INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

Course No: **EE206P**

Name of the Lab: **Digital Systems Laboratory**

Academic Year/Semester: **2024/B.Tech 3^{rd} Sem**

ASSIGNMENT NUMBER: **01**

Verilog Programs To Impliment Some Functions

MADHU SAI NAIK POOJARI
EE23B039
PRATHAM CHINTAMANI
EE23B041

Instructor:
Dr. Vikramkumar Pudi
Lab Assistant:
Mr. Kumar Bellikatti

19-Aug-2024

Contents

1 Problem Statement

Assignment-1

Write a Verilog program to implement the following functions ?

S	Function	
000	C = A and B	
001	C = A or B	
010	C = A nand B	
011	C = A nor B	
100	C = A xor B	
101	C = A xnor B	
110	C = 2 A	Multiplication by 2
111	C = A/2	Division by 2
A, B and C are N-bit binary unsigned numbers. S is 3-bit binary number		

Deadline
August 19th, 2024

2 Objectives of the Experiment

To implement a logic circuit with smaller logic gate modules, using verilog and to verify the output.

3 Design Approach

- Create modules for each of the 8 operations and one for the 8x1 mux.
- Create a top-level module that connects these modules. i.e, the wire outputs of each of the 8 modules is the input for the mux along with s.
- Create the test-bench for the top-level module and test for some sample input.

4 Implementation

And_2in_Nbit

```
module And_2in_Nbit #(parameter N = 8)(c,a,b);

input [N-1:0]a;
input [N-1:0]b;
output [N-1:0]c;

assign c = a&b;

endmodule
```

Or_2in_Nbit

```
module Or_2in_Nbit #(parameter N = 8)(c,a,b);

input [N-1:0]a;
input [N-1:0]b;
output [N-1:0]c;

assign c =  a | b;

endmodule
```

Nand_2in_Nbit

```
module Nand_2in_Nbit #(parameter N = 8)(c,a,b);

input [N-1:0]a;
input [N-1:0]b;
output [N-1:0]c;

wire [N-1:0]y;

assign y = a&b;

assign c = ~y;

endmodule
```

Nor_2in_Nbit

```
module Nor_2in_Nbit #(parameter N = 8)(c,a,b);

input [N-1:0]a;
input [N-1:0]b;
output [N-1:0]c;

wire [N-1:0]y;

assign y = a|b;

assign c = ~y;

endmodule
```

Xor_2in_Nbit

```
module Xor_2in_Nbit #(parameter N = 8)(c,a,b);

input [N-1:0]a;
input [N-1:0]b;
output [N-1:0]c;

wire [N-1:0]a_bar;
wire [N-1:0]b_bar;
wire [N-1:0]y1;
wire [N-1:0]y2;

assign a_bar = ~a;
assign b_bar = ~b;
assign y1 = a&b_bar;
assign y2 = a_bar&b;
assign c = y1|y2;

endmodule
```

Xnor_2in_Nbit

```
module Xnor_2in_Nbit #(parameter N = 8)(c,a,b);

input [N-1:0]a;
input [N-1:0]b;
output [N-1:0]c;

wire [N-1:0]a_bar;
wire [N-1:0]b_bar;
wire [N-1:0]y1;
wire [N-1:0]y2;
wire [N-1:0]y;

assign a_bar = ~a;
assign b_bar = ~b;
assign y1 = a&b_bar;
assign y2 = a_bar&b;
assign y = y1|y2;

assign c = ~y;

endmodule
```

Mul_by2_Nbit

```
module Mul_by2_Nbit #(parameter N = 8) (a,y);

input [N-1:0]a;
output [N-1:0]y;

assign y = {a[N-2:0],1'b0};

endmodule
```

Div_by2_Nbits

```
module Div_by2_Nbit #(parameter N = 8)(a,b);

input [N-1:0]a;
output [N-1:0]b;

assign b = {1'b0 , a[N-1:1]};

endmodule
```

Mux_8x1_Nbit

```
module Mux_8x1_Nbit #(parameter N = 8) (s,a,b,c,d,e,f,g,h,o);
input [2:0]s;
input [N-1:0] a,b,c,d,e,f,g,h;

output [N-1:0]o;

assign o = (s == 3'b000) ? a:
     (s == 3'b001) ? b:
     (s == 3'b010) ? c:
     (s == 3'b011) ? d:
     (s == 3'b100) ? e:
     (s == 3'b101) ? f:
     (s == 3'b110) ? g:
      h;

endmodule
```

Top Level

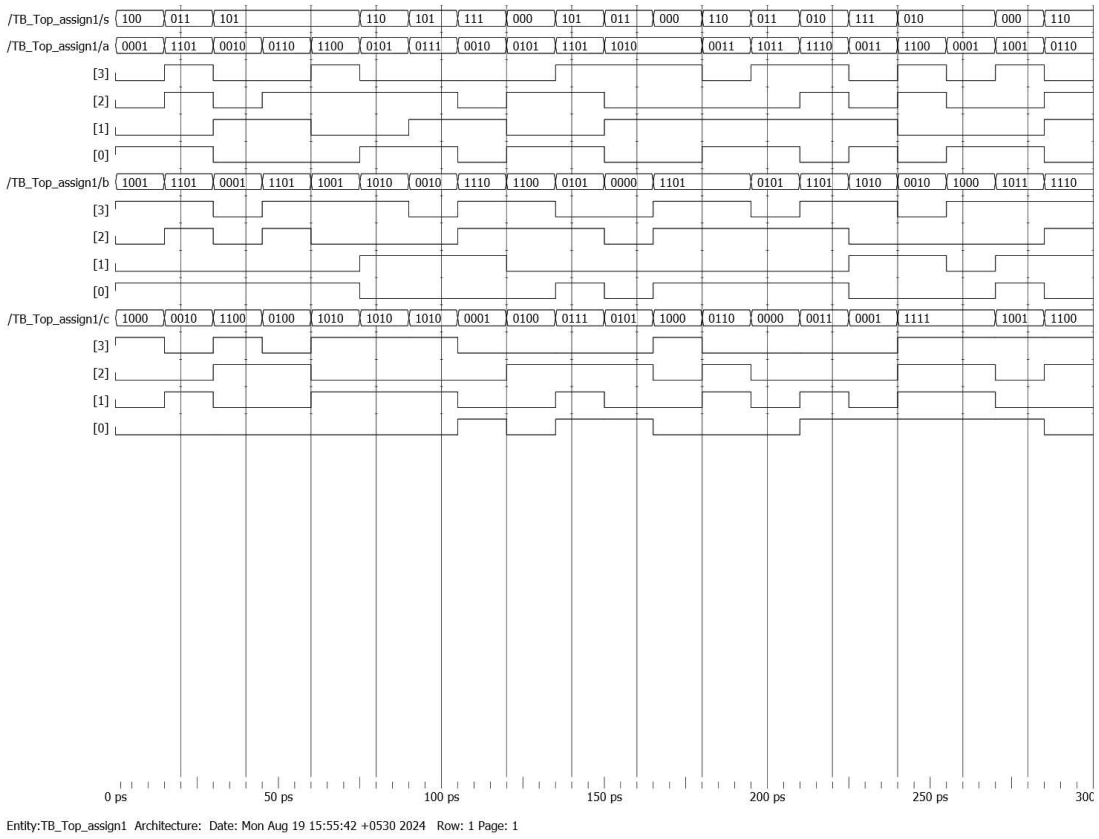
```
module Top_assign1#(parameter N = 8) (s,a,b,c);
input [2:0]s;
input [N-1:0] a,b;
output [N-1:0]c;

wire [N-1:0] w1,w2,w3,w4,w5,w6,w7,w8;

And_2in_Nbit #(.N(N)) And (.c(w1),.a(a),.b(b));
Or_2in_Nbit #(.N(N)) Or (.c(w2),.a(a),.b(b));
Nand_2in_Nbit #(.N(N)) Nand (.c(w3),.a(a),.b(b));
Nor_2in_Nbit #(.N(N)) Nor (.c(w4),.a(a),.b(b));
Xor_2in_Nbit #(.N(N)) Xor (.c(w5),.a(a),.b(b));
Xnor_2in_Nbit #(.N(N)) Xnor (.c(w6),.a(a),.b(b));
Mul_by2_Nbit #(.N(N)) Mul (.y(w7),.a(a));
Div_by2_Nbit #(.N(N)) Div (.b(w8),.a(a));

Mux_8x1_Nbit #(.N(N)) Mux (.s(s),.a(w1),.b(w2),.c(w3),.d(w4),.e(w5),.f(
   w6),.g(w7),.h(w8),.o(c));

endmodule
```

Test Bench

```
module TB_Top_assign1();

  parameter M = 4;

  reg [2:0]s;
  reg [M-1:0]a;
  reg [M-1:0]b;
  wire [M-1:0]c;

  Top_assign1 #(.N(M)) DUT(.a(a),.s(s),.b(b),.c(c));

  initial

  begin

/*    s = 3'b000; a = 4'b1001; b = 4'b1010;
    #10
    s = 3'b001; a = 4'b0100; b = 4'b1000;
    #10
    s = 3'b010; a = 4'b1001; b = 4'b1010;
    #10
    s = 3'b011; a = 4'b0100; b = 4'b1000;
    #10
    s = 3'b100; a = 4'b1110; b = 4'b0111;
    #10
    s = 3'b101; a = 4'b1110; b = 4'b0111;
    #10
    s = 3'b110; a = 4'b0011; b = 4'b0000;
    #10
    s = 3'b111; a = 4'b0110; b = 4'b0000;
    #10;
*/
repeat(20)
begin
s = $random();
a = $random();
b = $random();
```

```
38 #10;
39 #5 $display("%b, %b, %b, %b", s, a, b, c );
40 end
41 $finish;
42 end
43 initial
44
45
46
47 begin
48 $monitor("s = %b,a = %b,b = %b,c = %b", s, a, b, c );
49 end
50 endmodule
```

5 Simulation Results and Discussions



s = 100,a = 0001,b = 1001,c = 1000

100, 0001, 1001, 1000

s = 011,a = 1101,b = 1101,c = 0010

011, 1101, 1101, 0010

s = 101,a = 0010,b = 0001,c = 1100

101, 0010, 0001, 1100

s = 101,a = 0110,b = 1101,c = 0100

101, 0110, 1101, 0100

s = 101,a = 1100,b = 1001,c = 1010

101, 1100, 1001, 1010

s = 110,a = 0101,b = 1010,c = 1010

110, 0101, 1010, 1010

s = 101,a = 0111,b = 0010,c = 1010

101, 0111, 0010, 1010

s = 111,a = 0010,b = 1110,c = 0001
111, 0010, 1110, 0001
s = 000,a = 0101,b = 1100,c = 0100
000, 0101, 1100, 0100
s = 101,a = 1101,b = 0101,c = 0111
101, 1101, 0101, 0111
s = 011,a = 1010,b = 0000,c = 0101
011, 1010, 0000, 0101
s = 000,a = 1010,b = 1101,c = 1000
000, 1010, 1101, 1000
s = 110,a = 0011,b = 1101,c = 0110
110, 0011, 1101, 0110
s = 011,a = 1011,b = 0101,c = 0000
011, 1011, 0101, 0000
s = 010,a = 1110,b = 1101,c = 0011
010, 1110, 1101, 0011
s = 111,a = 0011,b = 1010,c = 0001
111, 0011, 1010, 0001
s = 010,a = 1100,b = 0010,c = 1111
010, 1100, 0010, 1111
s = 010,a = 0001,b = 1000,c = 1111
010, 0001, 1000, 1111
s = 000,a = 1001,b = 1011,c = 1001
000, 1001, 1011, 1001
s = 110,a = 0110,b = 1110,c = 1100
110, 0110, 1110, 1100

The circuit was found to be accurate on random inputs.

6 Conclusion

- Logic gates were implemented using verilog.
- These logic gates were connected into a circuit with a top level module.
- A test bench was made to verify the functionality and correctness of the verilog code with random inputs.
- The simulation was successful with zero errors.