TO DESIGN 16-BIT MICROPROCESSOR USING VHDL

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

in

Electronics and Communication Engineering

MAY-2014

Under the Supervision of

Mr. Akhil Ranjan

By

Ayush Bajpai (101028)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

CERTIFICATE

This is to certify that the work titled "**DESIGN OF 16-BIT MICROPROCESSOR USING VHDL**" submitted by **Ayush Bajpai** in partial fulfilment for the award of degree of **B.Tech Electronics and Communication** of **Jaypee University of Information Technology, Waknaghat** has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	……………………..
Name of Supervisor	……………………..
Designation	……………………..
Date	……………………..

ACKNOWLEDGEMENT

I express sincere appreciation to Mr. Akhil Ranjan for his guidance throughout the research and preparation of the thesis. I would like to thank him for his helpful comments, suggestions and giving me chance to work together.

Signature of the student	……………………..
Name of Student	……………………..
Date	……………………..

TABLE OF CONTENTS

Abstract

The aim of this project is to design and simulate a 16 bit processor. The design has been implemented using VHDL synthesis tool Xilinx 9.2i. Microprocessor is basically an electronic device that consists of ALU and control circuitry which is required to function as computer’s CPU. Microprocessor is integrated circuit that interprets and executes the program instructions and behaves intelligently. The processor operates at a speed of the internal clock and the speed of the clock depends upon the no. of pulses per second. With each clock pulse, the processor performs the function that corresponds to the instruction. Thus the power of the processor can be calculated by no. of instructions executed per second. During the execution of instructions, data are stored temporarily in memory units called registers. The control signal is the electronic signals used for communication between various processor units during the execution of the instruction. This report describes all the sections of microprocessor briefly using flowchart explained by a brief explanation and showing the output waveform of different blocks.

LIST OF FIGURES

LIST OF TABLES

<u>Symbols and abbreviations</u>

1) VHDL :- Very high speed integrated circuit Hardware Description Language
2) RAM :- Random Access Memory
3) ALU:- Arithmetical Logical Unit
4) ROM:- Read Only Memory
5) Bits :- Binary Digits
6) RTL:- Register Transfer Level

1. INTRODUCTION

The main aim of this project is to design 16 bit MICROPROCESSOR using VHDL (Very high speed Integrated circuit Hardware Description Language). While designing such a project it is better to have an idea about the functionality of the project as in such cases we need not to make a general system. This will limit both cost and the time required to develop the system. This is the basic idea behind the project. A Microprocessor basically contains a RAM, ALU, and Control unit to control the flow of data.

Before designing a microprocessor we should have a brief knowledge about the flow of data, the way in which data is to be processed after doing a particular processing on data. If we will have knowledge about the flow of data then it would be easy for us to design the system and map the output of one block to the input of other or to data bus depending upon the flow of data.

After describing a digital system in VHDL, we simulate and synthesize VHDL code for two reasons. First, we need to verify the VHDL code correctly implements the intended design; second, we need to verify that design meets its specifications. Before the VHDL model of digital system can be simulated, the VHDL code must first be compiled (see Figure 1). The VHDL compiler also called as an analyzer, first checks the VHDL source code to see whether it conforms to the syntax and semantic rules of VHDL. If there is a syntax error, a missing semicolon, or if there is a semantic error such as trying to add two signals of incompatible types, the compiler will output an appropriate error message. The compiler also checks to see that references to libraries are correct. If the VHDL code conforms to all the rules, the compiler generates intermediate code, which can be used by a simulator or by a synthesizer.

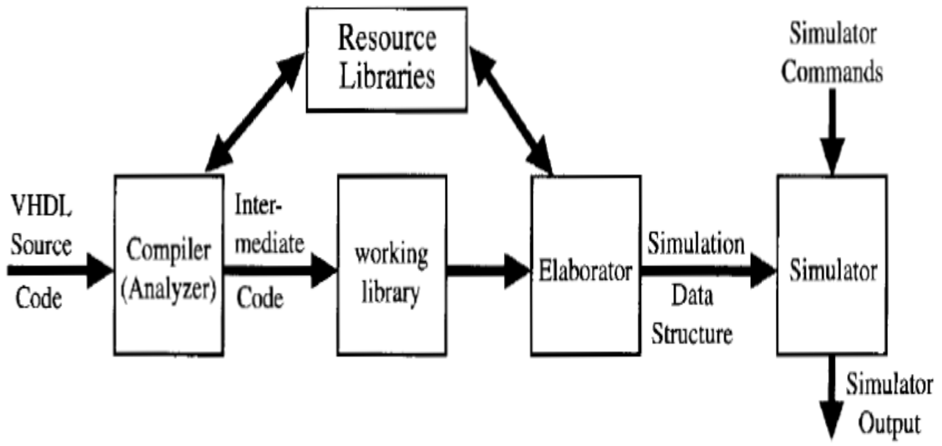


Figure 1:Compilation, Elaboration, and Simulation of VHDL code

In preparation for simulation, the VHDL intermediate code must be converted to a form that can be used by the simulator. This step is referred to as elaboration (synthesis). During the elaboration, ports are created for each instance of a component, memory storage is allocated for required signals, the interconnections among the ports signals are specified, and a mechanism is established for executing the VHDL in the proper sequence. The resulting data structure represents the digital system being simulated. After an initialization phase, simulation enters the execution phase.

With some idea on the design methodology and deciding on the simulation and synthesis tool, one can come with a complete design methodology for the project. The design process begins with architectural specifications, which are then translated into a VHDL behavioral model. This behavioral model is tested using VHDL test bench or command language of the simulator. After satisfactory verification of the microprocessor Control block, the behavioral code is synthesized to obtain gate level net list to prepare the layout and floor representation of the design.

2. Proposed methodology

This Microprocessor basically contains a RAM, a comparator, a shifter, an ALU block, registers, a control unit. This is a 16-bit microprocessor because the data on which processing is to be done is of 16-bit. Data bus is a 16-bit signal on which the data is written. The data is written in data bus during processing. Address bus in this case is 16-bit which is used only to signify the address of the next instruction to be performed by the microprocessor.

2.1 RAM

This microprocessor contains 8 blocks in RAM each of 16 bit. A select line that selects the block which is required for a particular operation. The size of RAM can also be varied based upon our requirements by simply adding more blocks and by changing the size of select line. The RAM block contains a select line which is of 3-bit which is basically used to select the Register block on which the data is to be written or from which it is to be read from.

2.2 Comparator

A comparator is a block that checks out of two data which one is greater than other, or less than other, or whether they are equal to each other. The operation to be performed by comparator block is also selected by its select line.

2.3 ALU

As the name suggests any set of arithmetical or logical operation is performed by this block. More set of instruction can again be added to this block depending upon the requirements.

2.4 Registers

This is basically a set of 16-bit registers to store data in the mid way of calculation or any processing of data .

2.5 Control block

This is a major block of the project as this particular block controls the flow of the data in the system. All the op-codes are written within this project, which decides the operation to be performed. Over here we can add more op-codes depending upon our requirement.

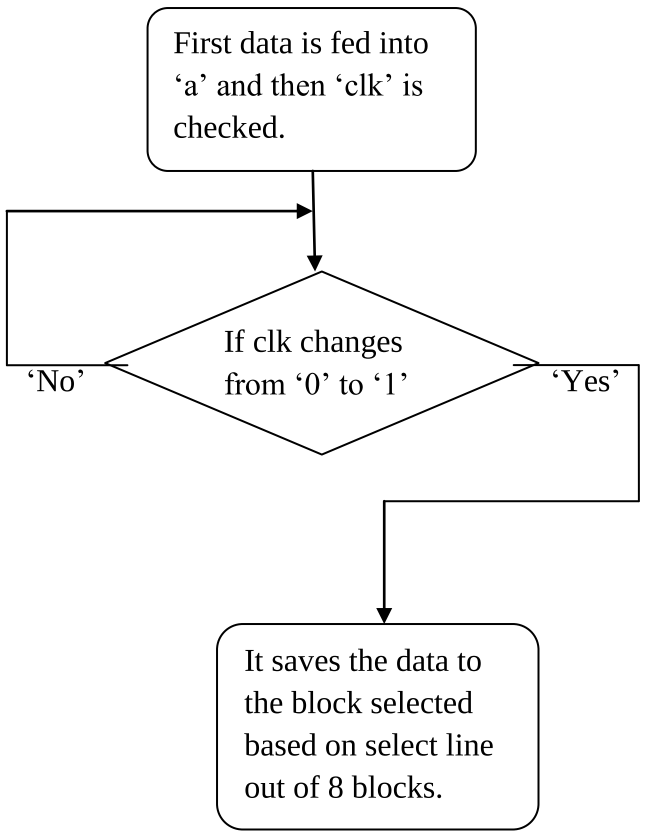
3. Working flow chart of each block

In this particular block we are basically going to explain the working of each block with the help of flow chart followed by detailed description of each and every flow chart in next section.

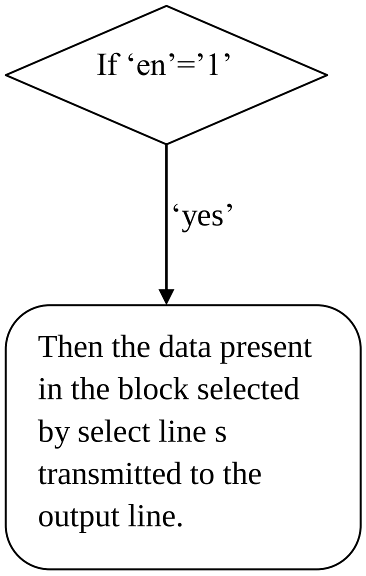
3.1 RAM Block

In this ‘a’ , ‘clk’,’en’ and ’sel’ are inputs and ‘y’ is output.

3.1.1 Flowchart for writing into block


First data is fed into ‘a’ and then ‘clk’ is checked.
If clk changes from ‘0’ to ‘1’
‘No’
‘Yes’
It saves the data to the block selected based on select line out of 8 blocks.

3.1.2 <u>Flowchart for reading from block</u>

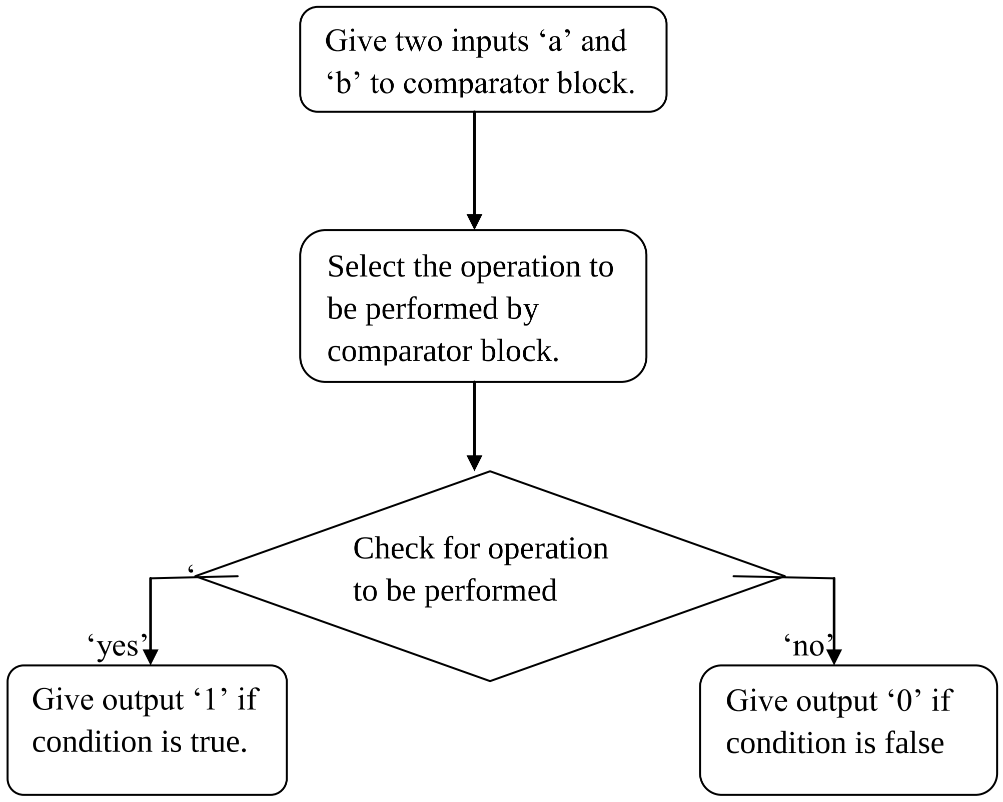

If ‘en’=’1’
‘yes’
Then the data present in the block selected by select line s transmitted to the output line.

The basic functionality of RAM block is that it acts as a storage for various results produced in the mid way of processing.

RAM works in following Manner:-

1 There are two lines based on which it decides whether the data is to be written in the RAM or data within is to be produced in the output. These two signals are ‘en’ and ‘clk’. It also has a select line. Select line is of 3- bit so the RAM designed contains 8- blocks each of 16- bits.
2 When the ‘clk’ signal changes from 0 to 1 the data on the input is stored on the register selected by select line.
3 When the ‘en’ signal changes from 0 to 1 the data on the selected register block is transferred to the output line.

3.2 Flowchart for Comparator


Give two inputs ‘a’ and ‘b’ to comparator block.
Select the operation to be performed by comparator block.
Check for operation to be performed
‘yes’
‘no’
Give output ‘1’ if condition is true.
Give output ‘0’ if condition is false

The basic functionality of Comparator block is that it just performs the comparison between the two inputs.

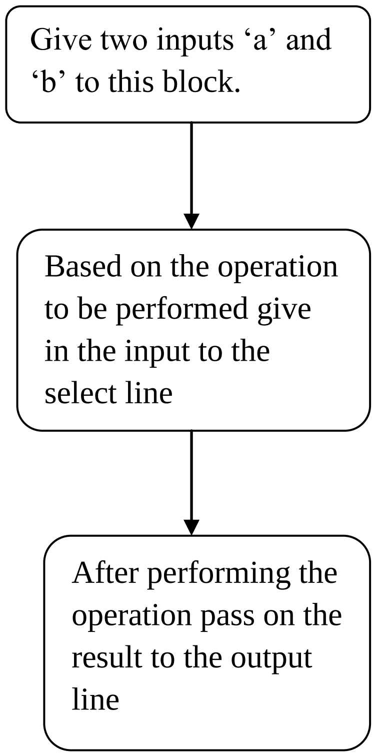
The comparison performed between the two inputs in this block are:-

1) Greater than
2) Greater than equal to
3) Less than
4) Less than equal to
5) Equal to
6) Not equal to

This block contains two inputs ‘a’ and ‘b’ each of 16-bit, a select line having enumerated data type to select the function to be performed on the inputs. A single bit output line to tell whether the condition is true or not. If output is ‘1’ then the condition is true otherwise its false.

For eg:- a and b are given to input and we have to check whether ‘a’ is greater than or equal to ‘b’ than we will give ‘gte’ to the select line if the condition is true than we will get ‘1’ in the output otherwise we will get ‘0’ in the output line.

3.3 Flowchart for ALU Block


Give two inputs ‘a’ and ‘b’ to this block.
Based on the operation to be performed give in the input to the select line
After performing the operation pass on the result to the output line

This is a block that performs various logical operations in this project. This block contains 2 inputs, one select line, one output. Again in this case select line selects the operation to be performed between two inputs or on one input.

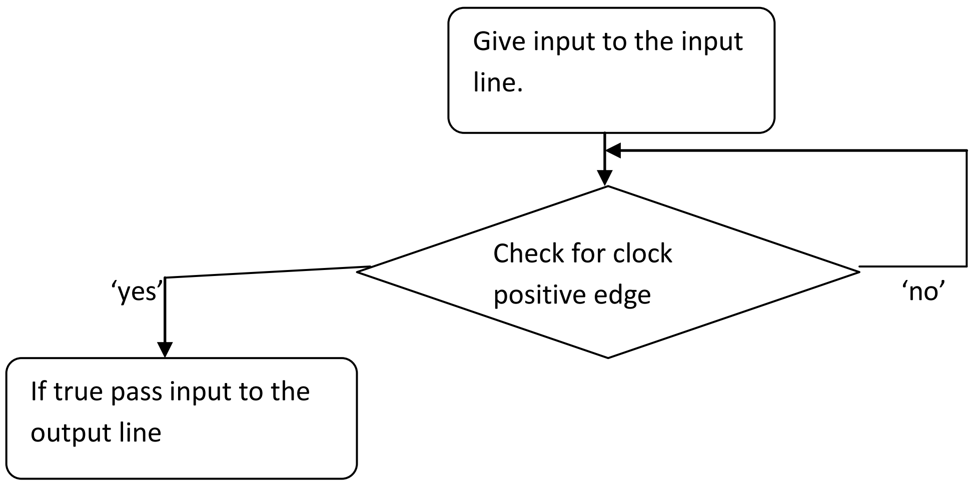
Suppose we give inputs to the ALU block both the inputs are of 16-bit each. A select line is there that selects the operation to be performed so we suppose set select line to ‘andop’ then we get the and of both the inputs on the output line.

3.4 Flowchart for Registers

There are two types of registers :

1. Bi-registers :- Pass input directly to output on positive edge of clk.
2. Tri- registers :- Stores input on one positive edge of clk and pass to output on positive edge of enable signal.

3.4.1 Bi-register


Give input to the input line.
Check for clock positive edge
'yes'
'no'
If true pass input to the output line

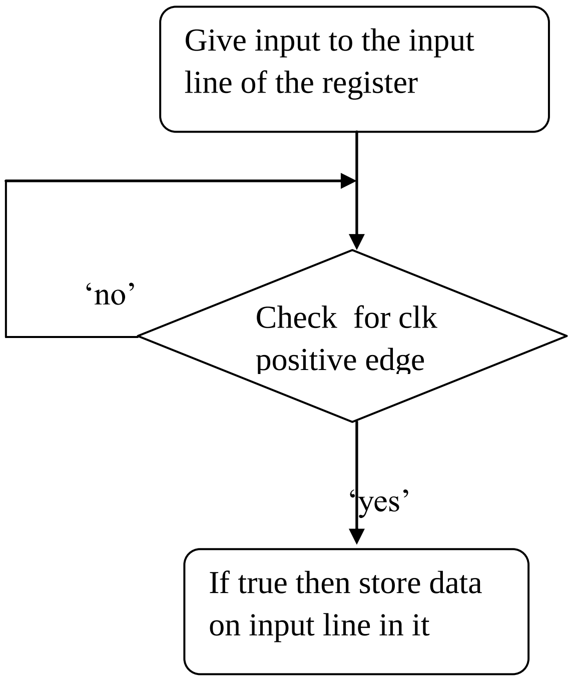
The basic need of this block is that some registers are required to simply pass the value to the output and keep it there.

Just taking example of how this bi-register is used in project. There is an operation register in the circuit controlled by control unit. The output of this register is input to ALU. When we need to perform operation on two inputs using ALU then since we have only one 16-bit data bus, so two inputs can't be stored there in such a case what we do is that first we select operation register since it is a bi- register the data automatically gets transferred to its output i.e. at one input of ALU ad then the second data is called to data bus and automatically comes to second input of ALU. Since this input is directly connected to the data bus, then we select the operation to be performed between both inputs in ALU and get the desired output.

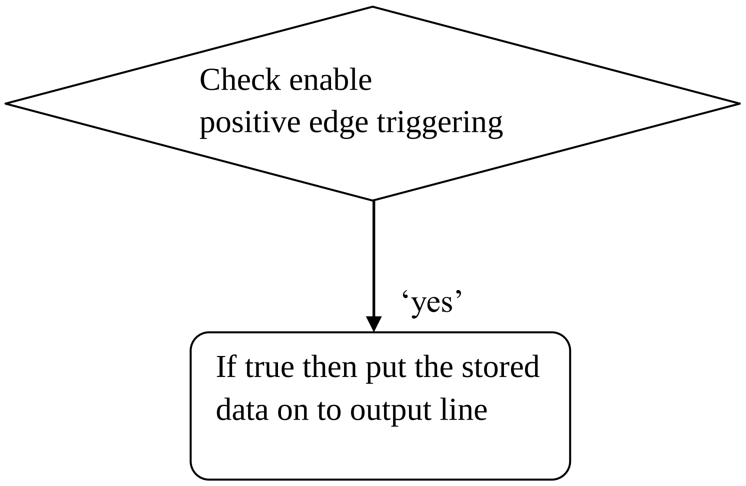
This register basically works in following manner when the positive edge of clock occurs then it basically transfers its input as it is to the output.

3.4.2 Tri-register

(a) Storing in register


Give input to the input line of the register
'no'
Check for clk positive edge
'yes'
If true then store data on input line in it

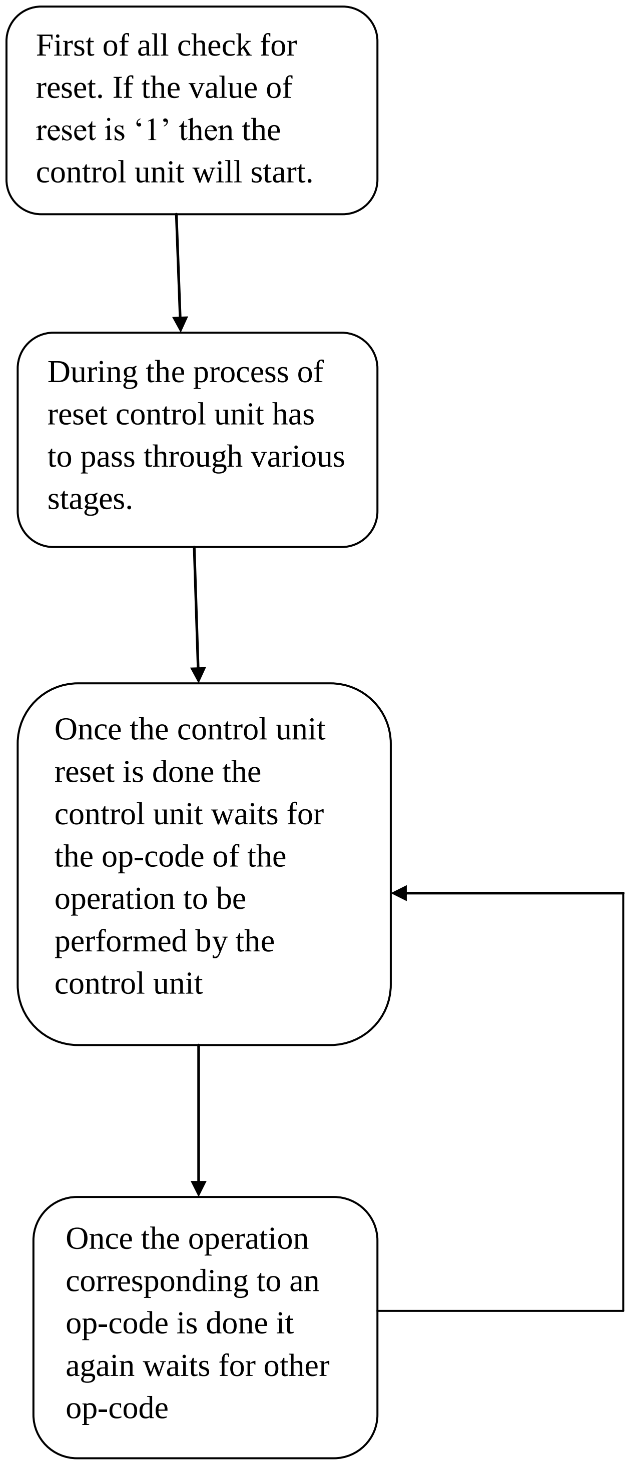
(b) Reading from register


Check enable positive edge triggering
'yes'
If true then put the stored data on to output line

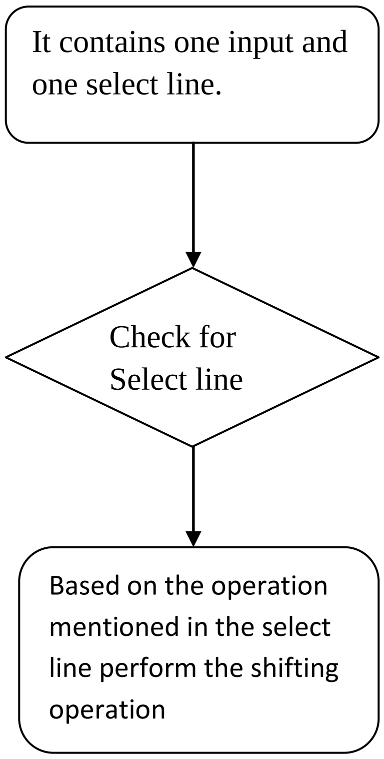
The basic need of this register is that some registers are required to save the value within it and then pass it to the output. The saving of value into the register occurs on the positive edge of the clock and the stored data is transferred to the output when ‘en’ i.e. enable signal gets a value’1’

Tri-register can be used at various places like in case of out-register in which the value after calculation from ALU block arrives. It cannot be directly passed to data bus as if we try to do such an operation then in that case the value on the data bus will get over written and new result will be produced taking in consideration this new value. So to avoid such problems we have used this tri-register using which data will be written into the register in one cycle and will be sent to output in other clock cycle.

3.5 <u>Flowchart for Control Unit</u>


First of all check for reset. If the value of reset is '1' then the control unit will start.
During the process of reset control unit has to pass through various stages.
Once the control unit reset is done the control unit waits for the op-code of the operation to be performed by the control unit
Once the operation corresponding to an op-code is done it again waits for other op-code

3.6 <u>Flowchart for Shift Units</u>


It contains one input and one select line.
Check for Select line
Based on the operation mentioned in the select line perform the shifting operation

The basic need of this block is that this block is required in the places where we need to shift the data or to rotate the data. The difference between shifting and rotating is that in one ‘0’ is added at one end while in one the data rotates completely i.e. the last data comes to first in case of rotate to left and vice versa in rotate to right.

There is a select line that selects the operation to be performed on the input. Using select line we select the operation to be performed on the data and depending upon the operation we will get the output on the output line.

4. RTL schematic of different blocks

4.1 RAM block

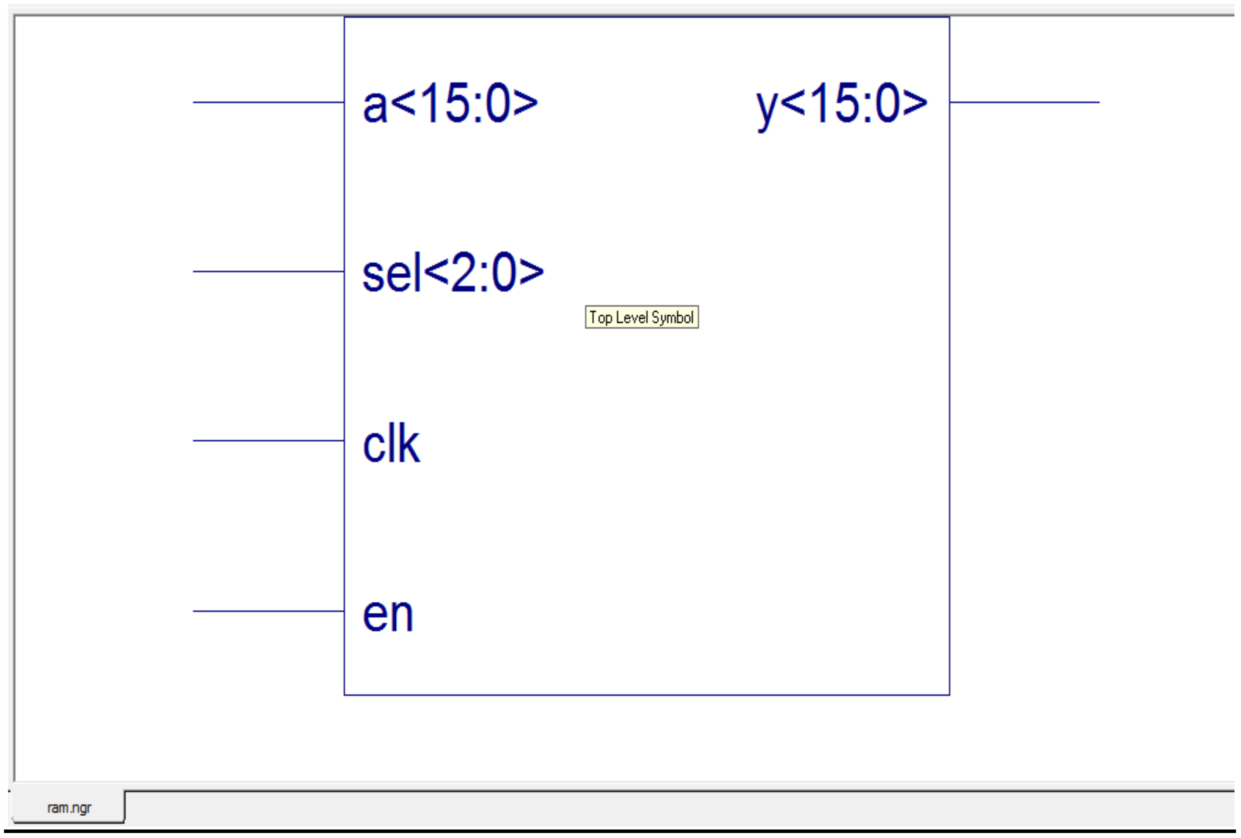


Figure 2 Top level symbol of RAM

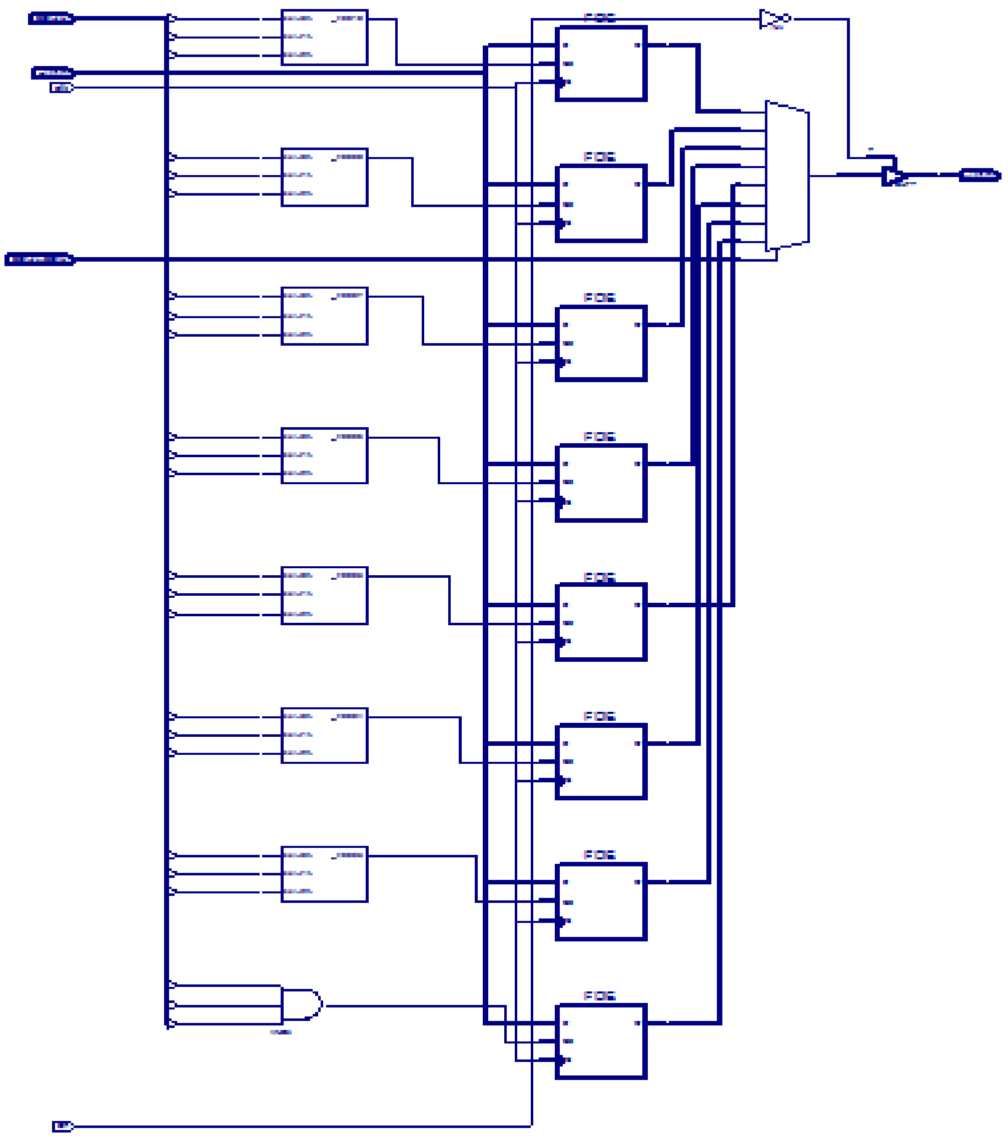


Figure 3 Top level schematic of RAM

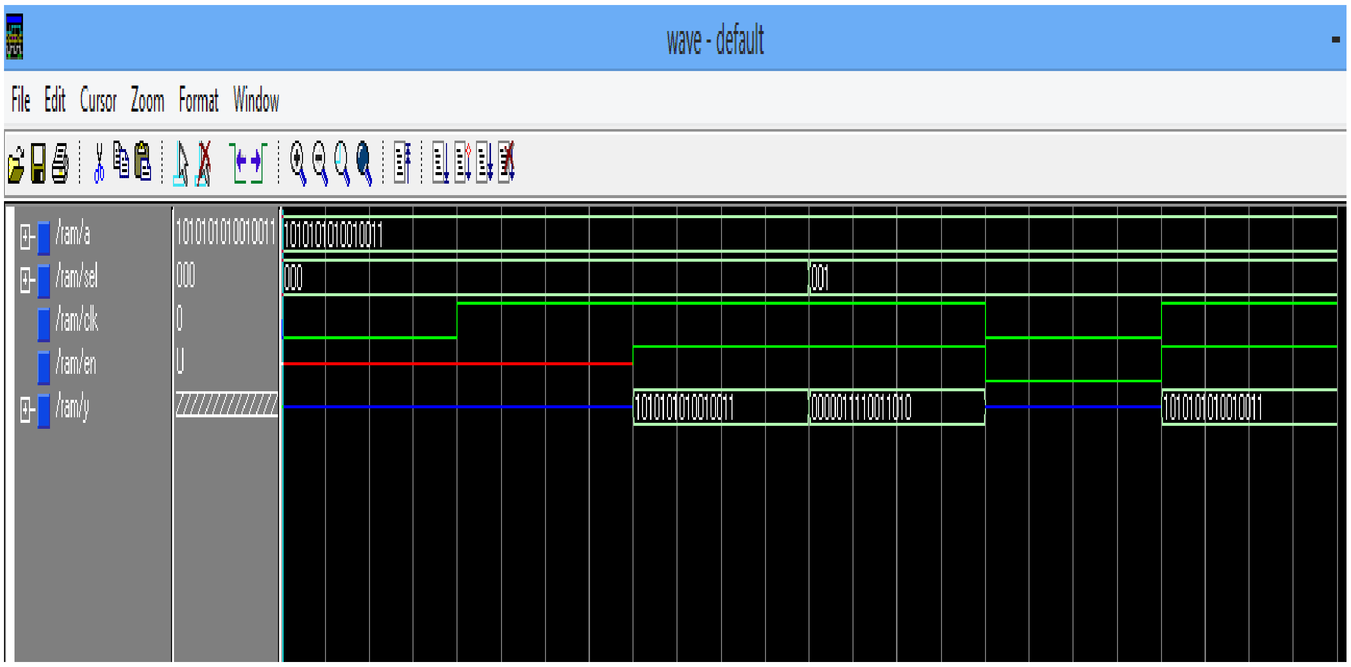


Figure 4 Simulation of RAM block

Table 1 RAM Simulation Explanation from table.

Input ‘a’	Input ‘sel’	Input ‘clk’	Input ‘en’	Output ‘y’
1010101010010011	000	0	u	zzzzzzzzzzzzzzzz
1010101010010011	000	1	u	zzzzzzzzzzzzzzzz
1010101010010011	000	1	1	1010101010010011
1010101010010011	001	1	1	0000011110011010
1010101010010011	001	0	0	zzzzzzzzzzzzzzzz
1010101010010011	001	1	1	1010101010010011

<u>Description of Simulation Table of RAM block</u>

As we can see from above table it is clear that RAM block contains four inputs and one output. When the input ‘a’ is forced to ‘1010101010010011’. Input ‘a’ is the input line to the RAM block. ‘Sel’ is to select the register block from RAM it has 8 blocks each of 16- bit , so from table 2 it is clear that when register block 1 is selected then in first row clk is ‘0’ and ‘en’ is ‘u’. so the output that we get corresponding to this is all 16-bit ‘z’. Now when the positive edge of ‘clk’ arrives then the value is stored in register block ‘1’ . Now when the ‘en’ signal is made ‘1’ then the output gets the value that was given as input to that block. Since in RAM there is some stored value in each and every block so in row -4 of table -1 when we selected block-2 keeping ‘en’ signal ‘1’ the value that was stored in it came as it is to the output i.e. ‘0000011110011010’.

Table 2 Register block Corresponding to 'sel' value

‘Sel’ value	**‘Register’ block it corresponds to**
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

4.2 Comparator block

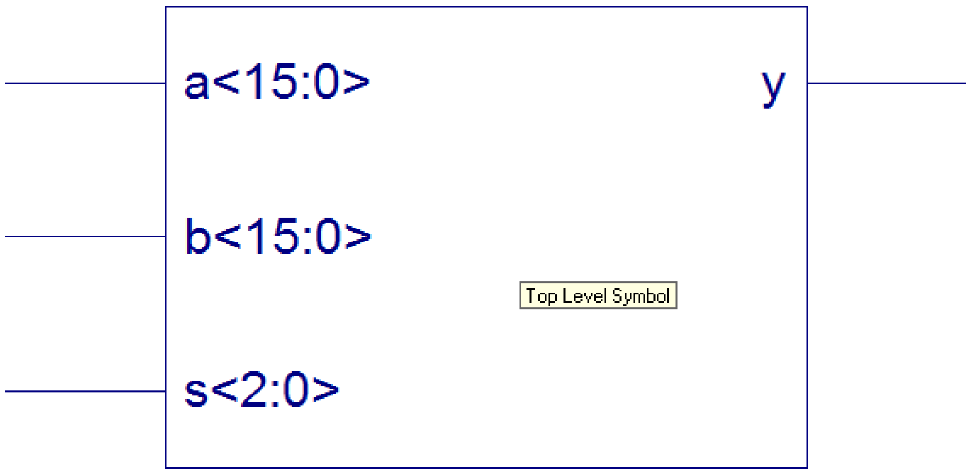


Figure 5 Top level Symbol of comparator

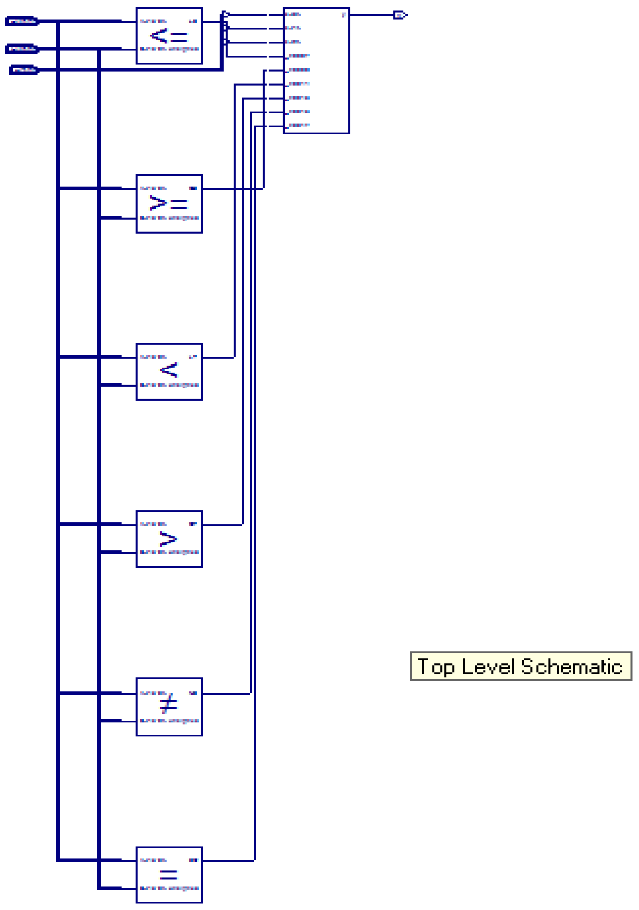
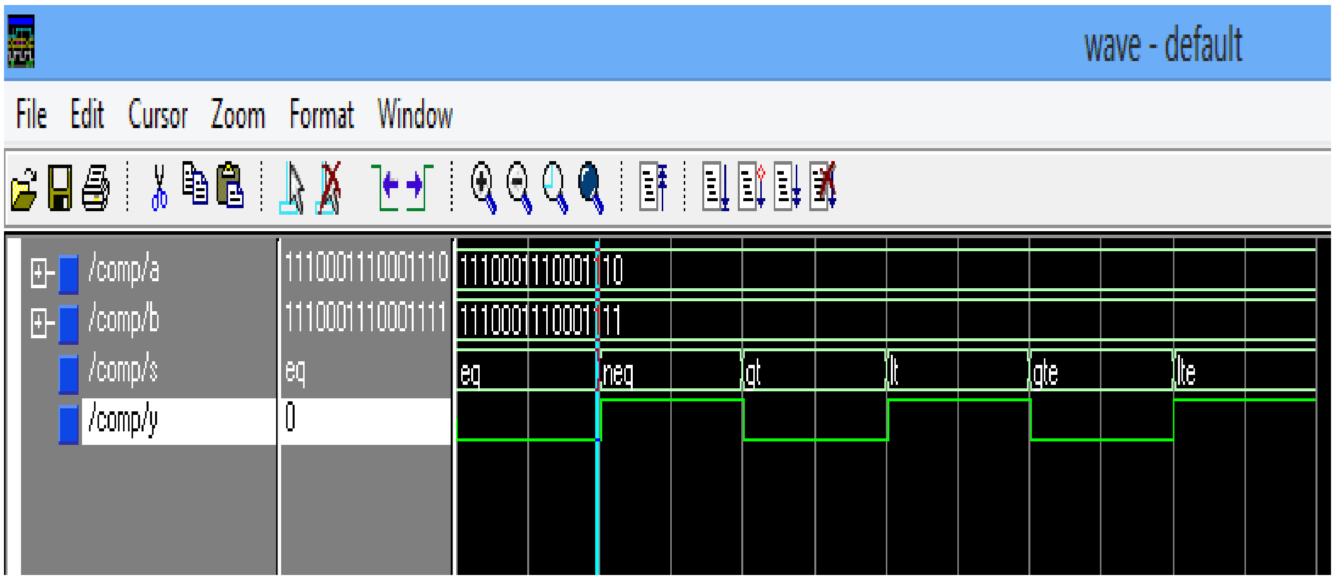


Figure 6 Top level Schematic of Comparator block



```
/comp/a   1110001110001110   1110001110001110
/comp/b   1110001110001111   1110001110001111
/comp/s   eq                 eq  neq  gt  lt  gte  lte
/comp/y   0
```

Figure 7 Simulation of Comparator block

Table 3 Comparator Simulation Explanation through table

Input ‘a’	Input ‘b’	Select ‘s’	Output ‘y’
1110001110001110	1110001110001111	eq	0
1110001110001110	1110001110001111	neq	1
1110001110001110	1110001110001111	gt	0
1110001110001110	1110001110001111	lt	1
1110001110001110	1110001110001111	gte	0
1110001110001110	1110001110001111	lte	1

Description of Simulation Table of Comparator block

As shown in table-3 it is clear that ‘a’ and ‘b’ are 16 –bit input line for this block. ‘s’ is the select line to select the operation to be performed by this block. ‘a’ is given a value ‘1110001110001110’ as input and ‘b’ is given a value ‘1110001110001111’ as input. From row-1 of table-3 it is clear that corresponding to this value of ‘a’ & ‘b’ select line ‘s’ is given a value ‘eq’ so now the comparator blocks checks whether ‘a=b’ or not. Since ‘a’ is not equal to ‘b’ so it gives ‘0’ on the output line. If the condition would have been true then the output corresponding to this would have been ‘1’, as shown in second row in which it is checking for not equal to condition. The output line is of 1-bit.

4.3 Bi-register Block

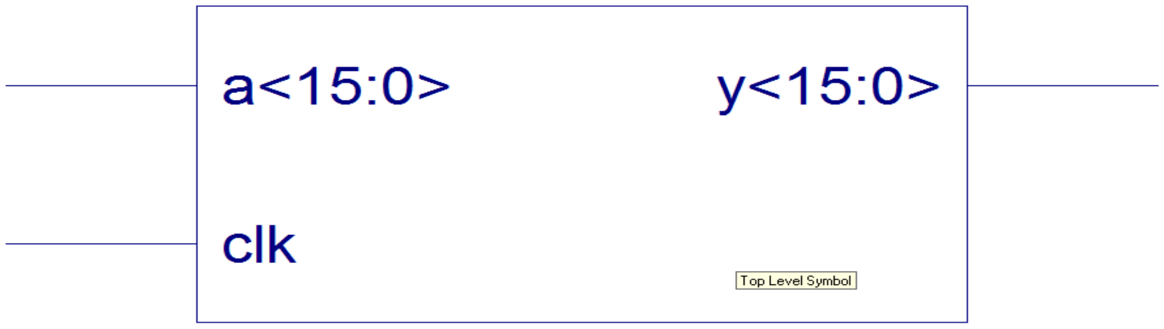


Figure 8 Top level Symbol of Bi-register

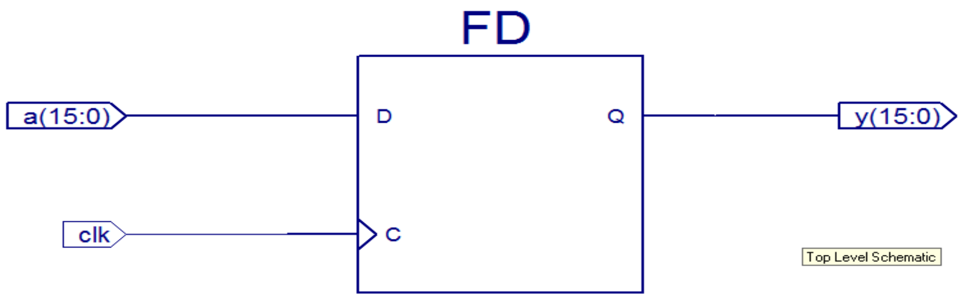


Figure 9 Top level Schematic of Bi-register

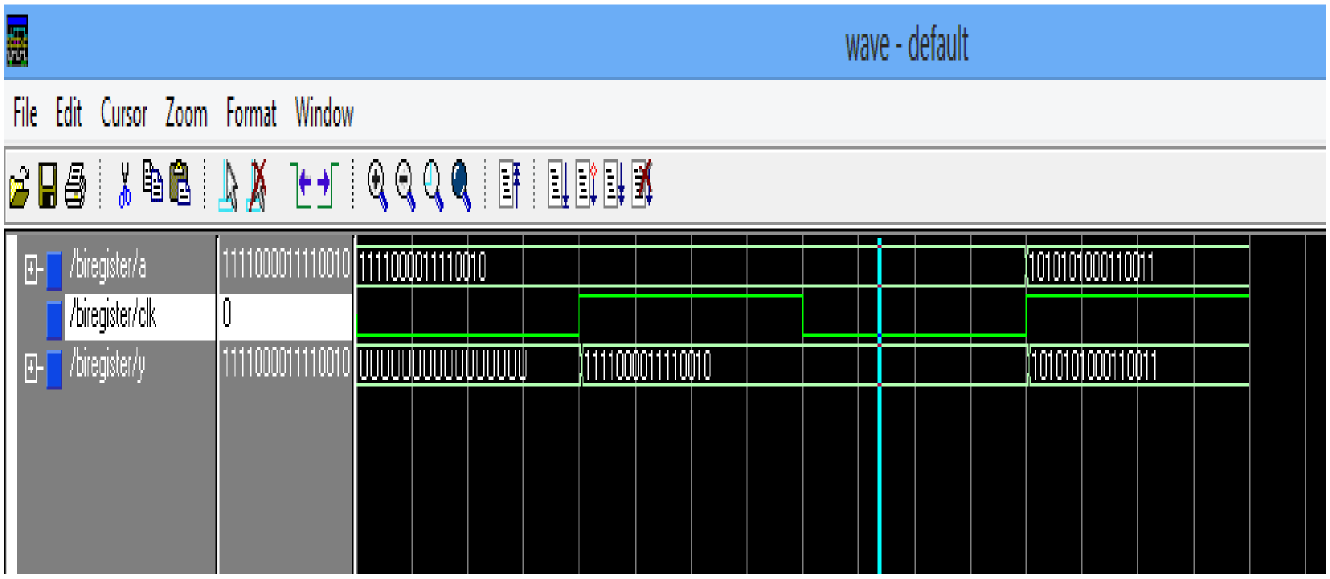


Figure 10 Simulation of Bi-register block

Table 4 Bi-register Simulation explanation through table

Input'a'	Input 'clk'	Output 'y'
1111000011110010	0	uuuuuuuuuuuuuuuu
1111000011110010	1	1111000011110010
1111000011110010	0	1111000011110010
1010101000110011	1	1010101000110011

Description of Simulation Table of Bi-register block

As mentioned earlier that in bi-register 'a' is input line and it has 'clk' signal and at the positive edge of 'clk' the input is transferred to the output. So from table- 4 it is clear that when initially input 'a' is assigned a value '1111000011110010' in the first row and 'clk' is assigned a value '0'. We get output as all the 16-bits 'u', and when the positive edge of 'clk' arrives i.e. in second row of the table the input gets transferred to the output.

4.4 Tri-register block

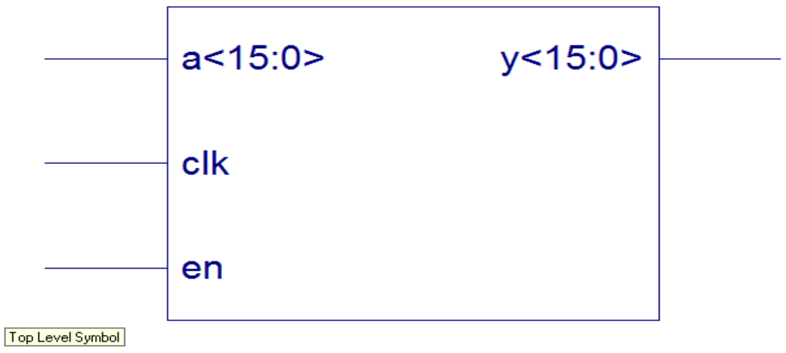


Figure 11 Top level Symbol of Tri-register

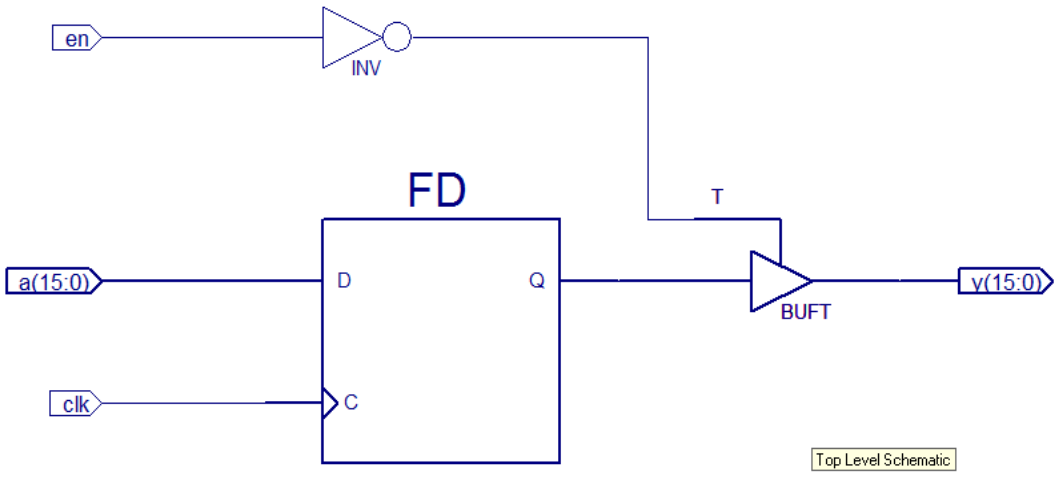


Figure 12 Top level Schematic of Tri-register

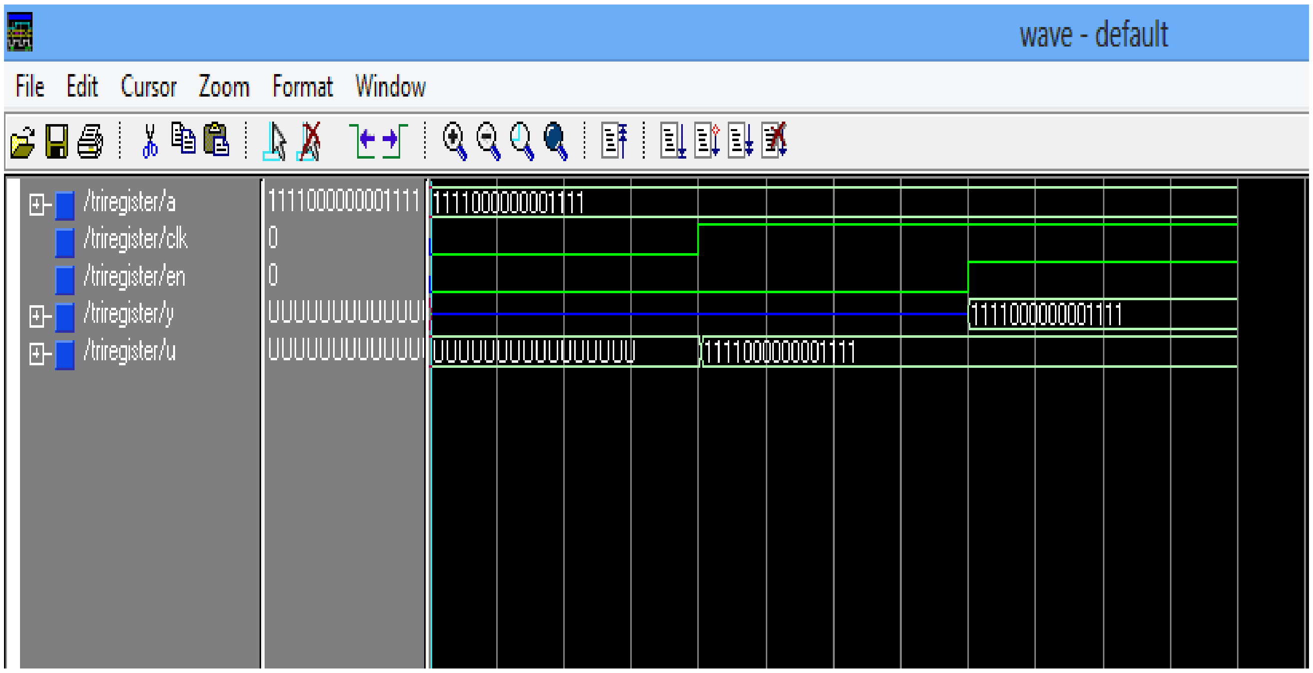


Figure 13 Simulation of Tri-register Block

Table 5 Tri-register Simulation explanation through table

Input 'a'	Input 'clk'	Input 'en'	Storage 'u'	Output 'y'
1111000000001111	0	0	uuuuuuuuuuuuuuuu	zzzzzzzzzzzzzzzz
1111000000001111	1	0	1111000000001111	zzzzzzzzzzzzzzzz
1111000000001111	1	1	1111000000001111	1111000000001111

<u>Description of Simulation Table of Tri-register block</u>

As we can see in table-5 it is clear that this block contains three input lines 'a' input that is of 16-bit , 'clk ' is an input, 'en' signal is there to pass input to output when 'en' signal is '1' the data stored is transferred to output as it is. As from table-5 we can see that when 'a' is given a value '1111000000001111' then initially when 'clk' was having value '0' and 'en' was having value '0' then storage was all 'u' and output all 'z' . In the second row i.e. when the positive edge of clock arrived then the value at the input got stored in the storage signal 'u' so the value corresponding to 'u' got changed to '1111000000001111' . Now in the third row when the 'en' signal got the value '1' the value in the storage signal 'u' got transferred to the output line 'y', so 'y' is having value '1111000000001111' in the third stage.

4.5 <u>Shifter block</u>

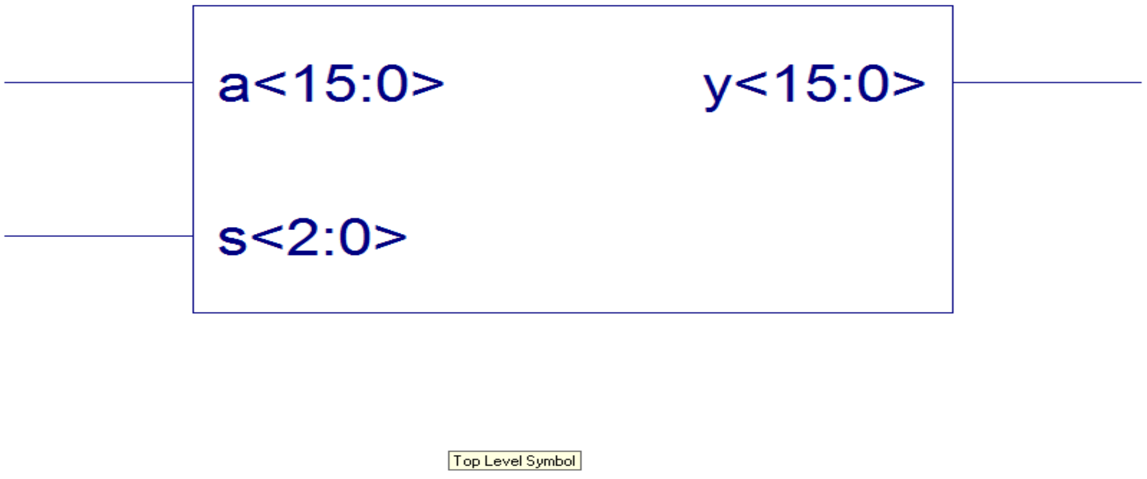


Figure 14 Top level Symbol of Shifter block

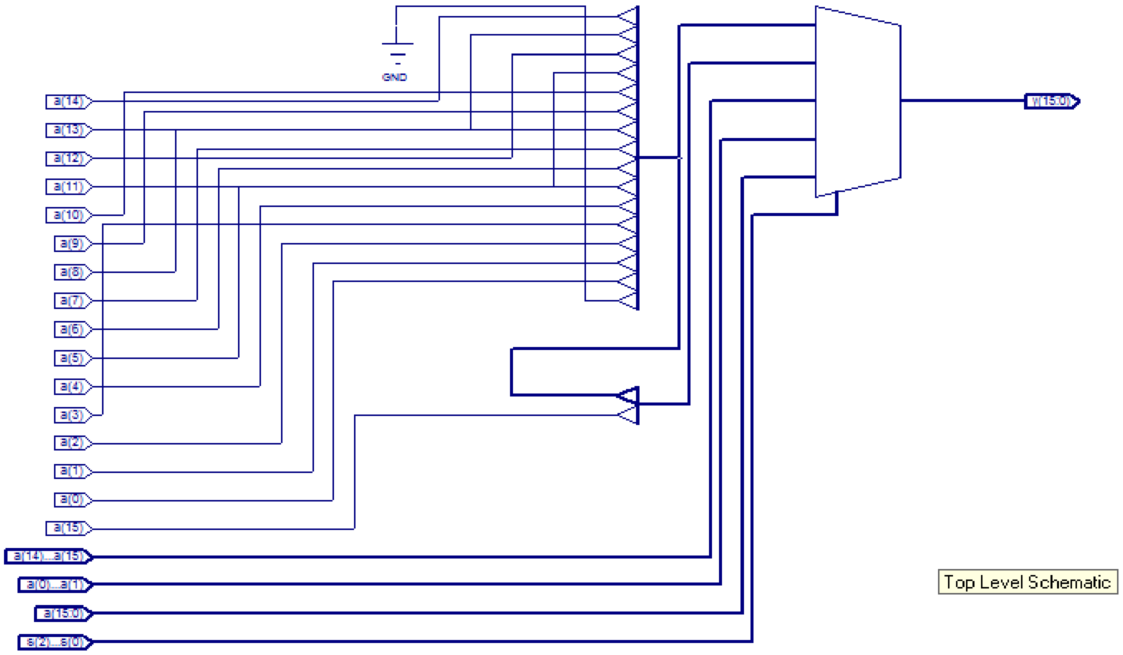
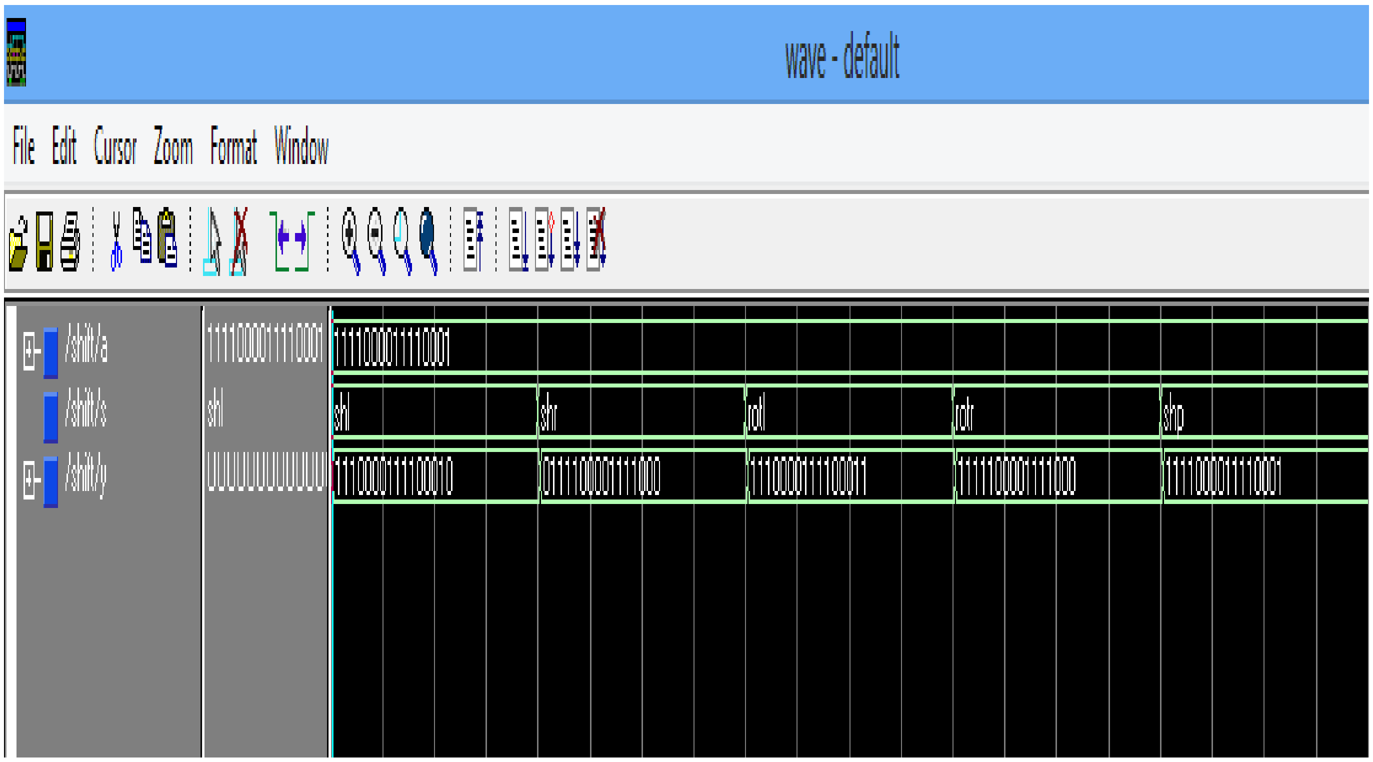


Figure 15 Top level schematic of shifter block



```
/shift/a  1111000011110001  1111000011110001
/shift/s  shl               shl               shr               rotl              rotr              shp
/shift/y  UUUUUUUUUUUUUUUU  1110000111100010  0111100001111000  1110000111100011  1111100001111000  1111000011110001
```

Figure 16 Simulation of Shifter Block

Table 6 Shifter Simulation explanation through table

Input 'a'	Select 's'	Output 'y'
1111000011110001	shl	1110000111100010
1111000011110001	shr	0111100001111000
1111000011110001	rotl	1110000111100011
1111000011110001	rotr	1111100001111000
1111000011110001	shp	1111000011110001

<u>Description of Simulation Table of Shifter block</u>

As mentioned earlier and also from table-6 it is clear that 'a' is the input line of shifter block. 's' is the select line that selects the operation to be performed by the shifter block. 'y' is a 16-bit output line. From the first row of the table it is clear when input '1111000011110001' is given to the shifter block and 'shl' is selected in select line then the input data is shifted to left by 1-bit and '0' is appended on the LSB of the input. Likewise in row-2 when 'shr' is selected as the operation then the data is shifted to right by one unit and '0'is appended to the MSB and 'y' has a value '1110000111100011'. When 'rotl' is selected then the entire data is rotated to left so the MSB of the input comes to the 'LSB' and 'y' has a value '0111100001111000' vice versa is case in 'rotr' so 'y' has value '1111100001111000'. And when 'shp' is selected input is as it is passed to the output so 'y' has a value '1111000011110001'.

4.6 <u>ALU block</u>

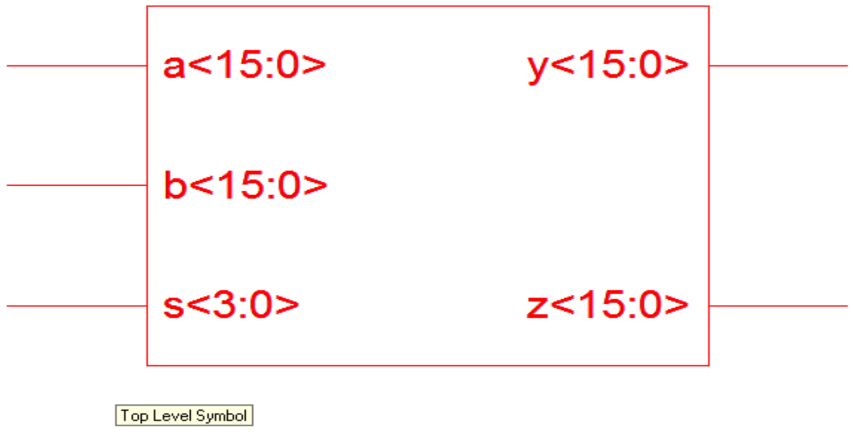
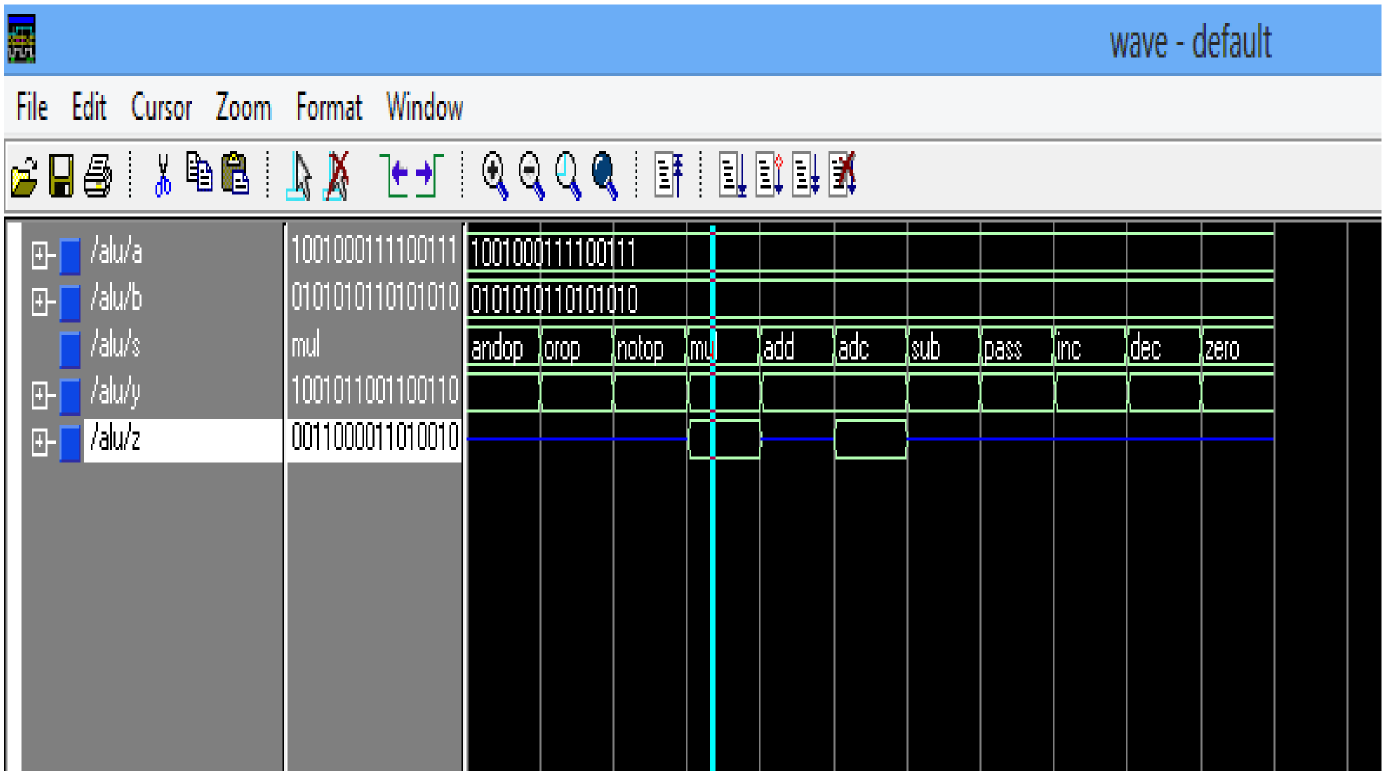


Figure 17 Top level symbol of ALU block



```
/alu/a   1001000111100111
/alu/b   0101010110101010
/alu/s   mul
/alu/y   1001011001100110
/alu/z   0011000011010010
```

Figure 18 Simulation of ALU block

Table 7 ALU Simulation explanation through table

Input 'a'	Input 'b'	Select input 's'	Output 'y'	Output 'z'
1001000111100111	0101010110101010	Andop	0001000110100010	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Orop	1101010111101111	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Notop	0110111000011000	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Mul	1001011001100110	0011000011010010
1001000111100111	0101010110101010	Add	1110011110010001	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Adc	1110011110010001	0000000000000000
1001000111100111	0101010110101010	Sub	0011110000111101	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Pass	1001000111100111	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Inc	1001000111101000	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Dec	1001000111100110	uuuuuuuuuuuuuuuu
1001000111100111	0101010110101010	Zero	0000000000000000	uuuuuuuuuuuuuuuu

Description of Simulation Table of Shifter block

As we can see from table-7 'a' and 'b' are two input lines, 's' is the select line which selects the operation to be performed by this block. 'y' and 'z' are two output lines. Two output lines are required for Multiplication and addition with carry as multiplication of two 16- bit number will be a 32-bit number so it requires two 16-bit output lines. Lower 16-bits will be stored in 'y' and upper 16- bits are stored in 'z'. Now initially as we can see from table that 'a' is assigned a value '1001000111100111' and 'b' is given a value '0101010110101010' select input is given 'andop' in the first row of table so the output 'y' is basically the and of 'a' and 'b' and its value is '0001000110100010'. When select input is given value 'orop' then we get following output '1101010111101111' which is the or of the values in 'a' and 'b'. When the select input gets the value 'notop' then the output we get is '0110111000011000', which is not of the value in input 'a'. When the select input gets value 'mul' then the output that we get is multiplication of 'a' and 'b' the value of outputs are 'y' = '1001011001100110' and 'z' = '0011000011010010'. Likewise corresponding to different set of input given to the select line the output is produced accordingly.

4.7 <u>Main Block</u>

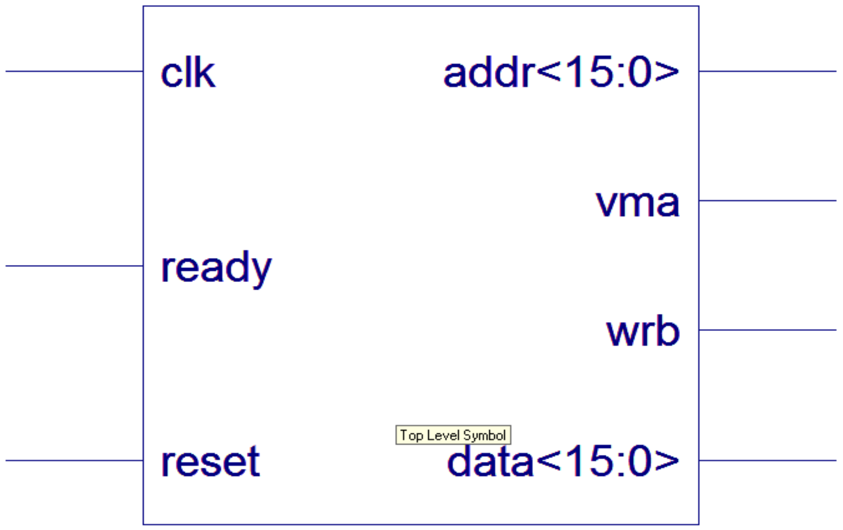


Figure 19 Top level Symbol of Main Block

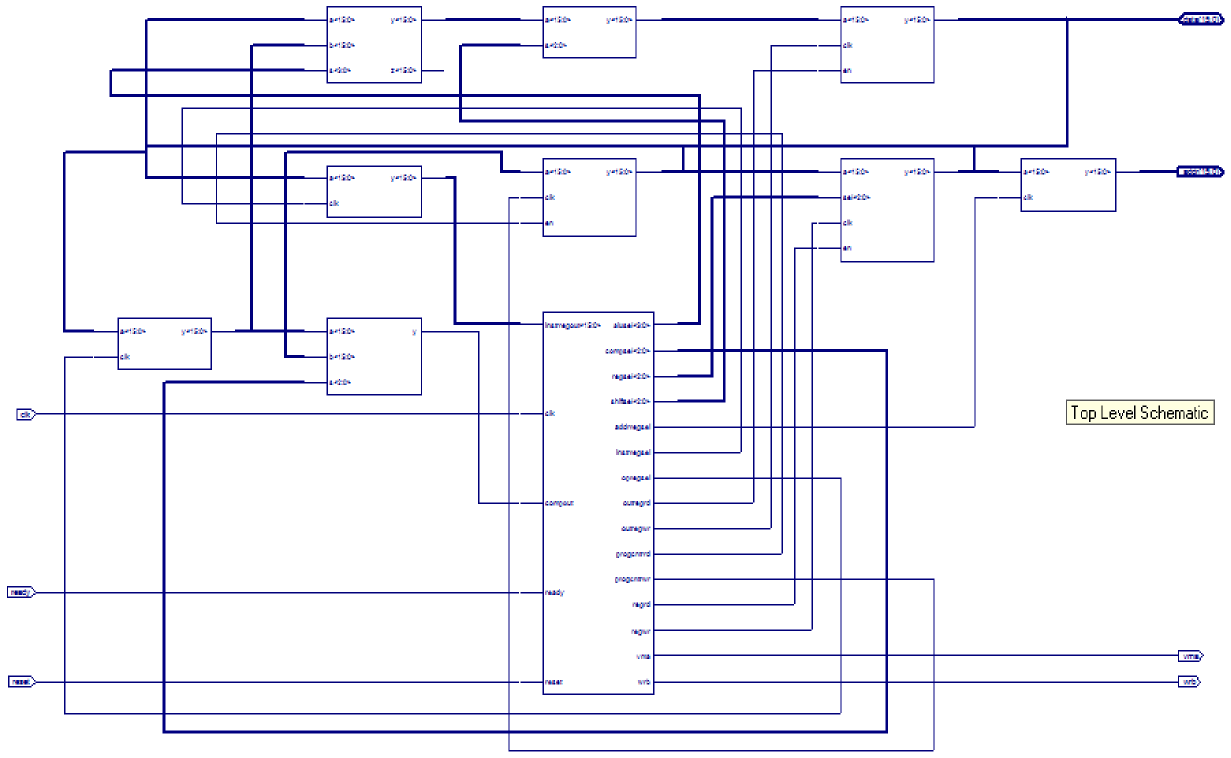


Figure 20 Top level schematic of Main block

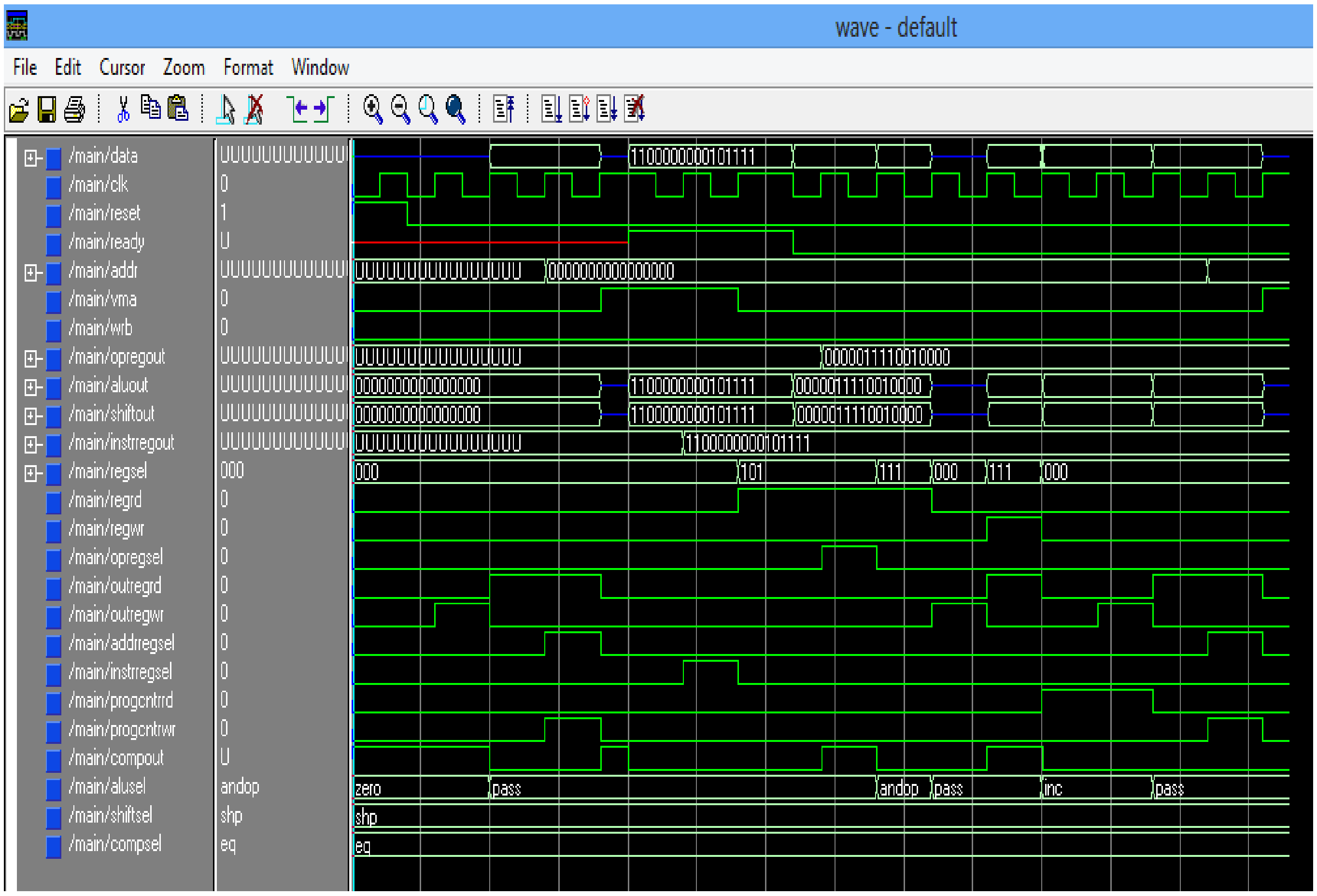


Figure 21 Simulation of Main block

<u>Detailed description of Simulation of main block</u>

The process to be followed in the simulation is as follows:-

1) Initially set ‘clk’ to 0 and reset to ‘1’ so as to reset the system as done in above Diagram.
2) Now set ‘clk’ to ‘1’ and again run
3) Now set ‘clk’ to ‘0’ and reset to ‘0’ and run.
4) Then set ‘clk’ to ‘1’ and run then outregwr is set to ‘1’ when you run this.
5) Then on the next positive edge of clk, outregrd signal is set to ‘1’. So now data bus will have value that was is outreg i.e. all 16-bits ‘0’
6) On the next positive edge of clk, progcntrwr gets value ‘1’ and all 16-bit ‘0’ is written to it. Same data is written to address register connected to external address bus.
7) On the next positive edge of clk, vma is set to ‘1’ and wrb to ‘0’ i.e. the processor is waiting for an op-code to arrive on data bus. So that operation corresponding to that particular op-code is performed. This value is written manually since, no ROM is present so we have to give it manually.

8) In the above simulation shown the data that was fed to it was ‘1100000000101111’ where the top 5-bits i.e. ‘11000’ corresponds to op-code lower 6-bits divided into 2- segments each of 3-bits represents the RAM block on which the operation is to be performed. i.e. ‘101’ and ‘111’ i.e. 6th and 8th block. And the data will be finally stored to the destination register as per opcode that in this case is ‘111’ i.e. 8th block of RAM. OPCODE ‘11000’ represents that the operation that will be on the two registers is ‘andop’.
9) Now in the first clk cycle the data will be sent to instruction register output which is given as input to control unit which will decode and identify the operation to be performed on the two register blocks.
10) Now on the next positive edge of the ‘clk’ the data is read from 6th block and bought on data bus and send to output of opreg which is a bi-register. Which is one input to ALU block.
11) On the next clk positive edge the data from other register is read and the ‘and’ operation is performed on the both the inputs and data is passed to the input of shift register which simply passes the input to output since it is selected as ‘shp’ now the output of shifter is input to ‘outreg’ which is a tri-register.
12) Now when the next positive edge of clk arrives data is written to outreg.
13) In the next positive edge of clk data is read from outreg and is now on data bus.
14) The data on the data bus is now written on to the destination register .
15) Then in the next few cycles the progcntr is incremented which is used to get the address of the next op-code.
16) And again the same process is performed corresponding to other op-code.

Conclusion

The design architecture of 16-bit microprocessor is written in VHDL code using Xilinx ISE and Modelsim 5.4a tools for synthesis and simulation. From synthesis report it is clear that minimum clock period that can be achieved from this proposed microprocessor is 3.526ns, which translate to a maximum operating frequency of 283.607MHz.

Appendix A

Signals and Their use

Signals	**Use**
Clk	This is basically clock signal given to all the blocks which are dynamic
En	This is enable signal used basically in RAM and tri-register to move input to output
andop	Enumerated data type in ALU to perform "AND" operation
Orop	Enumerated data type in ALU to perform "OR" operation
notop	Enumerated data type in ALU to perform "NOT" operation
xorop	Enumerated data type in ALU to perform "XOR" operation
Add	Enumerated data type in ALU to perform "ADD" operation
Adc	Enumerated data type in ALU to perform "Addition with carry" operation
Sub	Enumerated data type in ALU to perform "subtract" operation
Mul	Enumerated data type in ALU to perform "multiply" operation
Inc	Enumerated data type in ALU to perform "increment by 1" operation
Dec	Enumerated data type in ALU to perform "decrement by 1" operation
Pass	Enumerated data type in ALU to pass the input as it is to output
Zero	Enumerated data type in ALU to pass 16-bit '0' to output
Shl	Enumerated data type in shifter to perform left shift by 1-bit
Shr	Enumerated data type in shifter to perform right shift by 1-bit
Rotl	Enumerated data type in shifter to perform rotate to left shift by 1-bit
Rotr	Enumerated data type in shifter to perform rotate to right shift by 1-bit
Shp	Enumerated data type in shifter to pass input as it is to output
Eq	Enumerated data type in comparator to check whether both the inputs are equal or not
Gte	Enumerated data type in comparator to check whether input 'a' is greater than or equal to 'b' or not
Lte	Enumerated data type in comparator to check whether input 'a' is less than or equal to 'b' or not
Gt	Enumerated data type in comparator to check whether input 'a' is greater than 'b' or not
Lt	Enumerated data type in comparator to check whether input 'a' is less than 'b' or not
reset1	Symbolizes the state corresponding to first clock cycle of reset
reset2	Symbolizes the state corresponding to second clock cycle of reset
Reset3	Symbolizes the state corresponding to third clock cycle of reset
Reset4	Symbolizes the state corresponding to fourth clock cycle of reset
Reset5	Symbolizes the state corresponding to fifth clock cycle of reset
Reset6	Symbolizes the state corresponding to sixth clock cycle of reset
execute	Symbolizes the state corresponding to the start of execution of op-code
Move	Symbolizes the state corresponding to first clock cycle of move operation to move from one register to other
Move2	Symbolizes the state corresponding to second clock cycle of move operation to move from one register to other
Move3	Symbolizes the state corresponding to third clock cycle of move operation to move from one register to other

Move4	Symbolizes the state corresponding to fourth clock cycle of move operation to move from one register to other
Incpc	Symbolizes the state corresponding to first clock cycle of increment program counter
Incpc2	Symbolizes the state corresponding to second clock cycle of increment program counter
Incpc3	Symbolizes the state corresponding to third clock cycle of increment program counter
Incpc4	Symbolizes the state corresponding to fourth clock cycle of increment program counter
Incpc5	Symbolizes the state corresponding to fifth clock cycle of increment program counter
Incpc6	Symbolizes the state corresponding to sixth clock cycle of increment program counter
Inc2	Symbolizes the state corresponding to second clock cycle of increment
Inc3	Symbolizes the state corresponding to third clock cycle of increment
Inc4	Symbolizes the state corresponding to fourth clock cycle of increment
Add1	Symbolizes the state corresponding to first clock cycle of add operation
Add2	Symbolizes the state corresponding to second clock cycle of add operation
Add3	Symbolizes the state corresponding to third clock cycle of add operation
Add4	Symbolizes the state corresponding to fourth clock cycle of add operation
And1	Symbolizes the state corresponding to first clock cycle of and operation
And2	Symbolizes the state corresponding to second clock cycle of and operation
And3	Symbolizes the state corresponding to third clock cycle of and operation
And4	Symbolizes the state corresponding to fourth clock cycle of and operation
Or1	Symbolizes the state corresponding to first clock cycle of or operation
Or2	Symbolizes the state corresponding to second clock cycle of or operation
Or3	Symbolizes the state corresponding to third clock cycle of or operation
Or4	Symbolizes the state corresponding to fourth clock cycle of or operation

Appendix B

Control Signals and their requirements

Control signal	**Requirement**
Clk	Just works as normal clock to control unit.
Reset	Signal that symbolizes that block need to be reset when set to ‘1’
Instrregout	This is the output of instruction register which is input to this block to decode the op-code that it contains
Compout	It is the output of comparator block
Ready	Ready is a signal used to symbolize that the op-code from external memory has been fed into data bus when it has value ‘1’
progrcntrwr	This is just like clock signal to tri-register when its positive edge arrives data is written into it
Progcntrrd	This is just like enable signal to tri-register
Addrregsel	This is just like clock signal to bi-register
Outregwr	This is just like clock signal to tri-register when its positive edge arrives data is written into it
Outregrd	This is just like enable signal to tri-register
Shiftsel	This symbolizes select line corresponding to shift register
Alusel	This symbolizes select line corresponding to ALU
Compsel	This symbolizes select line corresponding to comparator
Opregsel	This is just like clock signal to bi-register
Instrregsel	This is just like clock signal to bi-register
Regsel	This is just like select line to RAM
Regwr	This is just like clock signal to RAM
Regrd	This is just like enable signal of RAM
Vma	“vma” when set to ‘1’ symbolizes that a valid memory address has been sent to address bus and processor is waiting for instruction op-code to arrive.
Wrb	“wrb” signifies that data bus is ready for signal to be written in it.

References

[1] Alpesh Kumar Dauda, Nalinikanta Barpanda, Nilamani Bhoi, Manoranjan Pradhan “Control Unit Design of a 16-bit Processor Using VHDL”, “International Journal of Advanced Research in Computer Science and Software Engineering” Volume 3, Issue 12, December 2013.

[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.

[4] 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.

[5] D.L. Perry, VHDL, 4 Edition, Mc Graw-Hill, 2002.

[6] Volnei A. Pedroni, Circuit Design with VHDL, 2004.

[7] Anjana & Krunal Gandhi “VHDL Implementation of a MIPS RISC Processor” International Journal of Advanced Research in   Computer Science and Software Engineering Volume 2, Issue 8, August 2012.