FPGA IMPLEMENTATION OF 16 BIT RISC CPU AND PERFORMANCE ANALYSIS

DINESH B BORUDE  
Student, Electronics and Telecommunication Engg, MIT College of Engineering, Aurangabad, Maharashtra, India  
PROF.S.V.VERMA  
Assistant professor, Electronics and Telecommunication Engg, MIT College of Engineering, Aurangabad, Maharashtra, India

ABSTRACT:
RISC (Reduced Instruction Set Computer) found several applications in the engineering. In this paper, authors have design, implement and performance analysis of a 16-bit Reduced Instruction Set (RISC) CPU using XILINX tool. The significant attribute of the RISC processor is that it is incredibly simple and sustains load/store architecture. The processor includes the ALU, Shifter, Register array, Instruction register, program counter, address register, Operand register, Comparator and Control unit. The performance parameters like area and propagation interruption are analyzed at 90 nm process tools using SPARTAN 3-XC3S400 FPGA and XILINX tool.

KEYWORDS: FPGA, Xilinx, VHDL, RISC.

INTRODUCTION:
Recently computers are crucial tools for most of the daily activities. RISC is an expansion of the architecture morality of the Reduced Instruction Set Computer (hereafter RISC). The effortless design provides outstanding recital and is ideal for use in a broad family of cost-effective, compatible systems.

The foremost purpose of this paper is to design and execute a 16-bit RISC processor using XILINX tool and Spartan 3-XC3S400 FPGA. The essential components of this processor comprise the Arithmetic Logic Unit, Shifter, output register, Instruction register, program counter, register array, address register, comparator and Control unit [1]. The architecture supports Arithmetic, Logical, Shifting, Branch and Rotational operations.

ARCHITECTURE OF 16-BIT RISC PROCESSOR:
The design of a 16-bit RISC processor is shown in “fig. 1”. [1]This consists of arithmetic logic unit, control unit, shifter and Register array, instruction register, program counter, comparator and output register. The processor is deliberate with load/store (Von Neumann) design. One mutual memory for instructions (program) and data with one data bus and one address bus between processor and memory. Instruction and data are fetched in chronological order so that the latency incurred between the machine cycles can be reduced. The instruction in RISC CPU would be executed in three stages i.e. fetch, decode, and execute in fetch, instruction and the necessary data are haggard from the memory. While in decode, the instruction and data that are derived from the memory are separated triggering the components and the data path so as to execute and finally in execution, the instruction is executed, data is manipulated and the result is stored.

![Fig-1: Architecture of 16-Bit RISC Processor](image)

The control unit interprets the opcode and instruction bits and then creates control signals as outputs that trigger the respective components and data path to perform the desired task [3]. The control unit with two instruction decoders which decodes instruction bits and decoded output of the control unit is nourished as control signal either into Arithmetic logic unit (ALU) or Universal shifter. From register A and register B operands are received by the ALU. Depending on control signal from control unit the ALU performs either mathematics or judgmental operations. After the carrying out of the instruction, the result is stored in the output register (Out Reg). The shifted data is accumulated in destination register which is nothing but the Out Reg.

2.1 CPU MODULES
It consists of different module designs like control unit, ALU, Universal shift register, comparator, address register,
output register, program counter, control unit, CPU unit, register array and general purpose registers[2].

I) ALU:

The first module illustrated is the ALU [2]. It executes arithmetic or logical operations on one or more input busses. A representation for ALU is shown in Fig- 2.

![Fig- 2: ALU Entity](Image)

Inputs “a” and “b” are the two input buses upon which the ALU functions are executed. Output bus “c” returns the outcome of the ALU operation. Input sel determines which operation is completed. The ALU can perform arithmetic actions such as add, subtract, increment, decrement, and some logical operations such as AND, OR, and XOR, NOT etc.

II) COMPARATOR:

The next module described is the comparator. It compares two values and returns either a ‘1’ or ‘0’ depending on the type of comparison requested and the values being compared. A symbol showing the ports of the comparator is shown in figure 3[4].

![Fig- 3: Comparator](Image)

The comparison type is resolved by the value on input port sel. For instance, to compare if inputs “a” and “b” are equal, apply the value eq to port sel. If ports “a” and “b” have the same value, port compout returns ‘1’. If the values are not equal, ‘0’ is returned.

III) REGISTER:

It is used for the address register and the instruction register. These registers need to be able to confine input data on a rising edge of the clk input and drive output q with the captured data. The value of input a is assigned to output q when a rising edge occurs on input clk. [4] The assignment is delayed by 1 nanosecond to remove delta delay problems during simulation.

![Fig- 4: Register Entity](Image)

IV) REGISTER ARRAY:

![Fig- 5: Register Array Entity](Image)

It is used to model the set of registers within the CPU that are used to store intermediate values during instruction processing [4]. These registers are read from and written to during the execution of instructions. The set of registers is modeled as a RAM of eight 16-bit words.[4] To write a location in the register array, set input sel to the location to be written, input data with the data to be written, and put a rising edge on input clk. To read a location from register array, set input sel to the location to be read and set input ld to a ‘1’; the data is output on port q[3].

V) SHIFTER:

The next module to be described is the shifter. The shifter is used to perform shifting and rotation operations within the CPU. The shift has a 16-bit input bus, a 16-bit output bus, and a sel input that determines which shift operation to perform.

![Fig- 6: Shifter Entity](Image)

Shifter block can shift the data towards left or right side by n-bit depending on sel signal. Same block can be used to rotate the Signal values.

VI) CONTROL UNIT:

It provides the essential signal interactions to compose the data flow properly through CPU and carry out the expected functions [5]. A representation for control block is shown in Figure below. The control symbol has only a few inputs, but many outputs. The control block present all of the control signals to regulate data traffic for the CPU. Architecture rtl contains two processes. The first is a combinational (not clocked) that inspect the current state and all inputs, produces output control values and next state output. The second is a sequential process (has a clock) that is used to store the current state and copy the next state to the current state [3]. The next state transitions occur on rising edges of the clock input. Executing all of the states for an instruction performs the necessary steps to complete the instruction. If the reset signal is high, the sequential process labeled controlffProc sets signal current_state to state value reset1.
DESIGN AND VERIFICATION:

Table -1: ALU Operations

<table>
<thead>
<tr>
<th>Sel input</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>bypass</td>
<td>C = A</td>
</tr>
<tr>
<td>andOp</td>
<td>C = A AND B</td>
</tr>
<tr>
<td>orOp</td>
<td>C = A OR B</td>
</tr>
<tr>
<td>notOp</td>
<td>C = NOT A</td>
</tr>
<tr>
<td>xorOp</td>
<td>C = A XOR B</td>
</tr>
<tr>
<td>add</td>
<td>C = A + B</td>
</tr>
<tr>
<td>sub</td>
<td>C = A - B</td>
</tr>
<tr>
<td>Inc</td>
<td>C = A + 1</td>
</tr>
<tr>
<td>Dec</td>
<td>C = A - 1</td>
</tr>
<tr>
<td>zero</td>
<td>C = 0</td>
</tr>
<tr>
<td>Comp2op</td>
<td>C = NOT A+1</td>
</tr>
</tbody>
</table>

The architecture uses a large case statement on input sel to determine which of the arithmetic or logical operations to perform. The possible values of signal sel are determined by type t_alu described in package cpu_lib in file cpulibvhdl. After the new value for output c is calculated, all of the resulting values are assigned with a 1-nanosecond time delay to eliminate delta delay problems during RTL simulation.

Table -2: Comparator Operation

<table>
<thead>
<tr>
<th>Sel input</th>
<th>Sel value</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>00</td>
<td>Compout = 1 when a equals b</td>
</tr>
<tr>
<td>NEQ</td>
<td>01</td>
<td>Compout = 1 when a is not equal to b</td>
</tr>
<tr>
<td>GT</td>
<td>10</td>
<td>Compout = 1 when a is greater than b</td>
</tr>
<tr>
<td>LT</td>
<td>11</td>
<td>Compout = 1 when a is less than b</td>
</tr>
</tbody>
</table>

SOFTWARE AND TOOLS USED:

Synthesis is done using Xilinx ISE 8.1i and implementation is done on Spartan 3 –XC3S400 FPGA kit.

4.1. SPARTAN -III FPGA FEATURES

- 400 k logic cell SPARTAN -III FPGA in PQ208 Plastic Quad Flat Package (MXS3FK-004-DSP)
- Three families Spartan 3 /Spartan 3L/Spartan 3 XA.
- Low cost, high-performance logic solution for high-volume, consumer-oriented applications.
- Densities as high as 74,680 logic cells.
- Three power rails for core (1.2V), I/O’s (1.2V to 3.3V) and Auxiliary purpose (2.5V).
- 326 MHz system clock rate.
- 90 nm process technology.
- 18 single-ended signal standards.
- Up to 1,672 Kbits of total block RAM.

4.2. VHDL:

VHDL or VHSIC (Very High Speed Integrated Circuits) Hardware Description Language is commonly used as a design entry language for field programmable gate arrays and application specified integrated circuits in electronic design automation of digital circuits. It expresses the performance of an electronic system, from which the physical circuit or system can be implemented. It is a typical technology/vendor sovereign language, and is therefore portable and reusable.

RESULTS AND DISCUSSION:

5.1 SIMULATION RESULTS:

I) ALU

II) COMPARATOR:
5.2 SYNTHESIS RESULT

The following Synthesis table shows the device utilization summary. These are estimated values which are generated by XILINX software by clicking on option of synthesis report. The logic utilization having different parameters like number of slices, Number of Slice Flip Flops etc. according to ratio between used parameters over available parameters we can determine the device utilization factor.

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices:</td>
<td>371</td>
<td>3584</td>
<td>10%</td>
</tr>
<tr>
<td>Number of Slice Flip Flops:</td>
<td>321</td>
<td>7168</td>
<td>4%</td>
</tr>
<tr>
<td>Number of 4 input LUTs:</td>
<td>527</td>
<td>7168</td>
<td>4%</td>
</tr>
<tr>
<td>Number of bonded IOBs:</td>
<td>18</td>
<td>141</td>
<td>12%</td>
</tr>
<tr>
<td>Number of GCLKs:</td>
<td>1</td>
<td>8</td>
<td>12%</td>
</tr>
</tbody>
</table>

CONCLUSION:

In this project I have implemented a 16 bit central processing unit using VHDL code. Design includes processor and a memory block which communicates through a bi-directional data bus, an address bus, and a few control lines. The instructions are stored in the instruction register (IR) and decoded by the control unit. This processor can perform 16 bit data transfer, simple arithmetic & logical operation and branching operation.

REFERENCES: