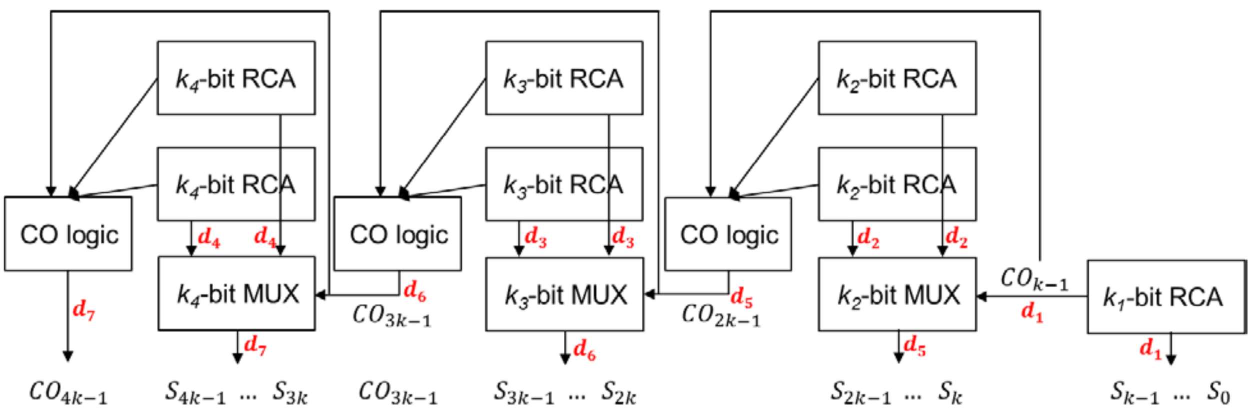
Homework Assignment 9

(Due May 3rd, 12pm)

1. [Carry Select Adder, **40 points**] In the lecture note, we used k-bit adders to design an N-bit carry select adder. However, we can use variable-length adders instead of fixed-length adders. The following shows the specification of logic blocks we are going to use:
   - N: 64
   - Delay of a full adder: $\Delta_{FA} = 100ps$
   - Delay of a k-bit ripple carry adder (RCA): $k \cdot \Delta_{FA}$
   - Delay of a k-bit MUX (when $k \geq 10$) and a CO logic: $\varepsilon = 150ps$
   - Architecture: We split N into four groups as follows:



Since N is 64, the above architecture should satisfy the following equation:

$$k_1 + k_2 + k_3 + k_4 = 64$$

We also assume that $k_i \geq 10$ ($i = 1 \sim 4$).

1) Represent delay of the 64-bit adder shown above as a function of k_1, k_2, k_3, k_4, and the MAX(a,b) function where "MAX(a, b) = a if (a>b) or b (if b>a)".
   - Delay of the k_1-bit RCA: $d_1 = k_1 \cdot \Delta_{FA} = 100k_1$
   - Delay of the k_2-bit RCA: $d_2 = k_2 \cdot \Delta_{FA} = 100k_2$
   - Delay of the k_3-bit RCA: $d_3 = k_3 \cdot \Delta_{FA} = 100k_3$
   - Delay of the k_4-bit RCA: $d_4 = k_4 \cdot \Delta_{FA} = 100k_4$
   - Arrival time (AT) at the output of the k_2-bit MUX and the first Carry-Out logic: $d_5 = \text{MAX}(d_1, d_2) + \varepsilon = 150 + \text{MAX}(100k_1, 100k_2)$
   - AT at the output of the k_3-bit MUX and the second Carry-Out logic: $d_6 = \text{MAX}(d_3, d_5) + \varepsilon = 150 + \text{MAX}\{100k_3, 150 + \text{MAX}(100k_1, 100k_2)\}$
   - AT at the output of the k_4-bit MUX and the third Carry-Out logic: $d_7 = \text{MAX}(d_4, d_6) + \varepsilon = \boxed{150 + \text{MAX}[100k_4, 150 + \text{MAX}\{100k_3, 150 + \text{MAX}(100k_1, 100k_2)\}]}$

2) Compute the total delay when $k_1 = k_2 = k_3 = k_4$.

- $k_1 = k_2 = k_3 = k_4 = 16$.
- $d_7 = 150 + MAX[1600, 150 + MAX\{1600, 150 + MAX(1600, 1600)\}]$
  $= 150 + MAX[1600, 150 + MAX\{1600, 150 + 1600\}]$
  $= 150 + MAX[1600, 150 + 1750]$
  $= 150 + 1900 = \boxed{2050ps}$

3) Compute $k_1, k_2, k_3, and\ k_4$ minimizing the delay and show the minimum delay. (Hint: (1) Use your intuition and some math. (2) If you want, you can program it to find k_1, k_2, k_3, k_4. In this case, you should show your program in your report).

1) We will first show that $k_1 = k_2$ will give us an optimal delay. Suppose MAX($100k_1$, $100k_2$) affects the final delay. Then, $k_1 = k_2$ will minimize MAX($100k_1$, $100k_2$), which will minimize the final delay.

$k_1 = k_2$ leads to $d_7 = 150 + \text{MAX}[100k_4, 150 + \text{MAX}\{100k_3, 150 + 100k_1\}]$.

By the same reason, $100k_3 = 150 + 100k_1$ will give us an optimal delay. From this, we get $k_3 = k_1 + 1.5$. However, k_3 should be an integer. If $k_3 \leq k_1 + 1$, we get $d_7 = 150 + \text{MAX}[100k_4, 150 + 150 + 100k_1]$. However, reducing k_3 will increase k_4, so let's set k_3 to $k_1 + 1$.

$d_7 = 150 + \text{MAX}[100k_4, 300 + 100k_1]$. Setting $100k_4 = 300 + 100k_1$ will minimize $d_7 \rightarrow k_4 = 3 + k_1$

From $k_1 + k_2 + k_3 + k_4 = 64$, we get $k_1 + k_1 + (k_1 + 1) + (k_1 + 3) = 64 \rightarrow$
$\boxed{k_1 = 15.\ k_2 = 15.\ k_3 = 16.\ k_4 = 18.}$

Delay = 150+MAX[1800, 150+MAX{1600, 150+MAX(1500, 1500)}]= $\boxed{\text{1950ps.}}$

2) We can also use a computer program to simulate this. The following C/C++ code simulates it.

```
#include <stdio.h>

int max (int a, int b) {
   if ( a > b )
      return a;
   return b;
}
```

```
int main () {
   int min_delay = 100000000;  // min. delay achieved

   for ( int k4 = 10 ; k4 <= 34 ; k4++ ) {
      for ( int k3 = 10 ; k3 <= (44–k4) ; k3++ ) {
         for ( int k2 = 10 ; k2 <= (54–k4–k3) ; k2++ ) {
            int k1 = 64 – (k4 + k3 + k2);
            int delay = 150+max(100*k4, 150+max(100*k3, 150+max(100*k1, 100*k2)));

            if ( delay <= min_delay ) {
               printf (“(k4, k3, k2, k1, d) = (%d, %d, %d, %d, %d)\n”, k4, k3, k2, k1, delay);
               min_delay = delay;
            }
         }
      }
   }
   return 0;
}
```

You can also download the following file:

http://eecs.wsu.edu/~ee434/Homework/add.cpp

and compile it as follows (in the ee434-466 server):

```
g++ add.cpp
```

which will generate a.out in your directory. Then, run it to see its usage:

```
> ./a.out
```

The following shows the usage:

```
./a   {delta_FA}  {delta_MUX}
```

Run the program for the above problem as follows:

```
./a.out  100  150
```

which gives the following result (format: k4 k3 k2 k1   total delay in ps):

```
delta_FA: 100 (ps)   delta_MUX: 150 (ps)
18  16  15  15     1950
```

You can also try some different combinations as follows:

```
./a.out 100 200
```

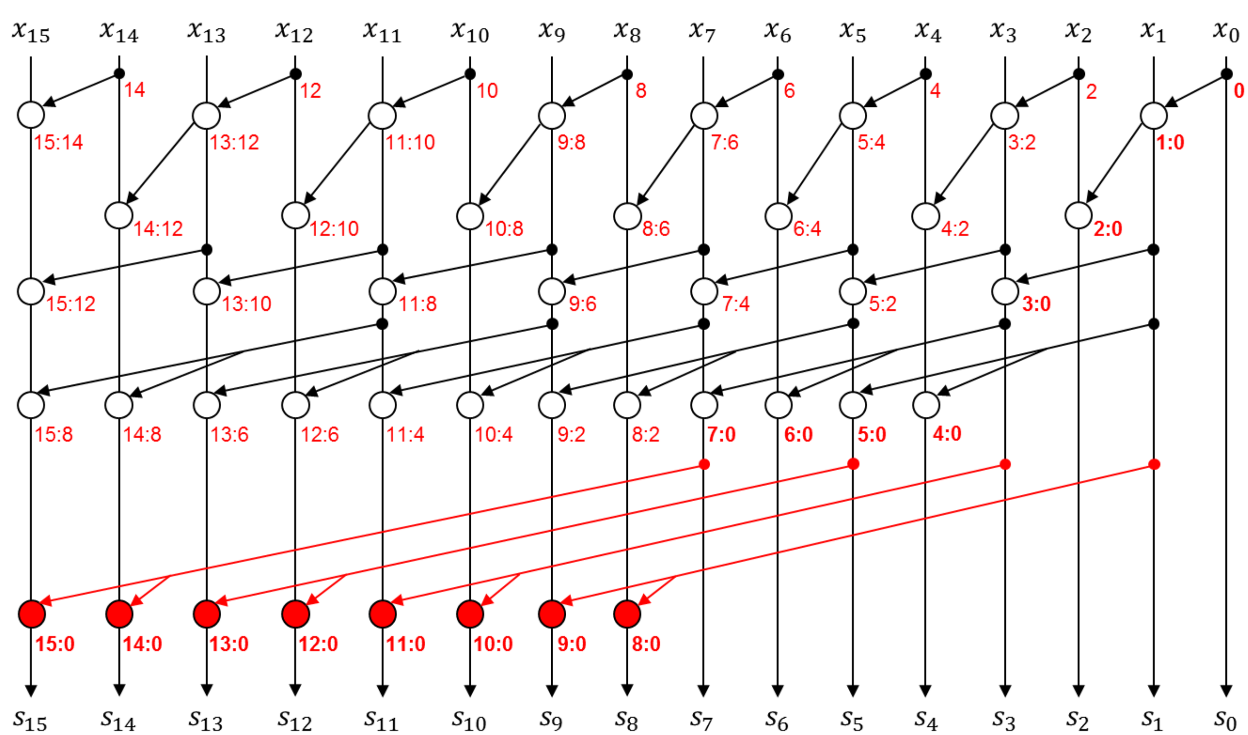
which gives the following result:

```
17 17 15 15    2100
18 16 15 15    2100
```

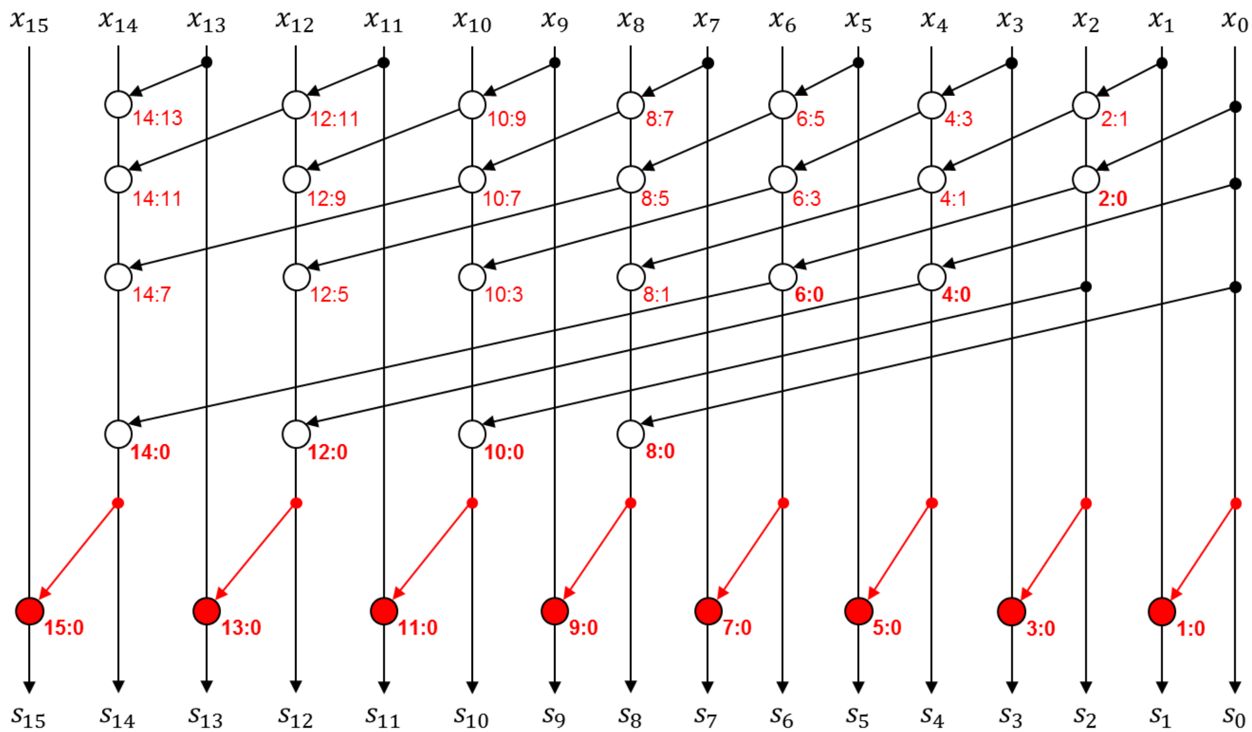
18 17 14 15	2100
18 17 15 14	2100
19 15 15 15	2100
19 16 14 15	2100
19 16 15 14	2100
19 17 13 15	2100
19 17 14 14	2100
19 17 15 13	2100

2. [Prefix Adder, **40 points**] Complete the following prefix adders by inserting merging blocks and drawing arrows (try to minimize the number of merging blocks inserted).

1)



2)



x_{15} x_{14} x_{13} x_{12} x_{11} x_{10} x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
14:13
12:11
10:9
8:7
6:5
4:3
2:1
14:11
12:9
10:7
8:5
6:3
4:1
2:0
14:7
12:5
10:3
8:1
6:0
4:0
14:0
12:0
10:0
8:0
15:0
13:0
11:0
9:0
7:0
5:0
3:0
1:0
s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0