

6.111 Digital Systems Laboratory
 Spring 2002
 Problem Set #3 Solutions

Problem 3.1

A. Use '163 to count 1-6 repeatedly

$$/\text{CLR} = 1$$

Load 1 when output 6.

$$/\text{LOAD} = \overline{(Q_2 * Q_1 * \overline{Q_0})}$$

$$A=1, B=C=D=0$$

B. Use '163 to count 5,6,7,12,13,14,15 repeatedly

Load 12 when output 7.

Load 5 when output 15.

$$/\text{CLR}=1$$

$$/\text{LOAD}=\overline{(Q_2*Q_1*Q_0)}$$

$$A=C=Q_3, B=D=\overline{Q_3}$$

C. Use '163 to count 0,1,2,4,7,11,13,14 repeatedly

Version 1: Ease the pain using **reduce**...

$$/\text{LOAD} = Q_2*Q_1*\overline{Q_0} + Q_3*\overline{Q_0} + \overline{Q_1}*Q_0 + \overline{Q_2}*Q_1 + Q_3*Q_2 + \overline{Q_3}*Q_2*Q_0$$

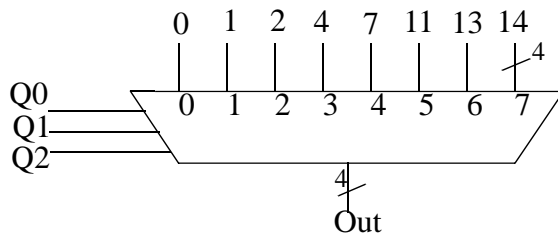
$$/\text{CLR} = \overline{(Q_3*Q_2*Q_1)}$$

$$A = Q_2 + Q_3, B = Q_2, C = \overline{Q_0} + \overline{Q_2}, D = Q_0$$

Version 2: Use a multiplexor

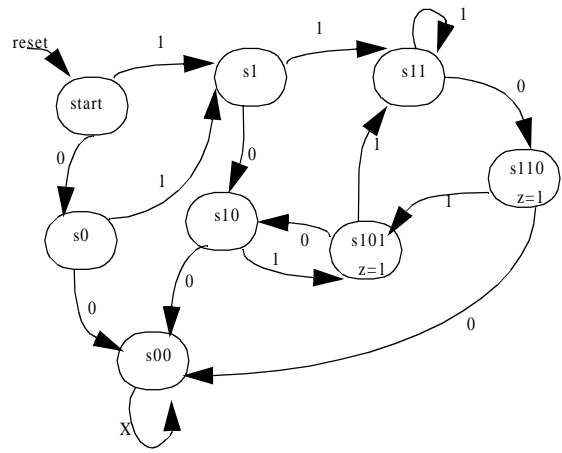
$$/\text{CLR} = \overline{(Q_2*Q_1*Q_0)}$$

$$/\text{LOAD} = 1$$



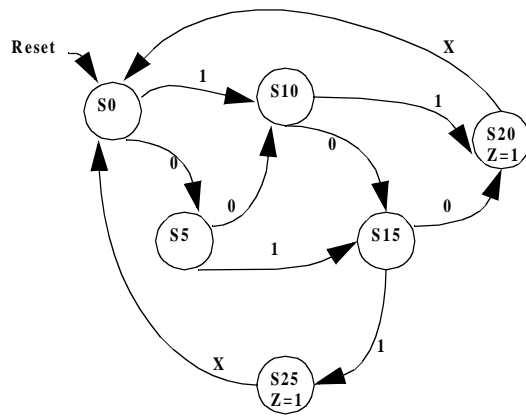
Problem 3.2

Since the Moore machine output string is given, we will design a Moore machine.



Problem 3.3

A.



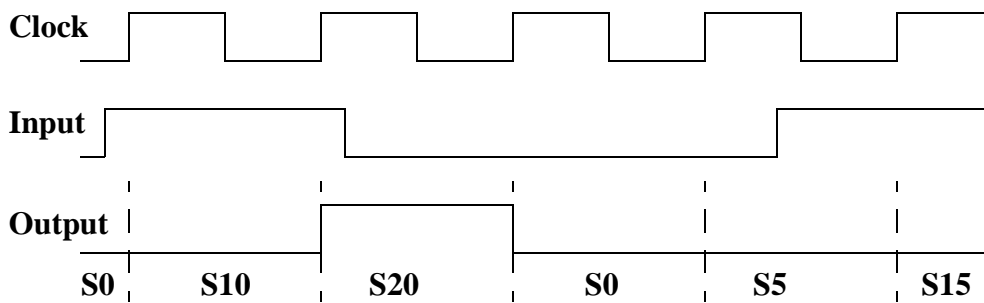
B. For 6 states, 3 state variables are needed.

C.

Table 1: Truth Table

State (n)	D2 D1 D0 (n)	X	State (n+1)	D2 D1 D0 (n+1)	Z
S0	000	0	S5	001	0
S0	000	1	S10	010	0
S5	001	0	S10	010	0
S5	001	1	S15	011	0
S10	010	0	S15	011	0
S10	010	1	S20	100	0
S15	011	0	S20	100	0
S15	011	1	S25	101	0
S20	100	0	S25	101	1
S20	100	1	S0	000	1
S25	101	0	S0	000	1
S25	101	1	S0	000	1

D.



E.

--6.111, Spring 2002
 --Homework 3 solution
 --Problem 3, Part E

```
library ieee;
use ieee.std_logic_1164.all;
```

```

entity soda is
  port (
    X, clk : in std_logic;
    Z      : out std_logic);
end soda;

```

```

architecture state_machine of soda is

```

```

  type StateType is (S0, S5, S10, S15, S20, S25);
  attribute enum_encoding of StateType : type is
    "000 001 010 011 100 101";
  signal p_s, n_s : StateType;

```

```

begin -- state_machine

```

```

  Z <= '1' when (p_s = S20) or (p_s = S25) else '0' ;

```

```

  state_clocked: process(clk)

```

```

  begin
    if rising_edge(clk) then p_s <= n_s;
    end if;
  end process state_clocked;

```

```

  FSM : process (p_s, X)

```

```

    --Combinational

```

```

    case p_s is
      when S0 =>
        if X='1' then n_s <=S10;
        else n_s      <=S5;
        end if;
      when S5 =>
        if X='1' then n_s <=S15;
        else n_s      <=S10;
        end if;
      when S10 =>
        if X='1' then n_s <=S20;
        else n_s      <=S15;
        end if;
      when S15 =>
        if X='1' then n_s <=S25;
        else n_s      <=S20;
        end if;
      when S20 =>
        n_s      <=S0;
      when S25 =>
        n_s      <=S0;
    end case;

```

```
when others =>
  n_s      <=S0;
end case;
end process FSM;
end architecture state_machine;
```