# EE466

# VLSI System Design

## Midterm Exam

**Oct. 15, 2020. (4:20pm – 5:35pm)**

**Instructor: Dae Hyun Kim (daehyun@eecs.wsu.edu)**

**Name:**

**WSU ID:**

| Problem | Points | |
|---|---|---|
| 1 | 10 | |
| 2 | 10 | |
| 3 | 10 | |
| 4 | 10 | |
| 5 | 10 | |
| 6 | 10 | |
| Total | 60 | |

* Allowed: Textbooks, cheat sheets, class notes, notebooks, calculators, watches

* Not allowed: Electronic devices (smart phones, tablet PCs, laptops, etc.) except calculators and watches

## Problem #1 (Kogge-Stone Adder, 10 points).

For the 1024-bit Kogge-Stone adder, show one of the critical paths to calculate $S_{789}$.

$$S_{789} = p_{789} \oplus C_{789}$$

$$C_{789} = g_{788:0} + p_{788:0} \cdot C_0$$

$$g_{788:0} = g_{788:277} + p_{788:277} \cdot g_{276:0}$$

$$g_{788:277} = g_{788:533} + p_{788:533} \cdot g_{532:277}$$

$$g_{788:533} = g_{788:661} + p_{788:661} \cdot g_{660:533}$$

$$g_{788:661} = g_{788:725} + p_{788:725} \cdot g_{724:661}$$

$$g_{788:725} = g_{788:757} + p_{788:757} \cdot g_{756:725}$$

$$g_{788:757} = g_{788:773} + p_{788:773} \cdot g_{772:757}$$

$$g_{788:773} = g_{788:781} + p_{788:781} \cdot g_{780:773}$$

$$g_{788:781} = g_{788:785} + p_{788:785} \cdot g_{784:781}$$

$$g_{788:785} = g_{788:787} + p_{788:787} \cdot g_{786:785}$$

$$g_{788:787} = g_{788} + p_{788} \cdot g_{787}$$

$$p_{788} = A_{788} \oplus B_{788}$$

## Problem #2 (Carry-Lookahead Adder, 10 points).

For the 1024-bit Carry-lookahead adder, show one of the critical paths to calculate $S_{789}$.

$$S_{789} = p_{789} \oplus C_{789}$$

$$C_{789} = g_{788} + p_{788} \cdot C_{788}$$

$$C_{788} = g_{787:784} + p_{787:784} \cdot C_{784}$$

$$C_{784} = g_{783:768} + p_{783:768} \cdot C_{768}$$

$$C_{768} = g_{767:512} + p_{767:512} \cdot g_{511:256} + p_{767:512} \cdot p_{511:256} \cdot g_{255:0} + p_{767:512} \cdot p_{511:256} \cdot p_{255:0} \cdot C_0$$

$$g_{255:0} = g_{255:192} + p_{255:192} \cdot g_{191:128} + p_{255:192} \cdot p_{191:128} \cdot g_{127:64} + p_{255:192} \cdot p_{191:128} \cdot p_{127:64} \cdot g_{63:0}$$

$$g_{63:0} = g_{63:48} + p_{63:48} \cdot g_{47:32} + p_{63:48} \cdot p_{47:32} \cdot g_{31:16} + p_{63:48} \cdot p_{47:32} \cdot p_{31:16} \cdot g_{15:0}$$

$$g_{15:0} = g_{15:12} + p_{15:12} \cdot g_{11:8} + p_{15:12} \cdot p_{11:8} \cdot g_{7:4} + p_{15:12} \cdot p_{11:8} \cdot p_{7:4} \cdot g_{3:0}$$

$$g_{3:0} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

$$p_0 = A_0 \oplus B_0$$

## Problem #3 (Conditional Sum Adder, 10 points)

How many 1-bit multiplexers do we need to implement a 64-bit conditional sum adder? (do not assume that $C_0$ is 0. $C_0$ could be 0 or 1.)

Step 1: 0 (In Step 1, we just use FAs.)

Step 1→2: 4*(64/2) = 128 (when we merge two bit positions, we need four muxes.)

Step 2→3: 6*(64/4) = 96 (when we merge four bit positions, we need six multiplexers.)

Step 3→4: 10*(64/8) = 80 (when we merge eight bit positions, we need ten multiplexers.)

Step 4→5: 18*(64/16) = 72

Step 5→6: 34*(64/32) = 68

Step 6→7: 33*(64/64) = 33

Total 477 multiplexers.

## Problem #4 (Carry Skip Adder, 10 points)

For the 64-bit Carry-skip adder, show how $S_{33}$ is generated from the primary input signals. Assume that the 4-bit blocks in the carry-skip adder are 4-bit ripple-carry adders with a separate logic generating $g_{i+3:i}$ and $p_{i+3:i}$ (i.e., the sum bits are generated purely by ripple-carry adders when $C_{4k}$ is available). For the output $Y$ of a two-input MUX ($S$: selection, $J$: input selected for $S = 0$, $K$: input selected for $S = 1$), you can use the following Boolean equation: $Y = \bar{S} \cdot J + S \cdot K$.

$$S_{33} = A_{33} \oplus B_{33} \oplus C_{33}$$

$$C_{33} = A_{32} \cdot B_{32} + C_{32} \cdot (A_{32} + B_{32})$$

$$C_{32} = \overline{p_{31:28}} \cdot g_{31:28} + p_{31:28} \cdot C_{28}$$

$$C_{28} = \overline{p_{27:24}} \cdot g_{27:24} + p_{27:24} \cdot C_{24}$$

$$C_{24} = \overline{p_{23:20}} \cdot g_{23:20} + p_{23:20} \cdot C_{20}$$

$$C_{20} = \overline{p_{19:16}} \cdot g_{19:16} + p_{19:16} \cdot C_{16}$$

$$C_{16} = \overline{p_{15:12}} \cdot g_{15:12} + p_{15:12} \cdot C_{12}$$

$$C_{12} = \overline{p_{11:8}} \cdot g_{11:8} + p_{11:8} \cdot C_{8}$$

$$C_{8} = \overline{p_{7:4}} \cdot g_{7:4} + p_{7:4} \cdot C_{4}$$

$$C_{4} = \overline{p_{3:0}} \cdot g_{3:0} + p_{3:0} \cdot C_{0}$$

$$g_{3:0} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$

$$p_0 = A_0 \oplus B_0$$

## Problem #5 (Carry Select Adder, 10 points)

We want to design an $n$-bit adder. The top-level architecture of the $n$-bit adder is the carry select adder architecture and is divided into $\frac{n}{m}$ groups and we use $m$-bit adders in each group. Now, we design the $m$-bit adders using the carry select adder architecture again. Each $m$-bit adder is divided into $\frac{m}{k}$ groups and we use $k$-bit adders in each group. (Thus, it is a recursively-designed carry select adder). The $k$-bit adders are ripple-carry adders. For the carry-out logic, we use a two-input MUX.

Express the worst-case delay as a function of $n, m, k, d, e$ where $d$ is the delay of a full adder and $e$ is the delay of a two-input MUX. (See page 8 and 9 in the lecture note.)

The delay of a $k$-bit adder is $k \cdot d$.

The delay of an $m$-bit carry select adder using k-bit adders is $k \cdot d + e \cdot \left(\frac{m}{k} - 1\right)$ (as shown in page 9).

If we recursively apply the delay to the top-level architecture, the delay is

$k \cdot d + e \cdot \left(\frac{m}{k} - 1\right) + e \cdot \left(\frac{n}{m} - 1\right)$.

## Problem #6 (Modified Booth Encoding, 10 points)

Use the modified Booth encoding technique to calculate the following multiplication (see page 9 in the multiplier lecture notes). Assume that all the numbers are unsigned.

```
      A          1 0 1 0 1 1 1 0
    * X          1 0 1 0 1 1 0 1
----------------------------------
  A              1 0 1 0 1 1 1 0
  * Y        0 1 0 1’0 1’0 1’0 1
----------------------------------
                              +A
                            -A
                         -A
                      -A
                   +A
----------------------------------
0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0
1 1 1 1 1 1 0 1 0 1 0 0 1 0
1 1 1 1 0 1 0 1 0 0 1 0
1 1 0 1 0 1 0 0 1 0
1 0 1 0 1 1 1 0
----------------------------------
0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 0
```