EECS 151/251 A Discussion 13

April 19, 2024

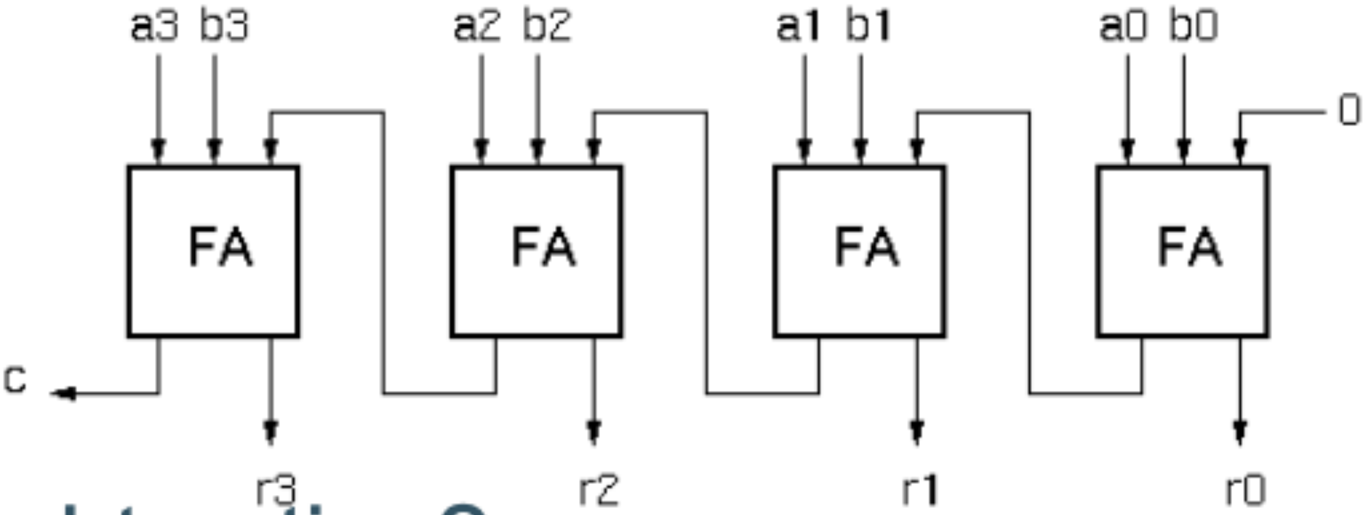
Berkeley
UNIVERSITY OF CALIFORNIA

Content

- Last content for the semester!
- Adders
- Multipliers
- Shifters

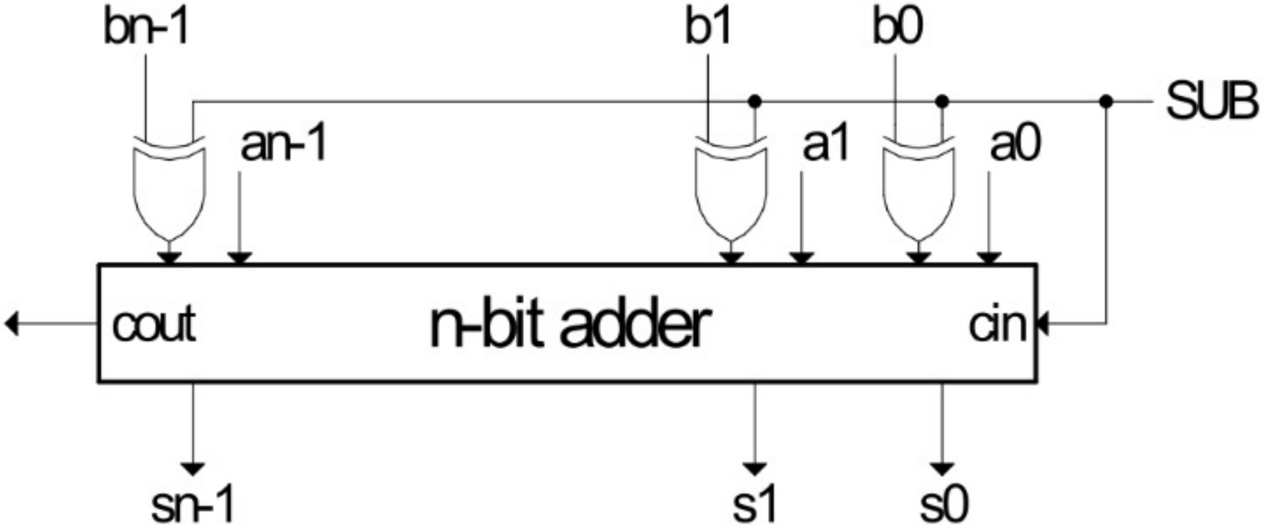
Adders

- From before: 1-bit Full Adder takes 3 bits (a,b,carry in) and outputs 2 bits (carry out and sum)
- Simplest n-bit adder = Carry-Ripple



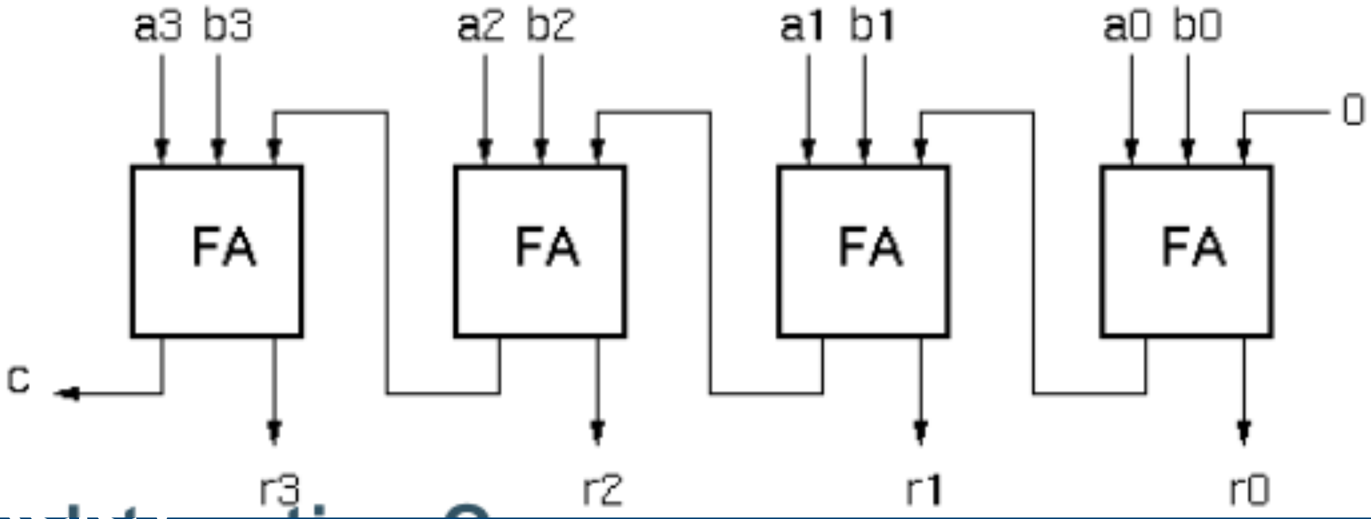
Subtractors

- Complement B and add 1 (as carry in) (2s complement)
- Simplest n-bit adder = Carry-Ripple
- XOR with 1 does complement, XOR with 0 does nothing



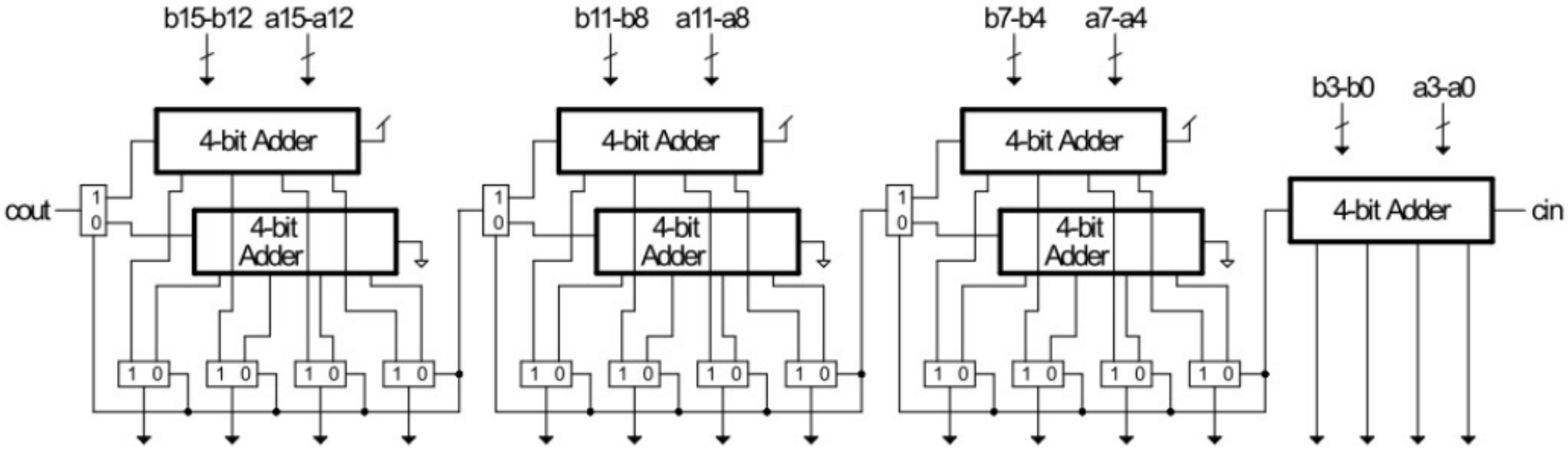
Adder Delay

- From before: 1-bit Full Adder takes 3 bits (a,b,carry in) and outputs 2 bits (carry out and sum)
- Simplest n-bit adder = Carry-Ripple
- Cost and Delay are proportional to the number of bits



Carry Select

- With more HW, calculate the sum if the carry in is 0 and if it is 1 in **parallel**
- Then, when the carry is computed, use it to the select the right output
- Can chain them, optimal is to do $\sqrt{N}$ stages with $\sqrt{N}$ bit adders (changes a little to match delays)
- What is the total cost proportional to? What is the delay proportional to?



Parallel-Prefix Carry Look-ahead

- Reformulate adder to output a propagate and generate signal
- Normal ripple with p and g signals offers no benefit
- We can group the p and g signals into P and G
- Then we generate hierarchically
- Runs in O(logn)

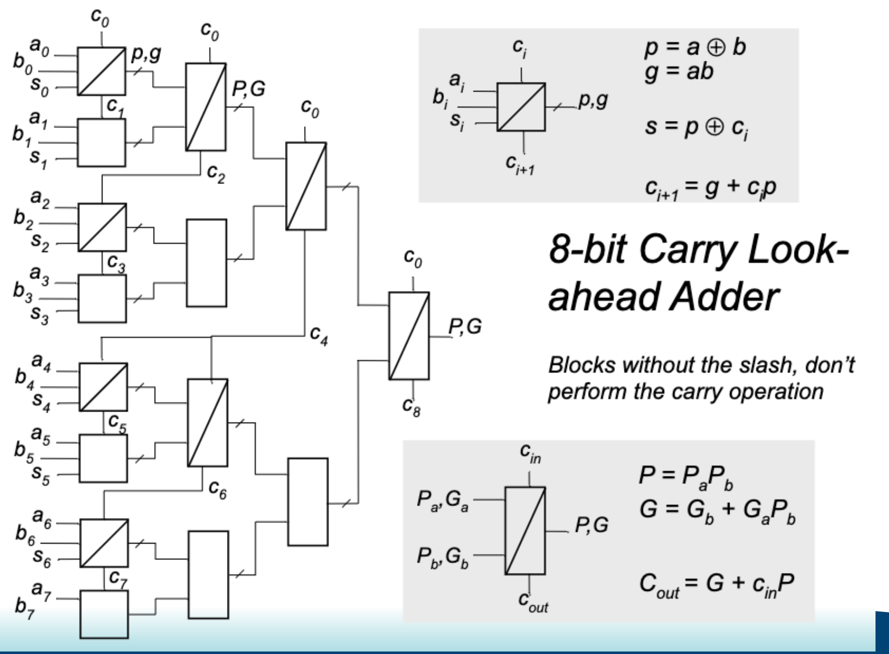
a b c_i	c_{i+1}	s
000	0	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	1	1

carry "kill"
$k_i = a_i' b_i'$

carry "propagate"
$p_i = a_i \oplus b_i$

carry "generate"
$g_i = a_i b_i$

$$\boxed{\begin{aligned} c_{i+1} &= g_i + p_i c_i \\ s_i &= p_i \oplus c_i \end{aligned}}$$



$p = a \oplus b$
$g = ab$

$s = p \oplus c_i$

$c_{i+1} = g + c_i p$

8-bit Carry Look-ahead Adder

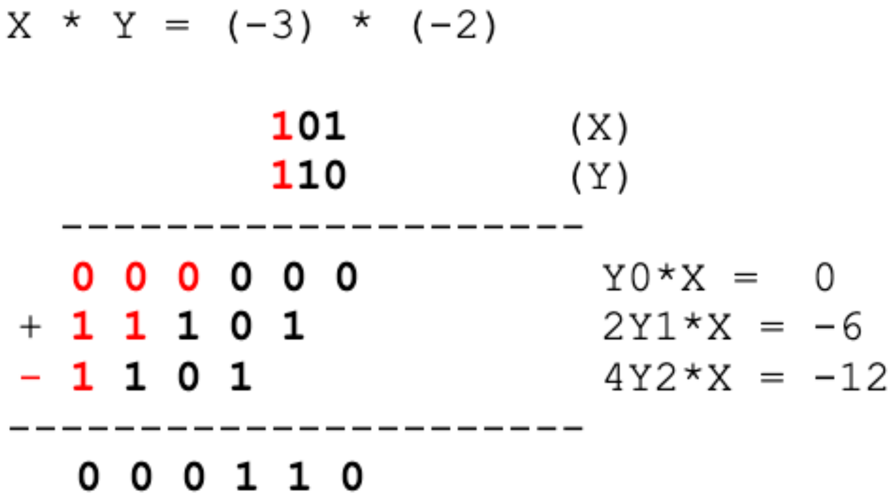
Blocks without the slash, don't perform the carry operation

$P = P_a P_b$
$G = G_b + G_a P_b$

$C_{out} = G + c_{in} P$

Multiplication

- Multiplication is a series of shifts and adds
- Negative numbers
  - For multiplier (second operand), you must subtract the final partial product
  - For multiplicand (first operand), you sign-extend partial products

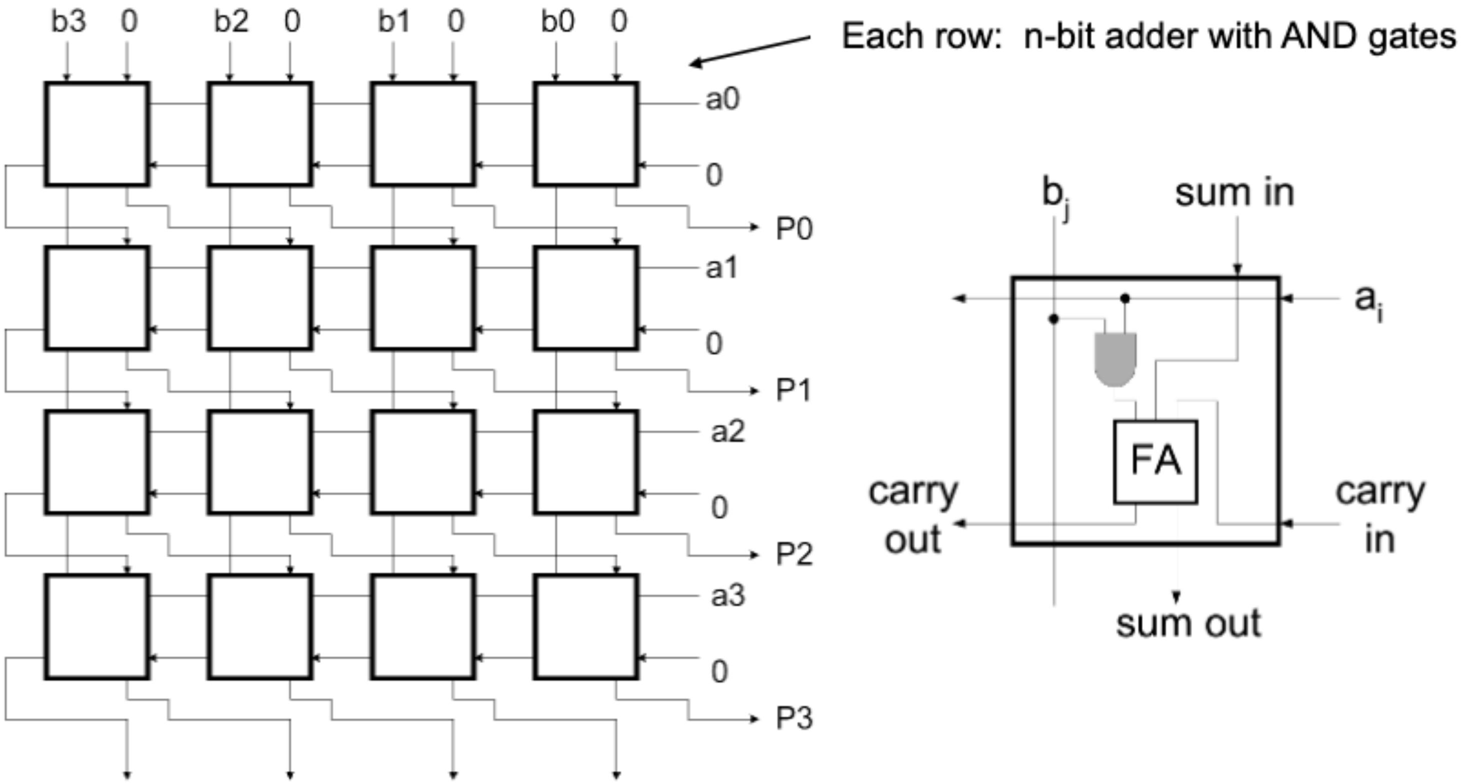


```
X * Y = (-3) * (-2)

          101          (X)
          110          (Y)
   --------------------
   0 0 0 0 0 0          Y0*X  =   0
 + 1 1 1 0 1            2Y1*X =  -6
 - 1 1 0 1              4Y2*X = -12
----------------------
   0 0 0 1 1 0
```

Array Multiplier

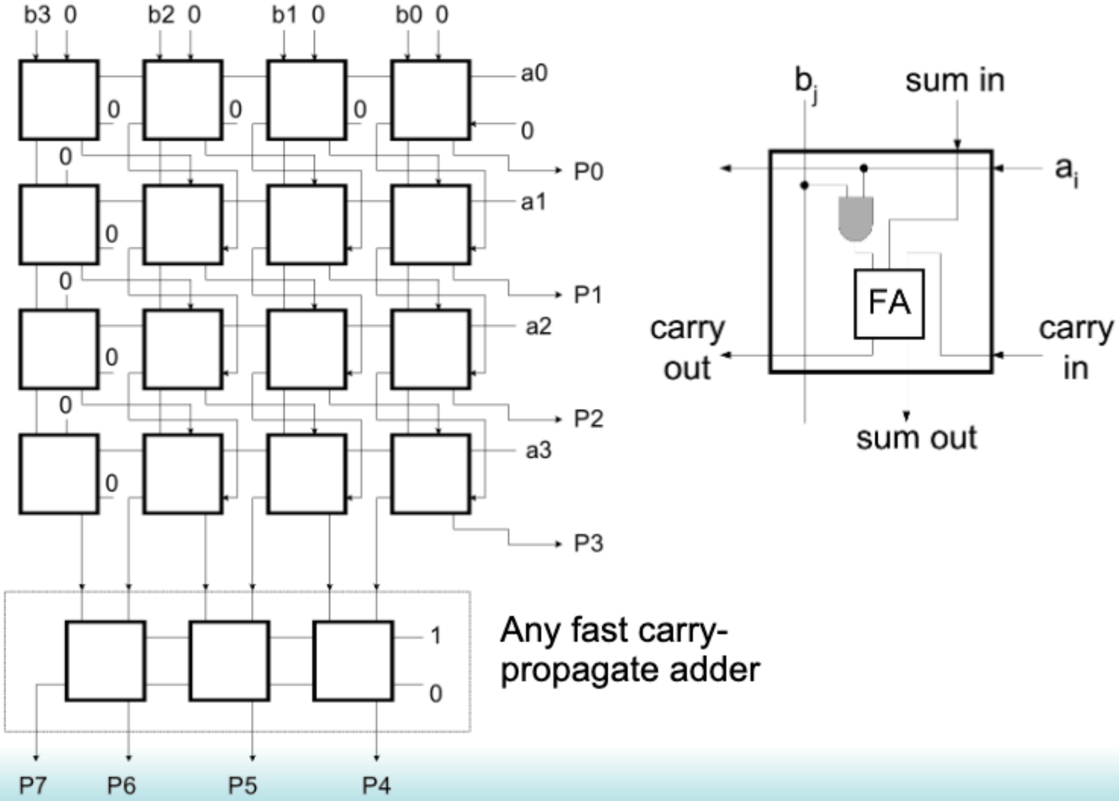
- Single cycle multiply
- Generate all n partial products simultaneously



Each row: n-bit adder with AND gates

Carry Save Addition

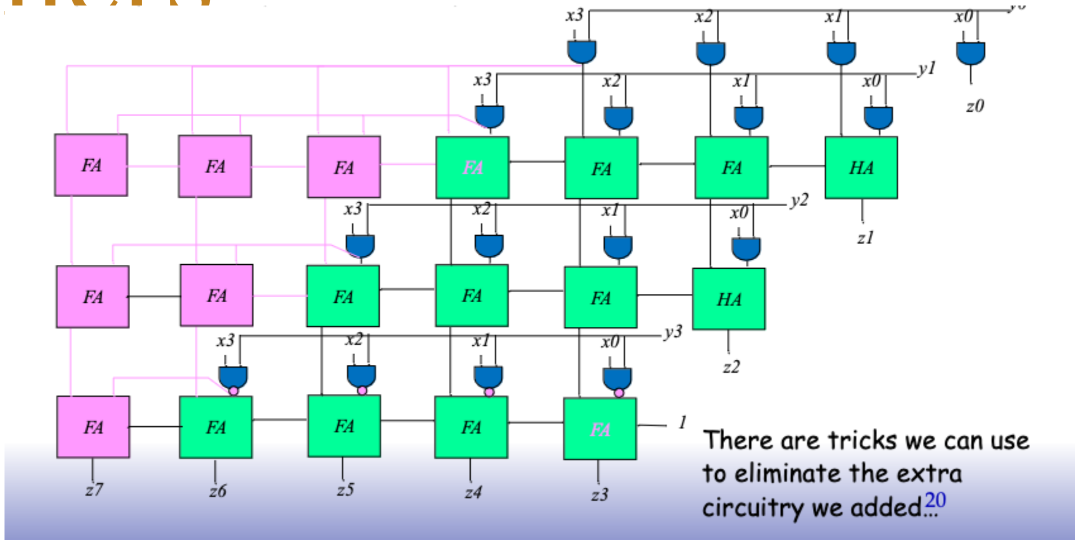
- Speed up summation of partial products
- Saves the carries of of each addition for the last step (then does carry-propagate to get the final sum)
- Takes 3 numbers and outputs 2



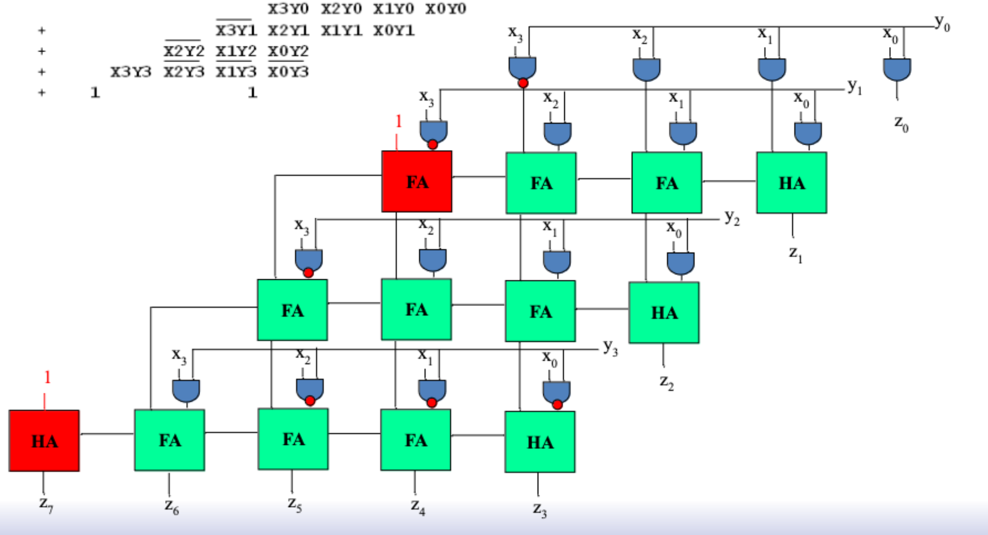
Any fast carry-propagate adder

Signed Multipliers

- In theory, need way more HW because of sign extension
- Baugh-Wooley does some fancy stuff, only the same number of adders!



There are tricks we can use to eliminate the extra circuitry we added…20

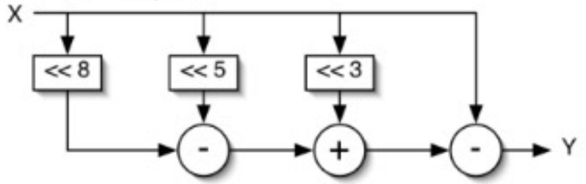


$$
\begin{array}{rcccccc}
 & & & & \overline{X3Y0} & X2Y0 & X1Y0 & X0Y0 \\
+ & & & \overline{X3Y1} & X2Y1 & X1Y1 & X0Y1 & \\
+ & & \overline{X2Y2} & X1Y2 & X0Y2 & & & \\
+ & X3Y3 & \overline{X2Y3} & \overline{X1Y3} & \overline{X0Y3} & & & \\
+ \;\; 1 & & & 1 & & & &
\end{array}
$$

Constant Multipliers

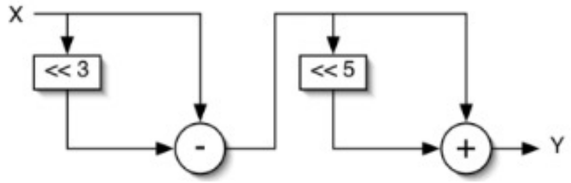
- Important in applications like DSP
- Can hardcode series of shifts and adds
- Instead of just doing adds, we can also do subtracts (CSD) to do at most n/2 adds/subtracts
- KCM Multiplier (Break down into factors and then do CSD) – no simpl eway to figure out optimal KCM

$Y = 231*X = (2^8 - 2^5 + 2^3 - 2^0)*X$



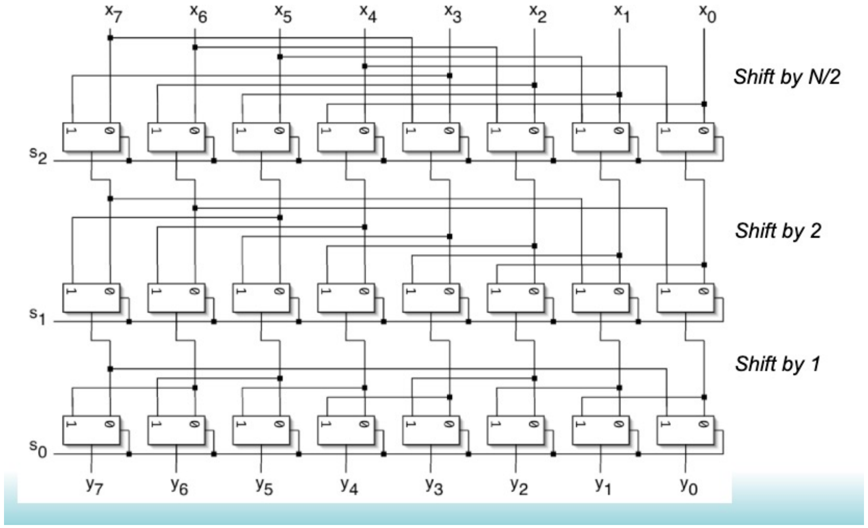
/shift/add ...?

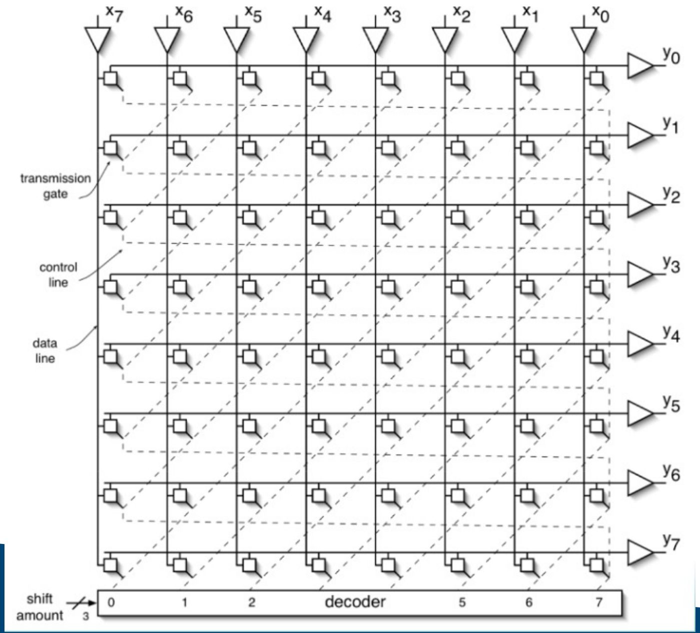
$Y = 231*X = 7*33*X = (2^3 - 2^0)*(2^5 + 2^0)*X$



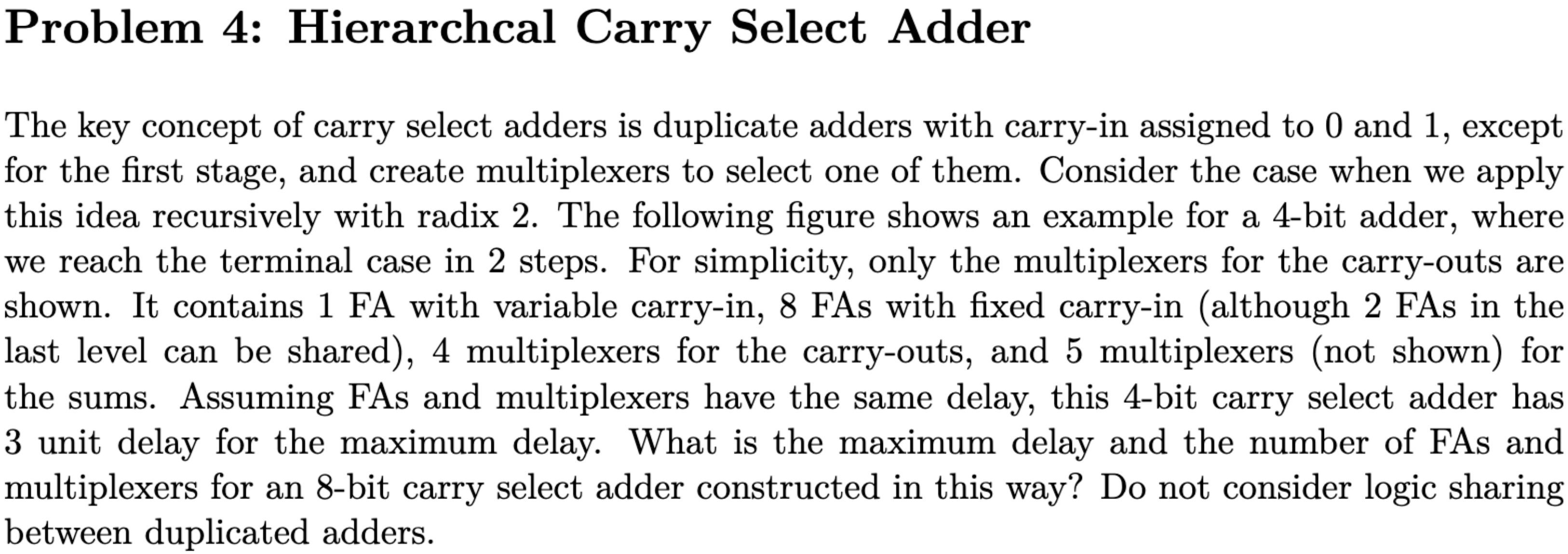
Shifters

- Log Shifter
  - Shift by decreasing powers of 2, with muxes
- Barrel Shifter
  - Cross bar, samt selects a diagonal to transmit data





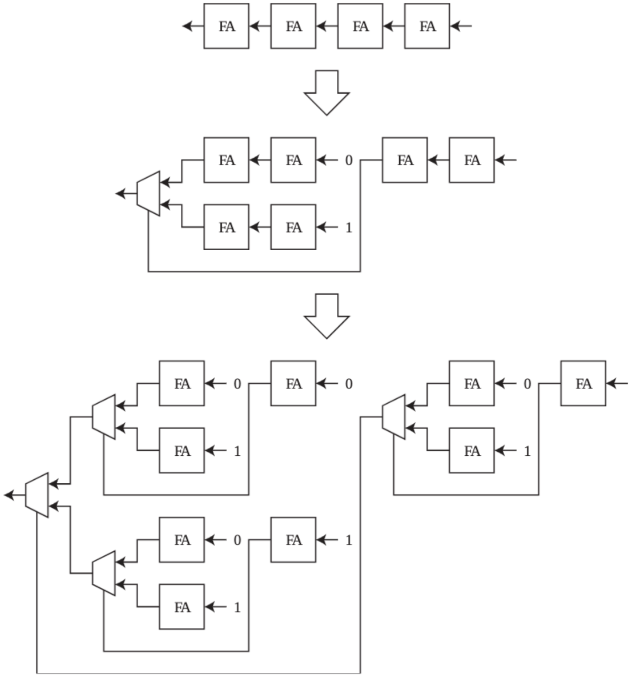
Problem 1

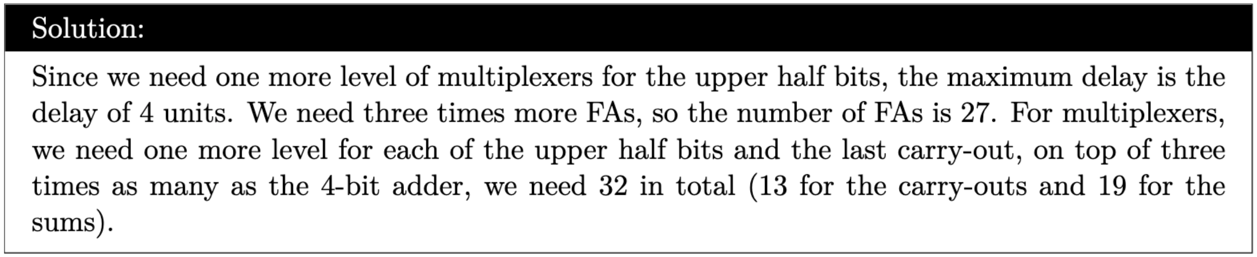


Problem 4: Hierarchcal Carry Select Adder

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

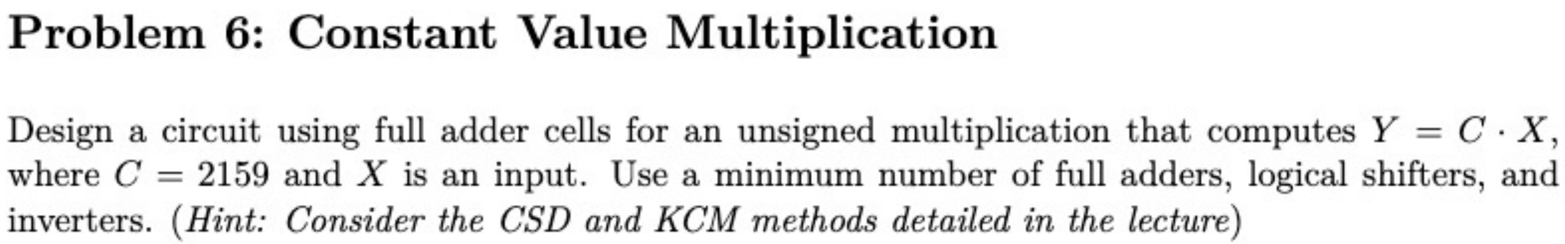




Solution:

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).

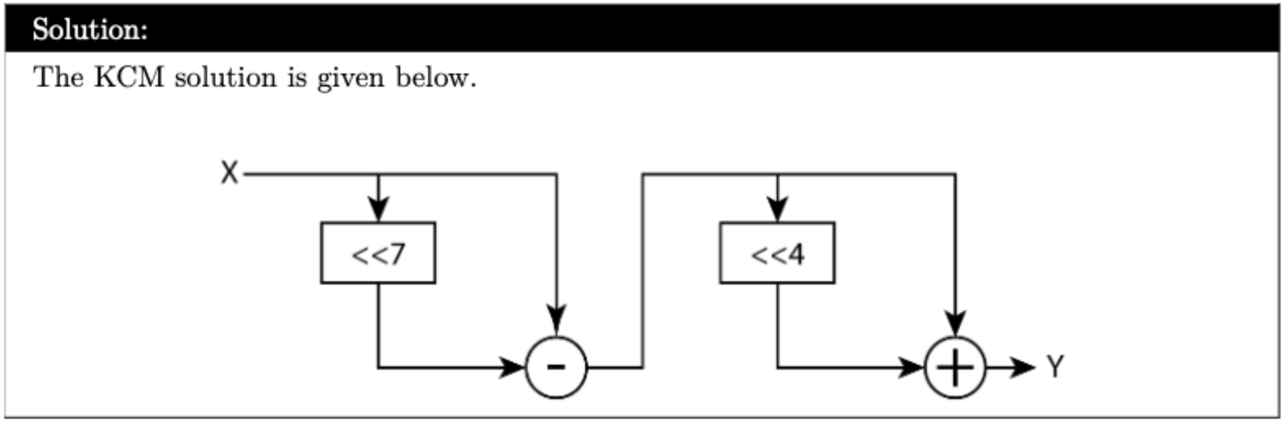
Problem 2



Problem 6: Constant Value Multiplication

Design a circuit using full adder cells for an unsigned multiplication that computes $Y = C \cdot X$, where $C = 2159$ and X is an input. Use a minimum number of full adders, logical shifters, and inverters. (*Hint: Consider the CSD and KCM methods detailed in the lecture*)

Solution



Solution:

The KCM solution is given below.