

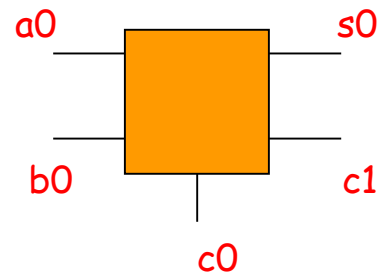
Fast Carry Propagation

During addition, the carry can trigger a "ripple" from the LSB to the MSB. **This slows down the speed of addition.**

$ \begin{array}{r} 01111111111111111111 \\ + \\ 000000000000000001 \\ \hline \end{array} $

How to overcome this? Consider the following:

$$\begin{aligned}
 c_1 &= a_0.b_0 + a_0.c_0 + b_0.c_0 \\
 &= a_0.b_0 + (a_0 + b_0).c_0 \\
 &= g_0 + p_0.c_0 \\
 &\text{(where } g_0 = a_0.b_0, p_0 = a_0+b_0)
 \end{aligned}$$



$$\begin{aligned}
 c_2 &= a_1.b_1 + (a_1 + b_1).c_1 \\
 &= g_1 + p_1.(g_0 + p_0.c_0) \\
 &= g_1 + p_1.g_0 + p_1.p_0.c_0
 \end{aligned}$$

G0

P0

$$c_4 = \underbrace{g_3 + p_3.g_2 + p_3.p_2.g_1 + p_3.p_2.p_1.g_0}_{G0} + \underbrace{p_3.p_2.p_1.p_0}_{P0}.c_0$$

$c_{32} = ?$

It will be **complex**. You can always use a **two-level circuit** to generate c_{32} , which will speed-up addition, but it is impractical due to the complexity.

Practical circuits use a two-phase approach. See the example of the 16-bit adder, designed from **four 4-bit adders**. Let

$$G_0 = g_3 + p_3.g_2 + p_3.p_2.g_1 + p_3.p_2.p_1.g_0$$

$$G_1 = g_7 + p_7.g_6 + p_7.p_6.g_5 + p_7.p_6.p_5.g_4$$

$$G_2 = g_{11} + p_{11}.g_{10} + p_{11}.p_{10}.g_9 + p_{11}.p_{10}.p_9.g_8$$

$$G_3 = g_{15} + p_{15}.g_{14} + p_{15}.p_{14}.g_{13} + p_{15}.p_{14}.p_{13}.g_{12}$$

$$P_0 = p_3.p_2.p_1.p_0$$

$$P_1 = p_7.p_6.p_5.p_4$$

$$P_2 = p_{11}.p_{10}.p_9.p_8$$

$$P_3 = p_{15}.p_{14}.p_{13}.p_{12}$$

Then

If C_1, C_2, C_3, C_4 are the output carry bit from the 1st, 2nd, 3rd, 4th 4-bit adders, then we can write

$$C_1 = G_0 + P_0.c_0$$

$$C_2 = G_1 + P_1.c_1 = G_1 + P_1.G_0 + P_1.P_0.c_0$$

$$C_3 = G_2 + P_2.c_2 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.c_0$$

$$C_4 = G_3 + P_3.c_3 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.c_0$$

How does it help? **Count the number of levels.** The smaller is this number, the faster is the implementation
This is implemented in the **carry look-ahead adder.**

There are other implementations too.

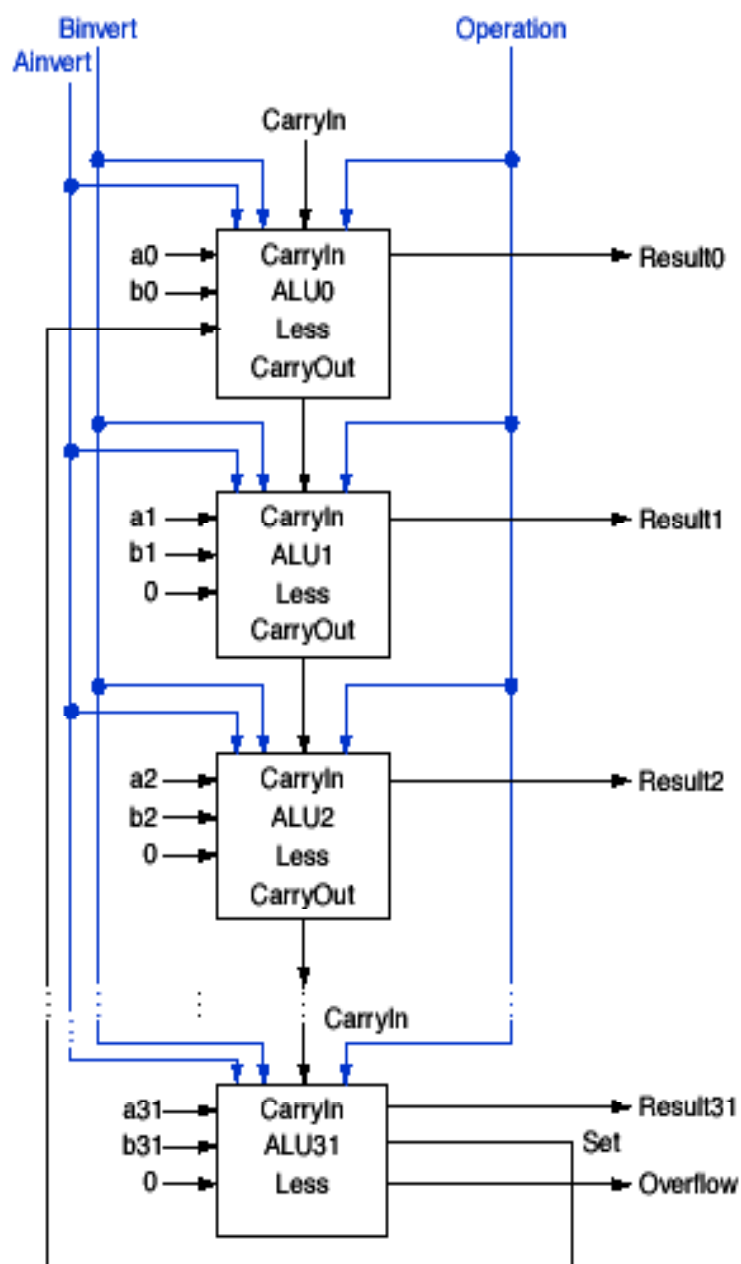


FIGURE B.5.11 A 32-bit ALU constructed from the 31 copies of the 1-bit ALU in the top of Figure B.5.10 and one 1-bit ALU in the bottom of that figure. The Less inputs are connected to 0 except for the least significant bit, which is connected to the Set output of the most significant bit. If the ALU performs $a - b$ and we select the input 3 in the multiplexor in Figure B.5.10, then $\text{Result} = 0 \dots 001$ if $a < b$, and $\text{Result} = 0 \dots 000$ otherwise.

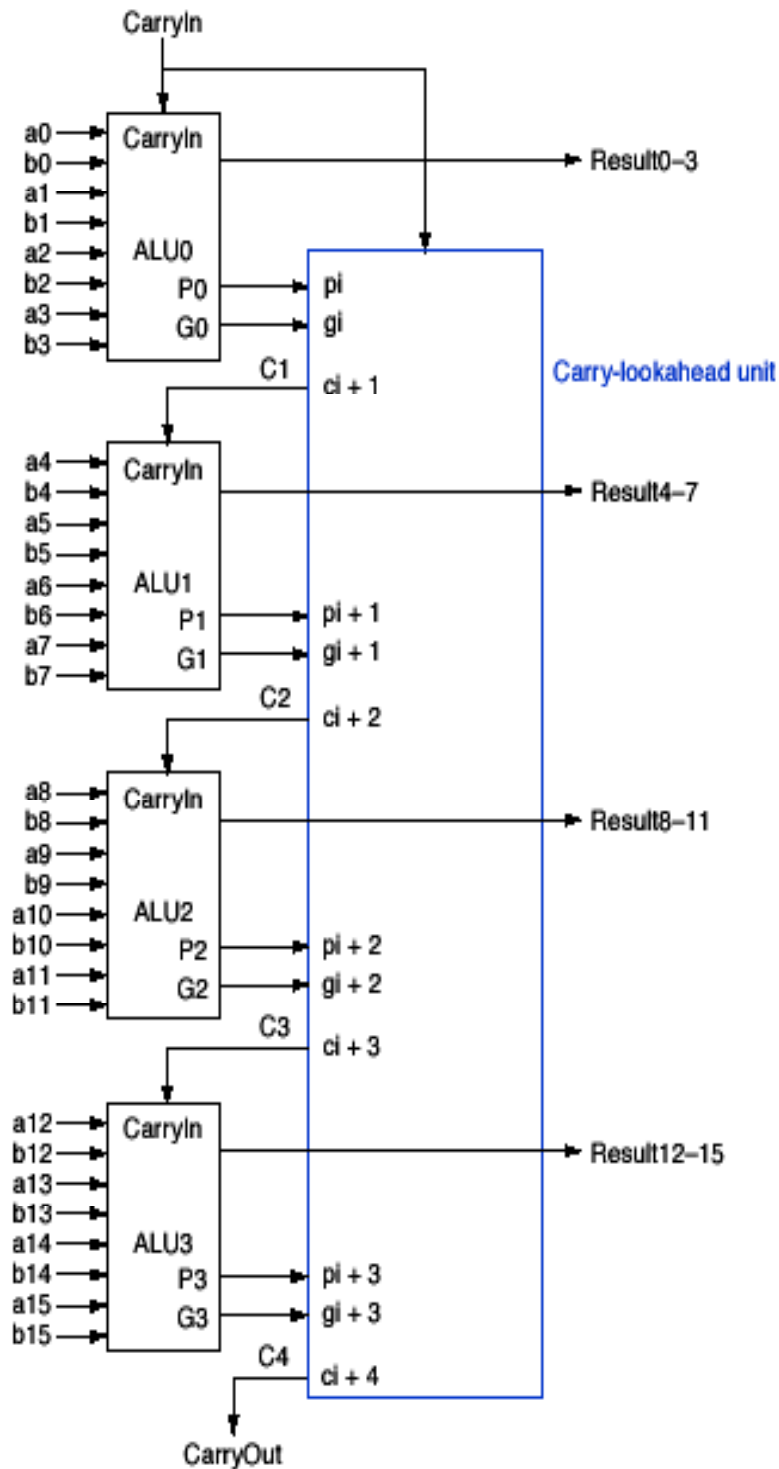


FIGURE B.6.3 Four 4-bit ALUs using carry lookahead to form a 16-bit adder. Note that the carries come from the carry-lookahead unit, not from the 4-bit ALUs.

