يوسف أحمد محمد ابراهيم

Assignment 2

Circuit_1 :

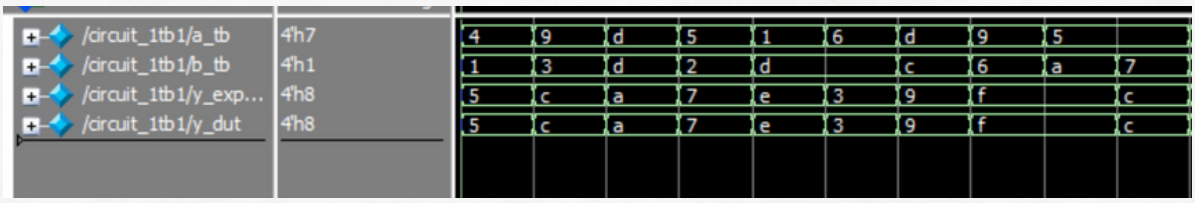
- **ALU module:**

```
module circuit_1#(parameter N = 4, parameter op = 0)
        (input [N-1:0] a, b,
        output [N:0]out);

    assign out = (op == 1) ? (a | b):
                 (op == 2) ? (a - b):
                 (op == 3) ? (a ^ b):
                 (a + b);
endmodule
```

- **Testbench1:**

```
module circuit_1tb1();
    parameter N = 4;
    reg [N-1:0]a_tb, b_tb, y_expected;
    wire [N-1:0]y_dut;

    circuit_1 #(.op(0)) dut(a_tb, b_tb, y_dut);
    initial begin
        repeat(100) begin
            a_tb = $random;
            b_tb = $random;
            y_expected = a_tb + b_tb;
            #10;
            if(y_dut != y_expected) begin
                $display("Error, a = %d, b = %d, expected value: %d, actual output: %d" ,a_tb, b_tb, y_expected, y_dut);
                $stop;
            end
        end
    end
endmodule
```

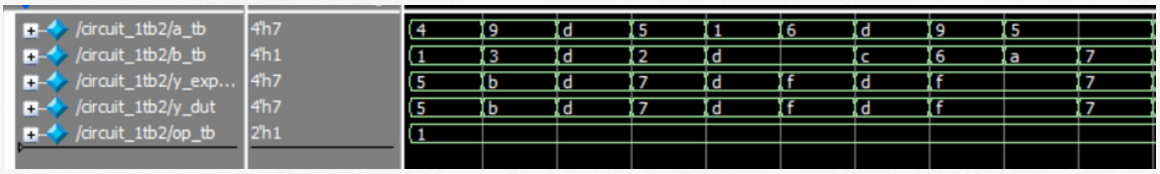
- **Output1 (100ns)**



- **Testbench2:**

```
21  module circuit_1tb2();
22      parameter N = 4;
23      reg [N-1:0]a_tb, b_tb, y_expected;
24      wire [N-1:0]y_dut;
25
26      circuit_1 #(.op(1)) dut(a_tb, b_tb, y_dut);
27      initial begin
28          repeat(100) begin
29              a_tb = $random;
30              b_tb = $random;
31              y_expected = a_tb | b_tb;
32              #10;
33              if(y_dut != y_expected) begin
34                  $display("Error, a = %d, b = %d, expected value: %d, actual output: %d" ,a_tb, b_tb, y_expected, y_dut);
35                  $stop;
36              end
37          end
38      end
39  endmodule
```

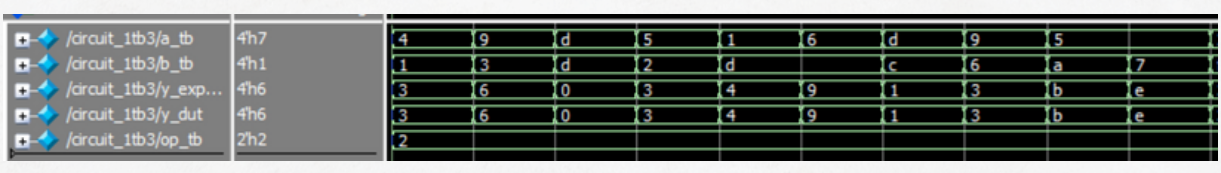
- **Output2(100ns)**



- **Testbench3:**

```
41  module circuit_1tb3();
42      parameter N = 4;
43      reg [N-1:0]a_tb, b_tb, y_expected;
44      wire [N-1:0]y_dut;
45
46      circuit_1 #(.op(2)) dut(a_tb, b_tb, y_dut);
47      initial begin
48          repeat(100) begin
49              a_tb = $random;
50              b_tb = $random;
51              y_expected = a_tb - b_tb;
52              #10;
53              if(y_dut != y_expected) begin
54                  $display("Error, a = %d, b = %d, expected value: %d, actual output: %d" ,a_tb, b_tb, y_expected, y_dut);
55                  $stop;
56              end
57          end
58      end
59  endmodule
```

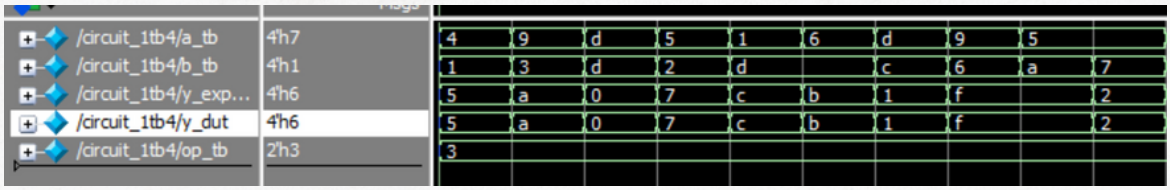
- **Output3 (100ns)**



- **Testbench4:**

```
module circuit_1tb4();
    parameter N = 4;
    reg [N-1:0]a_tb, b_tb, y_expected;
    wire [N-1:0]y_dut;

    circuit_1 #(.op(3)) dut(a_tb, b_tb, y_dut);
    initial begin
        repeat(100) begin
            a_tb = $random;
            b_tb = $random;
            y_expected = a_tb ^ b_tb;
            #10;
            if(y_dut != y_expected) begin
                $display("Error, a = %d, b = %d, expected value: %d, actual output: %d" ,a_tb, b_tb, y_expected, y_dut);
                $stop;
            end
        end
    end
endmodule
```

- **Output4(100ns)**

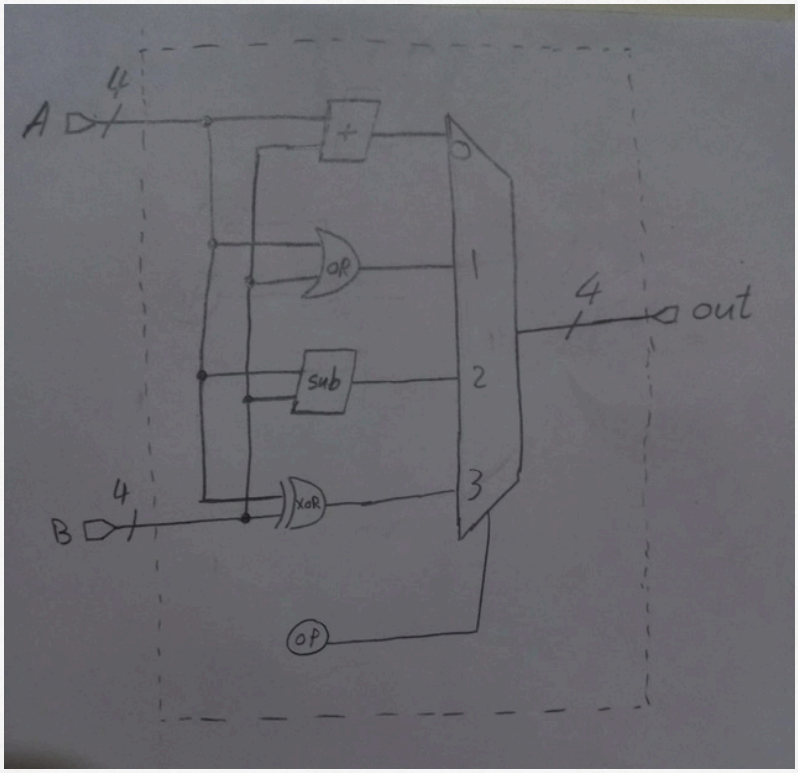


Circuit_2:

- code

```verilog
module circuit_2#(parameter N = 4, parameter op = 0)
        (input [N-1:0] a, b,
        input clk, rst,
        output [N:0]out);

    always @(posedge clk, posedge rst) begin
        if (rst)
            out = 0;
        else begin
            case (op)
                2'b1 : out = a | b;
                2'b2 : out = a - b;
                2'b3 : out = a ^ b;
                default : out = a + b;
            endcase
        end
    end
endmodule
```

- schematic:

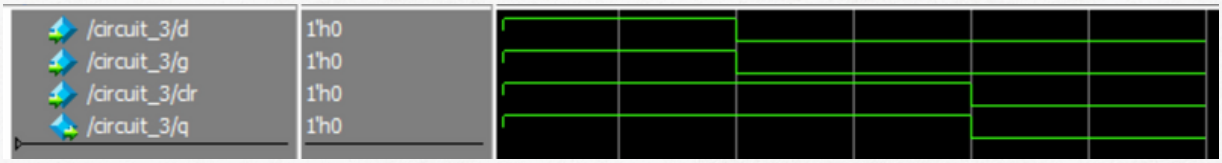


Circuit_3:

- code

```
1  module circuit_3(input d, g, clr, output q);
2      always @(*) begin
3          if(~clr)
4              q = 0;
5          else if(g)
6              q = d;
7      end
8  endmodule
```

- output:

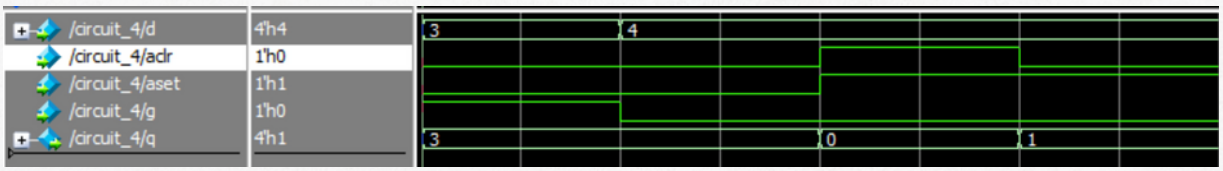


Circuit_4:

- code

```
1   module circuit_4#(parameter N = 4)
2                    (input [N-1:0]d,
3                    input aclr, aset, g,
4                    output [N-1:0]q);
5
6       always @(*) begin
7           if(aclr)
8               q = 0;
9           else if(aset)
10              q = 1;
11          else if(g)
12              q = d;
13      end
14  endmodule
```

- output:



Circuit_5a:

- code

```
module circuit_5a(input t, rstn, clk, output reg q, q_bar);
    always @(posedge clk) begin
        if (~rstn)
            q <= 0;
        else if(t)
            q <= ~q;
    end
    assign q_bar = ~q;
endmodule
```

Circuit_5b:

- code

```
module circuit_5b(input d, rstn, clk, output reg q, q_bar);
    always @(posedge clk) begin
        if (~rstn)
            q <= 0;
        else
            q <= d;
    end
    assign q_bar = ~q;
endmodule
```

Circuit_5c:

- code

```
21  module circuit_5c#(parameter FF_TYPE = "DFF")
22                    (input d, clk, rstn,
23                    output reg q, q_bar);
24      always @(posedge clk) begin
25          if (~rstn)
26              q <= 0;
27          else if (FF_TYPE == "TFF")
28              if(d)
29                  q <= ~  q;
30          else
31              q <= d;
32
33      end
34      assign q_bar = ~q;
35  endmodule
```