Volume 3, Issue 12, December 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Control Unit Design of a 16-bit Processor Using VHDL

Alpesh Kumar Dauda*
Deptt. Of El&TC Engg.
Pkace, Bargarh, Odisha
India

Nalinikanta Barpanda
Deptt. Of AEl&IE Engg.
G.I.E.T, Gunupur
India

Nilamani Bhoi
Deptt. Of El&TC Engg.
Vssut,Burla,Odisha
India

Manoranjan Pradhan
Deptt. Of El&TC Engg
Vssut,Burla, Odisha
India

Abstract—This paper describes the design and implementation of control unit of a 16-bit processor that is implemented in Spartan-II FPGA device. The CPU (Central Processing Unit) is the "brain" of the computer. Its function is to execute the programs stored in the main memory by fetching their instructions, examining them, and executing them one after another. The CPU is composed of several distinct parts, like data path, control path and memory units. For operating the data path CONTROL UNIT is needed to generate the control signals automatically at each clock cycle. The proposed architecture illustrates behavioral and structural description styles of a 16-bit control unit.

Keywords— CPU, FPGA, VHDL, RISC, INSTRUCTION REGISTER

I. INTRODUCTION

The control unit inside the processor is a finite state machine. By stepping through a sequence of states, the control unit controls the operations of the datapath. For each state the output logic that is inside the control unit will generate all of the appropriate control signals for the datapath to perform one data operation. These data operations are referred to as register-transfer operations. Each register-transfer operation consists of reading a value from a register, modifying the value by one or more functional units, and finally, writing the modified value back into the same or in a different register [1].

Neenu joseph et.al [3] tried to design a technique for power reduction in front end process. They are using pipelining architecture for designing a 32-bit RISC processor for embedded and portable application. The technique they are using for power reduction is clock gating. Clock gating is a method where the clock signal is prevented from reaching the various modules of the processor. The authors in [5] presented a design methodology of a single clock cycle MIPS RISC 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 MIPS 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.

In [6], authors analyzes MIPS instruction format, instruction data path, decoder module function and design theory based on RISC CPU instruction set. All MIPS instruction is 32-bit specified instruction and instruction address is word justification. MIPS has three instruction format

(1) Immediate format (I-format)
(2) Register format(R-format)
(3) Jump format (J-format)

Design of a 16-bit teaching microprocessor using the micro programmed control method, which is implemented on Xilinx xc2s150 FPGA have been presented in [7]. Where both the control memory and micro-instruction registers are 32 bits wide. The control memory capacity is 512 words (512x32 bits). The author in [8], presented a technique for asynchronous circuit design by using a novel asynchronous control unit. This is composed of transmission gates and inverters. The performance of this circuit is reduces because the control of asynchronous circuit is very hard. They also verify the control unit by designing and implementing an 8x8 multiplier. Similar research work also find in literature [9].

There are various techniques to implement control unit such as: Hardwired implementation and Micro programmed implementation. The micro programmed control unit is simple to design, but it will be slower in speed than a hardwired unit. For adding a new machine instruction in hardwired control is found to be very difficult. So here we proposed an easy method to design a 16-bit processor control unit by using micro programming.

This paper is organized as follows. In section-II, FPGA based design method is discussed. Section-III describes control unit architecture of a 16-bit processor. The Opcode chart, ALU chart and IR format as for different types of instructions are shown in section-III. The synthesis and simulation results are illustrated in section-IV. The conclusion is given in section-V.

II. DESIGN METHOD

The basic architecture of the system is designed, which is coded in a Hardware description Language like Verilog or VHDL. Design entry of the Control unit is done by using VHDL code i.e. the input to the design is given by writing the VHDL code for the control unit. Initial synthesis generates an initial circuit, based on data entered during the design entry stage. Functional simulation issued to verify the functionality of the circuit, based on inputs provided by the designer. Logic synthesis and use of optimization techniques leads to optimized circuits in the design process. 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. This VHDL code is written with the help of the VHDL language which is a part of the hardware description language (HDL) [2], [4].

III. PROPOSED ARCHITECTURE

The control unit generates all the control signals needed to control the coordination among the entire component of the processor. The control unit for a processor basically cycles through three main steps, usually referred to as the instruction cycle: 1) Fetch an instruction; 2) Decodes the instruction; and 3) Executes the instruction.

Each step is executed in one state of the finite-state machine. For step 3, each instruction is usually executed in one clock cycle, although some memory access instructions may require two or more clock cycles to complete, hence they may require several states for correct timing. For fetching the instruction in step 1, the control unit simply reads the memory location specified by the Program Counter (PC), and copies the content of that location into the instruction register (IR). The Program Counter (PC) is then incremented by 1. For decoding the instruction in step 2, the control unit extracts the opcode bits from the instruction register and determines what the current instruction is by jumping to the state that is assigned for executing that instruction. Once in that particular state, the control unit performs step 3 by simply asserting the appropriate control signals for controlling the data path to execute that instruction. Control unit composed of controller, program counter, instruction register and multiplexer [1].

1) *Individual modules of control unit:*

1.1 Controller: The controller consists of logic circuits to fetch program instructions. It moves data to, from, and through the data path according to those instructions. The Controller contains program counter (PC) that holds the address in memory of the next Program instruction to fetch. The controller also contains instruction register (IR) to hold the fetched instruction. The controller's control logic generates the appropriate signals to control the flow of data in the data path according to these instructions. Such data flows may include inputting two particular registers into the ALU, storing ALU results into a particular register, or moving data between memory and a register. The next-state logic determines the next value of the PC. For data transfer, arithmetic and logical instruction, this logic increments the PC. For a branch instruction, this logic looks at the data path status signals and the IR to determine the appropriate next address. The control unit generates all the control signals needed to control the coordination among the entire component of the processor. The input to this unit is the 4-bit operation code field of the instruction word. This unit generates signals that control all the read and write operations. The control unit of the processor generates timing and control signals for execution of instructions. It controls fetching and decoding operations. It generates appropriate control signals for instruction execution and also the signals required to interface memory. The processor support simple set of instructions with less complex addressing modes. So the speed and performance are comparatively higher. Here in this block (Fig.1) total 19 ports are used, among which 3 are inputs and 16 are outputs. Among the inputs clock and rst are the same clock signal and reset signal from the top module. Instruction register 16-bit (IR_word(15:0)) is another input signal which is output from instruction register and input to controller. All the output signals are control signals, which are input to different section of processor for performing different operation.

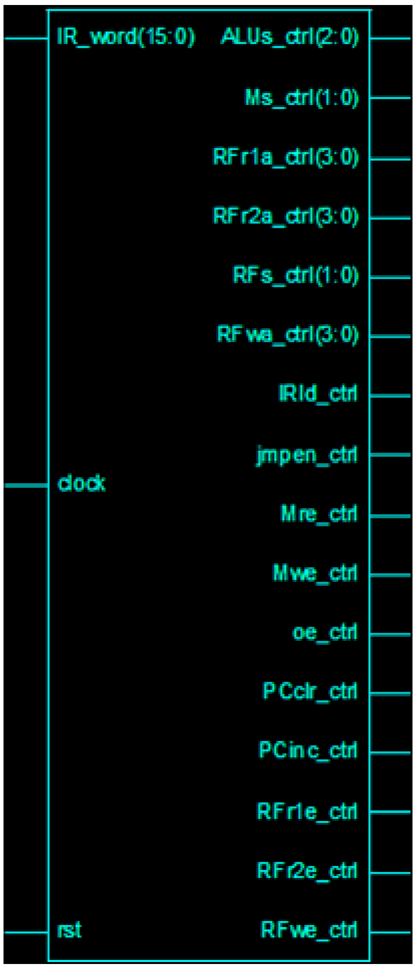


Fig. 1 Block structure over the controller

1.2 Instruction register: Instruction register is for storing the instruction being fetched from the memory The IR format has sixteen bits. The first four bits are opcode fields; the next four bits are register address fields, the least significant eight bits represent either memory address or immediate operand as per addressing modes. In Fig.2 when IRld is asserted (=1) then the 16-bit IRin becomes IRout and the least significant 8-bit of IRin goes to dir_addr.

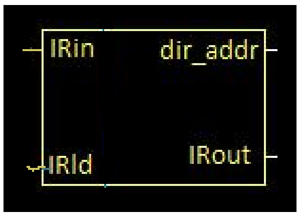


Fig.2 Block Structure over the Instruction Register

1.3 Program counter: The program counter holds the address of the memory of the current instruction. After the instruction is executed, the program counter advances to the next instruction. If there is branch instruction, the program counter is loaded with the address of the next instruction directly. The control unit copies the program counter values to the address register, which output the new address in address bus. The block structure of program counter is shown in the Fig.3 below. Here in this block total 5 ports are used among which 4 are input port and one is output port. “PCin” is a 16-bit input port and PCout is a 16- bit output port.

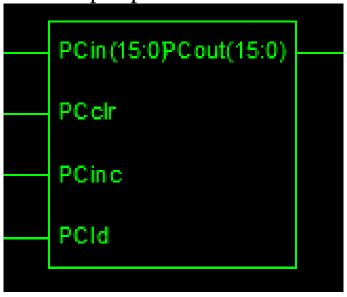


Fig.3 Block structure over the Program counter

1.4 Multiplexer: The multiplexer, allows the selection of one input signal among n signals, where n > 1, and is a power of two. Select lines connected to the multiplexer determine which input signal is selected and passed to the output of the multiplexer. In general, an n-to-1 multiplexer has n data input lines, m select lines where m = log2n, i.e. 2m = n, and one output line. For a 4-to-1 multiplexer, there are two select lines, “option (1:0)”, to select between the four inputs, Ia, Ib, Ic and Id. The block structure over the multiplexer is shown in Fig.4 below.

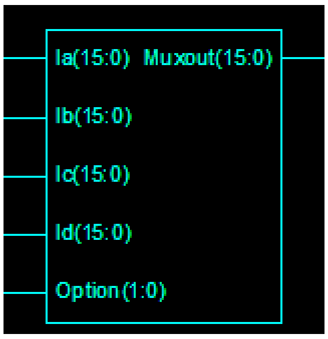


Fig.4 Block structure over the Multiplexer

1.5 Memory: Memories or registers are utilized for storage purpose of processor. For long term information memory is used and register is used for short term information. Basically the information to be stored is of two types, data information and program information. A sequence of instruction of program information makes the system to perform the desired function. The input and output values are represented by data information which is transformed by program. Memories are used for storage of both instructions and data. The process of storing data into memory is called writing and retrieving data or opcode from the memory is called reading. For reading and writing data, the particular memory location is identified and then reading or writing is done. Fig.5 shows the top view of memory block. There are 256 number of 16-bit memory locations. It has 8-bit address and 16-bit word.

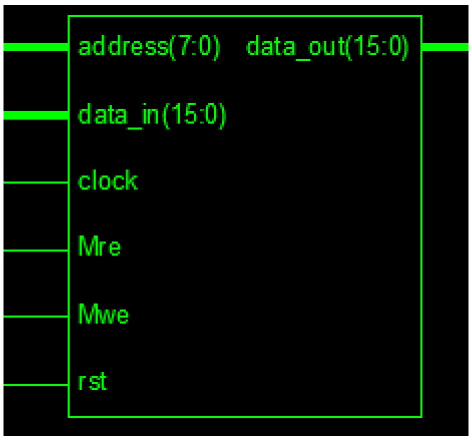
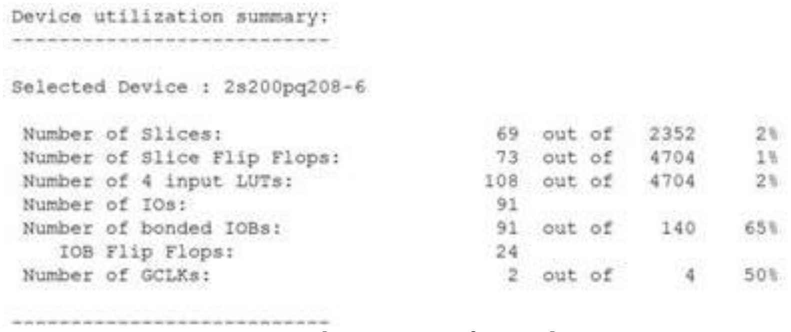


Fig.5 Top view of memory block

When reset signal is asserted (rst=1) the fetch operation perform and the instruction and data are written into the memory. When reset is deasserted (rst=0) then memory read or write operation performed.

IV. SYNTHESIS AND SIMULATION RESULTS

The performance of the approach used is determined on XILINX platform using VHDL as hardware description language. Optimization is achieved in terms of effective resource utilization in terms of LUT’s, no of input-output bonds, no of registers used etc. In this paper, the synthesis and simulation results of control unit of the 16-bit processor have been presented.



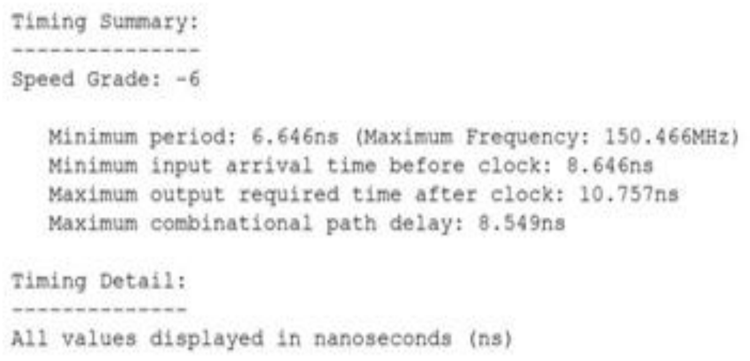
```
Device utilization summary:
---------------------------

Selected Device : 2s200pq208-6

 Number of Slices:                      69  out of   2352     2%
 Number of Slice Flip Flops:            73  out of   4704     1%
 Number of 4 input LUTs:               108  out of   4704     2%
 Number of IOs:                         91
 Number of bonded IOBs:                 91  out of    140    65%
    IOB Flip Flops:                     24
 Number of GCLKs:                        2  out of      4    50%

---------------------------
```

Fig.6 Synthesis report of control unit

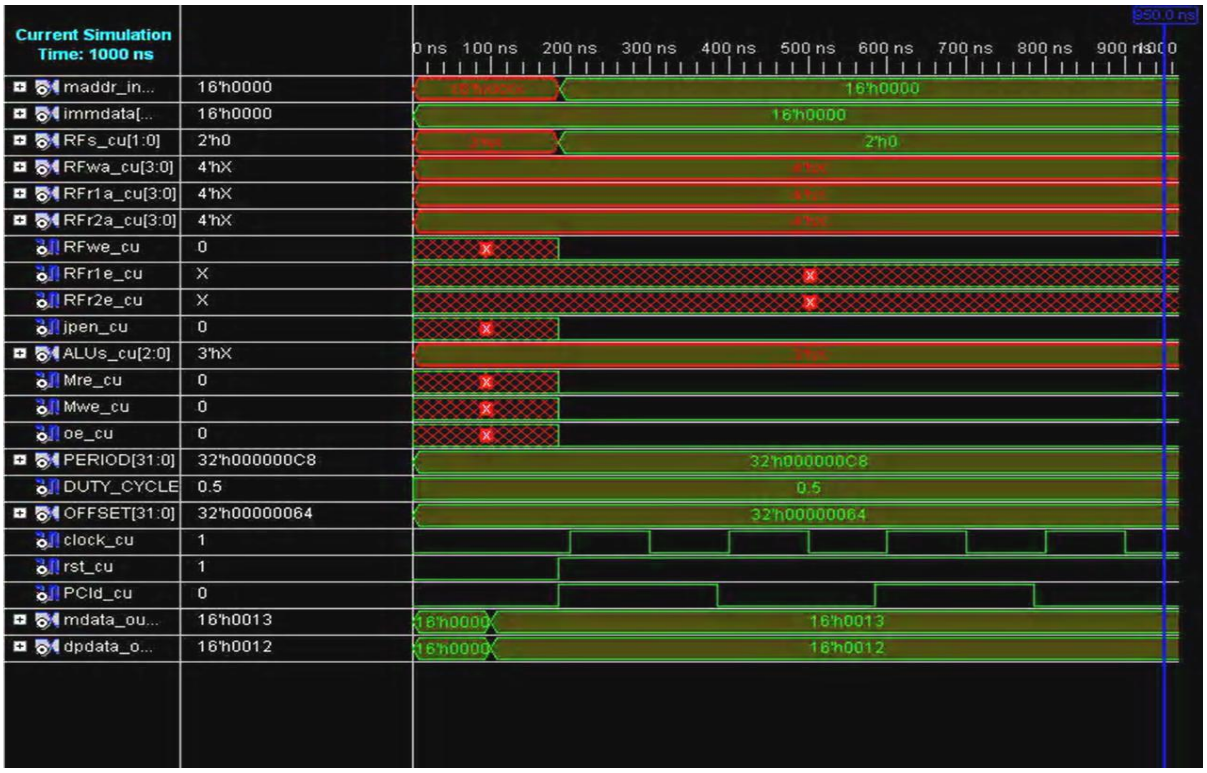


```
Timing Summary:
---------------
Speed Grade: -6

   Minimum period: 6.646ns (Maximum Frequency: 150.466MHz)
   Minimum input arrival time before clock: 8.646ns
   Maximum output required time after clock: 10.757ns
   Maximum combinational path delay: 8.549ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)
```

Fig.7 Timing report of control unit



```
Current Simulation
Time: 1000 ns

maddr_in...      16'h0000
immdata[...      16'h0000
RFs_cu[1:0]      2'h0
RFwa_cu[3:0]     4'hX
RFr1a_cu[3:0]    4'hX
RFr2a_cu[3:0]    4'hX
RFwe_cu          0
RFr1e_cu         X
RFr2e_cu         X
jpen_cu          0
ALUs_cu[2:0]     3'hX
Mre_cu           0
Mwe_cu           0
oe_cu            0
PERIOD[31:0]     32'h000000C8
DUTY_CYCLE       0.5
OFFSET[31:0]     32'h00000064
clock_cu         1
rst_cu           1
PCld_cu          0
mdata_ou...      16'h0013
dpdata_o...      16'h0012
```

Fig.8 Simulation results for control unit

V. CONCLUSIONS

The design architecture of a 16-bit processor's control unit is written in Very High Speed Integrated Circuit Hardware Description Language (VHDL) code using Xilinx ISE 10.1 tool for synthesis and simulation. From synthesis estimate, the minimum clock period that can be achieved in this proposed control unit architecture is 6.646 ns, which translate to a maximum operating frequency of 150.466 MHz & the combinational path delay is 8.549 ns. In **t**his design the simulation of the model is run using Xilinx ISE version 10.1**.** Simulation of design is done in a bottom-up fashion to verify the correctness of design.

REFERENCES

[1] David A. Patterson, John L. Hennessy, Computer Organization and Design (The Hardware /Software Interface), Third edition, Elsevier Inc., 2005.

[2] Hall. Douglas V., Microprocessors and Interfacing Programming and Hardware, McGraw-Hill International, Inc. 1992.

[3] Neenu joseph,Sabaninath.S, Sankarapandiammal K, "FPGA based Implementation of High Performance Archi tectural level Low Power 32-bit RISC Core", International Conference on Advances in Recent Technologies in Communication and Computing,2009, pp. 53-57.

[4] Douglas L. Perry, VHDL Programming by Examples, TMH.

[5] Mamun Bin Ibne Reaz, Md. Shabiul Islam, Mohd. S. Sulaiman, "**A** Single Clock Cycle MIPS RISC Processor Design using VHDL",ICSE2002 Proc. 2002, Penang, Malaysia, pp.199- 203, 2002.

[6] Kui YI, Yue-hua DING, "32-bit RISC CPU Based on MIPS Instruction Fetch Module Design", International Joint Conference on Artificial Intelligence, 2009, pp. 754-760.

[7] Xiao Tiejun, Liu Fang, "16-Bit Teaching Microprocessor Design and Application", Proceedings of 2008 IEEE International Symposium on IT in Medicine and Education, pp.160-163.

[8] Jen-Shiun Chiang and Jun-Yao Liao, "A novel asynchronous control unit and the application to a pipelined multiplier", IEEE 1998, pp. II 169-172.

[9] Ms. Aye Thi Ri Wai and Ms. Phyu Phyu Tar, "Translating a micro program to hardware control", Proceedings of ECTI-CON IEEE 2008, pp. 689-692.