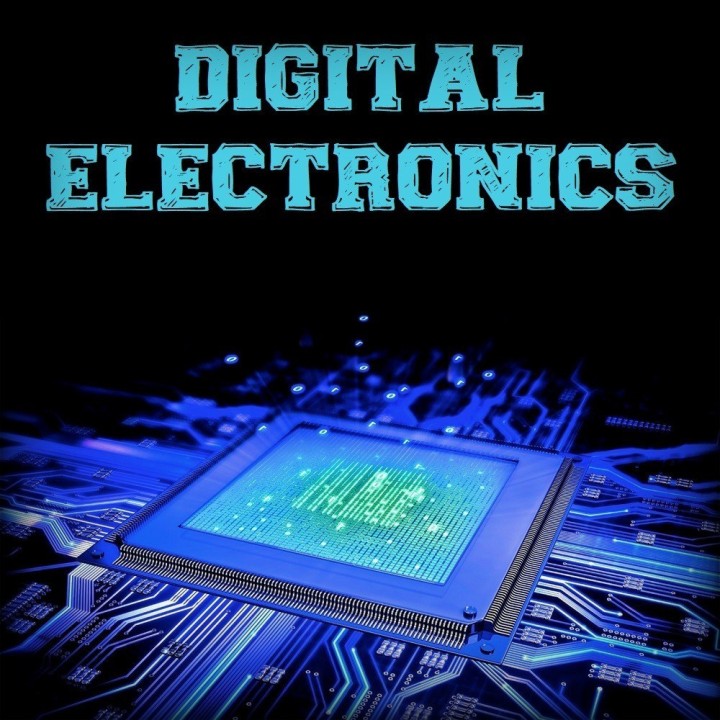
***Abanob Evram***

***Assignmen3***



A screenshot of a computer

Description automatically generated

**[Q1]**

**The design code:**

module Data\_Latch(CLR,D,G,Q);

input CLR,D,G;

output reg Q;

always @(\*) begin

if (~CLR)

Q<=0;

else if (G)

Q<=D;

end

endmodule

**The testbench code:**

module Data\_Latch\_tb();

reg CLR\_tb,G\_tb,D\_tb,Q\_expected;

wire Q\_dut;

Data\_Latch m0(CLR\_tb,D\_tb,G\_tb,Q\_dut);

integer i;

initial begin

CLR\_tb=0;

G\_tb=0;

D\_tb=0;

Q\_expected=0;

#10

if(Q\_expected!=Q\_dut) begin

$display("Errorrrr");

$stop;

end

for(i=0;i<99;i=i+1) begin

CLR\_tb=$random;

G\_tb=$random;

D\_tb=$random;

if (~CLR\_tb)

Q\_expected=0;

else if(G\_tb)

Q\_expected=D\_tb;

#10

if(Q\_expected!=Q\_dut) begin

$display("Errorrrr");

$stop;

end

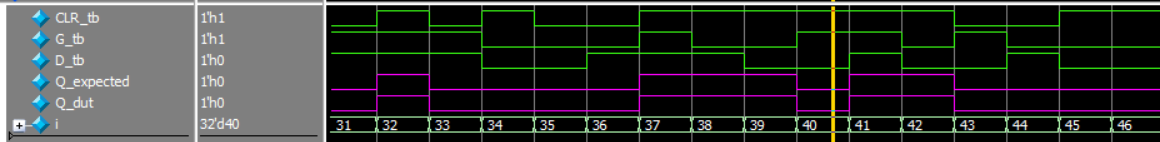
end

$stop;

end

endmodule





A white background with black text

Description automatically generated **[Q2]**

**(A)**

**The design code:**

module T\_Flipflop(t,rstn,clk,q,qpar);

input t,rstn,clk;

output reg q;

output qpar;

assign qpar = ~q ;

always @(posedge clk or negedge rstn) begin

if (~rstn)

q<=0;

else if (t)

q<=~q;

end

endmodule

**The testbench code:**

module T\_Flipflop\_tb();

reg t\_tb,rstn\_tb,clk\_tb,q\_expected;

wire q\_dut,qpar\_dut;

T\_Flipflop dut(t\_tb,rstn\_tb,clk\_tb,q\_dut,qpar\_dut);

initial begin

clk\_tb=0;

forever

#1 clk\_tb=~clk\_tb;

end

integer i;

initial begin

rstn\_tb=0;

t\_tb=0;

q\_expected=0;

@(negedge clk\_tb);

if(q\_expected!=q\_dut) $stop;

rstn\_tb=1;

for(i=0;i<1000;i=i+1)begin

t\_tb=$random;

@(posedge clk\_tb);

if(t\_tb) q\_expected=~q\_expected;

@(negedge clk\_tb);

if (q\_expected!=q\_dut) $stop;

end

$stop;

end

endmodule

A screenshot of a video game

Description automatically generated

A close up of a logo

Description automatically generated**(B)**

**The design code:**

module D\_Flipflop(d,rstn,clk,q,qpar);

input d,rstn,clk;

output reg q;

output qpar;

assign qpar = ~q ;

always @(posedge clk or negedge rstn) begin

if (~rstn)

q<=0;

else

q<=d;

end

endmodule

**The testbench code:**

module D\_Flipflop\_tb();

reg d\_tb,rstn\_tb,clk\_tb;

wire q\_dut,qpar\_dut;

D\_Flipflop dut(d\_tb,rstn\_tb,clk\_tb,q\_dut,qpar\_dut);

initial begin

clk\_tb=0;

forever

#1 clk\_tb=~clk\_tb;

end

integer i;

initial begin

rstn\_tb=0;

d\_tb=0;

@(negedge clk\_tb);

for(i=0;i<99;i=i+1) begin

rstn\_tb=$random;

d\_tb=$random;

@(negedge clk\_tb);

A screenshot of a video game

Description automatically generated end

$stop;

end

endmodule

A white background with black text

Description automatically generated**(C)**

A black and white text

Description automatically generated**The design code:**

module Flipflop(d,rstn,clk,q,qpar);

parameter FF\_TYPE="DFF";

input d,rstn,clk;

output reg q;

output qpar;

assign qpar = ~q ;

always @(posedge clk or negedge rstn) begin

if (~rstn)

q<=0;

else if (FF\_TYPE=="TFF")

q<=q^d; //when d is high then q is toggle

else

q<=d;

end

endmodule

**The testbench code:**

module Flipflop\_tb();

parameter FF\_TYPE\_tb="DFF";

reg d\_tb,rstn\_tb,clk\_tb;

wire q\_dut,qpar\_dut;

Flipflop #(FF\_TYPE\_tb) dut(d\_tb,rstn\_tb,clk\_tb,q\_dut,qpar\_dut);

initial begin

clk\_tb=0;

forever

#1 clk\_tb=~clk\_tb;

end

integer i;

initial begin

rstn\_tb=0;

d\_tb=0;

@(negedge clk\_tb);

rstn\_tb=1;

for(i=0;i<99;i=i+1) begin

d\_tb=$random;

@(negedge clk\_tb);

end

$stop;

end

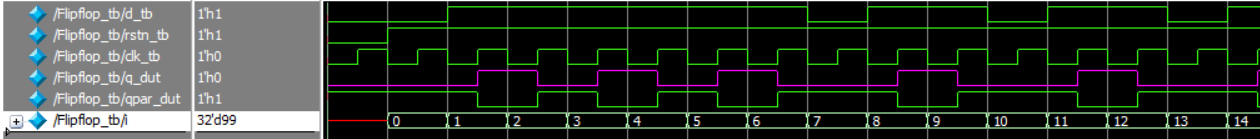
endmodule

**Wave of DFF:**

A black screen with green and purple lines

Description automatically generated

**Wave of TFF:**



A white paper with black text

Description automatically generated**(D)**

**The code of DFF:**

module Qd\_for\_DFF();

parameter FF\_TYPE\_tb="DFF";

reg d\_tb,rstn\_tb,clk\_tb;

wire q\_param,q\_golden,qpar\_param,qpar\_golden;

D\_Flipflop golden(d\_tb,rstn\_tb,clk\_tb,q\_golden,qpar\_golden);

Flipflop #(FF\_TYPE\_tb) param(d\_tb,rstn\_tb,clk\_tb,q\_param,qpar\_param);

initial begin

clk\_tb=0;

forever

#1 clk\_tb=~clk\_tb;

end

integer i;

initial begin

rstn\_tb=0;

d\_tb=0;

@(negedge clk\_tb);

if((q\_golden!=q\_param)||(qpar\_golden!=qpar\_param)) $stop;

for (i=0;i<1000;i=i+1) begin

d\_tb=$random;

rstn\_tb=$random;

@(negedge clk\_tb);

if((q\_golden!=q\_param)||(qpar\_golden!=qpar\_param)) $stop;

end

$stop;

end

endmodule

**The code of TFF:**

module Qd\_for\_TFF();

parameter FF\_TYPE\_tb="TFF";

reg d\_tb,rstn\_tb,clk\_tb;

wire q\_param,q\_golden,qpar\_param,qpar\_golden;

T\_Flipflop golden(d\_tb,rstn\_tb,clk\_tb,q\_golden,qpar\_golden);

Flipflop #(FF\_TYPE\_tb) param(d\_tb,rstn\_tb,clk\_tb,q\_param,qpar\_param);

initial begin

clk\_tb=0;

forever

#1 clk\_tb=~clk\_tb;

end

integer i;

initial begin

rstn\_tb=0;

d\_tb=0;

@(negedge clk\_tb);

if((q\_golden!=q\_param)||(qpar\_golden!=qpar\_param)) $stop;

for (i=0;i<99;i=i+1) begin

d\_tb=$random;

rstn\_tb=$random;

@(negedge clk\_tb);

if((q\_golden!=q\_param)||(qpar\_golden!=qpar\_param)) $stop;

end

$stop;

end

endmodule

**Wave of DFF:**

A screenshot of a computer game

Description automatically generated

**Wave of TFF:**

A screenshot of a video game

Description automatically generated

A diagram of a block diagram

Description automatically generated**[Q3]**

**The design code:**

module Ripple\_counter(clk,rstn,out);

input clk,rstn;

output [3:0] out;

wire q0,qn0,q1,qn1,q2,qn,q3,qn3;

D\_Flipflop DFF0(qn0,rstn,clk,q0,qn0);

D\_Flipflop DFF1(qn1,rstn,q0,q1,qn1);

D\_Flipflop DFF2(qn2,rstn,q1,q2,qn2);

D\_Flipflop DFF3(qn3,rstn,q2,q3,qn3);

assign out = {qn3,qn2,qn1,qn0};

endmodule

**The testbench code:**

module Ripple\_counter\_tb();

reg clk\_tb,rstn\_tb;

wire [3:0] out\_dut;

Ripple\_counter dut(clk\_tb,rstn\_tb,out\_dut);

initial begin

clk\_tb=0;

forever

#1 clk\_tb=~clk\_tb;

end

initial begin

rstn\_tb=0;

@(negedge clk\_tb);

rstn\_tb=1;

#200

$stop;

end

endmodule

**The do file code:**

vlib work

vlog D\_Flipflop.v Ripple\_counter.v Ripple\_counter\_tb.v

vsim -voptargs=+acc work.Ripple\_counter\_tb

add wave \*

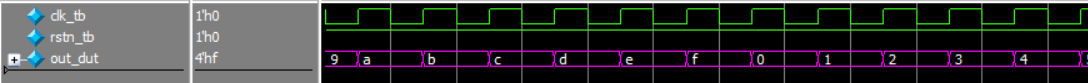
run -all

#quit -sim

A close up of text

Description automatically generated





A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated**[Q4]**

**The design code:**

module Param\_Shift\_register(sclr,sset,shiftin,load,data,clock,enable,aclr,aset,shiftout,q);

parameter SHIFT\_WIDTH=8,LOAD\_AVALUE =1,LOAD\_SVALUE=1,SHIFT\_DIRECTION="LEFT";

input sclr,sset,shiftin,load,clock,enable,aclr,aset ;

input [SHIFT\_WIDTH-1:0] data;

output shiftout;

output [SHIFT\_WIDTH-1:0] q;

reg temp\_shiftout;

reg [SHIFT\_WIDTH-1:0] temp\_q;

always @(posedge clock or posedge aclr or posedge aset) begin

if(aclr)

temp\_q <= 0;

else if (aset)

temp\_q <=LOAD\_AVALUE;

else if (enable) begin

if(sclr)

temp\_q<=0;

else if(sset)

temp\_q<=LOAD\_SVALUE;

else if(load)

temp\_q<=data;

else begin

if(SHIFT\_DIRECTION=="Right")

{temp\_q,temp\_shiftout}<={shiftin,temp\_q};

else

{temp\_shiftout,temp\_q}<={temp\_q,shiftin};

end

end

end

assign q = temp\_q ;

assign shiftout = temp\_shiftout ;

endmodule

**The testbench code:**

module Param\_Shif\_tb;

parameter SHIFT\_WIDTH=8,LOAD\_AVALUE =1,LOAD\_SVALUE=1,SHIFT\_DIRECTION="LEFT";

reg sclr,sset,shiftin,load,clock,enable,aclr,aset;

reg [SHIFT\_WIDTH-1:0] data;

wire shiftout;

wire [SHIFT\_WIDTH-1:0] q;

Param\_Shift\_register #(SHIFT\_WIDTH,LOAD\_AVALUE,LOAD\_SVALUE,SHIFT\_DIRECTION) dut(sclr,sset,shiftin,load,data,clock,enable,aclr,aset,shiftout,q);

initial begin

clock=0;

forever

#1 clock=~clock;

end

initial begin

sclr=0;sset=0;shiftin=0;load=1;data=8'b11001100;//load the nubmber to register

enable=1;aclr=0;aset=0; @(negedge clock);

load=0; @(negedge clock);//shift left with shiftin=0

shiftin =1; @(negedge clock);//shift left with shiftin=1

aclr=1; @(negedge clock);// asynchronouns clear

aclr=0;aset=1; @(negedge clock);// asynchronouns set

aset=0;sclr=1;sset=1; @(negedge clock);// synchronous clear

sclr=0; @(negedge clock);// anchronous set

$stop;

end

endmodule

A screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated**[Q5]**

**The design code:**

module SLE(D,CLK,EN,ALn,ADn,SLn,SD,LAT,Q);

input D,CLK,EN,ALn,ADn,SLn,SD,LAT;

output Q;

reg Q\_ff,Q\_latch;

always @(posedge CLK or negedge ALn ) begin

if (~ALn)

Q\_ff<=~ADn;

else if(EN&&~(LAT)) begin

if (~SLn)

Q\_ff<=SD;

else if(SLn)

Q\_ff<=D;

end

end

always @(\*) begin

if(~ALn)

Q\_latch<=~ADn;

if(LAT&&CLK&&EN) begin

if (~SLn)

Q\_latch<=SD;

else if(SLn)

Q\_latch<=D;

end

end

assign Q = (LAT)?Q\_latch:Q\_ff;

endmodule

**The testbench code:**

module SLE\_tb();

reg D,CLK,EN,ALn,ADn,SLn,SD,LAT;

wire Q\_dut;

SLE dut(D,CLK,EN,ALn,ADn,SLn,SD,LAT,Q\_dut);

initial begin

CLK=0;

forever

#1 CLK=~CLK;

end

integer i;

initial begin

SD=1;ADn=0;LAT=0;ALn=0;//note SD&ADn&LAT are static

D=0;EN=0;SLn=0;

@(negedge CLK);

ALn=1;

for(i=0;i<100;i=i+1)begin

D=$random;

EN=$random;

SLn=$random;

@(negedge CLK);

end

ALn=0;D=0;EN=0;SLn=0;// I will reset the values to change the LAT from 0to1

@(negedge CLK);

ALn=1;LAT=1;

for(i=0;i<100;i=i+1)begin

D=$random;

EN=$random;

SLn=$random;

@(negedge CLK);

end

$stop;

end

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedendmodule