EE466

VLSI Design

Final Exam

Dec. 12, 2018. (3:10pm – 5:10pm)

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

Name:

WSU ID:

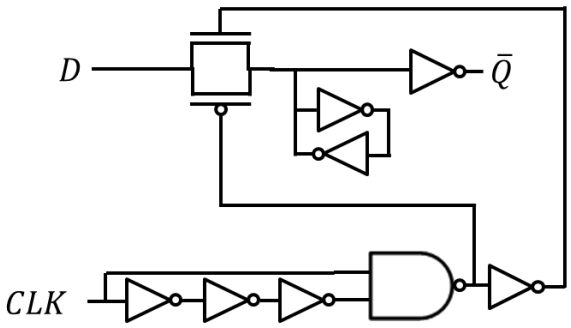
Problem	Points	
1	20	
2	10	
3	10	
4	20	
5	10	
6	10	
7	40	
8	50	
Total	170	

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

* Not allowed: Chat apps.

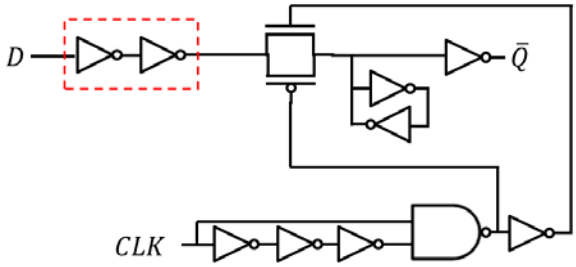
Problem #1 (Sequential Logic, 20 points)

Schematic A below shows an explicit-pulsed D flip-flop. The signal strength of input D is unknown (e.g., someone designed a circuit and its output is connected to input D).



<Schematic A>

Since we have no clue on the strength of input D, we suggest the following D-F/F design. We can properly size the two inverters in the dotted rectangle.



<Schematic B>

Question: Compare Schematic A and Schematic B (quantitatively and/or qualitatively) in terms of 1) setup time constraint, 2) hold time constraint, 3) clock-to-Q delay, and 4) output slew ($\Delta V_{\bar{Q}}/\Delta t$) where $V_{\bar{Q}}$ is the voltage at the output node $\bar{Q}$.

1) Setup time constraint: $t_{setup,A}$ < > = $t_{setup,B}$

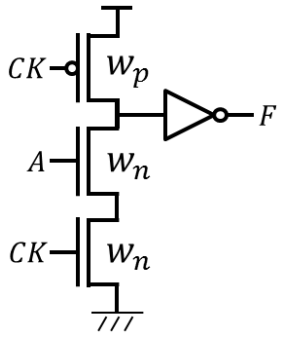
2) Hold time constraint: $t_{hold,A}$ < > = $t_{hold,B}$

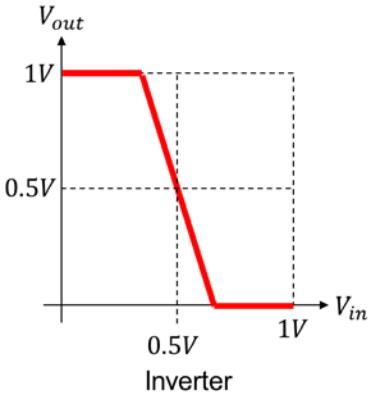
3) Clock-to-Q delay: $t_{C-Q,A}$ < > = $t_{C-Q,B}$

4) Output slew: s_A < > = s_B

Problem #2 (DC Analysis of a Domino Logic, 10 points)

The left figure shows a domino-logic-based buffer design and the right figure shows a DC curve of the inverter in the schematic. $\mu_n = 2\mu_p$. w: Transistor minimum width.





Draw (rough sketches will be accepted) three DC curves (x-axis: V_A, y-axis: V_F) for the buffer for

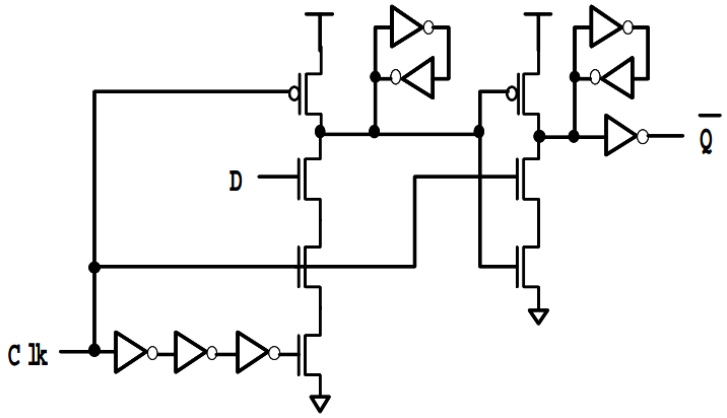
1) $w_p = w_n = w$

2) $w_p = 2w, w_n = w$

3) $w_p = w, w_n = 2w$.

Problem #3 (Sequential Logic, 10 points)

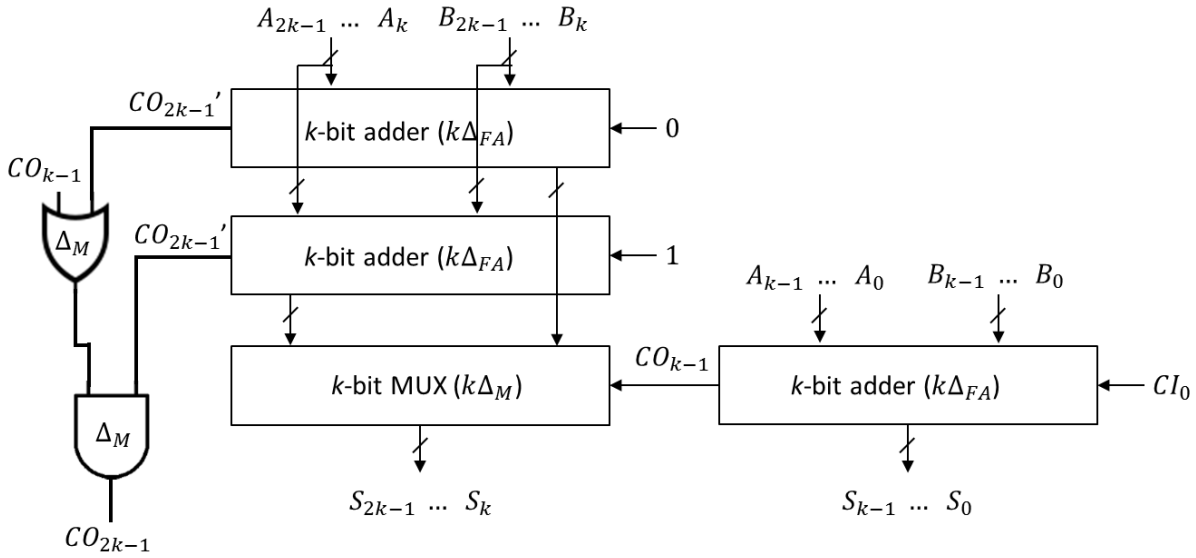
The following shows a schematic of a positive-edge triggered D-F/F.



Describe how you can estimate the hold time constraint of the F/F above. (Hold time constraint: input *D* should be stable (should not change) for some time after a clock rising edge.)

Problem #4 (Carry Select Adder, 20 points)

The following shows a schematic of a *2k*-bit adder designed using *k*-bit carry select adders. The delay of a *k*-bit adder is $k\Delta_{FA}$, the delay of a *k*-bit MUX is $k\Delta_M$, and the delay of a two-input AND (or OR) gate is Δ_M.

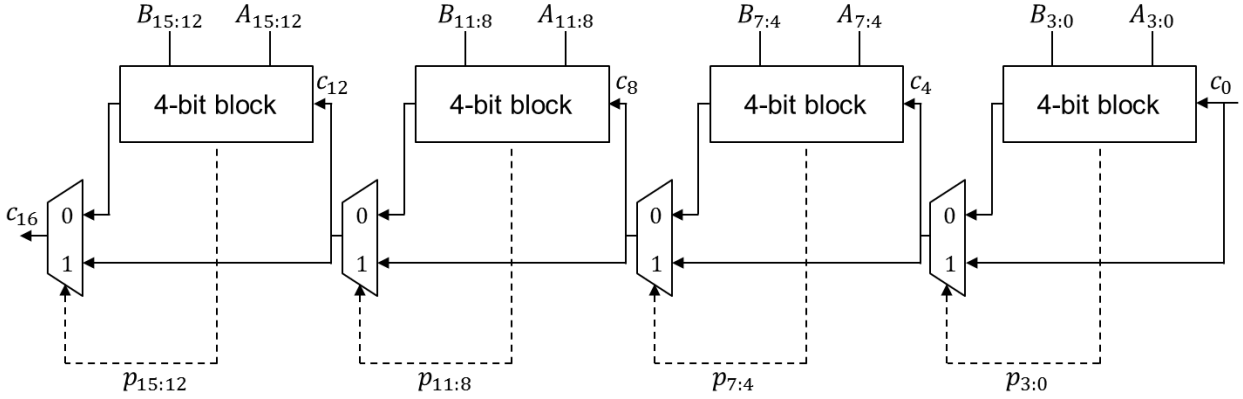


1) We are supposed to design an *N*-bit adder using carry select adders (# groups: $\frac{N}{k}$). Find k minimizing the delay of the *N*-bit adder (express the optimal *k* as a function of *N*, Δ_M, and Δ_{FA}). Notice that the worst-case delay occurs at C_N (the final carry out) or $S_{N-1:0}$ (the final sum).

2) Now, the *k*-bit adders are designed using conditional sum adders, so the delay of a *k*-bit adder is $\Delta_M \cdot \ln k$ instead of $k\Delta_{FA}$. Find k minimizing the delay of the new *N*-bit adder (express the optimal *k* as a function of *N* and Δ_M).

Problem #5 (Carry Skip Adder, 10 points)

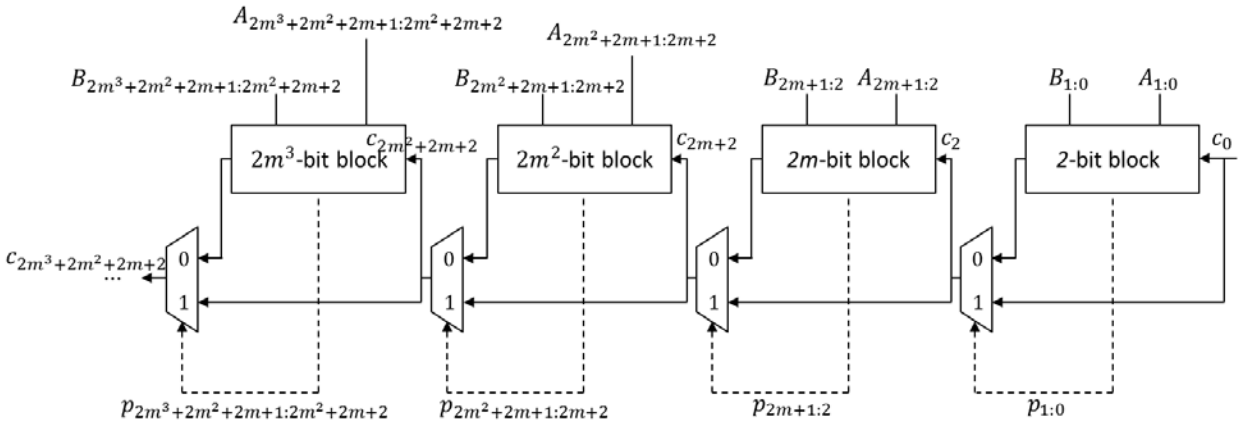
The following diagram shows a 16-bit carry-skip adder designed using 4-bit adders.



To improve the speed of the carry-skip adder, we replace the 4-bit adders (4-bit blocks designed using 4-bit ripple-carry adders) by 4-bit conditional sum adders. The delay of each multiplexer step in the conditional sum adders is Δ_M (so, if all the operands are available at time 0, the delay of a 4-bit conditional sum adder is $3\Delta_M$.) The delay of each 2:1 MUX in the schematic above is Δ_M. The delay of each $p_{i:i-3}$ is $2\Delta_M$. Calculate the delay of the new 16-bit carry-skip adder.

Problem #6 (Carry Skip Adder, 10 points)

To radically improve the delay of a carry-skip adder, we design an *N*-bit carry-skip adder as follows:



We design the *k*-bit adder blocks using *k*-bit conditional sum adders where k is $2m^i$ (i is an integer greater than or equal to 0). We also design the group-propagation signal $p_{i:j}$ using OR gates hierarchically. The following shows the delays of the components:

- *k*-bit adder: $(1 + \ln k) \cdot \Delta_M$
- *k*-bit group-propagation signal $p_{i:i-k+1}$: $(2 + \ln \frac{k}{4}) \cdot \Delta_M$
- 2-bit MUX: Δ_M

Find ***m*** minimizing the total delay of an *N*-bit carry-skip adder designed using the new architecture shown above.

Problem #7 (Prefix Adder, 40 points)

Use the following delay values:

- AND, OR, XOR: Δ
- Two-level (sum-of-product) logic: 2Δ
- $g_i = a_i \cdot b_i$, $p_i = a_i \oplus b_i$

We are designing a 1024-bit Kogge-Stone adder.

1) Represent s_{999} hierarchically using group-generated and group-propagated carries ($g_{i:k}$, $p_{i:k}$) and c_0 (primary carry-in), then compute the delay to compute s_{999} assuming all the primary input signals are available at time 0 (10 points).

2) Represent s_{768} hierarchically using group-generated and group-propagated carries ($g_{i:k}$, $p_{i:k}$) and c_0 (primary carry-in), then compute the delay to compute s_{768} assuming all the primary input signals are available at time 0 (10 points).

3) Calculate the total gate area to build the 1024-bit Kogge-Stone adder. Use the following area values. Notice that you should generate all sum ($s_{1023:0}$) and carry-out (c_{1024}) signals (10 points).

- Two-input AND, OR gate: k
- Two-input XOR: $4k$
- $g_i = a_i \cdot b_i$, $p_i = a_i \oplus b_i$

4) When we designed a Kogge-Stone adder, we used an XOR gate to calculate $p_i = a_i \oplus b_i$. Prove that we can also use $a_i + b_i$ (+ is an OR operation) for p_i, i.e., prove that replacing $p_i = a_i \oplus b_i$ by $p_i = a_i + b_i$ does not change the final sum and carry-out values. (10 points).

Problem #8 (Carry Look-Ahead Adder, 50 points)

The max. fanout is 4. Use the following delay values:

- AND, OR, XOR: Δ
- Two-level (sum-of-product) logic: 2Δ
- $g_i = a_i \cdot b_i$, $p_i = a_i \oplus b_i$

We are designing a 1024-bit carry look-ahead adder.

1) Represent s_{999} hierarchically using group-generated and group-propagated carries ($g_{i:k}$, $p_{i:k}$) and c_0 (primary carry-in), then compute the delay to compute s_{999} assuming all the primary input signals are available at time 0 (10 points).

2) Represent s_{768} hierarchically using group-generated and group-propagated carries ($g_{i:k}$, $p_{i:k}$) and c_0 (primary carry-in), then compute the delay to compute s_{768} assuming all the primary input signals are available at time 0 (10 points).

3) Calculate the total gate area to build the 1024-bit carry look-ahead adder. Use the following area values. Notice that you should generate all sum ($s_{1023:0}$) and carry-out (c_{1024}) signals (20 points).

- Two-input AND, OR gate: k
- Three-input AND, OR gate: $2k$
- Four-input AND, OR gate: $3k$
- Two-input XOR: $4k$
- $g_i = a_i \cdot b_i$, $p_i = a_i \oplus b_i$
- For a carry look-ahead unit for $a_{i+3:i}, b_{i+3:i}, c_i$, use the following formulae:
  - $c_{i+1} = g_i + p_i \cdot c_i$ (for this, you need a two-input OR and a two-input AND)
  - $c_{i+2} = g_{i+1} + p_{i+1} \cdot g_i + p_{i+1} \cdot p_i \cdot c_i$ (for this, you need a three-input AND, a two-input AND, a three-input OR)
  - $c_{i+3} = g_{i+2} + p_{i+2} \cdot g_{i+1} + p_{i+2} \cdot p_{i+1} \cdot g_i + p_{i+2} \cdot p_{i+1} \cdot p_i \cdot c_i$ (for this, you need a four-input AND, three-input AND, a two-input AND, a four-input OR)
- For a group carry look-ahead unit for $g_{i+3:i}, p_{i+3:i}, c_i$, use the following formulae:
  - g' (group generation) $= g_{i+3} + p_{i+3} \cdot g_{i+2} + p_{i+3} \cdot p_{i+2} \cdot g_{i+1} + p_{i+3} \cdot p_{i+2} \cdot p_{i+1} \cdot g_i$ (for this, you need a two-input AND, a three-input AND, a four-input AND, a four-input OR)
  - p' (group propagation) $= p_{i+3} \cdot p_{i+2} \cdot p_{i+1} \cdot p_i$ (for this, you need a four-input AND)
  - c' (group carry) $= g' + p' \cdot c_i$ (for this, you need a two-input AND and a two-input OR)

The max. fanout is 2. Use the following delay values:

- AND, OR, XOR: Δ
- Two-level (sum-of-product) logic: 2Δ
- $g_i = a_i \cdot b_i$, $p_i = a_i \oplus b_i$

We are designing a 1024-bit carry look-ahead adder.

4) Represent s_{999} hierarchically using group-generated and group-propagated carries ($g_{i:k}$, $p_{i:k}$) and c_0 (primary carry-in), then compute the delay to compute s_{999} assuming all the primary input signals are available at time 0 (10 points).