Simulation and Verification of Self Test 16-Bit Processor

Manoranjan Pradhan
Department of Electronics and Telecommunication Engineering,
VSS University of Technology, Burla, India

ABSTRACT
This paper presents the design and verification of 16 bit processor. The Booth multiplier and restoring division are integrated in to the ALU of the proposed processor. The processor is described in structural level to verify the general understanding of the system. The processor has 16-bit instruction based on three different format R-format, I-format and J-format. The control unit generates all the control signals needed to control the coordination among the entire component of the processor. All the modules in the design are coded in VHDL (very high speed integrated circuit hardware description language) to ease the description, verification, simulation and hardware implementation. The design entry, synthesis, and simulation of processor are done by using Xilinx ISE 10.1 software and implemented on XC2S200-6pq208 Spartan-II FPGA device.

General Terms
Booth Algorithms, Restoring division Algorithm.

Keywords
Register transfer level, Reduced instruction set computer, Very high speed integrated circuit hardware description language , Arithmetic logic unit, Field programmable gate array.

1. INTRODUCTION
The authors in [1] presented a design methodology of a single clock cycle Processor using VHDL to ease the description, verification, simulation and hardware realization. The RISC processor has fixed-length of 32-bit instructions based on three different formats: R-format, I-format and J-format, and 32-bit general-purpose registers with memory word of 32-bit. The processor is separated into five stages: instruction fetch, instruction decode, execution, data memory and write back. The control unit controls the operations performed in these stages. All the modules in the design are coded in VHDL, as it is very useful tool with its concept of concurrency to cope with the parallelism of digital hardware. The authors in [2] proposed a VHDL based rapid prototyping approach to simulate, synthesize, and implement a prototype computer system using commercial CAD tools, a Meta assembler, a C compiler, and FPGAs in a hardware emulator. The use of VHDL for the design and implementation of a CPU structure has been presented in [3]. Initially the CPU is described at the behavioral level. This description is used by the designer to verify the general understanding of the system under design. This design phase is followed by the actual design of the CPU. For this purpose, a more detailed description of hardware is developed. This description is at the dataflow level, and includes register and bus structure details of the hardware.

The authors in [4] proposed a processor which can execute two instruction sets simultaneously. The processor can execute VLIW (very long instruction word) instruction set for multimedia processing. Their processor is based on a VLIW pipeline. The processor decodes either of the two instruction sets in its corresponding front end. By these means, they have successfully united the two processors of different purposes into one specific processor. As a result, they have reduced hardware cost, footprint, and power consumption to meet the rising demands of portable media processing market. The authors in [5] presented hardware architecture for a thread level parallel processing processor which exploits the continuance concept based on the dataflow model and evaluate the performance of the processor.

The authors have proposed a graph theoretic model of pipelined processors and develop a systematic approach to path delay fault testing of such processor cores using the processor instruction set in [6]. The proposed methodology generates test vectors under the extracted architectural constraints. These test vectors can be applied in functional mode of operation. Hence, self-test becomes possible. Self-test in a functional mode can also be used for online periodic testing. Their approach uses a graph model for architectural constraint extraction and path classification.

In this paper, we have been motivated to integrate Booth multiplier and restoring division circuit in the design of 16-bit processor for availing more advantages of flexibility.

2. DESIGN OF 16-BIT PROCESSOR
Usually, all classical design methods involve one or more PCBs (printed circuit board) that contain many chips together with other components. Development of these products starts with the definition of the overall structure. After that the required integrated circuits chips are selected followed by the PCBs that house and connect the chips together are designed. Since the complexity of circuits implemented on individual chips and on the circuit boards is usually very high, it is very much essential to make use of Xilinx software [7].

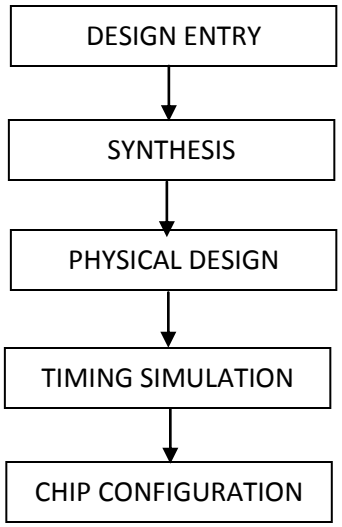

Fig.1. Design Method of proposed 16 bit processor

The design method of proposed 16 bit processor is shown in Fig.1. FPGA devices are software configured and field programmable. Hence, there is a significant cost saving in design and productions. Design entry of the processor is carried out by using VHDL code. Initial synthesis generates an initial circuit, based on data entered during the design entry stage. Functional simulation is done to verify the functionality of the circuit, based on inputs provided by the designer. Logic synthesis and optimization uses optimization techniques to derive optimized circuits. Physical design determines how to implement the optimized circuit in a FPGA chip. Timing simulation determines the propagation delays that are expected in the implemented circuit. Chip configuration configures the actual chip to realize the designed circuit.

3. ARCHITECTURE

The architecture of proposed 16-bit Processor is shown in Fig.2 which consists of processor block and memory block communicating through data bus, an address bus and a few control lines [8-10]. The architecture of the 16-bit processor is designed based on three 16-bit instruction formats R-format, I-format and J-format. The design of this processor consists of 16-bit instructions and 16-bit data path. The implementation of processor performs fetch, decode, and execute operation. The fetch stage obtains the requested instruction from memory. The operation of the fetch stage starts when the program counter (PC) a 16-bit register is sent out to fetch the instruction from memory into the instruction register (IR) and the PC is incremented to address the next sequential instruction. The IR is used to hold the instruction needed on subsequent clock cycles.

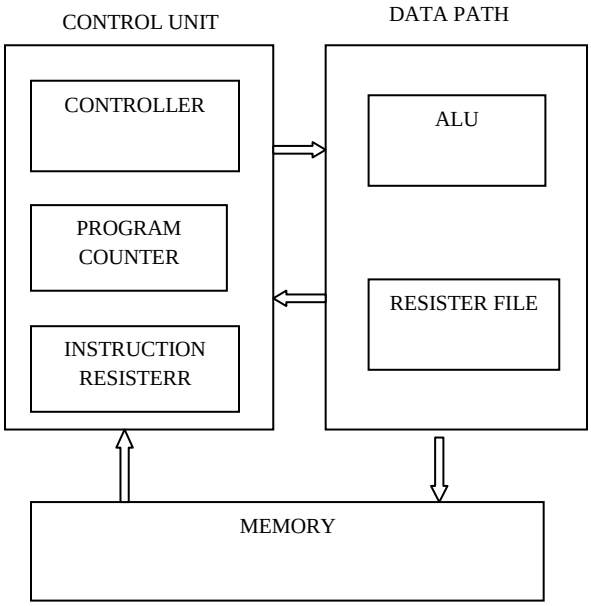

Fig.2 Proposed 16-bit Processor Architecture

The processor's 16-bit registers are stored in 'register file' that contains the register state of the machine. For R-format instructions, there are three register operands. Two data words are read from the register file and one data word is written into the register file for each instruction. The register number inputs are 4 bits wide to specify one of the 16 registers, whereas the data input and two data output buses are each 16 bits wide. When the instruction is fetched from the fetch stage, the instruction's operation code is sent to the control unit. The instruction's register address fields are used to address the two-port register file. The two-port register performs two independent reads and one write in one clock cycle .In the decode stage, the instructions are decoded and the register file is accessed to read the registers. The outputs of the general-purpose registers are read into two registers, register 1 and register 2. After the instructions decode stage, the execution stage performs calculations.

3.1 Data path

There are 16 number of 16 bit registers in register file. So Register file of data path consists of 4-bit address bus and 16-bit data bus. This is a two port register file which can perform two simultaneous read and one write operation. It contains sixteen 16-bit general purpose registers. The registers are named R0 through R15. When the Register Write signal is high, a write operation is performed to the register indicated by the write address, otherwise it outputs the value contained in the register indicated by the read address. The ALU is responsible for all arithmetic and logic operations that take place within the processor. These operations can have one operand or two, with these values coming from either the register file or from the immediate value from the instruction directly. The output of the ALU goes either to the memory (in the case where the output is an address) or through a multiplexer back to the register file. The ALU has two 16-bit data inputs for two operands and 16-bit output for result. It can perform Booth multiplication, restoring division, addition and subtraction functions. The small multiplexer of the data path has three 16-bit inputs and one 16-bit output. It selects one of the 16-bit inputs and sends to the

output depending on the condition of two-bit multiplexer select signal.

The multiplier unit has two inputs and one output. The internal structure consists of four units namely Booth Encoder, Partial Product generator, Carry Save Adder, and Carry Look Ahead Adder. The top design of Booth Encoder has one input and three outputs. The Booth Encoder generates the booth codes to encode the multiplicand into the partial products. The outputs of the Booth Encoder are fed to the inputs of the partial product generator (PPG) module. The PPG module reads the booth code signals generated from the Booth Encoder to encode the multiplicand into the partial products. The outputs of the partial product generators are taken as the inputs of the Wallace tree. The Wallace tree uses the 3 to 2 carry save adder to implement the Wallace tree. The outputs of the Wallace tree are fed to the inputs of the Carry Look Ahead (CLA) adder. The CLA is used to perform the addition of the final sum and carry vector. After implementing all the four components of the fast multiplier unit, the desired multiplication result is obtained.

In restoring division, the divisor is positioned appropriately with respect to the dividend. Then the divisor is subtracted from the dividend. If the remainder is zero or positive, a quotient bit of '1' is determined. The remainder is extended by another bit of the dividend and the divisor is repositioned and another subtraction is performed. If the remainder is negative, a quotient bit of '0' is determined and the dividend is restored by adding back the divisor. Then the divisor is repositioned for another subtraction.

3.2 Control unit

Control unit composed of controller, program counter, instruction register, and multiplexer. The controller provides the necessary signal interaction to make the data flow through the processor and perform expected function. The 16-bit program counter indicates the address of the next location where the next instruction is to be fetched. It has 16-bit program counter input, 16-bit program counter output, and some control signals like read, write, and clear. The Program Counter (PC) contains the address of the instruction that will be fetched from the Instruction Memory during the next clock cycle. Normally the PC is incremented by one during each clock cycle unless a branch instruction is executed. When a branch instruction is encountered, the PC is incremented or decremented by the amount indicated by the branch offset. The instruction register (IR) has one 16-bit input and two 16-bit outputs. The instruction set describes the bit-configurations allowed in the IR .The instruction consists of operation codes and operand. The processor support direct, register, registers indirect and immediate addressing modes. In immediate addressing; the operand field contains the data itself. In registers addressing, the operand field contains the address of a data path register in which the data resides. In registers indirect addressing, the operand field contains the address of a register, which in turn contains the address of a memory location in which the data resides. In direct addressing, the operand field contains the address of a memory location in which the data resides. In indirect addressing, the operand field contains the address of a memory location, which in turn contains the address of a memory location in which the data resides.

Memories are used for storage of both instructions and data. The process of storing data into memory is called writing and retrieving data or operation codes from the memory is called reading. For reading and writing data, the particular memory location is identified and then reading or writing is done. There are 256 number of 16-bit memory locations. It has 8-bit address and 16-bit word.

4. SIMULATION AND DISCUSSION

We have synthesized and simulated the VHDL code of the proposed processor using Xilinx Integrated Software Environment tool (Version 10.1). The synthesis results and simulation results of processor are presented for justification. The proposed 16 bit RISC processor is coded with VHDL (very high speed integrated circuit hardware description language). Using Xilinx ISE 10.1 software the code is tested and checked for error. When there is no error, the code is synthesized and simulated using Xilinx ISE 10.1 software. The synthesis and simulation results are compared with the theoretical results. Before the start of simulation, the memory is loaded by writing the instructions and data into the memory. The completed processor with memory is tested for addition, subtraction, multiplication, and division program. When the VHDL code is 100% synthesized, and then the code is downloaded to the Spartan FPGA device. After downloading the code, the functionality of 16 bit RISC processor is compared with theoretical result.

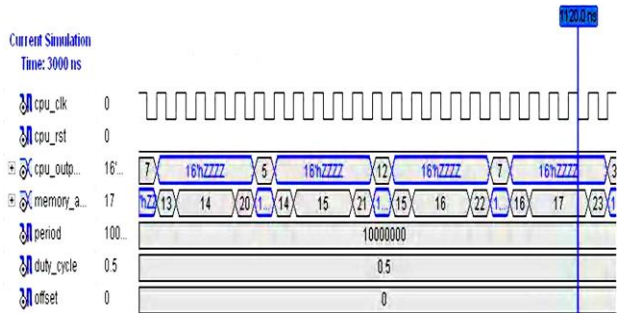

Fig.3. Simulation of proposed 16 bit processor

The simulation result of the proposed 16 bit RISC processor is shown in Fig.4. The two input 16 bit operand to the processor are 7 and 5.They can be either in register location, memory location, or given as immediate values. The two 16 bit outputs from processor are cpu_out (15:0) and memory_add (15:0).The cpu_out (15:0) and memory_add (15:0) represent arithmetic logic unit result and memory address respectively. Initially '7' decimal equivalent of 16 bit operand is copied into register address R1.Similarly '5' is copied into register address R2.The contents of R1 that is 7 is copied in memory address '19'.Similarly contents of R2 is stored in memory address '20'. Now '7' contents of R1 is added with '5' contents of R2 to produce result '12' and is stored in R1.The addition result in R1 is copied in memory address '21'. Similarly '5' contents of R2 is subtracted from '12' modified contents of R1 to produce result '7' and is stored in R1.The subtraction result in R1 is copied in memory address '22'. Likewise '7' modified contents of R1 is multiplied with '5' contents of R2 to produce result '35' and is

stored in R1.The multiplication result in R1 is copied in memory address '23'. Then '35' modified contents of R1 is divided with '5' contents of R2 to produce result '12' and is stored in R1.The division result in R1 is copied in memory address '24'.

5. CONCLUSION

We tested our processor architecture by running addition, subtraction, booth multiplication, and restoring division programs created using Xilinx Software. We have compared the simulated output results with the expected results. From synthesis report, the minimum clock period that can be achieved in this proposed architecture is 31.641 ns (31.605 MHz). The booth multiplier unit is tested and the functionality is found correct. Similarly, the functionality of restoring division circuit is tested and found correct. Finally, the performance of the processor is tested for addition, subtraction, booth multiplication, and restoring division programs and the functionality is found correct.

6. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template.

7. REFERENCES

[1] Mamun B., Shabiul I. and Sulaiman S. 2002. A Single Clock Cycle MIPS RISC Processor Design using VHDL. Penang, Malaysia, pp.199- 203, 2002.

[2] Hamblen J. Using Synthesis, Simulation, and Hardware Emulation to Prototype a Pipelined RISC Computer System. Atlanta, Georgia.

[3] Zainalabedin N. Using VHDL for Modeling and Design of Processing Units. Pp.315- 326, Boston, Massachusetts.

[4] Outline of OROCHI. 2007. A Multiple Instruction Set Executable SMT Processor., International Workshop on Innovative Architecture for Future generation Processors and Systems.

[5] Takanori M., Satoshi A., and Masaaki I. 2005. A Multi-thread Processor Architecture Based on the Continuation Model. Proceedings of the Innovative Architecture for Future Generation High-Performance Processors and Systems. Kasuga-Koen, Kasuga, Fukuoka,Japan.

[6] Virendra S. and Michiko I. 2006. Instruction-Based Self-Testing of Delay Faults in Pipelined Processors., IEEE Transaction on VLSI systems, vol. 14, no.11,pp.1203-1215.

[7] Hamacher V. and Zaky. 2002. Computer Organization. McGraw-Hill Companies, New York, 5th edition.

[8] Frank V. and Tony G. 1999. Embedded System Design, A Unified Hardware/Software Approach., Department of Computer Science and Engineering University of California, Draft version.

[9] Patterson A. and Hennessy J. 1999. Computer Organization & Design., Morgan Kaufmann Publishers.

[10] Weijun Z. 2001. VHDL Tutorial, Learn by Example.

[11] Data sheet of Spartan-II 2.5 FPGA Family. 2003. XILINX, DS001-2 (V2.2).