Specification of the Multicycle processor:
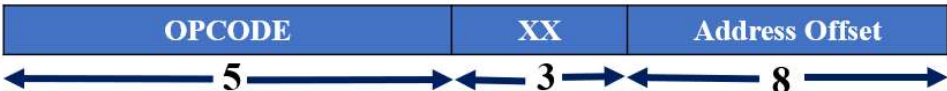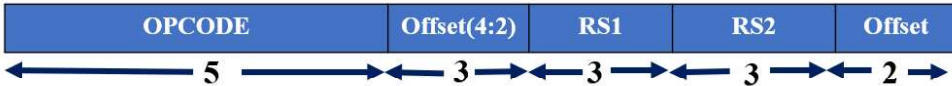
Separate data bus and instruction bus with each having 16-bit address space is assumed.
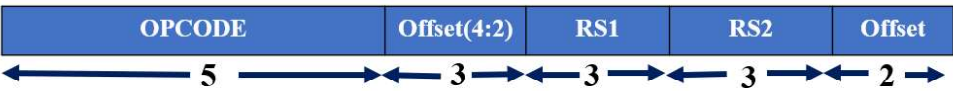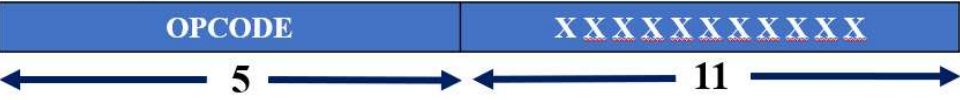
- Instruction memory:
    - Instruction size : 16 bits
    - Address line(PC) : 16 bits
- Data memory :
    - Data width   : 16 bits
    - Address width : 16 bits
- ALU
    - 16 bit ALU supporting Addition Subtraction OR, AND, XOR and Shift operations
    - ALU also generate flags required for doing conditional branch
- Register File
    - R0- R7 registers with 16 bit each
- Instruction set
  Custom designed instruction set of fixed instruction length of 16 bit supporting load, load immediate, store, conditional and unconditional branch, and Arithmetic Logic operations

Supported Instructions:

Design support total 14 instructions in total. Classification of instruction and instruction format is tabulated in the below table.

	Instruction	Opcode
R Type instructions (Arithmetic and Logic) Bit (15:11): Opcode Bit (10:8): Address of destination register Bit (7:5): Address of source 1 Register Bit (4:2): Address of source 2 Register	Add	01000
	Sub	01001
	And	01011
	OR	01100
	XOR	01010
R Type instructions (Shift left and shift right) Bit (15:11): Opcode	Shift Left	01101
	Shift Right	01110

Bit (10:8): Address of destination register
Bit (7:5): Address of source 1 Register
Bit (4:0): bit field for data amount at which the shift should occur; last 4 bits are used

Load instruction 	OPCODE | Rd | RS1 | Offset 5 — 3 — 3 — 5 Bit (15:11): Opcode Bit (10:8): Address of destination register Bit (7:5): Address of Base register for memory address Bit (4:0): 5-bit signed offset from a base register value	Load	10101

Load instruction

OPCODE	Rd	RS1	Offset
5	3	3	5

Bit (15:11): Opcode
Bit (10:8): Address of destination register
Bit (7:5): Address of Base register for memory address
Bit (4:0): 5-bit signed offset from a base register value

Load Immediate Instruction

OPCODE	Rd	DATA
5	3	8

Load Immediate will load LSB of a register with data specified in the instruction. MSB will be 0x00. Shifting and Oring with another register can be used to load any 16bit value to register
Bit (15:11): Opcode
Bit (10:8): Address of destination register
Bit (7:0): 8 bit data to be loaded to LSB

Unconditional branch instruction

OPCODE	XX	Address Offset
5	3	8

Unconditional branch instructions. This instruction jumps to PC+1±offset
Bit (15:11): Opcode
Bit (8:0): Signed offset by which PC should be changed

Conditional branch instructions

OPCODE	Offset(4:2)	RS1	RS2	Offset
5	3	3	3	2

Conditional branch instructions, branch will happen based on source register values.
Bit (15:11): Opcode
Bit (7:5): Address of source 1 Register
Bit (4:2): Address of source 2 Register
Bit (10:8) & Bit(1:0): Signed offset for branch from PC+1

Load — 10101

LoadI — 10111

Branch — 11010

Bneq (Brach not equal) — 11000

Bgt (Branch greater than) RS1>RS2 — 11001

Store Instruction	Store	11101

Store Instruction

| OPCODE	Offset(4:2)	RS1	RS2	Offset
5	3	3	3	2

Bit (15:11): Opcode
Bit (7:5): Base register address
Bit (4:2): Register to be stored
Bit (10:8) & Bit(1:0): Signed offset from base register value

Nop Instruction	Nop	00000

Nop Instruction

OPCODE	XXXXXXXXXXX
5	11

Bit (15:11): Opcode

Level-0 Design (Port Definitions)
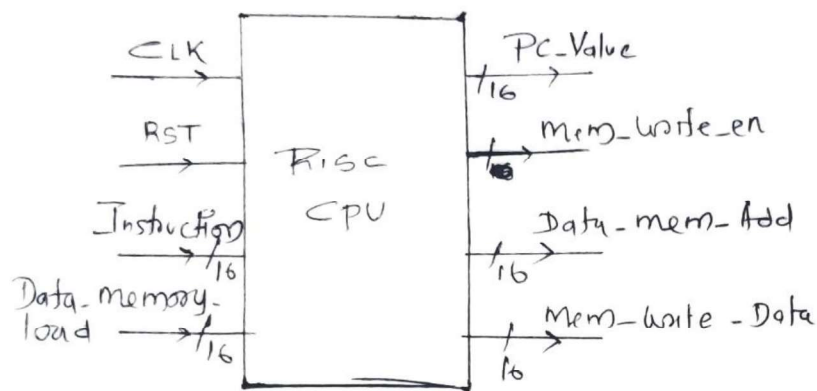
Port Name	Input /Output	Description
Clk	Input	Clock Input
Rst	Input	Reset signal
Instruction	Input	Instruction from instruction memory
Data_memory_load	Input	Data read from data memory
Pc_value	Output	Current PC value, Address to instruction memory
mem_write_en	Output	Data memory write enable signal to data memory 1 for write; 0 normally
Data_Mem_Add	Output	Address to data memory for read and write
Mem_write_Data	Output	Data to write to data memory, store operation

Detailed Datapath diagram along with control signals
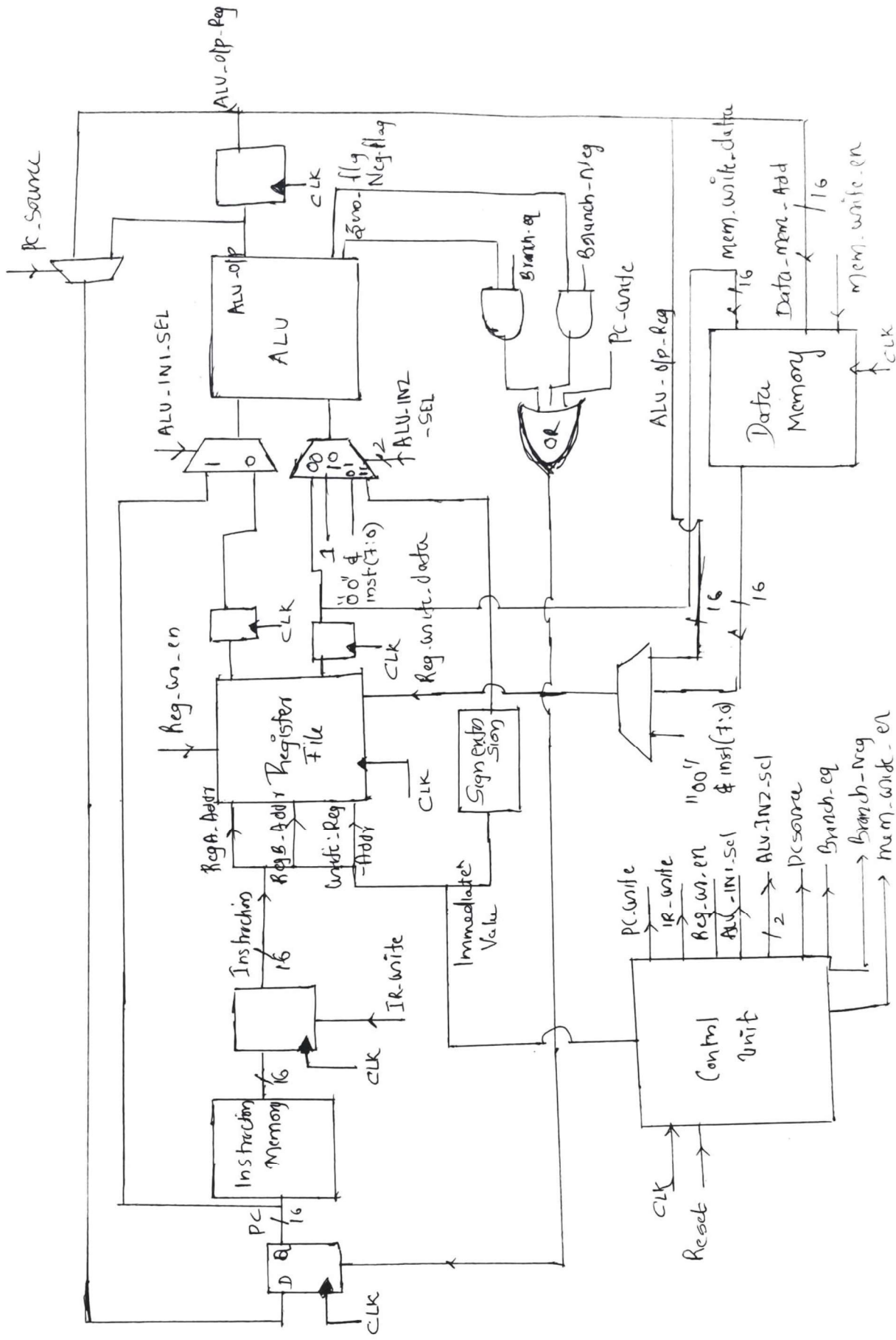
Data path Module Descriptions

1) Arithmetic Logic Unit (ALU)

Arithmetic Unit support Addition, Subtraction, bitwise OR, XOR and AND operations along with Shift right and shift left.
Arithmetic unit also generates the greater than and equal flag required for conditional branch instructions.

Port Details

Port Name	Input /Output	Description
data_in1	Input	16 bit input to ALU
data_in2	Input	16 bit input to ALU
Alu_op_sel	Input	3 bit input to ALU select the ALU function 000: Addition 001: Subtraction 010: XOR 011: AND 100: OR 101: Shift left 110: Shift right
Alu_output	Output	16 bit output to ALU
Zero_flag	Output	Zero flag for branch not equal function
Neg_flag	Output	Flag for branch grater instruction

2) Register File

Eight 16-bit registers with writing enabled on clock edge while read always transparent. One write port and two read port are provided.

Port Name	Input /Output	Description
Clk	Input	Clock input
Write_Reg_Data	Input	16 bit input data to register file
RegA_Addr	Input	3 bit register address for reading through Port A
RegB_Addr	Input	3 bit register address for reading through Port B
Write_Reg_Addr	Input	3 bit register address for writing
Write_En	Input	Register write enable signal

RegA_Out	Output	Register file PORT A output
RegB_Out	Output	Register file PPORT B output

3) Data_path Entity

This module integrates all the data path elements. Input output description of datapath module is given below.

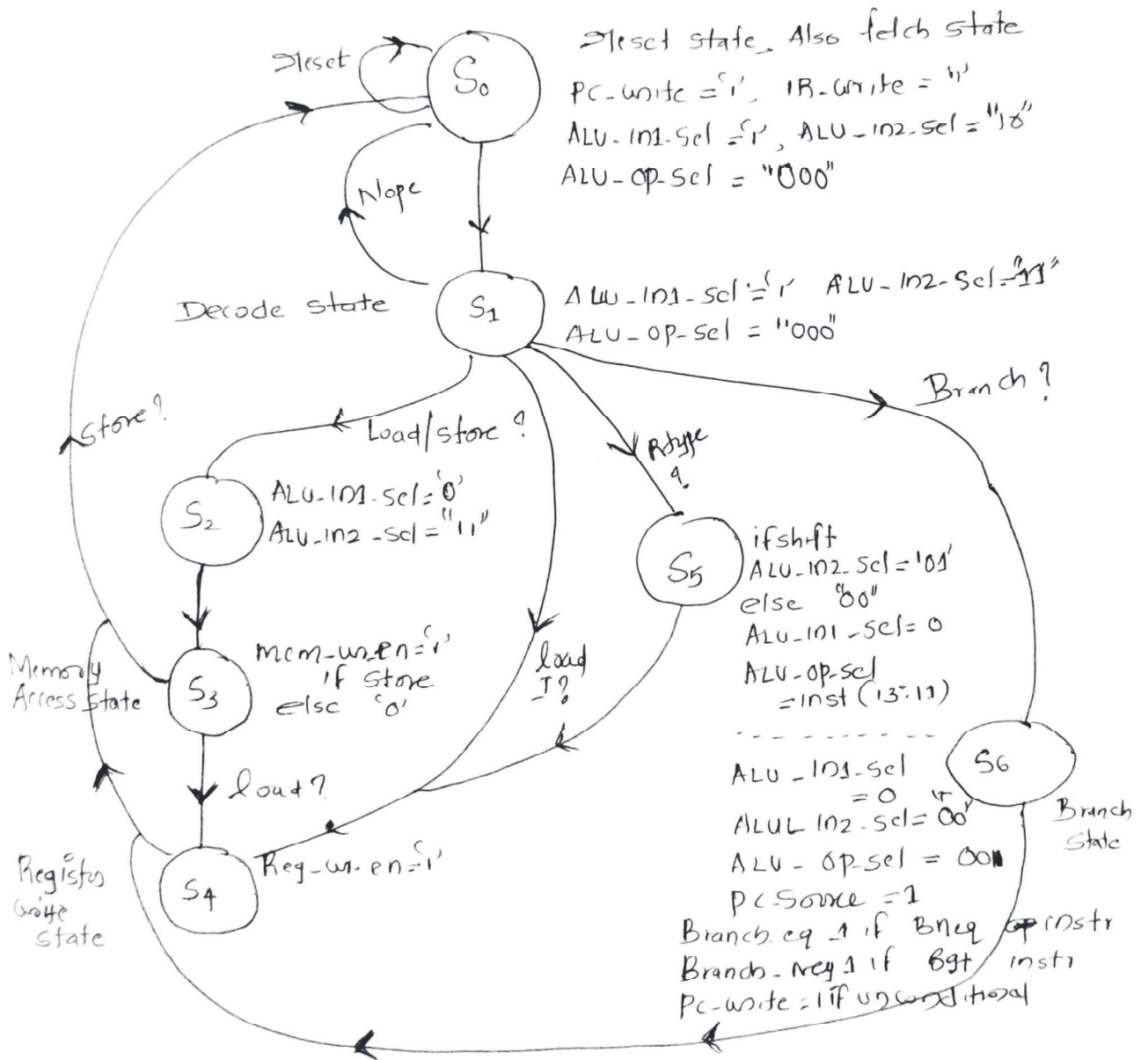| Port Name	Input /Output	Description
Clk	Input	Clock input
Reset	Input	Reset input
Instruction	Input	Registered Instruction from IM
PC_write	Input	Control Signal from the Control unit Signal will enable PC register updation
Reg_Wr_en	Input	Control Signal from the Control unit Signal will enable register file write
ALU_In1_Sel	Input	ALU input1 MUX selection line
ALU_In2_Sel	Input	ALU input2 MUX selection line
ALU_OP_SEL	Input	ALU function selection
PC_source	Input	PC update mux selection line
Branch_eq	Input	Indicate a bneq instruction being executed. Will and with zero flag and use to update the PC value
Branch_Neg	Input	Indicate a branch greater instruction being executed. Will and with neg flag from ALU and use to update the PC value
ALU_Output_Add	Output	Address to data memory
Write_Data	Output	Write Data to data memory
PC_value	Output	Current PC value to Instruction memory

4) Control Unit

Control Unit is a finite state machine which control the over all flow of instruction fetching, decoding and executing. Based on the instruction type number of clock cycles can vary. No of clock cycle for each instruction is tabulated in later section. This module generate the required control signals for the Datapath.

Port details of the control unit is given below.

| Port Name	Input /Output	Description
Clk	Input	Clock input
Reset	Input	Reset input
Instruction	Input	Registered Instruction from IM
PC_write	Output	Signal will enable PC register updation.
Reg_Wr_en	Output	Signal will enable register file write
ALU_In1_Sel	Output	ALU input1 MUX selection line 0 : Register input 1: PC value
ALU_In2_Sel	Output	ALU input2 MUX selection line 00: Register value 01: shift value 10: Load Immediate 11: Sign extended offset
ALU_OP_SEL	Output	ALU function selection
PC_source	Output	PC update mux selection line 0: Alu output 1: registered ALU output
Branch_eq	Output	Indicate a bneq instruction being executed.
Branch_Neg	Output	Indicate a branch greater instruction being executed.
IR_Write	Output	IR write enable signal
Mem_Write_en	Output	Data memory write enable 1 for write 0 for read

State Diagram of the Controller:

State diagram showing states S0 through S6:

S0 — =reset state, Also fetch state. PC-write = '1', IR-write = '1'. ALU-In1-Sel = '1', ALU-In2-Sel = "18". ALU-Op-Sel = "000". Self-loop labeled =reset.

S1 — Decode state. ALU-In1-Sel = '1', ALU-In2-Sel = "11". ALU-Op-Sel = "000". Nope loop back to S0.

Branch? → S6.

Load/store? → S2. ALU-In1-Sel = '0', ALU-In2-Sel = "11".

Rtype? → S5.

Store? path back.

S3 — Memory Access State. mem-wr-en = '1' if Store else '0'.

S5 — ifshift ALU-In2-Sel = '01' else "00". ALU-In1-Sel = 0. ALU-Op-Sel = Inst (13:11).

load I? / load I?

S4 — Register Write State. Reg-wr-en = '1'. load?

S6 — Branch State. ALU-In1-Sel = 0 & ALUL-In2-Sel = 00. ALU-Op-Sel = 00. PC Source = 1. Branch-eq 1 if Bneq or instr. Branch-neg 1 if Bgt instr. PC-write = 1 if unconditional.

No of clock cycle for each type of instruction

Instruction	No of clock cycle
Load Instruction	5
Load Immediate Instruction	3
Store Instruction	4
R type Instruction	4
Nope Instruction	2
Branch instruction	3

Testing and Simulation Results

To test the multicore CPU, Instruction Memory and Data Memory was instantiated using Xilinx IPs and loaded with appropriate values using .coe files
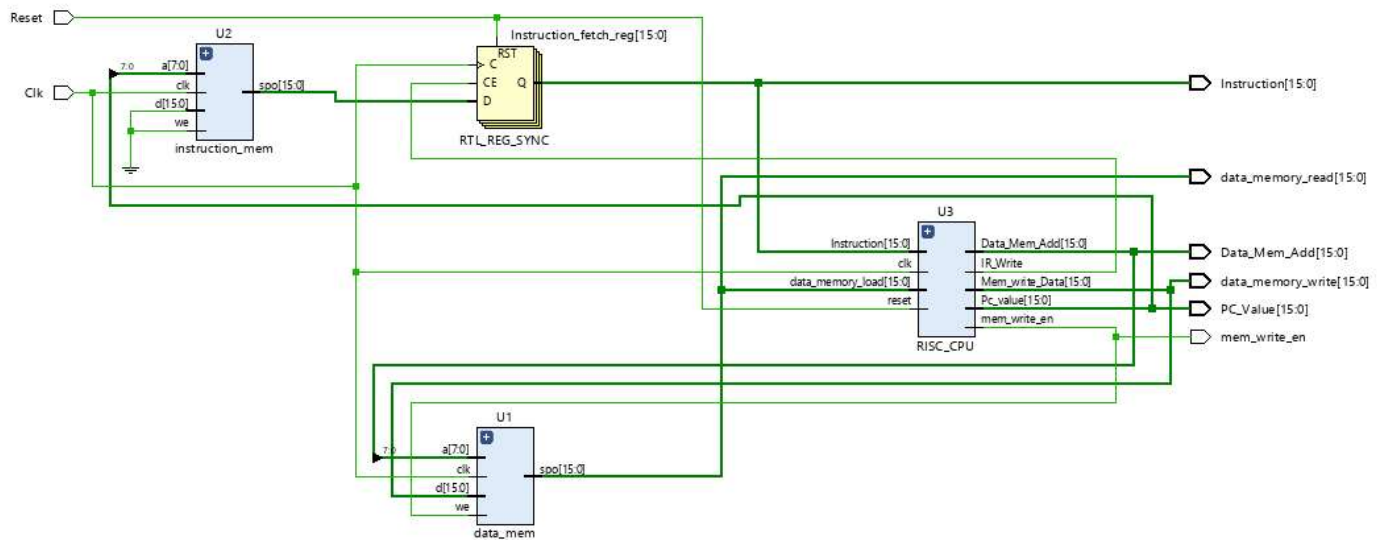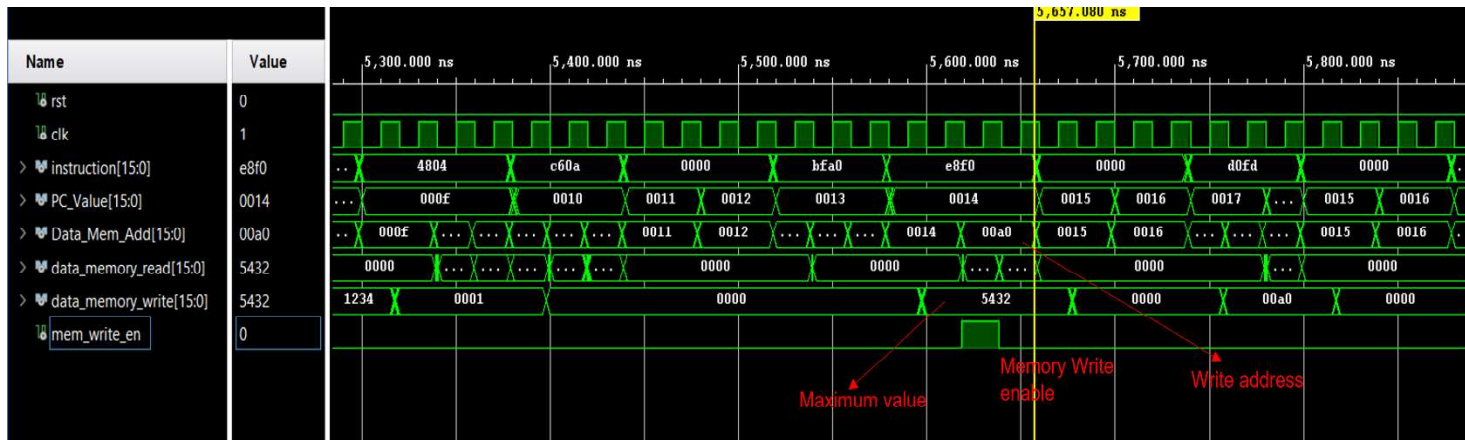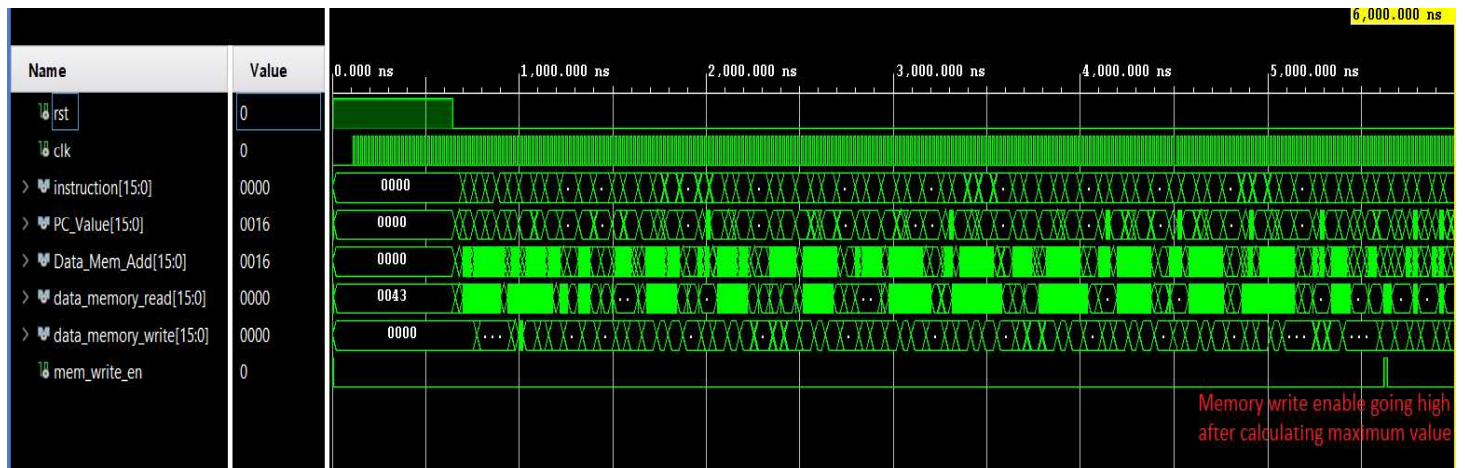

256*16bit data memory and Instruction memory is instantiated and LSB of the PC and Data memory address is mapped or connected for simulation purpose

A program for finding the maximum number from 10 random number stored in data memory was used to verify the performance. Code is given below

```
0: Nope                        0x0000
1: LOADI R0, 0x0A              0xB80A
2: LOADI R1, 0x01              0xB901
3: LOADI R2, 0x00              0xBA00
4: LOADI R3, 0x00              0xBB00
5: LOADI R7, 0xFF              0xBFFF
6: ShiftLeft R7,R7,0x08        0x6FE8
7: LOADI R6,0xFF               0xBEFF
8: OR R7,R7,R6                 0x67F8
9: LOAD R4,0x00(R3)            0xAC60
10:ADD R3,R3,R1                0x4364
11: LOAD R5,0x00(R3)           0xAD60
12:BGT R4,R5,0x01              0xC895
13: AND R4,R5,R7               0x5CBC
14: SUB R0,R0,R1               0x4804
15: BEQ R0, -6(R2)             0xC60A
16:NOP                         0x0000
17: NOP                        0x0000
18:LOADI R7,0xA0               0xBFA0
19: STORE R4,0x00(R7)          0xE8F0
20: NOP         ---label2      0x0000
21: NOP                        0x0000
22: Branch  label2             0xD0FD
```

Program will find out the maximum value and store the result to memory location 0xA0 at the end.

Post Implementation Timing simulation

Memory write enable going high
after calculating maximum value

Maximum value
Memory Write enable
Write address

Resource utilization

Resource utilization including of CPU, Data Memory and Instruction memory are given below.

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Bonded IOB (106)	BUFGCTRL (32)
∨ N RISC_TOP	388	122	66	32	124	236	152	83	1
> Ⅰ U1 (data_mem)	64	0	32	16	16	0	64	0	0
> Ⅰ U2 (instruction_mem)	64	0	32	16	16	0	64	0	0
> Ⅰ U3 (RISC_CPU)	260	91	2	0	83	236	24	0	0

Percentage Utilization of FPGA resources including Data memory and Instruction memory

| Resource	Utilization	Available	Utilization %
LUT	388	20800	1.87
LUTRAM	152	9600	1.58
FF	122	41600	0.29
IO	83	106	78.30

Clock constraint was given as 100MHz. Constraint file is attached in submission folder.

| Specification	Value
Worst Negative Slack (WNS)	1.064ns
Worst Hold Slack (WHS)	0.122ns
Setup between clocks (minimum clock period)	8.936ns Max freq=111.9MHz
Input port to setup	1.191ns (worst case)

Implementation2

FSM of control unit was modified in 2nd implementation such that no of clock cycle taken by the instruction except Branch and NOPE will be reduced by one cycle. This is done by latching the Instruction Memory output to IR in the register write state and avoiding an explicit state for fetching. Additional states are introduced in FSM for this purpose.
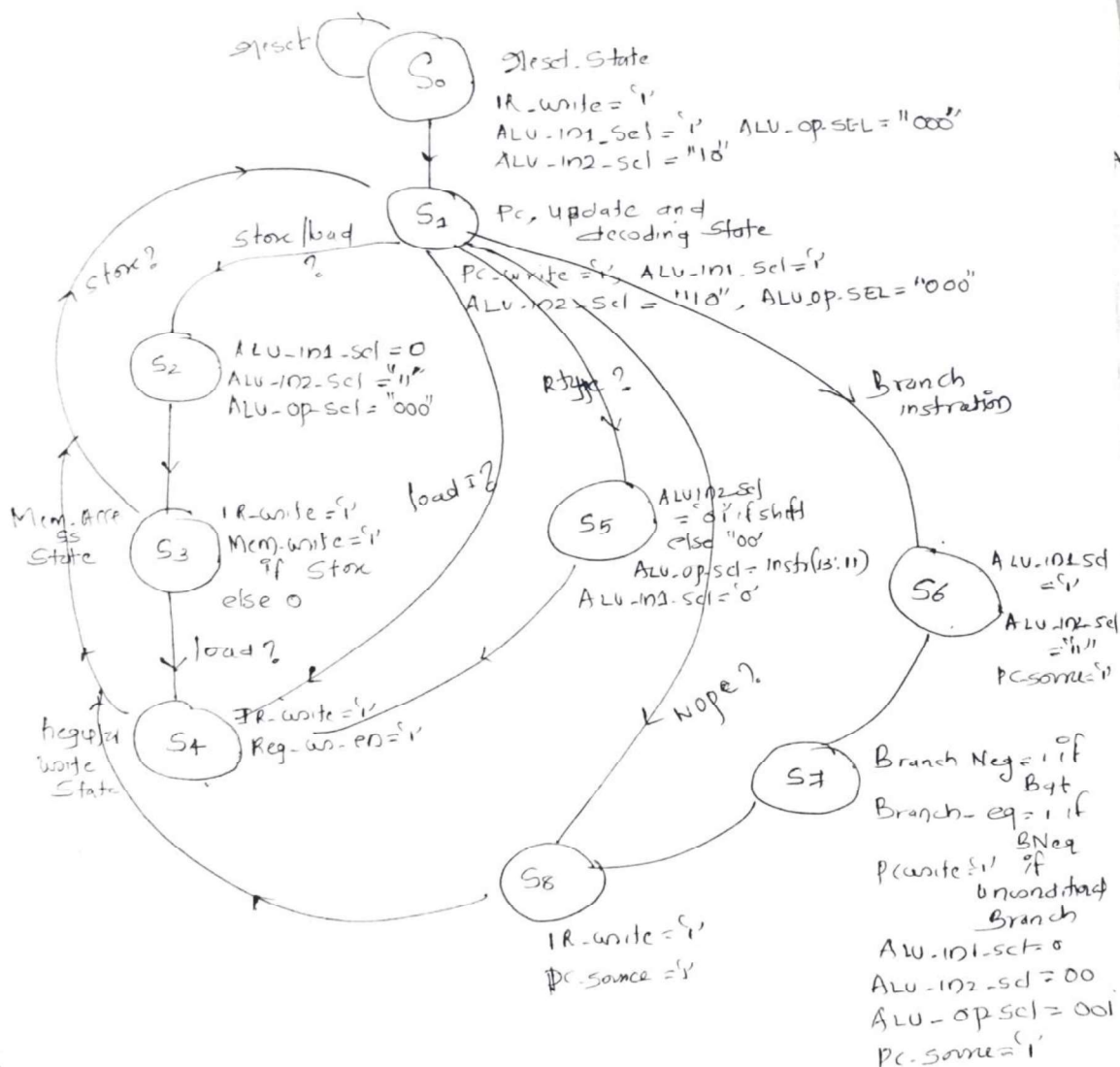
No of clock cycle for each type of instruction in modified implementation

Instruction	No of clock cycle
Load Instruction	4
Load Immediate Instruction	2
Store Instruction	3
R type Instruction	3
Nope Instruction	2
Branch instruction	4

Post Implementation Timing simulation
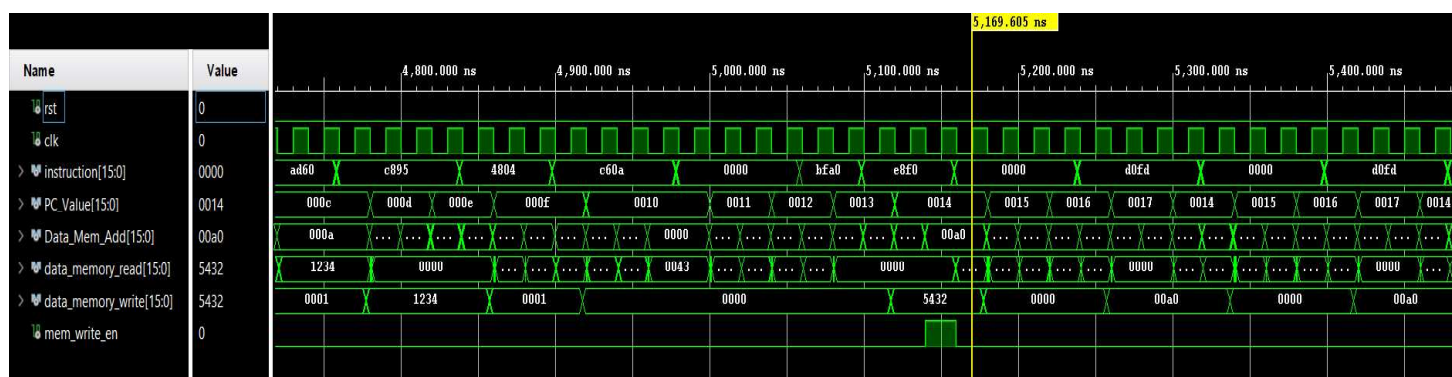
Resource utilization

Name	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	F8 Muxes (8150)	Slice (8150)	LUT as Logic (20800)	LUT as Memory (9600)	Bonded IOB (106)	BUFGCTRL (32)
N RISC_TOP	383	121	78	32	113	231	152	83	1
U1 (data_mem)	64	0	32	16	16	0	64	0	0
U2 (instruction_mem)	64	0	32	16	16	0	64	0	0
U3 (RISC_CPU)	255	94	14	0	72	231	24	0	0

Resource	Utilization	Available	Utilization %
LUT	383	20800	1.84
LUTRAM	152	9600	1.58
FF	121	41600	0.29
IO	83	106	78.30

Timing Summary: -

Specification	Value
Worst Negative Slack (WNS)	0.425ns
Worst Hold Slack (WHS)	0.072ns
Setup between clocks (minimum clock period)	9.575ns Max freq=104.5MHz
Input port to setup	0.521ns (worst case)