VE270 Lecture 12 Arithmetic Components

Carry Look-ahead Adder

Check the Carry-Ripple Adder, it is created with full adders.

For 4-bit adder, we add two 4-bit numbers and generate a 5-bit number (4-bit sum and 1-bit carry).

Faster Adder

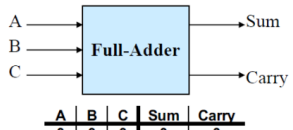
Use two-level combinational logic design process.

- pro: fast

  2 gate level delays

- con: large
  - 9 inputs and 5 outputs.
  - Truth table would have $2^{4+4+1} = 512$ rows

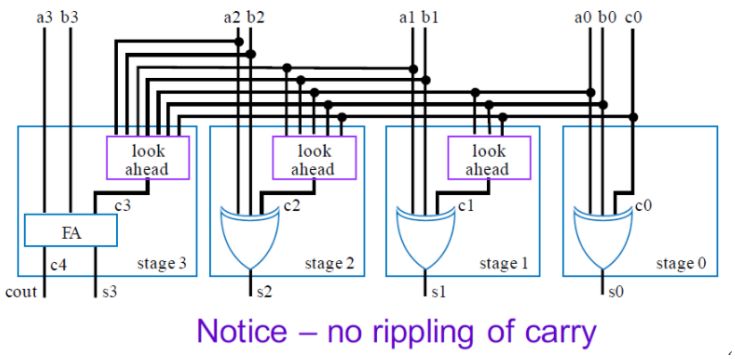
Full Adder



A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

sum is $\sum m(1,2,4,7) = A \oplus B \oplus C$, carry is $\sum m(3,5,6,7) = AB + AC + BC = (A \oplus B)C + AB$

Faster Adder - Intuitive Attempt at "Look ahead"



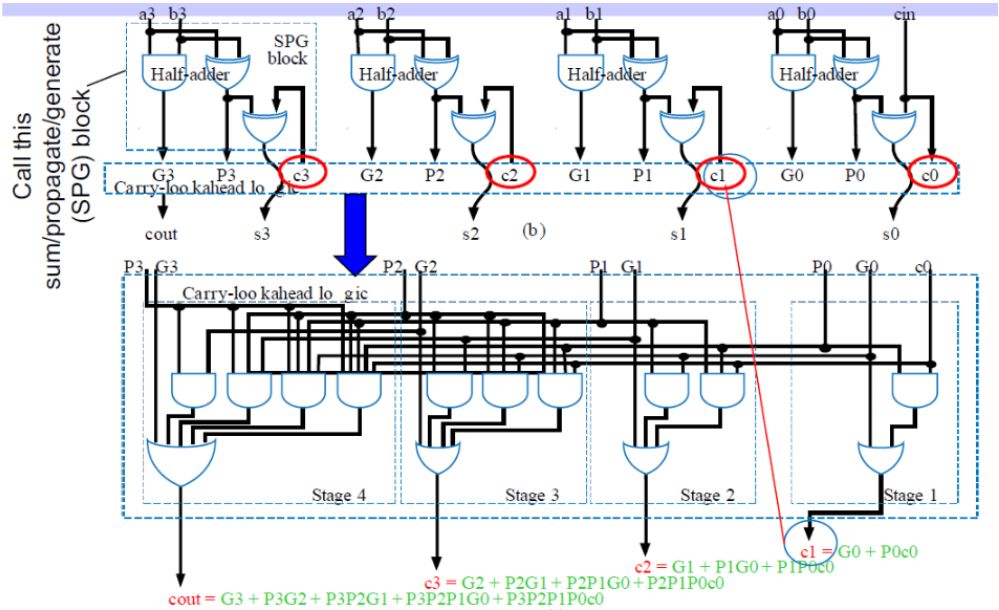
Then we produce carries directly and recursively: $c_{n+1} = a_n b_n + a_n c_n + b_n c_n$

Two layer SOP logic.

Better Form of Look Ahead

Since we get the $\textbf{Carry} = ab + (a \oplus b)c$ then we define $\textbf{Propagate}\ P = a \oplus b$ and $\textbf{Generate}\ G = ab$.

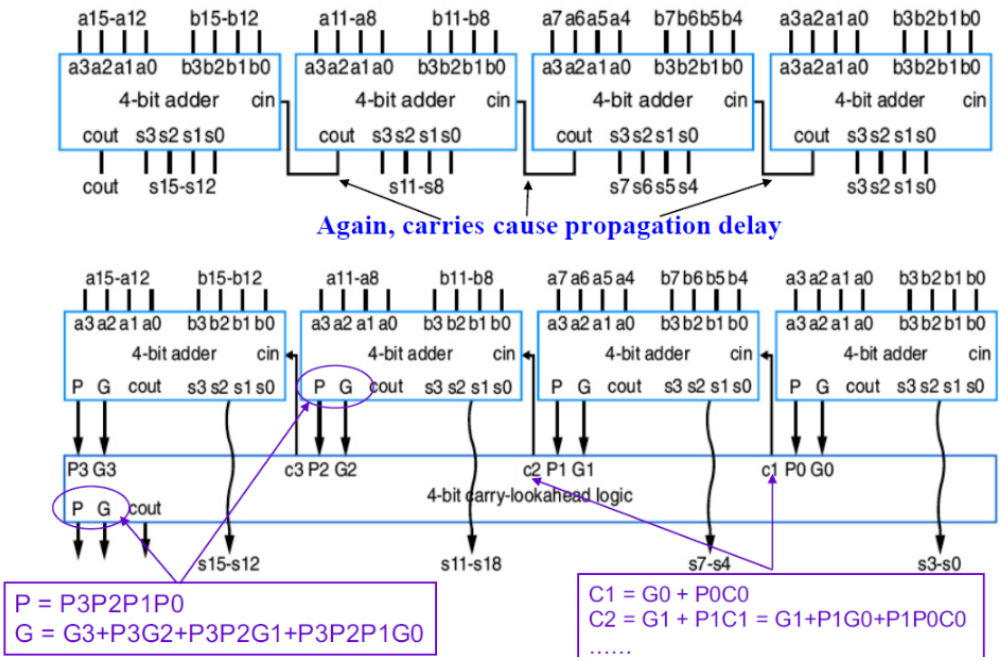
Then $\textbf{Cout} = G + Pc$, $c_{n+1} = G_n + P_n c_n$



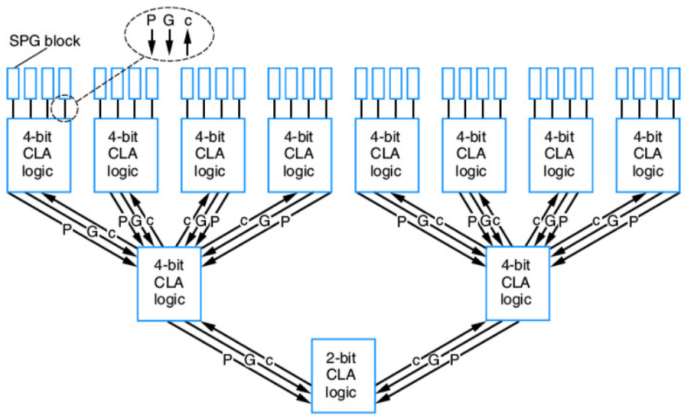
So only 4 gate level delay, 2 from SPG block and 2 from look ahead block.

Cascading Adder

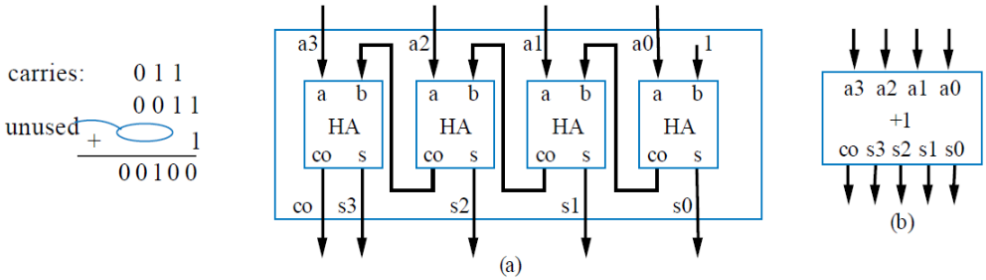
According to the $c = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0c_0$, we derive a P and G for cascading adder



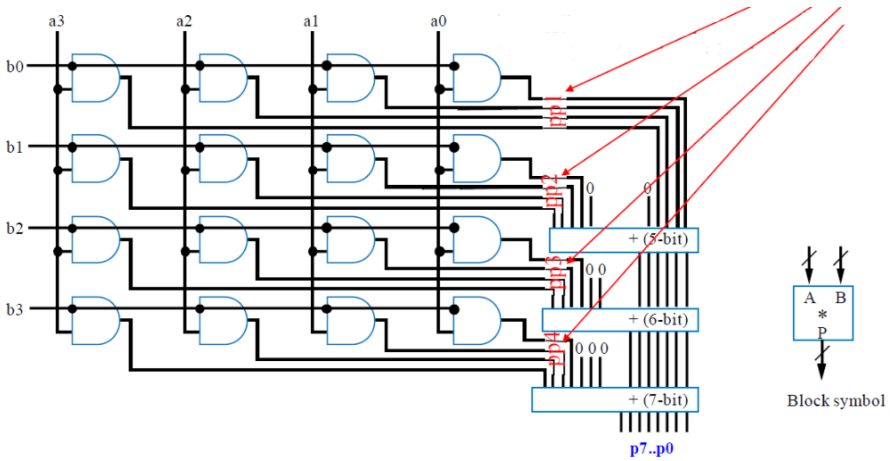
CLA Multi-Level Adder



Incrementer

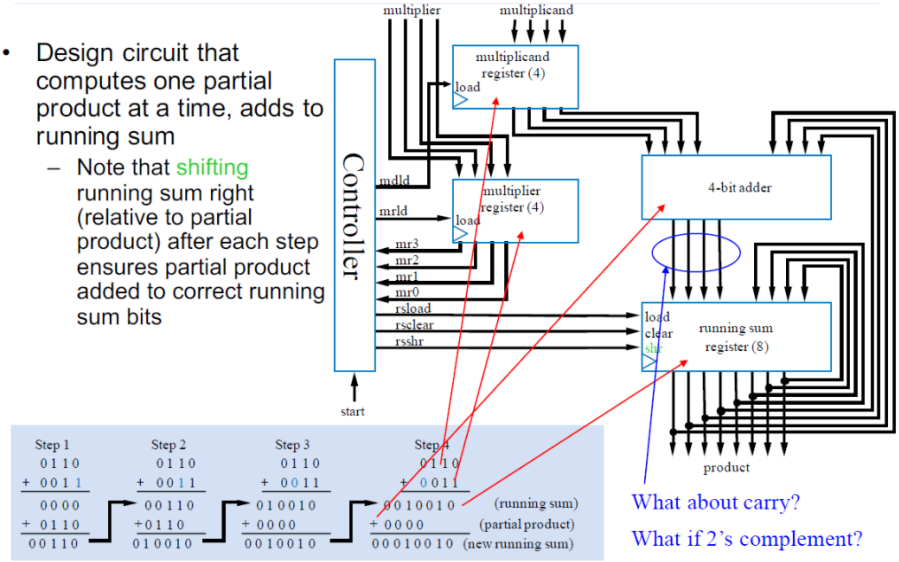


Multiplier - Array Style



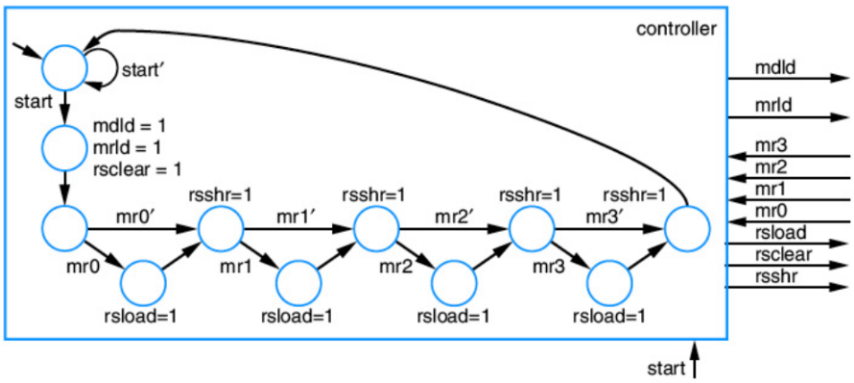
Multiplier - Sequential (Add and Shift) Style

- Design circuit that computes one partial product at a time, adds to running sum
  - Note that shifting running sum right (relative to partial product) after each step ensures partial product added to correct running sum bits



What about carry?

What if 2's complement?

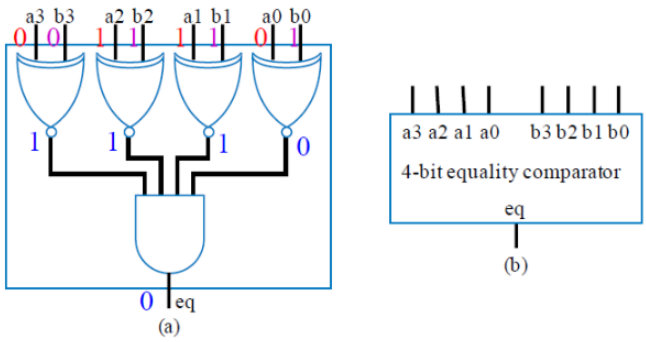


ALU

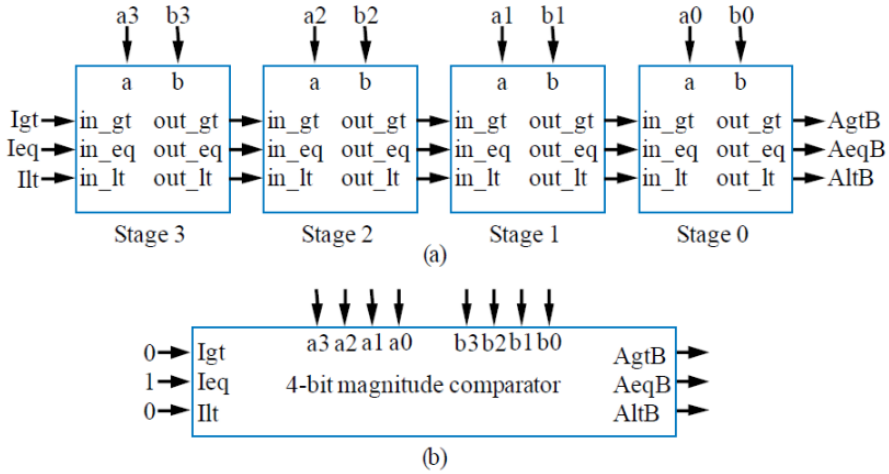
Refer the Lecture 5, we can see an example for the ALU operations.

Problem, the mux will generate too much wires, wasting too much power.

Comparators



Magnitude Comparator



For each stage, we have:

- $\text{out\_gt} = \text{in\_gt} + (\text{in\_eq} * a * b')$
- $\text{out\_eq} = \text{in\_eq} * (a \text{ XNOR } b)$
- $\text{out\_lt} = \text{in\_lt} + (\text{in\_eq} * a' * b)$