Teaching IP Core Development: An Example

Aleksandar Milenkovic, David Fatzer
Electrical and Computer Engineering, The University of Alabama in Huntsville
E-mail: milenka@ece.uah.edu, URL: http://www.ece.uah.edu/~milenka

Abstract

The increasing gap between design productivity and chip complexity, and emerging systems-on-a-chip (SoC) have led to the wide utilization of reusable intellectual property (IP) cores. Educators' responsibility is to provide future generations of SoC architects with knowledge necessary for successful design and use of IP cores, and to offer them a system perspective including both hardware and software. One way to accomplish this goal is through projects focused on soft CPU cores development. In this paper, we show the design flow and give the details of one such project aimed to develop a Microchip's PIC18 microcontroller core and implement it on an FPGA.

1. Introduction

The traditional integrated circuits design flow based on the schematic capture has been replaced by the design flow based on hardware description languages (HDL) such as VHDL, Verilog, or SystemC, and tools for logic synthesis. The HDL-based design flow offers portability, reduction of the design cycle, independence of technology, and automatic synthesis and logic optimization. In spite of its great benefits, this design flow cannot successfully bridge the ever-increasing gap between available chip complexity and design productivity - since 1980, the number of transistors on a chip increases 58% a year, while design productivity increases merely 21% a year [1].

According to the SIA Technology Roadmap, by the end of this decade the semiconductor industry will manufacture chips with four billion transistors, thousands of pins, and clock speeds of 10GHz. In order to increase design productivity, a new design flow has recently emerged, based on the reuse of portable IP cores. An IP core is a block of logic or data that is used in making a field programmable gate array (FPGA) or an application-specific integrated circuit (ASIC).

The future generations of designers must know both how to design an IP core, and how to build a system-on-a-chip using IP core libraries. In this paper we describe a project-based approach in teaching senior students of Electrical and Computer Engineering how to develop a soft IP core, and implement it on an FPGA. The proposed approach can be incorporated into senior level courses teaching hardware description languages, advanced digital design, rapid prototyping, or VLSI. The prerequisites include basic digital design and computer architecture courses.

In the choice of an IP core suitable for a student project, we used the following rationale. Having in mind that future designers will design systems rather than components, most project specifications ask students to design a CPU or its derivative. Students develop a synthesizable VHDL or Verilog model of the CPU, fully verify it, implement that model on an FPGA, and demonstrate its functioning. Project task should be manageable in size so that a small group of students can successfully complete it during the course (roughly 100-200K gates).

Expected benefits of this approach are the following. In addition to the skills in VHDL structural modeling, verification, and synthesis, students also get the real-world experience working with state-of-the-art CPUs that have available C compiler or assembler. Proposed projects include both software and hardware components, and offer students a unique opportunity to understand and explore various design tradeoffs in complexity, performance, and power. The developed soft IP cores form a publicly available library, which can be used in design of more complex systems or in research. Also, they can be used for demonstration purposes in junior level courses. Last but not least, all projects require efficient teamwork.

The proposed approach is illustrated using a successful project aimed to implement a Microchip's high-end PIC18 microcontroller [2].

2. Soft IP Core Development

The project design flow encompasses both hardware and software tracks, as illustrated in Figure 1. In the first step called instruction set analysis, students study the Reference Manual of PIC18 microcontrollers [3], in order to become familiar with the instruction set, formats of instructions, and PIC18 architecture and organization. This step provides knowledge necessary both for the CPU design and test programs development.

The next step is to come up with the register transfer level (RTL) design using standard logic blocks such as

registers, ALUs, adders, subtractors, multipliers, register files, etc. The PIC18 employs Harvard architecture with separate 8-bit data and 16-bit program spaces. The program space permits access of up to 2Mbytes of non-volatile memory for instruction and constant data storage. The data address space includes up to 4Kbytes (16 banks, each 256 bytes). An accumulator provides temporary storage for multiple instruction operations, and serves as the second operand for multi-operand instructions. A sixteen-bit instruction latch permits current instruction decoding and execution in tandem with the next sequential instruction fetch cycle, thus providing two-stage pipeline organization. In order to conserve silicon area, Microchip chose to implement a single internal 8-bit databus for both read and write operations resulting in a 4-clock cycle instruction execution. Students have been guided to make the design simpler by incorporating separate read and write busses. While this implementation requires more silicon area, it permits single cycle operation of non-branch instructions, making the design four times faster than the conventional PIC18 operating at the same clock frequency. The third step in the hardware track is VHDL structural modeling. The VHDL hierarchy is shown in Figure 2.

In the software track, a set of testbenches is prepared using Microchip's MPLAB free assembler integrated development environment to verify each of 77 PIC18 instructions. A hex file for each test program is generated, and a utility program *hex2rom* is used to fill corresponding ROM VHDL module, linked with other modules to create a full system model. Verification is done using ModelSim.

After successful model verification, the Xilinx free WebPack tool is used to synthesize and implement the design into the Spartan II 100K gate FPGA. The final design occupies 78% of the 100K gate device. The critical timing path of 54ns (18.450MH) is determined by the Multiplier module implemented as an array multiplier. Introducing a pipelined implementation of the array multiplier results in shortening of the critical path to 44ns. Program execution is demonstrated by synthesizing the design on a Spartan II Development Kit interfaced through the parallel ports to a 4-line by 20-character LCD display. A simple assembly language program is written to display "Hello world" on the display (Figure 1).

3. Conclusions

The proposed project-based approach encompasses whole engineering cycle, starting from specification, through design, modeling, simulation and verification, implementation, to performance measurement, and closing the cycle with the design improvements in order to maximize performance at minimal cost. Students have an opportunity to apply their theoretical knowledge of hardware description languages, digital design and computer architecture, and to gain real-world experience in developing IP cores. The work in small teams follows a real industry pattern, with one student designated as team leader, and instructor conducting design reviews every week. We feel that such projects are essential to educate future architects of complex systems-on-a-chip.

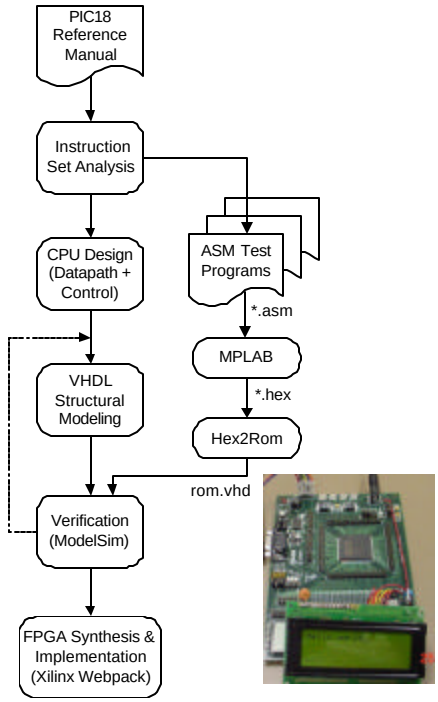

Figure 1. Design Flow.

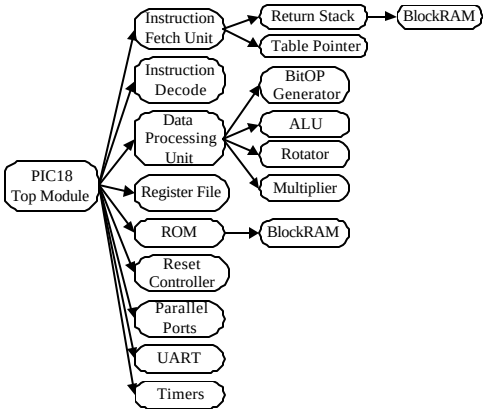

Figure 2. VHDL Hierarchy.

REFERENCES

[1] C. Rowen, "Reducing SoC Simulation and Development Time," *IEEE Computer*, vol. 35, no. 12, December 2002, pp. 29-35.
[2] PIC18 Soft Core: http://www.ece.uah.edu/~milenka/pic18/pic.htm
[3] PIC18C MCU Family Reference Manual, http://www.microchip.com/1010/suppdoc/refernce/golden/index.htm