CHAPTER 7

HARDWARE TESTING

In the previous Chapters, the complete low level design aspects of the ATM switch are discussed. The functions and design descriptions of the different modules and their simulations are discussed in detail. This Chapter deals with the hardware implementation issues of the ATM switch.

7.1 HARDWARE DESCRIPTION OF THE ATM SWITCH

The functions of each module and the entire ATM switch design are realized using a Hardware Descriptive Language – HDL. Verilog HDL is used for this purpose. The functions of each of the modules are coded using Verilog HDL as a behavioral description. In some cases mixed coding is also followed. The selection of Verilog HDL for hardware description for this design is quite obvious.

1. Verilog is used for coding the functions of each of the sub modules of the ATM switch because of the ease of describing the behavior of any circuit.
2. The language is more flexible in terms of describing hardware.
3. The hierarchical modeling such as Top-down and bottom-up design methodologies are available for digital design.
4. There are four levels of abstraction such as Behavioral, Structural (data flow level), Gate level, and Switch level to represent the same module, which offers flexibility of using all these abstractions in Verilog.

The functions of all the fifteen modules in the design that were explained in the Chapter 5 have references to the hardware that is instantiated in each of the modules. A hierarchical design approach is adopted for the design.
As shown in the figure 5.3 the **ATM Top** houses all the modules in the design namely:

1. Segmentation module
2. ATM host Interface Module
3. Reassembly module
4. Switch top

**Switch Top** houses four modules:

1. Main Switch FSM
2. VPI Translation Table
3. Port0 top
4. Port1 top

**Port 0 top** houses the following modules:

1. Switch Incoming interface P 0
2. Header Validation FSM P 0
3. VPI Routing Table For Port 0
4. Main Memory FSM P 0
5. Main Memory Control P 0

Similarly **Port 1** houses the following modules

1. Switch Incoming Interface P1
2. Header Validation FSM P1
3. VPI Routing Table For Port 1
4. Main Memory FSM P1
5. Main Memory Control P1
Shaded: Processes In Design Flow; Unshaded: Level Of Design

Figure 7.1 Flow Chart for VLSI IC design
7.2 HARDWARE TESTING

The functions of the different modules are tested by simulation and are shown in Chapter 6. The simulation shows that a very high switching speed in the order of 8.5 Gbps could be obtained in this new architecture. The architecture attained by including all the functional modules inside a single chip is the novelty of the design. The simulation results show that the single chip architecture will work satisfactorily when implemented in hardware, without affecting the functions of any of the modules. The hardware testing of the design is therefore carried out. This Chapter discusses the issues related to the hardware testing of the ATM switch and the significant results of testing.

7.2.1 IC Technologies

The flowchart in Figure 7.1 shows the steps involved in ASIC design. There are three methods of implementing the ATM switch design in hardware. They are explained as follows. The three types of IC technologies are:

- Full-custom VLSI
- Semi-custom ASIC (gate array and standard cell)
- PLD (Programmable Logic Device)

a. Full-custom VLSI: All the layers of the device are optimized for an embedded system’s particular digital implementation. The steps involved in the design are: Placing transistors, Sizing transistors and routing wires. The benefits are excellent performance, small size and low power consumption. The major drawback is that it requires weeks to months for development.

b. Semi-custom: Lower layers of the device are fully or partially built. Designers are left with routing of wires and placing some blocks. This offers good performance, small size, less cost than a full-custom implementation. The major drawback is that it requires weeks to months for development.
7.2.2 Programmable Logic Devices (PLD)

A PLD is an integrated circuit chip that can be configured by the end user to realize different designs and they are re-programmable and re-usable. Smaller PLDs are programmed using special programming units. Others are programmed, in-system using JTAG. The ATM switch design is tested using a PLD. Figure 7.2 shows the categorization of PLDs.

![Figure 7.2 PLD categorization](image)

In a PLD, all layers already exist and the designers can purchase an IC and development board. The connections on the IC are either created or destroyed to implement desired functionality, that is, they are re-programmable. One such IC is Field Programmable Gate Array (FPGA). The benefits offered by the PLDs are very low cost and almost instant IC availability and rapid prototyping facility to verify the functionality of the design and therefore complements the design verification efforts. Table 7.1 gives the characteristics of PLDs.
a. Programmable Logic Array (PLA)

Programmable Logic Arrays are designed using AND, OR gates and best suited for implementing complex combinatorial logic and eliminates discrete gates in the design and they are re-programmable and provides flexibility. A simple programmable AND/OR array is shown in figure 7.3.

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Criteria</th>
<th>CPLD</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Architecture</td>
<td>PLA based More Combinational</td>
<td>LUTs-Register based Registers + RAM</td>
</tr>
<tr>
<td>2</td>
<td>Density</td>
<td>Low-to-medium 0.5-100K logic gates</td>
<td>Medium to high 100K to 10M system gates</td>
</tr>
<tr>
<td>3</td>
<td>Performance</td>
<td>Predictable timing &gt; 200 MHz</td>
<td>Application &amp; Utilization Dependent</td>
</tr>
<tr>
<td>4</td>
<td>Interconnect</td>
<td>“Crossbar”</td>
<td>“Segmented”</td>
</tr>
<tr>
<td>5</td>
<td>Application</td>
<td>Glue Logic</td>
<td>Hardware Emulation</td>
</tr>
</tbody>
</table>

b. Complex Programmable Logic Devices (CPLD)

CPLDs consists of a number of macro cells (PLA based architecture) typically in the range 32-512 and has a capacity up to 100K gates and is an ideal candidate for low-medium density designs. CPLDs are best suited for combinatorial high-speed designs up to about 200 MHz. They offer predictable performance because of constant delay between 2 Macro cells. The vendor provides one step
synthesis and mapping tool for design entry using Verilog/VHDL and the device can be programmed using JTAG. The programme is erasable and re-programmable in system. The range of device utility is typically in the range of 100 user I/Os.

c. Field Programmable Gate Array (FPGA)

FPGAs are preferred to CPLDs for implementation of complex high-density design for the following reasons:

1. FPGAs provide cost effective solution to quickly prototype a design.
2. Validating the design on the FPGA hardware minimizes the risk when entering into fabrication.
3. FPGA hardware emulation enables development of software much ahead of implementation.
4. Array of logic cells called Configurable Logic Blocks (CLB) connected via routing channels can be programmed.
5. The gate densities from 40K to 10M Gates make this ideal for high-density designs.
6. Supports huge number of I/O standards like TTL, CMOS, LVCMOS, PCI, LVDS, etc (E.g.: Xilinx Spartan II FPGAs support 19 different single-ended and 6 different differential standards)

7. FPGAs with user pin outs greater than 100 user I/Os are available

8. FPGAs have built-in components like RAMs, Multipliers, Clock, Delay Locked Loops, ALU, Content Addressable Memories (CAM), Processors, and Intellectual Property (IP) cores are available for the user apart from logic cells

9. FPGAs consist of array of Configurable Logic Blocks. Each CLB comprises of 4 Slices and is tied to a Switch Matrix to access the Global Routing Matrix. Each Slice contains 4-Input Look Up Tables (Function Generators) and is capable of implementing any Boolean function

10. FPGAs contain multiplexers, Carry Logic, Arithmetic Logic Gates and two Storage Elements (Flip-Flop/Latch)

11. Plenty of Fast 18x18 Multipliers (in some FPGAs) are available for DSP Applications

12. FPGAs have large amounts of 18K-Bit Select RAM all over the chip (Synchronous operation, configurable as Single or Dual ported, configurable in different sizes like 2Kx9, 16Kx1, 4Kx4, etc) are available

13. FPGA's have multiple Digital Clock Managers (DCM) available to deskew and phase align all the internal clocks with respect to the input clock and they offer flexible Frequency Synthesis using DCM's

14. Multiple Global Clock Buffers are available for Clock distribution in the chip.

The advantages of using FPGAs for testing the ATM Switch design

- With FPGAs it is possible for full functional verification before fabricating the chip.

- Re-Programmable and hence it is possible to fix and verify functions in the design stage.
Hardware Emulation of the chip is possible with FPGA. Figure 7.4 shows the hardware and software verification involved in testing the design.

Reusable for future enhancements in the design.

7.3 STEPS INVOLVED IN FPGA EMULATION

- FPGA identification.
- FPGA based circuit board with other debug accessories to verify the design in hardware.
- Synthesis Tool (Convert RTL design to the targeted FPGA gates).
- Place & Route Tool (mostly from the FPGA vendor) that will place and route the synthesized gate net list.
- FPGA programming tools (e.g.: JTAG/parallel port cable).

Figure 7.4 Hardware & Software Verification
7.3.1 Hardware Selection Issues

Prior to implementation as a chip, the single chip ATM switch is to be tested in a programmable logic device in order to analyze the real time hardware functionalities of the new design. There are a number of programmable logic devices available. Hence the selection of a PLD becomes an important issue for hardware testing of this design.

Basically, the selection of a PLD for a specific application is based on one or more of the following requirements.

1. Number of memory elements used in the design.
2. Approximate Gate Counts for the Design.
3. Complexity of the design.
4. Availability of the associated chips or boards for testing.
5. Nature of inputs and outputs.

1. The ATM switch that is designed has got 8 memory elements each of 32-bit size. According to the design specifications, all data that is written or read from any memory of the switch is 32-bit information per clock cycle (1ns). The design requires at least 8 RAMs of size of 32 bit each. The modules in which memories are instantiated are shown in table 7.6.

Other hardware components instantiated in the design are:
- Counters are instantiated in all the fifteen modules of the design
- Registers are instantiated in all the modules of the design
- There are a number of flags used in the design
- There are a number of 32-bit data busses used in the design.

2. The complexity of the design can be seen from the number of hardware elements that are instantiated. Based on the complexity of the design, an approximate
prediction of gate counts for implementing the design in a programmable matrix would be at least 1 to 1.5 lakhs.

The discussion stated above, does not permit use of PLA or CPLD for testing this complex design. An FPGA with high gate counts and with enormous other associated features, would be much suitable for implementation and testing of the ATM switch. Accordingly, the Xilinx Spartan II series (XC2S100tq144) FPGA that has enough features for implementing the design is selected. The following section describes the various required features, Architectural and functional descriptions of the device.

7.4 DESCRIPTION OF SPARTAN II FPGA ARRAY

Spartan II FPGA (XC2s100tq144)

The Spartan™ -II 2.5V Field-Programmable Gate Array gives high performance, abundant logic resources, and a rich feature set. The chip offers densities ranging from 15,000 to 100,000 system gates, as shown in Table 7.2. System performance is supported up to 200 MHz. Spartan-II FPGA devices deliver more gates, I/Os, and features. Features include 10 block RAM (40K bits), distributed RAM (to 38,400 bits), 2,700 Logic cells, 600 CLBs, 16 selectable I/O standards, 1,00,000 system gates (including logic and RAM) and four DLLs.

7.4.1 Features of XC2s100tq144 Spartan II FPGA

- Densities as high as 2,700 logic cells with up to 100,000 system gates.
- Unlimited re programmability.
- Select RAM hierarchical memory.
- 16 bits/LUT distributed RAM
- Configurable 40K bit block RAM
• Fast interfaces to external RAM.
• Fully PCI compliant.
• Low-power segmented routing architecture.
• Full read back ability for verification/operability.
• Dedicated carry logic for high-speed arithmetic.
• Dedicated multiplier support.
• Cascade chain for wide-input functions.
• Abundant registers/latches with enable, set, reset.
• Four dedicated DLLs for advanced clock control.
• Four primary low-skew global clock distribution nets.
• IEEE 1149.1 compatible boundary scans logic.
• Versatile I/O and packaging.
• Compatibility in common packages.
• 16 high-performance interface standards.
• Hot swap compact PCI friendly.
• Zero hold time simplifies system timing.

The device is fully supported by powerful Xilinx development system

• Foundation ISE Series: Fully integrated software
• Alliance Series: For use with third-party tools
• Fully automatic mapping, placement, and routing

7.4.2 Functional Description of Spartan II - XC2s100tq144

The architecture of Spartan II FPGA is shown in figure 7.5. The Spartan-II FPGA has a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), surrounded by a perimeter of programmable Input/Output Blocks (IOBs). There are four Delay-Locked Loops (DLLs), one at each corner of the die. Two columns of block RAM lie on opposite sides of the die, between the CLBs and the IOB columns. A powerful hierarchy of versatile routing channels interconnects
these functional elements. Loading configuration customizes Spartan-II FPGA data into internal static memory cells. The device offers unlimited reprogramming cycles. The stored values in these cells determine logic functions and interconnections implemented in the FPGA. The configuration data can be read from an external serial PROM using master serial mode, or written into the FPGA in slave serial, slave parallel, or boundaries scan modes. The device provides system clock rates up to 200 MHz. In addition to the conventional benefits of high-volume programmable logic solutions, Spartan-II FPGA also offer on-chip synchronous single-port and dual-port RAM (block and distributed form), DLL clock drivers, programmable set and reset on all flip-flops, and fast carry logic.

### Table 7.2 Device Specifications for Spartan II family

<table>
<thead>
<tr>
<th>Device</th>
<th>Logic Cells</th>
<th>System Gates (Logic and RAM)</th>
<th>CLB Array (R x C)</th>
<th>Total CLBs</th>
<th>Maximum Available User I/O(1)</th>
<th>Total Distributed RAM Bits</th>
<th>Total Block RAM Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2S15</td>
<td>432</td>
<td>15,000</td>
<td>8 x 12</td>
<td>96</td>
<td>6,144</td>
<td>15K</td>
<td></td>
</tr>
<tr>
<td>XC2S30</td>
<td>972</td>
<td>30,000</td>
<td>12 x 18</td>
<td>216</td>
<td>13,824</td>
<td>24K</td>
<td></td>
</tr>
<tr>
<td>XC2S50</td>
<td>1,726</td>
<td>50,000</td>
<td>16 x 24</td>
<td>384</td>
<td>24,576</td>
<td>32K</td>
<td></td>
</tr>
<tr>
<td>XC2S100</td>
<td>2,700</td>
<td>100,000</td>
<td>20 x 30</td>
<td>600</td>
<td>38,400</td>
<td>40K</td>
<td></td>
</tr>
<tr>
<td>XC2S150</td>
<td>3,666</td>
<td>150,000</td>
<td>24 x 36</td>
<td>864</td>
<td>55,296</td>
<td>48K</td>
<td></td>
</tr>
<tr>
<td>XC2S200</td>
<td>5,282</td>
<td>200,000</td>
<td>28 x 42</td>
<td>1,176</td>
<td>75,264</td>
<td>56K</td>
<td></td>
</tr>
</tbody>
</table>

#### 7.4.3 Architectural Description of Spartan-II Array

The Spartan-II user-programmable gate array, shown in figure 7.5 is composed of five major configurable elements:

- IOBs provide the interface between the package pins and the internal logic.
- CLBs provide the functional elements for constructing most logic
Figure 7.5 Basic Spartan II FPGA block Diagram

- Dedicated block RAM memories of 4096 bits each.
- Clock DLLs for clock-distribution delay compensation and clock domain control.
- Multi-level inter-connect structure.

As can be seen in figure 7.5 the CLBs form the central logic structure with easy access to all support and routing structures. The IOBs are located around...
all the logic and memory elements for easy and quick routing of signals on and off the chip. Values stored in static memory cells control all the configurable logic elements and interconnect resources. These values load into the memory cells on power-up, and can reload if necessary to change the function of the device. Each of these elements will be discussed in detail in the following sections.

a. Input/Output Block

The Spartan-II IOB, as seen in figure 7.6, features inputs and outputs that support a wide variety of I/O signaling standards. These high-speed inputs and outputs are capable of supporting various memory and bus interfaces. The three IOB registers function either as edge-triggered D-type flip-flops or as level-sensitive latches. Each IOB has a clock signal (CLK) shared by the three registers and independent Clock Enable (CE) signals for each register.

b. Configurable Logic Block (CLB)

The basic building block of the Spartan-II CLB is the logic cell (LC), which is shown in figure 7.7. An LC includes a 4-input function generator; carry logic, and storage element. Output from the function generator in each LC drives the CLB output and the D input of the flip-flop. Each CLB contains four LCs, organized in two similar slices; a single slice is shown in fig 7.7. In addition to the four basic LCs, the Spartan-II CLB contains logic that combines function generators to provide functions of five or six inputs.

c. Look-Up Tables

Spartan-II function generators are implemented as 4-input look-up tables (LUTs). In addition to operating as a function generator, each LUT can provide a 16 x 1-bit synchronous RAM. Furthermore, the two LUTs within a slice can be combined to create a 16 x 2-bit or 32 x 1-bit synchronous RAM, or a 16 x 1-bit dual-port synchronous RAM. The LUT can also provide a 16-bit shift register that is ideal for capturing high-speed or burst-mode data. This mode can also be used to store data in applications such as Digital Signal Processing.
d. Storage Elements

Storage elements in the Spartan-II slice can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D inputs can be driven either by function generators within the slice or directly from slice inputs, bypassing the function generators.
e. Block RAM

Spartan-II FPGA incorporates several large block RAM memories. This complement the distributed RAM Look-Up Tables (LUTs) that provide shallow memory structures implemented in CLBs. Block RAM memory blocks are organized in columns. The device contains two such columns, one along each vertical edge. These columns extend the full height of the chip. Each memory block
is four CLBs high, and consequently, a Spartan-II device with eight CLBs high will contain two memory blocks per column, and a total of four blocks. Each block RAM cell, as illustrated in figure 7.8, is a fully synchronous dual-ported 4096-bit RAM with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion. The block RAM also includes dedicated routing to provide an efficient interface with both CLBs and other block RAMs. Table 7.3 illustrates the block RAM present in Spartan II FPGA and table 7.4 gives the port aspect ratios of the block RAM.

### Table 7.3 Block RAM of Spartan II family

<table>
<thead>
<tr>
<th>Spartan-II Device</th>
<th># of Blocks</th>
<th>Total Block RAM Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2S15</td>
<td>4</td>
<td>16K</td>
</tr>
<tr>
<td>XC2S30</td>
<td>6</td>
<td>24K</td>
</tr>
<tr>
<td>XC2S50</td>
<td>8</td>
<td>32K</td>
</tr>
<tr>
<td>XC2S100</td>
<td>10</td>
<td>40K</td>
</tr>
<tr>
<td>XC2S150</td>
<td>12</td>
<td>48K</td>
</tr>
<tr>
<td>XC2S200</td>
<td>14</td>
<td>56K</td>
</tr>
</tbody>
</table>

![Diagram of dual port Block RAM](image)

Figure 7.8 Block diagram of dual port Block RAM
### Table 7.4 Block RAM port aspect ratios

<table>
<thead>
<tr>
<th>Width</th>
<th>Depth</th>
<th>ADDR Bus</th>
<th>Data Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4096</td>
<td>ADDR&lt;11:0&gt;</td>
<td>DATA&lt;0&gt;</td>
</tr>
<tr>
<td>2</td>
<td>2048</td>
<td>ADDR&lt;10:0&gt;</td>
<td>DATA&lt;1:0&gt;</td>
</tr>
<tr>
<td>4</td>
<td>1024</td>
<td>ADDR&lt;9:0&gt;</td>
<td>DATA&lt;3:0&gt;</td>
</tr>
<tr>
<td>8</td>
<td>512</td>
<td>ADDR&lt;8:0&gt;</td>
<td>DATA&lt;7:0&gt;</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>ADDR&lt;7:0&gt;</td>
<td>DATA&lt;15:0&gt;</td>
</tr>
</tbody>
</table>

### f. Programmable Routing Matrix

It is the longest delay path that limits the speed of any worst-case design. Consequently, the Spartan-II FPGA routing architecture and its place-and-route software were defined in a single optimization process. This joint optimization minimizes long-path delays, and yields the best system performance. The joint optimization also reduces design compilation times because the architecture is software-friendly. Design cycles are correspondingly reduced due to shorter design iteration times.

### g. Local Routing

The local routing resources, as shown in figure 7.9, provide the following three types of connections:

- Interconnections among the LUTs, flip-flops, and General Routing Matrix (GRM)
- Internal CLB feedback paths that provide high-speed connections to LUTs within the same CLB, chaining them together with minimal routing delay
- Direct paths that provide high-speed connections between horizontally adjacent CLBs, eliminating the delay of the GRM
h. Clock Distribution

The FPGA provides high-speed, low-skew clock distribution through the primary global routing resources described above. Four global buffers are provided, two at the top center of the device and two at the bottom center. These drive the four primary global nets that in turn drive any clock pin. Four dedicated clock pads are provided, one adjacent to each of the global buffers. The input to the global buffer is selected either from these pads or from signals in the general purpose routing.

i. Delay-Locked Loop (DLL)

Associated with each global clock input buffer is a fully digital Delay-Locked Loop (DLL) that can eliminate skew between the clock input pad and internal clock-input pins throughout the device. Each DLL can drive two global clock networks. The DLL monitors the input clock and the distributed clock, and automatically adjusts a clock delay element. Additional delay is introduced such that clock edges reach internal flip-flops exactly one clock period after they arrive at the input. This closed-loop system effectively eliminates clock-distribution delay by
ensuring that clock edges arrive at internal flip-flops in synchronism with clock edges arriving at the input. In addition to eliminating clock-distribution delay, the DLL provides advanced control of multiple clock domains. The DLL provides four quadrature phases of the source clock which can be used to double the clock, or divide the clock by 1.5, 2, 2.5, 3, 4, 5, 8, or 16. It has six outputs. In order to guarantee that the system clock is operating correctly prior to the FPGA starting up after configuration, the DLL can delay the completion of the configuration process until after it has achieved lock.

7.5 FPGA BASED CIRCUIT BOARD WITH OTHER DEBUG ACCESSORIES

7.5.1 Selection of the Prototyping board

After selecting a suitable FPGA for testing the design, the next step is to select an appropriate prototyping board. This is more important due to the fact that the testing can be done only if appropriate prototyping board has all associated chips. The selection is usually done based on the following criteria.

b. Complexity of the design.
c. Testing method used.
d. Software for downloading the design.
e. Interfacing units.

a. **Nature of Inputs and Outputs:** The nature of inputs to the ATM switching processor is stream of data bits (for example 448 bytes) and the output of the ATM switch will be eight ATM cells carrying the output address according to the design specifications, each 56 bytes (3 bytes are dummy zeros that can be used for advanced flow control). Since the design requires stream of data to be forced into the ATM switch for testing, on-off switches cannot be used for giving inputs to the ATM switch. Similarly, the outputs are also stream of
bits of 53 byte data, each, which cannot be viewed using any output indicator such as LEDs. Even when Logic analyzer is used there must be enough number of channels for verification of the outputs. Therefore, for testing the design, it was decided that the input stream of bits be stored in an external RAM and the same is forced into the design, and the output from the ATM switch be again stored in the same RAM in a different location.

b. **Complex designs** require more number of gates in FPGA and other accessories.

c. **Testing Method:** A test control block is designed for this purpose.

d. **Software for downloading:** The downloading of the ATM switch and the other units of the test control block is done using software.

e. **Interfacing units:** The interfacing units are necessary for communication of data between the computer and the FPGA.

### 7.5.2 Layout of the FPGA Prototyping Board

The figure 7.10 shows the layout of the prototyping board used for fusing the design into FPGA. The board has the following modules.

- a. 16 MB SDRAM (8M x 16)
- b. Xilinx Xc2s100 Spartan II FPGA
- c. 256KB flash memory
- d. CPLD Xilinx XC9572
- e. Parallel port interface
- f. 100 MHz Programmable Oscillator
- g. Seven segment display and four DIP switches
- h. PC loaded with bit stream loading software
- i. Push Button Switches
This section describes the various accessories of the prototyping board and shows how the I/O of the FPGA and CPLD are connected to the rest of the circuitry. The figure 7.11 shows the photograph of the prototyping board.

a. **Programmable logic**
   - XC2S100 Spartan-II FPGA
   - XC9572XL CPLD.

The Prototyping Board contains two programmable logic chips:
1. A 100-Kgate Xilinx XC2S100 Spartan-II FPGA in a 144-pin QFP package. The FPGA is the main repository of programmable logic on the board.

2. A Xilinx XC9572XL CPLD that is used to manage the configuration of the FPGA via the parallel port. The CPLD also controls the programming of the Flash RAM.

Figure 7.11 Prototyping board

b. 100 MHz Programmable Oscillator

A Dallas DS1075 programmable oscillator provides a clock signal to both the FPGA and the CPLD. The DS1075 has a maximum frequency of 100 MHz that is divided to provide frequencies of 100 MHz, 50 MHz, 33.3 MHz, 25 MHz, ..., 48.7 KHz. The clock signal is connected to a dedicated clock input of the CPLD. The CPLD passes the clock signal on to the FPGA. This allows the CPLD to control the clock source for the FPGA. The divisor is stored in EEPROM in the DS1075 so it
will be restored whenever power is applied to the board. The clock signal enters a dedicated clock input of the CPLD. Then the CPLD can output a clock signal to a dedicated clock input of the FPGA. The circuit is shown in figure 7.12.

![Programmable clock Circuit diagram](image)

**Figure 7.12 Programmable clock Circuit**

c. 16 MB SDRAM

A Hynix HY57V281620AT-H SDRAM with 16 Mbytes of storage (8M x 16) is connected to the FPGA as shown in figure 7.13. The clock signal to the SDRAM is also re-routed back to a dedicated clock input of the FPGA. This makes it easy to synchronize the internal operations of the FPGA with the SDRAM operations.

d. Seven-Segment Display

The board has a 7-segment LED digit for use by the FPGA or the CPLD. The segments of this LED are active-high meaning that a segment will glow when a logic-high is applied to it. The LED shares the same pins as the eight bits of the Flash RAM data bus.
e. 256 Kbytes Flash RAM

An Atmel AT49F002 Flash RAM with 256 Kbytes of storage (256K × 8) is connected to both the FPGA and CPLD as shown in figure 7.14. The CPLD and FPGA both have access to the Flash RAM. Typically, the CPLD will program the Flash with data passed through the parallel port. If the data is an FPGA configuration bit stream, then the CPLD can be configured to program the FPGA with the bit stream from Flash whenever the board is powered up. After power-up, the FPGA can read and/or write the flash. (Of course, the CPLD and FPGA have to be programmed such that they do not conflict if both are trying to access the Flash.) The Flash can be
disabled by raising the CE pin to logic 1 in which case the I/O lines connected to the Flash can be used for general-purpose communication between the FPGA and the

---

**Figure 7.14 Flash RAM connection with CPLD and FPGA**

---
f. Push button Switches

The board has a single pushbutton that shares the FPGA pin connected to the data line of the PS/2 port. The pushbutton applies a low level to the FPGA pin when pressed and a resistor pulls the pin to a high level when the pushbutton is not pressed as shown in figure 7.15.

![Figure 7.15 Push Button Switch connection with FPGA](image)

---

g. VGA Monitor Interface

The FPGA can generate a video signal for display on a Video Graphics Adaptor (VGA) monitor interface, which is shown in figure 7.16. When the FPGA is generating VGA signals, the FPGA outputs two bits of red, green, and blue color information to a simple resistor-ladder DAC. The outputs of the DAC are sent to the RGB inputs of a VGA monitor along with the horizontal and vertical sync pulses (/HSYNC, /VSYNC) from the FPGA.

![Figure 7.16 VGA Interface with FPGA](image)
h. DIP Switch

The board has a bank of four Dipswitches accessible from the CPLD and FPGA. When closed or ON, each switch pulls the connected pin of the FPGA and CPLD to ground. Otherwise, the pin is pulled high through a resistor when the switch is open or OFF. When not being used, the Dipswitches should be left in the open or OFF configuration so the pins of the FPGA and CPLD are not tied to ground and can freely move between logic low and high levels. The Dipswitches also share the same pins as the uppermost four bits of the Flash RAM address bus. If the Flash RAM is programmed with several FPGA bit streams, then the Dipswitches can be used to select a particular bit streams, which will be loaded into the FPGA by the CPLD on power-up.

i. PS/2 Port

The board provides a PS/2-style interface (mini-DIN connector J4) to either a keyboard or a mouse. The FPGA receives two signals from the PS/2 interface: a clock signal and a serial data stream that is synchronized with the falling edges on the clock.

j. Parallel Port Interface

The parallel port is the main interface for communicating with the board is shown in figure 7.17. Control line C0 goes directly to the DS1075 oscillator and is used for setting the divisor as described previously, and status line S6 connects directly to the FPGA for use as a communication line from the FPGA back to the PC. The CPLD handles the fifteen remaining active lines of the interface to the parallel port.

Eleven of the active lines of the parallel port connect to General-purpose I/O pins on the CPLD. Three of the parallel port control lines, C1–C3, connect to the JTAG pins through which the CPLD is programmed. The C1 control line clocks
configuration data presented on the C3 line into the CPLD while the C2 signal steers the actions of the CPLD programming state machine.

Meanwhile, information from the CPLD returns to the PC through status line S7. The eight data lines, D0–D7, and the remaining three status lines, S3–S5, connect to general-purpose pins of the CPLD. The CPLD can be programmed to act as an interface between the FPGA and the parallel port. Schmitt-trigger inverters are inserted into the D1 line so it can carry a clean clock edge for use by any state machine programmed into the FPGA/ CPLD. The CPLD connects to the configuration pins of the Spartan-II FPGA so that it can pass configuration bit
The actual configuration data is presented on to the FPGA on the same 8-bit bus that connects the CPLD, Flash, seven-segment LED and FPGA. The CPLD also drives the configuration pins (CCLK, /PROGRAM, /CS, and /WR) of the FPGA that control the loading of a bit stream. The CPLD uses the M0 input of the FPGA to select either the slave-serial or master-select configuration mode (M1 and M2 are already hard-wired to VCC and GND, respectively).

The CPLD can monitor the status of the bit stream download through the /INIT, DONE, and BSY/DOUT pins of the FPGA. The CPLD also has access to the FPGA JTAG pins: TCK, TMS, TDI, TDO. The TMS, TDI, and TDO pins share the connections with the BSY/DOUT, /CS, and /WR pins. With these connections, the CPLD can be programmed with an interface that allows configuration of the Spartan-II FPGA through the Xilinx JTAG Programmer software utility.

The FPGA sends data back to the PC by driving logic levels onto pins 40, 29 and 28 which passes through the CPLD and onto the parallel port status lines S3, S4 and S5, respectively. Conversely, the PC sends data to the FPGA on parallel port data lines D0–D7 and the data passes through the CPLD and ends up on FPGA pins 50, 48, 42, 47, 65, 51, 58 and 43, respectively. The CPLD also drives the decimal point of the LED display to indicate when the FPGA is configured with a valid bit stream.

7.6 TESTING THE DESIGN IN FPGA

For verification of the design into FPGA four steps are followed. They are:

1. Creating test bench for testing the design
2. Synthesis and producing bit file
3. Downloading the design into FPGA
4. Testing the design
7.6.1 TEST CONTROL Block Design

The ATM switch design cannot be directly downloaded into FPGA for testing because of the nature of the inputs and outputs. The input to the ATM switch is a stream of bits of 144 bytes, which will be processed by the ATM switch and gives three ATM cells at the outputs. 144 bytes are taken as an example for testing. However, the ATM switch can process any incoming packets of greater size. So the input bit stream is stored in an external SDRAM. The idea involved in testing is that the stream of data from SDRAM should be forced to enter the ATM switch and the outputs from the switch are again stored in the same external SDRAM in a different location. In order to force the input from the RAM and to transfer the output to the SDRAM, a test control block is to be designed in Verilog as TESTCONTROL.V. The test control block should comprise a mechanism by which the inputs and outputs are processed by the ATM switch. The test control block is shown in figure 7.19. TESTCONTROL block has the following modules. The design of TESTCONTROL block is given in Appendix.

a. Clock Distributor (Divisor)

b. SDRAM controller (SDRAMCTL.V) consisting of
- I/O controller
- Read / Write controller

c. ATM TOP.V (Single chip ATM switch)

a. Clock distribution

The prototyping board has a programmable 100 MHz clock. The clock is programmed to 50 MHz and is given as input to the test control block. The ATM
switch that is designed can operate at the rate of 32 bits per clock cycle. Hence a
divisor divides the clock to 50/32 so that the entire test control block operates at the
same frequency.

b. SDRAM controller and R/W controller

The test control block has to communicate with the external SDRAM for
fetching the inputs and sending the outputs. For this purpose a SDRAM controller is
designed using Verilog HDL. This controller does the reading and writing operations
from and to the SDRAM. A Read / Write controller acts as an interface between the
SDRAM controller and the ATM switch design. Hence the R/W controller is
instantiated in the top module SDRAMCTL.V, a Verilog file and the entire test
control block is instantiated in the TESTCONTROL.V, which are the top most
modules in the hierarchy.

c. ATM TOP.V: This is the actual ATM switch design, which is to be tested. The
Reassembly module is not included in ATMTOP.V, the design that is fused to the
FPGA because the function of the switch could not be realized. The hierarchy is
given in figure 7.18.

Figure 7.18 Hierarchy of TESTCONTROL block
7.6.2 Testing methodology

1. The test control block shown in figure 7.19 is synthesized using Xilinx Synthesis tools and a corresponding bit file is generated. The bit stream is downloaded into the FPGA in the form of TESTCONTROL.bit file by a loading software tool as shown in figure 7.20.
2. The input data stream is downloaded into the 16 MB SDRAM by means of the downloading software provided by the prototyping board manufacturer.

3. The test control block is designed in such a way that the prototyping board when enabled the **SDRAM controller** reads the input file-containing stream of input data (448 bytes) through the R/W controller and transfers it to the **ATM TOP.V**. After the switching operations are over, the three output cells carrying the corresponding VPI address is stored in the same SDRAM through the R/W control block which is read by a PC through parallel port. A User Constraints File (UCF), SDRAM Controller, TESTCONTROL block design, is designed for this purpose and is given in Appendix 1.

Figure 7.20 Downloading the design
7.7 SYNTHESIS AND GENERATION OF BIT FILE

7.7.1 Design Flow

Xilinx Synthesis System – Overview

The Xilinx Synthesis system is a software tool, which has the design flow modules that are shown in table 7.5. The flow chart in figure 7.22 shows the detailed procedure involved in synthesizing and generating the bit stream for the design using Xilinx Synthesis technology (XST). The following section describes the method adopted for downloading the design into the target device.

<table>
<thead>
<tr>
<th>Design entry</th>
<th>Synthesis</th>
<th>Simulation</th>
<th>Implementation</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Project Navigator</td>
<td>Xilinx synthesis</td>
<td>1. HDL Bencher</td>
<td>1. Chip Viewer</td>
<td>1. IMPACT (Intelligent Multipurpose Programmable and Configuration Tool)</td>
</tr>
<tr>
<td>2. HDL Editor</td>
<td>Technology</td>
<td>2. Modelsim Simulator</td>
<td>2. X – Power</td>
<td>2. PROM file formatter</td>
</tr>
<tr>
<td>3. State CAD</td>
<td>(XST)</td>
<td>3. Timing Analyzer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Schematic Editor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Constraints Editor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Floor plan Editor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The design entry tools of the XST are used for entering the design modules in the form of Verilog HDL codes.
7.7.2 RTL description

The behavioral description of the functional modules is converted into Register Transfer Level (RTL) coding. The ATM switch is modeled using Verilog HDL using synthesizable constructs, specific for Xilinx Synthesis system. For example, blocking assignments are used for combinational logics and for variables that are assigned and used, all within an always statement. Non-blocking assignments are used for modeling sequential logics. The design consists of more sequential logics when compared to combinational logic. The synthesizable constructs for each module are given in Appendix.

7.7.3 SYNTHESIS

Synthesis is a process of creating gate level net list from a register transfer level model of a circuit described in Verilog HDL. That is, synthesis involves conversion of RTL Design (Verilog/VHDL) into gates present in the target FPGA library. This is shown in figure 7.21. This is done using a synthesis tool, which is software that is provided by the chip vendor. The Xilinx synthesis system used in this design as an intermediate step generates a net list that is comprised of register transfer level blocks such as flip-flops, registers, adders, and subtractors, arithmetic logic units and multiplexers, interconnected by wires. In such cases a module builder is used to build or acquire from a library of predefined components, each of the required RTL blocks in the user specified target technology (here, Xilinx FPGA).

![Figure 7.21 Synthesis Extractions](image-url)
Figure 7.22 Flow Chart for Synthesis and Testing the ATM switch (contd..)
Figure 7.22 Flow Chart for Synthesis and Testing the ATM switch

The synthesis system has a mapping mechanism or a construction mechanism that translates the Verilog HDL elements into their corresponding hardware elements using the library of the target device. The input files to the tool are synthesisable Verilog model of the TESTCONTROL.V, Constraints file, and the target device library (XC2s100 tq144). The output of the synthesis tool is gate level net list in the NCD file format. A Native Circuit Description file (NCD) is the one that represents a physical circuit description of the input design as applied to a specific device. Gate net list is a textual description of the circuit connectivity containing a list of connectors, a list of instances, and, for each instance, a list of the
signals connected to the instance terminals. The net list also contains attribute information.

7.7.4 Memory mapping to the target device

The Xilinx FPGA XC2s100 Spartan II has 10 block RAMs each 16 bit. But the design of the ATM switch is based on the writing or reading operations to take place in any memory module and data transfer throughout the chip as 32 bit per clock cycle. There are totally 8-memory modules instantiated in the design. They are shown in table 7.7. The UCF is given in Appendix.

Table 7.7 Modules in the design with Memory instantiation

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Module Name</th>
<th>Memory Type</th>
<th>No.</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Segmentation Module</td>
<td>RAM</td>
<td>1</td>
<td>65536 X 32</td>
</tr>
<tr>
<td></td>
<td>Switch Incoming Interface</td>
<td>RAM</td>
<td>2</td>
<td>32x28</td>
</tr>
<tr>
<td></td>
<td>Main Memory Control</td>
<td>RAM</td>
<td>2</td>
<td>32x140</td>
</tr>
<tr>
<td></td>
<td>VPI routing Table</td>
<td>RAM</td>
<td>2</td>
<td>2x12</td>
</tr>
<tr>
<td></td>
<td>VPI translation table</td>
<td>RAM</td>
<td>1</td>
<td>4x18</td>
</tr>
<tr>
<td></td>
<td><strong>TOTAL</strong></td>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Out of the 8 memory modules that are instantiated only 5 memory modules are mapped to the target device (FPGA) due to the following reasons.

a. Memory size of 65536 X 32 is instantiated in the Segmentation Module i.e. \((2^{16} \times 32)\) is a large memory space which cannot be accommodated in the PLD. Hence the memory size of this module is reduced to 256 x 32 \((2^8 \times 32)\). This memory is sufficient for storing 256 thirty-two bit data i.e. around 20 ATM cells. The switch is tested for 448 bytes of input. However when designed as an ASIC this memory capacity can be increased according to the requirement.
b. The VPI routing table, and VPI translation table has very small memory size. The two tables store VPI addresses for the corresponding output ports. The number of VPI addresses stored in the routing tables is 4 addresses of 12 bits each as shown in figure. Since they require a very small size memory and they simply store the VPI addresses they are replaced by parameter assignment statements in the modeling. This is done in order to accommodate the design in the Spartan II chip. However, for larger number of port, these memories store a lot of VPI addresses and will be big in size. In that case parameter assignment cannot be done and memories instantiated for these modules has to be mapped to the target device. Hence, effectively, the design contains only 5 memory modules of big size.

c. The target device (Xilinx Spartan II FPGA) has 10 block RAMs each of 16 bit size. Each block RAM cell, as illustrated in section 7.4.3, is a fully synchronous dual-ported 4096-bit RAM with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion. The Spartan-II block RAM also includes dedicated routing to provide an efficient interface with both CLBs and other block RAMs. But, the design requires 32 bit RAM for all the 5 modules. Hence two 16-bit blocks RAM of the FPGA are mapped for one design module. All the 10 block RAMs of the chip are configured and mapped in such a way that the five 32 bit memory modules of the design are mapped to the 10 16 bit block RAM of the chip. This is done by instantiating the modules in the User constraints file. The constraints editor of Xilinx synthesis system is used for this purpose.

7.7.5 Constraints editor

Constraints are instructions placed on symbols or nets in an FPGA schematic or textual entry file such as VHDL or Verilog. They indicate a number of things such as placement, implementation, naming, signal direction, and timing considerations. In the Xilinx development system, logical constraints are placed in a file called the UCF (User Constraints File). The Constraints Editor is a program that can be used to create and modify constraints.
Input files to the Constraints Editor: UCF (User Constraints File): The constraints created for the implementation of the design such as I/O pin locking and memory mapping etc. are written to this file. The UCF is an ASCII file specifying constraints on the logical design. These constraints affect the method by which the logical design is implemented in the target device. The TESTCONTRL.UCF is given in Appendix.

7.7.6 Translation

This tool of XST controls the Net list Translation Type property controls, on how NGD Build uses source files. The NGD build is a program that automatically invokes these programs as needed to convert the TESTCONTROL.NCD file that is generated after synthesis, to the required format for the Xilinx software tools for mapping it to the target device. The Translate process is run after the design files have been successfully analyzed. The Translate process will translate the design into gates and optimize it for the target architecture. The TESTCONTROL.NCD file generated by the Synthesis tool is converted into TESTCONTROL.NGD (Native Generic Data Base) by the NGD build programme. The output of translation is TESTCONTROL.NGD, which is a Native Generic Database file that describes a logical design reduced to Xilinx primitives.

Translate report

The report contains warning and error messages from the three translation processes: conversion of the EDIF/NCD or XNF style net list to the Xilinx NGD net list format, timing specification checks, and logical design rule checks.

The translation report lists the following:

- Missing or untranslatable hierarchical blocks
- Invalid or incomplete timing constraints
- Output contention, load less outputs, and source less inputs
7.7.7 Mapping

The process of assigning a design's logic elements to the specific physical elements that actually implement logic functions in a device. The MAP process is run after the design files have been successfully translated. The MAP process will map the logical design to the Xilinx FPGA (XC2s100). The input to the MAP process is TESTCONTROL.NGD file, which contains a logical description of the design in terms of both the hierarchical components used to develop the design and the lower level Xilinx primitives, and any number of NMC (macro library) files, each of which contains the definition of a physical macro. The Mapping process runs the technology-specific optimization of the Boolean structure that was created during the compile process.

MAP first performs a logical DRC (Design Rule Check) on the design in the NGD file. MAP then maps the logic to the components (logic cells, I/O cells, and other components) in the target Xilinx FPGA (XC2s100). The output design is a TESTCONTROL. NCD (Native Circuit Description) file - a physical representation of the design mapped to the components in the Xilinx FPGA and a Physical Constraints File (PCF). The NCD file can then be placed and routed.

a. Map Report

The report contains warning and error messages detailing logic optimization and problems in mapping logic to physical resources. The report lists the following information.

- Erroneously removed logic: Source less and leadless signals can cause a whole chain of logic to be removed. Each deleted element is listed with progressive indentation, so that the origins of removed logic sections are easily identifiable.
- Logic that has been added or expanded to optimize speed.
7.8 TESTING PROCEDURE

1. The bit stream is downloaded into the FPGA in the form of TESTCONTROL.bit FILE by a loading software tool.

2. The input data stream is also downloaded into the 16 MB SDRAM by means of the downloading software.

3. One Dipswitch is used for enabling the prototyping board.

4. Two Dipswitches are used for enabling the input ports Port 0 and Port 1.

5. The board is connected to the PC through the parallel port interface.

6. A power supply of 9 V is given to the board.

7. The board is tested using software. The software downloads a pre-designed bit file, which is downloaded into the FPGA and configures the FPGA and displays the result of the test.

8. The Dipswitch corresponding to enabling the prototyping board is enabled.

9. The TESTCONTROL.bit file that is generated by Xilinx Synthesis Technology is downloaded into the FPGA using the GUI based downloading software.

10. The input in the form of stream of bits is downloaded in the form of hex code, into the SDRAM by the downloading software.

11. The Dipswitch corresponding to Port 0 is enabled.

12. The port when enabled by a dipswitch, the SDRAM controller reads the input file-containing stream of input data bytes through the R/W controller and transfers it to the ATM TOP.V. After the switching operations are over, the output cells carrying the corresponding VPI
• The Design Summary section lists the number and percentage of used CLBs, IOBs, flip-flops, and latches. It also lists occurrences of architecturally specific resources like global buffers and boundary scan logic.

b. PAD report

• The report lists the design's pin out in three ways.
• Signals are referenced according to pad numbers.
• Pad numbers are referenced according to signal names.
• PCF file constraints are listed.

c. Post Map static timing report:

The Post-Map Static Timing Report process is used to view the pre place and route static timing report that will give a calculated worst-case timing for all signal paths in the design. It optionally includes a complete listing of all delays on each individual path in the design. It does not include insertion of stimulus vectors. The FPGA design must be mapped and can be partially or completely placed, routed, or both.

d. Asynchronous Delay report

The report lists all nets in the design and the delays of all loads on the net. This report shows the 20 worst net delays within the design. No worst delay report is observed in this design.

e. Back annotate pin report:

The Back annotate Pin Report process use the pin two UCF (pin2ucf) report. Each time the design is re-implemented, a file is created which contains the pin locations and logical pad names information. For FPGAs, pin locations and logical pad names are read from a placed NCD file (design_name.ncd), TESTCONTROL.NCD.
7.7.8 Floor planner

This process launches the Floor planner graphical user interface. The Floorplanner can be used to graphically place a design into a target Xilinx FPGA (XC2s100). When the Floor planner is launched, the design is run through Place and Route (PAR) and the mapped NCD file is loaded in the Floorplanner automatically. The Floor planner uses a hierarchy structure of lines and colors to distinguish the different hierarchical levels of the design. The Floorplanner provides information on design connectivity and resource requirements, target FPGA resource layout, and design mapping via location constraints. For a selected FPGA the logic can be placed into a floorplan either manually or automatically. The Floor planner is set in Auto mode for implementing the TESTCONTOL.V and the figure shows the floorplan view of the placed design. Floorplanning is an optional methodology that can be used to improve the performance of an automatically placed and routed design. Floorplanning is particularly useful on structured designs and data path logic. The Floorplanner helps in determining where to place logic in the floorplan for optimal results. The data paths can be exactly placed at the desired location on the die when manufactured as ASIC. The Floorplanner can be used prior to mapping, immediately after mapping, and following place and route. The Floorplanner can read in the following input files. The floor plan view of the implemented design is shown in chapter 8.

NGD: NGD Build generates this file. The NGD file represents the design prior to mapping to the target device resources. IOB placement can be performed with an NGD file alone.

NCD: Either MAP or PAR generates this file. The NCD file represents the design after mapping in terms of the physical resource usage of the design in the target device. This file may include placement and routing information.

FNF: The Floorplanner Net list File (FNF) is the Floorplanner's database. Its core function is to retain a record of all the (physical) constraints entered in the Floorplan window. If the FNF is generated using a placed NCD file, the placement information is also recorded in the FNF.
7.7.9. Place And Route (PAR)

The Place and Route process is run after the design has undergone the necessary translation to bring it into the NCD (Native Circuit Description) format. Place and Route (PAR) take the TESTCONTROL. NCD file, places and routes the design, and outputs an NCD file which is used by the bit stream generator which generates the bit stream corresponding to the TESTCONTROL.V.

a. Place And Route report:

The report contains information on device utilization, overall effort level of the placer, and the number of signals not completely routed. Placement and routing errors are also indicated in the report. The Map and Place and Route report for the design is given in Chapter 8.

b. Post-PAR static timing report:

The Generate Post- Place and Route Static Timing process is run after the design files have been successfully routed. The Post-Place and Route Static Timing process will create timing simulation data from which it can be ascertained if the timing requirements and functions of the design have been met after routing.

The Post Place and Route Static Timing Report process is used to view the post place and route static timing report that will give a calculated worst-case timing for all signal paths in the design. It optionally includes a complete listing of all delays on each individual path in the design.

c. Post Place and Route Simulation

This process runs the design through the Place and Route phase, uses the NGD Anno program on the TESTCONTROL.NCD file to back annotate the design, and then run the NGD2VER program to generate the simulation model. Once the simulation model has been generated, this process invokes Modelsim to simulate the post place and route simulation model.
d. Lock pin report

Each time when the design is implemented, a file is created which contains the pin locations and logical pad names information. After the pin-out is performed for the design the information is stored in the .gyd file and is appended to the end of the User Constraint File for your design (TESTCONTROL.UCF). This pin-out will then be applied to all subsequent design iterations that are run. After running the Lock Pins process, the report file can be viewed. The report will show any pin assignment conflicts that may have occurred.

7.7.10 Programming file generation

The Generate Programming File process is run on TESTCONTROL.NCD to create a bit stream file, or BIT file (TESTCONTROL.bit). The bit stream file is created from the placed and routed NCD file. The BIT file contains all of the configuration information from the NCD file defining the internal logic and interconnections of the FPGA, plus device-specific information from other files associated with the target device. The binary data in the BIT file can then be downloaded into the FPGA's memory cells, or it can be used to create a PROM file. The Programming File Generation Report process is used to view an error report created during the creation of the bit stream file that is generated by the Hardware Debugger.

7.7.11 Bit stream (Bit file)

A stream of data that contains location information for logic on the device; that is, the placement of CLBs, IOBs, TBUFFs, pins, and routing elements. The bit stream also includes empty placeholders that are filled with the logical states sent by the device during a read back. Only the memory elements, such as flip-flops, RAMs, and CLB outputs, are mapped to these placeholders, because their contents are likely to change from one state to another. When downloaded to a device, a bit stream configures the logic of a device and programs the device so that the states of that device can be read back. A bit stream file has a bit extension in the name of TESTCONTROL.BIT.
address of Port 2 and Port 3 is stored in the same SDRAM in a different location, through the R/W control block. The output cells that are stored in the SDRAM is uploaded and read from a note pad file named RAMUPLD through the parallel port of the PC.

13. The Dipswitch is set to input Port 1.

14. The same procedure is followed for downloading and reading the output.

15. The inputs and outputs of testing are shown in Tables 8.3, 8.4 and 8.5.