EECS 151/251A Homework 10

Due Monday, May 1, 2023

Problem 1: Vector Addition

Imagine you are asked to implement a vector addition module using single-stage (non-pipelined) adders that works at 100 MHz and consumes 10 µW (only due to switching power) at a nominal V_{dd}. Given a switching power constraint of 15 µW and an area constraint of 4 adders, what is the maximum possible throughput (number of elements output per second)? You are allowed to change V_{dd} arbitrarily. Assume that the frequency scales linearly with V_{dd} and that delay and power of other components are negligible.

Solution:

By using 4 adders at the nominal V_{dd}, we get a throughput of 400 MHz with 40 µW. Since the power is proportional to the cube of V_{dd}, we have to decrease V_{dd} by a factor of $\sqrt[3]{15/40} = \sqrt[3]{3}/2 \approx 0.72$ to meet the power constraints. Therefore, with 15 µW, we get a throughput of $400 \times 0.72 = 288$ MHz.

Problem 2: Power and Energy Efficiency

The following table shows the effect of optimization on the total capacitance C, supply voltage V_{dd}, clock frequency f, power P, time per task T, and energy per task E. Consider only the switching power.

Optimization	C	V_{dd}	f	P	T	E
Logic optimization	$1/k$	1	t			
Lower V_{dd}	1	$1/k$	$1/k$			
Denard scaling	$1/k$	$1/k$	k			
Pipelining	1	1	k	k	$1/k$	1
Multi-core	k	1	1	k	$1/k$	1

(a) Fill out the blank cells in the table.

(b) The effect of pipelining in the table is not accurate. What is wrong?

(c) The effect of multi-core in the table may not be accurate. Explain why.

(d) Clock gating is another possible optimization. Explain how it helps.

(e) Power gating turns off a part of chip to cut leakage power as well as switching power. However, it involves some overhead for activation. Explain an alternative technique to reduce leakage power without such an overhead.

Solution:

(a)

Optimization	C	V_{dd}	f	P	T	E
Logic optimization	$1/k$	1	t	t/k	$1/t$	$1/k$
Lower V_{dd}	1	$1/k$	$1/k$	$1/k^3$	k	$1/k^2$
Denard scaling	$1/k$	$1/k$	k	$1/k^2$	$1/k$	$1/k^3$

(b) It requires pipeline registers, so C should be larger. (It might also cause hazards increasing T.)
(c) If there is dependency among tasks, we cannot fully parallelize them. So, T may be larger. Also, C may be larger due to task management hardware.
(d) It reduces the activity factor.
(e) Using larger V_{th} in non-critical paths reduces leakage current.

Problem 3: 36-bit Carry Select Adder

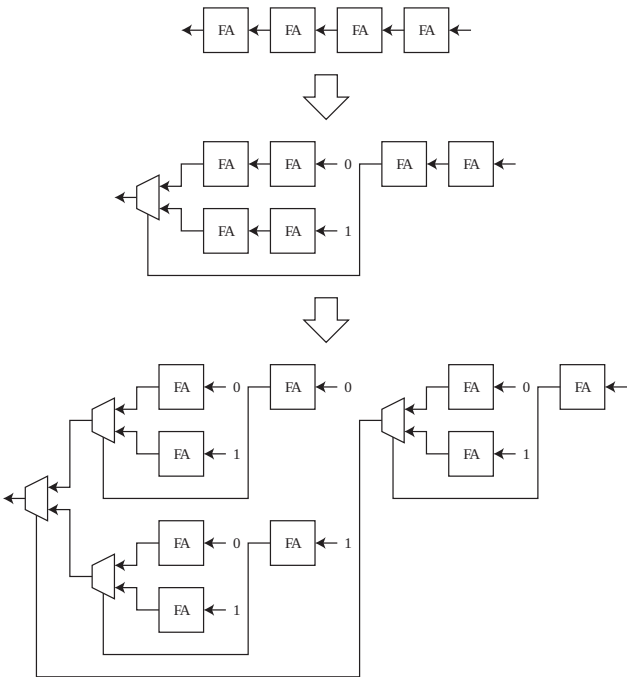
For a 36-bit carry selection adder, we need 6 stages of 6-bit adders in the square root method. Assuming an FA and a multiplexer take the same delay, can you minimize the delay furthermore? Explain how.

Solution:

Using 5 stages of 6-, 6-, 7-, 8-, and 9-bit adders, we get 10 unit delay for the maximum delay. This is lower than the one by the square root method, which has at most 11 unit delay.

Problem 4: Hierarchcal Carry Select Adder

The key concept of carry select adders is duplicate adders with carry-in assigned to 0 and 1, except for the first stage, and create multiplexers to select one of them. Consider the case when we apply this idea recursively with radix 2. The following figure shows an example for a 4-bit adder, where we reach the terminal case in 2 steps. For simplicity, only the multiplexers for the carry-outs are shown. It contains 1 FA with variable carry-in, 8 FAs with fixed carry-in (although 2 FAs in the last level can be shared), 4 multiplexers for the carry-outs, and 5 multiplexers (not shown) for the sums. Assuming FAs and multiplexers have the same delay, this 4-bit carry select adder has 3 unit delay for the maximum delay. What is the maximum delay and the number of FAs and multiplexers for an 8-bit carry select adder constructed in this way? Do not consider logic sharing between duplicated adders.

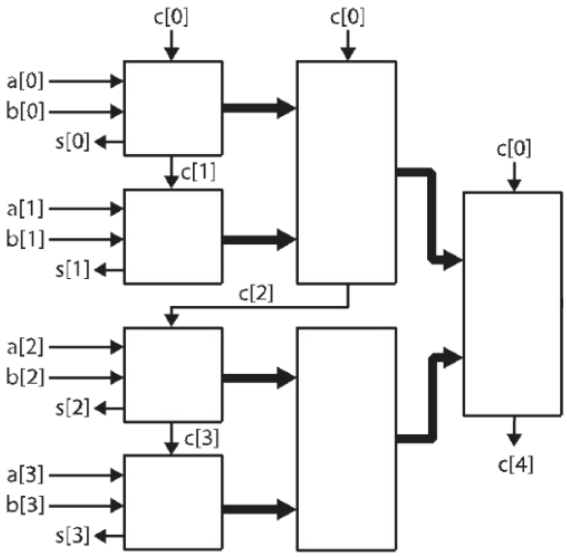


Solution:

Since we need one more level of multiplexers for the upper half bits, the maximum delay is the delay of 4 units. We need three times more FAs, so the number of FAs is 27. For multiplexers, we need one more level for each of the upper half bits and the last carry-out, on top of three times as many as the 4-bit adder, we need 32 in total (13 for the carry-outs and 19 for the sums).

Problem 5: Carry Look-ahead Adder

The figure below shows the structure of 4-bit carry look-ahead adder. Assuming each box takes the same delay uniformly from each input to dependent output, what is the maximum delay? Can you generalize it for N-bit carry look-ahead adder where N is power of 2 larger than 4?



Solution:

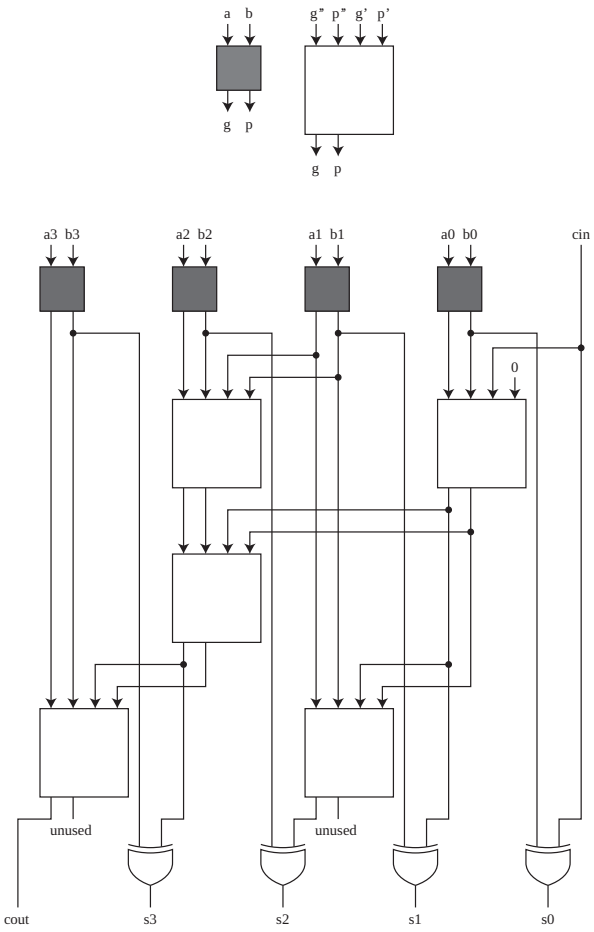
After simple analysis, $s[3]$ turns out to have the largest delay of 4 units coming through $c[2]$ from either top two boxes on the left hand side. Note that the propagate and generate from the second box does not depend on $c[1]$.

For 8-bit adder, we copy the same structure for the upper half bits and assign $c[4]$ to the place of $c[0]$, while one more box is added to the right end for $c[8]$. Since the path through $c[4]$ has larger delay than the paths coming from $a[4]$ or $a[5]$ to $s[7]$, the maximum delay is $3 + 4 - 1 = 6$. By induction, the path coming through $c[N/2], c[N/2 + N/4], \ldots$ to the box for the most significant bit is the critical path, which has 2 unit larger delay than the previoius power of 2. Therefore, the maximum delay for N-bit carry look-ahead adder is $2 \log_2 N$.

Problem 6: Parallel Prefix Adder

Draw a block diagram for a 4-bit Brent-Kung adder with carry-in. It takes 9 inputs $a0, \ldots, a3$, $b0, \ldots, b3$, and cin, and generates 5 outputs $s0, \ldots, s3$, and $cout$. You are allowed to define the building blocks such as partial full adder as you want.

Solution:



Gray box: $g = ab, p = a \oplus b$. Empty box: $g = g'' + g'p'', p = p'p''$.