JPL D-31335

Design of a Hardware/Software Platform for a Comprehensive Dynamic Burn-In Test of SRAM-Based Field Programmable Gate Arrays

February 14, 2005

Ramin Roosta, Jet Propulsion Laboratory

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Acknowledgement Statement

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The author would like to thank Michael Sadigursky for his work on driver board and UUT board design, Tony Wang for the VHDL codes , and Phil Tracton for the SW codes.

Contents

Abstract .... **xi**

1 Introduction **1**

1.1 Project Objectives 1

1.2 Project Motivation 1

2 High-Level Description **3**

2.1 Components 3

- 2.1.1 Driver Board 3
- 2.1.2 UUT Board 4
- 2.1.3 Board Connection Cables 4
- 2.1.4 PC Software 4
- 2.1.5 Power Supply 4

2.2 Design Examples 5

3 Detailed Description of the Driver Board **7**

3.1 Components 7

- 3.1.1 Xilinx Virtex-II 3000 FPGA 7
- 3.1.2 FPGA Mode Switch, Configuration LED, and Temperature Sensor 13
- 3.1.3 XC18V04 EEPROM 14
- 3.1.4 Parallel Port Interface 15
- 3.1.5 Clocks 17

3.2 Interfaces 18

- 3.2.1 Host-PC EPP interface 18
- 3.2.2 UUT FPGA download/readback interface 19
- 3.2.3 Test-vectors sending/retrieving interface 20
- 3.2.4 Internal/external test-vector storage interface 21
- 3.2.5 Internal/external fault-detection interfaces 22

3.3 FPGA Timing 23

- 3.3.1 Data Write Cycle 23
- 3.3.2 Address Write Cycle 23
- 3.3.3 Data Read Cycle 24
- 3.3.4 Address Read Cycle 24

4 Detailed Description of the UUT Board **27**

4.1 Components 27

- 4.1.1 UUT FPGA 27
- 4.1.2 Capacitors 27
- 4.1.3 Sockets and Connectors 28
  - 4.1.3.1 UUT Socket 28
  - 4.1.3.2 Clock Sockets 30
- 4.1.4 Clocks (Oscillators) 30

4.1.5 Interfaces and Clocking Scheme Options 31
4.1.6 FPGA Mode Switch and Temperature Sensor 32
4.2 Thermal Considerations and Analysis 32

5 Detailed Description of the Board Connection Cables 35
5.1 Controlling Ground Bounce 35
5.2 Limiting Ringing and Overshooting of the Signal 36
5.3 Locating Clocks to Minimize Noise 38
5.4 Other Good Design Practices 38

6 Printed Circuit Boards 39

7 Detailed Description of the Power Supply 41
7.1 Power Requirements 41
7.2 Power-up Timing and Power Sequencing Requirements 42
7.3 Power Supply Summary and Specifications 44

8 FPGA Design Examples 45
8.1 Shifter Example 45
8.2 FIFO (Test) Example 46
8.3 CRC (Calculator) Example 47

9 PC Software 49
9.1 WinIO 50
9.2 Parallel Layer 51
9.3 Driver 51
9.4 Log 51
9.5 Command Line Interface 51
9.6 Graphical User Interface 51

10 Test Results 53

11 Future Enhancements 57

Appendix A: Acronyms and Other Abbreviations

Appendix B: Bills of Materials
B.1 Driver Board
B.2 UUT Board

Appendix C: Schematic Drawings of the Driver Board
C.1 Driver Board
C.2 Unused Bank
C.3 Power Plane

Appendix D: Oscillator specifications

Appendix E: Schematic Drawings of the UUT (Burn-In) Board
E.1 UUT Board
E.2 Typical Input Section
E.3 Typical Output Section

Appendix F: PDF Conversion of Layout Gerber Files of the UUT Board

Appendix G: Connectors

Appendix H: BGA Socket

Appendix I: Cables

Appendix J: BGA Footprint

Appendix K: PCB Layering

Appendix L: Schematic Drawing of the Power Supply

Appendix M: Driver FPGA VHDL Source Code

Appendix N: Shifter-Test FPGA VHDL Source Code

Appendix O: FIFO Test FPGA VHDL Source Code

Appendix P: Calculator Test FPGA VHDL Source Code

Appendix Q: Behavioral Test Bench and Results

Appendix R: C/C++ Software Source Code

Figures

Figure 1: System block diagram. 3
Figure 2: Virtex-II 3000 FPGA basic block diagram. 8
Figure 3: BGA I/O bank organization 9
Figure 4: Virtex-II IOB block diagram 10
Figure 5: A block diagram of a Virtex-II FPGA slice 12
Figure 6: Block RAM used as dual-port RAM 13
Figure 7: XC18V04-series PROM block diagram 15
Figure 8: Cascading multiple EEPROMs (with the FPGA in SelectMAP Mode). 15
Figure 9: LVT, LVCMOS or PCI select I/O-ultra standards 16
Figure 10: Slice of 74ALVC164245 level shifter 17
Figure 11: 5-V transceivers on common bus with 3-V transceivers 17
Figure 12: Download/readback timing diagrams. 20
Figure 13: EPP data write cycle. 23
Figure 14: EPP address write cycle. 24
Figure 15: EPP data read cycle. 24
Figure 16: EPP address read cycle. 25
Figure 17: Socket for the UUT FPGA 29
Figure 18: BGA package construction. 30
Figure 19: Series termination. 37
Figure 20: DCI in a Virtex-II bank 38
Figure 21: 1.5-core-voltage ramp 43
Figure 22: 3.3-auxiliary-supply-voltage V_{CCAUX} and output-driver-supply voltage V_{CCO} ramp. 43
Figure 23: Graphical user interface (GUI) for the PC software. 49
Figure 24: Software-design block diagram. 50

Tables

Table 1: Virtex-II 3000 FPGA (XC2V3000) characteristics. 8
Table 2: Single-ended I/O standards supported by the Virtex-II FPGA. 10
Table 3: Virtex-II configuration mode pin settings. Master SelectMAP is the default mode for our configuration. 14
Table 4: EPP signals as defined by IEEE 1284.. 18
Table 5: Bits in the SelectMAP interface.. 19
Table 6: Data transaction protocols for sending results from the driver FPGA to the UUT FPGA. 21
Table 7: Parallel IO register definitions. 25
Table 8: Software routines for the I/O register.. 26
Table 9: Power estimates for the Xilinx Virtex-II FPGA (XC2V3000) used in the UUT board. Calculations assume 25-MHz clock and approximately 85% device utilization. 42
Table 10: Utilization of FPGA resources with the shifter configuration. 45
Table 11: Utilization of FPGA resources with the test configuration. 46
Table 12: Utilization of FPGA resources with the calculator configuration.................. 47

Abstract

Field-programmable gate arrays (FPGAs) play an increasingly important role in military and aerospace applications, where the need for a high-capacity logic—tens of million of gates per integrated circuit, and fast and complex routing—elevates long-term reliability issues. Although Xilinx Virtex-II 3000 static random-access memory (SRAM)-based FPGAs are extensively used in commercial applications, they have been used less frequently in space-flight applications. They are susceptible to single-event upsets (SEUs), and conventional test methods do not adequately test their reliability for these applications. Conventional methods test only a few of the discrete alternating current (AC) parameters (primarily switching) of a given integrated circuit.

This project presents the design and implementation of a dynamic burn-in/radiation-test platform that allows complete, comprehensive AC testing of the Xilinx Virtex-II 3000 FPGA. The platform also allows developers to collect information on the elevation of the junction temperature as a function of gate utilization, operating frequency, and functionality.

Supporting at-speed and SEU tests, the platform may be used to test any possible configuration of the FPGA and any associated performance parameters. A graphical user interface allows designers to fully control the programming of the FPGA and the configurations/functions to be stressed. The platform also offers error logging, user-selectable master clocks, readback, and expandability.

1 Introduction

This report documents a project to design and implement a dynamic burn-in/radiation-test platform that allows complete, comprehensive dynamic testing of the Xilinx Virtex-II 3000 field-programmable gate array (FPGA). Appendix A lists acronyms and other abbreviations used in the report.

1.1 PROJECT OBJECTIVES

The latest trend in the satellite and deep space application markets is to expand on designs with static random-access memory (SRAM)-based FPGAs. FPGAs store millions of configuration bits in internal latches. A common concern with these chips is their sensitivity to single-event upsets (SEU). Xilinx has measured and documented SEU sensitivity for several years, primarily for aerospace and military applications.

The mean time between (functional) failures (MTBF) for a logic error caused by SEU in a Virtex-II 3000 FPGA (XC2V1000) was measured and was reported to be 1,000 years (which is equivalent to 114 failure in time (FITs), where one FIT stands for one error per billion device hours). These are soft errors—that is, reloading the configuration can repair them.

The ability to optimize the functionality of onboard electronics or to change them to accommodate new objectives and to correct design errors remotely, any time after the launch of spacecraft (though very risky), is made possible by utilizing reconfigurable logic. A SRAM-based FPGA will reduce the cost of optimization or change by allowing common hardware to be used for many applications, as a mission progresses. Programmable logic is known for its portability, low cost, availability, and quicker time to market for new products. However, reconfigurable platforms for spacecraft applications must satisfy stringent reliability requirements.

The objective of this project is to design a fully programmable hardware/software platform that will allow the comprehensive, dynamic, burn-in/life test of SRAM-based FPGAs. This platform can also be used for SEU radiation testing.

1.2 PROJECT MOTIVATION

FPGAs have played increasingly important roles in military and aerospace applications. Xilinx SRAM-based FPGAs are extensively used in commercial applications. They have been used less frequently in space flight applications because of their susceptibility to SEUs. The reliability of these devices in space applications is a concern that has not been addressed thoroughly. Conventional methods test very few discrete AC parameters (primarily switching) of given integrated circuits.

The objective of this project is to design a fully programmable hardware/software platform that allows the comprehensive static/dynamic burn-in test for SRAM-based FPGAs, for at-speed tests and SEU tests. This flexible approach will test any possible configuration of the FPGA and any associated performance parameters. It allows complete or partial reprogramming of the FPGA and verification of the program by using read-back followed by dynamic testing. Designers have full control over which functional elements of the FPGA to stress. They can completely simulate all possible types of configurations and functions.

Another benefit of this platform is that it allows collecting information on elevation of the junction temperature as a function of gate utilization, operating frequency, and functionality.

A software tool demonstrates the various features of the system. The software consists of three major parts: the parallel interface driver, the main system procedure, and a graphical user interface (GUI).

2 High-Level Description

This section describes the components of the prototype and the algorithms it uses for the three design examples provided.

2.1 Components

The prototype is composed of five major components, as shown in Figure 1:

- Driver board
- Unit-under-test (UUT) board
- Board connection cables
- Host personal computer (PC) software
- Power supply

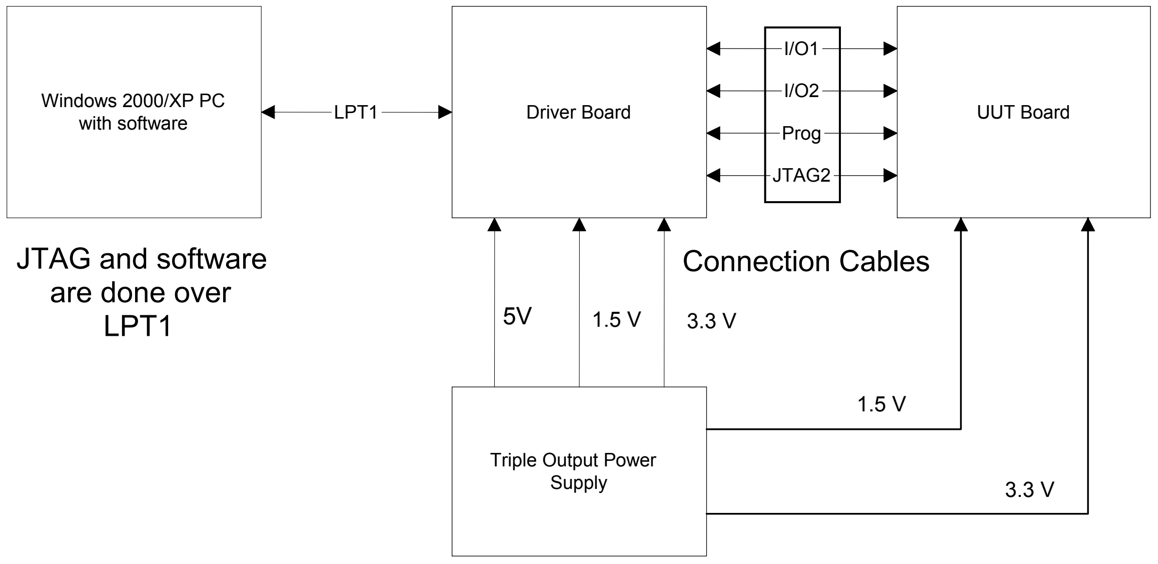

Windows 2000/XP PC with software
LPT1
Driver Board
I/O1
I/O2
Prog
JTAG2
UUT Board
JTAG and software are done over LPT1
Connection Cables
5V
1.5 V
3.3 V
1.5 V
3.3 V
Triple Output Power Supply

Figure 1: System block diagram.

2.1.1 Driver Board

The driver board configures the UUT FPGA on the UUT board, manages the traffic of the test vectors, collects test results, and passes them to the host PC. This board is connected to both the host PC via parallel ports and the UUT board via the board connection cables. Because it is not radiation-hardened (RH), this board is not placed inside the radiation/temperature oven.

The "brain" of the driver board is the driver FPGA, which is a commercial Xilinx Virtext-II 3000 FPGA (XC2V3000) on ball-grid array (BGA)728 package that is 33 x 3 3 mm^2 with 1.27-mm pitch. This FPGA is responsible for communications with the PC, programming the UUT FPGA, applying the test vectors to the UUT FPGA, collecting the results from the UUT FPGA, and detecting errors. The driver-board FPGA can be programmed through Joint Test Action Group (JTAG) interface or the programmable read-only memory (PROM) on the board.

2.1.2 UUT Board

The UUT ("burn-in") board contains the UUT FPGA—the FPGA image to be tested. The driver FPGA programs and applies the test vectors to the UUT FPGA. This board is connected only to the driver board. For prototyping, the same FPGA is used for testing as is used for the driver FPGA—the commercial version of the Virtex-II 3000 FPGA (P/N XC2V3000)—since radiation-hardened (RH) versions are not yet available. The industrial-grade version (P/N XC2V3000-4BG728I) is being used until the RH versions are released: the XQR2V3000-4CG717M and the XQR2V3000-4CG717V. The latter (ceramic) version is considered more suited for space applications.

A less expensive commercial version of this FPGA (XC2V2000) is packaged in BGA575 and is 100% pin compatible with the "Mil-Temp" version (XQR2V3000). It can be used for preparation/set-up trials.

2.1.3 Board Connection Cables

Specially designed cable assembly connect the two boards together. Because part of the cable will go in the oven with the UUT board, the cables are rated to operate at temperature range from -55 °C to 200C°. The operating range for the cable is 2 GHz with 103 dB per 100 ft loss. Cable complies with MIL-C-17.

2.1.4 PC Software

To control the whole system, a PC running special software communicates with the driver FPGA. The GUI was built using Microsoft's .NET technology and a third-party library to gain access to the hardware. The PC must use a Windows 2000 or Windows XP operating system and have an enhanced parallel port (EPP). The user running the software must use an administrator account.

2.1.5 Power Supply

A custom power supply was built to power both boards.

2.2 DESIGN EXAMPLES

The prototype includes three sample test algorithms ("design examples"), each of which configures the UUT FPGA differently:

- **Shifter example:** turns the slice flip-flops on the UUT FPGA into shift registers
- **FIFO example:** The Xilinx CoreGenerator is used to produce three large first-in-first-out (FIFO) buffers that consist of all the block RAM and one basic element of distributed RAM FIFO.
- **Cyclic-redundancy-check** (**CRC) example:** The look-up tables (LUTs) of the UUT FPGA are configured into four input-function generators

3 Detailed Description of the Driver Board

The bill of materials of the driver board is given in Appendix B. Complete schematics are given in Appendix C.

3.1 COMPONENTS

Essential components of the driver board are those essential to test operations. All of the following are essential components except the FPGA pin-setting switch, configuration light emitting diode (LED), and temperature sensor.

The driver board is constructed of eight layers of polyimide with an uninterrupted ground plane, one plane for core-supply V_{CCINT} (1.5 V), plus one plane for V_{CCAUX} (3.3 V). V_{CCO} (3.3 V) is distributed on wide signal traces with sufficient bypass capacitors.

3.1.1 Xilinx Virtex-II 3000 FPGA

The Virtex-II 3000 FGPA (XC2V1000) on the driver board is the brain of the entire system. It is responsible for communications with the user through the PC's parallel port. It also stores the user-defined test vectors and sends them to the UUT FPGA, upon the user's command. It provides the download/readback protocol signals and data stream necessary to configure and debug the UUT FPGA. It also provides the built-in-self-test logic to decide whether the incoming results from the UUT FPGA contain any errors.

Several projects at JPL/NASA are planning to use the Virtex-II 3000 FPGA, such as Ocean Surface Topography from Space Missions (OSTM), Electra, and Mars Telecommunications Orbiter (MTO). This FPGA was selected because of its advanced resources. Its architecture is designed for rapid deployment on advanced-process technologies below 100 nanometers (nm). The eight-layer aluminum process used by this FPGA offers the best performance and lowest power consumption available.

Figure 2 is an overall block diagram of the FPGA. The main characteristics of this device are shown in Table 1. Because, unlike the UUT FPGA, there is no need to replace the driver FPGA, the driver FPGA is soldered to the board, using a reflow method recommended by Xilinx.

The FPGA is built on a 0.15-micron, eight-layer aluminum process with high-speed, 0.12-micron transistors.

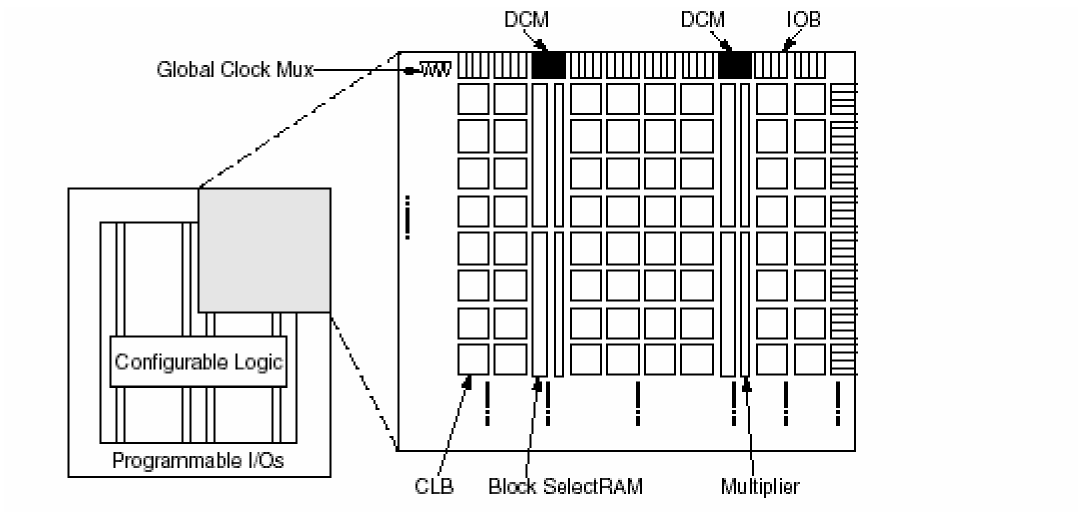


Figure 2: Virtex-II 3000 FPGA basic block diagram.

Table 1: Virtex-II 3000 FPGA (XC2V3000) characteristics.

Characteristic	Value
Maximum system gates	3 M
Logic cells	32,256
Number of slices	14,336
Number of configurable logic blocks (CLBs)	3,584
Embedded block RAM	1728 kbit
Maximum distributed RAM	448 kbit
Digital clock managers	12
18 x 18 multipliers	96
Maximum user inputs/outputs (I/Os)	516
Digital control management (DCM) buffers	12
Global-clock multiplexer buffers	16
Supported I/O standards	27
Maximum I/O pads	720

The FPGA require a 1.5-V direct-current (DC) internal (core) supply voltage. In addition, the device requires an auxiliary supply voltage, V_{CCAUX}, and an output driver supply voltage, V_{CCO}. The values for the last two depend on the given (bank) I/O standard. V_{CCO} and V_{CCAUX} were chosen to provide 3.3 V DC and to be driven by a common power supply.

The FPGA offers unique pinout/package migration paths, which maintains compatibility with printed-circuit-board (PCB) footprints across different device densities and packages. All the pin types are similar regardless of the device/ package combination. The number of control pins is always 16, including V_{BATT}.

The number of power/ground pins and user I/O pins, however, depends on each device/package combination.

The FPGA is divided into eight I/O banks. One of the main benefits of banking is the simplification of compliance to requirements for simultaneously switching (input/)output (SSO) blocks by correctly distributing active input/output buffers (IOBs) between the banks. Banking was taken into consideration when the configuration- and test-vector software was designed. The package organization is shown in Figure 3.

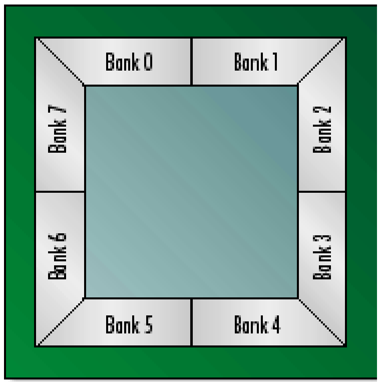


Figure 3: BGA I/O bank organization

A useful feature of the Virtex-II FPGA is its digital control impedance (DCI). It adds on-chip termination capability to the IOB, eliminating the need for external resistor and thus conserving valuable board space. As shown in Figure 4, a useful feature of the IOB is the built-in double-data-rate (DDR) support. Because the IOB contains two independent I/O flip-flops and a dedicated multiplexer (MUX), it clocks-in and clocks-out twice as fast as an IOB with a single-data-rate I/O flip-

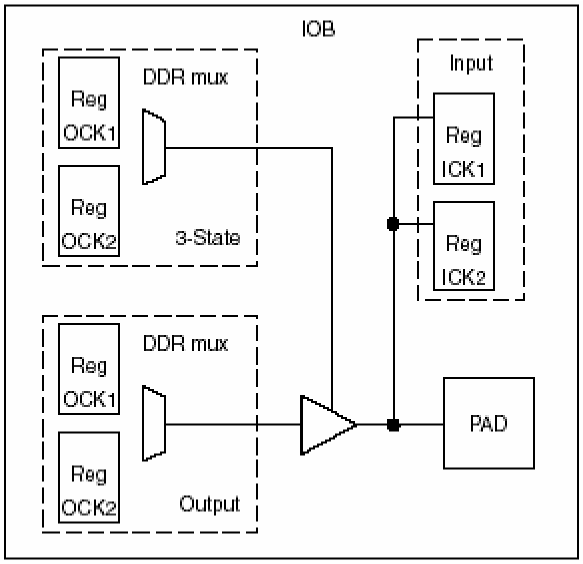


Figure 4: Virtex-II IOB block diagram

flop. Each bank, which can be independently configured, supports one of the output standards listed in Table 2.

Table 2: Single-ended I/O standards supported by the Virtex-II FPGA.

I/O standard	Output V_{CCO}	Input V_{CCO}	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	1.5	1.5	N/A	N/A
PCI33_3	3.3	3.3	N/A	N/A
PCI66_3	3.3	3.3	N/A	N/A
PCI-X	3.3	3.3	N/A	N/A
GTL	Note 1	Note 1	0.8	1.2
GTLP	Note 1	Note 1	1.0	1.5
HSTL_I	1.5	N/A	0.75	0.75
HSTL_II	1.5	N/A	0.75	0.75
HSTL_III	1.5	N/A	0.9	1.5
HSTL_IV	1.5	N/A	0.9	1.5
SSTL2_1	2.5	N/A	1.25	1.25
SSTL2_11	2.5	N/A	1.25	1.25
SSTL3_I	3.3	N/A	1.5	1.5
SSTL3_II	3.3	N/A	1.5	1.5
AGP-2X/AGP	3.3	N/A	1.32	N/A

1. V_{cco} of GTL or GTLP should not be lower than the termination voltage or the voltage seen at the I/O pad.

Each CLB contains four slices. Each slice consists of two 4-input LUTs and two edge-triggered flip-flops. Each LUT can be implemented as regular Boolean functions or as one 16 x 1 distributed RAM, or as one 16-bit shift register. Figure 5 is a block diagram of a Virtex-II FPGA slice.

Each block-select RAM is 18 kbit and can be configured into the following formats:

- 16k x 1 bit
- 8k x 2 bits
- 4k x 4 bits
- 2k x 9 bits (one parity bit)
- 1k x 18 bits (two parity bits)
- 512k x 36 bits (four parity bits)

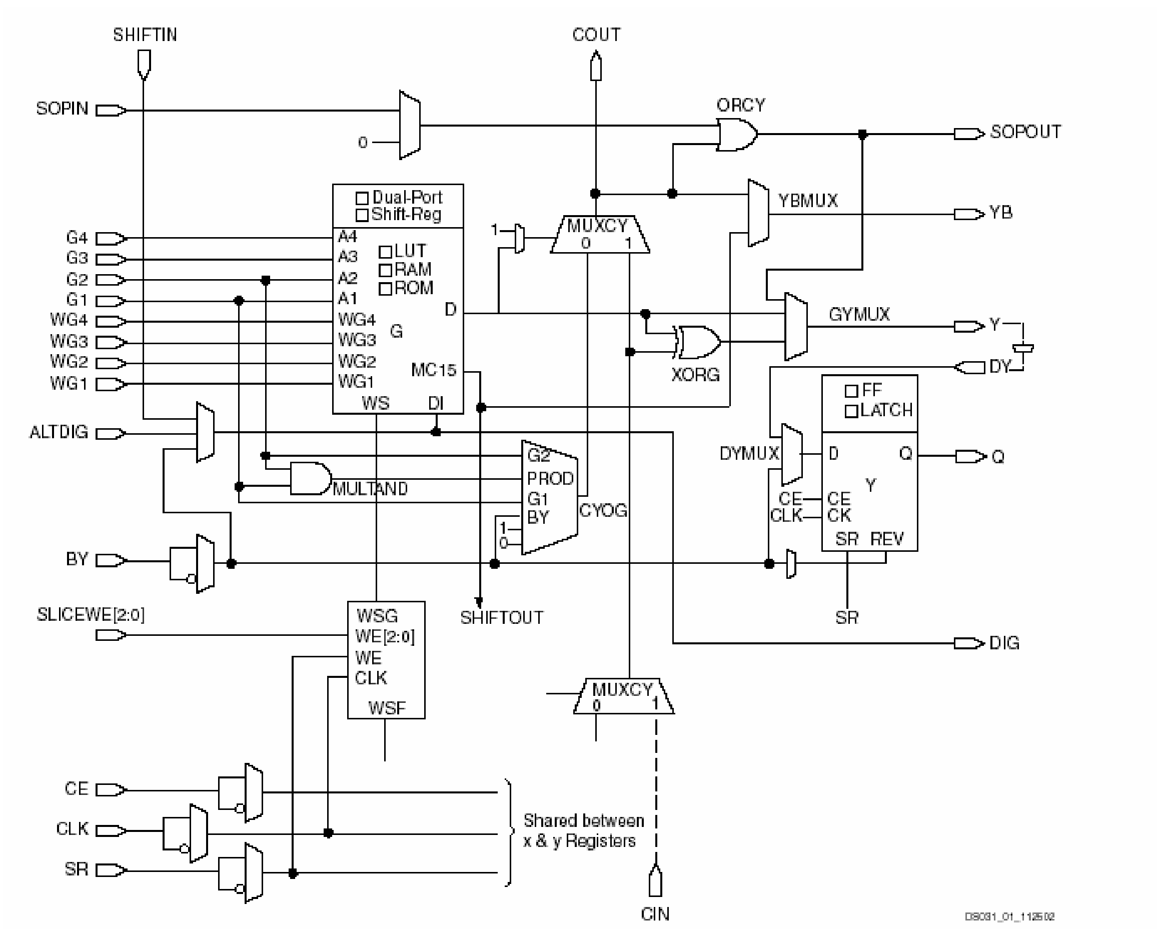


Figure 5: A block diagram of a Virtex-II FPGA slice

These true dual-port, synchronous RAMs can be cascaded or combined to form larger storage blocks or FIFO buffers. Figure 6 shows a typical usage of the block RAM as a dual-port RAM.

The 12 DCMs and 16 buffer general (BUFG) MUXs provide a complete solution for high-speed clocking schemes. The DCMs can phase-shift and synthesize (multiply or divide) frequencies. They can also deskew the internal clock with respect to a common external-clock source, to get rid of clock distribution delays. The BUFG MUX offers glitch-free clock-switching; up to four of them can be used for each DCM.

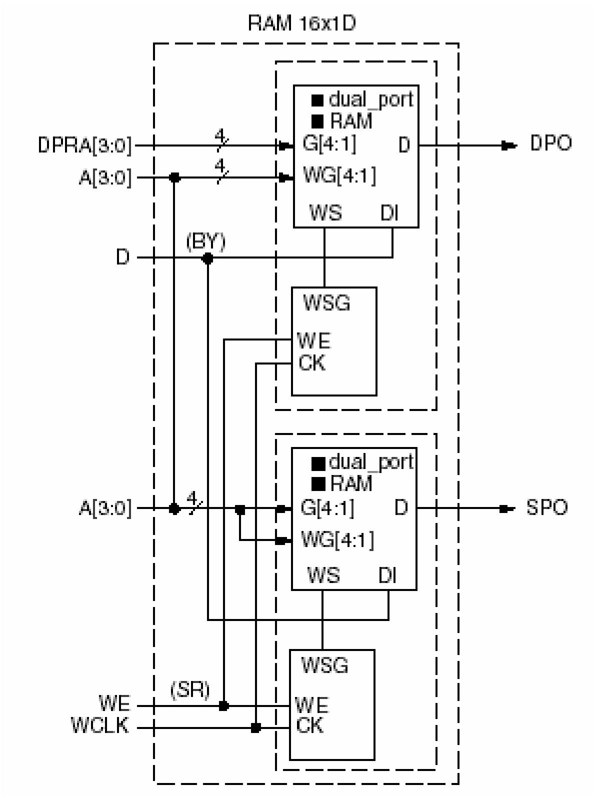


Figure 6: Block RAM used as dual-port RAM

3.1.2 FPGA Mode Switch, Configuration LED, and Temperature Sensor

Although not needed for the operation of the board, the FPGA pin-setting switch adds valuable options that can be used during testing. The SW1 switch allows the user to select a configuration shown in Table 3. The Master SelectMAP configuration is the default mode for the prototype.

An LED displays “PROGRAMMED” upon the successful completion of the configuration of the driver-board FPGA.

A temperature-sense connector indicates temperature overstress of the driver-board FPGA. However, because the driver board is outside of the temperature chamber, the sensing circuitry can by mounted directly next to DXN and DXP outputs of the FPGA.

Note: The preceding considerations depend on using the original system-maximum clock speed of 25 MHz. This value is used in the thermal consideration and analysis (see Section 4.2). Significantly increasing the clock speed for any reason (such as for speeding up the test or communication) will require the thermal load on the UUT to be reevaluated.

Table 3: Virtex-II configuration mode pin settings. Master SelectMAP is the default mode for our configuration.

Configuration mode[1]	M0	M1	M2	CCLK direction	Data width	Serial DOUT[2]
Master Serial	0	0	0	OUT	1	YES
Slave Serial	1	1	1	IN	1	YES
Master SelectMAP	**0**	**1**	**1**	**OUT**	**8**	**NO**
Slave SelectMAP[3]	1	1	0	IN	8	NO
Boundary Scan (IEEE 1532)	1	0	1	N/A	1	NO

1. The HSWAP_EN pin controls the pull-ups. Setting M2, M1, and M0 selects the configuration mode, whereas the HSWAP_EN pin controls whether the pull-ups are used.
2. Daisy chaining is possible only in modes where Serial DOUT is used. For example, in SelectMAP modes, the first device does not support daisy chaining of downstream devices.
3. An external oscillator is required for this mode.

3.1.3 XC18V04 EEPROM

Before a test is conducted, the configuration string for the driver board FPGA is downloaded from the PC, via the JTAG interface, into an XC18V04 electrically erasable PROM (EEPROM) (Xilinx P/N XC18VQ44C). This 4,194,304-bit EEPROM is designed for reprogramming and storing various sizes of Xilinx FPGA-configuration bit streams. The EEPROM supports both in-system programming and IEEE 1149.1 boundary-scan (JTAG) testing, via a single four-wire test-access port (TAP). Xilinx guarantees the endurance level of 20,000 program/erase cycles and data retention of at least 20 years.

The block diagram of the XC1804 EEPROM is shown in Figure 7. Users may configure the XC18V04 to either a serial or parallel mode, through a user control register in the XC18V04. This control register is accessed through the JTAG interface and is set using the “parallel mode” setting on the Xilinx iMPACT software. Serial output is the default configuration mode. An on-chip pull-up resistor automatically holds the mode at a defined level during normal operation.

The UUT FPGA requires 10,494,368-bit configuration string. If an FPGA is tested using the SelectMAP configuration mode, which requires a larger configuration memory, the EEPROMs on the driver board are cascaded to provide additional memory (Figure 8). There are three XC18V04 EEPROMs on the driver board. These EEPROMs can be concatenated by using the chip-enabled output (CEO) to drive the CE input of the downstream device.

When these EEPROMs are concatenated, the clock inputs and the data outputs of all EEPROMs in the chain are interconnected. After the last datum from the first PROM is read, the next clock signal to the PROM asserts its CEO output at a low-impedence state and drives its DATA line to a high-impedance state (“high”). The second PROM recognizes the low level on its CE input and enables its DATA line output. After the configuration is complete, address counters of all cascaded PROMs are reset if the PROM OE/RESET pin goes low or if the CE goes high.[1]

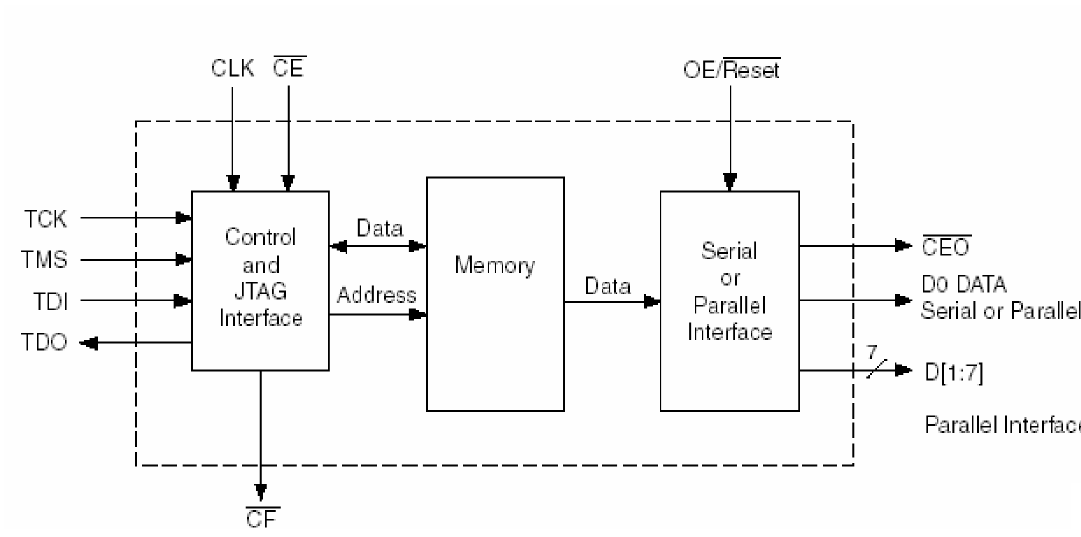

CLK
CE
OE/Reset
TCK
TMS
TDI
TDO
Control and JTAG Interface
Data
Address
Memory
Data
Serial or Parallel Interface
CEO
D0 DATA Serial or Parallel Mode
7
D[1:7]
Parallel Interface
CF

Figure 7: XC18V04-series PROM block diagram

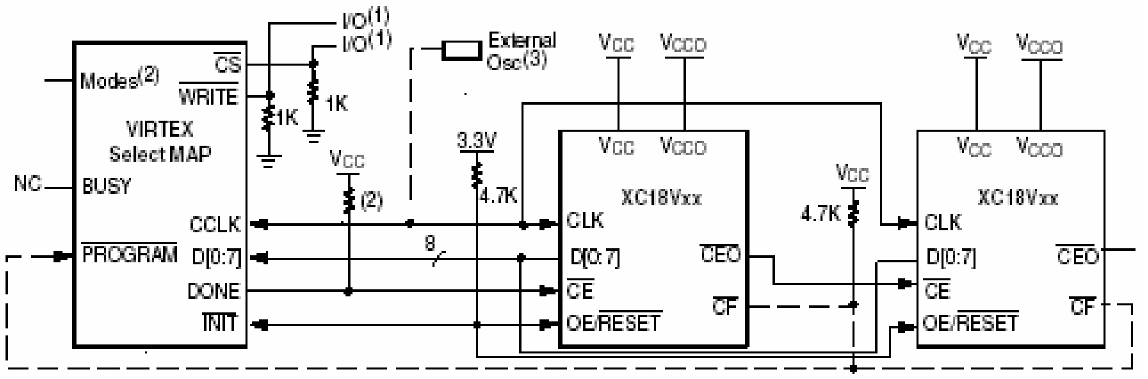


Figure 8: Cascading multiple EEPROMs (with the FPGA in SelectMAP Mode).

3.1.4 Parallel Port Interface

The driver board has two parallel ports. Only one is required for operation; the second one is for expansion/enhancement. The ports are used for bidirectional asynchronous communication with the host PC, using a parallel IV cable (Xilinx P/N HW-PC4-ND).

Upon the successful configuration of the driver board FPGA, the enhanced parallel port (EPP) interface between the driver board and the PC becomes operational. This interface uses the standard EPP protocol. Data transferred to the driver board consists of any number of configurations/bit streams for the UUT FPGA, test

[1] The CS and WRITE, must be either driven low, or pulled down externally.

vectors, and test control signals. Data flow from the driver board carries test results and test status signals.

Each Virtex-II FPGA I/O has a pair of clamp diodes that connects to V_{CCO} and ground (GND), as shown in Figure 9.

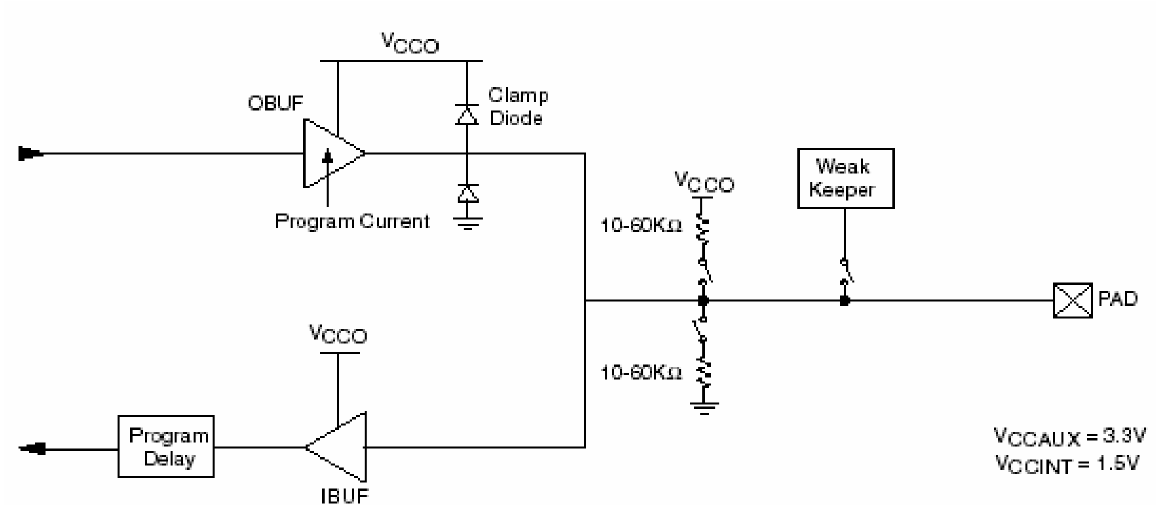


Figure 9: LVT, LVCMOS or PCI select I/O-ultra standards

In spite of all the advantages of using the EPP interface, there is a problem with hardware interface compatibility: Virtex I/Os configures as low-voltage transistor–transistor logic (LVT) using 3.3 V, whereas the PC uses 5.0-V transistor–transitor logic (TTL). Virtex-II FGPA I/Os are not 5-V tolerant without adding an external current-limiting resistor.

A workaround is to use a resistor in series to limit the current into the clamp diode. This workaround is reliable when the higher voltage is driving the FPGA input. It is not reliable when the higher voltage is driving FPGA output or a bidirectional signal, because the resulting V_{OH} may be lower than that specified for the other device.

To solve the problem for bidirectional communication, a 74ALVC164245 3-V–5-V, 16-bit-level shifter manufactured by Philips Semiconductor is used. Its block diagram is shown in Figure 10.

When either LVT or low-voltage CMOS (LVC) with 5-V tolerant outputs are used, a direct communication can be established between the FPGA and any 5-V TTL-level transceiver. When other low-voltage families are used as the UUT FPGA, the dual-VCC-level shifters provide the transceiver function, with built-in level shifting and the prevention of current flows between power supplies. Dual V_{CC} level shifters are superior to alternatives such as the input pull-up resistors, blocking diodes, and other circuits that normally degrade speed and/or noise margins.

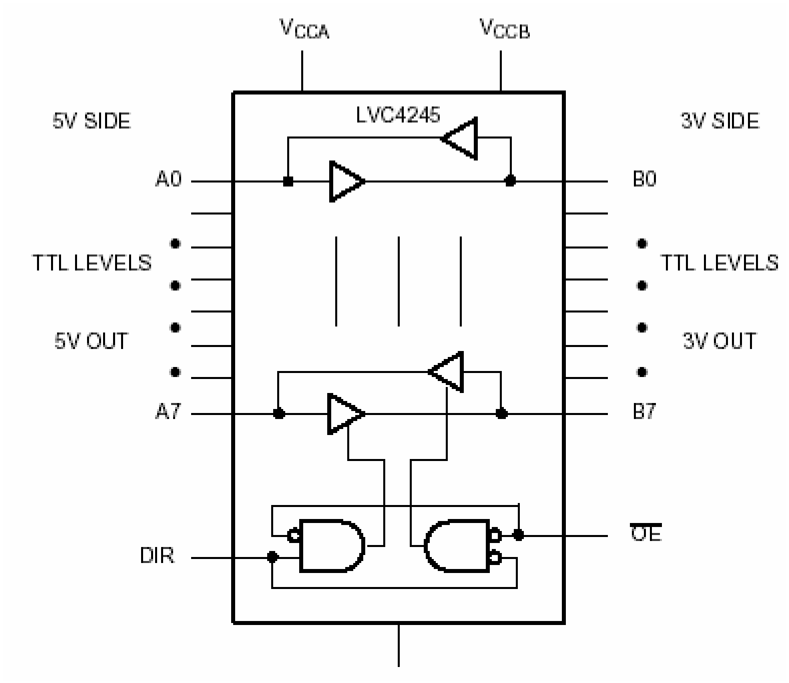


Figure 10: Slice of 74ALVC164245 level shifter

A general diagram for bidirectional data communication between 5-V and 3-V systems is shown in Figure 11.

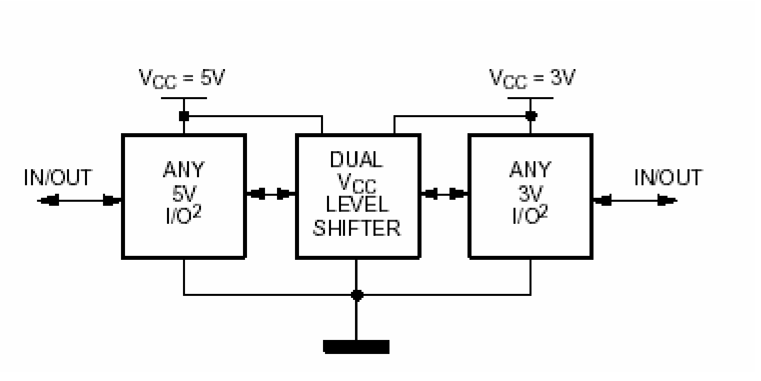


Figure 11: 5-V transceivers on common bus with 3-V transceivers

3.1.5 Clocks

The oscillators on the driver board operate at ambient temperatures and therefore can be of commercial grade. The driver board has three independent, tristate oscillators (Epson P/N SG-8002DC). Detailed specifications are given in Appendix D.

During testing, the operator can utilize from one to all three oscillators. Alternatively, the operator may also choose to use the oscillators on the UUT board. There are 120 clock combinations available to the operator, demonstrating the versatility and various resources offered in our design.

3.2 INTERFACES

The driver FPGA is equipped with five types of interfaces:

- Host-PC EPP interface
- UUT FPGA download/readback interface
- Test-vectors sending/retrieving interface
- Internal/external test-vector storage interface
- Internal/external fault-detection interfaces

Some of these interfaces act alone and are not aware of the others. Some of these interfaces work closely with other interfaces to function as a flow, to complete a complex task. Most of the interfaces are synchronous—i.e., they are triggered on the same clock edge that flows into the chip. The test-vector sending/receiving interfaces may contain some asynchronous FIFO buffers, to handle data communications between different clock sources. All coding is done in the Very-High-Speed Integrated-Circuit Hardware-Description Language (VHDL). (IEEE standard 1164–93).

3.2.1 Host-PC EPP interface

The host-PC EPP interface is crucial to keeping the entire system “alive” from the user’s point of view. It provides the only way of knowing that the system is responding to human interactions. An EPP interface is used because it is only Enhanced Parallel Port interface that supports LVTTL configuration of FPGA I/O’s eliminating the need for 5 V to 3.3 V level shifters. EPP offers data bandwidth is 500kByte–2MByte per second. Table 4 shows the definition of the EPP signals, per IEEE 1284.

Table 4: EPP signals as defined by IEEE 1284

SPP signal	EPP signal name	EPP signal description	Comments
nSTROBE	nWRITE	Out	Active low. Indicates a write operation is high for a read cycle.
nAUTOFEED	nDATASTB	Out	Active low. Indicates a Data_Read or Data_Write operation is in process.
nSELECTIN	nADDRSTB	Out	Active low. Indicates an Address_Read or Address_Write operation is in process.
nINIT	nRESET	Out	Active low. Peripheral reset.
NACK	nINTR	In	Peripheral interrupt. Used to generate an interrupt to the host.

Table 4: EPP signals as defined by IEEE 1284

SPP signal	EPP signal name	EPP signal description	Comments
BUSY	nWAIT	In	Handshake signal. When low, indicates that it is OK to start a cycle (assert a strobe); when high, indicates that it is OK to end the cycle (de-assert a strobe).
D[8:1]	AD[8:1]	Bi-Di	Bidirectional address/data lines.
PE	user defined	In	Can be used differently by each peripheral
SELECT	user defined	In	Can be used differently by each peripheral.
nERROR	user defined	In	Can be used differently by each peripheral.

3.2.2 UUT FPGA download/readback interface

The UUT FPGA download/readback interface is a SelectMAP eight-bit parallel interface that includes the bits shown in Table 5. It is used to download and readback bit-streams to/from the UUT FPGA. The host PC provides the configuration for the UTT FPGA. To utilize this interface, the user must set the driver FPGA to the slave SelectMAP mode. The FPGA's done pin requires some pull-up resistors to function properly. The same scenario applies to the readback function. Figure 12 shows download/readback timing diagrams.

Table 5: Bits in the SelectMAP interface.

Name	Functionality	Pin assignment
PROG_N	Master-device issue program attention	
INIT_N	Slave device acknowledge	
DONE	Configuration complete	
CCLK	Download bit stream synchronizing clock	
RDWR_N	Data bidirection control	
BUSY		
CS_N	Device chip select	
DATA (7:0)	8-bit bidirectional data shared with test-vector data bus	

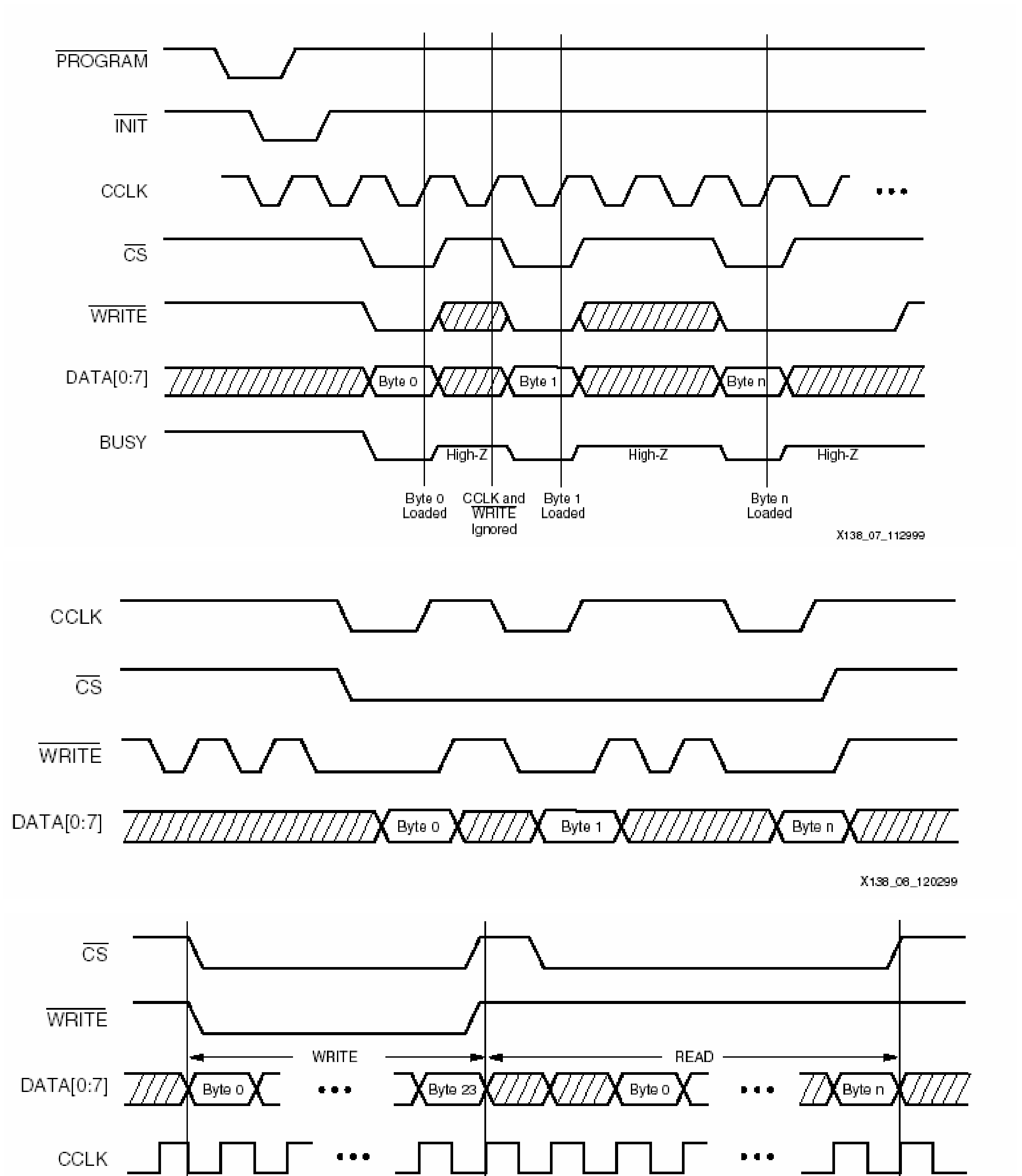


Figure 12: Download/readback timing diagrams.

3.2.3 Test-vectors sending/retrieving interface

The test-vector sending/retrieving interface sends test vectors between the driver FPGA and the UTT FPGA. As shown in Table 6, data-transaction protocols are defined in order to send and receive test vectors/test results back and forth between the driver FPGA and the UUT FPGA.

The test vectors are sent from the driver FPGA to UUT FPGA. After the UUT FPGA programming is downloaded and verified, the driver FPGA reads out test vectors from the synchronized SRAM (SSRAM), sets the necessary control bits, and then asserts the TX_EN while clocking out data to the UUT FPGA. In the meantime, if the UUT FPGA has valid data, it will also clock-out test results back to the driver FPGA via the RX_EN and RX_DATA bus.

Table 6: Data transaction protocols for sending results from the driver FPGA to the UUT FPGA.

Name	Definition
TX_CONTROL [2:0]	Specify modes of operation from the driver FPGA
TX_EN	TX data valid
TX_DATA [7:0]	8-bit data one directional, driving from the driver FPGA
RX_EN	UUT FPGA data valid
RX_DATA [7:0]	8-bit data one directional, driving from the UUT FPGA

3.2.4 Internal/external test-vector storage interface

The internal/external test-vector storage interface is used to move test vectors between the driver FPGA and the UUT FPGA. The software or the user may provide any random data as desired. It can be the same value—AA or 55—or entirely random, and with any kind of distribution. It will be stored initially in a text/data file on the host PC’s hard disk. (In the prototype, to simplify matters, the test vectors are not generated by the hardware but are provided by software.)

The user can change the test vectors at any time. Even with their relatively small total size, they can still result in a variety of data patterns to stress the test FPGA. For the prototype, eight of the internal block RAM chips are used. The block RAM chips are 16 kbit each, resulting in a total test-vector size of 128 kbit (16 kByte). Although not huge, the test vectors are large enough to demonstrate the hardware functionalities.

The eight-block RAM chips are combined into one 8-bit-wide, 14-bit-deep true dual-port RAM, which means one can independently read/write on both ports. Port A of the RAM chips is used as both a read and write port. Initially, it is used as the write port to set up the test vectors. Once all the addresses are walked through and the user gives the start command to start sending test vectors to the test FPGA, port A’s address will be driven by another counter/state machine, to act as a read port for pumping data out of it. The software moves the test vectors files from the host PC’s hard disk to the internal/external RAM onto the driver board.

3.2.5 Internal/external fault-detection interfaces

The internal and external fault-detection interfaces are used for fault detection. They move error statistics between the driver FPGA and the UUT FPGA. The fault detection approach used depends on the design example employed.

The fault detection method is the same for the shifter and FIFO design examples. In the prototype, Port A is used for sending out the test vectors to the test FPGA. Port B is then used for comparing the data coming back from the test FPGA. Once the RXEN signal goes from zero to one, data are read out, starting from address zero of Port B, and those data are compared to the received data. This approach works for the shifter and FIFO design samples because the data should remain unchanged.

If there is a discrepancy, an error bit is set for that clock, and the error counter increments by one, to keep track of the discrepancy. Two sets of eight-bit counters show the error statistics. Two sets are used based on the assumption that the error rate will not be big enough to overflow the eight-bit counter in any EPP software-polling interval; otherwise, the counter will roll over. One counter is used to count and another one is used to latch the counting one, in case the software wants to update the error-total counts. They ping-pong between each other; for example, if counter A is being read, it latches counter B's value, counter B's value is cleared to zero, and counter B starts counting from scratch. If there is a failure simultaneous with a read event, count B's initial value is set to one—not zero, like it is done normally. This way, it is impossible to miss any error count while the total error count is accumulated and maintained by the software.

The hardware merely keeps track of the incremental values—it is up to the user to start or stop the comparison. Once the RXEN goes from one to zero, the error counting and data comparison stop. The software continues to display the total error count from the last test.

In the calculator design example, fault detection is a bit complicated. It has multiple CRC chains, and results for each chain are not known until all the data bits are shifted. However, the driver FPGA knows the total number of CRC chains and knows when the data has been sent. After the data have been sent, the driver FPGA pulls the results back from the UUT FPGA. It compares the CRCs of each chain with its prerecorded value.

If there is a discrepancy, the driver FPGA report error counts by using the same counter mechanism as in the shifter and FIFO design samples.

3.3 FPGA TIMING

Timing diagrams for EPP interface are provided for these cycles:

- Data Write Cycle
- Address Write Cycle
- Data Read Cycle
- Address Read Cycle

3.3.1 Data Write Cycle

Figure 13 shows the data write cycle, in which:

1 The program writes to the EPP data register. (Base + 4)
2 nWrite is placed to low. (Low indicates the write operation.)
3 Data are placed on data lines 0–7.
4 The nData Strobe is asserted if Wait is low (i.e., it is O.K. to start the cycle).
5 The host PC waits for acknowledgment by nWait going to high (i.e., it is O.K. to end the cycle).
6 The nData Strobe is de-asserted.

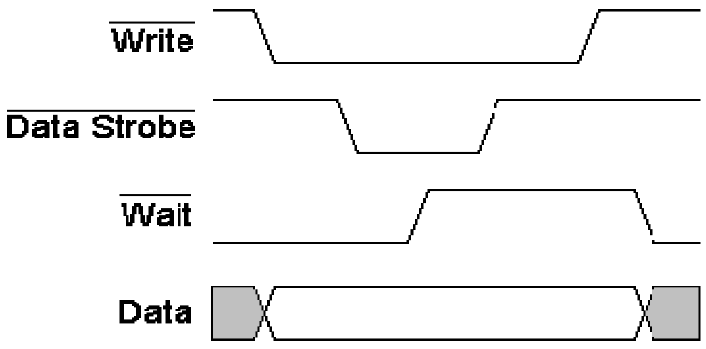


Figure 13: EPP data write cycle.

3.3.2 Address Write Cycle

Figure 14 shows the address write cycle, in which:

1 The program writes address to the EPP's address register (Base + 3)
2 Write is placed to low. (Low indicates the write operation.)
3 The address is placed on data lines 0–7.
4 The Address Strobe is asserted if Wait is low (i.e., it is O.K. to start the cycle).
5 The host PC waits for acknowledgment by Wait going to high (i.e. it is O.K. to end the cycle).
6 The nAddress Strobe is de-asserted, signaling the end of the cycle.

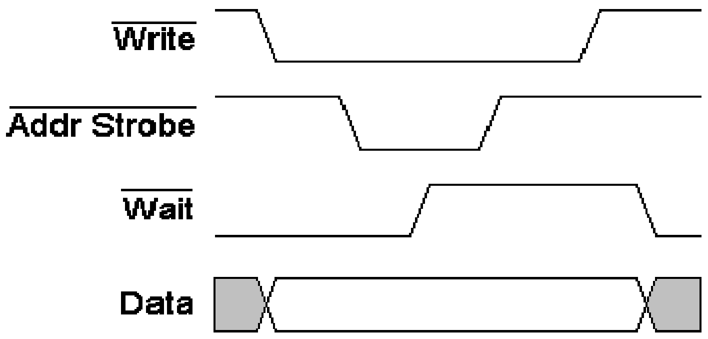


Figure 14: EPP address write cycle.

3.3.3 Data Read Cycle

Figure 15 shows the data read cycle, in which:

1 The program reads the EPP data register. (Base + 4)
2 The nData strobe is asserted if Wait is lowl (i.e., it is O.K. to the start cycle)
3 The host PC waits for acknowledgment by nWait going to high.
4 Data are read from the parallel-port pins.
5 The nData Strobe is de-asserted, signaling the end of the cycle.

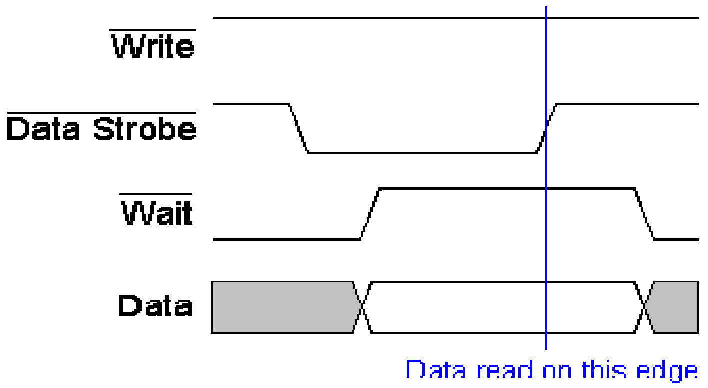


Figure 15: EPP data read cycle.

3.3.4 Address Read Cycle

Figure 16 shows the address read cycle, in which:

1 The program reads the EPP address register. (Base + 3)
2 The nAddr Strobe is asserted if Wait is low (i.e., it is O.K. to start the cycle).
3 The host PC waits for acknowledgment by nWait going to high.
4 Data is read from the parallel-port pins.
5 nAddr Strobe is de-asserted, signaling the end of the cycle.

Table 7 is a table of registers and their definition:

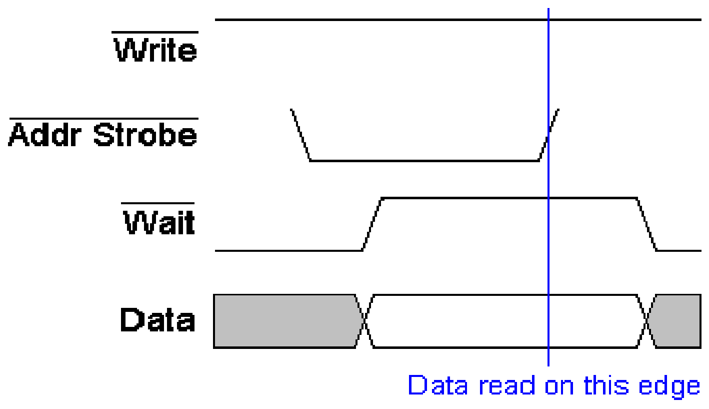


Figure 16: EPP address read cycle.

Table 7: Parallel IO register definitions.

Name	Bit definition	Notes
Control register	Bit7–6: chip select encoding: 00: Driver FPGA 01: UUT FPGA Bit5–4: reserved Bit3–0: 0001: read UUT FPGA version 0010: download 0011: readback 0100: writes synchronous static RAM (SSRAM) 0101: read test results 0110: reset board 0111: start test 1000: read driver FPGA version	Sized 0 x 01 Write only ≤ 15 commands
Status register	Bit7: Tx done (W) Bit6: download done (W) Bit5: Rx ready(R) BIt4: readback done(R) Bit3: test result ready(R) Bit2: UUT FPGA download complete Bit1: UUT FPGA ready to accept download data (R) Bit0: Driver FPGA RAM	Sized 0 x 02 Read/write
Data register	8-bit data	Sized 0 x 03 Read/write
Address register	Index to the three registers	

The software routines for accessing this register and to issue commands are described in Table 8:

Table 8: Software routines for the I/O register.

Routine	Description
Reset board	Any time the system has any problem or to abort the current operation and start over, 0 x 06 is written into the control register, to reset the board.
Read-revision sequence	Writes 0 x 01 to the control register, reads the status register, and if bit 6 is set, reads the data register.
Download sequence	Writes 0 x 12 to the control register, reads the status register if bit 6 is set, writes the UUT FPGA’s configuration data to the data register, and, when done writing the data, writes 0 x 20 to the status register, to indicate the end of the configuration data.
Readback sequence	Writes 0 x 13 to the control register and reads status register. If bit 6 is set and bit 4 is not set, reads the data register; if the bit 4 is set, the current FPGA configuration bit is done reading.
Write SSRAM/internal RAM sequence	Writes 0 x 14 to the control register, reads status register if bit 6 is set, writes to the data register the first byte of the FPGA’s test vector. When done writing the test vector, writes 0 x 80 to the status register to indicate the last byte of the test vectors.
Read-result sequence	After finishing sending the test patterns, reads the status register. If bit 3 is set, reads test result from the data register.

4 Detailed Description of the UUT Board

One of the major objectives in the design of the burn-in (UUT) board is to maximize reliability and signal integrity. The design achieves this by minimizing the number of components on the board while preserving the desired versatility of the board. The bill of materials for the UUT board is given in Appendix B. Complete schematics are given in Appendix E. The PDF conversion of the layout files (Gerber files) is given in Appendix F.

4.1 COMPONENTS

Essential components are those used during test operations. All essential components of the UUT board are military grade. Nonessential features of the UUT board are a mode switch and a temperature sensor.

The UUT board consists of 14 layers of polyimide assembly with an uninterrupted ground plane, one plane for core supply V_{CCINT} (1.5 V) and one plane for V_{CCAUX} (3.3 V). V_{CCO} (3.3 V) is distributed on wide signal traces with sufficient bypass capacitors.

4.1.1 UUT FPGA

The UUT FPGA used in the prototype is the same as the driver FPGA and is discussed in detail in Section 3.1.1.

The components of the FPGAs that are most utilized are also the most susceptible to failures. Such components include the LUTs, and block RAMs, and other arithmetic logics. For most designs, these are the three major components most in demand, both in terms of quantity and quality. The degree to which these components are utilized, how they are interconnected, and the timing that can be achieved directly dictate how well the FPGA performs. Thus, the focus of the platform is to test these specific components.

4.1.2 Capacitors

Because the power consumption is dynamic, it is nontrivial to ensure stable supply voltages at the device pins and to minimize ground differentials. Fast-changing Icc transitions are supplied by 104 local-decoupling capacitors, placed proximal to the V_{CC} device pins. These capacitors must have enough capacitance to supply Icc for a few ns, and they must have low intrinsic resistance and inductance. The 0.1-µF negative-and-positive-to-zero (NPO) ceramic, surface-mounted capacitors can each supply 1A for 2 ns, with a 20-mV voltage drop.

To compensate for trace inductance—a 10-mm trace represents an inductance of several nanohenries, defeating the purpose of the decoupling capacitor—twenty-four 4.7-µF and 2.2-µF tantalum capacitors were positioned in various locations.

Finally, four 100-µF power-supply-decoupling electrolytic capacitors are used to supply even more current for a portion of the supply-switching period. (As a general rule, multiple smaller capacitors in parallel always offer lower resistance and inductance than any single large capacitor.)

4.1.3 Sockets and Connectors

The following sections describe the sockets used in the prototype. The details of the configuration connectors may be found in Appendix G.

4.1.3.1 UUT Socket

There is an obvious need to test more than one FPGA. Because using multiple soldering processes to replace the UUT FPGA will rapidly destroy the UUT board or/and the UUT FPGA itself, the UUT FPGA is placed on an Enplas728-pin BGA socket in the center of the board (Figure 17). This socket allows for easy, solderless replacement of the UUT and is rated at an operating temperature range of –60°C ~ +150°C. A detailed drawing and specification of the socket is in Appendix H.

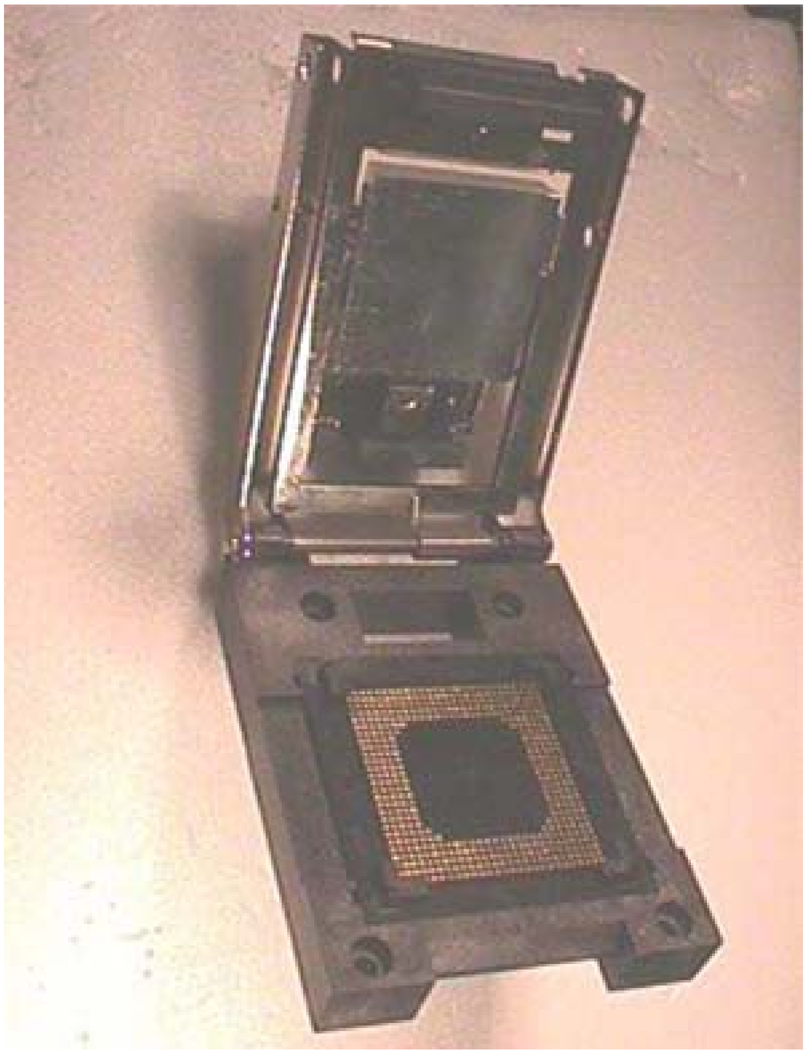


Figure 17: Socket for the UUT FPGA

The socket is fanned-out into six 76-pin, 50-ohm Mictor (or equivalent) connectors. These connectors are used for the general I/O interface, maximizing the number of I/Os possible for a given FPGA. One more 38-pin Mictor connector is dedicated to the configuration/readback function.

This socket was selected because it meets the following requirements:

- It is rated to work in harsh environments, to allow for high-temperature burn-in and reliability testing of the FPGA.
- It satisfies design frequency and power requirements.
- It allows easy access and solderless replacement of the UUT FPGA.
- Its minimal complexity ensures the maximum MTBF.
- It satisfies Xilinx’s mechanical specifications for mounting the FPGA: a sustained, direct, compressive force applied normally to the lid, using a tool head that does not exceed 4.0 grams per external ball.

To restate the last requirement, the UUT FPGA must not be crushed. The FPGA has a BGA flip-chip 35 x 35-mm^2 packaging with 1.27-mm spacing. Flip-chip packaging places the die in the package face down. The die is connected to I/Os by small “bumps” of solder. This configuration eliminates wire bonding and allows for shorter interconnections between circuits and the I/O pins, providing more robust, smaller and faster interconnects. A flip-chip package offers more available I/Os than a wire-bond package. Flip-chip package also improves heat dissipation (i.e., has lower thermal resistance). The package construction is shown in Figure 18.

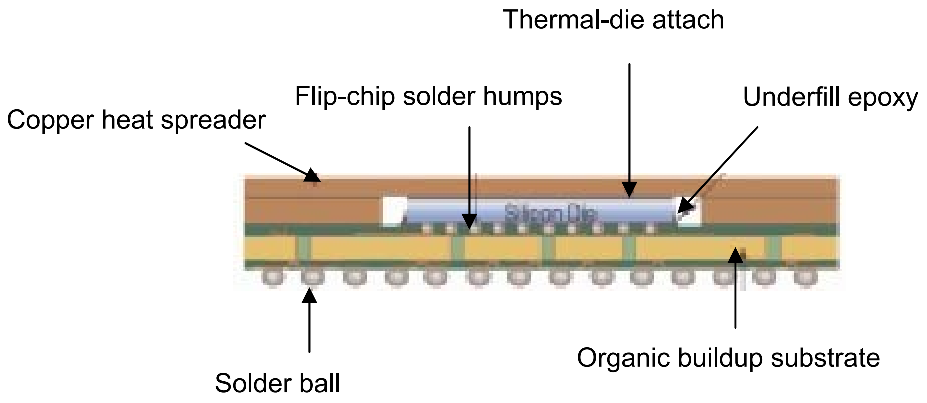


Figure 18: BGA package construction.

4.1.3.2 *Clock Sockets*

Adding versatility to the UUT board are two sockets for dedicated oscillators. These sockets allow the easy replacement of oscillators, should different frequency sources be desired. The reason for having two sockets is to provide multiple, simultaneous sources of clocks and to enhance resources during the testing and exercising of the DCM feature of the FPGA. The disadvantage of sockets is that they require via-holes, which in turn may have some negative effect on signal integrity at very high frequencies.

4.1.4 Clocks (Oscillators)

The optional, onboard oscillators on the UUT are high-reliability versions by the Sematec Company. Their operating temperature range is –55°C ~ +150°C. Detailed specifications of the oscillators are given in the Appendix D.

If onboard oscillators are not installed on the UUT board, one of the three oscillators on the driver board can be used. This approach make a few more clock combinations available to the user. All the oscillators feature tristate outputs, which allow software control over the clocks. An oscillator is selected by a jumper close to one of the clock inputs of the FPGA.

4.1.5 Interfaces and Clocking Scheme Options

The interface between driver FPGA and the UUT FPGA's consist of two 8-bit data buses. Due to signal integrity consideration each bus is unidirectional: one passes data from driver to UUT and another one delivers response from UUT back to the driver. Each bas consists of 12 lines: three control, one clock and eight data bits:

- TXEN as the data valid
- TXDATA [8:0] as the eight-bit data flowing into the UUT FPGA
- TXCLK, together in phase with the TXDATA
- Two bits of TXCONTROL, for four possible conditions

At higher (over 100 MHz) frequencies there aree several options to compensate for transmission line (cable) delay and synchronize the exchange of data between the two boards. As a first option, one clock is used on both boards. The driver board sends the TXCLK together with its data and the UUT FPGA either clocks all its internal logic with this TXCLK or with the falling edge of the TXCLK. Either way, when the UUT FPGA sends the data back to the driver FPGA, it does not send back the clock.

To compensate for the cable delay, the driver FPGA must phase-shift its clock, to be able to correctly clock-in the data. If the UUT FPGA uses the falling edge of the TXCLK, the driver FPGA still uses its original clock. Therefore, two clock domains may exist inside the driver FPGA but only one in the UUT FPGA. This option works fine if the operating frequency is relatively slow, since it is a lot easier and safer for one fixed-phase shift value to work for any board, under any temperature variation.

For a higher-speed clock rate, the phase-shift window is a lot smaller and the first choice is a gamble. As a second option, a special cable is provided that enables only one clock domain to be used for the entire system, to compensate the delay introduced by the long cable. The Virtex-II FPGA has DCM, which can deskew clocks, given the proper feedbacks. If an external trace is made on the cable so that the total length of the trace equals the total length of the cable, then the same clock source can be used for both boards.

The driver board can be made to clock out data "early" enough to compensate for the cable delay, so that when the UUT FPGA gets the data, the set-up and hold times are satisfied. However, this approach requires one clock source to equally distribute the clock to both boards, using the exactly the same-length feedback path. This is hard to do. It is also difficult to make a feedback path exactly the same length as the cable.

A third option is to make the two FPGAs operate under totally independent clocks—i.e., the two clocks have no phase relationship whatsoever, even if they are relatively equal in terms of frequencies. This option leads to the simplest PCB solution but at the cost of a more-complex interface. The driver FPGA send its

TXCLK along with its data to the UUT FPGA, and the UUT FPGA uses only the TXCLK to clock in data at its IOB. The UUT FPGA then uses an asynchronous FIFO buffer to convert the data into its own clock domain. When the UUT FPGA clocks out test results back to the driver FPGA, the driver FPGA does the same, so that the driver FPGA will also have another asynchronous FIFO buffer to reverse-convert the data into its own clock domain.

Although this third option works at any clock frequency, flow control becomes an issue over time: the differences between the two different clocks will build up, causing one side to starve and the other side to overflow. The interface logic design must be robust enough to tolerate such conditions.

The prototype uses the first option, since that is the simplest option under low frequencies. To drive good clock signals out of the IOB, not just any IOB may be used—its delay cannot be guaranteed and the signal may be distorted. The Xilinx double-data-rate I/O solves this issue by "regenerating" the clock, using two flip-flops clocked by two clocks that are 180 degrees out of phase. One flip-flop outputs a logic one ("1") and the other outputs a logic zero ("0"). This way, the IOB operates with exactly the same delay timing as the data that is clocked by the output flip-flop on (other) regular IOBs. This method is widely used for high-speed off-chip interconnection systems.

4.1.6 FPGA Mode Switch and Temperature Sensor

As with the driver board, a SW1 switch on the UUT board to allow the selection of the configuration modes described in Table 3. The UTT board modes should match the driver board modes. The truth table for mode selection is given in Table 3.

A temperature-sense connector on the UUT indicates temperature overstress of the UUT FPGA junctions. This optional feature takes advantage of DXN and DXP signals from the UUT FGPA. To activate this feature, these signals are routed outside the chamber, as differential outputs, to the Maxell temperature-sensing device (MAX11617AMEE). The MAX11617AMEE offers autonomous alarming without requiring interaction by a host controller, generating an interrupt when a user-programmable upper or lower threshold is exceeded, resulting in stand-alone temperature monitoring.

4.2 Thermal Considerations and Analysis

The UUT board has features that affect the UUT FPGA's thermal parameters.

The challenge in predicting power requirements is that they are directly related to the degree of chip utilization and the frequency of operation. Extensive utilization increases power consumption and may result in the integrated circuit generating excessive heat. The FPGA chip could easily exceed the maximum allowable junction temperature, which in turn may degrade performance. Maintaining the balance between chip utilization and power requirements ensures the proper operating conditions for the chip.

Four factors determine the temperature of an FPGA die:

- Total power dissipation
- Package thermal resistance
- Ambient temperature
- Airflow

These factors must be managed so that they stay below the maximum junction temperature of the die (125°C). Given the temperature inside the case or rack (T_{amb} = 125°C), the total power dissipation of the FPGA (P), and the thermal resistance of the package (ΘJA), the junction temperature (TJ) may be calculated as:

$TJ=T_{amb} + P(\Theta JA)$

Where the junction–to–ambient; ΘJA ranges from 12 to 20°C/watt,[2] and the value of P depends on the percent chip utilization.

For the worst case (e.g., 85%) chip utilization, power calculations tell us that P = 2 watts, and that the junction temperature (TJ) maximizes at:

$TJ = 125 + (2 * 20) = 165°C$

The chip utilization is the function of the test vector used during the given stage of the test. Because any number of test vectors can be used, the junction temperature may vary. Xilinx recommends using passive heat sinks for this type of power dissipation. An aluminum BGA socket is used to house the XC2V3000, as described in Section 4.1.3.1. The socket serves as a passive heat sink, which in turn allows us to reduce the junction temperature. Moderate airflow present in the temperature chamber will further help reduce the temperature of the device junctions. For intensive testing, the surface temperature of the device can be monitored with an inexpensive digital thermometer.

For the prototype, the junction-to-case thermal resistance (JC) is considered to be 1.5°C/watt. Simple calculations show that the surface temperature is 3°C below the junction temperature. This value is then taken into account and is used as offset in measuring the surface temperature.

Note: If passive burn-in testing is desired, the temperature of the junctions can be approximated to the ambient temperature of 125°C and will not adversely affect the UUT FPGA.

Calculations, assumptions, and the preceding analysis show that testing the Virtex-II FPGA at 125°C, using a 20-MHz main clock frequency with variable test

[2] From the Xilinx user guide for the BG728 package with a 1.27-mm pitch.

vectors at 80% chip utilization, requires careful temperature monitoring. Depending on the objectives of the given test, aggressive heat-sinking measures may also be required.

5 Detailed Description of the Board Connection Cables

The electromechanical part of the prototype design—the cable connecting the driver board and the UUT board—presented a big challenge. The original plan was to use a miniature, shielded ribbon cable manufactured by Precision Interconnect ("Blue Ribbon"). However, testing revealed that the operational temperature for the cable is only 60°C, which resulted in the manufacturer downgrading the cable specification. Currently there is no technology known to us that will allow the manufacturing of a miniature (30 AWG or less), high-temperature ribbon cable.

Thus the prototype uses bulkier individual coaxial cables. Even with this cable choice, over a month of searching resulted in only one supplier: Astrolab, which manufactures a 50-ohm Teflon coax cable.

The driver board is connected to the UUT board with three sets of coaxial cables:

- One set of 76 input lines
- One set of 76 output lines
- One set of 38 lines dedicated to the UUT programming, readback, and control.

There are two cable assemblies. Only the data cable is required. The data cable assembly consists of a 1.0"x 1.5" four-layer interface board; two high-density 76-pin, 50-ohm connectors; and 76 individual coaxial cables, each 3 feet long. Currently two of these cables are manufactured—one for source, and another one for inputs—with a total of 152 lines.

The purpose of the second cable assembly is to program the UUT FPGA and to support the readback test. The number of lines in this cable is 20. It uses 38-pin, 50-ohm connectors with a design similar to those for the data cable: two interface boards, two 38-pin connectors, and a cable.

Electrical and environmental information on the cable assemblies and photograph of the components used in cable assembly is given in Appendix I.

With the heavy traffic of high-speed logic signals, signal integrity becomes an important design requirement. The prototype design must create an environment in which undistorted signals can perform designated tasks. Decoupling, matching impedance, controlling ground bounce, and limiting cross-talk, ringing, and noise margins are essential design elements. Both the board connection cables and the PCB contribute to these elements.

5.1 Controlling Ground Bounce

In certain cases, a number of outputs may be driven in the same direction within the same time frame (e.g., SSO such as data output from the FPGA). If these outputs share the same ground pin, the elevated current may result in a significant IR drop, which will noticeably rise or drop the potential of this ground pin along with its self-inductance. This problem may cause a malfunction or even damage to the circuit.

To avoid this problem, the prototype follows the Xilinx recommendations for SSO, and it has its bypass capacitors placed close to the ground pins. The SSO guidelines limit the number of simultaneously switching IOBs at a specific current level to a fixed number per power- and ground-pin pair. Per Xilinx, the pairing number for the VC2V3000 is 8 and for the BGA 20. Therefore,

SSO/bank = 20 x 8 = 160.

The prototype design satisfies Xilinx's recommendations. However, the footprint of the BG728 used in the UUT FPGA (see Appendix J), reveals that the "window" in the center of the footprint, which is used in lower-density FPGAs, is not present. Instead, a solid pattern of BGA balls is revealed. Because distinct physical separation between the banks is no longer possible, some of the Xilinx recommendations may not be as effective as for lower-density FPGAs. Xilinx recommendations regarding placement of power supply bypass capacitors cannot be followed for the same reason.

The user may enhance SSO criteria and overall performance by employing the suggestions in Section 11.

The design gives the user a choice of multiple clocks. Working in conjunction with the DCM feature of Virtex-II FPGA, the software can move some of the IOB's switching by –2, –1, +1 and +2 ns, divide or multiply the clock, or alternate oscillators in a way that spreads out the switching currents and thus does not add to ground bounce.

As a result, the combined transient currents from the switching will keep the ground bounce to less than ±100 mV (200 mV peak to peak), allowing the FPGA to provide the best performance and the lowest jitter. To further reduce ground bounce, 0603-type, 0.1-µF surface-mount bypass capacitors were placed for each power- and ground-pin pair.

5.2 LIMITING RINGING AND OVERSHOOTING OF THE SIGNAL

When the length of the conductive media approaches the magnitude of the wavelength of the highest frequency of interest, the echo from the receiver end may arrive at the transmitter after the end of the transition, resulting in ringing. Then the trace must be analyzed as a transmission line.

In this case, the driver sees the trace not as a lumped capacitance, but rather as a pure resistance of Z0. The signal transition then travels along the trace. At any trace-impedance discontinuity, all or part of the signal is reflected back to the origin. If the far end is resistively terminated with R = Z0, then there is no reflection. If, however, the end is open, or loaded with only a LVTTL input, then the transition doubles its amplitude, and this new wave travels back to the driver, where it may be reflected again, resulting in the familiar ringing. Such ringing has a serious effect on signal integrity, reduces noise margins, and can lead to a malfunction, especially if an asynchronous signal or a clock signal crosses the input threshold voltage unpredictably.

Two alternate ways to avoid reflections and ensure signal integrity are parallel termination and series termination. Although parallel termination eliminates reflections, series termination relies on the reflection from the far end to achieve a full-amplitude signal. For series termination, the driver impedance is adjusted to equal Z0, thus driving a half-amplitude signal onto the transmission line. At the unterminated far end, the reflection creates a full-amplitude signal, which then travels back to the driver where it gets absorbed, since the output impedance equals Z0. An example of series termination is given in Figure 19

For an outer-layer trace (air on one side), the propagation delay is 140 ps/inch, or 55 ps/cm. For an inner-layer trace (FR4 with E = 4.5 on both sides), the propagation delay is 180 ps/inch, or 70 ps/cm.

Series termination dissipates no direct-current (DC) power, but the half-amplitude round-trip delay signal means that there must be no additional loads along the line. Series termination is ideal for single-source–single-destination interconnects such as the one used in our design.

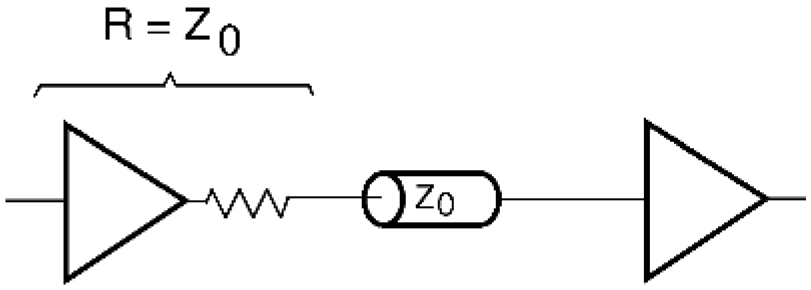


Figure 19: Series termination.

The Virtex-II FPGAs offer digitally controlled output impedance drivers and digitally controlled input (DCI) termination, thus eliminating the need for any external termination resistors. This feature is extremely valuable with high pin-count, high-density packages.

DCI operates independently on each I/O bank. In the prototype design, all I/Os of the UUT FPGA are divided into two halves. One half is configured as output (source) and terminated with 1% 50-ohm resistors (Figure 20). The other half is configured as inputs. When a DCI I/O standard is used in a particular I/O bank, external reference resistors must be connected to two dual-function pins on the bank. To reduce prototyping costs, the driver board utilizes only one bank of FPGAs as source and another one as receiver. However, the usage of I/Os on this board can be expanded if needed. All I/Os are connected to 50-ohm high-density connectors.

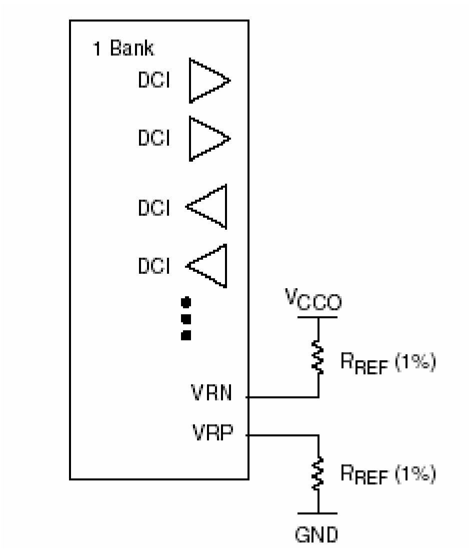


Figure 20: DCI in a Virtex-II bank

5.3 LOCATING CLOCKS TO MINIMIZE NOISE

Clock signals require special attention for two reasons. First, it is critical that their timing not be marginalized by noise. This can lead to false clocking of data. Second, because clock signals often run at a higher frequency than data, they can be more troublesome as noise sources.

From this consideration, all oscillators used in the design are placed within ½" from the input line. Optional clock lines routed from one board to another are shielded and placed as far from the lines as possible.

5.4 OTHER GOOD DESIGN PRACTICES

Unused I/O pins may be employed as virtual ground or supply pins. They should be programmed to drive a "1" or a "0" at the highest current drive strength and whenever possible should be tied to the PCB power or ground. These pins will function as additional power and ground pins, keeping the ground and power bounce under control.

Another technique used in FPGAs is to control the output slew rates. If delay is not an issue, the operator may use slow-output attributes to reduce cross talk and bounce.

6 Printed Circuit Boards

The PCB performs a critical function by providing stable power to the components and by maintaining signal integrity between devices. Three PCBs were designed for the prototype:

- Driver board
- UUT board
- Two types of interface boards for high-density impedance matching connectors and for the 50-ohm coaxial cable

See Section 7 for a discussion of the power supply board. The issue of V_{CC} decoupling is discussed in Section 3.

The input structure of the FPGA primarily represents an open circuit. As such, at high frequencies, the model is the PCB trace impedance, with a small capacitance at the end. The output structure of the FPGA represents low impedance, close to 25 ohms at high frequencies. With such low output impedance, large capacitive loads or long PCB traces may be driven. If the PCB trace is not controlled, the lack of control may lead to serious performance problems.

The same philosophy is used in the design the driver board and the UUT board: 0.1-µF NPO capacitors located close to V_{CC} can supply ~1A for 2 nsec with 20 mV voltage drop:

1A x 2 nsec = 2 nanocoulomb= 0.1 µF x 20 mV

The PCB's traces were analyzed as a transmission lines. The highest frequency of interest was not the clock frequency, but the highest frequency required by the pulses to define the rising and falling edges. This frequency was usually nine times the fundamental frequency.

For example, if the clock frequency is 33 MHz, the edges rise and fall in less than 3.3 nS. The frequencies of interest may approach 300 MHz. The wavelength of that frequency on the PCB is approximately half of what it is in the air, or about 50 cm. The length of the trace on the PCB trace becomes an important factor when it is around 1/8 of the wavelength of the highest frequency of interest. One-eighth of that frequency is about 6.25 cm. Traces less than 6.25 cm are unlikely to create the problems associated with transmission lines at 33 MHz. The longest trace on any of two boards is about 3.0 cm. Therefore, signals up to 66 MHz should travel freely.

What controls the impedance of a PCB trace? The major elements are the width and the distance to the V_{CC} or ground plane. To enable proper operation at higher frequencies, the ratio of trace width to the distance to the ground or V_{CC} plane is kept at 2 (e.g., width = 4 mil, depth = 2 mil); thus making the voltage-to-current ratio or characteristic impedance 50 ohms. To match this impedance, traces were routed into 50-ohm high-density connectors.

Secondary to the geometry is the material the board is made of. All PCBs are made out of epoxy fiberglass. Its permeability is the same as air (fiberglass does not affect magnetic fields), and the permitivity is about 4.5 that of air (fiberglass does affect the electric field).

The corresponding stack-up and spacing between the layers for the printed circuit boards is given in Appendix K.

7 Detailed Description of the Power Supply

Power consumption in Xilinx FPGAs depends upon the number of internal logic transitions and is proportional to the operating clock frequency. As device size increases, so does power consumption. It is common for a large, high-speed design to require several amperes of current. Power supply requirements, including initial conditions, transient behavior, turn-on, and turn-off are also important. Bypassing or decoupling the power supplies at the device, in the context of the device's application, requires careful attention. All these aspects of the power supply must be considered in order to achieve successful designs.

The power consumed in a Xilinx device is highly dependent on the design. Accurate power estimation methods must be used to ensure that a system power supply meets the FPGA's requirements.

7.1 Power Requirements

LVTTL is selected as the single-ended I/O standard. From this, in addition to internal supply voltage V_{CCINT} = 1.5 V (to power up the core), V_{CCO} = 3.3 V is required to support I/Os. The same voltage is used for the auxiliary power supply (V_{CCAUX} = 3.3 V).

For the correct operation of the power-on-reset (POR) and configuration controls, VCCO_4 and VCCO_5 (IOB input voltage for banks 4 and 5) must be connected to 3.3V regardless whether these banks are used.

To calculate power consumption for XC18V04 PROMS used to configure the UUT FPGA, the Virtex-II FPGA power-estimator worksheet and other applicable specifications were used. As shown in Table 9, the total estimated power needed from the power supply to operate the UUT board is under 2 W. The spreadsheet takes a conservative approach by overestimating the required power. It assumes that when a slice is selected, all of the LUTs and flip-flops within this slice will be used.

Software estimates the power consumption based on the FPGA configuration file. By analyzing the bit stream, the software algorithm estimates the design's resource usage, toggle rates, I/O power, and many other factors. The formulas used for the calculations in the program are based on test-design measurements. Because more than one configuration file may be used to address specific test goals, the worst-case scenario was considered in the calculations. The user may have to reevaluate the data's precise value corresponding to a given bit stream and/or clock frequency.

Table 9: Power estimates for the Xilinx Virtex-II FPGA (XC2V3000) used in the UUT board. Calculations assume 25-MHz clock and approximately 85% device utilization.

Item	Description/Value
Target Package	BGA728
Total Estimated Design Power	1957 mW
Estimated Design V_{CCINT} with 1.5-V Power	1880 mW
Estimated Design V_{CCAUX} with 3.3-V Power	66 mW
Estimated Design V_{CCO} with 3.3-V Power	11 mW
Estimated Design V_{CCO} with 2.5-V Power	0 mW
Estimated Design V_{CCO} with 1.8-V Power	0 mW
Estimated Design V_{CCO} with 1.5-V Power	0 mW
Estimated Design V_{CCO} with 1.2-V Power	0 mW

Note: The power required to operate the driver board is under 3 W. Besides using the same type of FPGA as the UUT board, this board has many other logic elements and discrete components. The power calculation for the driver board is more complex and is not presented here.

Note: Another constraint was imposed by thermal planning aimed at keeping the UUT die temperature below maximum rated value during the first stage of the burn–in test. Xilinx specifies a maximum of 80°C for commercial-, 100°C for industrial-, and 125°C for military-grade FPGAs. (The temperature may easily be raised over 100°C—up to 175°C—if needed.) The issues of thermal planning are discussed in greater detail in Section 4.2.

7.2 POWER-UP TIMING AND POWER SEQUENCING REQUIREMENTS

The actual current consumed depends on the power-on ramp rate of the power supply, and the duration of the V_{CCINT} ramp will depend on the amount of current available from the power supply. (Ramp on is defined as 0.1 VDC in addition to the minimum-specified supply voltage). If a large amount of current is available, the V_{CCINT} ramp will be very fast. Because our power supply can provide 10A of sustained and 12A of surge currents, no limiting factors are expected from the power supply, thus allowing FPGA its “natural” behavior.

When the final voltage has been reached, this high current is no longer required. Likewise, if the available current is limited, the rise time will be lengthened. A current trip or current fold back should not inhibit the rise.

The V_{CCINT}, V_{CCAUX}, and V_{CCO} power supplies must ramp on no faster than 200 µs and no slower than 50 ms. If this ramping requirement is not met, the FPGA will not perform POR properly. To avoid POR problems, timing measurements were conducted during the power-up cycle, using a simulated load. The ramps were captured using a simulated load, as shown in Figure 21 and Figure 22. As shown in these figures, the voltage rise is monotonic.

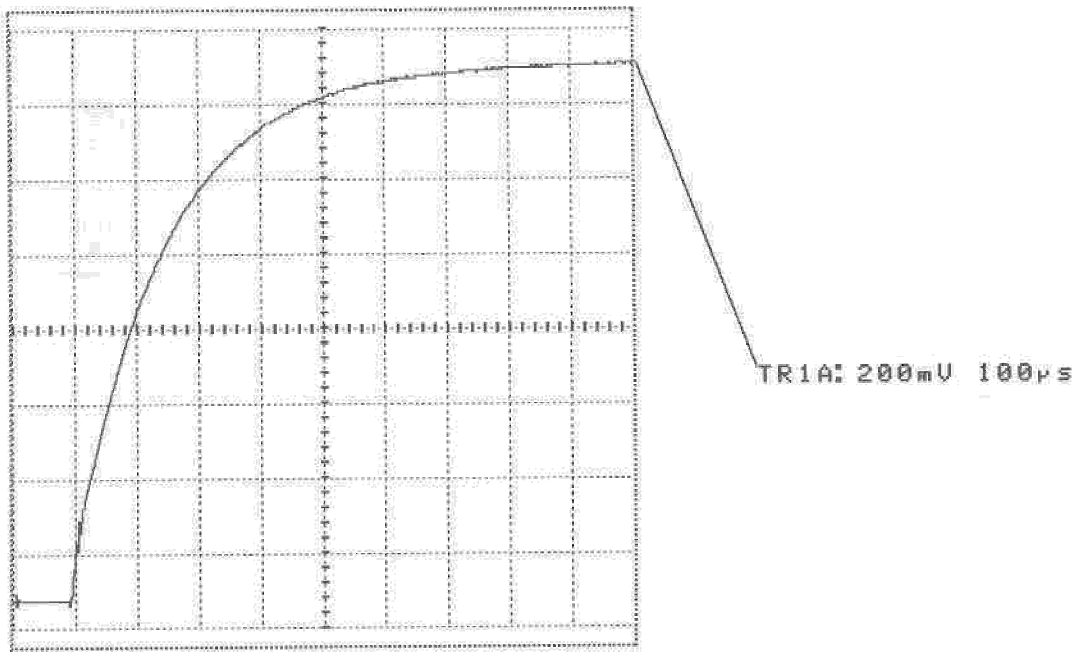


Figure 21: 1.5-core-voltage ramp

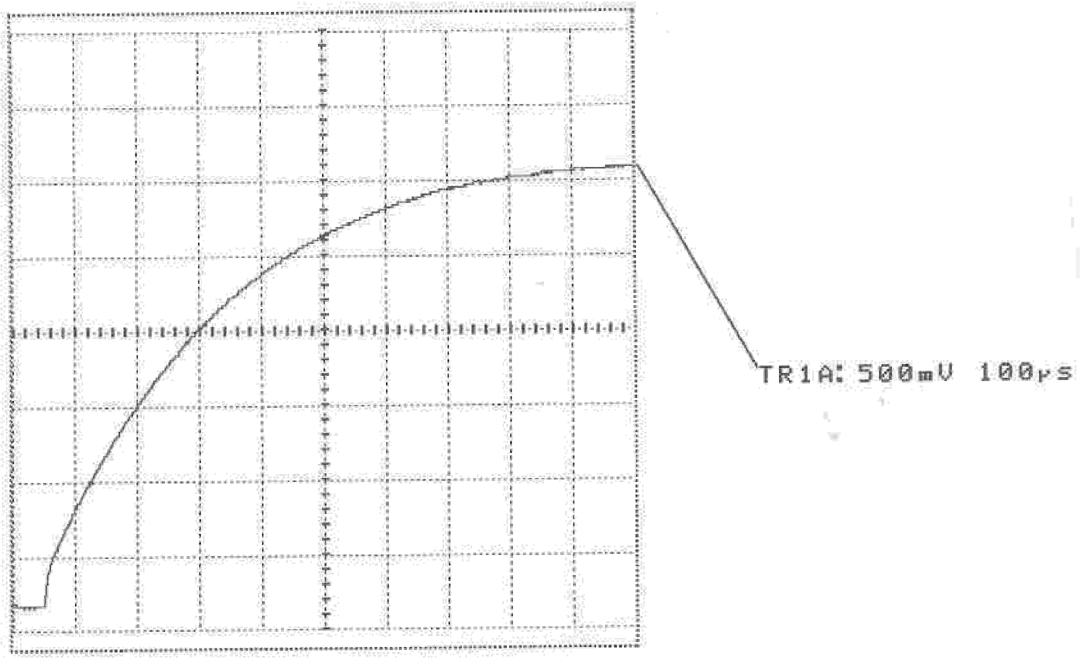


Figure 22: 3.3-auxiliary-supply-voltage V_{CCAUX} and output-driver-supply voltage V_{CCO} ramp.

Power supplies can be turned on in any sequence. If any V_{CCO} bank powers up before the V_{CCAUX}, then each bank draws up to 300 mA, worst case, until the

V_{CCAUX} powers on. This does not harm the device. If the current is limited to the minimum required value, the device powers on properly after all three supplies have passed through their POR threshold voltages.

The POR circuit of the Virtex-II FPGA is triggered when the following conditions are met:

$V_{CCINT} > 1.2$ V

$V_{CCAUX} > 2.5$ V

V_{CCO} (Bank 4) >1.5V

Once initialized and configured, the FPGA draws the power close to the one estimated by the calculator.

Note: The 300 mA is transient current (peak) and eventually disappears, even if V_{CCAUX} does not power up.

7.3 POWER SUPPLY SUMMARY AND SPECIFICATIONS

A standard off-the-shelf, linear-regulated power supply manufactured by Lambda is used in the prototype. To suit the project's needs, the power supply was modified to provide 3.3 and 1.5 voltages, using 5A National Semiconductor voltage regulators LM338 and few passive components. The power supply has its over-voltage protector set to 5.6 VDC. (Sensing voltage regulators would be more desirable but cannot be used because of the limited number of pins on the connector of the burn-in oven.)

The power supply offers 0.1% regulation at 50-W continuous output. Built by JPL, a digital readout allows the operator to check the output level for any of three outputs. The front panel has two separate sets of color-coded connectors. One set for the regular cable, another for the high temperature power cable.

In case of failure, the power supply has enough margin to make unusual current surges sustainable and therefore detectable.

A schematic of the power supply is provided in the Appendix L.

Note: If the voltage goes over the minimum operating voltage and then drops below it, incorrect power-on behavior may result. When the power-supply voltage falls below the absolute minimum-operating voltage when turned off, it should not rise immediately back to the nominal operating voltage without first discharging back down below 0.1 VDC. To ensure this condition is met, a resistor may be required to bleed off the charge on the filter and bypass capacitors.

8 FPGA Design Examples

The prototype includes three sample test designs ("design examples"). The VHDL code for these design samples, as well as for the test benches, can be found in the following appendixes:

- Appendix M: Driver FPGA VHDL Source Code
- Appendix N: Shifter-Test FPGA VHDL Source Code
- Appendix O: FIFO Test FPGA VHDL Source Code
- Appendix P: Calculator Test FPGA VHDL Source Code
- Appendix Q: Behavioral Test Bench and Results
- Appendix R: C/C++ Software Source Code

8.1 SHIFTER EXAMPLE

In the shifter example, the algorithm turns all the slice flip-flops of the UUT FPGA into shift registers. A small generic module contains eight "one-bit"-wide–by–16-bit shift registers consisting of LUTs. Another small generic module contains "eight-bit"-wide shift registers consisting of real slice flip-flops. Then, using the generics of the VHDL, the small eight-bit LUT shift registers are cascaded with the eight-bit flip-flop shift registers to form a long chain of shift registers occupying the entire chip.

These numbers can easily be changed since everything is coded in generic styles. All that is needed to produce a different design is to specify the total number of each shift register type. The incoming eight-bit data and the TXEN signal will be delayed together, to indicate the RXEN for the driver FPGA, to qualify the resulting data stream. Table 10 shows examples of how the FPGA resources are utilized, using this shifter configuration:

Table 10: Utilization of FPGA resources with the shifter configuration.

Characteristic	Value
Number of errors	0
Number of warnings	0
Number of slices	14,334 out of 14,336 (99%)
Number of slices containing unrelated logic	5,237 out of 14,334 (36%)
Number of slice flip-flops	18,241 out of 28,672 (63%)
Total number of input LUTs	9,000 out of 28,672 (31%)
Number used as shift registers	9,000
Number of bonded IOBs	28 out of 516 (5%)
IOB flip-flops	17
Number of GCLKs	1 out of 16 (6%)

Table 10: Utilization of FPGA resources with the shifter configuration.

Characteristic	Value
Total equivalent gate count for design	722,067
Additional JTAG gate count for IOBs	1,344

8.2 FIFO (TEST) EXAMPLE

The FIFO example demonstrates the test algorithm. In this example, the Xilinx CoreGenerator is used to produce three big FIFO buffers, each consisting of all the 96 block RAMs and one, 2-kbit basic element of distributed-RAM FIFO buffer. Each block-RAM FIFO buffer is 8 bits wide by 64k deep, with 512k total bits. The algorithm allows the user to set a threshold of how many data lines to buffer before the data are read out. This will dictate how many locations of each RAM are to be tested.

During testing, the data streams first flow in parallel though a configurable number of smaller FIFOs consisting only of the distributed RAM. The outputs of these FIFO buffers are multiplexed one at a time, to enter the bigger block-RAM FIFO buffers. The three block-RAMs receive the same data start rolling them in simultaneously. They then should start “spitting out” the same data.

Out of the three results, one is chosen at a time to be sent back to the driver FPGA. Two counters select results from all the parallel FIFO buffers. One large counter is used for the many distributed FIFO buffers. Starting from zero, it is incremented every time a value is read out of a distributed FIFO buffer. The counter ensures that, over the course of the test, each FIFO has a fair chance of being testing. A similar idea applies to the output selection of one of the three block-RAM FIFO buffers. A two-bit counter is incremented every time data is read. Table 11 shows one of the resource utilization summaries for this test configuration.

Table 11: Utilization of FPGA resources with the test configuration.

Characteristic	Summary
Number of errors	0
Number of warnings	0
Logic utilization: Number of slice flip-flops	1,885 out of 28,672 (6%)
Logic utilization: Number of 4 input LUTs	7,320 out of 28,672 (25%)
Logic distribution: Number of occupied slices	8,035 out of 14,336 (56%)
Logic distribution: Number of slices containing only related logic	8,035 out of 8,035 (100%)
Number of slices containing unrelated logic	0 out of 8,035 (0%)
Total number of four-input LUTs	15,694 out of 28,672 (54%)
Number used as logic	7,320

Table 11: Utilization of FPGA resources with the test configuration.

Characteristic	Summary
Number of errors	0
Number used as a route-through	182
Number used as shift registers	8,192
Number of bonded IOBs	31 out of 516 (6%)
IOB flip-flops	21
Number of block RAMs	96 out of 96 (100%)
Number of GCLKs	1 out of 16 (6%)
Total equivalent gate count for design	6,894,037
Additional JTAG gate count for IOBs	1,488

8.3 CRC (Calculator) Example

In the CRC (calculator) example, the LUTs of the Virtex-II FPGA are configured into four-input function generators. Each slice has built-in arithmetic units such as XOR gates and carry chains. This example supports two configurations:

- CRC engine, which mainly consists of shifting and modulo-2 additions
- Checksum engine, which mainly consists of adders with carry chains

A small generic block is written for eight-bit CRC or checksums. Then the entire chip is occupied with many of them working in parallel. As the data stream flows in from the driver FPGA, all the individual engines keep calculating results and storing them in their own eight-bit result registers. At the end of the test, an end of frame pulse coming from the TXCTRL will signal the end of tests for each engine and the results stop changing. Then a large MUX sends out the results one by one to the driver FPGA, to check them against its software-calculated results. If an error is detected, the error counter increases, to keep track of the results. Table 12 summarizes FPGA resource utilization in this calculator configuration.

Table 12: Utilization of FPGA resources with the calculator configuration.

Characteristic	Summary
Number of errors	0
Number of warnings	0
Logic utilization: Number of slice flip-flops	8,447 out of 28,672 (29%)
Logic utilization: Number of 4 input LUTs	19,681 out of 28,672 (68%)
Logic distribution: Number of occupied slices	10,733 out of 14,336 (74%)
Logic distribution: Number of slices containing only related logic	10,733 out of 10,733 (100%)
Logic Distribution: Number of slices containing unrelated logic[1]	0 out of 10,733 (0%)

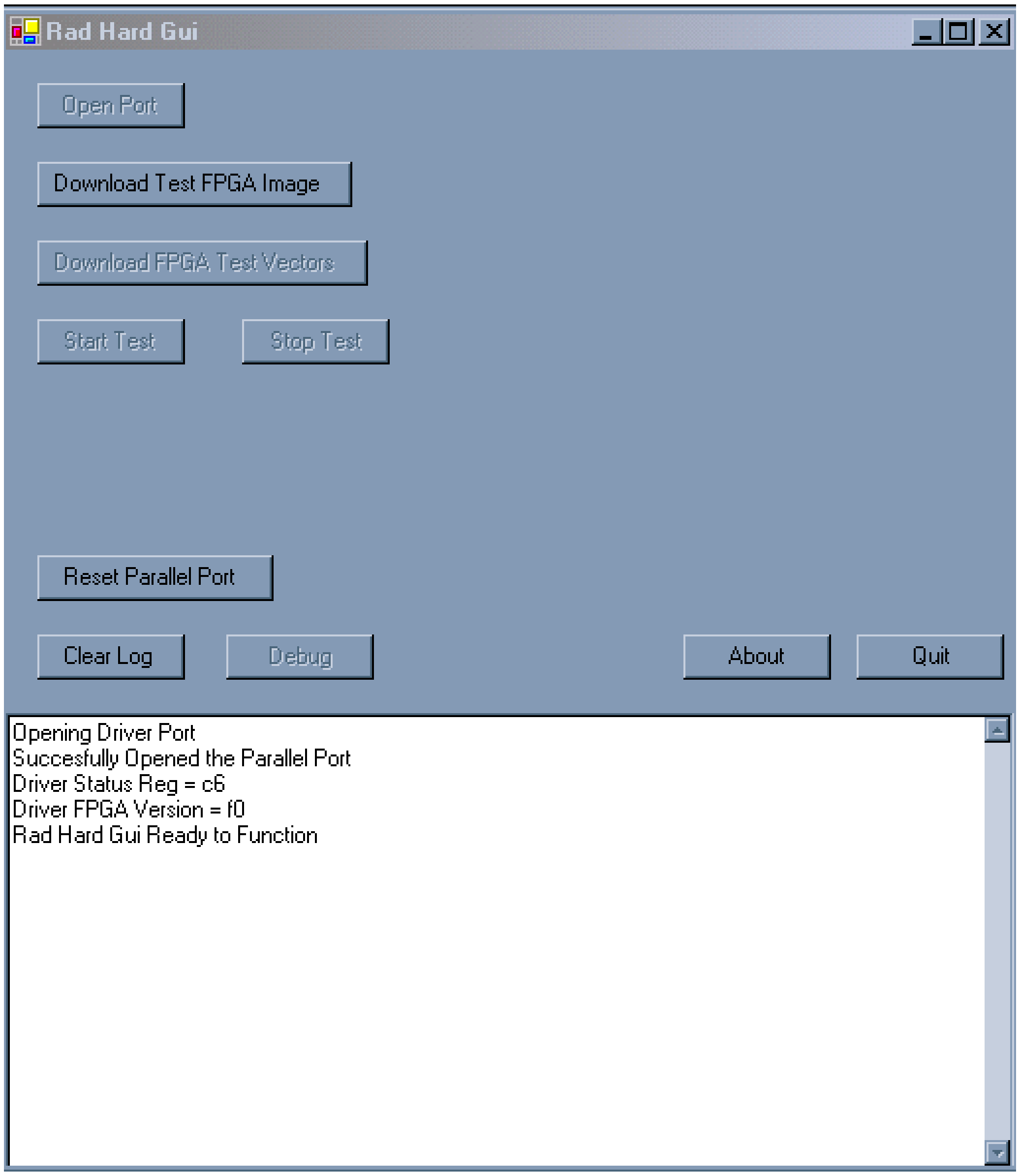
Table 12: Utilization of FPGA resources with the calculator configuration.

Characteristic	**Summary**
Total Number 4 input LUTs	19,686 out of 28,672 (68%)
Number used as logic	19,681
Number used as a route-throug:	5
Number of bonded IOB:	31 out of 516 (6%)
IOB flip-flops	21
Number of GCLKs	1 out of 16 (6%)
Total equivalent gate count for design	195,211
Additional JTAG gate count for IOBs	1,488

9 PC Software

The prototype includes a PC to control the platform and for long-term storage of data. The PC must use the Windows 2000 or XP operating system and have a parallel port through which it communicates with the driver FPGA.

The PC software was originally written with a command line interface. The prototype development was done using this interface. As the project progressed, the command-line interface was replaced with an easier-to-use GUI (Figure 23).



```
Opening Driver Port
Succesfully Opened the Parallel Port
Driver Status Reg = c6
Driver FPGA Version = f0
Rad Hard Gui Ready to Function
```

Figure 23: Graphical user interface (GUI) for the PC software.

The software was developed in Microsoft Visual Studio .Net V7.1. The development required V7.1 because this version introduced the concept of forms.

(V7.0 does not have this feature.) Although forms are not new technology (they have been used in Visual Basic for years), V7.1 provides a C++ interface to them, simplifying GUI development.

The software was designed to be modular. Each layer does a single task and relies on the other layers to handle other details, as shown in Figure 24. Each layer of the software is kept in a separate file. With the exception of the driver layer and the log layer, all other parts can be removed or replaced.

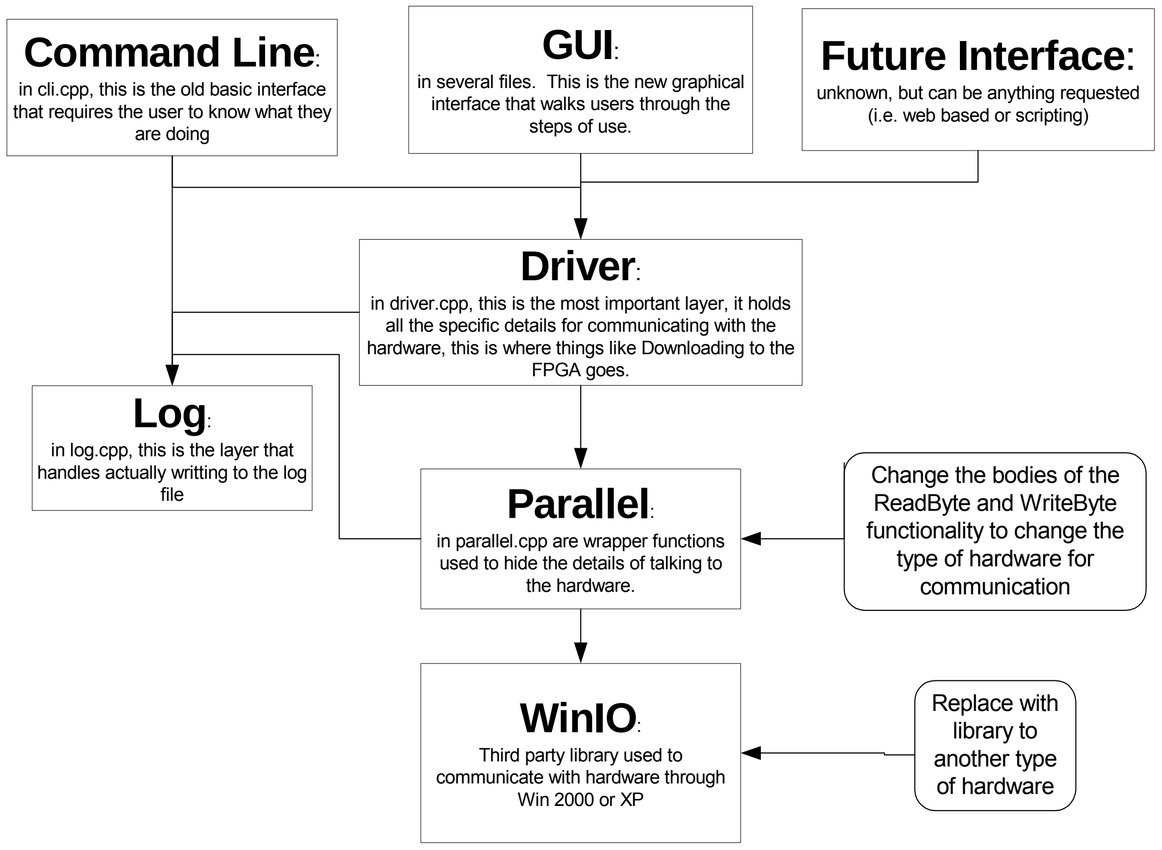

Command Line:
in cli.cpp, this is the old basic interface that requires the user to know what they are doing
GUI:
in several files. This is the new graphical interface that walks users through the steps of use.
Future Interface:
unknown, but can be anything requested (i.e. web based or scripting)
Driver:
in driver.cpp, this is the most important layer, it holds all the specific details for communicating with the hardware, this is where things like Downloading to the FPGA goes.
Log:
in log.cpp, this is the layer that handles actually writting to the log file
Parallel:
in parallel.cpp are wrapper functions used to hide the details of talking to the hardware.
Change the bodies of the ReadByte and WriteByte functionality to change the type of hardware for communication
WinIO:
Third party library used to communicate with hardware through Win 2000 or XP
Replace with library to another type of hardware

Figure 24: Software-design block diagram

The following sections describe the software modules in more detail.

9.1 WinIO

WinIO is a third-party library that can access the hardware on the PC. Since Windows NT, direct hardware access has not been allowed—the Microsoft Hardware Abstraction Layer handles it. To gain access to the actual hardware, both a kernel space and a user-space driver need to be installed. For the prototype, it was more efficient use a third-party tool than to write one from scratch.

9.2 Parallel Layer

The parallel layer handles the details of reading/writing to the parallel port. It is basically a wrapper around the WinIO function calls. The PAR_ReadByte and PAR_WriteByte functions handle the details of reading and writing to the parallel port. By rewriting these two functions to target different hardware, it should be possible to change from the parallel port to another communication mechanism, without rewriting any other parts of the software.

9.3 Driver

The driver layer has all the major details. This layer knows about specific registers on the driver FPGA. This is the one part of the software that is indispensable. This layer does **not** handle any display or memory but only handles the details of controlling the driver FPGA.

9.4 Log

The log layer handles the details of logging the information to the local hard drive. Since it needs to write a file, this layer also deals with the Windows OS. It writes a standard form to the file, consisting of the full date and time stamp and the specific message to be logged.

9.5 Command Line Interface

The original interface to the software was through the command line interface layer. This layer handled the display and memory handling details of the software. It was a simple loop continuously displaying a list of choices until the “quit” option was chosen. Each of the choices of action went to a new display along with the details for performing that action.

9.6 Graphical User Interface

The GUI is the new interface to the software. Compared to the command line interface, the GUI makes available a limited number of functions and enforces a strict flow of instructions. For example, it is no longer possible to start testing without downloading test vectors. To get into the debug features, a command line switch must be passed to the executable at start up. This adds an extra button on the main window, which opens a smaller window that can peek inside parts of the driver FPGA.

10 Test Results

```
YYYY/MM/DD hh:mm:ss     Action
2004/08/04 21:29:21     Opened Log File
2004/08/04 21:29:21     Command Line =
2004/08/04 21:29:21     Parallel Port has been opened
2004/08/04 21:29:21     GUI: Win Io Init = 1
2004/08/04 21:29:27     GUI: Opening File = C:\src\radhard\hex
files\Fifo.hex of size = 2623594 bytes
2004/08/04 21:29:30     Done pin went high and FPGA done downloading
2004/08/04 21:29:33     GUI: Opening File = C:\src\radhard\hex files\vector
of size = 95 bytes
2004/08/04 21:29:33     Downloaded Test Vectors
2004/08/04 21:29:37     GUI: Start Test
2004/08/04 21:29:37     DBG: Start Test Running = 0 --> 1
2004/08/04 21:30:27     GUI: Error Count = 1.000000
2004/08/04 21:30:27     GUI: Error Count = 2.000000
2004/08/04 21:30:27     GUI: Error Count = 3.000000
2004/08/04 21:30:27     GUI: Error Count = 4.000000
2004/08/04 21:30:27     GUI: Error Count = 5.000000
2004/08/04 21:30:27     GUI: Error Count = 6.000000
2004/08/04 21:30:27     GUI: Error Count = 7.000000
2004/08/04 21:30:27     GUI: Error Count = 8.000000
2004/08/04 21:30:27     GUI: Error Count = 9.000000
2004/08/04 21:30:27     GUI: Error Count = 10.000000
2004/08/04 21:30:27     GUI: Error Count = 11.000000
2004/08/04 21:30:27     GUI: Error Count = 12.000000
2004/08/04 21:30:27     GUI: Error Count = 13.000000
2004/08/04 21:30:27     GUI: Error Count = 14.000000
2004/08/04 21:30:27     GUI: Error Count = 15.000000
2004/08/04 21:30:27     GUI: Error Count = 16.000000
2004/08/04 21:30:27     GUI: Error Count = 17.000000
2004/08/04 21:30:27     GUI: Error Count = 18.000000
2004/08/04 21:30:27     GUI: Error Count = 19.000000
2004/08/04 21:30:27     GUI: Error Count = 20.000000
2004/08/04 21:30:27     GUI: Error Count = 21.000000
2004/08/04 21:30:27     GUI: Error Count = 22.000000
2004/08/04 21:30:27     GUI: Error Count = 23.000000
2004/08/04 21:30:27     GUI: Error Count = 24.000000
2004/08/04 21:30:27     GUI: Error Count = 25.000000
2004/08/04 21:30:27     GUI: Error Count = 26.000000
2004/08/04 21:30:27     GUI: Error Count = 27.000000
2004/08/04 21:30:27     GUI: Error Count = 28.000000
2004/08/04 21:30:27     GUI: Error Count = 29.000000
2004/08/04 21:30:27     GUI: Error Count = 30.000000
2004/08/04 21:30:27     GUI: Error Count = 31.000000
2004/08/04 21:30:27     GUI: Error Count = 32.000000
2004/08/04 21:30:27     GUI: Error Count = 33.000000
2004/08/04 21:30:27     GUI: Error Count = 34.000000
2004/08/04 21:30:27     GUI: Error Count = 35.000000
2004/08/04 21:30:27     GUI: Error Count = 36.000000
2004/08/04 21:30:27     GUI: Error Count = 37.000000
2004/08/04 21:30:27     GUI: Error Count = 38.000000
2004/08/04 21:30:27     GUI: Error Count = 39.000000
2004/08/04 21:30:27     GUI: Error Count = 40.000000
2004/08/04 21:30:27     GUI: Error Count = 41.000000
2004/08/04 21:30:27     GUI: Error Count = 42.000000
2004/08/04 21:30:27     GUI: Error Count = 43.000000
2004/08/04 21:30:27     GUI: Error Count = 44.000000
2004/08/04 21:30:27     GUI: Error Count = 45.000000
2004/08/04 21:30:27     GUI: Error Count = 46.000000
2004/08/04 21:30:28     GUI: Error Count = 47.000000
2004/08/04 21:30:28     GUI: Error Count = 48.000000
```

```
2004/08/04 21:30:28     GUI: Error Count = 49.000000
2004/08/04 21:30:28     GUI: Error Count = 50.000000
2004/08/04 21:30:28     GUI: Error Count = 51.000000
2004/08/04 21:30:28     GUI: Error Count = 52.000000
2004/08/04 21:30:28     GUI: Error Count = 53.000000
2004/08/04 21:30:28     GUI: Error Count = 54.000000
2004/08/04 21:30:28     GUI: Error Count = 55.000000
2004/08/04 21:30:28     GUI: Error Count = 56.000000
2004/08/04 21:30:28     GUI: Error Count = 57.000000
2004/08/04 21:30:28     GUI: Error Count = 58.000000
2004/08/04 21:30:28     GUI: Error Count = 59.000000
2004/08/04 21:30:28     GUI: Error Count = 60.000000
2004/08/04 21:30:28     GUI: Error Count = 61.000000
2004/08/04 21:30:28     GUI: Error Count = 62.000000
2004/08/04 21:30:28     GUI: Error Count = 63.000000
2004/08/04 21:30:28     GUI: Error Count = 64.000000
2004/08/04 21:30:28     GUI: Error Count = 65.000000
2004/08/04 21:30:28     GUI: Error Count = 66.000000
2004/08/04 21:30:28     GUI: Error Count = 67.000000
2004/08/04 21:30:28     GUI: Error Count = 68.000000
2004/08/04 21:30:28     GUI: Error Count = 69.000000
2004/08/04 21:30:28     GUI: Error Count = 70.000000
2004/08/04 21:30:28     GUI: Error Count = 71.000000
2004/08/04 21:30:28     GUI: Error Count = 72.000000
2004/08/04 21:30:28     GUI: Error Count = 73.000000
2004/08/04 21:30:28     GUI: Error Count = 74.000000
2004/08/04 21:30:28     GUI: Error Count = 75.000000
2004/08/04 21:30:28     GUI: Error Count = 76.000000
2004/08/04 21:30:28     GUI: Error Count = 77.000000
2004/08/04 21:30:28     GUI: Error Count = 78.000000
2004/08/04 21:30:28     GUI: Error Count = 79.000000
2004/08/04 21:30:28     GUI: Error Count = 80.000000
2004/08/04 21:30:28     GUI: Error Count = 81.000000
2004/08/04 21:30:28     GUI: Error Count = 82.000000
2004/08/04 21:30:28     GUI: Error Count = 83.000000
2004/08/04 21:30:28     GUI: Error Count = 84.000000
2004/08/04 21:30:28     GUI: Error Count = 85.000000
2004/08/04 21:30:28     GUI: Error Count = 86.000000
2004/08/04 21:30:28     GUI: Error Count = 87.000000
2004/08/04 21:30:28     GUI: Error Count = 88.000000
2004/08/04 21:30:28     GUI: Error Count = 89.000000
2004/08/04 21:30:28     GUI: Error Count = 90.000000
2004/08/04 21:30:28     GUI: Error Count = 91.000000
2004/08/04 21:30:28     GUI: Error Count = 92.000000
2004/08/04 21:30:28     GUI: Error Count = 93.000000
2004/08/04 21:30:28     GUI: Error Count = 94.000000
2004/08/04 21:30:28     GUI: Error Count = 95.000000
2004/08/04 21:30:29     GUI: Error Count = 96.000000
2004/08/04 21:30:29     GUI: Error Count = 97.000000
2004/08/04 21:30:29     GUI: Error Count = 98.000000
2004/08/04 21:30:29     GUI: Error Count = 99.000000
2004/08/04 21:30:29     GUI: Error Count = 100.000000
2004/08/04 21:30:29     GUI: Error Count = 101.000000
2004/08/04 21:30:29     GUI: Error Count = 102.000000
2004/08/04 21:30:29     GUI: Error Count = 103.000000
2004/08/04 21:30:29     GUI: Error Count = 104.000000
2004/08/04 21:30:29     GUI: Error Count = 105.000000
2004/08/04 21:30:29     GUI: Error Count = 106.000000
2004/08/04 21:30:29     GUI: Error Count = 107.000000
2004/08/04 21:30:29     GUI: Error Count = 108.000000
2004/08/04 21:30:29     GUI: Error Count = 109.000000
2004/08/04 21:30:29     GUI: Error Count = 110.000000
2004/08/04 21:30:29     GUI: Error Count = 111.000000
2004/08/04 21:30:29     GUI: Error Count = 112.000000
```

```
2004/08/04 21:30:29     GUI: Error Count = 113.000000
2004/08/04 21:30:29     GUI: Error Count = 114.000000
2004/08/04 21:30:29     GUI: Error Count = 115.000000
2004/08/04 21:30:29     GUI: Error Count = 116.000000
2004/08/04 21:30:29     GUI: Error Count = 117.000000
2004/08/04 21:30:29     GUI: Error Count = 118.000000
2004/08/04 21:30:29     GUI: Error Count = 119.000000
2004/08/04 21:30:29     GUI: Error Count = 120.000000
2004/08/04 21:30:29     GUI: Error Count = 121.000000
2004/08/04 21:30:29     GUI: Error Count = 122.000000
2004/08/04 21:30:29     GUI: Error Count = 123.000000
2004/08/04 21:30:29     GUI: Error Count = 124.000000
2004/08/04 21:30:29     GUI: Error Count = 125.000000
2004/08/04 21:30:29     GUI: Error Count = 126.000000
2004/08/04 21:30:29     GUI: Error Count = 127.000000
2004/08/04 21:30:29     GUI: Error Count = 128.000000
2004/08/04 21:30:29     GUI: Error Count = 129.000000
2004/08/04 21:30:29     GUI: Error Count = 130.000000
2004/08/04 21:30:29     GUI: Error Count = 131.000000
2004/08/04 21:30:29     GUI: Error Count = 132.000000
2004/08/04 21:30:29     GUI: Error Count = 133.000000
2004/08/04 21:30:29     GUI: Error Count = 134.000000
2004/08/04 21:30:29     GUI: Error Count = 135.000000
2004/08/04 21:30:29     GUI: Error Count = 136.000000
2004/08/04 21:30:29     GUI: Error Count = 137.000000
2004/08/04 21:30:29     GUI: Error Count = 138.000000
2004/08/04 21:30:29     GUI: Error Count = 139.000000
2004/08/04 21:30:29     GUI: Error Count = 140.000000
2004/08/04 21:30:29     GUI: Error Count = 141.000000
2004/08/04 21:30:29     GUI: Error Count = 142.000000
2004/08/04 21:30:29     GUI: Error Count = 143.000000
2004/08/04 21:30:29     GUI: Error Count = 144.000000
2004/08/04 21:30:30     GUI: Error Count = 145.000000
2004/08/04 21:30:30     GUI: Error Count = 146.000000
2004/08/04 21:30:30     GUI: Error Count = 147.000000
2004/08/04 21:30:30     GUI: Error Count = 148.000000
2004/08/04 21:30:30     GUI: Error Count = 149.000000
2004/08/04 21:30:30     GUI: Error Count = 150.000000
2004/08/04 21:30:30     GUI: Stop Test
2004/08/04 21:30:30     DBG: Stop Test Running = 1 --> 0
2004/08/04 21:30:30     GUI: Final Test Error Count = 150.000000
2004/08/04 21:30:32     Closing Log File and Exiting Program
```

11 Future Enhancements

For a first attempt at a new technology, this project has done well. The prototype has demonstrated all of its intended purposes and its ability to meet demands. This does not mean that everything is accomplished. There are many ways to expand and improve this design:

- Move off of the parallel port to a faster bus
- Add to the driver FPGA external memory to support much-larger test vectors
- Add to the driver FPGA the ability to store results and compare them with expected results
- Add to the software the writing of data to the hard drive as it writes to the file, to prevent possible data loss
- More UUT FPGA design samples that target different resource utilizations
- Reliability enhancements to the cables and connections

References:

1. C. Carmichael et al, "SEU Mitigation Techniques for Virtex FPGAs in Space Applications", 1999 MAPLD Conf.

2. Ghazanfar Asadi and Mehdi B. Tahoori, "An Analytical Approach for Soft Error Rate Estimation of SRAM-Based FPGAs", Dept. of Electrical & Computer Engineering, Northeastern University, Boston, MA, 2003

3. Xilinx Vertex II ^{TM} data books and application notes