CoolRunner-II CPLD Solutions for Cell Phone Handsets and Terminals

A Designer's Guide

Table of Contents

XC2C128 CoolRunner-II CPLD

XC2C256 CoolRunner-II CPLD

About This Handbook

We wrote this handbook to share many of the solutions Xilinx has created supporting handset and terminal customers over the last few years. To that end, we have included many of the basic functions that you will find in standard handsets, like keyboard and display interfaces; but, we also include solutions to extend beyond basic handset functionality. For instance, we include a compact flash interface that easily modifies to an IDE disk interface. As more applications like audio and video entertainment are being added into handsets, we also track the requirements for greater local storage.

This handbook consists of a collection of technical documents, including white papers, application notes and data sheets. Given the extremely fast pace of technology development, some content may become outdated over time. To find the latest technical product and application data from Xilinx, simply find the required content on the Xilinx website at www.xilinx.com/support/library.htm.

Two key messages are threaded throughout this handbook. First, Xilinx has become a high volume supplier of products into the portable consumer arena, with worldwide manufacturing, advanced product planning logistics and world class customer support – in local time zones with local languages. We ship tens of millions of CPLDs each quarter, all over the world.

Second, although handsets and terminals are well served by chipset solutions from Texas Instruments, Intel, Freescale, Qualcomm and others, these chipsets take time to develop and get “right”. In that time, evolving applications arise making it impossible to deliver to the world wide customer base in real time. Some handset developers have time horizons of nine months to a year, whereas others develop within two to four months. Being able to respond to the latest application requirements can only be done with low power, inexpensive, high speed programmable logic.

Programmable logic gives you fastest time-to-market and flexible product life cycle management available in the silicon world. There is no other equivalent solution for reducing design time, and nothing that gives you this level of flexibility to respond to changing market requirements.

Incidentally, much of the information we provide was “discovered” by ASIC designers needing to fix their gate array solutions with small, low power CPLDs. This occurred so often that they extended the idea into their product development cycle, to account for late arriving, last minute market requirements. You might say: “there’s always room for CoolRunner^{TM}-II.”

Please scan the table of contents to find the applications you need today, or work your way through the handbook following the capabilities of CoolRunner-II CPLDs, delivering low power, inexpensive solutions in tiny packages.

You will find full explanations of CoolRunner-II advanced features like clock dividing and DataGATE. Explanations will show how adding a CoolRunner-II to a design can actually reduce the power being used on the whole board. The application notes are backed up with full designs you can download and use today. The designs are fully documented, allowing you to expand or collapse functionality as required.

We have included the CoolRunner-II Family data sheet and those of the four most popular parts for handsets and terminals – the XC2C32A, XC2C64A, XC2C128 and the XC2C256 CoolRunner-II CPLDs.

Acknowledgements

This book would not have been possible without the efforts and cooperation of several applications engineers, and one FAE. I would like to thank them for their contributions to the ideas, designs, and hard work performed in the creation of application notes and white papers for the handset marketplace. They are:

- Nick Mehta
- Anita Schreiber
- Mark Ng
- Scott Lien
- Frank Wirtz
- John Hubbard
- Jennifer Jenkins
- Mike Gulotta

I would also like to extend my appreciation to the CPLD Marketing team for guidance and the creation of collateral marketing material to support the efforts of the handset initiative. They are:

- Mark Halfman
- Denny Steele
- Tony Grant
- Betsy Thibault
- LaToya Parker
- Roger Seaman

Sincerely,
Jesse Jenkins
Author

Additional Resources

To find the very latest data, see the Xilinx website at:

http://www.xilinx.com/literature/index.htm.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Introduction to Handset Design

This handbook provides information, white papers, data sheets, and application notes useful in the creation of low cost, low power electronic handsets, including cell phones, PDAs and other devices in which elegant design, cost, and battery life are paramount. You can improve your results in all three areas using Xilinx CoolRunner^{TM}-II CPLDs and the information provided in this handbook.

The handbook contains topics for the implementation of common handset functions, such as keypad scanners, as well as topics for improvement of low power design. Using this document you can reduce the number of components in your design, expand the features, lower your overall power profile, and save money in both design cost and component inventory.

This Chapter contains white papers and application notes written to assist a handset designer in the creation of new products. The topics listed are among the design applications our CPLDs are fully capable of performing. In most cases, Xilinx provides free reference designs to help speed up your design process. The reference designs can be found at:

http://www.xilinx.com/products/silicon_solutions/cplds/resources/coolvhdlq.htm

This Chapter contains the following topics:

- CPLDs in Cell Phone Handsets / Terminals
- Implementing Keypad Scanners
- Level Translation
- Character LCD Module Interface
- I^{2}C Bus Controller
- Interfacing to Mobile SDRAM
- Serial Peripheral Interface (SPI) Master
- Interfacing to a NAND Flash Memory Device
- CompactFlash Card Interface
- An SMBus / I^{2}C-Compatible Port Expander
- OMAP, XScale & Other Chip Sets
- Cell Phone Demo Board

XILINX®

WP198 (v1.1) July 4, 2005

CoolRunner-II CPLDs in Cell Phone Handsets/Terminals

Cell phone handsets (or “terminals,” as they’re called in Europe) are among the most dynamic products in the electronics market today. From their original analog roots, they have evolved into nearly pure digital devices with as much functionality as complex PDAs. Consumers who once evaluated handsets based on their ability to make high-quality local calls now take call clarity as a given. Their choices instead rest on characteristics ranging from a handset’s "skin" color to its ability to support streaming video. Buyers, even those shopping for low-cost handsets, increasingly demand these kinds of features: "extras" are well on their way to becoming standards. This shift puts manufacturers in a bind as they try to balance low cost with the ever-increasing consumer insistence on new features. Should customers pay for these features outright, or should their monthly payments subsidize the handset cost?

Each country has adopted an individual economic model to resolve this question. Common to all of these models, however, is a need to financially cope with increasing numbers of new features. Our goal in this white paper is to show how using CoolRunner™-II CPLDs in handsets will make it easier for handset developers to make these future changes—as well as baseline handset capability—financially viable.

Quick Look at a Handset

Figure 1 shows a fairly straightforward block diagram of a digital-type handset.

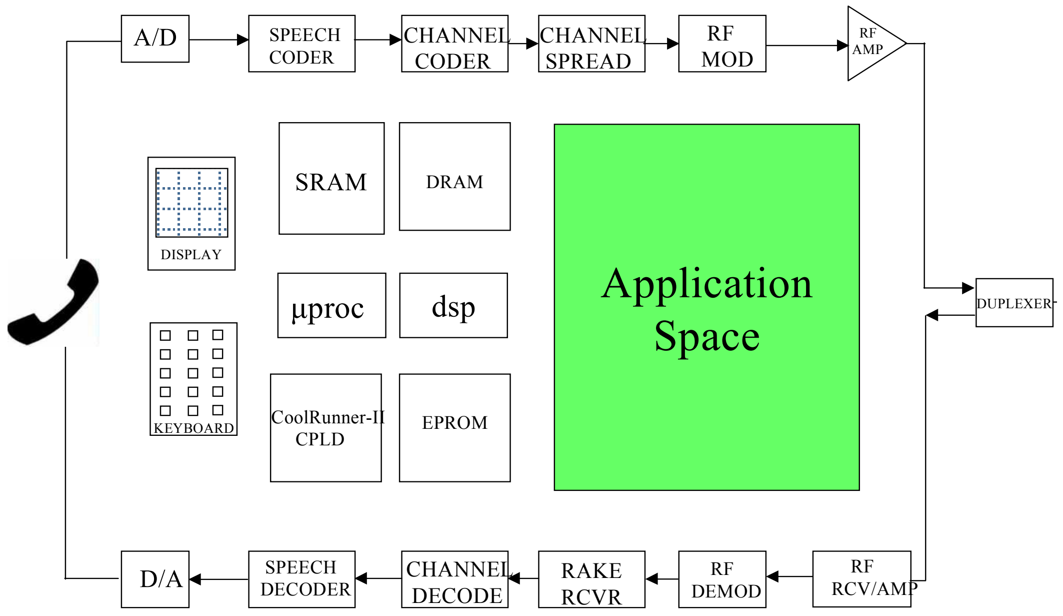


Figure 1: **Functional Components of a Digital Handset/Terminal**

In this diagram, we don't show specific connections between the microprocessor, the DSP, and the various boxes within the handset. Typically, they are all interconnected. In fact, some functions may be implemented by either processor. This saves board area at the expense of processor bandwidth. Let's assume the microprocessor handles baseline tasks like managing the display and the keyboard, but the DSP does channel coding / decoding and other signal processing tasks in the phone. The microprocessor handles dialing.

Let's simply look at sending and receiving audio. The microphone delivers audio signal to the A / D converter, creating a bit-stream. Fidelity and efficiency require redundancy elimination and compression. This is done in the speech coder (typically adaptive multirate). Forward Error Correction (FEC) occurs in the channel coder (Viterbi today, Turbo tomorrow). Coding adds back redundancy, but dropped bits can be recovered. CDMA channel spreading has been introduced in the U.S. and Korea, but others may use TDMA, SCDMA or WCDMA. The digital signal finally gets to the RF modulator and RF amplifier where it becomes one of several signal types (again, depending on the phone standard). At this point, it is analog. Today, these are in the gigahertz range and focused on octal phase shift keying. The duplexer is inserted to switch between receiving and sending at the antenna.

On the receiving end, the signal hits the antenna, slides through the duplexer and is amplified and demodulated, becoming digital. The Rake Receiver "despreads" the received bits and forwards the signal to the channel decoder, which reconstructs the possibly corrupted digital signal and passes it on to the speech decoder. The speech decoder corrects bit-dropout and should resemble the digital version of the original analog signal. This forwards to the D / A converter, which, when filtered and amplified (not shown), drives the earphone.

If the phone is going to support 3G capability, typically it must handle other standards (2G, 2.5G and GPRS) because different countries are at different levels of deployment. Why pay for a high-end phone and be able to only use it on the high end systems? Would you want to carry multiple phones, or should the manufacturer make sure your super handset just works wherever you are? Given that the latter is the case, your handset is basically three or four phones, depending on what communication link you are connecting to. It's a lot to ask, but 3G isn't about signaling, quality and compatibility, as much as it is about applications.

Applications

By now, you should have noticed the big box on the right hand side of Figure 1. That is reserved for advanced applications—things that go beyond making a call. Let's revise Figure 1 to set the stage for the kinds of things we will need to add applications. For starters, cell phone standards committees categorize applications according to parameters like guaranteed bit rate, variable / fixed delay, asymmetric / symmetric traffic and whether or not buffering is allowed. In terms of words we can relate to, those categories translate into conversational class, streaming class, interactive class and background class. These designations are based on the properties of those classes of applications. For instance, an ordinary conversation cannot tolerate long delays and echoes, as such things upset users. Lots of image dropout on a video data-stream also upsets users. So, by thinking through the needs of whole classes of applications, developers arrive at important quality factors. Listed below are some of potential cell phone applications; some may cross over more than one of the classes listed above.

1. MP3 music
2. Distance Learning
3. Online Retail
4. Travel booking
5. E-books
6. Distance Learning
7. Online Retail
8. Travel booking
9. M-commerce
10. Online trading
11. Mobile banking
12. Gambling
13. Games Interactive toys
14. Video movie rental
15. Virtual radio station
16. Virtual TV station
17. Newspapers
18. Photo album
19. Video Conferencing
20. Field service
21. Business kiosks
22. Secure monitoring
23. telemedecine
24. Mobile clinics
25. Email
26. Secure monitoring
27. Targeted advertising
28. Free trials
29. Remote metering
30. Remote control
31. Mobile speed cameras
32. Remote surveillance
33. Ticket purchases
34. Showtime information
35. Vending machines
36. Parking meters
37. Buying Crossword puzzles
38. Navigation
39. Finding ATM money machines
40. And many more . . .

Each application has its own particular needs, so it might be appropriate to consider a few of them individually. For instance, music via MP3 files could be done entirely in software, the same way that PCs can play files. If the processors are burdened with channel coding / decoding and possibly with decryption, it might be more feasible to include an MP3 decoder chip within the phone, or as an add-on. Video movie rental would probably have the files being downloaded via GPRS and linked through the Internet. Clearly, vendors renting the files would want to assure their MPEG-4 files weren't being delivered to pirates. Vendors would insist on encryption as well as compression. Adding these to the channel decoding exhausts today's processors and drains batteries. There would also be a need for buffering frames, to support dropout in the Internet protocol. Some applications might require JPEG, MPEG-4, MP3, encryption and channel coding. This suggests the need for additional offloading logic. Let's look at a commercial chipset for doing all of this, and then discuss offloading.

Cell Phone Chipsets

Figure 2 shows an Intel PXA 800F block diagram. Note that this multifaceted chipset is designed to integrate all the key functionality for the baseline, base-band operations, with a clear focus on using software to handle a broad range of other applications.

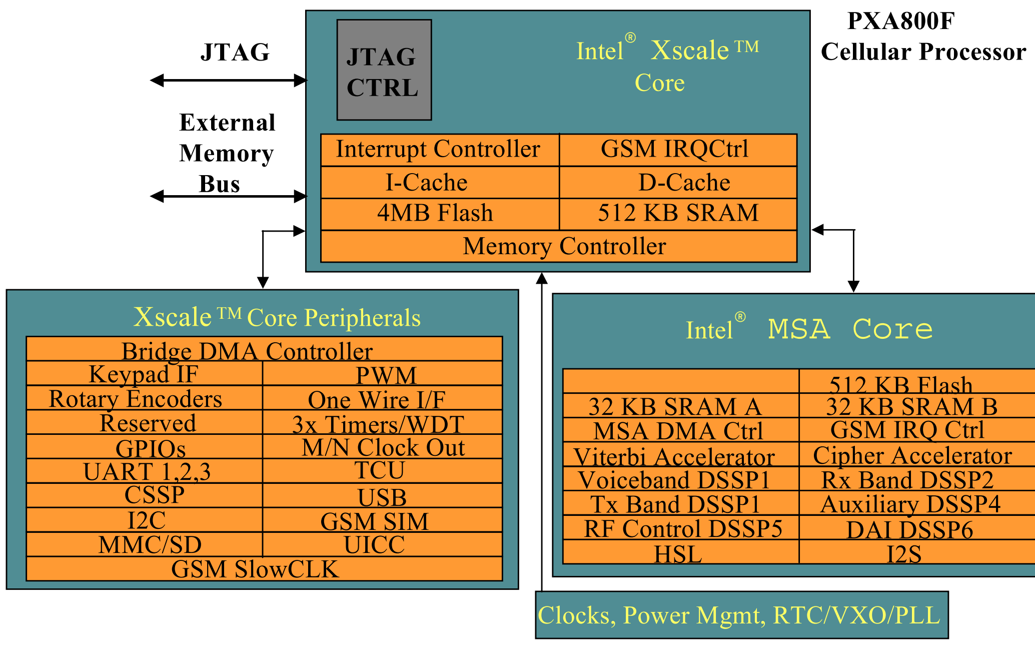


Figure 2: **Intel PXA 800F Cellular Chipset**

Manufacturers such as Motorola, Qualcomm and Texas Instruments offer their versions of chipsets that attempt do deliver the maximum capability in the minimum number of chips. Each is somewhat different in functional partitioning. The thing that is difficult for them to do is add enough flexible functionality to adapt to arbitrary future applications. We frequently see pins referring to keyboard interfaces, or display interfaces, which is great, but they are all different. Some of the European UMTS terminals resemble pocket computers with wide screen displays and full computer keyboards as opposed to 12-16 keys on a simple cell phone. Adaptation logic is needed to use these chipsets with broader applications.

Picking the Right Mix

Figure 3 shows an application that is not on the list—Global Positioning Satellite (GPS) service. It can use the same antenna as the phone receiver / transmitter, but it will drag the satellite signals into the chip, collect the data and deliver the co-ordinates of your location to the DSP. Emergency police / ambulance calls need this.

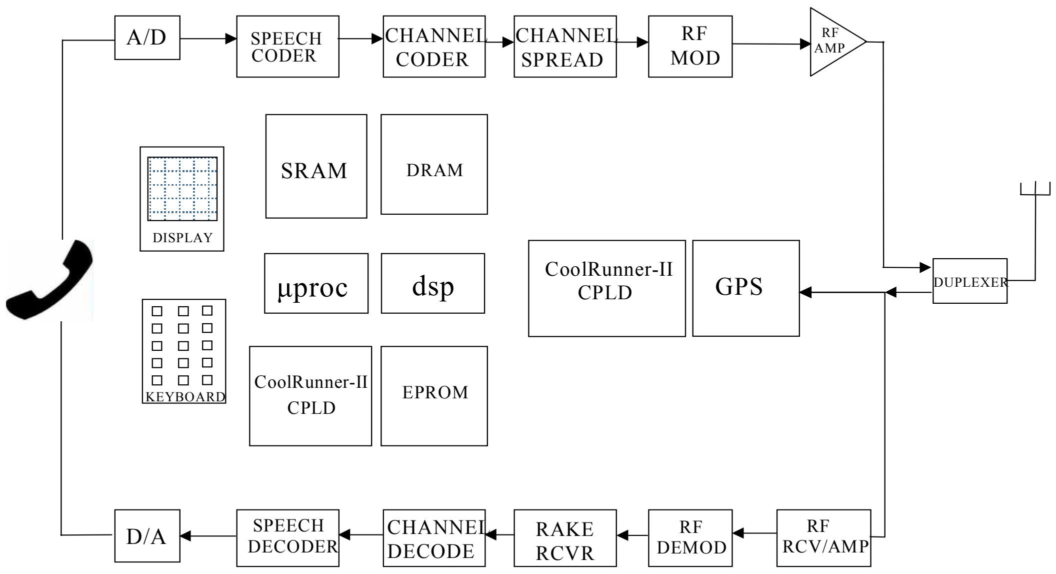


Figure 3: **GPS Added into a Phone**

In this case, it is impossible for the DSP or the microprocessor to "grow another radio chip." As long as the standard RF demodulator is being used, a second will be needed to support the GPS. The GPS will need to be interfaced to the processor buses to get data. The CoolRunner-II CPLD easily creates bus interfaces, interrupts and control operations for the processors, tying peripherals in.

Another example would be a camera. Adding a CMOS imaging chip requires interfacing it to the processor buses, along with additional memory to buffer images. Cameras come with either parallel or serial interfaces. Why be constrained to THEIR choice of peripherals in YOUR product?

For some time to come, applications will consist of adding in special application circuitry and interfacing it into the processor buses. Given that, how can a designer select the right application-specific circuits to add in, so that one ASIC could be built to interface them all? Basically, they can't. Below is an array of application modules that might be added into the phone:

- MP3
- MPEG-4
- Compact Flash +
- Video Camera
- GPS
- MMC / SDI
- More SRAM
- More DRAM
- More EPROM

But what about new applications that are not in this group? There are applications yet to come that we can envision, but whose exact specifications we cannot wholly anticipate. Designing for their arrival is sometimes called "future proofing."

It is true that some applications are better served by microprocessor code dropped into on board EPROM, but that can only happen as long as the processor bandwidth is available for the application. If this cannot be done, either more processors or additional silicon needs to be added. Either way, the application will need to interface into the phone bus network, and that will require programmable logic, very low power programmable logic.

MediPhone—A Speculative Example

To drive home some of these ideas, consider an idea for a product that probably does not exist today, but easily could in the near future. It will be marketed under the name "MediPhone" and will target segments of the population that require quick medical support. This would include the growing population of elderly citizens (frequently with enough money to buy these) as well as handicapped people needing close monitoring. See Figure 4 for an "artist" conception of this futuristic phone.

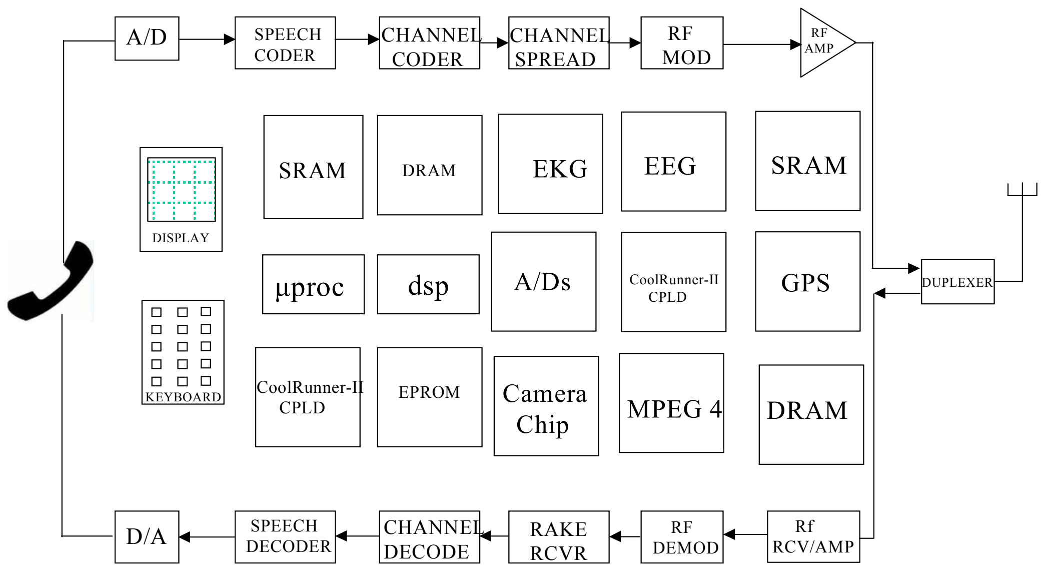


Figure 4: **MediPhone Block Diagram**

MediPhone works like this:

1. A heart attack (or other medical emergency occurs)
2. The victim or friend dials Emergency (911 in the U.S.)
3. Personnel receiving the call at a medical facility recognize the phone is "MediPhone" equipped and extract the geographic location of the emergency using GPS
4. The medic directs the friend to place the cell phone on the victim's face
5. A video camera scans the Iris for dilation to determine shock level
6. The friend is directed to attach small electrodes to the forehead / ear and chest of the victim, where pulse is taken and EEG / EKG measurements are driven into the Internet. Everything is attached to the phone.

7. Temperature is measured by physical contact of the forehead with the victim off the back of the phone
8. Microphone gain is adjusted to listen to the victim's breathing
9. It is determined that a heart attack has occurred and the friend is directed to elevate the feet and is given directions for CPR from the medic while an ambulance is dispatched
10. Ten minutes later, the ambulance team takes over and the MediPhone medic is released. For insurance reasons, all transaction data are encrypted, compressed and stored on CD for future reference.
11. The victim is saved and lives happily ever after . . .

In order to do this, MediPhone requires greater application functionality than discussed so far. Adding instrumentation amplifiers, A/Ds, transducers and support circuitry go beyond just having video and GPS as mentioned earlier. CoolRunner-II CPLDs would interface the various items needed to support MediPhone. The associated processor buses and memory interfaces are needed to support the capture of the information taken. Other ideas quickly come to mind (WeatherPhone, ToxicGas Phone, etc.)

Power and Packaging are Key!

We have not yet mentioned power in this discussion, but it may be a key element in figuring out a solution, at least at the outset. Power management requires a careful balance of dynamic operations within a cell phone. Do we add in specialty silicon, or do we just use a bigger EPROM and do it in software? As mentioned earlier, some choices are made for us. It would be ideal to just do it in the software, unless the bandwidth requirement on the processor(s) exceeds what could be done with a separate chip interfaced into the phone bus. Nonetheless, adding power consuming chips must be balanced against adding incremental code space, which also increases the power consumption. Whether you are interfacing more EPROM or additional ASSP silicon, CoolRunner-II is the ideal low power interface, with standby currents as low as 14 microamps. For power management, CoolRunner-II CPLDs are frequently used to enable/disable the power delivery to other chips within a system. So, if you are not using your camera or GPS capability, they can be automatically shut down to minimize power draw.

In addition to the unsurpassed power savings at high performance (clocking in the 300+ MHz range) provided by CoolRunner-II RealDigital technology, it is also critical to note that CoolRunner-II devices are offered in a wide range of very small packages. Table 1 shows part densities, available packages and other key features of CoolRunner-II CPLDs.

Table 1: **CoolRunner-II CPLD Family Parameters**

	XC2C32A	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
Macrocells	32	64	128	256	384	512
Max I/O	33	64	100	184	240	270
T_{PD} (ns)	3.8	4.6	5.7	5.7	7.1	7.1
T_{SU} (ns)	1.9	2.0	2.4	2.4	2.9	2.6
T_{CO} (ns)	3.7	3.9	4.2	4.5	5.8	5.8
$F_{SYSTEM1}$ (MHz)	323	263	244	256	217	179

Looking Forward to 4G

The fourth generation cell phones are already being referred to as Software Defined Radio (SDR), because so many things can be more easily (in theory) changed with software. Processor speeds in the gigahertz range result in swift, agile flexibility. However, as mentioned earlier, software cannot “grow” a camera. Software cannot add in a bus interface to Compact Flash +. Software cannot grow another radio channel. Only hardware can do some things, and when developers know neither when nor what the next new "thing" will be, only programmable logic can solve their development quandary. CoolRunner-II CPLDs provide the future proofing to help drive 4G cell phones.

Conclusion

We have focused, at a high level, on requirements of cell phones and on the capabilities of CoolRunner-II CPLDs for meeting those requirements. Given the current trend toward packing as much functionality as possible into handsets without increasing their power consumption, CoolRunner-II is an ideal choice for building phones that need to quickly adapt to new applications still on the horizon.

Product differentiation will ultimately result in winning designs. If developers want to cinch their winning positions, they must ensure that their designs have distinctive, useful applications that outdo those of their competitors. Using CoolRunner-II CPLDs to plan for future feature changes can contribute enormously to speedy and cost-effective introduction of such applications.

References

1. *UMTS Networks, Architecture, Mobility and Services*, H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian, V. Niemi, 2001, John Wiley & Sons, Ltd.
2. *Services for UMTS (Creating Killer Applications in 3G)*, T. Ahonen, J. Barrett, editors, 2002, John Wiley & Sons, Ltd.
3. *3G Wireless Demystified*, L. Harte, R. Levine, R. Kikta, 2002, McGraw-Hill
4. "The Mobile Phone Meets the Internet," M. W. Oliphant, August 1999, *I.E.E.E. Spectrum*
5. "Evolving Cellular Handset Architectures but a Continuing, Insatiable Desire for DSP MIPs," M. L. McMahan, March 2000, Texas Instruments Application Report SPRA650

Further Reading

The following links provide information useful for handset and CoolRunner-II design.

CoolRunner-II Design Kit

http://www.xilinx.com/products/cr2/design_kit.htm

Application Notes

Select the link below for a complete list of links to over 75 CoolRunner-II Application Notes. The list includes notes on the PicoBlaze Microcontroller, Keypad Scanners, Level Translation, Bus Controllers, Memory Interfaces, and Handheld Designs.

CoolRunner-II Application Notes

If you are looking at a printed copy of this document, go to www.xilinx.com and go to the documentation link.

CoolRunner-II Data Sheets

http://direct.xilinx.com/bvdocs/publications/ds090.pdf (CoolRunner-II Family Data Sheet)

http://direct.xilinx.com/bvdocs/publications/ds091.pdf (XC2C32 Data Sheet)

http://direct.xilinx.com/bvdocs/publications/ds092.pdf (XC2C64 Data Sheet)

http://direct.xilinx.com/bvdocs/publications/ds093.pdf (XC2C128 Data Sheet)

http://direct.xilinx.com/bvdocs/publications/ds094.pdf (XC2C256 Data Sheet)

http://direct.xilinx.com/bvdocs/publications/ds095.pdf (XC2C384 Data Sheet)

http://direct.xilinx.com/bvdocs/publications/ds096.pdf (XC2C512 Data Sheet)

CoolRunner-II White Papers

The following link takes you to the CoolRunner-II White Papers. If you are looking at a printed copy of this document, go to www.xilinx.com and go to the documentation link.

CoolRunner-II White Papers

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/30/03	1.0	Initial Xilinx release.
07/04/05	1.1	Update of specifications in Table 1.

XAPP512 (v1.1) May 6, 2005

Implementing Keypad Scanners with CoolRunner-II

Summary

This application note provides a functional description of Verilog source code for a keypad scanner. The code is used to target the lowest density, 32-macrocell CoolRunner^{TM}-II XC2C32A CPLD device in a CP56 package (6 mm x 6 mm). The keypad accommodated in this design has 8 rows and 8 columns. The design can easily be scaled to target keypads with more or less rows/columns. For instance, a keypad with 7 rows and 7 columns would allow the design to fit in the smallest QFG32 package (5 mm x 5 mm). To obtain the Verilog source code described in this document, see "Verilog Code," page 26, for instructions.

Introduction

As handheld devices such as cell phones pack more and more features into them, they require more effective ways of entering data. Most cell phones, for example, use the standard DTMF style keypad and a multi-tap process to enter alphanumeric data; however, for larger amounts of data multi-tapping becomes cumbersome. More and more high-end phones are therefore employing QWERTY keypads that make entering data easier and quicker.

Going from a DTMF to a QWERTY keypad requires more I/O. For instance, a DTMF keypad might have 4 rows and 3 columns, where a QWERTY keypad might have 8 rows and 8 columns. This can vary depending on the requirements.

Typically, a processor (or ASIC) is used to interface to the keypad's rows and columns. The processor scans the rows and monitors the columns for a logic change. When a change occurs, it indicates that one of the buttons in that column was pressed. By knowing which row was being scanned, and which column changed state, the processor can deduce which specific button was pushed. Additional functions such as debounce are also typically employed. Figure 1 shows how a simple 4 x 4 keypad uses 8 GPIO of a processor.

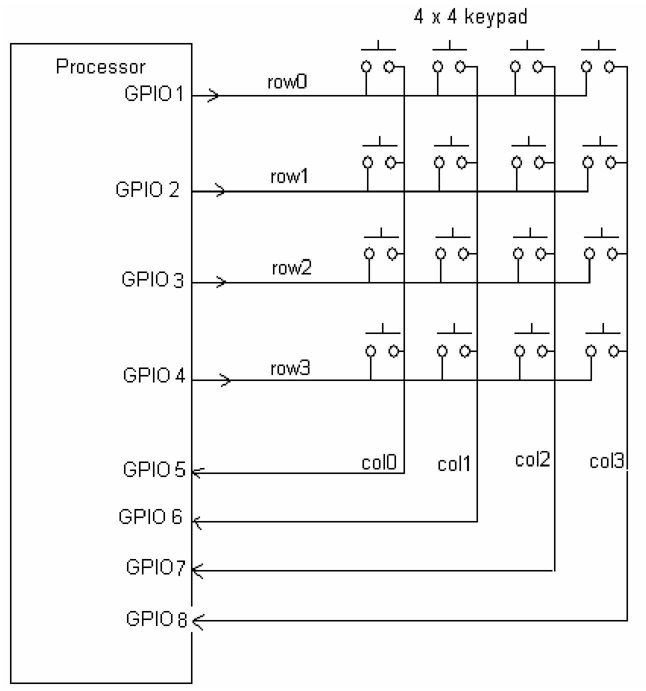


Figure 1: **Simple 4 x 4 Keypad Connected to a Processor Requiring 8 GPIO**

Expanding I/O

Designers faced with accommodating a keypad requiring more I/O might find their existing processor (or ASIC) does not have enough GPIO ports. One solution is to use a CPLD as an I/O expander that reduces the I/O requirement of the processor.

Figure 2 shows a CPLD interfacing to the keypad rows/columns on one side, and the processor's available GPIO on the other. In this example, an 8 x 8 keypad requires the same number of processor GPIO ports as the 4 x 4 keypad (actually one less) when a CPLD is used. Without a CPLD, the processor would require 16 GPIO ports instead of 7.

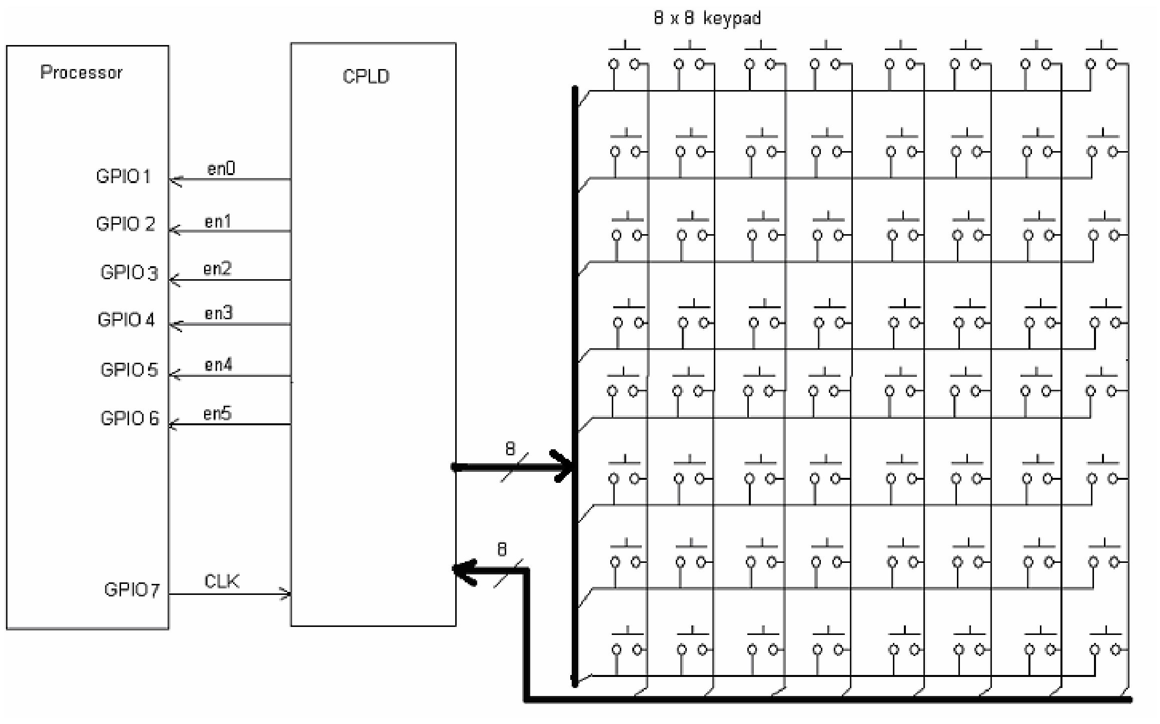


Figure 2: **CPLD Expands I/O and Reduces the Processor's GPIO**

Scanning and Encoding

Besides reducing the processor's GPIO requirements, the CPLD also scans the rows and monitors the columns for a change in state. When a key is pressed, the CPLD stops scanning and immediately sends an encoded word out to the processor. The encoded word indicates which key was pressed.

In the example shown in Figure 2, there are six bits used to represent the encoded word. Six bits provides 2^6 or 64 different values each representing a different key. However, one value needs to be used to represent the state when no keys are pressed. Therefore, only 63 keys can be represented in this example.

All 64 keys most likely would not be needed in a typical application. If they were, there are many options with a programmable CPLD. For instance, a CPLD could generate an enable signal to the processor that would indicate when a key is being pressed (that is, when the encoded value was valid). This would require one more GPIO on the processor.

The processor is still required to monitor for changes on its GPIO, only it would not have to deduce which key was pressed since this information is encoded in the six bit word. Debounce will also be required. This can be performed in the CPLD or the processor. Performing this in the processor would keep the size of the CPLD to a minimum.

CPLD Design Details

Note: The following details describe how the available Verilog reference design is implemented. The concept and design are relatively simple. Additional functionality can be added depending on specific design requirements.

To scan the keypad rows, a barrel shift register is initialized with all ones except for one bit preset to a zero. Each bit of the shift register drives a CPLD output pin that is connected to a row of the keypad. As the shift register is clocked the zero shifts through the barrel shifter and scans the rows by driving them low one at a time.

The columns are inputs to the CPLD. Each input is pulled up with an internal pull up resistor. When no keys are pushed, all column inputs to the CPLD are passively pulled up to a logic high state. All column inputs are AND'd together. A logic one at the output of the AND gate indicates no keys are pressed. The output of the AND is used as an enable to the shift register.

When a key is pressed, a connection between a row and column is made. The column with the key being pressed will be driven low by the row associated with that key. The output of the AND will go low and disable the shift register for how ever long the key is pressed.

At this point the shift register is driving the row of the key being pressed to a low, and the column of that key is also at a low. Two encoders are used, one for the row bits (outputs of the shift register), and another for the column inputs. The outputs of the two encoders are grouped together to form the resulting encoded word that is presented to the processor. Figure 3 shows a block diagram of these functions.

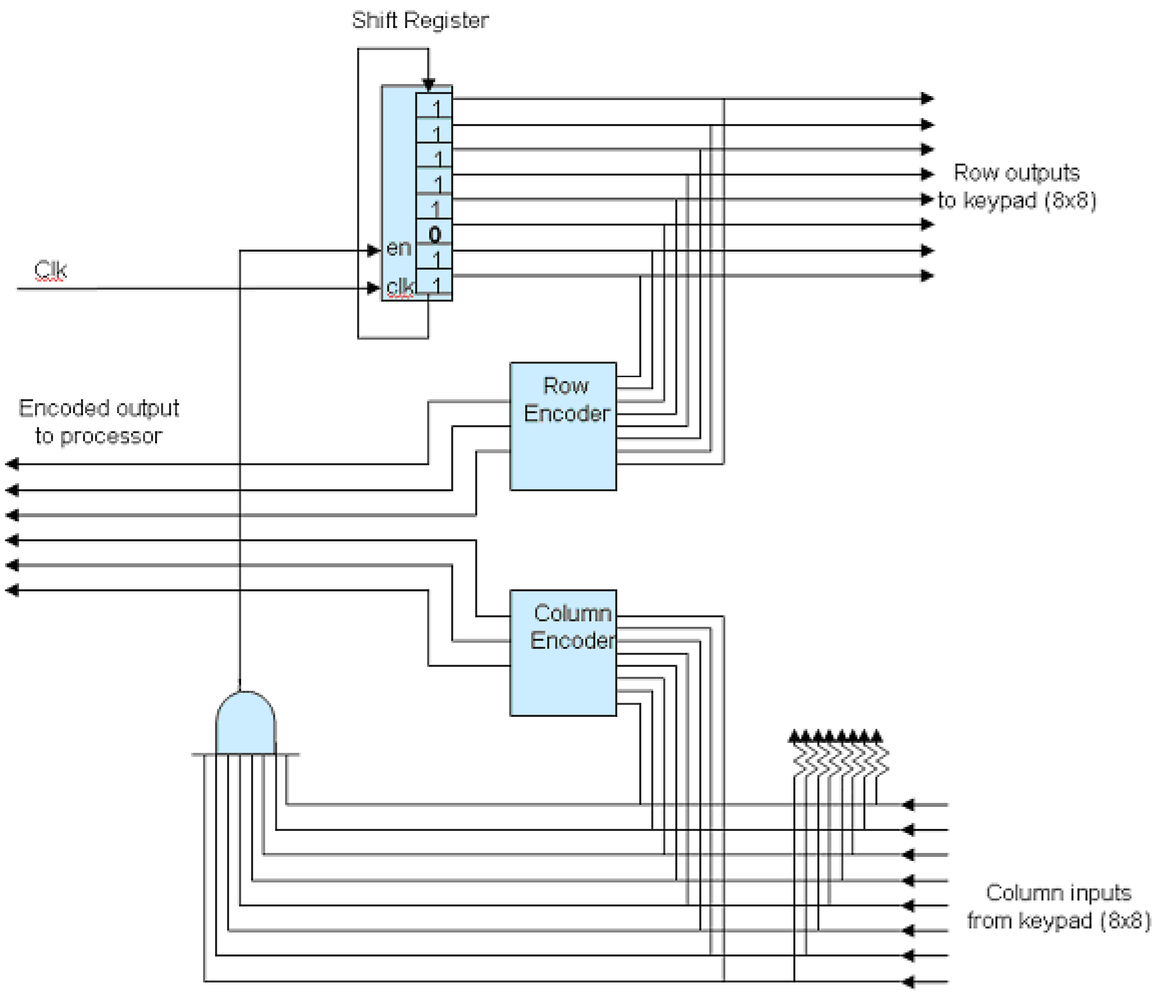

Shift Register
1
1
1
1
1
0
1
1
en
clk
Clk
Row outputs to keypad (8x8)
Row Encoder
Encoded output to processor
Column Encoder
Column inputs from keypad (8x8)

Figure 3: **Block Diagram**

Implementation and Verification

The reference design is implemented in Verilog. A Xilinx ISE^{TM} software project is zipped up, including the Verilog source file, Verilog testbench, and UCF file. If you do not have Xilinx software, you can obtain ISE WebPACK^{TM} for free from the Xilinx website. WebPACK will give you all the tools you need to complete any CPLD project.

The UCF file shows how to initialize the barrel shifter with a pattern of ones and a zero. It also shows how to program the column inputs with internal pull up resistors.

The testbench can be evoked from within Project Navigator, which will automatically run a custom ModelSim `.do` file. The `.do` file will compile the source code, open a waveform window to view signals, and run the simulation.

The testbench generates a clock for stimulus and also simulates the pressing of buttons by periodically connecting a row with a column. This is done until all possible combinations of rows and columns are simulated, and then repeats.

Verilog Code

XAPP512 - http://www.xilinx.com/bvdocs/appnotes/xapp512__verilog.zip

Conclusion

Since the CPLD is reprogrammable, adding a control line, changing the mapping of the encoded word, or accommodating different keypads is possible with the same device. Additionally, other "glue" functions can be absorbed into the CPLD, such as voltage translators.

A specific CPLD device (i.e., part number) can be used to accommodate different keypads and even different applications because it's programmable. This helps boost the volume (lowers cost), and reduces risk since changes can be made even after it's soldered down.

Coolrunner-II also is designed for low power, making it a good choice for battery powered applications, such as cell phones, PDAs, and other portable devices. They also have additional features that augment its low power. Multiple I/O banks can be used for voltage translation, which is another typical application in devices with a mixture of technologies.

Additional Information

CoolRunner-II Data Sheets, Application Notes, and White Papers

Access to all Xilinx Data Sheets, Application Notes, and White Papers

Device Packages

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/04/05	1.0	Initial Xilinx release.
05/06/05	1.1	NAND gate references on page 3 changed to AND gates.

XILINX®

XAPP785 (v1.0) June 22, 2005

Level Translation Using Xilinx CoolRunner-II CPLDs

Summary

As electronic design has advanced over the years, more and more I/O standards have been created. Since the days when the 5V CMOS and TTL standards were the prevalent standards with which to design logic circuits, there have been a lot of changes. Today, there are many voltage standards operating at different voltages with different thresholds. This application note explains how to use a Xilinx CoolRunner^{TM}-II CPLD to translate between different voltage levels.

Introduction

A typical electronic system will no longer operate at only one voltage. The most popular voltages used to interface between components on a board are 3.3V, 2.5V and 1.8V. However, more frequently, devices need to interface to ’unusual’ voltages.

Since the introduction of the CoolRunner-IIA parts (XC2C32A and XC2C64A), all Xilinx CoolRunner-II devices have multiple I/O banks. The XC2C32A, XC2C64A, XC2C128, and XC2C256 have two banks each, and the XC2C384 and XC2C512 have four banks each. This means that the V_{CCIO} rail (the power supply for the device I/O) is split, enabling I/O in different banks to be powered at different voltage levels. Each I/O bank can support one V_{CCIO} voltage at a time. The supported I/O standards for the CoolRunner-II device can be seen in Table 1 below.

Table 1: **Supported I/O Standards in CoolRunner-II Family**

IOSTANDARD Attribute	Output V_{CCIO}	Input V_{CCIO}	Input[1] V_{REF}	Board TerminationVoltage V_{TT}
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15[2]	1.5	1.5	N/A	N/A
HSTL_1[3]	1.5	1.5	0.75	0.75
SSTL2_1[3]	2.5	2.5	1.25	1.25
SSTL3_1[3]	3.3	3.3	1.5	1.5

1. For information on assigning Vref pins, see XAPP399
2. LVCMOS15 requires use of Schmitt-trigger inputs.
3. HSTL_1, SSTL2_1 and SSTL3_1 are supported on XC2C128 and larger

The I/O characteristics of each standard can be found in the device specific CoolRunner-II Datasheets, for example XC2C128. More information about each of the I/O standards can be found in XAPP382. For information about interfacing to 5V, see XAPP429.

Configuring I/O to Use I/O Standards

The designer specifies which I/O standard to use at the time of design entry. I/Os can be configured to operate at different I/O standards in a number of ways.

Default

The default I/O Standard can be set in the process properties for the Fit process in the ISE software. The use of the default I/O Standard will set all I/O used in the design to the I/O standard specified. This is useful if all the I/O are powered at the same voltage. However, this application note is discussing interfacing between different voltages, so another I/O standard assignment method is required.

PACE

The Xilinx Pinout Area and Constraints Editor (PACE) tool can be used to assign a variety of constraints, including pin locations, slew rate, Schmitt trigger and I/O standard. The package diagram distinguishes between I/O banks by using different colors.

The Design Object List then enables you to assign I/O standards to the I/O pins in the design. If the banking rules are broken, then PACE will issue a message that the I/O standard is not V_{CCIO} compatible.

UCF

The final method of assigning I/O standards in the software is to enter them directly into the User Constraint File (UCF). The syntax for entering I/O standard constraints into the UCF is as follows:

```
NET "user_net" IOSTANDARD = xx ;
```

Where ***xx*** is one of the permissible values: LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVTTL, and for devices of 128 macrocells and above, HSTL_1, SSTL2_1 and SSTL3_1.

Xilinx recommends the use of the Default assignment in co-operation with PACE to get the optimum results, since PACE has the design rules check that the UCF entry method lacks.

Using the CPLD as a Level Shifter

In addition to knowing of how to configure the I/O of the CPLD at different voltages, it is also necessary to have an understanding of how to use the device as a level shifter. It is possible to have this device act as a route-through to simply shift, for example, 1.8V inputs to 3.3V outputs. This would simply require a logical assignment of the input bus to the output bus, and would use one pin per input/output, and one product term and one macrocell per output.

Even in the smallest device in the CoolRunner-II family, the XC2C32A, performing this translation on an 8-bit bus would utilize less than one quarter of the resources. As the input signals must go through the central interconnect switch, they can be assigned to any output pin required by user. This makes the CPLD perfect for performing additional functions such as bit-swapping of the input bus, or changing from little-endian to big-endian format. CoolRunner-II CPLDs have a very fast T_{PD} (propagation delay), as fast as 3.8 ns in the XC2C32A -4, so very little delay will be incurred by using a CPLD in this situation. It is also important to note that, due to the uniform nature of the architecture, all signals going through a similar path in the device will incur a similar delay and will, therefore, have minimal skew from each other.

While this is a perfectly legitimate use for a CPLD, there are a lot of other resources available that can be used to perform operations on the incoming data.

Incorporating the Level Shifter into a Common Interface

It is common for devices on a board to operate at different I/O voltages. The information above explained how to use the CPLD as a simple level shifter, but all too often, the data coming into the device will need some operation performed upon it. There are many communication interfaces for use in electronic systems (USB, SPI, I^{2}C), several of which can fit into a Xilinx CPLD. Figure 1 shows a generic example that builds on the previous example. Here the 1.8V input bus goes into the device, has a few operations performed on it, such as shifting left or right, checksum calculation, parallelization, serialization, interrupt handling, and so on, and emerges from the device at 3.3V.

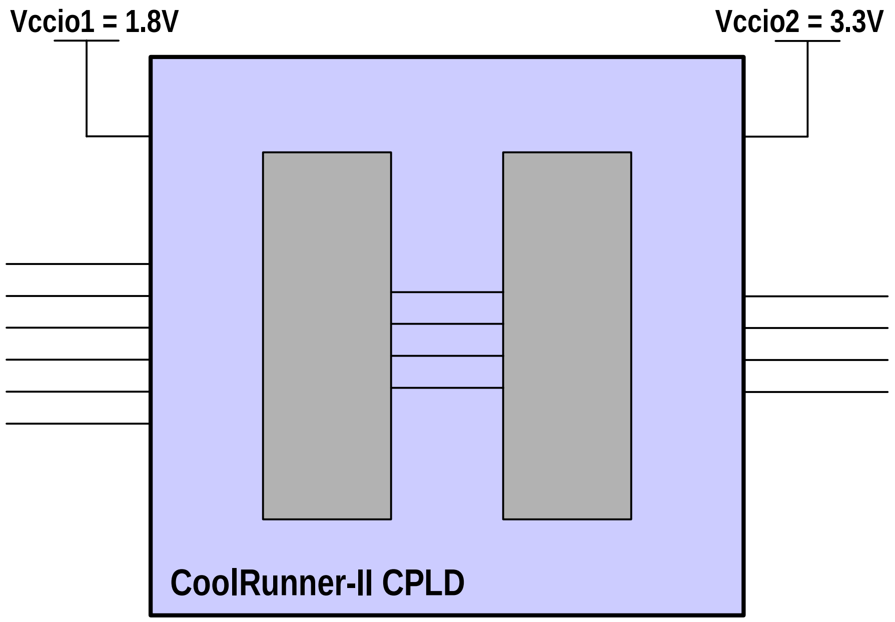


Figure 1: **Using CoolRunner-II as a Level Shifter Interface**

A few examples of common interfaces can be found in other application notes on the Xilinx website.

- **XAPP341 UARTs in Xilinx CPLDs**
- **XAPP354 Using Xilinx CPLDs tp Interface to a NAND Flash Memory Device**
- **XAPP384 Interfacing to DDR SDRAM with Xilinx CoolRunner-II CPLDs**

CPLDs Can be Used to Supply Current

An added advantage of using a CPLD in this situation is if the device providing the 1.8V signals to the CPLD cannot supply sufficient current. The CPLD can be used to supply the current required. To ascertain how much current the I/O of the CPLD can provide under typical conditions, the I-V plots for the I/O are needed. These can be seen in the device specific CoolRunner-II datasheets. Figure 2 is taken from the XC2C32A datasheet.

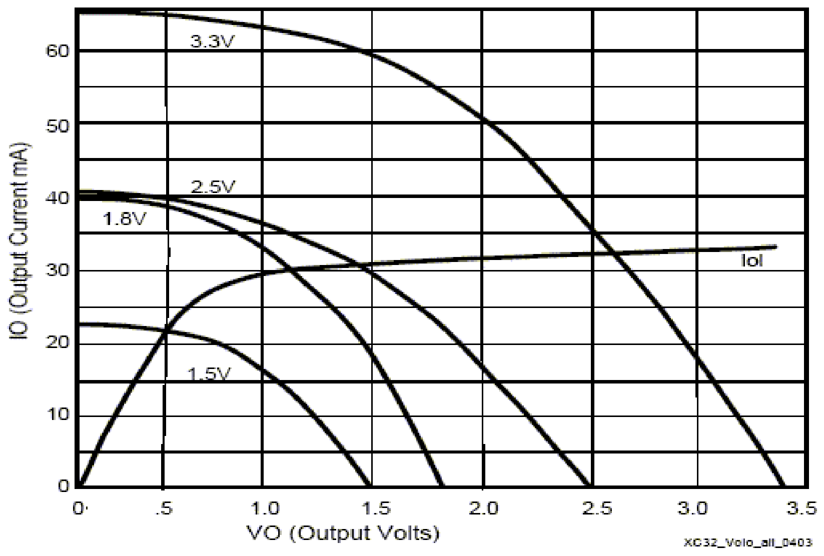


Figure 2: **IV Curves for the CoolRunner-II I/O Standards**

Powering V_{CCIO} Between Specifications

The CoolRunner-II architecture supports 1.5V, 1.8V, 2.5V and 3.3V I/O. However, 2.85V, 2.8V and 2.7V are also commonly used I/O voltages in certain applications. Can CoolRunner-II support 2.85V?

The input buffer and output buffer are the same when using a 2.5V I/O standard (LVCMOS25) as when using a 3.3V I/O standard (LVCMOS33 or LVTTL). The permissible V_{CCIO} range for LVCMOS25 is 2.3V to 2.7V, and the permissible V_{CCIO} range for LVCMOS33 is 3.0V to 3.6V. So, there is a clear gap from 2.7V up to 3.0V that is "out of spec." Due to the linear scaling nature of the input and output buffers, it is safe to assume that the V_{IH} min and V_{OH} min specifications for a pseudo-2.85V I/O standard can be interpolated from these specifications. V_{IH} will be in the region of 1.85V and V_{OH} will be V_{CCIO} minus 0.4V at (maximum) 8 mA loading.

Conclusion

Xilinx CoolRunner-II CPLDs are perfectly suited to perform level translation operations on signals. They offer split V_{CCIO} rails on all devices providing at least two I/O banks. Signals can enter the device at one voltage and be output from the device at another voltage without introducing any skew on the signals. This application, which previously required a dedicated IC, can now be incorporated into the CoolRunner-II, which can also be performing other system functions.

Additional Information

CoolRunner-II Data Sheets, Application Notes, and White Papers

Access to all Xilinx Data Sheets, Application Notes, and White Papers

Device Packages

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/22/05	1.0	Initial Xilinx release.

XILINX®

XAPP904 (v1.0) August 22, 2005

CoolRunner-II Character LCD Module Interface

Summary

CoolRunner^{TM}-II CPLDs can be used to control dot-matrix liquid crystal display (LCD) modules. The low-power characteristics of LCD modules make them an ideal complement to the CoolRunner-II family. These displays typically require 3.3V signals. However, this is not of concern because CoolRunner-II devices are 3.3V tolerant. Thus, it is possible to achieve ultra-low power at a 1.8V core voltage while using 3.3V at the I/O.

Introduction

Character LCD Module Background

There are many manufacturers of dot matrix LCD modules. However, most of these displays are similar. They all have on-board controllers and drivers capable of displaying alpha numerics and a wide variety of other symbols (including Japanese "Katakana" characters). The internal operation of LCD controller devices is determined by signals sent from a central processing unit (in this case, a CoolRunner-II CPLD).

These signals include:

1. Register Select (RS)
2. Read/Write (R/W)
3. Data Bus (DB7~DB0)
4. Enable Strobe (E)

Character LCD Instructions

Figure 1 shows the timing requirements for writing data to the LCD (this design does not perform read operations for the sake of simplicity). Notice that data is written with respect to the falling edge of the Enable Strobe (E) signal. The CoolRunner-II CPLD is responsible for making sure that LCD Module timing requirements are met. More specific timing requirements can be found in the LCD data sheets.

Figures 1 and 3 originated at Seiko, and were found at:

http://www.seiko-usa-ecd.com/lcd/products/char_mods/pdf/instructions.pdf

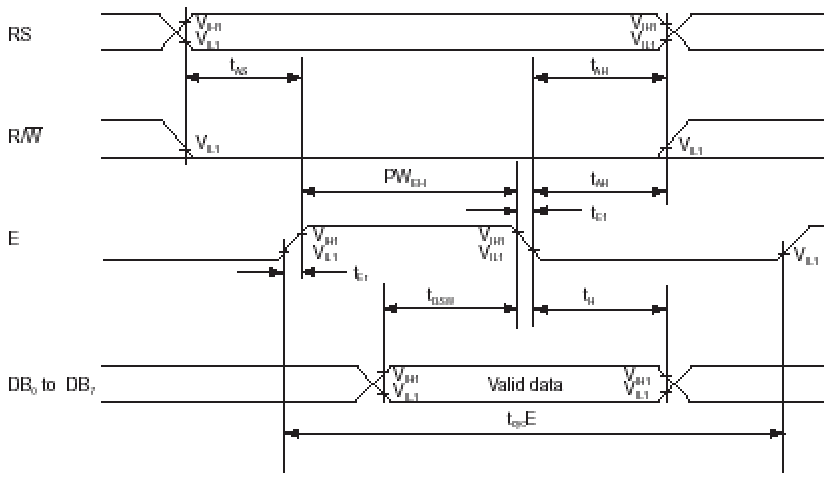


Figure 1: **Data Write from CPLD to LCD**

CPLD Implementation

Figure 2 shows a block diagram of an LCD Controller on a CoolRunner-II CPLD.

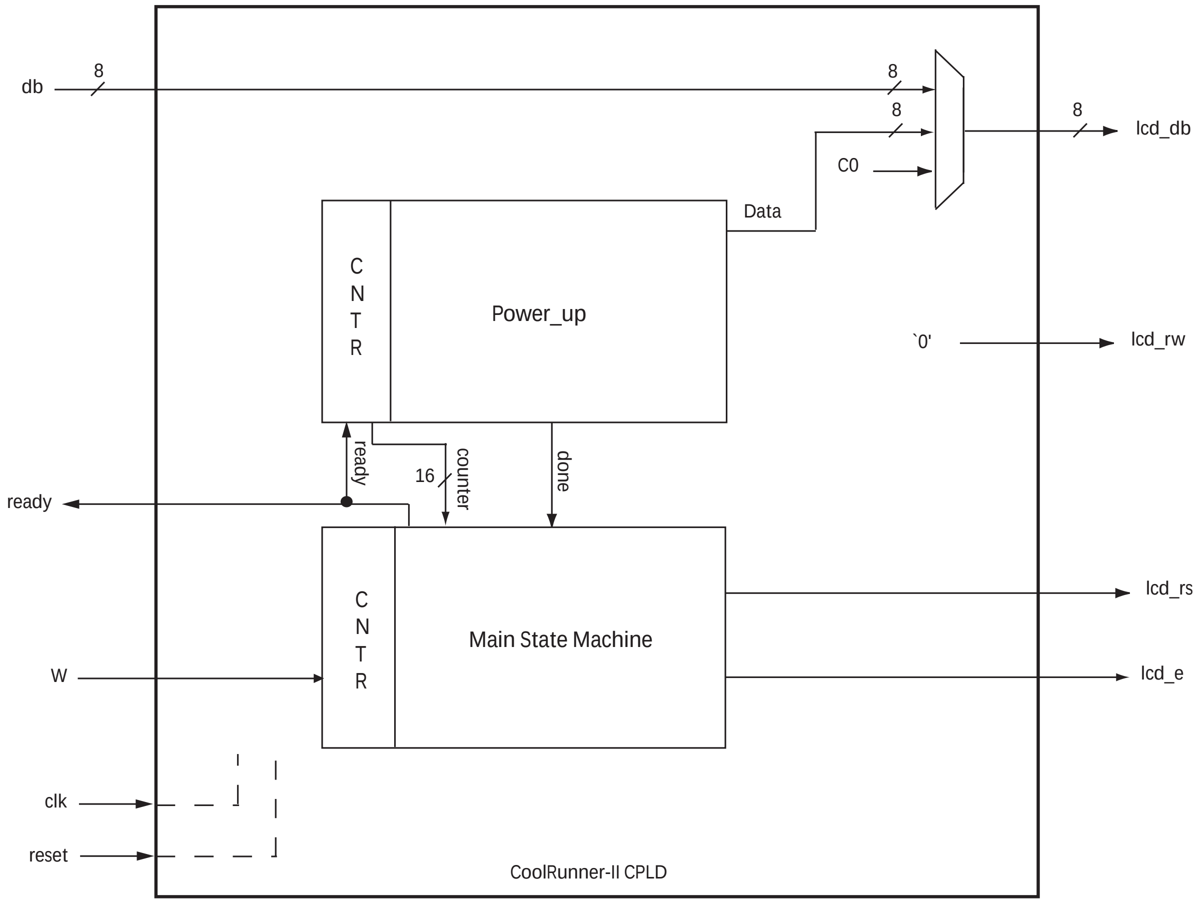


Figure 2: **LCD Controller Block Diagram**

The LCD Controller implementation is comprised of two state machines - a Power_Up State Machine, and an LCD Controller Main State Machine. These two blocks are discussed in the following sections.

Table 1 defines the primary inputs/outputs of this LCD Controller design.

Table 1: **Signal Descriptions**

Signal Name	Port Specification	Description
Reset	input	Reset Signal (Active High)
DB[7:0]	input	8-bit Data bus.
W	input	Write strobe (Active High)
Ready	output	Ready Signal - Asserted by CPLD to signal when more data can be sent (Active High)
Clk	input	Clock input (This design assumes 1.8MHz clock)
LCD_RS	output	LCD Register Select
LCD_RW	output	LCD Read/Write
LCD_E	output	LCD Enable Strobe
LCD_DB[7:0]	output	8-bit LCD Data Bus

Power-Up State Machine

LCDs require an initialization procedure to be executed each time the module is turned on or reset. The Power-Up State Machine is designed to automatically take care of this procedure by sending a sequence of hex codes from the CPLD to the LCD module. Upon completion, the LCD cursor will be enabled, the display will be cleared and the LCD module will be set for auto-increment mode. (See Figure 3)

During this sequence, the CPLD waits for 15 ms to allow the LCD to power up, then sends 0x38, 0x06, 0x0E and 0x01. This initialization sequence is automatic, and will also occur upon every power-up or reset. All timing requirements will be met, assuming a 1.8 MHz external clock frequency.

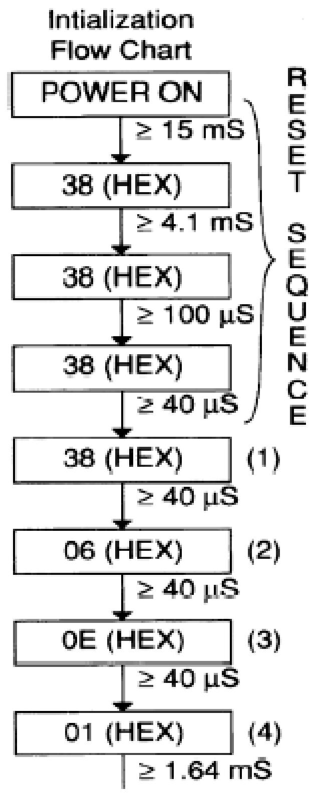


Figure 3: **LCD Module Initialization Sequence**

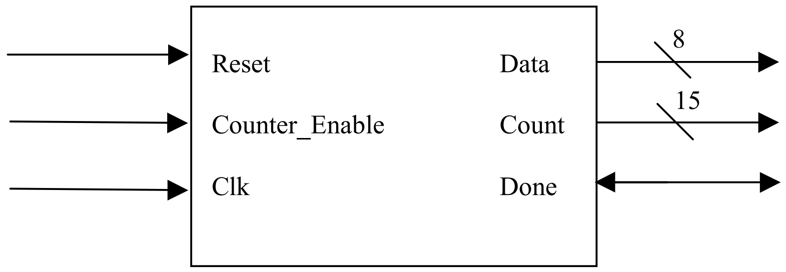


Figure 4: **Power_Up Module Block Diagram**

Figure 4 shows a block diagram of the Power_Up module. This Power_Up module utilizes a terminal counter to insure that data is held for the appropriate amount of time, as specified in Figure 3. As shown, the largest wait time required by the LCD is 15 ms (during the Power_ON sequence). Thus, to simplify the design, data is held for 16 ms in every subsequent block. Since this design assumes a 560 ns clock period (1.8 MHz), a terminal count value of 30,000 is used. Alter this terminal count if a different clock frequency will be used.

An internal 'Done' signal is output by this Power_Up module to let the main LCD State Machine know when the Power_Up sequence has completed.

LCD Controller Main State Machine

After the LCD has been initialized, control is released to the LCD Controller Main State Machine. The CPLD will drive the ’Ready’ line high in order to signify that the power up state machine has completed and that it is ready to accept data to be written to the CPLD. Hence, a CPU, or some other upstream device, sends a data byte corresponding to a specific LCD character to the CPLD along with a Write strobe. The CPLD will latch the data byte on the falling edge of the Write strobe, drive the ’Ready’ signal low, and drive the appropriate LCD signals in order to make the character appear. Input Data will be ignored by the CPLD until the ’Ready’ signal is high again.

Figure 5 shows the complete sequence of events, starting from the power up initialization process. As can be seen, the CPLD first sends the initialization codes 0x00, 0x38, 0x06, 0x01 followed by 0x80. Immediately after, the ’Ready’ line is asserted, and the CPU begins sending data on ’DB’ on every edge of the ’W’ line. The CPLD latches the data on every rising edge of W, and drives the ’LCD_DB’, ’LCD_RW’, ’LCD_RS’ and ’LCD_E’ pins accordingly in order to make the character appear. The CPU continuously monitors the ’Ready’ line and sends data only when Ready is high.

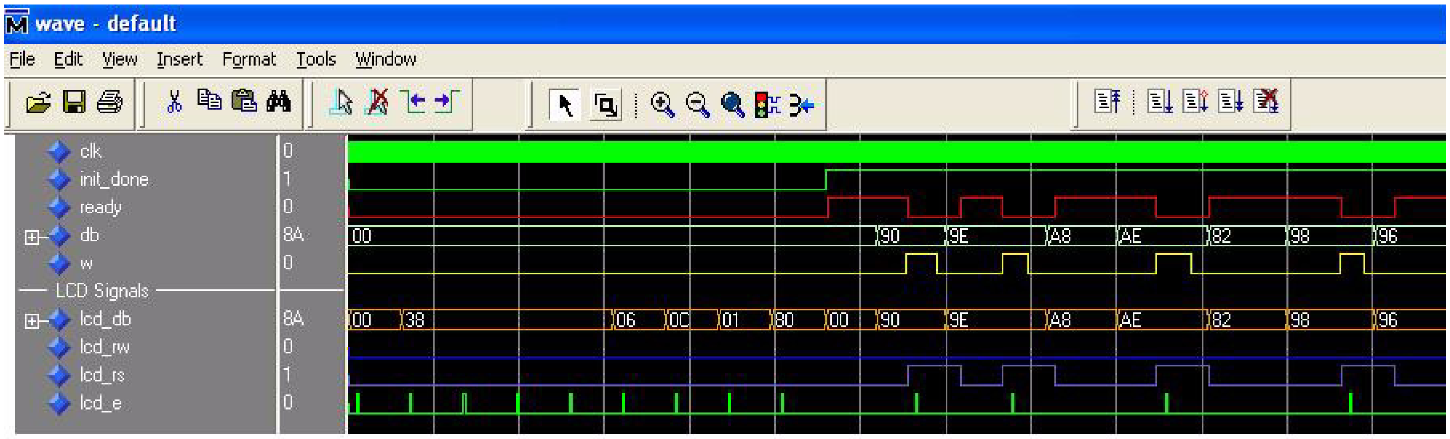


Figure 5: **Signal Sequence of Events**

Table 2: **CPLD Resource Use Summary**

Resource Summary

This design uses a total of only 40 macorcells, allowing it to fit into an XC2C64A device.

```
************************  Mapped Resource Summary  *******************

Macrocells     Product Terms    Function Block   Registers     Pins

Used/Tot       Used/Tot          Inps Used/Tot   Used/Tot      Used/Tot

40/64(62%)    64/224(29%)    61/160(38%)    26/64(41%)   23/33(70%)
```

Additional Information

CoolRunner-II Data Sheets and Application Notes

Conclusion

The LCD Interface presented in this application note is simple and straightforward. Additionally, CoolRunner-II devices are ideal candidates for driving LCDs due to their low cost, low power and ease of use.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/22/05	1.0	Initial Xilinx release.

XAPP385 (v1.1) December 30, 2003

CoolRunner-II CPLD I^{2}C Bus Controller Implementation

Summary

This document details the VHDL implementation of an I^{2}C controller in a Xilinx CoolRunner™-II 256-macrocell CPLD. CoolRunner-II CPLDs are the lowest power CPLDs available, making this the perfect target device for an I^{2}C controller. To obtain the VHDL code described in this document, go to section **VHDL Code Download**, page 55 for instructions. This design fits both XPLA3 and CoolRunner-II CPLDs. For the CoolRunner XPLA3 CPLD version, please refer to **XAPP333, CoolRunner CPLD I^{2}C Bus Controller Implementation**.

Introduction

The I^{2}C bus is a popular serial, two-wire interface used in many systems because of its low overhead. The two-wire interface minimizes interconnections so ICs have fewer pins, and the number of traces required on printed circuit boards is reduced. Capable of 100 KHz operation, each device connected to the bus is software addressable by a unique address with a simple Master/Slave protocol.

The CoolRunner-II I^{2}C Controller design contains an asynchronous microcontroller (µC) interface and provides I^{2}C Master/Slave capability. It is intended to be used with a microcontroller (µC) or microprocessor (µP) as shown in Figure 1.

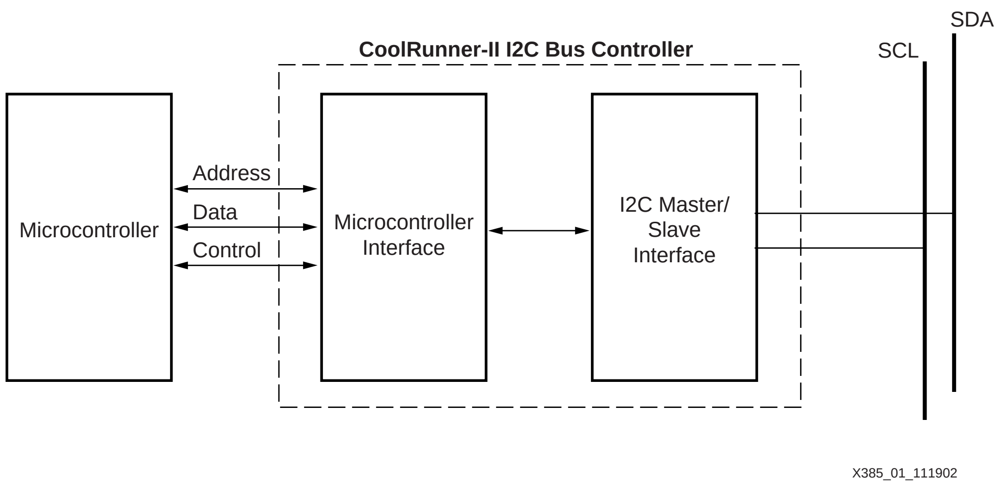

CoolRunner-II I2C Bus Controller
SDA
SCL
Microcontroller
Address
Data
Control
Microcontroller Interface
I2C Master/ Slave Interface
X385_01_111902

Figure 1: **CoolRunner-II I^{2}C Bus Controller**

I^{2}C Background

This section will describe the main protocol of the I^{2}C bus. For more details and timing diagrams, please refer to the I^{2}C specification.

The I^{2}C bus consists of two wires, serial data (SDA) and serial clock (SCL), which carry information between the devices connected to the bus. The number of devices connected to the same bus is limited only by a maximum bus capacitance of 400 pF. Both the SDA and SCL lines are bidirectional lines, connected to a positive supply voltage via a pull-up resistor. When the bus is free, both lines are High. The output stages of devices connected to the bus must have an open-drain or open-collector in order to perform the wired-AND function.

Each device on the bus has a unique address and can operate as either a transmitter or receiver. In addition, devices can also be configured as Masters or Slaves. A Master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any other device that is being addressed is considered a Slave. The I^2C protocol defines an arbitration procedure that insures that if more than one Master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted. The arbitration and clock synchronization procedures defined in the I^2C specification are supported by the CoolRunner-II I^2C Controller.

Data transfers on the I^2C bus are initiated with a START condition and are terminated with a STOP condition. Normal data on the SDA line must be stable during the High period of the clock. The High or Low state of the data line can only change when SCL is Low. The START condition is a unique case and is defined by a High-to-Low transition on the SDA line while SCL is High. Likewise, the STOP condition is a unique case and is defined by a Low-to-High transition on the SDA line while SCL is High. The definitions of data, START, and STOP insure that the START and STOP conditions will never be confused as data. This is shown in Figure 2.

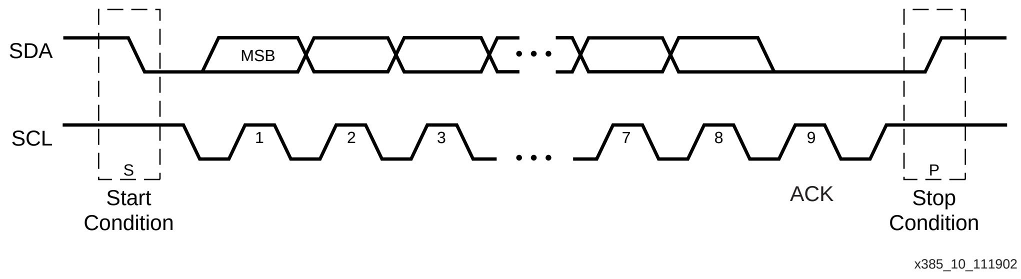


Figure 2: **Data Transfer on the I^2C Bus**

Each data packet on the I^2C bus consists of eight bits of data followed by an acknowledge bit so one complete data byte transfer requires nine clock pulses. Data is transferred with the most significant bit first (MSB). The transmitter releases the SDA line during the acknowledge bit and the receiver of the data transfer must drive the SDA line low during the acknowledge bit to acknowledge receipt of the data. If a Slave-receiver does not drive the SDA line Low during the acknowledge bit, this indicates that the Slave-receiver was unable to accept the data and the Master can then generate a STOP condition to abort the transfer. If the Master-receiver does not generate an acknowledge, this indicates to the Slave-transmitter that this byte was the last byte of the transfer.

Standard communication on the bus between a Master and a Slave is composed of four parts: START, Slave address, data transfer, and STOP. The I^2C protocol defines a data transfer format for both 7-bit and 10-bit addressing. The implementation of the I^2C controller in the Xilinx CoolRunner-II CPLD supports the seven-bit address format. After the START condition, a Slave address is sent. This address is seven bits long followed by an eighth-bit which is the read/write bit. A "1" indicates a request for data (read) and a "0" indicates a data transmission (write). Only the Slave with the calling address that matches the address transmitted by the Master responds by sending back an acknowledge bit by pulling the SDA line Low on the ninth clock.

Once successful Slave addressing is achieved, the data transfer can proceed byte-by-byte as specified by the read/write bit. The Master can terminate the communication by generating a STOP signal to free the bus. However, the Master may generate a START signal without generating a STOP signal first. This is called a repeated START.

CoolRunner-II I^{2}C Controller

The CoolRunner-II CPLD implementation of the I^{2}C Controller supports the following features:

- Microcontroller interface
- Master or Slave operation
- Multi-master operation
- Software selectable acknowledge bit
- Arbitration lost interrupt with automatic mode switching from Master to Slave
- Calling address identification interrupt with automatic mode switching from Master to Slave
- START and STOP signal generation/detection
- Repeated START signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- 100 KHz operation

Signal Descriptions

The I/O signals of the CoolRunner-II I^{2}C controller are described in Table 1. Pin numbers have not been assigned to this design, this can be done to meet the system requirements of the designer.

Table 1: **CoolRunner-II I^{2}C Controller Signal Description**

Name	Direction	Description
SDA	Bidirectional	**I^{2}C Serial Data.**
SCL	Bidirectional	**I^{2}C Serial Clock.**
ADDR_BUS[23:0]	Input	**µC Address Bus.**
DATA_BUS[7:0]	Bidirectional	**µC Data Bus**.
AS	Input	**Address Strobe**. Active Low µC handshake signal indicating that the address present on the address bus is valid.
DS	Input	**Data Strobe.** Active Low µC handshake signal indicating that the data present on the data bus is valid or that the µC is no longer driving the data bus and the I^{2}C Controller can place data on the data bus.
R_W	Input	**Read/Write.** "1" indicates a read, "0" indicates a write.
DTACK	Output	**Data Transfer Acknowledge.** Active Low µC handshake signal indicating that the I^{2}C Controller has placed valid data on the data bus for a read cycle or that the I^{2}C Controller has received the data on the bus for a write cycle.

Table 1: **CoolRunner-II I^{2}C Controller Signal Description**

IRQ	Output	**Interrupt Request.** Active Low.
MCF	Output	**Data Transferring Bit.** While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the ninth clock of a byte transfer. This bit is used to signal the completion of a byte transfer to the µC.
CLK	Input	**Clock.** This clock is input from the system. The constants used in generating a 100 KHz SCL signal assumes the frequency to be 1.832 MHz. Different clock frequencies can be used, but the constants in the VHDL source code must be recalculated.

Block Diagram

The block diagram of the CoolRunner-II I^{2}C Controller, shown in Figure 3 was broken into two major blocks, the µC interface and the I^{2}C interface.

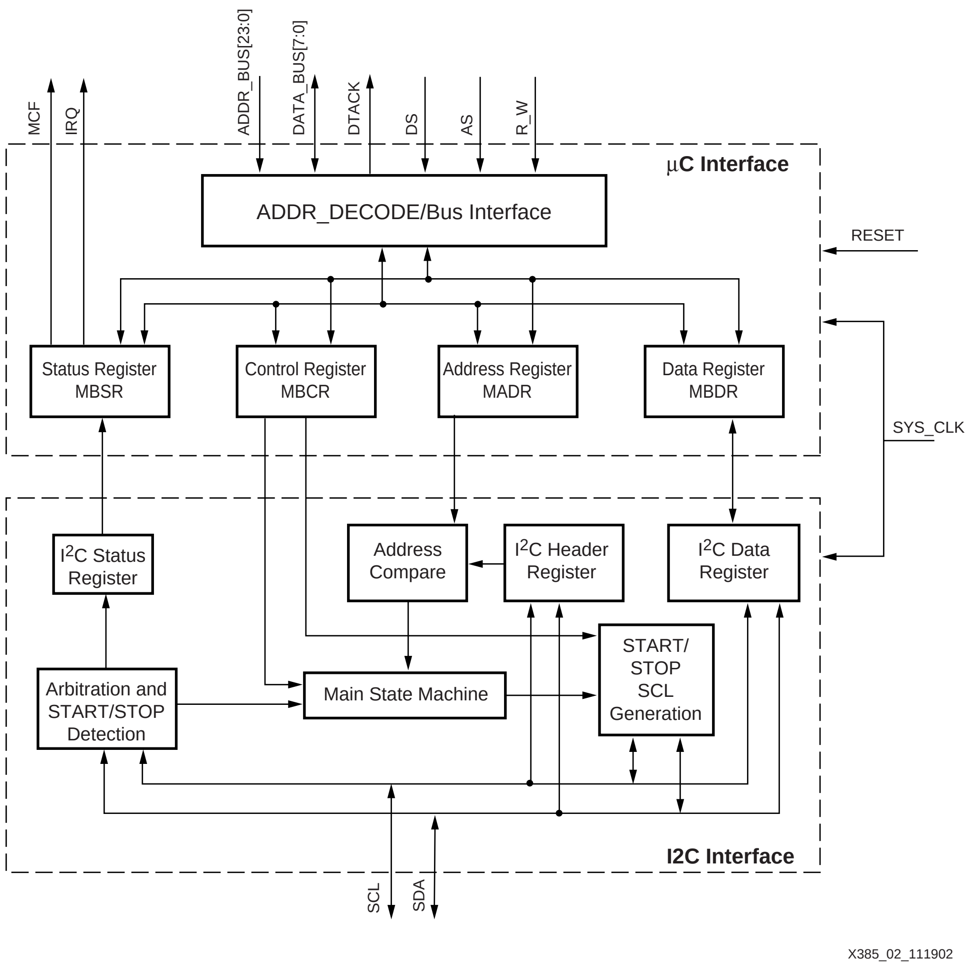


Figure 3: **CoolRunner-II I^{2}C Controller**

Microcontroller Logic

The µC interface for the I^{2}C controller design supports an asynchronous byte-wide bus protocol. This protocol is the method in which the µC reads and writes the registers in the design and is shown in Figure 4.

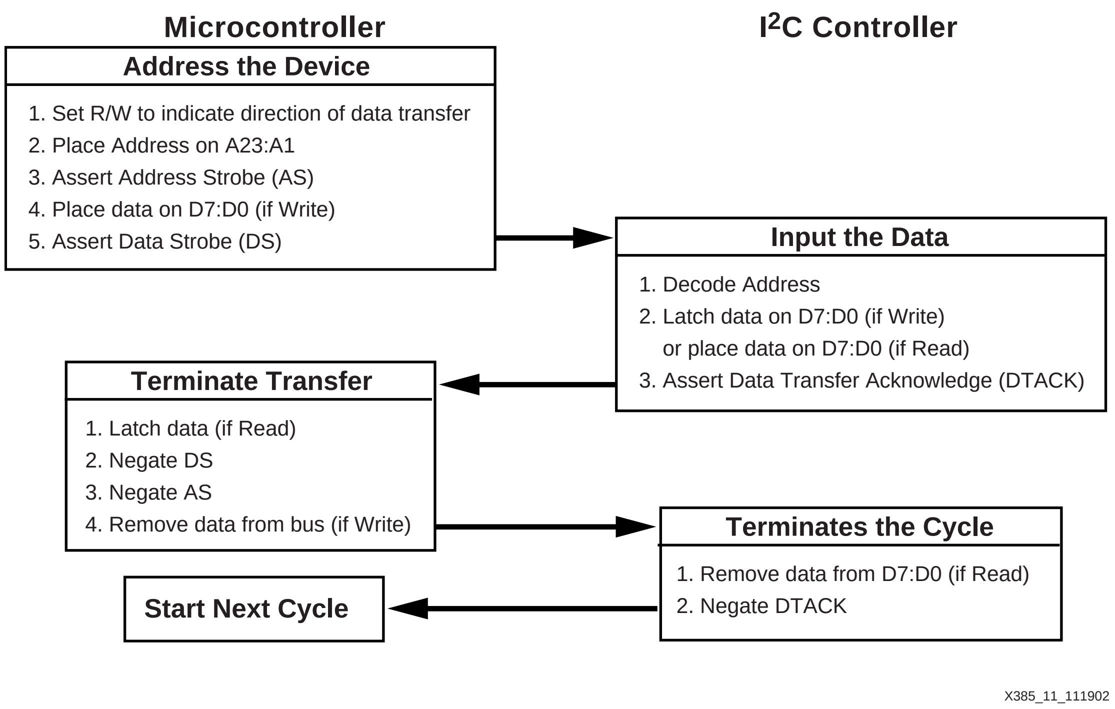

Microcontroller
I2C Controller
Address the Device
1. Set R/W to indicate direction of data transfer
2. Place Address on A23:A1
3. Assert Address Strobe (AS)
4. Place data on D7:D0 (if Write)
5. Assert Data Strobe (DS)
Input the Data
1. Decode Address
2. Latch data on D7:D0 (if Write)
or place data on D7:D0 (if Read)
3. Assert Data Transfer Acknowledge (DTACK)
Terminate Transfer
1. Latch data (if Read)
2. Negate DS
3. Negate AS
4. Remove data from bus (if Write)
Terminates the Cycle
1. Remove data from D7:D0 (if Read)
2. Negate DTACK
Start Next Cycle
X385_11_111902

Figure 4: **µC Read/Write Protocol**

Address Decode/Bus Interface Logic

The µC bus protocol is implemented in the CoolRunner-II I^{2}C Controller in the state machine shown in Figure 5.

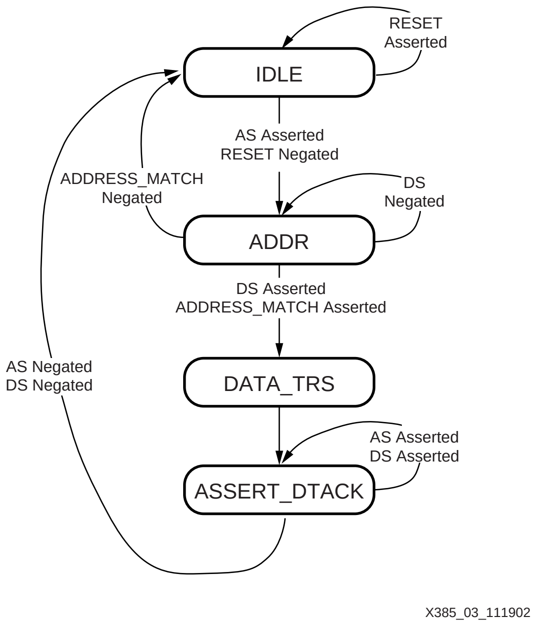


Figure 5: **µC Bus Interface State Machine**

In the first cycle, the µC places the address on the address bus, sets the read/write line to the correct state, and asserts address strobe (AS) and data strobe (DS). Address strobe indicates that the address present on the address bus is valid. If this is a write cycle, the µC also places the data on the data bus and DS indicates that valid data is present on the data bus. If this is a read cycle, the µC 3-states the data bus and DS indicates that the CoolRunner-II I^{2}C Controller can place data on the data bus.

Upon the assertion of AS, the CoolRunner-II I^{2}C Controller transitions to the ADDR state to decode the address and determine if it is the device being addressed. The enables for the internal registers are set in this state. If the CoolRunner-II I^{2}C Controller is being addressed and DS is asserted, the CoolRunner-II I^{2}C controller progresses to the DATA_TRS state. If this is a read cycle, the requested data is placed on the bus and if this is a write cycle, the data from the data bus is latched in the addressed register. The CoolRunner-II I^{2}C Controller automatically progresses to the ASSERT_DTACK state and asserts DTACK indicating that the data requested is ready if a read cycle or that the data has been received if a write cycle.

Upon the assertion of DTACK, the µC either removes data from the bus if this is a write cycle, or latches the data present on the bus if this is a read cycle. The read/write line is set to read and AS and DS are negated to indicate that the data transfer is complete. The negation of AS and DS causes the CoolRunner-II I^{2}C Controller to negate DTACK and transition to the IDLE state.

CoolRunner-II I^{2}C Controller Registers

The base address used for address decoding is set in the VHDL code via the constant BASE_ADDRESS. The base address is the upper 16 bits of the address bus. The lower address bits determine which register is being accessed.

The registers supported in the CoolRunner-II I^{2}C Controller are described in the Table 2. The µC interface logic of the CoolRunner-II I^{2}C Controller handles the reading and writing of these registers by the µC and supplies and/or retrieves these bits to/from the I^{2}C interface logic.

Table 2: **I^{2}C Controller Registers**

Address	Register	Description
MBASE + $8Dh	MADR	I^{2}C Address Register
MBASE + $91h	MBCR	I^{2}C Control Register
MBASE + $93h	MBSR	I^{2}C Status Register
MBASE + $95h	MBDR	I^{2}C Data I/O Register

Address Register (MADR)

This field contains the specific Slave address to be used by the I^{2}C Controller. This register is read/write. (Table 3).

Table 3: **Address Register Bits**

Bit Location	Name	µC Access	Description
7-1	Slave Address	Read/Write	Address used by the I^{2}C controller when in Slave mode.
0	-	-	Unused

Control Register (MBCR)

This register contains the bits to configure the I^{2}C controller. (Table 4).

Table 4: **Control Register Bits**

Bit Location	Name	μC Access	Description
7	MEN	Read/Write	**I^{2}C Controller Enable.** This bit must be set before any other MBCR bits have any effect "1" enables the I^{2}C controller "0" resets and disables the I^{2}C controller
6	MIEN	Read/Write	**Interrupt Enable.** "1" enables interrupts. An interrupt occurs if MIF bit in the status register is also set "0" disable interrupts but does not clear any currently pending interrupts
5	MSTA	Read/Write	**Master/Slave Mode Select.** When the μC changes this bit from "0" to "1", the I^{2}C controller generates a START condition in Master mode. When this bit is cleared, a STOP condition is generated and the I^{2}C controller switches to Slave mode. If this bit is cleared, however, because arbitration for the bus has been lost, a STOP condition is not generated.
4	MTX	Read/Write	**Transmit/Receive Mode Select.** This bit selects the direction of Master/Slave transfers. "1" selects an I^{2}C Master transmit "0" selects an I^{2}C Master receive
3	TXAK	Read/Write	**Transmit Acknowledge Enable.** This bit specifies the value driven onto the SDA line during acknowledge cycles for both Master and Slave receivers "1" - ACK bit = "1" - no acknowledge "0" - ACK bit = "0" - acknowledge Since Master receivers indicate the end of data reception by not acknowledging the last byte of the transfer, this bit is the means for the μC to end a Master receiver transfer.
2	RSTA	Read/Write	**Repeated Start.** Writing a "1" to this bit generates a repeated START condition on the bus if the I^{2}C controller is the current bus Master. This bit is always read as "0". Attempting a repeated START at the wrong time if the bus is owned by another Master results in a loss of arbitration.
1-0	Reserved		

Status Register (MBSR)

This register contains the status of the I^{2}C controller. This status register is read-only with the exception of the MIF and MAL bits, which are software clearable. All bits are cleared upon reset except the MCF and RXAK bits. (Table 5).

Table 5: **Status Register Bits**

Bit Location	Name	µC Access	Description
7	MCF	Read	**Data Transferring Bit.** While one byte of data is being transferred, this bit is cleared. It is set by the rising edge of SCL during the acknowledge cycle of the transfer and is only High for this SCL clock period. "1" transfer is complete "0" transfer in progress Note that in the CoolRunner-II I^{2}C controller, this bit is also an output pin so that a register read cycle is not required to determine that a transfer is complete.
6	MAAS	Read	**Addressed as Slave Bit.** When the address on the I^{2}C bus matches the Slave address in the MADR register, the I^{2}C controller is being addressed as a Slave and switches to Slave mode.
5	MBB	Read	**Bus Busy Bit.** This bit indicates the status of the I^{2}C bus. This bit is set when a START condition is detected and cleared when a STOP condition is detected. "1" indicates the bus is busy "0" indicates the bus is idle
4	MAL	Read Software Clearable	**Arbitration Lost Bit.** This bit is set by hardware when arbitration for the I^{2}C bus is lost. This bit must be cleared by the µC software writing a "0" to this bit.
3	Reserved		

Table 5: **Status Register Bits *(Continued)***

Bit Location	Name	µC Access	Description
2	SRW	Read	**Slave Read/Write Bit.** When the I^{2}C controller has been addressed as a Slave (MAAS is set), this bit indicates the value of the read/write bit sent by the Master. This bit is only valid when a complete transfer has occurred and no other transfers have been initiated. "1" indicates Master reading from Slave "0" indicates Master writing to Slave
1	MIF	Read Software Clearable	**Interrupt Bit.** This bit is set when an interrupt is pending, which causes a processor interrupt request if MIEN is set. This bit must be cleared by the µC software writing a "0" to this bit in the interrupt service routine.
0	RXAK	Read	**Received Acknowledge Bit.** This bit reflects the value of the SDA signal during the acknowledge cycle of the transfer. "1" indicates that no acknowledge was received "0" indicates that an acknowledge was received

Data Register (MBDR)

This register contains data to/from the I^{2}C bus. Physically, this register is implemented by two byte-wide registers at the same address, one for the I^{2}C transmit data and one for the I^{2}C received data. This eliminates any possible contention between the µC and the CoolRunner-II I^{2}C Controller. Since these registers are at the same address they appear as the same register to the µC and will continue to be described as such. In transmit mode, data written into this register is output on the I^{2}C bus, in receive mode, this register contains the data received from the I^{2}C bus. Note that in receive mode, it is assumed that the µC will be able to read this register during the next I^{2}C transfer. The received I^{2}C data is placed in this register after each complete transfer, the I^{2}C interface logic does not wait for an indication from the µC that this register has been read before proceeding with the next transfer. (Table 6)

Table 6: **I^{2}C Data Register Bit**

Bit Location	Name	µC Access	Description
7-0	D7:D0	Read/Write	I^{2}C Data

I^{2}C Interface Logic

The I^{2}C bus interface logic consists of several different processes as seen in Figure 3. Control bits from the µC interface registers determine the behavior of these processes.

Arbitration

Arbitration of the I^{2}C bus is lost in the following circumstances:

- The SDA signal is sampled as a "0" when the Master outputs a "1" during an address or data transmit cycle
- The SDA signal is sampled as a "0" when the Master outputs a "1" during the acknowledge bit of a data receive cycle
- A start cycle is attempted when the bus is busy

- A repeated start cycle is requested in Slave mode
- A STOP condition is detected when the Master did not request it

If the CoolRunner-II I^{2}C Controller is in Master mode, the outgoing SDA signal is compared with the incoming SDA signal to determine if control of the bus has been lost. The SDA signal is checked only when SCL is High during all cycles of the data transfer except for acknowledge cycles to insure that START and STOP conditions are not generated at the wrong time. If the outgoing SDA signal and the incoming SDA signals differ, then arbitration is lost and the MAL bit is set. At this point, the CoolRunner-II I^{2}C Controller switches to Slave mode and resets the MSTA bit.

The CoolRunner-II I^{2}C design will not generate a START condition while the bus is busy, however, the MAL bit will be set if the µC requests a START or repeated START while the bus is busy. The MAL bit is also set if a STOP condition is detected when this Master did not generate it.

If arbitration is lost during a byte transfer, SCL continues to be generated until the byte transfer is complete.

START/STOP Detection

This process monitors the SDA and SCL signals on the I^{2}C bus for START and STOP conditions. When a START condition is detected, the Bus Busy bit is set. This bit stays set until a STOP condition is detected. The signals, DETECT_START and DETECT_STOP are generated by this process for use by other processes in the logic. Note that this logic detects the START and STOP conditions even when the CoolRunner-II I^{2}C Controller is the generator of these conditions.

Generation of SCL, SDA, START and STOP Conditions

This process generates the SCL and SDA signals output on the I^{2}C bus when in Master mode. The clock frequency of the SCL signal is ~100 KHz and is determined by dividing down the input clock. The number of input clock cycles required for generation of a 100 KHz SCL signal is set by the constant CNT_100 KHZ and is currently calculated for a system clock of 1.832 MHz. This constant can easily be modified by a designer based on the clock available in the target system. Likewise, the constants START_HOLD and DATA_HOLD contain the number of system clock cycles required to meet the I^{2}C requirements on hold time for the SDA lines after generating a START condition and after outputting data.

The state machine that generates SCL and SDA when in Master mode is shown in Figure 6. Note that SCL and SDA are held at the default levels if the bus is busy. This state machine generates the controls for the system clock counter.

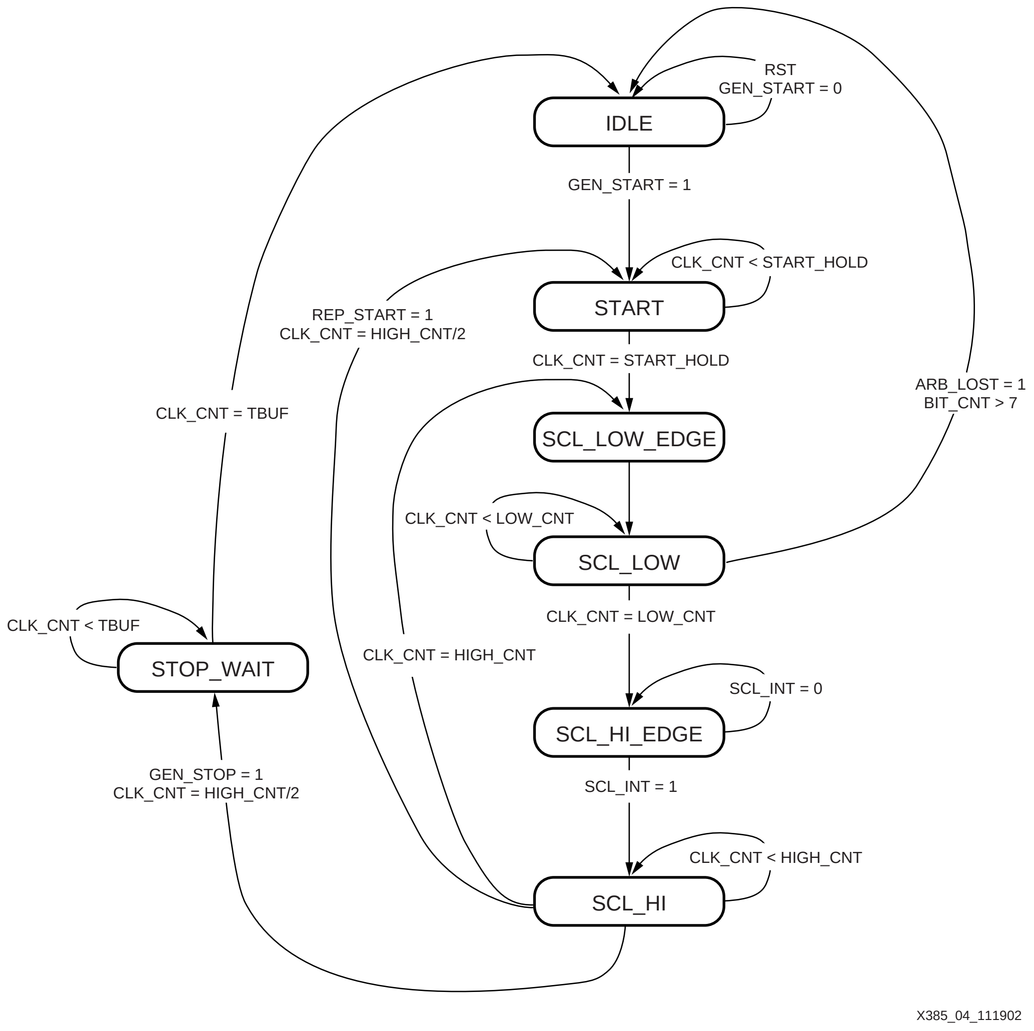


Figure 6: **SCL, SDA, START, and STOP Generation State Machine**

The internal SDA signal output from this design is either the SDA signal generated by this state machine for START and STOP conditions or the data from the MBDR register when the CoolRunner-II I^{2}C Controller is in transmit mode. Note that both SCL and SDA are open-collector outputs, therefore, they are only driven to a "0". When a "1" is to be output on these signals, the CoolRunner-II I^{2}C Controller 3-states their output buffers. The logic in the design will set internal SDA and SCL signals to "1" or "0". These internal signals actually control the output enable of the 3-state buffer for these outputs.

In the IDLE state, SCL and SDA are 3-stated, allowing any Master to control the bus. Once a request has entered to generate a start condition, the CoolRunner-II I^{2}C Controller is in Master mode, and the bus is not busy, the state machine transitions to the START state.

The START state holds SCL High, but drives SDA Low to generate a START condition. The system clock counter is started and the state machine stays in this state until the required hold time is met. At this point, the next state is SCL_LOW_EDGE.

The SCL_LOW_EDGE state simply creates a falling edge on SCL and resets the system clock counter. On the next clock edge, the state machine moves to state SCL_LOW. In this state, the SCL line is held Low and the system clock counter begins counting. If the REP_START signal is asserted then the SDA signal will be set High, if the GEN_STOP signal is asserted, SDA will be set Low.

When the SCL low time has been reached, the state machine will transition to the IDLE state if arbitration has been lost and the byte transfer is complete to insure that SCL continues until the end of the transfer. Otherwise the next state is the SCL_HI_EDGE state.

The SCL_HI_EDGE state generates a rising edge on SCL by setting SCL to "1". Note, however, that the state machine will not transition to the SCL_HI state until the sampled SCL signal is also High to obey the clock synchronization protocol of the I^{2}C specification. Clock synchronization is performed using the wired-AND connection of the SCL line. The SCL line will be held Low by the device with the longest low period. Devices with shorter low periods enter a high wait state until all devices have released the SCL line and it goes High. Therefore the SCL_HI_EDGE state operates as the high wait state as the SCL clock is synchronized.

The SCL_HI state then starts the system clock counter to count the high time for the SCL signal. If a repeated START or a STOP condition has been requested, the state machine will transition to the appropriate state after half of the SCL high time so that the SDA line can transition as required. If neither of these conditions has been requested, then the state machine transitions to the SCL_LOW_EDGE state when the SCL high time has been achieved.

The STOP_WAIT state is used to insure that the hold time requirement after a STOP condition is met.

I^{2}C Interface Main State Machine

The main state machine for the I^{2}C Interface logic is shown in Figure 7. This state machine is the same for both Slave and Master modes. In each state, the mode is checked to determine the proper output values and next state conditions. This allows for immediate switching from

Master to Slave mode if arbitration is lost or if the CoolRunner-II I2C Controller is addressed as a Slave.

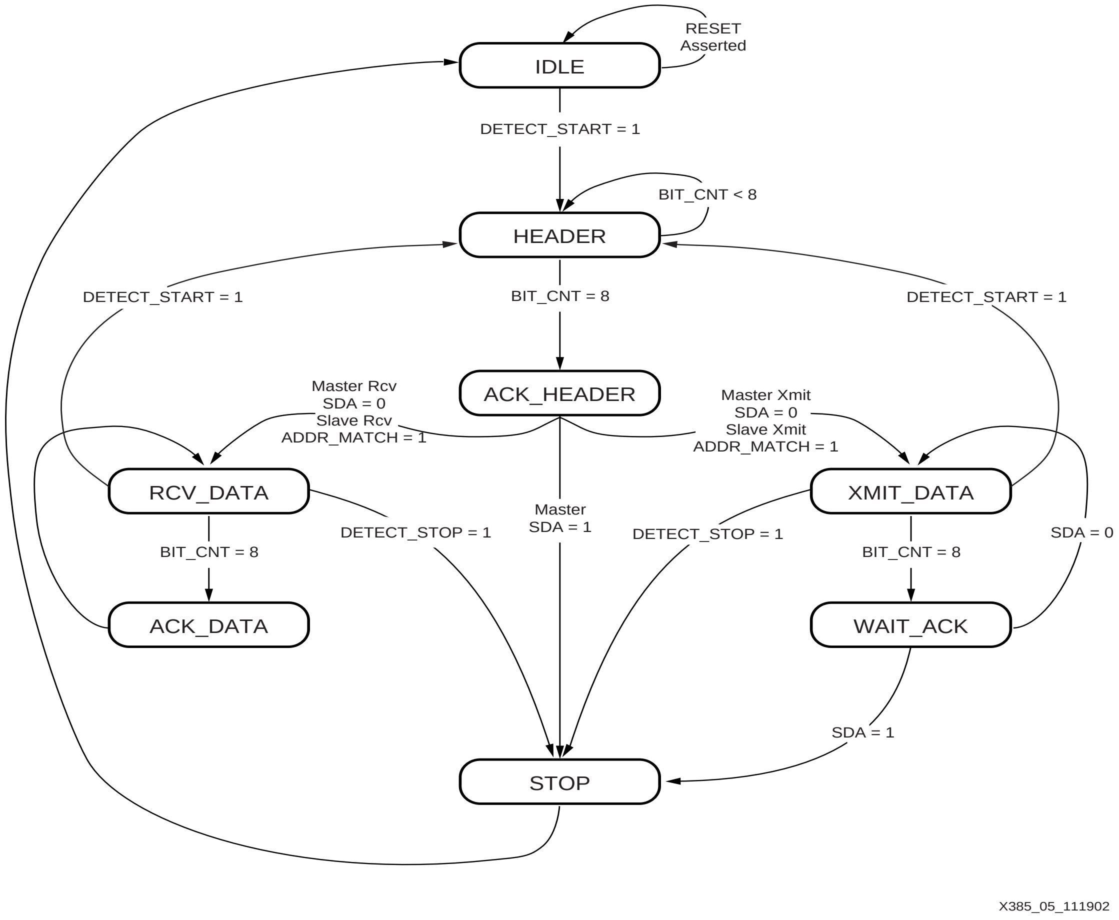


Figure 7: **I2C Interface Main State Machine**

This state machine utilizes and controls a counter that counts the I2C bits that have been received. This count is stored in the signal BIT_CNT. This state machine also controls two shift registers, one that stores the I2C header that has been received and another that stores the I2C data that has been received or is to be transmitted.

Note:

This state machine and the associated counters and shift registers are clocked on the falling edge of the incoming SCL clock. If the load is heavy on the SCL line, the rise time of the SCL signal may be very slow which can cause susceptibility to noise for some systems. This can be particularly dangerous on a clock signal. The designer is strongly encouraged to investigate the signal integrity of the SCL line and if necessary, use external buffers for the SCL signal.

When a START signal has been detected, the state machine transitions from the IDLE state to the HEADER state. The START signal detection circuit monitors the incoming SDA and SCL lines to detect the START condition. The START condition can be generated by the CoolRunner-II I2C controller or another Master—either source will transition the state machine to the HEADER state.

The HEADER state is the state where the I2C header is transmitted on the I2C bus from the MBDR register if in Master mode. In this state, the incoming I2C data is captured in the I2C Header shift register. In Master mode, the I2C Header shift register will contain the data that was just transmitted by this design. When all eight bits of the I2C header have been shifted in, the state machine transitions to the ACK_HEADER state.

In the ACK_HEADER state, the CoolRunner-II I^{2}C design samples the SDA line if in Master mode to determine whether the addressed I^{2}C Slave acknowledged the header. If the addressed Slave does not acknowledge the header, the state machine will transition to the STOP state which signals the SCL/START/STOP generator to generate a STOP. If the addressed Slave has acknowledged the address, then the LSB of the I^{2}C header is used to determine if this is a transmit or receive operation and the state machine transitions to the appropriate state to either receive data, RCV_DATA, or to transmit data, XMIT_DATA.

The I^{2}C Header shift register is constantly compared with the I^{2}C address set in the MADR register. If these values match in the ACK_HEADER state, the CoolRunner-II I^{2}C Controller has been addressed as a Slave and the mode immediately switches to Slave mode. The MAAS bit is then set in the MBSR status register. The SDA line will be driven as set in the TXAK register to acknowledge the header to the current I^{2}C bus Master. Again, the LSB of the I^{2}C header is used to determine the direction of the data transfer and the appropriate state is chosen.

The RCV_DATA state shifts the incoming I^{2}C data into the I^{2}C shift register for transfer to the μC. When the whole data byte has been received, the state machine transitions to the ACK_DATA state and the value of the TXAK register is output on the SDA line to acknowledge the data transfer. Note that in Master mode, the indication that the Slave has transmitted the required number of data bytes is to not acknowledge the last byte of data. The μC must negate the TXAK bit to prohibit the ACK of the last data byte. The state machine exits this pair of states when a STOP condition has been detected, otherwise, the transition between these two states continues. In Master mode, the μC requests a STOP condition by negating the MSTA bit.

The XMIT_DATA state shifts the data from the I^{2}C data register to the SDA line. When the entire byte has been output, the state machine transitions to the WAIT_ACK state. If an acknowledge is received, the state machine goes back to the XMIT_DATA to transmit the next byte of data. This pattern continues until either a STOP condition is detected, or an acknowledge is not received for a data byte.

Note that the data transfer states of this state machine assume that the μC can keep up with the rate at which data is received or transmitted. If interrupts are enabled, an interrupt is generated at the completion of each byte transfer. The MCF bit is set as well providing the same indication. Data is transferred to/from the I^{2}C data register to/from the μC data register during the acknowledge cycle of the data transfer. The state machine does not wait for an indication that the μC has read the received data or that new data has been written for transmission. The designer should be aware of the effective data rate of the μC to insure that this is not an issue.

The STOP state signals the SCL/START/STOP generator to generate a STOP condition if the CoolRunner-II I^{2}C design is in Master mode. The next state is always the IDLE state and the I^{2}C activity is completed.

Operational Flow Diagrams

The flow of the interface between the µC and the CoolRunner-II I^{2}C Controller is detailed in the following flow charts. These flow charts are meant to be a guide for utilizing the CoolRunner-II I^{2}C Controller in a µC system.

Initialization

Before the CoolRunner-II I^{2}C Controller can be utilized, certain bits and registers must be initialized as shown in Figure 8.

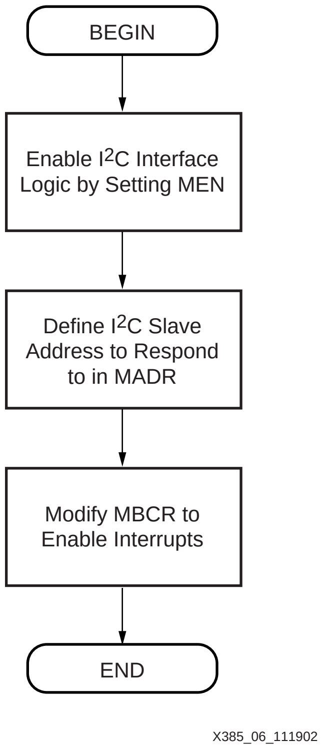

BEGIN
Enable I2C Interface Logic by Setting MEN
Define I2C Slave Address to Respond to in MADR
Modify MBCR to Enable Interrupts
END
X385_06_111902

Figure 8: **CoolRunner-II I^{2}C Controller Initialization Flow Chart**

Master Transmit/Receive

The flow charts for transmitting data and receiving data while I^{2}C bus Master are shown in Figure 9 and Figure 10. The major difference between transmitting and receiving is the additional step in the Master Receive flow chart of turning off the acknowledge bit on the second to last data word.

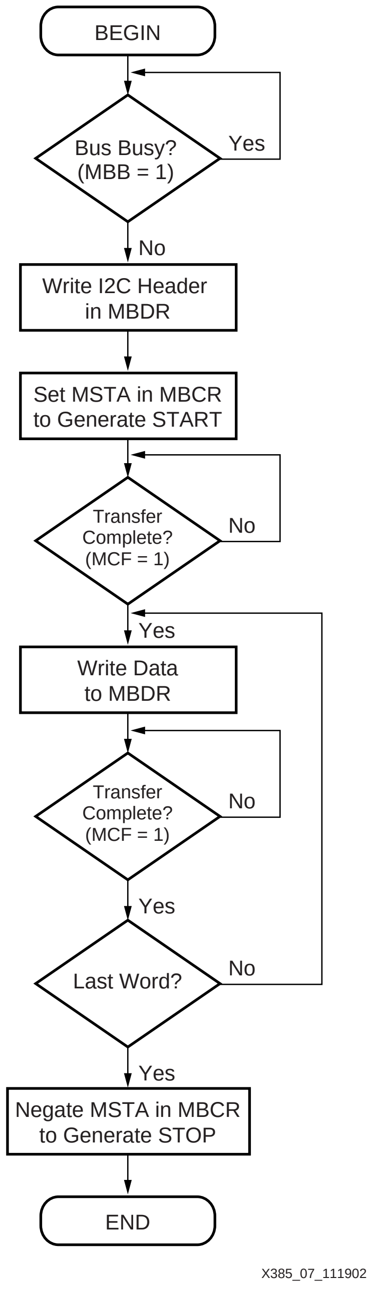

BEGIN
Bus Busy? (MBB = 1)
Yes
No
Write I2C Header in MBDR
Set MSTA in MBCR to Generate START
Transfer Complete? (MCF = 1)
No
Yes
Write Data to MBDR
Transfer Complete? (MCF = 1)
No
Yes
Last Word?
No
Yes
Negate MSTA in MBCR to Generate STOP
END
X385_07_111902

Figure 9: **Master Transmit Flow Chart**

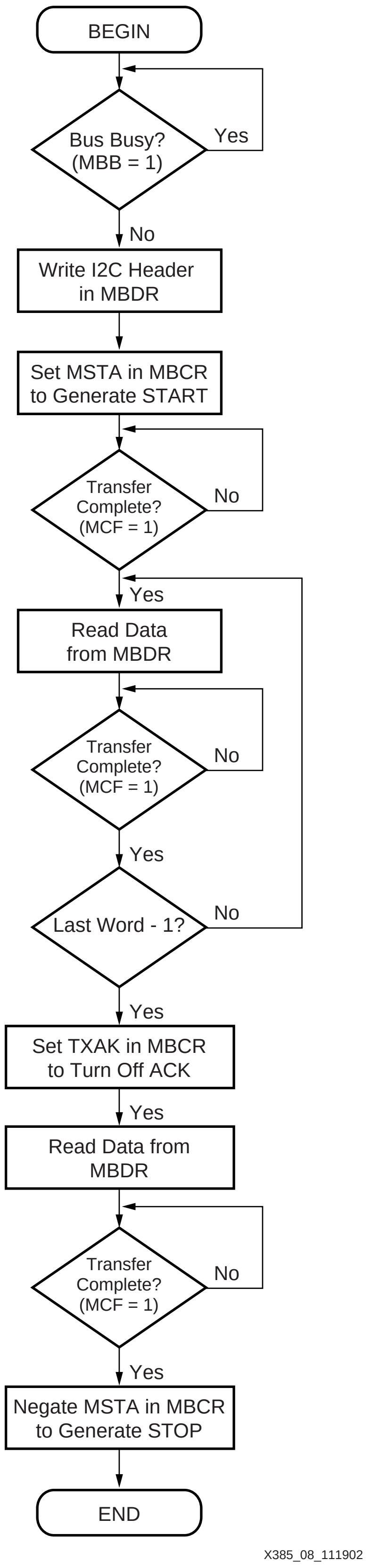

BEGIN
Bus Busy? (MBB = 1)
Yes
No
Write I2C Header in MBDR
Set MSTA in MBCR to Generate START
Transfer Complete? (MCF = 1)
No
Yes
Read Data from MBDR
Transfer Complete? (MCF = 1)
No
Yes
Last Word - 1?
No
Yes
Set TXAK in MBCR to Turn Off ACK
Yes
Read Data from MBDR
Transfer Complete? (MCF = 1)
No
Yes
Negate MSTA in MBCR to Generate STOP
END
X385_08_111902

Figure 10: **Master Receive Flow Chart**

Slave Flow Chart

The flow chart for receiving or transmitting data in Slave mode is shown in Figure 11. If in receive mode, the first read from the MBDR register is a dummy read because data has not yet been received. Since the CoolRunner-II I²C Controller is in Slave mode, the only way to know that the transaction is complete is to check that the bus is busy and that the Addressed as Slave bit is still set.

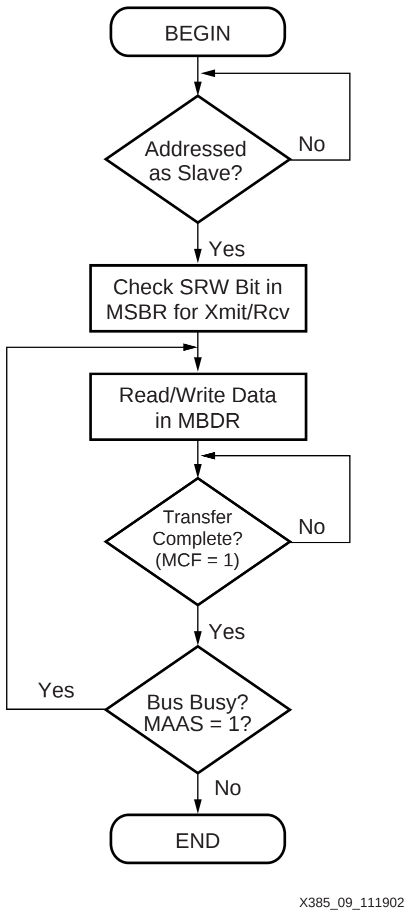


Figure 11: **Slave/Transmitter Flow Chart**

CoolRunner-II CPLD Implementation

The design of the CoolRunner-II I²C Controller was implemented in VHDL and targeted to a 256 macrocell CoolRunner-II CPLD in a 144-pin TQFP package (XC2C256-5TQ144) using Xilinx Project Navigator. (Xilinx Project Navigator software is available free-of-charge from the Xilinx website: www.xilinx.com/products/software/webpowered.htm.).

Note:

Since the system clock frequency was 1.832 MHz, the speed of the design was not critical and any speed grade part could have been used.

Note:

The I²C SCL line is used as a clock input into the CoolRunner-II I²C Controller. If there are many loads on the I²C bus, the rise time of the SCL line can be quite slow. The CoolRunner-II CPLD for this design requires a rise time no greater than 20ns, therefore, the designer is strongly encouraged to examine the characteristics of the SCL signal in the I²C system. If the rise time of the I²C signals are greater than 20 ns, the inputs can be configured as Schmitt Triggers providing for input hysteresis and noise immunity. Schmitt Trigger inputs will prevent adverse effects of slow rising/falling input signals.

The I²C design utilization in a CoolRunner-II 256-macrocell device is shown in Table 7. This utilization was achieved using certain fitter parameters, actual results may vary. As shown, there is plenty of room remaining in the device for the implementation of other logic in the system.

Table 7: **CoolRunner-II CPLD 256-Macrocell Utilization**

Resource	Available	Used	Utilization
Macrocells	256	127	50%
P-terms	896	352	39%
Registers Used	256	110	43%
I/O Pins	118	42	36%
Function Block Inputs Used	640	305	48%

Design Verification

The Xilinx Project Navigator software package outputs a timing VHDL model of the fitted design. This post-fit VHDL was simulated with the original VHDL test benches to insure design functionality. Also, the CoolRunner-II I²C Controller design was simulated with an independently generated VHDL model of an I²C Slave design to verify that the interface specifications were implemented correctly. Please note that all verification of this design has been done through simulations.

VHDL Code Download

VHDL source code and test benches are available for this design. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design has not been verified on hardware (as opposed to simulations), and it should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

VHDL - **ftp://ftp.xilinx.com/pub/applications/refdes/xapp333.zip**

Verilog - **ftp://ftp.xilinx.com/pub/applications/refdes/xapp333_ver.zip**

Disclaimer

I²C is a registered trademark of Philips Electronics N.V.. Xilinx provides this reference design as one possible implementation of this standard and claims no rights to this interface. To use this reference design, you must contact Philips to obtain any necessary rights associated with this interface.

Conclusion

This document has detailed the design of an I²C Controller design for a CoolRunner-II CPLD. Though the design has been extensively verified in simulations, Xilinx assumes no responsibility for the accuracy or the functionality of this design.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/24/02	1.0	Initial Xilinx release.
12/30/03	1.1	Change to control register address.

XILINX®

XAPP394 (v1.1) December 1, 2003

Interfacing to Mobile SDRAM with CoolRunner-II CPLDs

Summary

This document describes the VHDL design for interfacing CoolRunner™-II CPLDs with low power Mobile SDRAM memory devices. Mobile SDRAM is the ideal memory solution for wireless, handheld, and mobile computing applications, making this a perfect match with the Xilinx CoolRunner-II low power CPLD family. The VHDL code described here can be found in **VHDL Code**, page 62.

Introduction

CoolRunner-II CPLDs are the latest CPLD product offering from Xilinx. CoolRunner-II CPLDs combine high performance with low power operation. Standby current on CoolRunner-II CPLD devices is less than 100µA. More information on the CoolRunner-II CPLD family can be found at **http://www.xilinx.com/cr2**.

Key features of the CoolRunner-II CPLD family include DualEDGE triggered registers, a global clock divider, and voltage referenced I/O standards. These features provide the capability to interface a CoolRunner-II CPLD with low power memory devices such as Mobile SDRAM. Mobile SDRAM manufacturers include Micron and Samsung with data sheets available in **References**, page 62.

Signal Definitions

It is important to define the connections in this design before describing the CPLD logic to interface with the Mobile SDRAM. Table 1 defines the Mobile SDRAM interface signals described in this document.

Table 1: **Mobile SDRAM Signal Definitions**

Manufacturer Specification	Xilinx CPLD VHDL Code	Description
CLK	sdram_clk	Clock input to SDRAM. All input signals are referenced to positive edge of CLK.
CKE	sdram_cke	Clock enable.
CS#	sdram_cs	Command signals that define current operation.
RAS#	sdram_ras	
CAS#	sdram_cas	
WE#	sdram_we	
LDQM, UDQM	sdram_udqm, sdram_ldqm	Mask signal for write data operations (LDQM corresponds to DQ[7:0], while UDQM corresponds to DQ[15:8]).

Table 1: **Mobile SDRAM Signal Definitions**

Manufacturer Specification	Xilinx CPLD VHDL Code	Description
BA[1:0]	sdram_ba[1:0]	2-bit bank address bus.
A[12:0]	sdram_a[11:0]	13-bit row and column address bus.
DQ[15:0]	sdram_dq[7:0]	Bidirectional 16-bit data bus.

Mobile SDRAM

Mobile SDRAM devices feature low power technology to meet the needs of handheld electronics. Additional power savings are achieved with two self-refresh features, temperature-compensated self-refresh (TCSR) and partial-array self-refresh (PASR).

Read and write accesses to the Mobile SDRAM are burst-oriented, starting at a specific address. SDRAM is organized in banks, where each data location can be represented with a row and column address. Each access must begin with an ACTIVE command followed by the READ or WRITE operation. The ACTIVE command selects the bank and row address, while the individual READ or WRITE command select the column address. The Mobile SDRAM features AUTO PRECHARGE, in which the row being accessed is initiated with a PRECHARGE command at the end of the burst sequence.

The operation of the Mobile SDRAM device can be customized by writing different values to the mode register and extended mode register. Each register allows the designer to set parameters for interfacing with the memory device.

The following commands are available to execute with Mobile SDRAM devices:

- NOP (Deselect SDRAM device)
- ACTIVE (Opens row in specified bank for access)
- READ (Select bank and column, and start READ burst)
- WRITE (Select bank and column, and start WRITE burst)
- DEEP POWER DOWN (Maximum power savings, data is not retained)
- PRECHARGE (Deactivates open row in specified bank or all banks)
- AUTO REFRESH or SELF REFRESH (Retains data in SDRAM)
- LOAD MODE REGISTER (Defines operating mode of SDRAM)

More information on the Mobile SDRAM devices targeted in this design can be found in **References**, page 62.

CPLD Design

A CoolRunner-II CPLD is utilized as the controller for interfacing to the Mobile SDRAM memory device. The CPLD is responsible for interpreting the system level commands and creating the interface requirements to the Mobile SDRAM. Figure 1 illustrates the main control logic in the CoolRunner-II CPLD.

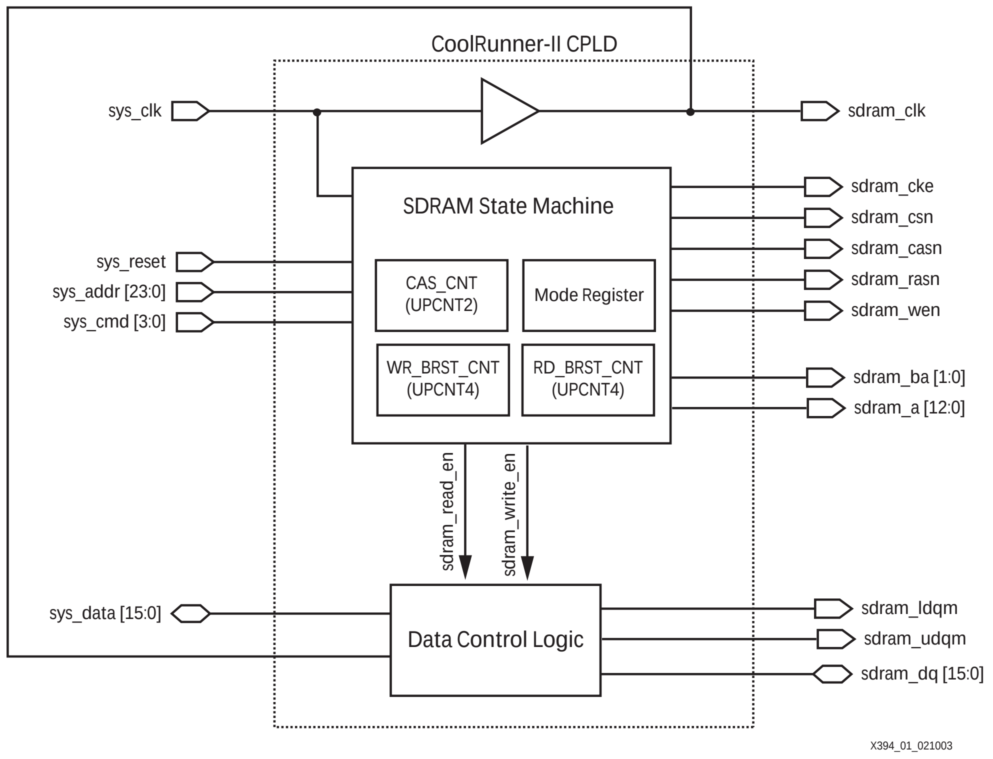


Figure 1: **CPLD Block Diagram**

The system level interface is illustrated in this design as an example interface. This system interface example is an asynchronous communication, with the signals registered in the CPLD. The SDRAM state machine is the main controller in this design and is responsible for generating the control, data, and address signals to the Mobile SDRAM device. The SDRAM state machine also asserts control signals that are used internally by the CPLD for reading/writing data and generating the SDRAM address signals. The SDRAM state machine in this design utilizes a 2-bit counter for waiting the CAS latency in a READ cycle, and two 4-bit counters, one for the length of a WRITE cycle burst and one for the length of a READ cycle burst.

A detailed description of the SDRAM state machine is illustrated in Figure 2. The SDRAM state machine remains in the IDLE state waiting for the next instruction to execute. The next instruction to execute is interpreted from the system command bus, sys_cmd. In this design, AUTO PRECHARGE is utilized, where an ACTIVE command must be issued prior to any READ or WRITE operation.

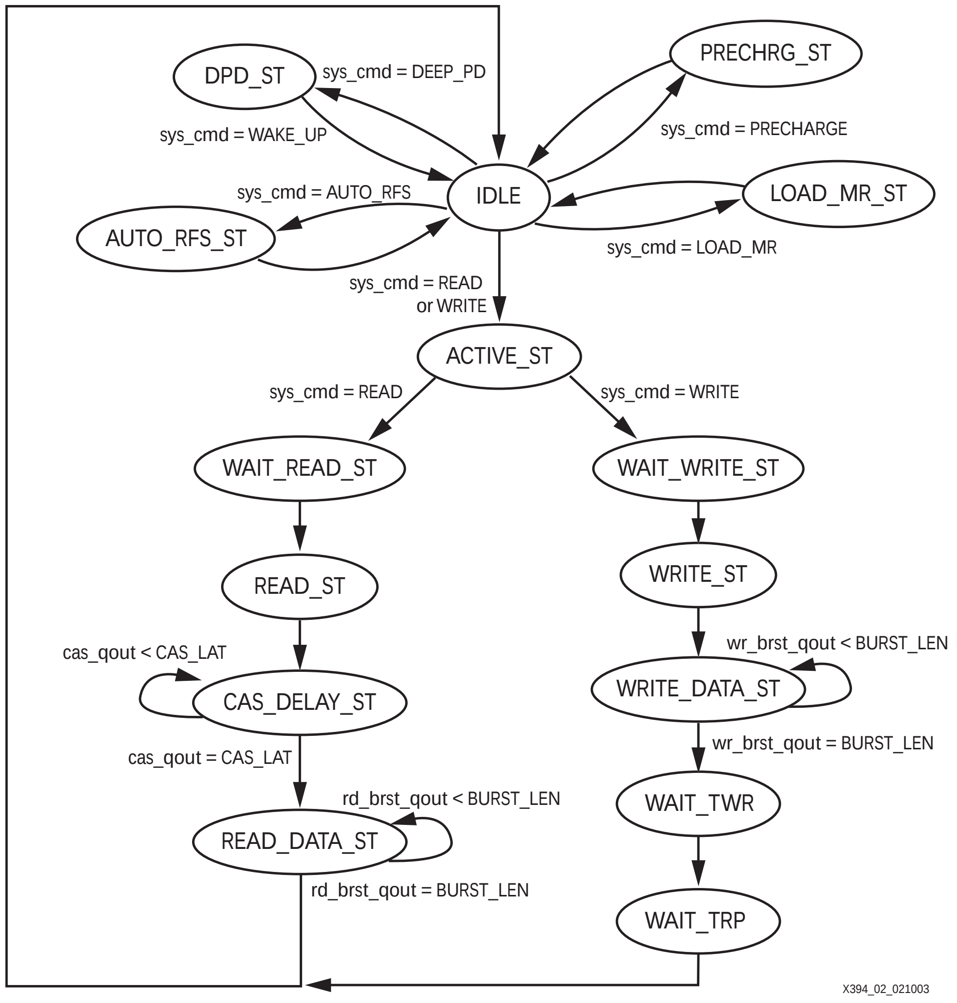


Figure 2: **SDRAM State Machine**

The function of each state in the SDRAM state machine is described in Table 2.

Table 2: **SDRAM State Machine State Description**

State Name	Function
IDLE	Assert NOP instruction control signals to SDRAM. Determines next operation to execute based on the value of sys_cmd signal.
AUTO_RFS_ST	Assign SDRAM command values to execute AUTO REFRESH instruction. Auto refresh retains data in SDRAM.
PRECHRG_ST	Assign SDRAM command values to execute PRECHARGE instruction. Precharge command deactivates specified row in one bank or all banks.
LOAD_MR_ST	Assign SDRAM command to execute LOAD MODE REGISTER. The mode register or extended mode register can be written to in this state, determined by bank address, sdram_ba signal.
DPD_ST	Executes DEEP POWER DOWN command. Data in SDRAM device is not retained in this state. Waits WAKE_UP for system command to return to IDLE state.

Table 2: **SDRAM State Machine State Description**

State Name	Function
ACTIVE_ST	Assign SDRAM command value to execute an ACTIVE command. Assign row address to SDRAM address bus.
WAIT_READ_ST	Execute NOP instruction. Necessary to meet timing specification for t_{RCD}.
READ_ST	Issue READ instruction by assigning SDRAM signals. Assign column address to address bus.
CAS_DELAY_ST	Wait for specified CAS latency of READ operation. Enable CAS latency counter, cas_cnt_en, is asserted.
READ_DATA_ST	Read data from SDRAM. Assert sdram_read_en signal to data control logic block. Enable rd_brst_qout counter. Remain in this state for length of specified burst to capture all data.
WAIT_WRITE_ST	Execute NOP instruction. Necessary to meet timing specification for t_{RCD}.
WRITE_ST	Assign WRITE command values to SDRAM to issue WRITE instruction. Assign column address to address bus.
WRITE_DATA_ST	Enable data write to SDRAM by asserting sdram_write_en signal to data control logic block. Enable wr_brst_qout counter. Wait for end of write burst length.
WAIT_TWR	Execute NOP instruction. Necessary to meet timing specification for write recovery, t_{WR}.
WAIT_TRP	Execute NOP instruction. Necessary to meet timing specification for precharge command period, t_{RP}.

Device Utilization

The SDRAM interface design utilizes a CoolRunner-II XC2C128-4TQ144 device. The system interface, SDRAM state machine, and SDRAM interface logic fits into this device with the utilization results shown in Table 3.

Table 3: **CoolRunner-II Design Utilization**

Parameter	Used	Available	% Utilization
I/O Pins	86	100	86%
Macrocells	102	128	80%
Product Terms	180	448	40%
Registers	100	128	78%
Function Block Inputs	126	320	39%

System Interface

The SDRAM design described in this document includes a generic system interface. The system interface illustrates the basic control signals needed for the SDRAM logic block. These signals include a system clock, reset, 24-bit address bus, 16-bit data bus, and 4-bit command bus. Excluding the system clock, all signals are asynchronous and registered in the CPLD.

Modeling of the system interface in this design is done with testbench logic. The testbench is responsible for generating the address, data and command signals for any operation with the

Mobile SDRAM device. The testbench controls the initialization requirements, single read/write operations, and tests the power down modes of the SDRAM.

VHDL Code

XAPP394 -**ftp://ftp.xilinx.com/pub/applications/refdes/xapp394.zip**

Conclusion

CoolRunner-II CPLD devices are the perfect compliment to low power Mobile SDRAM memory. These two low power devices are targeted for handheld, mobile computing applications. CoolRunner-II CPLD devices feature low power consumption and create a seamless interface to Mobile SDRAM memory.

References

1. Micron Data Sheet. MT48V16M16LFFG 256Mb: x16 Mobile SDRAM. Micron Technology, Inc. 2002.
2. Micron Technical Note. TN-48-10. Mobile SDRAM Power-Saving Features. Micron Technology, Inc. 2002.
3. Samsung Data Sheet. K4S64163LF-RG/S 4Mx16 Mobile SDRAM. Rev 1.0. Samsung Electronics 2002.

Additional Information

CoolRunner-II Data Sheets, Application Notes, and White Papers

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/23/03	1.0	Initial Xilinx release.
12/01/03	1.1	Errata

XAPP386 (v1.0) December 12, 2002

CoolRunner-II Serial Peripheral Interface Master

Summary

This document details the VHDL implementation of a Serial Peripheral Interface (SPI) master in a Xilinx CoolRunner™-II CPLD. CoolRunner-II CPLDs are the lowest power CPLDs available, making this the perfect target device for an SPI Master. To obtain the VHDL code described in this document, go to section **VHDL Code Download and Disclaimer**, page 81 for instructions. This design fits XC2C256 CoolRunner-II or XCR3256XL CoolRunner XPLA3 CPLDs.

Introduction

The Serial Peripheral Interface (SPI) is a full-duplex, synchronous, serial data link that is standard across many microprocessors, microcontrollers, and peripherals. It enables communication between microprocessors and peripherals and/or inter-processor communication. The SPI system is flexible enough to interface directly with numerous commercially available peripherals.

A SPI Master design has been implemented in a CoolRunner-II CPLD. The CoolRunner-II SPI Master design can be used to provide a SPI controller to those microcontrollers or microprocessors that do not contain a SPI interface. A high-level block diagram is shown in Figure 1. The microcontroller (μC) interface chosen in this SPI Master implementation is based on the popular 8051 microcontroller bus cycles, but can easily be modified to other microcontroller interfaces. For more information on the 8051 microcontroller interface, please refer to **XAPP388, CoolRunner-II CPLD 8051 Microcontroller Interface**.

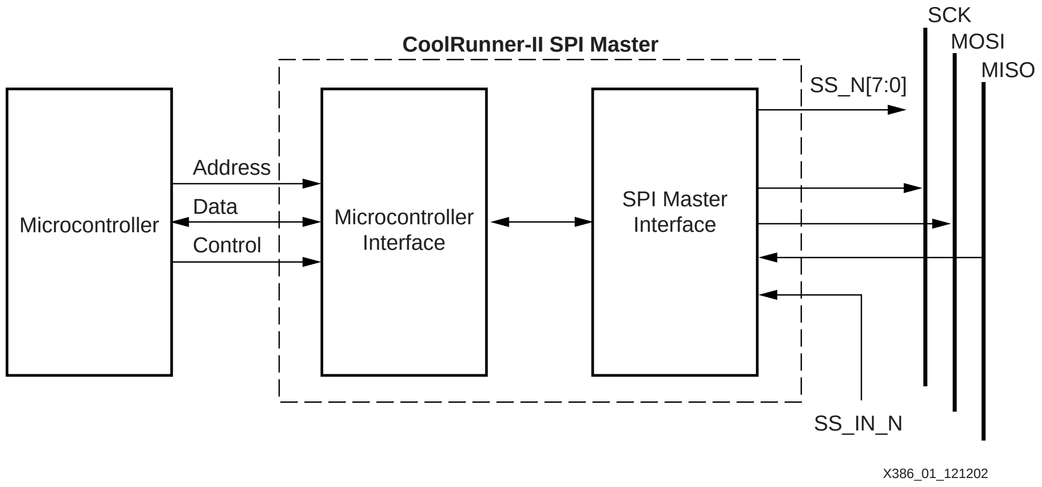


Figure 1: **CoolRunner-II SPI Master**

SPI Background

This section will describe the main protocol of the SPI bus. For more details and timing diagrams, please refer to the description of the SPI bus in the Motorola 68HC11 Reference Manual.

The SPI bus consists of four wires, Serial Clock (SCK), Master Out Slave In (MOSI), Master In Slave Out (MISO), and Slave Select (SS_N), which carry information between the devices connected to the bus.

The SCK control line is driven by the SPI Master and regulates the flow of data bits. The master may transmit data at a variety of baud rates; the SCK line transitions once for each bit in the transmitted data. The SPI specification allows a selection of clock polarity and a choice of two fundamentally different clocking protocols on an 8-bit oriented data transfer. The master selects one of four different bit rates for SCK. Data is shifted on one edge of SCK and is sampled on the opposite edge when the data is stable.

The MOSI data line carries the output data from the master which is shifted as the input data to the selected slave. The MISO data line carries the output data from the selected slave to the input of the master. There may be no more than one slave transmitting data during a particular transfer.

All SCK, MOSI, and MISO pins are tied together. A single SPI device is configured as a master; all other SPI devices on the SPI bus are configured as slaves. The single master drives data out its SCK and MOSI pins to the SCK and MOSI pins of the slaves. One selected slave device can drive data out its MISO pin to the MISO master pin.

The SS_N control line selects a particular slave via hardware control. This control line allows individual selection of a slave SPI device. Slave devices that are not selected do not interfere with SPI bus activities. Slave devices ignore the SCK signal and keeps the MISO output pin in a high impedance state unless the slave select pin is active low.

The SS_IN_N control line can be used as an input to the SPI Master indicating a multiple-master bus contention (SS_IN_N). If the SS_IN_N signal to the master is asserted, it indicates that some other device on the bus is attempting to be a master and address this device as a slave. Assertion of SS_IN_N automatically disables SPI output drivers in the master device if more than one device attempts to become master.

The clock phase and polarity can be modified for SPI data transfers. The clock polarity (CPOL) selects an active high or active low clock and has no significant effect on the transfer format. If CPOL = "0", then the idle state of SCK is low. If CPOL = "1", then the idle state of SCK is high. The clock phase (CPHA) can be modified to select one of two fundamentally different transfer formats. If CPHA = "0", data is valid on the first SCK edge (rising or falling) after SS_N has asserted. If CPHA = "1", data is valid on the second SCK edge (rising or falling) after SS_N has asserted. The clock phase and polarity should be identical for the master SPI device and the communicating slave device.

Figure 2 shows the timing diagram for a SPI data transfer when the clock phase, CPHA, is set to "0". The waveform is shown for both positive and negative clock polarities of SCK. The SCK signal remains inactive for the first half of the first SCK cycle (SCK Cycle 1). In this transfer format, the falling edge of SS_N indicates the beginning of a data transfer. Therefore, the SS_N line must be negated and reasserted between each successive serial byte. If the slave writes data to the SPI data register while SS_N is active low, a write-collision error results.

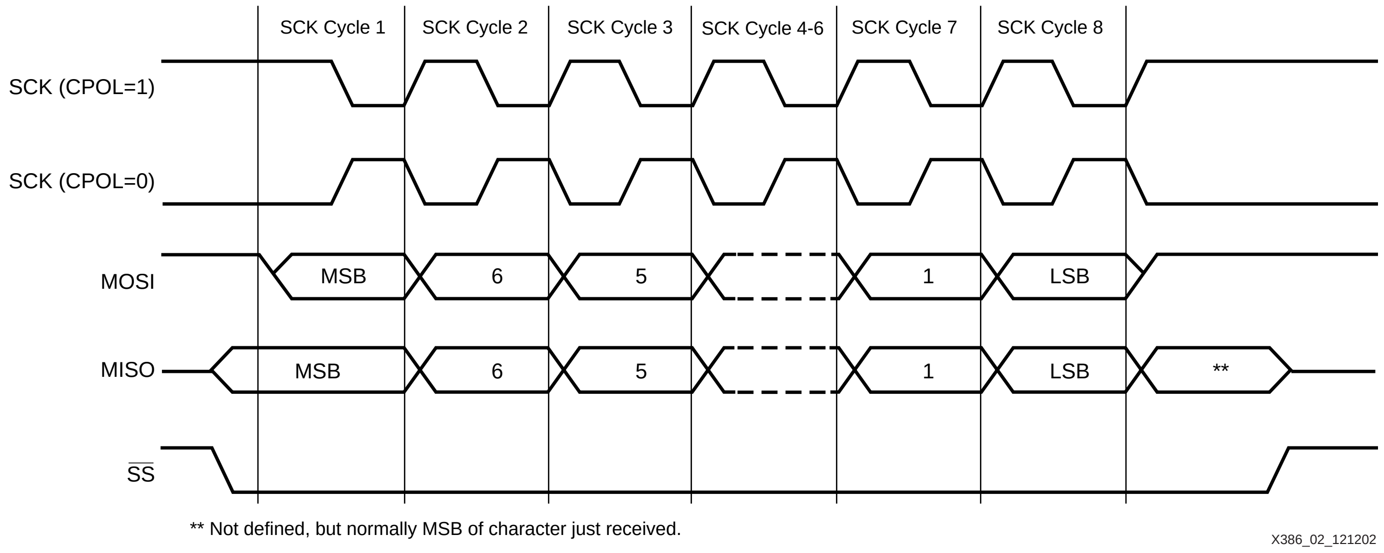


Figure 2: **Data Transfer on the SPI Bus with CPHA=0**

Figure 3 shows the timing diagram for a SPI transfer when the clock phase, CPHA, is set to "1". The waveform is shown for both positive and negative clock polarities of SCK. The first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The first edge of SCK indicates the start of the data transfer in this format. The SS_N line may remain active low between successive transfers. This format is useful in systems with a single master and single slave.

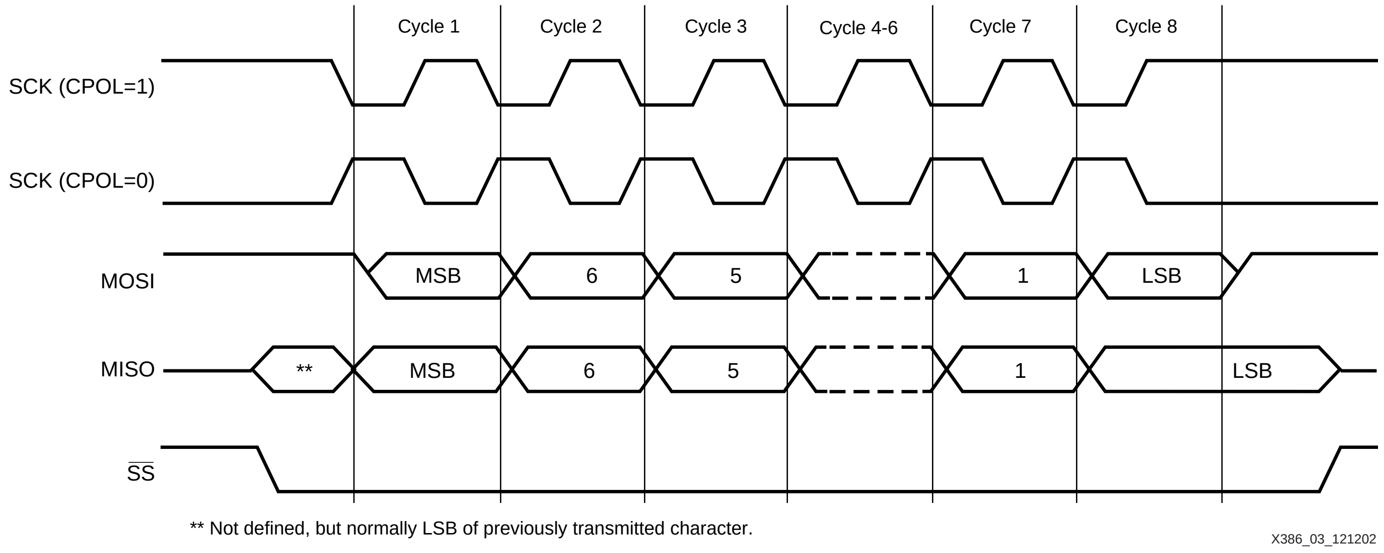


Figure 3: **Data Transfer on SPI Bus with CPHA=1**

When an SPI transfer occurs, an 8-bit data word is shifted out one interface pin while a different 8-bit data word is being shifted in on another interface pin. This can be viewed as an 8-bit shift register in the SPI Master device and another 8-bit shift register in a SPI slave device that are connected as a circular 16-bit shift register. When a transfer occurs, this 16-bit shift register is shifted 8 positions, thus exchanging the 8-bit data between the master and slave devices.

The SPI specification details the timing and wave forms for byte transfers on the SPI bus, but does not dictate the data protocol used in these transfers, i.e., the interface specification does not specify that the first byte contain an address or a read/write line. When communicating to devices over the SPI bus, it is important to review the specifications of these devices to determine the required communication protocol so that commands, addresses and the data direction (read/write) can be set correctly. The CoolRunner-II CPLD SPI Master implementation will not enforce a particular data protocol. It is expected that the μC will specify the data bytes to be transferred on the SPI bus in the correct order. All data received on the SPI bus will be stored in a receive register for the user logic to interpret. All data written to the transmit register will be transmitted on the SPI bus.

CoolRunner-II SPI Master Implementation

The CoolRunner-II CPLD implementation of an SPI Master supports the following features:

- Microcontroller interface
- Multi-master bus contention detection and interrupt
- Eight external slave selects
- Four transfer protocols available with selectable clock polarity and clock phase
- SPI transfer complete microcontroller interrupt
- Four different bit rates available for SCK

Signal Descriptions

The 36 I/O signals of the CoolRunner-II CPLD SPI Master are described in Table 1. Pin numbers have not been assigned to this design, this can be done to meet the system requirements of the designer.

Table 1: **CoolRunner-II SPI Master Signal Description**

Name	Direction	Description
MOSI	Output	**SPI Serial Data Output.** Serial data output from the SPI Master to a SPI slave.
MISO	Input	**SPI Serial Data Input.** Serial data input from a SPI slave to the SPI Master.
SS_IN_N	Input	**SPI Slave Select Input.** Active Low slave select input to the SPI Master. If this line is asserted, it indicates that another master on the bus was trying to select this SPI Master as a SPI slave and thus bus contention can occur. Assertion of this line causes the CoolRunner-II CPLD SPI Master to reset all communication and interrupt the μC.
SS_N[7:0]	Output	**SPI Slave Selects.** Active Low slave select signals to eight SPI slaves in the system.
SCK	Output	**SPI Serial Clock.** Clock output selectable as 1/2, 1/4, 1/8, or 1/16 of the system clock.
ADDR[15:8]	Input	**μC Address Bus.** High byte address bus.
ADDR_DATA[7:0]	Bidirectional	**μC Multiplexed Address/Data Bus.**
ALE_N	Input	**Address Latch Enable**. Active Low μC control signal indicating that the data present on the multiplexed address/data bus is a valid address.
PSEN_N	Input	**Program Store Enable.** Active Low μC control signal indicating that the current bus cycle is an access to the external program memory.
RD_N	Input	**Read Strobe.** Active Low μC control signal indicating that the current bus cycle is a read cycle.

Table 1: **CoolRunner-II SPI Master Signal Description**

WR_N	Input	**Write Strobe.** Active Low µC control signal indicating that the current bus cycle is a write cycle.
INT_N	Output	**Interrupt Request.** Active Low signal to generate an interrupt to the µC. This signal is asserted when interrupts are enabled and there is SPI bus contention as indicated by SS_IN_N or when the transmit register is empty (SPITR) during a transaction, or when the receive register is full and the transaction is complete.
XMIT_EMPTY	Output	**Transmit Register Empty.** Active High signal indicating that the transmit register (SPITR) is empty. This bit is used to signal the loading of data from the SPITR to the SPI transmit shift register indicating that the µC can load another byte of data into the SPITR. This signal could be connected to an µC interrupt or to an I/O port. This signal causes INT_N to assert during data transfers but does not cause an interrupt after the transfer is complete (i.e. START = 0). This signal is brought out of the CPLD as a separate I/O pin for systems that do not want to use interrupts.
RCV_FULL	Output	**Receive Register Full**. Active High signal indicating that the receive register (SPIRR) is full. This bit is used to signal the loading of data from the SPI receive shift register to the SPIRR. This signal could be connected to an µC interrupt or to an I/O port. This signal causes INT_N to assert only for the last word received from the transfer (i.e., START = 0). This signal is brought out of the CPLD as a separate I/O pin for systems that do not want to use interrupts.
CLK	Input	**Clock.** This clock is input from the system and is used to generate the SCK signal.
RESET	Input	**Reset.** Active High reset from the system. When asserted, all logic in the CoolRunner-II CPLD is reset.

Block Diagram

The block diagram of the CoolRunner-II CPLD SPI Master, shown in Figure 4 was broken into two major blocks, the µC interface and the SPI interface.

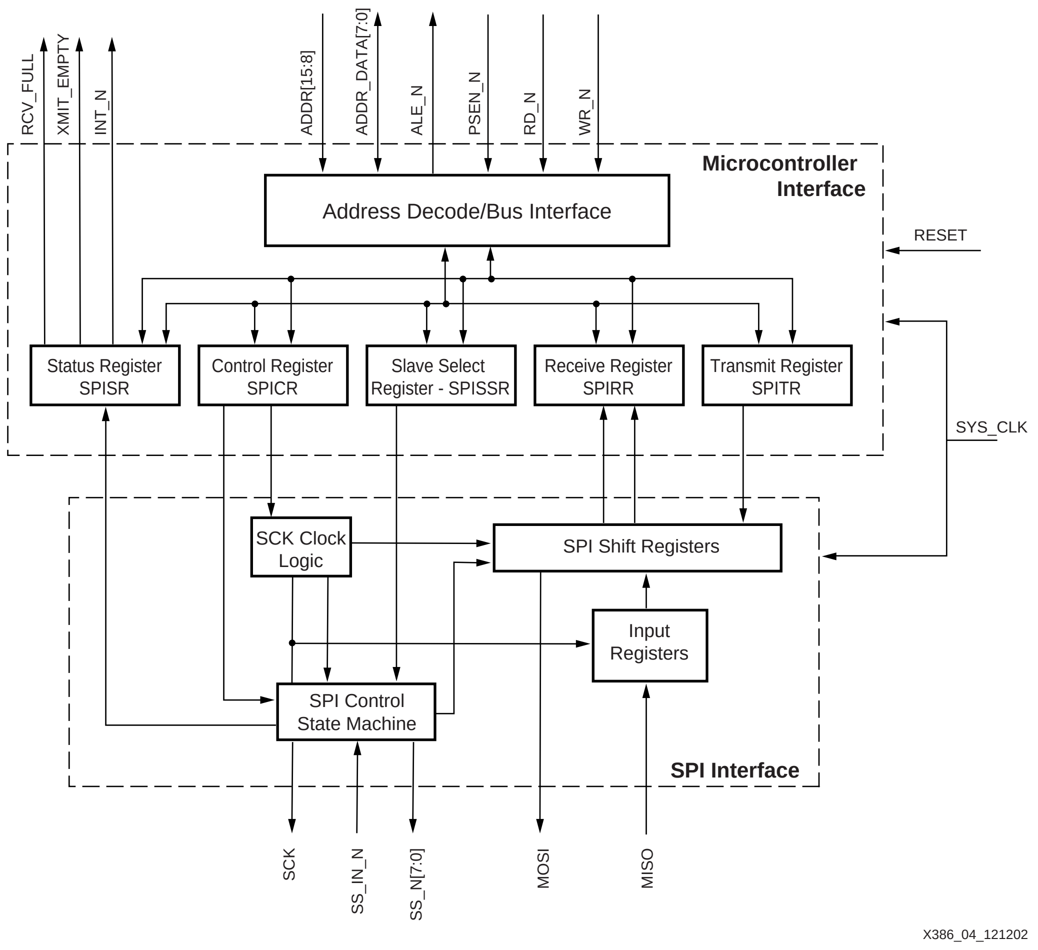


Figure 4: **SPI Master Block Diagram**

µC Interface Logic

The µC interface for the SPI Master design supports a byte-wide, multiplexed address/data bus protocol found in the popular 8051 µC. This protocol is the method in which the µC reads and writes the registers in the CoolRunner-II CPLD and is shown in Figure 5.

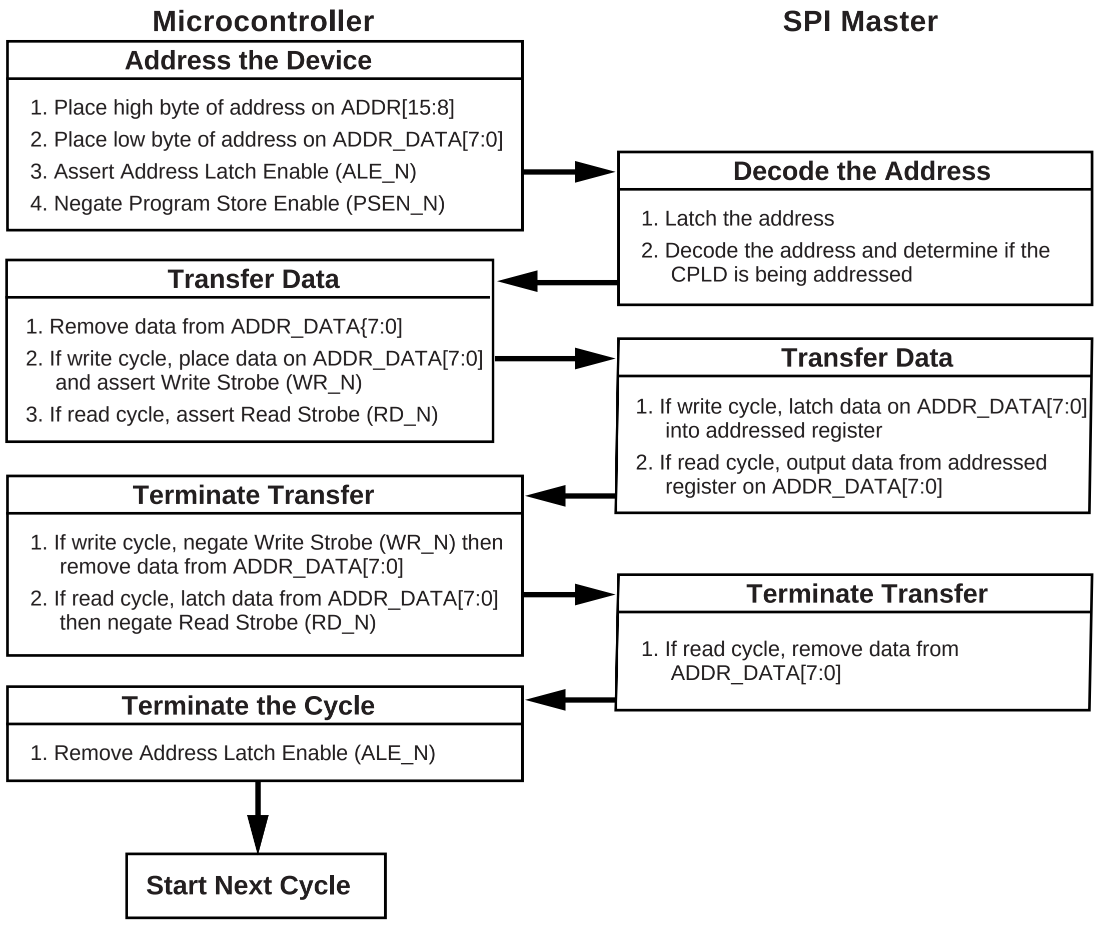

Microcontroller
SPI Master
Address the Device
1. Place high byte of address on ADDR[15:8]
2. Place low byte of address on ADDR_DATA[7:0]
3. Assert Address Latch Enable (ALE_N)
4. Negate Program Store Enable (PSEN_N)
Decode the Address
1. Latch the address
2. Decode the address and determine if the CPLD is being addressed
Transfer Data
1. Remove data from ADDR_DATA{7:0]
2. If write cycle, place data on ADDR_DATA[7:0] and assert Write Strobe (WR_N)
3. If read cycle, assert Read Strobe (RD_N)
Transfer Data
1. If write cycle, latch data on ADDR_DATA[7:0] into addressed register
2. If read cycle, output data from addressed register on ADDR_DATA[7:0]
Terminate Transfer
1. If write cycle, negate Write Strobe (WR_N) then remove data from ADDR_DATA[7:0]
2. If read cycle, latch data from ADDR_DATA[7:0] then negate Read Strobe (RD_N)
Terminate Transfer
1. If read cycle, remove data from ADDR_DATA[7:0]
Terminate the Cycle
1. Remove Address Latch Enable (ALE_N)
Start Next Cycle
X386_05_121202

Figure 5: **µC Read/Write Protocol**

Note that the code to interface to the 8051 µC is a separate VHDL module which interfaces to the SPI interface logic via a set of registers. Therefore, this code can easily be replaced with the µC interface of your choice.

Address Decode/Bus Interface Logic

The µC bus protocol is implemented in the CoolRunner-II SPI Master in the state machine shown in Figure 6.

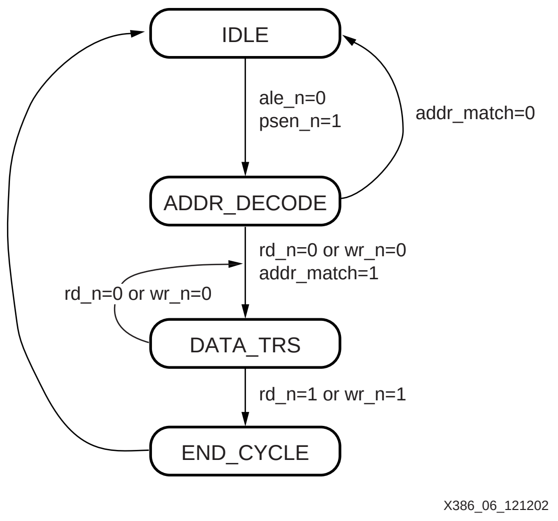


Figure 6: **µC Bus Interface State Machine**

In the first cycle, the µC places the address on the address bus and asserts address latch enable (ALE_N). ALE_N indicates that the data on the multiplexed address/data bus is a valid address and that the address on ADDR[15:0] is also valid.

Upon the assertion of ALE_N, the CoolRunner-II SPI Master transitions to the ADDR_DECODE state to decode the address and determine if it is the device being addressed. The enables for the internal registers are set in this state. ALE_N is also used to register the lower address bits from the multiplexed ADDR_DATA bus.

If this is a write cycle, the µC removes the address from the multiplexed address/data bus and places the data to be written onto these signals. The write strobe (WR_N) is then asserted. If this a read cycle, the µC 3-states the multiplexed address/data bus and asserts the read strobe (RD_N) indicating that the CoolRunner-II SPI Master can place data from the addressed register on the data bus.

If the CoolRunner-II SPI Master is being addressed and either RD_N or WR_N are asserted, the CoolRunner-II SPI Master progresses to the DATA_TRS state. If this is a read cycle, the requested data is placed on the bus and if this is a write cycle, the data from the data bus is latched in the addressed register.

The µC latches the data present on the bus if this is a read cycle and then negates the read strobe (RD_N). If this is a write cycle, the µC removes data from the bus and then negates the write strobe (WR_N). The negation of either RD_N or WR_N causes the CoolRunner-II SPI Master to progress to the END_CYCLE state. The CoolRunner-II CPLD will 3-state the multiplexed address/data bus in this state, removing the data if it is a read cycle.

At this point, the µC ends the cycle by negating address latch enable (ALE_N), which causes the CoolRunner-II CPLD to return to the IDLE state.

CoolRunner-II SPI Master Registers

The base address used for address decoding is set in the VHDL code via the constant BASE_ADDRESS. The lower four address bits determine which register inside the CPLD is being accessed. Each register address is set by a constant in the µC interface VHDL code that can easily be modified to meet the addressing scheme of the system.

The registers supported in the CoolRunner-II SPI Master are described in the Table 2. The µC interface logic of the CoolRunner-II SPI Master handles the reading and writing of these registers by the µC and supplies and/or retrieves these bits to/from the SPI interface logic.

Table 2: **SPI Master Registers**

Address	Register	VHDL Constant	Description
BASE + $80\h	SPISR	SPISR_ADDR	SPI Status Register
BASE + $84\h	SPICR	SPICR_ADDR	SPI Control Register
BASE + $88\h	SPISSR	SPISSR_ADDR	SPI Slave Select Register
BASE + $8A\h	SPITR	SPITR_ADDR	SPI Transmit Data Register
BASE + $8E\h	SPIRR	SPIRR_ADDR	SPI Receive Data Register

SPI Status Register (SPISR)

This register contains the status of the SPI controller. This status register is read-only with the exception of certain bits which are software clearable as described in Table 3.

Table 3: **Status Register Bits**

Bit Location	Name	µC Access	Description
7	Done	Read	**Done Bit.** While one byte of data is being transferred, this bit is cleared. It is set during the eighth SCK clock period of a byte transfer. • "1" transfer is complete • "0" transfer in progress
6	SPIERR	Read Software Clearable	**SPI Error Bit.** When the SS_IN_N input pin is asserted, this bit is set indicating an SPI error condition. The SPI interface resets and 3-states all signals on the SPI bus. An interrupt will be asserted to the µC when this bit is set if interrupts are enabled. This bit must be cleared by the µC writing a "0" to this bit in the interrupt service routine. See Figure 11 for details on how to handle an SPI error.
5	BB	Read	**Bus Busy Bit.** This bit indicates the status of the SPI bus. This bit is set when a slave select line (SS_N) is asserted and cleared when a slave select line (SS_N) is negated. • "1" indicates the bus is busy • "0" indicates the bus is idle The µC should examine this bit to insure that the bus is not busy before configuring the SPI interface for an SPI transaction.
4	INT_N	Read Software Clearable	**Interrupt Bit.** This bit is asserted (active low) when an interrupt is pending which causes a processor interrupt request if INTEN is set. An interrupt is asserted if any of the following conditions occur: • The transmit register (SPITR) is empty and there are more data words to transmit (START = 1) • The receive register (SPIRR) is full and there are no more data words to transmit (START = 0) • An SPI error has occurred This bit is negated whenever the µC writes data to the SPITR, reads data from the SPIRR, or resets the SPIERR bit.

Table 3: **Status Register Bits *(Continued)***

Bit Location	Name	µC Access	Description
3	XMIT_EMPTY	Read	**Transmit Register Empty.** This bit is set when the transmit register (SPITR) is empty. It is cleared when the µC writes data into this register. An interrupt will be asserted to the µC when this bit is set if interrupts are enabled and there are more data words to transmit (START = 1). Note that this bit is also an output pin.
2	RCV_FULL	Read	Receive Register Full. This bit is set whenever the receive register (SPIRR) is full. It is cleared when the µC reads from this register. An interrupt will be asserted to the µC when this bit is set if interrupts are enabled and there are no more data words to transmit (START =0). Note that this bit is also an output pin.
1-0	Unused		**Unused Bits.** These bits will read as "0" when the status register is read.

SPI Control Register (SPICR)

This register contains the bits to configure the SPI Master. (Table 4)

Table 4: **Control Register Bits**

Bit Location	Name	µC Access	Description
7	SPIEN	Read/Write	**SPI Master Enable.** This bit must be set before any other SPICR bits have any effect • “1” enables the SPI Master • “0” resets and disables the SPI Master
6	INTEN	Read/Write	**Interrupt Enable.** • “1” enables interrupts. An interrupt occurs if the INT_N bit in the status register is also set • “0” disables interrupts but does not clear the cause of any currently pending interrupts
5	START	Read/Write	**SPI Transfer Start.** When the µC changes this bit from “0” to “1”, the SPI Master begins to transfer the data in the SPI Transfer data register (SPITR) on the SPI bus if XMIT_EMPTY is negated. All data received from the SPI bus is captured in the SPI Receive data register (SPIRR). As long as this bit stays asserted, SPI transfers will occur. After the µC has written the last data word to be transferred in SPITR and XMIT_EMPTY asserts, the µC must negate this bit to indicate that this is the last word in the SPI transfer.
4-3	CLKDIV	Read/Write	**Clock Divider Bits.** These bits determine the frequency of SCK by selecting a divide by 4, 8,16, or 32 of the system clock. • "00" - SCK is 1/4 of the system clock • "01" - SCK is 1/8 of the system clock • "10" - SCK is 1/16 of the system clock • "11" - SCK is 1/32 of the system clock

Table 4: **Control Register Bits *(Continued)***

Bit Location	Name	μC Access	Description
2	CPHA[1]	Read/Write	Clock Phase Bit. This bit determines the clock phase of SCK in relationship to the serial data. • "0" - data is valid on first SCK edge (rising or falling) after slave select has asserted • "1" - data is valid on second SCK edge (rising or falling) after slave select has asserted
1	CPOL[1]	Read/Write	Clock Polarity Bit. This bit determines the polarity of SCK. • "0" - SCK is low when idle • "1" - SCK is high when idle
0	RCV_CPOL[1]	Read/Write	**Receive Clock Polarity.** This bit determines whether the MISO data is sampled on the rising or falling edge of the outgoing SCK. • "0" - MISO data is sampled on the falling edge of SCK • "1" - MISO data is sampled on the rising edge of SCK Note that in most cases, RCV_CPOL = "1" when CPHA is the same value as CPOL and is "0" otherwise. However, this bit should be set according to the slave that is being accessed.

Notes:

1. CPHA, CPOL, RCV_CPOL should be set to match the protocol expected by the SPI slave that is the target of the SPI bus cycle.

SPI Slave Select Register (SPISSR)

This register contains bits which indicate which slave select line should be asserted. A "1" in a bit position indicates that the corresponding bit in the slave select output bus is asserted according to the SPI timing specifications. A "0" in a bit position indicates that the corresponding bit in the slave select output bus stays negated. This allows the μC to specify which slave select line is asserted during a SPI data transfer. Note that only one slave select line can be asserted during a SPI data transfer. This must be adhered to by the μC, the hardware implementation of this specification does not enforce this requirement, in other words, if the μC sets multiple bits in this register, multiple slave select lines will be asserted for the SPI transfer. (Table 5)

Table 5: **SPI Slave Select Register**

Bit Location	Name	μC Access	Description
7 - 0	SS_N7 - SS_N0	Read/Write	SPI Slave Selects

SPI Transfer Data Register (SPITR)

This register contains data to be transmitted on the SPI bus on the MOSI pin (Table 6). Data written into this register is output on the SPI bus once the START bit in the control register (SPICR) has been asserted. As long as the START bit in the SPICR is asserted, additional bytes of data in this register will continue to be transmitted on the SPI bus.

Once this data has been loaded into the SPI transmit shift register, XMIT_EMPTY asserts and the μC can place the next data byte for transmission on the SPI bus into this register. Writing data into this register resets the XMIT_EMPTY flag. Note that XMIT_EMPTY must be negated and START must be asserted before the SPI state machine will begin transmission of data on the SPI bus.

Table 6: **SPI Transmit Data Register**

Bit Location	Name	µC Access	Description
7 - 0	D7 - D0	Read/Write	SPI Transmit Data

SPI Receive Data Register (SPIRR)

This register contains the data received from the SPI bus on the MISO pin. When a byte of data has been received from the SPI bus and transferred to the SPIRR, the RCV_FULL flag asserts. The µC then reads data from the SPIRR which resets the RCV_FULL flag. Because data is loaded from the SPI receive shift register to the SPIRR, the µC has an entire 8-bit SPI transfer to read the data from the SPIRR. (Table 7)

Table 7: **SPI Receive Data Register**

Bit Location	Name	µC Access	Description
7 - 0	D7 - D0	Read/Write	SPI Receive Data

SPI Interface Logic

The SPI bus interface logic consists of several different processes as seen in Figure 4. Control bits from the µC interface registers determine the behavior of these processes.

SPI Control State Machine

This process generates the slave select signals and controls the shift and load operations of the SPI transmit shift register. It also monitors the SPI bus and determines when a byte transfer is complete. This process also generates clock mask signals that control when the clock is output to the SPI bus. If the START signal stays asserted after a byte has been transferred, the state machine will continue to output the next byte and the SCK signal continues to transition. Note that if CPHA = 0, the slave select signal will negate and then re-assert between consecutive byte transfers as stated in the SPI specification. If the START signal is negated after a byte has been transferred, then the state machine first insures that the current transfer has been completed and the SCK output is placed in its inactive state as determined by CPOL.

The slave select is then negated after the required hold time. The SPI control state machine is shown in Figure 7.

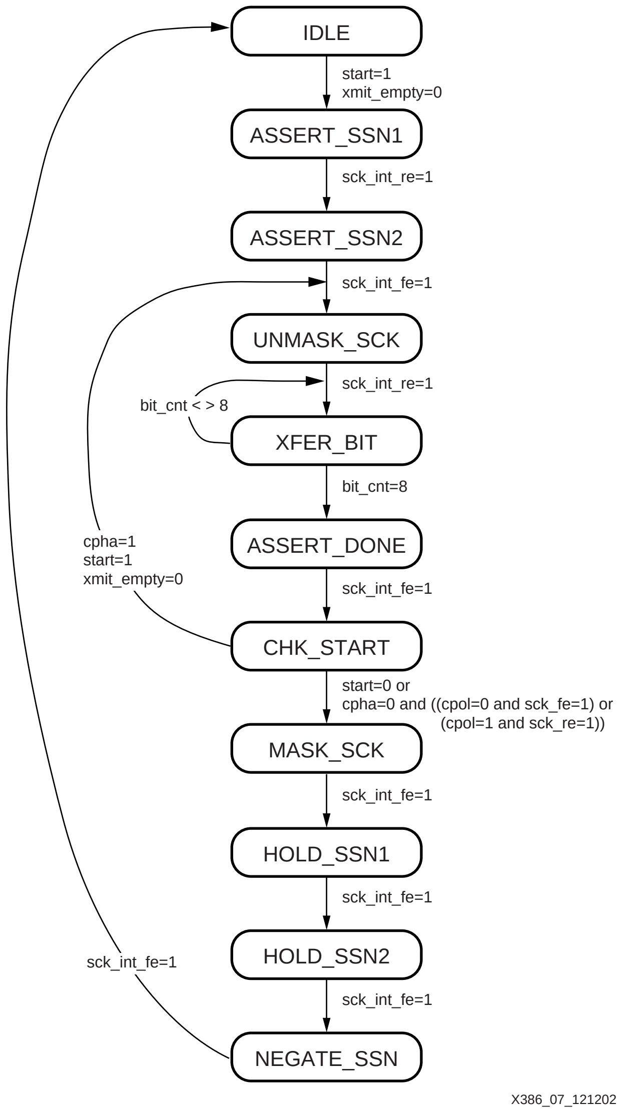


Figure 7: **SPI Control State Machine**

The SPI Control state machine remains in the IDLE state until the START bit in the SPI Control register is asserted and the XMIT_EMPTY signal is negated. At this point, the state machine moves to the ASSERT_SSN1 state to assert the internal SS_N signal. This signal is then masked with the SPI Slave Select Register (SPISSR) to generate the slave select signals output to the system. After a rising edge of the internal SCK (SCK_INT_RE=1), the state machine transitions to the ASSERT_SSN2 state, keeping SS_N asserted until the falling edge of the internal SCK. This state machine will insure that SS_N will assert ~1 SCK period before the first SCK edge, meeting the SS_N setup time requirement of most SPI slave devices. This timing parameter should be verified by the designer for the target system as the SS_N setup time requirement may vary between different SPI slaves.

After SS_N has been asserted for both the ASSERT_SSN1 and ASSERT_SSN2 states, the state machine transitions to the UNMASK_SCK state. The first edge of the phase 1 (CPHA=1) version of SCK occurs before the first data is output, therefore, this state unmasks the phase1 version of SCK. The SPI transmit shift register is loaded with data from the SPITR in this state. The SPI transmit shift register is clocked by the rising edge of the internal SCK signal. The state machine waits for a rising edge of the internal SCK signal (SCK_INT_RE) to leave this state to insure that the SPI transmit shift register has been loaded.

The state machine then moves to the XFER_BIT state. The first edge of the phase 0 (CPHA=0) version of SCK occurs after data has been output, therefore, this state unmasks the phase 0 version of SCK. This state shifts data from the SPI transmit shift register and the state machine remains in this state until the byte transfer is complete. Once the byte transfer has completed, the state machine transitions to the ASSERT_DONE state where the DONE signal in the SPI Status Register (SPISR) is asserted. The state machine will not transition to the CHK_START state until the next falling edge of the internal SCK to synchronize this state machine to the internal SCK.

The SPI specification requires that SS_N be negated and re-asserted between consecutive byte transfers when CPHA=0. If CPHA=1, SS_N can remain asserted during consecutive byte transfers. Therefore, if START is still asserted in the CHK_START state and CPHA=1 and XMIT_EMPTY is negated, the state machine transitions back to the UNMASK_SCK state and continues SPI transfers. If START is negated or CPHA=0, the state machine transitions to the MASK_SCK state which masks both the phase 0 and the phase1 versions of SCK. Note, however, that the state machine waits for either the rising edge (if CPOL=1) or the falling edge (if CPOL=0) of the external SCK before transitioning to this state to insure that the transmission has been completed before masking the external clock.

At the next falling edge of the internal SCK (SCK_INT_FE), the state machine transitions to the HOLD_SSN1 state. SS_N must stay asserted for some time period after the last SCK edge.To insure that the SS_N hold time is at least 2 SCK periods, the state machine transitions to HOLD_SSN2 after the next falling edge of the internal SCK (SCK_INT_FE) and remains in this state until another falling edge of the internal SCK has occurred. This will meet the SS_N hold time requirement of most SPI slave devices. This timing parameter should be verified by the designer for the target system as the SS_N hold time requirement may vary between different SPI slaves.

At this point, the state machine transitions to the NEGATE_SSN state and remains in this state until the next falling edge of the internal SCK. This insures that the pulse width of SS_N between SPI transfers is at least one SCK period. This will meet the SS_N pulse width requirement of most SPI slave devices. This timing parameter should be verified by the designer for the target system as the SS_N pulse width requirement may vary between different SPI slaves.

The state machine then transitions to the IDLE state. If START is asserted and XMIT_EMPTY is negated, the SPI transfer and state machine operation will repeat.

Note that if no further SPI transfers are required, the µC must negate the START signal after writing the last data word of the transmission to the SPITR as shown in Figure 11.

Also note that at any time, the assertion of SS_IN_N will cause the SPI Control state machine to return to the IDLE state and the MOSI, SS_N, and SCK outputs will be 3-stated. The SPI Master will remain in this state until the SS_IN_N signal is negated and the START signal is asserted. When a SPI error occurs, the system must be examined to determine the cause of the error. The µC can reset the bit in the SPI status register (SPISR) and then continue to read the SPI status register (SPISR) to determine if the system error has been corrected. Once the error has been fixed at the system level, the SPI interface should be reset to guarantee correct operation as shown in Figure 11. The µC can reset the SPI Master by negating the SPIEN bit in the SPI control register (SPICR).

Transmit Empty and Receive Full Flags

The transmit empty flag (XMIT_EMPTY) is set whenever data is loaded from the SPITR to the SPI transmit shift register. This signal is clocked from the internal SCK and is reset whenever the µC writes data into the SPITR or when the system reset signal is asserted.

The receive full flag (RCV_FULL) is set whenever data is loaded from the SPI receive shift register to the SPIRR. This signal is clocked from the system clock and is reset whenever the µC reads data from the SPIRR.

SCK Clock Logic

This process generates the SCK output based on the CLKDIV, CPHA, and CPOL settings in the SPI control register. The clock frequency of the SCK signal is determined by dividing down the input clock based on the entries in the control register. The signal, SCK_INT is the internal SCK used to clock serial data out of the device and is continually generated. The SPI Control state machine is synchronized to this internal signal. The signal SCK_1 represents SCK when CPHA = 1 and the signal SCK_0 represents SCK when CPHA = 0. The SPI control state machine generates the masks for these clocks (CLK0_MASK, CLK1_MASK) so that the output SCK has the correct phase relationship with the data and is held in its inactive state when there is no data to be transferred. A representation of the logic required to generate the SCK signal output to the SPI bus is shown in Figure 8.

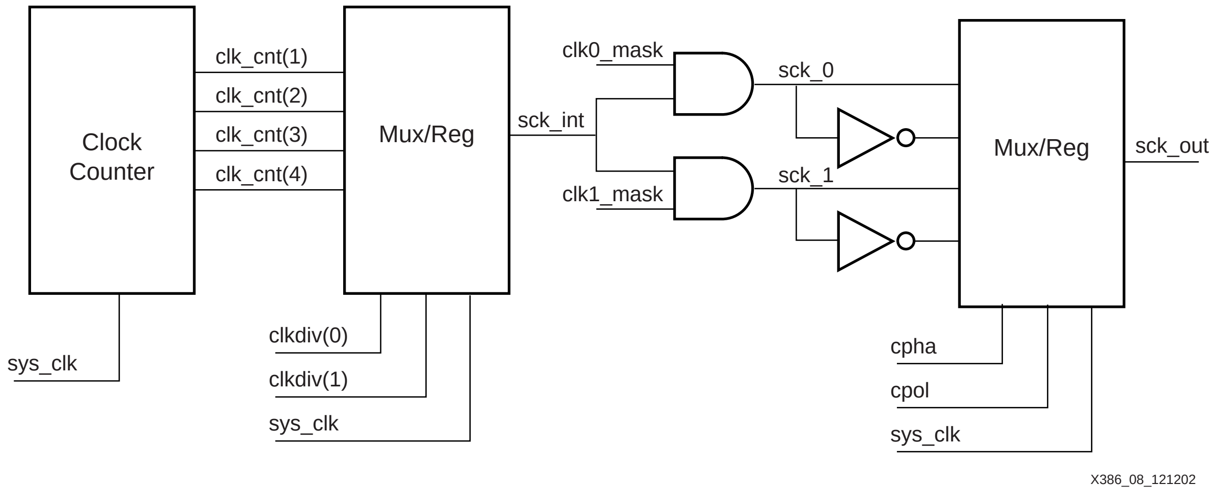


Figure 8: **SCK Clock Generation Logic**

SPI Shift Registers

SPI Transmit Shift Register

The SPI transmit shift register is an 8-bit loadable shift register containing SPI data. This shift register is loaded from the SPI Transmit Register (SPITR) via a load signal generated by the SPI Control state machine and is clocked by the rising edge of SCK_INT. The data shifting out is the MOSI data. Note that in Figure 8, SCK_OUT is one SYS_CLK delay from SCK_INT. Therefore, it is necessary to delay the data being shifted out from the SPI transmit shift register by one SYS_CLK as well so that the relationship between MOSI and SCK_OUT is maintained.

This is accomplished by a single register clocked from SYS_CLK which simply clocks the data output from the shift register as shown in Figure 9..

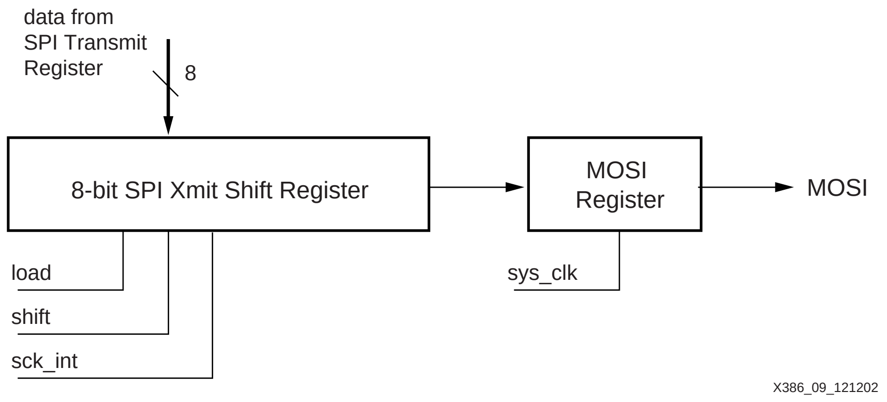


Figure 9: **SPI Transmit Shift Register**

SPI Receive Shift Register

A separate shift register is used to receive the MISO data since the clock phase and polarity of the SCK output can vary based on each transaction. The SPI receive shift register is clocked on the rising edge of the external SCK. The RCV_CPOL bit set in the control register allows the μC to specify which edge of the external SCK incoming MISO data is sampled on. This allows for flexibility in dealing with all types of different SPI slave devices as some SPI slaves will clock data out on the rising edge of SCK while others will clock data out on the falling edge of SCK. If a slave clocks data out on the falling edge of SCK, then RCV_CPOL should be set to "1" so that the CoolRunner-II SPI Master will clock data in on the rising edge of SCK. If a slave clocks data out on the rising edge of SCK, then RCV_CPOL should be set to "0" so that the CoolRunner-II SPI Master clocks data in on the falling edge of SCK. This eliminates any setup and hold timing issues. If slaves behave according to the SPI specification for CPHA and CPOL, RCV_CPOL will equal "1" whenever CPHA and CPOL are equal and "0" otherwise.

In the actual implementation, two input registers are used to sample MISO, one clocked on the rising edge of SCK, the other clocked on the falling edge of SCK. The outputs of these two registers are then multiplexed with the RCV_CPOL bit used as the select line. The output of this multiplexor then becomes the input to the SPI receive shift register which is clocked on the rising edge of the external SCK. Using this implementation method eliminates multiplexing clocks inside the CPLD. Multiplexing clocks within the CPLD places a delay between SCK and the actual data input to the shift register. This delay on SCK could impose an additional data hold time requirement on the slave device which is undesirable. Therefore, it was decided to multiplex the data instead, thus eliminating the need to specify a holdtime requirement on the slave device.

A counter, clocked off the rising edge of the external SCK, is used to count the bits shifted into the SPI receive shift register and to indicate when a byte transfer is complete. When the byte transfer is complete, the data in the SPI receive shift register is loaded into the SPI Receive

Register for use by the µC. The SPI receive shift register, MISO input registers, and receive bit counter are shown in Figure 10.

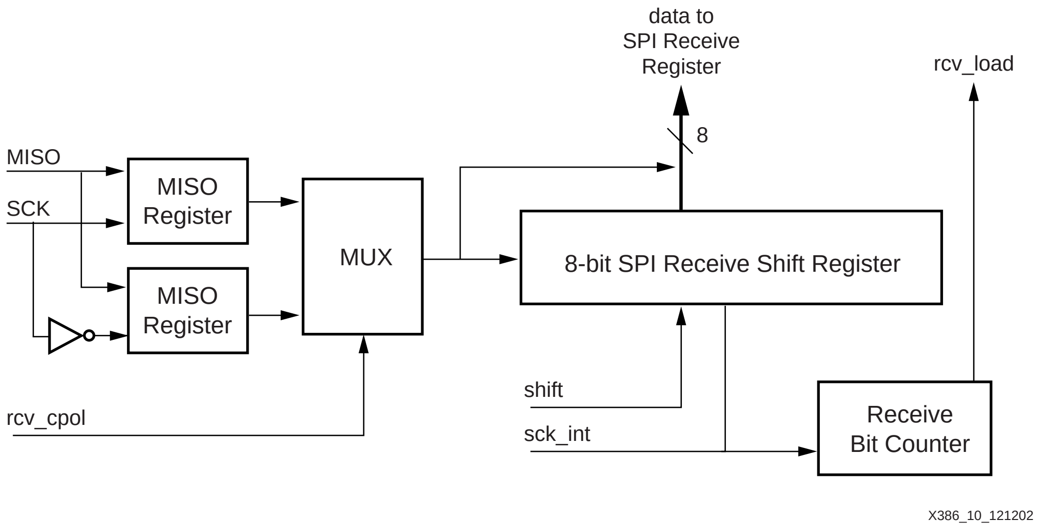


Figure 10: **SPI Receive Shift Register and MISO Input Data Registers**

Operational Flow Diagrams

The flow of the interface between the µC and the CoolRunner-II SPI Master is detailed in the following flow charts. These flow charts are meant to be a guide for utilizing the CoolRunner-II SPI Master in a µC system.

SPI Transaction Flow Chart

The flow chart for configuring and performing a SPI transaction for the CoolRunner-II SPI Master is shown in Figure 11. Since SPI is a full duplex communication protocol, data is received while data is transmitted.

Note that if an SPI error occurs indicating that another master has taken control over the bus, the system operation should be verified. The flow chart shows continually polling SPIERR in the status register to determine when the SPI error has been corrected. Since an SPI error also asserts an interrupt (if interrupts have been enabled), an alternative to polling the SPI status register (SPISR) is to service the interrupt to determine if the SPI error has been corrected. Note that either of these flows may not be the operational flow required by a system when an

SPI error has occurred; the designer should determine the correct operations to execute when an SPI error occurs based on the system requirements.

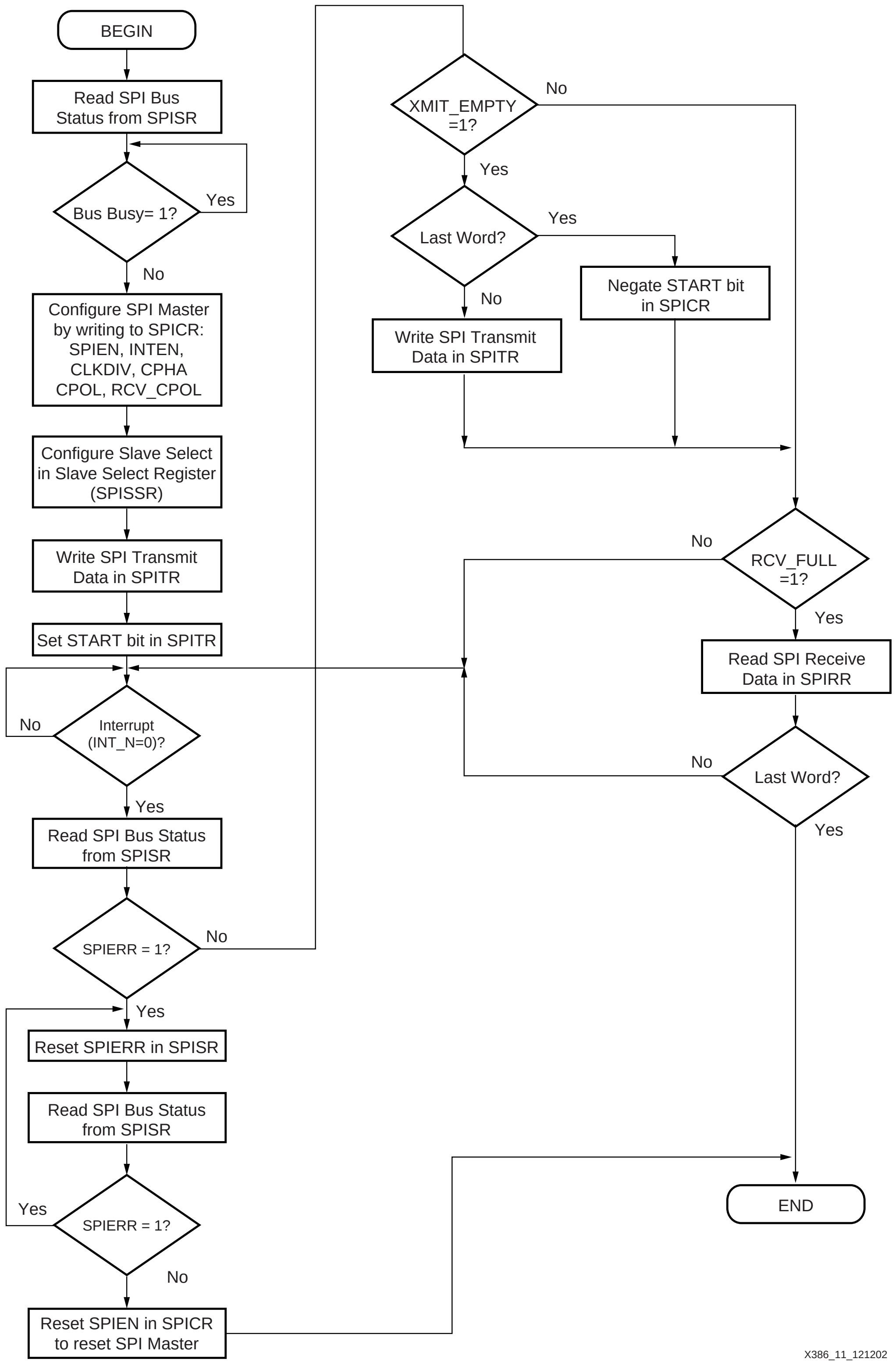

BEGIN
Read SPI Bus Status from SPISR
Bus Busy= 1?
Yes
No
Configure SPI Master by writing to SPICR: SPIEN, INTEN, CLKDIV, CPHA CPOL, RCV_CPOL
Configure Slave Select in Slave Select Register (SPISSR)
Write SPI Transmit Data in SPITR
Set START bit in SPITR
No
Interrupt (INT_N=0)?
Yes
Read SPI Bus Status from SPISR
SPIERR = 1?
No
Yes
Reset SPIERR in SPISR
Read SPI Bus Status from SPISR
Yes
SPIERR = 1?
No
Reset SPIEN in SPICR to reset SPI Master
XMIT_EMPTY =1?
No
Yes
Last Word?
Yes
Negate START bit in SPICR
No
Write SPI Transmit Data in SPITR
No
RCV_FULL =1?
Yes
Read SPI Receive Data in SPIRR
No
Last Word?
Yes
END
X386_11_121202

Figure 11: **CoolRunner-II SPI Master Transaction Flow Chart**

VHDL Testbench and Functional Simulation

A VHDL testbench has been developed that verifies the CoolRunner-II SPI Master implementation through all the possible combinations of CLKDIV, CPHA, CPOL, and RCV_CPOL. This testbench contains a process that emulates the bus cycles of the 8051 µC. Constants are provided at the top of the testbench file to set up the base address of the SPI Master device and all of the registers contained within the device. These constants should be modified to match the addressing scheme of the designer's system.

The VHDL testbench also provides a model of a simple SPI slave. The slave first transmits the hex value "CE" and then simply transmits the value that was received. The clock phase and polarity of the slave are dynamically set in the test bench based on the CPHA and CPOL values currently being tested.

To test the SPI Master reaction to an SPI error, the test bench contains the constant SS_IN_ASSERT_TIME which specifies the delay from the beginning of the simulation until SS_IN_N is asserted and also specifies the amount of time that SS_IN_N is asserted. Setting this constant to "0" disables assertion of SS_IN_N.

The test bench contains a signal, ERROR, which indicates that the data received did not match the expected data. The simulation has run correctly if ERROR stays "0" throughout all SPI transfers. Note, however, that ERROR may assert some time during the simulation if the testbench is set up to test SS_IN_N assertion. This is due to the fact that the data received may be out of alignment with the expected data value.

The ModelSim command file, *func_sim.do,* can be used to open the correct waveform window and run the simulation.

CoolRunner-II CPLD Implementation

The CoolRunner-II SPI Master design has been targeted to a CoolRunner-II 256 macrocell device. The speed grade chosen is dependent on the system clock frequencies and should be analyzed by the designer to determine which speed grade is required.

The CoolRunner-II SPI Master utilizes 128 of the 256 macrocells available in the device, leaving 50% of the device resources for user logic.

The top level file for the SPI Master has been entered as a heirarchical VHDL file showing the connection between the *uc_interface* block and the *spi_interface* block. The *spi_interface* block is also a hierarchical VHDL fileshowing the inter connectivity between the *spi_control_sm* block, the *sck_logic* block, the *spi_rcv_shift_reg* block and the *spi_xmit_shift_reg* block. The structural VHDL models are found in the files *spi_master.vhd* and *spi_interface.vhd.*

Post-fit Timing Simulation

The Xilinx Project Navigator software package outputs a timing VHDL model of the fitted design. This post-fit VHDL was simulated with the original VHDL test benches to insure design functionality using ModelTech Xilinx Edition (MXE). Please note that all verification of this design has been done through simulations.

The user of this design is strongly encouraged to thoroughly inspect the timing report for this design to insure that the design meets the timing specification of the system. The user is also strongly encouraged to perform post-fit timing simulations as well. The ModelSim command file, *post_sim.do*, can be used to open the correct waveform window and run the simulation.

VHDL Code Download and Disclaimer

All VHDL source code, VHDL testbenches, and software files associated with this design are available. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any

XAPP386 - ftp://ftp.xilinx.com/pub/applications/refdes/xapp386.zip

Conclusion

This document has detailed the design of a Serial Peripheral Interface Master design for a CoolRunner-II CPLD. Though the design has been extensively verified in simulations, Xilinx assumes no responsibility for the accuracy or the functionality of this design.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/12/02	1.0	Initial Xilinx release.

XAPP354 (v1.1) September 30, 2002

Using Xilinx CPLDs to Interface to a NAND Flash Memory Device

Summary

This application note describes the use of a Xilinx CoolRunner™ CPLD to implement a NAND Flash memory interface. CoolRunner CPLDs are the lowest power CPLD available and the ideal target device for memory interface applications. The code for this design may be downloaded from the Xilinx web site, refer to section **HDL Code**, page 96. This design fits XCR3032XL CooRunner or XC2C32 CoolRunner-II CPLDs.

Introduction

The flash memory market has been rapidly growing over the past several years due in part to increasing demands of portable and embedded devices. This market is driven by embedded code storage and bulk data storage applications. Flash technology has been optimized to meet the needs of these two target applications.

NAND Flash is a sequential access device appropriate for mass storage applications, while NOR Flash is a random access device appropriate for code storage applications. NAND technology organizes cells serially to achieve higher densities. This reduces the number of contacts needed in the memory array. The trade-off between the two technologies is NAND Flash data must be accessed sequentially compared with NOR Flash which offers fast random access.

NAND Flash memory offers low cost per bit, high performance, and the highest density non-volatile memory available. NAND Flash is ideal for applications ranging from MP3 players and digital cameras to applications requiring mass storage of data, especially when the data is packetized, or sequentially arranged.

This application note describes the design of a basic NAND Flash interface. The NAND devices used for testing this interface include the AMD UltraNAND™ Flash and the Samsung NAND Flash memory.

The AMD UltraNAND flash memory (AM30LV0064D) is a 64 Mbit storage device. This memory device utilizes a multiplexed command/data/address bus as well as other control signals for read, erase and program commands. This memory device has an initial page read access time of 7 µs, with subsequent byte access times of less than 50 ns per byte.

The Samsung K9F4008W0A is a 512K x 8-bit NAND Flash memory device. The command, data, and address are multiplexed through an 8-bit I/O port. This memory device supports a 32-byte frame read, with random access times of 15 µs and sequential access times of 120 ns.

NAND Interface

The NAND interface described here is implemented in a CoolRunner CPLD. The NAND interface design can interact with both AMD and Samsung NAND memory devices.

Figure 1 shows the overall system diagram for a single AMD UltraNAND Flash or Samsung memory device. The CoolRunner CPLD reads the 4 least significant bits in the system address in order to decode the flash interface commands. The interface signals to the Flash device are asserted by writing to a specific port of the CPLD.

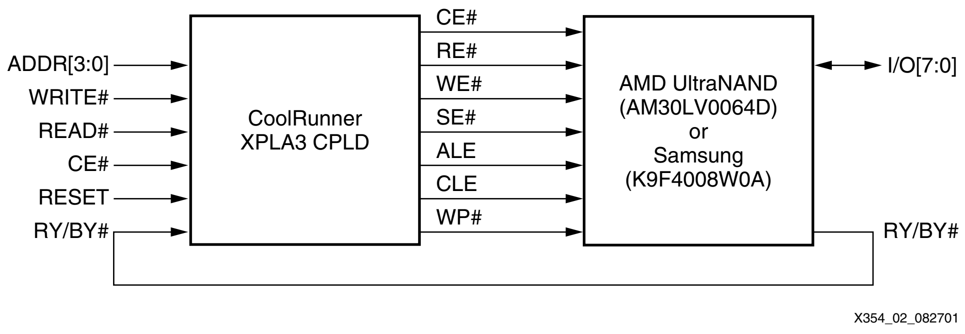


Figure 1: **System Block Diagram**

The NAND Flash interface signals and functionality is shown in Table 1.

Table 1: **UltraNAND Pin Descriptions**

Pin Name	Function
I/O[7:0]	I/O pins used to send commands, address, and data to the device, and receive data during read operations.
CLE	Command Latch Enable. The CLE input controls writing to the command register. When CLE is high, the command is loaded on the rising edge of WE#.
ALE	Address Latch Enable. The ALE input controls writing to the address register. When ALE is high, the address is loaded on the rising edge of WE#. ALE must remain high during the entire address sequence.
CE#	Chip Enable. The CE# input controls the active vs. standby mode of the device. During a command or address load sequence, CE# must be low prior to the falling edge of WE#.
RE#	Read Enable. The RE# input controls the data and status output on the I/O lines. The data output is triggered on the falling edge of RE#.
WE#	Write Enable. The WE# input controls the data and command on the I/O lines during a write sequence. The I/O lines are latched on the rising edge of the WE# signal.
WP#	Write Protect. The WP# input provides protection when programming or erasing the device. The internal voltage regulator is reset when WP# is low, preventing any program or erase operations.
SE#	Spare Area Enable. The SE# input controls access to the 16 bytes of spare area on each page. When SE# is not asserted (high), the spare area for the selected page is not enabled. When SE# is asserted (low), access to the spare area is enabled.
RY/BY#	Ready/Busy Output. The RY/BY# output indicates the operation status of the device. When RY/BY# is high, the device is ready for the next operation. When RY/BY# is low, an internal program, erase, or random read operation is in progress.

AMD UltraNAND Memory Device

The AM30LV0064D is organized as 8 kB (+ 256 byte spare area) blocks (1,024 blocks total). Each block has 16 pages of 512 bytes (+ 16 bytes spare area) or 16,384 pages total. Figure 2 is a block diagram of the AMD UltraNAND device.

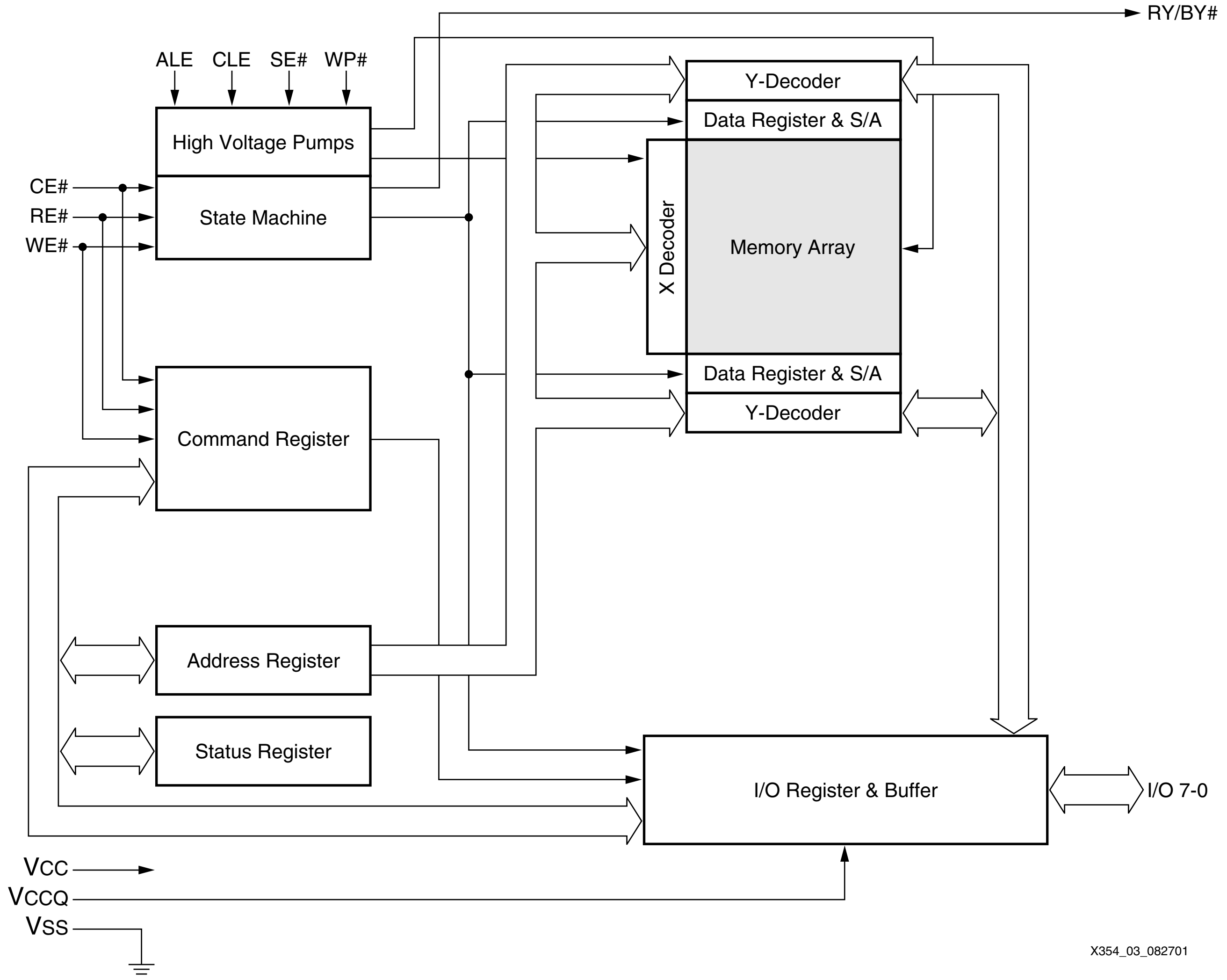


Figure 2: **AMD UltraNAND Block Diagram**

Table 2 describes the AMD UltraNAND command set and functionality. The command register does not occupy any addressable memory location. This register holds the command, along with any address and data information needed to execute the command. Programming data into the Flash array is a two step process. The data to be programmed is loaded into the data registers using the "Input Data" command. After the data is loaded, the "Page Program" command is performed to write data from the data registers to the Flash array.

Table 2: **AMD UltraNAND Command Set**

Operation	Cycle (Hex)	Functionality
Read Data	00h or 01h	Reads out flash array starting with First Half Page (00h) or reads out data starting with the Second Half Page (01h).
Gapless Read	02h	Allows reading from multiple pages with only one 7 µs latency occurring on the first page transfer
Read Spare Area	50h	Only reads data from the 16 byte spare area in each page (address locations 512 through 527).
Read ID	90h	Read the manufacturer and device ID.
Read Status	70h	Checks the device status to determine if the device is ready, in the write protect mode, erase suspended, or if the previous program/erase operation completed without error.
Input Data	80h	First command that allows the device to be programmed. This command loads the data registers from the I/O lines to program the device.
Page Program	10h	Issued after the "Input Data" operation has loaded the proper data. The command transfers information from the data registers to the Flash array in 200 µs or less, and the Flash device will appear busy during the data transfer operation.
Block Erase	60h & D0h	In the first command (60h), two address cycles are used to input the address of the block to erase. Once the second command (D0h) is issued, the Flash device will begin the "Block Erase" operation.
Erase Suspend	B0h	Only valid during a "Block Erase" command sequence. On the rising edge of WE#, the erase operation will be suspended.
Erase Resume	D0h	Only valid during an "Erase Suspend" command sequence. On the rising edge of WE#, the Flash device will resume the "Block Erase" operation.
Reset	FFh	Used to initialize the Flash device.

Samsung NAND Memory Device

The K9F4008W0A is a 512K x 8-bit NAND Flash memory. A Program operation programs a 32-byte frame in typically 500 µs and an Erase operation erase a 4 kB block in typically 6 ms. Data in a frame can be read out at a burst cycle rate of 120 ns/byte. The array organization of the Samsung device is such that 32 bytes are equal to one accessible frame. A row consists of four frames (or 128 bytes) and one block consists of 32 rows (or 4 kB). The total size of the device is 128 blocks (or 4 MB). Data is programmed via the Frame Register which holds 32 bytes of data. Figure 3 is a block diagram of the Samsung K9F4008W0A.

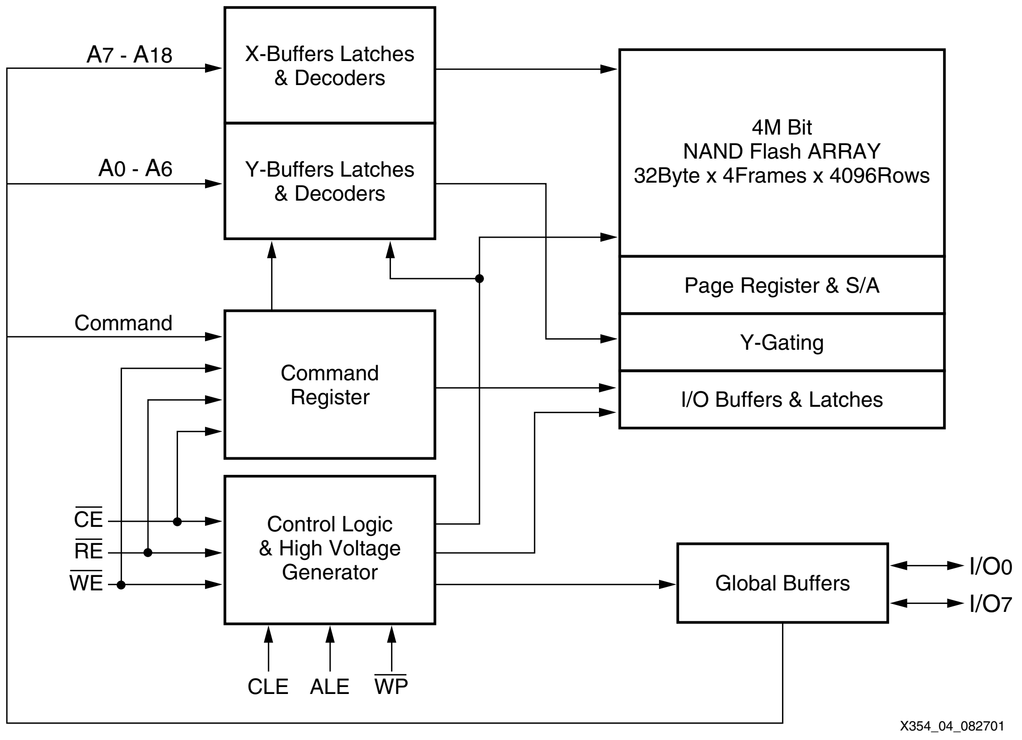

A7 - A18
X-Buffers Latches & Decoders
A0 - A6
Y-Buffers Latches & Decoders
4M Bit NAND Flash ARRAY 32Byte x 4Frames x 4096Rows
Page Register & S/A
Y-Gating
I/O Buffers & Latches
Command
Command Register
CE
RE
WE
Control Logic & High Voltage Generator
Global Buffers
I/O0
I/O7
CLE
ALE
WP
X354_04_082701

Figure 3: **Samsung Block Diagram**

All address and command instructions are multiplexed through the 8 I/O lines. Similar to the AMD UltraNAND, data is latched on the rising edge of WE# when CE# is low. The address is latched in when ALE = ’1’ and CLE = ’0’ and the command is latched when ALE = ’0’ and CLE = ’1’. Table 3 describes the Samsung NAND Flash memory command set.

Table 3: **Samsung Command Set**

Command	Cycle Data	Description
Read	00h	Device default mode. After the frame address is changed, 32 bytes of data are transferred to the data registers in less than 15 µs. Each data byte in the frame can be read on the high to low transition of the RE# signal.
Reset	FFh	Reset operation can abort a read, program, or erase operation. The device enters the Read mode after a Reset command.
Frame Program	80h & 10h	Frame Program consists of loading the 32 byte Frame Register and a nonvolatile programming period when the data is programmed into the appropriate cells.

Table 3: **Samsung Command Set**

Command	Cycle Data	Description
Block Erase	60h & D0h	The Erase operation is done 4K bytes (or 1 block) at a time. Requires a 2 cycle address load to specify block address (A_{18} to A_{12}).
Status Read	70h	The device contains a Status Register which may be read to determine if a program or erase operation is complete and successful. See the Samsung data sheet (refer to **References**, page 97) for a complete definition of each bit in the Status Register.
Read ID	90h	The device contains product identification, which can be read during a Read ID command. Two read cycles are required to read the manufacturer code and device code.

CPLD Design

For more information on the ABEL implementation of the CoolRunner CPLD design, refer to AMD Application Note # 22363 (see **References**, page 97). The design described here and available on the web is implemented in both VHDL and Verilog (see "HDL Code" on page 96 for more information).

The CPLD design decodes system address commands to interface with the AMD UltraNAND and Samsung Flash memory devices. The CPLD is responsible for the following functions:

- Decode read or write from address bus
- Interpret system address bus commands
- Assert interface signals to UltraNAND Flash device
- Monitor RY/BY# output from Flash memory device

The CPLD is configured to decode address 00h through 0Fh from a base address. The CPLD logic outputs are determined by the system writing to each port address. The port addresses for the CPLD are shown in Table 4 with a functional description of each.

Table 4: **CPLD Port Address Definition**

Port	Operation	Function
0	Read Data/ID/Status or Write Address/Data	Read information from previous command loaded. Write address (ALE high) or data (ALE low).
1	Command	All commands are written through this port with ALE low.
2	Set ALE	Set ALE (high) to allow addresses to be written.
3	Clear ALE	Clear ALE (low) to allow commands or data to be written.
4	Set SE#	Set SE# (low) to allow access to the spare area on each page.
5	Clear SE#	Clear SE# (high) to prevent access to the spare area on each page.
6	Set WP#	Set WP# (low) to prevent program/erase cycles.
7	Clear WP#	Clear WP# (high) to allow program/erase cycles.
8	Set CE#	Set CE# (low) to enable the UltraNAND device.
9	Clear CE#	Clear CE# (high) to disable the UltraNAND device.
A	N/A	Not used in this design.

Table 4: **CPLD Port Address Definition**

Port	Operation	Function
B	N/A	Not used in this design.
C	N/A	Not used in this design.
D	N/A	Not used in this design.
E	N/A	Not used in this design.
F	RY/BY# Status	Read the state of all RY/BY# pins through this port.

Table 4 is described in more detail in the AMD Application Note #22363 for the design implementation in ABEL. Refer to **References**, page 97 for more information.

Figure 4 is a block diagram of the VHDL/Verilog implementation of the NAND interface. All port signals (shown in Table 4) are decoded from the system address when CE# is asserted.

The ALE_SIG process asserts the ALE signal on a write to PORT2 and clears ALE on a write to PORT3. The SEN_SIG process asserts the SE# signal upon a write to PORT4 and clears SE# upon a write to PORT5. The OUTCE_SIG process asserts the OUTCE# signal upon a write to PORT8 and clears OUTCE# upon a write to PORT9. The WPN_SIG process asserts the WP# signal upon a write to PORT6 and clears WP# upon a write to PORT7. The READY_SIG process assigns the RY/BY# signal from the Flash device to the ready output signal. Otherwise, the ready signal is 3-stated.

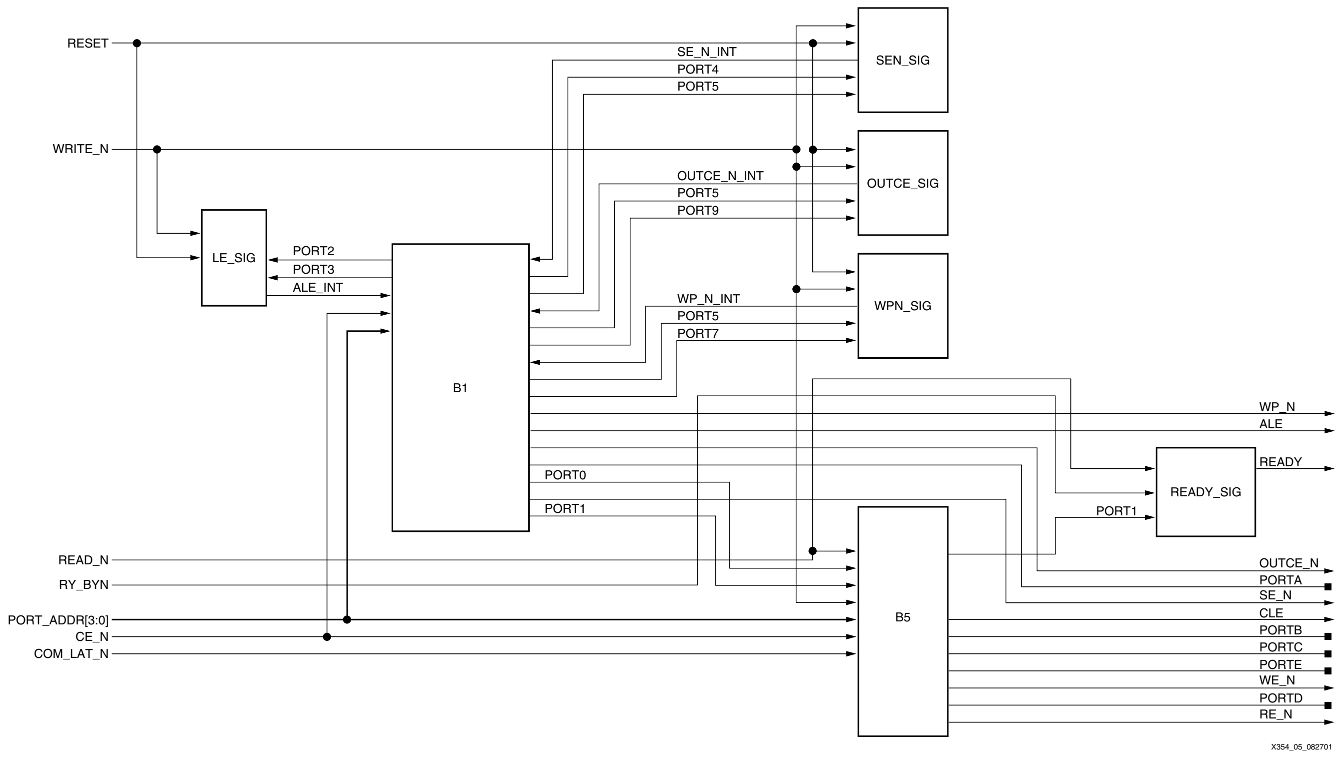


Figure 4: **CPLD Block Diagram**

The CLE signal is asserted on any access to PORT1. The RE# signal is asserted to the Flash device when a read is performed on PORT0. The WE# signal is asserted to the Flash device when a write is performed on PORT0 or PORT1.

CPLD Implementation

The NAND flash interface was implemented in VHDL and Verilog as described above; and in ABEL as described by AMD. Xilinx Project Navigator was used for design compilation, fitting, and simulation in a CoolRunner XPLA3 CPLD. Xilinx Project Navigator, which includes the ModelTech simulation tool, is available free-of-charge from the Xilinx website at www.xilinx.com/products/software/webpowered.htm. The design was targeted for a 3.3V, 32 macrocell CoolRunner XPLA3 CPLD (XCR3032XL-VQ44). The design utilization is shown in Table 5. This utilization was achieved using certain fitting parameters, actual results may vary.

Table 5: **NAND Interface XPLA3 32-Macrocell Utilization**

Resource	Available	Used	Utilization
Function Blocks	2	2	100%
Macrocells	32	10	31.25%
Product Terms	96	47	48.96%
I/O Pins	32	21	65.63%

Design Verification

Verification of the NAND interface was performed using the Xilinx WebPACK Project Navigator VHDL output timing model. The timing model was imported and compiled by Model Technology ModelSim. A test bench was utilized to instantiate the memory device (AMD UltraNAND or Samsung Flash) and the CPLD interface design as shown in Figure 5. The AMD UltraNAND and Samsung devices were instantiated using Denali Memory Maker.

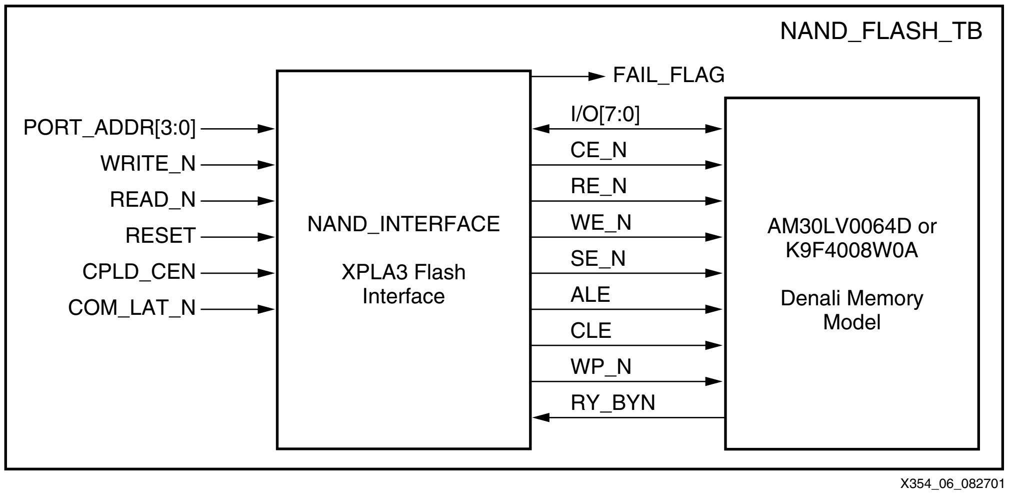


Figure 5: **Test Bench Diagram**

Figure 5 illustrates the operational flow used to test the Denali memory model. Figure 6 describes the test flow for the AMD UltraNAND device. The test flow for the Samsung Flash memory model was modified slightly to match Samsung command cycles.

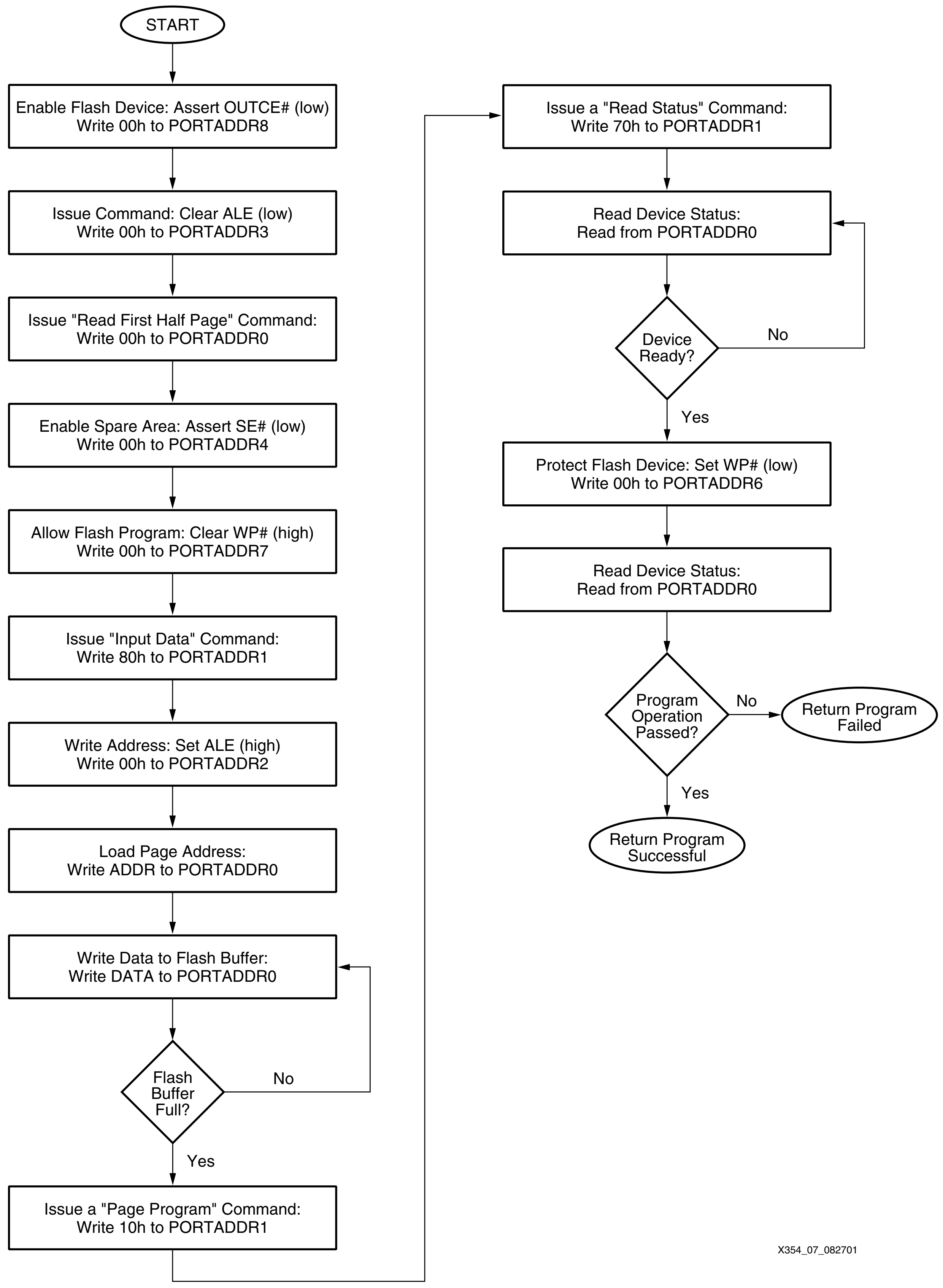
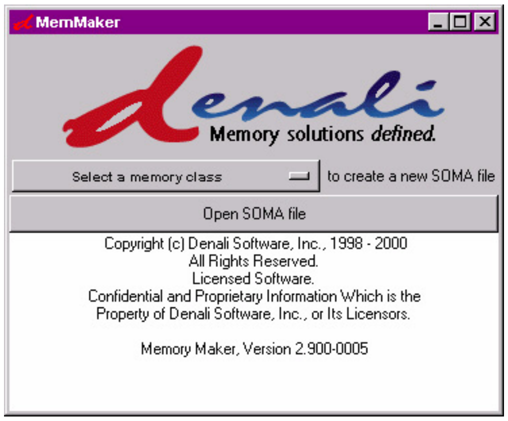


Figure 6: **Memory Operational Test Flow**

Denali Memory Maker

The Denali Memory Maker tool is used to simulate the AMD UltraNAND and Samsung devices. The Denali tool allows a VHDL or Verilog model to be instantiated in a test bench environment. In this simulation, Model Technology ModelSim is the target simulator. For a given memory device, a SOMA file (Specification Of Memory Architecture) represents unique timing, features, and functionality. The SOMA file is then imported to the Denali MemMaker tool. The SOMA file can be edited to meet the design signal names and timing requirements. Figure 7 illustrates how to bring a SOMA file into the MemMaker tool.

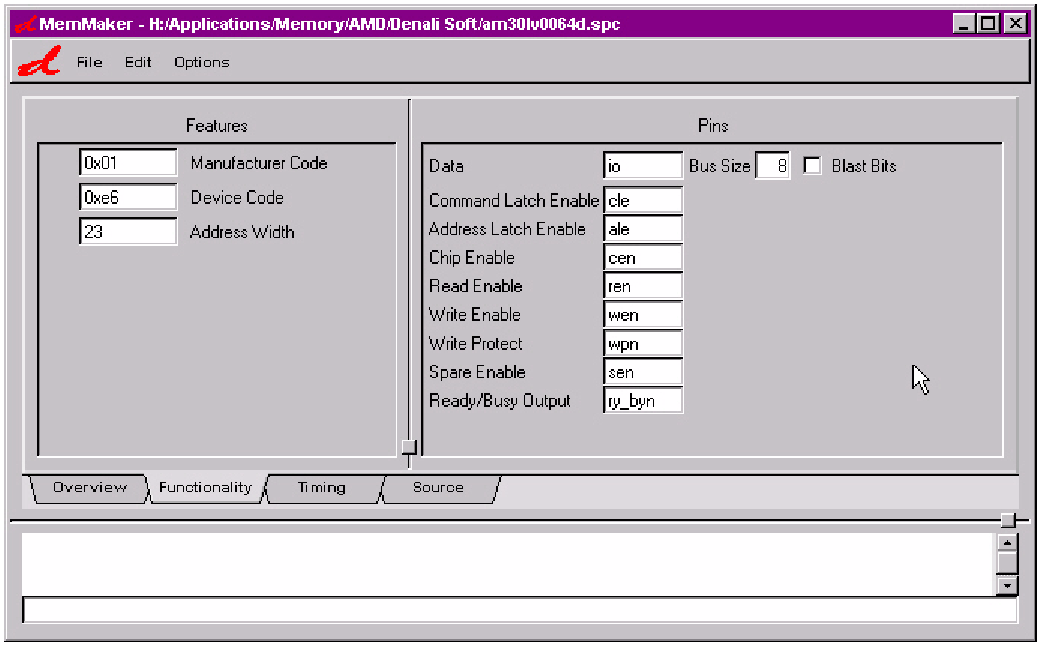


Copyright (c) Denali Software, Inc., 1998 - 2000
All Rights Reserved.
Licensed Software.
Confidential and Proprietary Information Which is the
Property of Denali Software, Inc., or Its Licensors.

Memory Maker, Version 2.900-0005

Figure 7: **Opening a SOMA File in MemMaker**

Figure 8 illustrates how to change any signal name requirements in the MemMaker tool.



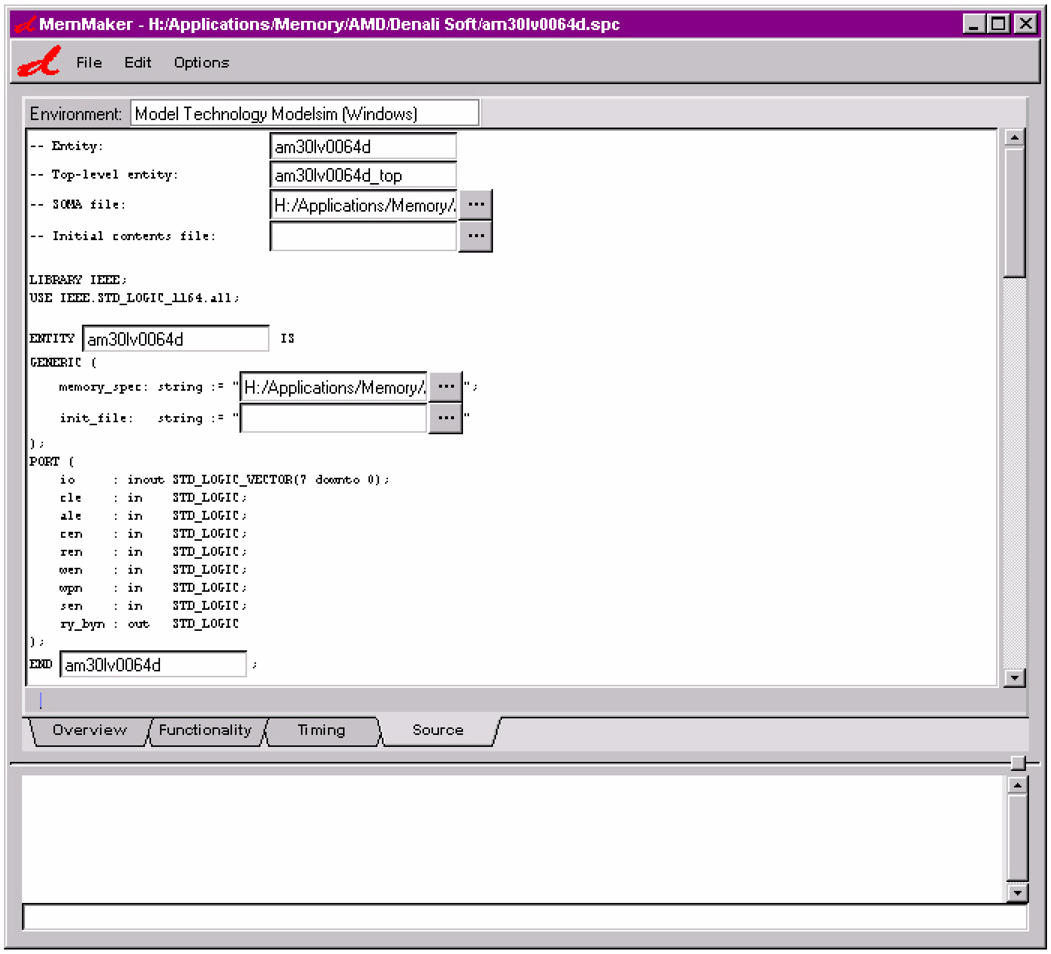
0x01	Manufacturer Code
0xe6	Device Code
23	Address Width

Data	io
Bus Size	8
Command Latch Enable	cle
Address Latch Enable	ale
Chip Enable	cen
Read Enable	ren
Write Enable	wen
Write Protect	wpn
Spare Enable	sen
Ready/Busy Output	ry_byn

Figure 8: **MemMaker Functional View**

Once all signal name and timing requirements have been specified, the VHDL or Verilog source code can be generated. To generate VHDL source code, select Options | Simulation

Environment | VHDL | Model Technology ModelSim (Windows). The VHDL code can then be saved for use in a test environment as shown in Figure 9.


MemMaker - H:/Applications/Memory/AMD/Denali Soft/am30lv0064d.spc
File
Edit
Options
Overview
Functionality
Timing
Source

Environment: Model Technology Modelsim (Windows)

```
-- Entity:                  am30lv0064d
-- Top-level entity:        am30lv0064d_top
-- SOMA file:               H:/Applications/Memory/
-- Initial contents file:

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;

ENTITY am30lv0064d IS
GENERIC (
    memory_spec: string := "H:/Applications/Memory/";
    init_file:   string := ""
);
PORT (
    io     : inout STD_LOGIC_VECTOR(7 downto 0);
    cle    : in    STD_LOGIC;
    ale    : in    STD_LOGIC;
    cen    : in    STD_LOGIC;
    ren    : in    STD_LOGIC;
    wen    : in    STD_LOGIC;
    wpn    : in    STD_LOGIC;
    sen    : in    STD_LOGIC;
    ry_byn : out   STD_LOGIC
);
END am30lv0064d ;
```

Figure 9: **MemMaker Source Code View**

ModelSim Implementation

Note:

Please refer to XAPP338: Using Xilinx WebPack and ModelTech ModelSim Xilinx Edition as a guide to using ModelSim with Project Navigator. The ModelSim Quick Start demo provides a good first step for getting familiar with ModelSim.

Model Technology ModelSim was the target simulator in this design. The test bench created for the CPLD design generates the necessary system address cycles. A separate test bench environment was used to test the AMD UltraNAND device and the Samsung Flash memory. This is due to the data buffer size during a program cycle and differences in command codes. The Denali MemMaker model is loaded in ModelSim as illustrated by the ModelSim script.

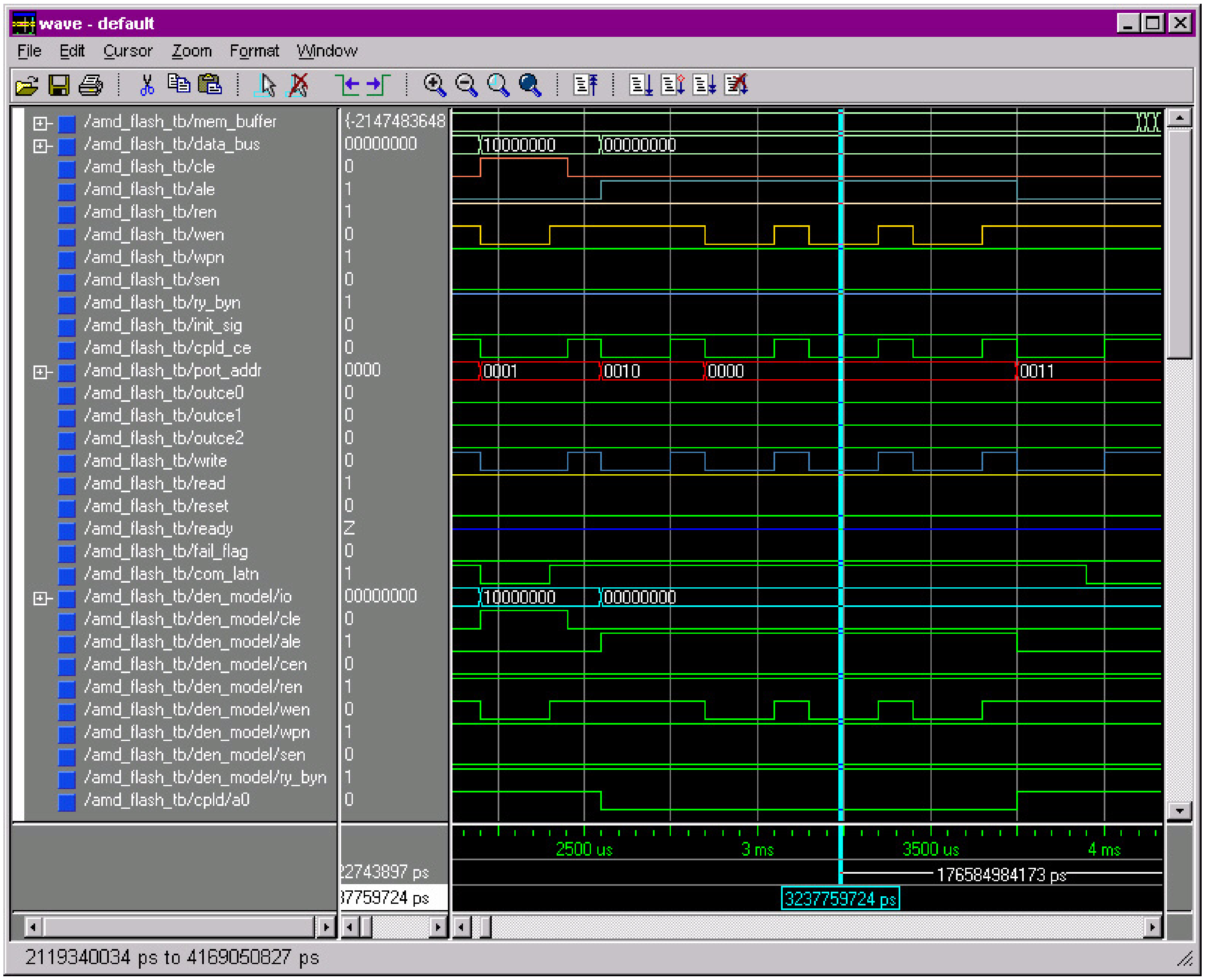
```
vcom -reportprogress 300 -work work{../amd_flash_tb.vhd}
# Model Technology ModelSim XE vcom 5.3d Compiler 2000.03 Mar 30 2000
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Loading package numeric_std
# -- Loading package pkg_convert
# -- Compiling entity amd_flash_tb
# -- Compiling architecture behavior of amd_flash_tb
# -- Loading entity am30lv0064d
# -- Loading package pxa_pkg
# -- Loading entity nand_int
vsim work.amd_flash_tb
```

```
# vsim work.amd_flash_tb
# Loading C:/Modeltech_xe/win32xoem/../std.standard
# Loading C:/Modeltech_xe/win32xoem/../ieee.std_logic_1164(body)
# Loading C:/Modeltech_xe/win32xoem/../ieee.numeric_std(body)
# Loading work.pkg_convert(body)
# Loading work.pxa_pkg
# Loading work.amd_flash_tb(behavior)
# Loading work.am30lv0064d(behavior)
# Loading C:\Denali\denali.dll
# *Denali* History enabled for Denali Memory Modeler.
# *Denali*     Debug information will also be saved.
# *Denali* Trace function enabled for Denali Memory Modeler.
# *Denali* Denali Memory Model Version 2.900-0005
# *Denali* Copyright (c) Denali Software, Inc., 1996-2001, All Rights
Reserved.
# *Denali* Class: flash_nand  Instance: "/den_model" Size: 8192Kx8
# *Denali* Class: internal  Instance: "/den_model(spare)" Size: 256Kx8
# Loading work.nand_int(structure)
# Loading work.pxa_bufif2(behavioral)
```

AMD UltraNAND Flash Memory

The test bench for the AMD UltraNAND Flash memory executes the following commands: "Page Program", "Read Status", "Read First Half Page", and "Input Data" as described in Figure 6. In executing these commands, the test bench fills a 528 byte buffer and programs the target page in the UltraNAND device. The status of the operations are checked through the "Read Status" operation.

Figure 10 illustrates loading the page address to the UltraNAND. Notice the ALE signal is high and WE# is asserted for each portion of the address write.



```
/amd_flash_tb/mem_buffer          {-2147483648
/amd_flash_tb/data_bus            00000000
/amd_flash_tb/cle                 0
/amd_flash_tb/ale                 1
/amd_flash_tb/ren                 1
/amd_flash_tb/wen                 0
/amd_flash_tb/wpn                 1
/amd_flash_tb/sen                 0
/amd_flash_tb/ry_byn              1
/amd_flash_tb/init_sig            0
/amd_flash_tb/cpld_ce             0
/amd_flash_tb/port_addr           0000
/amd_flash_tb/outce0              0
/amd_flash_tb/outce1              0
/amd_flash_tb/outce2              0
/amd_flash_tb/write               0
/amd_flash_tb/read                1
/amd_flash_tb/reset               0
/amd_flash_tb/ready               Z
/amd_flash_tb/fail_flag           0
/amd_flash_tb/com_latn            1
/amd_flash_tb/den_model/io        00000000
/amd_flash_tb/den_model/cle       0
/amd_flash_tb/den_model/ale       1
/amd_flash_tb/den_model/cen       0
/amd_flash_tb/den_model/ren       1
/amd_flash_tb/den_model/wen       0
/amd_flash_tb/den_model/wpn       1
/amd_flash_tb/den_model/sen       0
/amd_flash_tb/den_model/ry_byn    1
/amd_flash_tb/cpld/a0             0

2743897 ps
7759724 ps

2119340034 ps to 4169050827 ps
```

Figure 10: **Memory Address Load**

Figure 11 illustrates the completion of the "Page Program" command. Notice the RY/BY# signal is asserted, representing that the flash device is busy with an operation. After the "Page Program" command is sent to the flash, a "Read Status" command is sent. The "Read Status" command is used to read the device status. When I/O6 is equal to ’1’, the device is ready for the next command. To check if an operation is successful, reading I/O0 will determine the pass/fail status. When I/O0 is equal to ’0’, the operation passed and when equal to ’1’, the operation failed. Notice where the simulation is highlighted in Figure 11, the program operation was successful.

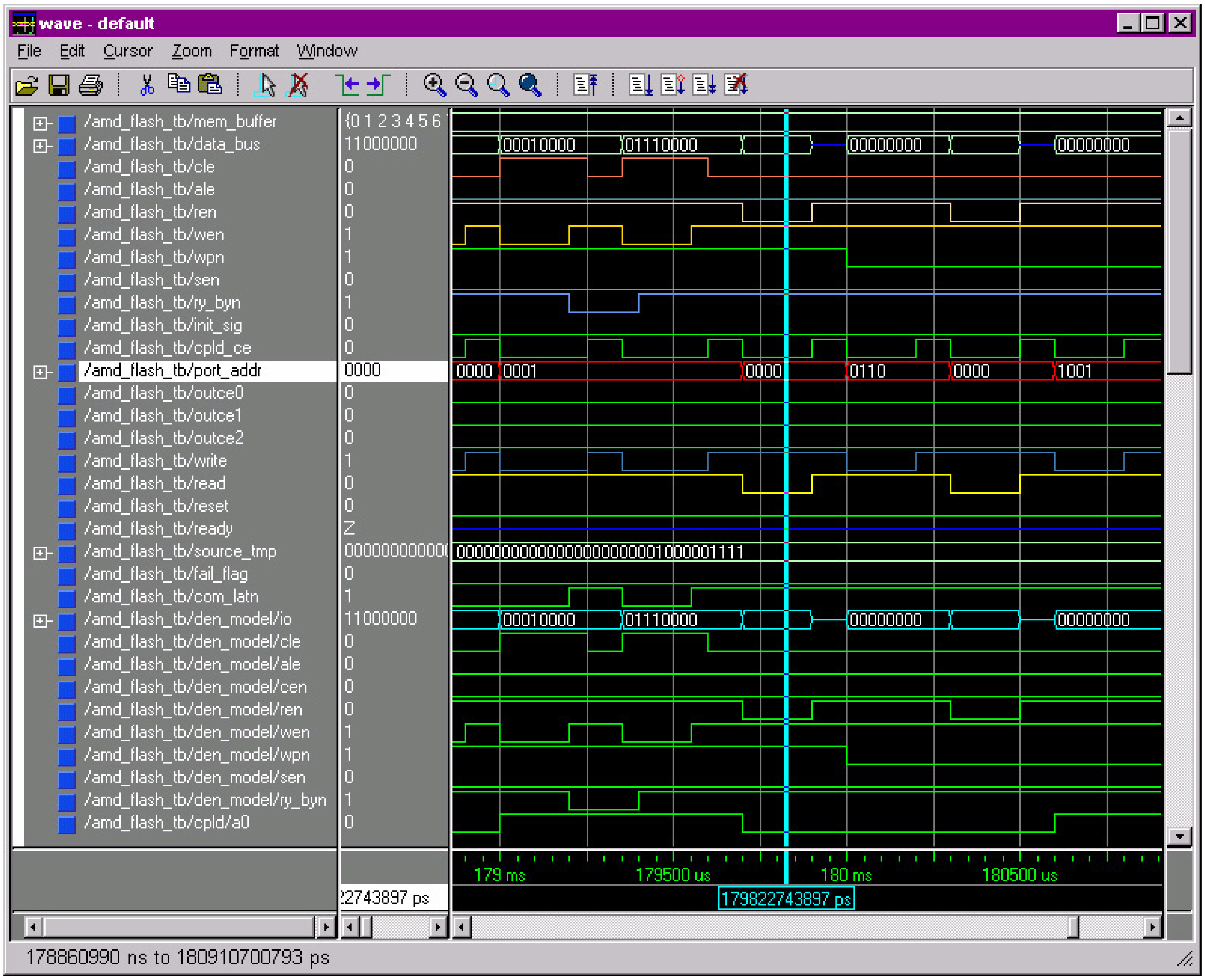


Figure 11: **Program Status: "Ready"**

Samsung Flash Memory

Testing the NAND interface with the Samsung K9F4008W0A model was performed similar to the AMD UltraNAND device. The following commands were executed with the Denali model (in ModelSim) with the test bench provided: "Frame Program" and "Status Read". To program a frame, the Frame Register must be loaded with data prior to executing the "Frame Program" command. The test bench provided loads the Frame Register with 32 bytes of data. The status of the device is checked with the "Status Read" command.

HDL Code

ABEL - **ftp://ftp.xilinx.com/pub/applications/refdes/xapp354_abel.zip**

VHDL - **ftp://ftp.xilinx.com/pub/applications/refdes/xapp354_vhdl.zip**

Verilog - **ftp://ftp.xilinx.com/pub/applications/refdes/xapp354_verilog.zip**

Conclusion

Xilinx CoolRunner XPLA3 CPLDs are the perfect target device for interfacing with system memory devices. This NAND Flash interface can be modified to support multiple memory banks and suit any application requirements. CoolRunner CPLDs are ideal for any memory interface needs in portable and handheld devices. CoolRunner CPLDs are low power and easy to design with using the WebPOWERED software tools.

References

The web sites shown here are valid as of the publication date of this note.

1. Samsung Flash Memory Devices (**http://samsungelectronics.com/semiconductors/Flash/Flash.htm**)
2. Denali Software, Inc. (**http://www.denalisoft.com/**)
3. eMemory (**http://www.ememory.com/**)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/30/01	1.0	Initial Xilinx release.
09/30/02	1.1	Minor changes.

XAPP398 (v1.0) September 23, 2003

CompactFlash Card Interface for CoolRunner-II CPLDs

Summary

This application note describes the card-side implementation of an 16-bit CompactFlash (CF+) card interface using a CoolRunner™-II CPLD. Included in this implementation are the CIS, Attribute Memory Control and Status Registers, 16-bit Common Memory, and 8-bit I/O Interface. This design can be easily modified to interface to any memory, DSP or microcontroller. This application note does not describe the Host Bus Adapter (HBA) side of the CompactFlash interface, which is resident on a host system such as a Personal Computer or PDA.

Introduction

There are two types of CompactFlash devices: CompactFlash Storage Cards (CF), and CF+ Cards. CF consists only of common memory data storage whereas CF+ is expanded to include I/O devices or magnetic disk data storage, depending on the specific application. This application note targets the CF+ type of device. There are three basic modes of operation called for in the CF specification. Typically, CF+ devices operate in two of the three basic modes: PC Card ATA using I/O Mode, and PC Card ATA using Memory Mode. The third, optional mode is True IDE Mode. CF cards are required to operate in all three modes. This application note implements the CF+ card side interface utilizing PC Card ATA using I/O Mode and PC Card ATA using Memory Mode. True IDE Mode is not addressed in this application note, but can be implemented by the designer using the CoolRunner-II CPLD and this reference design.

The CompactFlash interface consists of two main components: the Host Bus Adapter (HBA) and the card side interface. The former resides on the host side of the interface, which typically is built into the socket of a personal computer or Personal Digital Assistant (PDA). The card side interface resides on the CF card itself. Described in this application note is the implementation of a controller which resides in the card side of the interface.

Within the card, there is Attribute Memory, Common Memory, and I/O Interface. Attribute memory consists of the Card Information Structure (CIS) and the Configuration and Control Registers. Common Memory, in this reference design, interfaces to the Intel StrataFlash 28F320J3 memory. The I/O logic interfaces to the Analog Devices, Inc. ADSP-218xN Series DSP using the 8 bit I/O Space of the DSP.

In this implementation, the CF+ interface is a 16-bit data bus. The I/O interface within the card consists of an 8-bit bus to the DSP.

It is necessary to refer to the CompactFlash Specification and the PCMCIA Specification to fully understand the discussion in this application note.

A free downloadable zip file containing the VHDL source code and a testbench for this reference design is available as described below in **Source Code Download**, page 117.

Electrical Interface

Power Considerations

CoolRunner-II CPLDs are 1.8V core voltage devices with I/O banking features that allow for various I/O voltages and tolerances up to 3.3V. The CF+ specification calls for either 3.3V or

5.0V operation. The core power supply to the CPLD, therefore, must be regulated on the CF+ card to 1.8V while the I/Os must be configured to operate at 3.3V as described in the following section. For the specific application, if 1.8V regulation is not available or the I/Os will be subjected to 5.0V, consider using the CoolRunner XPLA3 CPLD, which is a 5V tolerant 3.3V device and therefore will not require the additional power regulation. I/O considerations for the CoolRunner-II CPLD are discussed in the following section.

To indicate to the host that the CF+ card is expecting 3.3V signaling, -VS1 must be held at GND. The signal -VS2 is reserved by PCMCIA for a secondary voltage and therefore must be left open. These two signals are not implemented in this application note and therefore must be electrically considered external to the CPLD on the PCB by the designer.

CF+ devices that are configured for 3.3V operation are limited to 75mA max (Power Level 0) or 500mA max. (Power Level 1). Further, during power up and after reset, the CF+ card is limited to Power Level 0. To assist the designer with more efficiently utilizing the CF+ power budget, CoolRunner-II CPLDs are the perfect choice since these devices exhibit very low static and dynamic current consumption.

I/O Considerations

Inputs

The CompactFlash specification calls for three basic types of input configurations in the Pin Assignments and Pin Type description: IxZ, IxU and IxD. "I" represents the pin is an input, "Z" represents an input with no resistive termination, "U" represents a weak pullup termination, and "D" represents a weak pulldown termination. These three configurations are also specified with one of three electrical characteristics which is denoted by a number 1, 2 or 3 in the "x" position of the designation. The three numbers correspond to various input threshold levels for the type of input required. Details of these values and configurations can be found in the CompactFlash specification.

Outputs

Similarly, the CF+ specification calls for three basic types of output configurations: OTx, OZx, OPx, and ONx. "O" designates the signal to be an output, "T" represents a totem pole type output, "Z" a CMOS P-channel/N-channel output with 3-state capabilities, "P" a PMOS only output, and "N" an NMOS only output. These also have three electrical characteristics denoted by 1, 2, or 3 in the "x" position of the designation. These three values represent V_{OH} and V_{OL} levels under certain test conditions. All output types also have a 3-state characteristic and, together with the voltage levels, are described in the CompactFlash Specification.

Implementation

The CF specification calls for 3.3V or 5.0V for the card power supply and I/O signals. CoolRunner-II devices are 1.8V devices, but have I/O banking capabilities. To implement the CF+ interface, the CoolRunner-II CPLD I/Os should be configured as 3.3V LVCMOS. CoolRunner-II CPLDs are not 5V tolerant. However, external circuitry can be used to interface to 5V systems. See XAPP364. Alternately, CoolRunner XPLA3 CPLDs are 3.3V devices with 5.0V tolerance. If these voltage requirements are an issue for the application, the CF+ implementation can target the CoolRunner XPLA3 CPLD. Since the CF+ card tells the HBA what voltage it expects, 5.0V is not present on the I/O pins until the HBA determines the required voltage. CoolRunner-II CPLDs are therefore acceptable for CF+ applications since 5.0V will never be applied to its I/Os, as long as -VS1 and -VS2 are configured correctly.

Since this application note describes the implementation of the two modes PC Card ATA using I/O Mode and Memory Mode, the I/Os are required to be configured as I1Z, I2Z, I1U, I3U, OZ3, OT1, and OT3. The CoolRunner-II CPLD can be configured to support all I/O modes with the following caveats.

I/O configuration I2Z requires that V_{IH} be at a lower voltage than the CoolRunner-II CPLD is specified in the data sheet. The signal of interest is RESET whose V_{IH} should be analyzed for the particular application of this CF+ design. If it is determined that the supplied signal is always high enough to meet the V_{IH} value of CoolRunner-II devices, then this CPLD should be sufficient. If not, an external buffer with these characteristics must be supplied to satisfy the system requirements.

I/O configurations OT1 and OT3 require a totem pole type of driver. CoolRunner CPLDs only provide CMOS type output buffers and therefore cannot explicitly meet these requirements. Therefore, an external buffer of totem pole type should be implemented to satisfy the CF+ specification. However, CoolRunner-II CPLDs can meet the I/O drive requirements of V_{OH} and V_{OL} at the specified I_{OH} and I_{OL} test conditions respectively, regardless of CMOS or totem pole type configuration. Analysis of the particular application is necessary to determine if CoolRunner-II CPLD I/Os can be used without the use of external totem pole buffers.

Fixed Level and Unused I/Os

Nomenclature of all signals in the VHDL source code matches that of the CF+ specification for PC Card I/O Mode.

The signals -CD1 and -CD2 are card detect pins which indicate to the host that the card is fully inserted when it detects both signals are low. This application note does not implement these signals within the CPLD, but instead relies on the system designer to hold these two signals at GND external to the CPLD.

The signal -CSEL is not used by this application as described by the CF+ specification.

-SPKR, binary audio, is not used in this reference design, but can be added if required.

-VS1 and -VS2, voltage sense signals, are also not used in this implementation, but must be hardwired on the PCB so the host system can correctly determine the card's required voltage levels. For this reference design, it is assumed that -VS1 is held LOW while -VS2 is left floating. These two signals are held HIGH by the host system, thereby allowing -VS2 to be read as a logic HIGH when the card leaves it in a floating state.

Block Diagram

CF+ Card

Figure 1 shows a block diagram of the CF+ card where the major components are the CoolRunner-II CPLD, Intel StrataFlash and the Analog Devices DSP.

The CoolRunner-II CPLD implements the direct interface to the CF+ slot, an interface to the Common Memory space and an interface to the I/O Space. Control logic is included to synchronize data between the three interfaces. The Attribute memory as a whole is realized in the CPLD which includes control and status registers as well as the CIS. External ROM is not needed for the CIS since it has been compactly implemented in the CPLD as a lookup table constructed of product terms.

The Intel StrataFlash is used for the Common Memory space and is limited to 2 kB due to the 11 available address lines of the CF+ interface. To access further memory space, the CF+ card must be configured to utilize the I/O Space or redesigned to implement the True IDE mode. The Common Memory space is configured with a 16-bit data bus.

The I/O Space, for this implementation, consists of an Analog Devices DSP. In Figure 1, there is a reference to Additional Functions. This is included for illustrative purposes and can be any function, such as a GPS device. An 8-bit data bus is used to interface to the DSP.

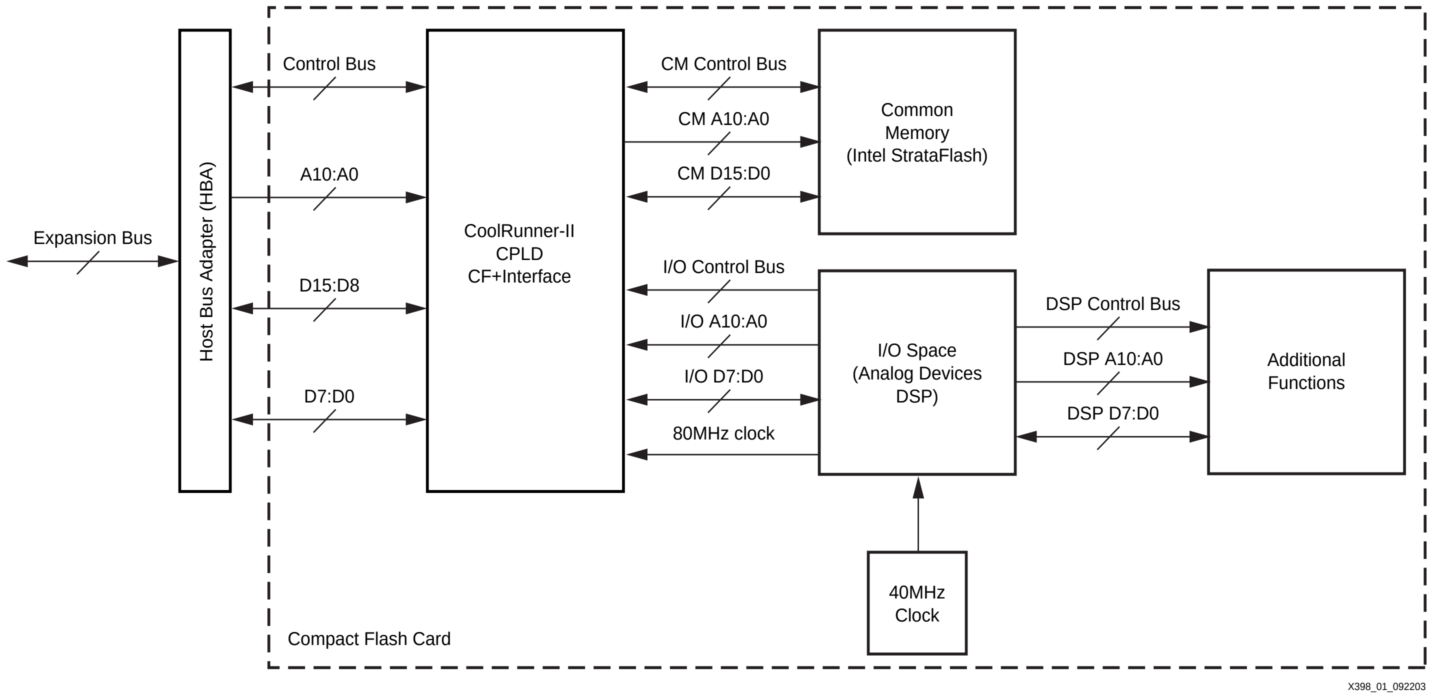


Figure 1: **CompactFlash Card General Block Diagram**

CPLD Implementation

Figure 2 represents an overview of the CF+ controller reference design contained in the CPLD (as described in this application note). There are two controllers for the three interfaces: CF+ Address Decode and Control Logic (CF+ Card Controller), and I/O Space Address Decode and I/O Control Logic (I/O Space Controller).

The former controls transactions with the CompactFlash interface and directs data to the correct locations—more specifically, the Attribute Memory, Common Memory, and the I/O Space Controller. It also provides the necessary status and control signals required by the HBA to effectively communicate with the card.

The I/O Space controller interfaces the CF+ card controller with the devices connected to the I/O Space bus, which is in this case the DSP. To interface more effectively and synchronously with the DSP, there are three I/O register banks: Address Register, Data Register, and Status Register.

As mentioned earlier, the Attribute memory is included in the CPLD reference design. The Attribute Memory, in this case includes the CIS, COR, CSR and PRR, whose functionality is described later in this document.

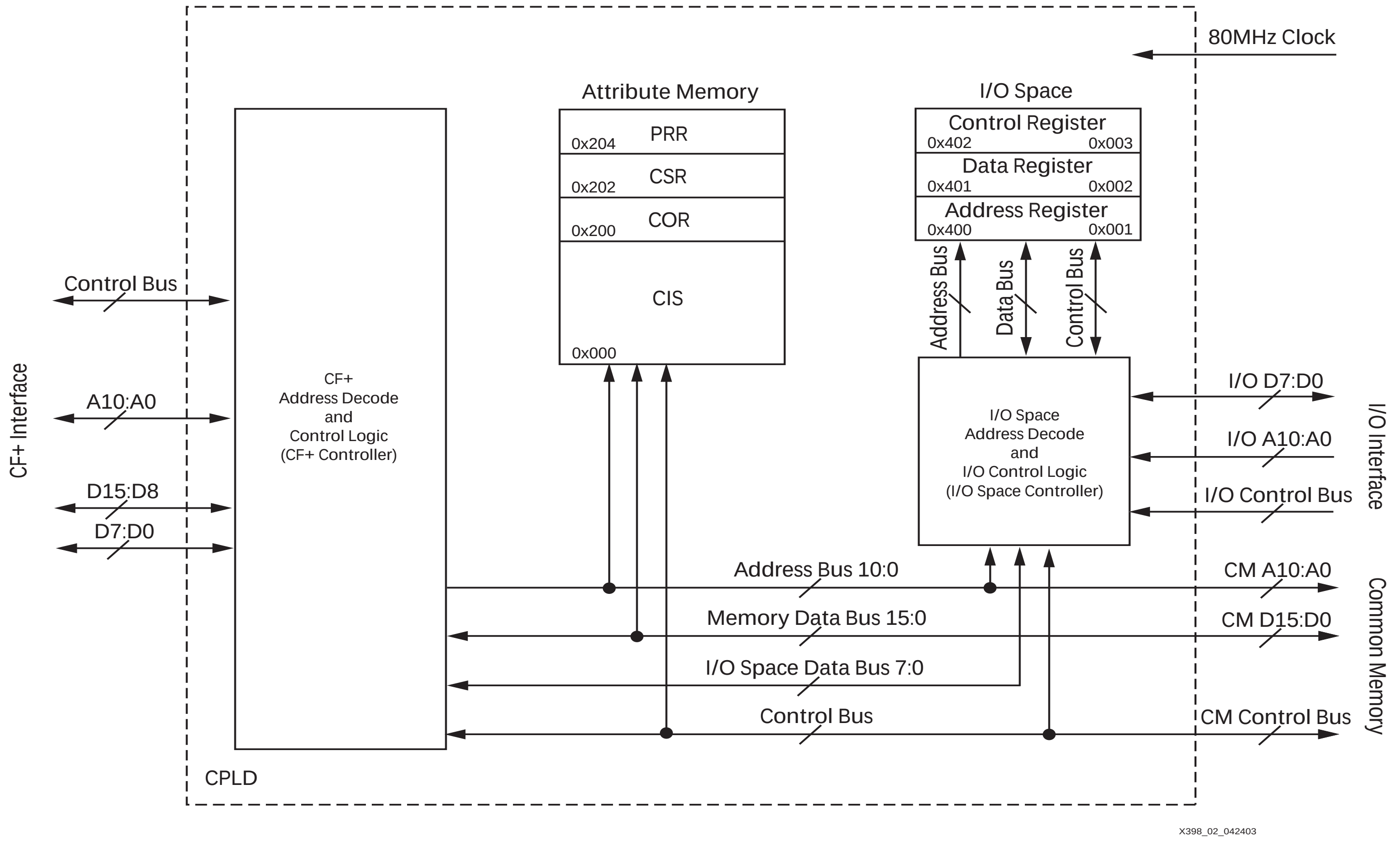


Figure 2: **CPLD Controller General Block Diagram**

Signal Descriptions

Table 1 displays the pin descriptions of the CoolRunner-II CF+ Interface. Note that according to the CompactFlash specification, some pin names change depending on the card configuration mode. For these signals, the I/O Mode nomenclature is used in the table, throughout this document, and in the VHDL source code.

Table 2 describes the pins of the Common Memory Interface to the Intel StrataFlash device.

Similarly, Table 3 indicates pin names and their functions for the I/O Space Interface to the Analog Devices DSP.

No pin assignments have been forced, leaving it up to the user to custom fit the design per their requirements.

Table 1: **CF+ Interface Pin Descriptions**

Name	Direction	Description
host_addr(10:0)	Input	A10:A0 in the Compact Flash Specification which are address lines from the HBA.
ce1_n	Input	Active LOW card select signal. Together with ce2_n and host_addr(0), selects between 16/8-bit data transfers and odd/even byte transfers. See Table 4 for details.
ce2_n	Input	Active LOW card select signal. When ce1_n is LOW, selects the odd or even byte of the data word depending on host_addr(0). See Table 4 for details.
iord_n	Input	Used in I/O Mode, this is an active LOW I/O read strobe from the host. Data is placed on the CF+ bus by the CF+ card.
iowr_n	Input	Also used in I/O Mode, this signal clocks data into the CF+ card when the host has placed valid data on the data bus of the CF+ interface. iowr_n is active LOW.
oe_n	Input	In Memory mode, this signal is used as a strobe by the host to read data from Common Memory or Attribute Memory including the CIS. In I/O Mode, this signal is used to read data from the Attribute Memory and CIS. oe_n is active LOW.
reg_n	Input	Active LOW signal that distinguishes between Common Memory (HIGH) and Attribute Memory (LOW). When in I/O Mode, this signal must be LOW to access I/O Space.
reset	Input	This is an active HIGH signal to initialize and reset all registers in the CF+ card.
we_n	Input	Active LOW signal to strobe data into the Attribute memory and the Common Memory.
stschg_n	Output	If enabled in the CSR, this active LOW pin indicates the status of the rdy/-bsy pin. When in I/O Mode, rdy/-bsy is renamed to ireq_n and its functionality no longer reflects the ready/busy condition. For the host to see ready/busy conditions, stschg_n is available. When in Memory Mode, stschg_n is always HIGH.
inpack_n	Output	When configured in I/O Mode, the CF+ card uses this active LOW signal to indicate the card is responding to an I/O Space read cycle at the address present on the host_addr bus.

Table 1: **CF+ Interface Pin Descriptions *(Continued)***

Name	Direction	Description
ireq_n	Output	ireq_n is active LOW. In Memory Mode, this signal indicates a ready/busy state (rdy/-bsy) where a LOW signal indicates a busy state. The CompactFlash specification requires this signal to be LOW during power up. During power up/configuration, the CoolRunner-II 3-states the I/Os with weak pullup, making it impossible to meet this requirement. However, the specification requires the HBA not access the card until >1ms after power has stabilized after which the HBA must assert the reset signal for 10µs. Since configuration times for CoolRunner-II CPLDs is much less than 1ms, this allows the card to have a HIGH rdy/-bsy signal during power up without adverse effects. The rdy/-bsy signal in Memory Mode is LOW during a reset, either a reset directed by the reset pin or a soft reset when the SRESET bit has been written in the COR. This signal also reflects the status of the Common Memory pin, cm_sts. During I/O Mode, this pin takes on the ireq_n functionality and masks the rdy/-bsy functionality. This active LOW signal now indicates that data is ready to be read from the I/O Space (DSP) Address or Data registers. To obtain the status of rdy/-bsy, the PRR bit 1 must be read. A change in status of this bit is reflected on the stschg_n pin, if enabled.

Table 1: **CF+ Interface Pin Descriptions *(Continued)***

Name	Direction	Description
wait_n	Output	As an active LOW signal, wait_n tells the HBA to extend the current memory or I/O cycle. This signal has been implemented such that the I/O Space and Common Memory have control over this signal. The dsp_ioms_n signal is sampled when accessing the I/O Space and indicates the DSP is reading or writing to the I/O Space. As implemented with the Analog Devices DSP, the signal ioms_n is used to drive dsp_ioms_n. Since the DSP has configured with three wait states, this assists the HBA to wait the appropriate length of time during an I/O Space access. The Common Memory uses cm_wait to drive wait_n during this type of memory access. However, as implemented with the Intel StrataFlash, this memory has no wait signal and therefore cm_wait must be driven HIGH external to the CPLD or removed from the source code.
host_data_low(7:0)	Bidirectional	D7:D0 in the CompactFlash specification. These are the lower data signals to/from the HBA. This corresponds to the even byte described in the CompactFlash specification.
host_data_high(7:0)	Bidirectional	D15:D8 in the CompactFlash specification. These are the upper data signals to/from the HBA.This corresponds to the odd byte described in the CompactFlash specification.

Table 2: **Common Memory Interface Pin Descriptions**

Name	Direction	Description
cm_sts	Input	Effectively a ready/busy signal for the Intel StrataFlash. This implementation assumes the memory is configured with level mode sts signalling and therefore supplies the CF+ rdy/-bsy logic (named ireq_n) a busy state (LOW) when the memory is busy performing a lengthy operation. cm_sts is active LOW.
cm_wait	Input	Available for Common Memory wait type status inputs. As the Intel StrataFlash memory used in this implementation does not contain a wait signal, cm_wait is unused and driven high in the test bench. The user must either permanently drive this signal HIGH or remove it from the source code so as to allow proper functionality of wait_n. cm_wait is active LOW.
cm_byte_n	Output	Active LOW, cm_byte_n selects the 8-bit or 16-bit data bus access modes of the Intel StrataFlash as requested by the HBA. When in 8-bit mode, cm_addr(0) selects the high or low byte of the addressed memory location. When in 16-bit mode, cm_addr(0) is ignored and cm_addr(1) becomes the LSB of the Common Memory address bus.
cm_addr(0)	Output	Byte-select address for the Intel StrataFlash. This signal selects the high or low byte of the addressed memory location. A high byte from Common Memory corresponds to an odd byte with respect to the CF+ interface. Similarly, a low byte from Common Memory corresponds to an even byte on the CF+ interface. High bytes are accessed when cm_addr(0) is HIGH and low bytes are accessed when cm_addr(0) is LOW.
cm_addr(10:1)	Output	Address bus for the Common Memory.
cm_reset	Output	Active LOW reset for the Common Memory
cm_read_n	Bidirectional	Active LOW output enable signal to read data from the Common Memory.
cm_data(15:0)	Bidirectional	Data bus for the Common Memory.
cm_ce_n	Bidirectional	Active LOW chip enable signal for the Common Memory.
cm_write_n	Bidirectional	Active LOW write enable signal for the Common Memory.

Table 3: **I/O Space Interface Pin Descriptions**

Name	Direction	Description
dsp_clk	Input	This clock input to the CPLD must be 80MHz
dsp_addr(10:0)	Input	11-bit address bus from the DSP. Most of these pins can be removed from the design to obtain more I/Os as needed. This implementation only utilizes the two LSBs since there are only three registers accessed by the DSP.
dsp_ioms_n	Input	I/O memory select from the DSP. This active LOW signal enables the I/O Space from the DSP side of the CF+ interface. The CF+ interface signal wait_n samples this signal, and when low, indicates the DSP is either reading or writing to the I/O Space interface.
dsp_rd_n	Input	Active low-read strobe from the DSP which accesses data contained in the I/O Space registers.
dsp_wr_n	Input	Active low-write strobe from the DSP which accesses data contained in the I/O Space registers.
dsp_data(7:0)	Bidirectional	8-bit bidirectional data bus from the DSP.

CF+ Controller

All primary control functions for the CF+ interface are contained in the CF+ controller (cf_plus_control.vhd). The following subsections describe the CF+ controller’s functionality.

Addressing Modes

This CF+ interface consists of a 16-bit data bus, which may or may not be used to its fullest capabilities, depending on the architecture of the HBA and the host system. Some host systems can only support 8-bit transfers which therefore requires the HBA to request 8-bit data bytes from the CF+ card. It is the card controller’s responsibility to provide 8 bits of data to the HBA. In the case of the 8-bit host, it is the HBAs responsibility to request the upper or lower byte of a 16-bit data word from the CF card using specific control signals and then sending that data to the host over the 8-bit data bus in the correct order.

The CF+ implementation in the CPLD reads the control signals (host_addr(0), ce1_n and ce2_n) from the interface and interprets them in the correct order for providing data over the 8-bit or 16-bit data bus as required by the HBA. Table 4 describes in detail this interpretation, which is compliant to the CompactFlash specification. This table references Odd and Even bytes as described in the CompactFlash specification. The Intel StrataFlash maintains a nomenclature of High and Low bytes. Table 4 conveniently shows the correlation between Odd/Even and High/Low nomenclatures.

Currently, I/O Space access is limited to 8-bit in this implementation. If-16 bit I/O Space access is required, comments have been added throughout the source code to assist the user to this end. However, 16-bit I/O Space access has not been tested.

Table 4: **Addressing Even and Odd Bytes from CF+ Interface**

Addressing Mode	ce1_n	ce2_n	host_addr(0)	host_data_high	host_data_low
No access	1	1	X	3-State	3-State
8/16-bit Mode (even byte)	0	1	0	3-State	Even Byte (Low Byte)
8-bit Mode (odd byte)	0	1	1	3-State	Odd Byte (High Byte)
16-bit Mode (odd byte only)	1	0	X	Odd Byte (High Byte)	High-Z
16-bit Mode (even & odd bytes)	0	0	X	Odd Byte (High Byte)	Even Byte (Low Byte)

Memory and I/O Space Access

The HBA gains access to the different memory spaces via a few control pins. The signal reg_n allows the HBA to write or read from Common Memory when this signal is HIGH. Note that when reg_n is LOW, the HBA has access to either the Attribute Memory or the I/O Space, dictated by the state of iord_n, iowr_n and, of course, the address.

When reg_n is LOW, the HBA can perform read and write operations on the I/O Space registers. To perform a read, iord_n is held LOW. Similarly, holding iowr_n LOW will access the I/O space registers in a write mode.

All memory spaces (Attribute Memory, Common Memory and I/O Space) are read and write capable. There are two exceptions: The CIS is read-only, and the I/O Space Status Register, which is read-only as well.

Interrupt Request and Ready/Busy

A CF+ card can be configured for several modes of operation, as mentioned earlier. Two of these modes are supported in this implementation: Memory Mode and I/O Mode. Each mode has unique I/O signal descriptions as described in the CompactFlash specification. In other words, when the card is configured in Memory Mode, the I/Os are defined to have a specific set of I/O functions and names. But when the card has been reconfigured for I/O Mode, some of these pins take on a new name and functionality. Of interest is the rdy/–-bsy pin named ireq_n in this implementation. The CompactFlash specification names this pin rdy/–bsy for Memory Mode and changes it to –ireq in I/O Mode. See Table 1, page 104.

In Memory Mode, this signal indicates a ready/busy state (rdy/-bsy) where a LOW signal indicates a busy state. The CompactFlash specification requires this signal to be LOW during power up. During power up/configuration, the CoolRunner-II 3-states the I/Os with weak pullup, making it impossible to meet this requirement. However, the specification requires the HBA not access the card until >1ms after power has stabilized, after which the HBA must assert the reset signal for 10µs. Since the configuration time for CoolRunner-II CPLDs is much less than 1ms, this allows the card to have a HIGH rdy/-bsy signal during power up without adverse effects.

The rdy/–bsy signal in Memory Mode is LOW during a reset, either a reset directed by the reset pin or a soft reset when the SRESET bit has been written in the COR. This signal also reflects the status of the Common Memory pin, cm_sts. The Intel StrataFlash provides a signal named STS which, in level mode, acts as a ready/busy signal. This CF+ implementation uses this STS signal to reflect the ready/'busy status of the Common Memory when it is being accessed.

During I/O Mode, this pin takes on the ireq_n functionality and masks the rdy/–bsy functionality. This active LOW signal now indicates that data is ready to be read from the I/O Space (DSP) Address or Data registers. When new data is placed in the Address or Data registers by the DSP, ireq_n goes LOW. Once data has been read from these registers by the HBA, ireq_n returns to its inactive HIGH state.

Since rdy/–bsy status is masked during I/O Mode, the PRR bit 1 must be read to determine the status of the Common Memory. A change in status of this PRR bit is reflected on the stschg_n pin, if enabled.

Attribute Memory

The Attribute Memory is divided into two basic sections: The CIS and the Configuration Registers. There are at least six optional Configuration Registers, but this implementation integrates three of these registers. The CIS may be found in the reference design source file named cis.vhd. The Attribute Memory may be found in attribute_memory.vhd, but some associated control signals for this memory space are found in cf_plus_control.vhd.

Addressing Attribute Memory

The CIS is found at memory location 000h per the CompactFlash specification and is read-only. The CompactFlash specification states that for CompactFlash and other cards with data storage, the configuration registers must reside at a base offset (BASE) of 200h. Conversely, non-data storage cards BASE can be located anywhere and is found by parsing the CIS tuples. This implementation, since data storage is used, sets BASE to 200h. Table 5 describes the addressing scheme of the Attribute Memory.

Table 5: **Attribute Memory Registers**

Address	Register	Description
000h	CIS	Card Information Structure
BASE + 00h	COR	Configuration Option Register
BASE + 02h	CSR	Card Configuration and Status Register
BASE + 04h	PRR	Pin Replacement Register

Card Information Structure

The Card Information Structure (CIS) is maintained in the Attribute Memory as a read-only memory space and is intended as nonvolatile memory. In this CoolRunner-II implementation, the CIS is built using product terms and therefore retains its nonvolatile properties without using external memory.

This data structure is comprised of tuples which tell the software driver of the host system what characteristics the CF+ card contains, such as size, speed and resources required. The host then configures the card and the HBA to efficiently take advantage of the card's features. The structure of the tuples and the definitions of the values within the tuples is not described here. It is up to the user to investigate this information as found in the CompactFlash specification and the PCMCIA specification.

Included in this implementation is an example CIS which must be modified to the user's application. To modify the CIS, edit the source file cis.vhd.

Configuration Option Register

The Configuration Option Register (COR) is one of the registers included in the Attribute Memory and is used to configure the card. Configuration options include soft reset, interrupt types and address decoding.

The COR is comprised of 8 bits as shown in Table 6.

Table 6: **Configuration Option Register**

Operation	D7	D6	D5	D4	D3	D2	D1	D0
R/W	SRESET	LevelReq	Conf5	Conf4	Conf3	Conf2	Conf1	Conf0

SRESET - Soft Reset

When this bit is written by the HBA, a soft reset of the card will occur. To obtain this functionality, the HBA must write a HIGH then a LOW where the reset is performed during the HIGH cycle. A soft reset differs from a hard reset in that this bit is not cleared by a soft reset. Otherwise, both types of reset have the same functionality. All registers and state machines in the card will be reset to zero upon hard or soft reset, including registers in the I/O space. A reset condition is also presented to the Common Memory. This bit is initially set LOW by a hardware reset condition when driven by the reset pin.

LevelReq

ireq_n can be set to indicate interrupts on a level or pulse mode basis. Setting this bit HIGH enables level mode interrupts, whereas setting this bit LOW enables pulse mode interrupts. Pulse mode is set to 0.5µs per the CompactFlash specification and uses a 6-bit binary up counter (upcnt6.vhd) to implement the pulse width. This bit is set to zero by reset.

Conf

These 6 bits select the operation mode of the card. The specification states that for CompactFlash Cards these bits are used for Memory Mapping or I/O Mapping, depending on the application. But since this implementation is of a CF+ card, the specification states that these bits specify either Memory Mapping where I/O cycles are ignored (all bits zero), or the bits are user-defined. The specification also states that multiple function CF+ cards bits 0 through 2 have specific functionality and bits 3 through 5 are reserved for user implementation.

This CF+ implementation handles the Conf bit functionality where an all zeros condition disables the I/O space and effectively allows for memory mapping only. Also, since single function CF+ cards use these bits optionally, and since this implementation is of a single function CF+ card, these bits are not required. However, they have been included for convenience. All Conf bits are set to zero when a reset condition occurs.

Conf0 enables the I/O space when set HIGH, and disables the I/O function if LOW.

Conf1 is used for Base and Limit Registers, but since these registers are not used in this implementation, Conf1 is non-functional.

Conf2 enables ireq_n routing. Its functionality depends on the state of Conf0. If Conf0 is HIGH and Conf2 is HIGH, interrupts from the I/O Space are routed to the ireq_n pin. If Conf0 is HIGH and Conf2 is LOW, interrupts are disabled. If Conf0 is LOW, Conf2 is undefined.

Conf3 through Conf5 are reserved for user implementation.

Card Configuration and Status Register

Another register in the Attribute Memory, the Card Configuration and Status Register (CSR) contains information regarding the card's condition.

The CSR is comprised of 8 bits as described in Table 7.

Table 7: **Card Configuration and Status Register**

Operation	D7	D6	D5	D4	D3	D2	D1	D0
Read	Changed	SigChg	IOis8	XE_n	Audio	PwrDwn	Int	0
Write	0	SigChg	IOis8	XE_n	Audio	PwrDwn	0	0

Changed

This bit reflects the status of CRdy_Bsy_n and CWProt in the PRR. Note that the functionality of CWProt is not implemented since WProt (Write Protect) is not supported by the CompactFlash specification.

When CRdy_Bsy_n is HIGH, indicating Rdy_Bsy_n has changed state, the Changed bit goes HIGH. Additionally, this bit will drive the stschg_n pin LOW indicating Rdy_Bsy_n has changed states, but only if the SigChg bit is HIGH and the card is configured to function in the I/O mode.

The Changed bit is reset to zero by a reset condition. Note that this bit is read-only.

SigChg

As a readable and writable bit, SigChg controls whether the stschg_n pin reflects the Changed bit status. If SigChg is HIGH, the stschg_n pin indicates the status of the Changed bit when the card is configured with I/O Space. If SigChg is LOW, the stschg_n pin will be held HIGH when the card is configured with I/O Space.

This bit is reset LOW during a reset condition.

IOis8, XE_N, Audio, and PwrDwn

These bits are unused in this implementation of the CF+ card. If the user desires to enable the functionality of these bits, code must be added to the source files. If a read is performed on the CSR, all of these bits will be LOW with the exception of IOis8, which will be HIGH.

A reset has no effect on these bits.

Int

The Int bit reflects the status of an interrupt request from the I/O Space, regardless of the setting of Conf2, the interrupt enable/disable bit. When HIGH, this bit indicates an interrupt is pending. The bit will clear when the interrupt has been serviced. This bit is read only and is reset to zero during a reset condition.

Pin Replacement Register

Located in the Attribute Memory space, this is an optional register that indicates the status of pins that have been remapped during I/O Space configuration. Some pins lose their functionality during I/O Space configuration, compared to Memory Mode configuration. Specifically, the rdy/–bsy pin from Memory Mode becomes ireq_n during I/O Mode. This register is 8 bits wide and is described in Table 8.

Table 8: **Pin Replacement Register**

Operation	D7	D6	D5	D4	D3	D2	D1	D0
Read	0	0	CRdy_Bsy_n	CWProt	1	1	Rdy_Bsy_n	WProt
Write	0	0	Host_CRdy_Bsy_n	CWProt	0	0	MRdy_Bsy_n	MWProt

Bits D2, D3, D6 and D7 are not writable. Bits D2 and D3 will be read as logic HIGH, whereas bits D6 and D7 will be read as a logic LOW.

CRdy_Bsy_n

When Rdy_Bsy_n changes state, this bit is set to a logic HIGH. This bit may also be written to by the host and is designated as Host_CRdy_Bsy_n in the source files. This bit is set LOW during a reset.

CWProt

Since write protect is not used as detailed in the CompactFlash specification, CWProt is unused in the source files and will read logic LOW.

Rdy_Bsy_n

This bit represents the internal state of the rdy/–bsy pin when it has been reallocated as ireq_n when the card has been configured for I/O Mode. When rdy/–bsy changes states (due to the cm_sts pin in this implementation), this bit reflects that state. When written HIGH by the host, this bit acts as a mask so that Host_CRdy_Bsy_n may be written by the Host. Writing to this bit is by using the MRdy_Bsy_n bit in the source code. This bit is set LOW during a reset condition.

WProt

Since write protect is not used as specified by the CompactFlash specification, this bit and MWProt are unused in this implementation. This bit is read as a logic LOW.

Common Memory

The functionality for the Common Memory space may be found in the reference design source file named cf_plus_control.vhd.

Common Memory has been implemented assuming an Intel StrataFlash 28F320J3 flash memory is used. Since there are only 11 address lines on the CF+ interface, there are only 2 kB of addressable common memory for the card. Common Memory is typically used as a working memory space which contains mapping of the larger memory arrays found in the I/O Space using True IDE Mode. Larger memory arrays are not included in this implementation, but can be added by the user.

The CF+ controller implements all functionality when interfacing the Common Memory. This includes addressing, byte mode select, reset, memory status, chip select and reset.

Addressing

The Intel StrataFlash memory can be accessed in either 16-bit or 8-bit modes. This works well with the CompactFlash specification since the interface is accessed by the HBA in either 16-bit or 8-bit modes. To access the Common Memory, reg_n must be HIGH. By using ce1_n and ce2_n the 16-bit or 8-bit modes can be selected as required. When in 16-bit mode, cm_addr(0) is driven HIGH or LOW depending on host_addr(0). When in 8-bit mode with cm_addr(0) HIGH, the high (odd) byte is accessed from Common Memory. Conversely, when in 8-bit mode, cm_addr(0) is LOW accessing the low (even) byte of Common Memory. When in 16-bit mode, cm_addr(0) is ignored by the flash memory. Addresses cm_addr(10:1) are used to specify the location of the byte or word in memory. Table 9 describes this relationship.

Table 9: **Common Memory Addressing Modes**

Addressing Mode	cm_addr(10:1)	cm_addr(0)	cm_byte_n
8-bit	host_addr(10:1)	host_addr(0)	0
16-bit	host_addr(10:1)	1	1

The signal cm_byte_n is used to indicate to the memory that byte mode is selected and is active LOW.

Memory status is reflected on the ireq_n pin, acting as rdy/–bsy in Memory Mode. This status is also available in the Rdy_Bsy_n bit of the PRR. The state of rdy/–bsy is directly related to the cm_sts pin of the Common Memory Interface.

I/O Space

Most I/O Space functionality is found in the reference design source file named dsp_interface.dsp. The interface state machine is located in this file. Some control signals are located in cf_plus_control.vhd. The pulse mode IRQ requires use of the counter found in upcnt6.vhd.

For the I/O Space, the CF+ specification states that either 8-bit or 16-bit I/O access is allowable. This application note describes the implementation of an 8-bit I/O Space as provided in the reference design. If 16-bit access is required, the reference design can easily be modified to support this requirement. There are comments placed throughout the source files (VHDL) to convert the I/O Space to 16-bit access, although 16-bit I/O Space access has not been tested.

Clock

This Compact Flash implementation requires an 80 MHz clock signal to be supplied to the CPLD. This particular implementation with the Analog Devices ADSP-218xN series DSP, uses the CLKOUT signal found on the DSP as the clock source. CLKOUT is a signal provided by the DSP and is double the input clock signal to the DSP found on pin CLKIN. This implementation assumes a 40 MHz clock on CLKIN of the DSP which subsequently becomes 80 MHz on CLKOUT. The CPLD requires this 80 MHz signal on the dsp_clk pin to function correctly. If a different clock speed is required for another implementation, it is recommended that a functional and post-route simulation be performed to ensure correct functionality is retained.

Although the CF+ interface itself contains no clock, this 80 MHz clock can be used to ensure proper timing of the interface signals.

Addressing I/O Space Registers

Three registers are provided as the communications venue to/from the I/O Space interface: Address Register, Data Register, and Status Register. There is no direct communication with the I/O Space—the communication is instead provided indirectly through these three registers. Their functionality is described in the following sections.

To gain access to these registers, there are two sets of addresses: one for the DSP and one for the HBA. This is intended to provide access to the same registers using two different memory maps. Table 10 describes this addressing scheme.

Table 10: **Addressing I/O Space Registers**

DSP Address	HBA Address	Register	Description
001h	400h	io_addr_reg	Address Register
002h	401h	io_data_reg	Data Register
003h	402h	io_status_reg	Status Register

Address Register

An 8-bit register is provided to pass address data to the DSP from the HBA, or in the reverse direction, from the DSP to the HBA.

Data Register

Similarly, the Data Register is an 8-bit register that is used to pass data to/from the I/O Space. It is read/write capable from both the DSP and the HBA.

Status Register

As a read-only register, Status Register contains information regarding the current Data Register and Address Register transactions. Table 11 describes the relationship of the bits in the register. All unused bits (D5:D2) are available for user implementation, as in the case of inserting additional I/O Space registers.

Table 11: **I/O Space Status Register**

Operation	D7	D6	D5	D4	D3	D2	D1	D0
Read	IRQ_CF	IRQ_DSP	0	0	0	0	DataReg	AddrReg
Write	0	0	0	0	0	0	0	0

IRQ_CF

When new data is available in the Data Register or the Address Register given by the DSP, this bit supplies an interrupt request to the HBA. A HIGH condition triggers an active LOW interrupt on ireq_n, provided interrupts are enabled in Conf(0) of the COR. An interrupt condition is recorded in the Int bit of the CSR regardless of the Conf(0) bit state.

To safeguard data in the Address and Data Registers prior to the HBA reading the new data, the HBA is not permitted to write to these registers until the IRQ_CF bit has cleared. This bit is cleared when the HBA reads data from either the Address or Data Registers or a card reset has been performed.

The IRQ_CF bit is found in the reference design source files as io_status_reg(7).

IRQ_DSP

When new data is available in the Data Register or the Address Register given by the HBA, this bit, in a HIGH state, indicates an interrupt request is pending to the DSP. This reference design does not utilize an interrupt request pin for the DSP, and therefore, this bit is unused throughout this implementation. This bit is available only for the purposes of the user if an interrupt request pin is desired to be added for a specific application.

To safeguard data in the Address and Data Registers prior to the DSP reading the new data, the DSP is not permitted to write to these registers until the IRQ_DSP bit has cleared. This bit is cleared when the DSP reads data from either the Address or Data Registers or a card reset has been performed.

This bit is found in the reference design source files as io_status_reg(6).

DataReg

When the Data Register has been written by either the DSP or the HBA, this bit is set HIGH. This bit is used to indicate the Data Register has been written when an interrupt has been detected by the system monitoring interrupts. This bit is cleared when the Data Register has been read by the target system, or a card reset has been issued.

This bit is found in the reference design source files as io_status_reg(1).

AddrReg

When the Address Register has been written by either the DSP or the HBA, this bit is set HIGH. This bit is used to indicate the Address Register has been written when an interrupt has been detected by the system monitoring interrupts. This bit is cleared when the Address Register has been read by the target system, or a card reset has been issued.

This bit is found in the reference design source files as io_status_reg(0).

State Machine

The I/O Space interface state machine synchronizes the communication between the DSP and the HBA and relies on the 80MHz clock provided to the CPLD. This state machine consists of seven states, including the IDLE state. Two basic paths in the state machine are available: communication initiated by the DSP and communication initiated by the HBA. Figure 3 displays the flow as described in the following paragraphs.

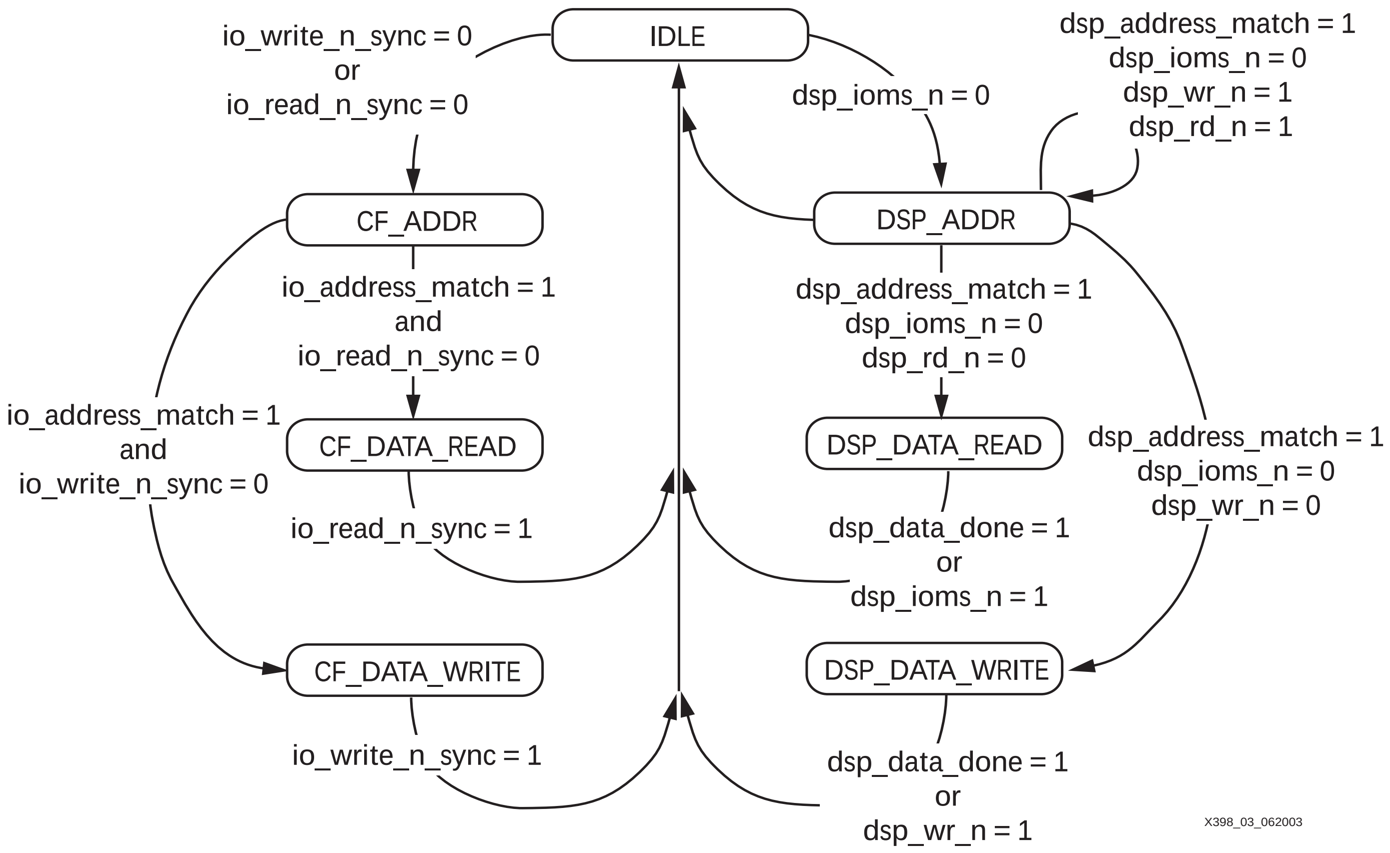


Figure 3: **I/O Space State Machine**

IDLE State

The default state upon reset, is the IDLE state. In this state, the machine waits for either the DSP or the HBA to initiate communications with the I/O Space registers. For a DSP initiated data transfer, the state machine follows the right hand branch shown in Figure 3, whereas an HBA data transfer is contained in the left hand path of Figure 3. The signal dsp_ioms_n represents the ioms_n pin as it is driven by the DSP. The signals io_write_n_sync and io_read_n_sync are the active low write and read request conditions as driven by the HBA. The latter two are synchronized to the state machine using the 80 MHz clock.

DSP Read/Write

When a chip select condition has been detected as found on the ioms_n pin, the state machine transitions to the DSP_ADDR state. During this branch of the machine, no HBA transactions can occur with the I/O Space registers. The DSP_ADDR state waits for a valid address for one of the registers in the I/O space. Once a valid address has been detected, the state machine transitions to the DSP_DATA_READ or DSP_DATA_WRITE state, depending on the status of the dsp_rd_n and dsp_wr_n pins.

The DSP_DATA_READ state allows data from the I/O space registers to be placed on the DSP data bus. Once the data has been transferred to the DSP, the state machine transitions to the IDLE state.

The DSP_DATA_WRITE state permits the DSP to transfer data from the data bus to the register being addressed. Once data has been captured in these registers, the state machine transitions to the IDLE state.

HBA Read/Write

When the HBA issues a request for data from the I/O space, as indicated using the io_write_n_sync and io_read_n_sync signals, and the state machine is in the IDLE state, the machine branches to the CF_ADDR state. It is in this state that the machine waits for a valid address from the HBA to access the I/O space registers. Once a valid address has been detected from the HBA as indicated with io_address_match at a HIGH level, the state machine transitions to either the CF_DATA_READ_STATE or CF_DATA_WRITE_STATE state. The signals io_read_n_sync and io_write_n_sync direct the state machine to the respective state when one of these signals is LOW.

Once in the CF_DATA_READ_STATE state, data is placed from the addressed I/O Space register to the HBA data bus. Once the HBA has received the data and removed the read condition from the CF+ interface, the state machine transitions to the IDLE state.

The CF_DATA_WRITE_STATE state allows data to be written to the addressed I/O Space register from the HBA data bus. After the HBA has written the data and removed the write condition from the CF+ interface, the state machine moves to the IDLE state.

Source Code Download

The VHDL source code and test benches are available for this design. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design has not been verified on hardware (as opposed to simulations), and it should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

XAPP398 - http://www.xilinx.com/products/xaw/coolvhdlq.htm

To simulate the files included in the download, it is necessary to obtain the VHDL model of the Intel StrataFlash 28F320J3 memory from the Intel website. To obtain this VHDL model, visit **http://appzone.intel.com/toolcatalog/listtools.asp?pid=5319&cid=683&pfamily=**

Disclaimer

CompactFlash and CF+ are registered trademarks of CompactFlash Association. Xilinx provides this reference design as one possible implementation of this standard and claims no rights to this interface. To use this reference design, you must contact the CompactFlash Association to obtain any necessary rights associated with this interface.

Conclusion

The CoolRunner-II CPLD is the perfect device to use as an interface to a CompactFlash host. These CPLDs exhibit low power consumption as well as other features that are beneficial to these cards. Features such as 3.3V I/Os, Schmitt Triggers and DualEDGE clocking allow easy integration to this type of system as well as provide flexibility to other functions on a CompactFlash card.

Further Reading

CoolRunner-II Data Sheets, Application Notes, and White Papers

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/23/03	1.0	Initial Xilinx release.

XILINX®

XAPP799 (v1.1) August 2, 2005

An SMBus/I^{2}C-Compatible Port Expander

Summary

Today's microcontrollers and microprocessors often limit the number of General Purpose I/O (GPIO) ports in order to conserve pin count and to reduce package sizes. Unfortunately, for many designs, the number of I/O ports required exceeds the number of available I/O ports on a microprocessor. Hence, Complex Programmable Logic Devices (CPLDs) can often be found in a system alongside a microcontroller functioning as a port expander. This Application Note presents a design of a port expander that fits into a CoolRunner™-II XC2C32A device -- A port expander that is SMBus and I^{2}C compatible.

CoolRunner-II Advantages

A key advantage of using any Xilinx CPLD as a port expander is the integration of logic functions across the entire board. The flexibility of programmable fabric allows for a variety of dissimilar functions to be implemented on one chip. A CPLD can be used as a port expander, an LED driver, an address decoder and a memory controller, all at once.

CoolRunner-II devices add further value being the lowest cost and lowest power CPLDs on the market. These CoolRunner-II parts also provide I/O Banking capability, allowing for level translation between the microcontroller and its external peripherals. Both features are favorable for an SMBus or I^{2}C design, as these environments are typically multi-voltage environments requiring low power.

Table 1 summarizes the level translation abilities of the CoolRunner-II family.

Table 1: **I/O Standards for the CoolRunner-II XC2C32A**

IOSTANDARD Attribute	Output V_{CCIO}	Input V_{CCIO}	Input V_{REF}	Board Termination Voltage V_T
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15(1)	1.5	1.5	N/A	N/A

(1) LVCMOS15 requires the use of Schmitt Trigger

Using the CoolRunner-II SMBus/I^{2}C Port Expander Design

The port expansion reference design shown in Figure 1 provides 8 output ports and 8 input ports controlled through an I^{2}C compatible serial interface. This design is scalable. The code can be modified to increase or decrease the number of desired output ports and input ports. This section describes the default code implementing 8 outputs and 8 inputs. The fully working

project, with source code, is available for download at
http://www.xilinx.com/bvdocs/publications/XAPP799_designfiles.zip.

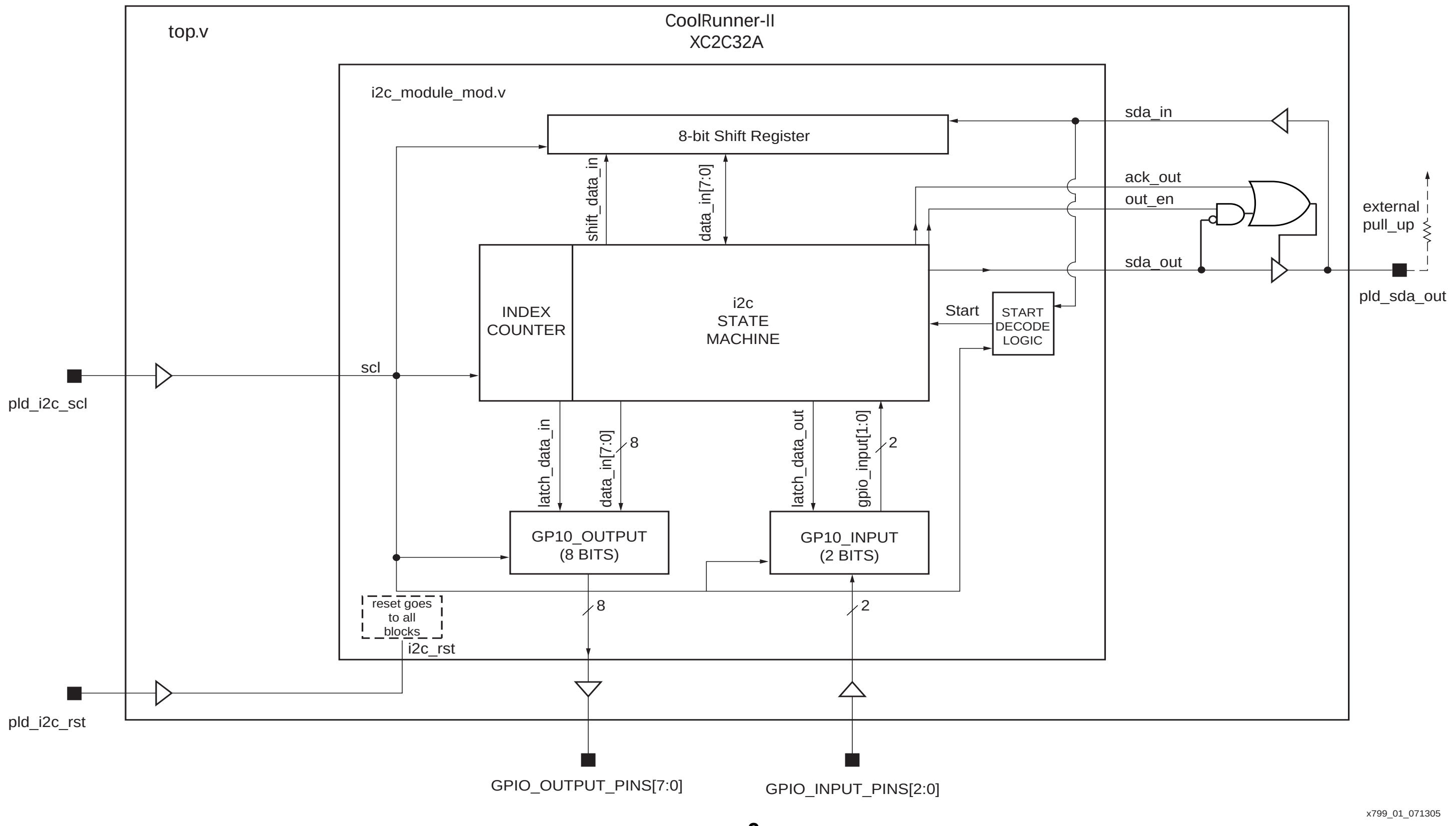


Figure 1: **I^{2}C Port Expander Block Diagram**

Table 2: **Pin Descriptions**

Name	Function
pld_i2c_scl	Serial Clock Line
pld_i2c_sda	Serial Data Line
pld_i2c_rst	Active High Reset
GPIO_Output_Pins	Port Expansion Outputs
GPIO_Input_Pins	Port Expansion Inputs

Serial Interface

This port expander design uses a serial data line (SDA) and a serial clock line (SCL) to achieve bidirectional communication with a master. The master, typically a microcontroller or microprocessor, initiates all data transfers to and from the CPLD. The master is also responsible for generating the SCL clock that synchronizes the data transfer.

Each transaction consists of a start condition sent by a master, followed by the CPLD’s predefined 7-bit slave address plus an $R/\overline{W}$ bit, and one data byte that is either transmitted from the master to the CPLD (for a write operation) or one data byte that is transmitted by the CPLD to the master (for a read operation). An Acknowledge bit is used at the end of each data byte, and data is transferred on every rising clock pulse.

Start Condition

Both SCL and SDA remain high when the interface is inactive. The master signals the start condition by transitioning SDA from high to low while SCL is high (Figure 2). Once a start condition is recognized by the CPLD, an internal ‘start’ pulse is generated, and the state machine will begin.

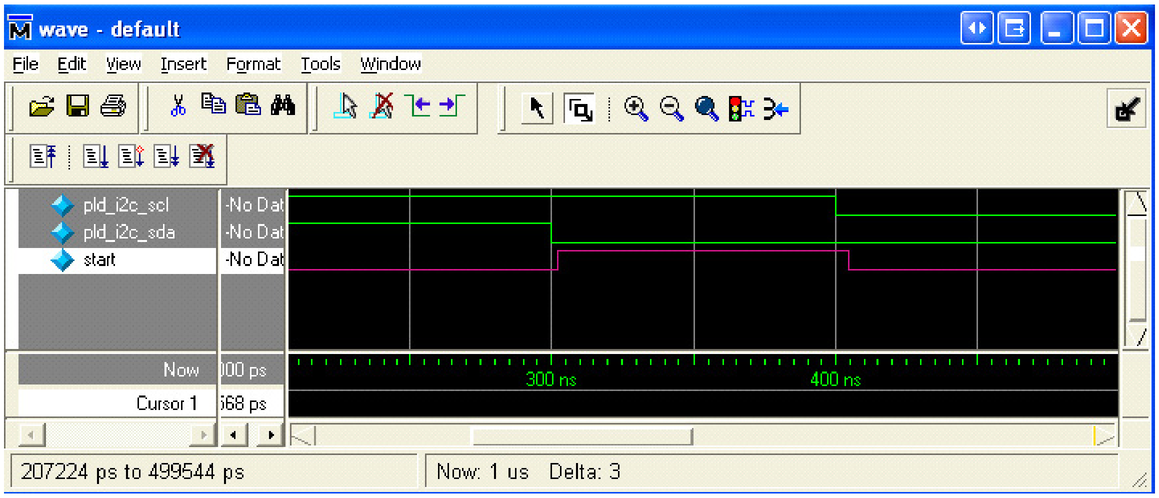


Figure 2: **Start Condition Initiated**

Acknowledge

The acknowledge bit is sent every 9th bit, indicating receipt of each data byte. Each byte transferred effectively requires 9 bits. The master generates the 9th clock pulse, and the recipient pulls down SDA during the acknowledge pulse. This means that when the master is transmitting to the CPLD, the CPLD generates the acknowledge bit. When the CPLD is transmitting to the master, the master generates the acknowledge bit.

I^{2}C Slave Address

The CPLD has a 7-bit long I^{2}C slave address that can be preprogrammed into the device through the source code. Edit the top level Verilog source file (top.v) and define your desired 7-bit binary value for the “my_i2c_addr” parameter. The 8th bit following the 7-bit slave address is the R/$\overline{\text{W}}$ bit. Set this bit low for a write command and high for a read command.

Performing a Write Command

After a slave address with a write command has been sent, one final data byte must be sent from the master to complete the transaction. This data byte defines the state of each of the 8 I/O’s. The MSB of the data byte defines the value on ‘GPIO_OUTPUT_PINS[7]’ and the LSB of the data byte defines the value on ‘GPIO_OUTPUT_PINS[0]’. The outputs assume their defined values during the 9th clock cycle, together with the acknowledge pulse.

Figure 3 shows a full Write Command transaction. The master initiates a start command, then sends a write request to the CPLD, located at slave address 0x56. The CPLD acknowledges the write request on the 9th clock edge, and the master continues delivering data to the CPLD dictating the values on the GPIO output ports. On the 9th clock cycle, the CPLD acknowledges this second data byte, and sets the GPIO output pins accordingly.

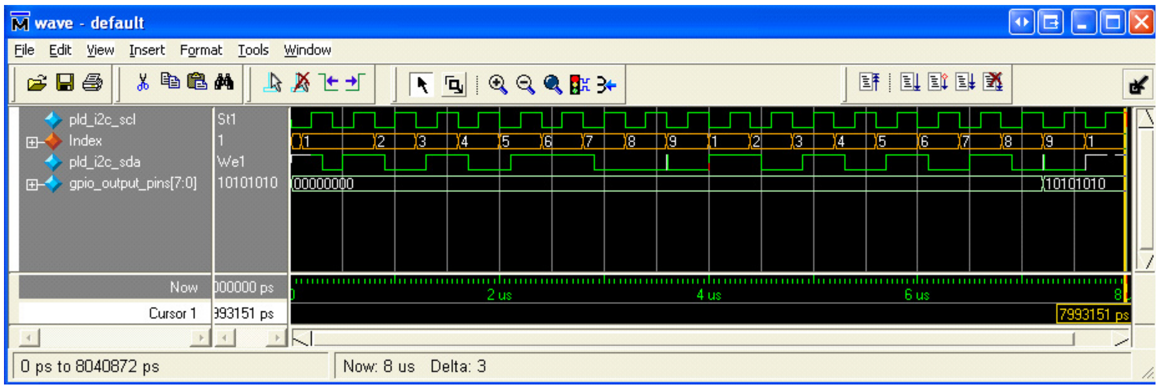


Figure 3: **Write Command Transaction**

Performing a Read Command

If a slave address with a read command has been sent, one final data byte is sent from the CPLD to the master. The data byte relays the values present on the 'GPIO_INPUT_PINS' bus.

Figure 4 shows a full Read Command transaction. The master initiates a start command, then sends a read request to the CPLD located at slave address 0x56. The CPLD acknowledges the read request on the 9th clock edge. On the next data byte, the CPLD returns the value on the GPIO input pins. In this example, since there are only 2 GPIO input pins in the design, only the data on the 7th and 8th clock are valid. The first 6 data bits are high by default, and are ignored by the master. On the 9th clock cycle, the master acknowledges this second data byte, and the read transaction completes.

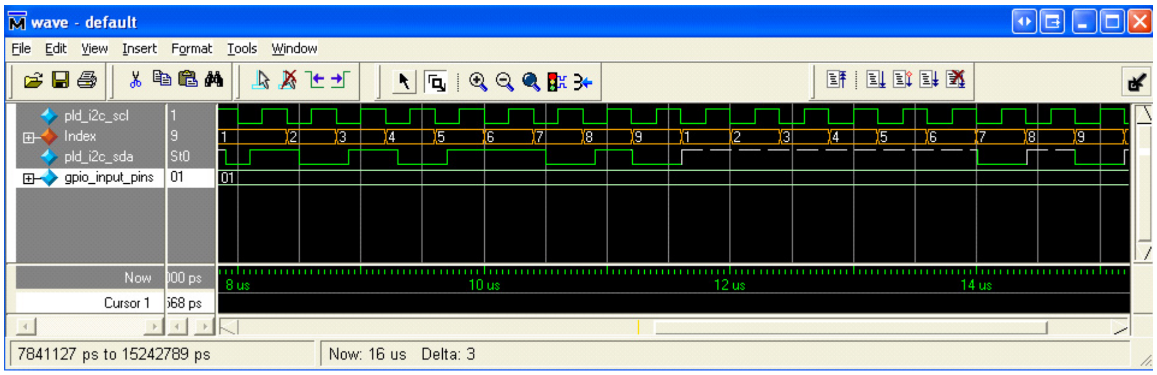


Figure 4: **Read Command Transaction**

Standby

The CoolRunner-II family of CPLDs automatically enters "standby" when all pins are set to Vcc or GND. Standby current is specified in the individual device data sheets. This design fits into an XC2C32A device, which has a typical standby number of 22 uA -- ideal for SMBus and I^{2}C applications.

Utilization

The following are the utilization statistics:

Table 3: **Device Utilization**

Macrocells Used/Total	Product Terms Used/Total	Function Block Inputs Used/Total	Registers Used/Total	Pins Used/Total
32/32 (100%)	71/112 (63%)	36/80 (45%)	32/32 (100%)	13/33 (39%)

Customizing the Design

This reference design can be customized to fit any specific design target. If the design requires more GPIOs, the state machine in the source code can be modified accordingly. Should bidirectional I/O's be required, the design can be modified to use the $R/\overline{W}$ bit as a signal to decode whether the I/O pin should function as an input or output. There are many possibilities, and the source code has been written for the most general case.

Conclusion

This port expander design can be used as a complete turnkey design that fits readily into a CoolRunner-II XC2C32A device. You can easily edit the port expansion module to fit your needs, and additional functionality can be integrated into the CPLD. The CoolRunner-II family of CPLDs is unique in that it is the only low cost, easy to use, low power device available today.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/19/05	1.0	Initial Xilinx release.
08/02/05	1.1	Fixed broken link to design files.

XAPP905 (v1.0) August 25, 2005

Using CoolRunner-II with OMAP, XScale, i.MX & Other Chipsets

Introduction

Getting it Right Every Time

Consumer electronics product designs, such as cell phone handsets and MP3 players, typically are very high volume products. To that end, most product designers choose ASIC or ASSP methodologies to pack the greatest functionality into tiny, portable packages. This “hits the mark” for dense functionality, and usually has acceptable power consumption. But the consumer world is rapidly changing, so features envisioned at one point in time become quickly obsolete, as competitors must deliver differentiated solutions to seek greater market success. The term cutthroat is frequently used to describe this level of competition! Mistakes are not tolerated, and mistakes are expensive. Choosing the correct ASSP, or designing an ASIC correctly every time is nearly impossible. Mitigating these circumstances is crucial to sustaining market share, and that is where CoolRunner™-II CPLDs enter the picture.

CoolRunner-II CPLDs are the leading, low power, low price programmable solution for digital consumer designs today.

This discussion will show you ways to expand beyond the limitations of today’s ASIC/ASSP solutions, with simple, cost effective, low power programmable logic using CoolRunner-II CPLDs. As well, we will show solutions to some of the problems mentioned here, with links to application notes that give in-depth details.

Level Translation

Interfacing two chips of different voltage standards is a common problem. Every type of memory is not made at every voltage standard, and microprocessors are offered at many voltages. Matching standards can be as simple as introducing level translators, but they are expensive and take more area than might be desired. Using a CPLD is a better solution, and offers substantially greater flexibility. All Xilinx CoolRunner-II CPLDs are capable of translating between two voltages, and some can handle as many as four.

CoolRunner-II CPLD I/O banks easily translate between voltages ranging from 1.5 to 3.6V, in a single chip. But, this totally disregards the programmability of the devices. You get the translation as part of the whole package, which means you get a bundle of logic, flip flops, power reduction resources, and I/O buffers frequently priced below level translator chips! XAPP785 explains the details on how you can take advantage of this powerful feature, to expand the capabilities of your OMAP, XScale or i.MX designs.

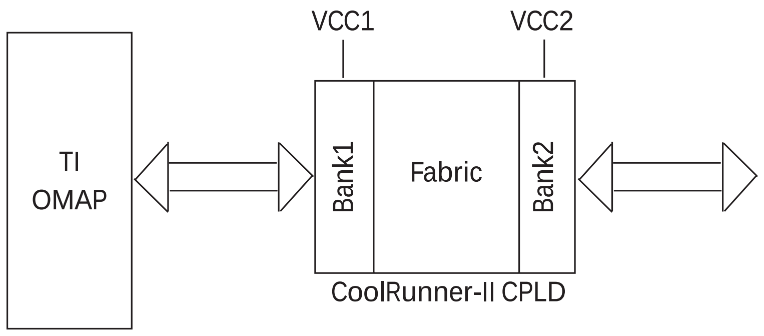


Figure 1: **CoolRunner-II Level Translation of TI OMAP Signals**

Pin Expansion

High pin count ASICs are more expensive than low pin count ASICs, in general. If your logic needs dictate a low capacity, but your I/O requirements dictate a high capacity, you may be paying for logic you will never use, to gain the pins. One solution to this is adding a CoolRunner-II CPLD to operate as a "pin expander". The basic idea is to identify GPIO pins that typically operate at a slow speed. Then, rather than assign ASIC pins to them, attach CoolRunner-II CPLD pins to the slow moving GPIO signals, serialize the signals and import them to the ASIC on fewer net pins.

Serializing/deserializing is done through simple, efficient shifting, and can drop the pin counts dramatically on expensive ASICs. XAPP799 shows how to do this through an I^2C port, but other methods can be used.

As an alternate viewpoint, OMAP, XScale and i.MX processors provide specific pin mixes to support the applications their vendors deem appropriate. This doesn't mean that you must agree, as a designer. CoolRunner-II pin expansion permits you to create your own GPIO pins, of assorted voltages and additional capabilities (pulsing, PWM, individually 3-stated). Increasing the effectiveness of your solution is our goal.

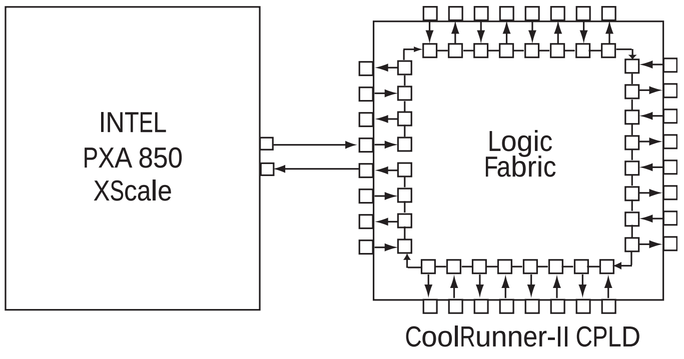


Figure 2: **CoolRunner-II Pin Expansion of XScale Processor**

Pin Swizzling

CPLDs offer the ability to rearrange your pinouts when PCB layout errors occur. This valuable quality is key to keeping you on schedule and within financial and power budgets.

Correcting misconnections on a board, without having to re-spin the PCB can shave weeks to months out of product schedules. CoolRunner-II CPLDs are built from powerful logic blocks using Programmable Logic Arrays, that can reassign pin logic "at will." You will be amazed at how well these devices retain pinouts through multiple edits, yet permit re-assigning a design onto different pins as needed. The CoolRunner-II Family data sheet explains the architecture and points you to application notes that give all the detail you will need to understand the value of PLAs.

Power Control

Quick power up is one of the strengths of CPLDs. Containing their own configuration cells permits CoolRunner-II CPLDs to power up and direct the activities of other chips, as they subsequently arise. This includes some power regulators, which may be sequenced by the Coolrunner-II CPLDs, as well as other controlling signals that need to be well defined early in board operation. XAPP436 describes some of these capabilities.

Power Reduction

XScale, OMAP and i.MX based chipsets all include some version of the ARM microprocessor. This is not a surprise. Advanced RISC Machines started early with developing low power methods to operate microprocessors, and subsequently, the licensing vendors have all added their own methods to further reduce processor power. Typical power reduction operations are: clock gating, voltage throttling and on board memory management to reduce transfers within the device. These are sometimes referred to as run, wait, doze, sleep, hibernate, and so on. Also, operating systems like Symbian have added "power awareness" to the mix, so that unused resources can be parked in the lowest power mode possible for the current tasks being executed. This all works well and lowers processor power. However, lowering power in the rest of the system exceeds the scope of these methods. Enter CoolRunner-II CPLDs.

CoolRunner-II CPLDs are designed to be inherently low power parts. That is important, but alone is not enough. CoolRunner-II special features also can be used to lower the power in other devices. Using clock dividers and Xilinx's patented DataGATE^{TM} technology can reduce power in many (if not all) of the chips on your design. XAPP378 shows how to do this, and WP227 shows just how much power you can save with DataGATE. Blocking power to other chips can also reduce electromagnetic fields being propagated on your board and emanating from your system. This powerful signal blocking technique can pay off in many ways!

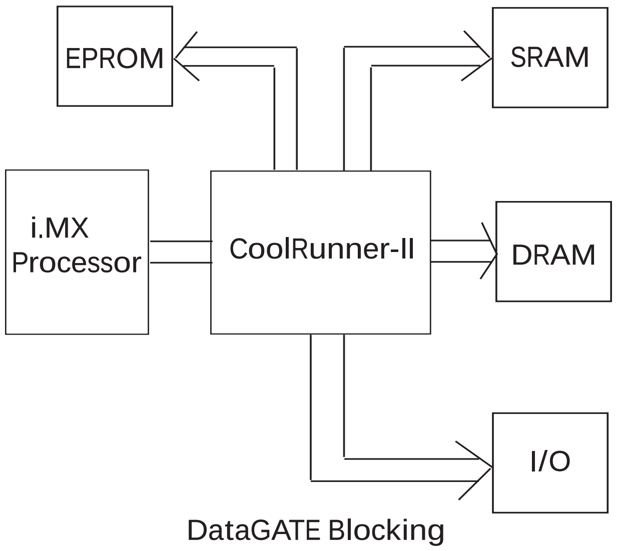


Figure 3: **DataGATE Blocking Extraneous Switching to Various Devices**

Logic Consolidation

Having three, two input AND gates, two three input OR gates and a Schmitt buffer package on your board can burden your bill of materials (BOM), eat away at your power and cost budgets, and lower your reliability. Collecting all that stray logic into a consolidated, low power CoolRunner-II not only solves these problems, but stores additional unused logic right there, on your board – ready to use with future improvements/edits. WP214 shows what you can expect from collecting logic gates/flip flops into CoolRunner-II CPLDs. Table 1 summarizes the "burn rate" for logic.

Table 1: **Macrocell "Burn Rate" for Common TTL Functions**

Function	Macrocells	P-terms	Flip-Flops
Shift register (simple)	1 per bit	1 per bit	1 per bit
Counter (simple)	1 per bit	1 per bit	1 per bit
2:1 Mux	1	2	0
4:1 Mux	1	4	0
8:1 Mux	1	8	0
8 bit loadable shifter	8	16	8
8 bit loadable/SL/SR shifter	8	24	8
8 bit loadable counter	8	16	8
8 bit load/up/dn counter	8	24	8

Table 1: **Macrocell "Burn Rate" for Common TTL Functions**

Function	Macrocells	P-terms	Flip-Flops
Full Adder / bit	2	7	0/1 (optional)
2:4 Decoder	4	4	0
3:8 Decoder	8	8	0
4:16 Decoder	16	16	0
8 bit Equality Comparator	1	16	0
And/Nand gate (1-40 inputs)	1	1	0
Or/Nor gate (1-40 inputs)	1	11	0
Ex-or/Ex-nor (2-3 inputs)	1	2-3	0
Level translator (per bit)	1	1	0

Additional Information

CoolRunner-II Data Sheets and Application Notes

Conclusion - The Future

CoolRunner-II CPLDs are quickly becoming the standard for low power, low cost, high volume, handheld consumer products. This application note has focused on how these powerful products can make life easier when building systems with OMAP, XScale and i.MX processors, but CoolRunner-II works just as well with many other processors, to add functionality, save power and get products to market fast.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/25/05	1.0	Initial Xilinx release.

XILINX®

XAPP430 (v1.0) July 8, 2003

Cell Phone Security Demoboard

On The Fly Reconfiguration Technique

Summary

This document describes the operation of the cell phone demo board, and provides a simple example of OTF operability of CoolRunner™-II CPLDs.

Introduction

This demo board was created specifically to demonstrate the unique ability of the CoolRunner-II device's capability of performing On The Fly (OTF) operations. This particular application illustrates a cell phone that utilizes a SIM card identification system.

In some parts of the world, cell phones are mated with SIM cards for operation. The SIM card contains the account information for the user. A user may insert their card into any phone to make a call. While this allows for a high degree of flexibility in terms of ease of hardware interchangeability, it also opens the cell phone up to a high risk in terms of theft. Unfortunately, an intelligent, resourceful and dedicated thief typically is successful. However, if the cost of stealing and modifying the phone is increased such that it is more expensive than just purchasing a phone outright, the economics of the theft should result in a reduction of this particular crime.

This demo board is not meant to be a solution to cell phone theft. It is a proof of concept to illustrate to engineers how the advanced features of the CoolRunner-II CPLDs might assist in cell phone security. Obviously additional techniques would need to be used in conjunction with this demo, such as package selection (ball grid), mounting techniques, PCB layout (such as burying JTAG lines), or even "chip on board" technology. The cell phone security issue is a complex one, and this demo illustrates how Xilinx CPLDs can add some margin of flexibility to the solution.

Demonstration Overview

Important: Read all instructions carefully on the following pages before use in order to avoid damaging the cell phone.

The demo board comes with two small 'SIM' cards that can be inserted into the reverse side of the 'phone'.

1) Program the CPLD using Impact and a download cable.

2) Insert one of the SIM cards.

3) The front display will either show a circular 'chasing' pattern, or will scroll out CodE? The chasing pattern is indicative of the normal operation of the phone. This is just a simple 'I am alive' pattern, indicating that the SIM card that is in the phone is mated to the phone.

4) If the front display shows CodE? Enter in the user code, which is 5,7,9,3. It will ask for the code again. Enter in 5,7,9,3. This will program the onboard EE to accept the new SIM card. The phone should now show the chasing pattern on the LCD.

5) Once the phone has been reprogrammed to accept the new SIM, put in the other SIM, so that the CodE? Starts again. Enter in a four digit code that is not the master code. Enter this code in again. The display should turn off. At this point the CPLD has erased itself and the

phone is useless. Attach the phone to the Impact tool, and perform a 'Blank Check' to illustrate that the phone is completely dead.

6) If two dissimilar codes are entered, such as 2345, and then 2346, the phone will keep requesting CodE? Until two matching codes are entered. It will then either accept the new SIM (if the codes were both 5,7,9,3) or erase the CPLD if the codes were something else.

Setup

Batteries

The phone runs on two lithium coin cell batteries. Any 20mm 3V cell will work. I have run about 30 demos so far on one set of batteries, and they are still working fine, but keep some spares on hand, as the lithium cells do droop fairly quickly. Important! The phone does not have reverse polarity protection. Make sure that the batteries are stacked so that the + side of the batteries are 'up'. Refer to Figure 1 for an illustration.

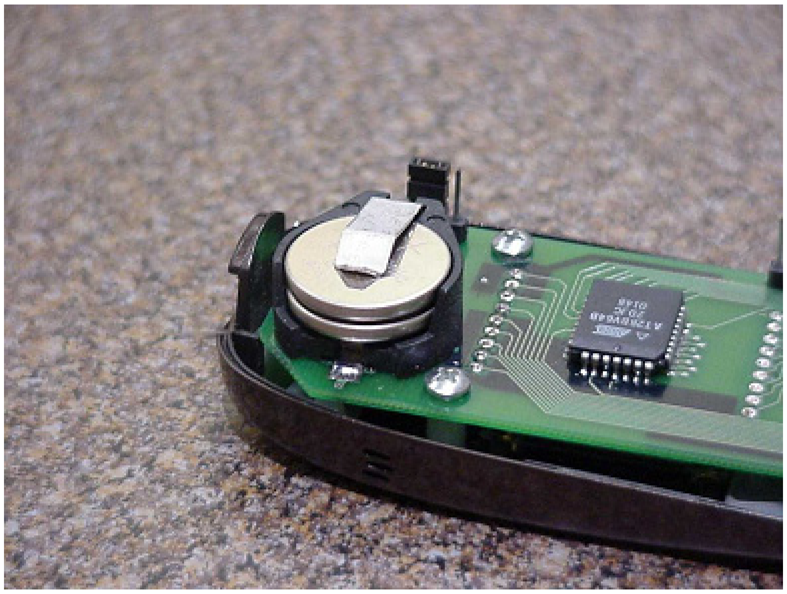


Figure 1: **Coin Cell Holder**

On / Off:

Turning on power is accomplished by moving the jumper next to the battery to the lower two pins. The upper two pins provide an 'off' position jumper holder. This below picture indicates the phone is in the 'off' position. Refer to Figure 2.

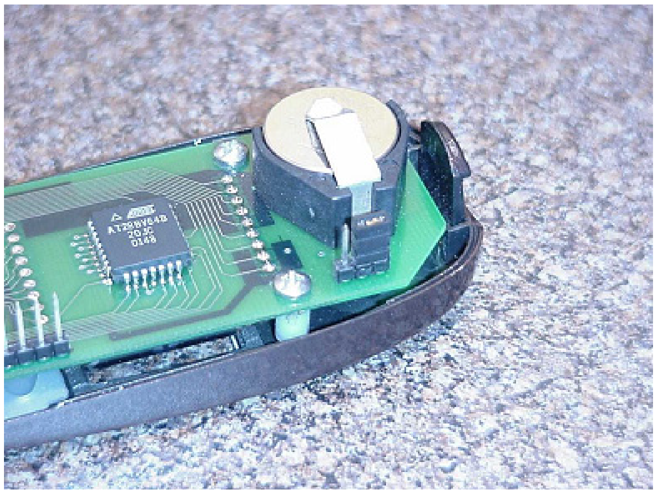


Figure 2: **Power Jumper**

JTAG Cable:

The board is not marked for the JTAG cable. The header on the board is in the order of the flying lead cable that is provided with the Xilinx download cables. In order from the top of the phone: VCC, GND, TCK, TDO, TDI, TMS. See the below picture, Figure 3, for details.

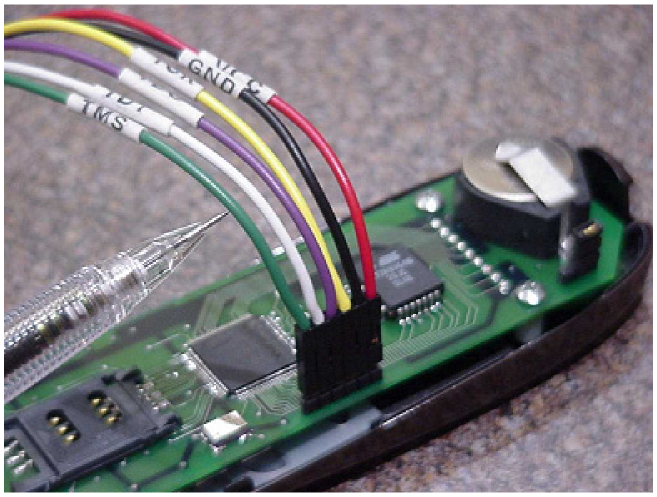


Figure 3: **JTAG Cable connections**

Caution

The mounting of the board to the faceplate is fragile. When removing the leads from the board, make sure that you hold onto the PCB and not the faceplate when disconnecting the cable. Refer to Figure 4.

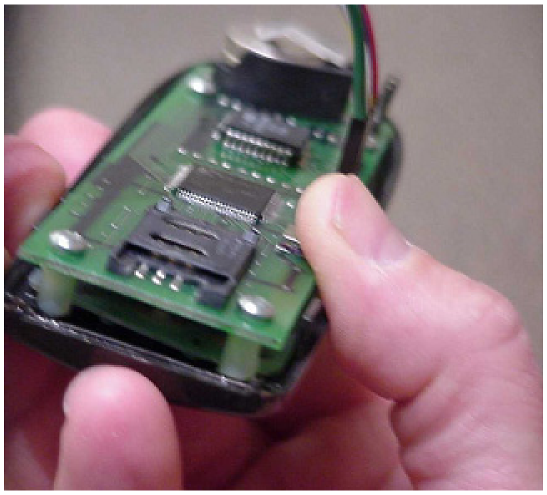


Figure 4: **Care When Disconnecting JTAG**

Insert Sim Card:

The SIM card slides in from below the phone into the socket. Copper side down. Use extreme care when inserting the card! This is not the normal operating mode of this type of socket, and it may be damaged easily. When inserting the card, make an attempt to keep the card going in 'square' with the socket, and do not over insert. Refer to the below pictures for additional detail.

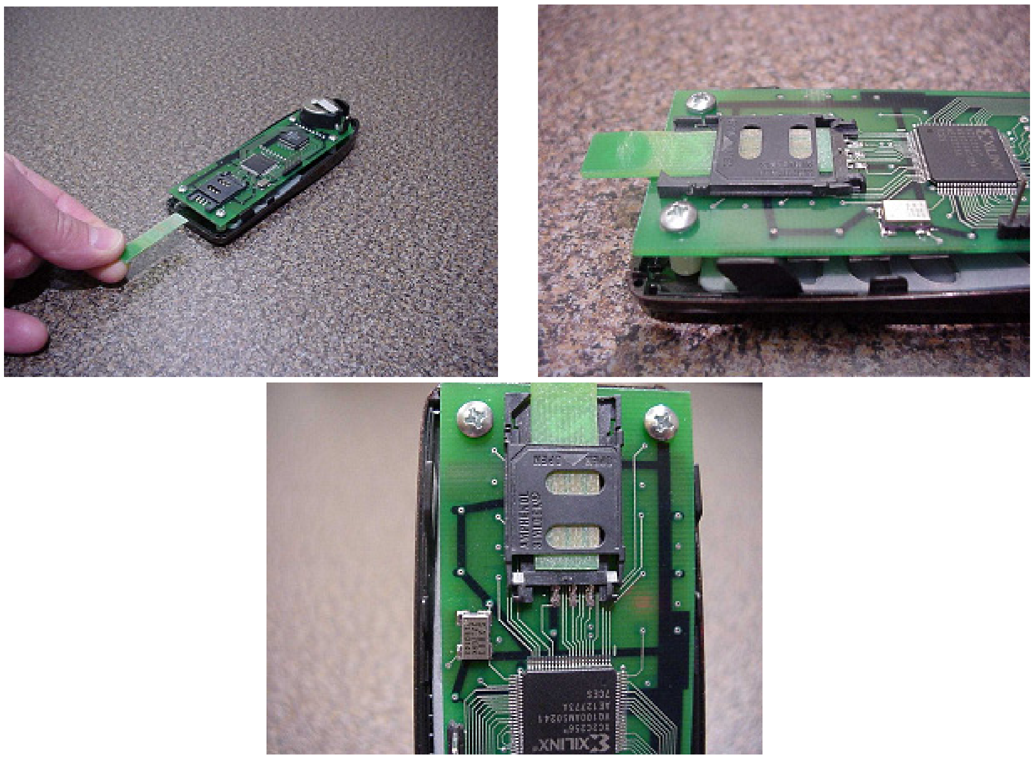


Figure 5: **SIM Card Insertion**

Operation:

Initial Operation

A mated SIM card inserted into a programmed phone will result in a 'chasing' pattern on the LCD. 'Mated' means that the phone has been programmed to accept that particular SIM card. The demo comes with two independent cards with unique addresses. The chasing pattern indicates that the phone is in its normal operating condition. Refer to Figure 6

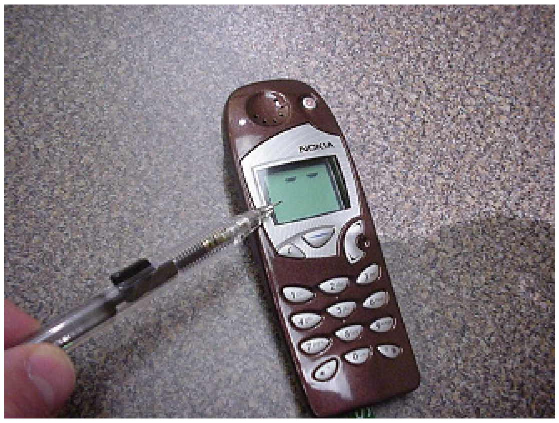


Figure 6: **Initial 'Chasing' Pattern**

A foreign (non-mated) SIM card will result in the phone requesting a code and then the confirmation of that code to be entered as shown in Figure 7.

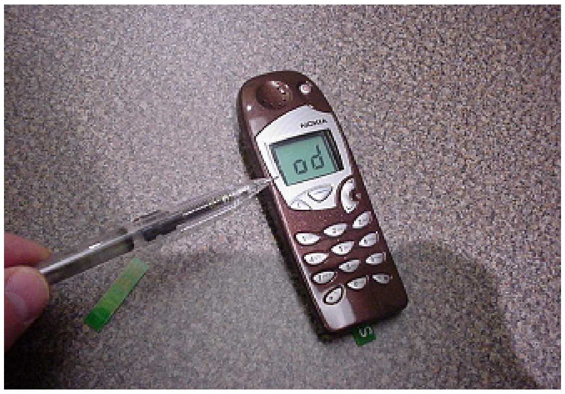


Figure 7: **Invalid SIM Requires User Code**

Enter User Code

Entering in the appropriate code (5,7,9,3) and then a confirmation of that code will result in the new SIM card being mated to the phone.

Figure 8: **Entering User Code**

Miscellaneous Details

The only buttons that have impact on the operation of the phone are the number keys and the * buttons. The asterisk is a system reset button and may be pushed if the phone exhibits unusual behavior.

Do not push in any of the other buttons as it may result in the rubber keypad becoming dislodged.

Always put the power jumper in the 'off' position when not being used.

Design Information

Demonstration State Diagram

This demonstration utilizes a few simple state machines that can be represented by a top level diagram as seen in Figure 9.

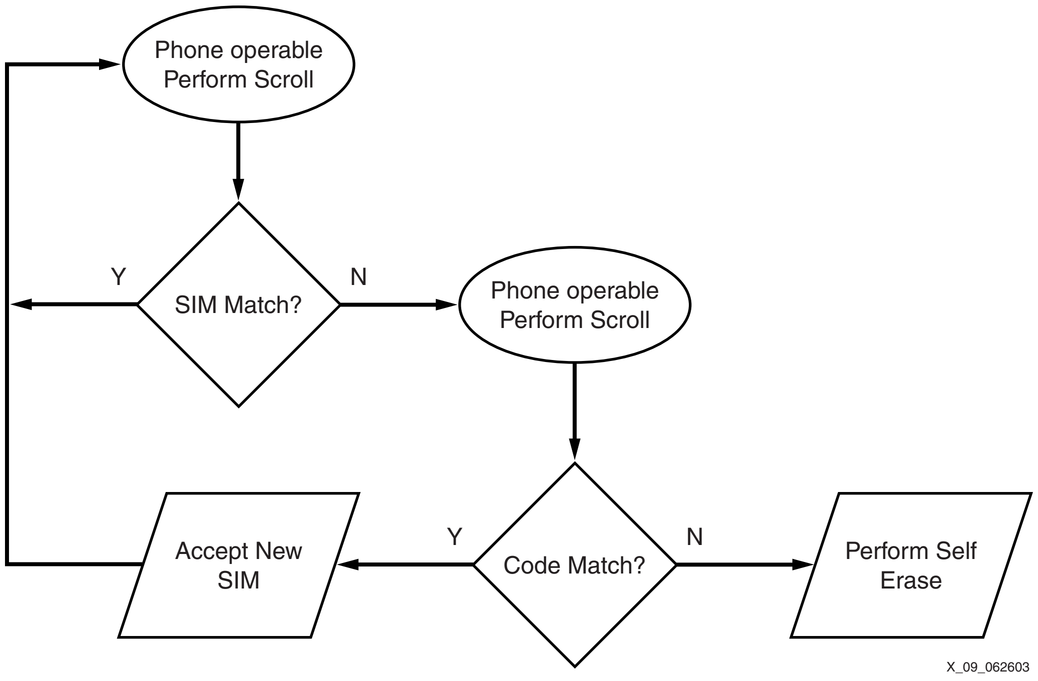


Figure 9: **Demonstration State Machine**

The phone idles in a small loop that operates the display in a chasing pattern. Once every loop cycle, it accesses the SIM card address and compares it to an onboard EE memory device to ensure that the correct card is inserted. As long as the phone recognizes the card, it stays in this normal operation loop.

If the card is not recognized, the phone breaks from the idle loop and queries the user with a user code request. The phone reads the keyboard for 4 key presses, and stores this value. It then queries for a confirmation of the user code, and compares it to the first 4 digits. If the user codes match, then the entered user code is compared against the known user code stored in memory. If the user codes mismatch, then the user is asked to re-enter the user codes again.

Upon receipt of two matching user codes, the phone compares this value to an internal stored user code value. If the user codes match the internal value, the new SIM card is programmed into EE and the phone enters the normal operation idle loop.

If the user codes do not match the master user code, then the phone enters into a state machine that performs an OTF erase of the CPLD.

OTF Erase Capability

The erase capability of the CPLD is facilitated through the capability of the CPLD to do On The Fly (OTF) operations. This means that the CPLD can have its non-volatile memory reprogrammed while it is fully functional with a different pattern. This is done by issuing an 'Enable OTF' command early in the JTAG sequence.

The erase algorithm itself is very easy to implement. In order to do any JTAG operation, TDI and TMS are assigned and then a rising edge on TCK is given. A single thread state machine was implemented to step through the different TDI / TMS pairs and present a TCK at the appropriate time. Additionally, at certain points during the program / erase operations, a delay is required for cell discharge and erase burn time. The state machine also accesses a timer for delay at the needed times.

Refer to Figure 1 for detailed instructions on OTF Erase.

Table 1: **OTF ERase JTAG Sequence**

Step	Transition Conditions	TAP State	Event Description	Programmer Action
0	TMS = 1	Test-Logic/Reset	Begin	Begin

Loop 0	TMS = 1
	TCK = rise
		Test-Logic/Reset	Ensure device in Test Logic Reset State	Loop 5 times

1	TMS = 0
	TCK = rise
		Run-Test/Idle		

2	TMS = 1
	TCK = rise
		Select DR-Scan		

3	TMS = 1
	TCK = rise
		Select IR Scan		

4	TMS = 0
	TCK = rise
		Capture IR		

5	TMS = 0
	TCK = rise
		Shift IR		

Loop1	TMS = 0
	TCK = rise
		Shift IR	Shift in Instruction Bits (0-6)	TDI = 0010011 (Enable OTF)

Table 1: **OTF ERase JTAG Sequence *(Continued)***

Step	Transition Conditions	TAP State	Event Description	Programmer Action

6	TMS = 1
	TCK = rise
		Exit1-IR	Shift in Instruction Bit 7	TDI = 1
				Enable_OTF
				(MSB)

7	TMS = 1
	TCK = rise
		Update-IR	Load the Instruction Register, Set enable flip flop	

8	TMS = 1
	TCK = rise
		Select DR-Scan		

9	TMS = 1
	TCK = rise
		Select IR-Scan		

10	TMS = 0
	TCK = rise
		Capture-IR		

11	TMS = 0
	TCK = rise
		Shift-IR		

Loop2	TMS = 0
	TCK = rise
		Shift-IR	Shift in Instruction Bits (0-6)	TDI = 1011011
				(Erase)

12	TMS = 1
	TCK = rise
		Exit1-IR	Shift in Instruction Bit 7	TDI = 1
				Erase (MSB)

13	TMS = 0
	TCK = rise
		Update-IR	Load the instruction Register	

14	TMS = 0
	TCK = rise
		Run-Test Idle	Execute: Erase the Device	Loop here for 100ms

15	TMS = 1
	TCK = rise
		Select DR-Scan		

16	TMS = 1
	TCK = rise
		Select IR-Scan		

17	TMS = 0
	TCK = rise
		Capture-IR		

18	TMS = 0
	TCK = rise
		Shift-IR		

Conclusion

This demoboard provides an illustration of the power of OTF operations by applying simple OTF reconfiguration techniques in a Cell Phone security demonstration. Requiring only 27 macrocells to implement the self erase algorithm portion of the design, this function will fit into any Xilinx CoolRunner-II CPLD

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/08/03	1.0	Initial Xilinx release

Chapter 2

Low Power Design

CoolRunner-™II is the ultimate in low power consumption and low power control for handset designs. CoolRunner-II is an inherently ultra low power device, and has advanced features to further lower the power use of your design. This section documents how to get the longest battery life out of your handset by using the advanced low power features of CoolRunner-II, as well as demonstrating how to use the CPLD as a low power controller for the rest of your design.

This Chapter contains the following topics:

- Low Power Design
- Advanced Features
- Using DataGATE
- The Real Value of DataGATE
- Managing Power

XAPP377 (v1.0) May 8, 2002

Low Power Design with CoolRunner-II CPLDs

Summary

CoolRunner™-II CPLDs are the only CPLD to combine both high performance and low power to form the next generation CPLD. This application note describes the design methodologies that can be employed to obtain the lowest power possible using the CoolRunner-II CPLD by utilizing its unique power saving features.

Introduction

CoolRunner-II CPLDs continue to employ the features of Fast Zero Power™ (FZP) technology as found in the earlier CoolRunner CPLD generations. But due to unique Xilinx design techniques, smaller geometries, and state of the art process technology, FZP technology has further advanced CoolRunner-II CPLDs as the low power standard. Other CPLD manufacturers have attempted to reproduce the FZP true CMOS technology of the CoolRunner CPLDs, but have not been able to meet the CoolRunner benchmark. For the first time in the CPLD industry, CoolRunner-II devices deliver both true high performance and low power at the same time, along with the lowest standby current in the industry without the use of power down modes. In addition, these devices offer unique power saving features such as CoolCLOCK, DataGATE, and DualEDGE flip-flops.

Traditional CPLDs use sense amp type product terms to provide fast propagation delay times. Sense amp product terms are biased in a manner to detect a small change in voltage levels on the word line which indicates a change in the logic state of the product term. The transistor biasing constantly draws current, even at standby. With this in mind, these types of CPLDs cannot provide a low power solution, as can CoolRunner-II CPLDs. As sense amp type device sizes become larger in macrocell count, power grows significantly since there are many more product terms to consume power.

As process technology shrinks, sense amp type CPLDs will inherently consume more power to maintain performance. Smaller process technology demands lower power supply voltages thereby reducing the gain of the sense amp. Further, product term transistors leak more with the smaller geometries. To maintain performance, a sense amp CPLD will need to be designed such that its biasing compensates for the leakage and boosts gain to detect the smaller voltage swing of the word line. Higher biasing will cause more current to flow through the bias network, thereby increasing total power consumption. Other CPLD manufacturers that use sense amp based technology will inevitably go through a learning process to re-design their current products to compensate for the effect of ever shrinking process technology.

CMOS product terms, as used in the FZP technology, inherently consume less power when not switching states. Since CMOS logic exhibits low standby current, CoolRunner-II CPLDs use this technology to reap the benefits of low power. Additionally, CMOS technology benefits from smaller geometries where the device consumes less power and becomes faster.

Power Saving Features

New architectural features have been added to the CoolRunner-II product line to enhance the power saving capabilities of the FZP technology. This section describes these features and how they can be used to save power.

All new features described below are available with the XC2C128 and larger devices. The smaller devices, XC2C32 and XC2C64, do not include some features, specifically DataGATE, Clock Divider, and CoolCLOCK.

DataGATE

Many times devices are connected to a data bus which are not being addressed by the master device. When a CPLD is in this situation, it will use more power than necessary since the data on the bus is not useful to the CPLD, but the data lines continue to toggle, which subsequently toggles the internal logic of the CPLD. Any logic that changes state within the CPLD will consume power. Therefore, it follows that disconnecting the CPLD from the data bus when the device is not addressed conserves power, as highlighted in this example.

DataGATE solves this issue by disconnecting external signal activity from the internal logic, consequently reducing power consumption of the device. This is achieved by using a unique, software enabled, CoolRunner-II pass gate at the I/O pin (when configured as an input) which is controlled by the DataGATE global net. This control net originates from a specific pin/macrocell and can be driven by either an external signal or an internally developed signal using logic elements. When the pass gate on the input pin is disabled by the DataGATE control net, an internal latch drives the CPLD logic network maintaining the same logic level that was present on the input pin just prior to asserting the DataGATE control net. This preserves the current logic state internal to the CPLD while external data changes states.

For example, a CoolRunner-II CPLD that shares a data bus with other devices will most likely have its own address and typically will not be addressed continuously. Two options exist to disconnect the data bus from the CoolRunner-II CPLD when not addressed. First, if the device is addressed using a chip select signal, this signal can be assigned to the DataGATE control pin and used to isolate the inputs from the data bus. Second, if using an address bus, the address of the device can be decoded internally to the CPLD which can then be used to enable, via DataGATE, the data bus inputs to receive data when addressed. When the CPLD is not addressed, DataGATE would disconnect the external data bus signals using address decoding internally to the CPLD. In this case, the result of the internal decoding is routed to the DataGATE pin to disconnect the toggling bus.

DataGATE can be configured to affect any or all I/O pins, with the exception of JTAG pins and the DataGATE pin itself. The above example discusses DataGATE configured to affect a few macrocell I/O pins. It may be beneficial in other applications to disconnect the system clock or clocks from the CPLD. The two elements that consume the most power are output buffers and the clock tree. If the CPLD is not needed for a specific amount of time, the DataGATE feature could be used to gate the clock. Doing so will dramatically reduce power consumption. However, caution must be exercised when gating a clock since undesired logic transitions may occur.

There are other cases where the unique CoolRunner-II DataGATE feature is useful to reduce power. Also, DataGATE is flexible such that the entire device or only parts of the device can be isolated from external signals, depending on the application. For example, it can be used in CoolRunner-II CPLDs to enhance board troubleshooting procedures.

Schmitt Trigger

CoolRunner-II Schmitt trigger inputs are useful for applications that require hysteresis on the inputs. A useful application might be where noise is an issue on specific pins. Hysteresis provides added noise immunity. Another application could implement an oscillator circuit which is constructed external to the CPLD, but CPLD logic is used to implement portions of the oscillator circuit.

To ensure the lowest power consumption in the CoolRunner-II CPLDs, disable the Schmitt trigger inputs since these types of buffers consume more power than the regular input buffer. It is important to design and operate the system in a low noise environment when Schmitt triggers are disabled to prevent inadvertent transitions on the CPLD inputs.

Clock Divider

Global clock networks tend to be the largest power consuming elements in CPLDs. Any effort to reduce the frequency of the global clock network greatly benefits the system with respect to power consumption. Therefore, CoolRunner-II devices have been designed to include a clock divider network on global clock, GCK2. Without introducing additional clock delays, the clock divider has the capability of dividing the system clock by even integers ranging from 2 to 16, as shown in Figure 1.

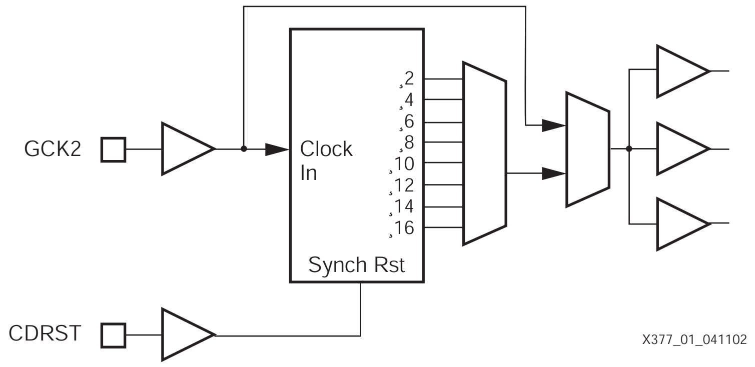


Figure 1: **Clock Division Circuitry for GCK2**

Some systems use state machines, for example, that do not require the full speed of the external system clock. A clock divider is the perfect tool to reduce system power in this case. The clock divider provides an excellent alternative to adding a user defined clock divider built from logic, which would waste logic otherwise usable for more features in the design. A lower frequency on the global clock network, provided by the clock divider, will reduce the power consumed by the CoolRunner-II CPLD.

Generally speaking, designing with the slowest system clock possible will reduce power consumption. To this end, the clock divider provided in the CoolRunner-II architecture will greatly assist the designer.

DualEDGE Registers

By utilizing both edges of the clock signal, the macrocell can do twice the work when configured as a DualEDGE flip-flop. Figure 2 displays the macrocell configured as a DualEDGE flip-flop. A system without the aid of the DualEDGE flip-flop would need to provide a clock at twice the frequency to obtain the same work output at the macrocell. Since the macrocells with DualEDGE flip-flops operate on both the rising and falling edges of the clock, the clock network is used more efficiently. Consequently, power consumption is reduced when the global clock net is operating at a lower frequency.

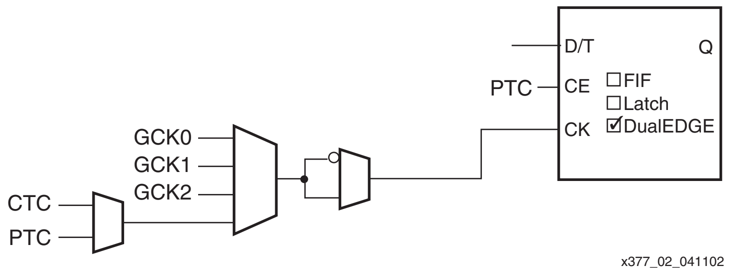


Figure 2: **Macrocell Clock Chain with DualEDGE Option Shown**

DualEDGE flip-flops further enhance the functional possibilities of the clock divider and therefore improve power savings. The global clock can be effectively divided by odd integers of 3, 5, and 7 if used with the DualEDGE flip-flop. For example, if a divide by 3 clock is desired for the design, the Clock Divider can be set to a divide by 6 and then effectively doubled by the DualEDGE flip-flop resulting in a divide by 3 characteristic. This is important to note since, again, lower clock frequencies always save power. The DualEDGE flip-flop effectively adds more functionality to the clock divider network.

Perhaps the design contains two state machines. One state machine can efficiently operate at 1/4th the system clock frequency, yet the other state machine can much more efficiently operate at 1/8th the system clock frequency. DualEDGE flip-flops can be employed in conjunction with the clock divider to obtain such a scenario. The state machine running at 1/8th the system clock frequency can simply use the clock divider configured as divide by 8. Enabling the DualEDGE flip-flop on macrocells assigned to the other state machine and assigning the divide by 8 clock divider to those macrocells as well will yield an effective clock frequency that is 1/4th the system clock frequency. This means that the single clock divider can obtain virtual dual functionality of a divide by 8 and a divide by 4 counter. Notice that both clocks are synchronized with each other and the system clock so that the two state machines operate concurrently.

Summarizing the previous discussion, when utilizing the CoolRunner-II clock divider and/or the DualEDGE flip-flops, it is possible to obtain clock divisors of 2, 3, 4, 5, 6, 7, 8, 10, 12, 14 and 16. Additionally, when a group of macrocells use the clock divider and some of those macrocells use DualEDGE flip-flops, the clock divider network can effectively deliver two clock frequencies of divisors 1-2 (using CoolCLOCK described later), 2-4, 3-6, 4-8, 5-10, 6-12, 7-14, and 8-16. When DualEDGE flip-flops are used with the clock divider, lower power is achieved while more efficiently operating the design.

CoolCLOCK

CoolRunner-II CPLDs are equipped with a unique feature, CoolCLOCK, to further reduce the power consumption of the global clock network without affecting the speed of the clock. Note that CoolCLOCK does not impose additional clock delays. As explained earlier, any efforts to reduce the clock frequency of the global clock net will significantly reduce power consumption.

CoolCLOCK reduces power consumed by the global clock net by dividing the external clock frequency by 2 before it is applied to the global clock network. This clock division occurs early in the clock tree near the clock input buffer so that the divided clock affects the majority of the clock network. Since the global clock network contains a relatively large amount of internal capacitance, a slower frequency will significantly reduce power consumed by this net, GCK2. This divided clock signal is then effectively doubled at the macrocell using the DualEDGE clocking feature of the flip-flops in the macrocell, shown in Figure 3. This ensures that the original clock frequency is applied to the macrocell as intended by the external system. Only macrocells that require the original clock frequency will be configured to utilize the DualEDGE flip-flop feature. Other macrocells can be configured to use the divided clock frequency to further reduce power when those macrocells don't require clocking at full speed.

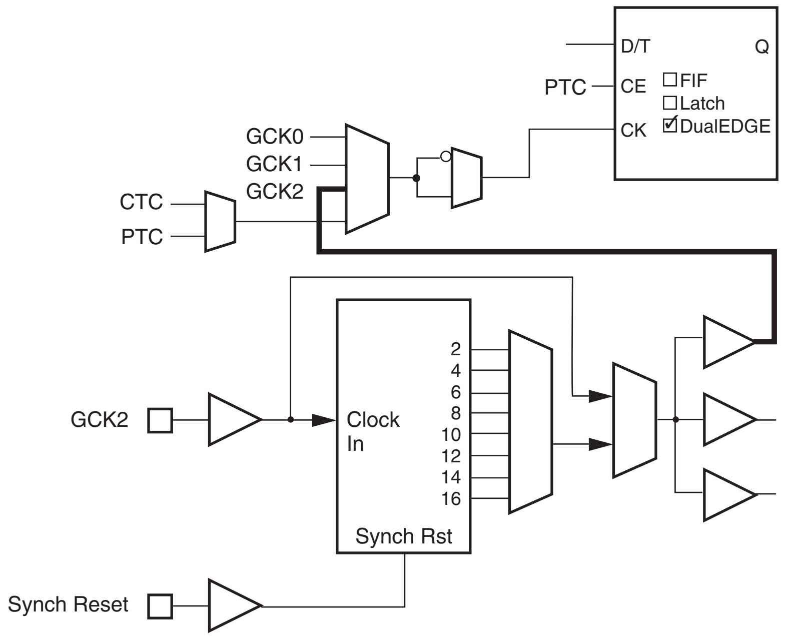


Figure 3: **CoolCLOCK Created by Cascading Clock Divider and DualEDGE Option**

Weak Pull-up and Bus Hold

All CoolRunner-II CPLDs include I/O termination options to reduce power consumption of the I/O due to externally 3-stated busses. The I/O termination circuitry can be configured in three ways: weak pull-up, bus hold, and no termination. Pull-up and bus hold are selected on a global basis and are mutually exclusive. Subsequently, the use or absence of the selected termination is specified on a per pin basis. Weak pull-up connects a high-impedance resistive load onto the I/O pin to prevent a floating situation on the pin. Bus hold is essentially a full latch on the I/O pin which drives the last state present on the I/O, either High or Low, prior to the bus going to the high impedance state. Bus hold is referred to in the software as "Keeper".

Floating inputs, an input which is not driven to a High or Low logic state, can use excessive power since the voltage on the gate of the input buffer may wander to a voltage level between standard logic levels. In this case, the input buffer is driven into the linear region where the P and N channel transistors are both switched on. Therefore, to avoid this situation, I/Os can be configured to use internal pull-ups or bus hold circuitry. I/Os configured as inputs or bidirectional should utilize this feature if it is known that the bus to which the I/O is attached will float at some point in its regular operation.

There are some cases where weak pull-up is undesirable. If the bus is pulled down the majority of time, current will flow through the weak pull-up to ground via the buffer, pulling the bus Low. A design such as this would benefit by using the bus hold circuitry.

A rule of thumb for any CMOS device is to not allow inputs to float. These two features of CoolRunner-II CPLDs, weak pull-up and bus hold, will minimize the chance of consuming excessive power on input pins.

I/O termination should be considered for each application and each I/O to determine the best combination to reduce power consumption. Again, avoid floating inputs whenever possible.

Slew Rate

This is a feature of the I/O structure that regulates the rate at which the output buffer changes states. There are two modes: FAST and SLOW. Designers concerned with reducing reflections on circuit board traces, minimizing RFI or minimizing EMI should specify a SLOW slew rate. Most design engineers considering low power designs will usually be designing slower speed systems and therefore will not be as concerned with reflections. In a case such as this, the system will benefit from a lower power perspective when slew rate is specified to be FAST.

Although an I/O is configured as an output, the input buffer is still connected to the pin and therefore senses changes in voltage on that pin. If the output is configured with a SLOW slew rate, the output voltage switches states less quickly, thereby lingering longer between standard logic levels. The input buffer will therefore be driven in the linear region (the input voltage between standard logic levels) for a longer period of time than if the output was configured in the FAST slew rate mode. Current will flow through the input buffer P and N channel transistors for a longer period of time resulting in higher power consumption. It is therefore recommended to configure the output with a FAST slew rate whenever reflections are not a concern.

Power Saving Techniques

Several rules of thumb should be followed when designing any circuit with CMOS devices to reduce power consumption. A basic understanding of power consumption must be reached prior to discussing these rules of thumb. Therefore, a derivation of the current equation for CMOS devices is necessary. Once the mathematical model of current is understood, it becomes easy to follow the theory of the rules of thumb. To maximize power savings, the designer should apply these concepts when implementing any CMOS device.

Derivation of Current Equation in CMOS Devices

Since the CMOS device is constructed of PMOS and NMOS transistors, the dynamic model is simply a capacitive structure for each transistor. Of course, the basic structure of a capacitor is the dielectric between two plates. In this case, the dielectric of the capacitor is the oxide layer on the silicon wafer and the plates are the poly or metal gate together with the inversion layer in the channel. The interconnecting metal/poly routing is also modeled as a capacitor where the plates are the routing itself together with any underlying routing or conductive material and the dielectric is any non-conductive structure between the plates. With these capacitive structures, the CMOS device is largely modeled as a collection of capacitors. Recall the basic equation for current through a capacitor:

$$I = C \cdot \frac{dV}{dt}$$

Breaking the derivative into its components we can extract the basic equation for frequency, which is the inverted value of a change in time, and simplify the equation:

$$I = C \cdot dV \cdot \frac{1}{dt}$$

$$I = C \cdot dV \cdot f$$

Since the voltage for capacitive structures in CMOS devices changes as a square wave with discontinuities between logic HighHigh and Low and is ideally rail to rail, we can further simplify the equation. For example, in a system of supply voltage, V, the change in voltage, dV, is V to 0V as shown and reduced here:

$$dV = V_2 - V_1$$

$$dV = V - 0$$

$$dV = V$$

Therefore the current equation for each capacitive structure becomes:

$$I = C \cdot V \cdot f$$

where:

- **I** = current in Amperes
- **C** = the capacitance of the capacitive structure in Farads
- **V** = the system voltage in Volts
- **f** = the toggle frequency of the capacitive structure in Hz

Dynamic device current is the summation of all capacitive structures toggling over time. Voltage remains the same for all equations and can be assumed to be a constant. A device of n capacitive structures can be represented as follows:

$$I_{Dynamic} = V \cdot \sum_{i=1}^{n} C_i \cdot f_i$$

To obtain total device current, static current must be added to the dynamic current:

$$I_{Total} = I_{Static} + V \cdot \sum_{i=1}^{n} C_i \cdot f_i$$

For illustrational purposes, it may be easier to discuss total current using a simplified version of this equation:

$$I_{CC} = I_{CCQ} + C \cdot V \cdot f$$

where:

- I_{CC} = *total* device current in Amperes
- I_{CCQ} = *quiescent* device current in Amperes
- **C** = the *lumped* capacitance of the device in Farads
- **V** = the system voltage in Volts
- **f** = the *average* device toggle frequency in Hz

Recall that to obtain power, multiply current by voltage to yield Watts.

Reduce System Speed

It becomes obvious from the equation that for a fixed device capacitance and voltage, reducing the average device toggle rate will reduce power consumption.

Limiting the system clock speed as well as the data bus speed to slower values will reduce power consumption since average device toggle rate will become smaller. Careful analysis of the required CPLD clock speed is essential for low power design. Over-clocking the CPLD design beyond the needs of the logic unnecessarily consumes extra power. Evaluating the minimum required speed of the CPLD logic will ensure that the system clock will have a minimal effect on power consumption.

Drive Inputs to Standard Logic Levels

Power consumption will rise dramatically when a CMOS input buffer is not driven to known, standard logic levels, otherwise known as allowing the input to "float". A voltage level between standard logic levels causes the input buffer transistors, typically P and N channel, to be biased in a manner where both are in the ON state. When biased in this way, a large amount of current will flow between the power and ground supplies of the device via this channel. It is therefore imperative to drive the input buffer to a known logic level High or Low state to turn off one of these two transistors and avoid this situation.

The beauty of CMOS logic is that power consumption is nearly zero when logic gates are held at a known logic input level. FZP technology found in CoolRunner-II CPLDs uses this principle to provide dramatic power savings over other CPLDs.

Further, it is advised to drive the input to the full voltage rail on a High logic level and fully to ground on a logic Low level. Even though the voltage level is within the acceptable logic levels, i.e., V_{IH} and V_{IL}, the closer the voltage is to the absolute voltage rails, the less current will be consumed in the input buffer. This effect is much less than the scenario where the input voltage floats between logic levels, but nevertheless can have a significant impact on total power consumption when summed across several I/Os.

Increase Input Edge Speed

CMOS device inputs that are driven by a slowly switching source will consume more power since the input spends more time biased in the linear region. Again, the linear region is found when an input is biased at a voltage between standard logic High or Low levels. When the CMOS input buffer is biased in this manner, both the P and N channel transistors are turned on, allowing a relatively large amount of current to pass to ground. The longer the buffer is biased in this fashion, the more power will be consumed by the device. Therefore, it is recommended to quickly switch the signal as it is applied to the inputs of the CMOS device. This applies to all clock pins, dedicated input pins, or I/Os configured as inputs or bidirectional.

Eliminate Bus Conflicts

Occasionally, bus conflicts occur where two output buffers are driving a line at the same time in opposite directions. This adversely affects logic performance as well as power consumption. Two drivers attempting to swing the bus at opposite voltage levels will draw excessive current.

A similar situation occurs when a bus is pulled High via a pull-up resistor, for example, and the output buffer is driving the bus Low. In this example, current will flow from the power source, through the pull-up resistor, the N channel transistor in the output buffer, and to ground. Current, in this case, is a function of the value of the pull-up resistor and the N channel impedance. A weak pull-up resistor, on the order of 10k ohms or more, is a good place to start if it is desired to use such a component. The larger the resistor, the less power will be consumed, but this will slow down the bus response time when the resistor is required to charge the bus to the High level if the bus is in the 3-state condition.

It is recommended to continuously drive a bus line High or Low with one device at a time and remove pull-up resistors whenever possible to obtain the lowest power condition. In some situations, this may not be possible. For example, an SMBus or I^2C SDA line is required to be released by all components but be at a High level when released. In this case, a pull-up resistor is required.

Bus Terminations

A different case where pull down resistors are necessary is when shunt bus termination is required to reduce reflections in high speed designs. These terminations are usually designed to be pull down resistors whose value equals the equivalent impedance of the data bus transmission line and are positioned as close to the load pin as possible. Whenever the output buffer is driving a transmission line with shunt resistors, the P channel transistor will source current into the load and therefore will raise power consumption.

It may be possible to avoid using shunt termination resistors by using a very short transmission line. The rule of thumb for a short transmission line is one whose length is less than one-sixth the electrical length of the rise time, and is described using the following equation:

$$\text{Length} \ll \frac{1}{6} \cdot \frac{T_{\text{rise}}}{\sqrt{LC}}$$

where:

- **Length** = maximum line length in inches
- **T_{rise}** = rise time in seconds
- **L** = the line inductance in Henries/inch
- **C** = the line capacitance in Farads/inch

By using the CoolRunner-II CPLD slew rate feature configured to SLOW, the rise time becomes longer. Using the above equation, it can be determined if the length of the transmission line can be effectively long enough to avoid reflections altogether. If by using the SLOW slew rate feature, the length of the transmission line is short by the above rule of thumb, shunt bus termination resistors can be avoided thereby saving power.

If either method is not an option, insert a series termination resistor positioned at the source pin with the same value as the transmission line impedance. This option will avoid excessive power consumption (since there is no shunt load) and eliminate reflections at the source. However, a series termination resistor will allow one reflection at the load (since the load impedance does not match the transmission line impedance) which implies that any component mid way between the source and the load will see two transitions: the incident wave and the single reflected wave created at the load.

Reduce System Voltage

Using the total current equation with fixed capacitance and average frequency, it becomes readily apparent that reducing system voltage will reduce power consumption, provided the system voltage remains within recommended operating specifications. Therefore, it makes sense to use a 1.8V device in lieu of a 3.3V or 2.5V device. CoolRunner-II CPLDs are 1.8V devices and therefore utilize this concept. Further, a device made with a smaller process technology, such as CoolRunner-II CPLDs, will generally have lower lumped internal capacitance values thereby reaching additional power savings. Combining these two factors with reducing average system frequency will also cut power consumption.

With any voltage device, there is a recommended operating range, and it may be advantageous to operate low in that voltage range to further reduce power consumption. Voltages that are outside the recommended operating range may cause excessive power consumption including adverse functional performance.

Reduce Bus Loading

Connecting an external bus to a CMOS device will increase power consumption due to the loading effects of the bus on the device. The primary factor from loading is given by capacitive or resistive bus components as seen by the output buffer looking into the bus.

Capacitive loading comes in two forms: lumped and distributed. Lumped capacitance is typically found from the gate of the input buffer of other connected CMOS devices. It is also developed from any capacitive element that is attached to the bus. Distributed capacitance is present due to the routing of the trace on the PCB. Both types will charge and discharge during logic transitions based on the previous logic state of the bus. Capacitive loading will draw current from the CMOS device whose output buffer is driving the bus, thereby increasing apparent power consumption of that device as seen at its power pins. To minimize power consumption based on this capacitive loading effect, it is necessary to reduce the size of the lumped capacitance found in attached devices, and to shorten the PCB traces. Doing so will also increase potential system speed since reflections are less likely.

Resistive loading is usually found when devices with a resistive element to their impedance is attached to the bus. For example, a pull down resistor attached to a PCB trace will allow for current to flow from the CMOS device power rail, through the P channel of the output buffer transistor, through the resistor, and to ground, increasing observed power consumption at the CMOS device power pins. Reducing resistive loads will reduce power consumption.

Keep in mind that many components are comprised of more than one type of impedance element. For example, capacitors also have resistive and inductive elements, albeit small.

Further Power Saving Techniques

For further discussions regarding power saving techniques, particularly those involving logic design, review XAPP346 - Low Power Tips for CoolRunner Design found at http://direct.xilinx.com/bvdocs/appnotes/xapp346.pdf. Although the referenced application note discusses CoolRunner XPLA3 CPLDs, the basic principles apply to CoolRunner-II CPLDs.

Conclusion

For the first time in the CPLD industry, CoolRunner-II products combine true high speed logic with ultra low power. Unique features of the CoolRunner-II CPLD, such as CoolCLOCK and DataGATE, allow the designer to further reduce power consumption. By using these features combined with good design practice, as outlined in this document, designers can be assured that their design will experience optimal low power benefits without sacrificing high speed.

References

1. Johnson H. and Martin G. (1993): High-Speed Digital Design: A Handbook of Black Magic Prentice Hall PTR

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/08/02	1.0	Initial Xilinx release.

XAPP378 (v1.2) June 5, 2005

Using CoolRunner-II Advanced Features

Summary

This application note describes how to implement the CoolRunner™-II advanced features in the Xilinx software. These features include the DualEDGE triggered registers, clock divider, CoolCLOCK, DataGATE, Schmitt trigger inputs, and I/O termination types. HDL code examples are available for download, see **Code Examples Download**, page 160.

CoolRunner-II

Xilinx CoolRunner-II CPLDs combine performance and low power in a single device. For more information on the architecture of CoolRunner-II CPLDs, see **References**, page 160. CoolRunner-II CPLDs feature enhanced clocking flexibility and provide design capabilities that significantly reduce power consumption.

All design features discussed in this application note are supported in Foundation™ ISE and WebPACK™ ISE software from Xilinx. For more information on Xilinx software, see **References**, page 160.

Software Attributes

There are two main methods of attribute entry, each with their own advantages. A User Constraint File (UCF) has the advantage of being easily edited while being separate from the design source files. It has the benefit of allowing changes without needing to re-synthesize the source code. For more information on entering constraints with the UCF, see **References**, page 160.

Attribute entry within the source has the advantage of not needing to maintain a separate file for the design constraints. All CoolRunner-II VHDL and Verilog attribute examples are only applicable to the XST synthesis tool.

The attributes that are user definable and specific to the CoolRunner-II CPLD include: CoolCLOCK, DataGATE, Schmitt trigger input, keeper or pullup I/O termination, I/O standards, VREF, and open drain outputs. Some design constraints from this list are software selectable via the ISE Project Navigator GUI. Individual signal constraints must be explicity defined in the methods described above.

Please note that several syntax and help guides for attributes exist within the Xilinx ISE Project Navigator. These help files are updated with each release of software and provide a reliable source of information. The help menu can be located from within ISE under the menu option **Help | ISE Help Contents | CPLD Attributes**.

DualEDGE Registers

CoolRunner-II DualEDGE triggered registers allows designers to reach unprecedented performance levels. CoolRunner-II CPLDs can double system performance by creating DualEDGE triggered (DET) registers. CoolRunner-II DET registers allow data to be registered on both the rising and falling edge of a clock.

CoolRunner-II DET registers can be used for logic functions that include shift registers, counters, comparators, and state machines. Designers must evaluate the desired performance of the CPLD logic to determine use of DET registers.

The DET register can be inferred in any ABEL, HDL, or schematic design. Table 1 lists the inference methods available to create a DET register in CoolRunner-II.

Table 1: **DET Register Inference Summary**

Design Entry	Instantiation Method
ABEL	Use the following syntax: QOUT:=data; QOUT.DEC=clock;
VHDL/Verilog	Infer a dual edge triggered register.
Schematic	Instantiate a FDDn[S][R][E] component.

Examples

A designer can infer a single edge triggered (SET) register in any HDL design. A SET register active on the rising edge of the input clock would require the following VHDL or Verilog syntax.

```
VHDL: if (clock'event) and (clock = '1') then
Verilog: always @ (posedge clock)
```

The required syntax to infer a CoolRunner-II DET register requires the register be active on both the rising and falling edge of the clock. The following VHDL syntax would be used to infer a CoolRunner DET register.

```
process (clock)
begin
  if (clock'event) then
    ...
  end if;
end process;
```

The following Verilog syntax would be used to infer a DET register in CoolRunner-II.

```
always @ (negedge clock or posedge clock)
  ...
```

The DET register is available with all macrocells in all devices of the CoolRunner-II family.

Clock Divider

CoolRunner-II CPLDs provide additional clocking flexibility to the DET register feature with the clock divider. The CoolRunner-II clock divider provides the capability to divide an incoming clock and globally distribute the divided clock to all macrocells. The clock divider provides additional power savings by reducing the toggle frequency of the internal clock network.

The CoolRunner-II clock divider is available on global clock, GCK2, and can divide the incoming clock by 2, 4, 6, 8, 10, 12, 14, and 16. The clock divider creates a 50-50 duty cycle divided clock without affecting T_{CO}. The clock divider output is initialized low by the CPLD power up reset circuitry.

The clock divider circuit includes an active high synchronous reset, referred to as CDRST. When the CDRST signal is asserted, the clock divider output is disabled after the current cycle. When the CDRST signal is de-asserted the clock divider output will become active upon the first edge of GCK2.

Figure 1 illustrates the CoolRunner-II clock divider.

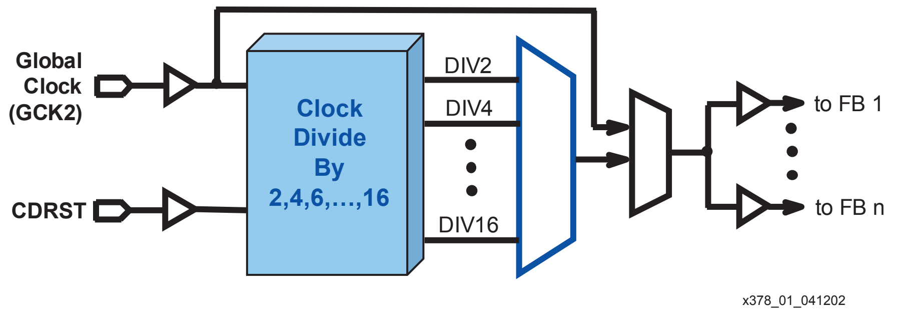


Figure 1: **CoolRunner-II Clock Divider**

The CoolRunner-II clock divider includes a built in delay circuit. With the delay feature enabled, the output of the clock divider will be delayed for one full count cycle. When used, the clock divider does not output a rising clock edge until after the divider reaches the terminal count value. The delay feature is either enabled or disabled upon configuration. The type of clock divider component instantiated will determine if the delay is enabled or disabled. Figure 2 illustrates a timing waveform of the CoolRunner-II clock divider with the delay enabled and disabled.

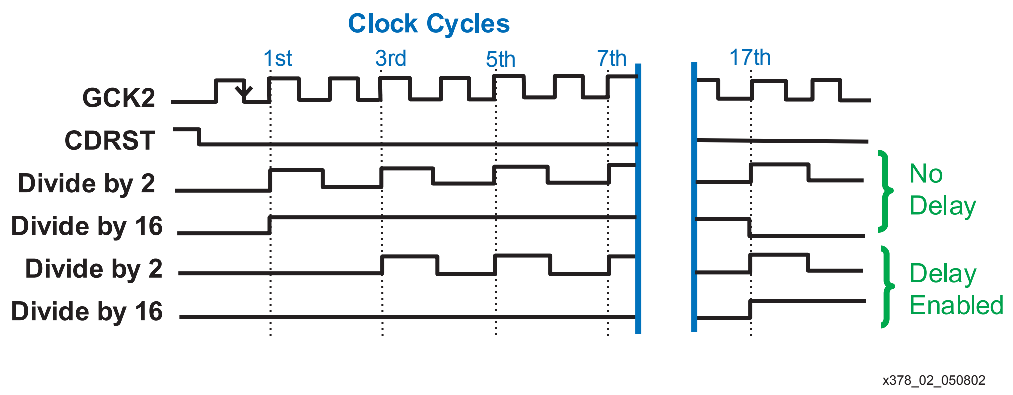


Figure 2: **Clock Divider Waveform**

Xilinx Synthesis Technology (XST) allows a clock divider component to be instantiated directly in the HDL source code. Table 2 lists the available clock divider components that can be instantiated in any ABEL, HDL, or schematic design.

Table 2: **Clock Divider Library Components**

Component	Description
CLK_DIVn	Global Clock Divider Component. No support of the synchronous reset or start delay features. Available: CLK_DIV2, 4, 6, 8, 10, 12, 14, 16
CLK_DIVnR	Global Clock Divider with Synchronous Reset. No support of the start delay feature. Available: CLK_DIV2, 4, 6, 8, 10, 12, 14, 16R
CLK_DIVnSD	Global Clock Divider with Start Delay. No support of the synchronous reset. Available: CLK_DIV2, 4, 6, 8, 10, 12, 14, 16SD
CLK_DIVnRSD	Global Clock Divider with Synchronous Reset and Start Delay. Available: CLK_DIV2, 4, 6, 8, 10, 12, 14, 16RSD

VHDL Example

To design with the CoolRunner-II clock divider in VHDL requires both a component declaration and component instantiation. The component declaration declares the name and interface of the clock divider unit. The VHDL component declaration syntax for using a clock divide by 2, the CLK_DIV2 component is shown here.

```
component CLK_DIV2 is
port (
  CLKIN : in STD_LOGIC;
  CLKDV : out STD_LOGIC );
end component;
```

The component instantiation associates signals with the ports of the clock divider component. If a clock divide by 2 is desired, the CLK_DIV2 component must be instantiated. The incoming clock signal, *clk*, is declared on the CLKIN port and the clock divider output signal, *clk_div_by_2*, is declared on the CLKDV output port. The VHDL syntax is shown here for instantiating the CLK_DIV2 component.

```
U1: CLK_DIV2
port map(
  CLKIN => clk,
  CLKDV => clk_div_by_2 );
```

If a clock divide by 16 with a synchronous reset and start delay is desired, the CLK_DIV16RSD component must be declared and instantiated. The VHDL syntax for the component declaration is shown here.

```
component CLK_DIV16RSD is
port (
  CLKIN : in STD_LOGIC;
  CDRST : in STD_LOGIC;
  CLKDV : out STD_LOGIC );
end component;
```

The component instantiation assigns the port signals. The input clock signal, *clk*, is declared on the CLKIN port. The clock divider reset signal, *clk_div_rst*, is declared on the CDRST port. The clock divider output, *clk_div_by_16*, is declared on the CLKDV port. This component will also enable the start delay circuitry in the CoolRunner-II clock divider. The syntax to instantiate the CLK_DIV16RSD component in VHDL is shown here.

```
U1: CLK_DIV16RSD
port map (
  CLKIN => clk,
  CDRST => clk_div_rst,
  CLKDV => clk_div_by_16 );
```

Verilog Example

Verilog design entry with XST does not require a component declaration; only the component instantiation is necessary. The Verilog syntax to instantiate the CLK_DIV16RSD component is shown here. The input clock signal, *clk*, is assigned to the CLKIN port. The clock divider reset signal, *clk_div_rst*, is assigned to the CDRST port. The clock divider output, *clk_div_by_16*, is assigned to the CLKDV port.

```
CLK_DIV16RSD U1 (
  .CLKIN (clk),
  .CDRST (clk_div_rst),
  .CLKDV (clk_div_by_16) );
```

ABEL Example

Designing with clock dividers in CoolRunner-II requires both the component declaration and component instantiation. The clock divider must be declared as an external component in an ABEL design as shown here.

```
CLK_DIV2R external (CLKIN, CDRST -> CLKDV);
```

Component instantiation in an ABEL design occurs by assigning an identifier to the clock divider component. In this example, U1 is the identier assigned to the clock divider component, CLK_DIV2R, using the functional_block ABEL keyword.

```
U1 functional_block CLK_DIV2R;
```

The following equations illustrate the component port mapping. The input clock, *clk*, is mapped to the CLKIN port. The clock divider reset signal, *clk_div_rst*, is mapped to the CDRST port. The clock divider output signal, *clk_div_by_2*, is assigned to the CLKDV output port.

```
U1.CLKIN = clk;
U1.CDRST = clk_div_rst;
clk_div_by_2 = U1.CLKDV;
```

Notes:

1. The signal assigned to the CDRST port will automatically be mapped to the CDRST/I/O pin.
2. The output of the clock divider circuit is only available as a clock input to registers within the CPLD. The clock divider output can not be used in combinational logic or routed off-chip for external use.
3. The clock divider is available on CoolRunner-II 128 macrocell devices and larger.

CoolCLOCK

The CoolRunner-II CoolCLOCK feature is the technique of combining the global clock divider and DET registers. Power savings are achieved by dividing the global clock by 2, distributing a lower frequency clock on the internal clock network, and then doubling the clock at each macrocell. Zero clock skew can be acheived due to the zero insertion delay of the clock divider and the DET registers. Figure 3 illustrates the CoolRunner-II CoolCLOCK feature.

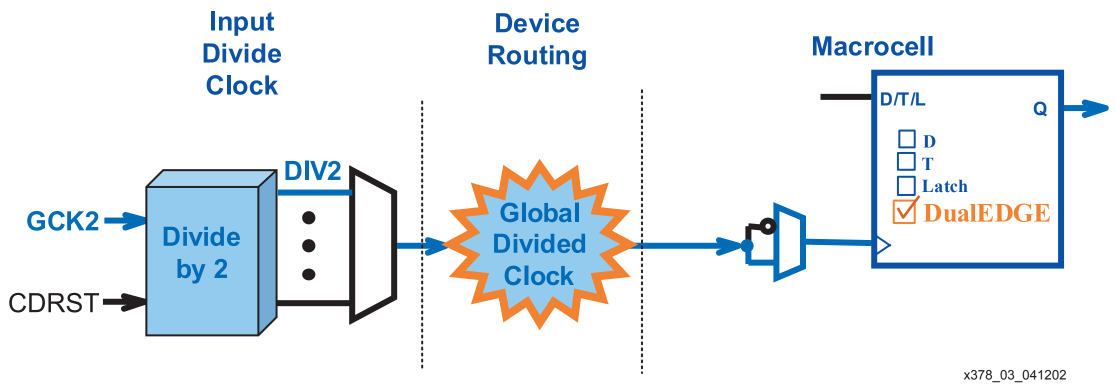


Figure 3: **CoolCLOCK**

Since GCK2 is the only clock network that can be divided, the CoolCLOCK feature is only available on GCK2. The CoolCLOCK feature can be implemented by assigning an attribute to an input clock. The CoolCLOCK attribute replaces the need to instantiate the clock divider and infer DET registers. Table 3 lists the methods available to use the CoolCLOCK attribute

Table 3: **CoolCLOCK Attribute**

Attribute Format	Syntax	Example
UCF	NET <*clock name*> COOL_CLK;	NET clk COOL_CLK;
ABEL	XILINX PROPERTY 'COOL_CLK <*clock name*>';	XILINX PROPERTY 'COOL_CLK clk';

VHDL	attribute COOL_CLK : string;
	attribute COOL_CLK of <*clock name*>: signal is "TRUE";
		attribute COOL_CLK : string;
		attribute COOL_CLK of clk : signal is "TRUE";

Verilog	//SYNTHESIS attribute COOL_CLK of <*clock name*>: signal is "TRUE"; Note: The comment delimiters are intentional and necessary for XST.	//SYNTHESIS attribute COOL_CLK of clk : signal is "TRUE";

Note: The CoolCLOCK feature is available on CoolRunner-II 128 macrocell devices and larger.

DataGATE

CoolRunner-II designers can block specified inputs under the control of the DataGATE function. By blocking inputs, switching signals do not drive internal chip capacitance and thereby reduce overall power consumption. The last value on the input pin prior to the assertion of the DataGATE rail is latched and used by the CPLD internally. Figure 4 illustrates the DataGATE feature in CoolRunner-II.

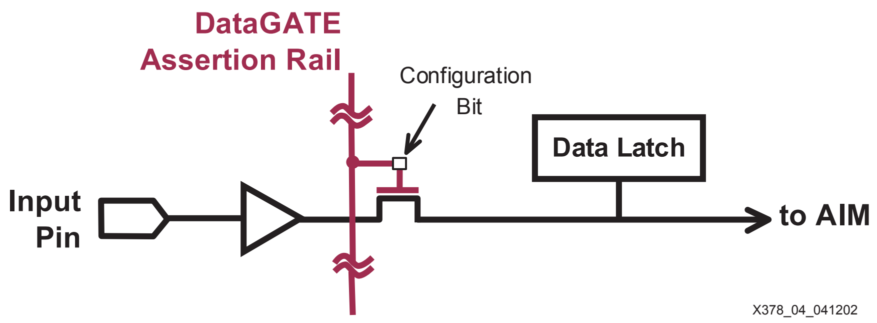


Figure 4: **DataGATE Block Diagram**

There are two attributes associated with the DataGATE feature in CoolRunner-II. The first attribute specifies if an input will be affected by DataGATE and the second designates the DataGATE control signal.

The DataGATE feature is selectable on a per pin basis. Each input pin that uses DataGATE must be assigned a DATA_GATE attribute. Table 4 illustrates the syntax for enabling DataGATE on an input signal.

Table 4: **DataGate Attribute**

Attribute Format	Syntax	Example
UCF	NET <*signal name*> DATA_GATE;	NET data_in DATA_GATE;
ABEL	XILINX PROPERTY 'DATA_GATE <*signal name*>';	XILINX PROPERTY 'DATA_GATE data_in';

VHDL	attribute DATA_GATE : STRING;
	attribute DATA_GATE of <*signal name*>: signal is "TRUE";
	Note: The string attribute need only be declared once for all DATA_GATE attributes.
		attribute DATA_GATE : STRING;
		attribute DATA_GATE of data_in : signal is "TRUE";

Verilog	//SYNTHESIS attribute DATA_GATE of <*signal name*>: signal is "TRUE"; Note: The comment delimiters are intentional and necessary for XST.	//SYNTHESIS attribute DATA_GATE of data_in : signal is "TRUE";

The DataGATE assertion rail can be driven from either an I/O pin or internal logic. The DataGATE enable signal is a dedicated DGE/I/O pin for each package in CoolRunner-II. Upon implementation, the software recongnizes a design using DataGATE and automatically assigns this I/O pin to the DataGATE enable control function, DGE. Internally generated DataGATE contol logic can be assigned to this I/O pin with the BUFG=DATA_GATE attribute. The methods of assigning the DataGATE enable signal are shown in Table 5.

Table 5: **DataGate Control Attribute**

Attribute Format	Syntax	Example
UCF	NET <*signal name*> BUFG=DATA_GATE;	NET dg_en BUFG=DATA_GATE;
ABEL	XILINX PROPERTY 'BUFG=DATA_GATE <*signal name*>';	XILINX PROPERTY 'BUFG=DATA_GATE dg_en';

VHDL	attribute BUFG : STRING;
	attribute BUFG of <*signal name*>: signal is "DATA_GATE";
	Note: The string attribute need only be declared once for all BUFG attributes.
		attribute BUFG : STRING;
		attribute BUFG of dg_en : signal is "DATA_GATE";

Verilog	//SYNTHESIS attribute BUFG of <*signal name*>: signal is "DATA_GATE"; Note: The comment delimiters are intentional and necessary for XST.	//SYNTHESIS attribute BUFG of dg_en : signal is "DATA_GATE";

Schmitt Trigger

Each CoolRunner-II I/O has multiple input buffers used for various I/O standard configurations. One of these input buffers behaves as a Schmitt trigger input and is enabled upon CPLD configuration. The Schmitt trigger input allows the board designer the flexibility to utilize the CoolRunner-II with both high speed signals as well as slow switching signals on the same device. Slowly switching signals can cause havoc on digital systems by causing double clocking or glitches on a CMOS input, however this can be virtually eliminated by using the Schmitt trigger input. The Schmitt trigger input use is only at the cost of a few nanosecond delay (refer to the CoolRunner-II datasheet, see **References**, page 160).

Table 6 illustrates the attribute syntax to assign the Schmitt trigger input buffer to a specific signal.

Table 6: **Schmitt Trigger Attribute**

Attribute Format	Syntax	Example

UCF	NET <*signal name*> SCHMITT_TRIGGER;	NET data_in SCHMITT_TRIGGER;
		NET clock SCHMITT_TRIGGER;

ABEL	XILINX PROPERTY 'SCHMITT_TRIGGER <*signal name*>';	XILINX PROPERTY 'SCHMITT_TRIGGER data_in';
		XILINX PROPERTY 'SCHMITT_TRIGGER clock';

VHDL	attribute SCHMITT_TRIGGER : STRING;
	attribute SCHMITT_TRIGGER of <*signal name*>: signal is "TRUE";
	Note: The string attribute need only be declared once for all SCHMITT_TRIGGER attributes.
		attribute SCHMITT_TRIGGER : STRING;
		attribute SCHMITT_TRIGGER of data_in: signal is "TRUE";
		attribute SCHMITT_TRIGGER of clock: signal is "TRUE";

Verilog	//SYNTHESIS attribute SCHMITT_TRIGGER of <*signal name*>;
	Note: The comment delimiters are intentional and necessary for XST.
		//SYNTHESIS attribute SCHMITT_TRIGGER of data_in;
		//SYNTHESIS attribute SCHMITT_TRIGGER of clock;

I/O Termination

CoolRunner-II pins may be terminated in the following ways: keeper (also referred to as bushold) and pullup. Usage of the keeper and the pullup circuitry is exclusive on a global basis. When one of these two (keeper and pullup) termination modes is selected for any number of signals, the other termination mode is no longer available to any other signal.

Keeper

The keeper circuitry provides the ability to hold the last known value on an I/O pin using weak pullup/down resistors. If an unterminated I/O pin was in high-impedence and floating, this would cause excessive leakage current. The keeper circuitry eliminates the need for external termination that would resolve this. Table 7 illustrates the attribute syntax for specifying the keeper termination on any I/O pin.

Table 7: **Keeper Attribute**

Attribute Format	Syntax	Example

UCF	NET <*signal name*> KEEPER;	NET data_in KEEPER;
		NET clock KEEPER;

ABEL	XILINX PROPERTY 'KEEPER <*signal name*>';	XILINX PROPERTY 'KEEPER data_in';
		XILINX PROPERTY 'KEEPER clock';

VHDL	attribute KEEPER : STRING;
	attribute KEEPER of <*signal name*>: signal is "TRUE";
	Note: The string attribute need only be declared once for all KEEPER attributes.
		attribute KEEPER : STRING;
		attribute KEEPER of data_in: signal is "TRUE";
		attribute KEEPER of clock: signal is "TRUE";

Verilog	//SYNTHESIS attribute KEEPER of <*signal name*>;
	Note: The comment delimiters are intentional and necessary for XST.
		//SYNTHESIS attribute KEEPER of data_in;
		//SYNTHESIS attribute KEEPER of clock;

Pullup

The internal pullup allows the designer to eliminate external pullup resistors on the board, thereby reducing cost and simplifying board layout. Table 8 illustrates the attribute syntax for specifying the pullup I/O termination.

Table 8: **Pullup Attribute**

Attribute Format	Syntax	Example

UCF	NET <*signal name*> PULLUP;	NET data_in PULLUP;
		NET clock PULLUP;

ABEL	XILINX PROPERTY 'PULLUP <*signal name*>';	XILINX PROPERTY 'PULLUP data_in';
		XILINX PROPERTY 'PULLUP clock';

VHDL	attribute PULLUP : STRING;
	attribute PULLUP of <*signal name*>: signal is "TRUE";
	Note: The string attribute need only be declared once for all PULLUP attributes.
		attribute PULLUP : STRING;
		attribute PULLUP of data_in: signal is "TRUE";
		attribute PULLUP of clock: signal is "TRUE";

Verilog	//SYNTHESIS attribute PULLUP of <*signal name*>;
	Note: The comment delimiters are intentional and necessary for XST.
		//SYNTHESIS attribute PULLUP of data_in;
		//SYNTHESIS attribute PULLUP of clock;

I/O Configuration

CoolRunner-II devices support multiple I/O banks in a single device, allowing for easy interfacing to different voltage standards on one chip. A device can support one I/O standard per bank (i.e., XC2C128 has two banks and can therefore support up to two I/O standards). Regardless of which I/O voltage standard is selected, any pin may be configured as an open-drain output.

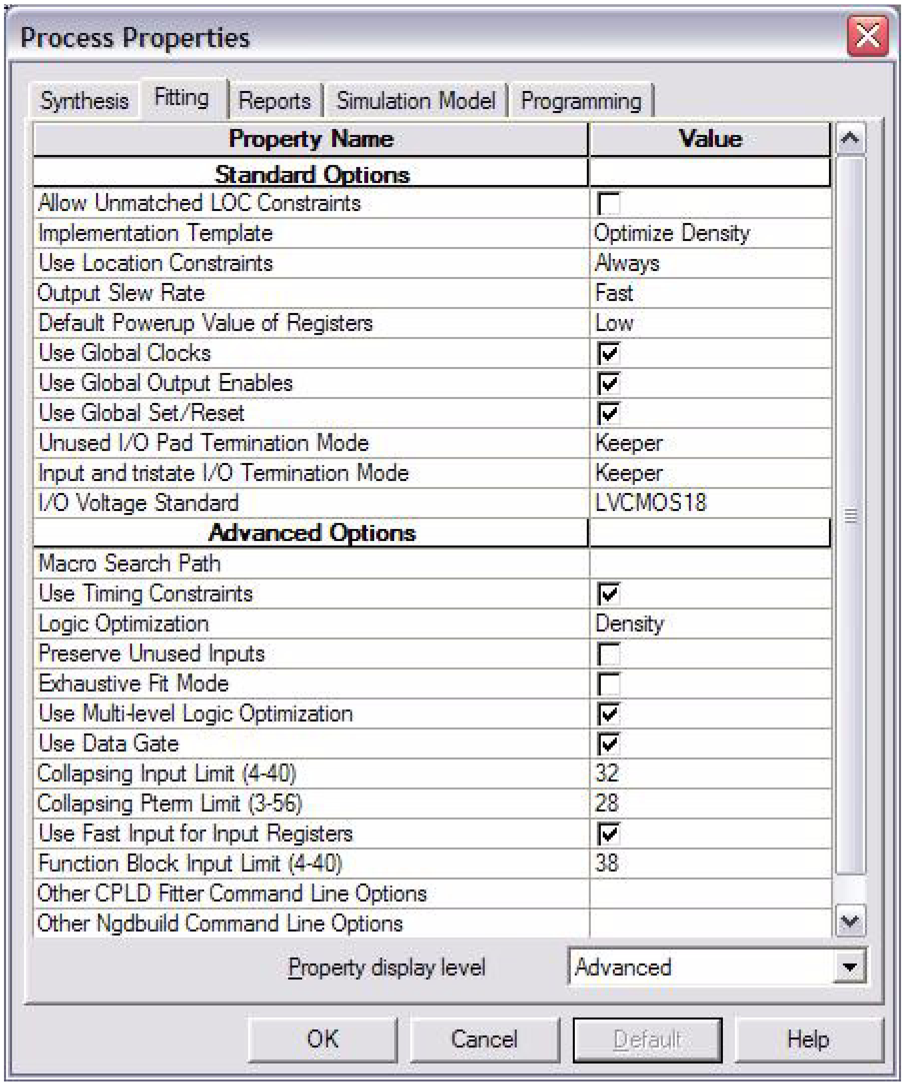
I/O Standards

Table 9 lists the supported I/O standards on CoolRunner-II devices. Note that all standards are not supported in every density.

Table 9: **CoolRunner-II Supported I/O Standards**

	XC2C32A	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
I/O Banks	2	2	2	2	4	4
LVTTL	Yes	Yes	Yes	Yes	Yes	Yes
LVCMOS33, LVCMOS25, & LVCMOS18	Yes	Yes	Yes	Yes	Yes	Yes
1.5V I/Os	Yes	Yes	Yes	Yes	Yes	Yes
SSTL2-1 & SSTL3-1	No	No	Yes	Yes	Yes	Yes
HSTL-1	No	No	Yes	Yes	Yes	Yes

Figure 5 illustrates how to specify the default I/O standard for all pins in a design. The I/O standard can be selected in the Implement Design Process Properties window under the Basic Tab.



Property Name	Value
Standard Options	
Allow Unmatched LOC Constraints	☐
Implementation Template	Optimize Density
Use Location Constraints	Always
Output Slew Rate	Fast
Default Powerup Value of Registers	Low
Use Global Clocks	☑
Use Global Output Enables	☑
Use Global Set/Reset	☑
Unused I/O Pad Termination Mode	Keeper
Input and tristate I/O Termination Mode	Keeper
I/O Voltage Standard	LVCMOS18
Advanced Options	
Macro Search Path	
Use Timing Constraints	☑
Logic Optimization	Density
Preserve Unused Inputs	☐
Exhaustive Fit Mode	☐
Use Multi-level Logic Optimization	☑
Use Data Gate	☑
Collapsing Input Limit (4-40)	32
Collapsing Pterm Limit (3-56)	28
Use Fast Input for Input Registers	☑
Function Block Input Limit (4-40)	38
Other CPLD Fitter Command Line Options	
Other Ngdbuild Command Line Options	

Property display level: Advanced

Figure 5: **Global I/O Standard Selection**

If a design requires multiple I/O standards on the same device, each pin must be manually declared with the appropriate I/O standard attribute. Table 10 illustrates the available I/O standard attributes that can be declared.

Table 10: **I/O Standard Attributes**

I/O Standard	Attribute Name
LVTTL	LVTTL
LVCMOS 3.3V	LVCMOS33
LVCMOS 2.5V	LVCMOS25
LVCMOS 1.8V	LVCMOS18
1.5V I/O	LVCMOS15

Table 10: **I/O Standard Attributes**

I/O Standard	Attribute Name
SSTL 2-1	SSTL2_I
SSTL 3-1	SSTL3_I
HSTL-1	HSTL_I

Table 11 illustrates the syntax for specifying an I/O standard attribute. The examples shown in Table 11 are for specifying the LVCMOS18 I/O standard. For other standards, LVCMOS18 can be replaced with the appropriate attribute name shown in Table 10.

Table 11: **I/O Standard Attribute Syntax**

Attribute Format	Syntax	Example

UCF	NET <*signal name*> <*I/O standard attribute name*>;	NET data_in IOSTANDARD=LVCMOS18;
		NET clock IOSTANDARD=LVCMOS18;

ABEL	XILINX PROPERTY 'IOSTANDARD=LVCMOS18 <*signal name*>';	XILINX PROPERTY 'IOSTANDARD=LVCMOS18 data_in';
		XILINX PROPERTY 'IOSTANDARD=LVCMOS18 clock';

VHDL	attribute IOSTANDARD : STRING;
	attribute IOSTANDARD of <*signal name*>: signal is "<*I/O standard attribute name*>";
	Note: The string attribute need only be declared once for all IOSTANDARD attributes.
		attribute IOSTANDARD : STRING;
		attribute IOSTANDARD of data_in: signal is "LVCMOS18";
		attribute IOSTANDARD of clock: signal is "LVCMOS18";

Verilog	//SYNTHESIS attribute IOSTANDARD of <*signal name*> is "<*I/O standard attribute name*>";
	Note: The comment delimiters are intentional and necessary for XST.
		//SYNTHESIS attribute IOSTANDARD of data_in is "LVCMOS18";
		//SYNTHESIS attribute IOSTANDARD of clock is "LVCMOS18";

Open Drain

A signal that is grounded when "false" and is in high-impedence when "true" is considered an open-drain signal. An output on CoolRunner-II can be configured as open drain by simply declaring the OPEN_DRAIN attribute in the Xilinx software. Table 12 illustrates the syntax for specifing an open drain output with the OPEN_DRAIN attribute.

Table 12: **Open Drain Attribute**

Attribute Format	Syntax	Example
UCF	NET <*signal name*> OPEN_DRAIN;	NET data_out OPEN_DRAIN;
ABEL	XILINX PROPERTY 'OPEN_DRAIN <*signal name*>';	XILINX PROPERTY 'OPEN_DRAIN data_out';

VHDL	attribute OPEN_DRAIN : STRING;
	attribute OPEN_DRAIN of <*signal name*>: signal is "TRUE";
	Note: The string attribute need only be declared once for all OPEN_DRAIN attributes.
		attribute OPEN_DRAIN : STRING;
		attribute OPEN_DRAIN of data_out: signal is "TRUE";

Verilog	//SYNTHESIS attribute OPEN_DRAIN of <*signal name*>; Note: The comment delimiters are intentional and necessary for XST.	//SYNTHESIS attribute OPEN_DRAIN of data_out;

Code Examples Download

Example source code is available for download. Both VHDL and Verilog code with test benches are available for using the CoolRunner-II advanced features.

XAPP378 - ftp://ftp.xilinx.com/pub/applications/refdes/xapp378.zip

Conclusion

Performance and low power have finally come together with CoolRunner-II CPLDs. The available advanced features furthur reduce power consumption and provide advanced clocking management options. For additional assistance with the CoolRunner-II advanced features, please contact the Xilinx hotline or refer to the Xilinx web support (http://support.xilinx.com/).

References

1. CoolRunner-II family data sheet
2. CoolRunner-II Application Notes
3. Xilinx ISE design entry software
4. Application Note: XAPP352: Utilizing a UCF for CoolRunner XPLA3 CPLDs

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/27/02	1.0	Initial Xilinx release.
12/04/02	1.1	Added clock divider notes.
06/05/05	1.2	Updated for CoolRunner-IIA, including Table 9; Updated Figure 5 for ISE 7.1i.

XILINX®

XAPP395 (v1.2) September 22, 2003

Using DataGATE in CoolRunner-II CPLDs

Summary

This application note outlines the various ways designers can utilize the DataGATE feature of CoolRunner™-II CPLDs.

Introduction

CoolRunner-II CPLDs deliver the lowest power consumption in today's CPLD marketplace. Built from standard CMOS structures, they achieve the classic I_{CC} = CVF relationship where "C" is driven capacitance, "V" is voltage swing and "F" is the switching frequency of gates driven within the part. The capacitance and voltage values in the equation are essentially demanded by the 0.18 micron process, but the "F" part of the expression is due to the design characteristics of the customer design inside. The desire to lower the "F" aspect resulted in the DataGATE feature in CoolRunner-II CPLDs.

As the name implies, the feature "gates" data. This application note shows how to get as near to the origin of the I_{CC} versus "F" curve as we can with today's technology. It describes the practical aspects of how DataGATE works, the timing model, and what you will need to do in the software to make it work. Further, it shows four different ways the circuitry can be used for power reduction, as well as for circuit debugging, printed circuit board "hot plugging," and device security. These are just some ideas on how to use this feature, and others may come to mind as users learn to operate DataGATE.

The final section of this application note lists additional application notes which cover other aspects of Xilinx CoolRunner-II CPLDs.

Operation

Figure 1 details the delivery of the DataGATE Assertion Rail throughout the I/O structure of a CoolRunner-II CPLD. This image doesn't convey the generation of the DataGATE activation method, but simply the delivery of the control signal to each input pin, with independent control of whether that pin will participate in the gating action. Specifically, two items should be noted. First, each pin can be programmed to participate in the DataGATE operation or not—the standard default is not to participate. The second aspect is that the individual pass transistors permit signal entry into the CPLD, or blocking of the input pin. Figure 2 expands the detail for a single input pin. If the input pin becomes blocked, the last driven value into the CPLD will automatically be latched and held so a solid binary value is delivered into the CPLD core. It should be noted that the Assertion Rail is driven from a specific macrocell within the CPLD, and that when that macrocell drives high, input data will be blocked at participating pins. If the Assertion Rail is low, data passes freely into the chip. The condition of the DataGATE Assertion Rail is manifested at a specific pin (designated DGE), which will vary from chip to chip within the CoolRunner-II family.

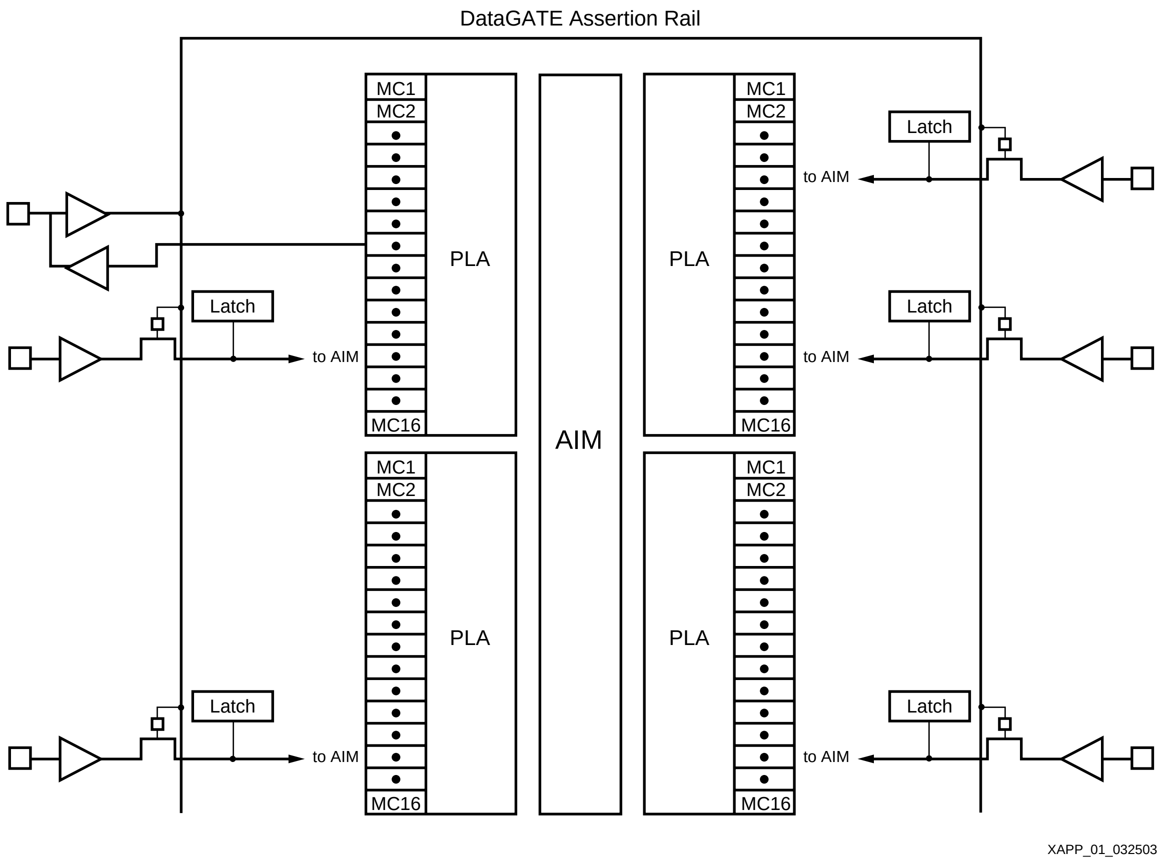


Figure 1: **DataGATE Assertion Rail**

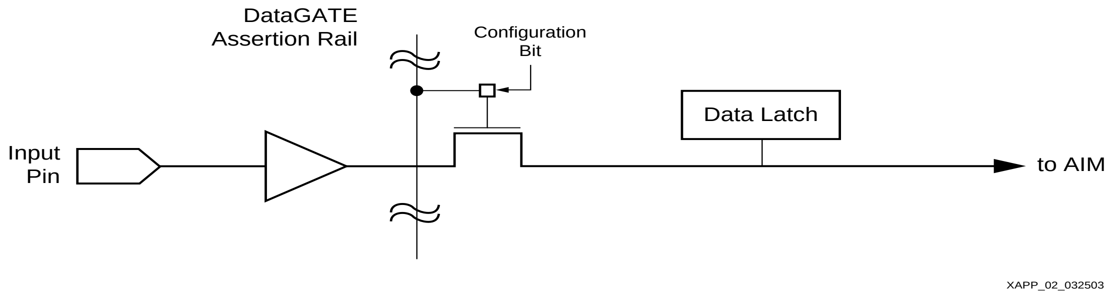


Figure 2: **DataGATE Assertion Rail Input Pin, Latching Old State**

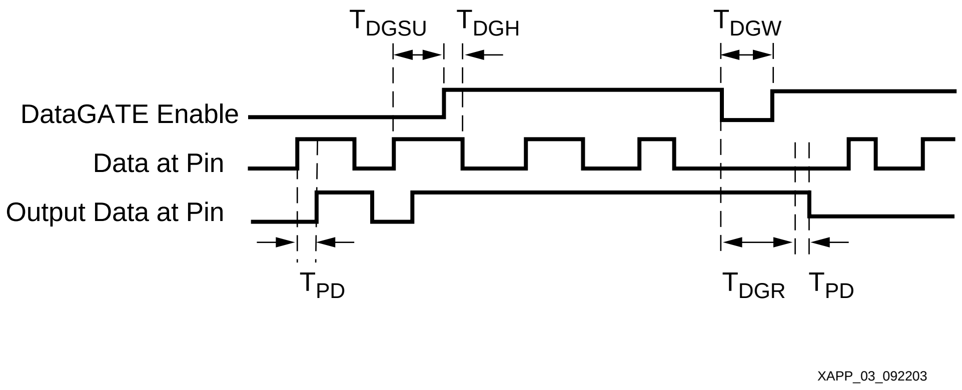


Figure 3: **DataGATE Timing Parameters**

Figure 3 details the timing action of the DataGATE operation. Specifically, the DataGATE Enable pin, if driven from an external pin, can be viewed as a clocking type function. Given that, it has standard clock type timing parameters, a setup time (T_{DGSU}), hold time (T_{DGH}), a minimum width (T_{DGW}) and DataGATE recovery time (T_{DGR}). For clarity, Figure 3 also shows the propagation time (T_{PD}) from input to output of a simple signal. DataGATE operation is described in detail in XAPP378, which describes how to declare the appropriate conditions in ABEL, VHDL and Verilog. DataGATE pin participation is also described, so you can easily dictate which pins will participate in the operation.

Applications

Simple Power saving

Figure 4 shows the simplest DataGATE configuration, and possibly the most frequently used method. Here, a simple external signal is delivered on a pin to the CPLD, which connects that signal through the Advanced Interconnect Matrix (AIM) to the DataGATE macrocell. The DataGATE macrocell is a specific macrocell in each part, which can drive the assertion rail. The logic created at the macrocell is arbitrary, so for this example, we can assume it is a simple connection. Hence, when the external pin drives high, that signal drives the Assertion Rail high. When the external pin drives low, so goes the Assertion Rail. As discussed earlier, when driven low, participating and nonparticipating input pins will have their signals forwarded into the CPLD. When driven high, participating input pins will be blocked and latched, while nonparticipating pins will still pass their signals into the CPLD. It is possible for all input pins to be “gated,” so care should be used in choosing signals. Blocking clocks typically has the highest payoff, but also the highest risk. Figure 4 shows the case where an external condition decides when to save power.

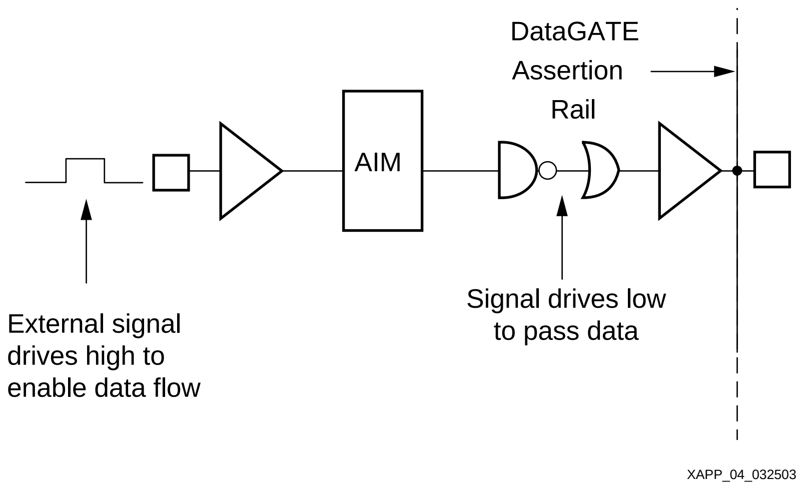

DataGATE
Assertion
Rail
AIM
External signal drives high to enable data flow
Signal drives low to pass data
XAPP_04_032503

Figure 4: **Simple External DataGATE Control Signal**

Timer

Figure 5 expands on the idea of Figure 4, including a timeout circuit (timer) that is employed to drive the Assertion Rail whenever the timeout signal asserts. We can assume that the timer delivers a “0” when the timer is counting clocks, but when it hits a previously chosen timeout period, it asserts a logical "1" and drives the blocking signal to the Assertion Rail. Because most timers are fairly simple counters, and frequently, it is not important to be too fussy about exactly which count value is chosen (ie, plus or minus a clock or two won’t matter), it may be smart to pick the simplest logic structure that can solve the timer problem. In all likelihood, this will be a Linear Feedback Shift Register (LFSR), which will produce most of the values of a binary counter, but in a non-consecutive order and with minimum logic beyond the macrocell flip flops. By either "And"ing the appropriate state, or choosing a state variable for direct connection to the Assertion Rail, the timer can drive the Assertion Rail as needed.

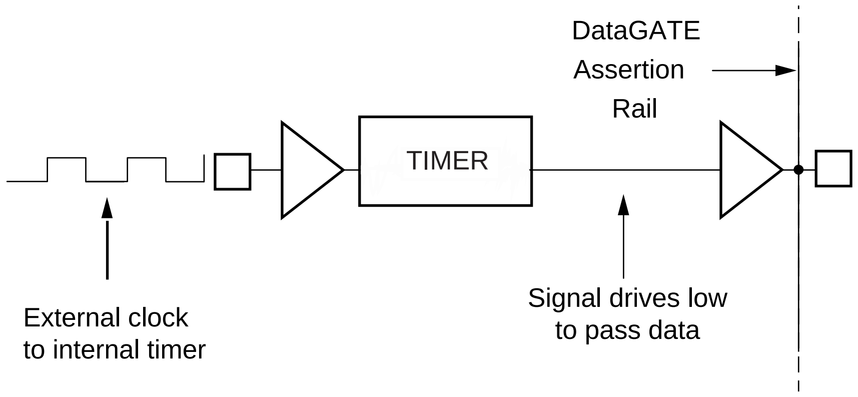


Figure 5: **DataGATE Under Control of Internal Timer**

It would also be possible to use the timer circuit to lock out signals that are deemed to be infrequently used, but periodically “wake up” to sample the pins for whatever reason that might be needed. Alternately, selected signals simply become permanently blocked at the completion of a timeout period. Another view would not have the counter counting “time,” but rather some other number of events that drive the “timer” clock input site—say, count 400 interrupt signals then block this set of pins.

Controller

As suggested earlier, any logic can be constructed to drive the Assertion Rail, so state machines can be created, too. In this sense, external variables can be combined with state variables to create the condition that drives the rail. As described with the timer, external signals can create a tally that is used to power down, but also, combinations of events and states can create the blocking event. Figure 6 shows the insertion of an arbitrary controller module, which includes flip flops and logic to interact with the external signals and clocks.

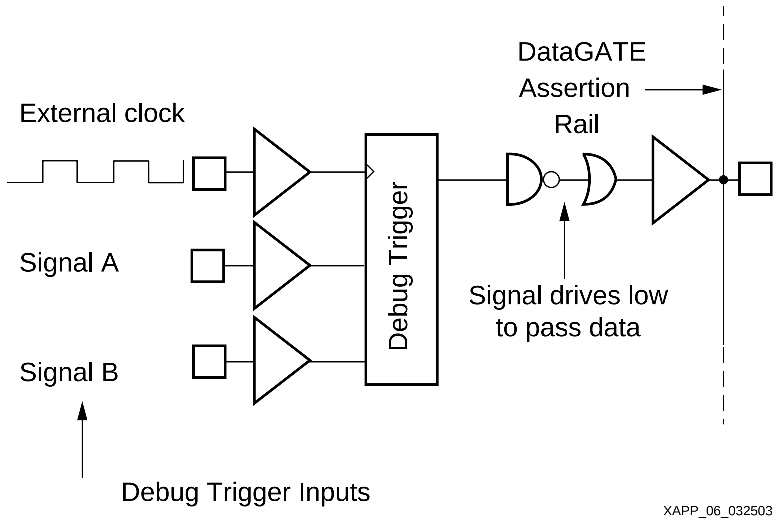


Figure 6: **DataGATE Rail Driven by Internal State Machine Controller**

Another important point to consider when using DataGATE is the overall time operation of a system. At initial power-on, certain actions occur in many systems—processor bootstrap, memory initialization, memory space and I/O space definition, etc. After these initial activities, many items remain in their initial condition and stay there throughout the operation. If certain registers or logic are only used at the beginning, but free running strobe signals, clocks, or data continue unnecessary switching, then those logic sections might benefit from “DataGating.”

Debugging with DataGATE

Latching the input pins when the Assertion Rail blocks inputs creates an additional capability: an on-chip debugger. Logic analyzers typically capture the contents of binary signals—the “state”—by triggering on an event, then snapshotting the data into a memory buffer for later readback. Combining the previously described controller with the JTAG circuitry inherent in a CoolRunner-II permits a simpler, but effective capability. For instance, assume that most of the pins are participating in the DataGATE operation. Assume next that we wish to isolate the state of the CPLD when a certain combination of inputs and internal CPLD states occurs—say an “error” state. By creating a state machine out of scrap gates and flops within the CPLD, the inputs can be blocked and held until the JTAG circuit reads back the entire state of the CPLD for dissection and analysis with a PC and JTAG cable. Alternately, the triggering event of a logic analyzer can be taken over to the CPLD and applied just as in the first, simple “power saving” model described above. Figure 7 gives some detail on how this might occur.

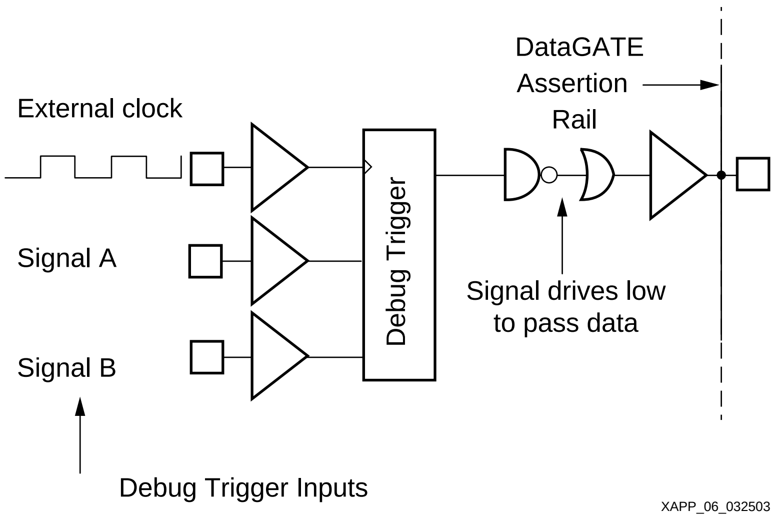


Figure 7: **Debugging with DataGATE**

This model of debugging is one way to use the input latches on the CPLD, although others view this as a very large set of input latches that can be controlled for simple data capture at the pins. The only drawback to this is that the timing for the data capture tends to increase on the larger parts, so users should plan for a little extra time delay.

Hot Plugging with DataGATE

Hot Plugging, Hot Socketing and Live Insertion are all terms that are used interchangeably. The standard situation is this: there is a rack of equipment that is powered up, and a user needs to insert a board into that rack without turning the power off. The methodology for this is somewhat nebulous because expectations vary widely between engineers: depending on how much they know about the situation, engineers all expect different behavior . Some expect the board insertion to go perfectly, without any control signals being disturbed or data bits being dropped. Others expect the board insertion to result in possible data corruption, but not in any catastrophic crashing of the systems. Experienced designers cross their fingers and are satisfied if the event doesn’t result in the rack catching on fire.

A user’s expectation should be that there will be no actual damage, but that extra care must be taken to prevent corrupt behavior. For instance, all the devices on the “cold board” are uncharged where most of the elements within the “hot” rack are charged up (bus lines, decoupling capacitors, whatever). Inserting a discharged board presents a substantial capacitive load to a system that has finite charge on it. This will require charging the “cold” board to the appropriate level before operation can occur. One solution to this has been the “extended finger” method whereby little extension tabs are placed on the cold board, so electricity is delivered to the board before the logic enters the rack connectors; however, this requires advanced planning to make the little tabs. We present an alternative method which uses the CoolRunner-II DataGATE feature to eliminate corrupt data.

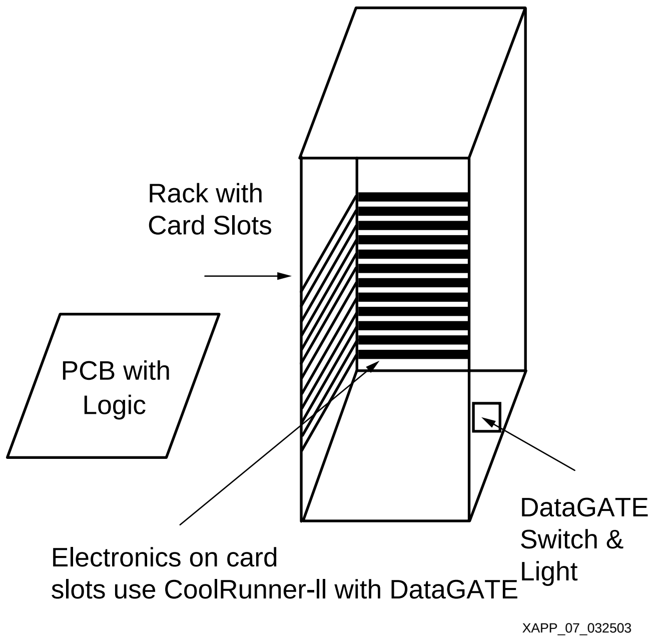


Figure 8: **Hot Plugging with DataGATE**

For this description, we assume all cards in the system, including the cold one to be inserted, have CoolRunner-II CPLDs attached to the slot connector pins. Also, all of the rack boards are connected to a control signal, which is either located on the rack or controlled by software on a card within the rack. The control signal can come from an internal processor signal, or from a switch on the rack (Figure 8). Before the hotplug event occurs, the control signal asserts to make all inputs within the rack latch their current state until the plug completes and the control signal releases. The release of the control signal should occur after the cold board is warmed up and initialized. The system then proceeds to operate. Depending on the system requirements, additional buffering or protocol actions should be included to account for the brief pause in the operation.

Security

Figure 9 shows another DataGATE application. In this case, the CPLD is involved in securing some aspect of the system operation. For simplicity, let's assume a password is involved in the operation. The password is delivered into the CoolRunner-II CPLD, which—if the password is correct—permits other signals to continue entering the device. If the password is incorrect, the DataGATE blocks all the signals coming into the CPLD to forbid future trials. Designing the rest of the CPLD to perform some "mission critical" aspect of the whole system is critical here, so the system won't work while the chip is being gated. To attempt more passwords, the whole system must be power cycled. A more permanent action would be to erase the CPLD if the password doesn't match, but doing so would use the On the Fly Reconfiguration capability instead of only DataGATE.

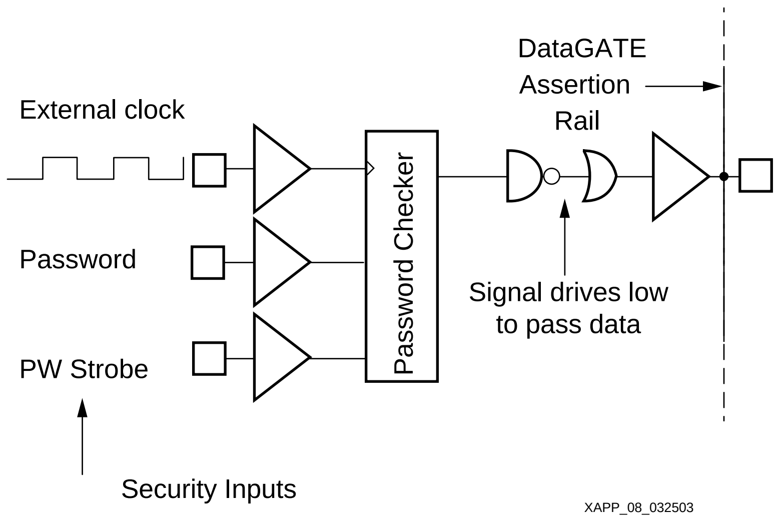


Figure 9: **DataGATE Dynamic Security**

Conclusions

DataGATE is a versatile and helpful feature designers can utilize when designing with CoolRunner-II CPLDs, although they are not required to use it and it can be used only over chosen pins, as needed.

A simple strategy for using DataGATE is to design with no initial regard for it, except in saving the macrocell that drives the assertion rail. Once the design is complete, evaluate the current drawn against the target current budget. Next, consider implementing CoolClock or other power savings options. Again, evaluate consumed current against target current. Review XAPP 377 for other things to try.

Next, if the design does not hit its target, or you simply wish to see how low the current can be taken, *then* consider using DataGATE. Consider sections of your design that initially “wake up” and perform active behavior, then go “quiet” for the rest of the operation. Assuming they are driven from active pins, can you identify some time or event that would permit them to be blocked by the DataGATE circuit? If so, define the time or event, create a logic circuit to manifest at that time and have it drive the DataGATE rail. Evaluate the current again. If you need less current, can you include more pins? Can you slightly change the block point (in time) to include more pins? If so, alter your design and re-evaluate. Should you encounter a “tweak” that adversely affects your design, you can back up a revision to the last working one and know that it is better than your original solution.

Additional Reading

CoolRunner-II Data Sheets, Application Notes, and White Papers

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/01/03	1.0	Initial Xilinx release.
05/15/03	1.1	Minor revisions
09/22/03	1.2	Fixed Figure 3.

WP227 (v1.1) June 29, 2005

The Real Value of CoolRunner-II DataGATE

By: Mark Ng

DataGATE™ is a CoolRunner™-II CPLD feature that permits input signal blocking, stops input switching, and reduces power. DataGATE can block any input pins you select. Low power design can be attained without using DataGATE, but even greater results are possible for your entire design using it. With DataGATE, CoolRunner-II devices are the only CPLDs on the market that can quote a low standby current and have it actually mean something. This white paper will demonstrate the dramatic results that can be obtained for your design using DataGATE.

Introduction

No other CPLD approaches specified standby current without using external logic to block switching inputs. Adding external logic increases system power and cost. Data sheet statements about static current are simply incomplete, and potentially useless or dangerous. You cannot measure static current without external modification. This has little value, in real world circuits, except to state: *If all inputs were stopped, then current drawn would be X microamps.*

Unfortunately, for real world designs this statement has little meaning. To gain the benefits of low static current may require massive design changes. DataGATE gives you additional power reduction using no external resources.

How Good is DataGATE?

It's excellent! The results you get will depend on the design and how DataGATE is used. To clarify, we will provide guidelines for best results. Table 1 shows how much V_{CCINT} current is saved under various input blocking conditions. As shown in the first row of Table 1, the standby current (defined as the total amount of current drawn at 0 MHz) of this particular XC2C128 unit under test is 0.02 mA. Without DataGATE, current increases linearly as the number of inputs switching increase.

However, with DataGATE the CPLD can approach standby current without forcing all inputs to stop switching. DataGATE allows up to 99% power savings. Other CPLDs can try to specify a meaningless standby current, but CoolRunner-II is the only CPLD that can actually save power in a real world design, one that has actual inputs and outputs, and interacts with other devices.

V_{CCINT} Current Savings

The current savings on V_{CCINT} are substantial, as shown in Table 1.

Table 1: **Current Drawn on V_{CCINT} from Switching Inputs with/without DataGATE at 50 MHz.**

	Current Drawn on V_{CCINT} (mA)		
Inputs Switching	**No DataGATE**	**With DataGATE**	**Savings**
0	0.02	0.02	0%
1	0.82	0.02	97%
2	1.62	0.02	98%
3	3.09	0.02	99%
4	5.40	0.02	99%

Figure 1, Figure 2, Figure 3, and Figure 4 graphically show how V_{CCINT} current savings increase versus input signal frequency.

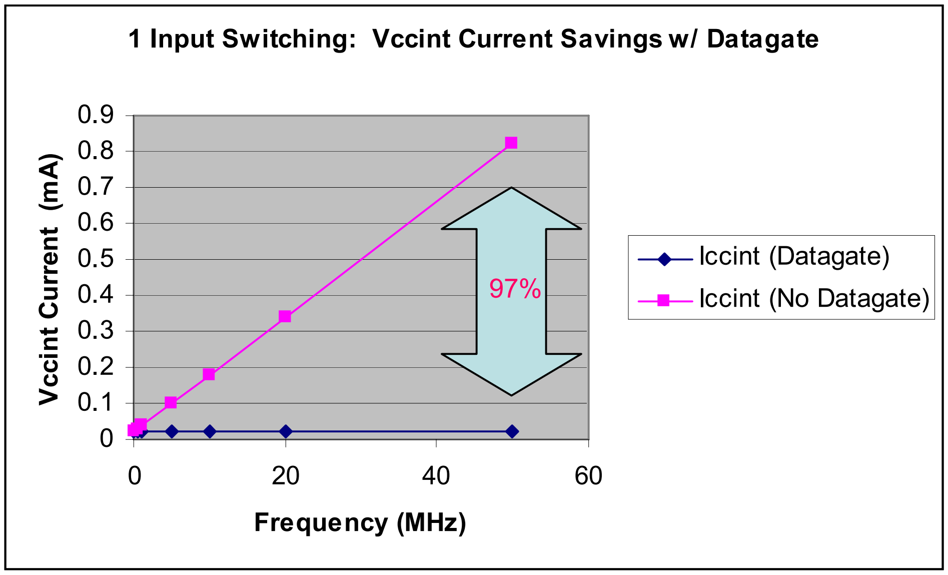


Figure 1: **V_{CCINT} Current Savings, Single Input Switching**

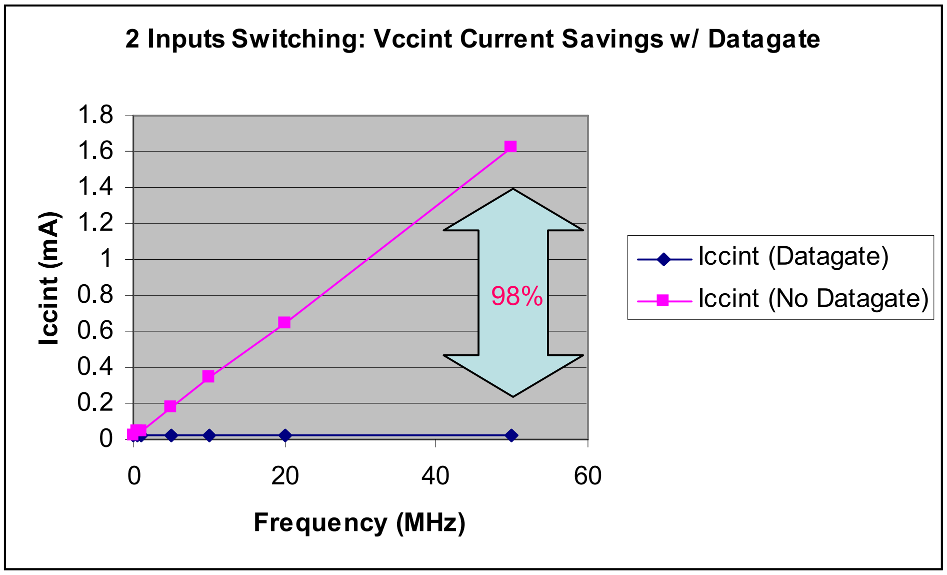


Figure 2: **V_{CCINT} Current Savings, Two Inputs Switching**

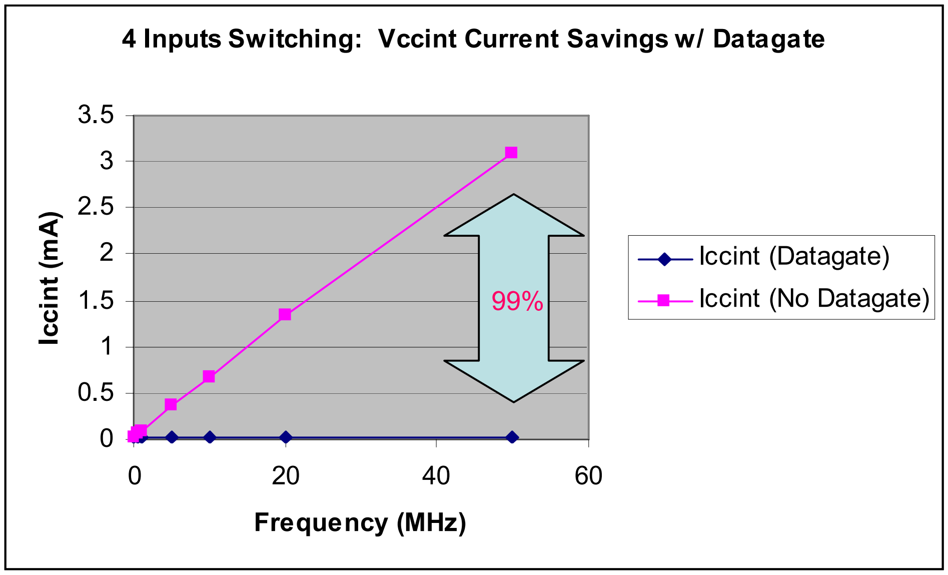


Figure 3: **V_{CCINT} Current Savings, Four Inputs Switching**

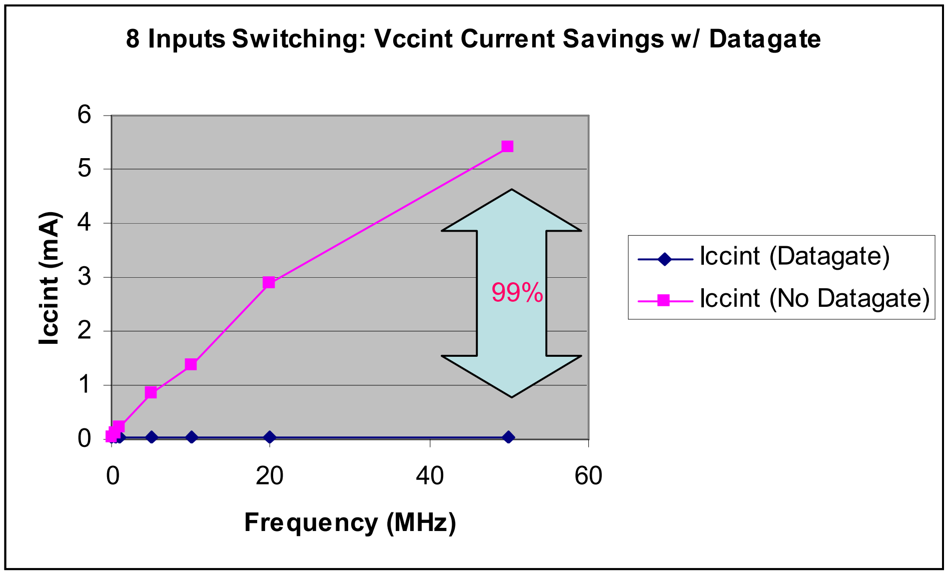


Figure 4: **V_{CCINT} Current Savings, Eight Inputs Switching**

What about V_{CCIO} Current?

At this point, we have established that a 'Standby Current' specification is relatively useless. No real design can operate with zero inputs toggling. We have also established that CoolRunner-II is the only CPLD in the world that allows a user to approach standby current without physically disconnecting all inputs. But all discussion thus far has concentrated on current drawn through the V_{CCINT} rail. What about current drawn through V_{CCIO}?

The entire industry avoids discussing current drawn through V_{CCIO}. No PLD manufacturer provides any specification whatsoever regarding how much current the I/Os will draw. Examine any CPLD data sheet. You will find that all I_{CC} versus Frequency graphs are specific to V_{CCINT} current. No reference is ever made to V_{CCIO}.

Why? This is primarily because current drawn through the I/Os is difficult for manufacturers to determine because it is dependent upon too many external variables (capacitive loading, frequency, current requirements, input rise time, and so on). In addition, as V_{CCIO} current can be significant (so significant that it can invalidate a device's low power message), most manufacturers have tended to avoid it.

Let's examine the effect of simply switching a few inputs. This time, instead of looking at V_{CCINT}, let's look at V_{CCIO}. How much current does that draw, and, can DataGATE do anything to reduce V_{CCIO} current?

Table 2: **Current Drawn on V_{CCIO} from Switching Inputs with/without DataGATE at 50 MHz**

	Current Drawn on V_{CCIO} (mA)		
Inputs Switching	**No DataGATE**	**With DataGATE**	**Savings**
0	0	0	0%
1	8.58	0.05	99%
2	14.86	0.11	99%
4	25.95	0.22	99%
8	43.82	0.44	99%

As can be seen in Table 2, this V_{CCIO} current can be quite large. However, with DataGATE asserted, the CoolRunner XC2C128 device can essentially shut down the internal I/O buffers and accomplish 99% power savings on the V_{CCIO} rail. Figure 5, Figure 6, Figure 7, and Figure 8 show V_{CCIO} current savings versus input switching frequency.

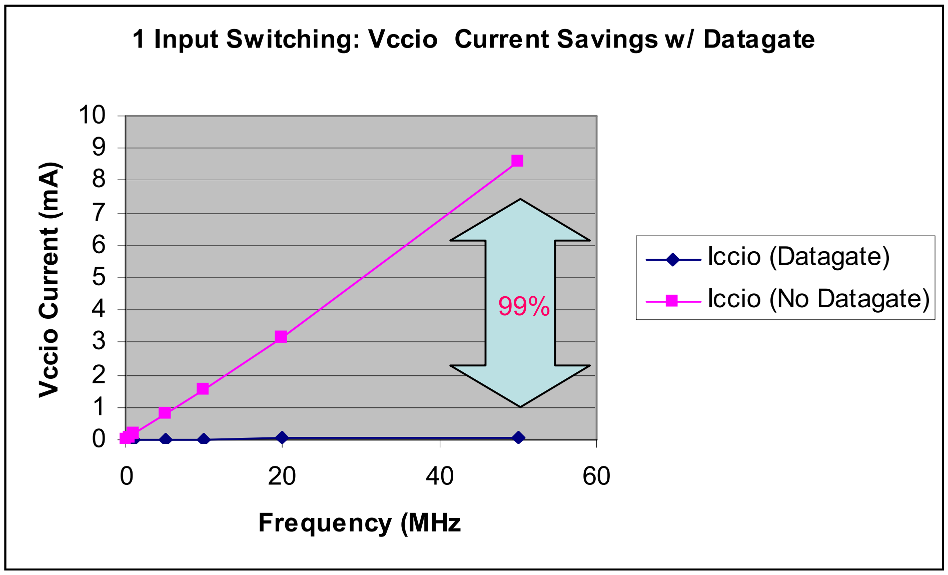


Figure 5: **V_{CCIO} Current Savings, Single Input Switching**

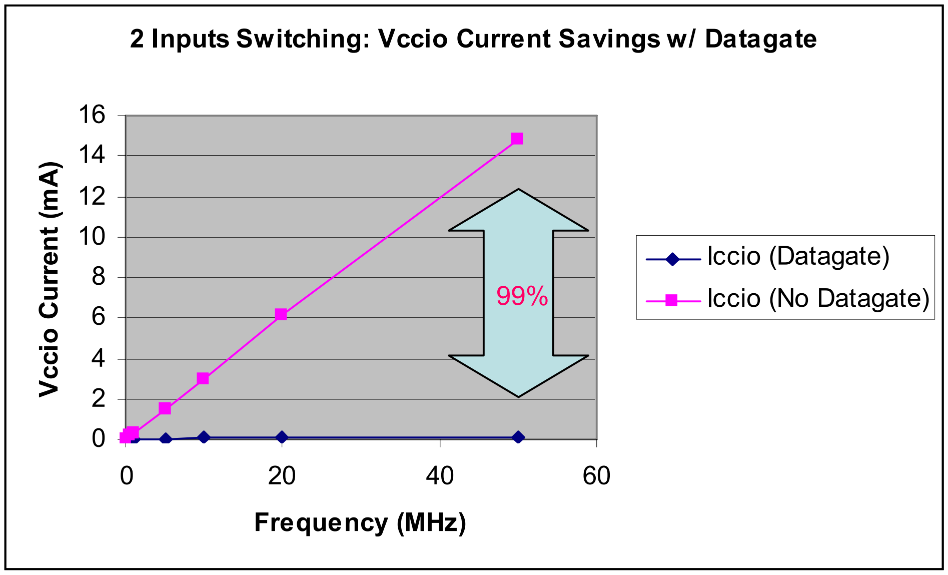


Figure 6: **V_{CCIO} Current Savings, Two Inputs Switching**

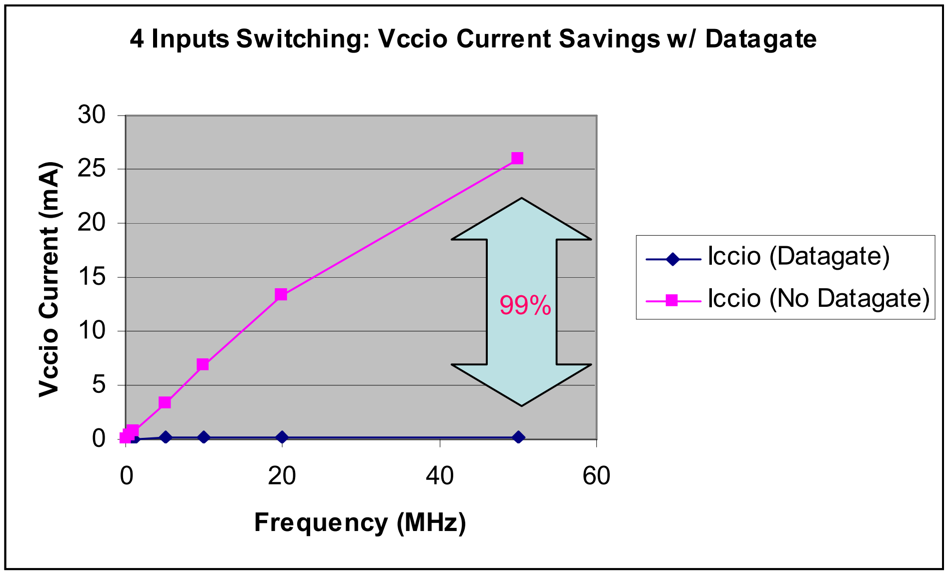


Figure 7: **V_{CCIO} Current Savings, Four Inputs Switching**

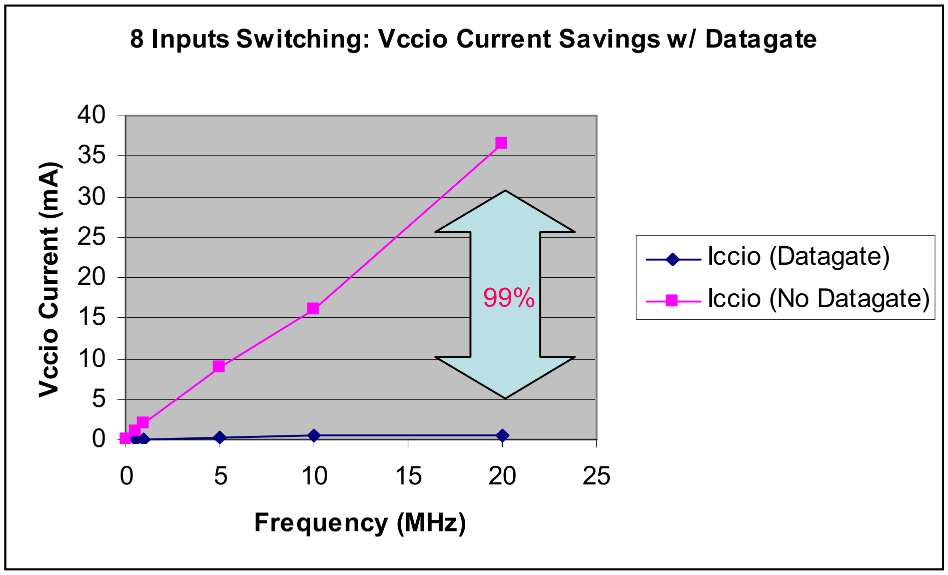


Figure 8: **V_{CCIO} Current Savings, Eight Inputs Switching**

We have already demonstrated one of the greatest kept secrets in the CPLD world -- although V_{CCINT} dynamic current can be quite low, V_{CCIO} current can exceed V_{CCINT} current by as much as 4x! It is no wonder that manufacturers do not specify V_{CCIO} current. Instead, they focus on what appears to be an extremely low standby current, which is basically useless. They focus on dynamic V_{CCINT} current, which is substantially less than V_{CCIO} current.

Other devices may appear to be low power, but only one device actually is low power. CoolRunner-II devices are the only CPLDs providing 99% savings on V_{CCINT} and 99% power savings on V_{CCIO}.

Conclusion

Table 1 and Table 2 understate the problem. Data was taken with simple buffers, showing the effect of blocking inputs into the chip. If those inputs connected to multiple sites within the CPLD, additional power would be drawn, driving the capacitance of the additional connections. Hence, the more complex the design, the more power is saved by blocking inputs. Because we cannot anticipate how much logic your inputs will drive, it is difficult to estimate how much current will be saved for a particular design. However, one thing is certain: CoolRunner-II is the leading low power device not only because it has low dynamic power consumption, but also because it is the only CPLD that allows a design to approach standby current during full operation.

Additional Resources

Xilinx has invested considerable time in developing the best ways to reduce power in digital systems that use our parts. The following documents give an idea of the many ways to approach the problem, so please become familiar with them as you select the methods that work best for you.

XAPP 395 describes how DataGATE works and outlines a general approach for reducing your power. Briefly, you simply create your design as you wish and measure your power (typically I_{CC}). Then, you identify signals that may be blocked, and redefine your design to block them with the DataGATE signal. You then measure your current and see if enough reduction occurs. If it works correctly and you wish to remove more current, block some more signals. Keep blocking and measuring to reduce current. If you block signals to the extent that the design no longer works, simply go back one step to the last point that worked. There are other approaches, but this one works well.

XAPP378 shows how to drive the design software to take advantage of the CoolRunner-II advanced features. DataGATE is one of many such features, as are advanced clocking (division, DualEDGE), Schmitt trigger inputs, and slew rate control.

XAPP436 shows how a CoolRunner-II CPLD can reduce power in other chips, along with the CPLD itself. This approach uses DataGATE to block switching signals that are not needed in the other chip, and contributes to the overall system power usage. If you are using CoolRunner-II as a level translator, you get DataGATE power management for free on devices of 128 macrocells and above. This application note explains how.

XAPP 377 shows a set of low power design practices, including DataGATE. There are many ways to reduce power, and Xilinx CPLDs show more ways to do that than any other competitor.

If you are not interested in measuring power, XAPP 317 shows how to apply the CoolRunner-II power equation to arrive at a reasonable estimate of your application's power usage. Just working with the equation factors can provide insight into ways to reduce it, as well.

Finally, if you want to understand the deeper workings, U.S. Patent #6,172,518 goes through the original approach for DataGATE. It was originally invented for the Xilinx XC9500/XL/XV family of CPLDs, but was only used in the CoolRunner-II Family, where dramatic power reduction would be most appreciated.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/30/05	1.0	Initial Xilinx release.
06/29/05	1.1	Web Publication

XAPP436 (v1.2) September 28, 2005

Managing Power with CoolRunner-II CPLDs

Summary

This application note demonstrates how multiple devices, including Virtex^{TM}-II, Spartan^{TM}-3, and Spartan^{TM}-3L FPGAs, can be effectively power managed by a single CoolRunner^{TM}-II CPLD. It is written with battery powered applications in mind.

Introduction

All chips draw power, but some applications are more sensitive than others to the amount drawn. Portable applications are sensitive simply because they draw from a battery. Most digital chips are designed to operate at 5V, 3.3V, 2.5V, 1.8V and so forth. This does not match well with today's battery voltages. Hence, there will be a regulator or two on most boards. Managing power will involve managing those regulators.

CoolRunner-II CPLDs were designed to operate with a core voltage of 1.8V, well suited to its 0.18 micron core, but its I/O structure supports 3.3V, 2.5V, 1.8V and 1.5V operation. Being standard low power CMOS, the I/Os also operate within that range, but are only speed specified at those voltages.

CoolRunner-II CPLDs have been successfully used as voltage translators, logic collection sites and even power management solutions. Their "early on" behavior makes them ideal for managing other chips' power. This application note focuses on using CoolRunner-II to manage power for Xilinx FPGAs, providing greater FPGA utility in a portable products. Additional CoolRunner-II qualities are shown to add value to reducing the portable product power budget in many cases.

The Power Equation

As expected, the starting point is physics, but CMOS power consumption can be deceptive. Simplistically, it follows an equation relating the switching speed of a logic gate output, times the voltage range over which it swings, times the load capacitance being driven. Adding in the static leakage component says it all.

Equation 1: $\boldsymbol{Power = V(VCF) + static\ power}$

where:

V = output voltage swing

C = load capacitance

F = gate output switching frequency

Static power = somewhat constant value for many parts (often negligible)

This equation is a guideline. Originally developed for a simple CMOS inverter, it shows a trend, but only suggests how to reduce power. Equation 1 drives most CMOS power estimation tools. Xilinx provides several ways to estimate power, including spreadsheets, application notes and the XPower software, available with the ISE design suite.

The CoolRunner-II Solution

The CoolRunner-II approach to power management attacks both static and dynamic current. A battery is a charge reservoir with varying capabilities on voltage regulation, and in some cases an ability to be recharged. Each battery has its own qualities for delivering charge. Many batteries can deliver high currents for brief time periods, or smaller currents for longer times. Trade-offs can and must be made.

To get a feeling for power activity, let's look at a real world example, that was documented by Portelligent in their Report #116.02- 031023-1d. Figure 1 is a reconstruction of their measurements made on an LG cell phone that was able to record video or take a picture. Figure 1 shows power use while taking a photograph, with the cell phone camera.

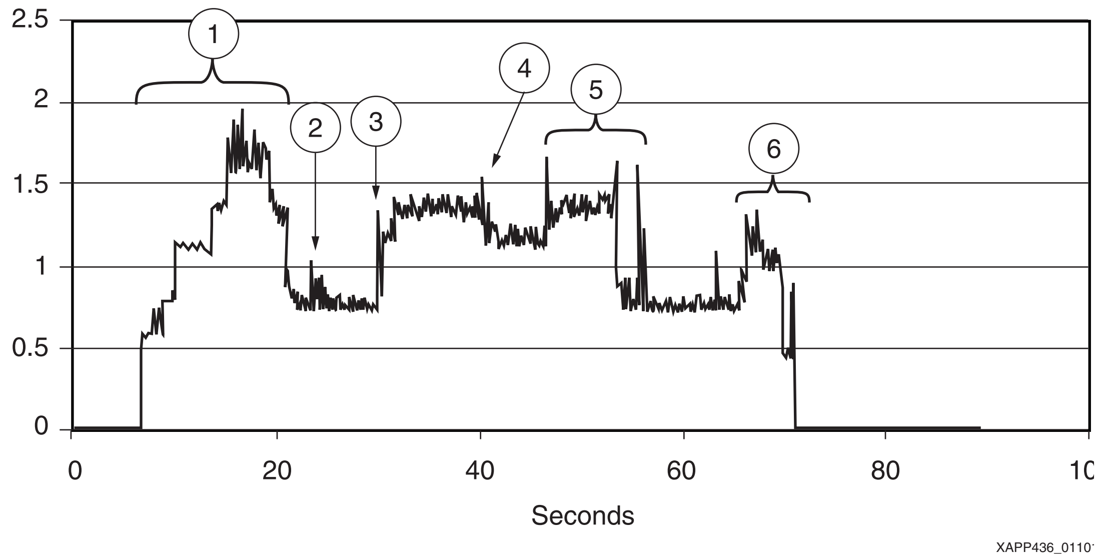


Figure 1: **Power Profile of LG Cell Phone Taking Still Photograph**

The steps in taking a still photo are summarized as:

1. Power up and turn on the LCD backlight
2. Select the camera from menu
3. Select taking a still photo
4. Take the photo
5. Return to the main menu
6. Power down

The numbers in the diagram correspond to the activity list above. The Portelligent report explains the activity of the various chips within the camera, but Figure 1 shows the composite power. There are many variables at play. Several chips within the camera receive power at different times. The display, camera chip and memories are involved here, but none of the standard cell phone capabilities - like making a call. That requires a different profile. Nonetheless, these actions are typical of a large number of today's portable systems ranging from, walkie-talkies to PDAs, to video cameras, to the emerging software defined radios (SDR). 3G cell phones, in particular, with their added on applications and multi-band operation take this type of power management to new heights. In fact, these systems may rival some of the very power conscious projects like the Mars rover or other deep space probes in complexity. Power management is vital to all of these products.

Now, let's shift our attention to Figure 2 which shows a small system using two FPGAs, an ASSP and a CoolRunner-II CPLD. Note, there are also multiple regulator chips shown (LDO), deriving their voltages by regulating down 5 volts. Another approach would be to regulate down from a battery, which most regulators can manage.

This particular example has the FPGAs being powered through the LDOs, but the ASSP simply connects to the distributed supply through a power FET. The particular FET will depend on the channel drop that can be tolerated, the turn on voltage and the power requirements of the ASSP. All power sources are controlled from logic signals coming off CoolRunner-II pins. The idea here is simple. When CoolRunner-II remains turned on, its power consumption is only in microamps (standby), whereas the FPGA current ranges into milliamps, depending on family and density.

In this example, we have combined a 150 nanometer Virtex-II FPGA and a 90 nanometer Spartan-3 FPGA. Each has different power needs, and illustrates choices and trade-offs to be made. As we will see later, the CoolRunner-II will be able to introduce an extra power down mode to the FPGAs, to reduce dynamic power. That will give a choice on whether to turn the part off, and pay for reconfiguration again, or just reduce dynamic power during standard operation. A reasonable regulator for the LDO 2 module is the TPS75003 from Texas Instruments^{TM}.

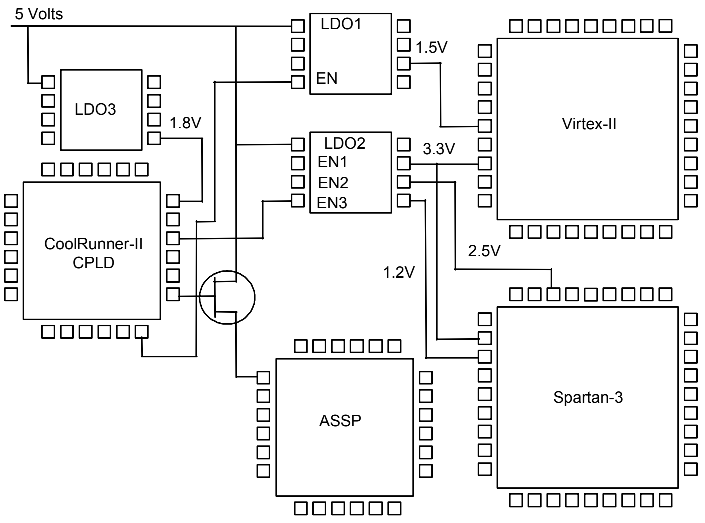


Figure 2: **Multiple Chips "Power Managed" by a CoolRunner-II CPLD**

The Virtex-II FPGA has reduced power up surge, but larger density parts may draw more current than a portable budget allows. Each FPGA might be a candidate for shutdown, except when needed. Table 1 summarizes quiescent internal current drawn by various Virtex-II family members. The dynamic current is a function of signal switching rates within the parts, and is design dependent. To estimate the power, you can use any of the methods mentioned earlier, or simply build up the design and measure the requirement. Additional estimation resources are listed at the end. Table 2 gives similar data for Spartan-3 FPGAs.

Table 1: **Virtex-II FPGA Typical Internal Quiescent Currents**

Virtex-II Device	Typical Internal Quiescent Supply Current[1]
XC2V40	3 mA
XC2V80	5 mA
XC2V250	8 mA
XC2V500	10 mA
XC2V1000	12 mA

1. Refer to the data sheet for the most accurate and up-to-date information

Table 2: **Spartan-3/L FPGA Typical Internal Quiescent Currents**

Spartan-3/L Device	Spartan-3 Typical Internal Quiescent Supply Current[1]	Spartan-3L Typical Internal Quiescent Supply Current[1]	Spartan-3L Quiescent[1] Max (Hibernate Mode)[2]
XC3S50	10 mA		
XC3S200	20 mA		
XC3S400	35 mA		
XC3S1000/L	65 mA	30 mA	6 mA
XC3S1500/L	65 mA	50 mA	8 mA

1. Refer to the data sheet for the most accurate and up-to-date information
2. Hibernate Mode available for Spartan-3L. **See DS313**.

Figure 3 shows how a current profile might look (averaged) over time. Usually, there is an initial surge as all on board capacitance becomes charged, various parts undergo configuration, initialization, bootstrapping, and so forth. Then things settle down, and various chips can be turned off, placed into low power, or whatever, as dictated by the application. The average current draw of this profile is substantially less than the initial surge value, so there may be a payoff for turning chips off all together, or placing them in a low power mode. Table 3 and Table 4 give configuration times in byte wide mode at maximum speed.

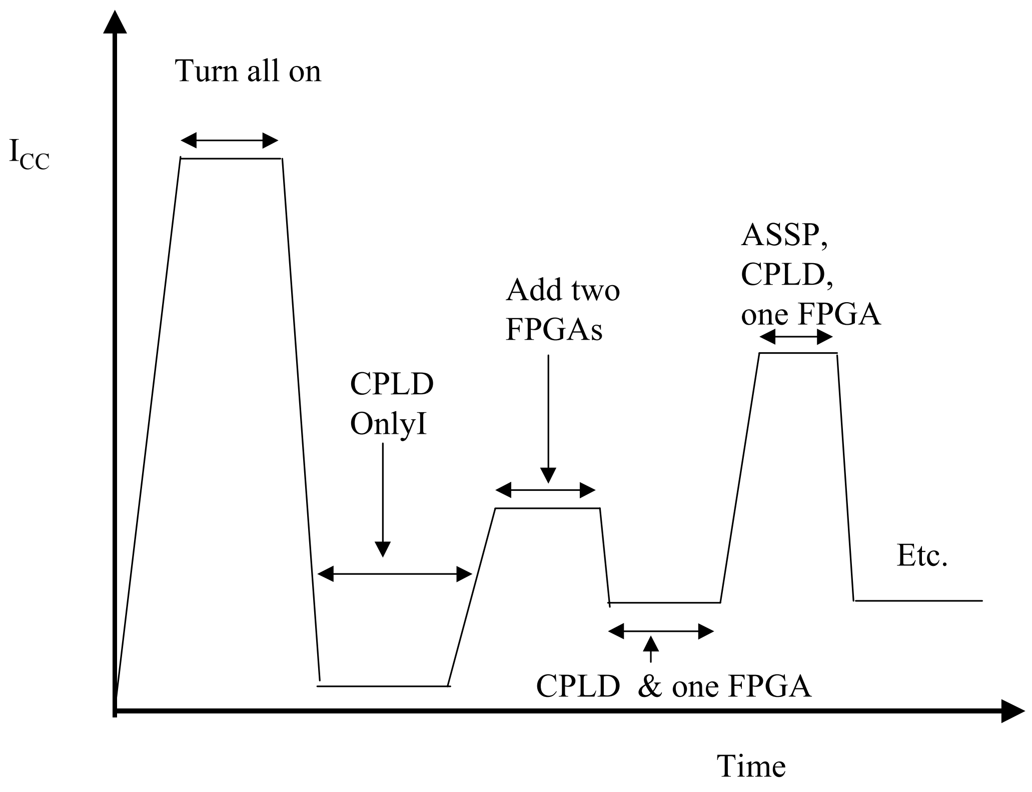


Figure 3: **Current Profile of a System with Various Active Devices Over Time**

Table 3: **Virtex-II Configuration Parameters**

Virtex-II Device	Configuration Bits	Configuration Time in Microseconds (at 50 MHz)
XC2V40	338,976	42.372
XC2V80	598,816	74.852
XC2V250	1,593,632	199.204
XC2V500	2,560,544	320.068
XC2V1000	4,082,592	510.324

Table 4: **Spartan-3 Configuration Parameters**

Spartan-3 Device	Configuration Bits	Configuration Time in Microseconds (at 50 MHz)
XC3S50	439,264	54.9
XC3S200	1,047,616	130.95
XC3S400	1,699,136	212.392
XC3S1000	3,223,488	402.936
XC3S1500	5,214,784	651.848

By lowering the average power, CoolRunner-II CPLDs can dramatically extend the battery life of a system in a way that brings the high flexibility and value of FPGAs into the portable world. Let’s show how more value is gained using CoolRunner-II DataGATE.

DataGATE

DataGATE was designed to stop unwanted input switching from continuously draining power in CoolRunner-II CPLDs. Additional applications evolved from testing to security, and are documented in the Advanced Features and DataGATE application notes. However, one additional application is simply to “DataGATE“ other chips.

Figure 4 shows how the DataGATE feature works. A metal rail (DataGATE Assertion Rail) circles the whole chip inside, near the pins. Each input site provides a place where the received signal can be blocked by a pass transistor, depending on two conditions. The first condition is an enable bit, selecting that pin to participate in the DataGATE decision. The second condition is simply whether the DataGATE Rail is asserted. If the rail is asserted and that input’s participation selected, the input signal is blocked from penetrating the chip, until the rail releases assertion. It’s that simple. When the rail asserts, blocking follows immediately. The previous input level automatically latches, so static CMOS logic signals forward into the CPLD core. The signal freezes until released. When the rail releases, switching action resumes.

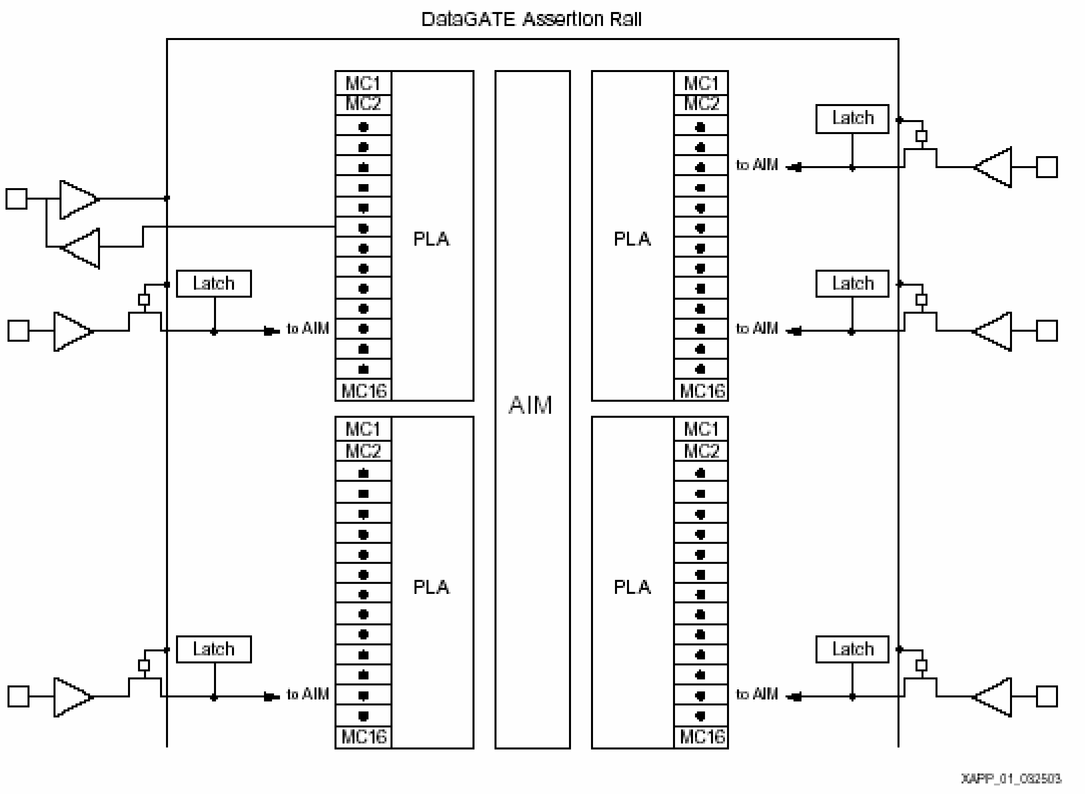


Figure 4: **DataGATE Architecture**

Figure 5 shows a close-up of how the pass transistor, enable cell and latch all connect to automatically block and freeze input signals that forward through the CoolRunner-II core.

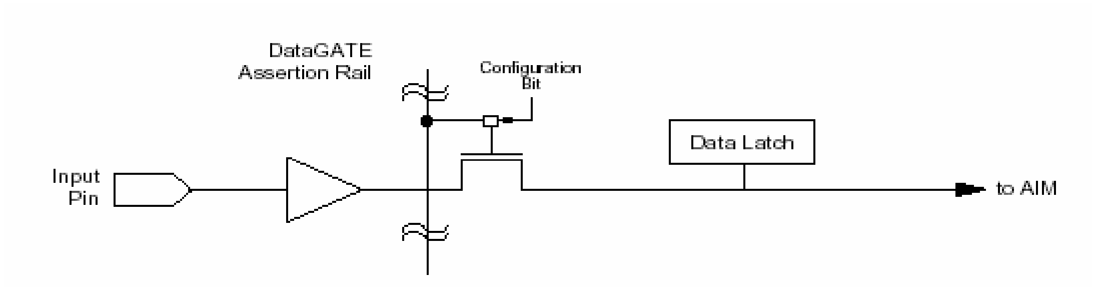


Figure 5: **Close-up of DataGATE Switch Mechanism**

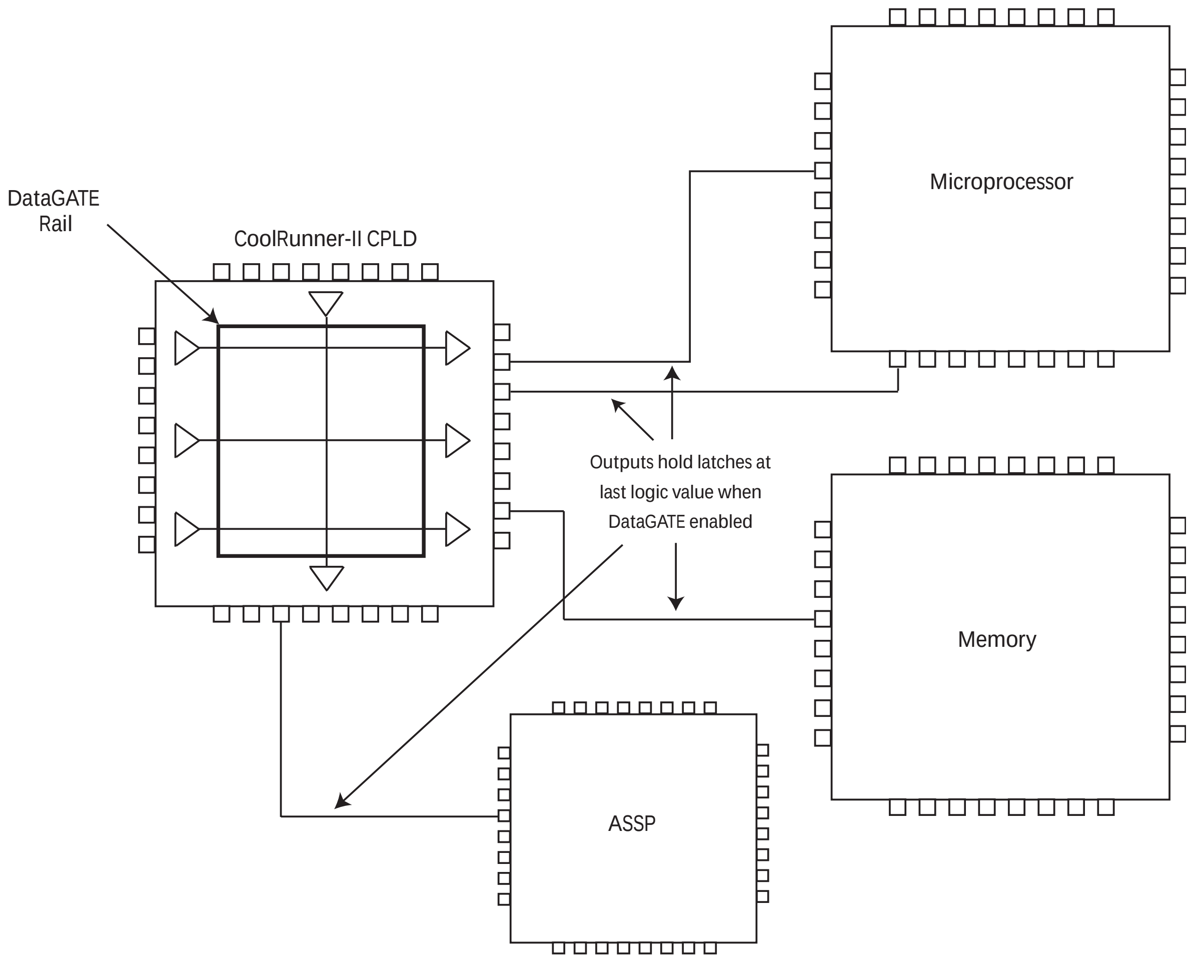


Figure 6: **DataGATE Blocking Switching Activity to Other Chips**

Figure 6 shows how signals passing through the CoolRunner-II DataGATE freeze signals to other chips. In this situation, we passed signals through the CoolRunner-II part, directly to the outside, where they drive to other chips. These will be held at the logic level that was last on the input pad, when DataGATE asserts. Should a 3-state signal be forwarded through the CoolRunner-II output pins, it will naturally be pulled to a high or low value, by the weak keeper that is on them, thereby covering the case when a “frozen” output gets 3-stated by another condition. An interesting proposition of passing signals through the CoolRunner-II is that it might be done as a natural side product of simply translating voltage levels on signals as they interface through the CPLD. The DataGATE action, is really for free under those conditions, which occur frequently.

DataGATE is offered on CoolRunner-II CPLDS that have 128, 256, 384 and 512 macrocells. This extends into the hundreds of I/O pins that can be blocked as needed by the DataGATE facility. If multiple sets of pins need to be blocked, under different circumstances, then multiples of the smaller parts can be used to cover the number and condition needs of any given design.

At this point, we have not described what drives the DataGATE rail. It is very simple. One macrocell within a particular CoolRunner-II CPLD is designated as the “DataGATE” macrocell. It is identical to all other macrocells, which means any logic situation a designer sets up to drive that macrocell, asserts the DataGATE signal, if enabled in the design software. The DataGATE signal releases whenever the logic driving that macrocell dictates. An event as simple as a switching input can trigger the DataGATE macrocell, and an event as complex as a conditional state machine driving a timer can trigger the macrocell. Designers are free to dream up whatever they want. It is possible to block any input chosen, including clocks, but extreme care must be used when designing that way! All, none or any subset of the input pins can be blocked. Being a reprogrammable CPLD, this facility can be used experimentally to determine the best set of signals to “freeze” and the best set of circumstances to assert and release the rail.

So, how do you know what to “freeze?” That will vary, from system to system. Here’s an example. When a microprocessor first bootstraps, it frequently sends values from its databus into a CPLD. These might be address values being loaded into comparators to select ranges of memory and I/O devices. Once those registers are initialized they need only compare against the address lines to operate. The databus connections are never needed again, but are still attached. DataGATE lets designers identify the time when the connection is no longer needed, and eliminate the extraneous switching that draws unneeded current. To learn more about other things you can do with DataGATE, check the references at the end.

Conclusion

We have omitted some details. What is the power impact of unpowered I/O pins attached to powered up termination resistors? How much leakage per pin occurs if an unpowered pin is driven by a powered one? Many questions can only be answered by assessing the specific situations with the particular device’s data sheet in hand. We hope the methods described here will have some value by simply increasing your choices on power reduction methods.

CoolRunner-II CPLDs are designed to be inherently low power devices. Additional features within them – including DataGATE, can help other chips also reduce their overall power, when properly applied.

References

Portelligent Report #116.02-031023-1d

Estimation equation: http://www.xilinx.com/bvdocs/appnotes/xapp317.pdf

Low power design methods: http://www.xilinx.com/bvdocs/appnotes/xapp346.pdf

Decreasing power: http://www.xilinx.com/bvdocs/appnotes/xapp347.pdf

Accurate XPLA3 estimation: http://www.xilinx.com/bvdocs/appnotes/xapp360.pdf

Accurate CoolRunner-II Design estimation: http://www.xilinx.com/bvdocs/appnotes/xapp377.pdf

Powering CoolRunner-II: http://www.xilinx.com/bvdocs/appnotes/xapp389.pdf

DataGATE: http://www.xilinx.com/bvdocs/appnotes/xapp395.pdf

XPower http://www.xilinx.com/xlnx/xebiz/designResources/

Power Estimator (Spartan) http://www.xilinx.com/cgi-bin/power_tool/power_Spartan3

Power Estimator (Virtex-II) http://www.xilinx.com/cgi-bin/power_tool/power_Virtex2

Additional Information

CoolRunner-II Data Sheets, Application Notes, and White Papers

Device Packages

Spartan-3 Data Sheets, Application Notes, and White Papers

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/29/04	1.0	Initial Xilinx release.
03/30/05	1.1	Removed incorrect reference to power surge. Added Spartan-3L.
09/28/05	1.2	Corrected part number on TI regulator, page 3.

CoolRunner-II Data Sheets

This appendix contains the following data sheets:

- CoolRunner™-II Family Data Sheet
- XC2C32A Data Sheet
- XC2C64A Data Sheet
- XC2C128 Data Sheet
- XC2C256 Data Sheet

These data sheets are included as they are the most popular devices for use in portable consumer applications. The remaining CoolRunner-II CPLD data sheets, and all other Xilinx CPLD data sheets can be found on the Xilinx website at www.xilinx.com.

XILINX®

CoolRunner-II CPLD Family

DS090 (v2.5) June 28, 2005 **Product Specification**

Features

- Optimized for 1.8V systems
  - Industry's fastest low power CPLD
  - Densities from 32 to 512 macrocells
- Industry's best 0.18 micron CMOS CPLD
  - Optimized architecture for effective logic synthesis
  - Multi-voltage I/O operation — 1.5V to 3.3V
- Advanced system features
  - Fastest in system programming
    - 1.8V ISP using IEEE 1532 (JTAG) interface
  - On-The-Fly Reconfiguration (OTF)
  - IEEE1149.1 JTAG Boundary Scan Test
  - Optional Schmitt trigger input (per pin)
  - Multiple I/O banks on all devices
  - Unsurpassed low power management
    - DataGATE external signal control
  - Flexible clocking modes
    - Optional DualEDGE triggered registers
    - Clock divider (÷ 2,4,6,8,10,12,14,16)
    - CoolCLOCK
  - Global signal options with macrocell control
    - Multiple global clocks with phase selection per macrocell
    - Multiple global output enables
    - Global set/reset
  - Abundant product term clocks, output enables and set/resets
  - Efficient control term clocks, output enables and set/resets for each macrocell and shared across function blocks
  - Advanced design security
  - Open-drain output option for Wired-OR and LED drive
  - Optional bus-hold, 3-state or weak pullup on select I/O pins
  - Optional configurable grounds on unused I/Os
  - Mixed I/O voltages compatible with 1.5V, 1.8V, 2.5V, and 3.3V logic levels on all parts
  - SSTL2_1,SSTL3_1, and HSTL_1 on 128 macrocell and denser devices
  - Hot pluggable
- PLA architecture
  - Superior pinout retention
  - 100% product term routability across function block
- Wide package availability including fine pitch:
  - Chip Scale Package (CSP) BGA, Fine Line BGA, TQFP, PQFP, VQFP, PLCC, and QFN packages
  - Pb-free available for all packages
- Design entry/verification using Xilinx and industry standard CAE tools
- Free software support for all densities using Xilinx WebPACK™
- Industry leading nonvolatile 0.18 micron CMOS process
  - Guaranteed 1,000 program/erase cycles
  - Guaranteed 20 year data retention

Family Overview

Xilinx CoolRunner™-II CPLDs deliver the high speed and ease of use associated with the XC9500/XL/XV CPLD family with the extremely low power versatility of the XPLA3™ family in a single CPLD. This means that the exact same parts can be used for high-speed data communications/computing systems and leading edge portable products, with the added benefit of In System Programming. Low power consumption and high-speed operation are combined into a single family that is easy to use and cost effective. Clocking techniques and other power saving features extend the users' power budget. The design features are supported starting with Xilinx ISE 4.1i ISE WebPACK. Additional details can be found in **Further Reading**, page 201.

Table 1 shows the macrocell capacity and key timing parameters for the CoolRunner-II CPLD family.

Table 1: **CoolRunner-II CPLD Family Parameters**

	XC2C32A	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
Macrocells	32	64	128	256	384	512
Max I/O	33	64	100	184	240	270
T_{PD} (ns)	3.8	4.6	5.7	5.7	7.1	7.1
T_{SU} (ns)	1.9	2.0	2.4	2.4	2.9	2.6
T_{CO} (ns)	3.7	3.9	4.2	4.5	5.8	5.8
$F_{SYSTEM1}$ (MHz)	323	263	244	256	217	179

Table 2: **CoolRunner-II CPLD DC Characteristics**

	XC2C32A	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
I_{CC} (µA), 0 MHz, 25°C (typical)	16	17	19	21	23	25
I_{CC} (mA), 50 MHz, 70°C (max)	2.5	5	10	27	45	55

I_{CC} is dynamic current.

Table 2 shows key DC characteristics for the CoolRunner-II family.

Table 3 shows the CoolRunner-II CPLD package offering with corresponding I/O count. All packages are surface mount, with over half of them being ball-grid technologies. The ultra tiny packages permit maximum functional capacity in the smallest possible area. The CMOS technology used in CoolRunner-II CPLDs generates minimal heat, allowing the use of tiny packages during high-speed operation.

With the exception of the new Pb-free QF packages, there are at least two densities present in each package with three in the VQ100 (100-pin 1.0mm QFP) and TQ144 (144-pin 1.4mm QFP), and in the FT256 (256-ball 1.0mm spacing FLBGA). The FT256 is particularly important for slim dimensioned portable products with mid- to high-density logic requirements.

Table 3: **CoolRunner-II CPLD Family Packages and I/O Count**

	XC2C32(2)	XC2C32A	XC2C64(2)	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
QFG32(1)		21	-	-	-	-	-	-
PC44	33	33	33	33	-	-	-	-
PCG44(1)		33		33	-	-	-	-
VQ44	33	33	33	33	-	-	-	-
VQG44(1)		33		33	-	-	-	-
QFG48(1)	-	-	-	37	-	-	-	-
CP56	33	33	45	45	-	-	-	-
CPG56(1)		33		45	-	-	-	-
VQ100	-	-	64	64	80	80	-	-
VQG100(1)	-	-		64	80	80	-	-
CP132	-	-	-	-	100	106	-	-
CPG132(1)	-	-	-	-	100	106	-	-
TQ144	-	-	-	-	100	118	118	-
TQG144(1)	-	-	-	-	100	118	118	-
PQ208	-	-	-	-	-	173	173	173
PQG208(1)	-	-	-	-	-	173	173	173
FT256	-	-	-	-	-	184	212	212
FTG256(1)	-	-	-	-	-	184	212	212
FG324	-	-	-	-	-	-	240	270
FGG324(1)	-	-	-	-	-	-	240	270

Notes:
1. The letter "G" as the third character indicates a Pb-free package.
2. The XC2C32 and XC2C64 are not recommended for new designs. Use the XC2C32A and XC2C64A.

Table 4 details the distribution of advanced features across the CoolRunner-II CPLD family. The family has uniform basic features with advanced features included in densities where they are most useful. For example, it is very unlikely that four I/O banks are needed on 32 and 64 macrocell parts, but very likely they are for 384 and 512 macrocell parts. The I/O banks are groupings of I/O pins using any one of a subset of compatible voltage standards that share the same V_{CCIO} level. (See Table 5 for a summary of CoolRunner-II I/O standards.)

Table 4: **CoolRunner-II CPLD Family Features**

	XC2C32(2)	XC2C32A	XC2C64(2)	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
IEEE 1532	✓	✓	✓	✓	✓	✓	✓	✓
I/O banks	1	2	1	2	2	2	4	4
Clock division	-	-	-	-	✓	✓	✓	✓
DualEDGE Registers	✓	✓	✓	✓	✓	✓	✓	✓
DataGATE	-	-	-	-	✓	✓	✓	✓
LVTTL	✓	✓	✓	✓	✓	✓	✓	✓
LVCMOS33, 25, 18, and 15(1)	✓	✓	✓	✓	✓	✓	✓	✓
SSTL2_1	-	-	-	-	✓	✓	✓	✓
SSTL3_1	-	-	-	-	✓	✓	✓	✓
HSTL_1	-	-	-	-	✓	✓	✓	✓
Configurable ground	✓	✓	✓	✓	✓	✓	✓	✓
Quadruple data security	✓	✓	✓	✓	✓	✓	✓	✓
Open drain outputs	✓	✓	✓	✓	✓	✓	✓	✓
Hot plugging	✓	✓	✓	✓	✓	✓	✓	✓
Schmitt Inputs	✓	✓	✓	✓	✓	✓	✓	✓

1. LVCMOS15 requires the use of Schmitt-trigger inputs.
2. The XC2C32 and XC2C64 are not recommended for new designs. Use the XC2C32A and XC2C64A.

Architecture Description

CoolRunner-II CPLD is a highly uniform family of fast, low power CPLDs. The underlying architecture is a traditional CPLD architecture combining macrocells into Function Blocks (FBs) interconnected with a global routing matrix, the Xilinx Advanced Interconnect Matrix (AIM). The Function Blocks use a Programmable Logic Array (PLA) configuration which allows all product terms to be routed and shared among any of the macrocells of the FB. Design software can efficiently synthesize and optimize logic that is subsequently fit to the FBs and connected with the ability to utilize a very high percentage of device resources. Design changes are easily and automatically managed by the software, which exploits the 100% routability of the Programmable Logic Array within each FB. This extremely robust building block delivers the industry's highest pinout retention, under very broad design conditions. The architecture will be explained by expanding the detail as we discuss the underlying Function Blocks, logic and interconnect.

The design software automatically manages these device resources so that users can express their designs using completely generic constructs without knowledge of these architectural details. More advanced users can take advantage of these details to more thoroughly understand the software's choices and direct its results.

Figure 1 shows the high-level architecture whereby Function Blocks attach to pins and interconnect to each other within the internal interconnect matrix. Each FB contains 16 macrocells. The BSC path is the JTAG Boundary Scan Control path. The BSC and ISP block has the JTAG controller and In-System Programming Circuits.

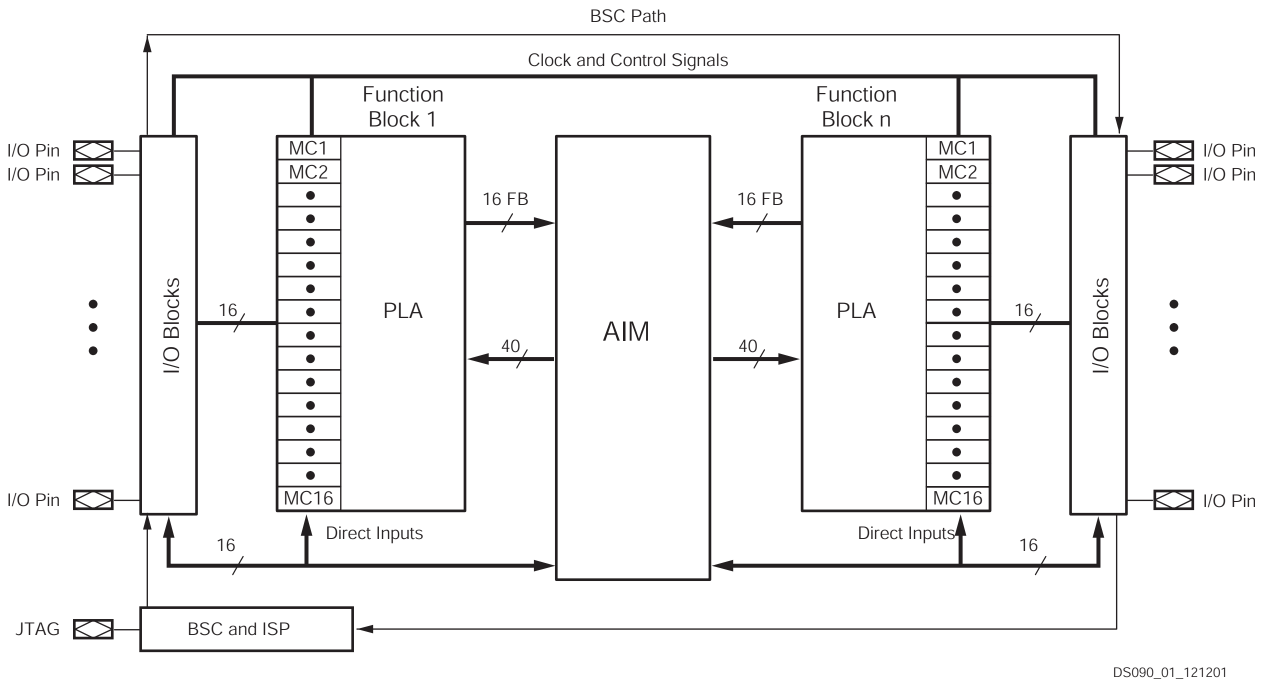


Figure 1: **CoolRunner-II CPLD Architecture**

Function Block

The CoolRunner-II CPLD Function Blocks contain 16 macrocells, with 40 entry sites for signals to arrive for logic creation and connection. The internal logic engine is a 56 product term PLA. All Function Blocks, regardless of the number contained in the device, are identical. For a high-level view of the Function Block, see Figure 2.

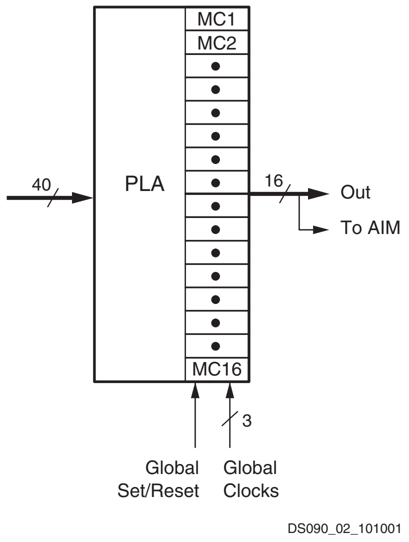


Figure 2: **CoolRunner-II CPLD Function Block**

At the high level, it is seen that the product terms (p-terms) reside in a programmable logic array (PLA). This structure is extremely flexible, and very robust when compared to fixed or cascaded product term function blocks.

Classic CPLDs typically have a few product terms available for a high-speed path to a given macrocell. They rely on capturing unused p-terms from neighboring macrocells to expand their product term tally, when needed. The result of this architecture is a variable timing model and the possibility of stranding unusable logic within the FB.

The PLA is different — and better. First, any product term can be attached to any OR gate inside the FB macrocell(s). Second, any logic function can have as many p-terms as needed attached to it within the FB, to an upper limit of 56. Third, product terms can be re-used at multiple macrocell OR functions so that within a FB, a particular logical product need only be created once, but can be re-used up to 16 times within the FB. Naturally, this plays well with the fitting software, which identifies product terms that can be shared.

The software places as many of those functions as it can into FBs, so it happens for free. There is no need to force macrocell functions to be adjacent or any other restriction save residing in the same FB, which is handled by the software. Functions need not share a common clock, common set/reset or common output enable to take full advantage of the PLA. Also, every product term arrives with the same time delay incurred. There are no cascade time adders for putting more product terms in the FB. When the FB product term budget is reached, there is a small interconnect timing penalty to route signals to another FB to continue creating logic. Xilinx design software handles all this automatically.

Macrocell

The CoolRunner-II CPLD macrocell is extremely efficient and streamlined for logic creation. Users can develop sum of product (SOP) logic expressions that comprise up to 40 inputs and span 56 product terms within a single function block. The macrocell can further combine the SOP expression into an XOR gate with another single p-term expression. The resulting logic expression's polarity is also selectable. As well, the logic function can be pure combinatorial or registered, with the storage element operating selectably as a D or T flip-flop, or transparent latch. Available at each macrocell are independent selections of global, function block level or local p-term derived clocks, sets, resets, and output enables. Each macrocell flip-flop is configurable for either single edge or DualEDGE clocking, providing either double data rate capability or the ability to distribute a slower clock (thereby saving power). For single edge clocking or latching, either clock polarity may be selected per macrocell. CoolRunner-II macrocell details are shown in Figure 3. Note that in Figure 4, standard logic symbols are used except the trapezoidal multiplexers have input selection from statically programmed configuration select lines (not shown). Xilinx application note XAPP376 gives a detailed explanation of how logic is created in the CoolRunner-II CPLD family.

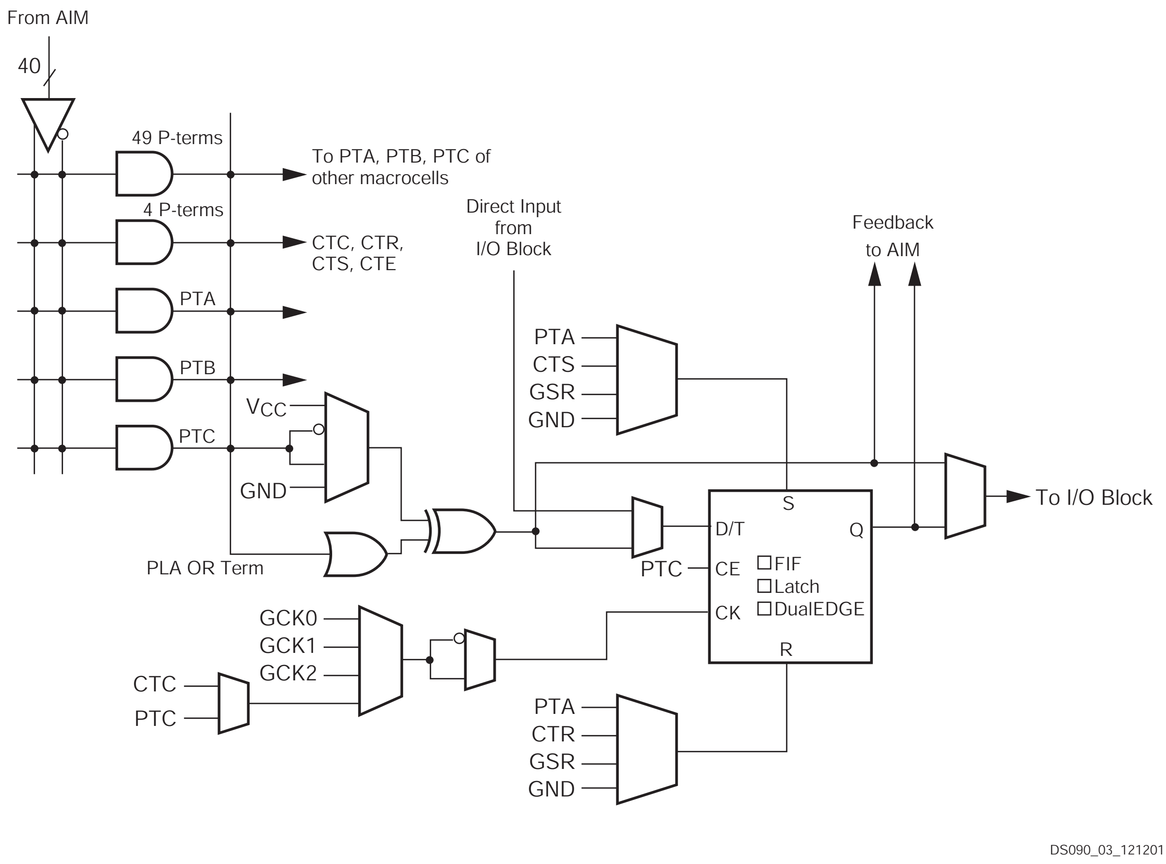


Figure 3: **CoolRunner-II CPLD Macrocell**

When configured as a D-type flip-flop, each macrocell has an optional clock enable signal permitting state hold while a clock runs freely. Note that Control Terms (CT) are available to be shared for key functions within the FB, and are generally used whenever the exact same logic function would be repeatedly created at multiple macrocells. The CT product terms are available for FB clocking (CTC), FB asynchronous set (CTS), FB asynchronous reset (CTR), and FB output enable (CTE).

Any macrocell flip-flop can be configured as an input register or latch, which takes in the signal from the macrocell's I/O pin, and directly drives the AIM. The macrocell combinational functionality is retained for use as a buried logic node if needed. F_{Toggle} is the maximum clock frequency to which a T flip-flop can reliably toggle.

Advanced Interconnect Matrix (AIM)

The Advanced Interconnect Matrix is a highly connected low power rapid switch. The AIM is directed by the software to deliver up to a set of 40 signals to each FB for the creation of logic. Results from all FB macrocells, as well as, all pin inputs circulate back through the AIM for additional connection available to all other FBs as dictated by the design software. The AIM minimizes both propagation delay and power as it makes attachments to the various FBs.

I/O Block

I/O blocks are primarily transceivers. However, each I/O is either automatically compliant with standard voltage ranges or can be programmed to become so. See XAPP382 for detailed information on CoolRunner-II I/Os.

In addition to voltage levels, each input can selectively arrive through Schmitt-trigger inputs. This adds a small time delay, but substantially reduces noise on that input pin. Approximately 500 mV of hysteresis will be added when Schmitt-trigger inputs are selected. All LVCMOS inputs can have hysteresis input. Hysteresis also allows easy generation of external clock circuits. The Schmitt-trigger path is best seen in Figure 4. See Table 5 for Schmitt-trigger compatibility with I/O standards.

Outputs can be directly driven, 3-stated or open-drain configured. A choice of slow or fast slew rate output signal is also available. Table 5 summarizes various supported voltage standards associated with specific part capacities. All inputs and disabled outputs are voltage tolerant up to 3.3V.

The CoolRunner-II family supports SSTL2-1, SSTL3-1 and HSTL-1 high-speed I/O standards in the 128-macrocell and larger devices. Figure 4 details the I/O pin, where it is noted that the inputs requiring comparison to an external reference voltage are available. These I/O standards all require VREF pins for proper operation. The CoolRunner-II CPLD allows any I/O pin to act as a VREF pin, granting the board layout engineer extra freedom when laying out the pins.However, if VREF pin placement is not done properly, additional VREF pins may be required, resulting in a loss of potential I/O pins or board re-work. See XAPP399 for details regarding VREF pins and their placement.

V_{REF} has pin-range requirements that must be observed. The Xilinx software aids designers in remaining within the proper pin range.

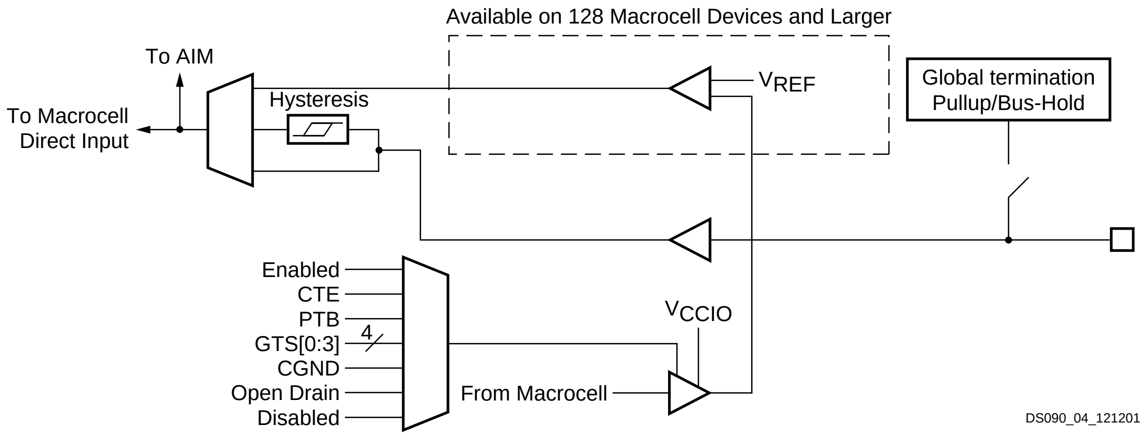

Available on 128 Macrocell Devices and Larger
To AIM
To Macrocell Direct Input
Hysteresis
VREF
Global termination Pullup/Bus-Hold
Enabled
CTE
PTB
GTS[0:3]
4
CGND
Open Drain
Disabled
VCCIO
From Macrocell
DS090_04_121201

Figure 4: **CoolRunner-II CPLD I/O Block Diagram**

Table 5 summarizes the single ended I/O standard support and shows which standards require V_{REF} values and board termination. V_{REF} detail is given in specific data sheets.

Table 5: **CoolRunner-II CPLD I/O Standard Summary**

IOSTANDARD Attribute	V_{CCIO}	Input V_{REF}	Board Termination Voltage (V_{TT})	Schmitt-trigger Support
LVTTL	3.3	N/A	N/A	Optional
LVCMOS33	3.3	N/A	N/A	Optional
LVCMOS25	2.5	N/A	N/A	Optional
LVCMOS18	1.8	N/A	N/A	Optional
LVCMOS15	1.5	N/A	N/A	Not optional
HSTL_1	1.5	0.75	0.75	Not optional
SSTL2_1	2.5	1.25	1.25	Not optional
SSTL3_1	3.3	1.5	1.5	Not optional

Output Banking

CPLDs are widely used as voltage interface translators. To that end, the output pins are grouped in large banks. The XC2C32 and XC2C64 devices are not banked, but the new XC2C32A and XC2C64A devices have two banks. The medium parts (128 and 256 macrocell) support two output banks. With two, the outputs will switch to one of two selected output voltage levels, unless both banks are set to the same voltage. The larger parts (384 and 512 macrocell) support four output banks split evenly. They can support groupings of one, two, three or four separate output voltage levels. This kind of flexibility permits easy interfacing to 3.3V, 2.5V, 1.8V, and 1.5V in a single part.

DataGATE

Low power is the hallmark of CMOS technology. Other CPLD families use a sense amplifier approach to creating product terms, which always has a residual current component being drawn. This residual current can be several hundred milliamps, making them unusable in portable systems. CoolRunner-II CPLDs use standard CMOS methods to create the CPLD architecture and deliver the corresponding low current consumption, without doing any special tricks. However, sometimes designers would like to reduce their system current even more by selectively disabling circuitry not being used.

The patented DataGATE technology was developed to permit a straightforward approach to additional power reduction. Each I/O pin has a series switch that can block the arrival of free running signals that are not of interest. Signals that serve no use may increase power consumption, and can be disabled. Users are free to do their design, then choose sections to participate in the DataGATE function. DataGATE is a logic function that drives an assertion rail threaded through the medium and high-density CoolRunner-II CPLD parts. Designers can select inputs to be blocked under the control of the DataGATE function, effectively blocking controlled switching signals so they do not drive internal chip capacitances. Output signals that do not switch, are held by the bus hold feature. Any set of input pins can be chosen to participate in the DataGATE function. Figure 5 shows the familiar CMOS I_{CC} versus switching frequency graph. With DataGATE, designers can approach zero power, should they choose to, in their designs

Figure 6 shows how DataGATE basically works. One I/O pin drives the DataGATE Assertion Rail. It can have any desired logic function on it. It can be as simple as mapping an input pin to the DataGATE function or as complex as a counter or state machine output driving the DataGATE I/O pin through a macrocell. When the DataGATE rail is asserted high, any pass transistor switch attached to it is blocked. Note that each pin has the ability to attach to the AIM through a DataGATE pass transistor, and thus be blocked. A latch automatically captures the state of the pin when it becomes blocked. The DataGATE Assertion Rail threads throughout all possible I/Os, so each can participate if chosen. Note that one macrocell is singled out to drive the rail, and that macrocell is exposed to the outside world through a pin, for inspection. If DataGATE is not needed, this pin is an ordinary I/O.

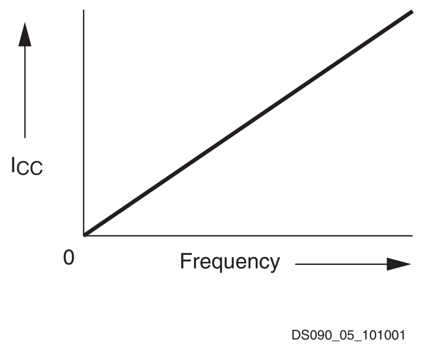


Figure 5: **CMOS I_{CC} vs. Switching Frequency Curve**

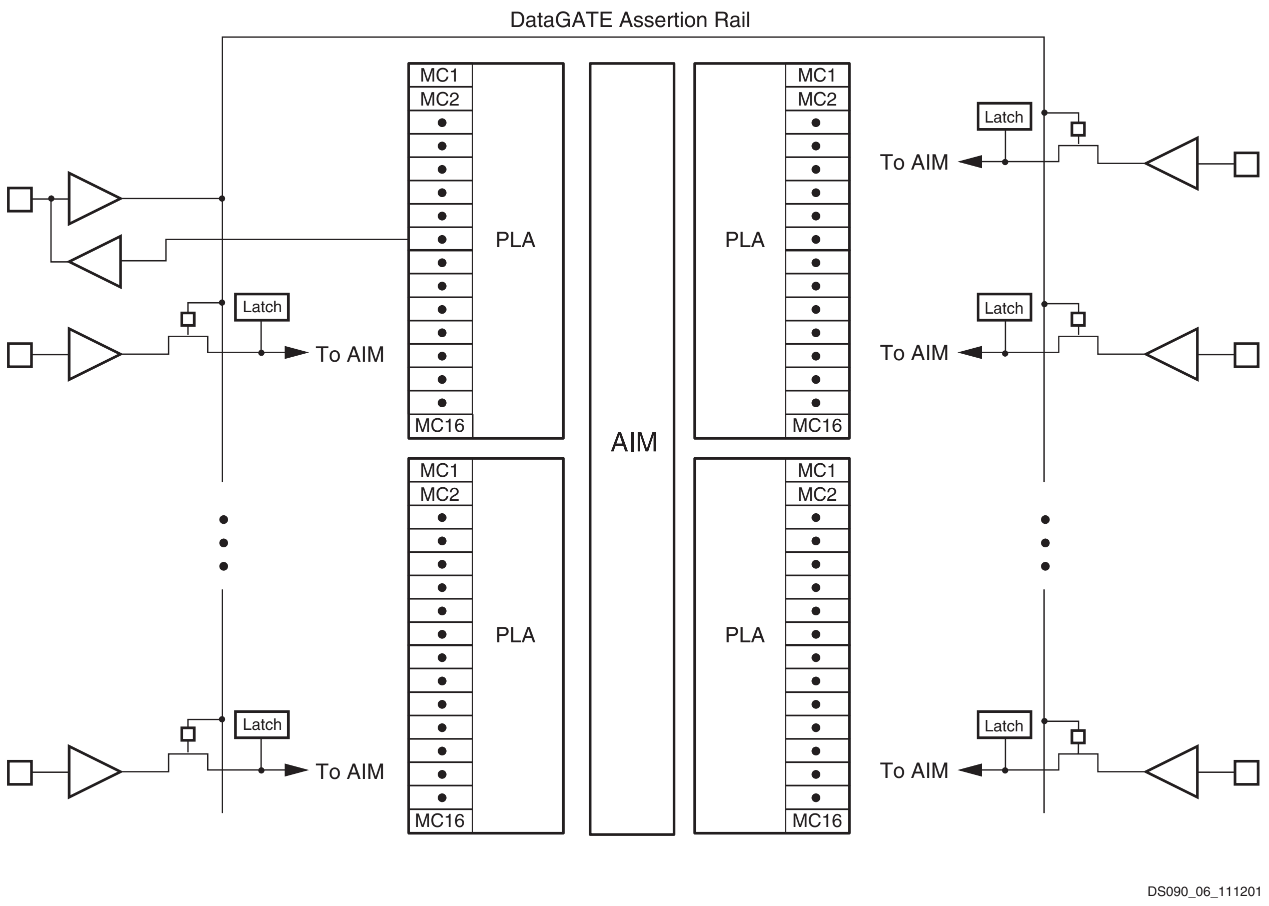


Figure 6: **DataGATE Architecture (output drivers not shown)**

Global Signals

Global signals, clocks (GCK), sets/resets (GSR) and output enables (GTS), are designed to strongly resemble each other. This approach enables design software to make the best utilization of their capabilities. Each global capability is supplemented by a corresponding product term version. Figure 7 shows the common structure of the global signal trees. The pin input is buffered, then drives multiple internal global signal traces to deliver low skew and reduce loading delays. The DataGATE assertion rail is also a global signal.

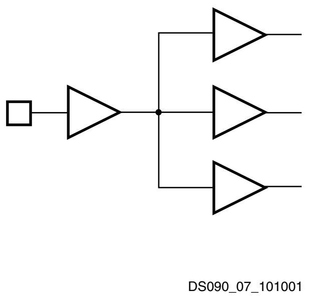


Figure 7: **Global Clocks (GCK), Sets/Resets (GSR) and Output Enables (GTS)**

Additional Clock Options: Division, DualEDGE, and CoolCLOCK

Division

Circuitry has been included in the CoolRunner-II CPLD architecture to divide one externally supplied global clock by standard values. Division by 2,4,6,8,10, 12, 14 and 16 are the options (see Figure 8). This capability is supplied on the GCK2 pin. The resulting clock produced will be 50% duty cycle for all possible divisions. Note that a Synchronous Reset (CDRST) is included to guarantee no runt clocks can get through to the global clock nets. Note that again, the signal is buffered and driven to multiple traces with minimal loading and skew.

DualEDGE

Each macrocell has the ability to double its input clock switching frequency. Figure 9 shows the macrocell flip-flop with the DualEDGE option (doubled clock) at each macrocell. The source to double can be a control term clock, a product term clock or one of the available global clocks. The ability to switch on both clock edges is vital for a number of synchronous memory interface applications as well as certain double data rate I/O applications.

CoolCLOCK

In addition to the DualEDGE flip-flop, additional power savings can be had by combining the clock division circuitry with the DualEDGE circuitry. This capability is called CoolCLOCK and is designed to reduce clocking power within the CPLD. Because the clock net can be an appreciable power drain, the clock power can be reduced by driving the net at half frequency, then doubling the clock rate using DualEDGE triggering at the macrocells. Figure 10 shows how CoolCLOCK is created by internal clock cascading with the divider and DualEDGE flip-flop working together. See **XAPP378** for more detail.

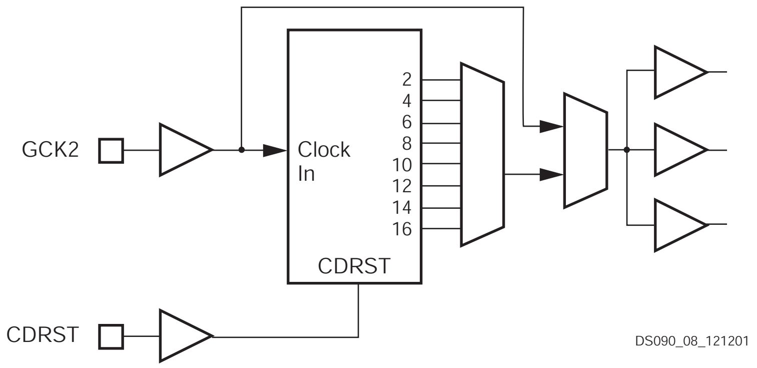


Figure 8: **Clock Division Circuitry for GCK2**

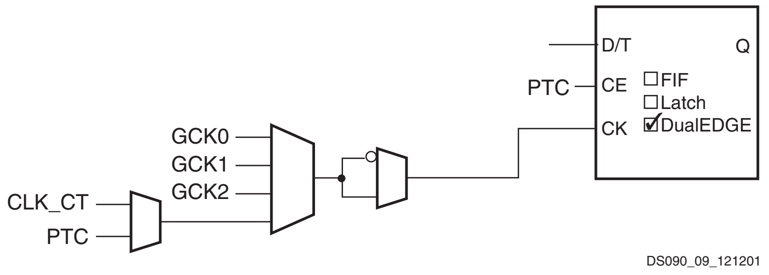


Figure 9: **Macrocell Clock Chain with DualEDGE Option Shown**

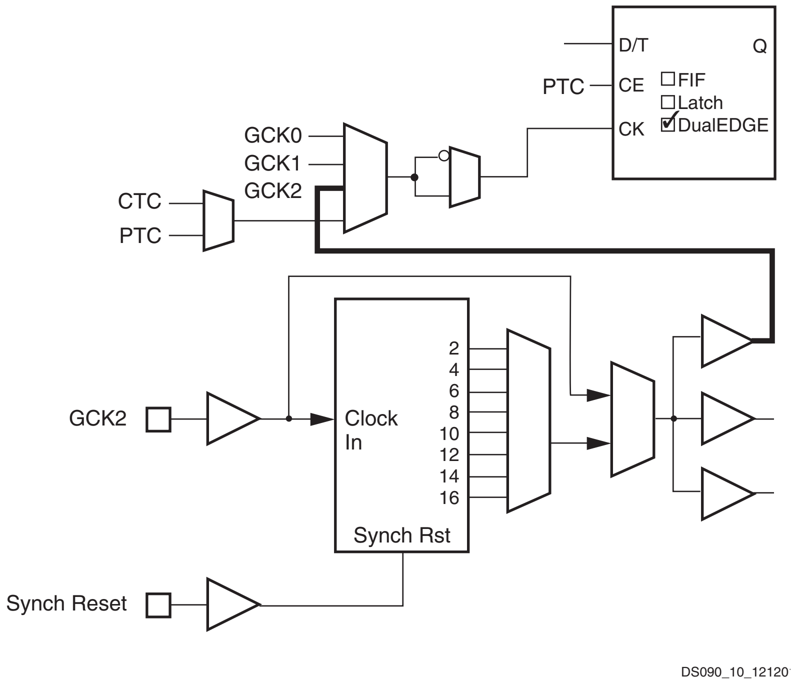


Figure 10: **CoolCLOCK Created by Cascading Clock Divider and DualEDGE Option**

Design Security

Designs can be secured during programming to prevent either accidental overwriting or pattern theft via readback. Four independent levels of security are provided on-chip, eliminating any electrical or visual detection of configuration patterns. These security bits can be reset only by erasing the entire device. See WP170 for more detail.

Timing Model

Figure 11 shows the CoolRunner-II CPLD timing model. It represents one aspect of the overall architecture from a timing viewpoint. Each little block is a time delay that a signal will incur if the signal passes through such a resource. Timing reports are created by tallying the incremental signal delays as signals progress within the CPLD. Software creates the timing reports after a design has been mapped onto the specific part, and knows the specific delay values for a given speed grade. Equations for the higher level timing values (i.e., T_{PD} and F_{SYSTEM}) are available. Table 6 summarizes the individual parameters and provides a brief definition of their associated functions. Xilinx application note XAPP375 details the CoolRunner-II CPLD family timing with several examples.

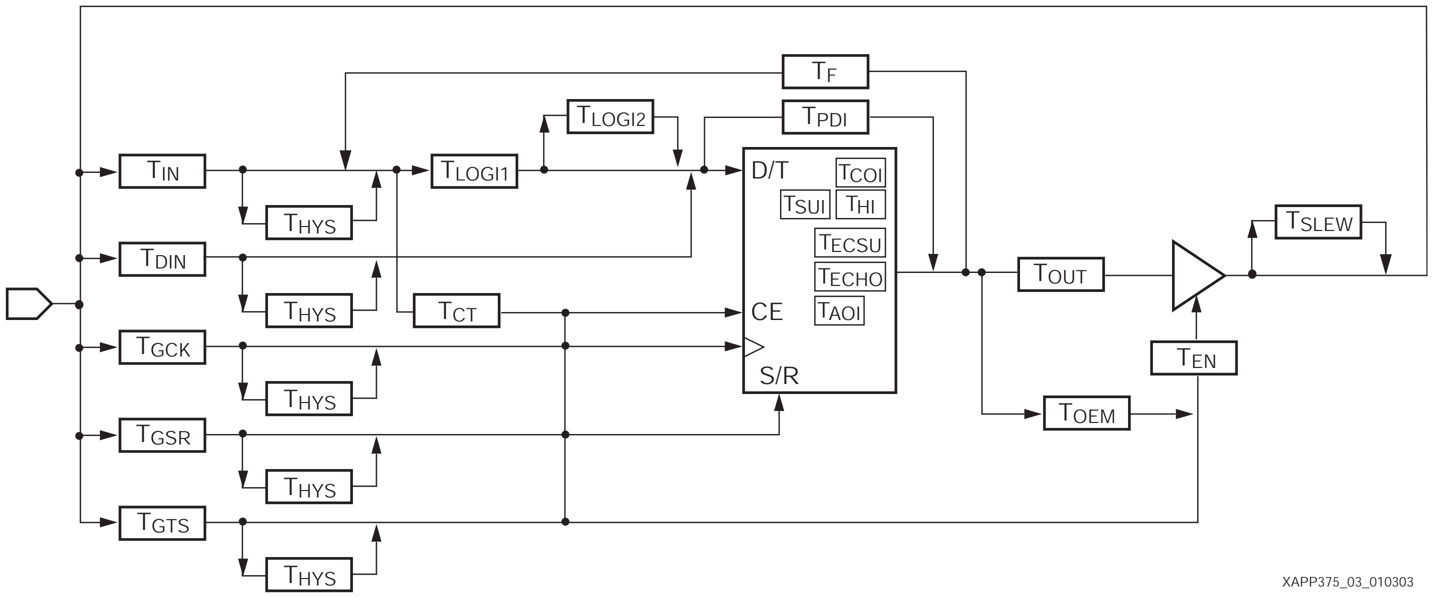


Figure 11: **CoolRunner-II CPLD Timing Model**

Table 6: **Timing Parameter Definitions**

Symbol	Parameter
Buffer Delays	
T_{IN}	Input Buffer Delay
T_{DIN}	Direct data register input delay
T_{GCK}	Global clock (GCK) buffer delay
T_{GSR}	Global set/reset (GSR) buffer delay
T_{GTS}	Global output enable (GTS) buffer delay
T_{OUT}	Output buffer delay
T_{EN}	Output buffer enable/disable delay
T_{SLEW}	Output buffer slew rate control delay
P-term Delays	
T_{CT}	Control Term delay (single PT or FB-CT)
T_{LOGI1}	Single P-term logic delay
T_{LOGI2}	Multiple P-term logic delay adder
Macrocell Delays	
T_{PDI}	Macro cell input to output valid
T_{SUI}	Macro register setup before clock
T_{HI}	Macro register hold after clock
T_{ECSU}	Macro register enable clock setup time
T_{ECHO}	Macro register enable clock hold time
T_{COI}	Macro register clock to output valid
T_{AOI}	Macro register set/reset to output valid
T_{HYS}	Hysteresis selection delay adder
Feedback Delays	
T_F	Feedback delay
T_{OEM}	Macrocell to Global OE delay

Programming

The programming data sequence is delivered to the device using either Xilinx iMPACT software and a Xilinx download cable, a third-party JTAG development system, a JTAG-compatible board tester, or a simple microprocessor interface that emulates the JTAG instruction sequence. The iMPACT software also outputs serial vector format (SVF) files for use with any tools that accept SVF format, including automatic test equipment. See **CoolRunner-II Application Notes** for more information on how to program.

In System Programming

All CoolRunner-II CPLD parts are 1.8V in system programmable. This means they derive their programming voltage and currents from the 1.8V V_{CC} (internal supply voltage) pins on the part. The V_{CCIO} pins do not participate in this operation, as they may assume another voltage ranging as high as 3.3V down to 1.5V. A 1.8V V_{CC} is required to properly operate the internal state machines and charge pumps that reside within the CPLD to do the nonvolatile programming operations. The JTAG interface buffers are powered by a dedicated power pin, V_{CCAUX}, which is independent of all other supply pins. V_{CCAUX} must be connected. Xilinx software is provided to deliver the bit-stream to the CPLD and drive the appropriate IEEE 1532 protocol. To that end, there is a set of IEEE 1532 commands that are supported in the CoolRunner-II CPLD parts. Programming times are less than one second for 32 to 256 macrocell parts. Programming times are less than four seconds for 384 and 512 macrocell parts. Programming of CoolRunner-II CPLDs is only guaranteed when operating in the commercial temperature and voltage ranges as defined in the device-specific data sheets.

On-The-Fly Reconfiguration (OTF)

Xilinx ISE 5.2i supports OTF for CoolRunner-II CPLDs. This permits programming a new nonvolatile pattern into the part while another pattern is currently in use. OTF has the same voltage and temperature specifications as system programming. During pattern transition I/O pins are in high impedance with weak pullup to V_{CCIO}. Transition time typically lasts between 50 and 300 µs, depending on density. See **XAPP388** for more information.

JTAG Instructions

Table 7 shows the commands available to users. These same commands may be used by third party ATE products, as well. The internal controllers can operate as fast as 66 MHz.

Table 7: **JTAG Instructions**

Code	Instruction	Description
`00000000`	EXTEST	Force boundary scan data onto outputs
`00000011`	PRELOAD	Latch macrocell data into boundary scan cells
`11111111`	BYPASS	Insert bypass register between TDI and TDO
`00000010`	INTEST	Force boundary scan data onto inputs and feedbacks
`00000001`	IDCODE	Read IDCODE
`11111101`	USERCODE	Read USERCODE
`11111100`	HIGHZ	Force output into high impedance state
`11111010`	CLAMP	Latch present output state

Power-Up Characteristics

CoolRunner-II CPLD parts must operate under the demands of both the high-speed and the portable market places, therefore, they must support hot plugging for the high-speed world and tolerate most any power sequence to its various voltage pins. They must also not draw excessive current during power-up initialization. To those ends, the general behavior is summarized as follows:

1. I/O pins are disabled until the end of power-up.
2. As supply rises, configuration bits transfer from nonvolatile memory to SRAM cells.
3. As power up completes, the outputs become as configured (input, output, or I/O).
4. For specific configuration times and power up requirements, see the device specific data sheet.

CoolRunner-II CPLD I/O pins are well behaved under all operating conditions. During power-up, CoolRunner-II devices employ internal circuitry which keeps the devices in the quiescent state until the V_{CCINT} supply voltage is at a safe level (approximately 1.3V). In the quiescent state, JTAG pins are disabled, and all device outputs are disabled with the pins weakly pulled high, as shown in Table 8. When the supply voltage reaches a safe level, all user registers become initialized, and the device is immediately available for operation, as shown in Figure 12. Best results are obtained with a smooth V_{CC} rise in less than 4 ms. Final V_{CC} value should occur within 1 second.

If the device is in the erased state (before any user pattern is programmed), the device outputs remain disabled with a weak pull-up. The JTAG pins are enabled to allow the device to be programmed at any time. All devices are shipped in the erased state from the factory.

Applying power to a blank part may result in a higher current flow as the part initializes. This behavior is normal and may persist for approximately 2 seconds, depending on the power supply ramp.

If the device is programmed, the device inputs and outputs take on their configured states for normal operation. The JTAG pins are enabled to allow device erasure or boundary-scan tests at any time.

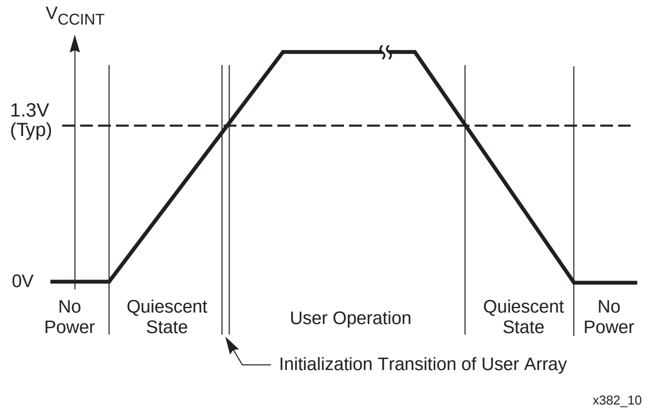


Figure 12: **Device Behavior During Power Up**

Table 8: **I/O Power-Up Characteristics**

Device Circuitry	Quiescent State	Erased Device Operation	Valid User Operation
IOB Bus-Hold/Weak Pullup	Weak Pull-up	Weak Pull-up	Bus-Hold/Weak Pullup
Device Outputs	Disabled	Disabled	As Configured
Device Inputs and Clocks	Disabled	Disabled	As Configured
Function Block	Disabled	Disabled	As Configured
JTAG Controller	Disabled	Enabled	Enabled

I/O Banking

CoolRunner-II CPLD XC2C32 and XC2C64 macrocell parts support a single V_{CCIO} rail that can range from 3.3V down to 1.5V operation. Two V_{CCIO} rails are supported on the XC2C32A, XC2C64A, 128 and 256 macrocell parts where outputs on each rail can independently range from 3.3V down to 1.5V operation. Four V_{CCIO} rails are supported on the 384 and 512 macrocell parts. Any of the V_{CCIO} rails can assume any one of the V_{CCIO} values of 1.5V, 1.8V, 2.5V, or 3.3V. Designers should assign input and output voltages to a bank with V_{CCIO} set at the voltage range of that input or output voltage. The V_{CC} (internal supply voltage) for a CoolRunner-II CPLD must be maintained within 1.8V ±5% for correct speed operation and proper in system programming.

Mixed Voltage, Power Sequencing, and Hot Plugging

As mentioned in I/O Banking, CoolRunner-II CPLD parts support mixed voltage I/O signals. It is important to assign signals to an I/O bank with the appropriate I/O voltage. Driving a high voltage into a low voltage bank can result in negative current flow through the power supply pins. The power applied to the V_{CCIO} and V_{CC} pins can occur in any order and the CoolRunner-II CPLD will not be damaged. For best results, we recommend that V_{CCIO} be applied before V_{CCINT}. CoolRunner-II CPLDs can reside on boards where the board is inserted into a “live” connector (hot plugged) and the parts will be well-behaved as if powering up in a standard way.

Development System Support

Xilinx CoolRunner-II CPLDs are supported by all configurations of Xilinx standard release development software as well as the freely available ISE WebPACK software available from www.xilinx.com. Third party development tools include synthesis tools from Cadence, Exemplar, Mentor Graphics, Synplicity, and Synopsys.

ATE Support

Third party ATE development support is available for both programming and board/chip level testing. Vendors providing this support include Agilent, GenRad, and Teradyne. Other third party providers are expected to deliver solutions in the future.

Absolute Maximum Ratings[1]

Symbol	Parameter	Min.	Max.	Unit
V_{CC}[2]	Supply voltage relative to GND	–0.5	2.0	V
V_I[3]	Input voltage relative to GND	–0.5	4.0	V
T_J[4]	Maximum junction temperature	–40	150	°C
T_{STR}	Storage temperature	–65	150	°C

Notes:
1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.
2. The chip supply voltage should rise monotonically.
3. Maximum DC undershoot below GND must be limited to either 0.5V or 10 mA, whichever is easier to achieve. During transitions, the device pins may undershoot to –2.0V or overshoot to 4.5 V, provided this over- or undershoot lasts less than 10 ns and with the forcing current being limited to 200 mA. The I/O voltage may never exceed 4.0V.
4. For soldering guidelines and thermal considerations, see the Device Packaging information on the Xilinx website. For Pb-free packages, see XAPP427.

Quality and Reliability Parameters

Symbol	Parameter	Min	Max	Units
T_{DR}	Data retention	20	-	Years
N_{PE}	Program/erase cycles (Endurance)	1,000	-	Cycles
V_{ESD}	Electrostatic discharge(1)	2,000	-	Volts

Notes:
1. ESD is measured to 2000V using the human body model. Pins exposed to this limit can incur additional leakage current to a maximum of 10 μA when driven to 3.9V.

Further Reading

Application Notes

http://direct.xilinx.com/bvdocs/appnotes/xapp375.pdf
(Timing Model)

http://direct.xilinx.com/bvdocs/appnotes/xapp376.pdf
(Logic Engine)

http://direct.xilinx.com/bvdocs/appnotes/xapp377.pdf
(Low Power Design)

http://direct.xilinx.com/bvdocs/appnotes/xapp378.pdf
(Advanced Features)

http://direct.xilinx.com/bvdocs/appnotes/xapp379.pdf
(High Speed Design)

http://direct.xilinx.com/bvdocs/appnotes/xapp380.pdf
(Cross Point Switch)

http://direct.xilinx.com/bvdocs/appnotes/xapp381.pdf
(Demo Board)

http://direct.xilinx.com/bvdocs/appnotes/xapp382.pdf
(I/O Characteristics)

http://direct.xilinx.com/bvdocs/appnotes/xapp383.pdf
(Single Error Correction Double Error Detection)

http://direct.xilinx.com/bvdocs/appnotes/xapp384.pdf
(DDR SDRAM Interface)

http://direct.xilinx.com/bvdocs/appnotes/xapp387.pdf
(PicoBlaze Microcontroller)

http://direct.xilinx.com/bvdocs/appnotes/xapp388.pdf
(On the Fly Reconfiguration)

http://direct.xilinx.com/bvdocs/appnotes/xapp389.pdf
(Powering CoolRunner-II)

http://direct.xilinx.com/bvdocs/appnotes/xapp393.pdf
(8051 Microcontroller Interface)

http://direct.xilinx.com/bvdocs/appnotes/xapp394.pdf
(Interfacing with Mobile SDRAM)

http://direct.xilinx.com/bvdocs/appnotes/xapp399.pdf
(Assigning CoolRunner-II VREF Pins)

CoolRunner-II Data Sheets

http://direct.xilinx.com/bvdocs/publications/ds090.pdf
(CoolRunner-II Family Datasheet)

http://direct.xilinx.com/bvdocs/publications/ds091.pdf
(XC2C32 Datasheet)

http://direct.xilinx.com/bvdocs/publications/ds092.pdf
(XC2C64 Datasheet)

http://direct.xilinx.com/bvdocs/publications/ds093.pdf
(XC2C128 Datasheet)

http://direct.xilinx.com/bvdocs/publications/ds094.pdf
(XC2C256 Datasheet)

http://direct.xilinx.com/bvdocs/publications/ds095.pdf
(XC2C384 Datasheet)

http://direct.xilinx.com/bvdocs/publications/ds096.pdf
(XC2C512 Datasheet)

CoolRunner-II White Papers

http://direct.xilinx.com/bvdocs/whitepapers/wp170.pdf
(Security)

Packages

Package Drawings

Online Store

Xilinx Online Store

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/03/02	1.0	Initial Xilinx release
07/04/02	1.1	Revisions and updates
07/24/02	1.2	Revisions and updates
09/24/02	1.3	Additions to "Power Characteristics" section
01/28/03	1.4	Addition of the "Further Reading" section
02/26/03	1.5	Multiple minor revisions
03/12/03	1.6	Minor revision to "Quality and Reliability Parameters"
10/09/03	1.7	Update Hewlett-Packard to Agilent, OFR to OTF, and other revisions
01/26/04	1.8	Incorporate links to Data Sheets, Application Notes, and Device Packages
02/26/04	1.9	Change to Power-Up Characteristics, page 11. Change T_{FIN} to T_{DIN}. Add Schmitt-trigger I/O compatibility information. Added T_{SOL} specification.
05/21/04	2.0	Add XC2C32A and XC2C64A devices.
07/30/04	2.1	Pb-free documentation. Changes to T_{SU} and F_{system} to match individual data sheets.
01/10/05	2.2	Added information about programming options, page 11.
03/07/05	2.3	Changes to Table 1, T_{PD}, T_{SU}, T_{CO}, and $F_{SYSTEM1.}$ Removed link to obsolete White Paper. Modifications to Table 5, IOSTANDARDs. Added Table 2, DC Characteristics.
04/15/05	2.4	Change to $F_{SYSTEM1}$ for XC2C128.
06/28/05	2.5	Move to Product Specification

XILINX®

XC2C32A CoolRunner-II CPLD

DS310 (v1.7) June 28, 2005 **Product Specification**

Features

- Optimized for 1.8V systems
  - As fast as 3.8 ns pin-to-pin logic delays
  - As low as 12 µA quiescent current
- Industry's best 0.18 micron CMOS CPLD
  - Optimized architecture for effective logic synthesis
  - Multi-voltage I/O operation: 1.5V through 3.3V
- Available in multiple package options
  - 32-land QFN with 21 user I/O
  - 44-pin PLCC with 33 user I/O
  - 44-pin VQFP with 33 user I/O
  - 56-ball CP BGA with 33 user I/O
  - Pb-free available for all packages
- Advanced system features
  - Fastest in system programming
    - 1.8V ISP using IEEE 1532 (JTAG) interface
  - IEEE1149.1 JTAG Boundary Scan Test
  - Optional Schmitt-trigger input (per pin)
  - Two separate I/O banks
  - RealDigital 100% CMOS product term generation
  - Flexible clocking modes
  - Optional DualEDGE triggered registers
  - Global signal options with macrocell control
    - Multiple global clocks with phase selection per macrocell
    - Multiple global output enables
    - Global set/reset
  - Efficient control term clocks, output enables and set/resets for each macrocell and shared across function blocks
  - Advanced design security
  - Open-drain output option for Wired-OR and LED drive
  - Optional configurable grounds on unused I/Os
  - Optional bus-hold, 3-state or weak pullup on selected I/O pins
  - Mixed I/O voltages compatible with 1.5V, 1.8V, 2.5V, and 3.3V logic levels
  - PLA architecture
    - Superior pinout retention
    - 100% product term routability across function block
  - Hot pluggable

Refer to the CoolRunner™-II family data sheet for architecture description.

Description

The CoolRunner™-II 32-macrocell device is designed for both high performance and low power applications. This lends power savings to high-end communication equipment and high speed to battery operated devices. Due to the low power stand-by and dynamic operation, overall system reliability is improved

This device consists of two Function Blocks interconnected by a low power Advanced Interconnect Matrix (AIM). The AIM feeds 40 true and complement inputs to each Function Block. The Function Blocks consist of a 40 by 56 P-term PLA and 16 macrocells which contain numerous configuration bits that allow for combinational or registered modes of operation.

Additionally, these registers can be globally reset or preset and configured as a D or T flip-flop or as a D latch. There are also multiple clock signals, both global and local product term types, configured on a per macrocell basis. Output pin configurations include slew rate limit, bus hold, pull-up, open drain and programmable grounds. A Schmitt trigger input is available on a per input pin basis. In addition to storing macrocell output states, the macrocell registers may be configured as "direct input" registers to store signals directly from input pins.

Clocking is available on a global or Function Block basis. Three global clocks are available for all Function Blocks as a synchronous clock source. Macrocell registers can be individually configured to power up to the zero or one state. A global set/reset control line is also available to asynchronously set or reset selected registers during operation. Additional local clock, synchronous clock-enable, asynchronous set/reset and output enable signals can be formed using product terms on a per-macrocell or per-Function Block basis.

The CoolRunner-II 32-macrocell CPLD is I/O compatible with standard LVTTL and LVCMOS18, LVCMOS25, and LVCMOS33 (see Table 1). This device is also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

Another feature that eases voltage translation is I/O banking. Two I/O banks are available on the CoolRunner-II 32A macrocell device that permit easy interfacing to 3.3V, 2.5V, 1.8V, and 1.5V devices.

RealDigital Design Technology

Xilinx CoolRunner-II CPLDs are fabricated on a 0.18 micron process technology which is derived from leading edge FPGA product development. CoolRunner-II CPLDs employ RealDigital, a design technique that makes use of CMOS technology in both the fabrication and design methodology. RealDigital design technology employs a cascade of CMOS gates to implement sum of products instead of traditional sense amplifier methodology. Due to this technology, Xilinx CoolRunner-II CPLDs achieve both high performance and low power operation.

Supported I/O Standards

The CoolRunner-II 32 macrocell features both LVCMOS and LVTTL I/O implementations. See Table 1 for I/O standard voltages. The LVTTL I/O standard is a general purpose EIA/JEDEC standard for 3.3V applications that use an LVTTL input buffer and Push-Pull output buffer. The LVCMOS standard is used in 3.3V, 2.5V, 1.8V applications. CoolRunner-II CPLDs are also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

Table 1: **I/O Standards for XC2C32A**

IOSTANDARD Attribute	Output V_{CCIO}	Input V_{CCIO}	Input V_{REF}	Board Termination Voltage V_T
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15(1)	1.5	1.5	N/A	N/A

(1) LVCMOS15 requires Schmitt-trigger inputs.

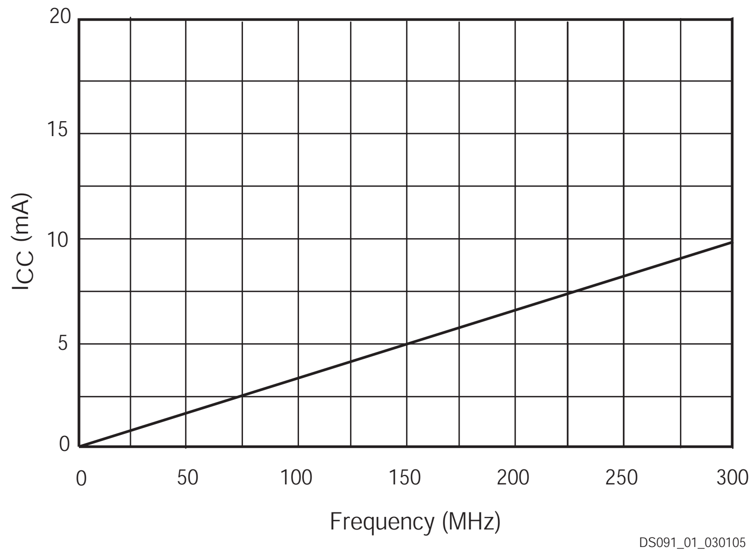


Figure 1: **I_{CC} vs Frequency**

Table 2: **I_{CC} vs Frequency (LVCMOS 1.8V T_A = 25°C)(1)**

	Frequency (MHz)										
	0	25	50	75	100	150	175	200	225	250	300
Typical I_{CC} (mA)	0.016	0.87	1.75	2.61	3.44	5.16	5.99	6.81	7.63	8.36	9.93

Notes:
1. 16-bit up/down, resettable binary counter (one counter per function block).

Absolute Maximum Ratings

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to ground	–0.5 to 2.0	V
V_{CCIO}	Supply voltage for output drivers	–0.5 to 4.0	V
V_{JTAG}(2)	JTAG input voltage limits	–0.5 to 4.0	V
V_{CCAUX}	JTAG input supply voltage	–0.5 to 4.0	V
V_{IN}(1)	Input voltage relative to ground	–0.5 to 4.0	V
V_{TS}(1)	Voltage applied to 3-state output	–0.5 to 4.0	V
T_{STG}(3)	Storage Temperature (ambient)	–65 to +150	°C
T_J	Junction Temperature	+150	°C

Notes:
1. Maximum DC undershoot below GND must be limited to either 0.5V or 10 mA, whichever is easiest to achieve. During transitions, the device pins may undershoot to –2.0v or overshoot to +4.5V, provided this over or undershoot lasts less than 10 ns and with the forcing current being limited to 200 mA.
2. Valid over commercial temperature range.
3. For soldering guidelines and thermal considerations, see the Device Packaging information on the Xilinx website. For Pb free packages, see XAPP427.

Recommended Operating Conditions

Symbol	Parameter		Min	Max	Units
V_{CC}	Supply voltage for internal logic and input buffers	Commercial T_A = 0°C to +70°C	1.7	1.9	V
		Industrial T_A = –40°C to +85°C	1.7	1.9	V
V_{CCIO}	Supply voltage for output drivers @ 3.3V operation		3.0	3.6	V
	Supply voltage for output drivers @ 2.5V operation		2.3	2.7	V
	Supply voltage for output drivers @ 1.8V operation		1.7	1.9	V
	Supply voltage for output drivers @ 1.5V operation		1.4	1.6	V
V_{CCAUX}	JTAG programming pins		1.7	3.6	V

DC Electrical Characteristics (Over Recommended Operating Conditions)

Symbol	Parameter	Test Conditions	Typical	Max.	Units
I_{CCSB}	Standby current Commercial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	22	90	µA
I_{CCSB}	Standby current Industrial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	38	150	µA
I_{CC} (1)	Dynamic current	f = 1 MHz	-	0.25	mA
		f = 50 MHz	-	2.5	mA
C_{JTAG}	JTAG input capacitance	f = 1 MHz	-	10	pF
C_{CLK}	Global clock input capacitance	f = 1 MHz	-	12	pF
C_{IO}	I/O capacitance	f = 1 MHz	-	10	pF
I_{IL}(2)	Input leakage current	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/-1	µA
I_{IH}(2)	I/O High-Z leakage	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/-1	µA

Notes:
1. 16-bit up/down resettable binary counter (one per Function Block) tested at V_{CC} = V_{CCIO} = 1.9V.
2. See Quality and Reliability section of the CoolRunner-II family data sheet.

LVCMOS 3.3V and LVTTL 3.3V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		3.0	3.6	V
V_{IH}	High level input voltage		2	3.9	V
V_{IL}	Low level input voltage		–0.3	0.8	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 3V	-	0.2	V

LVCMOS 2.5V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		2.3	2.7	V
V_{IH}	High level input voltage		1.7	3.9	V
V_{IL}	Low level input voltage		–0.3	0.7	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 2.3V	-	0.4	V
		I_{OL} = 0.1mA, V_{CCIO} = 2.3V	-	0.2	V

LVCMOS 1.8V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		1.7	1.9	V
V_{IH}	High level input voltage		0.65 x V_{CCIO}	3.9	V
V_{IL}	Low level input voltage		–0.3	0.35 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.2	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.7V	-	0.45	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.7V	-	0.2	V

LVCMOS 1.5V DC Voltage Specifications(1)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		1.4	1.6	V
V_{T+}	Input hysteresis threshold voltage		0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}			0.2 x V_{CCIO}	0.5 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.2	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.4V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.4V	-	0.2	V

Notes:
1. Hysteresis used on 1.5V inputs.

Schmitt Trigger Input DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		1.4	3.9	V
V_{T+}	Input hysteresis threshold voltage		0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}			0.2 x V_{CCIO}	0.5 x V_{CCIO}	V

AC Electrical Characteristics Over Recommended Operating Conditions

Symbol	Parameter	-4		-6		Units
		Min.	Max.	Min.	Max.	
T_{PD1}	Propagation delay single p-term	-	3.8	-	5.5	ns
T_{PD2}	Propagation delay OR array	-	4.0	-	6.0	ns
T_{SUD}	Direct input register clock setup time	1.7	-	2.2	-	ns
T_{SU1}	Setup time fast (single p-term)	1.9	-	2.6	-	ns
T_{SU2}	Setup time (OR array)	2.1	-	3.1	-	ns
T_{HD}	Direct input register hold time	0.0	-	0.0	-	ns
T_H	P-term hold time	0.0	-	0.0	-	ns
T_{CO}	Clock to output	-	3.7	-	4.7	ns
F_{TOGGLE}(1)	Internal toggle rate	-	500	-	300	MHz
$F_{SYSTEM1}$(2)	Maximum system frequency	-	323	-	200	MHz
$F_{SYSTEM2}$(2)	Maximum system frequency	-	303	-	182	MHz
F_{EXT1}(3)	Maximum external frequency	-	179	-	137	MHz
F_{EXT2}(3)	Maximum external frequency	-	172	-	128	MHz
T_{PSUD}	Direct input register p-term clock setup time	0.4	-	0.9	-	ns
T_{PSU1}	P-term clock setup time (single p-term)	0.6	-	1.3	-	ns
T_{PSU2}	P-term clock setup time (OR array)	0.8	-	1.8	-	ns
T_{PHD}	Direct input register p-term clock hold time	1.5	-	1.6	-	ns
T_{PH}	P-term clock hold	1.3	-	1.2	-	ns
T_{PCO}	P-term clock to output	-	5.0	-	6.0	ns
T_{OE}/T_{OD}	Global OE to output enable/disable	-	4.7	-	5.5	ns
T_{POE}/T_{POD}	P-term OE to output enable/disable	-	6.2	-	6.7	ns
T_{MOE}/T_{MOD}	Macrocell driven OE to output enable/disable	-	6.2	-	6.9	ns
T_{PAO}	P-term set/reset to output valid	-	5.5	-	6.8	ns
T_{AO}	Global set/reset to output valid	-	4.5	-	5.5	ns
T_{SUEC}	Register clock enable setup time	2.0	-	3.0	-	ns
T_{HEC}	Register clock enable hold time	0.0	-	0.0	-	ns
T_{CW}	Global clock pulse width High or Low	1.4	-	2.2	-	ns
T_{PCW}	P-term pulse width High or Low	4.0	-	6.0	-	ns
T_{APRPW}	Asynchronous preset/reset pulse width (High or Low)	4.0	-	6.0	-	ns
T_{CONFIG}(4)	Configuration time	-	50	-	50	µs

Notes:

1. F_{TOGGLE} is the maximum clock frequency to which a T-Flip Flop can reliably toggle (see the CoolRunner-II family data sheet).
2. $F_{SYSTEM1}$ ($1/T_{CYCLE}$) is the internal operating frequency for a device fully populated with one 16-bit counter through one p-term per macrocell while $F_{SYSTEM2}$ is through the OR array.
3. F_{EXT1} ($1/T_{SU1}+T_{CO}$) is the maximum external frequency using one p-term while F_{EXT2} is through the OR array.
4. Typical configuration current during T_{CONFIG} is 500 µA.

Internal Timing Parameters

Symbol	Parameter[(1)]	-4 Min.	-4 Max.	-6 Min.	-6 Max.	Units
Buffer Delays						
T_{IN}	Input buffer delay	-	1.3	-	1.7	ns
T_{DIN}	Direct register input delay	-	1.5	-	2.4	ns
T_{GCK}	Global Clock buffer delay	-	1.3	-	2.0	ns
T_{GSR}	Global set/reset buffer delay	-	1.6	-	2.0	ns
T_{GTS}	Global 3-state buffer delay	-	1.3	-	2.1	ns
T_{OUT}	Output buffer delay	-	1.8	-	2.0	ns
T_{EN}	Output buffer enable/disable delay	-	2.9	-	3.4	ns
P-term Delays						
T_{CT}	Control term delay	-	1.3	-	1.6	ns
T_{LOGI1}	Single p-term delay adder	-	0.4	-	1.1	ns
T_{LOGI2}	Multiple p-term delay adder	-	0.2	-	0.5	ns
Macrocell Delay						
T_{PDI}	Input to output valid	-	0.3	-	0.7	ns
T_{LDI}	Setup before clock (transparent latch)	-	1.5	-	2.5	ns
T_{SUI}	Setup before clock	1.5	-	1.8	-	ns
T_{HI}	Hold after clock	0.0	-	0.0	-	ns
T_{ECSU}	Enable clock setup time	0.7	-	1.7	-	ns
T_{ECHO}	Enable clock hold time	0.0	-	0.0	-	ns
T_{COI}	Clock to output valid	-	0.6	-	0.7	ns
T_{AOI}	Set/reset to output valid	-	1.1	-	1.5	ns
Feedback Delays						
T_{F}	Feedback delay	-	0.6	-	1.4	ns
T_{OEM}	Macrocell to global OE delay	-	0.2	-	0.8	ns
I/O Standard Time Adder Delays 1.5V CMOS						
T_{HYS15}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT15}	Output adder	-	0.8	-	1.0	ns
T_{SLEW15}	Output slew rate adder	-	4.0	-	5.0	ns
I/O Standard Time Adder Delays 1.8V CMOS						
T_{HYS18}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT18}	Output adder	-	0.0	-	0.0	ns
T_{SLEW}	Output slew rate adder	-	4.0	-	5.0	ns

Internal Timing Parameters *(Continued)*

Symbol	Parameter[1]	-4 Min.	-4 Max.	-6 Min.	-6 Max.	Units
I/O Standard Time Adder Delays 2.5V CMOS						
T_{IN25}	Standard input adder	-	0.5	-	0.6	ns
T_{HYS25}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT25}	Output adder	-	0.6	-	0.7	ns
T_{SLEW25}	Output slew rate adder	-	4.0	-	5.0	ns
I/O Standard Time Adder Delays 3.3V CMOS/TTL						
T_{IN33}	Standard input adder	-	0.5	-	0.6	ns
T_{HYS33}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT33}	Output adder	-	1.0	-	1.2	ns
T_{SLEW33}	Output slew rate adder	-	4.0	-	5.0	ns

Notes:
1. 1.5 ns input pin signal rise/fall.

Switching Characteristics

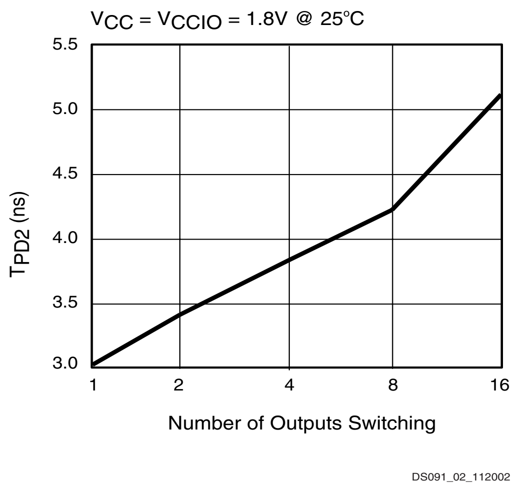
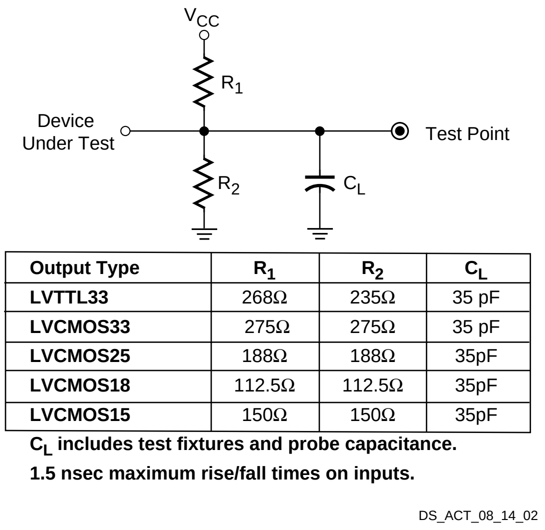


Figure 2: **Derating Curve for T_{PD}**

AC Test Circuit



Output Type	R_1	R_2	C_L
LVTTL33	268Ω	235Ω	35 pF
LVCMOS33	275Ω	275Ω	35 pF
LVCMOS25	188Ω	188Ω	35pF
LVCMOS18	112.5Ω	112.5Ω	35pF
LVCMOS15	150Ω	150Ω	35pF

C_L includes test fixtures and probe capacitance.
1.5 nsec maximum rise/fall times on inputs.

Figure 3: **AC Load Circuit**

Typical I/O Output Curves

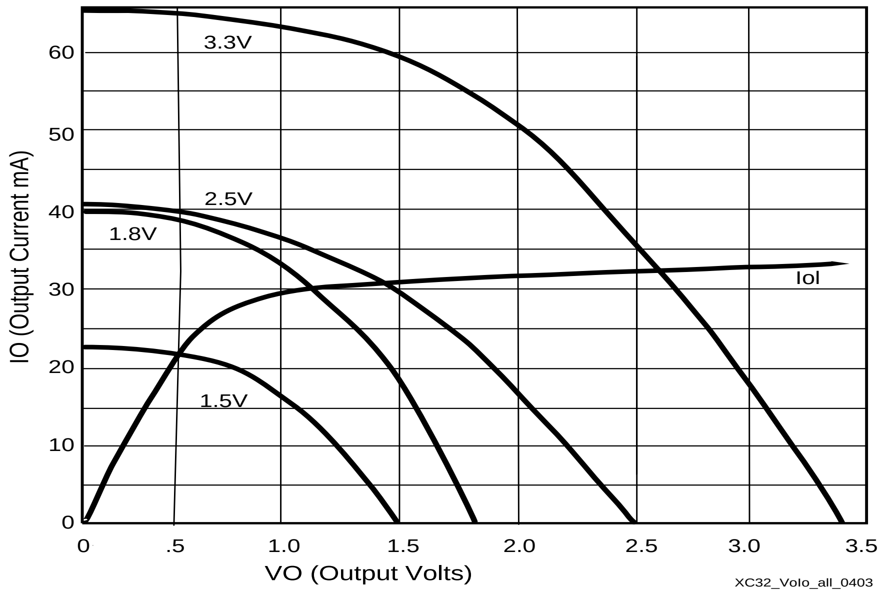


Figure 4: **Typical I/V Curve for XC2C32A**

Pin Descriptions

Function Block	Macrocell	QFG32	PC44	VQ44	CP56	I/O Bank
1	1		44	38	F1	Bank 2
1	2		43	37	E3	Bank 2
1	3		42	36	E1	Bank 2
1(GTS1)	4	3	40	34	D1	Bank 2
1(GTS0)	5	2	39	33	C1	Bank 2
1(GTS3)	6	1	38	32	A3	Bank 2
1(GTS2)	7	32	37	31	A2	Bank 2
1(GSR)	8	31	36	30	B1	Bank 2
1	9	30	35	29	A1	Bank 2
1	10	29	34	28	C4	Bank 2
1	11	28	33	27	C5	Bank 2
1	12	24	29	23	C8	Bank 2
1	13		28	22	A10	Bank 2
1	14	23	27	21	B10	Bank 2
1	15		26	20	C10	Bank 2
1	16		25	19	E8	Bank 2
2	1	5	1	39	G1	Bank 1
2	2		2	40	F3	Bank 1
2	3		3	41	H1	Bank 1
2	4		4	42	G3	Bank 1
2(GCK0)	5	6	5	43	J1	Bank 1
2(GCK1)	6	7	6	44	K1	Bank 1
2(GCK2)	7	8	7	1	K2	Bank 1

Pin Descriptions *(Continued)*

Function Block	Macrocell	QFG32	PC44	VQ44	CP56	I/O Bank
2	8	9	8	2	K3	Bank 1
2	9	10	9	3	H3	Bank 1
2	10		11	5	K5	Bank 1
2	11		12	6	H5	Bank 1
2	12	13	14	8	H8	Bank 1
2	13	17	18	12	K8	Bank 1
2	14	18	19	13	H10	Bank 1
2	15	19	20	14	G10	Bank 1
2	16		22	16	F10	Bank 1

Notes:
1. GTS = global output enable, GSR = global set reset, GCK = global clock

XC2C32A Global, JTAG, Power/Ground and No Connect Pins

Pin Type	QFG32	PC44[1]	VQ44[1]	CP56[1]
TCK	16	17	11	K10
TDI	14	15	9	J10
TDO	25	30	24	A6
TMS	15	16	10	K9
Input Only	22 (bank 2)	24 (bank 2)	18 (bank 2)	D10 (bank 2)
V_{CCAUX} (JTAG supply voltage)	4	41	35	D3
Power internal (V_{CC})	20	21	15	G8
Power bank 1 I/O (V_{CCIO1})	12	13	7	H6
Power bank 2 I/O (V_{CCIO2})	27	32	26	C6
Ground	11, 21, 26	10,23,31	4,17,25	H4, F8, C7
No connects		-	-	K4, K6, K7, H7, E10, A7, A9, D8, A5, A8, A4, C3
Total user I/O (includes dual function pins)	21	33	33	33

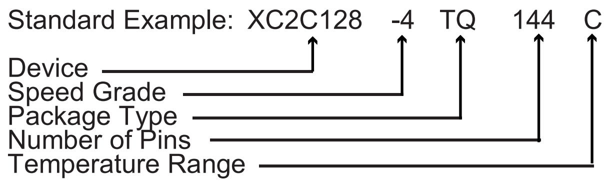
Notes:
1. All packages pin compatible with larger macrocell densities

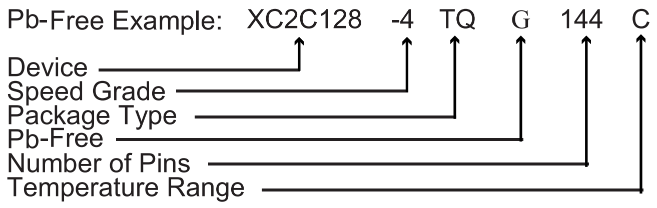
Ordering Information

Part Number	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Comm. (C) Ind. (I)(1)
XC2C32A-4QFG32C	0.5mm	35.5	24.0	Quad Flat No Lead; Pb-free	5mm x 5mm	21	C
XC2C32A-6QFG32C	0.5mm	35.5	24.0	Quad Flat No Lead; Pb-free	5mm x 5mm	21	C
XC2C32A-4PC44C	1.27mm	55.1	35.3	Plastic Leaded Chip Carrier	16.5mm x 16.5mm	33	C
XC2C32A-6PC44C	1.27mm	55.1	35.3	Plastic Leaded Chip Carrier	16.5mm x 16.5mm	33	C
XC2C32A-4VQ44C	0.8mm	47.7	8.2	Very Thin Quad Flat Pack	10mm x 10mm	33	C
XC2C32A-6VQ44C	0.8mm	47.7	8.2	Very Thin Quad Flat Pack	10mm x 10mm	33	C
XC2C32A-4CP56C	0.5mm	66.0	14.9	Chip Scale Package	6mm x 6mm	33	C
XC2C32A-6CP56C	0.5mm	66.0	14.9	Chip Scale Package	6mm x 6mm	33	C
XC2C32A-4PCG44C	1.27mm	55.1	35.3	Plastic Leaded Chip Carrier; Pb-free	16.5mm x 16.5mm	33	C
XC2C32A-6PCG44C	1.27mm	55.1	35.3	Plastic Leaded Chip Carrier; Pb-free	16.5mm x 16.5mm	33	C
XC2C32A-4VQG44C	0.8mm	47.7	8.2	Very Thin Quad Flat Pack; Pb-free	10mm x 10mm	33	C
XC2C32A-6VQG44C	0.8mm	47.7	8.2	Very Thin Quad Flat Pack; Pb-free	10mm x 10mm	33	C
XC2C32A-4CPG56C	0.5mm	66.0	14.9	Chip Scale Package; Pb-free	6mm x 6mm	33	C
XC2C32A-6CPG56C	0.5mm	66.0	14.9	Chip Scale Package; Pb-free	6mm x 6mm	33	C
XC2C32A-6QFG32I	0.5mm	35.5	24.0	Quad Flat No Lead; Pb-free	5mm x 5mm	21	I
XC2C32A-6PC44I	1.27mm	55.1	35.3	Plastic Leaded Chip Carrier	16.5mm x 16.5mm	33	I
XC2C32A-6VQ44I	0.8mm	47.7	8.2	Very Thin Quad Flat Pack	10mm x 10mm	33	I
XC2C32A-6CP56I	0.5mm	66.0	14.9	Chip Scale Package	6mm x 6mm	33	I
XC2C32A-6PCG44I	1.27mm	55.1	35.3	Plastic Leaded Chip Carrier; Pb-free	16.5mm x 16.5mm	33	I

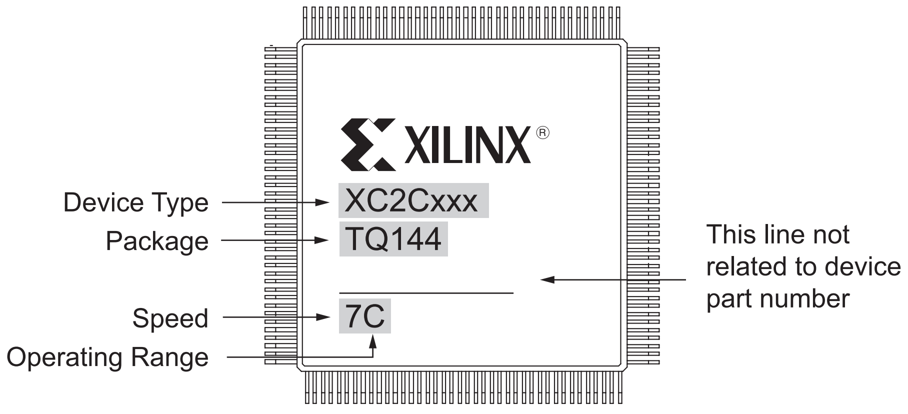
Part Number	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Comm. (C) Ind. (I)[1]
XC2C32A-6VQG44I	0.8mm	47.7	8.2	Very Thin Quad Flat Pack; Pb-free	10mm x 10mm	33	I
XC2C32A-6CPG56I	0.5mm	66.0	14.9	Chip Scale Package; Pb-free	6mm x 6mm	33	I

Notes:
1. C = Commercial (T_A = 0°C to +70°C); I = Industrial (T_A = –40°C to +85°C)





Device Part Marking



Part marking for non-chip scale package

Figure 5: **Sample Package with Part Marking**

Note: Due to the small size of chip scale and quad flat no lead packages, the complete ordering part number cannot be included on the package marking. Part marking on chip scale and quad flat no lead packages by line are:

- Line 1 = X (Xilinx logo) then truncated part number
- Line 2 = Not related to device part number
- Line 3 = Not related to device part number
- Line 4 = Package code, speed, operating temperature, three digits not related to device part number. Package codes: C3 = CP56, C4 = CPG56, Q1 = QFG32.

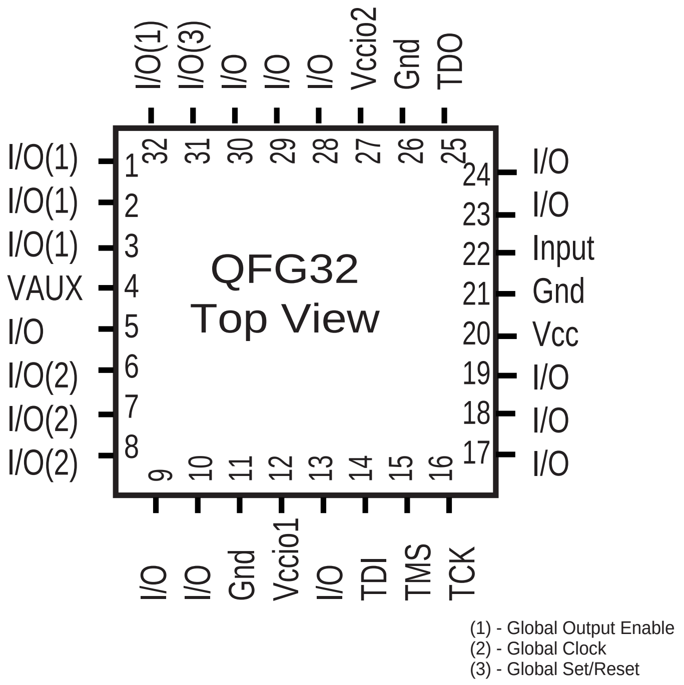


Figure 6: **QFG32 Package**

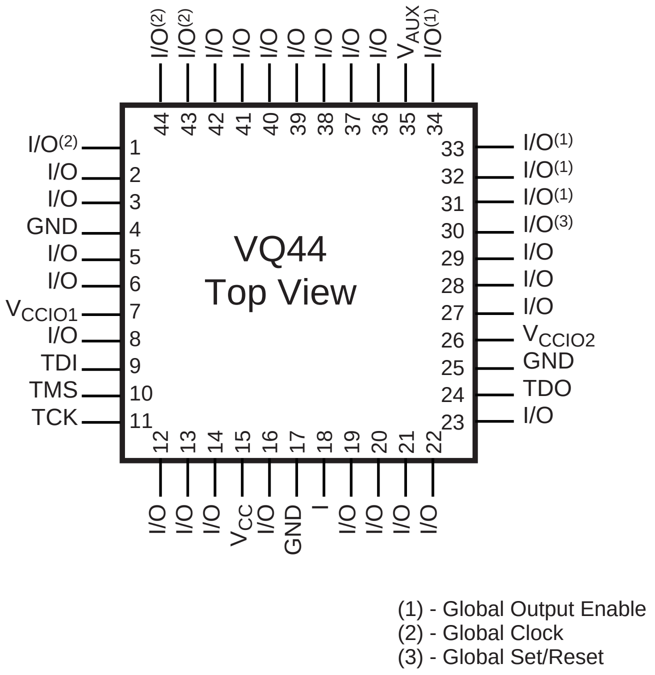


Figure 7: **VQ44 Package**

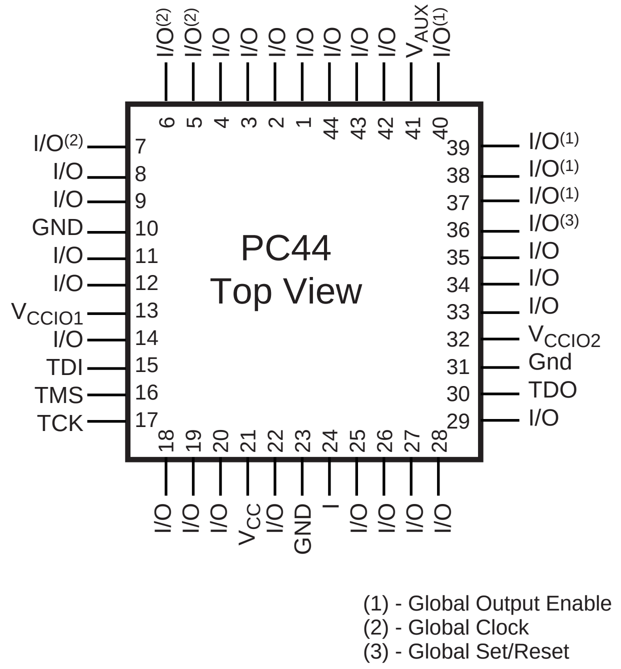
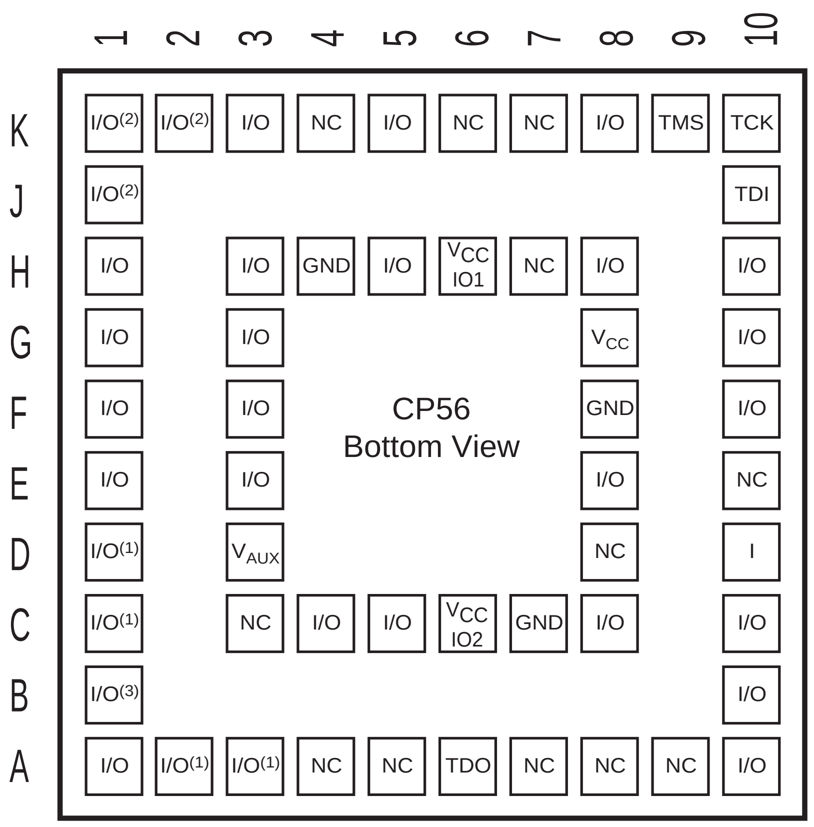


Figure 8: **PC44 Package**



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset

Figure 9: **CP56 Package**

Additional Information

[CoolRunner-II Data Sheets and Application Notes](#) [Device Package Drawings](#)

[Online Store](#)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
6/15/04	1.0	Initial Xilinx release.
8/30/04	1.1	Pb-free documentation
10/01/04	1.2	Add Asynchronous Preset/Reset Pulse Width specification to AC Electrical Characteristics.
11/08/04	1.3	Product Release. No changes to documentation.
11/22/04	1.4	Changes to output enable/disable specifications; changes to I_{CCSB}.
02/17/05	1.5	Changes to f_{TOGGLE}, t_{SLEW25}, and t_{SLEW33}
03/07/05	1.6	Improvement of pin-to-pin logic delay, page 1. Modifications to Table 1, IOSTANDARDs.
06/28/05	1.7	Move to Product Specification. Change to T_{IN25}, T_{OUT25}, T_{IN33}, and T_{OUT33}.

XC2C64A CoolRunner-II CPLD

DS311 (v1.8) June 28, 2005 **Product Specification**

Features

- Optimized for 1.8V systems
  - As fast as 4.6 ns pin-to-pin logic delays
  - As low as 15 µA quiescent current
- Industry's best 0.18 micron CMOS CPLD
  - Optimized architecture for effective logic synthesis
  - Multi-voltage I/O operation — 1.5V to 3.3V
- Available in multiple package options
  - 44-pin PLCC with 33 user I/O
  - 44-pin VQFP with 33 user I/O
  - 48-land QFN with 37 user I/O
  - 56-ball CP BGA with 45 user I/O
  - 100-pin VQFP with 64 user I/O
  - Pb-free available for all packages
- Advanced system features
  - Fastest in system programming
    - 1.8V ISP using IEEE 1532 (JTAG) interface
  - IEEE1149.1 JTAG Boundary Scan Test
  - Optional Schmitt-trigger input (per pin)
  - Two separate I/O banks
  - RealDigital™ 100% CMOS product term generation
  - Flexible clocking modes
    - Optional DualEDGE triggered registers
  - Global signal options with macrocell control
    - Multiple global clocks with phase selection per macrocell
    - Multiple global output enables
    - Global set/reset
  - Efficient control term clocks, output enables and set/resets for each macrocell and shared across function blocks
  - Advanced design security
  - Optional bus-hold, 3-state or weak pullup on selected I/O pins
  - Open-drain output option for Wired-OR and LED drive
  - Optional configurable grounds on unused I/Os
  - Mixed I/O voltages compatible with 1.5V, 1.8V, 2.5V, and 3.3V logic levels
  - PLA architecture
    - Superior pinout retention
    - 100% product term routability across function block
  - Hot pluggable

Refer to the CoolRunner™-II family data sheet for architecture description.

Description

The CoolRunner-II 64-macrocell device is designed for both high performance and low power applications. This lends power savings to high-end communication equipment and high speed to battery operated devices. Due to the low power stand-by and dynamic operation, overall system reliability is improved

This device consists of four Function Blocks inter-connected by a low power Advanced Interconnect Matrix (AIM). The AIM feeds 40 true and complement inputs to each Function Block. The Function Blocks consist of a 40 by 56 P-term PLA and 16 macrocells which contain numerous configuration bits that allow for combinational or registered modes of operation.

Additionally, these registers can be globally reset or preset and configured as a D or T flip-flop or as a D latch. There are also multiple clock signals, both global and local product term types, configured on a per macrocell basis. Output pin configurations include slew rate limit, bus hold, pull-up, open drain and programmable grounds. A Schmitt trigger input is available on a per input pin basis. In addition to storing macrocell output states, the macrocell registers may be configured as "direct input" registers to store signals directly from input pins.

Clocking is available on a global or Function Block basis. Three global clocks are available for all Function Blocks as a synchronous clock source. Macrocell registers can be individually configured to power up to the zero or one state. A global set/reset control line is also available to asynchronously set or reset selected registers during operation. Additional local clock, synchronous clock-enable, asynchronous set/reset and output enable signals can be formed using product terms on a per-macrocell or per-Function Block basis.

A DualEDGE flip-flop feature is also available on a per macrocell basis. This feature allows high performance synchronous operation based on lower frequency clocking to help reduce the total power consumption of the device.

The CoolRunner-II 64-macrocell CPLD is I/O compatible with standard LVTTL and LVCMOS18, LVCMOS25, and LVCMOS33 (see Table 1). This device is also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

Another feature that eases voltage translation is I/O banking. Two I/O banks are available on the CoolRunner-II 64A macrocell device that permit easy interfacing to 3.3V, 2.5V, 1.8V, and 1.5V devices.

RealDigital Design Technology

Xilinx CoolRunner-II CPLDs are fabricated on a 0.18 micron process technology which is derived from leading edge FPGA product development. CoolRunner-II CPLDs employ RealDigital, a design technique that makes use of CMOS technology in both the fabrication and design methodology. RealDigital design technology employs a cascade of CMOS gates to implement sum of products instead of traditional sense amplifier methodology. Due to this technology, Xilinx CoolRunner-II CPLDs achieve both high performance and low power operation.

Supported I/O Standards

The CoolRunner-II 64 macrocell features both LVCMOS and LVTTL I/O implementations. See Table 1 for I/O standard voltages. The LVTTL I/O standard is a general purpose EIA/JEDEC standard for 3.3V applications that use an LVTTL input buffer and Push-Pull output buffer. The LVCMOS standard is used in 3.3V, 2.5V, 1.8V applications. CoolRunner-II CPLDs are also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

Table 1: **I/O Standards for XC2C64A**

IOSTANDARD Attribute	Output V_{CCIO}	Input V_{CCIO}	Input V_{REF}	Board Termination Voltage V_T
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15(1)	1.5	1.5	N/A	N/A

(1) LVCMOS15 requires Schmitt-trigger inputs.

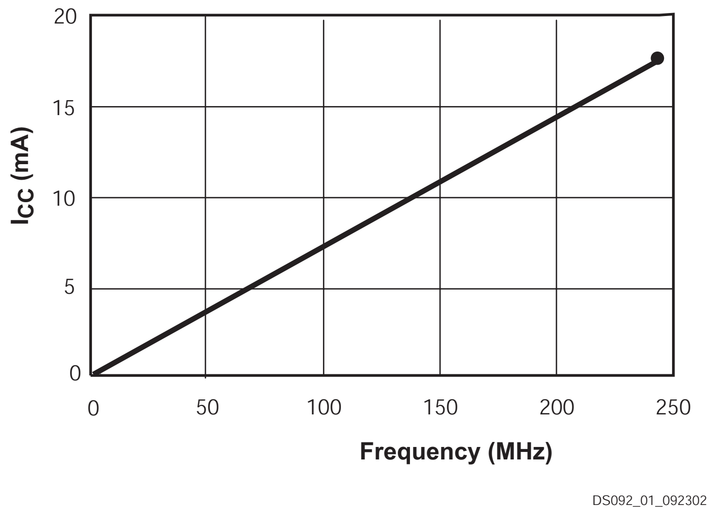


Figure 1: **I_{CC} vs Frequency**

Table 2: **I_{CC} vs Frequency (LVCMOS 1.8V T_A = 25°C)(1)**

	Frequency (MHz)									
	0	25	50	75	100	150	175	200	225	240
Typical I_{CC} (mA)	0.017	1.8	3.7	5.5	7.48	11.0	12.7	14.6	15.3	17.77

Notes:
1. 16-bit up/down, Resetable binary counter (one counter per function block).

Absolute Maximum Ratings

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to ground	–0.5 to 2.0	V
V_{CCIO}	Supply voltage for output drivers	–0.5 to 4.0	V
V_{JTAG}(2)	JTAG input voltage limits	–0.5 to 4.0	V
V_{CCAUX}	JTAG input supply voltage	–0.5 to 4.0	V
V_{IN}(1)	Input voltage relative to ground(1)	–0.5 to 4.0	V
V_{TS}(1)	Voltage applied to 3-state output(1)	–0.5 to 4.0	V
V_{STG}(3)	Storage Temperature (ambient)	–65 to +150	°C
T_J	Junction Temperature	+150	°C

Notes:
1. Maximum DC undershoot below GND must be limited to either 0.5V or 10 mA, whichever is easiest to achieve. During transitions, the device pins may undershoot to –2.0v or overshoot to +4.5V, provided this over or undershoot lasts less than 10 ns and with the forcing current being limited to 200 mA.
2. Valid over commercial temperature range.
3. For soldering guidelines and thermal considerations, see the Device Packaging information on the Xilinx website. For Pb free packages, see XAPP427.

Recommended Operating Conditions

Symbol	Parameter		Min	Max	Units
V_{CC}	Supply voltage for internal logic and input buffers	Commercial T_A = 0°C to +70°C	1.7	1.9	V
		Industrial T_A = –40°C to +85°C	1.7	1.9	V
V_{CCIO}	Supply voltage for output drivers @ 3.3V operation		3.0	3.6	V
	Supply voltage for output drivers @ 2.5V operation		2.3	2.7	V
	Supply voltage for output drivers @ 1.8V operation		1.7	1.9	V
	Supply voltage for output drivers @ 1.5V operation		1.4	1.6	V
V_{CCAUX}	JTAG programming pins		1.7	3.6	V

DC Electrical Characteristics (Over Recommended Operating Conditions)

Symbol	Parameter	Test Conditions	Typical	Max.	Units
I_{CCSB}	Standby current Commercial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	31	100	µA
I_{CCSB}	Standby current Industrial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	43	165	µA
I_{CC}(1)	Dynamic current	f = 1 MHz	-	500	µA
		f = 50 MHz	-	5	mA
C_{JTAG}	JTAG input capacitance	f = 1 MHz	-	10	pF
C_{CLK}	Global clock input capacitance	f = 1 MHz	-	12	pF
C_{IO}	I/O capacitance	f = 1 MHz	-	10	pF
I_{IL}(2)	Input leakage current	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/–1	µA
I_{IH}(2)	I/O High-Z leakage	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/–1	µA

Notes:
1. 16-bit up/down, Resetable binary counter (one counter per function block) tested at V_{CC}=V_{CCIO}= 1.9V.
2. See Quality and Reliability section of the CoolRunner-II family data sheet.

LVCMOS 3.3V and LVTTL 3.3V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		3.0	3.6	V
V_{IH}	High level input voltage		2	3.9	V
V_{IL}	Low level input voltage		–0.3	0.8	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 3V	-	0.2	V

LVCMOS 2.5V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		2.3	2.7	V
V_{IH}	High level input voltage		1.7	3.9	V
V_{IL}	Low level input voltage		–0.3	0.7	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 2.3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 2.3V	-	0.2	V

LVCMOS 1.8V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	1.7	1.9	V
V_{IH}	High level input voltage	-	0.65 x V_{CCIO}	3.9	V
V_{IL}	Low level input voltage	-	–0.3	0.35 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.2	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.7V	-	0.45	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.7V	-	0.2	V

LVCMOS 1.5V DC Voltage Specifications(1)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	1.4	1.6	V
V_{T+}	Input hysteresis threshold voltage	-	0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}		-	0.2 x V_{CCIO}	0.5 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.2	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.4V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.4V	-	0.2	V

Notes:
1. Hysteresis used on 1.5V inputs.

Schmitt Trigger Input DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	1.4	3.9	V
V_{T+}	Input hysteresis threshold voltage	-	0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}		-	0.2 x V_{CCIO}	0.5 x V_{CCIO}	V

AC Electrical Characteristics Over Recommended Operating Conditions

Symbol	Parameter	-5		-7		Units
		Min.	Max.	Min.	Max.	
T_{PD1}	Propagation delay single p-term	-	4.6	-	6.7	ns
T_{PD2}	Propagation delay OR array	-	5.0	-	7.5	ns
T_{SUD}	Direct input register clock setup time	2.4	-	3.3	-	ns
T_{SU1}	Setup time (single p-term)	2.0	-	2.5	-	ns
T_{SU2}	Setup time (OR array)	2.4	-	3.3	-	ns
T_{HD}	Direct input register hold time	0	-	0	-	ns
T_H	P-term hold time	0	-	0	-	ns
T_{CO}	Clock to output	-	3.9	-	6.0	ns
F_{TOGGLE}(1)	Internal toggle rate(1)	-	500	-	300	MHz
$F_{SYSTEM1}$(2)	Maximum system frequency(2)	-	263	-	159	MHz
$F_{SYSTEM2}$(2)	Maximum system frequency(2)	-	238	-	141	MHz
F_{EXT1}(3)	Maximum external frequency(3)	-	169	-	118	MHz
F_{EXT2}(3)	Maximum external frequency(3)	-	159	-	108	MHz
T_{PSUD}	Direct input register p-term clock setup time	0.9	-	1.7	-	ns
T_{PSU1}	P-term clock setup time (single p-term)	0.6	-	0.9	-	ns
T_{PSU2}	P-term clock setup time (OR array)	1.0	-	1.7	-	ns
T_{PHD}	Direct input register p-term clock hold time	1.3	-	1.4	-	ns
T_{PH}	P-term clock hold	1.5	-	1.7	-	ns
T_{PCO}	P-term clock to output	-	6.0	-	8.4	ns
T_{OE}/T_{OD}	Global OE to output enable/disable	-	8.0	-	10.0	ns
T_{POE}/T_{POD}	P-term OE to output enable/disable	-	9.0	-	11.0	ns
T_{MOE}/T_{MOD}	Macrocell driven OE to output enable/disable	-	9.0	-	11.0	ns
T_{PAO}	P-term set/reset to output valid	-	7.3	-	9.7	ns
T_{AO}	Global set/reset to output valid	-	6.0	-	8.3	ns
T_{SUEC}	Register clock enable setup time	3.0	-	3.7	-	ns
T_{HEC}	Register clock enable hold time	0	-	0	-	ns
T_{CW}	Global clock pulse width High or Low	1.4	-	2.2	-	ns
T_{PCW}	P-term pulse width High or Low	5.0	-	7.5	-	ns
T_{APRPW}	Asynchronous preset/reset pulse width (High or Low)	5.0	-	7.5	-	ns
T_{CONFIG}(4)	Configuration time	-	50.0	-	50.0	µs

Notes:
1. F_{TOGGLE} ($1/2*T_{CW}$) is the maximum frequency of a dual edge triggered T flip-flop with output enabled.
2. F_{SYSTEM} ($1/T_{CYCLE}$) is the internal operating frequency for a device fully populated with 16-bit up/down, Resetable binary counter (one counter per function block).
3. F_{EXT} ($1/T_{SU1}+T_{CO}$) is the maximum external frequency.
4. Typical configuration current during T_{CONFIG} is 2.3 mA.

Internal Timing Parameters

Symbol	Parameter[1]	-5		-7		Units
		Min.	Max.	Min.	Max.	
Buffer Delays						
T_{IN}	Input buffer delay	-	1.7	-	2.4	ns
T_{DIN}	Direct data register input delay	-	2.6	-	4.0	ns
T_{GCK}	Global clock buffer delay	-	1.6	-	2.5	ns
T_{GSR}	Global set/reset buffer delay	-	2.4	-	3.5	ns
T_{GTS}	Global 3-state buffer delay	-	2.7	-	3.9	ns
T_{OUT}	Output buffer delay	-	1.9	-	2.8	ns
T_{EN}	Output buffer enable/disable delay	-	5.3	-	6.1	ns
P-term Delays						
T_{CT}	Control term delay	-	2.0	-	2.5	ns
T_{LOGI1}	Single P-term delay adder	-	0.5	-	0.8	ns
T_{LOGI2}	Multiple P-term delay adder	-	0.4	-	0.8	ns
Macrocell Delay						
T_{PDI}	Input to output valid	-	0.5	-	0.7	ns
T_{SUI}	Setup before clock	1.4	-	1.8	-	ns
T_{HI}	Hold after clock	0.0	-	0.0	-	ns
T_{ECSU}	Enable clock setup time	0.9	-	1.3	-	ns
T_{ECHO}	Enable clock hold time	0	-	0	-	ns
T_{COI}	Clock to output valid	-	0.4	-	0.7	ns
T_{AOI}	Set/reset to output valid	-	1.7	-	2.0	ns
T_{CDBL}	Clock doubler delay	-	0	-	0	ns
Feedback Delays						
T_{F}	Feedback delay	-	1.5	-	2.0	ns
T_{OEM}	Macrocell to global OE delay	-	1.7	-	1.7	ns
I/O Standard Time Adder Delays 1.5VCMOS						
T_{HYS15}	Hysteresis input adder	-	4.0	-	6.0	ns
T_{OUT15}	Output adder	-	0.9	-	1.5	ns
T_{SLEW15}	Output slew rate adder	-	4.0	-	6.0	ns
I/O Standard Time Adder Delays 1.8V CMOS						
T_{HYS18}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT18}	Output adder	-	0	-	0	ns
T_{SLEW}	Output slew rate adder	-	3.5	-	5.0	ns

Internal Timing Parameters *(Continued)*

Symbol	Parameter(1)	-5		-7		Units
		Min.	Max.	Min.	Max.	
I/O Standard Time Adder Delays 2.5V CMOS						
T_{IN25}	Standard input adder	-	0.5	-	0.6	ns
T_{HYS25}	Hysteresis input adder	-	2.5	-	3.0	ns
T_{OUT25}	Output adder	-	0.8	-	0.9	ns
T_{SLEW25}	Output slew rate adder	-	4.0	-	5.0	ns
I/O Standard Time Adder Delays 3.3V CMOS/TTL						
T_{IN33}	Standard input adder	-	0.5	-	0.6	ns
T_{HYS33}	Hysteresis input adder	-	2.0	-	3.0	ns
T_{OUT33}	Output adder	-	1.2	-	1.4	ns
T_{SLEW33}	Output slew rate adder	-	4.0	-	5.0	ns

(1) 1.5 ns input pin signal rise/fall.

Switching Characteristics

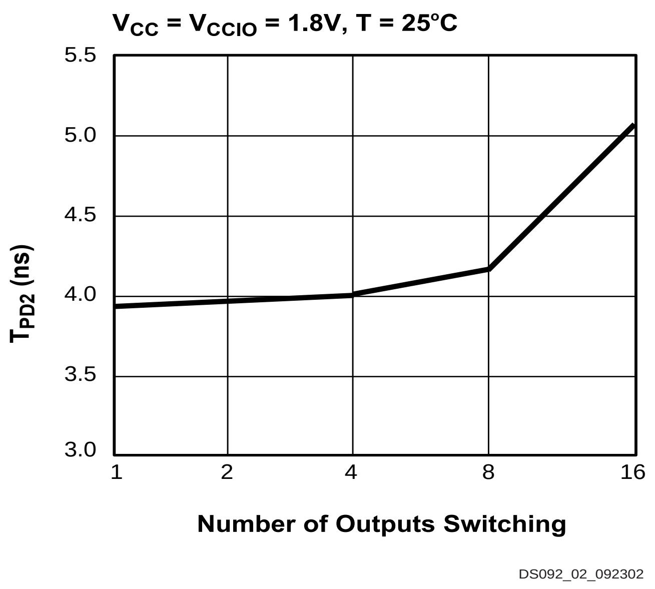


Figure 2: **Derating Curve for T_{PD}**

Typical I/O Output Curves

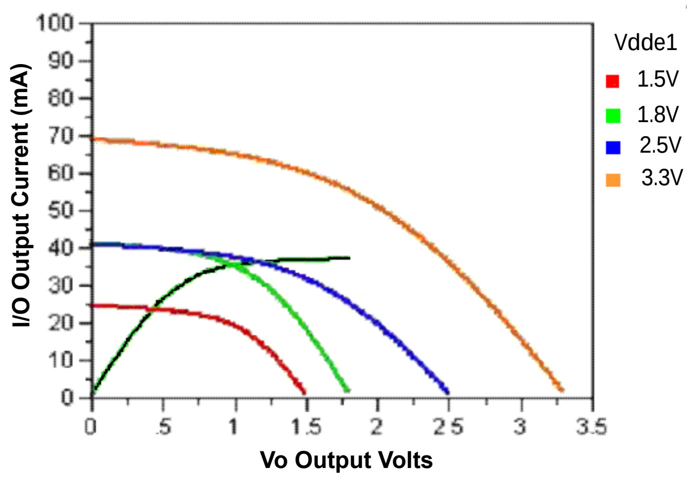
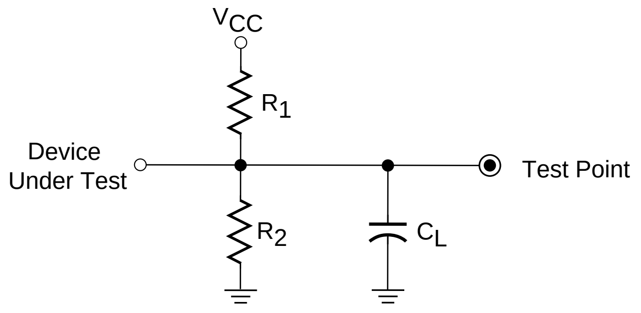


Figure 4: **Typical I/O Output Curves**

AC Test Circuit



Output Type	R_1	R_2	C_L
LVTTL33	268Ω	235Ω	35 pF
LVCMOS33	275Ω	275Ω	35 pF
LVCMOS25	188Ω	188Ω	35 pF
LVCMOS18	112.5Ω	112.5Ω	35 pF
LVCMOS15	150Ω	150Ω	35 pF

Notes:
1. C_L includes test fixtures and probe capacitance.
2. 1.5 nsec maximum rise/fall times on inputs.

DS092_03_092302

Figure 3: **AC Load Circuit**

Pin Descriptions

Function Block	Macrocell	PC44	VQ44	QFG48	CP56	VQ100	I/O Banking
1	1	44	38		F1	13	Bank 2
1	2	43	37	5	E3	12	Bank 2
1	3	42	36	4	E1	11	Bank 2
1	4	-	-		-	10	Bank 2
1	5	-	-		-	9	Bank 2
1	6	-	-		-	8	Bank 2
1	7	-	-		-	7	Bank 2
1	8	-	-		-	6	Bank 2
1(GTS1)	9	40	34	2	D1	4	Bank 2
1(GTS0)	10	39	33	1	C1	3	Bank 2
1(GTS3)	11	38	32	48	A3	2	Bank 2
1(GTS2)	12	37	31	47	A2	1	Bank 2
1(GSR)	13	36	30	46	B1	99	Bank 2
1	14	-	-		A1	97	Bank 2
1	15	-	-		C3	94	Bank 2
1	16	-	-		-	92	Bank 2
2	1	1	39	6	G1	14	Bank 1
2	2	2	40	7	F3	15	Bank 1
2	3	-	-	8	-	16	Bank 1
2	4	-	-	9	-	17	Bank 1
2	5	3	41	10	H1	18	Bank 1
2	6	4	42		G3	19	Bank 1
2(GCK0)	7	5	43	11	J1	22	Bank 1
2(GCK1)	8	6	44	12	K1	23	Bank 1
2	9	-	-		K4	24	Bank 1
2(GCK2)	10	7	1	13	K2	27	Bank 1
2	11	-	-		-	28	Bank 1
2	12	8	2	14	K3	29	Bank 1
2	13	9	3	15	H3	30	Bank 1
2	14	-	-		K5	32	Bank 1
2	15	-	-		-	33	Bank 1
2	16	-	-		-	34	Bank 1

Pin Descriptions *(Continued)*

Function Block	Macrocell	PC44	VQ44	QFG48	CP56	VQ100	I/O Banking
3	1	35	29	45	C4	91	Bank 2
3	2	34	28	44	A4	90	Bank 2
3	3	33	27	43	C5	89	Bank 2
3	4	-	-		A7	81	Bank 2
3	5	-	-	39	C8	79	Bank 2
3	6	29	23	38	A8	78	Bank 2
3	7	-	-		A9	77	Bank 2
3	8	-	-		-	76	Bank 2
3	9	-	-	37	A5	74	Bank 2
3	10	28	22	36	A10	72	Bank 2
3	11	27	21	35	B10	71	Bank 2
3	12	26	20	34	C10	70	Bank 2
3	13	-	-		D8	68	Bank 2
3	14	25	19	33	E8	67	Bank 2
3	15	24	18	32	D10	64	Bank 2
3	16	-	-		-	61	Bank 2
4	1	11	5	17	K6	35	Bank 1
4	2	12	6	18	H5	36	Bank 1
4	3	-	-		K7	37	Bank 1
4	4	-	-		-	39	Bank 1
4	5	-	-		H7	40	Bank 1
4	6	-	-		-	41	Bank 1
4	7	14	8	20	H8	42	Bank 1
4	8	-	-		-	43	Bank 1
4	9	-	-		-	49	Bank 1
4	10	-	-	24	K8	50	Bank 1
4	11	18	12	25	H10	52	Bank 1
4	12	-	-	26	-	53	Bank 1
4	13	19	13	27	G10	55	Bank 1
4	14	20	14	28	-	56	Bank 1
4	15	22	16		F10	58	Bank 1
4	16	-	-	30	E10	60	Bank 1

1. GTS = global output enable, GSR = global set reset, GCK = global clock.

XC2C64A Global, JTAG, Power/Ground and No Connect Pins

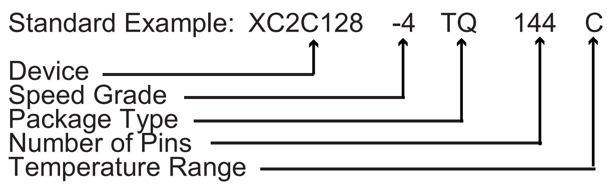
Pin Type	PC44	VQ44	QFG48	CP56	VQ100
TCK	17	11	23	K10	48
TDI	15	9	21	J10	45
TDO	30	24	40	A6	83
TMS	16	10	22	K9	47
V_{CCAUX} (JTAG supply voltage)	41	35	3	D3	5
Power internal (V_{CC})	21	15	29	G8	26,57
Power bank 1 I/O (V_{CCIO1})	13	7	19	H6	38, 51
Power bank 2 I/O (V_{CCIO2})	32	26	42	C6	88, 98
Ground	10, 23, 31	4,17,25	16, 31, 41	H4, F8, C7	21, 31, 62, 69, 84,100
No connects					20, 25, 44, 46, 54, 59, 63, 65, 66, 73, 75, 80, 82, 85, 86, 87, 93, 95, 96
Total user I/O	33	33	37	45	64

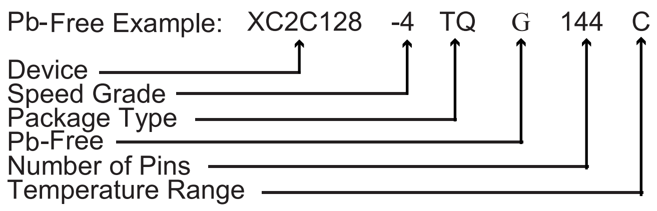
Ordering Information

Device Ordering No. and Part Marking No.	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Comm (C) Ind. (I)[1]
XC2C64A-5QFG48C	0.5mm	31.2	21.2	Quad Flat No Lead	7mm x 7mm	37	C
XC2C64A-7QFG48C	0.5mm	31.2	21.2	Quad Flat No Lead	7mm x 7mm	37	C
XC2C64A-5PC44C	1.27mm	53.1	28.7	Plastic Leaded Chip Carrier	16.5mm x 16.5mm	33	C
XC2C64A-7PC44C	1.27mm	53.1	28.7	Plastic Leaded Chip Carrier	16.5mm x 16.5mm	33	C
XC2C64A-5VQ44C	0.8mm	46.6	8.2	Very Thin Quad Flat Pack	10mm x 10mm	33	C
XC2C64A-7VQ44C	0.8mm	46.6	8.2	Very Thin Quad Flat Pack	10mm x 10mm	33	C
XC2C64A-5CP56C	0.5mm	65.0	15.0	Chip Scale Package	6mm x 6mm	45	C
XC2C64A-7CP56C	0.5mm	65.0	15.0	Chip Scale Package	6mm x 6mm	45	C
XC2C64A-5VQ100C	0.5mm	53.2	14.6	Very Thin Quad Flat Pack	14mm x 14mm	64	C
XC2C64A-7VQ100C	0.5mm	53.2	14.6	Very Thin Quad Flat Pack	14mm x 14mm	64	C
XC2C64A-5PCG44C	1.27mm	53.1	28.7	Plastic Leaded Chip Carrier; Pb-free	16.5mm x 16.5mm	33	C
XC2C64A-7PCG44C	1.27mm	53.1	28.7	Plastic Leaded Chip Carrier; Pb-free	16.5mm x 16.5mm	33	C
XC2C64A-5VQG44C	0.8mm	46.6	8.2	Very Thin Quad Flat Pack; Pb-free	10mm x 10mm	33	C
XC2C64A-7VQG44C	0.8mm	46.6	8.2	Very Thin Quad Flat Pack; Pb-free	10mm x 10mm	33	C
XC2C64A-5CPG56C	0.5mm	65.0	15.0	Chip Scale Package; Pb-free	6mm x 6mm	45	C
XC2C64A-7CPG56C	0.5mm	65.0	15.0	Chip Scale Package; Pb-free	6mm x 6mm	45	C

Device Ordering No. and Part Marking No.	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Comm (C) Ind. (I)(1)
XC2C64A-5VQG100C	0.5mm	53.2	14.6	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	64	C
XC2C64A-7VQG100C	0.5mm	53.2	14.6	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	64	C
XC2C64A-7PC44I	1.27mm	53.1	28.7	Plastic Leaded Chip Carrier	16.5mm x 16.5mm	33	I
XC2C64A-7VQ44I	0.8mm	46.6	8.2	Very Thin Quad Flat Pack	10mm x 10mm	33	I
XC2C64A-7QFG48I	0.5mm	31.2	21.2	Quad Flat No Lead; Pb-free	7mm x 7mm	37	I
XC2C64A-7CP56I	0.5mm	65.0	15.0	Chip Scale Package	6mm x 6mm	45	I
XC2C64A-7VQ100I	0.5mm	53.2	14.6	Very Thin Quad Flat Pack	14mm x 14mm	64	I
XC2C64A-7PCG44I	1.27mm	53.1	28.7	Plastic Leaded Chip Carrier; Pb-free	16.5mm x 16.5mm	33	I
XC2C64A-7VQG44I	0.8mm	46.6	8.2	Very Thin Quad Flat Pack; Pb-free	10mm x 10mm	33	I
XC2C64A-7CPG56I	0.5mm	65.0	15.0	Chip Scale Package; Pb-free	6mm x 6mm	45	I
XC2C64A-7VQG100I	0.5mm	53.2	14.6	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	64	I

Notes:
1. C = Commercial (T_A = 0°C to +70°C); I = Industrial (T_A = –40°C to +85°C).





Device Part Marking

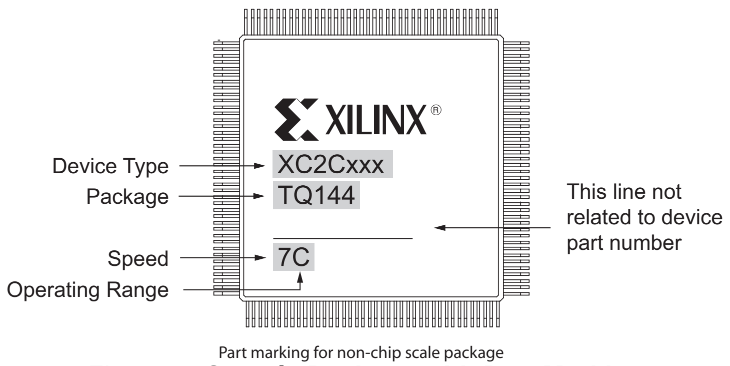
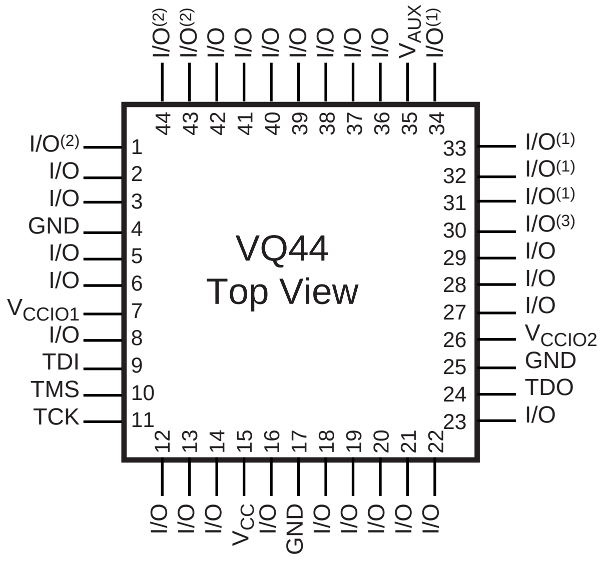

XILINX
XC2Cxxx
TQ144
7C
Device Type
Package
Speed
Operating Range
This line not related to device part number
Part marking for non-chip scale package

Figure 5: **Sample Package with Part Marking**

Note: Due to the small size of chip scale and quad flat no lead packages, the complete ordering part number cannot be included on the package marking. Part marking on chip scale and quad flat no lead packages by line are:

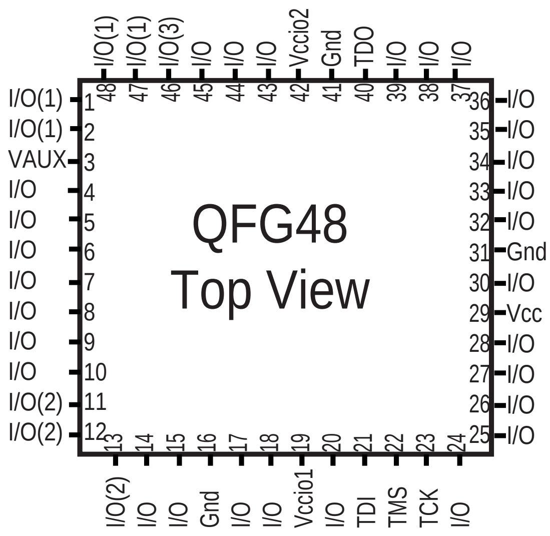
1. X (Xilinx logo) then truncated part number
2. Not related to device part number
3. Not related to device part number
4. Device code, speed, operating temperature, three digits not related to device part number. Device codes: C3 = CP56, C4 = CPG56, Q2 = QFG48.

Package Pinout Diagrams



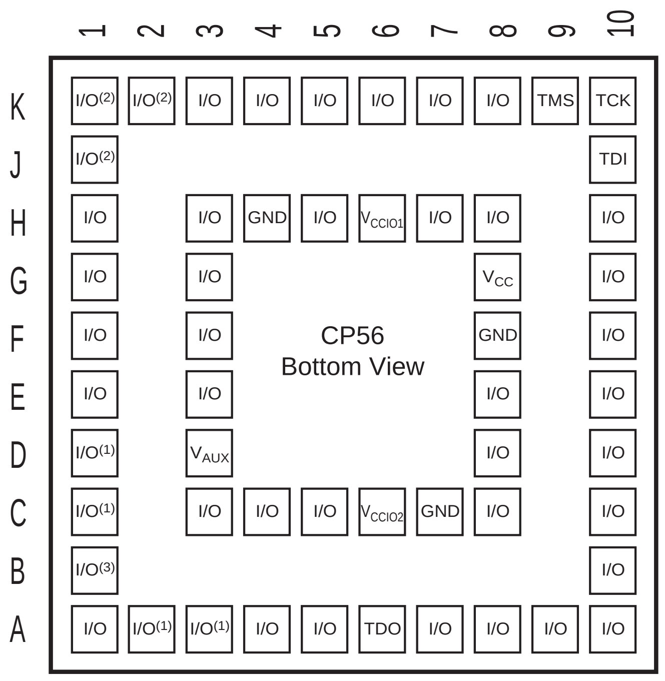
(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset

Figure 6: **VQ44 Package**



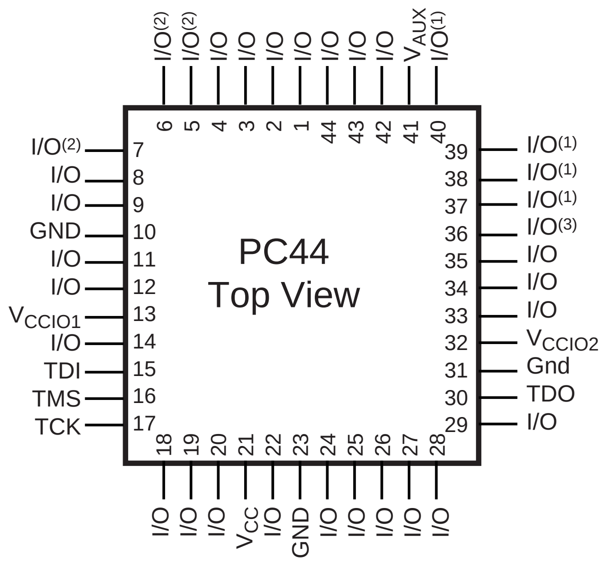
(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset

Figure 8: **QFG48 Package**


1 2 3 4 5 6 7 8 9 10
K I/O(2) I/O(2) I/O I/O I/O I/O I/O I/O TMS TCK
J I/O(2) TDI
H I/O I/O GND I/O VCCIO1 I/O I/O I/O
G I/O I/O VCC I/O
F I/O I/O GND I/O
E I/O I/O I/O I/O
D I/O(1) VAUX I/O I/O
C I/O(1) I/O I/O I/O VCCIO2 GND I/O I/O
B I/O(3) I/O
A I/O I/O(1) I/O(1) I/O I/O TDO I/O I/O I/O I/O
CP56
Bottom View

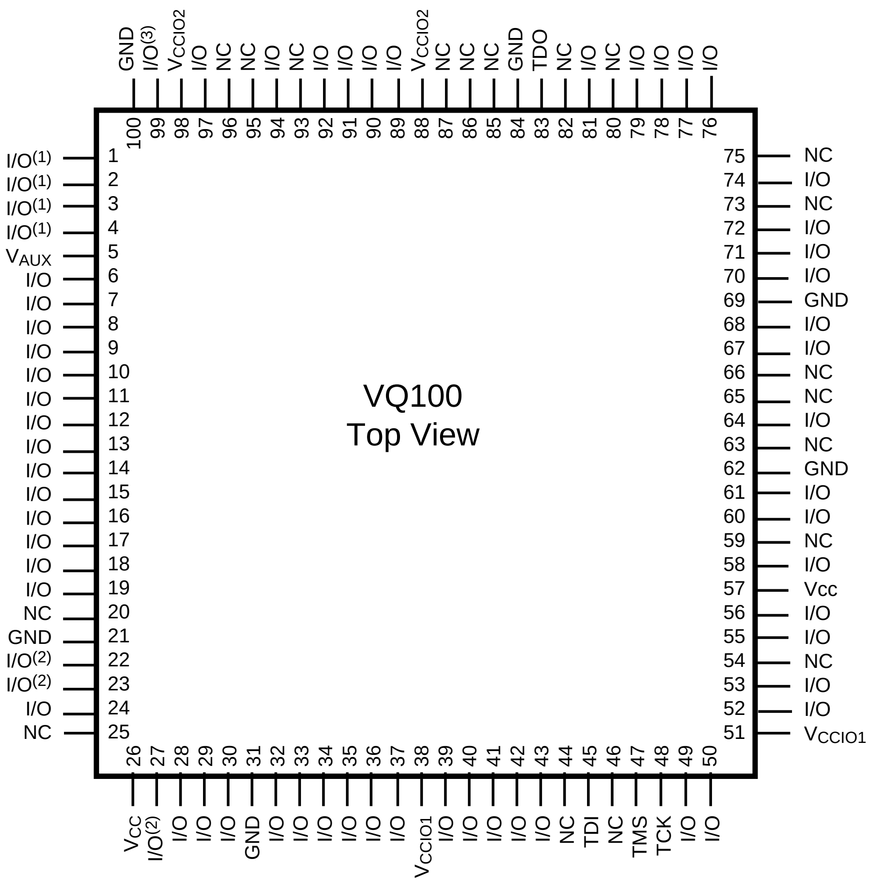
(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset

Figure 9: **CP56 Package**



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset

Figure 7: **PC44 Package**



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset

Figure 12: **VQ100 Package**

Additional Information

CoolRunner-II Data Sheets and Application Notes

Package Drawings

Online Store

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
5/15/04	1.0	Initial Xilinx release.
8/30/04	1.1	Pb-free documentation
10/01/04	1.2	Add Asynchronous Preset/Reset Pulse Width specification to AC Electrical Characteristics.
11/08/04	1.3	Product Release. No change to documentation.
11/29/04	1.4	Change to QFG package drawing (Figure 8). Pin 29 relabelled.
12/14/04	1.5	Changes to Figure 4, Typical I/O Output Curves; Changes to t_{OUT25} and t_{OUT33}, Internal Timing Parameters, page 8.
01/18/05	1.6	Changes to I_{CCSB}, f_{TOGGLE}, t_{PSU1}, t_{PSU2}, t_{PHD}, t_{CW}, t_{SLEW25}, and t_{SLEW33}
03/07/05	1.7	Format change to specifications I_{IL} and I_{IH}, page 3. Improvement to pin-to-pin logic delay, page 1. Modifications to Table 1, IOSTANDARDs.
06/28/05	1.8	Move to Product Specification. Change to T_{IN25}, T_{OUT25}, T_{IN33}, and T_{OUT33}.

XC2C128 CoolRunner-II CPLD

DS093 (v2.9) June 28, 2005 **Product Specification**

Features

- Optimized for 1.8V systems
  - As fast as 5.7 ns pin-to-pin delays
  - As low as 13 μA quiescent current
- Industry's best 0.18 micron CMOS CPLD
  - Optimized architecture for effective logic synthesis
  - Multi-voltage I/O operation — 1.5V to 3.3V
- Available in multiple package options
  - 100-pin VQFP with 80 user I/O
  - 144-pin TQFP with 100 user I/O
  - 132-ball CP (0.5mm) BGA with 100 user I/O
  - Pb-free available for all packages
- Advanced system features
  - Fastest in system programming
    - 1.8V ISP using IEEE 1532 (JTAG) interface
  - IEEE1149.1 JTAG Boundary Scan Test
  - Optional Schmitt-trigger input (per pin)
  - Unsurpassed low power management
    - DataGATE enable (DGE) signal control
  - Two separate I/O banks
  - RealDigital 100% CMOS product term generation
  - Flexible clocking modes
    - Optional DualEDGE triggered registers
    - Clock divider (divide by 2,4,6,8,10,12,14,16)
    - CoolCLOCK
  - Global signal options with macrocell control
    - Multiple global clocks with phase selection per macrocell
    - Multiple global output enables
    - Global set/reset
  - Advanced design security
  - Open-drain output option for Wired-OR and LED drive
  - PLA architecture
    - Superior pinout retention
    - 100% product term routability across function block
  - Optional bus-hold, 3-state or weak pull-up on selected I/O pins
  - Optional configurable grounds on unused I/Os
  - Mixed I/O voltages compatible with 1.5V, 1.8V, 2.5V, and 3.3V logic levels
    - SSTL2-1, SSTL3-1, and HSTL-1 I/O compatibility
  - Hot pluggable

Refer to the CoolRunner™-II family data sheet for architecture description.

Description

The CoolRunner-II 128-macrocell device is designed for both high performance and low power applications. This lends power savings to high-end communication equipment and high speed to battery operated devices. Due to the low power stand-by and dynamic operation, overall system reliability is improved

This device consists of eight Function Blocks inter-connected by a low power Advanced Interconnect Matrix (AIM). The AIM feeds 40 true and complement inputs to each Function Block. The Function Blocks consist of a 40 by 56 P-term PLA and 16 macrocells which contain numerous configuration bits that allow for combinational or registered modes of operation.

Additionally, these registers can be globally reset or preset and configured as a D or T flip-flop or as a D latch. There are also multiple clock signals, both global and local product term types, configured on a per macrocell basis. Output pin configurations include slew rate limit, bus hold, pull-up, open drain and programmable grounds. A Schmitt-trigger input is available on a per input pin basis. In addition to storing macrocell output states, the macrocell registers may be configured as direct input registers to store signals directly from input pins.

Clocking is available on a global or Function Block basis. Three global clocks are available for all Function Blocks as a synchronous clock source. Macrocell registers can be individually configured to power up to the zero or one state. A global set/reset control line is also available to asynchronously set or reset selected registers during operation. Additional local clock, synchronous clock-enable, asynchronous set/reset and output enable signals can be formed using product terms on a per-macrocell or per-Function Block basis.

A DualEDGE flip-flop feature is also available on a per macrocell basis. This feature allows high performance synchronous operation based on lower frequency clocking to help reduce the total power consumption of the device.

Circuitry has also been included to divide one externally supplied global clock (GCK2) by eight different selections. This yields divide by even and odd clock frequencies.

The use of the clock divide (division by 2) and DualEDGE flip-flop gives the resultant CoolCLOCK feature.

DataGATE is a method to selectively disable inputs of the CPLD that are not of interest during certain points in time.

By mapping a signal to the DataGATE function, lower power can be achieved due to reduction in signal switching.

Another feature that eases voltage translation is I/O banking. Two I/O banks are available on the CoolRunner-II 128 macrocell device that permit easy interfacing to 3.3V, 2.5V, 1.8V, and 1.5V devices.

The CoolRunner-II 128 macrocell CPLD is I/O compatible with various JEDEC I/O standards (see Table 1). This device is also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

RealDigital Design Technology

Xilinx CoolRunner-II CPLDs are fabricated on a 0.18 micron process technology which is derived from leading edge FPGA product development. CoolRunner-II CPLDs employ RealDigital technology, a design technique that makes use of CMOS technology in both the fabrication and design methodology. RealDigital technology employs a cascade of CMOS gates to implement sum of products instead of traditional sense amplifier methodology. Due to this technology, Xilinx CoolRunner-II CPLDs achieve both high-performance and low power operation.

Supported I/O Standards

The CoolRunner-II 128 macrocell features LVCMOS, LVTTL, SSTL and HSTL I/O implementations. See Table 1 for I/O standard voltages. The LVTTL I/O standard is a general purpose EIA/JEDEC standard for 3.3V applications that use an LVTTL input buffer and Push-Pull output buffer. The LVCMOS standard is used in 3.3V, 2.5V, 1.8V applications. Both HSTL and SSTL make use of a V_{REF} pin for JEDEC compliance. CoolRunner-II CPLDs are also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

Table 1: **I/O Standards for XC2C128**[(1)]

IOSTANDARD Attribute	Output V_{CCIO}	Input V_{CCIO}	Input V_{REF}	Board Termination Voltage V_{TT}
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15[(2)]	1.5	1.5	N/A	N/A
HSTL_1	1.5	1.5	0.75	0.75
SSTL2_1	2.5	2.5	1.25	1.25
SSTL3_1	3.3	3.3	1.5	1.5

(1) For information on assigning Vref pins, see XAPP399

(2) LVCMOS15 requires use of Schmitt-trigger inputs.

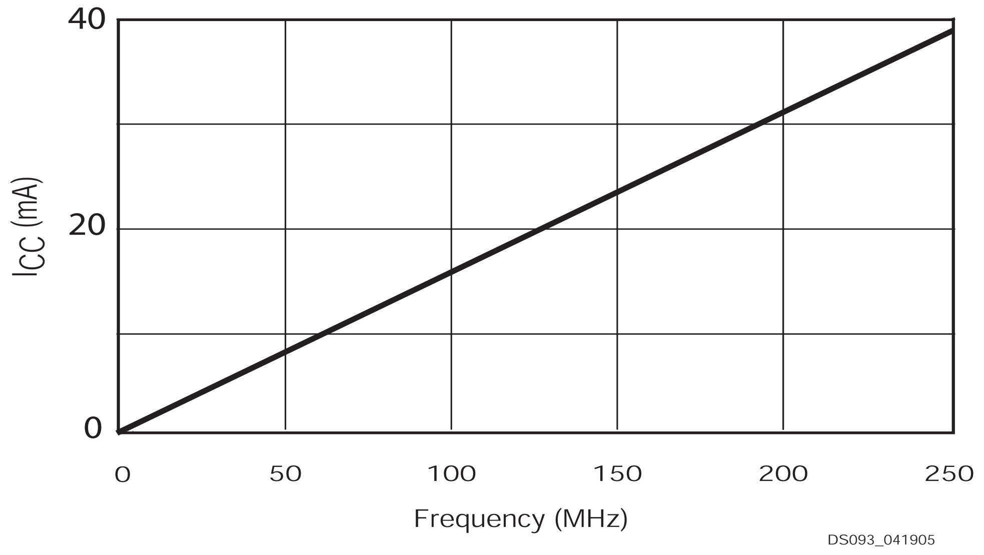


Figure 1: **I_{CC} vs Frequency**

Table 2: **I_{CC} vs Frequency (LVCMOS 1.8V T_A = 25°C)**[(1)]

	Frequency (MHz)									
	0	25	50	75	100	150	175	200	225	250
Typical I_{CC} (mA)	0.019	3.97	7.95	11.92	15.89	23.83	27.80	31.93	35.73	39.70

Notes:

1. 16-bit up/down, Resetable binary counter (one counter per function block).

Absolute Maximum Ratings

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to ground	–0.5 to 2.0	V
V_{CCIO}	Supply voltage for output drivers	–0.5 to 4.0	V
V_{JTAG}(2)	JTAG input voltage limits	–0.5 to 4.0	V
V_{CCAUX}	JTAG input supply voltage	–0.5 to 4.0	V
V_{IN}(1)	Input voltage relative to ground	–0.5 to 4.0	V
V_{TS}(1)	Voltage applied to 3-state output	–0.5 to 4.0	V
T_{STG}(3)	Storage Temperature (ambient)	–65 to +150	°C
T_J	Junction Temperature	+ 150	°C

Notes:
1. Maximum DC undershoot below GND must be limited to either 0.5V or 10 mA, whichever is easiest to achieve. During transitions, the device pins may undershoot to –2.0V or overshoot to +4.5V, provided this over or undershoot lasts less than 10 ns and with the forcing current being limited to 200 mA.
2. Valid over commercial temperature range.
3. For soldering guidelines and thermal considerations, see the Device Packaging information on the Xilinx website. For Pb-free packages, see XAPP427.

Recommended Operating Conditions

Symbol	Parameter		Min	Max	Units
V_{CC}	Supply voltage for internal logic and input buffers	Commercial T_A = 0°C to +70°C	1.7	1.9	V
		Industrial T_A = –40°C to +85°C	1.7	1.9	V
V_{CCIO}	Supply voltage for output drivers @ 3.3V operation		3.0	3.6	V
	Supply voltage for output drivers @ 2.5V operation		2.3	2.7	V
	Supply voltage for output drivers @ 1.8V operation		1.7	1.9	V
	Supply voltage for output drivers @ 1.5V operation		1.4	1.6	V
V_{CCAUX}	Supply voltage for JTAG programming		1.7	3.6	V

DC Electrical Characteristics (Over Recommended Operating Conditions)

Symbol	Parameter	Test Conditions	Typical	Max.	Units
I_{CCSB}	Standby current Commercial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	30	120	µA
I_{CCSB}	Standby current Industrial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	60	200	µA
I_{CC} (1)	Dynamic current	f = 1 MHz	-	500	µA
		f = 50 MHz	-	10	mA
C_{JTAG}	JTAG input capacitance	f = 1 MHz	-	10	pF
C_{CLK}	Global clock input capacitance	f = 1 MHz	-	12	pF
C_{IO}	I/O capacitance	f = 1 MHz	-	10	pF
I_{IL}(2)	Input leakage current	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/–1	µA
I_{IH}(2)	I/O High-Z leakage	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/–1	µA

Notes:
1. 16-bit up/down, Resetable binary counter (one counter per function block).
2. See Quality and Reliability section in CoolRunner-II family data sheet for details.

LVCMOS and LVTTL 3.3V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		3.0	3.6	V
V_{IH}	High level input voltage		2.0	3.9	V
V_{IL}	Low level input voltage		–0.3	0.8	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 3V	-	0.2	V

LVCMOS 2.5V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		2.3	2.7	V
V_{IH}	High level input voltage		1.7	3.9	V
V_{IL}	Low level input voltage		–0.3	0.7	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 2.3V	V_{CCIO} –0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 2.3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 2.3V	-	0.2	V

LVCMOS 1.8V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		1.7	1.9	V
V_{IH}	High level input voltage		0.65 x V_{CCIO}	3.9	V
V_{IL}	Low level input voltage		–0.3	0.35 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.2	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.7V	-	0.45	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.7V	-	0.2	V

LVCMOS 1.5V DC Voltage Specifications(1)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		1.4	1.6	V
V_{T+}	Input hysteresis threshold voltage		0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}			0.2 x V_{CCIO}	0.5 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.45		V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.2		V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.4V		0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.4V		0.2	V

Notes:
1. Hysteresis used on 1.5V inputs.

Schmitt Trigger Input DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage		1.4	3.9	V
V_{T+}	Input hysteresis threshold voltage		0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}			0.2 x V_{CCIO}	0.5 x V_{CCIO}	V

SSTL2-1 DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Units
V_{CCIO}	Input source voltage		2.3	2.5	2.7	V
V_{REF}(1)	Input reference voltage		1.15	1.25	1.35	V
V_{TT}(2)	Termination voltage		V_{REF} – 0.04	1.25	V_{REF} + 0.04	V
V_{IH}	High level input voltage		V_{REF} + 0.18	-	3.9	V
V_{IL}	Low level input voltage		–0.3	-	V_{REF} – 0.18	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.62	-	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 2.3V	-	-	0.54	V

Notes:
1. V_{REF} should track the variations in V_{CCIO}, also peak to peak ac noise on V_{REF} may not exceed ±2% $V_{REF.}$
2. V_{TT} of transmitting device must track V_{REF} of receiving devices.

SSTL3-1 DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Units
V_{CCIO}	Input source voltage		3.0	3.3	3.6	V
V_{REF}(1)	Input reference voltage		1.3	1.5	1.7	V
V_{TT}(2)	Termination voltage		V_{REF} – 0.05	1.5	V_{REF} + 0.05	V
V_{IH}	High level input voltage		V_{REF} + 0.2	-	V_{CCIO} + 0.3	V
V_{IL}	Low level input voltage		–0.3	-	V_{REF} – 0.2	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 3V	V_{CCIO} – 1.1	-	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 3V	-	-	0.7	V

Notes:
1. V_{REF} should track the variations in V_{CCIO}, also peak to peak ac noise on V_{REF} may not exceed ±2% $V_{REF.}$
2. V_{TT} of transmitting device must track V_{REF} of receiving devices.

HSTL1 DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Units
V_{CCIO}	Input source voltage		1.4	1.5	1.6	V
V_{REF}(1)	Input reference voltage		0.68	0.75	0.90	V
V_{TT}(2)	Termination voltage			V_{CCIO} x 0.5		V
V_{IH}	High level input voltage		V_{REF} + 0.1	-	1.9	V
V_{IL}	Low level input voltage		–0.3	-	V_{REF} – 0.1	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.7V	V_{CCIO} -0.4	-	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.7V	-	-	0.4	V

Notes:
1. V_{REF} should track the variations in V_{CCIO}, also peak to peak ac noise on V_{REF} may not exceed ±2% $V_{REF.}$
2. V_{TT} of transmitting device must track V_{REF} of receiving devices.

AC Electrical Characteristics Over Recommended Operating Conditions

Symbol	Parameter	-6		-7		Units
		Min.	Max.	Min.	Max.	
T_{PD1}	Propagation delay single p-term	-	5.7	-	7.0	ns
T_{PD2}	Propagation delay OR array	-	6.0	-	7.5	ns
T_{SUD}	Direct input register set-up time	3.6	-	4.6	-	ns
T_{SU1}	Setup time fast (single p-term)	2.4	-	3.0	-	ns
T_{SU2}	Setup time (OR array)	2.7	-	3.5	-	ns
T_{HD}	Direct input register hold time	0.0	-	0.0	-	ns
T_H	Hold time (Or array or p-term)	0.0	-	0.0	-	ns
T_{CO}	Clock to output	-	4.2	-	5.4	ns
F_{TOGGLE}(1)	Internal toggle rate	-	500	-	300	MHz
$F_{SYSTEM1}$(2)	Maximum system frequency	-	244	-	152	MHz
$F_{SYSTEM2}$(2)	Maximum system frequency	-	227	-	141	MHz
F_{EXT1}(3)	Maximum external frequency	-	152	-	114	MHz
F_{EXT2}(3)	Maximum external frequency	-	145	-	108	MHz
T_{PSUD}	Direct input register p-term clock setup time	2.5	-	3.1	-	ns
T_{PSU1}	P-term clock setup time (single p-term)	1.3	-	1.5	-	ns
T_{PSU2}	P-term clock setup time (OR array)	1.6	-	2.0	-	ns
T_{PHD}	Direct input register p-term clock hold time	0.2	-	0.2	-	ns
T_{PH}	P-term clock hold	0.7	-	1.0	-	ns
T_{PCO}	P-term clock to output	-	5.9	-	7.3	ns
T_{OE}/T_{OD}	Global OE to output enable/disable	-	5.9	-	7.5	ns
T_{POE}/T_{POD}	P-term OE to output enable/disable	-	7.0	-	8.5	ns
T_{MOE}/T_{MOD}	Macrocell driven OE to output enable/disable	-	7.7	-	9.9	ns
T_{PAO}	P-term set/reset to output valid	-	6.6	-	8.1	ns
T_{AO}	Global set/reset to output valid	-	5.0	-	7.6	ns
T_{SUEC}	Register clock enable setup time	3.1	-	3.5	-	ns
T_{HEC}	Register clock enable hold time	0.0	-	0.0	-	ns
T_{CW}	Global clock pulse width High or Low	1.1	-	1.6	-	ns
T_{APRPW}	Asynchronous preset/reset pulse width (High or Low)	6.0	-	7.5	-	ns
T_{PCW}	P-term pulse width High or Low	6.0	-	7.5	-	ns
T_{DGSU}	Set-up before DataGATE latch assertion	0.0	-	0.0	-	ns
T_{DGH}	Hold to DataGATE latch assertion	4.0	-	6.0	-	ns
T_{DGR}	DataGATE recovery to new data	-	8.2	-	9.0	ns
T_{DGW}	DataGATE low pulse width	3.0	-	4.0	-	ns
T_{CDRSU}	CDRST setup time before falling edge GCLK2	1.3	-	2.0	-	ns
T_{CDRH}	Hold time CDRST after falling edge GCLK2	0.0	-	0.0	-	ns
T_{CONFIG}(4)	Configuration time	-	350	-	350	us

Notes:
1. F_{TOGGLE} is the maximum clock frequency to which a T flip-flop can reliably toggle (see the CoolRunner-II family data sheet).
2. $F_{SYSTEM1}$ is the internal operating frequency for a device with 16-bit resetable binary counter through one p-term per macrocell while $F_{SYSTEM2}$ is through the OR array (one counter per function block).
3. F_{EXT1} ($1/T_{SU1}+T_{CO}$) is the maximum external frequency using one p-term while F_{EXT2} is through the OR array.
4. Typical configuration current during T_{CONFIG} is 10 mA.

Internal Timing Parameters

Symbol	Parameter(1)	-6		-7		Units
		Min.	Max.	Min.	Max.	
Buffer Delays						
T_{IN}	Input buffer delay	-	2.0	-	2.6	ns
T_{DIN}	Direct data register input delay	-	3.3	-	4.9	ns
T_{GCK}	Global Clock buffer delay	-	1.5	-	2.7	ns
T_{GSR}	Global set/reset buffer delay	-	1.6	-	3.5	ns
T_{GTS}	Global 3-state buffer delay	-	1.8	-	3.0	ns
T_{OUT}	Output buffer delay	-	2.3	-	2.6	ns
T_{EN}	Output buffer enable/disable delay	-	3.8	-	4.0	ns
P-term Delays						
T_{CT}	Control term delay	-	0.6	-	1.4	ns
T_{LOGI1}	Single P-term delay adder	-	0.5	-	1.1	ns
T_{LOGI2}	Multiple P-term delay adder	-	0.3	-	0.5	ns
Macrocell Delay						
T_{PDI}	Input to output valid	-	0.9	-	0.7	ns
T_{LDI}	Setup before clock (transparent latch)	-	2.1	-	2.5	ns
T_{SUI}	Setup before clock	1.4	-	1.8	-	ns
T_{HI}	Hold after clock	0.0	-	0.0	-	ns
T_{ECSU}	Enable clock setup time	1.4	-	1.8	-	ns
T_{ECHO}	Enable clock hold time	0.0	-	0.0	-	ns
T_{COI}	Clock to output valid	-	0.4	-	0.7	ns
T_{AOI}	Set/reset to output valid	-	1.1	-	1.5	ns
T_{CDBL}	Clock doubler delay	-	0.0	-	0.0	ns
Feedback Delays						
T_F	Feedback delay	-	1.4	-	3.0	ns
T_{OEM}	Macrocell to global OE delay	-	1.5	-	2.0	ns
I/O Standard Time Adder Delays 1.5V CMOS						
T_{HYS15}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT15}	Output adder	-	0.8	-	1.0	ns
T_{SLEW15}	Output slew rate adder	-	4.0	-	4.0	ns
I/O Standard Time Adder Delays 1.8V CMOS						
T_{HYS18}	Hysteresis input adder	-	2.0	-	4.0	ns
T_{OUT18}	Output adder	-	0.0	-	0.0	ns
T_{SLEW18}	Output slew rate adder	-	2.5	-	4.0	ns

Internal Timing Parameters *(Continued)*

Symbol	Parameter[1]	-6 Min.	-6 Max.	-7 Min.	-7 Max.	Units
I/O Standard Time Adder Delays 2.5V CMOS						
T_{IN25}	Standard input adder	-	0.6	-	0.7	ns
T_{HYS25}	Hysteresis input adder	-	1.5	-	3.0	ns
T_{OUT25}	Output adder	-	0.8	-	0.9	ns
T_{SLEW25}	Output slew rate adder	-	3.0	-	4.0	ns
I/O Standard Time Adder Delays 3.3V CMOS/TTL						
T_{IN33}	Standard input adder	-	0.5	-	0.6	ns
T_{HYS33}	Hysteresis input adder	-	1.2	-	3.0	ns
T_{OUT33}	Output adder	-	1.2	-	1.4	ns
T_{SLEW33}	Output slew rate adder	-	3.0	-	4.0	ns
I/O Standard Time Adder Delays HSTL, SSTL						
SSTL2-1	Input adder to T_{IN}, T_{DIN}, T_{GCK}, T_{GSR}, T_{GTS}	-	0.8	-	2.5	ns
	Output adder to T_{OUT}	-	0.5	-	0.5	ns
SSTL3-1	Input adder to T_{IN}, T_{DIN}, T_{GCK}, T_{GSR}, T_{GTS}	-	0.8	-	2.5	ns
	Output adder to T_{OUT}	-	0.5	-	0.5	ns
HSTL-1	Input adder to T_{IN}, T_{DIN}, T_{GCK}, T_{GSR}, T_{GTS}	-	2.0	-	2.5	ns
	Output adder to T_{OUT}	-	0.0	-	0.0	ns

Notes:
1. 1.5 ns input pin signal rise/fall.

Switching Characteristics

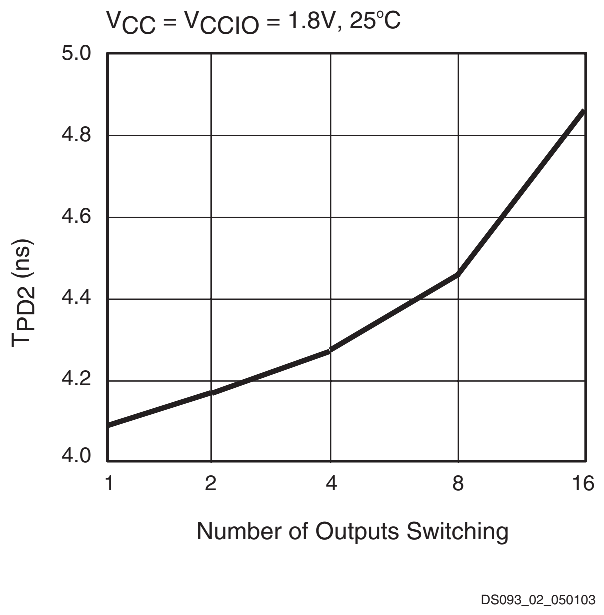
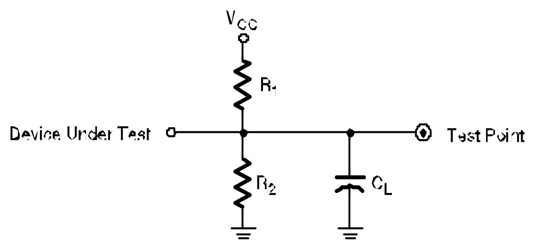


Figure 2: **Derating Curve for T_{PD}**

Switching Test Conditions



Output Type	V_{CCIO}	V_{CC}	R_1	R_2	C_L
LVTTL33	3.3V	3.0V	268Ω	235Ω	35 pF
LVCMOS33	3.3V	3.0V	275Ω	275Ω	35 pF
LVCMOS25	2.5V	2.3V	188Ω	188Ω	35pF
LVCMOS18	1.8V	1.7V	112.5Ω	112.5Ω	35pF
LVCMOS15	1.5V	1.4V	150Ω	150Ω	35pF

C_L Includes test fixtures and probe capacitance

DS_ACT_08_14_02

Figure 3: **AC Load Circuits**

Typical I/V Output Curves

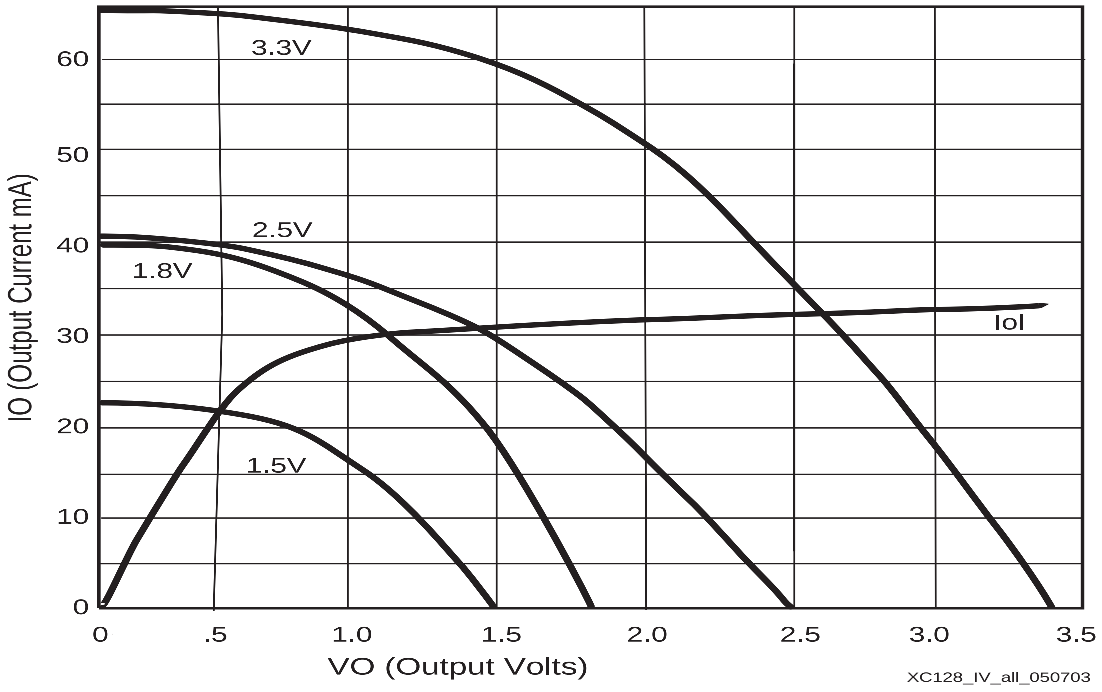


Figure 4: **Typical I/V Curves for XC2C128**

Pin Descriptions

Function Block	Macro-cell	VQ100	CP132	TQ144	I/O Bank
1	1	13	G1	17	2
1	2	-	F1	16	2
1	3	12	F2	15	2
1	4	11	F3	14	2
1	5	10	E1	13	2
1	6	9	E2	12	2
1	7	-	-	-	-
1	8	-	-	-	-
1	9	-	-	-	-
1	10	-	-	-	-
1	11	8	E3	11	2
1	12	7	D1	10	2
1	13	6	D2	9	2
1	14	-	C1	7	2
1(GTS1)	15	4	C2	6	2
1(GTS0)	16	3	C3	5	2

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	I/O Bank
2	1	-	G2	19	1
2	2	14	G3	21	1
2	3	15	H1	22	1
2	4	16	H2	23	1
2	5	17	H3	24	1
2	6	18	J1	25	1
2	7	-	-	-	-
2	8	-	-	-	-
2	9	-	-	-	-
2	10	-	-	-	-
2	11	19	J2	26	1
2	12	-	K1	28	1
2(GCK0)	13	22	K3	30	1
2(GCK1)	14	23	L2	32	1
2(CDRST)	15	24	M2	35	1
2(GCK2)	16	27	N2	38	1

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	I/O Bank
3	1	-	B1	4	2
3(GTS3)	2	2	B2	3	2
3(GTS2)	3	1	A1	2	2
3(GSR)	4	99	A3	143	2
3	5	97	B4	140	2
3	6	96	A4	138	2
3	7	95	C5	136	2
3	8	-	-	-	-
3	9	-	-	-	-
3	10	-	-	-	-
3	11	94	B5	134	2
3	12		A5	133	2
3	13	93	C6	132	2
3	14	92	B6	131	2
3	15	91	A6	130	2
3	16	90	C7	129	2
4(DGE)	1	28	P2	39	1
4	2	-	M3	40	1
4	3	-	N3	41	1
4	4	29	P3	43	1
4	5	30	M4	45	1
4	6	32	M5	49	1
4	7	33	N5	50	1
4	8	-	-	-	-
4	9	-	-	-	-
4	10	-	-	-	-
4	11	34	P5	51	1
4	12	35	M6	52	1
4	13	36	N6	53	1
4	14	37	P6	54	1
4	15	39	N7	56	1
4	16	40	M7	57	1

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	I/O Bank
5	1	65	G13	94	2
5	2	66	G12	95	2
5	3	67	F14	96	2
5	4	-	F13	97	2
5	5	68	F12	98	2
5	6	-	E13	100	2
5	7	70	E12	101	2
5	8	-	-	-	-
5	9	-	-	-	-
5	10	-	-	-	-
5	11	71	D14	102	2
5	12	72	D13	103	2
5	13	73	D12	104	2
5	14	74	C14	105	2
5	15	76	B13	110	2
5	16	-	A13	111	2
6	1	64	H12	92	1
6	2	63	H13	91	1
6	3	61	J13	88	1
6	4	60	J12	87	1
6	5	59	K14	86	1
6	6	58	K13	85	1
6	7	-	-	-	-
6	8	-	-	-	-
6	9	-	-	-	-
6	10	-	-	-	-
6	11	-	L14	83	1
6	12	56	L13	82	1
6	13	-	L12	81	1
6	14	55	M14	80	1
6	15	-	M13	79	1
6	16	54	M12	78	1

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	I/O Bank
7	1	77	C12	112	2
7	2	78	B12	113	2
7	3	-	A12	115	2
7	4	79	C11	116	2
7	5	80	B11	117	2
7	6	81	A11	118	2
7	7	-	C10	119	2
7	8	-	-	-	-
7	9	-	-	-	-
7	10	-	-	-	-
7	11	82	A10	120	2
7	12	-	C9	121	2
7	13	85	A8	124	2
7	14	86	B8	125	2
7	15	87	C8	126	2
7	16	89	B7	128	2

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	I/O Bank
8	1	-	N14	77	1
8	2	53	N13	76	1
8	3	52	P14	74	1
8	4	50	P12	71	1
8	5	-	M11	70	1
8	6	49	N11	69	1
8	7	-	-	-	-
8	8	-	-	-	-
8	9	-	-	-	-
8	10	-	-	-	-
8	11	-	P11	68	1
8	12	46	P10	64	1
8	13	44	P9	61	1
8	14	43	M8	60	1
8	15	42	N8	59	1
8	16	41	P8	58	1

Notes:
1. GTS = global output enable, GSR = global reset/set, GCK = global clock, CDRST = clock divide reset, DGE = DataGATE enable.

XC2C128 JTAG, Power/Ground, No Connect Pins and Total User I/O

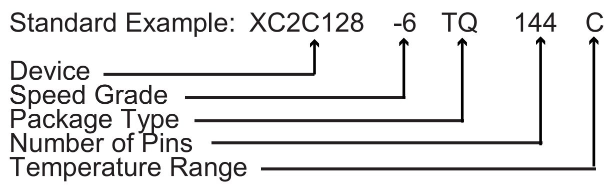
Pin Type	VQ100[(1)]	CP132[(1)]	TQ144[(1)]
TCK	48	M10	67
TDI	45	M9	63
TDO	83	B9	122
TMS	47	N10	65
V_{CCAUX} (JTAG supply voltage)	5	D3	8
Power internal (V_{CC})	26, 57	P1, K12, A2	1, 37, 84
Power Bank 1 I/O (V_{CCIO1})	20, 38, 51	J3, P7, G14, P13	27, 55, 73, 93
Power Bank 2 I/O (V_{CCIO2})	88, 98	A14, C4, A7	109, 127, 141
Ground	21, 25, 31, 62, 69, 75, 84, 100	K2, N1, P4, N9, N12, J14, H14, E14, B14, A9, B3	29, 36, 47, 62, 72, 89, 90, 99, 108, 123, 144
No connects	-	L1, L3, M1, N4, C13, B10	18, 20, 31, 33, 34, 42, 44, 46, 48, 66, 75, 106, 107, 114, 135, 137, 139, 142
Total user I/O (including dual function pins)	80	100	100

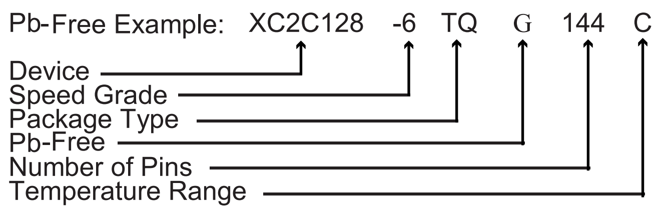
Notes:
1. Pin compatible with all larger and smaller densities except where I/O banking is used.

Ordering Information

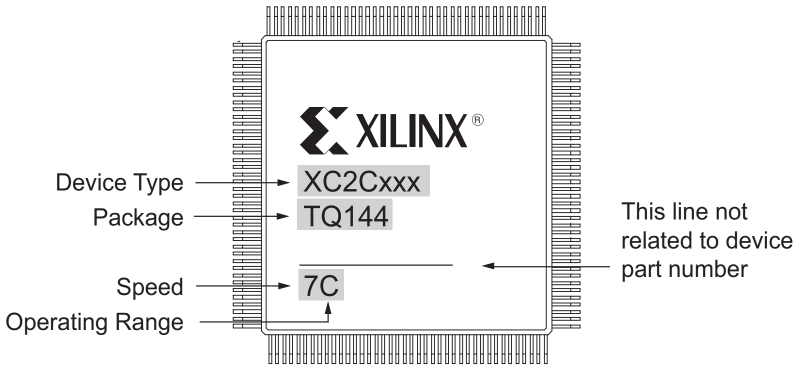
Part Number	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Comm. (C) Ind. (I)[1]
XC2C128-6VQ100C	0.5mm	47.5	12.5	Very Thin Quad Flat Pack	14mm x 14mm	80	C
XC2C128-7VQ100C	0.5mm	47.5	12.5	Very Thin Quad Flat Pack	14mm x 14mm	80	C
XC2C128-6CP132C	0.5mm	72.4	15.7	Chip Scale Package	8mm x 8mm	100	C
XC2C128-7CP132C	0.5mm	72.4	15.7	Chip Scale Package	8mm x 8mm	100	C
XC2C128-6TQ144C	0.5mm	46.1	7.9	Thin Quad Flat Pack	20mm x 20mm	100	C
XC2C128-7TQ144C	0.5mm	46.1	7.9	Thin Quad Flat Pack	20mm x 20mm	100	C
XC2C128-6VQG100C	0.5mm	47.5	12.5	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	80	C
XC2C128-7VQG100C	0.5mm	47.5	12.5	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	80	C
XC2C128-6CPG132C	0.5mm	72.4	15.7	Chip Scale Package; Pb-free	8mm x 8mm	100	C
XC2C128-7CPG132C	0.5mm	72.4	15.7	Chip Scale Package; Pb-free	8mm x 8mm	100	C
XC2C128-6TQG144C	0.5mm	46.1	7.9	Thin Quad Flat Pack; Pb-free	20mm x 20mm	100	C
XC2C128-7TQG144C	0.5mm	46.1	7.9	Thin Quad Flat Pack; Pb-free	20mm x 20mm	100	C
XC2C128-7VQ100I	0.5mm	47.5	12.5	Very Thin Quad Flat Pack	14mm x 14mm	80	I
XC2C128-7CP132I	0.5mm	72.4	15.7	Chip Scale Package	8mm x 8mm	100	I
XC2C128-7TQ144I	0.5mm	46.1	7.9	Thin Quad Flat Pack	20mm x 20mm	100	I
XC2C128-7VQG100I	0.5mm	47.5	12.5	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	80	I
XC2C128-7CPG132I	0.5mm	72.4	15.7	Chip Scale Package; Pb-free	8mm x 8mm	100	I
XC2C128-7TQG144I	0.5mm	46.1	7.9	Thin Quad Flat Pack; Pb-free	20mm x 20mm	100	I

Notes:
1. C = Commercial (T_A = 0° C to +70° C); I = Industrial (T_A = –40° C to +85° C).





Device Part Marking

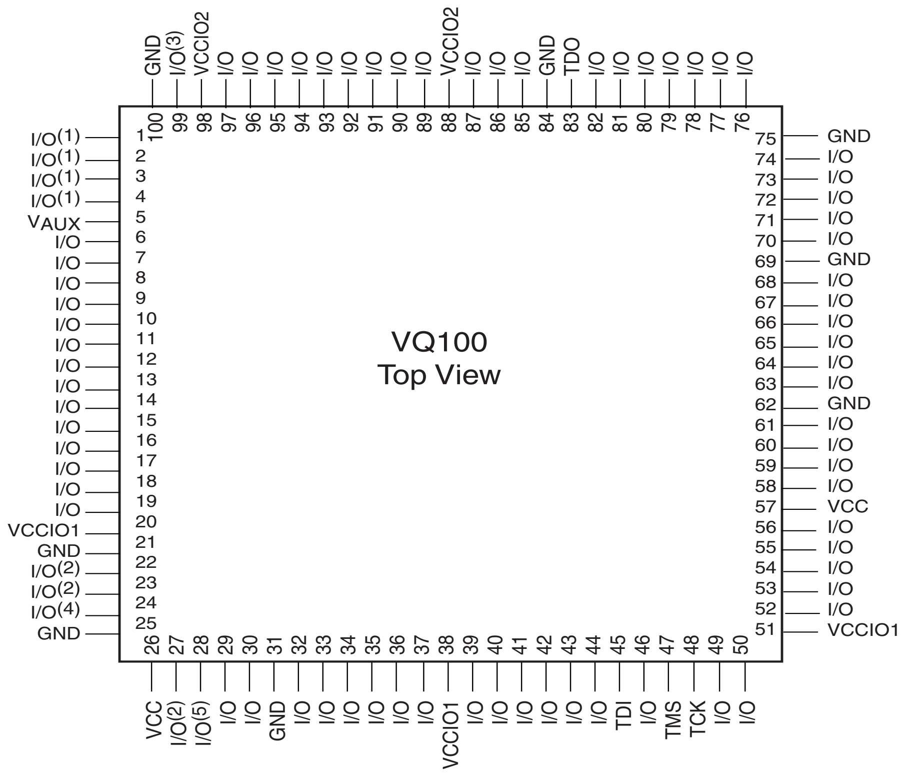


Part Marking for all non chip scale packages

Figure 5: **Sample Package with Part Marking**

Note: Due to the small size of chip scale packages, the complete ordering part number cannot be included on the package marking. Part marking on chip scale packages by line are:

- Line 1 = X (Xilinx logo) then truncated part number
- Line 2 = Not related to device part number
- Line 3 = Not related to device part number
- Line 4 = Package code, speed, operating temperature, three digits not related to device part number. Package codes: C5 = CP132, C6 = CPG132.



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - Data Gate

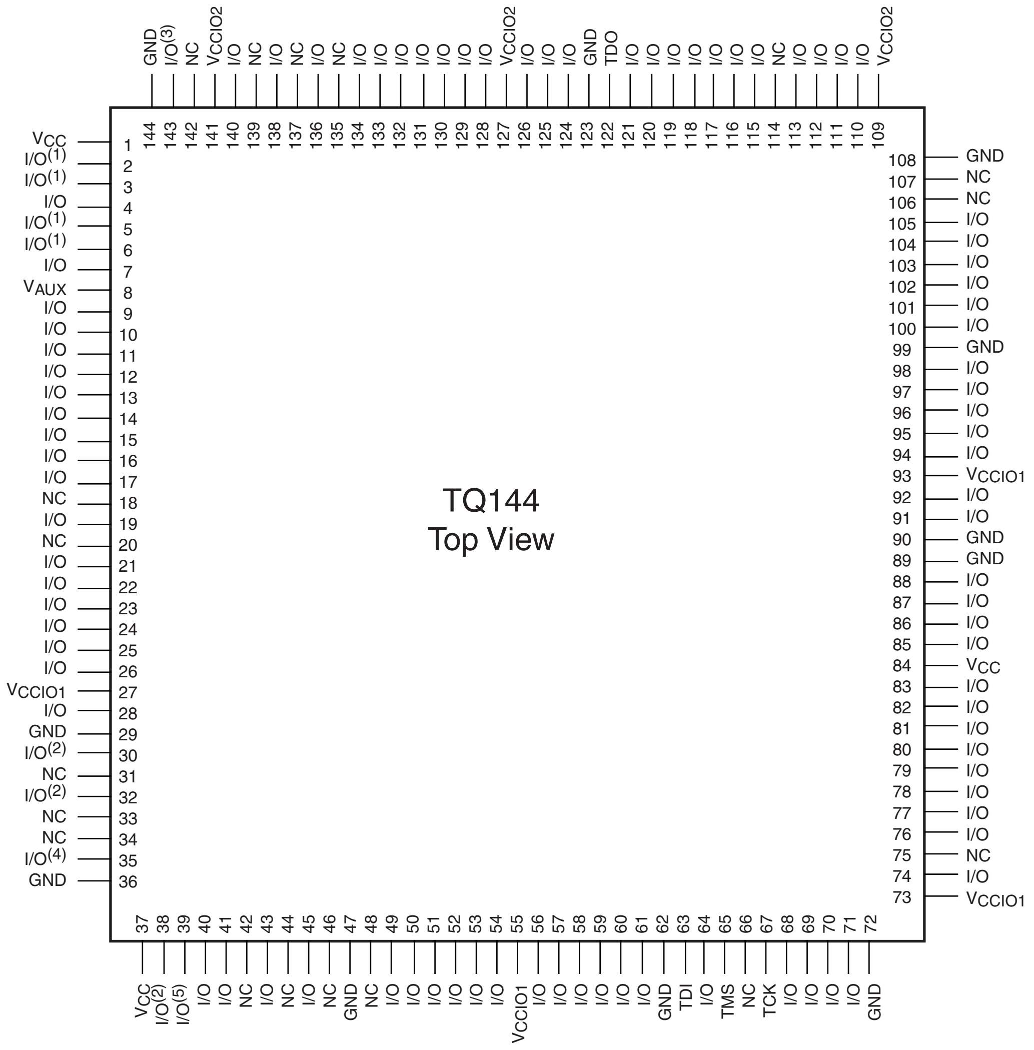
Figure 6: **VQ100 Very Thin Quad Flat Pack**

CP132
Bottom View

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P	VCC	I/O(5)	I/O	GND	I/O	I/O	VCCIO1	I/O	I/O	I/O	I/O	I/O	VCCIO1	I/O
N	GND	I/O(2)	I/O	NC	I/O	I/O	I/O	I/O	GND	TMS	I/O	GND	I/O	I/O
M	NC	I/O(4)	I/O	I/O	I/O	I/O	I/O	I/O	TDI	TCK	I/O	I/O	I/O	I/O
L	NC	I/O(2)	NC									I/O	I/O	I/O
K	I/O	GND	I/O(2)									VCC	I/O	I/O
J	I/O	I/O	VCCIO1									I/O	I/O	GND
H	I/O	I/O	I/O									I/O	I/O	GND
G	I/O	I/O	I/O									I/O	I/O	VCCIO1
F	I/O	I/O	I/O									I/O	I/O	I/O
E	I/O	I/O	I/O									I/O	I/O	GND
D	I/O	I/O	VAUX									I/O	I/O	I/O
C	I/O	I/O(1)	I/O(1)	VCCIO2	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	NC	I/O
B	I/O	I/O(1)	GND	I/O	I/O	I/O	I/O	I/O	TDO	NC	I/O	I/O	I/O	GND
A	I/O(1)	VCC	I/O(3)	I/O	I/O	I/O	VCCIO2	I/O	GND	I/O	I/O	I/O	I/O	VCCIO2

(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - DataGATE Enable

Figure 7: **CP132 Chip Scale Package**



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - DataGATE Enable

Figure 8: **TQ144 Thin Quad Flat Pack**

Additional Information

CoolRunner-II Datasheets and Application Notes

Device Packages

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/01/02	1.0	Initial Xilinx release.
5/19/03	2.0	Added bin 6, 7 characterization data.
8/25/03	2.1	Edit Package diagram, other minor formatting edits.
01/26/04	2.2	Update links.
03/01/04	2.3	Fixed cropping on Figure 6.
7/30/04	2.4	Added Pb-free documentation.
10/01/04	2.5	Add Asynchronous Preset/Reset Pulse Width specification to AC Electrical Characteristics.
01/30/05	2.6	Change to I_{CCSB} MAX for Commercial and Industrial.
03/07/05	2.7	Delete -4 speed grade. Modifications to Table 1, IOSTANDARDs.
04/21/05	2.8	Recharacterization of AC Specifications
06/28/05	2.9	Move to Product Specification.

XC2C256 CoolRunner-II CPLD

DS094 (v2.8) June 28, 2005 **Product Specification**

Features

- Optimized for 1.8V systems
  - As fast as 5.7 ns pin-to-pin delays
  - As low as 13 μA quiescent current
- Industry's best 0.18 micron CMOS CPLD
  - Optimized architecture for effective logic synthesis. Refer to the CoolRunner™-II family data sheet for architecture description.
  - Multi-voltage I/O operation — 1.5V to 3.3V
- Available in multiple package options
  - 100-pin VQFP with 80 user I/O
  - 144-pin TQFP with 118 user I/O
  - 132-ball CP (0.5mm) BGA with 106 user I/O
  - 208-pin PQFP with 173 user I/O
  - 256-ball FT (1.0mm) BGA with 184 user I/O
  - Pb-free available for all packages
- Advanced system features
  - Fastest in system programming
    - 1.8V ISP using IEEE 1532 (JTAG) interface
  - IEEE1149.1 JTAG Boundary Scan Test
  - Optional Schmitt-trigger input (per pin)
  - Unsurpassed low power management
    - DataGATE enable (DGE) signal control
  - Two separate I/O banks
  - RealDigital 100% CMOS product term generation
  - Flexible clocking modes
    - Optional DualEDGE triggered registers
    - Clock divider (divide by 2,4,6,8,10,12,14,16)
    - CoolCLOCK
  - Global signal options with macrocell control
    - Multiple global clocks with phase selection per macrocell
    - Multiple global output enables
    - Global set/reset
  - Advanced design security
  - PLA architecture
    - Superior pinout retention
    - 100% product term routability across function block
  - Open-drain output option for Wired-OR and LED drive
  - Optional bus-hold, 3-state or weak pull-up on selected I/O pins
  - Optional configurable grounds on unused I/Os
  - Mixed I/O voltages compatible with 1.5V, 1.8V, 2.5V, and 3.3V logic levels
    - SSTL2-1, SSTL3-1, and HSTL-1 I/O compatibility
  - Hot pluggable

Description

The CoolRunner™-II 256-macrocell device is designed for both high performance and low power applications. This lends power savings to high-end communication equipment and high speed to battery operated devices. Due to the low power stand-by and dynamic operation, overall system reliability is improved

This device consists of sixteen Function Blocks inter-connected by a low power Advanced Interconnect Matrix (AIM). The AIM feeds 40 true and complement inputs to each Function Block. The Function Blocks consist of a 40 by 56 P-term PLA and 16 macrocells which contain numerous configuration bits that allow for combinational or registered modes of operation.

Additionally, these registers can be globally reset or preset and configured as a D or T flip-flop or as a D latch. There are also multiple clock signals, both global and local product term types, configured on a per macrocell basis. Output pin configurations include slew rate limit, bus hold, pull-up, open drain and programmable grounds. A Schmitt-trigger input is available on a per input pin basis. In addition to storing macrocell output states, the macrocell registers may be configured as "direct input" registers to store signals directly from input pins.

Clocking is available on a global or Function Block basis. Three global clocks are available for all Function Blocks as a synchronous clock source. Macrocell registers can be individually configured to power up to the zero or one state. A global set/reset control line is also available to asynchronously set or reset selected registers during operation. Additional local clock, synchronous clock-enable, asynchronous set/reset and output enable signals can be formed using product terms on a per-macrocell or per-Function Block basis.

A DualEDGE flip-flop feature is also available on a per macrocell basis. This feature allows high performance synchronous operation based on lower frequency clocking to help reduce the total power consumption of the device.

Circuitry has also been included to divide one externally supplied global clock (GCK2) by eight different selections. This yields divide by even and odd clock frequencies.

The use of the clock divide (division by 2) and DualEDGE flip-flop gives the resultant CoolCLOCK feature.

DataGATE is a method to selectively disable inputs of the CPLD that are not of interest during certain points in time.

By mapping a signal to the DataGATE function, lower power can be achieved due to reduction in signal switching.

Another feature that eases voltage translation is I/O banking. Two I/O banks are available on the CoolRunner-II 256 macrocell device that permit easy interfacing to 3.3V, 2.5V, 1.8V, and 1.5V devices.

The CoolRunner-II 256 macrocell CPLD is I/O compatible with various I/O standards (see Table 1). This device is also 1.5V I/O compatible with the use of Schmitt-trigger inputs.

RealDigital Design Technology

Xilinx CoolRunner-II CPLDs are fabricated on a 0.18 micron process technology which is derived from leading edge FPGA product development. CoolRunner-II CPLDs employ RealDigital, a design technique that makes use of CMOS technology in both the fabrication and design methodology. RealDigital design technology employs a cascade of CMOS gates to implement sum of products instead of traditional sense amplifier methodology. Due to this technology, Xilinx CoolRunner-II CPLDs achieve both high-performance and low power operation.

Supported I/O Standards

The CoolRunner-II 256 macrocell features LVCMOS, LVTTL, SSTL and HSTL I/O implementations. See Table 1 for I/O standard voltages. The LVTTL I/O standard is a general purpose EIA/JEDEC standard for 3.3V applications that use an LVTTL input buffer and Push-Pull output buffer. The LVCMOS standard is used in 3.3V, 2.5V, 1.8V applications. Both HSTL and SSTL I/O standards make use of a V_{REF} pin for JEDEC compliance. CoolRunner-II CPLDs are also 1.5V I/O compatible with the use of Schmitt-trigger inputs

Table 1: **I/O Standards for XC2C256**(1)

IOSTANDARD Attribute	Output V_{CCIO}	Input V_{CCIO}	Input V_{REF}	Board Termination Voltage V_{TT}
LVTTL	3.3	3.3	N/A	N/A
LVCMOS33	3.3	3.3	N/A	N/A
LVCMOS25	2.5	2.5	N/A	N/A
LVCMOS18	1.8	1.8	N/A	N/A
LVCMOS15 (2)	1.5	1.5	N/A	N/A
HSTL_1	1.5	1.5	0.75	0.75
SSTL2_1	2.5	2.5	1.25	1.25
SSTL3_1	3.3	3.3	1.5	1.5

(1)For information on Vref, see XAPP399.

(2) LVCMOS15 requires Schmitt-trigger inputs.

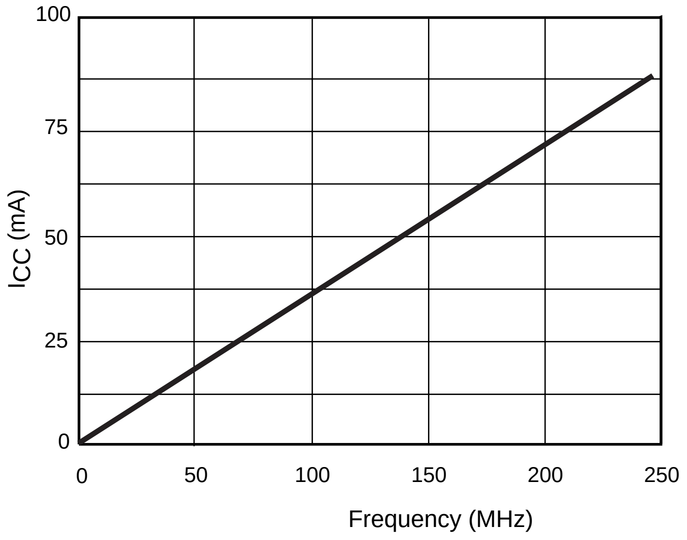


Figure 1: **I_{CC} vs Frequency**

Table 2: **I_{CC} vs Frequency (LVCMOS 1.8V T_A = 25°C)**(1)

	Frequency (MHz)										
	0	30	50	70	100	120	150	170	190	220	240
Typical I_{CC} (mA)	0.021	11.68	19.40	27.01	38.18	45.54	56.32	63.37	70.40	80.90	88.03

Notes:
1. 16-bit up/down, resettable binary counter (one counter per function block).

Absolute Maximum Ratings

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to ground	–0.5 to 2.0	V
V_{CCIO}	Supply voltage for output drivers	–0.5 to 4.0	V
V_{JTAG}(2)	JTAG input voltage limits	–0.5 to 4.0	V
V_{CCAUX}	JTAG input supply voltage	–0.5 to 4.0	V
V_{IN}(1)	Input voltage relative to ground	–0.5 to 4.0	V
V_{TS}(1)	Voltage applied to 3-state output	–0.5 to 4.0	V
T_{STG}(3)	Storage Temperature (ambient)	–65 to +150	°C
T_J	Junction Temperature	+150	°C

Notes:
1. Maximum DC undershoot below GND must be limited to either 0.5V or 10 mA, whichever is easiest to achieve. During transitions, the device pins may undershoot to –2.0v or overshoot to +4.5V, provided this over or undershoot lasts less than 10 ns and with the forcing current being limited to 200 mA.
2. Valid over commercial temperature range.
3. For soldering guidelines and thermal considerations, see the Device Packaging information on the Xilinx website. For Pb free packages, see XAPP427.

Recommended Operating Conditions

Symbol	Parameter		Min	Max	Units
V_{CC}	Supply voltage for internal logic and input buffers	Commercial T_A = 0°C to +70°C	1.7	1.9	V
		Industrial T_A = –40°C to +85°C	1.7	1.9	V
V_{CCIO}	Supply voltage for output drivers @ 3.3V operation		3.0	3.6	V
	Supply voltage for output drivers @ 2.5V operation		2.3	2.7	V
	Supply voltage for output drivers @ 1.8V operation		1.7	1.9	V
	Supply voltage for output drivers @ 1.5V operation		1.4	1.6	V
V_{CCAUX}	JTAG programming		1.7	3.6	V

DC Electrical Characteristics (Over Recommended Operating Conditions)

Symbol	Parameter	Test Conditions	Typical	Max.	Units
I_{CCSB}	Standby current Commercial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	33	150	µA
I_{CCSB}	Standby current Industrial	V_{CC} = 1.9V, V_{CCIO} = 3.6V	54	300	µA
I_{CC}	Dynamic current	f = 1 MHz	-	410	µA
		f = 50 MHz	-	27	mA
C_{JTAG}	JTAG input capacitance	f = 1 MHz	-	10	pF
C_{CLK}	Global clock input capacitance	f = 1 MHz	-	12	pF
C_{IO}	I/O capacitance	f = 1 MHz	-	10	pF
I_{IL}(2)	Input leakage current	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/–1	µA
I_{IH}(2)	I/O High-Z leakage	V_{IN} = 0V or V_{CCIO} to 3.9V	-	+/–1	µA

Notes:
1. 16-bit up/down, resettable binary counter (one counter per function block) tested at V_{CC}= V_{CCIO} = 1.9V
2. See Quality and Reliability section of the CoolRunner-II family data sheet

LVCMOS 3.3V and LVTTL 3.3V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	3.0	3.6	V
V_{IH}	High level input voltage	-	2	3.9	V
V_{IL}	Low level input voltage	-	–0.3	0.8	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 3V	-	0.2	V

LVCMOS 2.5V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	2.3	2.7	V
V_{IH}	High level input voltage	-	1.7	3.9	V
V_{IL}	Low level input voltage	-	–0.3	0.7	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.4V	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.2V	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 2.3V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 2.3V	-	0.2	V

LVCMOS 1.8V DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	1.7	1.9	V
V_{IH}	High level input voltage	-	0.65 x V_{CCIO}	3.9	V
V_{IL}	Low level input voltage	-	–0.3	0.35 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.7V	V_{CCIO} – 0.2	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.7V	-	0.45	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.7V	-	0.2	V

LVCMOS 1.5V DC Voltage Specifications(1)

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	1.4	1.6	V
V_{T+}	Input hysteresis threshold voltage	-	0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}		-	0.2 x V_{CCIO}	0.5 x V_{CCIO}	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.45	-	V
		I_{OH} = –0.1 mA, V_{CCIO} = 1.4V	V_{CCIO} – 0.2	-	V

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.4V	-	0.4	V
		I_{OL} = 0.1 mA, V_{CCIO} = 1.4V	-	0.2	V

Notes:
1. Hysteresis used on 1.5V inputs.

Schmitt Trigger Input DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Max.	Units
V_{CCIO}	Input source voltage	-	1.4	3.9	V
V_{T+}	Input hysteresis threshold voltage	-	0.5 x V_{CCIO}	0.8 x V_{CCIO}	V
V_{T-}		-	0.2 x V_{CCIO}	0.5 x V_{CCIO}	V

SSTL2-1 DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Typ	Max.	Units
V_{CCIO}	Input source voltage	-	2.3	2.5	2.7	V
V_{REF}(1)	Input reference voltage	-	1.15	1.25	1.35	V
V_{TT}(2)	Termination voltage	-	V_{REF} – 0.04	1.25	V_{REF} + 0.04	V
V_{IH}	High level input voltage	-	V_{REF} + 0.18	-	3.9	V
V_{IL}	Low level input voltage	-	–0.3	-	V_{REF} – 0.18	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 2.3V	V_{CCIO} – 0.62	-	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 2.3V	-	-	0.54	V

Notes:
1. V_{REF} should track the variations in V_{CCIO}, also peak to peak AC noise on V_{REF} may not exceed ± 2% V_{REF}
2. V_{TT} of transmitting device must track V_{REF} of receiving devices

SSTL3-1 DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Typ	Max.	Units
V_{CCIO}	Input source voltage	-	3.0	3.3	3.6	V
V_{REF}(1)	Input reference voltage	-	1.3	1.5	1.7	V
V_{TT}(2)	Termination voltage	-	$V_{REF} – 0.05$	1.5	$V_{REF} + 0.05$	V
V_{IH}	High level input voltage	-	$V_{REF} + 0.2$	-	$V_{CCIO} + 0.3$	V
V_{IL}	Low level input voltage	-	–0.3	-	$V_{REF} – 0.2$	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 3V	$V_{CCIO} – 1.1$	-	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 3V	-	-	0.7	V

Notes:
1. V_{REF} should track the variations in V_{CCIO}, also peak to peak AC noise on V_{REF} may not exceed ± 2% V_{REF}
2. V_{TT} of transmitting device must track V_{REF} of receiving devices

HSTL1 DC Voltage Specifications

Symbol	Parameter	Test Conditions	Min.	Typ	Max.	Units
V_{CCIO}	Input source voltage	-	1.4	1.5	1.6	V
V_{REF}(1)	Input reference voltage	-	0.68	0.75	0.90	V
V_{TT}(2)	Termination voltage	-	-	V_{CCIO} x 0.5	-	V
V_{IH}	High level input voltage	-	$V_{REF} + 0.1$	-	1.9	V
V_{IL}	Low level input voltage	-	–0.3	-	$V_{REF} – 0.1$	V
V_{OH}	High level output voltage	I_{OH} = –8 mA, V_{CCIO} = 1.7V	$V_{CCIO} – 0.4$	-	-	V
V_{OL}	Low level output voltage	I_{OL} = 8 mA, V_{CCIO} = 1.7V	-	-	0.4	V

Notes:
1. V_{REF} should track the variations in V_{CCIO}, also peak-to-peak AC noise on V_{REF} may not exceed ± 2% V_{REF}
2. V_{TT} of transmitting device must track V_{REF} of receiving devices

AC Electrical Characteristics Over Recommended Operating Conditions

Symbol	Parameter	-6 Min.	-6 Max.	-7 Min.	-7 Max.	Units
T_{PD1}	Propagation delay single p-term	-	5.7	-	6.7	ns
T_{PD2}	Propagation delay OR array	-	6.0	-	7.5	ns
T_{SUD}	Direct input register clock setup time	2.6	-	3.0	-	ns
T_{SU1}	Setup time (single p-term)	2.4	-	2.8	-	ns
T_{SU2}	Setup time (OR array)	2.7	-	3.3	-	ns
T_{HD}	Direct input register hold time	0	-	0	-	ns
T_H	P-term hold time	0	-	0	-	ns
T_{CO}	Clock to output	-	4.5	-	6.0	ns
F_{TOGGLE}(1)	Internal toggle rate	-	500	-	300	MHz
$F_{SYSTEM1}$(2)	Maximum system frequency	-	256	-	152	MHz
$F_{SYSTEM2}$(2)	Maximum system frequency	-	238	-	141	MHz
F_{EXT1}(3)	Maximum external frequency	-	145	-	114	MHz
F_{EXT2}(3)	Maximum external frequency	-	139	-	108	MHz
T_{PSUD}	Direct input register p-term clock setup time	0.9	-	1.7	-	ns
T_{PSU1}	P-term clock setup time (single p-term)	0.7	-	1.5	-	ns
T_{PSU2}	P-term clock setup time (OR array)	1.0	-	2.0	-	ns
T_{PHD}	Direct input register p-term clock hold time	0.9	-	1.2	-	ns
T_{PH}	P-term clock hold	0.7	-	1.0	-	ns
T_{PCO}	P-term clock to output	-	6.2	-	7.3	ns
T_{OE}/T_{OD}	Global OE to output enable/disable	-	5.6	-	7.0	ns
T_{POE}/T_{POD}	P-term OE to output enable/disable	-	7.0	-	8.0	ns
T_{MOE}/T_{MOD}	Macrocell driven OE to output enable/disable	-	7.4	-	9.9	ns
T_{PAO}	P-term set/reset to output valid	-	7.0	-	8.1	ns
T_{AO}	Global set/reset to output valid	-	5.5	-	7.6	ns
T_{SUEC}	Register clock enable setup time	2.5	-	3.1	-	ns
T_{HEC}	Register clock enable hold time	0	-	0	-	ns
T_{CW}	Global clock pulse width High or Low	1.4	-	2.2	-	ns
T_{PCW}	P-term pulse width High or Low	6.0	-	7.5	-	ns
T_{APRPW}	Asynchronous preset/reset pulse width (High or Low)	6.0	-	7.5	-	ns
T_{DGSU}	Set-up before DataGATE latch assertion	0	-	0	-	ns
T_{DGH}	Hold to DataGATE latch assertion	4.0	-	6.0	-	ns
T_{DGR}	DataGATE recovery to new data	-	8.2	-	9.0	ns
T_{DGW}	DataGATE low pulse width	2.5	-	3.5	-	ns
T_{CDRSU}	CDRST setup time before falling edge GCLK2	1.3	-	2.0	-	ns

Symbol	Parameter	-6		-7		Units
		Min.	Max.	Min.	Max.	
T_{CDRH}	Hold time CDRST after falling edge GCLK2	0	-	0	-	ns
T_{CONFIG}(4)	Configuration time	150	-	150	-	µs

Notes:
1. F_{TOGGLE} is the maximum clock frequency to which a T-Flip Flop can reliably toggle (see the CoolRunner-II family data sheet for more information).
2. $F_{SYSTEM1}$ (1/T_{CYCLE}) is the internal operating frequency for a device fully populated with one 16-bit counter through one p-term per macrocell while $F_{SYSTEM2}$ is through the OR array.
3. F_{EXT1} (1/T_{SU1}+T_{CO}) is the maximum external frequency using one p-term while F_{EXT2} is through the OR array.
4. Typical configuration current during T_{CONFIG} is approximately 7.7 mA.

Internal Timing Parameters

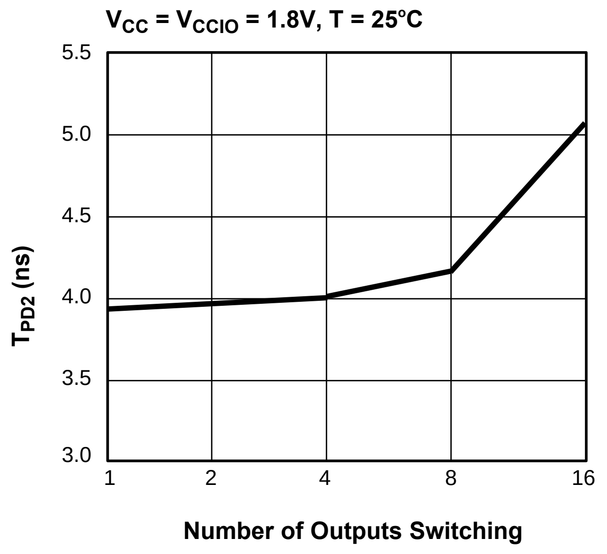
Symbol	Parameter[(2)]	-6		-7		Units
		Min.	Max.	Min.	Max.	
Buffer Delays						
T_{IN}	Input buffer delay	-	2.4	-	2.6	ns
T_{DIN}	Direct data register input delay	-	3.1	-	3.9	ns
T_{GCK}	Global Clock buffer delay	-	1.8	-	2.7	ns
T_{GSR}	Global set/reset buffer delay	-	2.0	-	3.5	ns
T_{GTS}	Global 3-state buffer delay	-	2.1	-	3.0	ns
T_{OUT}	Output buffer delay	-	2.3	-	2.6	ns
T_{EN}	Output buffer enable/disable delay	-	3.5	-	4.0	ns
P-term Delays						
T_{CT}	Control term delay	-	1.1	-	1.4	ns
T_{LOGI1}	Single P-term delay adder	-	0.5	-	1.1	ns
T_{LOGI2}	Multiple P-term delay adder	-	0.3	-	0.5	ns
Macrocell Delay						
T_{PDI}	Input to output valid	-	0.5	-	0.7	ns
T_{SUI}	Setup before clock	1.3	-	1.8	-	ns
T_{HI}	Hold after clock	0	-	0	-	ns
T_{ECSU}	Enable clock setup time	0.8	-	1.8	-	ns
T_{ECHO}	Enable clock hold time	0	-	0	-	ns
T_{COI}	Clock to output valid	-	0.4	-	0.7	ns
T_{AOI}	Set/reset to output valid	-	1.2	-	1.5	ns
T_{CDBL}	Clock doubler delay	-	0	-	0	ns
Feedback Delays						
T_{F}	Feedback delay	-	1.7	-	3.0	ns
T_{OEM}	Macrocell to global OE delay	-	1.7	-	2.5	ns
I/O Standard Time Adder Delays 1.5V CMOS						
T_{IN15}	Standard input adder	-	0.8	-	1.0	ns
T_{HYS15}	Hysteresis input adder	-	3.0	-	4.0	ns
T_{OUT15}	Output adder	-	0.8	-	1.0	ns
T_{SLEW15}	Output slew rate adder	-	4.0	-	5.0	ns
I/O Standard Time Adder Delays 1.8V CMOS						
T_{HYS18}	Hysteresis input adder	-	2.0	-	3.0	ns
T_{OUT18}	Output adder	-	0	-	0	ns
T_{SLEW}	Output slew rate adder	-	2.0	-	4.0	ns

Internal Timing Parameters *(Continued)*

Symbol	Parameter[2]	-6 Min.	-6 Max.	-7 Min.	-7 Max.	Units
I/O Standard Time Adder Delays 2.5V CMOS						
T_{IN25}	Standard input adder	-	0.6	-	0.7	ns
T_{HYS25}	Hysteresis input adder	-	1.5	-	3.0	ns
T_{OUT25}	Output adder	-	0.8	-	1.0	ns
T_{SLEW25}	Output slew rate adder	-	3.0	-	4.0	ns
I/O Standard Time Adder Delays 3.3V CMOS/TTL						
T_{IN33}	Standard input adder	-	0.5	-	0.7	ns
T_{HYS33}	Hysteresis input adder	-	1.2	-	3.0	ns
T_{OUT33}	Output adder	-	1.2	-	1.6	ns
T_{SLEW33}	Output slew rate adder	-	3.0	-	4.0	ns
I/O Standard Time Adder Delays HSTL, SSTL						
SSTL2-1	Input adder to T_{IN}, T_{DIN}, T_{GCK}, T_{GSR}, T_{GTS}	-	0.4	-	1.0	ns
	Output adder to T_{OUT}	-	-0.5	-	0.0	ns
SSTL3-1	Input adder to T_{IN}, T_{DIN}, T_{GCK}, T_{GSR}, T_{GTS}	-	0.4	-	1.0	ns
	Output adder to T_{OUT}	-	-0.5	-	0.0	ns
HSTL-1	Input adder to T_{IN}, T_{DIN}, T_{GCK}, T_{GSR}, T_{GTS}	-	0.6	-	1.0	ns
	Output adder to T_{OUT}	-	0	-	0	ns

Notes:
1. 1.5 ns input pin signal rise/fall.

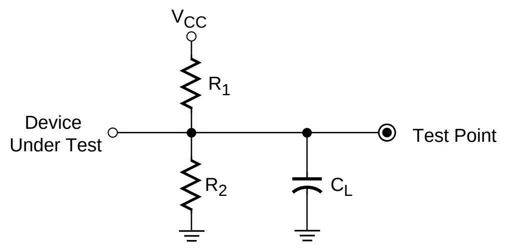
Switching Characteristics



DS092_02_092302

Figure 2: **Derating Curve for T_{PD}**

AC Test Circuit



Output Type	**R_1**	**R_2**	**C_L**
LVTTL33	268Ω	235Ω	35 pF
LVCMOS33	275Ω	275Ω	35 pF
LVCMOS25	188Ω	188Ω	35pF
LVCMOS18	112.5Ω	112.5Ω	35pF
LVCMOS15	150Ω	150Ω	35pF

C_L includes test fixtures and probe capacitance.
1.5 nsec maximum rise/fall times on inputs.

DS_ACT_08_14_02

Figure 3: **AC Load Circuit**

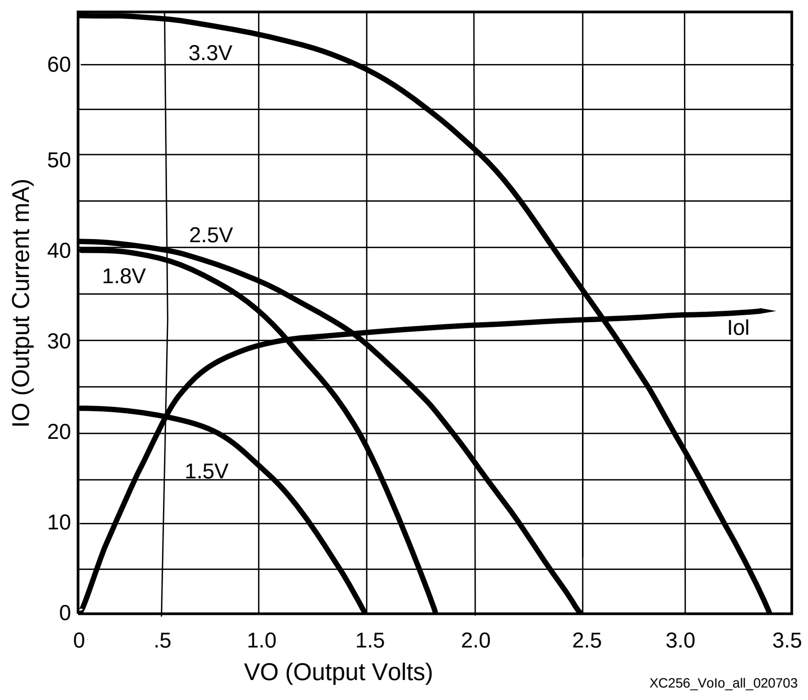


Figure 4: **Typical I/V Curve for XC2C256**

Pin Descriptions

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
1	1	-	-	-	2	B3	2
1	2	-	-	-	208	B4	2
1(GSR)	3	99	A3	143	206	C4	2
1	4	-	-	142	205	A2	2
1	5	-	-	-	203	A3	2
1	6	97	B4	140	202	A4	2
1	7	-	-	-	-	-	-
1	8	-	-	-	-	-	-
1	9	-	-	-	-	-	-
1	10	-	-	-	-	-	-
1	11	-	-	-	-	-	-
1	12	96	-	139	201	B5	2
1	13	95	-	138	200	A5	2
1	14	94	A4	137	199	E8	2
1	15	-	-	-	198	B6	2
1	16	-	C5	-	197	C7	2
2(GTS2)	1	1	A1	2	3	D3	2
2	2	-	-	-	4	C3	2
2(GTS3)	3	2	B2	3	5	E3	2
2	4	-	B1	4	6	B2	2
2(GTS0)	5	3	C3	5	7	D4	2
2	6	-	-	-	8	D2	2
2	7	-	-	-	-	-	-
2	8	-	-	-	-	-	-
2	9	-	-	-	-	-	-
2	10	-	-	-	-	-	-
2	11	-	-	-	-	-	-
2(GTS1)	12	4	C2	6	9	E5	2
2	13	-	C1	7	10	B1	2
2	14	6	D2	9	12	E4	2
2	15	7	-	10	14	C1	2
2	16	-	D1	-	-	E2	2

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
3	1	-	-	136	196	A6	2
3	2	-	B5	135	195	D7	2
3	3	-	-	134	194	B7	2
3	4	-	A5	-	193	E9	2
3	5	93	-	133	192	A7	2
3	6		C6		191	D8	2
3	7	-	-	-	-	-	-
3	8	-	-	-	-	-	-
3	9	-	-	-	-	-	-
3	10	-	-	-	-	-	-
3	11	-	-	-	-	-	-
3	12	92	-	-	189	B8	2
3	13	-	B6	-	188	C8	2
3	14	91	A6	132	187	A8	2
3	15	-	C7	-	186	E11	2
3	16	90	B7	131	185	E10	2
4	1	8	E3	11	15	F2	2
4	2	9	-	12	16	F3	2
4	3	10	E2	13	17	G4	2
4	4	-	E1	14	18	G3	2
4	5	11	F3	15	19	F5	2
4	6	12	F2	16	20	G5	2
4	7	-	-	-	-	-	-
4	8	-	-	-	-	-	-
4	9	-	-	-	-	-	-
4	10	-	-	-	-	-	-
4	11	-	-	-	-	-	-
4	12	-	F1	17	21	H2	2
4	13	13	G1	-	22	H4	2
4	14	-	-	18	23	H3	2
4	15	-	-	-	-	H1	2
4	16	-	-	-	25	H5	2

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
5	1	-	L3	-	49	R1	1
5	2	-	-	33	48	N4	1
5	3	-	-	-	47	N2	1
5(GCK1)	4	23	L2	32	46	M3	1
5	5		L1	31	45	P1	1
5(GCK0)	6	22	K3	30	44	M2	1
5	7	-	-	-	-	-	-
5	8	-	-	-	-	-	-
5	9	-	-	-	-	-	-
5	10	-	-	-	-	-	-
5	11	-	-	-	-	-	-
5	12	-	-	-	43	L3	1
5	13	-	-	-	41	N1	1
5	14	-	-	28	40	L4	1
5	15	-	-	-	39	M1	1
5	16	-	K1	-	38	L5	1
6	1	-	M1	34	50	N3	1
6 (CDRST)	2	24	M2	35	51	P2	1
6	3	-	-	-	54	P4	1
6(GCK2)	4	27	N2	38	55	P5	1
6	5	-	-	-	56	R2	1
6	6	-	-	-	57	T1	1
6	7	-	-	-	-	-	-
6	8	-	-	-	-	-	-
6	9	-	-	-	-	-	-
6	10	-	-	-	-	-	-
6	11	-	-	-	-	-	-
6(DGE)	12	28	P2	39	58	T2	1
6	13	-	M3	40	60	N5	1
6	14	29	N3	41	61	R4	1
6	15	-	P3	42	62	M5	1
6	16	30	M4	43	63	R5	1

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
7	1	-	-	-	37	K4	1
7	2	-	-	-	36	L2	1
7	3	-	-	-	35	K3	1
7	4	-	-	-	34	L1	1
7	5	19	J2	26	32	K5	1
7	6	18	J1	25	31	K2	1
7	7	-	-	-	-	-	-
7	8	-	-	-	-	-	-
7	9	-	-	-	-	-	-
7	10	-	-	-	-	-	-
7	11	17	H3	24	30	J4	1
7	12	16	H2	23	29	K1	1
7	13	15	H1	22	28	J3	1
7	14	14	G3	21	27	J2	1
7	15	-	G2	20	-	J5	1
7	16	-	-	19	-	J1	1
8	1	-	N4	44	64	R6	1
8	2	-	-	45	65	N6	1
8	3	-	-	46	66	R3	1
8	4	-	-	-	67	M6	1
8	5	-	-	48	69	T3	1
8	6	32	-	49	70	P6	1
8	7	-	-	-	-	-	-
8	8	-	-	-	-	-	-
8	9	-	-	-	-	-	-
8	10	-	-	-	-	-	-
8	11	33	M5	50	71	T4	1
8	12	34	N5	51	72	P7	1
8	13	35	P5	52	73	T5	1
8	14	36	M6	-	74	N7	1
8	15	37	N6	-	75	R7	1
8	16	-	-	-	76	M7	1

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
9	1	78	C12	112	160	B13	2
9	2	79	B12	113	161	B14	2
9	3	-	-	-	162	C13	2
9	4	80	A12	114	163	A15	2
9	5				164	C12	2
9	6	81	C11	115	165	B12	2
9	7	-	-	-	-	-	-
9	8	-	-	-	-	-	-
9	9	-	-	-	-	-	-
9	10	-	-	-	-	-	-
9	11	-	-	-	166	D13	2
9	12	82	B11	116	167	A14	2
9	13	-	-	117	168	E13	2
9	14	-	A11	118	169	A13	2
9	15	-	-	119	170	C11	2
9	16	-	C10	-	171	A12	2
10	1	77	A13	111	159	A16	2
10	2	76	B13	110	158	B15	2
10	3	74	C13	107	155	C14	2
10	4	73	C14	106	154	G11	2
10	5	72	D12	105	153	B16	2
10	6	71	D13	104	152	D15	2
10	7	-	-	-	-	-	-
10	8	-	-	-	-	-	-
10	9	-	-	-	-	-	-
10	10	-	-	-	-	-	-
10	11				151	E14	2
10	12	70	D14	103	150	C16	2
10	13	-	-	-	149	F14	2
10	14	-	E12	102	148	F13	2
10	15	-	-	-	147	E15	2
10	16	-	E13	101	146	G13	2

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
11	1	-	B10	-	-	B11	2
11	2	-		-	173	D11	2
11	3	-	A10	-	174	A11	2
11	4	-	-	-	175	D10	2
11	5	-	C9	120	-	B10	2
11	6	-	-	121	-	E12	2
11	7	-	-	-	-	-	-
11	8	-	-	-	-	-	-
11	9	-	-	-	-	-	-
11	10	-	-	-	-	-	-
11	11	85	A8	124	178	F12	2
11	12	86	B8	125	179	B9	2
11	13	87	C8	126	180	C9	2
11	14	89	-	128	182	C10	2
11	15	-	-	129	183	A9	2
11	16	-	-	130	184	D9	2
12	1	-	-	-	145	F15	2
12	2	-	-	100	144	G14	2
12	3	-	-	-	143	E16	2
12	4	-	-	-	142	H12	2
12	5	-	F12	-	140	F16	2
12	6	-	F13	-	139	H16	2
12	7	-	-	-	-	-	-
12	8	-	-	-	-	-	-
12	9	-	-	-	-	-	-
12	10	-	-	-	-	-	-
12	11	68	F14	98	138	G15	2
12	12	-	G12	97	137	H13	2
12	13	67	G13	96	136	G16	2
12	14	66	-	95	135	H14	2
12	15	65	-	94	134	H15	2
12	16	-	-	-	-	J12	2

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
13	1	-	N13	75	107	R15	1
13	2	53	N14	76	108	T16	1
13	3	-	M12	77	109	N14	1
13	4	54	-	-	110	R16	1
13	5	-	M13	78	111	N15	1
13	6	55	-	79	112	M15	1
13	7	-	-	-	-	-	-
13	8	-	-	-	-	-	-
13	9	-	-	-	-	-	-
13	10	-	-	-	-	-	-
13	11	-	-	-	-	-	-
13	12	-	M14	80	113	M13	1
13	13	56	-	81	114	P16	1
13	14	-	L12	82	115	N16	1
13	15	-	-	-	116	L14	1
13	16	-	L13	-	117	M14	1
14	1	52	P14	74	106	P15	1
14	2	-	-	71	103	P14	1
14	3	50	P12	70	102	P13	1
14	4	-	M11	69	101	R13	1
14	5	49	N11	-	100	N13	1
14	6	-	P11	68	-	R14	1
14	7	-	-	-	-	-	-
14	8	-	-	-	-	-	-
14	9	-	-	-	-	-	-
14	10	-	-	-	-	-	-
14	11	-	-	-	-	-	-
14	12	-	-	-	99	T15	1
14	13	-	-	66	97	R12	1
14	14	46	P10	64	95	N11	1
14	15	44	-	-	-	M11	1
14	16	-	P9	61	91	N10	1

Pin Descriptions *(Continued)*

Function Block	Macro-cell	VQ100	CP132	TQ144	PQ208	FT256	I/O Bank
15	1	-	-	-	118	L15	1
15	2	-	L14	83	119	L13	1
15	3	-	-	-	120	M12	1
15	4	-	-	-	121	M16	1
15	5	-	-	-	122	K14	1
15	6	-	-	-	123	L16	1
15	7	-	-	-	-	-	-
15	8	-	-	-	-	-	-
15	9	-	-	-	-	-	-
15	10	-	-	-	-	-	-
15	11	58	K13	85	125	K15	1
15	12	59	K14	86	126	L12	1
15	13	60	J12	87	127	K16	1
15	14	61	J13	88	128	J14	1
15	15	63	H13	91	-	J15	1
15	16	64	H12	92	131	J13	1
16	1	-	-	-	90	P10	1
16	2	-	-	-	89	R10	1
16	3	-	M8	-	88	T10	1
16	4	-	-	-	87	R9	1
16	5	43	N8	60	86	N9	1
16	6	42	-	59	85	M8	1
16	7	-	-	-	-	-	-
16	8	-	-	-	-	-	-
16	9	-	-	-	-	-	-
16	10	-	-	-	-	-	-
16	11	41	P8	58	84	T8	1
16	12	40	M7	57	83	P8	1
16	13	39	N7	56	82	R8	1
16	14	-	-	-	80	T7	1
16	15	-	-	54	78	N8	1
16	16	-	P6	53	77	T6	1

Notes:
1. GTS = global output enable, GSR = global reset/set, GCK = global clock, CDRST = clock divide reset, DGE = DataGATE enable.

XC2C256 JTAG, Power/Ground, No Connect Pins and Total User I/O

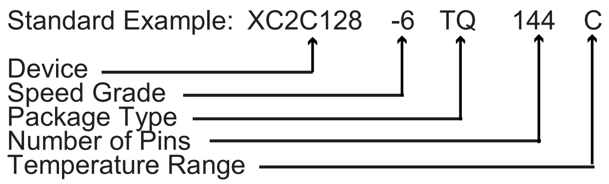
Pin Type	VQ100	CP132	TQ144	PQ208	FT256
TCK	48	M10	67	98	P12
TDI	45	M9	63	94	R11
TDO	83	B9	122	176	A10
TMS	47	N10	65	96	N12
V_{CCAUX} (JTAG supply voltage)	5	D3	8	11	F4
Power internal (V_{CC})	26, 57	P1, K12, A2	1, 37, 84	1, 53, 124	P3, K13, D12, D5
Power Bank 1 I/O (V_{CCIO1})	20, 38, 51	J3, P7, G14, P13	27, 55, 73, 93	33, 59, 79, 92, 105, 132	J6, K6, L7, L8, J11, K11, L10, L9
Power Bank 2 I/O (V_{CCIO2})	88, 98	A14, C4, A7	109, 127, 141	26, 133, 157, 172, 181, 204	F7, F8, G6, H6, F10, F9, H11
Ground	21, 25, 31, 62, 69, 75, 84, 100	K2, N1, P4, N9, N12, J14, H14, E14, B14, A9, B3	29, 36, 47, 62, 72, 89, 90, 99, 108, 123, 144	13, 24, 42, 52, 68, 81, 93, 104, 129, 130, 141, 156, 177, 190, 207	F11, F6, G10, G7, G8, G9, H10, H7, H8, H9, J10, J7, J8, J9, K10, K7, K8, K9, L11, L6
No connects	-	-	-	-	A1, C2, E6, D1, E1, G2, F1, G1, M4, T9, P9, M9, M10, T11, T12, T13, P11, T14, J16, K12, D16, G12, C15, D14, D6, C6, E7, C5
Total user I/O	80	106	118	173	184

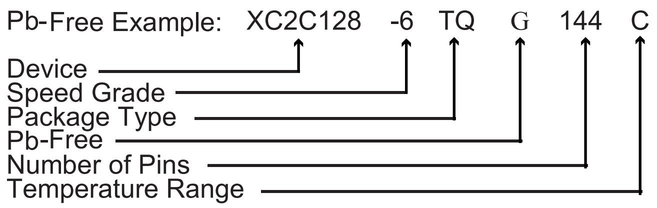
Ordering Information

Part Number	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Commercial (C) Industrial (I)[1]
XC2C256-6VQ100C	0.5mm	43.1	10.9	Very Thin Quad Flat Pack	14mm x 14mm	80	C
XC2C256-7VQ100C	0.5mm	43.1	10.9	Very Thin Quad Flat Pack	14mm x 14mm	80	C
XC2C256-6CP132C	0.5mm	65.0	15.0	Chip Scale Package	8mm x 8mm	106	C
XC2C256-7CP132C	0.5mm	65.0	15.0	Chip Scale Package	8mm x 8mm	106	C
XC2C256-6TQ144C	0.5mm	37.2	7.2	Thin Quad Flat Pack	20mm x 20mm	118	C
XC2C256-7TQ144C	0.5mm	37.2	7.2	Thin Quad Flat Pack	20mm x 20mm	118	C
XC2C256-6PQ208C	0.5mm	36.9	9.7	Plastic Quad Flat Pack	28mm x 28mm	173	C
XC2C256-7PQ208C	0.5mm	36.9	9.7	Plastic Quad Flat Pack	28mm x 28mm	173	C
XC2C256-6FT256C	1.0mm	34.6	6.1	Fine Pitch Thin BGA	17mm x 17mm	184	C
XC2C256-7FT256C	1.0mm	34.6	6.1	Fine Pitch Thin BGA	17mm x 17mm	184	C
XC2C256-6VQG100C	0.5mm	43.1	10.9	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	80	C
XC2C256-7VQG100C	0.5mm	43.1	10.9	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	80	C
XC2C256-6CPG132C	0.5mm	65.0	15.0	Chip Scale Package; Pb-free	8mm x 8mm	106	C
XC2C256-7CPG132C	0.5mm	65.0	15.0	Chip Scale Package; Pb-free	8mm x 8mm	106	C
XC2C256-6TQG144C	0.5mm	37.2	7.2	Thin Quad Flat Pack; Pb-free	20mm x 20mm	118	C
XC2C256-7TQG144C	0.5mm	37.2	7.2	Thin Quad Flat Pack; Pb-free	20mm x 20mm	118	C
XC2C256-6PQG208C	0.5mm	36.9	9.7	Plastic Quad Flat Pack; Pb-free	28mm x 28mm	173	C
XC2C256-7PQG208C	0.5mm	36.9	9.7	Plastic Quad Flat Pack; Pb-free	28mm x 28mm	173	C
XC2C256-6FTG256C	1.0mm	34.6	6.1	Fine Pitch Thin BGA; Pb-free	17mm x 17mm	184	C
XC2C256-7FTG256C	1.0mm	34.6	6.1	Fine Pitch Thin BGA; Pb-free	17mm x 17mm	184	C
XC2C256-7VQ100I	0.5mm	43.1	10.9	Very Thin Quad Flat Pack	14mm x 14mm	80	I
XC2C256-7CP132I	0.5mm	65.0	15.0	Chip Scale Package	8mm x 8mm	106	I
XC2C256-7TQ144I	0.5mm	37.2	7.2	Thin Quad Flat Pack	20mm x 20mm	118	I
XC2C256-7PQ208I	0.5mm	36.9	9.7	Plastic Quad Flat Pack	28mm x 28mm	173	I
XC2C256-7FT256I	1.0mm	34.6	6.1	Fine Pitch Thin BGA	17mm x 17mm	184	I

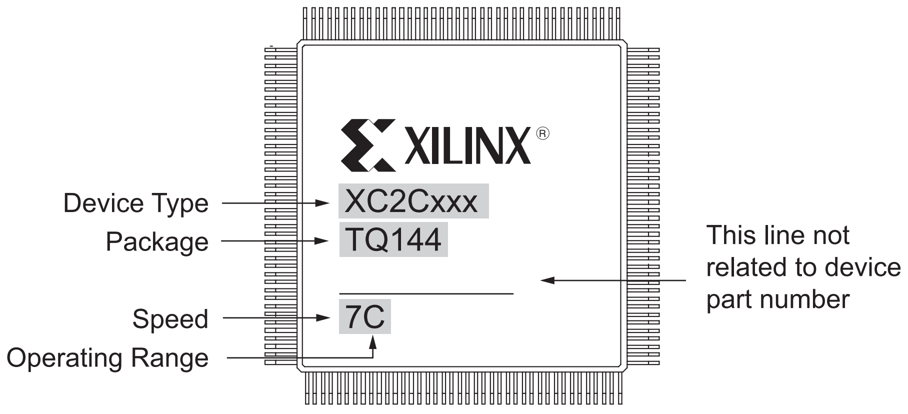
Part Number	Pin/Ball Spacing	θ_{JA} (C/Watt)	θ_{JC} (C/Watt)	Package Type	Package Body Dimensions	I/O	Commercial (C) Industrial (I)[1]
XC2C256-7VQG100I	0.5mm	43.1	10.9	Very Thin Quad Flat Pack; Pb-free	14mm x 14mm	80	I
XC2C256-7CPG132I	0.5mm	65.0	15.0	Chip Scale Package; Pb-free	8mm x 8mm	106	I
XC2C256-7TQG144I	0.5mm	37.2	7.2	Thin Quad Flat Pack; Pb-free	20mm x 20mm	118	I
XC2C256-7PQG208I	0.5mm	36.9	9.7	Plastic Quad Flat Pack; Pb-free	28mm x 28mm	173	I
XC2C256-7FTG256I	1.0mm	34.6	6.1	Fine Pitch Thin BGA; Pb-free	17mm x 17mm	184	I

Notes:
1. C = Commercial (T_A = 0°C to +70°C); I = Industrial (T_A = –40°C to +85°C).





Device Part Marking



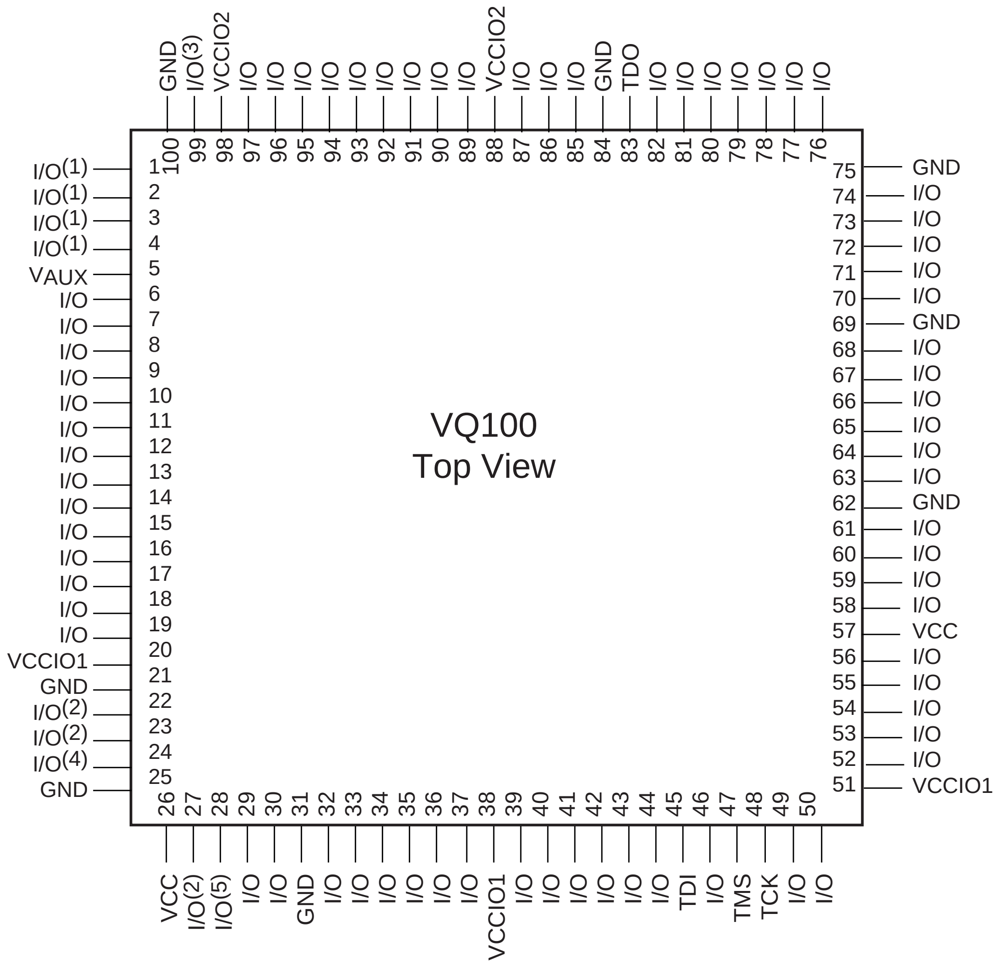
Part marking for non-chip scale package

Figure 5: **Sample Package with Part Marking**

Note: Due to the small size of chip scale packages, the complete ordering part number cannot be included on the package marking. Part marking on chip scale packages by line are:

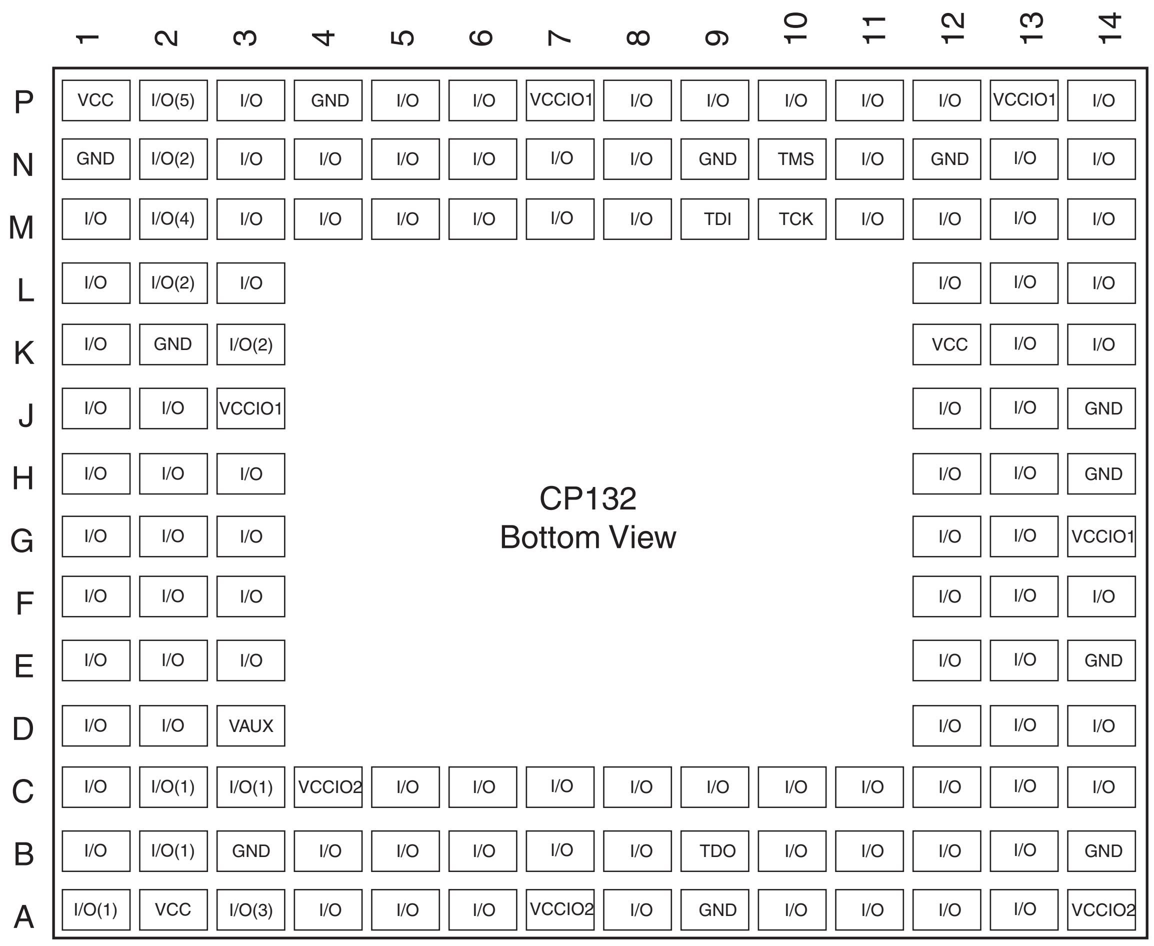
- Line 1 = X (Xilinx logo) then truncated part number
- Line 2 = Not related to device part number
- Line 3 = Not related to device part number

1. Line 4 = Package code, speed, operating temperature, three digits not related to device part number. Package codes: C5 = CP132, C6 = CPG132.



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - Data Gate

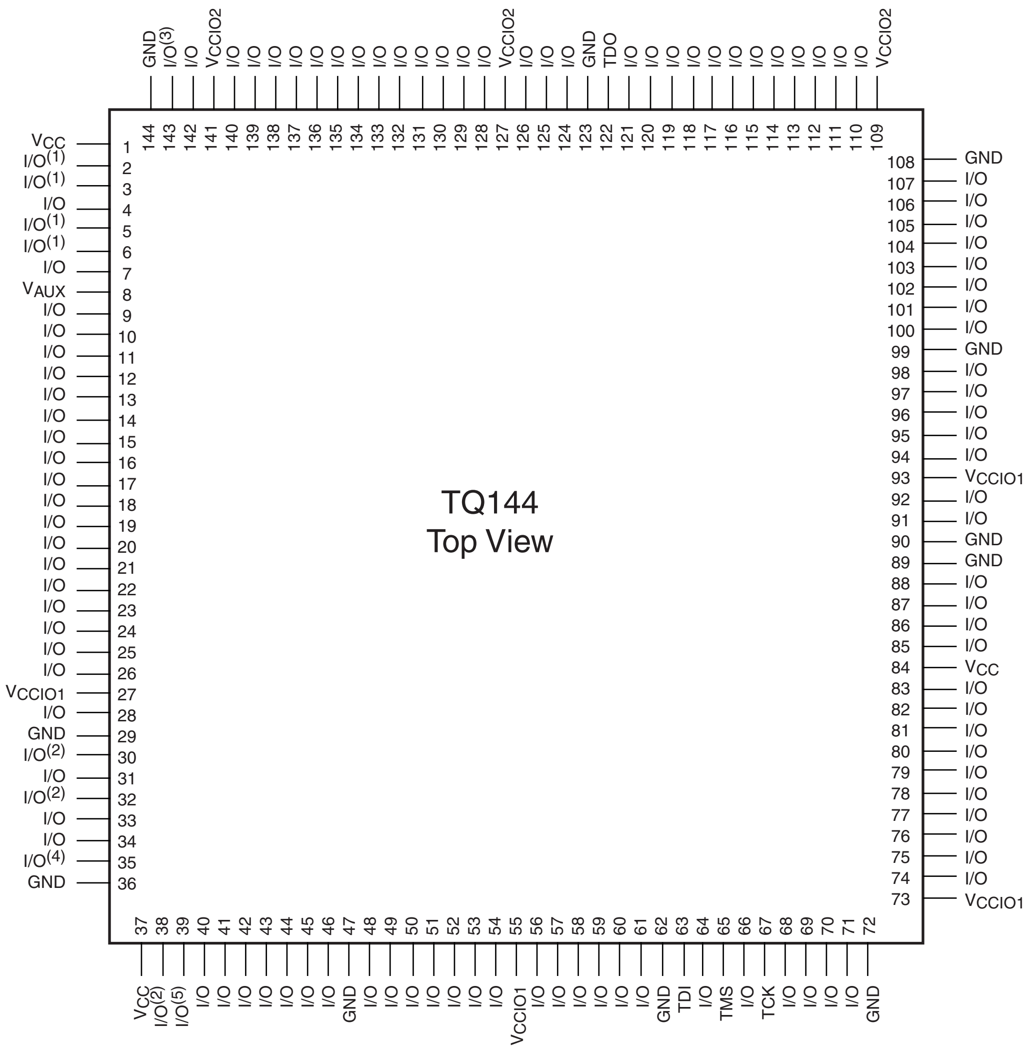
Figure 6: **VQ100 Very Thin Quad Flat Pack**



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
P	VCC	I/O(5)	I/O	GND	I/O	I/O	VCCIO1	I/O	I/O	I/O	I/O	I/O	VCCIO1	I/O
N	GND	I/O(2)	I/O	I/O	I/O	I/O	I/O	I/O	GND	TMS	I/O	GND	I/O	I/O
M	I/O	I/O(4)	I/O	I/O	I/O	I/O	I/O	I/O	TDI	TCK	I/O	I/O	I/O	I/O
L	I/O	I/O(2)	I/O									I/O	I/O	I/O
K	I/O	GND	I/O(2)									VCC	I/O	I/O
J	I/O	I/O	VCCIO1									I/O	I/O	GND
H	I/O	I/O	I/O									I/O	I/O	GND
G	I/O	I/O	I/O									I/O	I/O	VCCIO1
F	I/O	I/O	I/O									I/O	I/O	I/O
E	I/O	I/O	I/O									I/O	I/O	GND
D	I/O	I/O	VAUX									I/O	I/O	I/O
C	I/O	I/O(1)	I/O(1)	VCCIO2	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
B	I/O	I/O(1)	GND	I/O	I/O	I/O	I/O	I/O	TDO	I/O	I/O	I/O	I/O	GND
A	I/O(1)	VCC	I/O(3)	I/O	I/O	I/O	VCCIO2	I/O	GND	I/O	I/O	I/O	I/O	VCCIO2

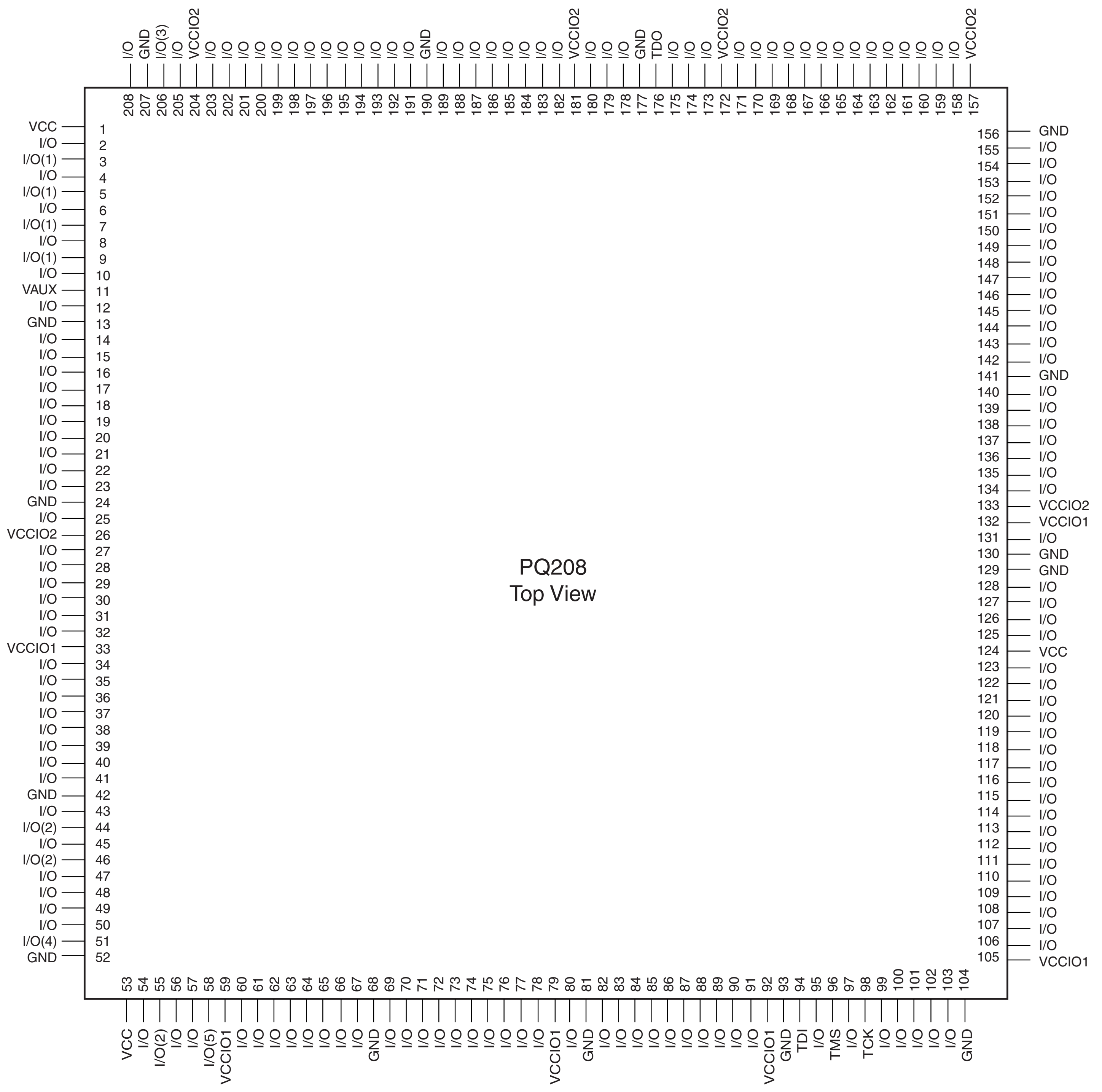
(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - DataGATE Enable

Figure 7: **CP132 Chip Scale Package**



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - DataGATE Enable

Figure 8: **TQ144 Thin Quad Flat Pack**



(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - DataGATE Enable

Figure 9: **PQ208 Quad Flat Package**

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
A	I/O	I/O	I/O	I/O	I/O	I/O	TDO	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	NC
B	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
C	I/O	NC	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	NC	NC	I/O(3)	I/O	NC	I/O
D	NC	I/O	NC	I/O	VCC	I/O	I/O	I/O	I/O	I/O	NC	VCC	I/O(1)	I/O(1)	I/O	NC
E	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	NC	NC	I/O(1)	I/O	I/O(1)	I/O	NC
F	I/O	I/O	I/O	I/O	I/O	GND	VCCIO2	VCCIO2	VCCIO2	VCCIO2	GND	I/O	VAUX	I/O	I/O	NC
G	I/O	I/O	I/O	I/O	NC	I/O	GND	GND	GND	GND	VCCIO2	I/O	I/O	I/O	NC	NC
H	I/O	I/O	I/O	I/O	I/O	VCCIO2	GND	GND	GND	GND	VCCIO2	I/O	I/O	I/O	I/O	I/O
J	NC	I/O	I/O	I/O	I/O	VCCIO1	GND	GND	GND	GND	VCCIO1	I/O	I/O	I/O	I/O	I/O
K	I/O	I/O	I/O	VCC	NC	VCCIO1	GND	GND	GND	GND	VCCIO1	I/O	I/O	I/O	I/O	I/O
L	I/O	I/O	I/O	I/O	I/O	GND	VCCIO1	VCCIO1	VCCIO1	VCCIO1	GND	I/O	I/O	I/O	I/O	I/O
M	I/O	I/O	I/O	I/O	I/O	I/O	NC	NC	I/O	I/O	I/O	I/O	NC	I/O(2)	I/O(2)	I/O
N	I/O	I/O	I/O	I/O	TMS	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
P	I/O	I/O	I/O	I/O	TCK	NC	I/O	NC	I/O	I/O	I/O	I/O(2)	I/O	VCC	I/O(4)	I/O
R	I/O	I/O	I/O	I/O	I/O	TDI	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
T	I/O	I/O	NC	NC	NC	NC	I/O	NC	I/O	I/O	I/O	I/O	I/O	I/O	I/O(5)	I/O

FT256 Bottom View

(1) - Global Output Enable
(2) - Global Clock
(3) - Global Set/Reset
(4) - Clock Divide Reset
(5) - DataGATE Enable

Figure 10: **FT256 Fine Pitch Thin BGA**

Additional Information

CoolRunner-II Data Sheets and Application Notes

Device Packages

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/09/02	1.0	Initial Xilinx release.
05/13/02	1.1	Updated AC Electrical Characteristics and added new parameters.
10/31/02	1.2	Corrected package user I/O, added Voltage Referenced DC tables.
03/17/03	2.0	Added Characterization numbers for product release and device part marking
04/02/03	2.1	Updated T_{SOL} max from 260 to 220. Changed I_{CCSB} units from mA to μA.
01/26/04	2.2	Updated Device Part Marking. Updated links and Tsol.
02/26/04	2.3	Corrected Theta JC value on XC2C256-7TQ144.
08/03/04	2.4	Pb-free documentation
08/19/04	2.5	Changes to I_{CCSB} maximum specifications in DC Electrical Characteristics table, on page 3.
10/01/04	2.6	Add Asynchronous Preset/Reset Pulse Width specification to AC Electrical Characteristics.
03/07/05	2.7	Removed -5 speed grade. Changes to Table 1, I/O Standards.
06/28/05	2.8	Move to Product Specification. Change to T_{IN25}, T_{OUT25}, T_{IN33}, and T_{OUT33} for -7 speed grade.