International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 10, October- 2014

FPGA Based Pipelined Controller Design and Implementation

Vaibbhav Taraate ,
Senior Design Engineer RV VLSI Design Center
Bangalore Karnataka

***Abstract*— The proposed architecture is for the design, development and implementation of a 16-bit 4 stage pipelined Reduced Instruction Set Computer (RISC) IP core. In this paper it is represented as pipelined controller. It is observed that the pipelined architecture is efficient and by micro-architecture tweaking the execution time can be reduced for the pipelined instruction execution. The architecture presented in this paper is designed by using mainly three units and they are instruction memory, data memory and pipelined controller. The pipelined controller is designed by using four units and they are fetch unit, decode unit, execute unit and internal register unit. The major emphasis of this paper is to design and implement the 4 stage pipelined controller on FPGA. The pipelined controller design is implemented using Verilog RTL. In addition to the pipelined feature the design is having programmable register selections and data transfer mechanism. The pipelined controller is synthesized using Altera Quartus II and verified by using QuestaSim. The design is implemented by using Altera Cyclone II FPGA EP2C20F484C7**

Keywords—FPGA, Verilog, pipelining, Resource Sharing, Logic Duplication., Opcode, RISC

I. INTRODUCTION

The design with microcontroller has the advantages and these are reduction in overall cost due to integration of peripherals into a single chip, the product is small size, easy to troubleshoot as well as maintaining, it can be expandable by having on-chip RAM, ROM and I/O ports if required additional memories can be added externally. One of the main aspects of the architecture is the design of instruction set for the Controller. Here the proposed RISC controller which is efficient for specific and suitable applications and can be used as IP core with suitable timing signals for small applications.

The pipelined architecture executes the instructions in pipelined manner. During the first clock cycle the instruction is fetched from the instruction memory and during the second clock cycle the fetched instruction is decoded and also another instruction is fetched from the instruction memory. So it is overlapping the decode cycle of first instruction over fetch cycle of next immediate instruction. Now during the third clock cycle the first fetched instruction is executed, the second fetched instruction is decoded and third instruction is fetched. So at the end of fourth cycle one instruction will be executed.

But in non-pipelined architecture the instructions are executed in a sequence. The pipelined architecture involves clock latency but speed of design is high as compare to non-pipelined architecture. In other terms it improves overall performance of the design and throughput, which is termed as the speed of execution of the processor [5].

The paper presents very simple 16 bit 4 stage pipelined controller on FPGA. The major objective is to design the architecture, micro-architecture and implement design to perform the instructions listed in section II on FPGA for target operating frequency of 50 MHz

The paper is organized as follows. There are seven sections in this paper, section I describes the introduction, section II describes the instructions. Section III describes about the architecture, pin description, opcode and register addressing. Section IV describes about the micro-architecture and the instruction execution in pipelined manner. Section V describes the RTL Coding, synthesis and Verification aspects; Section VI describes the implementation results. Finally conclusion and future work in section VII.

II. INSTRUCTIONS

I. ***nop***: Don't perform any operation

II. ***move [source1] [destination]:*** Transfer the contents of register pointed by source1 to destination.

III. ***add [source1] [source2] [destination]***: Addition of the contents of registers pointed by source1 and source2 and stores the result in the register pointed by destination address.

IV. ***sub [source1] [source2] [destination]***: Subtraction of the contents of registers pointed by source1 and source2 and stores the result in the register pointed by destination address.

V. ***mult [source1] [source2] [destination]***: Multiplication of the contents of registers pointed by source1 and source2 and stores the result in the register pointed by destination address.

VI. ***cjeq [source1] [source2] [code***] Compare the contents of registers pointed by source1 and source2 and jump if equal to the location pointed by code.

VII. ***loadi [value] [destination]***: Load the data value in the register pointed by destination address.

VIII. ***readi [destination]*** : Read the contents of register pointed by destination address

.

III. ARCHITECTURE

The pipelined controller consists of instruction memory, data memory and pipelined controller. The architecture is evolved from the design specifications. Every Controller has the processing unit, internal register storage, and instruction memory with instruction cache.

The pipelined processor is designed to perform the operations on 16 bit binary inputs and has maximum 8 instructions. So it needs 3 bit instruction code also called as Opcode. The design has 8 internal registers of 16 bit each and to address these registers it requires 3 bit address. The processor operates on 16 bit data input and generates the 16 bit or 32 bit result depending on the operation performed.

The top level architecture is shown in the Figure 1 and it consists of pipelined Controller, instruction memory and data memory. The major emphasis of this paper is to describe the micro-architecture of pipelined controller and to implement the pipelined controller on FPGA.

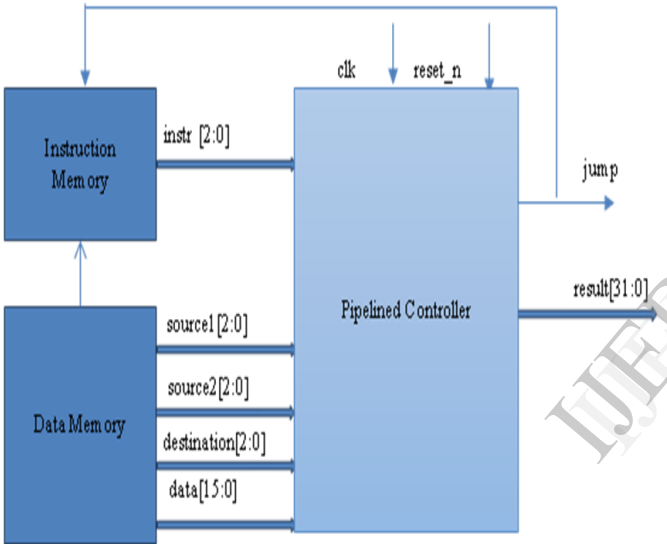

Figure 1: Architecture of Pipelined Controller

A. Pin Description

The Table 1 gives information about the pin/signals of the pipelined controller.

Sr. No.	***Pin/Signal***	***Description***
1	instr[2:0]	Three bit opcode to perform maximum 8 instructions
2	source1[2:0]	Three bit address of internal register bank
3	source2[2:0]	Three bit address of internal register bank
4	destination[2:0]	Three bit address of internal register bank
5	data[15:0]	16-bit data input to pipelined controller for load instruction
6	jump	One bit output from controller to enable jump
7	result[31:0]	32 bit output from pipelined Controller.
8	clk	Input Clock signal
9	reset_n	Active low asynchronous reset

Table 1: Pin/Signal Description

B. Instruction Definition

The Table 2 gives information about the operational code (Opcode) for the eight instructions respectively. As maximum instructions defined for the said design are 8, it uses 3-bit Opcode.

Sr. No.	Instruction	opcode
1	nop	000
2	move	001
3	add	010
4	sub	011
5	mult	100
6	loadi	101
7	readi	110
8	cjeq	111

Table 2: Opcode

C. Register Addressing

The Table 3 describes about the register addresses for source1, source2 and destination. For the design only 8 registers are used and it requires 3-bit address.

Sr. No.	Source1/source2/destination	Register Name
1	000	A
2	001	B
3	010	C
4	011	D
5	100	E
6	101	H
7	110	L
8	111	W

Table3: Register Addressing

IV. MICRO-ARCHITECTURE

The Micro-architecture for pipelined processor is derived from the architecture. The micro-architecture is detail low level abstraction of architecture. For the pipelined processor shown in Figure 1, the micro-architecture is derived to implement the required instructions with four stage pipeline. The Micro- architecture sub blocks are: fetch, decode, execute and Internal Register unit and are shown in Figure 2.

- Fetch Unit**:** It interfaces with the external instruction cache and instruction module. This module receives the 3 bit instruction opcode and data required for execution of the instruction. This unit also has interface with internal register unit. This unit also enables the communication between the internal register unit and execute unit.
- Decode Unit: It decodes the signals received from the fetch unit and passes to the execute block at the next clock cycle. For the instructions **add, sub, mult, cjea, readi** this block takes one extra clock cycle. During this it is required to pass the values from internal register block to execute block.
- Internal Register Unit: This block is used to hold the data and it consists of 8 registers **(A, B, C, D, E, H, L, W)** each of 16 bit**.** It interfaces the fetch unit with execution unit. It accepts signals from the fetch unit and drives the data required to the execute unit. After completion of execution of instruction, it also receives the signals from the execute unit.
- Execute Unit: This unit is responsible for the execution of the instruction depending on the operational code. For example for sub instruction the contents of [source1] and, [source2] are subtracted and the results are stored in internal register block pointed by [destination]

Figure2 describes the micro-architecture for the pipelined Controller [Note: It is assumed that all the units shown in the Figure 2 has synchronous clock signal 'clk' and asynchronous active low reset signal 'reset_n']

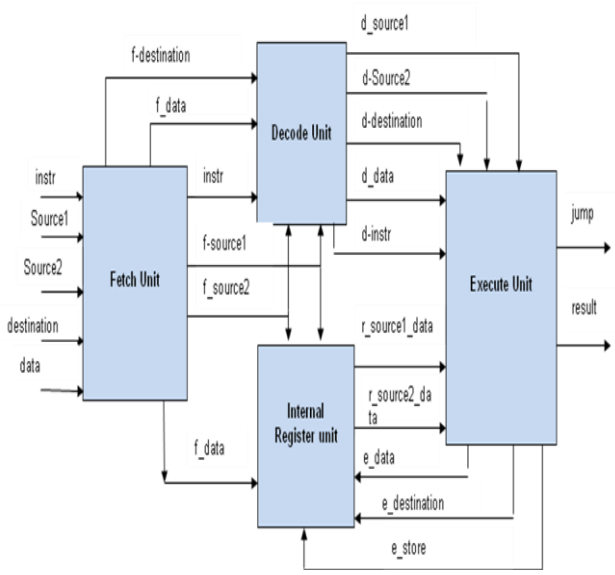

Figure 2: Micro- Architecture for Pipelined Controller

As shown in the architecture the instruction passes through fetch, decode, internal register unit and execute units. For example consider the execution of following instructions:

add B,C,A
add D,E,W
loadi FFh H
readi A

The above instructions are executed in pipelined manner as shown in Table4 below:

Clock	Fetch Unit	Decode Unit	Execute Unit	Internal Register Unit
I	add B,C,A			
II	add D,E,W	add B,C,A		
III	loadi FFh H	add D,E,W	add B,C,A	
IV	readi A	loadi FFh H	add D,E,W	add B,C,A
V		readi A	loadi FFh H	add D,E,W
VI			readi A	loadi FFh H
VII				readi A

Table 4: Instruction Execution

V. RTL CODING VERIFICATION AND SYNTHESIS

The major challenge is to write an efficient RTL Verilog code for the micro- architecture shown in Figure 2. The RTL is implemented for the individual functional block of the pipelined controller by using Verilog RTL and following are key considerations while synthesizing the design

a. Use of efficient area/resources of FPGA: For area optimization the resource sharing and logic duplication techniques are used. These techniques are proven to be very powerful while implementing the multiplier block and even for overall design [2]. Due to use of effective resource sharing the area had reduced by almost around 1% from initial device utilization value. It is observed that the resource sharing technique is very effective for the design of multiplier.
b. Use of Registered inputs and registered Outputs: The registered inputs and outputs increase the number of sequential cells but it provides the proper register to register path and encourages the pipelining. It is observed that the pipelining technique improves the speed of design at the cost of latency [3].
c. Functional Verification: The functional verification of individual block is carried out by QuestaSim for the coverage target of 98%. The overall coverage achieved is 98%.
d. Constraining Design: The design is constrained for the operating clock frequency of 50MHz and for the target area of 4% of CYCLONE II EP2C20F484C7 [4].

e. Compilation: Compilation is performed with Top-Down approach.

VI. IMPLEMENTATION RESULTS

Implementation is performed by using the Altera Quartus II for the target FPGA device Cyclone II EP2C20F484C7 [4] and following are the results shown in Table 5. For the said implementation the low power design constraints are not considered.

a.

Sr. No	Design Parameters	Result
1	Device utilization	4%
2	Timing Violation Reported	None
3	Multicycle Paths	None
4	Design Speed	50MHz
5	Data Required time	20nsec
6	Data Arrival Time	18.7nsec

Table 5: Implementation Results

VII. CONCLUSION AND FUTURE WORK

The presented architecture of 16 bit 4 stage pipelined processor is implemented on Altera Cyclone II EP2C20F484C7. The design is implemented by using Verilog hardware description language and synthesized using Quartus II . The design is verified by using QuestSim simulator. The total available logic elements in Cyclone II EP2C20 are around 18 K [4] and are sufficient to implement the design. The design just utilizes only 4% of the available logic elements for the performance of 50 MHz

The architecture can be modified by adding few additional instructions and features. The additional features includes use of stack pointer, program counter, flag registers, efficient multiply and Accumulate (MAC) unit and floating point unit. The architecture can be constrained for the low power.

ACKNOWLEDGMENT

The author would like to express sincere gratitude to Venkatesh Prasad CEO RV VLSI for constant encouragement and for providing the research set up with research environment.

REFERENCES

[1] Samir Palnitkar.1996, “Verilog HDL A guide to Digital Design and Synthesis”, SunSoft Pre
[2] D. J. Smith. (2010), “HDL Chip Design”, International Edition, Doone Publications
[3] Wayne Wolf. 2005, “FPGA Based System Design”, Prentice Hall
[4] Altera Quartus II Software documentation www.altera.com
[5] Weng Fook Lee , “ VHDL Coding and Logic Synthesis” Academic Press