BBM 203: Logic Design Lab

Fall 2020

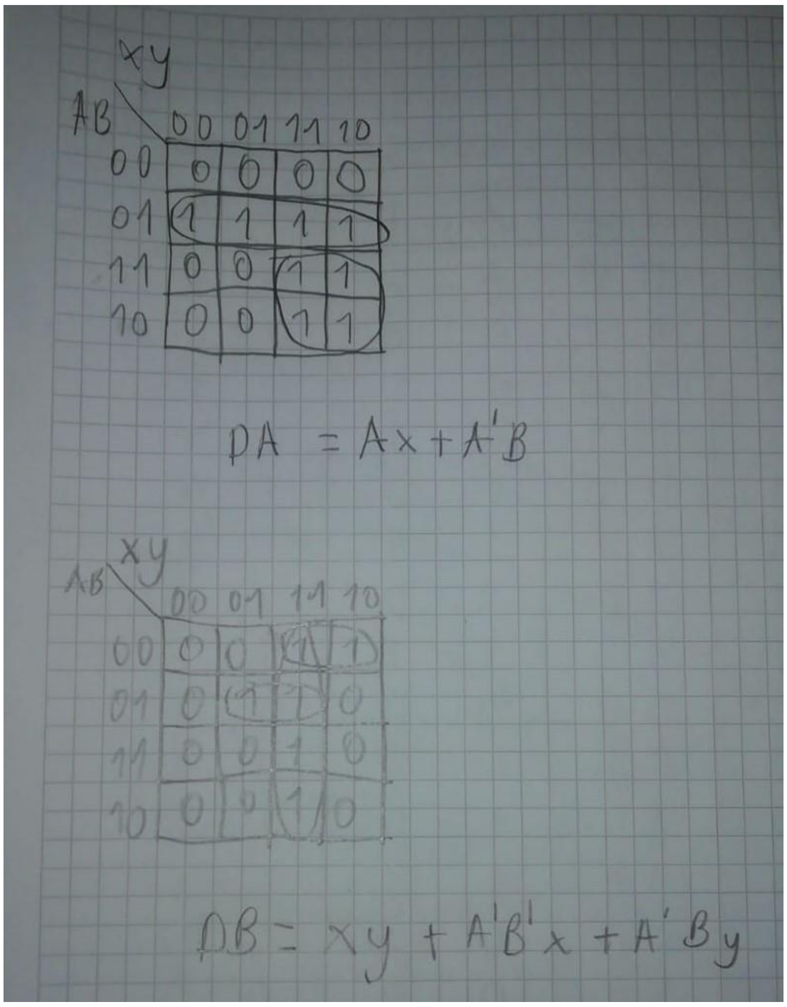
Lab Experiment 5 – Report

Emirhan Topcu 21827899

Problem

Designing and simulating sequential circuits in Verilog HDL.

Convert the given state transition diagram to state transition table.



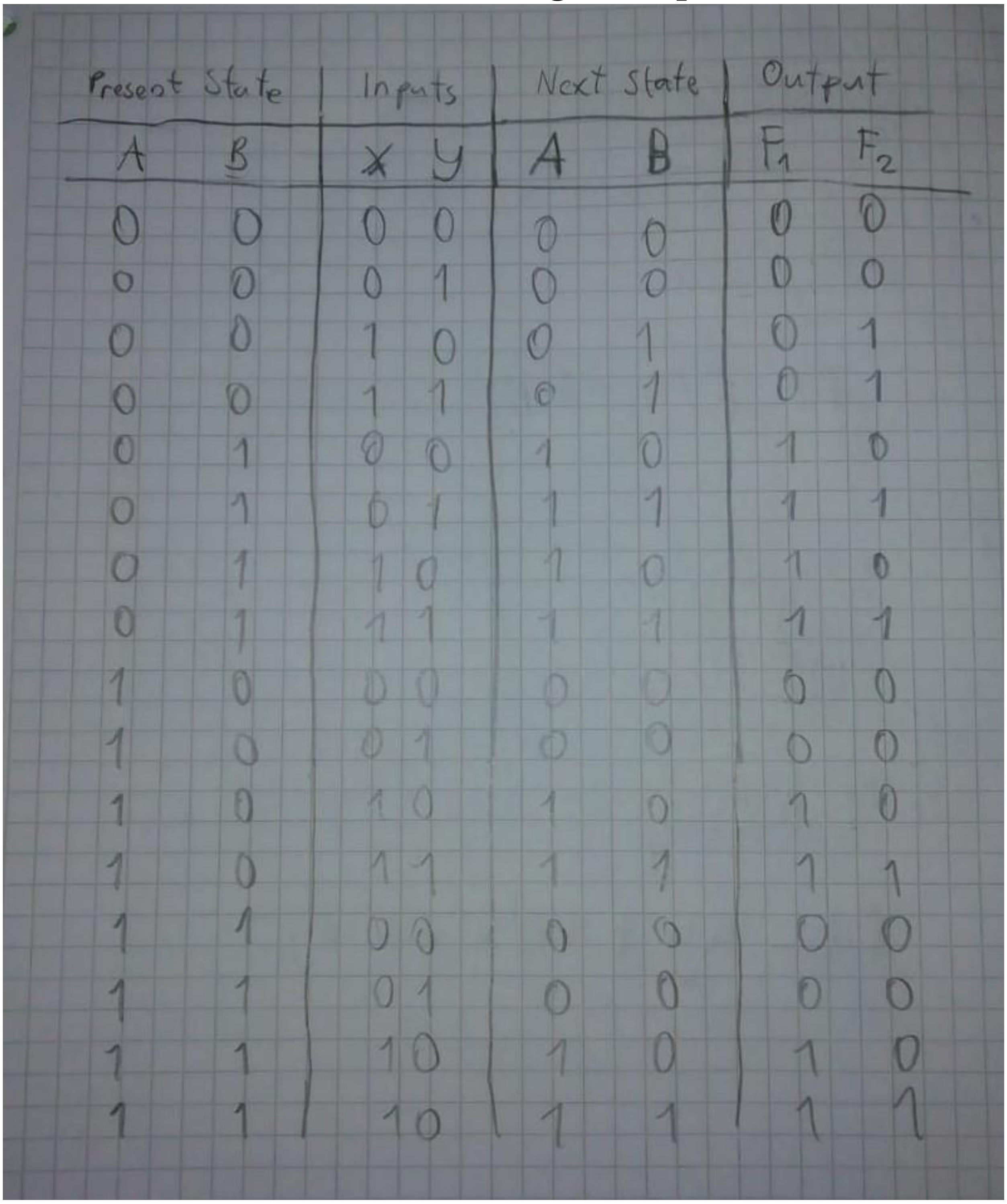
AB \ xy	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	1	1

$DA = Ax + A'B$

AB \ xy	00	01	11	10
00	0	0	1	1
01	0	1	1	0
11	0	0	1	0
10	0	0	1	0

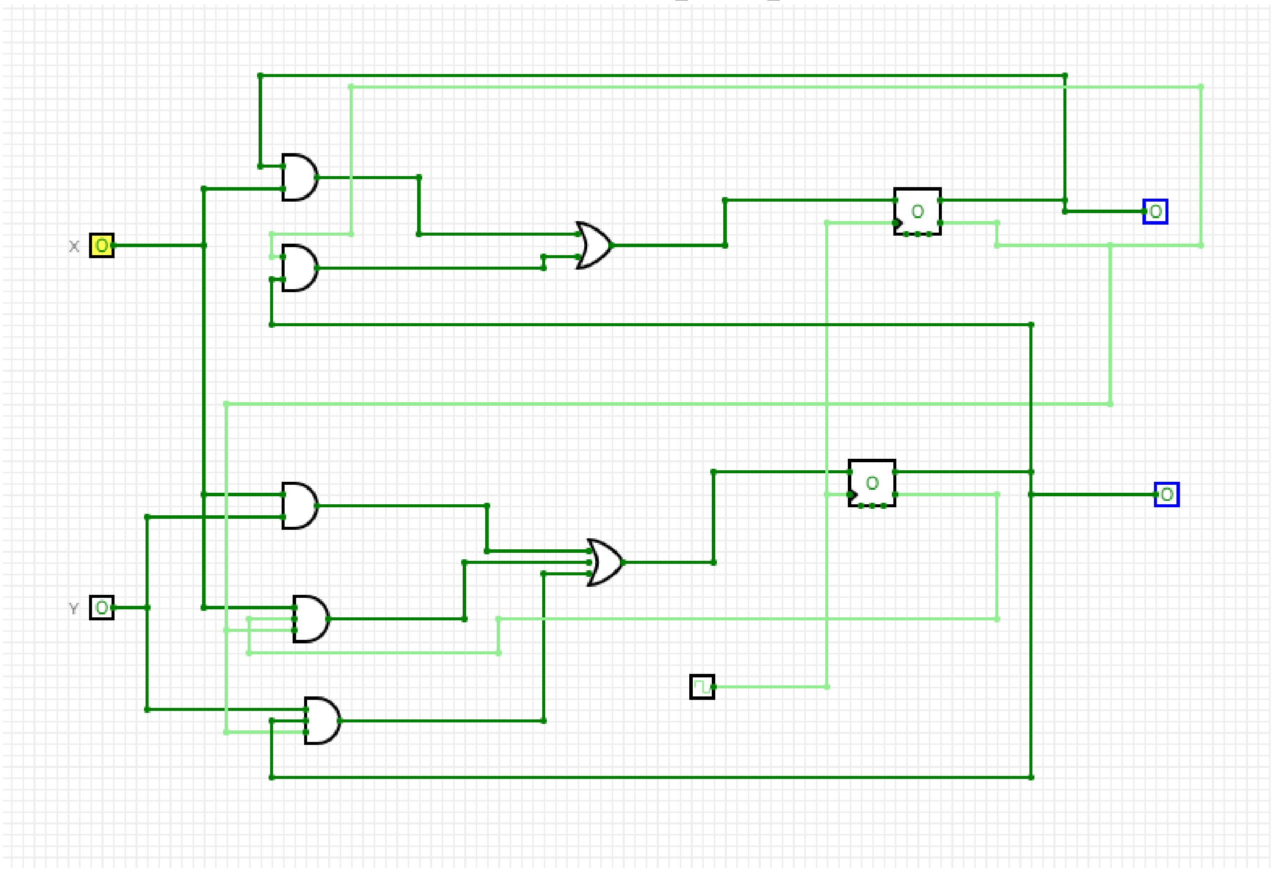
$DB = xy + A'B'x + A'By$

Determine the number of D flip flops you need to use to store the state information, and derive the input and output equations from the state transition table. Minimize the functions using K-Maps.



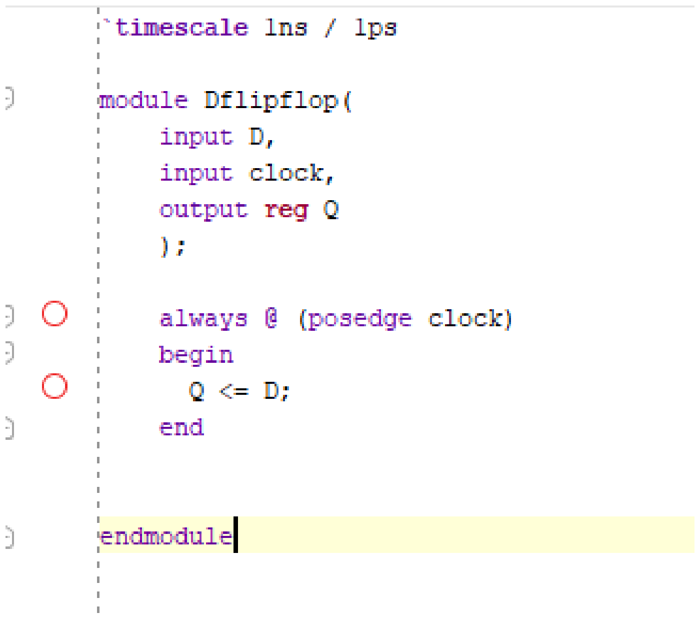
Present State		Inputs		Next State		Output	
A	B	x	y	A	B	F_1	F_2
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	0	1
0	1	0	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	1	0	1	0
1	0	1	1	1	1	1	1
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	1	0	1	0
1	1	1	0	1	1	1	1

Use the simplified functions to obtain the design schematic that uses D flip flops.



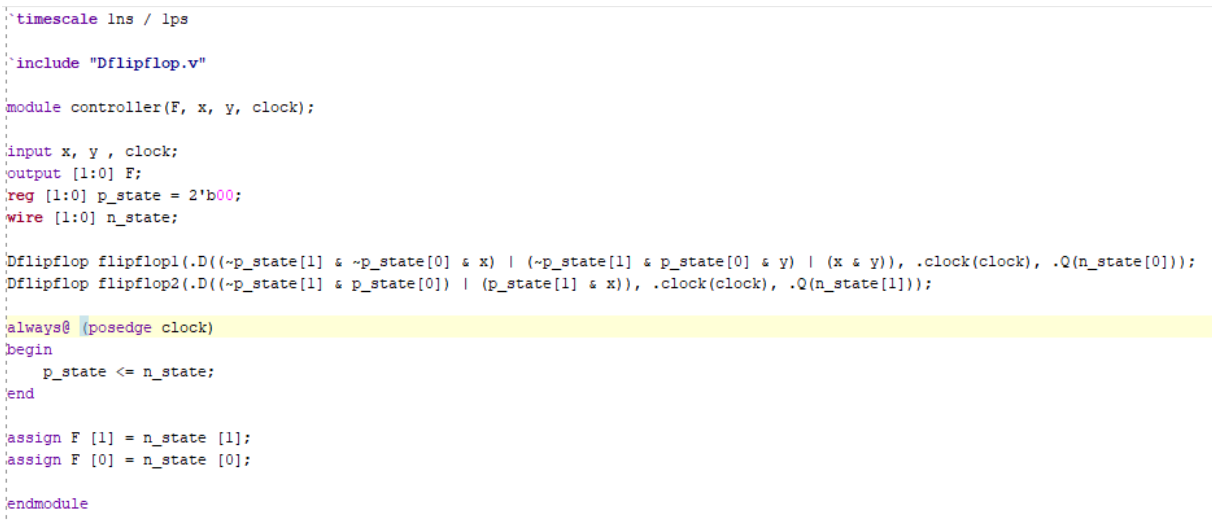
Include three Verilog code solutions for all specified modules:

Dflipflop.v



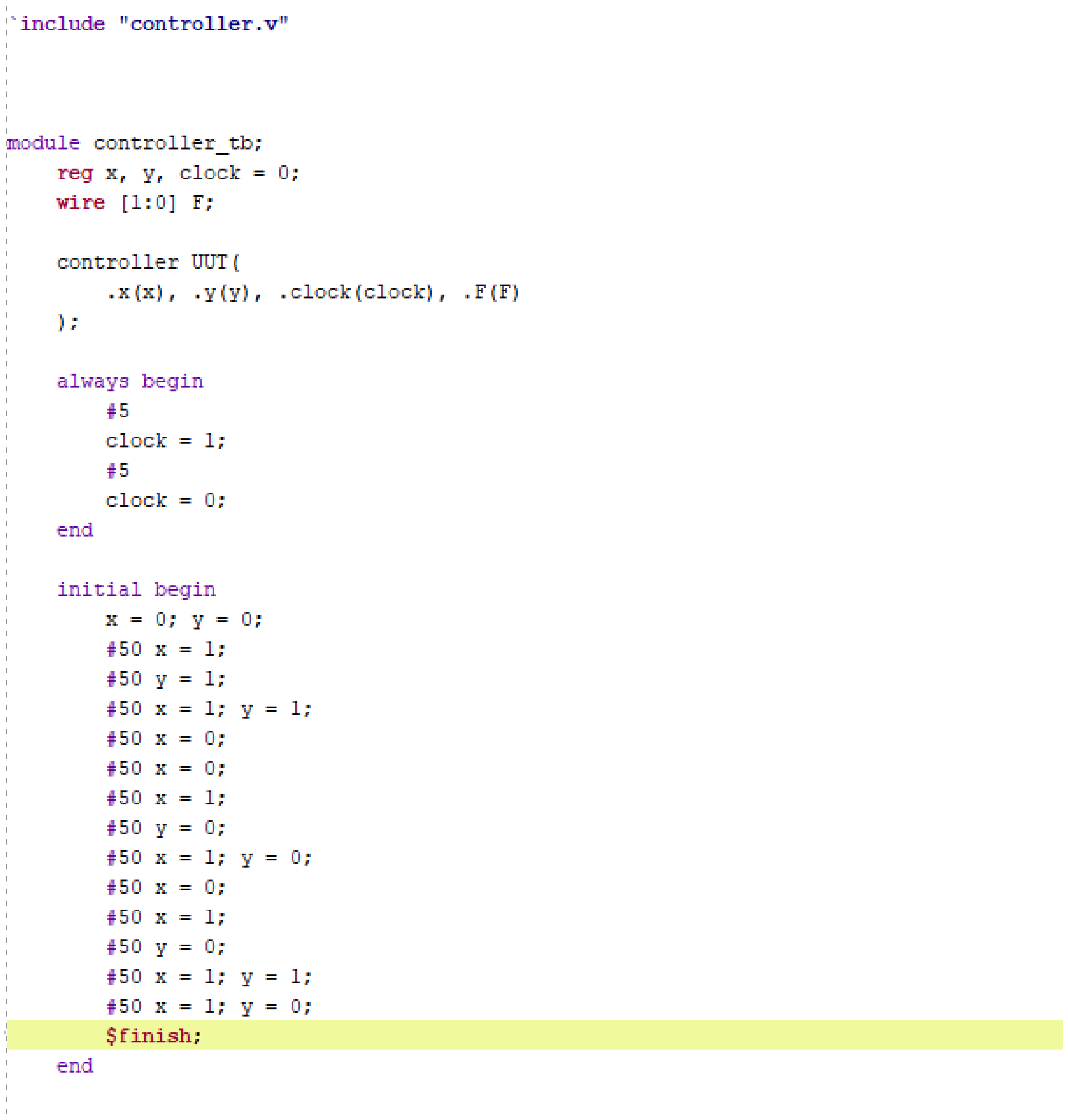
```
`timescale 1ns / 1ps

module Dflipflop(
    input D,
    input clock,
    output reg Q
    );

    always @ (posedge clock)
    begin
      Q <= D;
    end


endmodule
```

controller.v



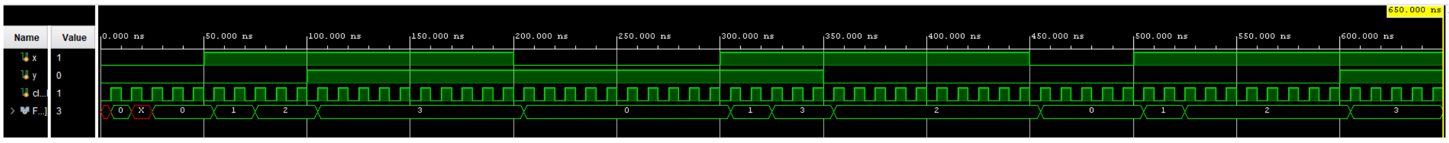
```verilog
`timescale 1ns / 1ps

`include "Dflipflop.v"

module controller(F, x, y, clock);

input x, y , clock;
output [1:0] F;
reg [1:0] p_state = 2'b00;
wire [1:0] n_state;

Dflipflop flipflop1(.D((~p_state[1] & ~p_state[0] & x) | (~p_state[1] & p_state[0] & y) | (x & y)), .clock(clock), .Q(n_state[0]));
Dflipflop flipflop2(.D((~p_state[1] & p_state[0]) | (p_state[1] & x)), .clock(clock), .Q(n_state[1]));

always@ (posedge clock)
begin
    p_state <= n_state;
end

assign F [1] = n_state [1];
assign F [0] = n_state [0];

endmodule
```

controller_tb.v



```verilog
`include "controller.v"



module controller_tb;
    reg x, y, clock = 0;
    wire [1:0] F;

    controller UUT(
        .x(x), .y(y), .clock(clock), .F(F)
    );

    always begin
        #5
        clock = 1;
        #5
        clock = 0;
    end

    initial begin
        x = 0; y = 0;
        #50 x = 1;
        #50 y = 1;
        #50 x = 1; y = 1;
        #50 x = 0;
        #50 x = 0;
        #50 x = 1;
        #50 y = 0;
        #50 x = 1; y = 0;
        #50 x = 0;
        #50 x = 1;
        #50 y = 0;
        #50 x = 1; y = 1;
        #50 x = 1; y = 0;
        $finish;
    end
```

Include the obtained waveform and explain.



I’ve checked for all the inputs and it works as it supposed to. For all the edges in the state transition diagram, I included a test block in the test bench and every execution for every edge works perfectly.