Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.111 – Introductory Digital Systems Laboratory

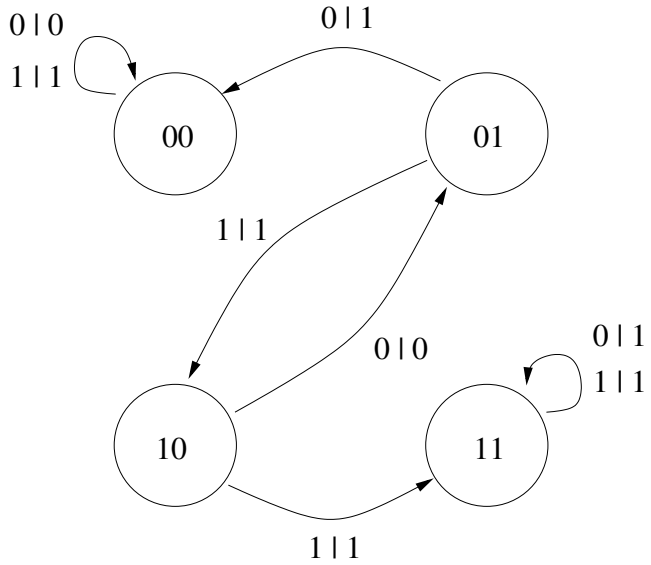
Problem Set 3 Solutions Issued: September 30, 2002

Problem 1

(a)

Q_A	Q_B	X	D_A	D_B	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

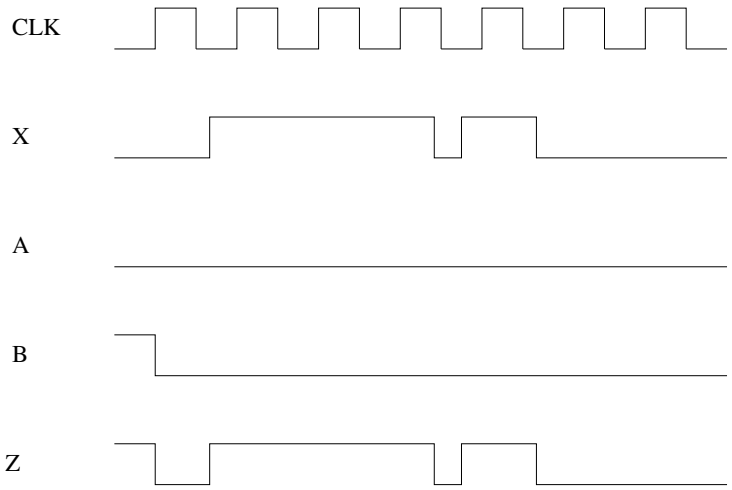


(c)

```
-- This is a sample solution to Problem set 3

library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;
entity cd_fsm is port(
clk      : in std_logic;
        reset    : in std_logic;
        X        : in std_logic;
        Z        : out std_logic;
        stateout : out std_logic_vector(1 downto 0));
end cd_fsm;

architecture state_machine of cd_fsm is
  type StateType is (ZeroZero, ZeroOne, OneZero, OneOne);
signal present_state, next_state : StateType;

begin --state_machine
        with present_state select
          stateout <=
          "00" when ZeroZero,
          "11" when OneOne,
          "01" when ZeroOne,
          "10" when OneZero,
          "10" when others;


        -- combinational process, Z changes even when state does not change
state_comb:process(present_state, X)
begin
  case present_state is
            when ZeroZero =>
              next_state <= ZeroZero;
              Z <= X;
            when ZeroOne =>
              if X = '1' then
                next_state <= OneZero;
              else
                next_state <= ZeroZero;
              end if;
              Z <= '1';
            when OneZero =>
              if X = '1' then
                next_state <= OneOne;
              else
                next_state <= ZeroOne;
              end if;
              Z <= X;
            when OneOne =>
                next_state <= OneOne;
              Z <= '1';
```

```
    when others =>
      next_state <= OneZero;
            Z <= '0';
  end case;                    --present_state;
end process;                   --state_comb;

state_clocked:process(clk, reset) begin  -- state transitions
       if (reset = '1') then
                present_state <= ZeroOne;
            elsif rising_edge(clk) then
      present_state <= next_state;
       end if;
end process state_clocked;

end state_machine;
```

(d)



Problem 2

(a)

Q_C	Q_B	Q_A	U/D	D_C	D_B	D_A
0	0	0	0	1	1	1
0	0	0	1	0	0	1
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	1
0	1	1	0	0	1	0
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	1
1	1	0	1	1	1	1
1	1	1	0	1	1	0
1	1	1	1	0	0	0

On reset, D_C, D_B, D_A should be set to 0.

(b)

Dc

QaU \ QcQb	00	01	11	10
00	1	0	1	0
01	0	0	1	1
11	0	1	0	1
10	0	0	1	1

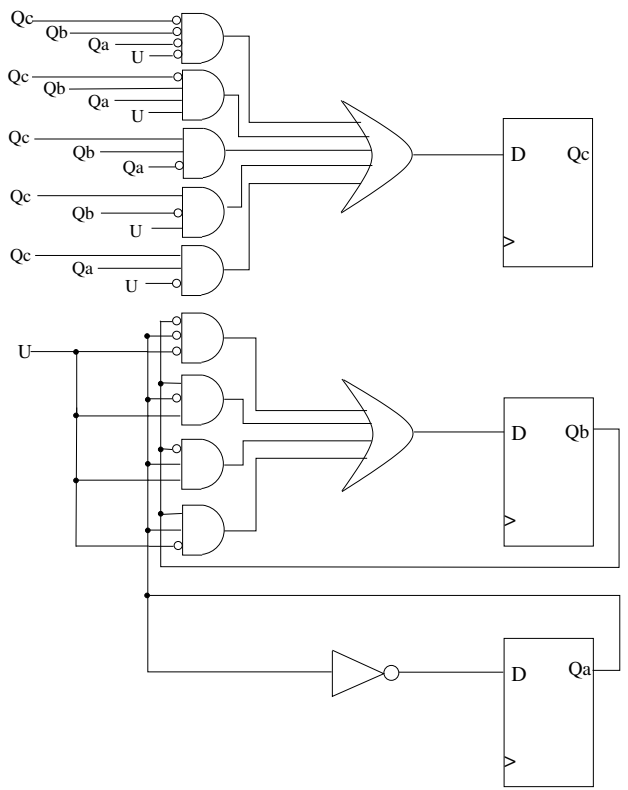
Db

QaU \ QcQb	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	1	0	0	1
10	0	1	1	0

Da

QaU \ QcQb	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

(c)



Problem 3

(a)

Inputs should be connected as follows:
A = 1
B = C = D = 0
/LD = Q_A nand Q_D
/CLR = 1

(b)

Inputs should be connected as follows:
A = 0
B = not Q_B
C = Q_C or (Q_B and not Q_D)
D = (not Q_D and Q_C and Q_B) or
(Q_D and not Q_B) or
(Q_D and not Q_C)
/LD = 0
/CLR = 1

(c)

Inputs should be connected as follows:
A = Q_B and not Q_A
B = (not Q_D and not Q_C) or (Q_D and Q_C)
C = (not Q_D and Q_B and Q_A) or (Q_D and Q_B and not Q_A)
D = Q_C
/LD = 0
/CLR = 1