

Massachusetts Institute of Technology  
Department of Electrical Engineering and Computer Science

6.111 - Introductory Digital Systems Laboratory  
Problem Set #2 Solutions

Problem 1: Fun with Combinational Logic

(a)

$$F = \overline{\overline{(A \cdot \bar{B})((B \cdot C) + (\bar{B} \cdot \bar{C}))(\bar{C} + \bar{D})}}$$

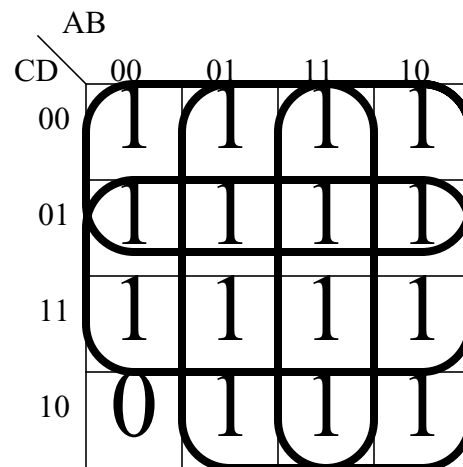
$$F = (A \cdot \bar{B}) + (B \cdot C) + (\bar{B} \cdot \bar{C}) + \bar{C} + D$$

$$F = (A \cdot \bar{B}) + (B \cdot C) + \bar{C} + D$$

(b)

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

(c)



$$F = A + B + \bar{C} + D$$

(d) Unique? yes  
Hazard free? yes

```

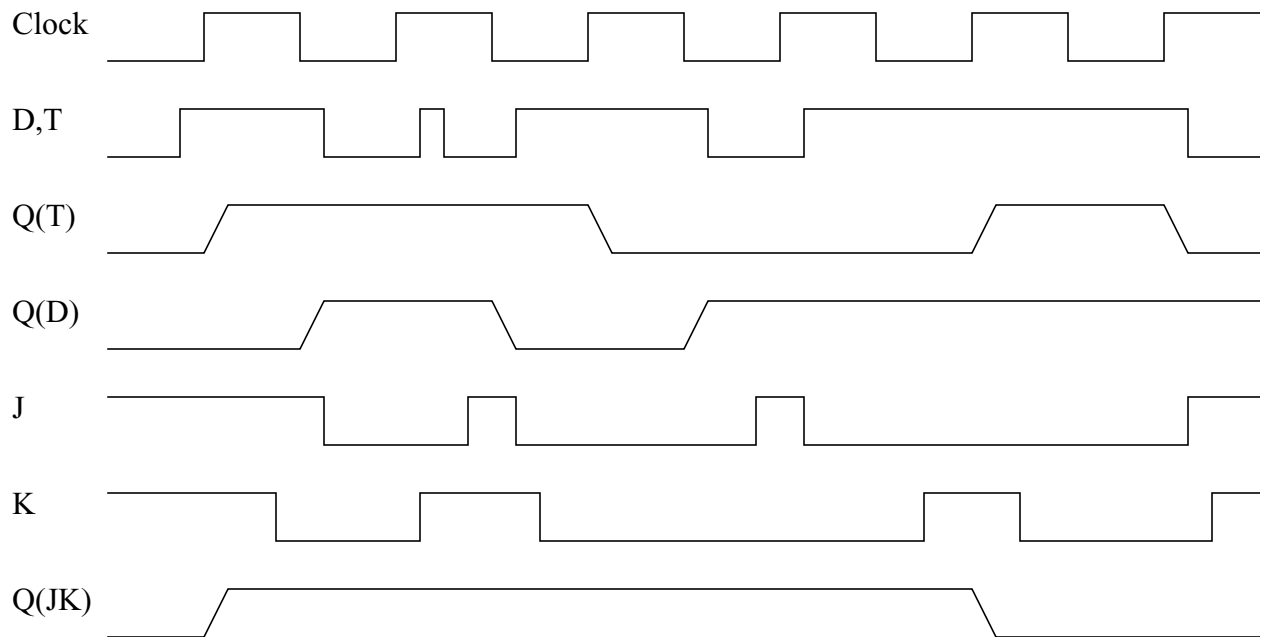
(e)
library ieee;
use ieee.std_logic_1164.all;

entity p1e is
  port (clk, a, b, c, d : in std_logic;
        f : out std_logic);
end p1e;

architecture behavioral of p1e is
begin
  process(clk)
  begin
    if rising_edge(clk) then
      f<= a or b or (not c) or d;
    end if;
  end process;
end behavioral;

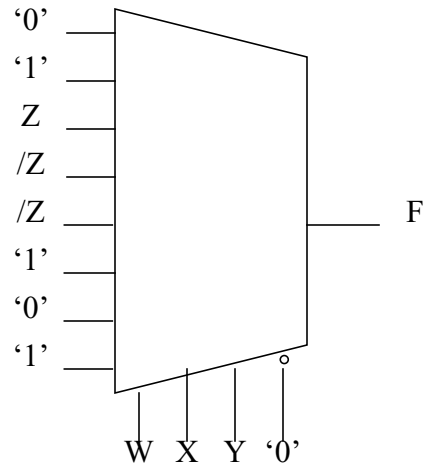
```

Problem 2: Thou shall know his Flip-Flops



### Problem 3: Multiplexers are your Friends

W	X	Y	Z	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	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

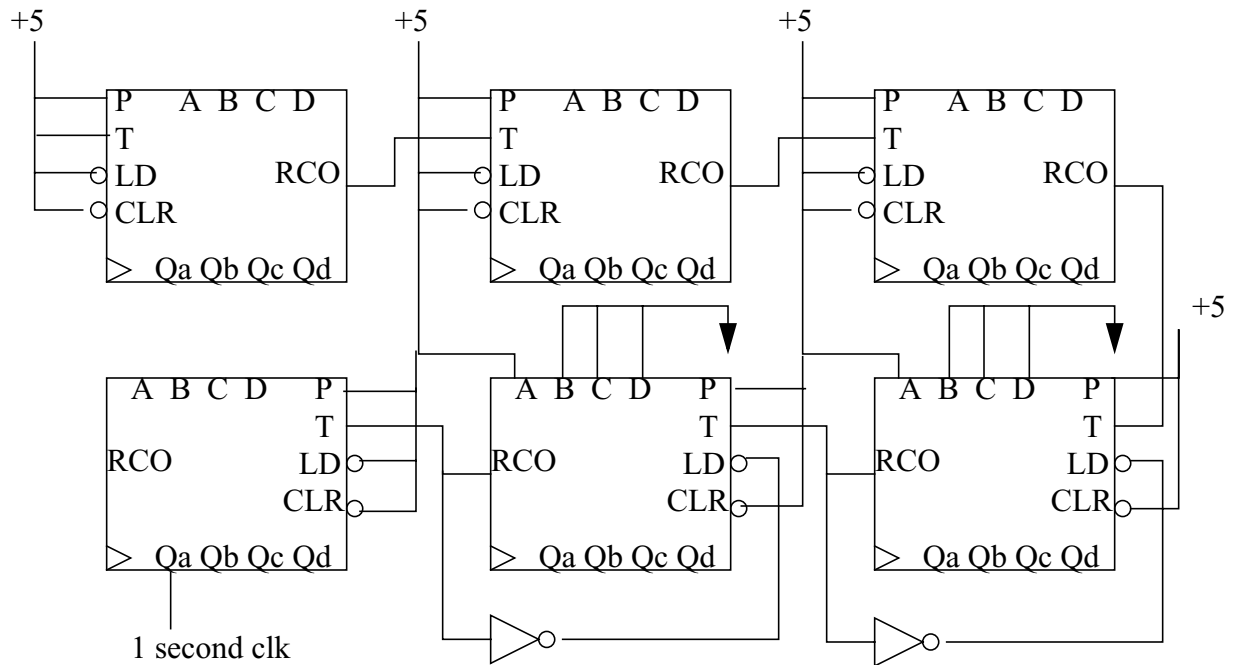


### Problem 4: Counters are Crucial

(a) All counters are constructed out of flip flop components. A synchronous counter has flip flops which are all clocked on the same signal whereas a ripple counter uses the output of one flip flop as the clock of the next. Because of this, a synchronous counter has less delay from a transition on the clock to the next stable state.

(b) The RCO can have glitches, so the second counter may use a glitch as a clock edge and increment the count incorrectly.

(c) There are several ways to implement the solution. This is one example.



(d) There are several ways to implement the solution. This is one example.

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

```
entity p4d is
  port (clk, a, b, c, d, p, t, ld, clr: in std_logic;
        Q : out std_logic(4 downto 0)); -- RCO is bit 4
end p4d;
```

```
architecture behavioral of p4d is
begin
  process(clk)
  begin
    if rising_edge(clk) then
      if ld='0' then
        Q<=d&c&b&a;
      elsif clr='0' then
        Q<="0000";
      elsif (p='1') and (t='1') then
        Q<=Q+1;
      end if;
    end if;
  end process;
end architecture;
```

```
    else
      Q<=Q;
    end if;
  end process;
end behavioral;
```

```
entity top is
  port (clk : in std_logic;
        sec_clk : out std_logic);
end top;
```

```
architecture behavioral of top is
  component p4d is
    port(clk, a, b, c, d, p, t, ld, clr: in std_logic;
         Q : out std_logic(4 downto 0)); -- RCO is bit 4
  end component;
```

```
  signal one, zero: std_logic;
  signal ld1, ld2: std_logic;
  signal Q1, Q2, Q3, Q4, Q5, Q6 : std_logic_vector(4 downto 0);
```

```
begin
  one<='1';
  zero<='0';
  ld1<=not Q4(4);
  ld2<=not Q5(4);
  sec_clk<=Q6(0);
```

```
  count1: p4d
    port map (
      clk=>clk,
      a=>zero,
      b=>zero,
      c=>zero,
      d=>zero,
      p=>one,
      t=>one,
      ld=>one,
      clr=>one,
      Q=>Q1);
```

```
  count2: p4d
    port map (
      clk=>clk,
      a=>zero,
```

```
b=>zero,  
c=>zero,  
d=>zero,  
p=>one,  
t=>Q1(4),  
ld=>one,  
clr=>one,  
Q=>Q2);
```

```
count3: p4d  
port map (  
  clk=>clk,  
  a=>zero,  
  b=>zero,  
  c=>zero,  
  d=>zero,  
  p=>one,  
  t=>Q2(4),  
  ld=>one,  
  clr=>one,  
  Q=>Q3);
```

```
count4: p4d  
port map (  
  clk=>clk,  
  a=>one,  
  b=>zero,  
  c=>zero,  
  d=>zero,  
  p=>one,  
  t=>Q3(4),  
  ld=>ld1,  
  clr=>one,  
  Q=>Q4);
```

```
count5: p4d  
port map (  
  clk=>clk,  
  a=>one,  
  b=>zero,  
  c=>zero,  
  d=>zero,  
  p=>one,  
  t=>Q4(4),  
  ld=>ld2,  
  clr=>one,
```

```
Q=>Q5);
```

```
count1: p4d
```

```
port map (
```

```
  clk=>clk,
```

```
  a=>zero,
```

```
  b=>zero,
```

```
  c=>zero,
```

```
  d=>zero,
```

```
  p=>one,
```

```
  t=>Q5(4),
```

```
  ld=>one,
```

```
  clr=>one,
```

```
  Q=>Q6);
```

```
end behavioral;
```