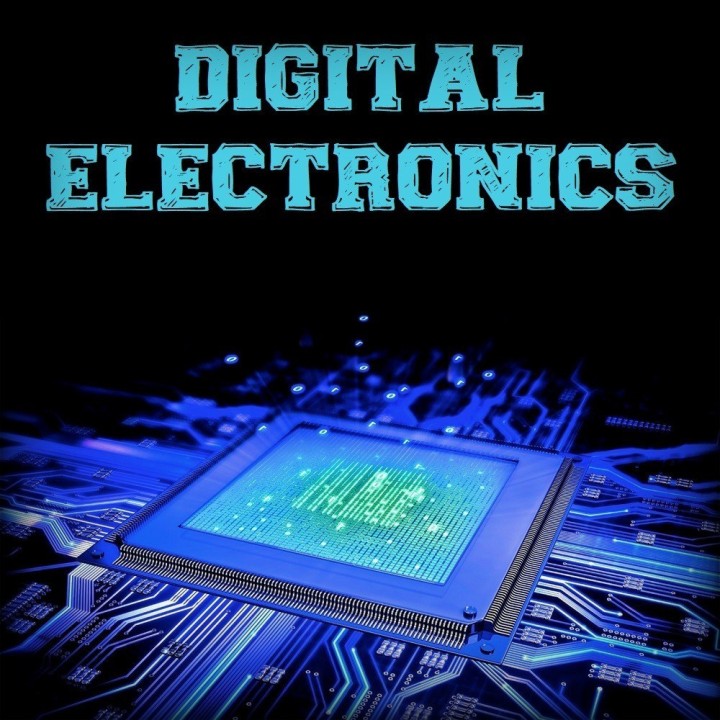
***Abanob Evram***

***Assignmen3***



A screenshot of a computer program

Description automatically generated **[Q1]**

**The design code:**

module latch(aset,data,gate,aclr,q);

parameter LAT\_WIDTH=6;

input aset,gate,aclr;

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

output reg [LAT\_WIDTH-1:0] q;

always @(\*) begin

if(aclr)

q<=0;

else if (aset)

q<={LAT\_WIDTH{1'b1}};

else if (gate)

q<=data;

end

endmodule

**The testbench code:**

module latch\_tb();

parameter LATCH\_WIDTH\_tb = 6;

reg aset\_tb,gate\_tb,aclr\_tb;

reg [LATCH\_WIDTH\_tb-1:0] data\_tb,q\_excpected;

wire [LATCH\_WIDTH\_tb-1:0] q\_dut;

latch #(LATCH\_WIDTH\_tb) dut(aset\_tb,data\_tb,gate\_tb,aclr\_tb,q\_dut);

integer i;

initial begin

aset\_tb=0;aclr\_tb=1;data\_tb=0;gate\_tb=0;q\_excpected=0;

if(q\_dut!=q\_excpected) begin

$display("errorr");

$stop;

end

#10

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

aset\_tb=$random;

aclr\_tb=$random;

gate\_tb=$random;

data\_tb=$random;

if(aclr\_tb)

q\_excpected<=0;

else if (aset\_tb)

q\_excpected<={LATCH\_WIDTH\_tb{1'b1}};

else if (gate\_tb)

q\_excpected<=data\_tb;

#10

if(q\_excpected!=q\_dut)begin

$display("Errrror");

$stop;

end

end

$stop;

end

endmodule

A black screen with multiple lines

Description automatically generated with medium confidence

**[Q2]**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

A notebook with writing on it

Description automatically generated

A white background with black text

Description automatically generated **[Q3]**

**The design code:**

module Counter(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

**The testbench code:**

module Counter\_golden\_tb();

reg clk,set;

wire [3:0] out\_behavioral,out\_structural;

Ripple\_counter golden(clk,set,out\_structural);

Counter dut(clk,set,out\_behavioral);

initial begin

clk=0;

forever

#1 clk=~clk;

end

integer i;

initial begin

set=0;

@(negedge clk);

if(out\_behavioral!=out\_structural)begin

$display("Errrrorr");

$stop;

end

set=1;

for(i=0;i<50;i=i+1)begin

@(negedge clk);

if(out\_behavioral!=out\_structural)begin

$display("Errrrorr");

$stop;

end

end

$stop;

end

endmodule

**The do file code:**

vlib work

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

vsim -voptargs=+acc Counter\_golden\_tb

add wave \*

run -all

#quit -sim

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated**[Q4]**

**The design code:**

module Extend\_counter(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];

assign div\_4 = out[1] ;

endmodule

**The testbench code:**

module Extend\_counter\_tb();

reg clk,set;

wire [3:0] out\_extend,out\_structural;

wire div\_2\_extend,div\_4\_extend,div\_2\_structural,div\_4\_structural;

Ripple\_counter golden(clk,set,out\_structural);

Extend\_counter dut(clk,set,out\_extend,div\_2\_extend,div\_4\_extend);

assign div\_2\_structural = out\_structural[0];

assign div\_4\_structural = out\_structural[1] ;

initial begin

clk=0;

forever

#1 clk=~clk;

end

integer i;

initial begin

set=0;

@(negedge clk);

if(out\_extend!=out\_structural) $display("Errrrorr in output");

if(div\_2\_extend!=div\_2\_structural) $display("Errrrorr in div\_2");

if(div\_4\_extend!=div\_4\_structural) $display("Errrrorr in div\_4");

set=1;

for (i=0;i<50;i=i+1) begin

@(negedge clk);

if(out\_extend!=out\_structural) $display("Errrrorr in output");

if(div\_2\_extend!=div\_2\_structural) $display("Errrrorr in div\_2");

if(div\_4\_extend!=div\_4\_structural) $display("Errrrorr in div\_4");

end

$stop;

end

endmodule

A screen shot of a computer game

Description automatically generated