An FPGA Project for use in a Digital Logic Course

Daniel C. Gray, Thomas D. Wagner
United States Military Academy

Abstract

The Digital Computer Logic Course offered at the United States Military Academy teaches cadets the principles of combinational and sequential logic, with an emphasis on programmable logic design. Classroom principles are reinforced with six lab exercises and two projects. In previous versions of the course, cadets were given a digital alarm clock kit that they constructed as credit for one lab.

In 1995, a decision was made to replace the alarm clock with a new project. The new project is a scrolling sign that interfaces through the parallel port of a PC. Two versions of the project have been built, the first using discrete MSI components, and the second using VHDL and a Xilinx FPGA. The FPGA implementation will be used in the future as one of the labs in the Digital Design Course.

This project proved to be a learning experience for the faculty in terms of VHDL, CAD tools, and synthesis onto an FPGA. This paper describes the process of designing the scrolling sign project and the intended use of the project in the EE curriculum at USMA. Lessons learned throughout the process are described as they occurred. The tools used in the design and why they were chosen are described.

I. Introduction

The Digital Computer Logic Course offered at the United States Military Academy teaches cadets the principles of combinational and sequential logic, with an emphasis on programmable logic design. Classroom principles are reinforced with six lab exercises and two projects. In previous versions of the course, cadets were given a digital alarm clock kit that they constructed as credit for one lab.

In 1995, a decision was made to replace the alarm clock with a new project. Two factors contributed to this decision. First, the supplier stopped providing the clock chip used in the design and another supplier could not be found. Second, it was time for a new project for cadets to build in the digital logic course. Two versions of a new project were built. The first version used discrete MSI (medium scale integration) components. The second version of the project used VHDL and a Xilinx FPGA (field programmable gate array).

Existing CAD tools were used to determine if they would support VHDL simulation and synthesis. The first iteration of the VHDL design was simulated in ViewLogic Powerview, synthesized with ViewSynthesis, and mapped onto a Xilinx FPGA using Xilinx Alliance M1.4. Both tools ran on a Sun SPARC1000. For the second iteration of the VHDL design, Xilinx

Foundation was used, running on a 233 MHz Pentium MMX. Both iterations gave satisfactory results.

This project proved to be a learning experience for the faculty in terms of VHDL, CAD tools, and synthesis onto an FPGA. Using VHDL to describe a circuit is different from gate-level design using traditional schematic capture techniques. The designer must be able to work without the visual representation provided by a gate-level circuit diagram. The benefit is that often much of the gate-level details can be abstracted away with a higher level VHDL description. However, an abstracted description may not synthesize. While VHDL allows you to write a standard "sequential" program, not all VHDL programs are guaranteed to synthesize into hardware.

A VHDL design can be described at several different levels, the two most common being behavioral and structural. A structural description mirrors a circuit layout, and leads to a design that more rapidly configures onto hardware. A behavioral description (or architecture) tends to look more like a traditional program. Some behavioral descriptions such as counters and state machines synthesize well into hardware. Other behavioral descriptions, particularly those that involve memory, or operations such as multiplication and division, may not lead to a design that can be synthesized onto an FPGA. Simply writing a design as a traditional sequential program without considering the hardware that is being described may not lead to a synthesizable design. A better approach is to describe some sub-components behaviorally, but keep in mind the overall hardware architecture when describing the circuit, and write structural architectures where necessary.

Selecting the correct CAD tool for design is also critical. The combination of VHDL, FPGA's, and CAD tools creates a steep learning curve. Two goals were kept in mind when selecting a CAD tool. First, the CAD tool had to support the entire design process through synthesis, placement, and routing on the FPGA. Second, the CAD tool had to be available and usable by undergraduates.

What follows is a high level architectural description of the chosen new project, comments on the MSI-based approach to the design, and a description of the VHDL/FPGA-based approach to the design. Lessons learned throughout the process are described as they occurred.

II. High Level Design

The project that was chosen was a scrolling sign. Like the digital clock, the scrolling sign project was envisioned to be something the cadets could build and then use in their rooms. The basic idea was to end up with a scrolling message display that could be connected to the cadets' personal computers. The project would require interfacing the parallel port of a PC with an Optrex LCD display through hardware that required design. The end product allows the user to type a message on a PC. The message is then sent through the parallel port to the designed hardware on a printed circuit board. The circuit stores the message and then scrolls the message across the Optrex display. The hardware design contains the following components: an eight-bit input register, an eight-bit output register, a seven-bit address counter, 128 bytes of SRAM, and a finite state machine. Figure One shows the data and control flow of the final design.

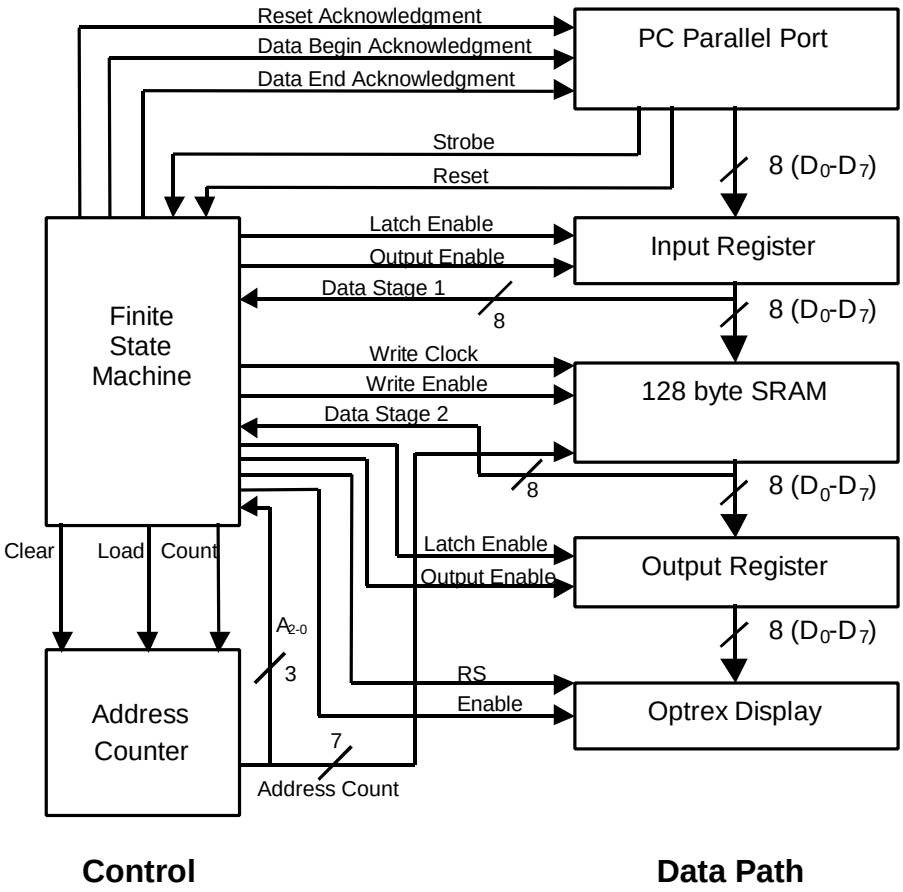

Figure 1: High Level Architecture

The data sent from the PC is in eight-bit ASCII character form. The data path flows from the parallel port, through the eight-bit input register, to a byte-wide static RAM memory chip. Once the message is stored, the data flows from the memory, through the eight-bit output register, to the Optrex display. The finite state machine controls the entire operation.

The finite state machine (FSM) is used to control the data flow. First, the FSM handshakes with the PC to receive each character. The FSM sends an acknowledgment to the PC when it is ready to receive a byte of data, and also after it has received the byte. This prevents the speed of the PC from outpacing the hardware receiving the data by forcing the software to wait until it receives the acknowledgments from the FSM. After receiving the byte, the FSM controls the timing sequence to latch the data into the input register, store the byte into an address location on the memory, and increment the address counter. While the FSM is storing the character data, the software on the PC is waiting for the next handshake signal from the FSM before sending the next byte. This sequence of states is looped in the state machine until the FSM receives a stop character.

The standard parallel port contains three byte-wide ports: the data port, status port, and control port. The software must ensure that it sets the control port bits correctly to enable data flow in the correct direction through the data port. The handshaking signals from the FSM are received

through the status port. The software also adds Optrex display initialization data to the front of the message and a stop character to the end of the message. All of the data is stored in sequential locations of the on-board memory, starting at address location zero.

Upon receipt of the stop character, the FSM exits the loop and begins sending data from the memory to the Optrex display. The first four characters are initialization characters for the display. The remainder of the characters contain the actual message to be scrolled. The FSM must send the appropriate control signals to read data from the correct memory location, latch it into the output register, and send the data from the output register to the Optrex display. Also, the display must be controlled for initialization and enabling data entry to the display. Once the message is stored in the memory, the sign can be disconnected from the PC. The FSM will ensure that the message is continuously scrolled.

III. The First Attempt

The first attempt at the scrolling sign project was designed and constructed using discrete MSI components. The first design issue was an interface that would allow sending data through the parallel port to an SRAM chip. This was designed using TTL logic components for input and output registers, and an address counter. A 2K x 8 static RAM chip was used to store the message. For the finite state machine, two AMD PALCE22V10 programmable chips were used. Two PLD's (programmable logic devices) were required because of the high number of control outputs. The CUPL PLD programming language, from Logical Devices, was used to program the PLD's.

Initially, this parallel port interface was designed using these hardware components and DOS debug. The debug commands compiled into a C program, which eventually became the software used to get the message from the user and send it to the hardware.

Another design issue was the speed difference between the PC sending the message and the hardware receiving the message. The PC sending the message operates at a higher clock rate than the hardware design. This obstacle was overcome by using the previously mentioned handshaking signals between the finite state machine and the software. This ensures that the FSM is in the correct state to receive the next ASCII character from the PC, and that the PC holds that character on the parallel port long enough for the hardware to read the data into the input register.

The first design attempt proved that data could be written through the parallel port, saved in the on-board memory, and written back to the parallel port. This required significant thought to correctly interface the software sending the message with the finite state machine controlling the hardware.

At this point, extending the design to control the sign display required frequent modifications to the FSM and "reburning" the resultant FSM onto the two PLD's. It was determined that if the existing design could be rewritten in VHDL and used to configure an FPGA, the design would be much more flexible and could more easily be used to prototype in hardware when the design was extended to control the Optrex display. Furthermore, the FPGA would produce a much more compact final product.

IV. Final Product

For the second attempt, it was decided to continue the design on an FPGA. This required rewriting the existing design in VHDL, synthesizing it onto an FPGA, and then testing to ensure that data could still be written through the parallel port, saved in the FPGA's memory, and then sent back to the PC as had been done with the MSI-based design. The first attempt provided confidence in the soundness of the design. It also provided a baseline on which to gauge the performance of the VHDL/FPGA design.

The FPGA was configured to contain the following synthesized components: an eight-bit input register, an eight-bit output register, a seven-bit address counter, 128 bytes of SRAM, and a finite state machine. Each MSI component was written using a behavioral description in VHDL. As each component was completed, it was simulated for correctness. The components were then tied together with a structural representation of the entire circuit and the overall system was simulated. The top-level structural VHDL architecture was conceptually identical to the MSI design. This proved to be an exceptional way for an experienced digital designer to learn VHDL.

Synthesis of the existing design onto a Xilinx XC5202 FPGA was attempted. This proved to be a bad choice, because the XC5202 does not support on-chip memory. Next, a design with the XC5202 and a separate memory chip was implemented. While this worked, it increased the pincount on the FPGA due to address and data lines between the memory chip and the FPGA. To address that problem a Xilinx XC4005E was chosen because it includes support for on-chip memory. The CAD tools configured the design for the XC4005E; however, there were new problems that using FPGA on-chip memory presented.

Although successful in the end, on-chip memory use proved to be challenging on an FPGA. First, as mentioned above, not all FPGA's would support on-chip memory. Second, the method used to describe the memory in VHDL had a significant impact on synthesis time and final operation in hardware. Initially, the memory was designed using a behavioral VHDL description. The design simulated correctly. However, the behavioral design required excessive time to synthesize. This occurred because the CAD tool read the behavioral description as a series of thousands of Boolean equations and then attempted to map them onto the FPGA. Also, while the behavioral SRAM simulated correctly and synthesized, it did not operate correctly when placed on the XC4005E. As a result, a structural VHDL implementation of the SRAM was written. This greatly reduced synthesis time because the CAD tool recognized the hardware-oriented structure of the memory. The structural VHDL description also performed correctly when configured on the actual FPGA.

As an alternative to structural memory descriptions in VHDL, Logiblox, from Xilinx, was used to create an SRAM memory model. This performed as well as the structural VHDL description and alleviated the need to write a structural SRAM architecture. This allows the designer to add SRAM to a design without needing to define the precise inner-workings of the memory. Simply understanding the memory's interface and control timing will suffice. Logiblox created a synthesized SRAM construct for FPGA configuration, and a behavioral VHDL description for simulation. To use the SRAM created by Logiblox, we simply instantiated the SRAM component in our top-level VHDL design.

The final product contained behavioral descriptions of the input and output registers, an address counter, and the finite state machine. The SRAM was created in Logiblox, as mentioned above. The entire design was tied together with a structural VHDL architecture. As an added benefit, the Foundation CAD tool allowed us to place an on-chip oscillator in the FPGA, alleviating the need to generate a clock signal off-chip. The resultant design used 77% of the FPGA's capacity, producing a circuit equivalent of approximately 6,253 logic gates. The VHDL/FPGA prototype was much easier to troubleshoot and implement than the previous MSI prototype. Furthermore, the FPGA design facilitated printed circuit board layout because there was a need for fewer board-level components. Also, flexibility of pin-assignments on the FPGA simplified board routing.

V. Conclusions and Further Work

This project will be a good teaching tool for digital design and VHDL. The project will be given to cadets as a kit to build in the Digital Computer Logic course. The follow-on computer architecture courses can then use the design to teach VHDL. There are many different ways that the design can be implemented by varying the behavioral/structural architecture mix of the components.

The tools to support teaching VHDL and FPGA's were validated. In a classroom of Sun workstations, Powerview lends itself well to VHDL simulation. Foundation requires an add-on tool for VHDL simulation. Or, the design can be simulated in Foundation after synthesis. Both companies offer student editions of their tools that provide schematic capture and simulation, but no VHDL support. We chose to use the full version of both tools for this project due to the requirement to use VHDL. For more advanced designs, the cadets will have the option of using either tool.

Once the initial steep learning curve is overcome, these tools and techniques lend themselves well to rapid systems prototyping. Because of the ease with which digital designs can be prototyped and implemented, a VHDL/FPGA based clock chip is being designed to replace the original clock chip that is no longer provided by the suppliers. In the near future, multiple FPGA projects may enhance VHDL instruction at the Military Academy.

Bibliography

1. K. C. Chang. *Digital Design and Modeling with VHDL and Synthesis*. Los Alamitos, CA: IEEE Computer Society Press (1997).

2. J. F. Wakerly. *Digital Design Principles and Practices*, 2nd Ed. Englewood Cliffs, NJ: Prentice Hall (1994).

3. Xilinx, Inc. *The Programmable Logic Data Book*. San Jose, CA: Xilinx, Inc. (1998).

4. S. Yalamanchili. *VHDL Starter's Guide*. Upper Saddle River, NJ: Prentice Hall (1998).

DANIEL C. GRAY
Daniel C. Gray is an Assistant Professor of Electrical Engineering in the Department of Electrical Engineering and Computer Science. He received a Master's Degree in Electrical Engineering from Duke University in 1994. His primary research interests are digital design using VHDL and FPGA's, and parallel computing.

THOMAS D. WAGNER
Dr. Thomas D. Wagner is an Assistant Professor of Computer Science in the Department of Electrical Engineering and Computer Science. He received the PhD from Vanderbilt in 1992. His primary research interests include parallel computing and digital image halftoning using error diffusion neural nets. Current work includes custom Windows NT device drivers and the application of the error diffusion neural nets to high speed analog to digital conversion.