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

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

c32 = ?

It will be complex. But you can use a two-level circuit to generate c4. This will expedite 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 in p.246. 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

$C1 = G0 + P0.c0$

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

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

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

This is implemented in the **carry look-ahead adder**.

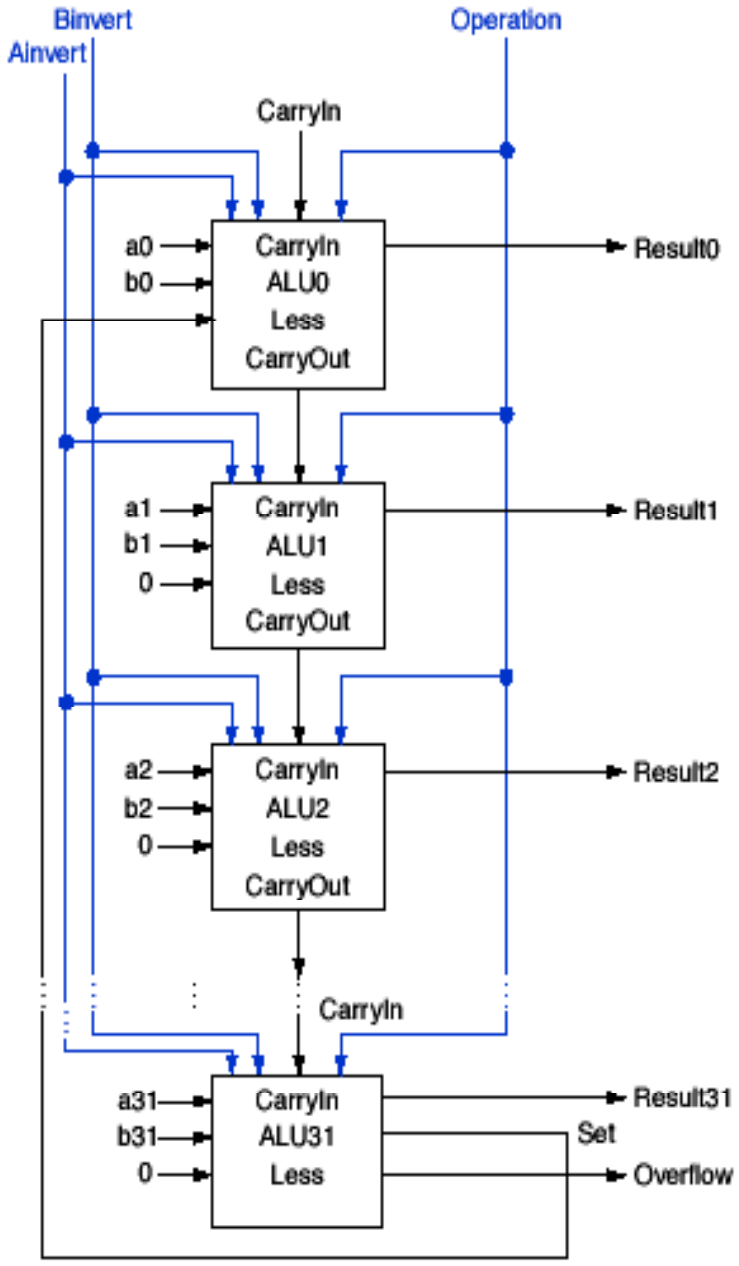


FIGURE B.5.11 A 32-bit ALU constructed from the 31 copies of the 1-bit ALU in the top of Figure B.5.10 and one 1-bit ALU in the bottom of that figure. The Less inputs are connected to 0 except for the least significant bit, which is connected to the Set output of the most significant bit. If the ALU 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.

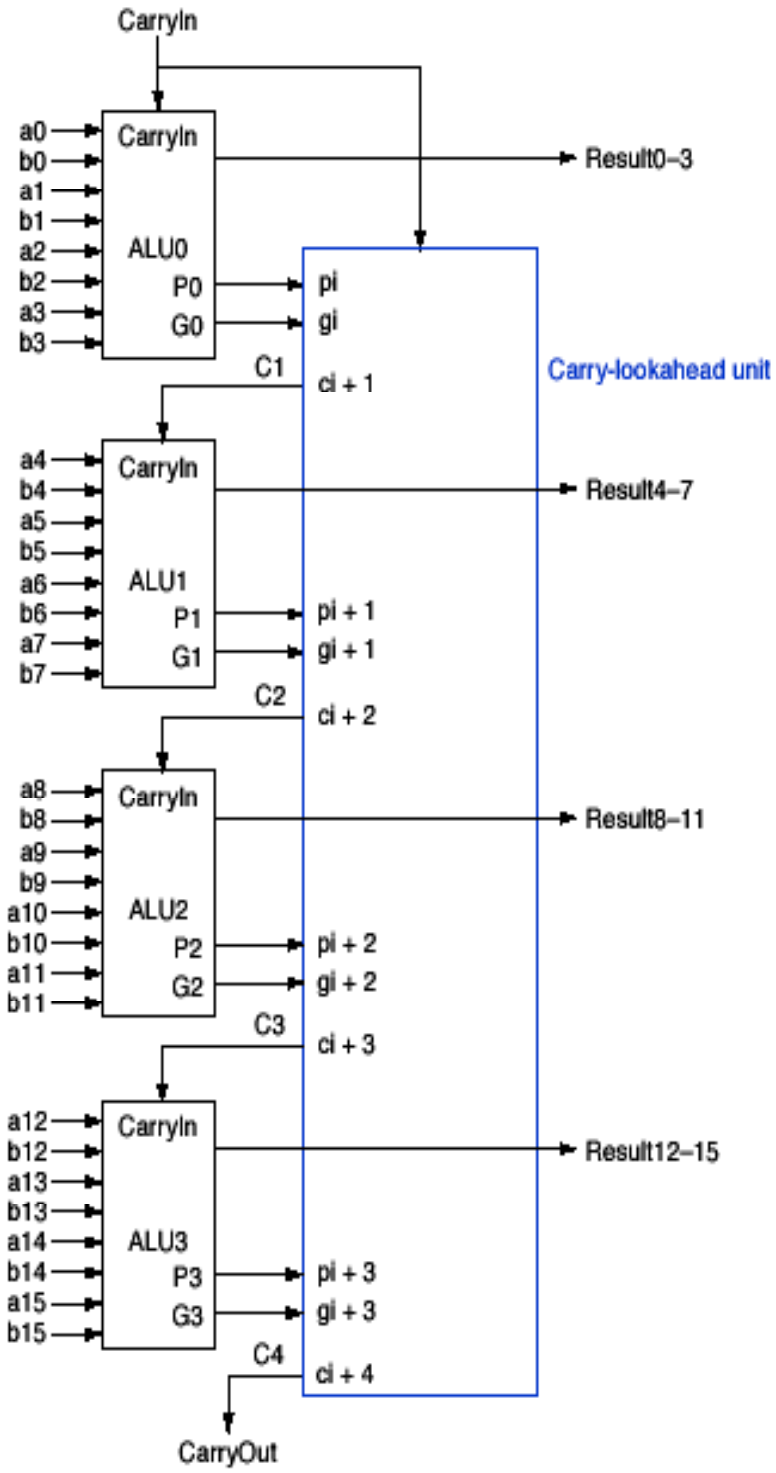
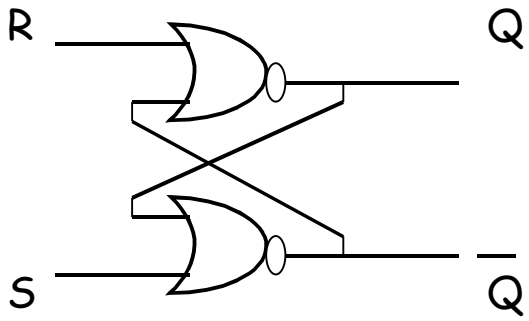


FIGURE B.6.3 Four 4-bit ALUs using carry lookahead to form a 16-bit adder. Note that the carries come from the carry-lookahead unit, not from the 4-bit ALUs.

Sequential Circuits

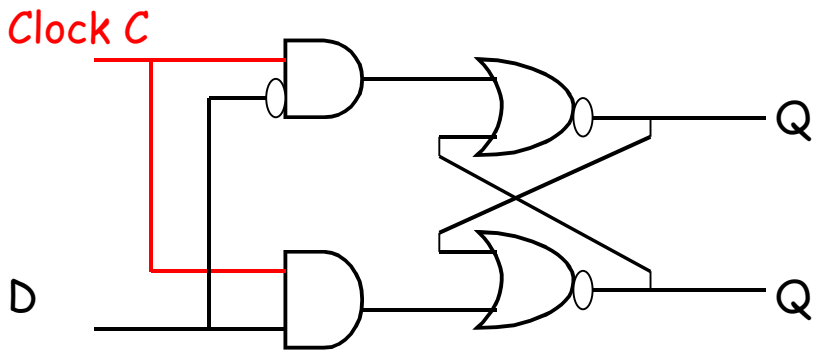
The output depends not only on the current inputs, but also on the past values of the inputs.



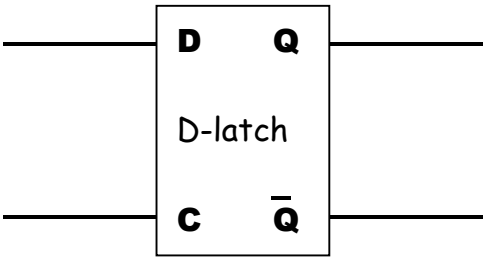
An SR Latch

S	R	Q	$\overline{Q}$	Comment
0	0	0/1	1/0	Old state continues
1	0	1	0	Set state
0	1	0	1	Reset state
1	1	0	0	Illegal inputs

A clocked D-latch

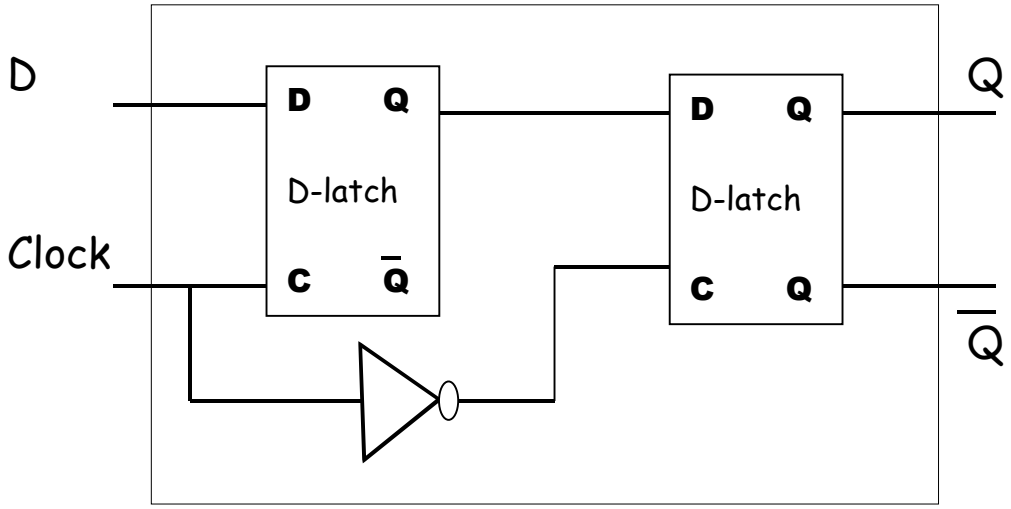


Clock is the enabler. If C=0, Q remains unchanged. When C=1, then Q acquires the value of D. We will use it as a building block of sequential circuits.

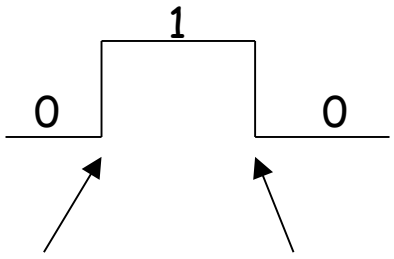


The main complaint is the "transparency". An edge triggered circuit (or a master-slave circuit) solves this problem (to be discussed in the class)

A Master-Slave D flip-flop



The output Q acquires the value of D, only when one complete pulse (i.e. 0 1 0) is applied to the clock input. The external output Q changes after the falling edge.



A clock pulse

rising edge falling edge