Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.111 - Introductory Digital Systems Laboratory

Problem Set 4 Solutions

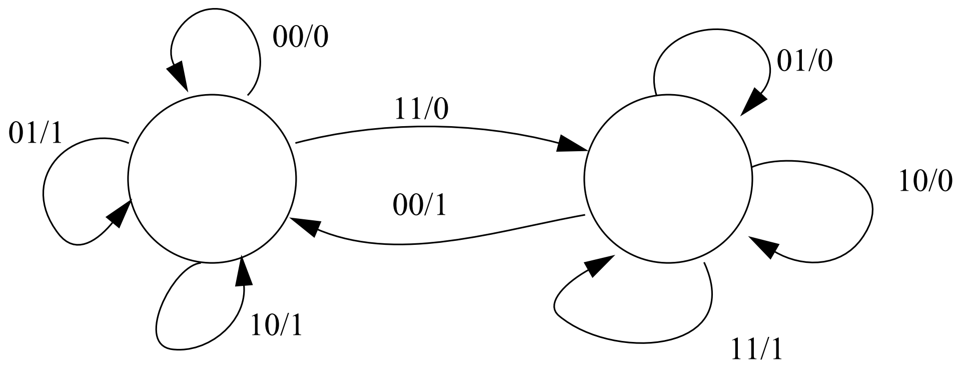
Issued: Lecture 12 Day

Problem 1:
1)

x	y	z	s	c
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2)

inputs: xy   output: S



3)
Mealy. S depends on the state and the inputs.

4)
After a clock edge, it takes 2 us for the output of the flip flop to change. Then 4 us later, the output of the adder changes. This value needs to be held for 2 us to satisfy the setup time of the flip flop. Since the contamination delay of each part is at least 2 us, we could have another edge now and the setup time would be satisfied. So the minimum clock period is 6 us.

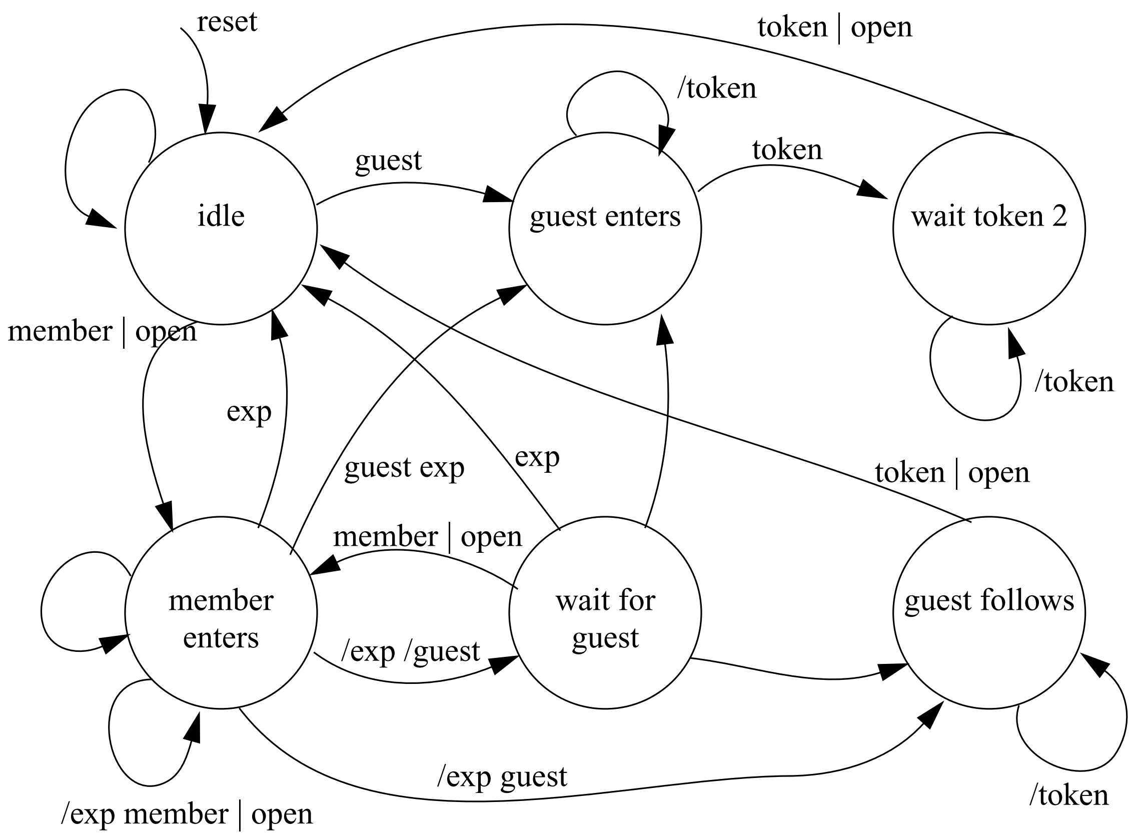
Problem 2:

1)
Below is a truth table that corresponds to the FSM. The state labels can be mapped to a three bit state variable. All entries not entered below are illegal.

state	member	guest	exp	token	next state	open
idle	0	0	X	X	idle	0
idle	0	1	X	X	guest enters	0
idle	1	0	X	X	member enters	1
member enters	0	0	0	X	wait for guest	0
member enters	0	0	1	X	idle	0
member enters	0	1	0	X	guest follows	0
member enters	0	1	1	X	guest enters	0
member enters	1	0	X	X	member enters	1
wait for guest	0	0	0	X	wait for guest	0
wait for guest	0	0	1	X	idle	0
wait for guest	0	1	0	X	guest follows	0
wait for guest	0	1	1	X	guest enters	0
wait for guest	1	0	X	X	member enters	1
guest follows	X	X	X	0	guest follows	0
guest follows	X	X	X	1	idle	1
guest enters	X	X	X	0	guest enters	0
guest enters	X	X	X	1	wait token 2	0
wait token 2	X	X	X	0	wait token 2	0
wait token 2	X	X	X	1	idle	1
XXX	1	1	X	X	illegal	0

2)
Not graded.

3)
Showing all dependencies is messy. See truth table for clarifications.



4)
Mealy. Open depends on inputs and state.

Problem 3:

This code will implement the 3-bit shifter:

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

entity shifter is
    port (
        clk : in std_logic;  -- clock signal
        i : in std_logic_vector(3 downto 0);   -- input bits
        s : in std_logic_vector(1 downto 0);   -- select bits
        o : out std_logic_vector(3 downto 0);-- output bits
end shifter;

architecture structure of shifter is
    signal internal : std_logic_vector(3 downto 0);    -- stores output values
                                                       -- internally for use by circuit
begin   -- structure
    stuff: process (clk)
    begin
        if rising_edge(clk) then
            if s=”00” then                -- if select is 00, hold
                internal <= internal;
            elsif s=”01” then             -- if select is 01, shift left
                internal(0) <= i(3);
                internal(1) <= i(0);
                internal(2) <= i(1);
                internal(3) <= i(2);
            elsif s=”10” then             -- if select is 10, shift right
                internal(0) <= i(1);
                internal(1) <= i(2);
                internal(2) <= i(3);
                internal(3) <= i(0);
            elsif s=”11” then             -- if select is 11, load input bits
                internal <= i;
            else                          -- this should not happen, but if it does,
                                          -- we will do a paralell load
                internal <= i;
            end if;
        end if;
    end process stuff;

    o<= internal;       -- now we send the internal count to the output

end structure;
```