Digital System Design

Submission Date: 21st Oct, 2019

1. **Write RTL VERILOG Code to implement the logic shown in figure 1 representing a full adder where A, B are 1 bit input whereas Cin 1 bit carry. Write another module for implementing a 4-bit adder, use "module for figure-1" to implement the four bit adder shown in figure 2. Write a stimulus to test the 4-bit adder.**

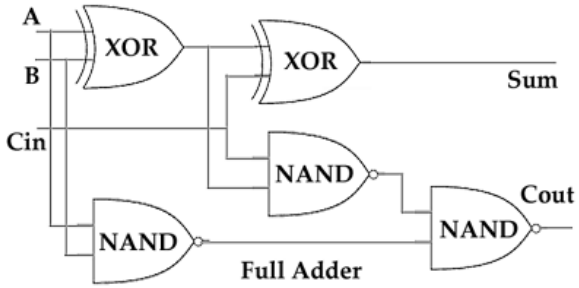


Figure 1: Logic representing Full-Adder

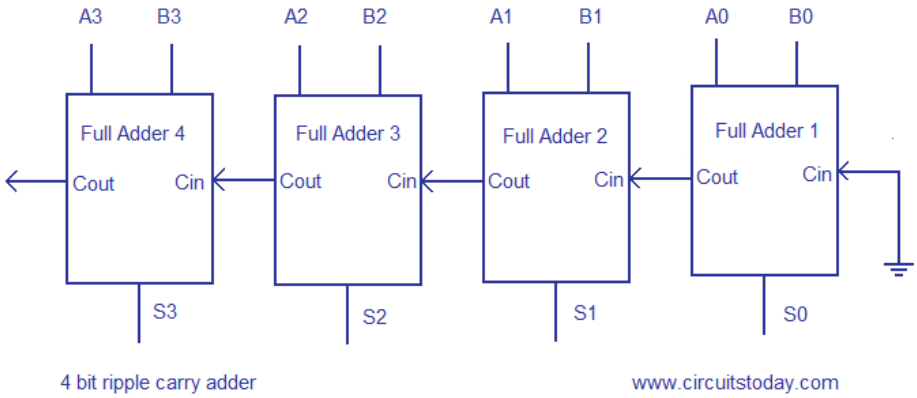


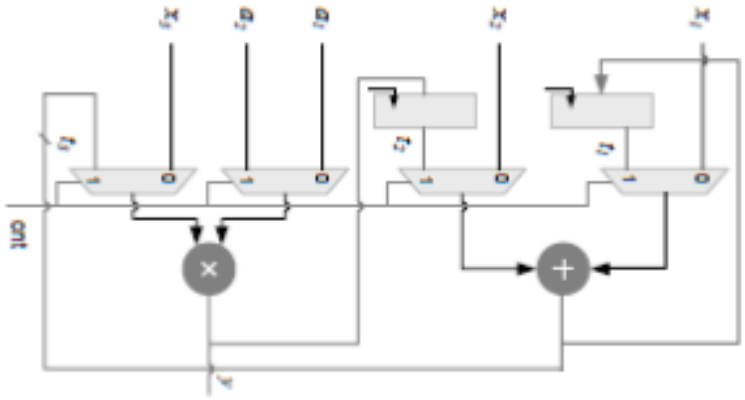
Figure 2: 4-bit Carry Ripple Adder

2. **Write RTL Verilog Code to implement Multiply Accumulator (MAC) architecture. The design implements the following:**

$$Acc = A * B + C * D + Acc$$

The signals A, B, C, D and Acc are 8, 8, 8, 8 and 32-bit wide unsigned numbers respectively. Verify your logic using a stimulus.

3. **Write RTL VERILOG code to implement the design shown below. Also design a stimulus to verify your RTL code of the design. Assume all inputs as 3 bit wide and output as 6 bit wide.**



4. **4. Draw RTL diagram representing the following VERILOG code. Clearly specify all the details (signals, bus width, directions etc) in the RTL diagram.**

```
module Question_4(a, b, sel, clk, rst_n, c);
input [6:0] a, b;
input clk, rst_n;
input [2:0] sel;
output reg [6:0] c;
reg [6:0] x1, x2, x3, t_x1, t_x2, t_x3;
always @ (posedge clk or negedge rst_n)
begin
if (!rst_n)
begin
x1<= 0; x2<=0; x3<=0;
end
else
begin
x1<= t_x1; x2<= t_x2; x3<= t_x3;
end
end

always@(*)
begin
        if (sel == 0)
        begin
                t_x1 <= x2;
                t_x2 <= x1;
                t_x3 <= 1;
        end
        else if (sel == 1)
        begin
                t_x1 <= x3;
                t_x2 <= x2;
                t_x3 <= x2;
        end
```

```
        else
        begin
                t_x1 <= 0;
                t_x2 <= x3;
                t_x3 <= 0;
        end
end
always@(*)
begin
        case(|(a&b))
        0: c = x1<<2;
        1: c = x2<<2;
        endcase
endmodule
```

5. **Define a module to design an 8-function ALU that takes two 4-bit numbers a and b and computes a 5-bit result out based on a 3-bit select signal. Ignore overflow or underflow bits.**

Select Signal	Function Output
3'b000	a
3'b001	a + b
3'b010	a - b
3'b011	a / b
3'b100	a % b (remainder)
3'b101	a << 1
3'b110	a >> 1
3'b111	(a > b) (magnitude compare)

NOTE:

- Code should be properly ***indented*** and ***commented***.
- Write your ***code in MODELSIM***, verify it using ***stimulus*** and ***attach output screen shots of your simulation*** in a separate docx file. Code/Screen shots should not be copied, ***in case of copying assignment will not be marked.***
- MODELSIM Projects (VERILOG files and all) along with screen shots of your simulation results should be ***emailed as a single ".rar" file*** on assignments.sajid@gmail.com.
- ***RAR*** file should be ***named*** as ***"YOURNAME_ASSIGN#_DSD"***.
- Subject of your ***email MUST*** be as ***"YOURNAME_ASSIGN#_DSD"***.
- ***VERY IMPORTANT:*** RTL diagrams should either be drawn in VISIO for software submission, for submission in hard form ***NEATLY*** draw the RTL diagrams with your ***HANDS.***