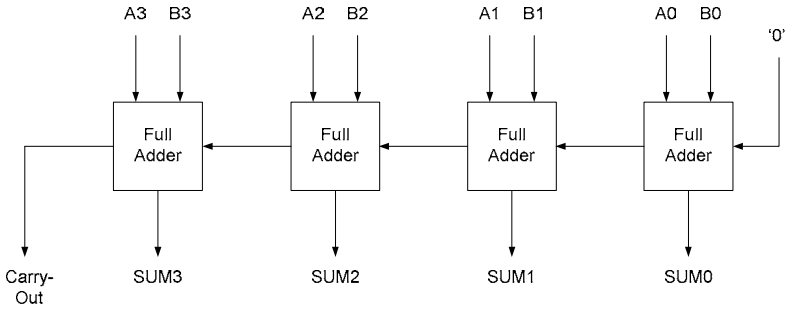
Carry-Propagate Adder

Connecting full-adders to make a multi-bit carry-propagate adder:



Right-most adder adds least-significant bits. Carry-out is passed to next adder, which adds it to the next-most significant bits, etc.

Can extend this to any number of bits

1

Carry-LookAhead Adders

By pre-computing the major part of each carry equation, we can make a much faster adder.

We start by computing the partial results (for each bit):

$p_i = a_i + b_i$ (called "propagate term")
$g_i = a_i \; b_i$ (called the "generate term")

Then:

$c_{i+1} = g_i + p_i \, c_i$ (carry_out from bit i)

We can compute these for each bit independently (no ripple)
Each of the carries can now be expressed in terms of p and g:

$c_{i+2} = g_{i+1} + p_{i+1} \, c_{i+1}$ (carry_from bit i+1)
$c_{i+3} = g_{i+2} + p_{i+2} \, c_{i+2}$ (carry_from bit i+2)
$c_{i+4} = g_{i+3} + p_{i+3} \, c_{i+3}$ (carry_from bit i+3)

4

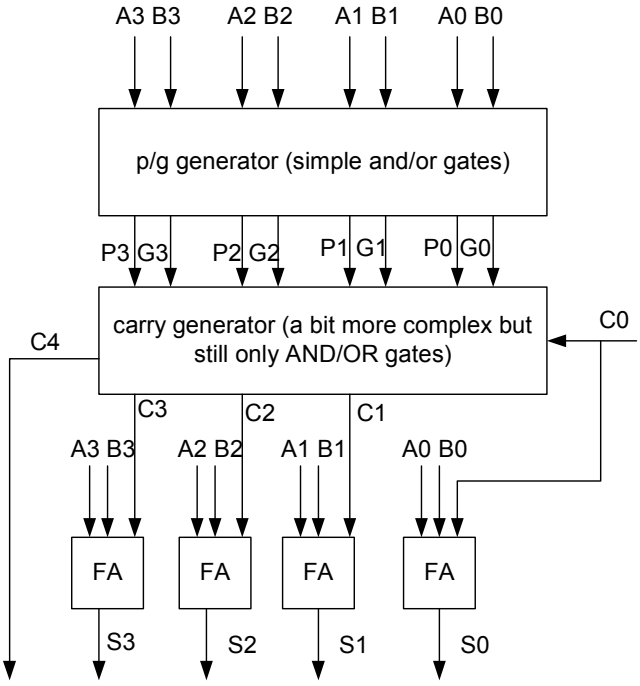
Carry-LookAhead Adders

Now, by forward substitution (show this and check for errors!)

$$c_{i+1} = g_i + p_i c_i$$
$$c_{i+2} = g_{i+1} + p_{i+1} g_i + p_{i+1} p_i c_i$$
$$c_{i+3} = g_{i+2} + p_{i+2} g_{i+1} + p_{i+2} p_{i+1} g_i + p_{i+2} p_{i+1} p_i c_i$$
$$c_{i+4} = g_{i+3} + p_{i+3} g_{i+2} + p_{i+3} p_{i+2} g_{i+1} + p_{i+3} p_{i+2} p_{i+1} g_i + p_{i+3} p_{i+2} p_{i+1} p_i c_i$$

So we can express each carry as a function of generate and propagate signals; in a two level AND-OR circuit and without any ripple effect!

5



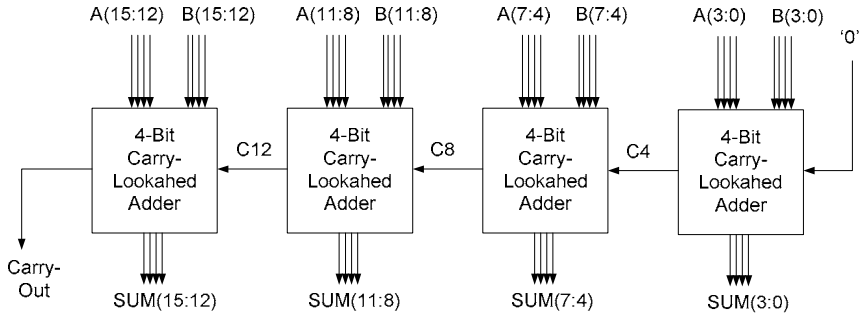
8

Hierarchical Carry-LookAhead Adders

Theoretically, we could create a carry-lookahead adder for any N

But these equations are complex. It is unreasonable to extend this to beyond more than 4 bits or so. Why?

Can combine carry look-ahead and carry-propagate schemes:

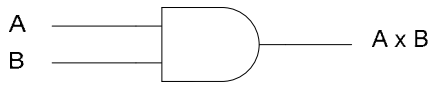


9

Multipliers

Consider a 1 bit x 1 bit multiplier:

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1



Multiplication of two bits is simply an AND gate

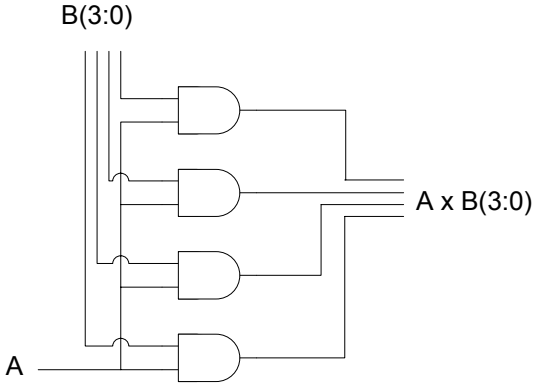
10

Multipliers

One bit number x N-bit number

eg. 10111011 x 1 = 10111011

10111011 x 0 = 00000000



11

Multipliers

N bit x N bit number (consider 4x4):

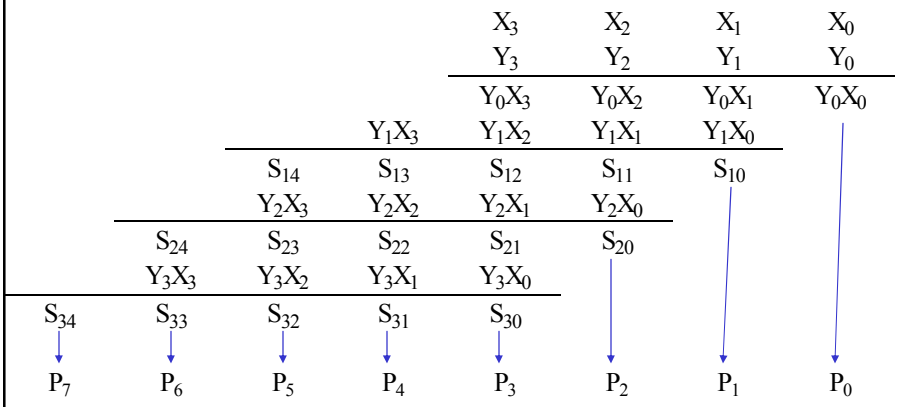
				X_3	X_2	X_1	X_0
				Y_3	Y_2	Y_1	Y_0
				Y_0X_3	Y_0X_2	Y_0X_1	Y_0X_0
			Y_1X_3	Y_1X_2	Y_1X_1	Y_1X_0	
		Y_2X_3	Y_2X_2	Y_2X_1	Y_2X_0		
	Y_3X_3	Y_3X_2	Y_3X_1	Y_3X_0			
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

This would require 16 AND gates and a 8 bit, 4 input adder, and lots of wiring (wiring is becoming a big problem in integrated circuits).

12

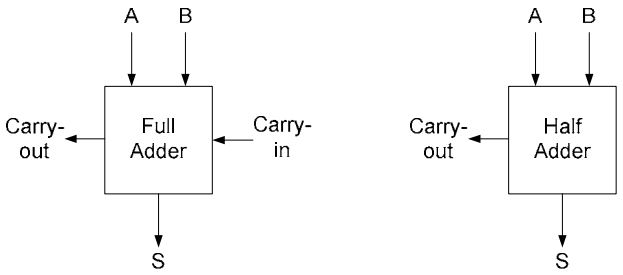
Multipliers

We can reduce the wiring by distributing the adder:



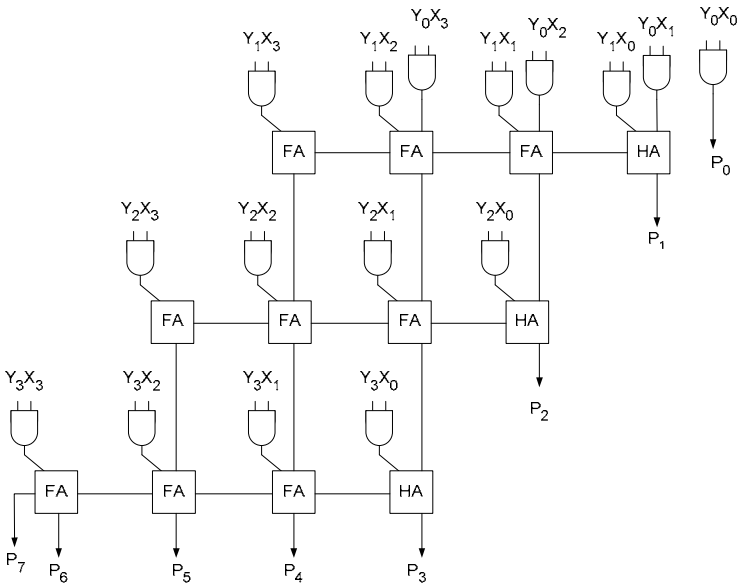
Multipliers

Recall: Half-Adder vs. Full-Adder:



By combining half-adders, full-adders, and AND gates, we can implement our array multiplier

Multipliers



15