

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

Problem Set 2 Solutions

Issued: February 19, 2003

Problem 1

(a)

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

This circuit is a two input multiplexor with a selectable inverter on the output. Input B selects between the C and D input, and A is the inversion selector.

(b)

CD \ AB	AB			
	00	01	11	10
00			1	1
01		1		1
11	1	1		
10	1		1	

The MSP expression here is $F = \bar{A}\bar{B}C + \bar{A}BD + AB\bar{D} + A\bar{B}\bar{C}$

(c)

The MSP expression is not free from hazards. There are two hazards that exist; the first is when $A=0$, $C=D=1$. The second is when $A=1$, $C=D=0$. In both cases, a transition on the unnamed variable can cause a glitch. To solve this problem, horizontal circlings represented by $\bar{A}CD$ and $A\bar{C}\bar{D}$ can remove the hazards.

(d)

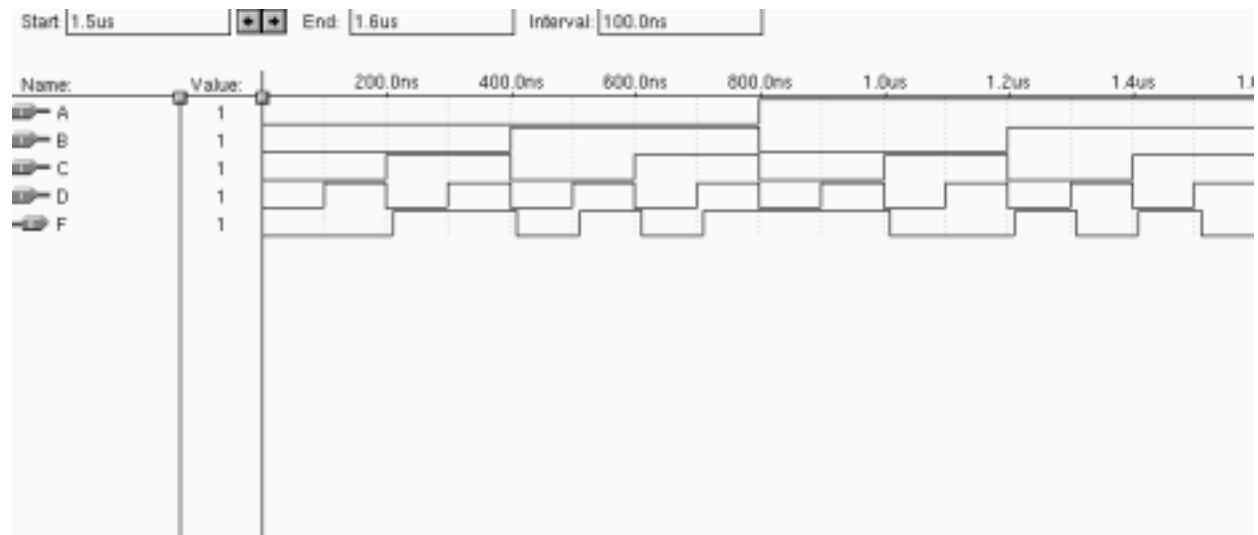
-- This is a sample solution to PS2 Problem 1d

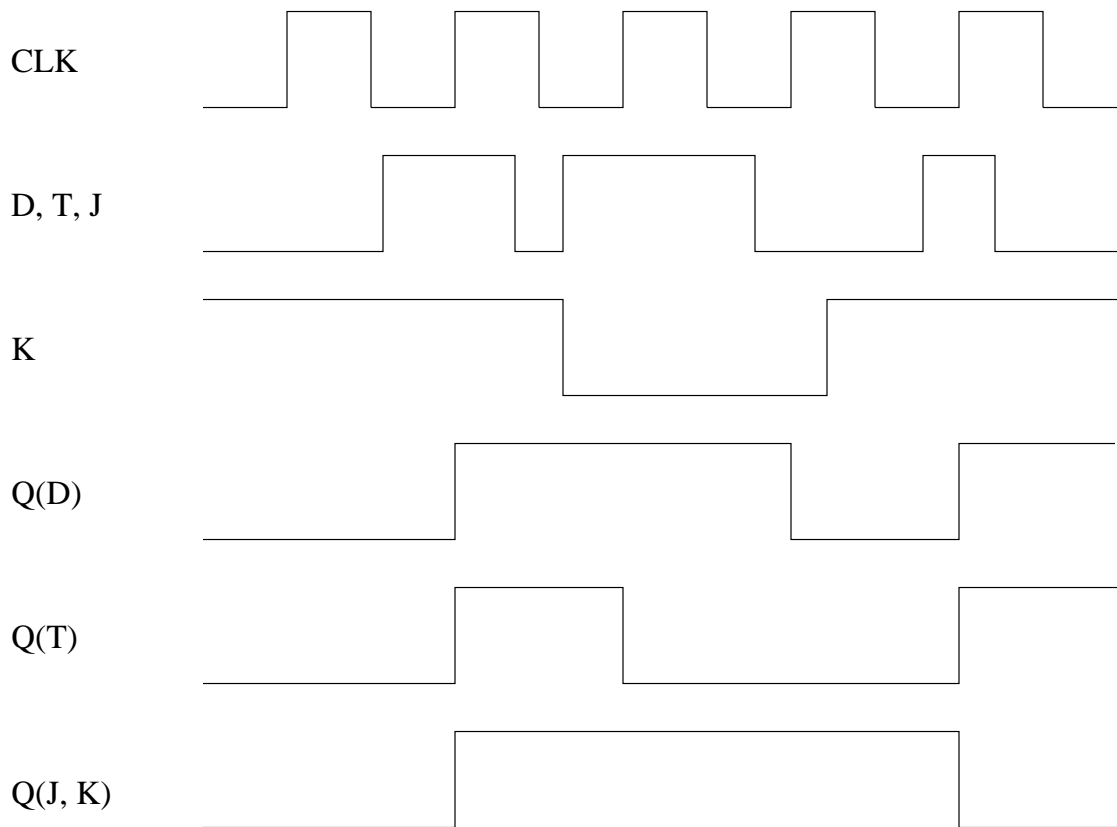
```
library ieee;
use ieee.std_logic_1164.all;

entity ps2 is port(
A      : in std_logic;
B      : in std_logic;
C      : in std_logic;
D      : in std_logic;
F      : out std_logic);
end ps2;

architecture mysteryfunc of ps2 is

begin
    F <= (not A and not B and C) or
        (not A and B and D) or
        (A and B and not D) or
        (A and not B and not C);
end mysteryfunc;
```



Problem 2**Problem 3****(a)**

Counters are composed of a set of flip-flops that hold a certain state, namely, the count. The design of these two different types of counters has several advantages and disadvantages.

The '393 is a very simple design; it is best modeled as four negative-edge triggered T flip-flops connected together in series. A falling edge causes a bit to flip, and can successively flip higher bit flip-flops. Because the negative-edge triggered flip-flops make the logic much simpler compared to the '163, the '393 takes up the least amount of space. However, this has its disadvantages since for the most significant bit to change, all the predecessor bits must sequentially flip; the worst case propagation delay for a 4-bit counter is four times the propagation delay for a flip-flop.

The '163 is more efficient in terms of performance, but it consumes much more space to build. The worst case propagation delay is only the propagation delay for a single flip-flop, since all of the flip-flops are processing in parallel. However, since the '163 is positive-triggered, it is much more difficult to determine the next count state; extra combinational logic using the current count as inputs and the next count will allow the count transitions to work properly.

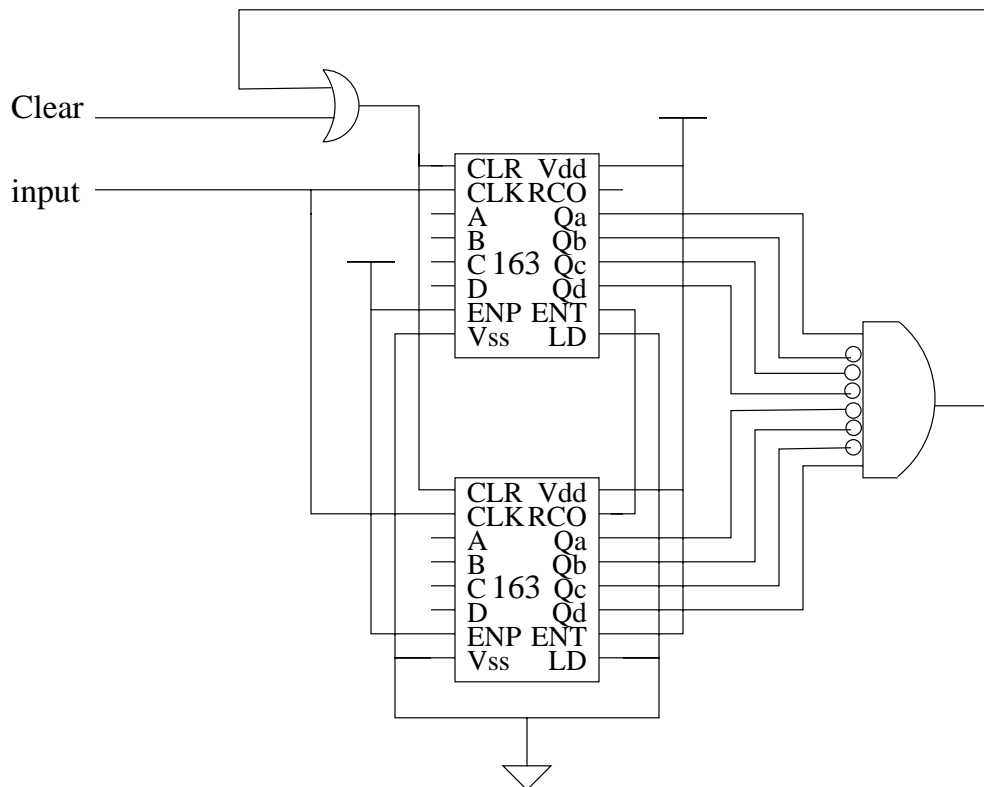
(b)

Since the RCO is determined by an AND of the output bits of the '163, the RCO output can be glitchy; while all the counter bits do change in parallel, they can still transition at slightly different times due to clock skew between the flip-flops and specific characteristics of each flip-flop. For this reason, connecting the RCO to the CLK input of the next counter could cause the next counter to count at inappropriate times. Instead, connect the RCO to the T input of the next counter, and connect the system clock to both counters. This will only allow the top counter to count when RCO is high.

(c)

The following circuit diagram follows the specifications on the problem set. The lower '163 outputs the lower order four bits of the total count; the upper '163 outputs the additional bit necessary for a range from 0 to 24 inches.

The circuit allows a reset of the count at any time by asserting COUNT.



(d)

The code for the '163:

```
-- This is a sample solution to PS2 Problem 3d
```

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

entity ls163 is port(
    clk      : in std_logic;
    clear    : in std_logic;
    load     : in std_logic;
    ent      : in std_logic;
    enp      : in std_logic;
    inputs   : in std_logic_vector(3 downto 0);
    rco      : out std_logic;
    q        : out std_logic_vector(3 downto 0));
end ls163;
```

```
architecture ls163arch of ls163 is
```

```
    signal count : std_logic_vector(3 downto 0);
```

```

begin
  q <= count;
  rco <= count(3) and count(2) and count(1) and count(0) and ent;

  counter : process(clk)
  begin
    if rising_edge(clk) then
      if (clear = '0') then
        count <= "0000";
      elsif (load = '0') then
        count <= inputs;
      elsif (ent = '1' and enp = '1') then
        count <= count + 1;
      end if;
    end if;
  end process counter;
end ls163arch;

```

The code for the snow counter:

-- This is a sample solution to PS2 Problem 3d

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity snowfall is port(
  input      : in std_logic;
  clear      : in std_logic;
  vdd        : in std_logic;
  fourzero  : in std_logic_vector(3 downto 0);
  overflow   : out std_logic;
  snow       : out std_logic_vector(7 downto 0));
end snowfall;

architecture snowfallarch of snowfall is

  signal carry      : std_logic;
  signal snowintlow : std_logic_vector(3 downto 0);
  signal snowinhigh : std_logic_vector(3 downto 0);
  signal clearsig   : std_logic;
  component ls163
  port (clk      : in std_logic;
        clear    : in std_logic;
        load     : in std_logic;
        ent      : in std_logic;
        enp      : in std_logic;
        inputs   : in std_logic_vector(3 downto 0);
        rco      : out std_logic;
        q        : out std_logic_vector(3 downto 0));
  end component;

begin

  clearsig <= not (clear or (not snowinhigh(3) and
                          not snowinhigh(2) and

```

```
        not snowinhigh(1) and
        snowinhigh(0) and
        snowintlow(3) and
        not snowintlow(2) and
        not snowintlow(1) and
        not snowintlow(0)));

snow <= snowinhigh&snowintlow;

lower : ls163
  port map (clk    => input,
            clear  => clearsig,
            load   => vdd,
            ent    => vdd,
            enp    => vdd,
            inputs => fourzero,
            rco    => carry,
            q      => snowintlow);

upper : ls163
  port map (clk    => input,
            clear  => clearsig,
            load   => vdd,
            ent    => carry,
            enp    => vdd,
            inputs => fourzero,
            rco    => overflow,
            q      => snowinhigh);

end snowfallarch;
```