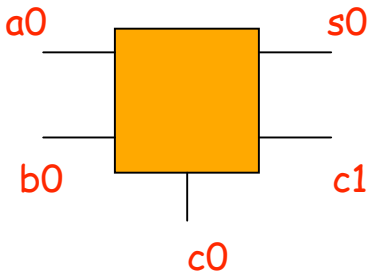
Fast Carry Propagation

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

```
  0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  +
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Calculate the max time it takes to complete a 32-bit addition if each stage takes 1 ns. How to overcome this?

Consider the following:



c1 = a0.b0 + a0.c0 + b0.c0

= a0.b0 + (a0 + b0).c0

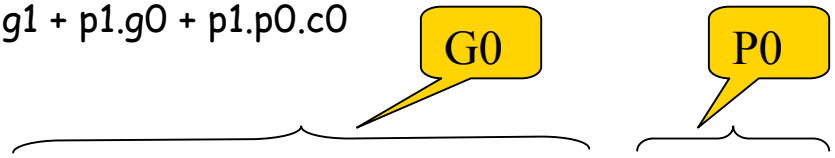
= g0 + p0.c0

(where **g0 = a0.b0**, **p0 = a0+b0**)

c2 = a1.b1 + (a1 + b1).c1

= g1 + p1.(g0 + p0.c0)

= g1 + p1.g0 + p1.p0.c0



c4 = g3 + p3.g2 + p3.p2.g1 + p3.p2.p1.g0 + p3.p2.p1.p0. c0

We could calculate c32 in this way.

It will be **complex**. But how much time will it take now? Assume that each gate takes 1 ns.

You can always use a two-level circuit to generate c32, which will speed-up addition (do 32-bit addition in 2 ns), but it is impractical due to the complexity.

Many practical circuits use a two-phase approach. Consider the example of a 16-bit adder, designed from four 4-bit adders. Let

$G0 = g3 + p3.g2 + p3.p2.g1 + p3.p2.p1.g0$

$G1 = g7 + p7.g6 + p7.p6.g5 + p7.p6.p5.g4$

$G2 = g11 + p11.g10 + p11.p10.g9 + p11.p10.p9.g8$

$G3 = g15 + p15.g14 + p15.p14.g13 + p15.p14.p13.g12$

$P0 = p3.p2.p1.p0$

$P1 = p7.p6.p5.p4$

$P2 = p11.p10.p9.p8$

$P3 = p15.p14.p13.p12$

Then if C1, C2, C3, C4 are the output carry bit from the 1st, 2nd, 3rd, 4th 4-bit adders, then we can write

C1 = G0 + P0.c0

C2 = G1 + P1.C1 = G1 + P1.G0 + P1.P0.c0

C3 = G2 + P2.C2 = G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.c0

C4 = G3 + P3.C3 = G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.c0

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.

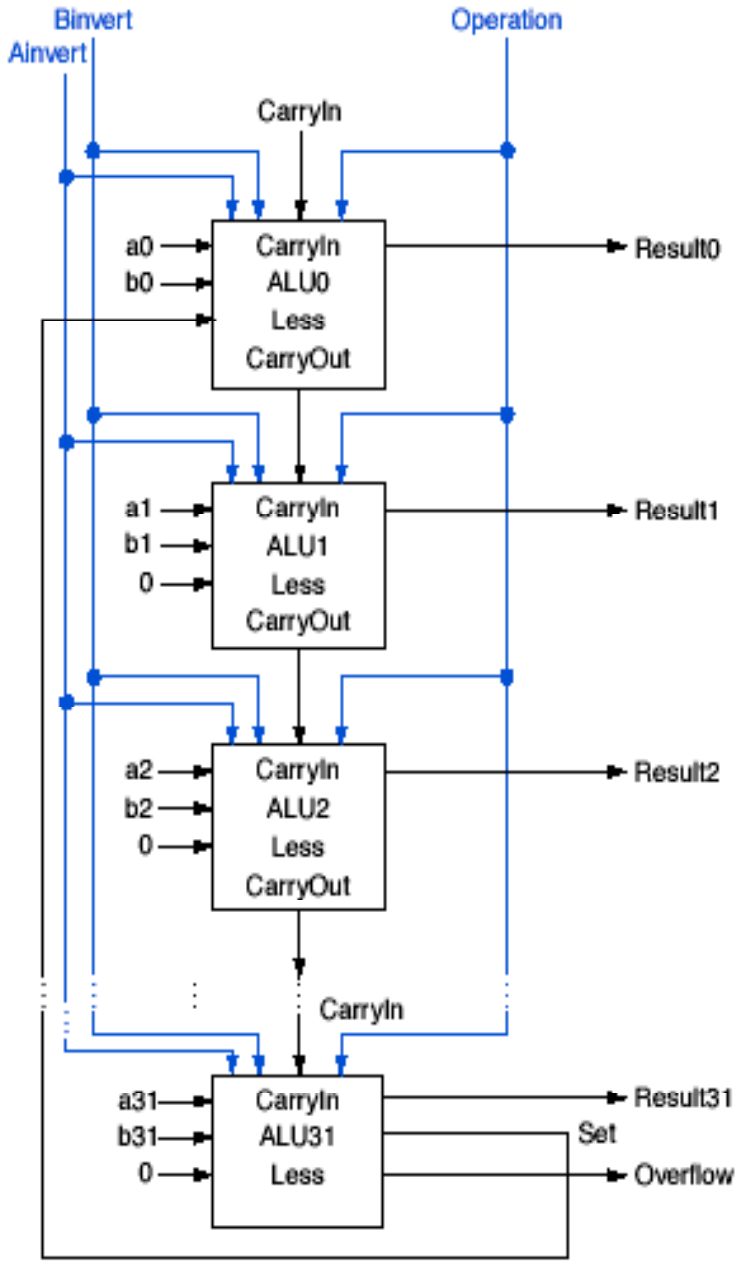


FIGURE B.5.11 A 32-bit ALU constructed from the 31 copies of the 1-bit ALU in the top of Figure B.5.10 and one 1-bit ALU in the bottom of that figure. The Less inputs are connected to 0 except for the least significant bit, which is connected to the Set output of the most significant bit. If the ALU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

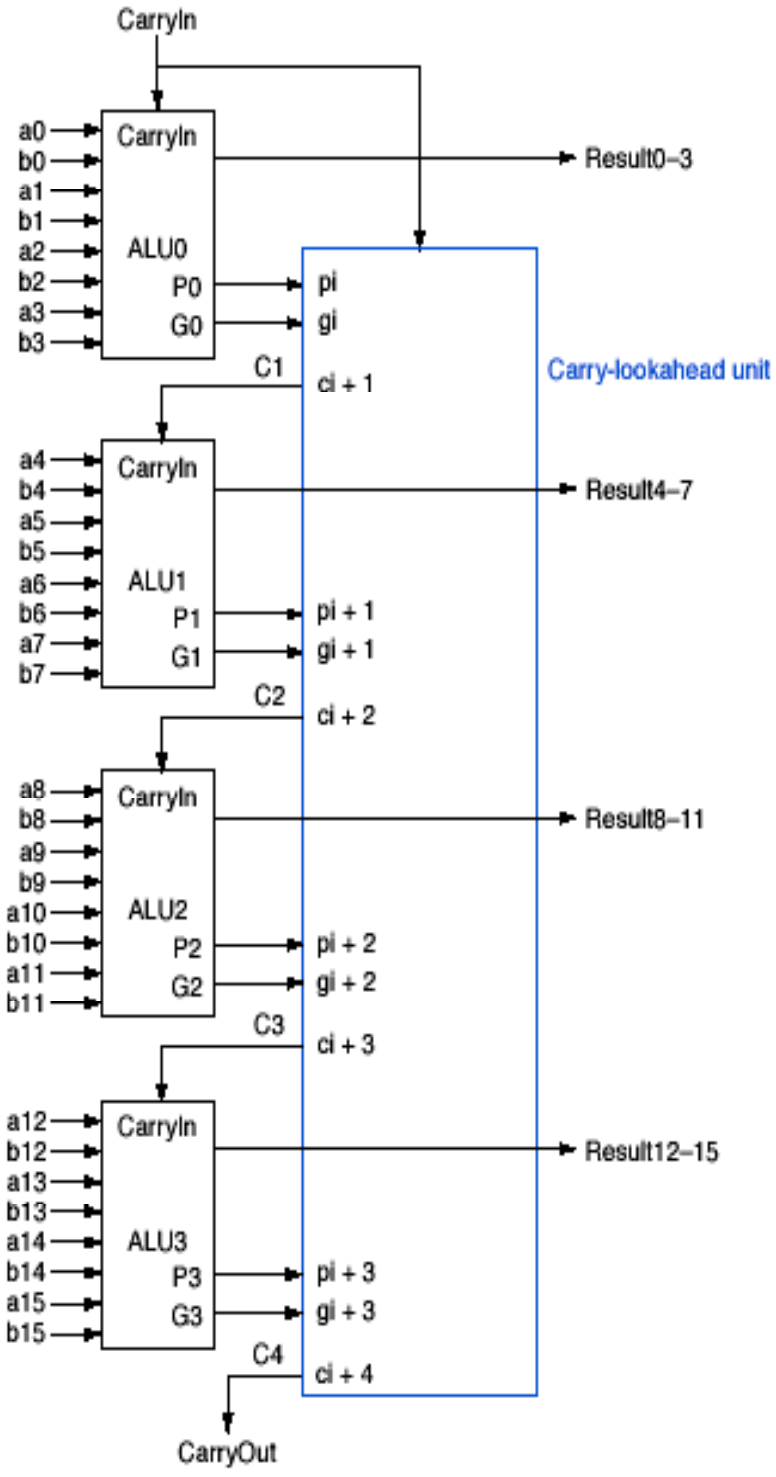


FIGURE B.6.3 Four 4-bit ALUs using carry lookahead to form a 16-bit adder. Note that the carries come from the carry-lookahead unit, not from the 4-bit ALUs.