A VLSI PROTOTYPE OF A 32-BIT RISC PROCESSOR

Flávio du Pin Calmon and Ricardo Pezzuol Jacobi

Departamento de Engenharia Elétrica, Universidade de Brasília, CP 4386, Brasília, DF 70904-970, Brazil

ABSTRACT

This paper presents the prototyping of a 32-bit RISC processor using VHDL and describes the necessary steps for validation of the architecture, implementation, simulation and test of the project. The processor used is the RISCO, presented in [1]. Strategies for optimization of VHDL code are studied, analyzing the implementation of a bank of 32-bit registers. RISCO is a strong candidate for being used as a co-processor or an auxiliary module in SoC (System on Chip) applications.

1. INTRODUCTION

Hardware Description Languages (HDLs) are used for programming PLDs (Programmable Logic Devices). The use of HDLs allows a well-defined description of a project and permits a faster design. As a consequence, a preliminary version of a digital project can be quickly compiled, synthesized and programmed on PLDs, allowing the validation of a device's architecture and the testing of its integration with other elements of a system [2].

This work presents the modeling of a 32-bit RISC processor using VHDL (Very High Speed Integrated Circuit Hardware Description Language) for its synthesis on a FPGA, describing the necessary steps for validation of the architecture, implementation, simulation and test of the project. The processor used is the RISCO, described in [1]. This processor has certain characteristics that allow future changes and enhancements, such as the possibility of expanding its instruction set without great modifications of the architecture, increasing its functionality. RISCO is a strong candidate for being used as a co-processor or an auxiliary module (e.g. configurable processor [3]) in SoC (System on Chip) applications.

The paper is divided in five parts. Initially, the general characteristics of the RISCO processor are presented. In the third topic, the steps for the implementation in VHDL are described. The next item shows the results of tests and simulations. Finally, the conclusions are presented.

2. RISCO'S CHARACTERISTICS

RISCO is a 32-bit processor with RISC architecture. Therefore, its data, instruction and addresses are referred by 32-bit words. This processor has a three stage pipeline, with a peak operation of one instruction per machine cycle. Each machine cycle is formed by three different phases. "Jump" instructions have their execution retarded by one machine cycle.

The architecture can be divided in "operational part", referring to the memory, communication (e.g. buses) and transformation elements (e.g. ALU), and the "control part", designating the state machine and pipeline control that defines the processor's operation.

3. VHDL IMPLEMENTATION

The RISCO model created with VHDL implements a comprehensive and unambiguous functional description of the processor. This approach was necessary due to the lack of a simple and clear reference of the register transfer level characteristics of the device. A simple and well commented VHDL code was developed.

The project uses a "down-up" approach, describing the most basic elements of the architecture first. The implementation was divided in two blocks (figure 1), connecting them in a final entity.

3.1. Operational block

The structures of the operational block were created following basic project rules for avoiding errors when joining them in a final entity. All structures were described to keep their outputs in a high-impedance state when not being accessed. This measure was essential for the operation of the model, due to the extensive use of shared buses in the architecture. The project aimed coherence between the implemented structures and those found in the original conception of RISCO.

VHDL allows multiple ways for describing a certain structure, resulting in a greater versatility and speed for code generation. To study the effect of synthesizing a component described using different VHDL structures, five different implementations of the RISCO's register bank were created, comparing the synthesis results using Xilinx ISE 6 software [4]. All the syntheses were made for a Xilinx XC2S50E FPGA. The characteristics of each implementation can be found on Table I, and the synthesis results on Table II.

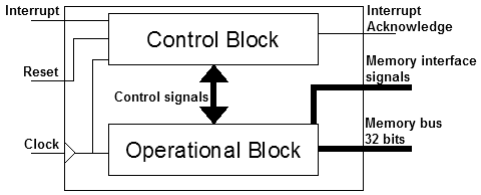

Figure 1 - RISCO's block diagram. On the left are the input signals and on the right, the output signals.

TABLE I - DIFFERENT IMPLEMENTATION STRATEGIES FOR A 32-BIT REGISTER BANK

Implementation number	Characteristics
1	Described with sequential instructions, using the *conv_integer* function for decoding selection signals. The resulting code is simple and clear.
2	Same as the first implementation, except with explicit address decoding using *case-when* type instructions. Results in a larger code that is harder to debug, but functionally more direct.
3	Uses a structural, low-level description, joining in a final entity address decoders, multiplexers for controlling bus access and 32 individual registers.
4	Uses a structural description, inserting more complexity to the individual registers, using fewer components for external access control, such as decoders and multiplexers. The resulting code is simple and direct.
5	Based on the synthesis results of the previous four implementations. Uses the *conv_integer* function for decoding, with little complexity in the individual registers. The bus access was described using 32-bit buffer banks.

The synthesis results indicate clearly that the implementation that generates the least device utilization is the fifth one. Therefore, a good strategy, in terms of resource occupation, for creating VHDL code is to use structural descriptions, using a reduced logic complexity in individual components (such as the registers, in this case) and standardized functions whenever possible, present in the language's libraries.

3.2. Control block

The RISCO's control block was implemented using sequential instructions. The Pipeline was described as a state machine, always executing the three necessary phases for each instruction, using additional phases for memory access instructions. A general reset was also introduced, along with the necessary signaling for the external memory. The use of VHDL allowed the development of a concise description of the Pipeline's dynamics. As a consequence, the code may be used as a reference for future studies of the processor. Also, the analysis of any changes that may be done on the processor's architecture will be considerably simpler.

TABLE II -PERCENTAGE OF DEVICE UTILIZATION FOR EACH IMPLEMENTATION OF THE REGISTER BANK ON A XILINX XC2S50E FPGA.

Resources	Implementation number				
	1	2	3	4	5
Number of *Slices*	160%	216%	143%	148%	74%
Number of *Slice Flip Flop*	72%	72%	66%	137%	64%
Number of 4 *input LUTs*	141%	206%	80%	24%	20%
Number of *GCLKs*	-	-	25%	25%	25%

TABLE III - RESULT OF RISCO´S SYNTHESIS FOR A XILINX XC3S400 FPGA.

Parameter	Result
Number of *Slices*	84%
Number of *Slice Flip Flops*	18%
Number of 4 *input LUTs*	78%
Number of *IOBs*	17%
Number of *GCLKs*	25%
Maximum operation frequency	50.774 MHz

4. TESTS, SIMULATIONS AND RESULTS

Once finalized the description of the RISCO processor, a series of simple tests validated each component of the operational block and the control block. After verifying the correct operation of each block, a test structure was created. An external memory was described using VHDL, storing the instructions and data for simulating the processor's operation. A final block instantiated and connected both entities, resulting in the final test-bench entity.

The simulations of the test-bench entity used the ModelSim XE III 6.0a software. The tests realized were sufficient to validate the architecture, with the execution of various instructions being verified, along with the Pipeline's behavior. The resource utilization of the final synthesis using Xilinx ISE 7.1.03i software [5] can be found on Table III.

5. CONCLUSION

A 32-bit RISC processor, the RISCO, was successfully validated and tested using VHDL. With an implementation of the processor using a Hardware description language, it is considerably easier to understand the RISCO's operation and study the effect of future changes or enhancements on its operation. An analysis of different strategies for describing logic devices in VHDL was also made, allowing synthesis optimization.

6. REFERENCES

[1] A. A. Junqueira, "RISCO – Microprocessador RISC CMOS de 32 bits", M.Sc. dissertation, Informatics Institute of the Federal University of Rio Grande do Sul, Porto Alegre, RS, 1993.

[2] K. Skahill, *VHDL for programmable logic*. California: Addison-Wesley Longman, 1996. 594p.

[3] S. Leibson, J. Kim, "Configurable Processors: A New Era in Chip Design" *IEEE Computer Magzine*, vol. 38, no. 7, pp. 51–59, Jul. 2005.

[4] *ISE 6 In-Depth Tutorial*, Xilinx Inc., 2005 [Online]. Available: http://direct.xilinx.com/direct/ise6_tutorials/

[5] *ISE 7 In-Depth Tutorial*, Xilinx Inc., 2005 [Online]. Available: http://direct.xilinx.com/direct/ise7_tutorials/