CHAPTER 7

DESIGN STRATEGIES & TOOLS UTILIZED

7-1. Field Programmable Gate Array

The internal architecture of an FPGA consist of several uncommitted logic blocks in which the design is to be encoded. The internal logic blocks consist of several universal gates that can be programmed to operate like multiplexers, logic gates, transistors and random access memory. The internal cell blocks are connected through different types of devices, such as static random access memory, electrically erasable programmable read only memory, or anti-fuse [18].

In conclusion, FPGA's offer fast time-to-market, low risk, low cost, low stress, high density and high speed-performance [22]. The FPGA kit used for this project was appropriate for testing and verification and has proven to be a very important tool for successful project completion.

7-2. Verilog HDL

Verilog HDL is the hardware descriptive language software that was used in this RISC design. It has syntax very similar to the high-level language known as 'C'. Hardware design languages such as Verilog allow designers to model the concurrence of processes found in hardware elements. Initially, HDL's were popular for logic verification but proved difficult to manually translate the HDL-based design into a

schematic circuit with interconnections between gates. The advent of logic synthesis tools such as Synplify by Synopsis Inc. and ModelSim by Mentor Graphics Inc., pushed HDL's into the forefront of digital design.

Hardware design languages have many advantages compared to traditional schematic-based design. For example,

- Designers can write the software without choosing a specific fabrication technology and Logic Synthesis tools can automatically convert the design to any fabrication technology.
- Functional verification of the design can be done early in the design cycle. Most design bugs can be eliminated at this point [16].

The design strategy is shown in Figure 7.1, below.

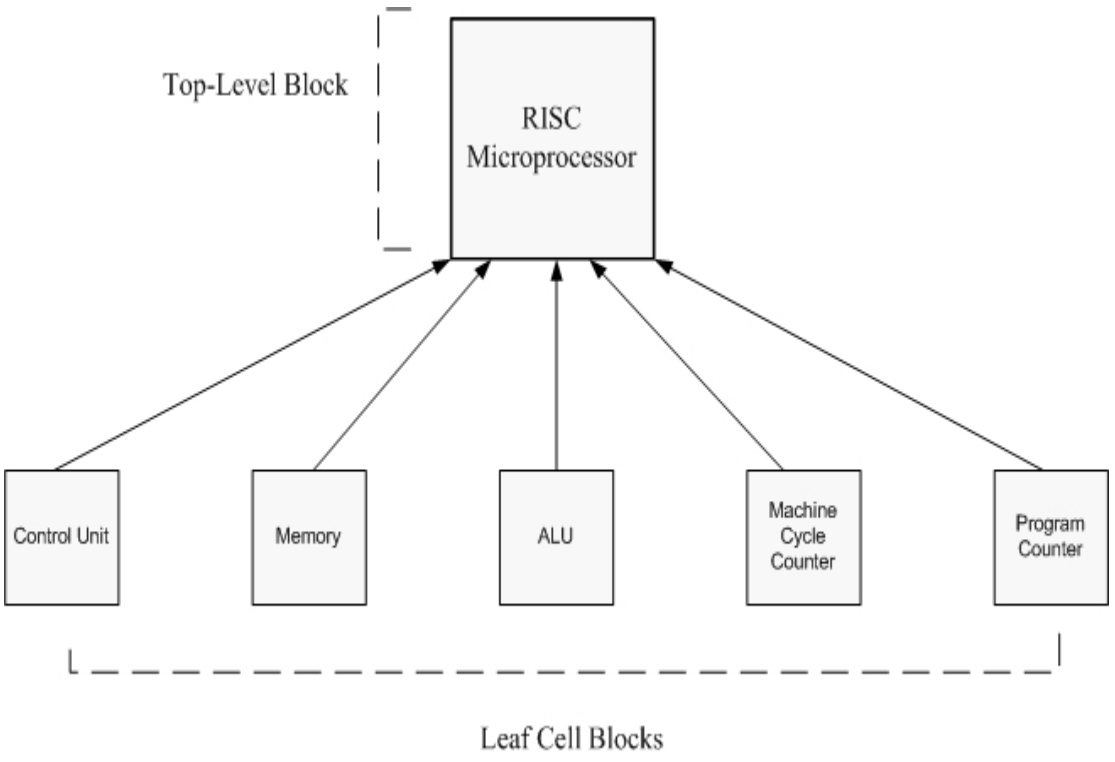

Figure 7.1: Top-Down Design.

7-3. Bottom – up Vs Top – down design

Verilog has two main design flow capabilities known as top-down and bottom-up design. In a top-down design methodology, the top-level block is defined and then sub-blocks are created as necessary. In a bottom-up design methodology, the building blocks that are available are defined. Bigger cells are built using these building blocks and these cells are then used for higher-level blocks to eventually build the top-level block in the design.

This particular project uses a bottom-up design flow methodology. The preliminary design strategy consisted of creating a module for each hardware entity. Verilog provides the concept of a module. A module provides the necessary functionality to the higher-level block through its port interface (inputs and outputs), but hides the internal implementation. There is a module for the Arithmetic Logic Unit, Latch, Transceiver, Tristate Buffer, Register, Display and Control Unit. These hardware blocks were used to create the higher level block of the processor, thus creating a top-level hierarchy that behaves like a processor [16].

7-4. Stimulus

One of the major advantages of designing using hardware design languages is the ability to test the design without ever moving outside the Verilog environment. This can be accomplished by applying the stimulus and checking the results. In Verilog, this type of block is known as the *Stimulus* Block. The Stimulus block is a block of Verilog code that instantiates the design block and directly drives the signals that control the processor. Figure 7.2 illustrates how the stimulus block was used to become the temporary top-level block during testing. Throughout the design process for the RISC machine, there have been numerous stimulus blocks created as the design progressed.

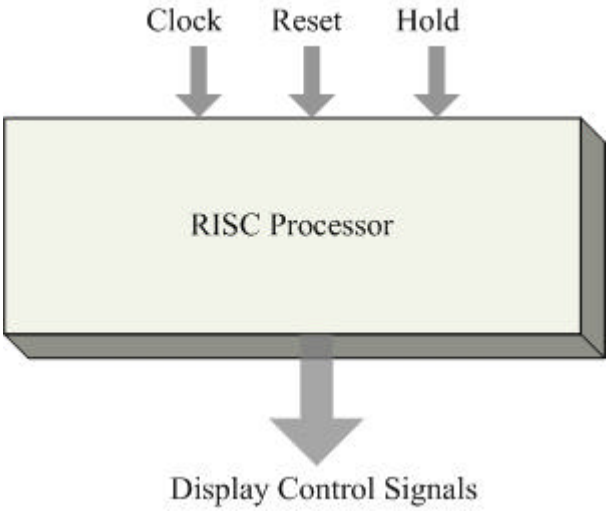

Figure 7.2: Stimulus Block Structure.

7-5. Xilinx Tools

Three primary tools made by Xilinx were used in the design flow for realizing and optimizing the RISC design and the XST is used for synthesizing the designed into a netlist.

7.5-1. ISE PROJECT NAVIGATOR

The Xilinx-ISE Project Navigator as shown in Figure 7.3 provides the basic platform for the design development of the '.v' file designed by the user. At this point the designer should be ready to define the speed, size of the target device. This tool also provides the necessary interface for accessing other development tools.

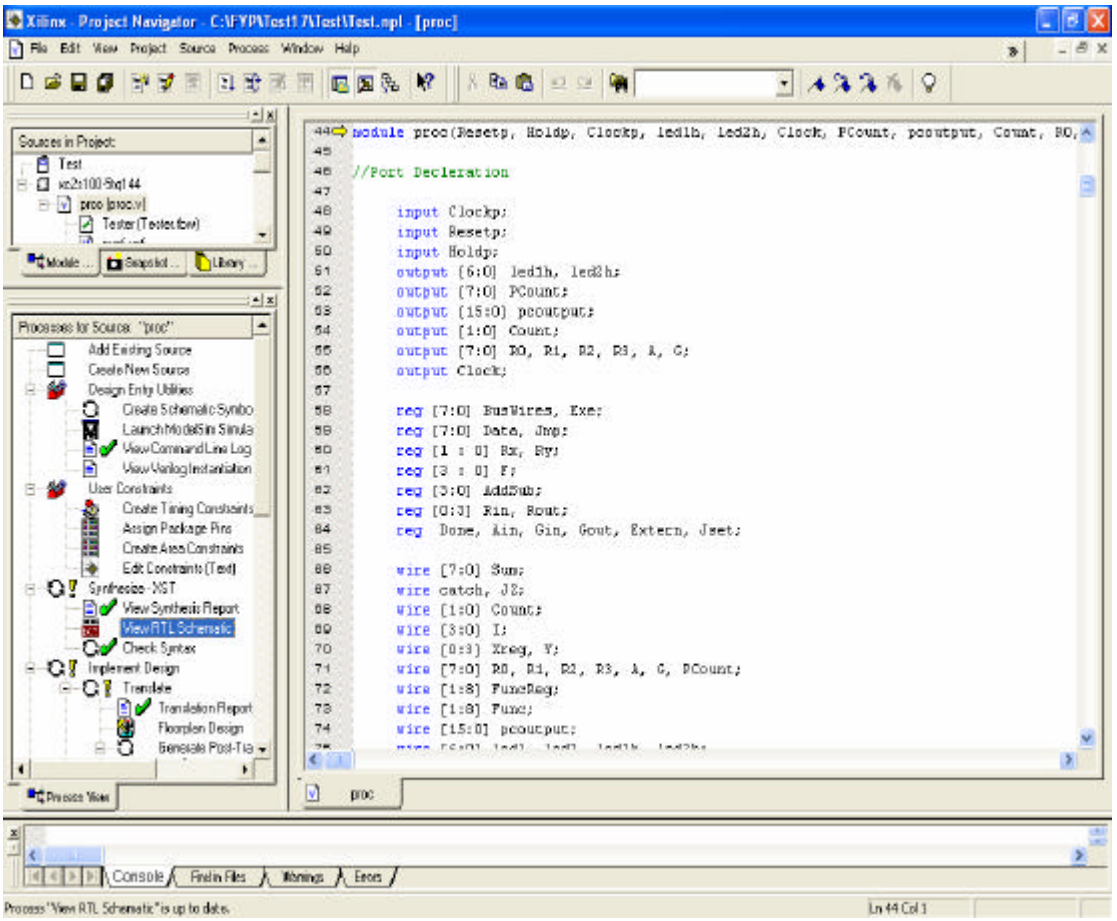

Figure 7.3: ISE Project Navigator.

7.5-2. XILINX SYNTHESIS TECHNOLOGY (XST)

You can synthesize your design once design files have been created. The synthesis process will check code syntax, and analyze the hierarchy of your design. These processes will ensure your design is optimized for the design architecture you have selected.

The synthesis process can be used with the following synthesis technology tools.

- Xilinx Synthesis Technology (XST)
- LeonardoSpectrum from Mentor Graphics Inc.
- Synplify and Synplify Pro from Synplicity Inc.

LeonardoSpectrum from Mentor Graphics, Inc, and Synplify and Synplify Pro from Synplicity Inc. can be purchased separately as synthesis programs. When these synthesis programs are installed, ISE provides the necessary interface to use the Xilinx implementation tools.

First select the source file in the **Sources in Project** window you want to synthesize and then double-click on the **Synthesize** process in the **Processes for Source** window. All processes necessary to successfully complete the Synthesis process run automatically. XST will create an NGC file and place it in your project directory. Default property values are used for the synthesize process unless you modify them. You can set the Synthesis Property Options in the Process Properties dialog box.

XST is executed from the main window of Xilinx-ISE Project Navigator. When the HDL code is completed with Verilog there are several steps that must be performed in order to synthesize the HDL design. The "check_syntax" command verifies the design and identifies any errors to be corrected. Using XST can be a very time consuming task and for this particular design, it took approximately one or two minutes to create a netlist file. The RTL schematic option will display the graphical structure of the verilog code as shown in Figure 7.4.

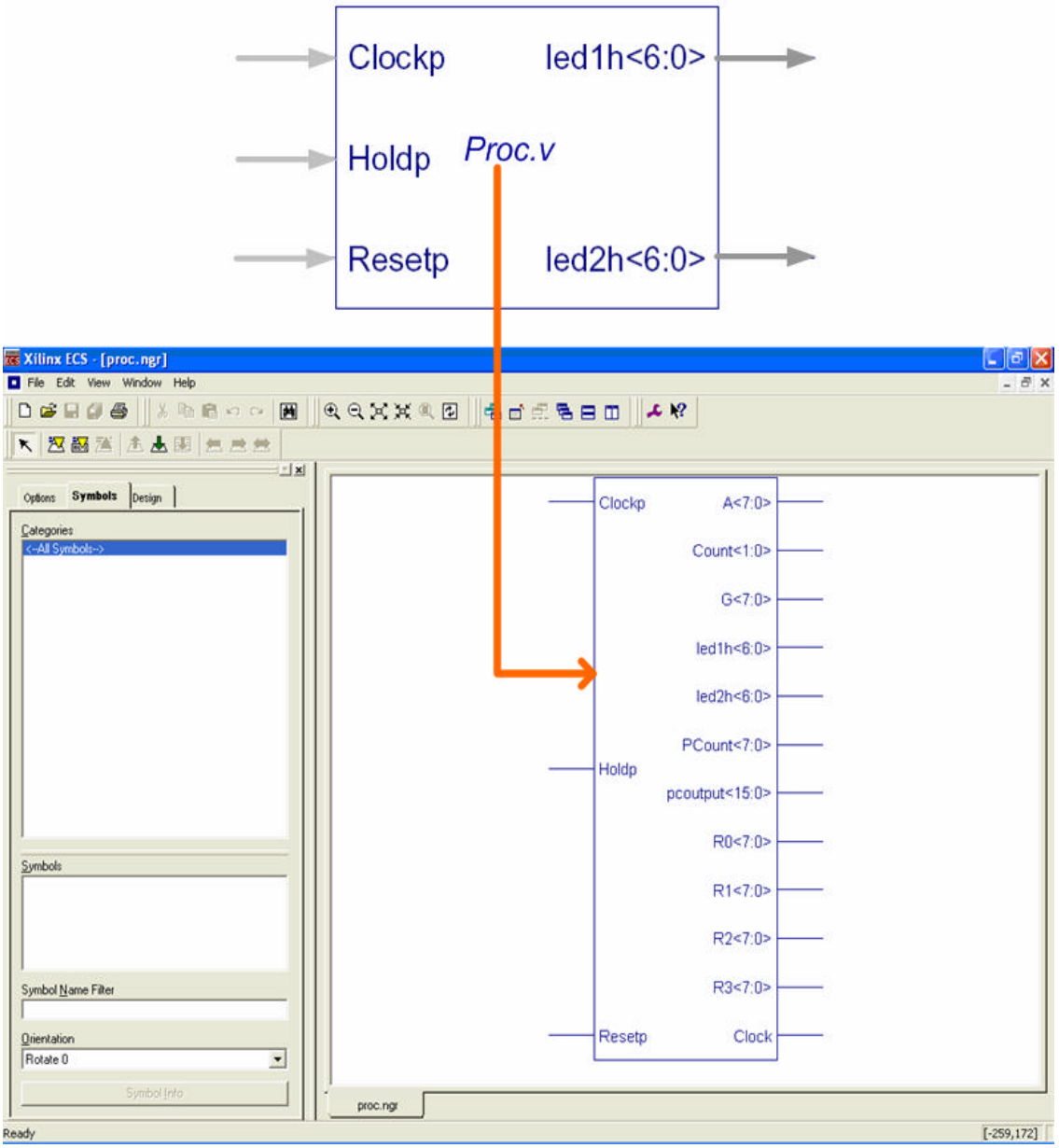

Figure 7.4: XST Synthesis Tool.

7.5-3. FLOOR PLANNER

The Xilinx Floor Planner tool can be used to view the interconnection between the CLBs in the FPGA and monitor the consumed surface area of the FPGA.

The Xilinx Flow Engine tool translates a XST NGC gate-level netlist file into the FPGA chip specific gate-level, based on the chip's libraries. This tool is activated by selecting "Design-> Implement" from the Xilinx-ISE Project Navigator window as shown in figure.

The Map, Place & Route stages of the Flow Engine tool maps the gate level design into CLBs (Cell Logic Blocks). The tool then determines where to place and route each of the design CLBs for optimal mapping and timing. The designer can select the effort level on how to best fit the design into the chip, area-wise and/or timing-wise.

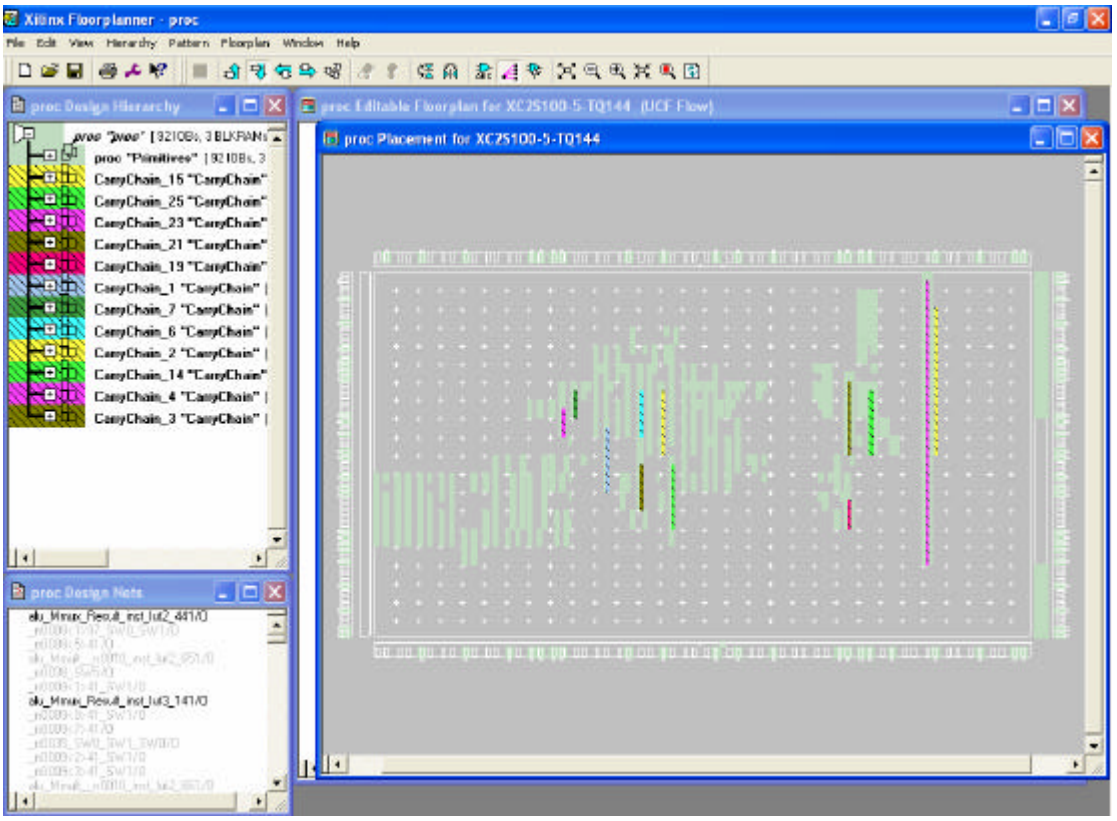

Figure 7.5: Xilinx Floor Planner Tool.

The Generate Timing report tool will determine the maximum and minimum timing delays through gates of the design. The maximum guaranteed speed (MHz) at which the design should be run is reported by the tool at this stage.

7.5-4. GENERATE PROGRAMMING FILE

The Generate Programming File tool then completes the design development cycle by creating a '.bit' file which can be downloaded via the LPT Cable using the iMPACT tool as shown in below figure or the tools provided by the Xess