Laboratory Manual

For

COMPUTER ORGANIZATION

(IT 402)

B.Tech. (IT)

SEM IV

July 2016
Faculty of Technology
Dharmsinh Desai University
Nadiad.
www.ddu.ac.in

Table of Contents

LABWORK BEYOND CURRICULA

1. **Program for implementing gcd of two numbers using 'c' language.**
2. **Write a program to implement 16-bit ALU in verilog.**

Sample Experiment

1 AIM: Write a program to implement nand gate in verilog.

2 TOOLS: Icarus Verilog.

3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:
Create a file with file_name.v. Define the main module and number of inputs and outputs.

3.2 Implementing the Solution
Create a new file in editor with name nandgate.v. Define a module nandgate with two input variables & one output variable. Define another module to create object of the main module, declare number of registers and wires required and give the input values for testing.

3.3.1 Writing Source Code

nandgate.v

```
module nandgate
(
input x,
input y,
output z
);

assign {z}=~(x&y);

endmodule

module xyz;

reg input1;
reg input2;
wire out;

andgate ob(.x(input1),.y(input2),.z(out));

initial
begin
input1 =0;
input2 =0;
```

```
#20; input1 =1;
#20; input2 =1;
#20; input1 =0;
end


initial
begin
$monitor("time = %2d,IN1=%1b, IN2=%1b,OUT=%1b", $time, input1, input2, out);
end
endmodule
Code_here  Ends
        End Start
```

3.3.2 Compilation /Running and Debugging the Solution

Step: 1

Compile File Using Verilog

iverilog nandgate.v

Step: 2

Run the program:

vvp nandgate

4 Conclusions

We can write a code and test various logic gates in verilog.

EXPERIMENT-1

Aim: Study of Introduction to verilog and installation
Apparatus (Software): Icarus Verilog
Procedure: Study of Introduction to verilog and installation
1) Installation steps:

1. Download verilog-version.tar.gz
2. Untar - tar -zxvf verilog-version.tar.gz
3. cd verilog-version
4 ./configure
5. make
6. su root
   root password
7. #make install

EXPERIMENT-2

Aim: Write a basic program of verilog and understand the syntax.
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:
1) Write a simple program to display "Hello World".
Steps:

1. Create a new file with name hello.v

```
module main;
initial begin
    $display("Hello, World");
    $finish ;
end
endmodule
```

2. Compiling program:

```
iverilog -o hello hello.v
```

3. Run:

```
vvp hello
```

EXPERIMENT-3

Aim: Write a program to implement different logic gates in verilog.
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:

1. Create a new file andgate.v
2. Write the program code to implement and gate in verilog.
3. Similarly write a program code in verilog for all the logic gates.

EXPERIMENT-4

Aim: Write a program to implement half adder and full adder in verilog**.**
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:

1. Create a new file halfadder.v
2. Write the program code to implement half adder in verilog.
3. Similarly write a program code in verilog for Full adder.

EXPERIMENT-5

Aim: Write a program to implement multiplexer in verilog.
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:

1. Create a new file mux.v
2. Write the program code to implement 2-to-1 multiplexer in verilog.
3. Similarly write a program code in verilog for 3-to-1 multiplexer.

EXPERIMENT-6

Aim: Write a program to implement D-flip-flop in verilog.
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:

1. Create a new file dflipflop.v
2. Write the following program code to implement D flipflop in verilog.

```
module d_ff
(
  input wire clock,
  input wire D,
  output reg Q
);
always@(posedge clock)

        Q=D;
endmodule

module stimulus;
  reg clock;
  reg D;
  wire Q;

d_ff d1(
     .clock(clock),.D(D),.Q(Q)
       );
integer i;
initial begin

D=1;
#10 D=0;
end

initial begin
clock=1;
$monitor("clock=%d,D=%d,q=%d\n",clock,D,Q);

for(i=0;i<=2;i++)
#10 clock=~clock;
end

endmodule
```

3. Similarly write a program code in verilog for JK flipflop.

EXPERIMENT-7

Aim: Write a program to implement ring counter in verilog**.**
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:

1. Create a new file ringcounter.v
2. Write the following program code to implement ring counter in verilog.

```
// 4 bit ring counter example
module four_bit_ring_counter (
     input clock,
     input reset,
    output [3:0] q
   );

   reg[3:0] a;

   always @(posedge clock or negedge clock)
    if (reset)
      a = 4'b0001;

    else
      begin
      a <=  a<<1; // Notice the blocking assignment
      a[0]<=a[3];  // <= non blocking
      end

   assign q = a;

 endmodule
//`timescale 1ns / 1ps
module stimulus;
       // Inputs
       reg clock;
       reg reset;
       // Outputs
 wire[3:0] q;
       // Instantiate the Unit Under Test (UUT)
       four_bit_ring_counter r1 (
    .clock(clock),
    .reset(reset),
    .q(q)
      );

 always #10 clock = ~clock;

       initial begin
```

```
    // Initialize Inputs
      clock = 1;
      reset = 1;

       #20 reset = 0;
       #20 $finish;
      end

                initial begin
        $monitor($time, " clock=%1b,reset=%1b,q=%4b",clock,reset,q);
                 end
endmodule
```

EXPERIMENT-8

Aim: Write a program to implement 8-bit ALU in verilog.
Apparatus (Software): Icarus Verilog
Procedure: To do this program follows these steps:

1. Create a new file alu.v
2. Write the following program code to implement 8-bit alu in verilog.

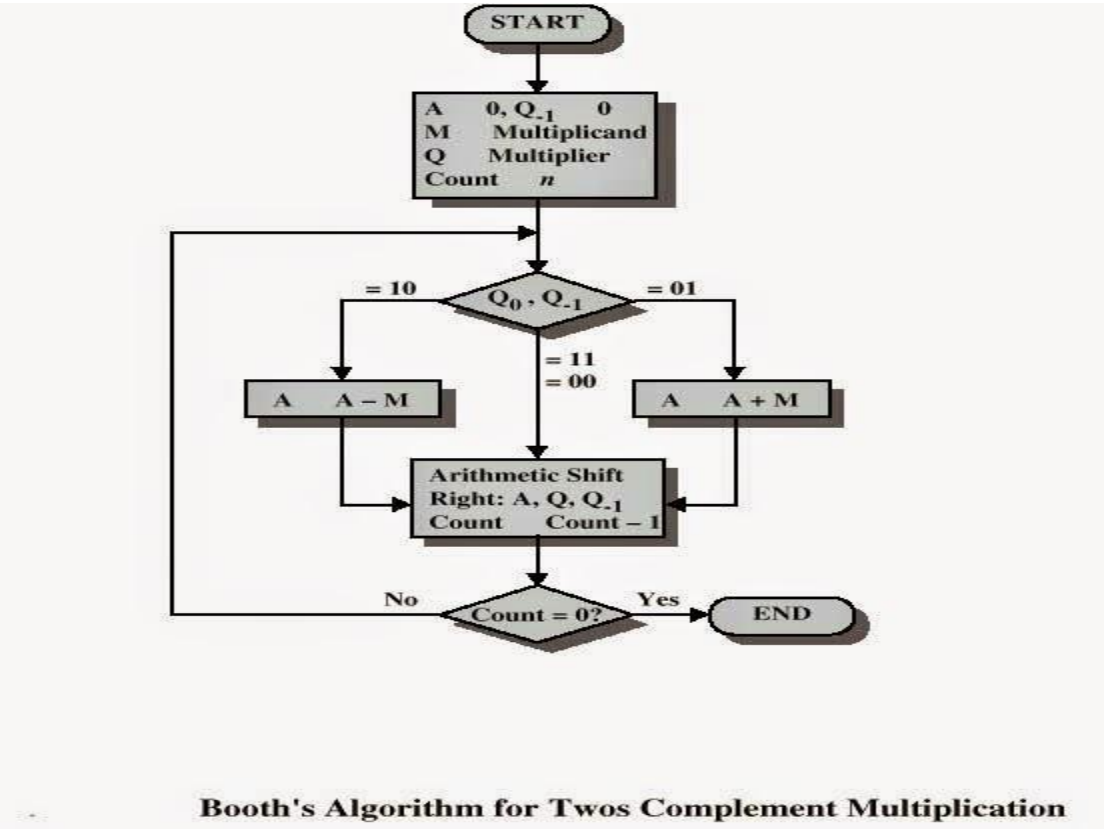
```
 module ALU(
A,
B,
Op,
R   );

//inputs,outputs and internal variables declared here
input [7:0] A,B;
input [2:0] Op;
output [7:0] R;
wire [7:0] Reg1,Reg2;
reg [7:0] Reg3;

//Assign A and B to internal variables for doing operations
assign Reg1 = A;
assign Reg2 = B;
//Assign the output
assign R = Reg3;

//Always block with inputs in the sensitivity list.
always @(Op or Reg1 or Reg2)
begin
   case (Op)
      0 : Reg3 = Reg1 + Reg2;  //addition
    1 : Reg3 = Reg1 - Reg2; //subtraction
    2 : Reg3 = ~Reg1;  //NOT gate
    3 : Reg3 = ~(Reg1 & Reg2); //NAND gate
    4 : Reg3 = ~(Reg1 | Reg2); //NOR gate
    5 : Reg3 = Reg1 & Reg2;  //AND gate
    6 : Reg3 = Reg1 | Reg2;  //OR gate
    7 : Reg3 = Reg1 ^ Reg2; //XOR gate
   endcase
end
endmodule
module tb_alu;

// Inputs
reg [7:0] A;
reg [7:0] B;
```

```
    reg [2:0] Op;

    // Outputs
    wire [7:0] R;

    // Instantiate the Unit Under Test (UUT)
    ALU uut (
        .A(A),
        .B(B),
        .Op(Op),
        .R(R)
    );

    initial begin
        // Apply inputs.
        A = 8'b01101010;
        B = 8'b00111011;
        Op = 0; #100;
        Op = 1; #100;
        Op = 2; #100;
        Op = 3; #100;
        Op = 4; #100;
        Op = 5; #100;
        Op = 6; #100;
        Op = 7; #100;
    end

endmodule
```

EXPERIMENT-9

Aim: Write a program to implement Booth's multiplication algorithm in C language.
Apparatus (Software): Turbo C/C++
Procedure: To do this program follows these steps:
1) The algorithm works for positive multipliers.
2) Booth`s algorithm examination of multiplier bits and shifting of the partial product.
3) Prior to the shifting, the multiplicand may be added to the partial product, subtracted or left unchanged.

Step 1:-In step 1 firstly we take a multiplicand M and multiplier Q and set value of AC, Q_{-1}, count are 0,0,0 respectively.
Step 2:-In step 2 We check Q_0 and Q_{-1}.
Step 3:-In step 3 if bits are 0,1 then add M with AC and after that perform Right Shift Operation.
If bits are 1,0 then perform AC+(M)'+1 then perform Right Shift Operation.
If bits are 0,0 or 1,1 then perform Right Shift Operation.
Step 4:-Check if count is set as o.
Step 5:-Repeat Step 2,3 until count<--0.



Booth's Algorithm for Twos Complement Multiplication

EXPERIMENT-10

Aim: To write and read data in RAM IC 6264
Apparatus (Hardware): RAM 6264 IC, power supply, bread board, connecting wires.
Procedure: RAM 6264 IC is 8k*8 configuration .some operation like memory read and memory write can be done using this IC. It has bi-directional data bus and unidirectional address bus.
Procedure:
To do this program follows these steps:
1) To perform read write operations connect the IC in following manner:
2) Memory write(read)operation takes place if WE`=0(write)WE`=1(read).
3) Chip select (CS`=0 or ground)triggers memory operation.
4) Chip enable(CE=1 or VCC)enables the chip.
5) Outputs enables(OE`)controls bi-directional data bus i.e. OE`=1(write)and OE`=0(read).