UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Homework 5 Solutions

1 From Mano: Problem 5-17.

a) Design the circuit described in the exercise.

The output is 0 for all 0 inputs until the first 1 occurs at which time the output is 1. Thereafter, the output is the complement of the input.

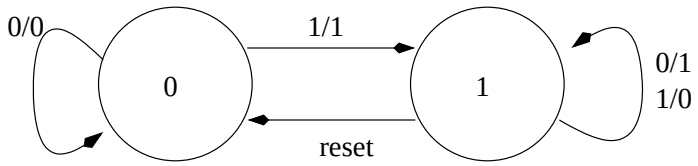


Figure 1: State diagram

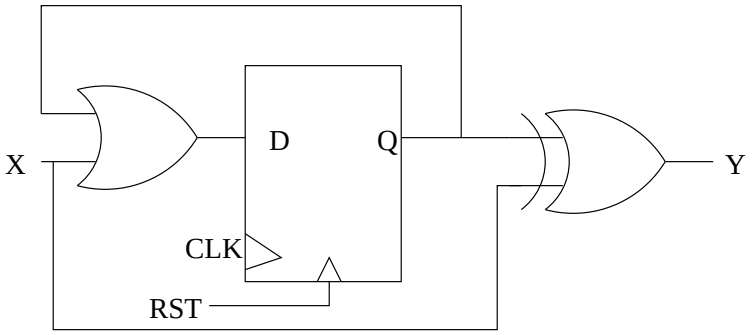


Figure 2: Circuit

b) Write the behavioral Verilog description of the circuit based on your state transition diagram (similar to the lab exercise).

```
module TwosComplementer (in, clk, rst, out);
  input in,clk,rst;
  output out;
  reg curState,nextState,out;

  always @(posedge clk or posedge rst)
    if(rst) curState = 1'b0;
    else curState = nextState;

  always @(curState or in)
    case(curState)
```

```
      1'b0:
        if (in) begin
          nextState = 1'b1;
          out = 1'b1;
        end else begin
          nextState = 1'b0;
          out = 1'b0;
        end
      1'b1: begin
          nextState = 1'b1;
          out = ~in;
        end
     endcase
endmodule
```

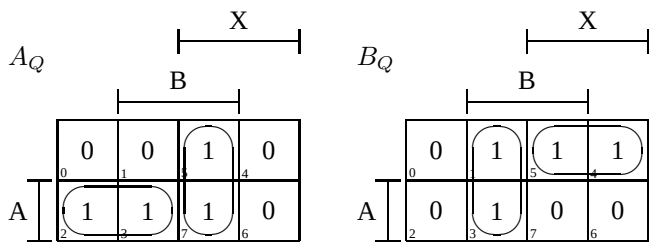
c) Write the structural Verilog description of the detailed circuit you derived in part a) using low-level gates for the combinational logic and cimplied flip-flip(s) for the state register. Write three modules– the combinational logic, the state register, and the composition of the two as the FSM.

```
module StateRegister(D,clk,rst,Q);
  input D,clk,rst;
  output Q;
  reg Q;

  always @(posedge clk or posedge rst)
    if (rst) Q = 0;
    else Q = D;

endmodule

module CombinationalLogic(curState,in,nextState,out);
  input curState, in;
  output nextState, out;

  assign nextState = curState | in;
  assign out = curState ^ in;
endmodule

module FSM(in, clk, rst, out);
  input in,clk,rst;
  output out;
  wire curState, nextState;

  StateRegister sr(.D(nextState),.clk(clk),.rst(rst),.Q(curState));
  CombinationalLogic cl(.curState(curState),.in(in),
                        .nextState(nextState),.out(out));
endmodule
```

5-16

Truth table:

X	A_D	B_D	A_Q	B_Q
0	0	0	0	0
0	0	1	0	1
0	1	1	1	1
0	1	0	1	0
1	0	0	0	1
1	0	1	1	1
1	1	1	1	0
1	1	0	0	0



$A_Q = Ax' + Bx$
$B_Q = A'x + Bx'$

5-21

The initial statement executes only once. The always statement executes every time that the specified even occurs. Initial statements are also for simulation only.

5-22

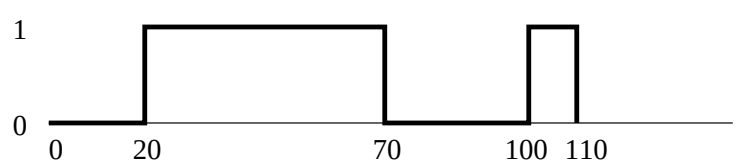


Figure 3: Waveform

5-24

```
module DFF(D,clk,clr,pre,Q);
  input D,clk,clr,pre;
  output Q;
  reg Q;
```

```
  always @(posedge clk or negedge clr or negedge pre)
    if (~pre) Q = 1'b1;
    else if (~clr) Q =1'b0;
    else Q = D;
endmodule
```

5-25

```
module DFF(D1,D2,ctrl,clk,rst,Q);
  input D1,D2,clk,rst,ctrl;
  output Q;
  reg Q;

  always @(posedge clk or posedge rst)
    if (rst)  Q = 0;
    else if (ctrl) Q = D2;
    else Q = D1;
endmodule
```

5-28

a)

```
module fsm(x,y,clk,A);
input x,y,clk;
output A;
reg A,nextState;

always @(posedge clk)
  A = nextState;

always @(x or y or A)
begin
  if (x != y) nextState = ~A;
  else nextState = A;
end

endmodule
```

b)

```
module circuit(x,y,clk,A);
input x,y,clk;
output A;
reg A;

always @(posedge clk)
```

```
  A <= (x ^ y) ^ A;

endmodule
```

c)

```
module testbench;
reg x,y,clk;
wire c_A,f_A;

circuit c(.x(x),.y(y),.clk(clk),.A(c_A));
fsm f(.x(x),.y(y),.clk(clk),.A(f_A));

always @(clk)
  #5 clk = ~clk;

initial
begin
  $monitor("x=%b y=%b clk=%b Circuit_A=%b FSM_A=%b",x,y,clk,c_A,f_A);
  clk = 0;
  x = 0; y = 0;
  #10 x = 0, y = 1;
  #10 x = 1, y = 1;
  #10 x = 1, y = 0;
end

endmodule
```

5-30

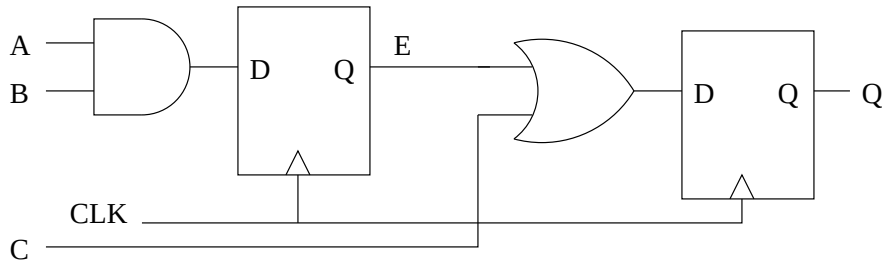


Figure 4: Non-blocking assign

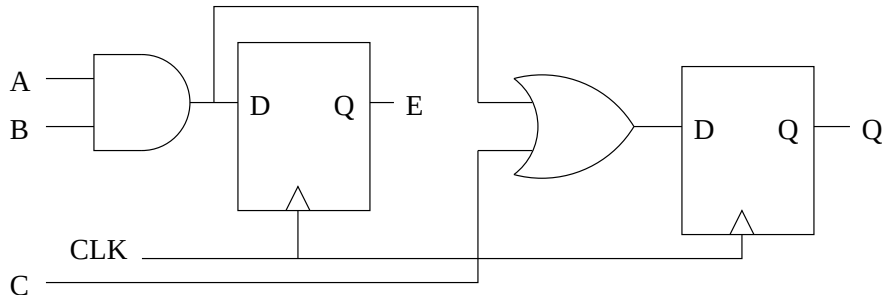


Figure 5: Blocking assign