



Universiteit Antwerpen
| Faculteit Wetenschappen

Computerarchitectuur

Adders

Optellen (addition) decimale getallen

$$\begin{array}{r} 111 \\ 3986 \\ + 1889 \\ \hline 5875 \end{array}$$

Optellen (addition) decimale getallen

$$\begin{array}{r} 111 \\ 3986 \\ + 1889 \\ \hline 5875 \end{array}$$

A red box highlights the tens column of the addition, showing the sum of the digits 6 and 9, which is 15. To the right of the box, the text "= 15" is written in red.

Optellen (addition) decimale getallen

$$\begin{array}{r} 111 \\ 3986 \\ + 1889 \\ \hline 5875 \end{array}$$

The diagram illustrates the addition of 3986 and 1889. A red box highlights the 8 in 3986 and the 8 in 1889, with an equals sign and a red 15 next to it, indicating the sum of these two digits. A green arrow points from this 15 to the 5 in the tens place of the result, 5875, which is also circled in green, showing the carry-over process.

Optellen (addition) decimale getallen

$$\begin{array}{r} 111 \\ 3986 \\ + 1889 \\ \hline 5875 \end{array}$$

Diagram illustrating the addition of 3986 and 1889, showing a carry of 15 from the tens place to the hundreds place.

The diagram shows the addition of 3986 and 1889. A red box highlights the tens place (8 and 9) and the resulting carry of 15. A green arrow points from the carry of 15 to the hundreds place (1 and 8), indicating the carry-over.

Optellen (addition) binaire getallen

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

Optellen (addition) binaire getallen

CarryOut

||

0101110 = CarryIn

0010110 = A

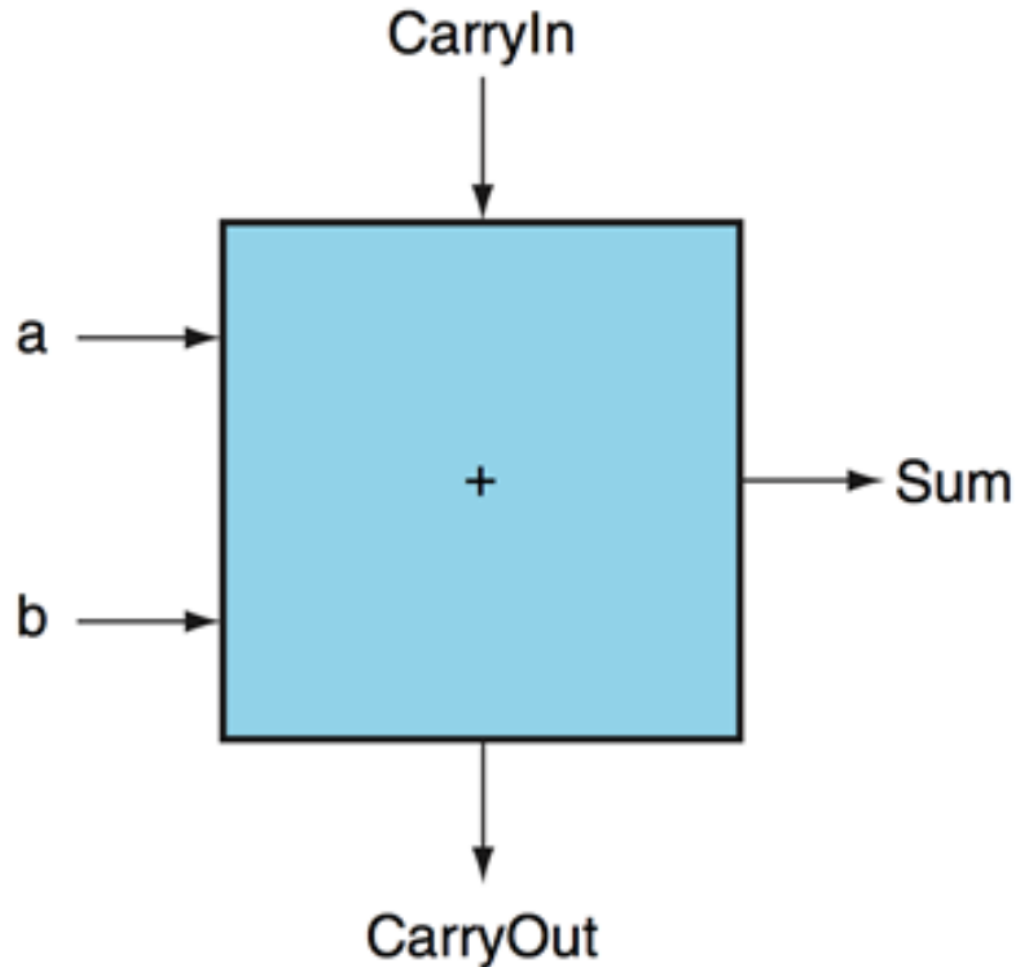
+ 0010011 = B

01010101 = Sum

Optellen (addition) binaire getallen

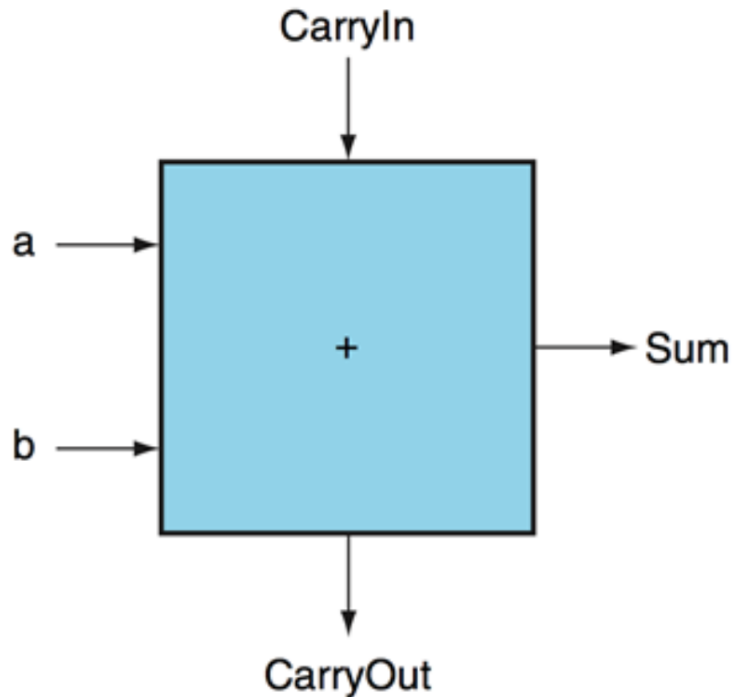
Inputs			Outputs		Comments
A	B	CarryIn	CarryOut	Sum	
0	0	0	0	0	$0 + 0 + 0 = 00_2$
0	0	1	0	1	$0 + 0 + 1 = 01_2$
0	1	0	0	1	$0 + 1 + 0 = 01_2$
0	1	1	1	0	$0 + 1 + 1 = 10_2$
1	0	0	0	1	$1 + 0 + 0 = 01_2$
1	0	1	1	0	$1 + 0 + 1 = 10_2$
1	1	0	1	0	$1 + 1 + 0 = 10_2$
1	1	1	1	1	$1 + 1 + 1 = 11_2$

1-Bit Full Adder



- 1-bit optelling (addition):
 - **CarryOut** output is 1 wanneer **ten minste twee inputs** gelijk zijn aan 1
 - **Sum** output is 1 wanneer **exact één input** gelijk is aan 1, of wanneer **alle drie de inputs** gelijk zijn aan 1

1-Bit Full Adder



- 1-bit optelling (addition):
 - CarryOut** output is 1 wanneer **ten minste twee inputs** gelijk zijn aan 1

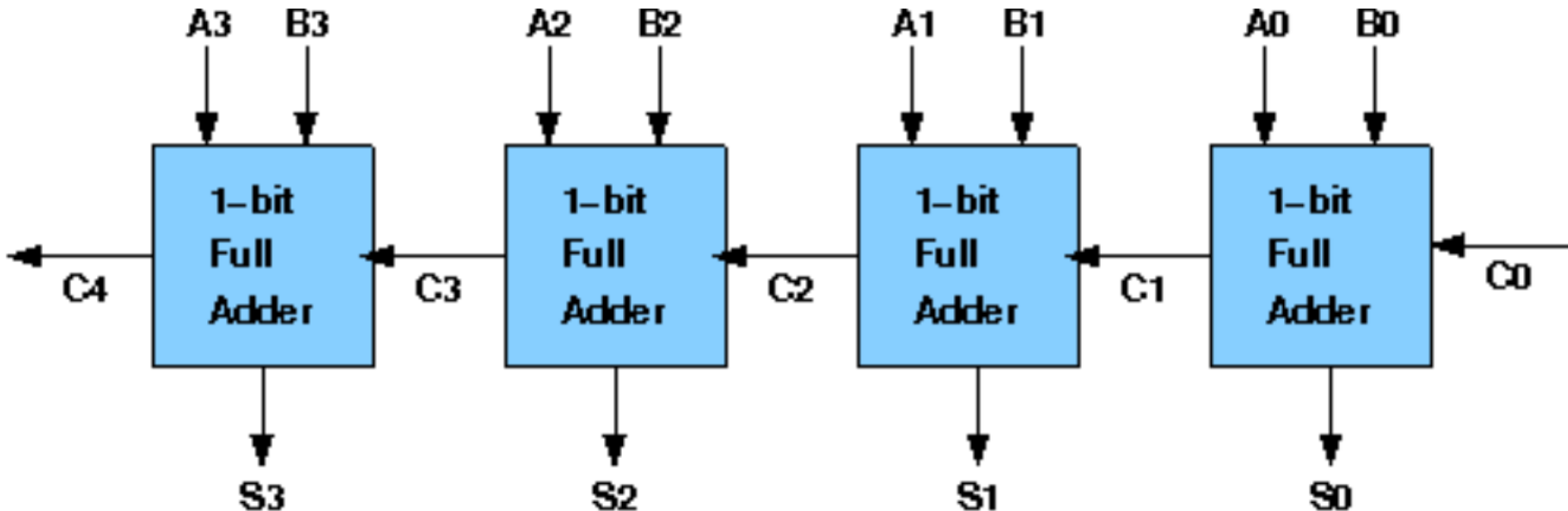
$$\text{CarryOut} = (b \cdot \text{CarryIn}) + (a \cdot \text{CarryIn}) + (a \cdot b)$$

- Sum** output is 1 wanneer **exact één input** gelijk is aan 1, of wanneer **alle drie de inputs** gelijk zijn aan 1

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

Ripple Carry Adder

- **N-bit optelling (addition):**
 - Series van 1-bit full adders
 - Carry vloeit (ripples) door de optelling = Traag!



Bereken Carry

????????
00101110
+ 00100111

Propagate en Generate

- **Generate: “Wanneer genereren a_i en b_i een carry?”**
 - Er wordt een CarryOut gegenereerd onafhankelijk van wat de CarryIn is?
 - $g_i = a_i \cdot b_i$
- **Propagate: “Wanneer propageren a_i en b_i een carry?”**
 - Wanneer wordt een CarryOut gegenereerd als er een CarryIn is?
 - $p_i = a_i + b_i$
- **CarryIn:**
 - $c_{i+1} = g_i + p_i \cdot c_i$

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

Carry Lookahead Adder

- $c_1 = g_0 + (p_0 \cdot c_0)$
- $c_2 = g_1 + (p_1 \cdot g_0) + (p_1 \cdot p_0 \cdot c_0)$
- $c_3 = g_2 + (p_2 \cdot g_1) + (p_2 \cdot p_1 \cdot g_0) + (p_2 \cdot p_1 \cdot p_0 \cdot c_0)$
- $c_4 = g_3 + (p_3 \cdot g_2) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0) + (p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0)$

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

Carry Lookahead Adder

????	????
0010	1110
<u>+ 0010</u>	<u>+ 0111</u>

Super Propagates en Super Generates

- Bereken CarryIn van elke 4-bit carry-lookahead adder
- Super propagate P_i en super generate G_i :
 - $P_0 = p_3 \cdot p_2 \cdot p_1 \cdot p_0$
 - $G_0 = g_3 + (p_3 \cdot g_2) + (p_3 \cdot p_2 \cdot g_1) + (p_3 \cdot p_2 \cdot p_1 \cdot g_0)$
- Bereken 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

