

Computer Architecture: Adders

Brent van Bladel

Stephen Pauwels

Addition

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



Addition

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

Addition

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

Addition

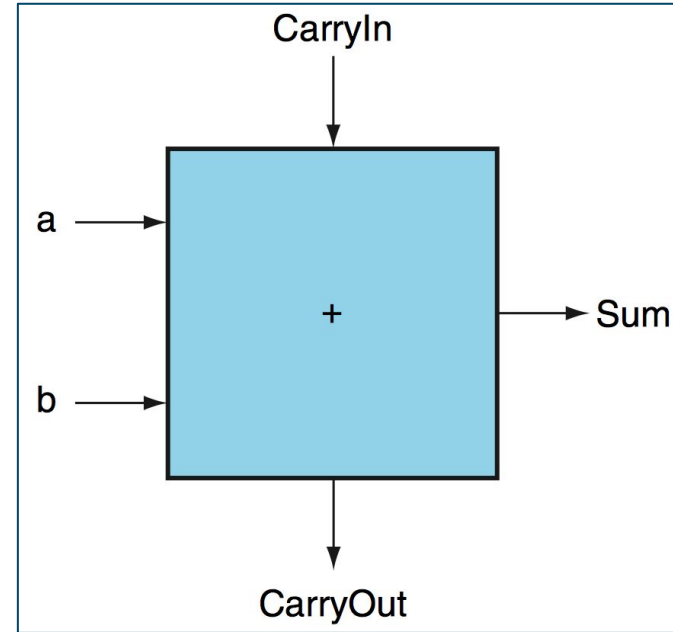
| Inputs | | | Outputs | | Comments |
|--------|---|---------|----------|-----|-------------------------------|
| a | b | CarryIn | CarryOut | Sum | |
| 0 | 0 | 0 | 0 | 0 | $0 + 0 + 0 = 00_{\text{two}}$ |
| 0 | 0 | 1 | 0 | 1 | $0 + 0 + 1 = 01_{\text{two}}$ |
| 0 | 1 | 0 | 0 | 1 | $0 + 1 + 0 = 01_{\text{two}}$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



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

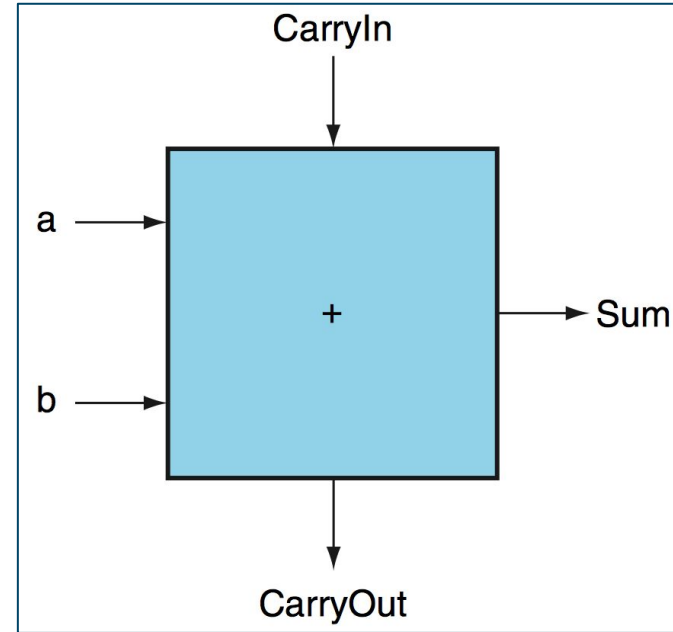
1-bit addition:

- 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

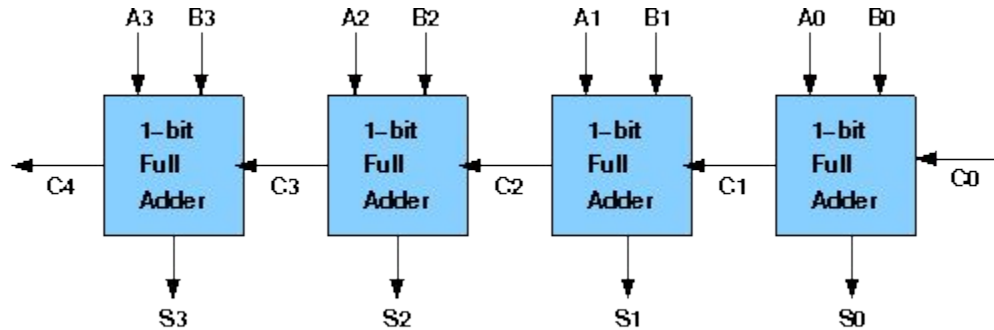
$$\text{Sum} = (a \cdot \bar{b} \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot b \cdot \overline{\text{CarryIn}}) + (\bar{a} \cdot \bar{b} \cdot \text{CarryIn}) + (a \cdot b \cdot \text{CarryIn})$$

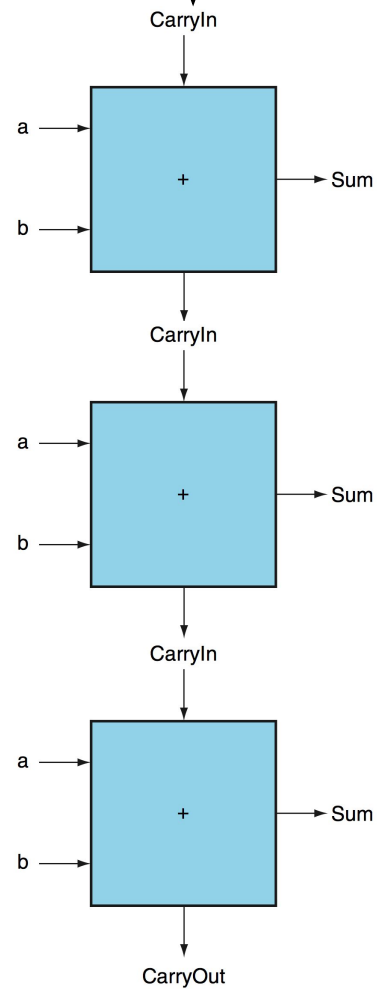
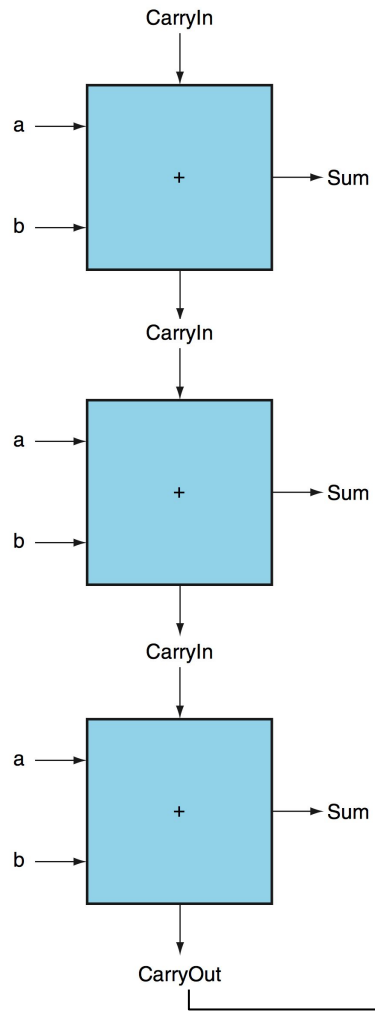


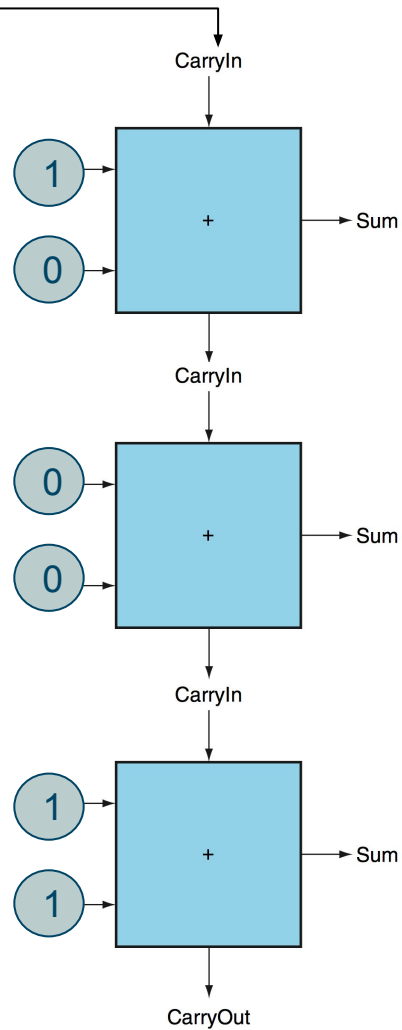
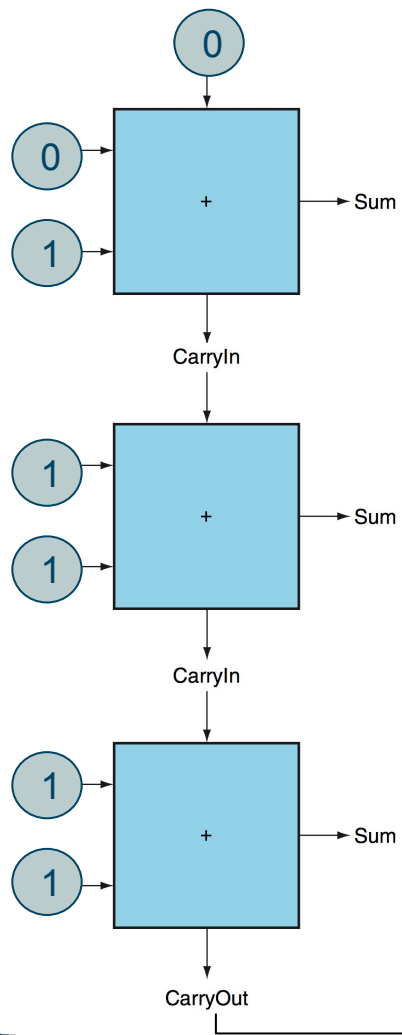
Ripple Carry Adder

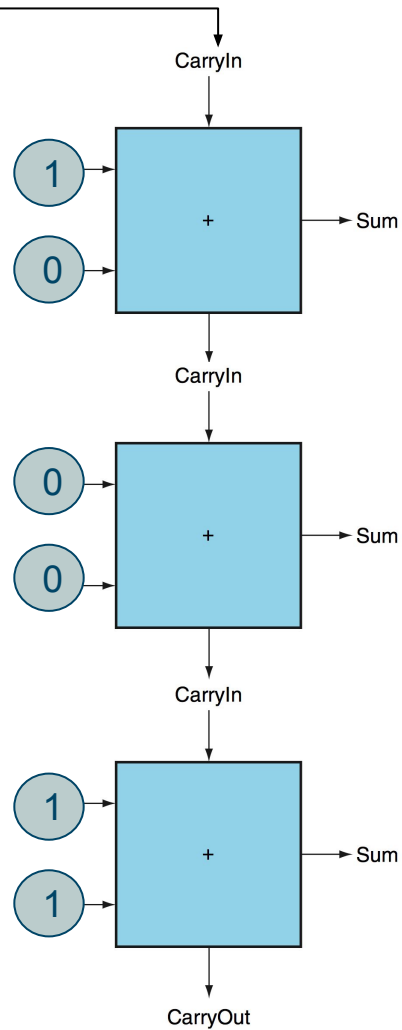
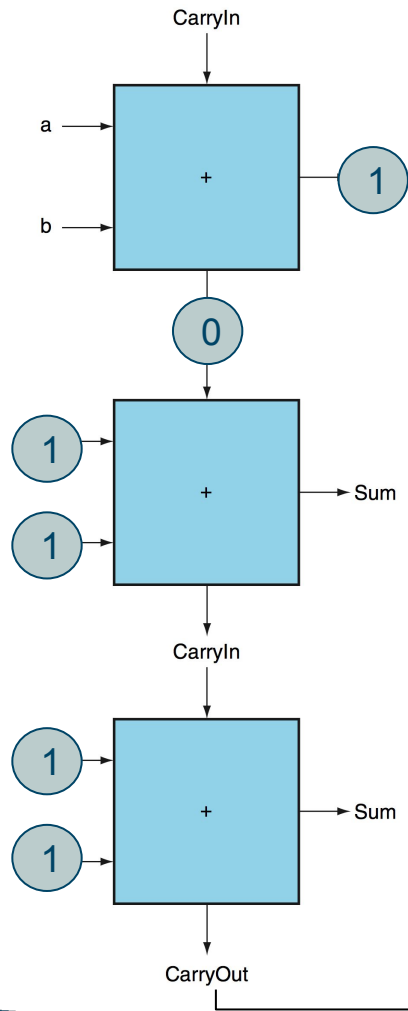
n-bit addition:

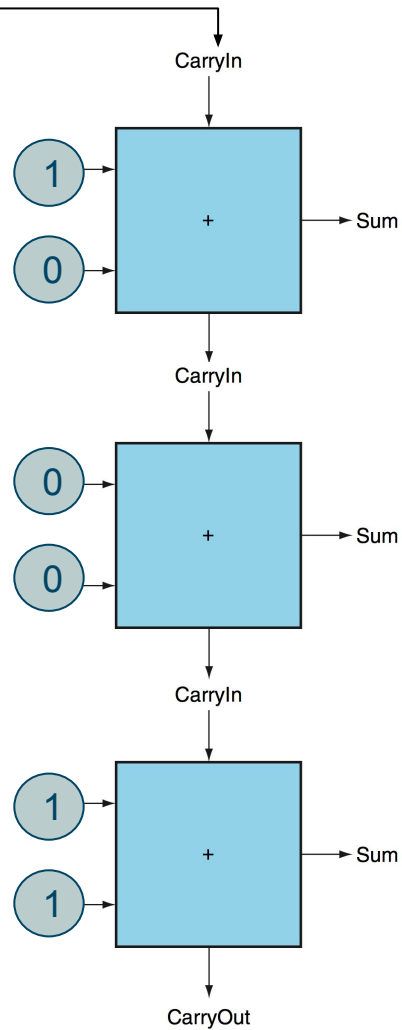
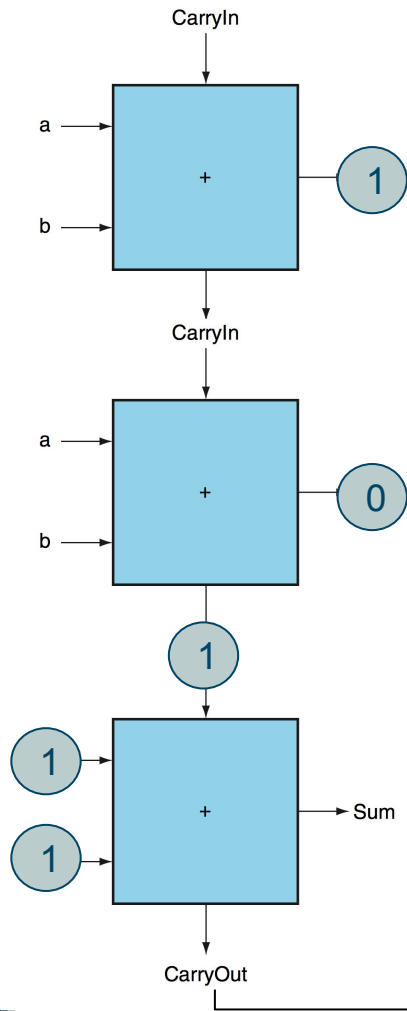
- Series of 1-bit full adders
- Carry ripples through addition = Slow!

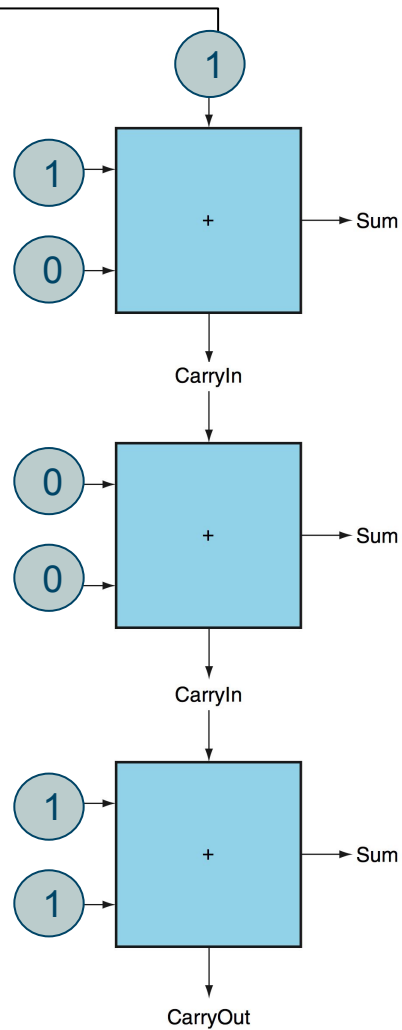
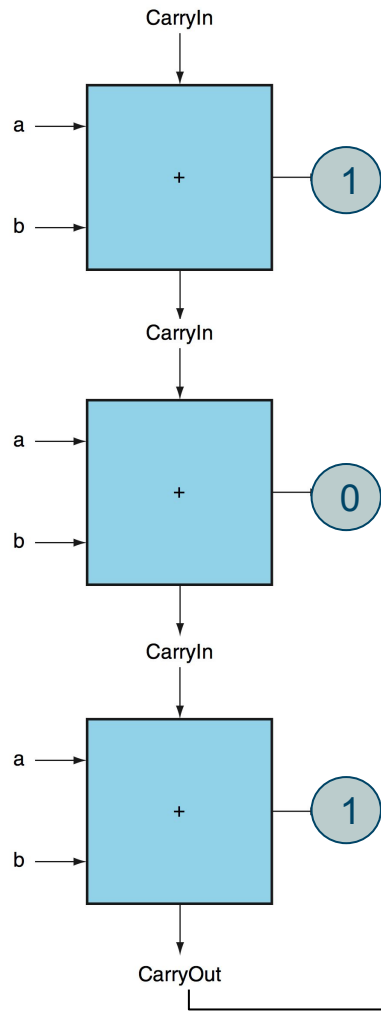


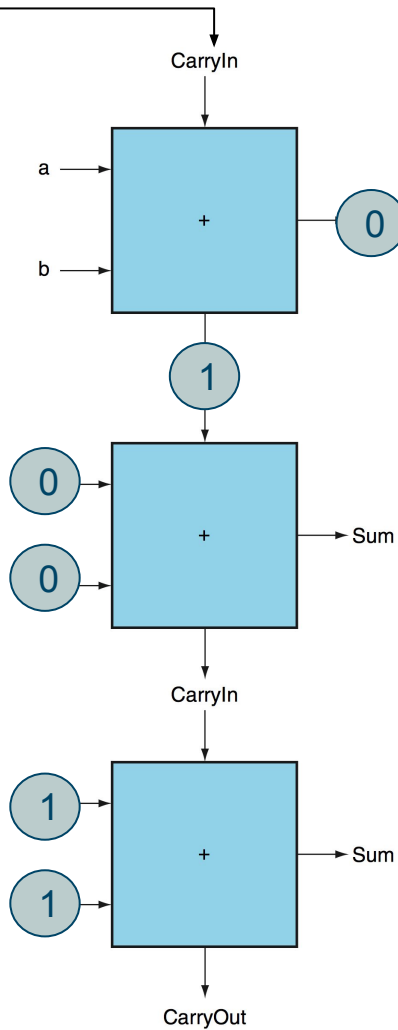
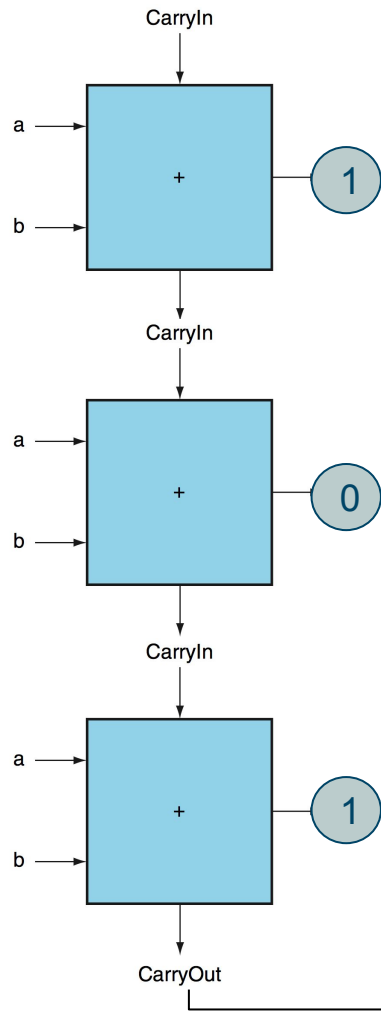


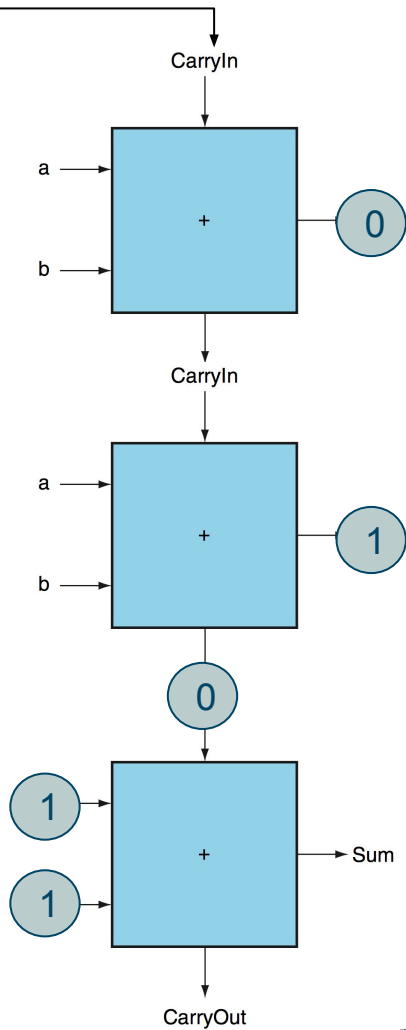
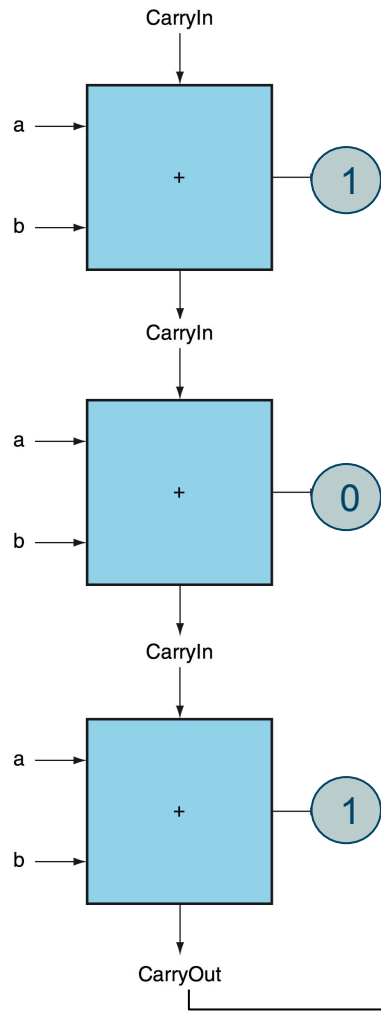


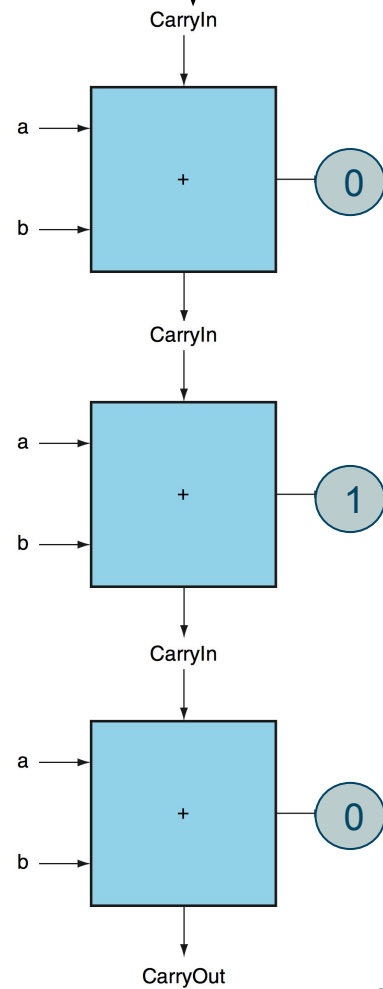
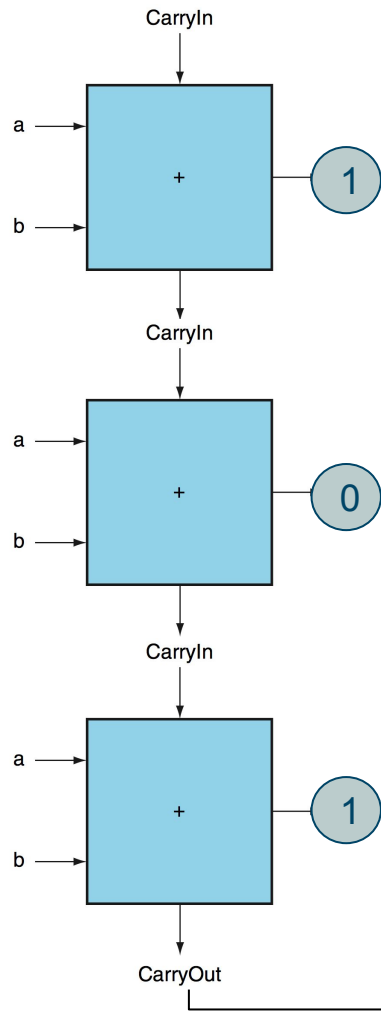












Calculate Carry

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

Fast Carry Adder using “Infinite Hardware”

$\text{CarryIn1} = \text{CarryOut0}$



Fast Carry Adder using “Infinite Hardware”

$$\text{CarryIn1} = (b0 \cdot \text{CarryIn0}) + (a0 \cdot \text{CarryIn0}) + (a0 \cdot b0)$$



Fast Carry Adder using “Infinite Hardware”

$$\text{CarryIn1} = (b_0 \cdot \text{CarryIn0}) + (a_0 \cdot \text{CarryIn0}) + (a_0 \cdot b_0)$$

$$\text{CarryIn2} = (b_1 \cdot \text{CarryIn1}) + (a_1 \cdot \text{CarryIn1}) + (a_1 \cdot b_1)$$



Fast Carry Adder using “Infinite Hardware”

$$c1 = (b0 \cdot c0) + (a0 \cdot c0) + (a0 \cdot b0)$$

$$c2 = (b1 \cdot c1) + (a1 \cdot c1) + (a1 \cdot b1)$$

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



Fast Carry Adder using “Infinite Hardware”

$$c1 = (b0 \cdot c0) + (a0 \cdot c0) + (a0 \cdot b0)$$

$$c2 = (b1 \cdot c1) + (a1 \cdot c1) + (a1 \cdot b1)$$

$$c2 = (a1 \cdot a0 \cdot b0) + (a1 \cdot a0 \cdot c0) + (a1 \cdot b0 \cdot c0) \\ + (b1 \cdot a0 \cdot b0) + (b1 \cdot a0 \cdot c0) + (b1 \cdot b0 \cdot c0) + \\ (a1 \cdot b1)$$

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

Fast Carry Adder using “Infinite Hardware”

$$c1 = (b0 \cdot c0) + (a0 \cdot c0) + (a0 \cdot b0)$$

$$c2 = (b1 \cdot c1) + (a1 \cdot c1) + (a1 \cdot b1)$$

$$c2 = (a1 \cdot a0 \cdot b0) + (a1 \cdot a0 \cdot c0) + (a1 \cdot b0 \cdot c0) \\ + (b1 \cdot a0 \cdot b0) + (b1 \cdot a0 \cdot c0) + (b1 \cdot b0 \cdot c0) + \\ (a1 \cdot b1)$$

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

→ “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?”

$$g_i = a_i \cdot b_i$$

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

| | | | | |
|---|-------|-------|-------|-------|
| | c_3 | c_2 | c_1 | c_0 |
| | a_3 | a_2 | a_1 | a_0 |
| + | b_3 | b_2 | b_1 | b_0 |
| | <hr/> | | | |
| | s_3 | s_2 | s_1 | s_0 |

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:

| | | | | |
|---|-------|-------|-------|-------|
| | c_3 | c_2 | c_1 | c_0 |
| | a_3 | a_2 | a_1 | a_0 |
| + | b_3 | b_2 | b_1 | b_0 |
| | <hr/> | | | |
| | s_3 | s_2 | s_1 | s_0 |

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:

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

| | | | | |
|---|-------|-------|-------|-------|
| | c_3 | c_2 | c_1 | c_0 |
| | a_3 | a_2 | a_1 | a_0 |
| + | b_3 | b_2 | b_1 | b_0 |
| | <hr/> | | | |
| | s_3 | s_2 | s_1 | s_0 |

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:

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

$$\begin{array}{rcccc} & c_3 & c_2 & c_1 & c_0 \\ & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline & s_3 & s_2 & s_1 & s_0 \end{array}$$



Carry Lookahead Adder

$$c_1 = g_0 + (p_0 \cdot c_0)$$

$$c_2 = g_1 + (p_1 \cdot g_0) \\ + (p_1 \cdot p_0 \cdot c_0)$$

$$c_3 = g_2 + (p_2 \cdot g_1) \\ + (p_2 \cdot p_1 \cdot g_0) \\ + (p_2 \cdot p_1 \cdot p_0 \cdot c_0)$$

$$c_4 = g_3 + (p_3 \cdot g_2) \\ + (p_3 \cdot p_2 \cdot g_1) \\ + (p_3 \cdot p_2 \cdot p_1 \cdot g_0) \\ + (p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0)$$

$$\begin{array}{rcccc} & c_3 & c_2 & c_1 & c_0 \\ & a_3 & a_2 & a_1 & a_0 \\ + & b_3 & b_2 & b_1 & b_0 \\ \hline s_3 & s_2 & s_1 & s_0 & \end{array}$$

Carry Lookahead Adder

| | |
|--------------|---------------|
| ???(?) | ???? |
| 0010 | 1110 |
| <u>+0010</u> | <u>+ 0111</u> |

Super Propagates and -Generates

Second Layer of Abstraction

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

Superpropagate P_i and supergenerate G_i :

$$P_0 = p_3 \cdot p_2 \cdot p_1 \cdot p_0$$

$$G_0 = g_3 + (p_3 \cdot g_2) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0)$$

Calculate C_i :

$$C_1 = G_0 + (P_0 \cdot c_0)$$

$$C_2 = G_1 + (P_1 \cdot G_0) + (P_1 \cdot P_0 \cdot c_0)$$

$$C_3 = \dots$$



8-bit Carry Lookahead Adder

