EXPERIMENT 5 REPORT

BBM233
LOGIC DESIGN LABORATORY

Name and Surname : Meltem Kaya
Student ID : 21827555
Due Date : 26/12/2020

HACETTEPE
University

Department of Computer Engineering

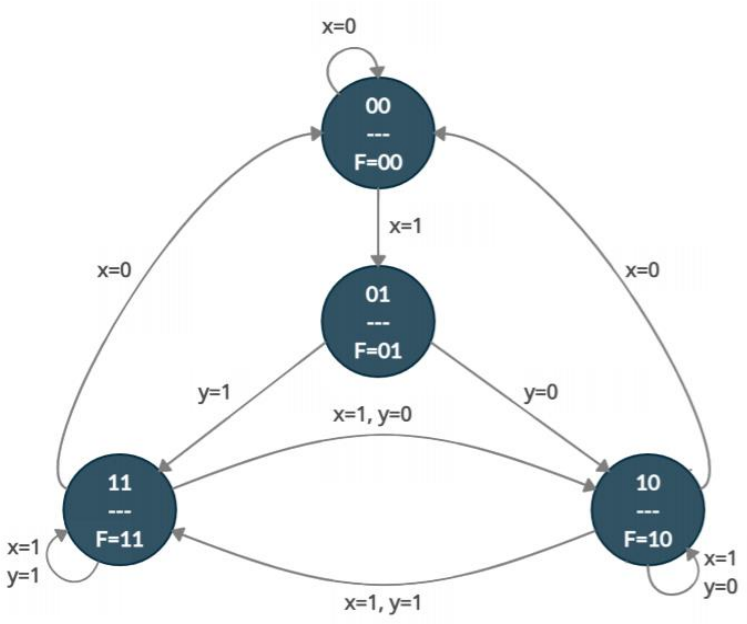
—

Teaching Assistant
Selma Dilek

Table of Contents

SOFTWARE DESIGN NOTES

Problem Statement



The state diagram of a control unit is shown in figure above. It has four states, two inputs x and y, and one 2-bit output F. Use D flip flops as storage elements to store the state information. Implement the design in Verilog using structural design approach.
Main purpose of this experiment is designing and simulating a sequential circuit by using D flip flops.

State Transition Table

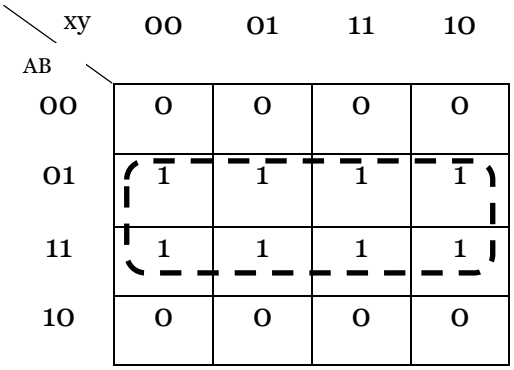
Present State		Next State		Flip Flop Inputs		Output F	
A	B	A^+	B^+	x	y	F_2	F_1
0	0	0	0	0	**X**	0	0
0	0	0	1	1	**X**	0	0
0	1	1	0	**X**	0	0	1
0	1	1	1	**X**	1	0	1
1	0	0	0	0	**X**	1	0
1	0	1	0	1	0	1	0
1	0	1	1	1	1	1	0
1	1	0	0	0	**X**	1	1
1	1	1	0	1	0	1	1
1	1	1	1	1	1	1	1

Note: ***X*** *-> Don't cares*
$\boldsymbol{B^+}$ *-> Next state B*
$\boldsymbol{A^+}$ *-> Next state A*

Number of D-Flip Flops Needed to Store the State Information

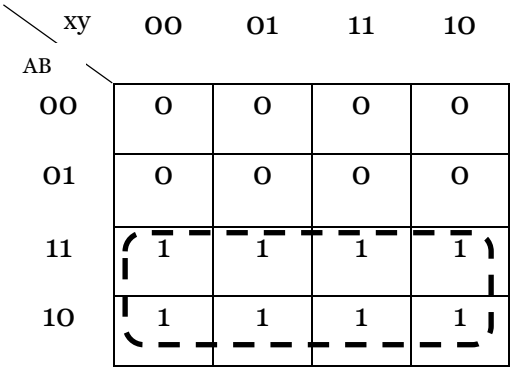
We need two D-Flip Flops to store state information because the number of states is four.

Evaluating Output Functions by Using Karnaugh Map



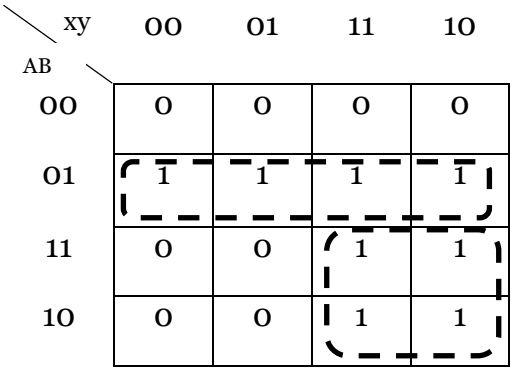
Simplifying F_1 Function

F_1 = B



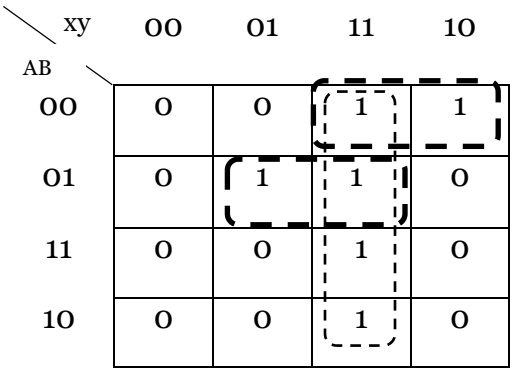
Simplifying F_2 Function

F_2 = A



Simplifying A^+ Function

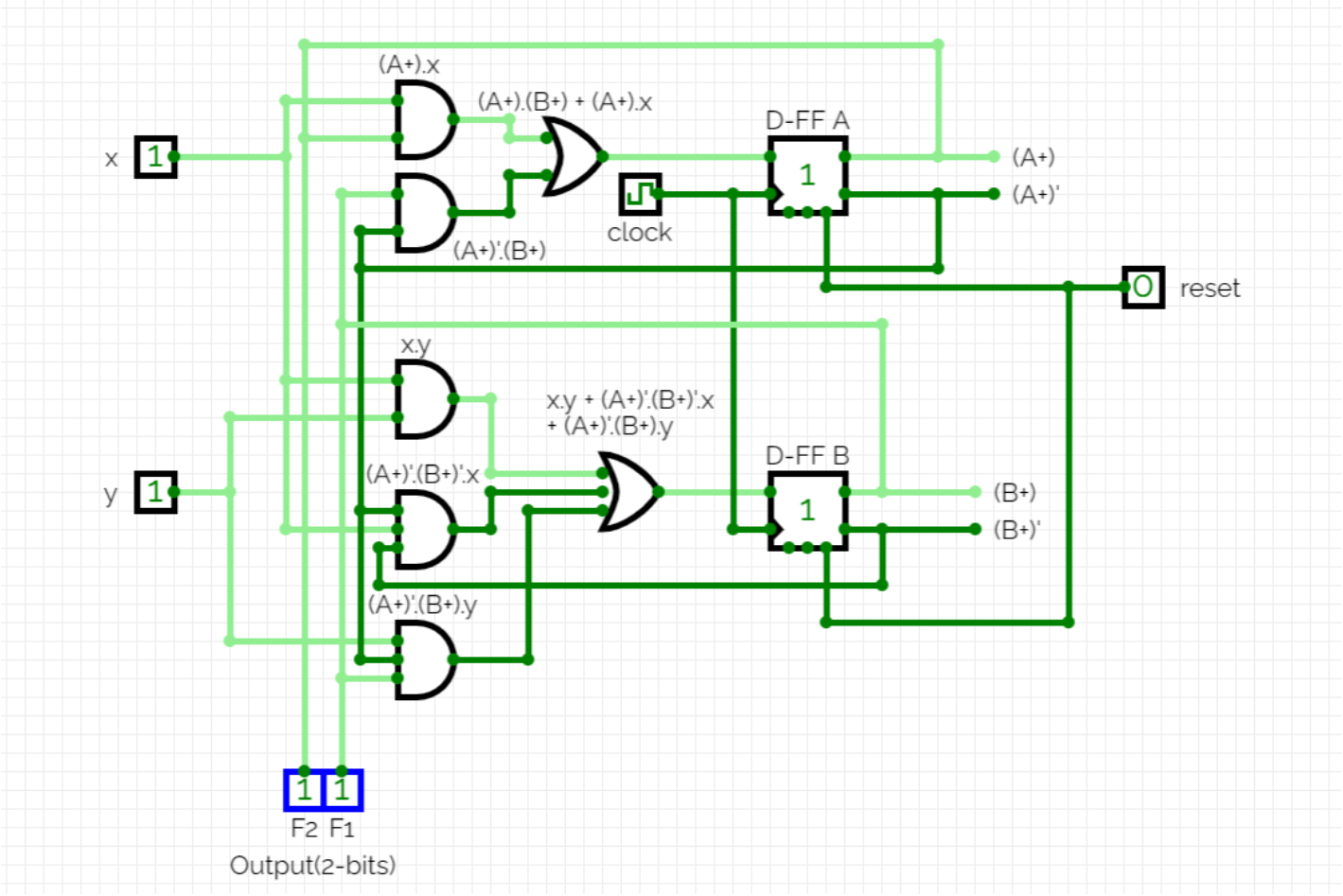
A^+ = A'B + Ax



Simplifying B^+ Function

B^+ = A'B'x + A'By + xy

Design Shematic



Dflipflop.v

```
1  `timescale 1ns / 1ps
2
3  module Dflipflop(
4  output reg Q,
5  input D,
6  input clk,rst // clk -> clock, rst -> reset
7      );
8      always @(posedge clk,posedge rst) begin
9
10         if (rst)
11             Q <= 1'b0;  //if reset is 1, assign output to 0
12
13         else
14            Q <= D; //non-blocking assign statement, assign output to D
15     end
16 endmodule
```

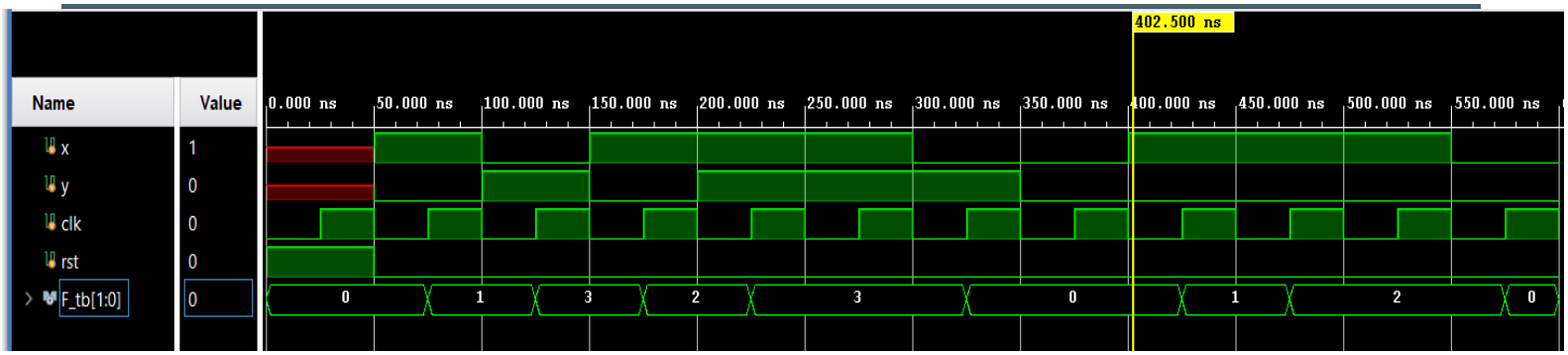
controller.v

```
`timescale 1ns / 1ps
`include "Dflipflop.v"

module controller(
output [1:0]F,  // 2-bits output
input x, y, clk,reset    // clk -> clock
    );

    reg [1:0] present_state = 2'b0;  // initialization of present state with 2'b00 value
    wire [1:0] next_state;          // declaration of next_state

    Dflipflop D2(.Q(next_state[1]),.D(~present_state[1] & present_state[0] | present_state[1] & x),.clk(clk),.rst(reset));
    Dflipflop D1(.Q(next_state[0]),.D(~present_state[1] & ~present_state[0] & x | ~present_state[1] & present_state[0] & y | x & y),.clk(clk),.rst(reset));

    always @ (next_state) begin

        present_state <= next_state;  // updating present state
    end
    assign F = present_state;   //assign output


endmodule
```

controller_tb.v

```
`timescale 1ns / 1ps
`include "controller.v"

module controller_tb;

     reg x,y,clk,rst;   // x,y -> inputs , clk -> clock, rst -> reset
     wire [1:0]F_tb;     // 2-bits output

    controller UUT(.F(F_tb),.x(x),.y(y),.clk(clk),.reset(rst));    //unit under test

     initial begin
          rst = 1'b1;      //initialization reset
          clk = 1'b0;     //initialization clock
          #50 rst= 1'b0;    //set reset to 0
          #550 $finish;
     end

          always #25 clk = ~clk;    //clock generator. period of clock is 50

     always begin
          #50 {x,y} = 2'b10;      //test inputs.
          #50 {x,y} = 2'b01;
          #50 {x,y} = 2'b10;
          #50 {x,y} = 2'b11;
          #50 {x,y} = 2'b11;
          #50 {x,y} = 2'b01;
          #50 {x,y} = 2'b00;
          #50 {x,y} = 2'b10;
          #50 {x,y} = 2'b10;
          #50 {x,y} = 2'b10;
          #50 {x,y} = 2'b00;
          end
endmodule
```

Obtained Waveform



Explanations

We can see that it is a Moore Machine from state diagram, means that output depends on only current state. If we look obtained waveform, we can see that all possible transitions are used to confirm whether circuit design is working correctly.
Reset equals to 1 between 0-50.000ns. In that range, inputs are not initialized. From 50.000 to 600.000ns reset equals to zero and we can see all possible transitions by the same line with positive edges of clock and outputs equal to present state.

Resources

TOBB ETU University Logic Design Lecture Records
BBM233 Lecture records and documents
https://www.youtube.com/watch?v=X75NM8eNzro
https://mil.ufl.edu/3701/classes/joel/16%20Lecture.pdf
https://en.wikipedia.org/wiki/State-transition_table
https://www.sciencedirect.com/topics/computer-science/moore-machine