16bit Processor with Memory

Anisha Sharma

B20029@students.iitmandi.ac.in

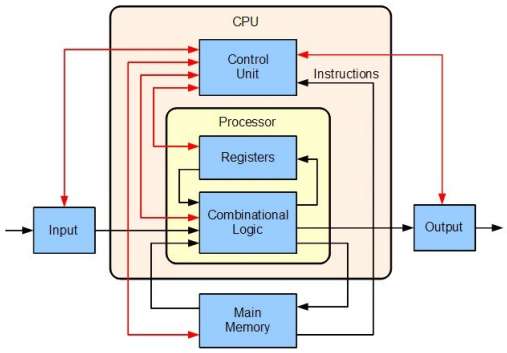
1. Aim :-

The main objective of this lab is to understand and implement a simple processor in VHDL with the following functionalities and instructions

Operation	Function
mv Rx, Ry	Rx←[Ry]
mvi Rx, #D	Rx←D
add Rx, Ry	Rx←[Rx]+[Ry]
sub Rx, Ry	Rx←[Rx]-[Ry]
and Rx, Ry	Rx←[Rx]Λ[Ry]
or Rx, Ry	Rx←[Rx]V[Ry]
xor Rx, Ry	Rx←[Rx]⊕[Ry]
not Rx	Rx←¬[Rx]

2. Theory :-

The main theory behind this working of the simple processor is as below.



We can see that there are registers the controlling unit and ALU unit in the CPU. Registers just simply store the data and output the data when the inputs are triggered. Control unit controls the state of a register whether to load data or dump the data to bus. It takes use of multiplexer to control the switching of required register among the rest that is which register to switch. ALU simply performs the operation.

Registers and control unit need the clock to trigger i.e. they are synchronous circuits, ALU and demultiplexer doesn't require any clock (they are combinational circuits.)

For implementing this simple processor given in the lab, these are the list of components required

1. Register (16bit) – R0 to R7,A,G
2. Decoder/rev Mux (1:16 with 4 select lines)
3. Control Unit (FSM)
4. ALU (Combinational logic) The

   mechanism is as below:

Register: This can store the data when the clock transition from 0 to 1 triggers and the load is high, and gives the output when the enable is high. There we can connect a not gate to load pin and connect these 2 combinedly naming the mode pin now the TT is as below

Mode	Function
1	Dump to output
0	Load from input

This mode pin is controlled by the decoder.

Also let us assign some addresses to these registers

Register	Address
R0	0000
R1	0001
R2	0010
R3	0011
R4	0100
R5	0101
R6	0110
R7	0111
DIN	1000
A	1001
G	1010
IR	1011

Decode:

The decode activates the required register and send the value from CU to Reg.

The table for register selection with the 4 select lines is as below

Address	16 bit output
0000	ZZZZZZZZZZZZZZZ"&val”
0001	ZZZZZZZZZZZZZZ"&val&"Z
0010	ZZZZZZZZZZZZZ"&val&"ZZ
0011	ZZZZZZZZZZZZ"&val&"ZZZ
0100	ZZZZZZZZZZZ"&val&"ZZZZ
0101	ZZZZZZZZZZ"&val&"ZZZZZ
0110	ZZZZZZZZZ"&val&"ZZZZZZ
0111	ZZZZZZZZ"&val&"ZZZZZZZ
1000	ZZZZZZZ"&val&"ZZZZZZZZ
1001	ZZZZZZ"&val&"ZZZZZZZZZ
1010	ZZZZZ"&val&"ZZZZZZZZZZ
1011	ZZZZ"&val&"ZZZZZZZZZZZ

Here the “val” is the mode of the register i.e. which controls the register to load in or dump out the data.

Control Unit:

For the given operations we can tabulate the steps as below (Collectively for all the instructions)

State	Process
0	DIN -> BUS
1	BUS -> IR
2	DIN -> BUS
3	BUS -> Rx
4	Rx -> BUS
5	BUS -> A
6	Ry -> BUS
7	f(A, BUS) -> G
8	G -> BUS
9	BUS -> Rx

Now the instructions can be realised as below

mv – 0,1,6,9 mvi – 0,1,2,3

all the rest – 0,1,4,5,6,7,8,9

Now for every state we can perform the required operation accordingly by just giving the signal to addr and val to de-mux. Defining a simple FSM for this control unit serves the purpose.

The format of instructions are as given in the question. “IIIXXXYYY” – Lower 9 bits

ALU:

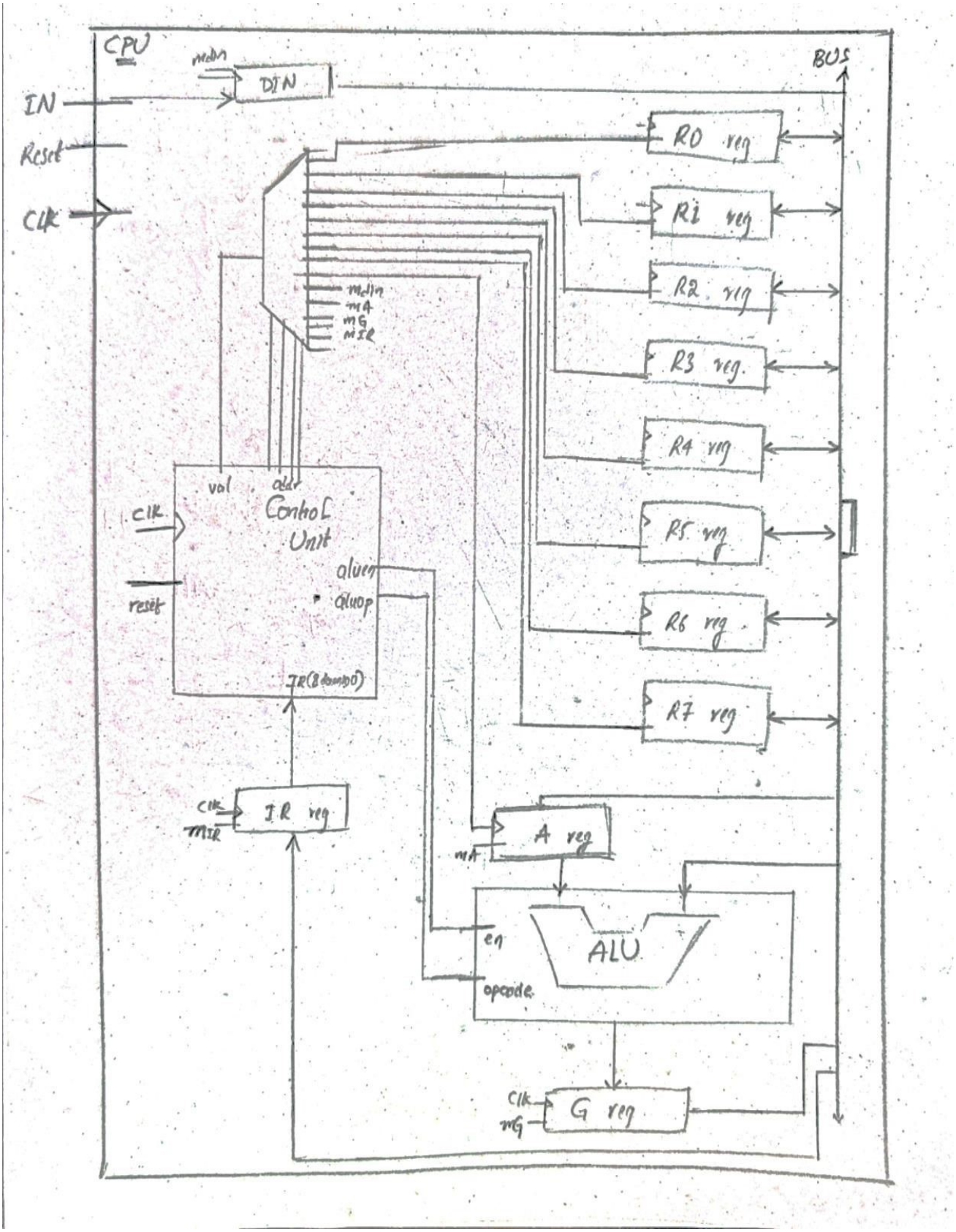
This takes the inputs, opcode. According to the opcode it performs the operation and dumps out the result.

For this project the opcodes are considered as below because they will directly correspond to the instruction at required state.

Opcode	Operation
010	Add
011	Subtract
100	AND
101	OR
110	XOR
111	Not

3. Circuit Diagram:-

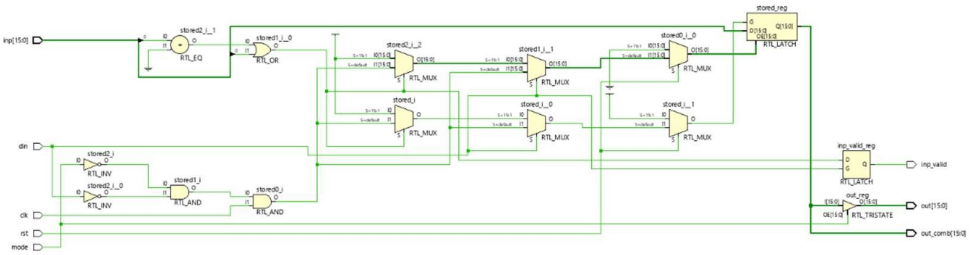
The circuit / architecture can be realised as follows



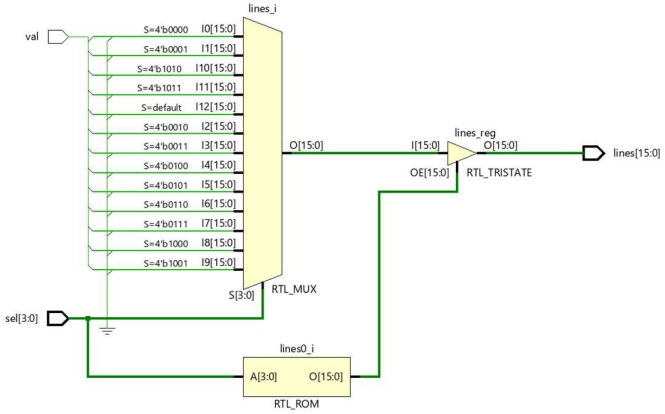
Note: The schematics and implementation designs are attached at the Discussion section.

4. Designs:

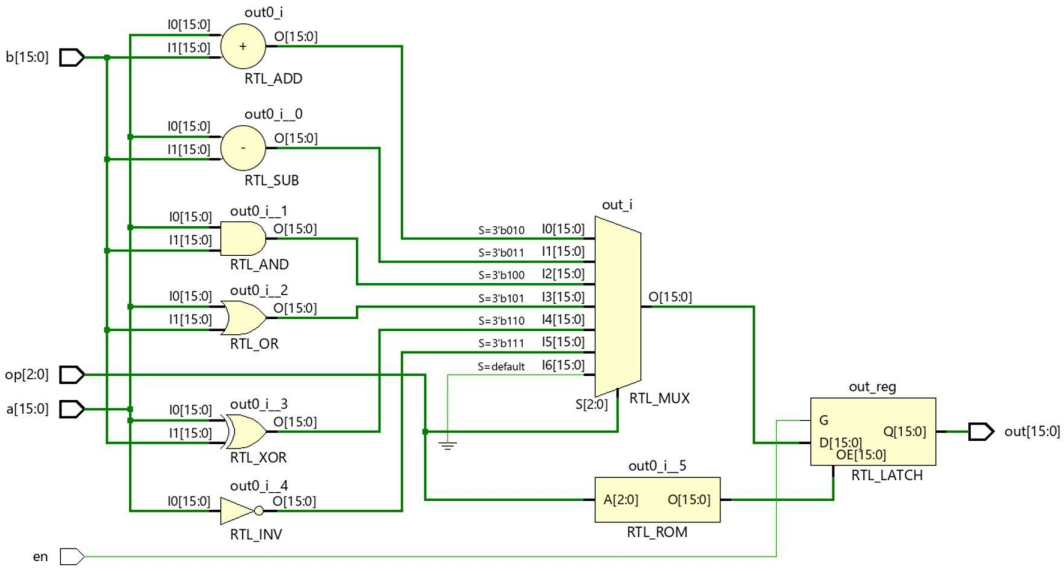
a. Register



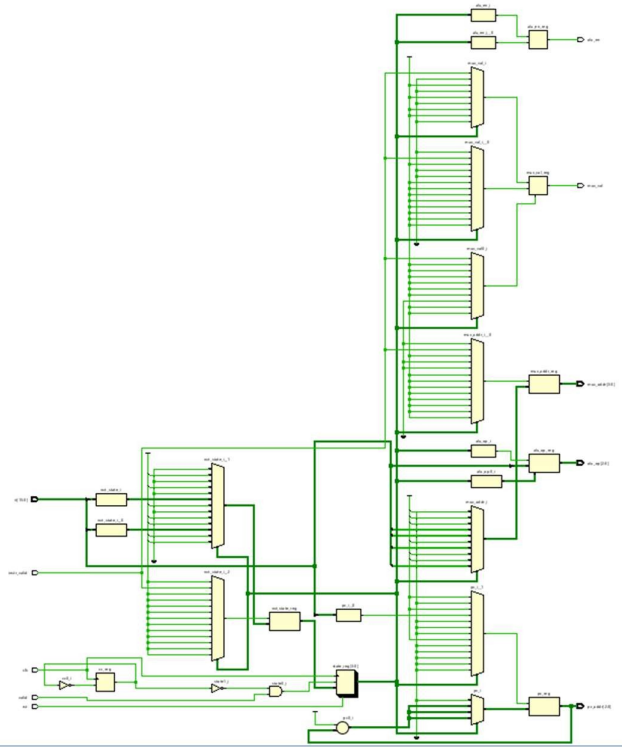
b. MUX



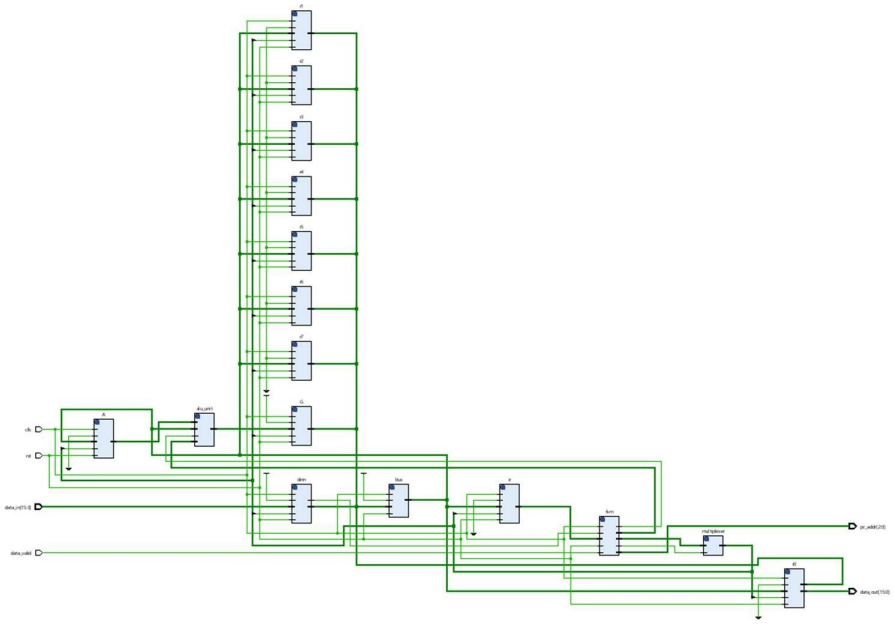
c. ALU



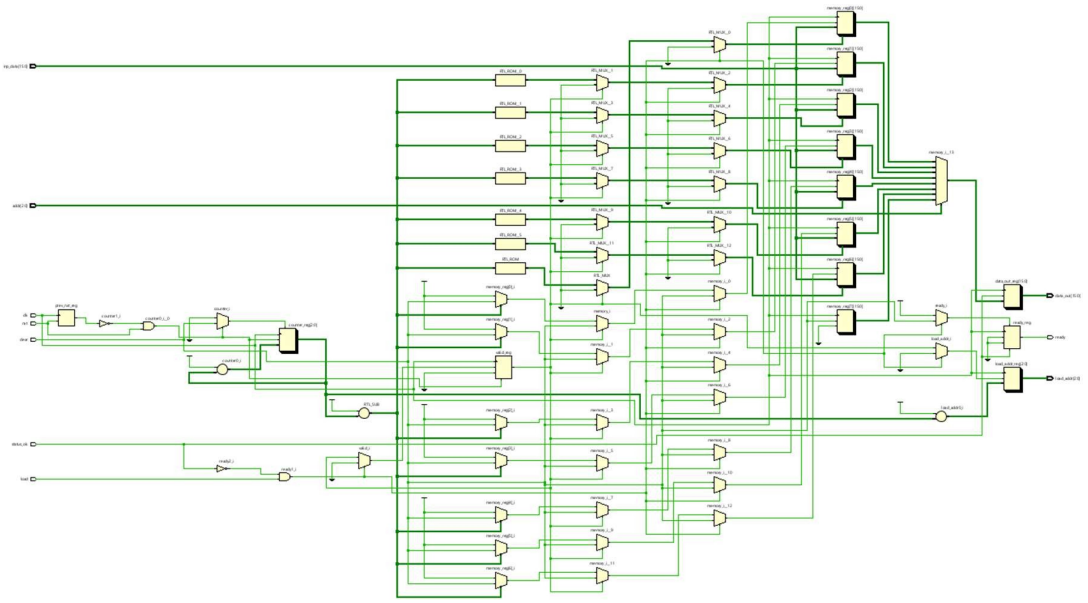
d. FSM



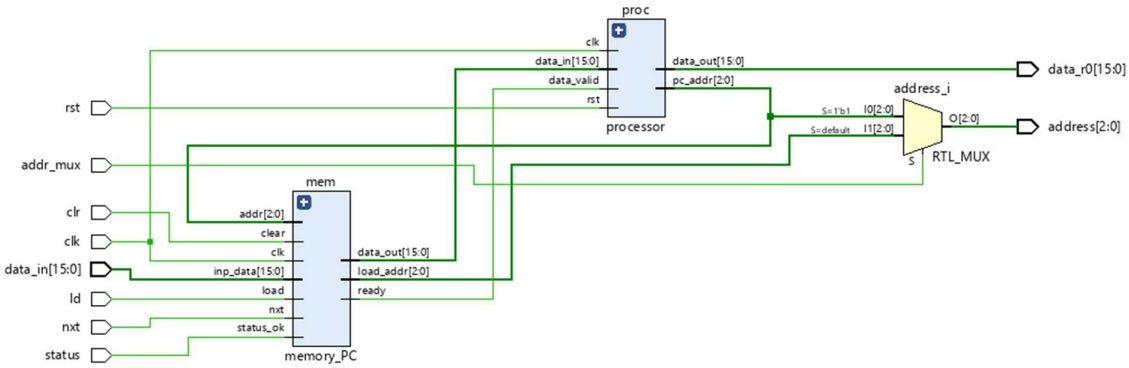
e. Processor



f. Memory

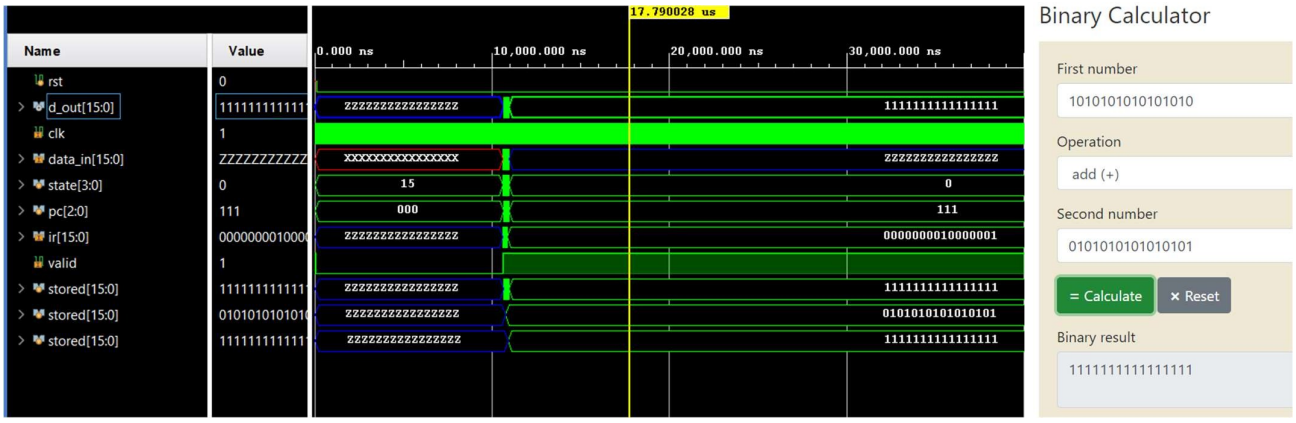


g. Wrapper



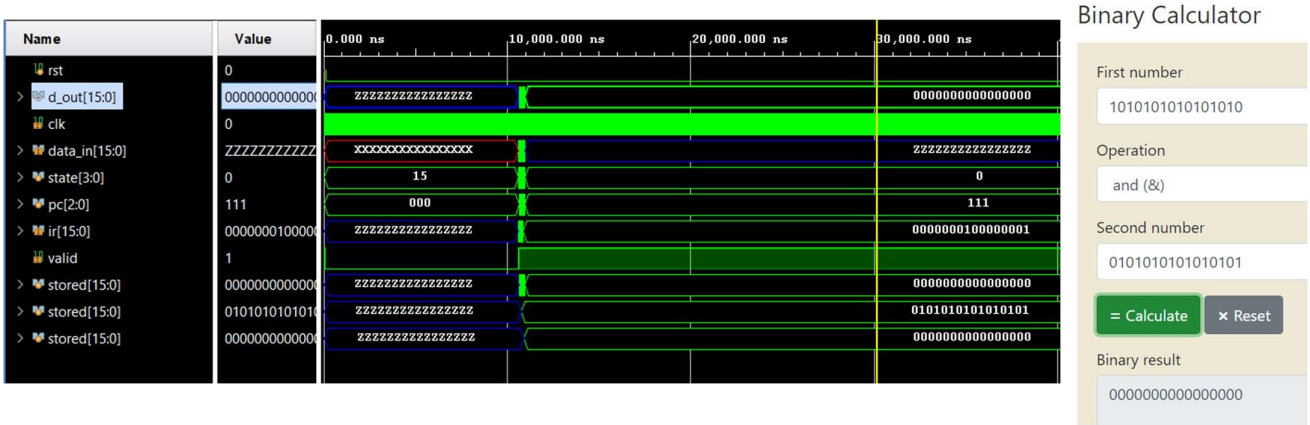
5. Simulations & Results

a. ADD (+mvi, mv)

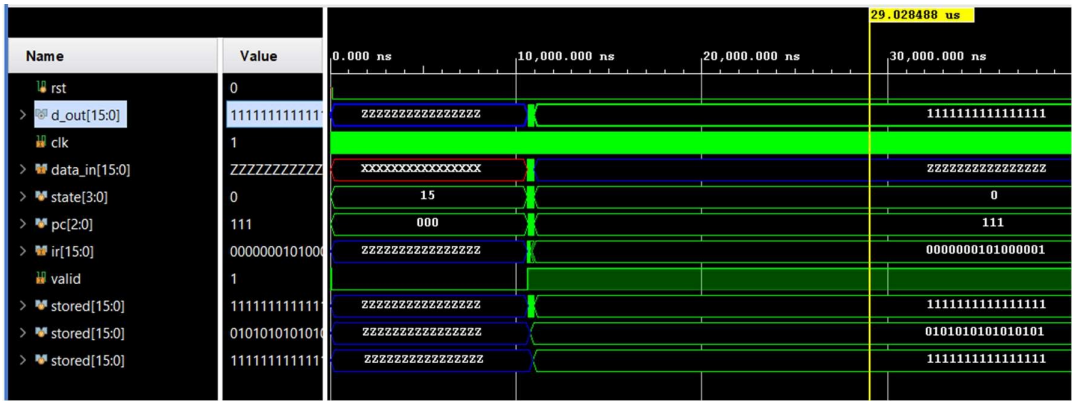


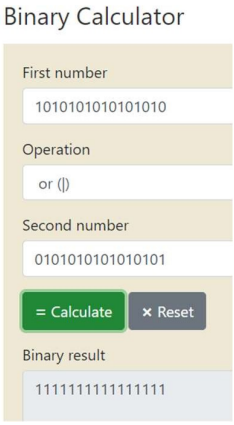
b. SUB

c. AND

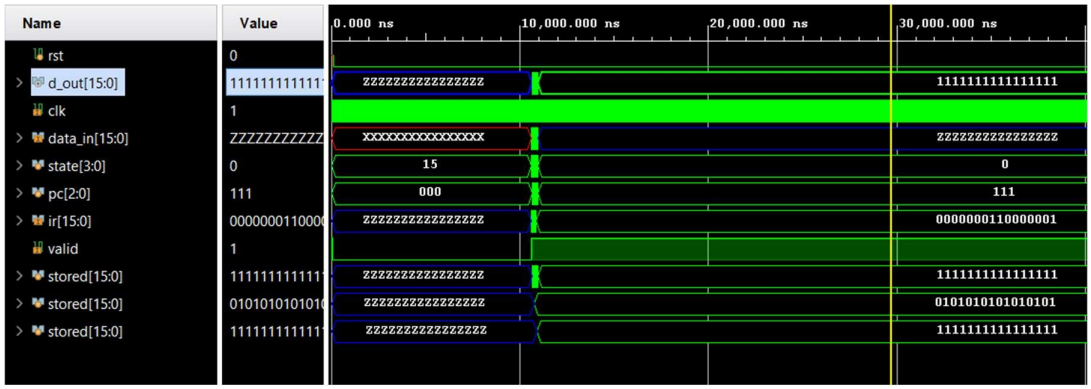


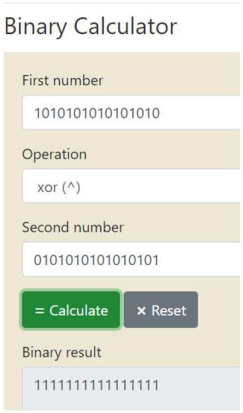
d. OR



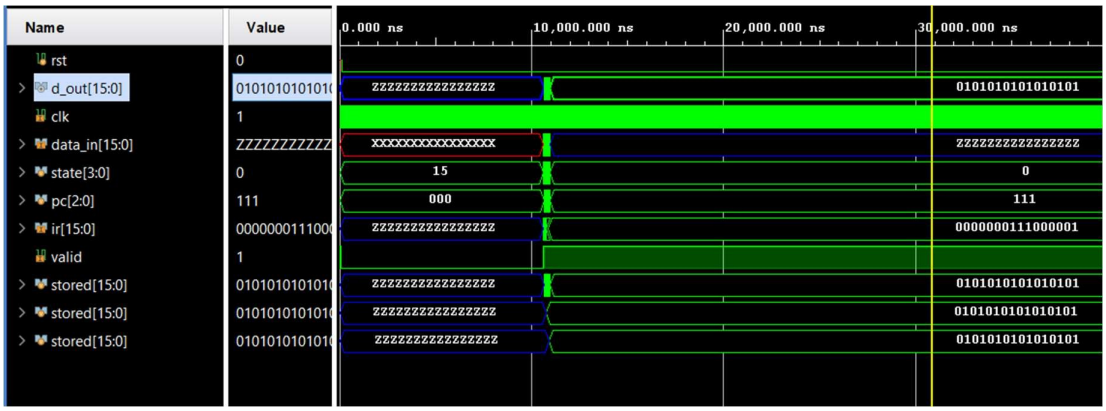


e. XOR





f. NOT



6. Timing Analysis:-

Considering the given delays we have the effective delays as follows:

MV, MVI – 282ns
Operations – 428ns

Note:

1. Please find the attached file for the delay calculations.
2. These are not included in simulations as the FPGA has differentstructure of execution which is using LUTs

7. Inferences of Discussion:

From the testbench results (inferences of clock cycles and correctness ofprocessor)

- We can verify the correctness of each operation. The screenshot attachednext to code is taken from the calculator. We can observe that our result matches.
- For the mv, mvi operations it took 8 clock cycles to complete. (As we usedhalf frequency clock for FSM and normal frequency clock for registers. So, to complete 4 states in FSM it takes 8 clock cycles)
- For the rest of operation which require the ALU, it took 8 clock cycles ofFSM i.e. 16 clock cycles to complete.

8. Conclusion:-

By this lab project we are able to understand the complete working of a simpleprocessor and able to implement it in VHDL.

(And also

- Compared and verified our results of Operations
- Analysed the number of required clock cycles and triggering.)

9.References:-

[1] https://www.partitionwizard.com/partitionmanager/cpu-architecture.html

The screenshots of Binary calculator are taken from https://www.rapidtables.com/calc/math/binary-calculator.html

All the rest of the images are of my own simulations and drawings .