Computer Architecture: Adders

Brent van Bladel

Stephen Pauwels

Universiteit Antwerpen

Addition

$$\begin{array}{r} 00101110 \\ +\ 00100111 \\ \hline \end{array}$$

Addition

$$\begin{array}{r} \mathit{0101110} \\ 00101110 \\ +\ 00100111 \\ \hline 01010101 \end{array}$$

Addition

$$\begin{array}{rcccccccc}
 & \mathit{0} & \mathit{1} & \mathit{0} & \mathit{1} & \mathit{1} & \mathit{1} & \mathit{0} & \\
 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
+ & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
\hline
 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}$$

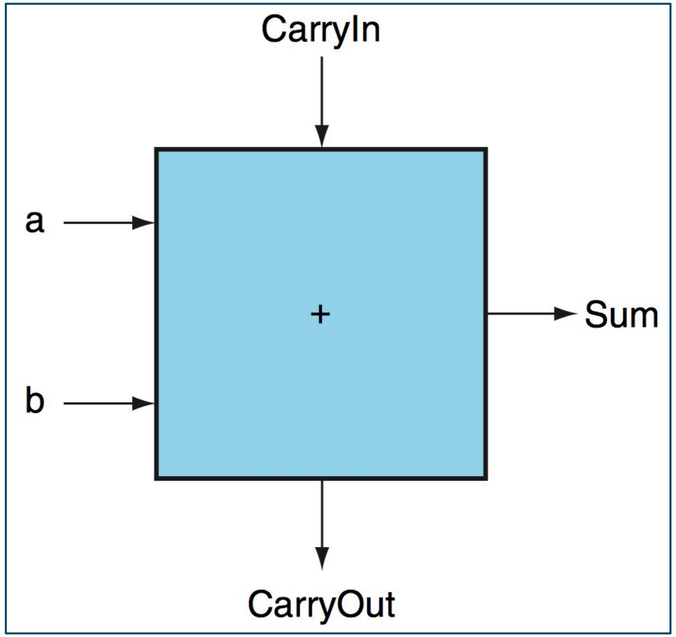
Addition

Inputs			Outputs		
a	b	CarryIn	CarryOut	Sum	Comments
0	0	0	0	0	$0 + 0 + 0 = 00_{two}$
0	0	1	0	1	$0 + 0 + 1 = 01_{two}$
0	1	0	0	1	$0 + 1 + 0 = 01_{two}$
0	1	1	1	0	$0 + 1 + 1 = 10_{two}$
1	0	0	0	1	$1 + 0 + 0 = 01_{two}$
1	0	1	1	0	$1 + 0 + 1 = 10_{two}$
1	1	0	1	0	$1 + 1 + 0 = 10_{two}$
1	1	1	1	1	$1 + 1 + 1 = 11_{two}$

1-Bit Full Adder

1-bit addition:

- CarryOut output is 1 when at least two inputs are 1
- Sum output is 1 when exactly one input is 1 or all three inputs are 1



1-Bit Full Adder

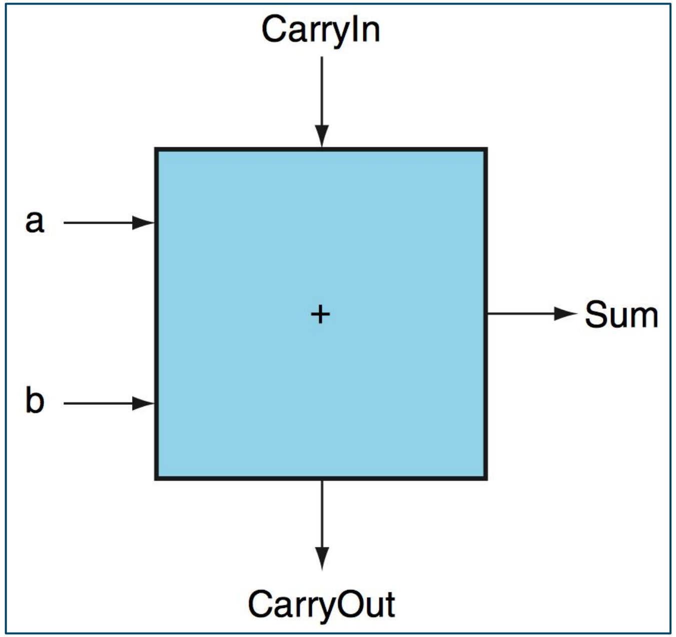
1-bit addition:

- CarryOut output is 1 when at least two inputs are 1

$$\text{CarryOut} = (b \cdot \text{CarryIn}) + (a \cdot \text{CarryIn}) + (a \cdot b)$$

- Sum output is 1 when exactly one input is 1 or all three inputs are 1

$$\text{Sum} = (a \cdot \overline{b} \cdot \overline{\text{CarryIn}}) + (\overline{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\overline{a} \cdot \overline{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

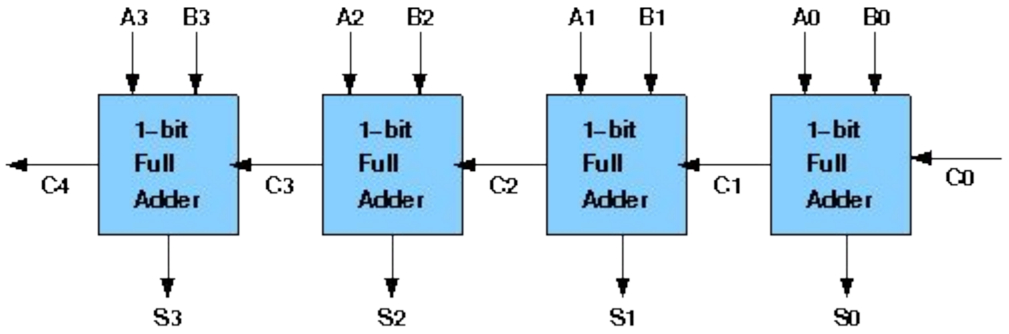


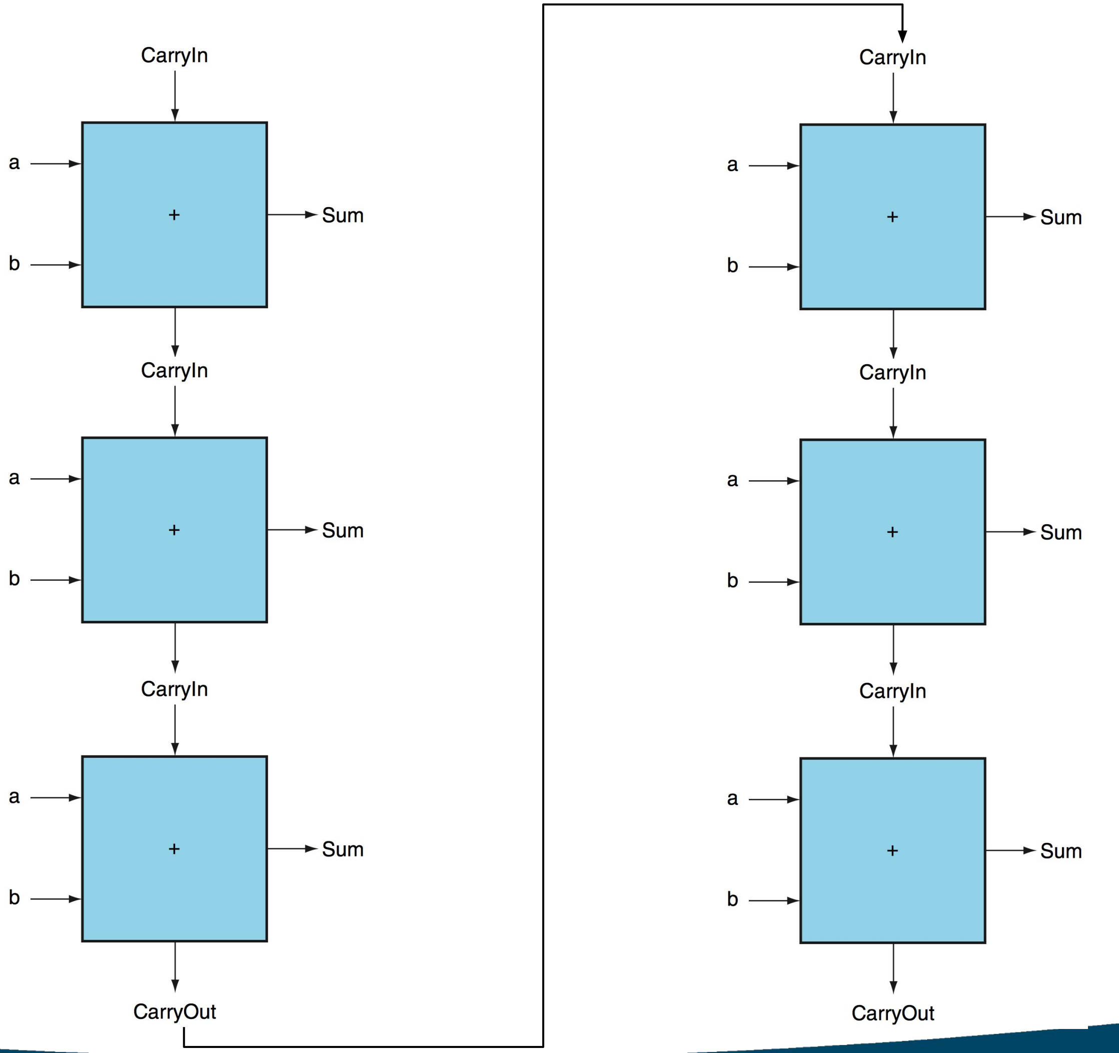
Ripple Carry Adder

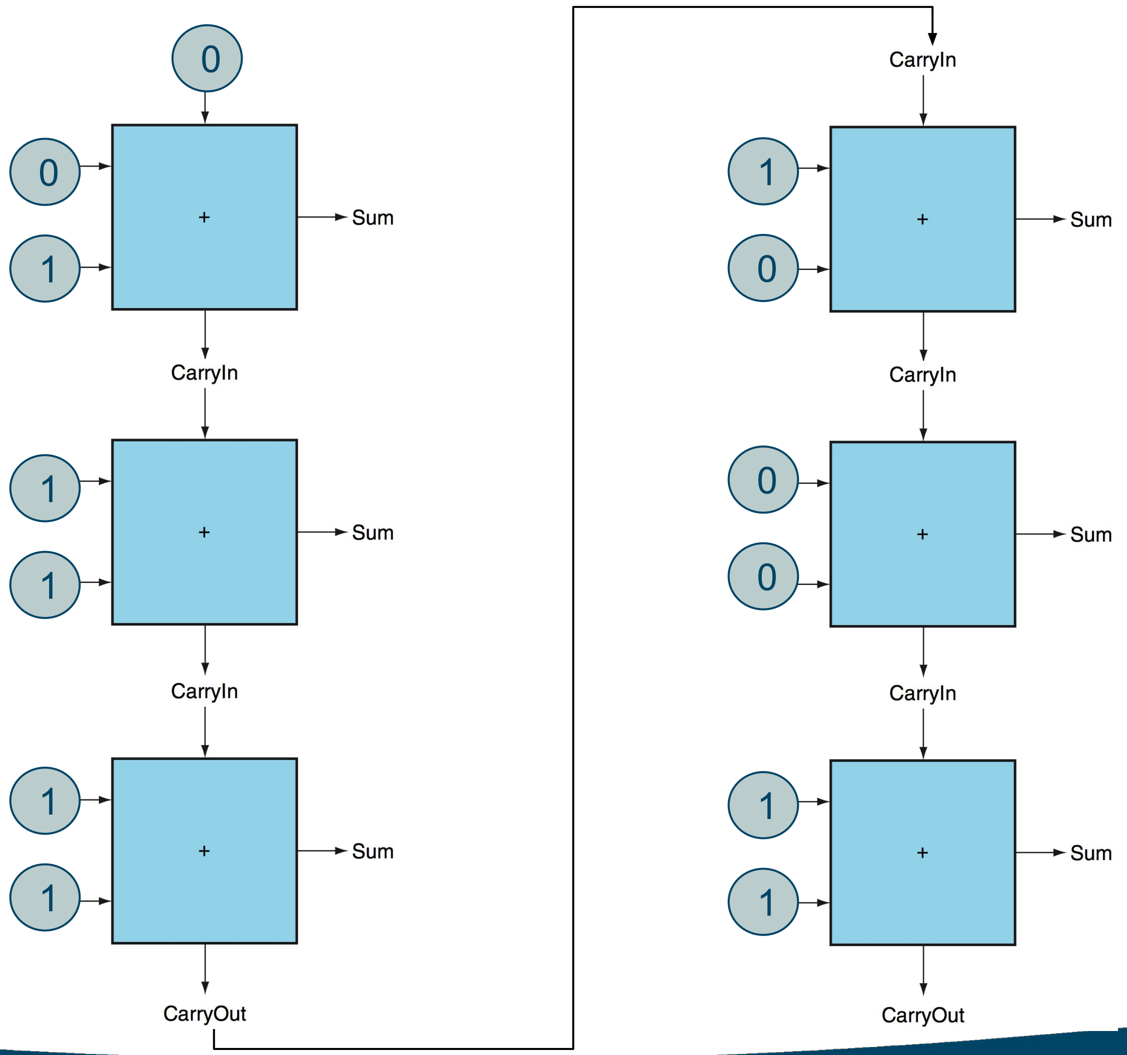
n-bit addition:

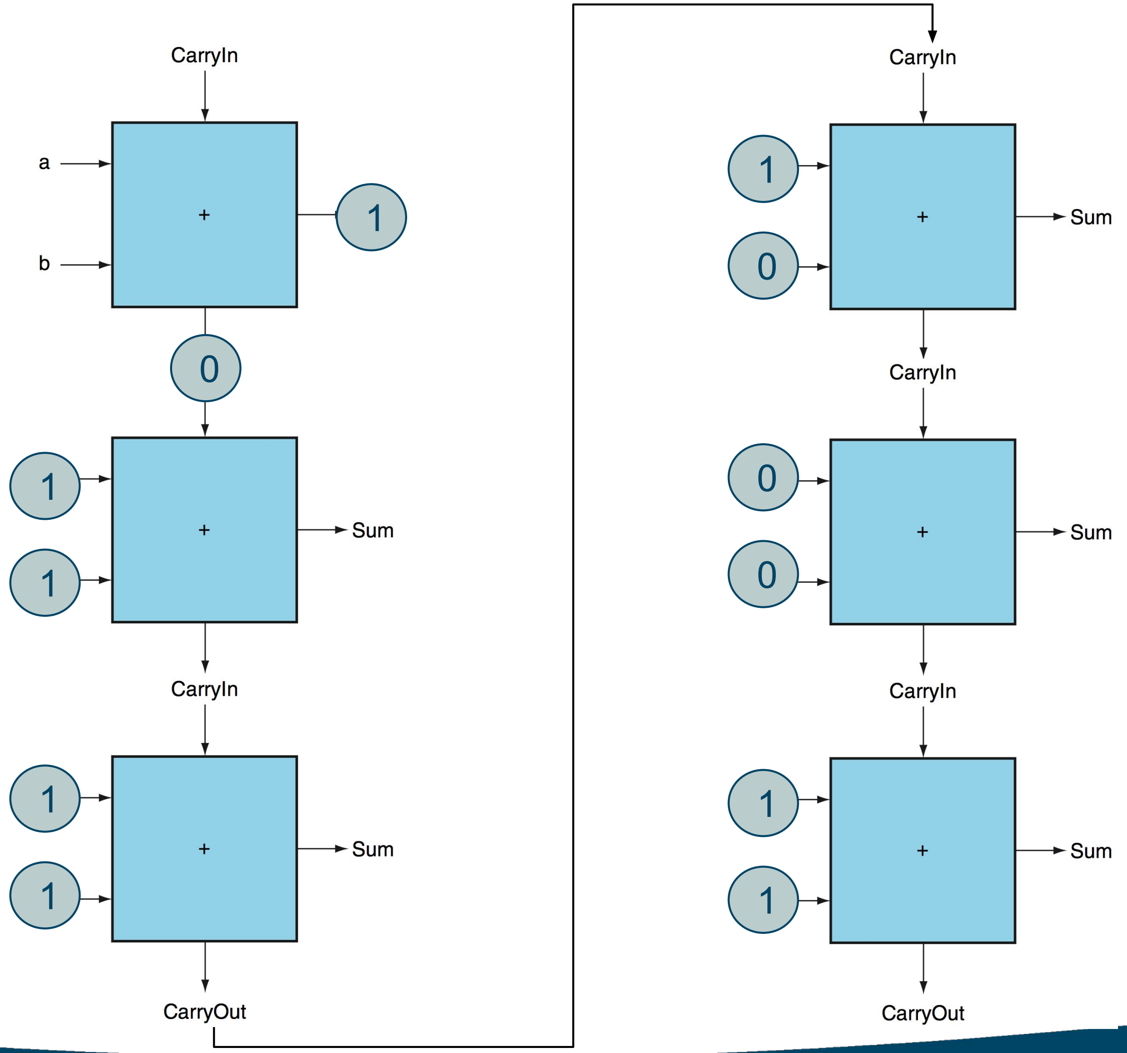
- Series of 1-bit full adders

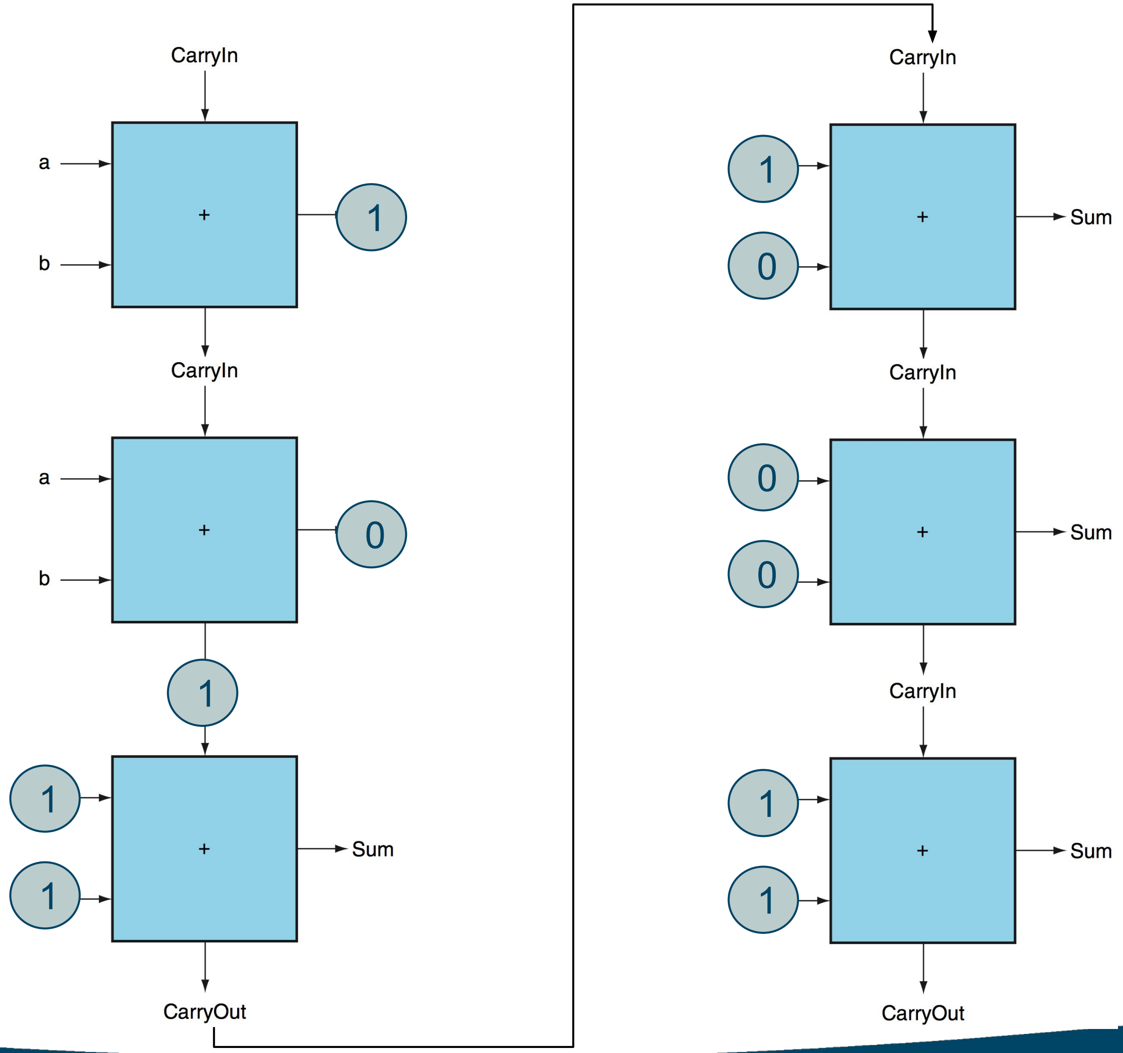
- Carry ripples through addition = Slow!

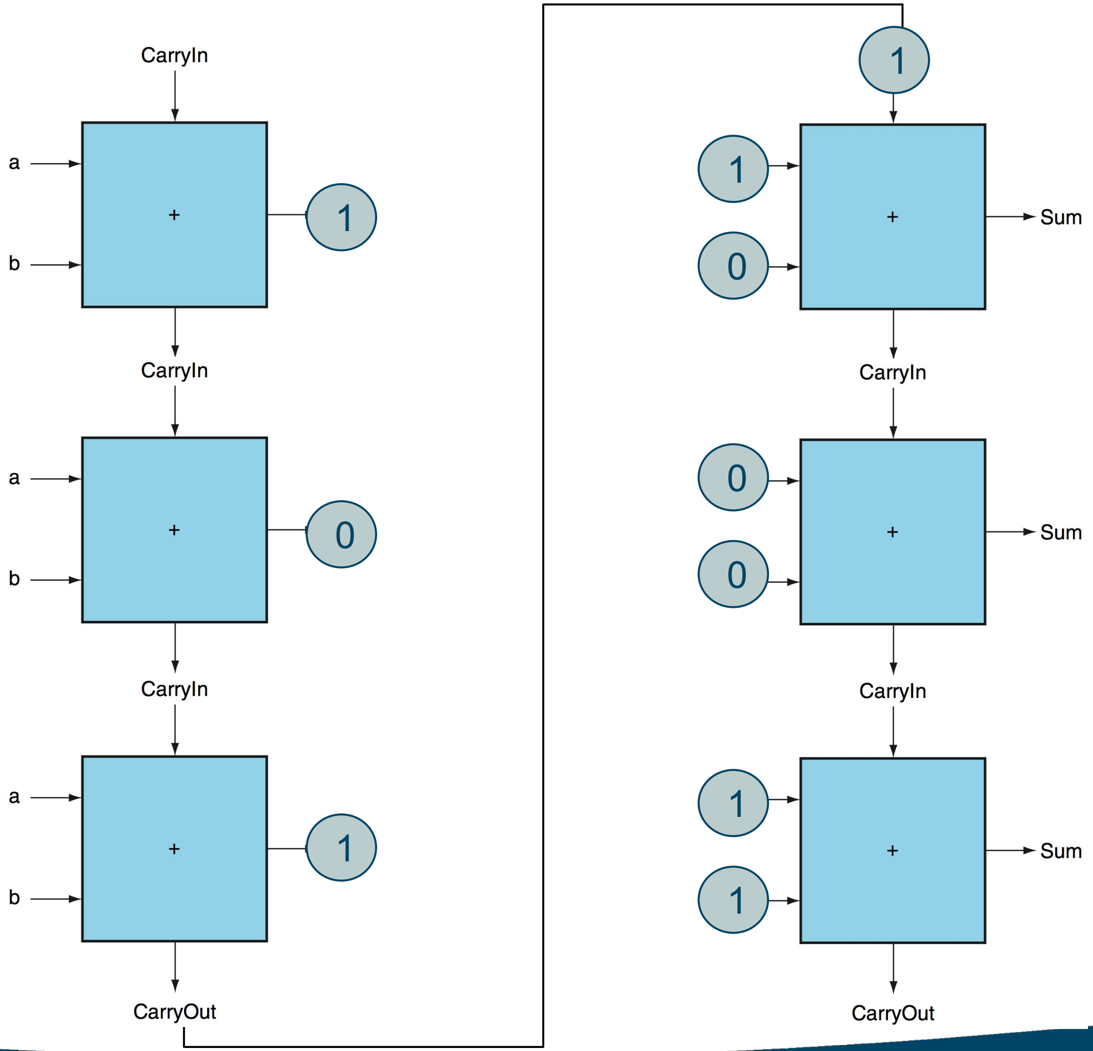



CarryIn
a
+
Sum
b
CarryIn
a
+
Sum
b
CarryIn
a
+
Sum
b
CarryOut
CarryIn
a
+
Sum
b
CarryIn
a
+
Sum
b
CarryIn
a
+
Sum
b
CarryOut

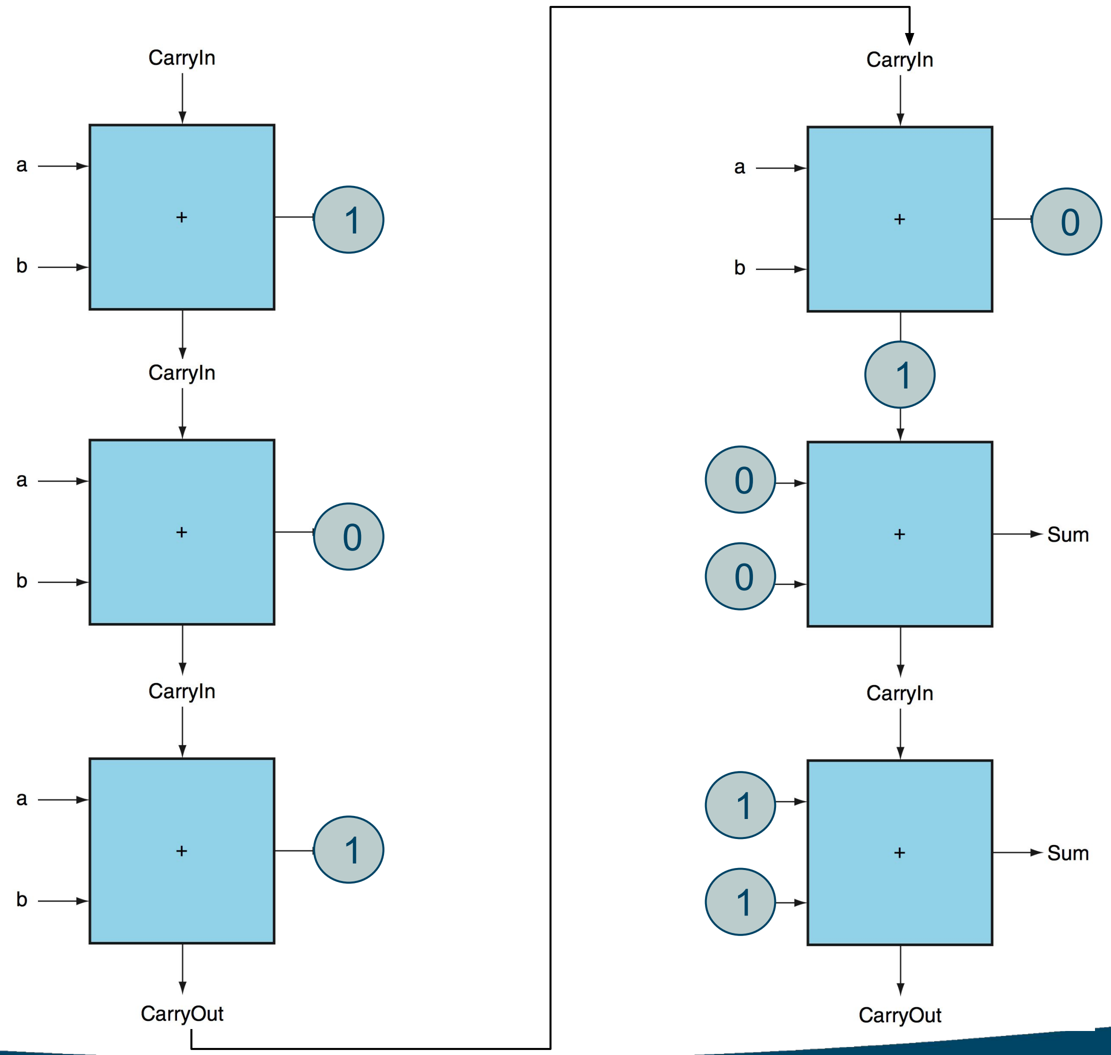

0
0
1
+
Sum
CarryIn
1
1
+
Sum
CarryIn
1
1
+
Sum
CarryOut
CarryIn
1
0
+
Sum
CarryIn
0
0
+
Sum
CarryIn
1
1
+
Sum
CarryOut

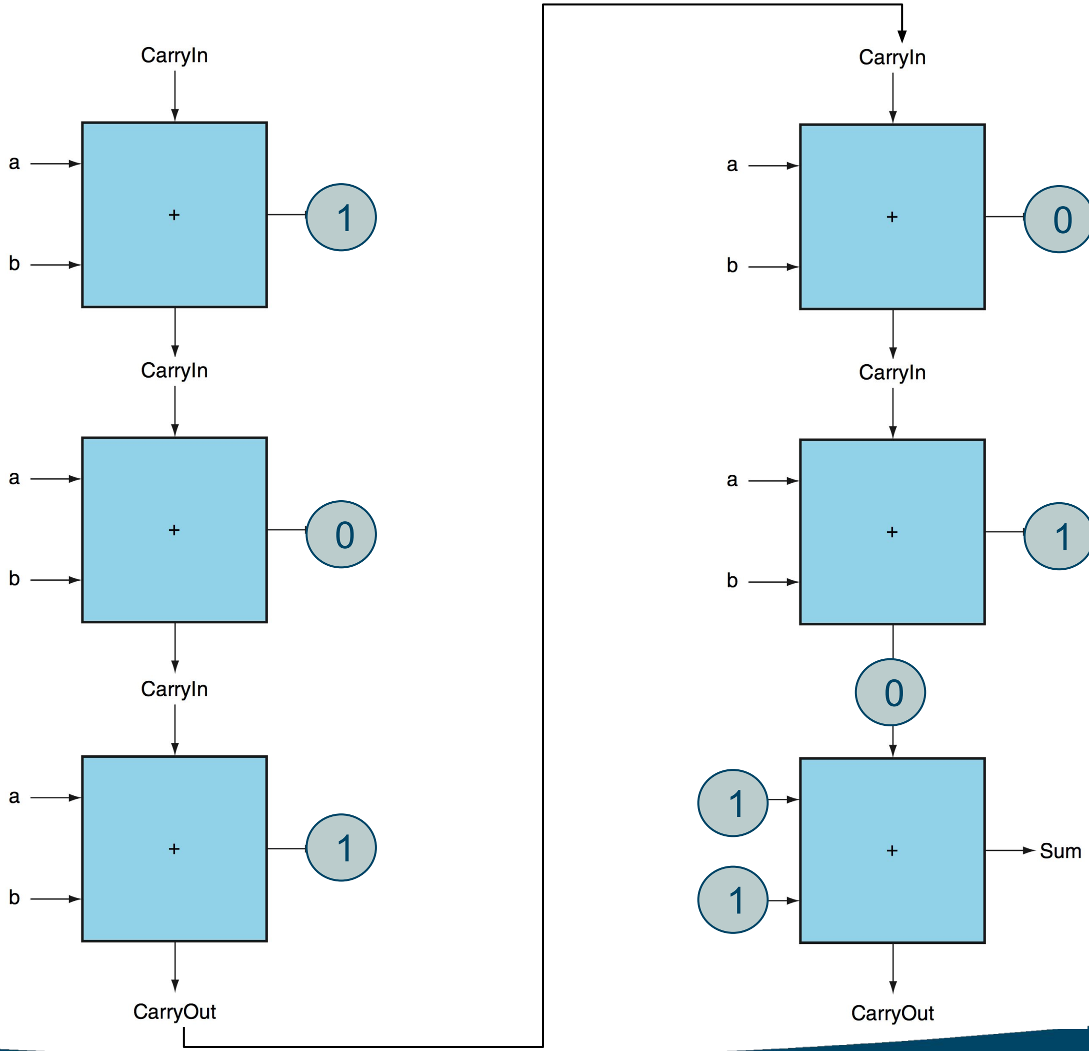

CarryIn
a
b
+
1
0
1
1
+
Sum
CarryIn
1
1
+
Sum
CarryOut
CarryIn
1
0
+
Sum
CarryIn
0
0
+
Sum
CarryIn
1
1
+
Sum
CarryOut

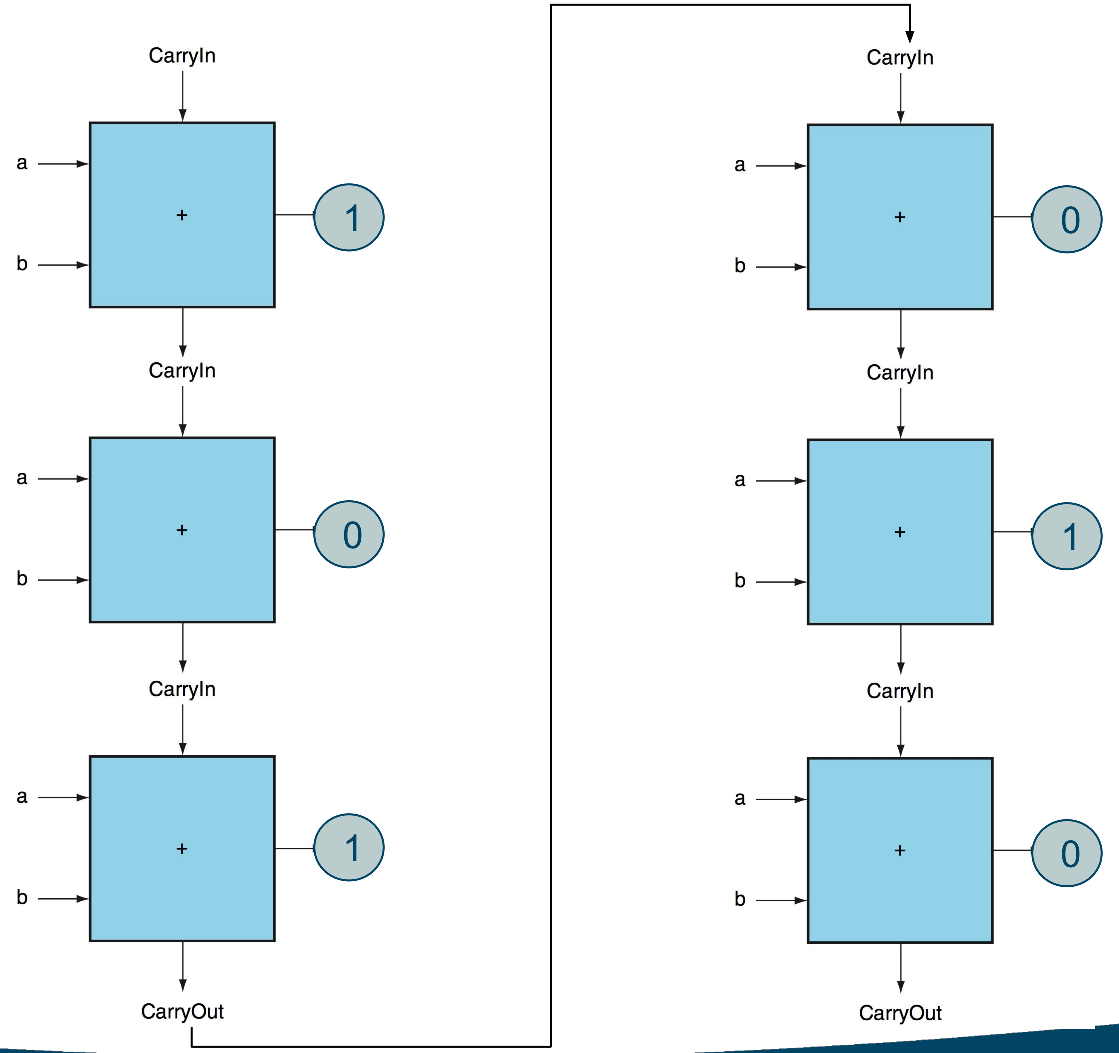

CarryIn
a
b
+
1
CarryIn
a
b
+
0
1
1
1
+
Sum
CarryOut
CarryIn
1
0
+
Sum
CarryIn
0
0
+
Sum
CarryIn
1
1
+
Sum
CarryOut



CarryIn
a
b
+
1
CarryIn
a
b
+
0
CarryIn
a
b
+
1
CarryOut
1
1
0
+
Sum
CarryIn
0
0
+
Sum
CarryIn
1
1
+
Sum
CarryOut


CarryIn
a
b
+
1
CarryIn
a
b
+
0
CarryIn
a
b
+
1
CarryOut
CarryIn
a
b
+
0
1
0
0
+
Sum
CarryIn
1
1
+
Sum
CarryOut


CarryIn
a
b
+
1
CarryIn
a
b
+
0
CarryIn
a
b
+
1
CarryOut
CarryIn
a
b
+
0
CarryIn
a
b
+
1
0
1
1
+
Sum
CarryOut


CarryIn
a
b
+
1
CarryIn
a
b
+
0
CarryIn
a
b
+
1
CarryOut
CarryIn
a
b
+
0
CarryIn
a
b
+
1
CarryIn
a
b
+
0
CarryOut

Calculate Carry

```
  ???? ????
  0010 1110
+ 0010 0111
-----------
```

Fast Carry Adder using “Infinite Hardware”

CarryIn1 = CarryOut0

$$\begin{array}{rcccc}
 & c3 & c2 & c1 & c0 \\
 & a3 & a2 & a1 & a0 \\
+ & b3 & b2 & b1 & b0 \\
\hline
 & s3 & s2 & s1 & s0
\end{array}$$

Fast Carry Adder using “Infinite Hardware”

c1 = (b0 . c0) + (a0 . c0) + (a0 . b0)

$$\begin{array}{rcccc}
 & c3 & c2 & c1 & c0 \\
 & a3 & a2 & a1 & a0 \\
+ & b3 & b2 & b1 & b0 \\
\hline
 & s3 & s2 & s1 & s0
\end{array}$$

Fast Carry Adder using “Infinite Hardware”

c1 = (b0 . c0) + (a0 . c0) + (a0 . b0)

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

```
   c3 c2 c1 c0
   a3 a2 a1 a0
+  b3 b2 b1 b0
   -----------
   s3 s2 s1 s0
```

Fast Carry Adder using “Infinite Hardware”

c1 = (b0 . c0) + (a0 . c0) + (a0 . b0)

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

c2 = (a1 . a0 . b0) + (a1 . a0 . c0) + (a1 . b0 . c0) + (b1 . a0 . b0) + (b1 . a0 . c0) + (b1 . b0 . c0) + (a1 . b1)

$$\begin{array}{rcccc} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}$$

Fast Carry Adder using “Infinite Hardware”

c1 = (b0 . c0) + (a0 . c0) + (a0 . b0)

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

c2 = (a1 . a0 . b0) + (a1 . a0 . c0) + (a1 . b0 . c0) + (b1 . a0 . b0) + (b1 . a0 . c0) + (b1 . b0 . c0) + (a1 . b1)

$$\begin{array}{rcccc} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}$$

→ “direct” computation of carry (sum of products)

→ fast, but complex

Propagate and Generate

Generate: “When does a_i and b_i generate a carry?”

$$
\begin{array}{rcccc}
 & c3 & c2 & c1 & c0 \\
 & a3 & a2 & a1 & a0 \\
+ & b3 & b2 & b1 & b0 \\
\hline
 & s3 & s2 & s1 & s0
\end{array}
$$

Propagate and Generate

Generate: “When does a_i and b_i generate a carry?”

$g_i = a_i . b_i$

Propagate: “When does a_i and b_i propagate a carry?”

$$\begin{array}{cllll} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}$$

Propagate and Generate

Generate: “When does a_i and b_i generate a carry?”

$g_i = a_i \cdot b_i$

Propagate: “When does a_i and b_i propagate a carry?”

$p_i = a_i + b_i$

CarryIn:

$$\begin{array}{cccc} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}$$

Propagate and Generate

Generate: “When does a_i and b_i generate a carry?”

$g_i = a_i . b_i$

Propagate: “When does a_i and b_i propagate a carry?”

$p_i = a_i + b_i$

CarryIn:

$c_{i+1} = g_i + p_i . c_i$

$$\boxed{\begin{array}{rcccc} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}}$$

Propagate and Generate

First Layer of Abstraction

Generate: “When does a_i and b_i generate a carry?”

$g_i = a_i \cdot b_i$

Propagate: “When does a_i and b_i propagate a carry?”

$p_i = a_i + b_i$

CarryIn:

$c_{i+1} = g_i + p_i \cdot c_i$

$$\begin{array}{rcccc} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}$$

Carry Lookahead Adder

$$c1 = g0 + (p0 \cdot c0)$$

$$\begin{aligned} c2 = g1 &+ (p1 \cdot g0) \\ &+ (p1 \cdot p0 \cdot c0) \end{aligned}$$

$$\begin{aligned} c3 = g2 &+ (p2 \cdot g1) \\ &+ (p2 \cdot p1 \cdot g0) \\ &+ (p2 \cdot p1 \cdot p0 \cdot c0) \end{aligned}$$

$$\begin{aligned} c4 = g3 &+ (p3 \cdot g2) \\ &+ (p3 \cdot p2 \cdot g1) \\ &+ (p3 \cdot p2 \cdot p1 \cdot g0) \\ &+ (p3 \cdot p2 \cdot p1 \cdot p0 \cdot c0) \end{aligned}$$

$$\begin{array}{rcccc} & c3 & c2 & c1 & c0 \\ & a3 & a2 & a1 & a0 \\ + & b3 & b2 & b1 & b0 \\ \hline & s3 & s2 & s1 & s0 \end{array}$$

Carry Lookahead Adder

$$\begin{array}{r} ???\boxed{?} \\ 0010 \\ +0010 \\ \hline \end{array} \qquad \begin{array}{r} ???? \\ 1110 \\ +\ 0111 \\ \hline \end{array}$$

Super Propagates and -Generates

Second Layer of Abstraction

Calculate CarryIn of each 4-bit carry-lookahead adder.

Superpropagate Pi and supergenerate Gi :

$P0 = p3 \cdot p2 \cdot p1 \cdot p0$

$G0 = g3 + (p3 \cdot g2) + (p3 \cdot p2 \cdot g1) + (p3 \cdot p2 \cdot p1 \cdot g0)$

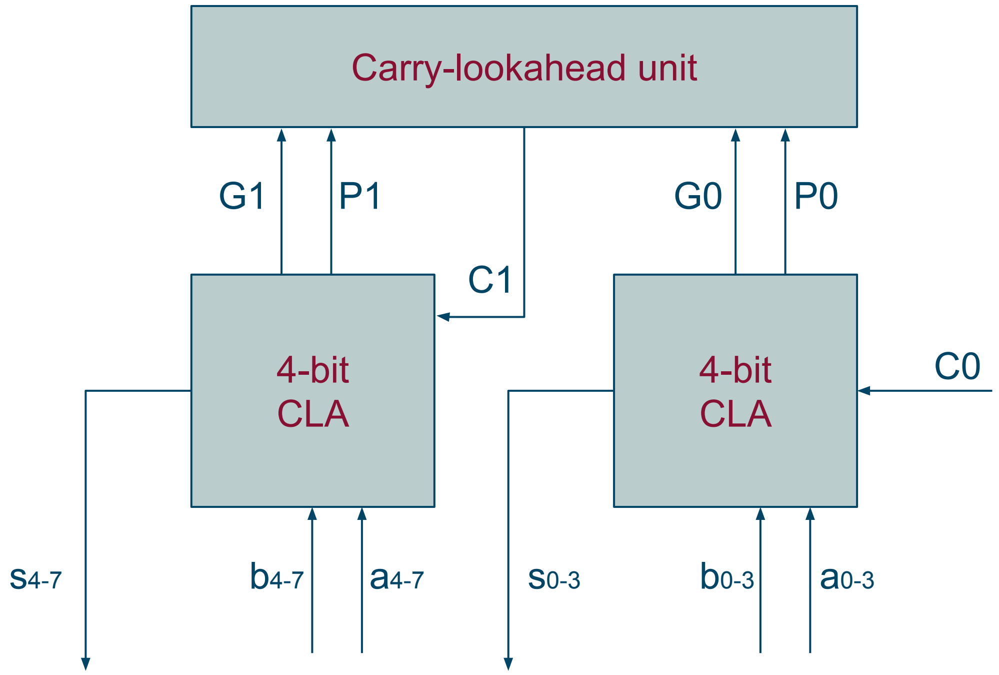
Calculate Ci :

$C1 = G0 + (P0 \cdot c0)$

$C2 = G1 + (P1 \cdot G0) + (P1 \cdot P0 \cdot c0)$

$C3 = \ldots$

8-bit Carry Lookahead Adder


Carry-lookahead unit
G1
P1
C1
G0
P0
4-bit CLA
4-bit CLA
C0
s4-7
b4-7
a4-7
s0-3
b0-3
a0-3