Assignment_5

(under supervision of Eng/Karim Wassim)

Question 1]

```
1   module question1(A,B,Cin,opcode,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst,out,leds);
2
3   parameter INPUT_PRIORITY ="A";
4   parameter FULL_ADDER="ON";
5
6   input [2:0] A,B,opcode;
7   input Cin,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst;
8
9   reg [2:0] A_ff,B_ff,opcode_ff;
10  reg Cin_ff,serial_in_ff,direction_ff,red_op_A_ff,red_op_B_ff,bypass_A_ff,bypass_B_ff;
11
12  output reg [5:0] out;
13  output reg [15:0] leds;
14
15
16  always @(posedge clk) begin
17      A_ff<=A;
18      B_ff<=B;
19      opcode_ff<=opcode;
20      Cin_ff<=Cin;
21      serial_in_ff<=serial_in;
22      direction_ff<=direction;
23      red_op_A_ff<=red_op_A;
24      red_op_B_ff<=red_op_B;
25      bypass_A_ff<=bypass_A;
26      bypass_B_ff<=bypass_B;
27  end
28
29  always @(posedge clk or posedge rst) begin
30      if(rst) begin
31          out<=0;
32          leds<=0;
33      end
34   else if(bypass_A_ff==1 && bypass_B_ff==0) begin
35          out<=A_ff;
36          leds<=0;
37          end
38      else if(bypass_A_ff==0 && bypass_B_ff==1) begin
39          out<=B_ff;
40          leds<=0;
41          end
42      else if(bypass_A_ff==1 && bypass_B_ff==1) begin
43          out<=A_ff;
44          leds<=0;
45          end
46      else begin
47          case(opcode_ff)
48          3'b000: begin
49              if(red_op_A_ff==1 && red_op_B_ff==0) begin
50              out<=&A_ff;
51              leds<=0;
52              end
53              else if(red_op_A_ff==0 && red_op_B_ff==1) begin
54              out<=&B_ff;
55              leds<=0;
```

Figure (1) : design code

```
                end
                else if(red_op_A_ff==1 && red_op_B_ff==1) begin
                out<=&A_ff;
                leds<=0;
                end
                else begin
                out<=A_ff & B_ff;
                leds<=0;
                end
            end
            3'b001: begin
                if(red_op_A_ff==1 && red_op_B_ff==0) begin
                out<=^A_ff;
                leds<=0;
                end
                else if(red_op_A_ff==0 && red_op_B_ff==1) begin
                out<=^B_ff;
                leds<=0;
                end
                else if(red_op_A_ff==1 && red_op_B_ff==1) begin
                out<=^A_ff;
                leds<=0;
                end
                else begin
                out<=A_ff ^ B_ff;
                leds<=0;
                end
            end
            3'b010: begin
              case({red_op_A_ff,red_op_B_ff})
                2'b11,2'b01,2'b10: begin
                     out<=0;
                     leds<=~leds;
                end
                2'b00: begin
                if(FULL_ADDER=="ON")
                out<=A_ff+B_ff+Cin_ff;
                else
                out<=A_ff+B_ff;

                leds<=0;
                end
              endcase
            end
            3'b011: begin
              case({red_op_A_ff,red_op_B_ff})
                2'b11,2'b01,2'b10: begin
                     out<=0;
                     leds<=~leds;
                end
                2'b00: begin
                out<= A_ff * B_ff;
                leds<=0;
                end
                    endcase
                end
                3'b100: begin
                   case({red_op_A_ff,red_op_B_ff})
                     2'b11,2'b01,2'b10: begin
                          out<=0;
                          leds<=~leds;
                     end
                     2'b00: begin
                   if(direction)
                     out<={out[4:0],serial_in_ff};
                   else
                     out<={serial_in_ff,out[5:1]};

                     leds<=0;
                     end
                  endcase
                end
                3'b101: begin
                   case({red_op_A_ff,red_op_B_ff})
                     2'b11,2'b01,2'b10: begin
                          out<=0;
                          leds<=~leds;
                     end
                     2'b00: begin
                   if(direction)
                     out<={out[4:0],out[5]};
                   else
                     out<={out[0],out[5:1]};
                     leds<=0;
                    end
                endcase
                end
                3'b110,3'b111: begin
                          out<=0;
                          leds<=~leds;
                end
      endcase
        end
    end
    endmodule
```

Figure (2) : continue design code

```
module question1_tb();

parameter N1 ="A";
parameter N2="ON";

reg [2:0] A,B,opcode;
reg Cin,serial_in,direction,red_op_A,red_op_B,bypass_A,bypass_B,clk,rst;

wire [5:0] out;
wire [15:0] leds;
integer i=0;

question1 #(.INPUT_PRIORITY(N1),.FULL_ADDER(N2)) DUT(.A(A),.B(B),.opcode(opcode),.Cin(Cin),.serial_in(serial_in),.direction(direction),.red_op_A(red_op_A),.red_op_B(red_op_B),.bypass_A(

initial begin
    clk=0;
    forever
    #25 clk=~clk;  // this mean that clock period will be 50 ns
end

initial begin
    rst=1;
    #100;
    rst=0;

    // directed testing for inputs bypass_A & bypass_B

    opcode=000; A=5; B=1; bypass_A=1; bypass_B=0;  // we expect output=A=5 (don't care the value of opcode)
    #50;
    opcode=001; A=2; B=3; bypass_A=0; bypass_B=1;  // we expect output=B=3 (don't care the value of opcode)
    #50;
    opcode=010; A=7; B=4; bypass_A=1; bypass_B=1;  // we expect output=A=7 (as A has higher piriority than B)
    #50;

    bypass_A=0; bypass_B=0;

    // directed testing for invalid cases


    opcode=010; A=4; B=6; red_op_A=1; red_op_B=0;   // we expect out=0 & leds will toggling
    #50;
    opcode=011; A=5; B=3; red_op_A=0; red_op_B=1;   // we expect out=0 & leds will toggling
    #50;
    opcode=100; A=2; B=1; red_op_A=1; red_op_B=1;   // we expect out=0 & leds will toggling
    #50;
    opcode=101; A=0; B=7; red_op_A=1; red_op_B=1;   // we expect out=0 & leds will toggling
    #50;
    opcode=110; A=3; B=2; red_op_A=1; red_op_B=0;  // we expect out=0 & leds will toggling (don't care to the input values)
    #50;
    opcode=111; A=6; B=1; red_op_A=0; red_op_B=1;  // we expect out=0 & leds will toggling  (don't care to the input values)
    #50;

    red_op_A=0;  red_op_B=0;

        for(i=0;i<20;i=i+1) begin        //randomized testing to check the validity of the operations

            @(negedge clk);
            A=$random;
            B=$random;
            Cin=$random;
            serial_in=$random;
            direction=$random;
            opcode=$urandom_range(0,5);

        end

        $stop;

    end

    endmodule
```

Figure (3) :testbench code

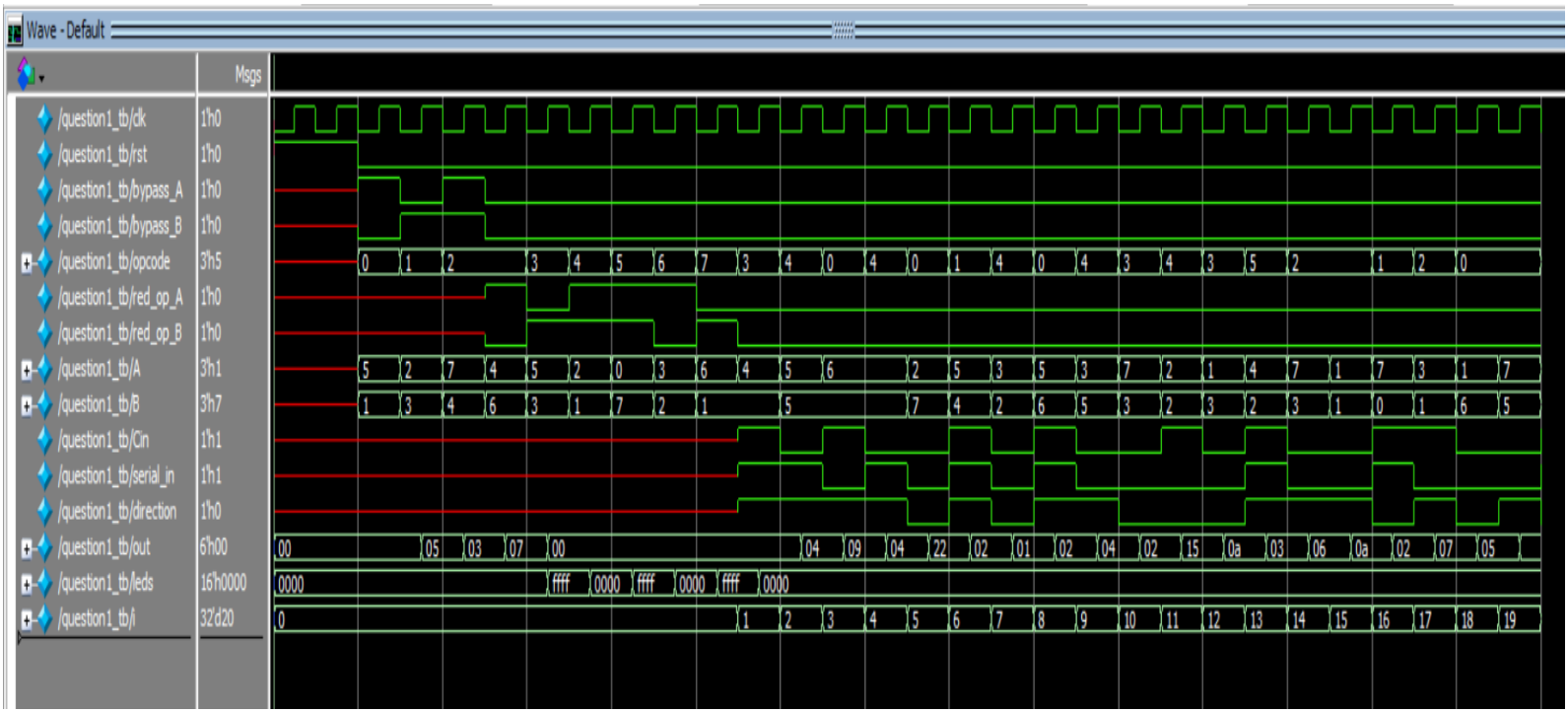

Figure (4) : waveform snippet

```
1  vlib work
2  vlog design.v testbench.v
3  vsim -voptargs=+acc work.question1_tb
4  add wave *
5  run -all
```

Figure (5) : do file

➔ **We test this design using both methods (directed testing & randomized testing) as we use directed testing for the asynchronous inputs as reset and for the inputs that I want to test them separately few times as bypass_A & bypass_B & red_op_A & red_op_B.**

➔ **The waveform shown above in figure (4) shows all the corner cases that I tested in the testbench. But we notice that there is a delay with one clock cycle due to the inputs that I get from the flip flop before performing any operation.**

Question 2]

design.v | testbench.v | sim_file.do

```
1  module question2(A,B,C,D,clk,rst_n,P);
2
3  parameter OPERATION="ADD";
4
5  input [17:0] A,B,D;
6  input [47:0] C;
7  input clk,rst_n;
8  output reg [47:0] P;
9
10 reg [47:0] multiplier_out;
11
12 always @(posedge clk) begin
13
14  if(!rst_n) begin
15     P<=0;
16     multiplier_out<=0;
17  end
18  else begin
19     if(OPERATION=="ADD") begin
20         multiplier_out<= (D + B) * A;
21         P<= multiplier_out + C;
22     end
23     else begin
24         multiplier_out<= (D - B) * A;
25         P<= multiplier_out - C;
26     end
27  end
28
29 end
30
31 endmodule
```

Figure (6): design code

```
1  vlib work
2  vlog design.v testbench.v
3  vsim -voptargs=+acc work.question2_tb
4  add wave *
5  run -all
```

Figure (7): do file

```
1  module question2_tb();
2
3  parameter N="ADD";
4
5  reg clk,reset;
6  reg [17:0] A,B,D;
7  reg [47:0] C;
8  wire [47:0] P;
9  integer i=0;
10
11 question2 #(.OPERATION(N)) DUT (.A(A),.B(B),.C(C),.D(D),.clk(clk),.rst_n(reset),.P(P));
12
13 initial begin
14     clk=0;
15     forever
16     #25 clk=~clk;
17 end
18
19 initial begin
20
21 reset=0;
22 #100;
23 reset=1;
24 for(i=0;i<15;i=i+1) begin
25
26     @(negedge clk);
27     A=$urandom_range(0,1000);
28     B=$urandom_range(0,1000);
29     C=$urandom_range(0,1000);
30     D=$urandom_range(0,1000);
31 end
32 $stop;
33 end
34 endmodule
```

Figure (8): testbench code

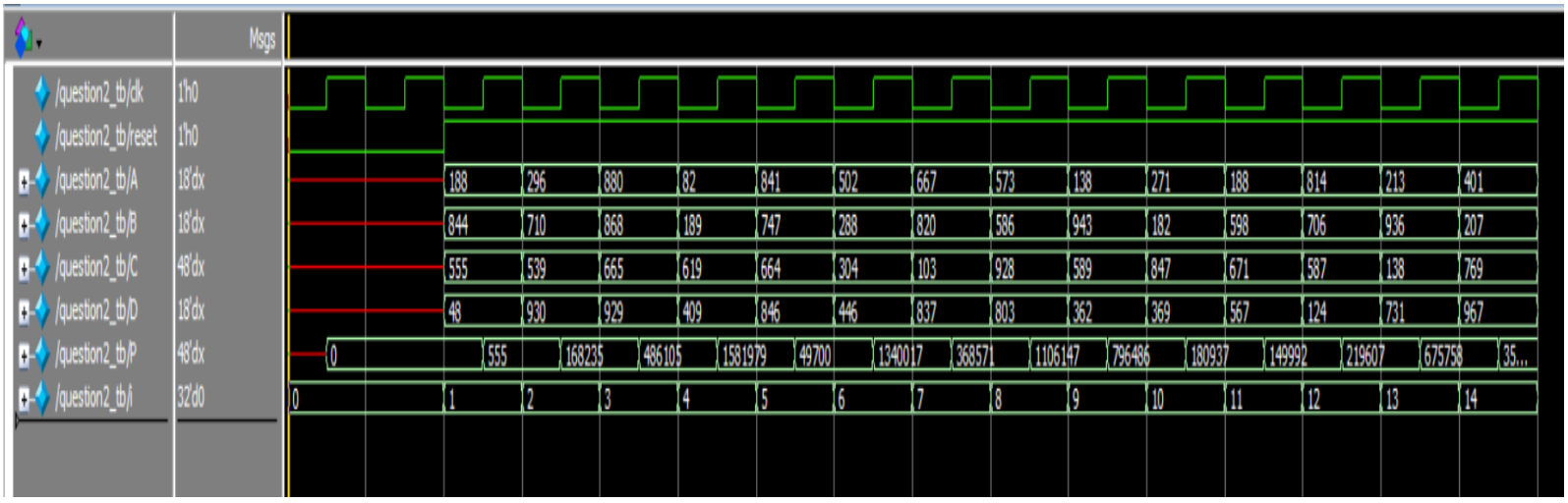

Figure (9): waveform snippet