Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2020 Fall

Experiment 5 - Sequential Circuits in Verilog

December 26, 2020

Student name
İlayda Atmaca

Student Number:
b21827101

1 Include the problem statement.

In this experiment,we aim to designing and simulating sequential circuits in Verilog HDL. A flip flop is a basic building block of sequential logic circuits .We have 2 D Flip Flop in the experiment and each of D Flip Flops has two input(x,y), a clock input ,reset input and outputs Q and Q' (the inverse of Q). Additionally, we use the sequential circuit so outputs depend on both the present inputs and the sequence of past inputs.

2 Convert the given state transition diagram to state transition table.

present state		input		next state		output	
A	B	x	y	A	B	F1	F2
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	1	0	1	0	1	0
0	1	0	1	1	1	1	1
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	1	1	1	1	1

Figure 1: State transition table

3 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

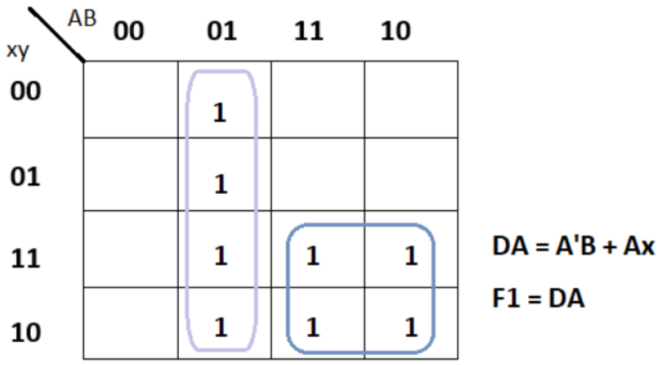

Figure 2: K-Map

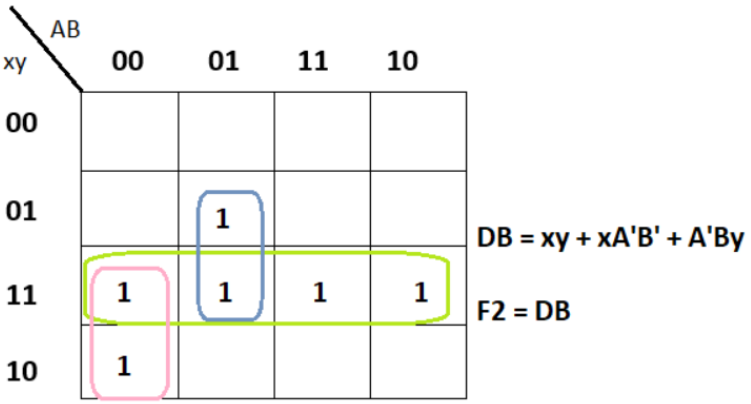

Figure 3: K-Map

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

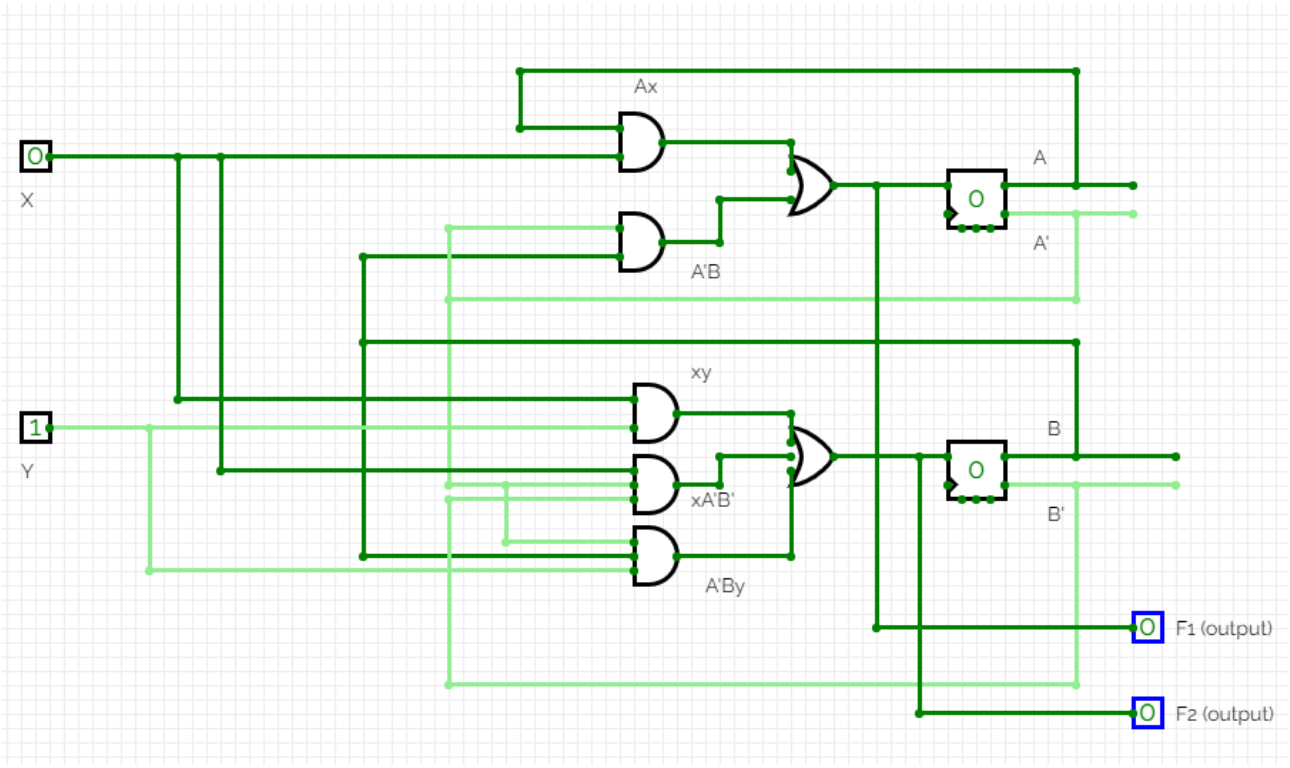

Figure 4:

5 Include three Verilog code solutions for all specified modules: Dflipflop.v, controller.v, and controller$_tb.v$

```

`timescale 1ns / 1ps

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

    always @(posedge clock) begin
    if(reset)
    Q<=0;
    else
    Q<=D;
    end
endmodule
```

```


`timescale 1ns / 1ps

module controller(input x,
input y,input clock,
input reset,
output [1:0] out
);

reg [1:0] present_state = 2'b00;
wire [1:0] next_state;
wire w1,w2,w3,w4,w5;


and A1(w1,~present_state[1],present_state[0]);
and A2(w2,present_state[1],x);
and A3(w3,x,~present_state[0],~present_state[1]);
and A5(w5,y,x);
and A4(w4,y,~present_state[1],present_state[0]);
or o1(out[1],w1,w2);
or o2(out[0],w3,w4,w5);

//Instantiate two D flip flops
Dflipflop df(.D((present_state[0]&~present_state[1])|(present_state[1]&x)),
.clock(clock),.reset(reset),
.Q(next_state[1]));

Dflipflop df1(.D((~present_state[0]&~present_state[1]&x)
```

```
|(present_state[0]&~present_state[1]&y)|(y&x)),
.clock(clock),.reset(reset),
.Q(next_state[0]));

always @(reset or next_state)begin
    if(reset) begin present_state <= 2'b00; end
    else begin present_state <= next_state; end
    end

endmodule
```

```


`timescale 1ns / 1ps
module controller_tb;

reg x;   //input x,y,clock,reset
reg y;
reg clock;
reg reset;
wire [1:0] out;  //output out

controller c(.x(x),.y(y),.clock(clock),.reset(reset),.out(out));
initial begin
  clock  = 0;
  reset   = 1;
  #100;
  reset = 0;
  #100;
 end
//Generate a clock with period
   always #10 clock = ~clock;
initial begin  //Test sequence
    x=0; y=0;
    #100 x=0; y=0;
    #100 x=1; y=0;
    #100 x=0; y=0;
    #100 x=0; y=1;
    #100 x=1; y=1;
    #100 x=1; y=0;
    #100 x=1; y=1;
    #100 x=0; y=1;
    end
endmodule
```

6 Include the obtained waveform.

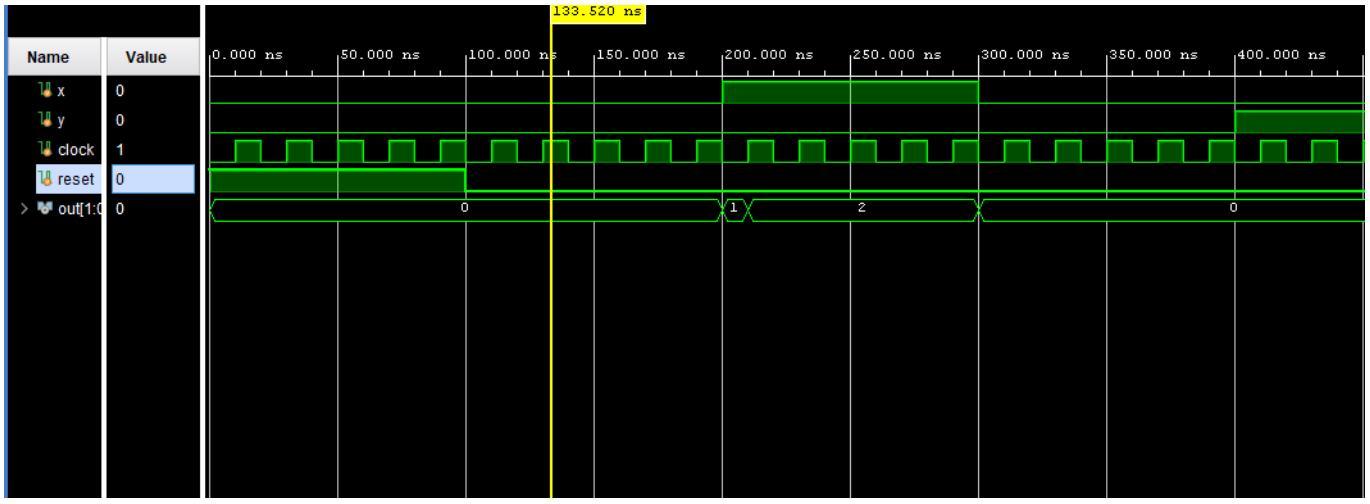

Figure 5: Waveform

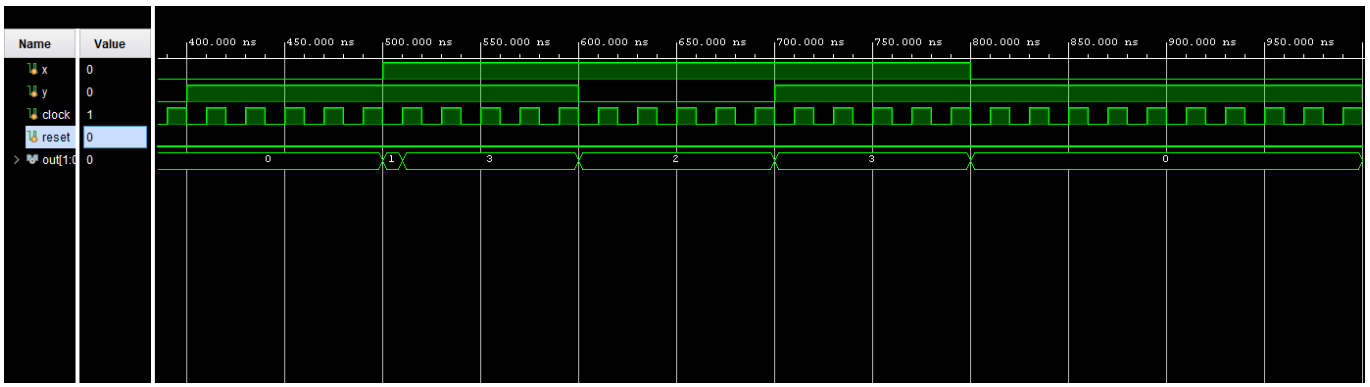

Figure 6: Waveform

7 Explain the obtained results and how they show that your design is working correctly.

First, I created my truth table and K-map from my state diagram. and I obtained the boolean equations from the K-map. I also got information on how output depends on the inputs. The outputs can change when clock edge will be either rising edge or falling edge. and when I compared my truth table and vaweform, I saw that I got the correct values.

References

- Reference 1: https://en.wikipedia.org/wiki/Flip-flop_(electronics)#
- Reference 2: https://www.electronics-tutorials.ws/sequential/seq_4.html)#