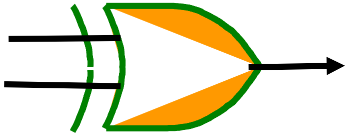


The last two gates

XOR - XNOR

… Two more gates

- ✓ XOR
- ✓ XNOR

… two more gates

- ✓ XOR
- ✓ XNOR

…are introduced to facilitate the design of binary ADDERS

OR gate

A	B	OR gate
0	0	0
0	1	1
1	0	1
1	1	1

XOR (eXclusiveOR) gate

A	B	OR gate	XOR gate
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

XOR (eXclusiveOR) gate

⊕

A	B	OR gate	XOR gate
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

$$A \textbf{ XOR } B = A \oplus B$$

$$= \overline{A}\,B + A\,\overline{B}$$

It produces a high output whenever the two inputs are at opposite levels

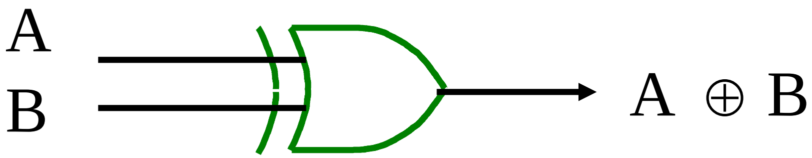
XOR (eXclusiveOR) gate

A	B	OR gate	XOR gate
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

A **XOR** B $= A \oplus B$

$$= \bar{A}B + A\bar{B}$$

It produces a high output whenever the two inputs are at opposite levels



Another gate … XNOR

$$\overline{A \oplus B} = ?$$

XNOR (eXclusiveNOR) gate

A	B	XOR gate	XNOR gate
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

⊙

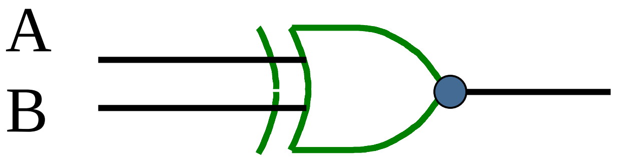
XNOR (eXclusiveNOR) gate ⊙

A	B	XOR gate	XNOR gate
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

⊙

$$A \text{ XNOR } B = \overline{A \oplus B}$$

$$= \overline{A}\,\overline{B} + A\,B$$



$$\overline{A \oplus B} = A \odot B$$

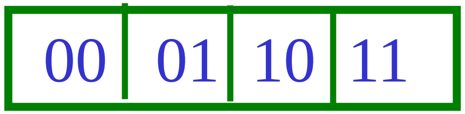
Parity generator module … Data

- The parity generators are used in data communications as error detectors

Parity generator module … Data

- The parity generators are used in data communications as error detectors

A	B	P
0	0	
0	1	
1	0	
1	1	



00 01 10 11

DATA

Design a 2-bit Even Parity generator module

- The parity generators are used in data communications as error detectors

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

00	01	10	11

DATA

Design a 2-bit Even Parity generator module

- The parity generators are used in data communications as error detectors

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

$$P = A \oplus B$$

00	01	10	11

DATA

Data Communication with error detection

- The parity generators are used in data communications as error detectors

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

$$P = A \oplus B$$

00	01	10	11

DATA

Data Communication with error detection

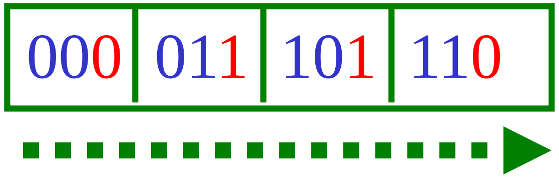
- The parity generators are used in data communications as error detectors

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

$P = A \oplus B$

00	01	10	11

DATA



000	011	101	110

Transmit … DATA + ParityBits

Total we have 2^4 = 16 gates …

AB	0	AND		A		B	XOR	OR	NOR	XNOR	NOTB		NOTA		NAND	1
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Only 7 gates are useful

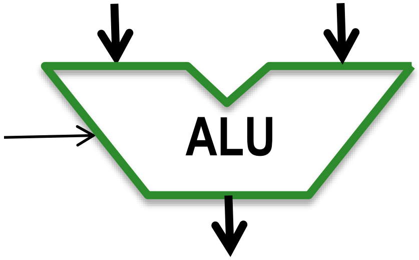
AND, OR, NOT, NAND, NOR, XOR, XNOR

XOR & XNOR: Gate delay

Gate Delay (nsec)	
4.2	(XOR) with 2 inputs
3.2	(XNOR) with 2 inputs

Binary Adders

- The binary Adder is part of the Arithmetic logic Unit (ALU)



Problem: 1-bit binary adder

- Design a binary logic circuit to Add 2 binary digits: (a, b) …… (1-bit Adder).

$$\begin{array}{r} 1 \\ +\ 0 \\ \hline \end{array} \qquad \begin{array}{r} a \\ +\ b \\ \hline \end{array}$$

1-bit adder: Truth table

a	b	c	s
0	0		
0	1		
1	0		
1	1		

1-bit adder: Truth table

a	b	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

1-bit Adder: Carry logic equation

a	b	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Therefore,

$$C = a\,b$$

1-bit Adder: Sum logic equation

a	b	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Therefore,

$$C = a\,b$$

$$S = \overline{a}\,b + a\,\overline{b}$$

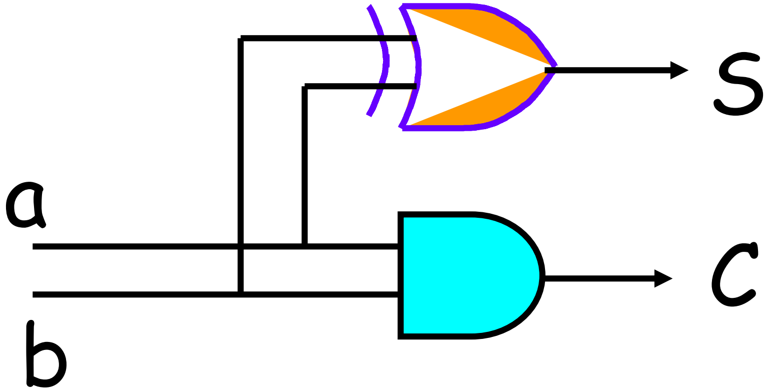
$$= a \oplus b$$

1-bit Adder: Logic circuit

$$C = a\,b$$

$$S = \overline{a}\,b + a\,\overline{b}$$

$$= a \oplus b$$

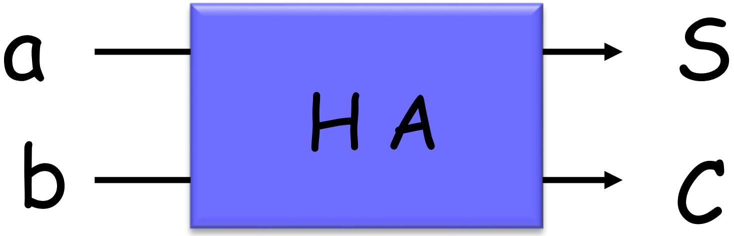


1-bit Adder: Graphical symbol

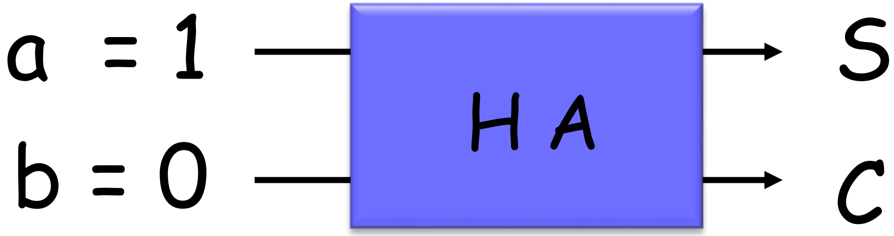
$$C = a\,b$$

$$S = \overline{a}\,b + a\,\overline{b}$$

$$= a \oplus b$$



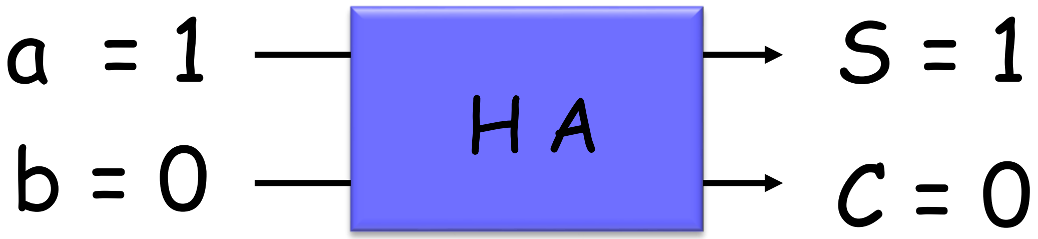
Example-1



Example-1: Addition

$$\begin{array}{r} 1 \\ +\ 0 \\ \hline \end{array}$$

0 is the carry and 1 is the sum



a = 1

b = 0

H A

S = 1

C = 0

Example-2

$$\begin{array}{r} 11 \\ +\ 01 \\ \hline ?? \end{array}$$

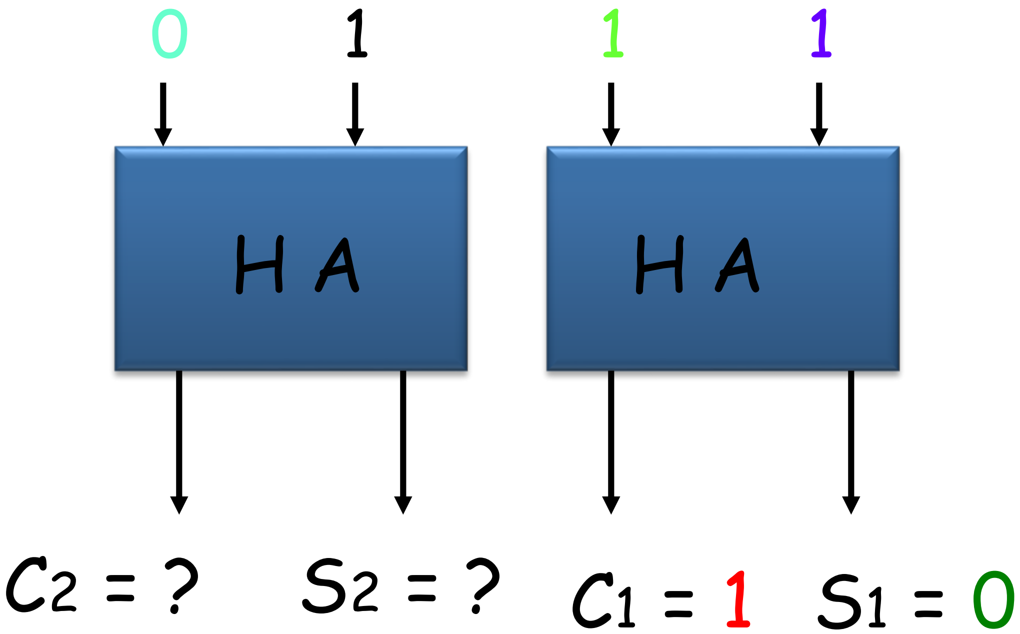
Example-2: Addition

$$\begin{array}{r} \color{red}{1} \\ 11 \\ +\ 01 \\ \hline 100 \end{array}$$

What about the logic circuit ?

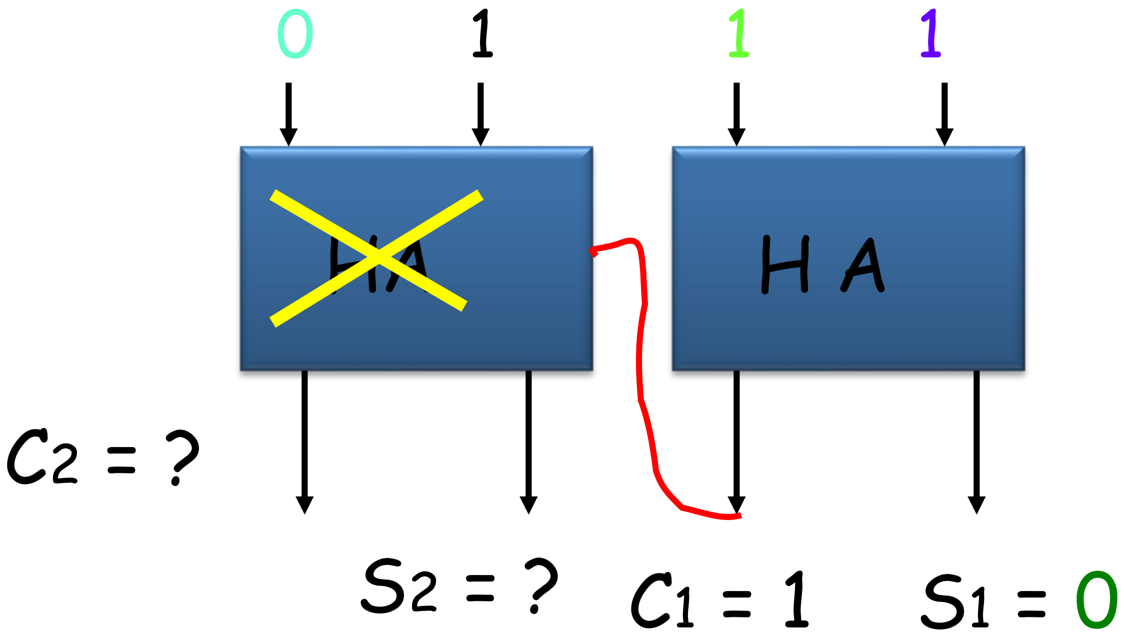
Example-2: Logic circuit

```
  1
  11
+ 01
----
 100
```



Example-2: Logic circuit

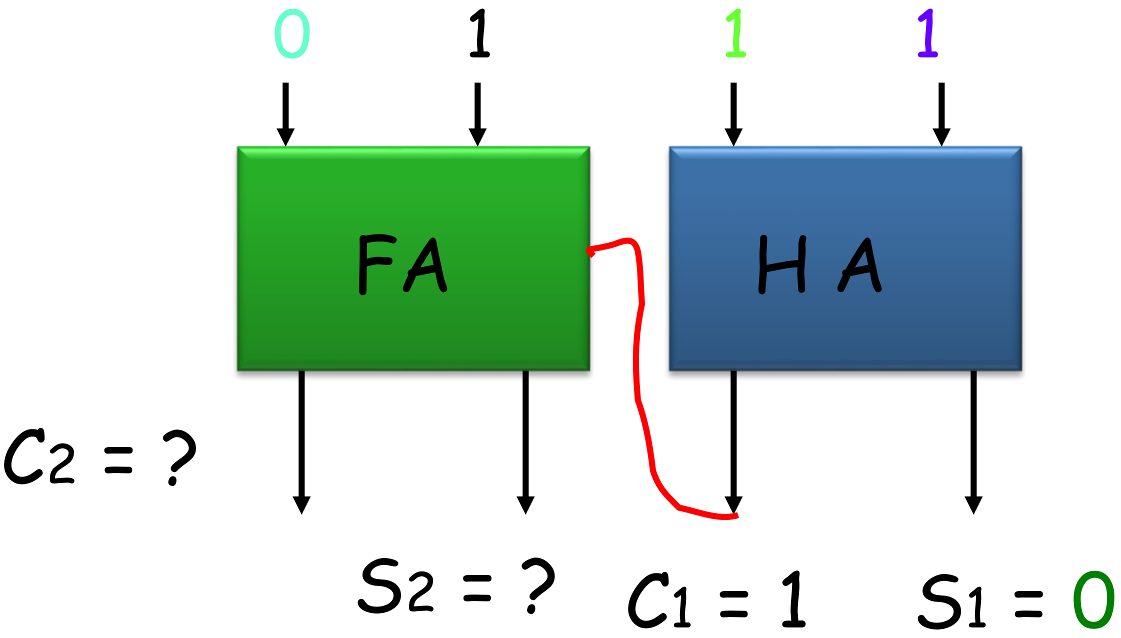
```
  1
  11
+ 01
----
 100
```



Therefore we need logic adders with three inputs = ???

Full-adder ?

```
   1
   11
 + 01
 ----
  100
```



Therefore we need logic adders with three inputs = ***Full Adders***

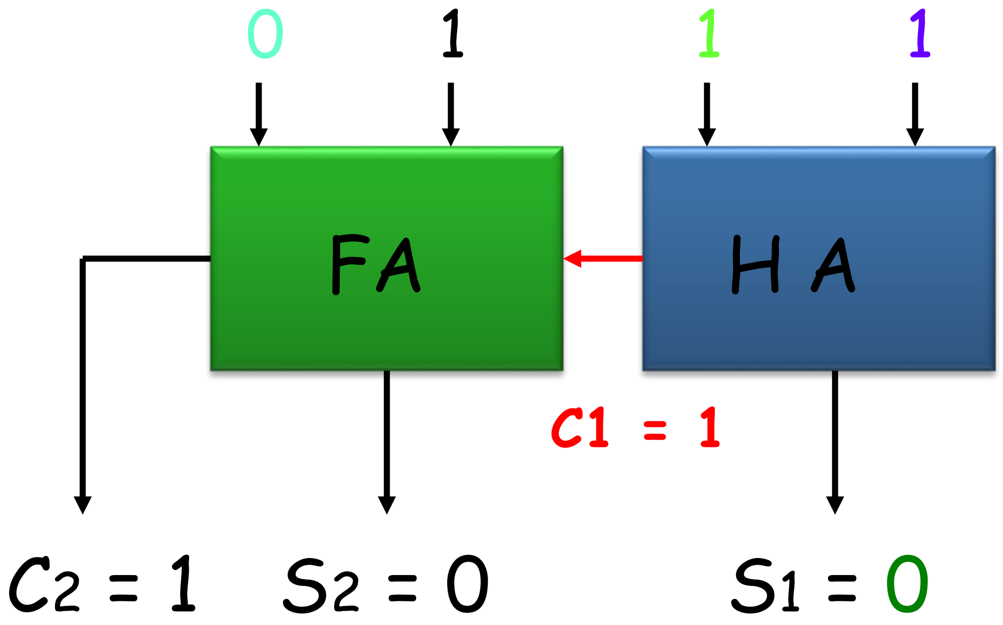
Result

1

11

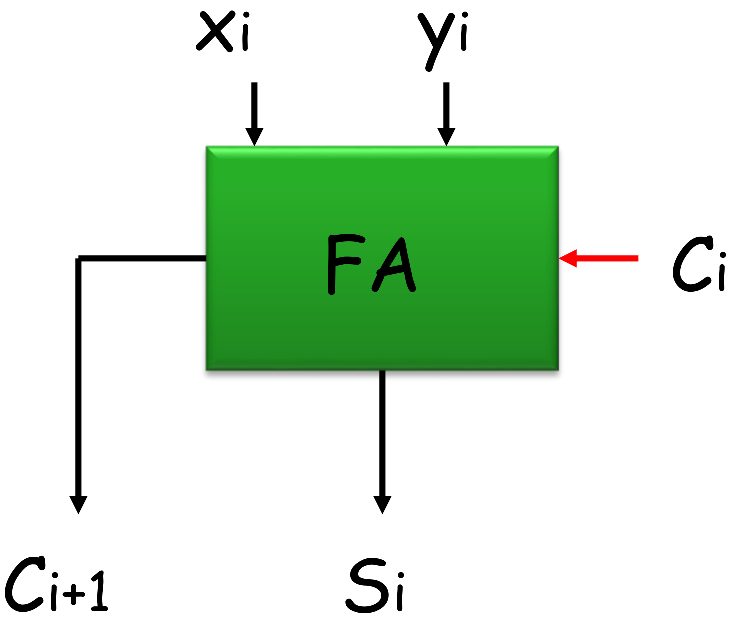
+ 01

100



Therefore we need logic adders with three inputs = ***Full Adders***

Design a Full-adder



Full-adder: Truth table

C_i	x_i	y_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Full-adder: Truth table

C_i	x_i	y_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Full-adder: Logic equations

C_i	x_i	y_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$C_{i+1} = \overline{C_i}\, x_i\, y_i + C_i\, \overline{x_i}\, y_i + C_i\, x_i\, \overline{y_i} + C_i\, x_i\, y_i$$

$$S_i = \overline{C_i}\, \overline{x_i}\, y_i + \overline{C_i}\, x_i\, \overline{y_i} + C_i\, \overline{x_i}\, \overline{y_i} + C_i\, x_i\, y_i$$

Full-adder: Simplification … C_{i+1}

$$C_{i+1} = \overline{C_i}\, x_i\, y_i + C_i\, \overline{x_i}\, y_i + C_i\, x_i\, \overline{y_i} + C_i\, x_i\, y_i$$

Full-adder: Simplification … C_{i+1}

$$C_{i+1} = \overline{C_i}\, x_i\, y_i + C_i\, \overline{x_i}\, y_i + C_i\, x_i\, \overline{y_i} + C_i\, x_i\, y_i$$

$$= C_i(\overline{x_i}\, y_i + x_i\, \overline{y_i}) + x_i\, y_i\, (\overline{C_i} + C_i)$$

Full-adder: Simplification … C_{i+1}

$$C_{i+1} = \overline{C_i}\, x_i\, y_i + C_i\, \overline{x_i}\, y_i + C_i\, x_i\, \overline{y_i} + C_i\, x_i\, y_i$$

$$= C_i(\overline{x_i}\, y_i + x_i\, \overline{y_i}) + x_i\, y_i\, (\overline{C_i} + C_i)$$

$$\boxed{C_{i+1} = C_i(x_i \oplus y_i) + x_i\, y_i}$$

Full-adder: Simplification … S_i

$$S_i = \overline{C_i}\,\overline{x_i}\,y_i + \overline{C_i}\,x_i\,\overline{y_i} + C_i\,\overline{x_i}\,\overline{y_i} + C_i\,x_i\,y_i \quad ?$$

Cixi \ yi	0	1
00		1
01	1	
11		1
10	1	

Full-adder: Simplification … S_i

$$S_i = \overline{C_i}\,\overline{x_i}\,y_i + \overline{C_i}\,x_i\,\overline{y_i} + C_i\,\overline{x_i}\,\overline{y_i} + C_i\,x_i\,y_i = C_i \oplus x_i \oplus y_i$$

Why ?

Cixi \ yi	0	1
00		1
01	1	
11		1
10	1	

Proof

$$S_i = \underbrace{\overline{C_i}\,\overline{x_i}\,y_i + \overline{C_i}\,x_i\,\overline{y_i}} + \underbrace{C_i\,\overline{x_i}\,\overline{y_i} + C_i\,x_i\,y_i} \overset{?}{=} C_i \oplus x_i \oplus y_i$$

Proof

$$S_i = \underbrace{\overline{C_i}\,\overline{x_i}\,y_i + \overline{C_i}\,x_i\,\overline{y_i}} + \underbrace{C_i\,\overline{x_i}\,\overline{y_i} + C_i\,x_i\,y_i} \overset{?}{=} C_i \oplus x_i \oplus y_i$$

$$S_i = \overline{C_i}(x_i \oplus y_i) + C_i(x_i\,y_i + \overline{x_i}\,\overline{y_i})$$

Proof

$$S_i = \underbrace{\overline{C_i}\,\overline{x_i}\,y_i + \overline{C_i}\,x_i\,\overline{y_i}} + \underbrace{C_i\,\overline{x_i}\,\overline{y_i} + C_i\,x_i\,y_i} \overset{?}{=} C_i \oplus x_i \oplus y_i$$

$$S_i = \overline{C_i}(x_i \oplus y_i) + C_i(x_i\,y_i + \overline{x_i}\,\overline{y_i})$$

$$S_i = \overline{C_i}(x_i \oplus y_i) + C_i\overline{(x_i \oplus y_i})$$

Done

$$S_i = \underbrace{\overline{C_i}\,\overline{x_i}\,y_i + \overline{C_i}\,x_i\,\overline{y_i}} + \underbrace{C_i\,\overline{x_i}\,\overline{y_i} + C_i\,x_i\,y_i} = C_i \oplus x_i \oplus y_i$$

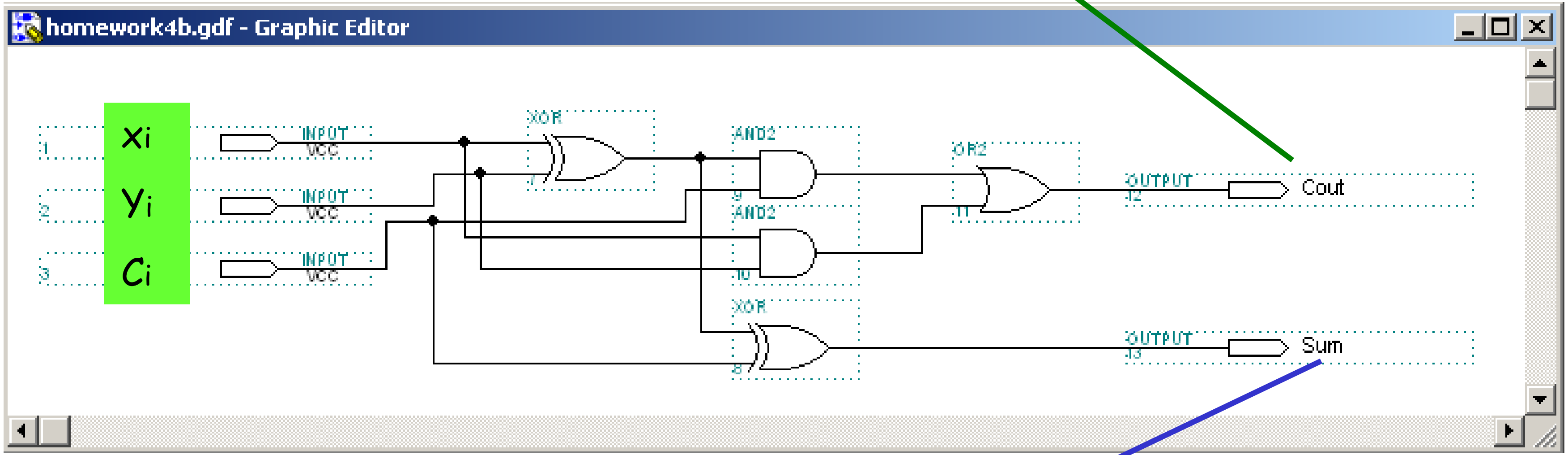
$$S_i = \quad \overline{C_i}(\,x_i \oplus y_i) \quad + \quad C_i(x_i\,y_i + \overline{x_i}\,\overline{y_i})$$

$$S_i = \quad \overline{C_i}(\,\underbrace{x_i \oplus y_i}) \quad + \quad C_i\overline{(\underbrace{x_i \oplus y_i})} \quad =$$

$$\overline{A}\;\;B \quad + \quad A\;\;\overline{B} \quad = \quad A \oplus B$$

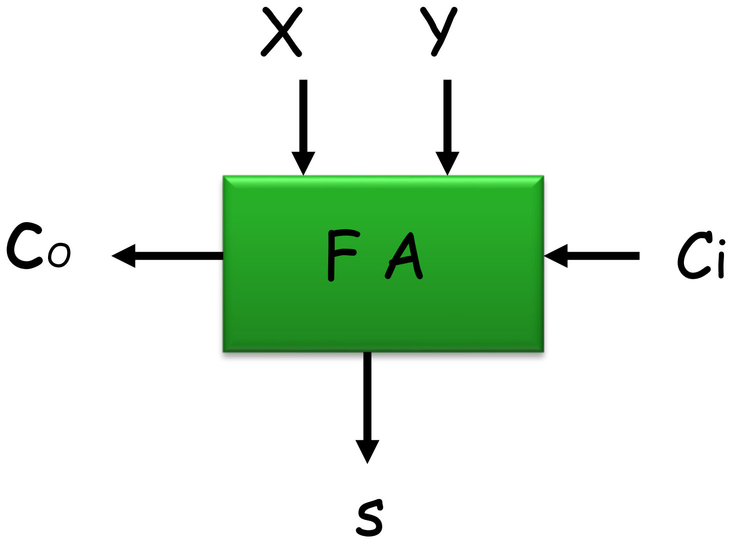
Full-adder … VHDL

$C_{i+1} = C_i(x_i \oplus y_i) + x_i y_i$

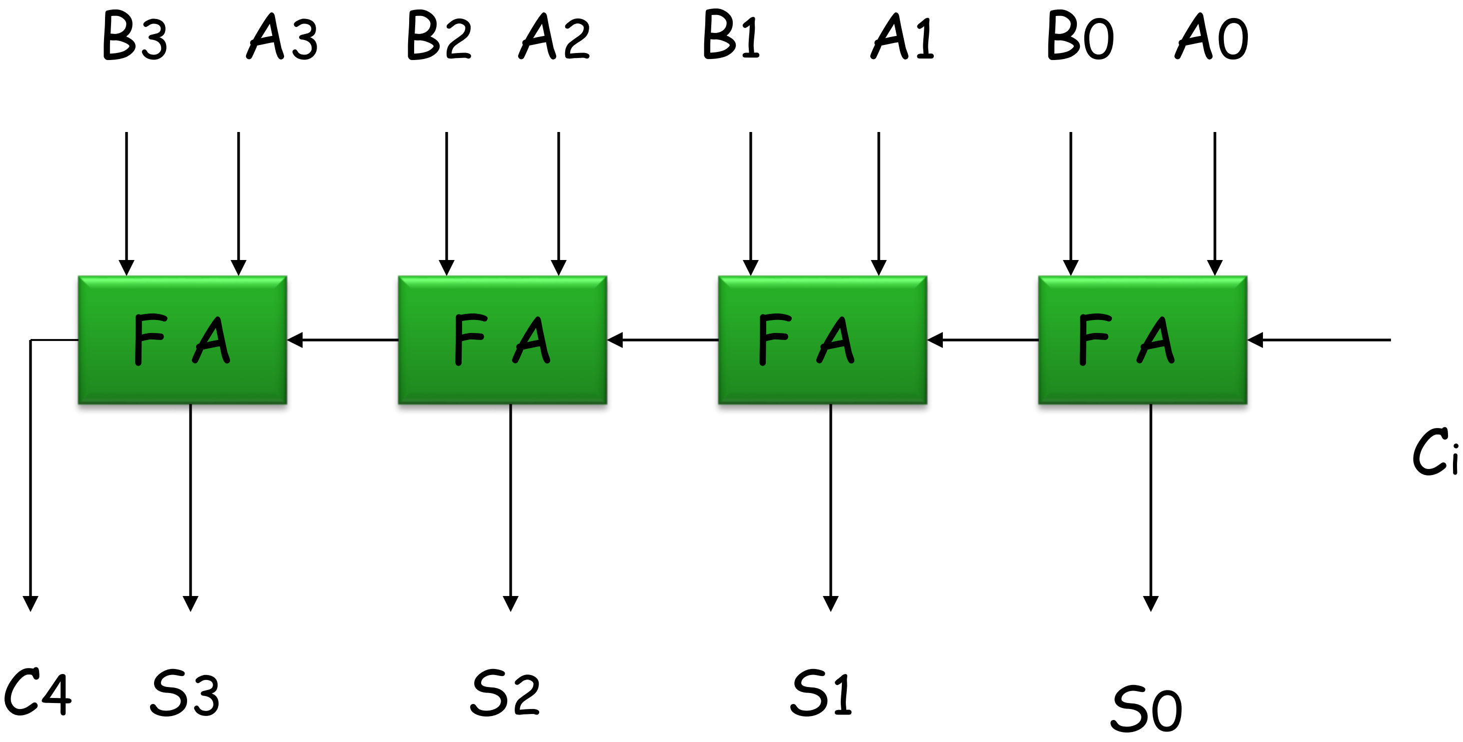


$S_i = C_i \oplus x_i \oplus y_i$

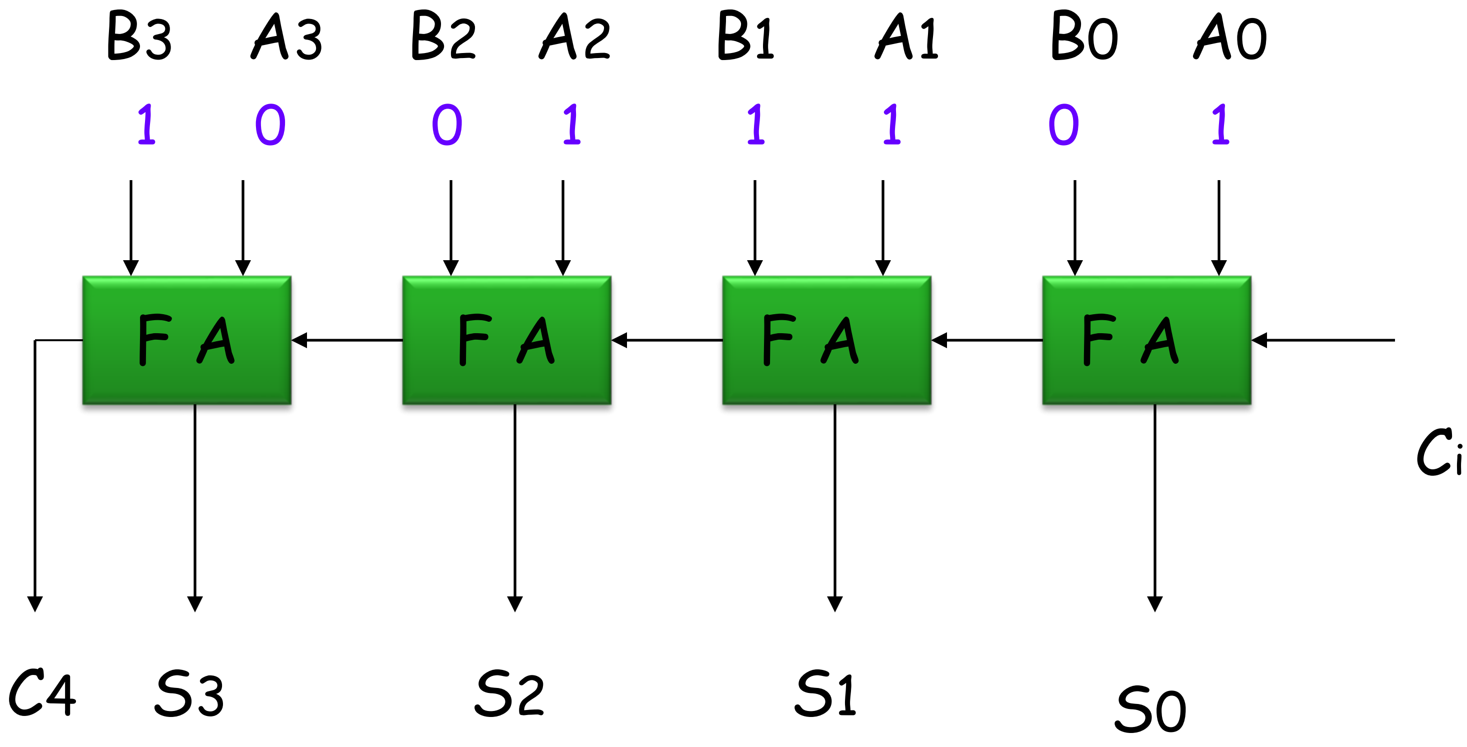
1-bit Full-adder



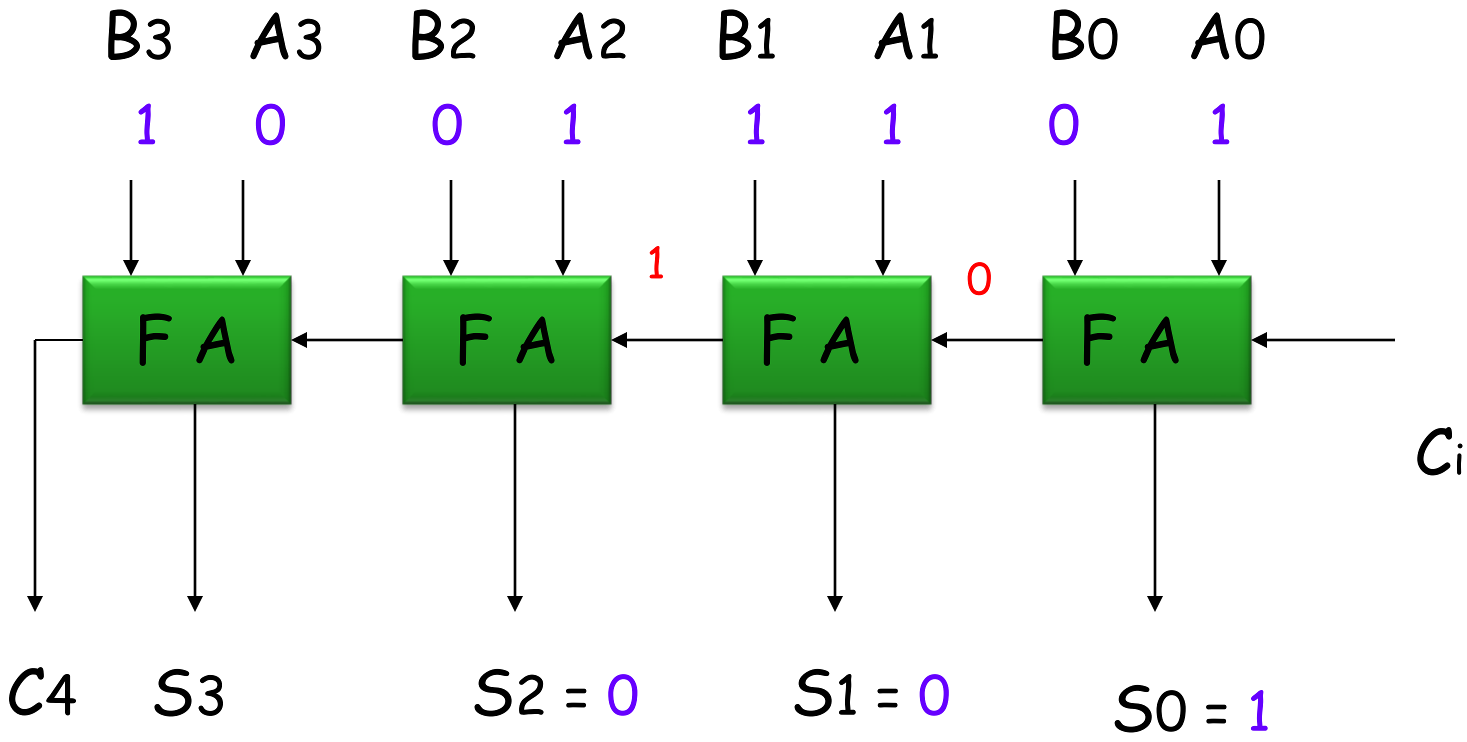
4-bit adder



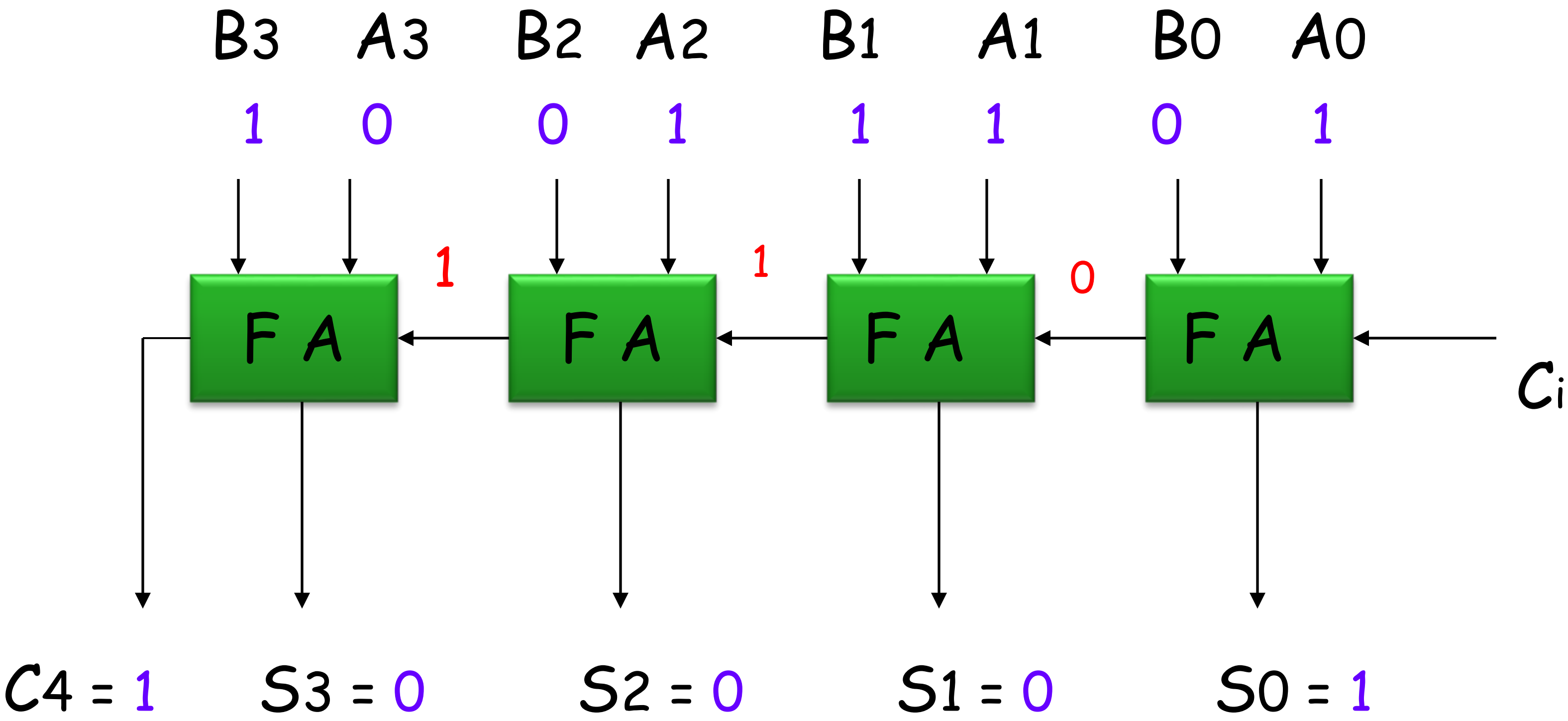
4-bit adder example



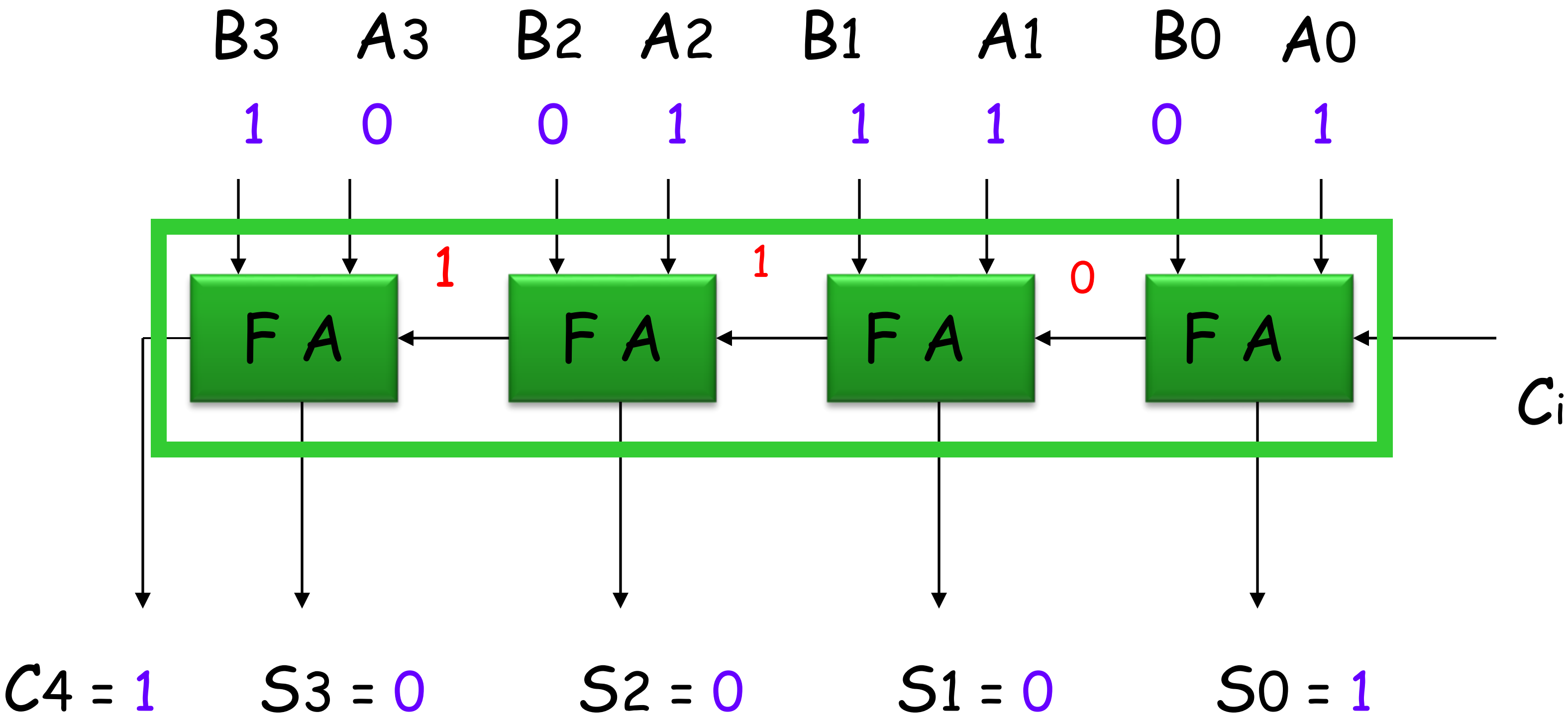
4-bit adder example



4-bit adder example



4-bit adder example



B3 A3 B2 A2 B1 A1 B0 A0

1 0 0 1 1 1 0 1

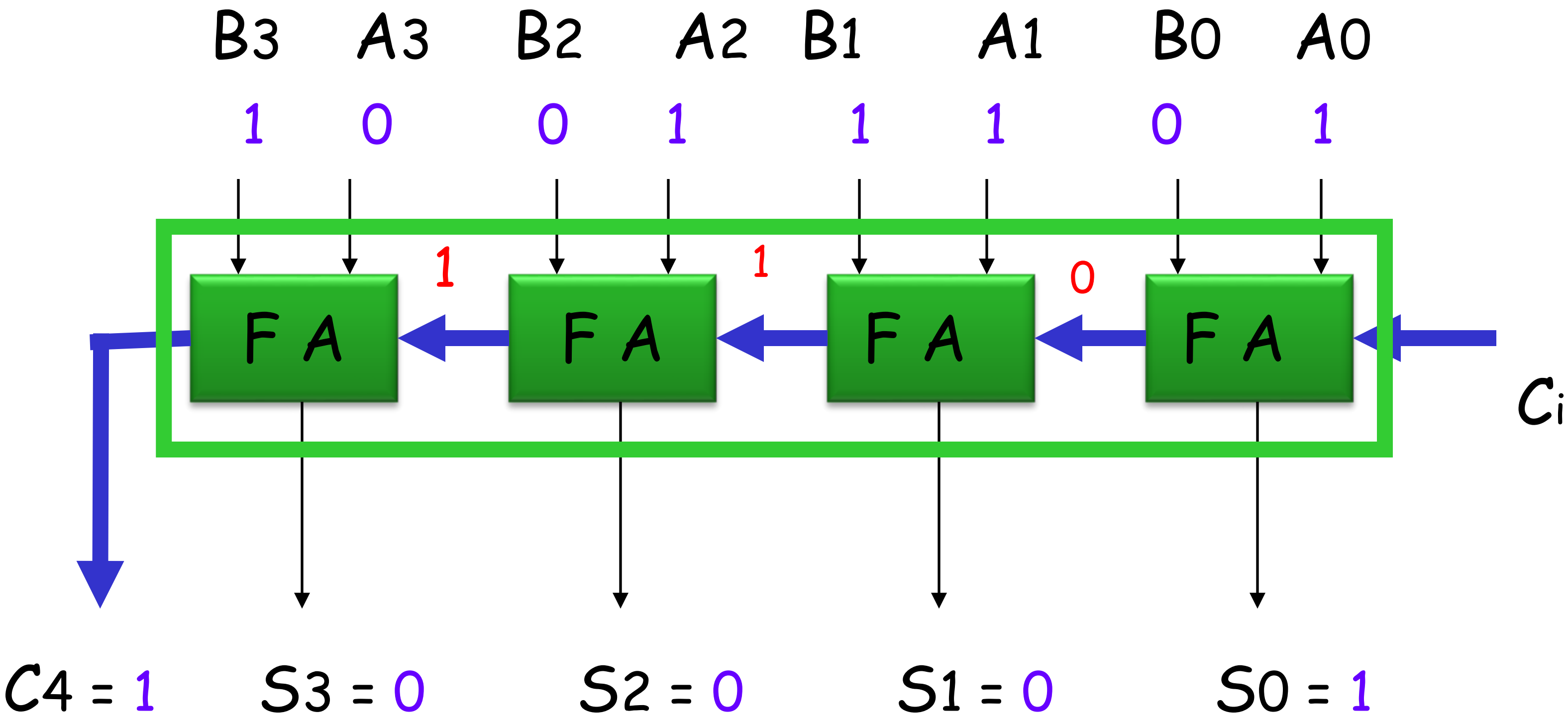
F A F A F A F A

1 1 0

Ci

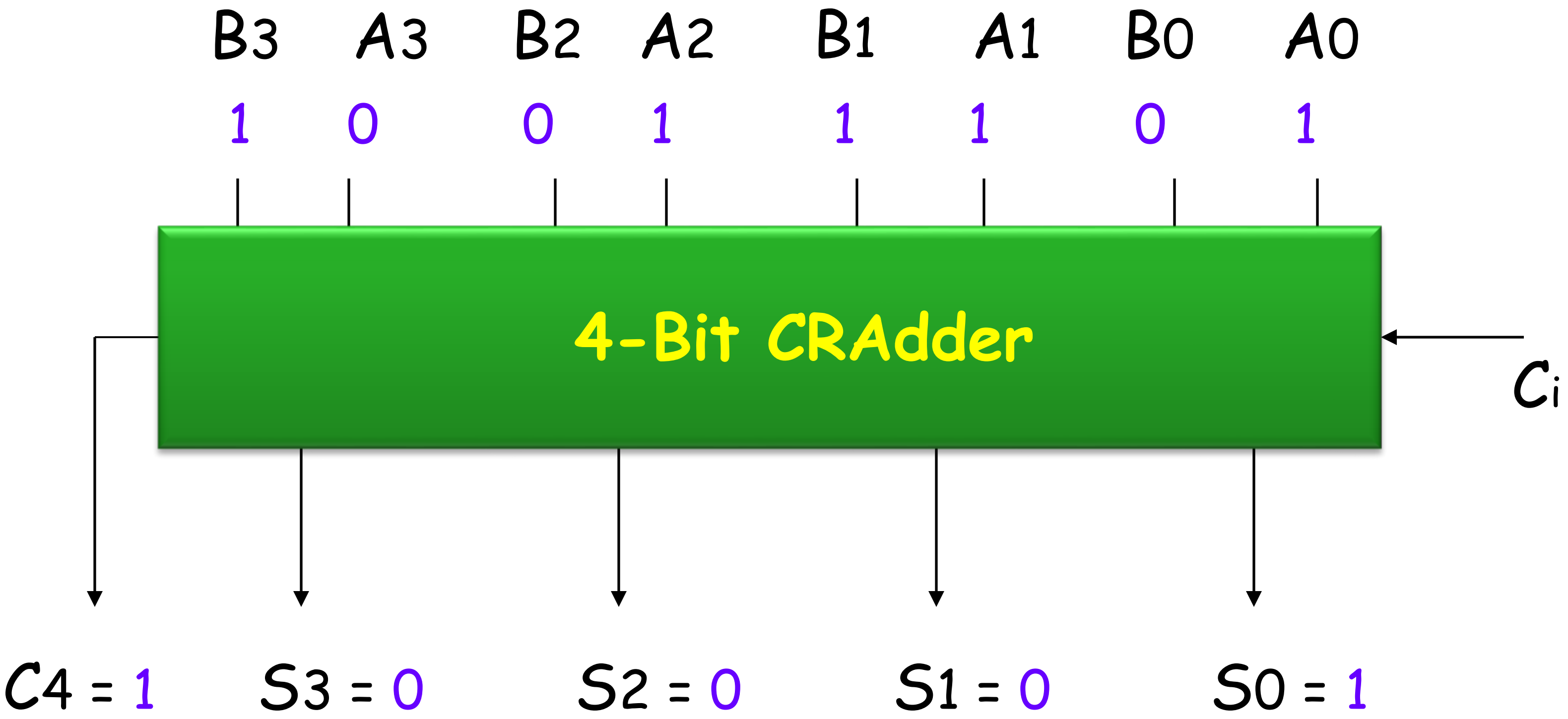
C4 = 1 S3 = 0 S2 = 0 S1 = 0 S0 = 1

4-bit Carry Ripple Adder (CRA)

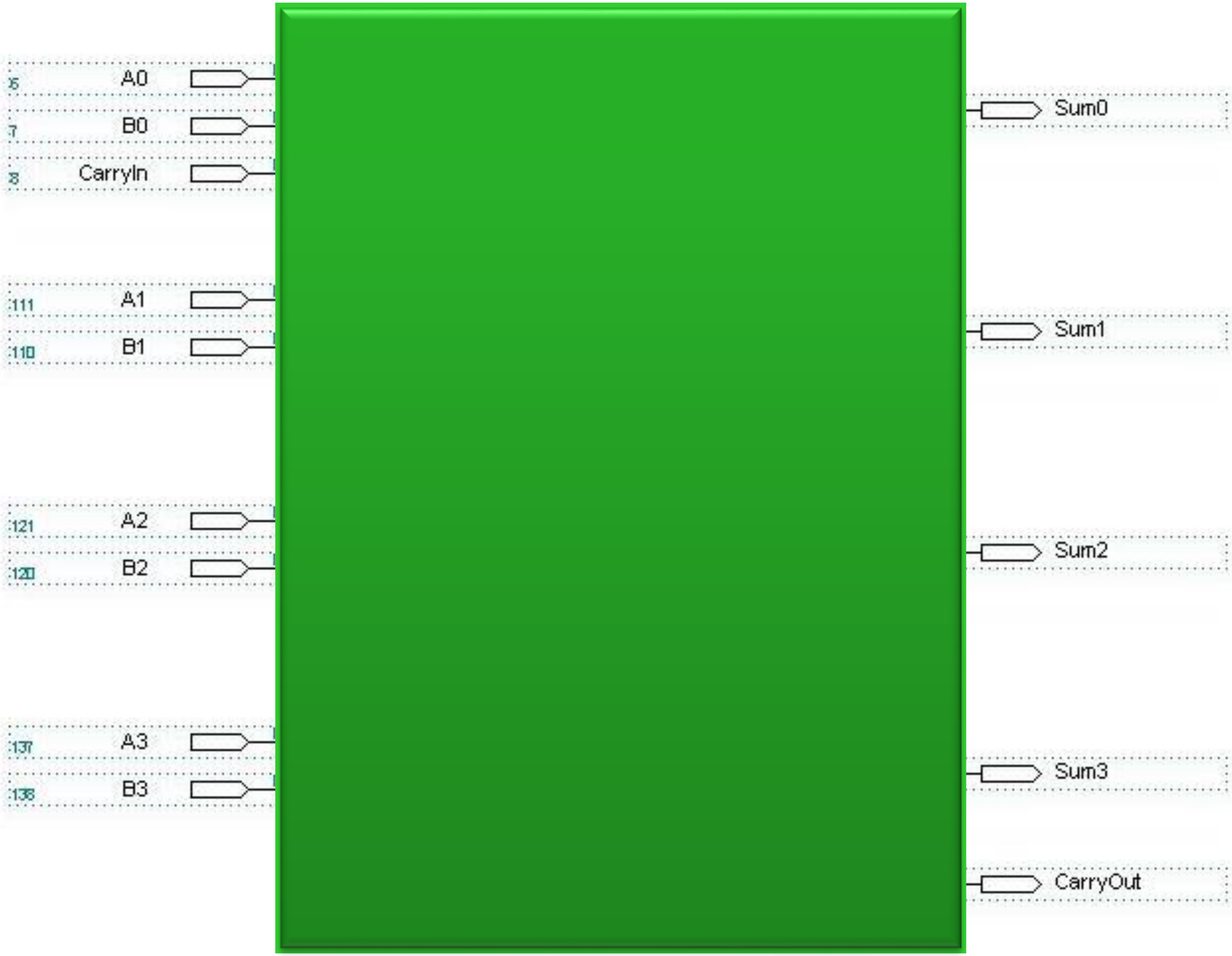


The carry "ripples" through the full adders

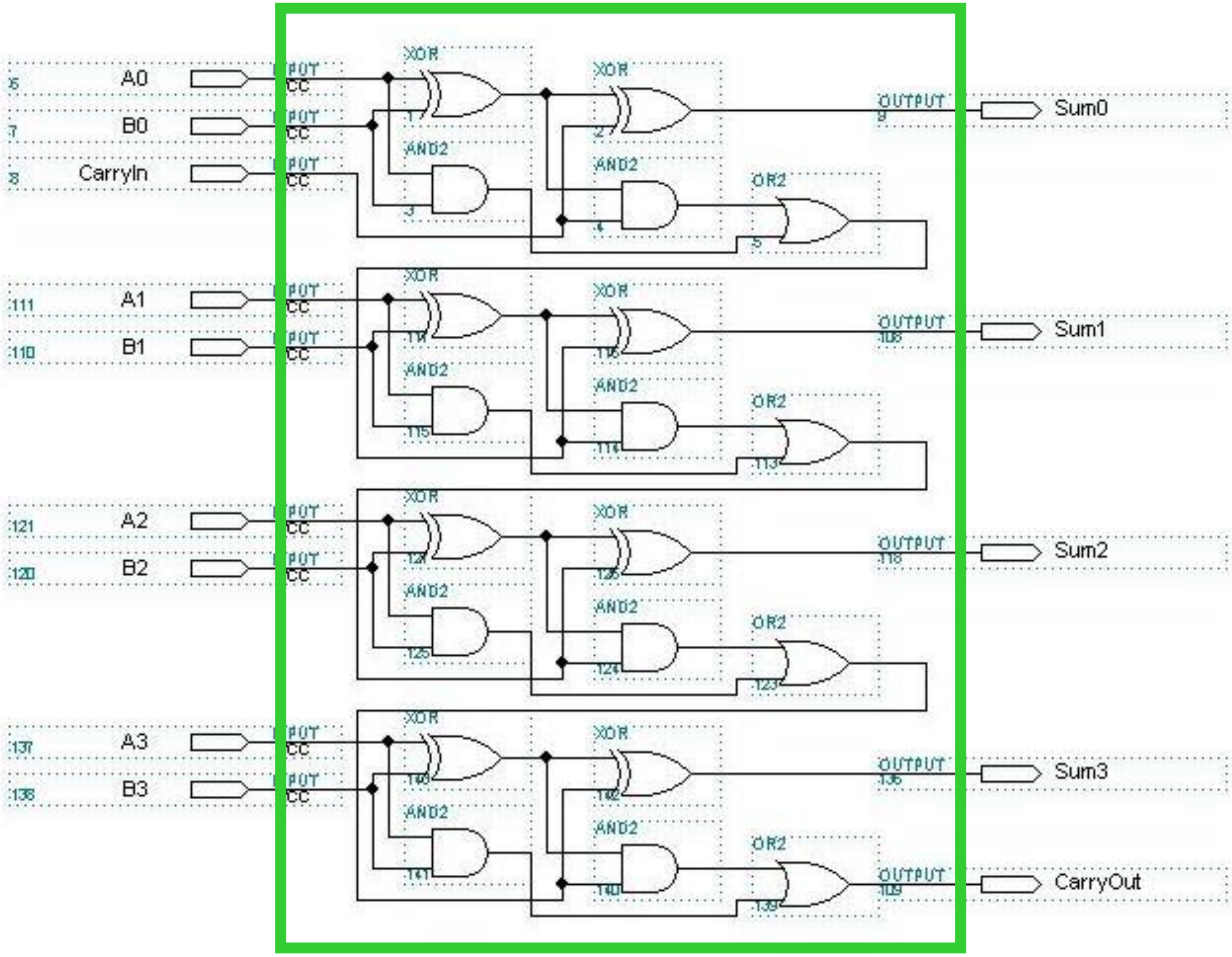
4-bit CRA: Compact form



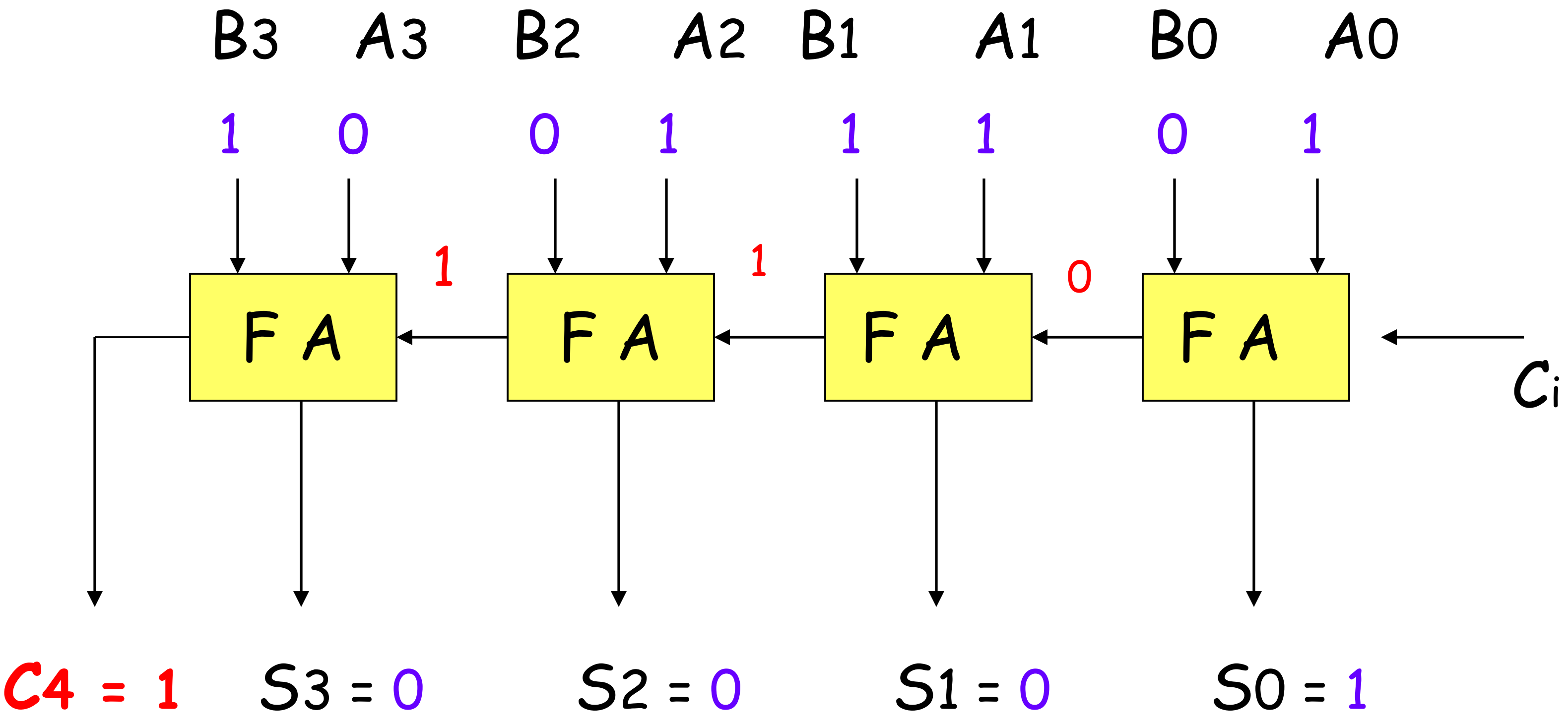
4-bit CRA … using VHDL



4-bit CRA … using VHDL



Overflow … result is 5 bits



[redacted]

Overflow …

- When the addition result has an extra bit (5-bits) than the inputs (4-bits), this is called overflow
- Overflow occurs in case where the carry-out is one (unsigned numbers addition)
- Overflow is a hardware related "problem"…