Abdalrahman Ali Eltaher

Group 2

Assignment_4_extra

- Q[1]

➢ Code :

Assignment_4_Extra > Q1_code.v

```
//dff_rst
module dff_rst(rst,clk,d,q);
input rst,clk,d;
output reg q;
always @(posedge clk or posedge rst) begin
    if(rst)
        q<=0;
    else
        q<=d;
end
endmodule
//xor
module xor_comb(a,b,c);
input a,b;
output c;
assign c = a^b;
endmodule
//dff_set
module dff_set(set,clk,d,q);
input set,clk,d;
output reg q;
always @(posedge clk or posedge set) begin
    if(set)
        q<=1;
    else
        q<=d;
end
endmodule
module LASER(clk,rst,set,out);
input set,rst,clk;
output [3:0]out ;
wire [3:0]internal_q;
wire in_xor;
//set
dff_set ff1(.clk(clk),.set(set),.d(internal_q[3]),.q(internal_q[0]));
//xor
xor_comb comb(.a(internal_q[0]),.b(internal_q[3]),.c(in_xor));
//rst
dff_rst ff2 (.clk(clk),.rst(rst),.d(in_xor),.q(internal_q[1]));
dff_rst ff3 (.clk(clk),.rst(rst),.d(internal_q[1]),.q(internal_q[2]));
dff_rst ff4 (.clk(clk),.rst(rst),.d(internal_q[2]),.q(internal_q[3]));
assign out = internal_q ;
endmodule
```

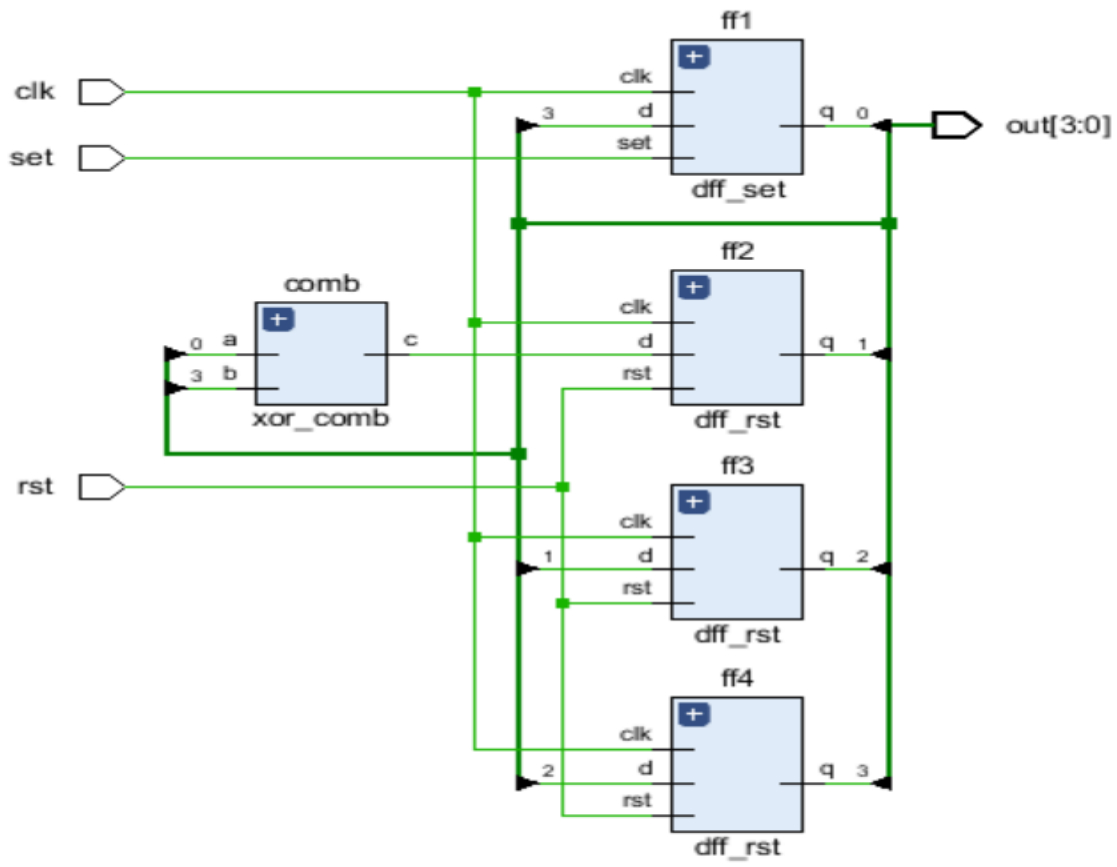
➢ Testbench:

Assignment_4_Extra > Q1_tb.v

```
module LASER_tb();
reg clk_tb,rst_tb,set_tb;
reg [3:0]out_expect;
wire [3:0] out_dut;
LASER dut(clk_tb,rst_tb,set_tb,out_dut);
initial begin
    clk_tb=0;
    forever
    #1 clk_tb =~clk_tb;
end
initial begin
    set_tb=1;rst_tb=1;out_expect=4'b0001;
    @(negedge clk_tb);
    if(out_expect!=out_dut)begin
        $display("erorr: out_expect=%h,out_dut=%h",out_expect,out_dut);
        $stop;
    end
    set_tb=0;rst_tb=0;
    repeat(50)begin
        @(negedge clk_tb);out_expect=4'b0010;
        @(negedge clk_tb);out_expect=4'b0100;
        @(negedge clk_tb);out_expect=4'b1000;
        @(negedge clk_tb);out_expect=4'b0011;
        @(negedge clk_tb);out_expect=4'b0110;
        @(negedge clk_tb);out_expect=4'b1100;
        @(negedge clk_tb);out_expect=4'b1011;
        @(negedge clk_tb);out_expect=4'b0101;
        @(negedge clk_tb);out_expect=4'b1010;
        @(negedge clk_tb);out_expect=4'b0111;
        @(negedge clk_tb);out_expect=4'b1110;
        @(negedge clk_tb);out_expect=4'b1111;
        @(negedge clk_tb);out_expect=4'b1101;
        @(negedge clk_tb);out_expect=4'b1001;
        @(negedge clk_tb);out_expect=4'b0001;
    end
    $stop;
end
endmodule
```

➢ Wave:

➢ Schematic:



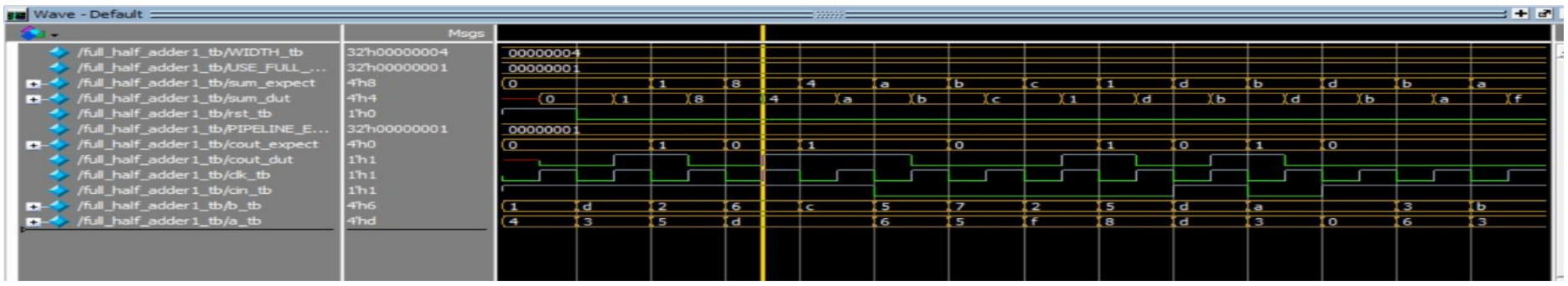
- Q [2]
- Code:

```
Assignment_4_Extra > Q2_code.v
1   module full_half_adder(a,b,clk,cin,rst,sum,cout);
2   parameter WIDTH = 4,PIPELINE_ENABLE=1,USE_FULL_ADDER=1;
3   input clk,rst,cin;
4   input [WIDTH-1:0] a,b;
5   output reg [WIDTH-1:0]sum;
6   output reg cout;
7   generate
8       if(PIPELINE_ENABLE)begin
9           always @(posedge clk) begin
10              if(rst)begin
11                  sum<=0;
12                  cout<=0;
13              end
14              else begin
15                  if(USE_FULL_ADDER)
16                      {cout,sum}<=a+b+cin;
17                  else 1
18                      {cout,sum}<=a+b;
19              end
20          end
21      end
22      else begin
23          always @(*) begin
24          if(USE_FULL_ADDER)
25              {cout,sum}=a+b+cin;
26          else
27              {cout,sum}=a+b;
28          end
29      end
30  endgenerate
31  endmodule
```

- **Testbench_1**: Where PIPELINE_ENABLE =1 and USE_FULL_ADDER=1

```
Assignment_4_Extra > Q2_tb1.v
1   module full_half_adder1_tb();
2   parameter WIDTH_tb = 4,PIPELINE_ENABLE_tb=1,USE_FULL_ADDER_tb=1;
3   reg [WIDTH_tb-1:0] a_tb,b_tb,cout_expect,sum_expect;
4   reg clk_tb,rst_tb,cin_tb;
5   wire [WIDTH_tb-1:0]sum_dut;
6   wire cout_dut;
7   full_half_adder #(WIDTH_tb,PIPELINE_ENABLE_tb,USE_FULL_ADDER_tb)dut (a_tb,b_tb,clk_tb,cin_tb,rst_tb,sum_dut,cout_dut);
8   initial begin
9       clk_tb=0;
10      forever
11      #1 clk_tb=~clk_tb;
12  end
13  initial begin
14      rst_tb= 1;a_tb=$random;b_tb=$random;cin_tb=$random;cout_expect=0;sum_expect=0;
15      @(negedge clk_tb);
16      if({cout_dut,sum_dut}!={cout_expect,sum_expect})begin
17          $display("error :cout_dut=%h,sum_dut=%h, cout_expect=,sum_expect",cout_dut,sum_dut,cout_expect,sum_expect);
18          $stop;
19      end
20      rst_tb = 0 ;
21      repeat(100)begin
22          a_tb=$random;b_tb=$random;cin_tb=$random;
23          @(negedge clk_tb);
24          {cout_expect,sum_expect}=a_tb+b_tb+cin_tb;
25          if({cout_dut,sum_dut}!={cout_expect,sum_expect})begin
26              $display("error :cout_dut=%h,sum_dut=%h, cout_expect=,sum_expect",cout_dut,sum_dut,cout_expect,sum_expect);
27              $stop;
28          end
29      end
30      $stop;
31  end
32  endmodule
```

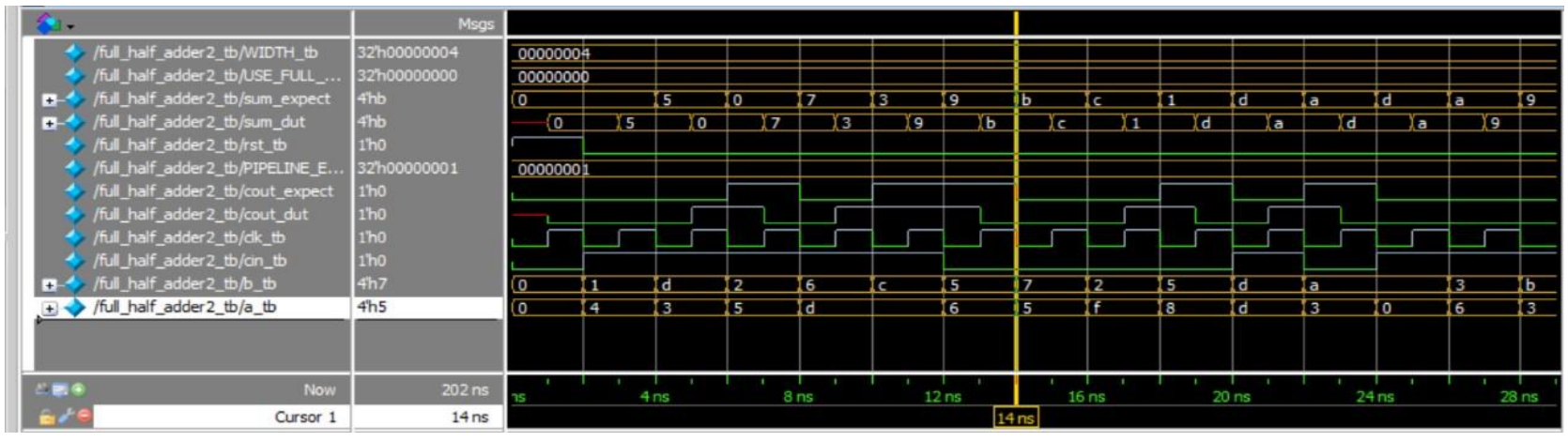
- Wave tb_1:



- **Testbench_2** : Where PIPELINE_ENABLE =1 and USE_FULL_ADDER=0

```
Assignment_4_Extra > Q2_tb2.v
1   module full_half_adder2_tb();
2   parameter WIDTH_tb = 4,PIPELINE_ENABLE_tb=1,USE_FULL_ADDER_tb=0;
3   reg [WIDTH_tb-1:0] a_tb,b_tb,sum_expect;
4
5   reg clk_tb,rst_tb,cin_tb,cout_expect;
6   wire [WIDTH_tb-1:0]sum_dut;
7   wire cout_dut;
8   full_half_adder #(WIDTH_tb,PIPELINE_ENABLE_tb,USE_FULL_ADDER_tb)dut (a_tb,b_tb,clk_tb,cin_tb,rst_tb,sum_dut,cout_dut);
9   initial begin
10      clk_tb=0;
11      forever
12      #1 clk_tb=~clk_tb;
13  end
14  initial begin
15      rst_tb= 1;a_tb=0;b_tb=0;cin_tb=0;cout_expect=0;sum_expect=0;
16      @(negedge clk_tb);
17      if({cout_dut,sum_dut}!={cout_expect,sum_expect})begin
18          $display("error :cout_dut=%h,sum_dut=%h, cout_expect=,sum_expect",cout_dut,sum_dut,cout_expect,sum_expect);
19          $stop;
20      end
21      rst_tb = 0 ;
22      repeat(100)begin
23          a_tb=$random;b_tb=$random;cin_tb=$random;
24          @(negedge clk_tb);
25          {cout_expect,sum_expect}=a_tb+b_tb;
26          if({cout_dut,sum_dut}!={cout_expect,sum_expect})begin
27              $display("error :cout_dut=%h,sum_dut=%h, cout_expect=,sum_expect",cout_dut,sum_dut,cout_expect,sum_expect);
28              $stop;
29          end
30      end
31      $stop;
32  end
33  endmodule
```

- Wave tb_2:



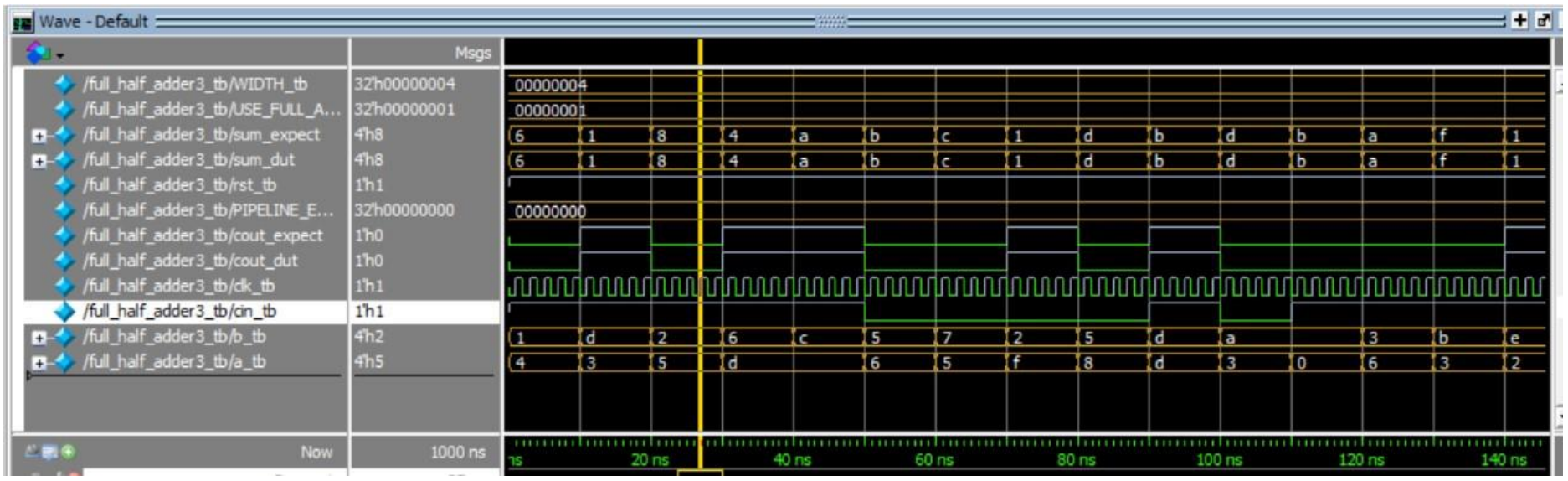
➢ **Testbench_3**: Where PIPELINE_ENABLE = 0 and USE_FULL_ADDER=1

Assignment_4_Extra > Q2_tb3.v

```
module full_half_adder3_tb();
parameter WIDTH_tb = 4,PIPELINE_ENABLE_tb=0,USE_FULL_ADDER_tb=1;
reg [WIDTH_tb-1:0] a_tb,b_tb,sum_expect;

reg clk_tb,rst_tb,cin_tb,cout_expect;
wire [WIDTH_tb-1:0]sum_dut;
wire cout_dut;
full_half_adder #(WIDTH_tb,PIPELINE_ENABLE_tb,USE_FULL_ADDER_tb)dut (a_tb,b_tb,clk_tb,cin_tb,rst_tb,sum_dut,cout_dut);
initial begin
    clk_tb=0;
    forever
    #1 clk_tb=~clk_tb;
end
initial begin
    rst_tb= 1;a_tb=0;b_tb=0;cin_tb=0;cout_expect=0;sum_expect=0;
    repeat(100)begin
        a_tb=$random;b_tb=$random;cin_tb=$random;
        {cout_expect,sum_expect}=a_tb+b_tb+cin_tb;
        #10;
        if({cout_dut,sum_dut}!={cout_expect,sum_expect})begin
            $display("error :cout_dut=%h,sum_dut=%h, cout_expect=,sum_expect",cout_dut,sum_dut,cout_expect,sum_expect);
            $stop;
        end
    end
    $stop;
end
endmodule
```

➢ Wave tb_3:



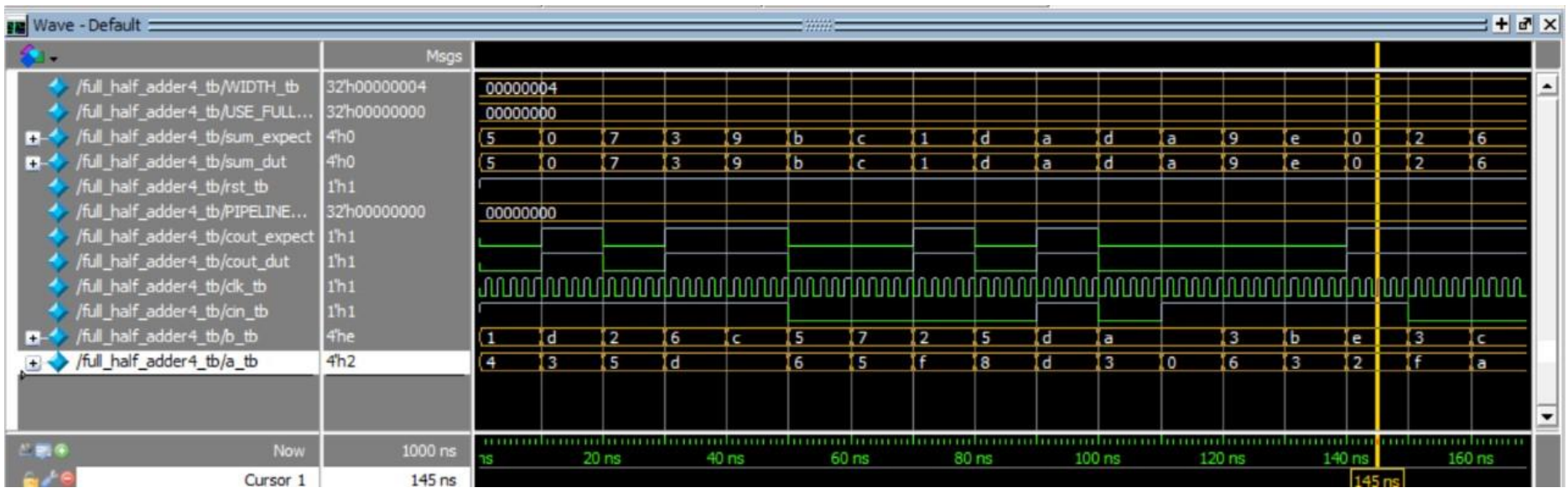
➢ **Testbench_4**: Where PIPELINE_ENABLE = 0 and USE_FULL_ADDER=0

Assignment_4_Extra > Q2_tb4.v

```
module full_half_adder4_tb();
parameter WIDTH_tb = 4,PIPELINE_ENABLE_tb=0,USE_FULL_ADDER_tb=0;
reg [WIDTH_tb-1:0] a_tb,b_tb,sum_expect;

reg clk_tb,rst_tb,cin_tb,cout_expect;
wire [WIDTH_tb-1:0]sum_dut;
wire cout_dut;
full_half_adder #(WIDTH_tb,PIPELINE_ENABLE_tb,USE_FULL_ADDER_tb)dut (a_tb,b_tb,clk_tb,cin_tb,rst_tb,sum_dut,cout_dut);
initial begin
    clk_tb=0;
    forever
    #1 clk_tb=~clk_tb;
end
initial begin
    rst_tb= 1;a_tb=0;b_tb=0;cin_tb=0;cout_expect=0;sum_expect=0;
    repeat(100)begin
        a_tb=$random;b_tb=$random;cin_tb=$random;
        {cout_expect,sum_expect}=a_tb+b_tb;
        #10;
        if({cout_dut,sum_dut}!={cout_expect,sum_expect})begin
            $display("error :cout_dut=%h,sum_dut=%h, cout_expect=,sum_expect",cout_dut,sum_dut,cout_expect,sum_expect);
            $stop;
        end
    end
    $stop;
end
endmodule
```

➢ Wave tb_4:



- Q [3]
  - Code:

Assignment_4_Extra > Q3_code.v

```v
module shift_register(clk,rst,load,load_value,PO);
parameter SHIFT_DIRECTION = "LEFT" , SHIFT_AMOUNT = 1 ;
input clk,rst,load;
input[7:0] load_value;
output reg [7:0] PO ;
always @(posedge clk or posedge rst) begin
    if(rst)
        PO<=0;
    else begin
        if(load)
            PO<=load_value;
        else begin
            if(SHIFT_DIRECTION=="LEFT")
                PO<=PO << SHIFT_AMOUNT;
            else
                PO<=PO>>SHIFT_AMOUNT;
        end
    end
end
endmodule
```

  - Testbench_1: SHIFT_DIRECTION_tb = "LEFT"SHIFT_AMOUNT_tb= 1

Assignment_4_Extra > Q3_tb1.v

```v
module shift_register1_tb();
parameter SHIFT_DIRECTION_tb = "LEFT" , SHIFT_AMOUNT_tb = 1 ;
reg clk_tb,rst_tb,load_tb;
reg [7:0] load_value_tb,PO_expect;
wire  [7:0] PO_dut;
shift_register #(SHIFT_DIRECTION_tb,SHIFT_AMOUNT_tb) dut(clk_tb,rst_tb,load_tb,load_value_tb,PO_dut);
initial begin
    clk_tb=0;
    forever
    #1 clk_tb=~clk_tb;
end
initial begin
    rst_tb=1;PO_expect=0;load_tb=0;load_value_tb=0;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    rst_tb = 0 ;load_tb=1;
    repeat (10)begin
        load_value_tb = $random;
        PO_expect = load_value_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    load_tb=0;
     repeat (10)begin
        PO_expect = PO_expect<<SHIFT_AMOUNT_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    load_tb=1;
    repeat (10)begin
        load_value_tb = $random;
        PO_expect = load_value_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    load_tb=0;
     repeat (10)begin
        PO_expect = PO_expect<<SHIFT_AMOUNT_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    $stop;
end
endmodule
```

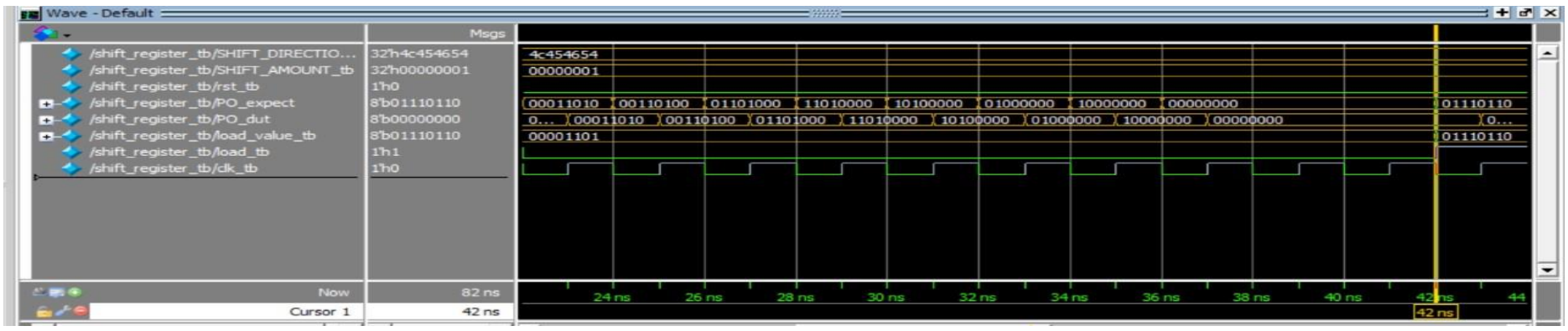
  - Testbench_2: SHIFT_DIRECTION_tb = "RIGHT" , SHIFT_AMOUNT_tb = 2

Assignment_4_Extra > Q3_tb2.v

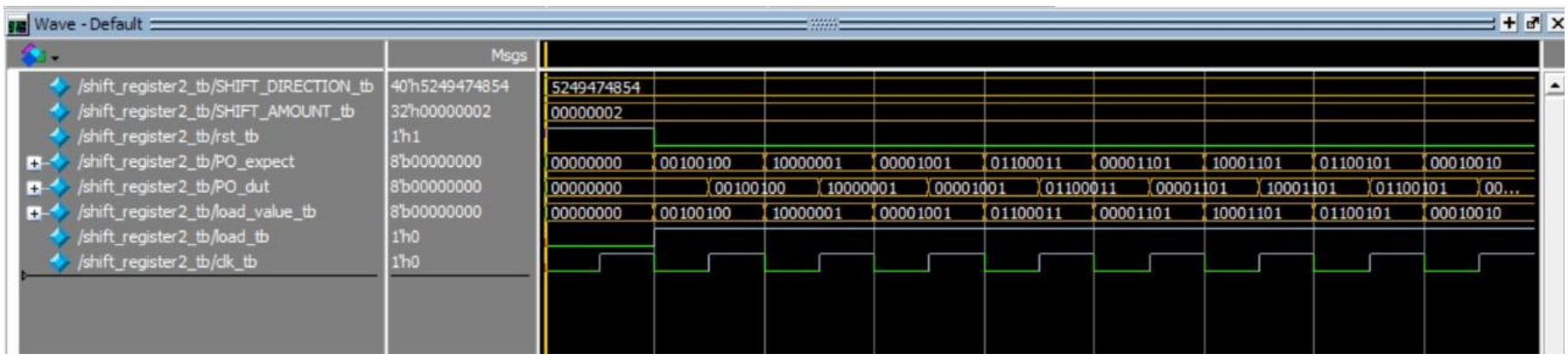
```v
module shift_register2_tb();
parameter SHIFT_DIRECTION_tb = "RIGHT" , SHIFT_AMOUNT_tb = 2 ;
reg clk_tb,rst_tb,load_tb;
reg [7:0] load_value_tb,PO_expect;
wire  [7:0] PO_dut;
shift_register #(SHIFT_DIRECTION_tb,SHIFT_AMOUNT_tb) dut(clk_tb,rst_tb,load_tb,load_value_tb,PO_dut);
initial begin
    clk_tb=0;
    forever
    #1 clk_tb=~clk_tb;
end
initial begin
    rst_tb=1;PO_expect=0;load_tb=0;load_value_tb=0;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    rst_tb = 0 ;load_tb=1;
    repeat (10)begin
        load_value_tb = $random;
        PO_expect = load_value_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    load_tb=0;
     repeat (10)begin
        PO_expect = PO_expect>>SHIFT_AMOUNT_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    load_tb=1;
    repeat (10)begin
        load_value_tb = $random;
        PO_expect = load_value_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    load_tb=0;
     repeat (10)begin
        PO_expect = PO_expect>>SHIFT_AMOUNT_tb;
        @(negedge clk_tb);
        if(PO_dut!=PO_expect)begin
            $display("error : PO_dut=%h,PO_expect=%h",PO_dut,PO_expect);
            $stop;
        end
    end
    $stop;
end
endmodule
```

➢ Wave_tb1 as load =1 ;

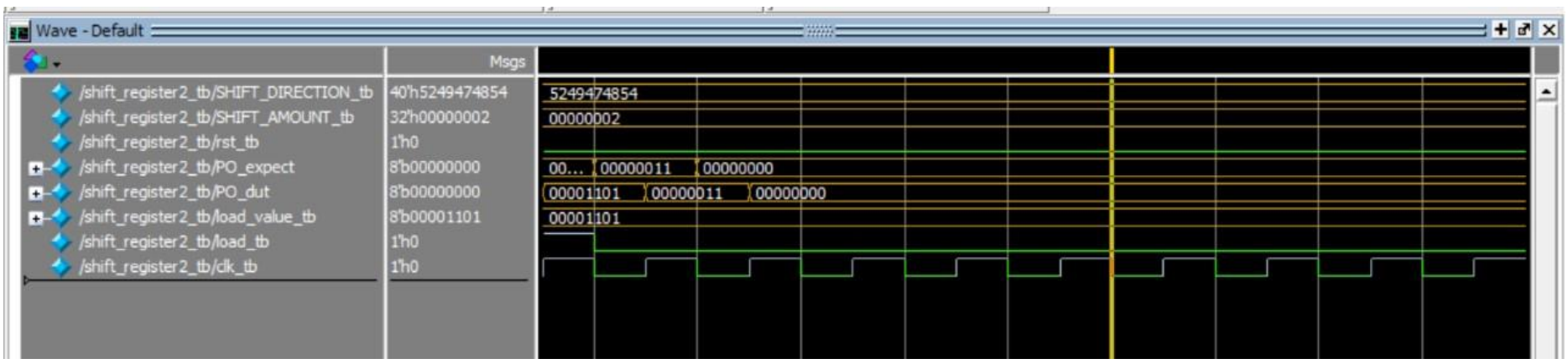
➢ Wave_tb1 as load =0;



➢ Wave tb2 as load =0 :



➢ Wave tb2 as load = 1 ;



#FINALLY