Assignment_4

(under supervision of Eng/ Karim Waseem)

Question1]

design.v testbench.v

```
module q1(clk,sset,sclr,aset,aclr,enable,load,shiftin,data,load_avalue,load_svalue,q,shiftout);
parameter SHIFT_WIDTH=8;
parameter SHIFT_DIRECTION="LEFT";
input clk,sset,sclr,aset,aclr,enable,load,shiftin;
input [SHIFT_WIDTH-1:0] data,load_avalue,load_svalue;
output reg shiftout;
output [SHIFT_WIDTH-1:0] q;
reg [SHIFT_WIDTH-1:0] temp;
always @(posedge clk or posedge aset or posedge aclr) begin
    if(aclr) begin
    temp<=0;
    shiftout<=0;
    end
    else if(aset)
    temp<=load_avalue;
    else if(enable) begin
        if(sclr) begin
            temp<=0;
            shiftout<=0;
        end
    else if(sset)
        temp<=load_svalue;
    else if(load)
    temp<=data;
    else begin
        case(SHIFT_DIRECTION)
        "LEFT" : begin
            temp<={q[SHIFT_WIDTH-2:0],shiftin};
            shiftout<=temp[SHIFT_WIDTH-1];
        end
        "RIGHT" : begin
            temp<={shiftin,q[SHIFT_WIDTH-1:1]};
            shiftout<=temp[0];
        end
        endcase
    end
    end
end
assign q=temp;
endmodule
```

Figure (1) : design code

design.v testbench.v

```
module q1_tb();
parameter N=8;
parameter shift="LEFT";
reg clk,sset,sclr,aset,aclr,en,load,shiftin;
reg [N-1:0] d,load_a,load_s;
wire [N-1:0] q;
wire shiftout;
integer i;

q1 #(.SHIFT_WIDTH(N),.SHIFT_DIRECTION(shift))DUT(.clk(clk),.sset(sset),.sclr(sclr),.aset(aset),.aclr(aclr),.load(load),.enable(en),.data(d),.q(

initial begin
clk=0;
forever
#25 clk=~clk;
end

initial begin
aclr=1; sclr=0; en=0;
#50;
aclr=0; sclr=1; en=1;
#50;
aclr=0; sclr=0; en=1;
for(i=0;i<20;i=i+1) begin
@(negedge clk);
sset=$random;
aset=$random;
load=$random;
shiftin=$random;
d=$random;
load_a=$random;
load_s=$random;
end
$stop;
end

initial
$monitor("sset=%b\t aset=%b\t load=%b\t shiftin=%b\t data=%b\t load_avalue=%b\t load_svalue=%b\t output(q)=%b\t shiftout=%b\t",sset,aset,load,s
endmodule
```

Figure (2) : testbench code

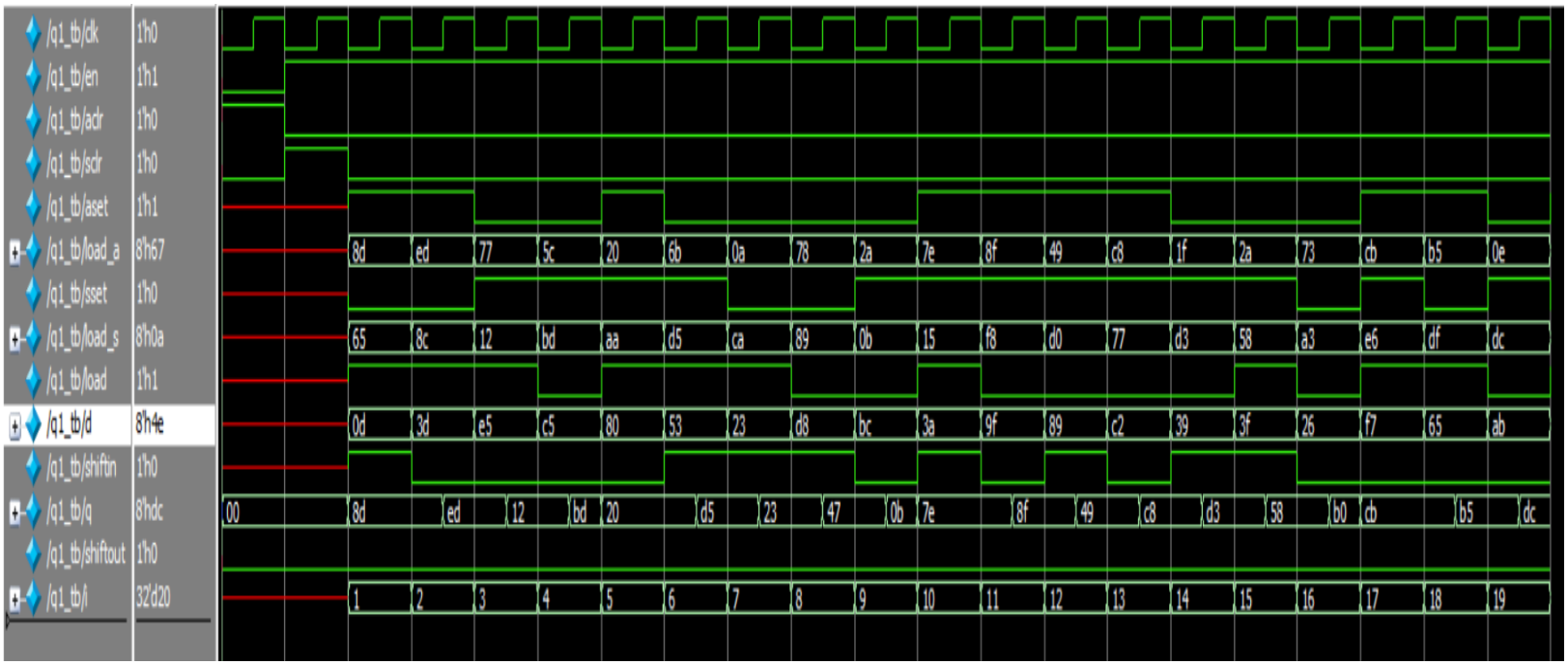


Figure (3) : waveform snippet

Question2]

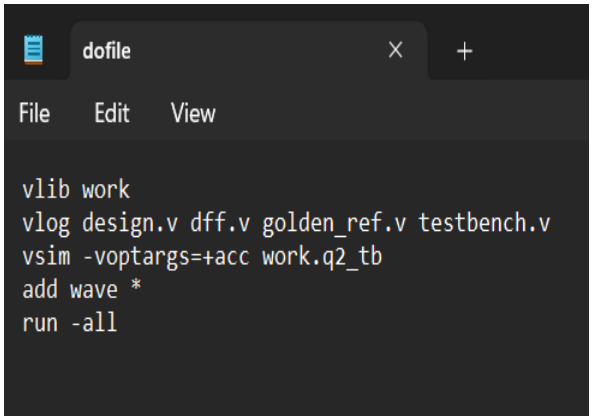


Figure (5) : Do file

```
module q2(clk,set,out);

input clk,set;
output reg [3:0] out;

always @(posedge clk or negedge set) begin

    if(!set)
        out<=4'b1111;
    else
        out<=out+1;

end

endmodule
```

Figure (4) : design code

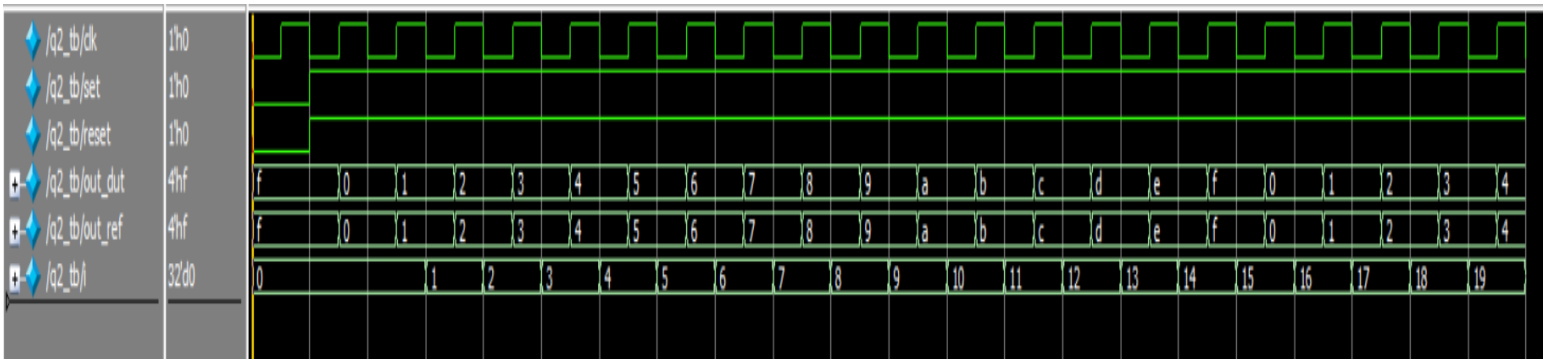


Figure (6) : waveform

```
module q2_tb();

reg clk,set,reset;
wire [3:0] out_dut;
wire [3:0] out_ref;
integer i=0;

q2 DUT (.clk(clk),.set(set),.out(out_dut));
counter REF (.RST_N(reset),.CLK(clk),.out(out_ref));

initial begin
    clk=0;
    forever
    #25 clk=~clk;
end

initial begin
    set=0;   reset=0;
    #50;
    set=1;   reset=1;
        #100;
   for(i=0;i<20;i=i+1) begin
     @(negedge clk);
     if(out_dut!==out_ref) begin
        $display("Error, this design is incorrect!!");
        $stop;
     end
    end
     $stop;
end

initial
$monitor("reset=%b\tset=%b\tout_refrence=%b\t out_dut=%b",reset,set,out_ref,out_dut);

endmodule
```

Figure (7) : testbench code

```
module counter (RST_N,CLK,out);

input RST_N,CLK;
output [3:0] out;
wire w1,w2,w3,w4;

dff f1(.d(out[0]),.clk(CLK),.rst_n(RST_N),.q(w1),.qbar(out[0]));

dff f2(.d(out[1]),.clk(w1),.rst_n(RST_N),.q(w2),.qbar(out[1]));

dff f3(.d(out[2]),.clk(w2),.rst_n(RST_N),.q(w3),.qbar(out[2]));

dff f4(.d(out[3]),.clk(w3),.rst_n(RST_N),.q(w4),.qbar(out[3]));


endmodule
```

Figure (8) : golden reference design

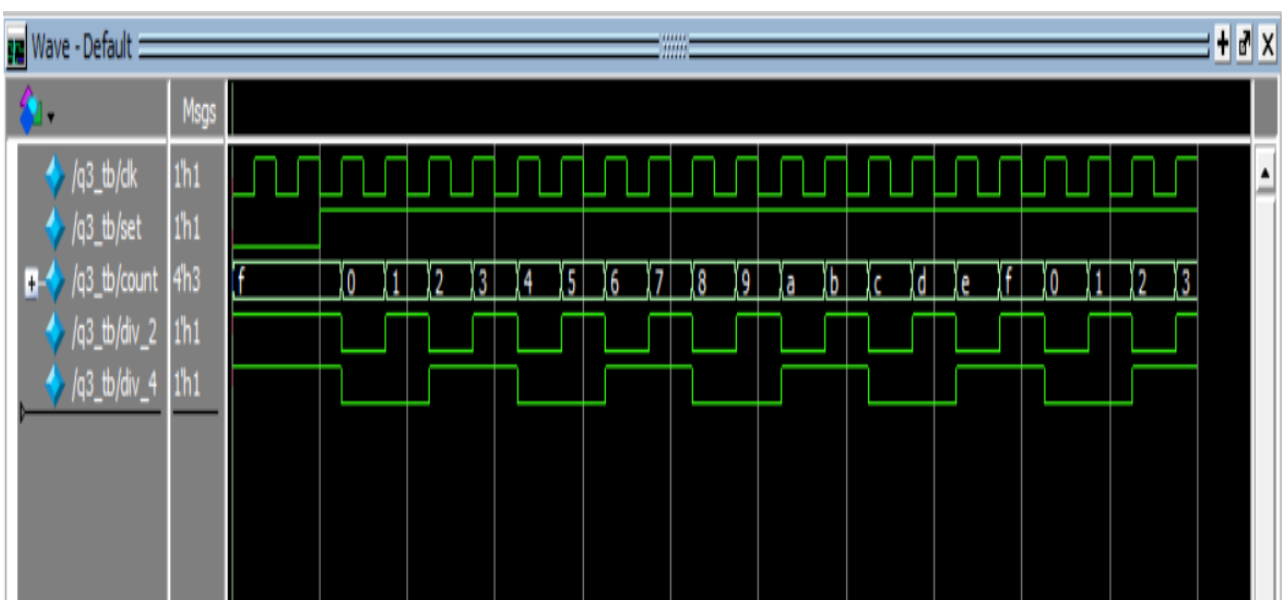
```
module dff(d,clk,rst_n,q,qbar);

input d,clk,rst_n;
output reg q;
output qbar;

always @(posedge clk or negedge rst_n) begin

if(~rst_n)
q<=0;
else
q<=d;

end

assign qbar=~q;

endmodule
```

Figure (9) : D flip flop code used by golden reference design

Question3]

```
module q3(clk,set,out,div_2,div_4);

input clk,set;
output reg [3:0] out;
output div_2,div_4;

always @(posedge clk or negedge set) begin

    if(!set)
        out<=4'b1111;
    else
        out<=out+1;

end

assign div_2=out[0];      // as the period of out[0] is double the period of the input clock
assign div_4=out[1];      // as the period of out[1] is four times the period of the input clock

endmodule
```

Figure(10) : design code



Figure(11) : waveform

```
module q3_tb();

reg clk,set;
wire [3:0] count;
wire div_2,div_4;

q3 DUT (.clk(clk),.set(set),.out(count),.div_2(div_2),.div_4(div_4));

initial begin
    clk=0;
    forever
    #25 clk=~clk;

end

initial begin

set=0;
#100;
set=1;
#1000;
$stop;

end

initial
$monitor("the output counter =%b \t div_2=%b\t div_4=%b",count,div_2,div_4);

endmodule
```

Figure(12) : testbench code

Question4]

```
module q4(clk,rst,set,out);

input clk,rst,set;
output reg [3:0] out;

always @(posedge clk or posedge rst or posedge set) begin

    if(rst && set)
        out<=4'b0001;
    else begin
        out[0]<=out[3];
        out[1]<=out[3]^out[0];
        out[2]<=out[1];
        out[3]<=out[2];
    end

end

endmodule
```

Figure (13) : design code

```
module q4_tb();

reg clk,reset,set;
wire [3:0] out_dut;

q4 DUT(.clk(clk),.rst(reset),.set(set),.out(out_dut));

initial begin
clk=0;

forever
#25 clk=~clk;

end

initial begin
    reset=1;  set=1;
    #100;
    reset=0;  set=0;
    #1000;
 $stop;

end

initial
$monitor("the output =%b",out_dut);

endmodule
```

Figure (14) : testbench code

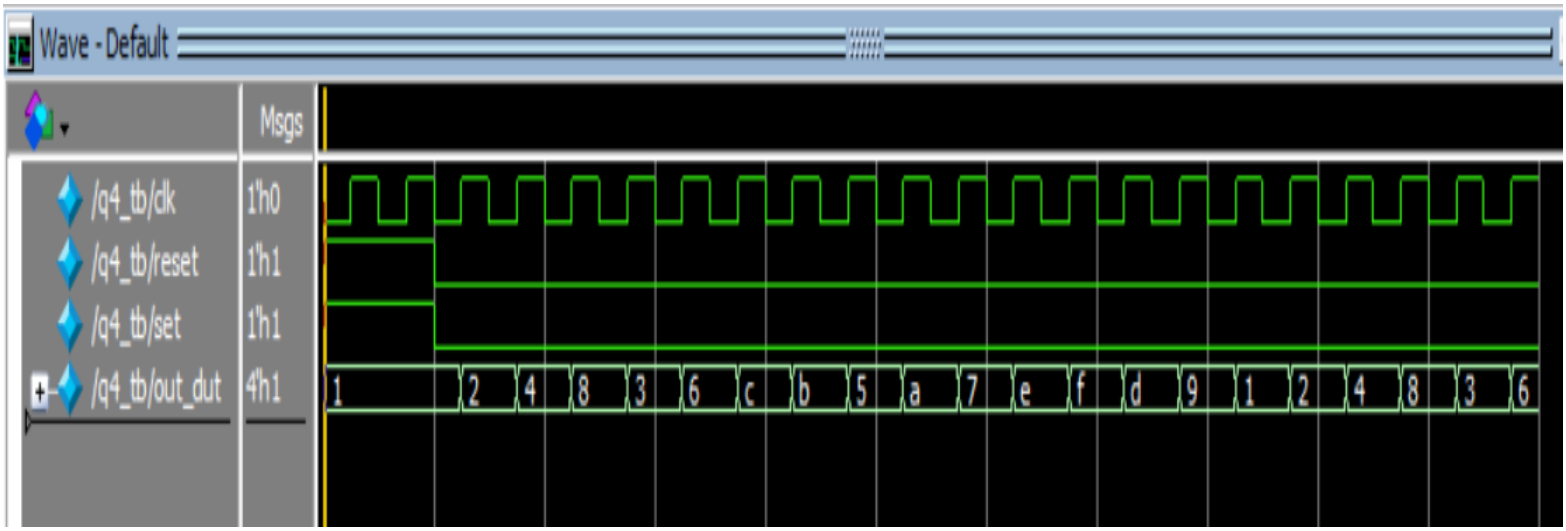


Figure (15) : waveform snippet

Question5]

```
module q5(a,b,cin,clk,rst,sum,cout);
parameter WIDTH =1;
parameter PIPELINE_ENABLE =1;
parameter USE_FULL_ADDER =1;
input [WIDTH-1:0] a,b;
input cin,clk,rst;
output [WIDTH-1:0] sum;
output cout;
reg [WIDTH-1:0] sum1;     // used for the sequential design
wire [WIDTH-1:0] sum2;    // used for pure combinational design
reg cout1;     //used for sequential design
wire cout2;     //used for combinational design
generate
if(PIPELINE_ENABLE) begin
    always @(posedge clk) begin
        if(rst) begin
            sum1<=0;
            cout1<=0;
        end
    else if(USE_FULL_ADDER) begin
         sum1<=a^b^cin;
         cout1<=(a&b)|(cin&(a^b));
       end
    else begin
        sum1<=a^b;
        cout1<=a&b;
    end
  end
end
 else begin
       if(USE_FULL_ADDER)  begin
            assign sum2=a^b^cin;
            assign cout2=(a&b)|(cin&(a^b));
         end
    else begin
          assign sum2=a^b;
          assign cout2=a&b;
      end
  end
  endgenerate
  assign sum=(PIPELINE_ENABLE)?sum1:sum2;
  assign cout=(PIPELINE_ENABLE)?cout1:cout2;
  endmodule
```

Figure (16) : design code

```
module q5_tb();

parameter N1=4;
parameter N2=1;
parameter N3=1;
reg [N1-1:0] a,b;
reg clk,rst,cin;
wire [N1-1:0] sum;
wire cout;
integer i=0;

q5 #(.WIDTH(N1),.PIPELINE_ENABLE(N2),.USE_FULL_ADDER(N3)) DUT(.a(a),.b(b),.cin(cin),.clk(clk),.rst(rst),.sum(sum),.cout(cout));

initial begin
clk=0;
forever
#25 clk=~clk;
end

initial begin
  rst=1;
  #100;
  rst=0;
  for(i=0;i<20;i=i+1) begin

    @(negedge clk);
    a=$random;
    b=$random;
    cin=$random;

  end
$stop;

end

initial
$monitor("a=%b\tb=%b\tcin=%b\tsum=%b\tcout=%b",a,b,cin,sum,cout);

endmodule
```

Figure(17) : testbench code

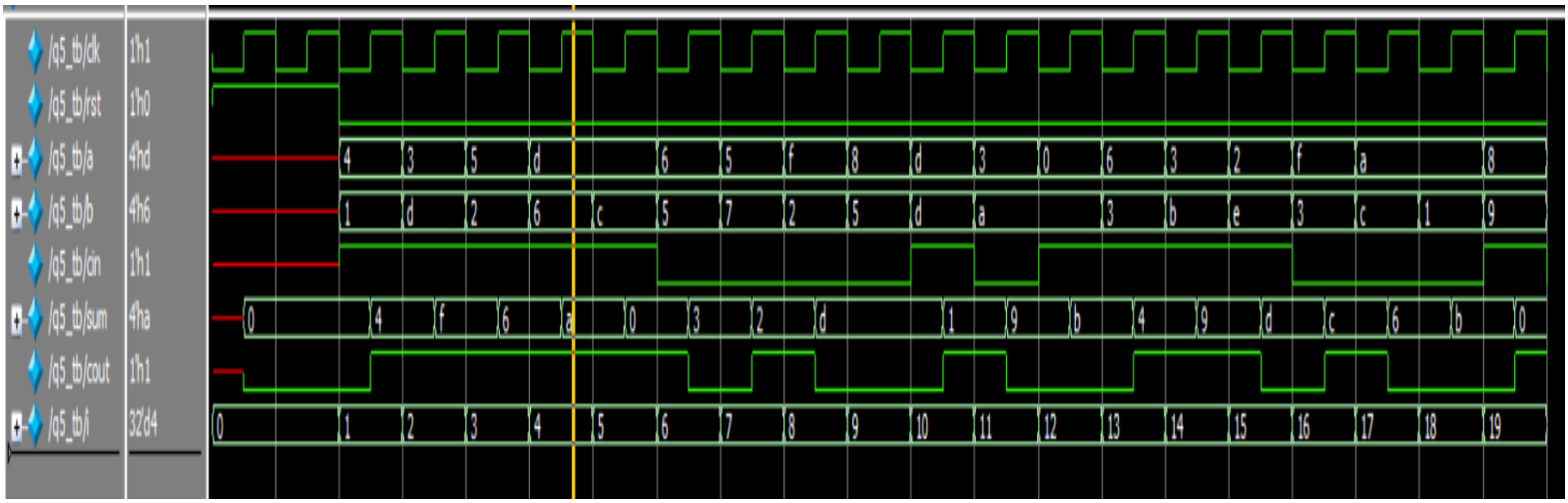


Figure (18) : waveform snippet