Binary Subtractor

(Multipler)

Binary Subtractor: Principle

$A - B = ?$

Minus

Binary Subtractor: Principle

A – B = A + 2's comp. of B

Minus

Adder

Read the Lecture about complements

Binary Subtractor: Principle

$$A - B = A + \text{2's comp. of } B$$

$$= A + \{\, (\text{1's comp. of } B) + 1 \,\}$$

Adder

Add

Read the Lecture about complements

Binary Subtractor: 2's

A – B = A + { (1's comp. of B) + 1 }

Therefore,

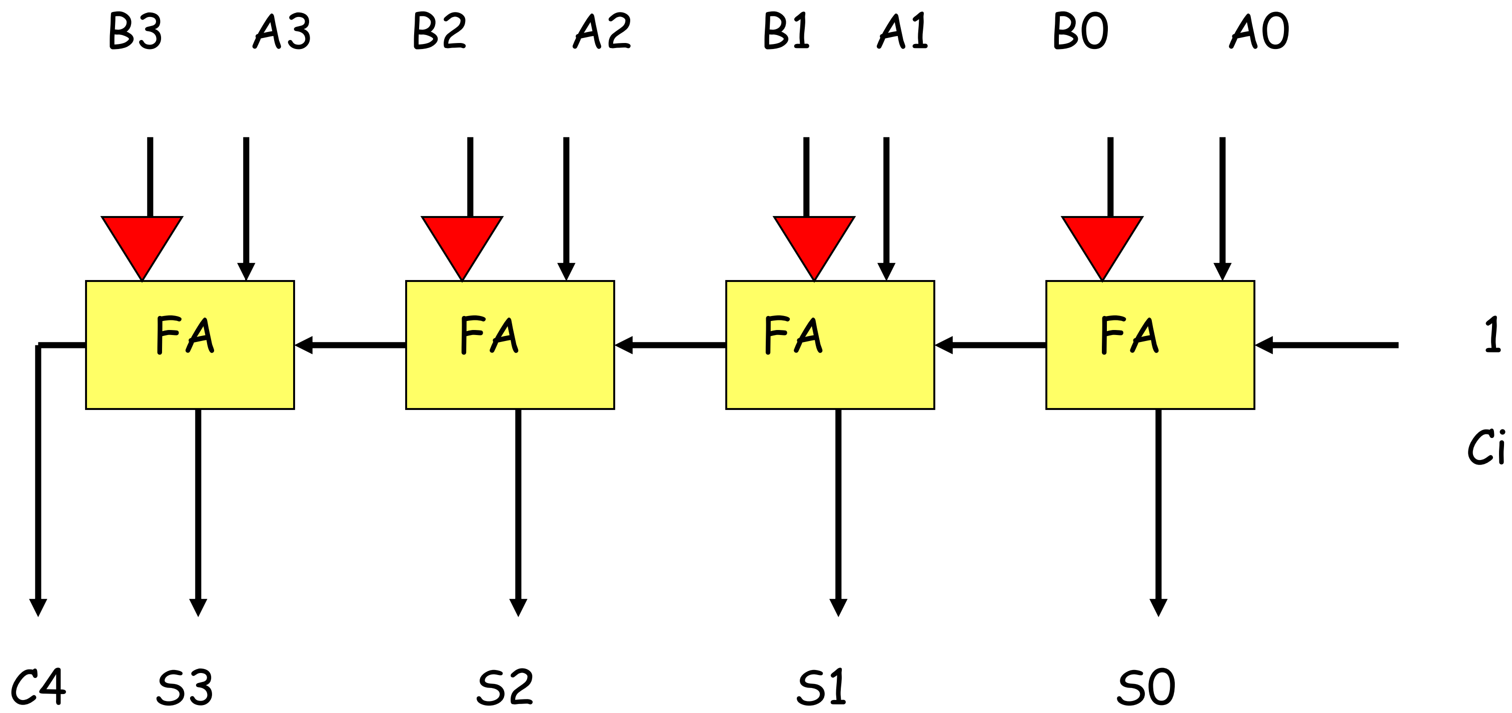
- 1's complement can be implemented with an Inverter.
- The +1 can be implemented by adding 1, or making the Carry-in equal to 1.

4-Bit Subtractor

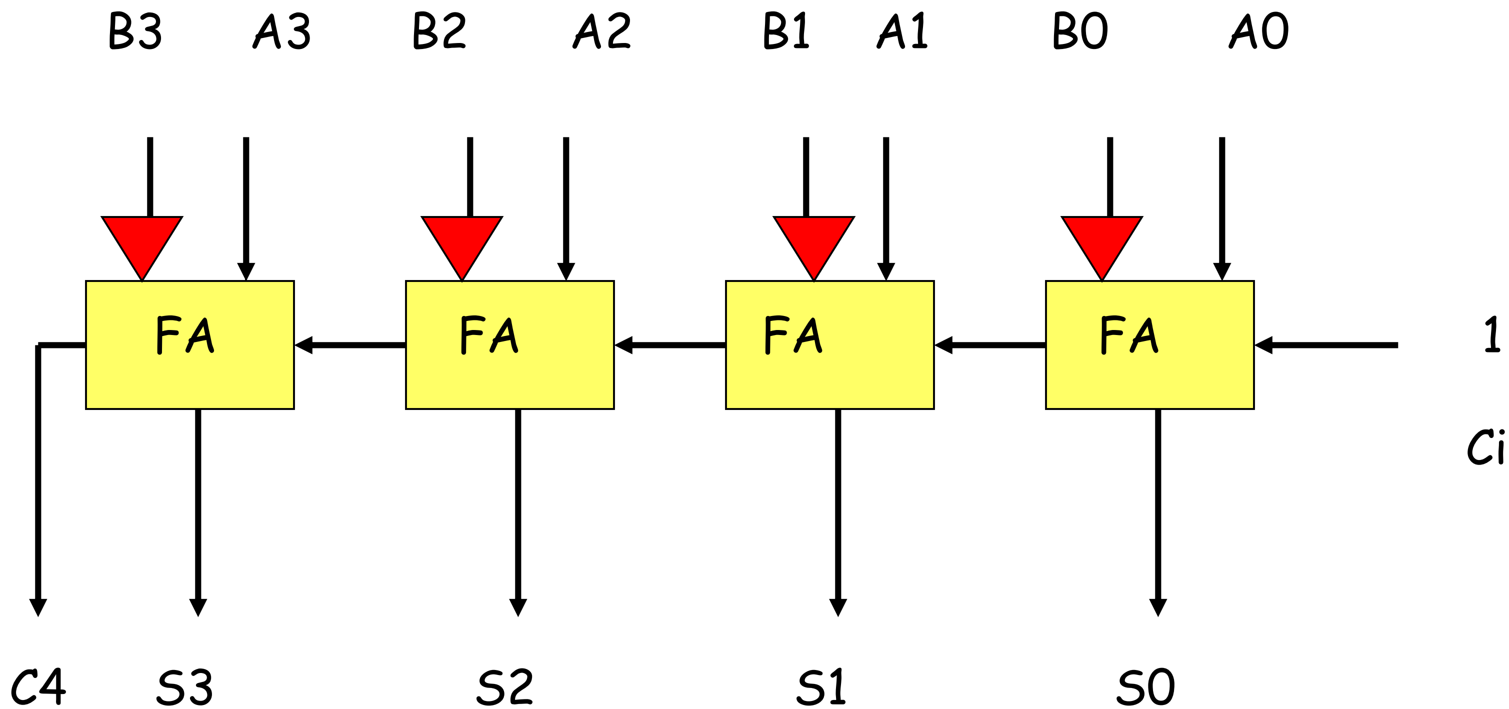
4-bit binary subtraction: A-B

4-bit Binary Subtractor

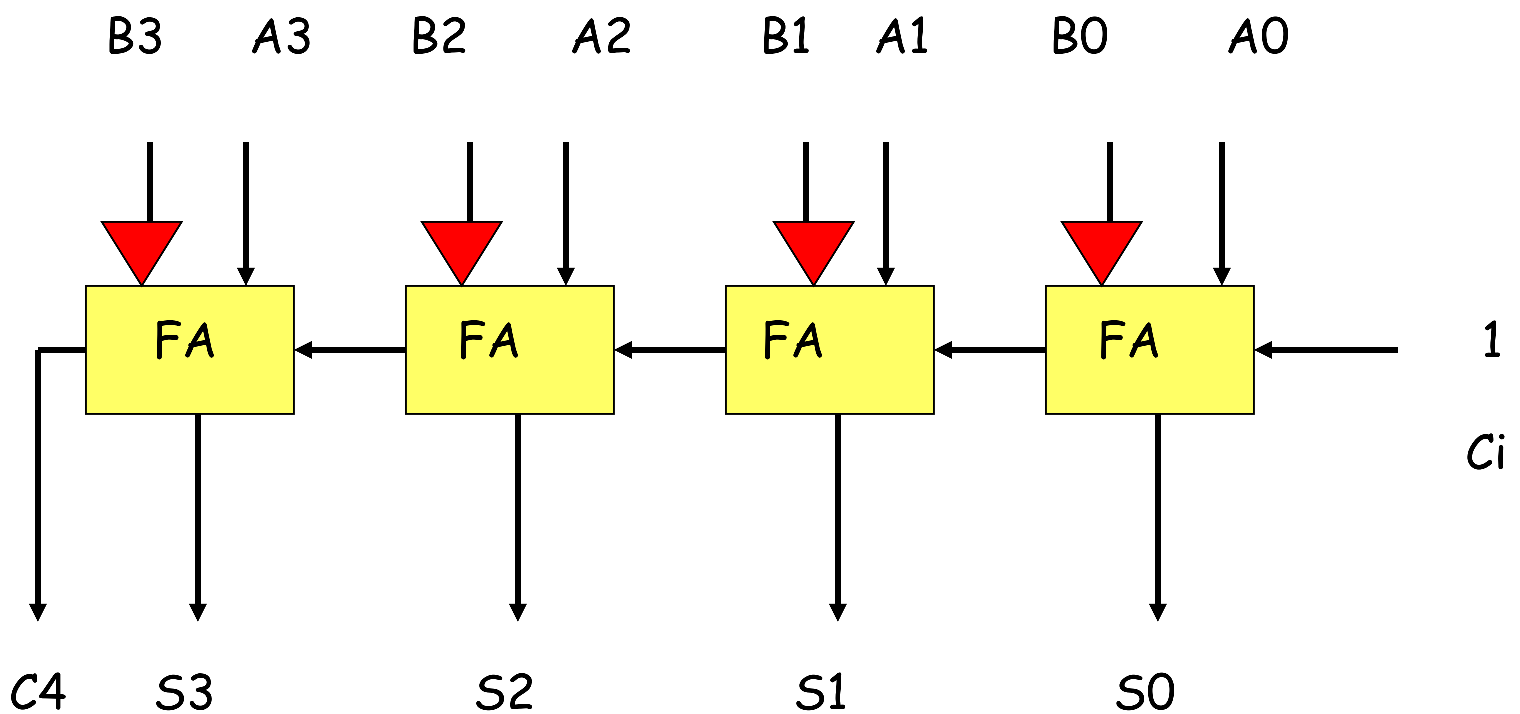
A – B = A + { (1's comp. of B) + 1 }



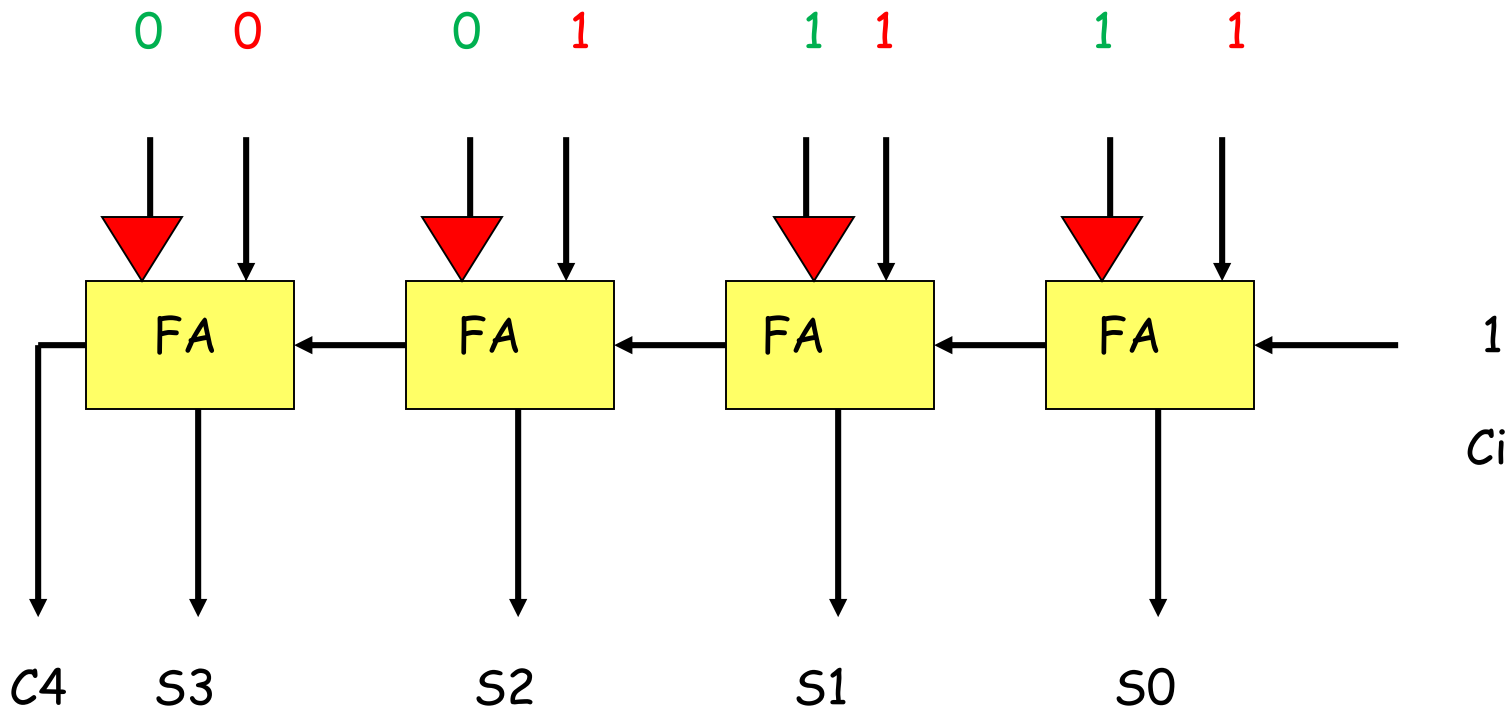
If C4 = 1; S3S2S1S0



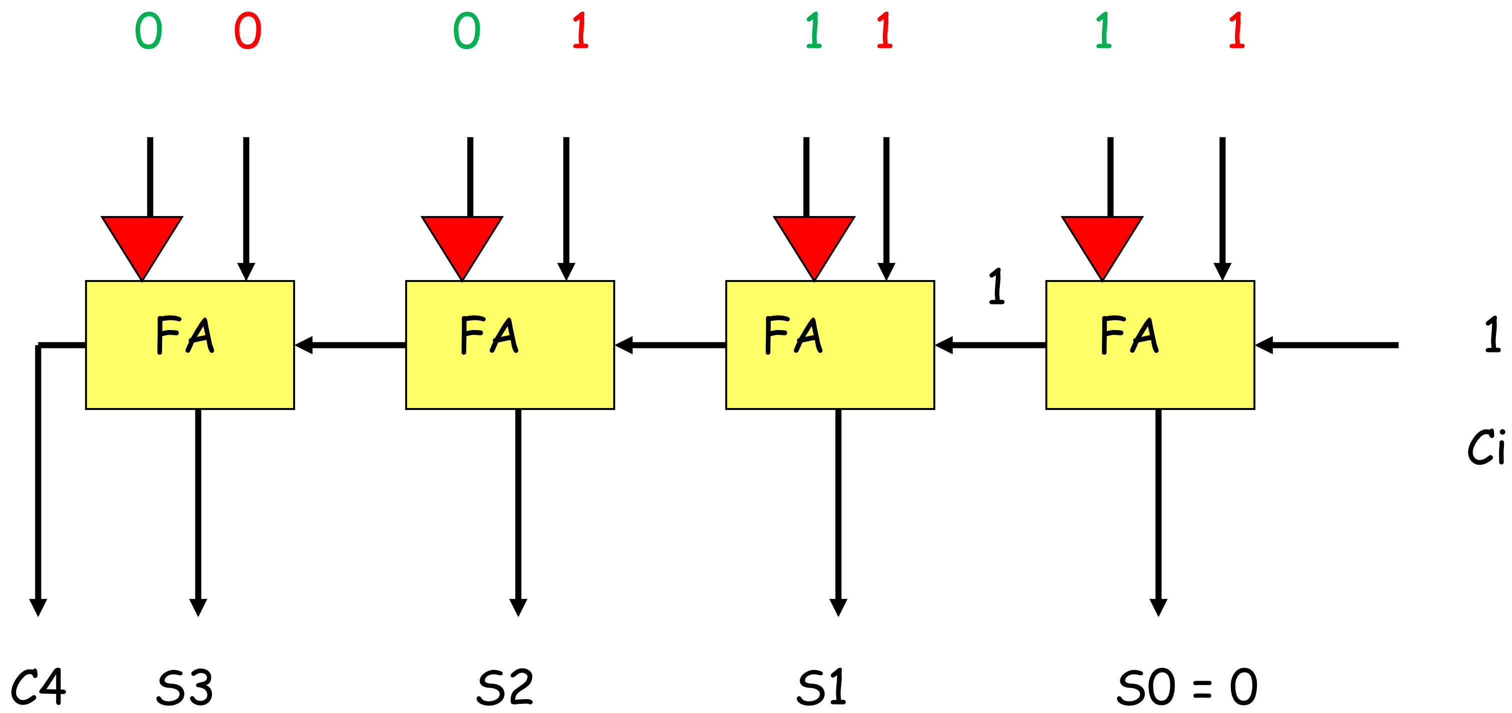
Numerical example-1: 7 (0111) – 3 (0011)



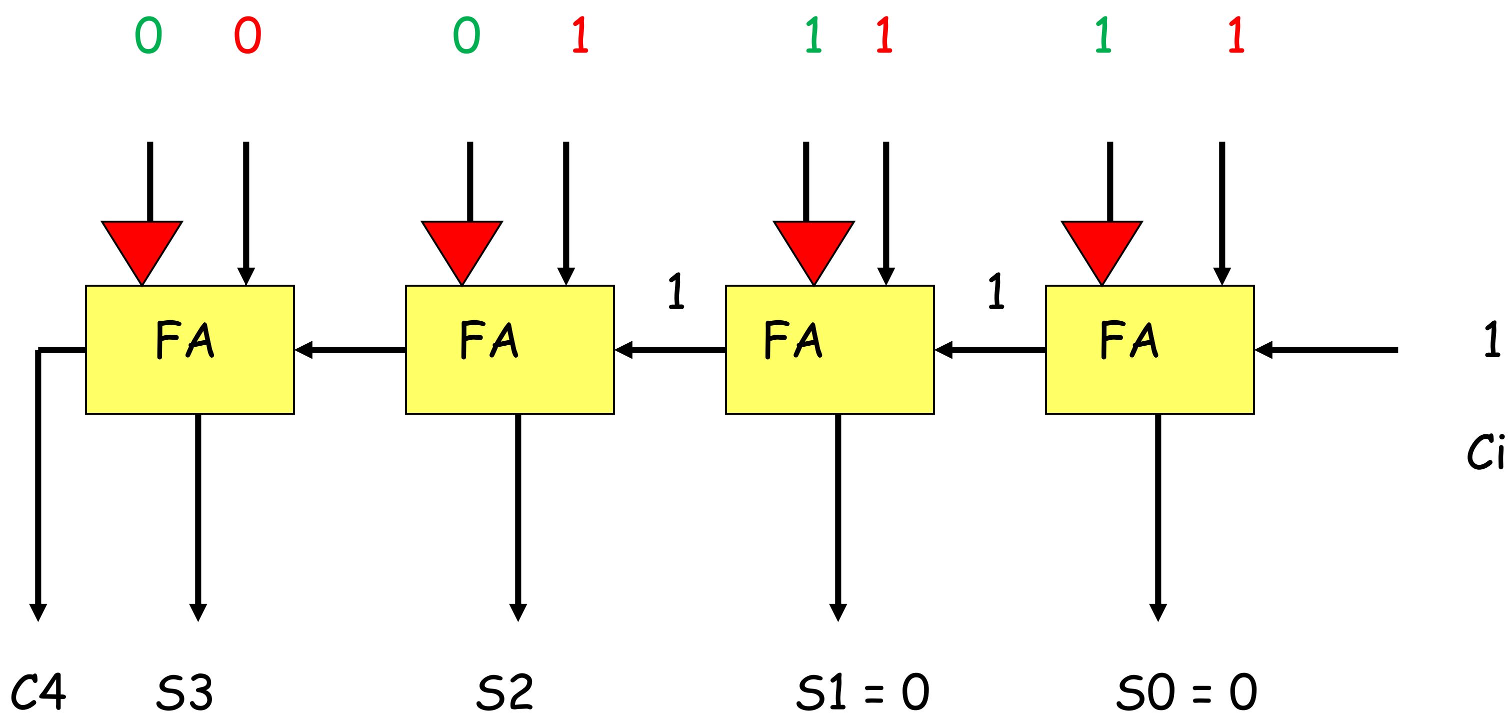
Numerical example-1: 7 (0111) – 3 (0011)



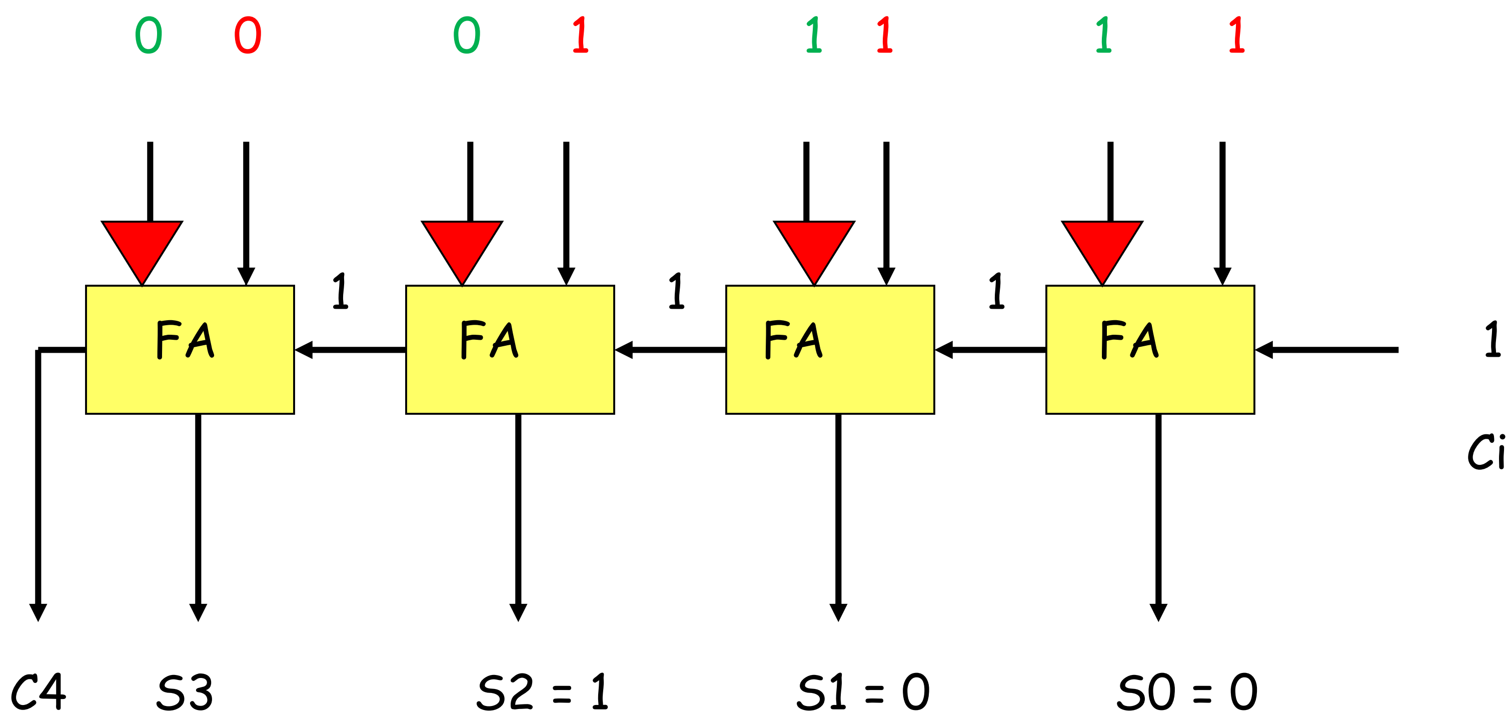
Numerical example-1: 7 (0111) – 3 (0011)



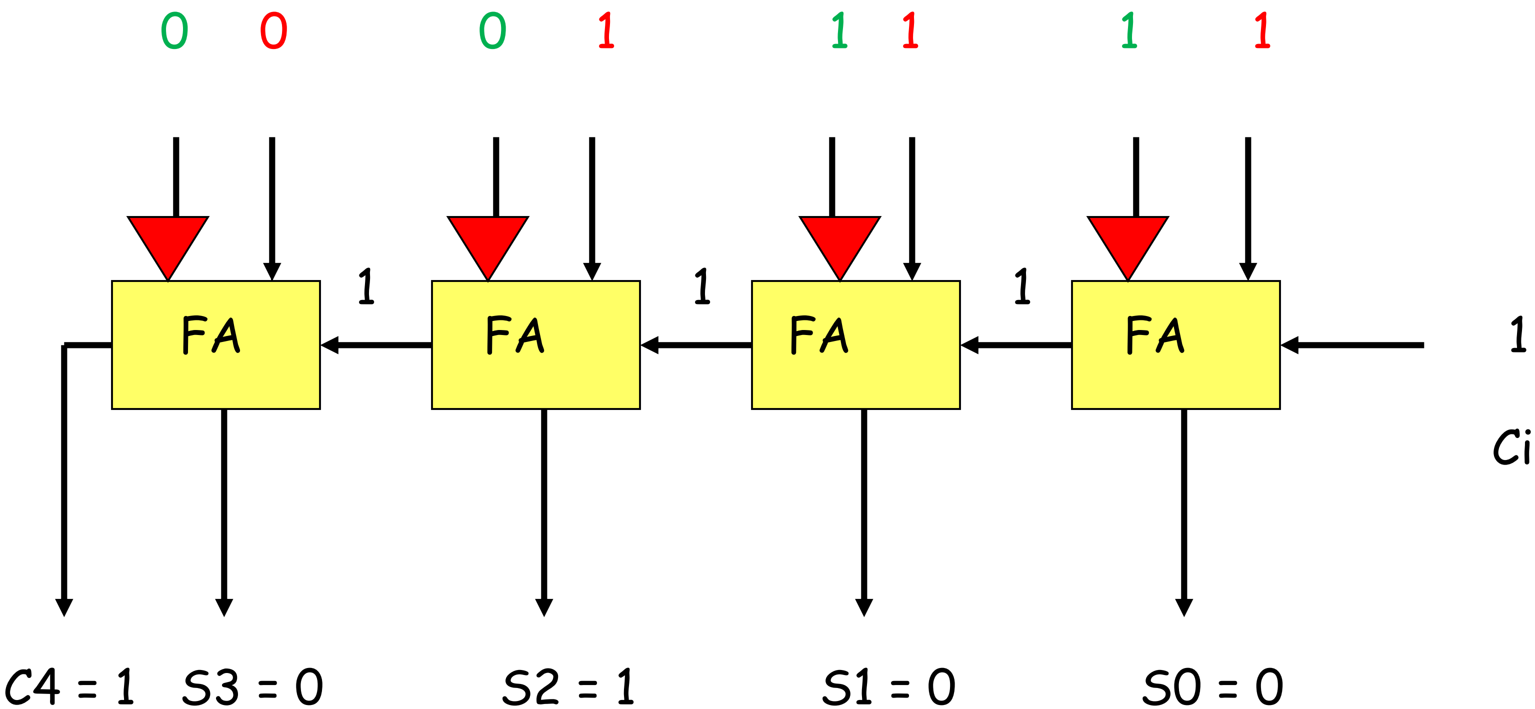
Numerical example-1: 7 (0111) – 3 (0011)



Numerical example-1: 7 (0111) – 3 (0011)



Numerical example-1: 7 (0111) – 3 (0011)



Answer = $0100_2 = 4_{10}$

Remember !

  0111

+1101

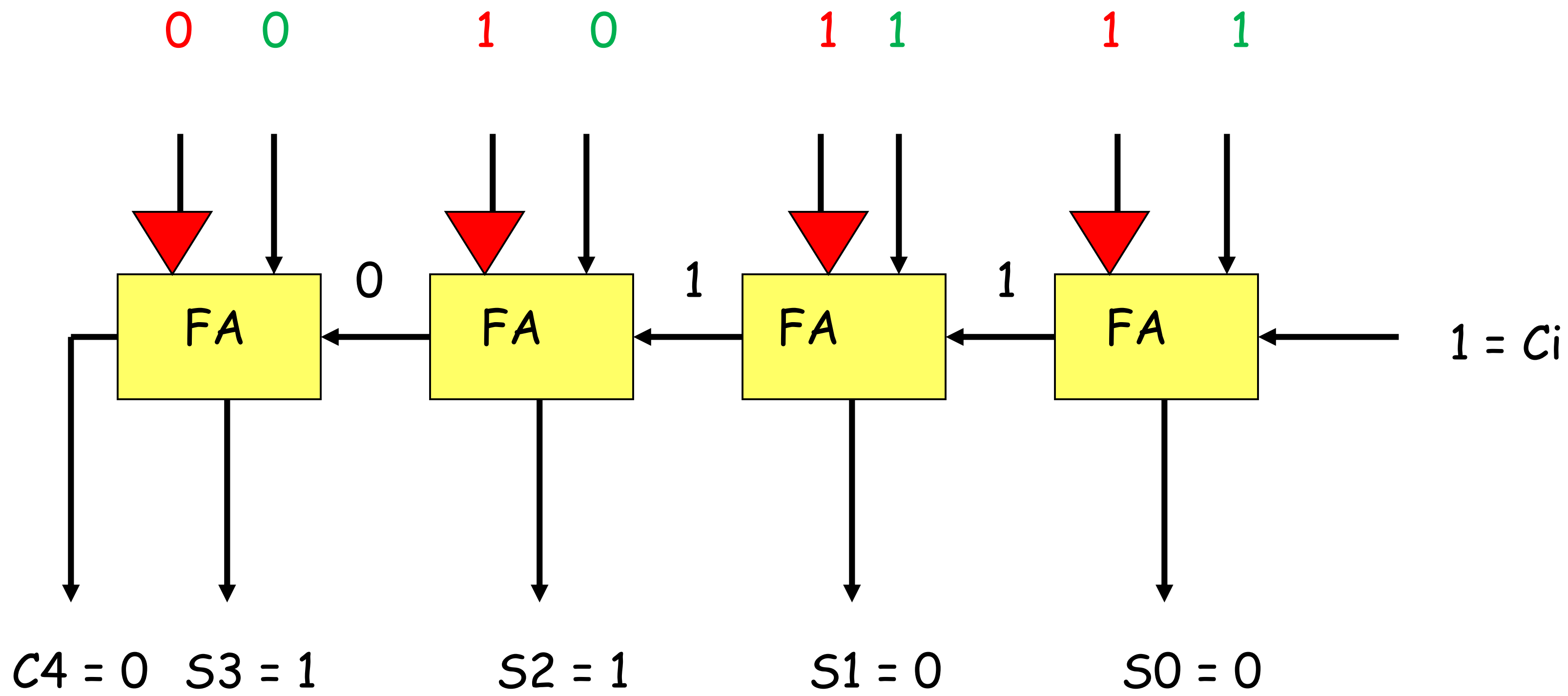
10100

Discard carry

Numerical Example-2

- $3 - 7 = ?$

Numerical example-1: 3 (0011) – 7 (0111)



Answer = 1100_2

3 (0011) – 7 (0111)

- In our example, the result of the 2's complement binary subtraction is in 2's complement form: $(1100)_2$
- $(0011 + 1)_2 = (0100)_2 = 4_{10}$
- Since there is no carry out, the final result: -4_{10}

Why do we prefer 2’s complement?

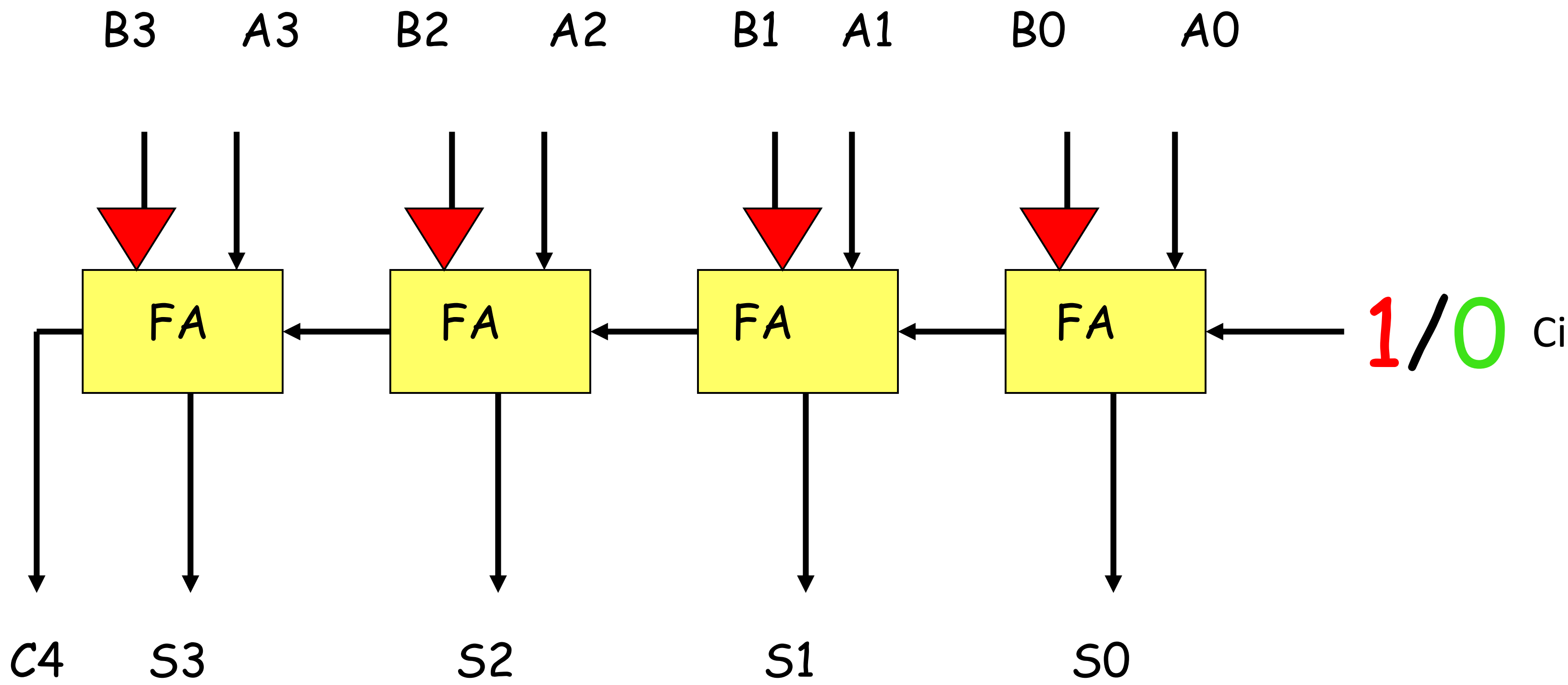
- For subtraction, the 2’s complement is used over the 1’s complement
- 2’s complement magnitude representation has only one zero for representing negative numbers and it is used in today’s computing technology.

4-bit (2's Complement uses one zero)

Decimal	S.Mag	1's Mag.	2's Mag.
7	0111	0111	0111
6	0110	0110	0110
5	0101	0101	0101
4	0100	0100	0100
3	0011	0011	0011
2	0010	0010	0010
1	0001	0001	0001
0	0000	0000	0000
-0	1000	1111	—
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	—	—	1000

Combined (Subtractor/Adder)

Combined (Subtractor/Adder)



A trick with the input: Bi (i = 0,…,n)

$$Bi \oplus 1 = ?$$

$$Bi \oplus 0 = ?$$

$\oplus$

Prove it … 5 min

Logic Circuit?

$$B_i \oplus 1 = \overline{B_i}$$

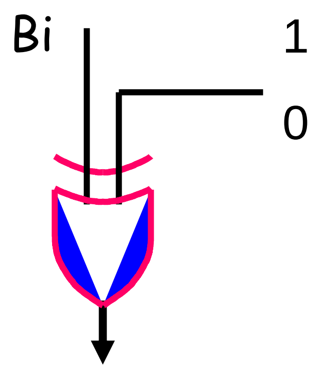
$$B_i \oplus 0 = B_i$$

$\oplus$

Just a XOR gate

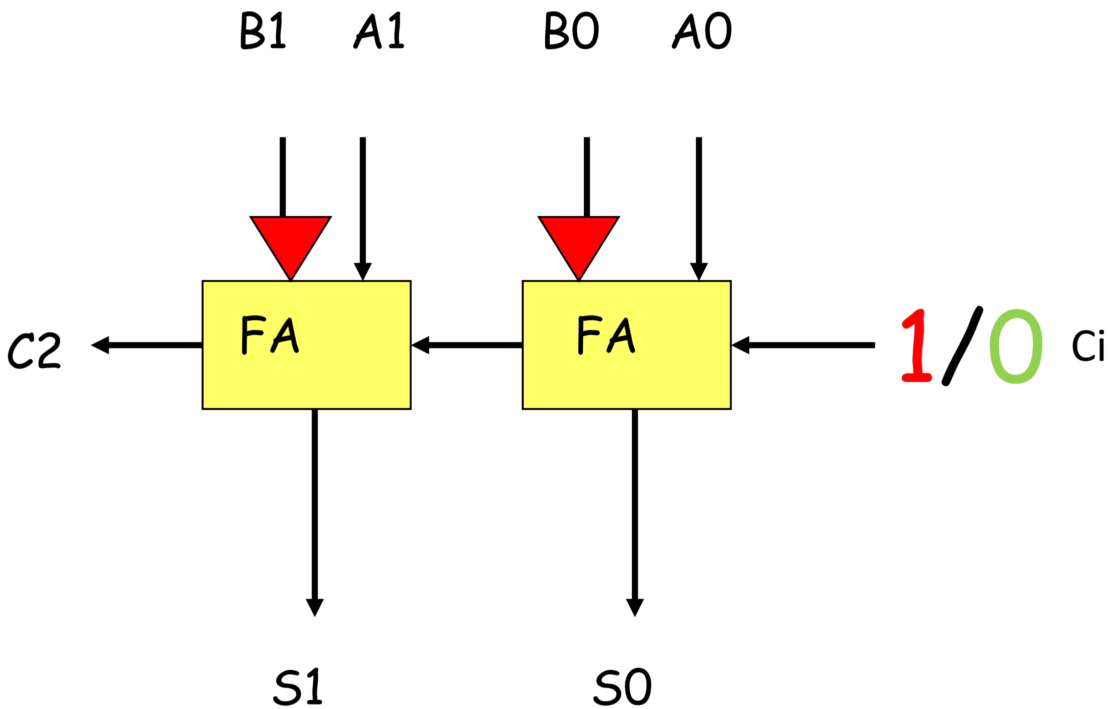
$$Bi \oplus 1 = \overline{Bi}$$

$$Bi \oplus 0 = Bi$$



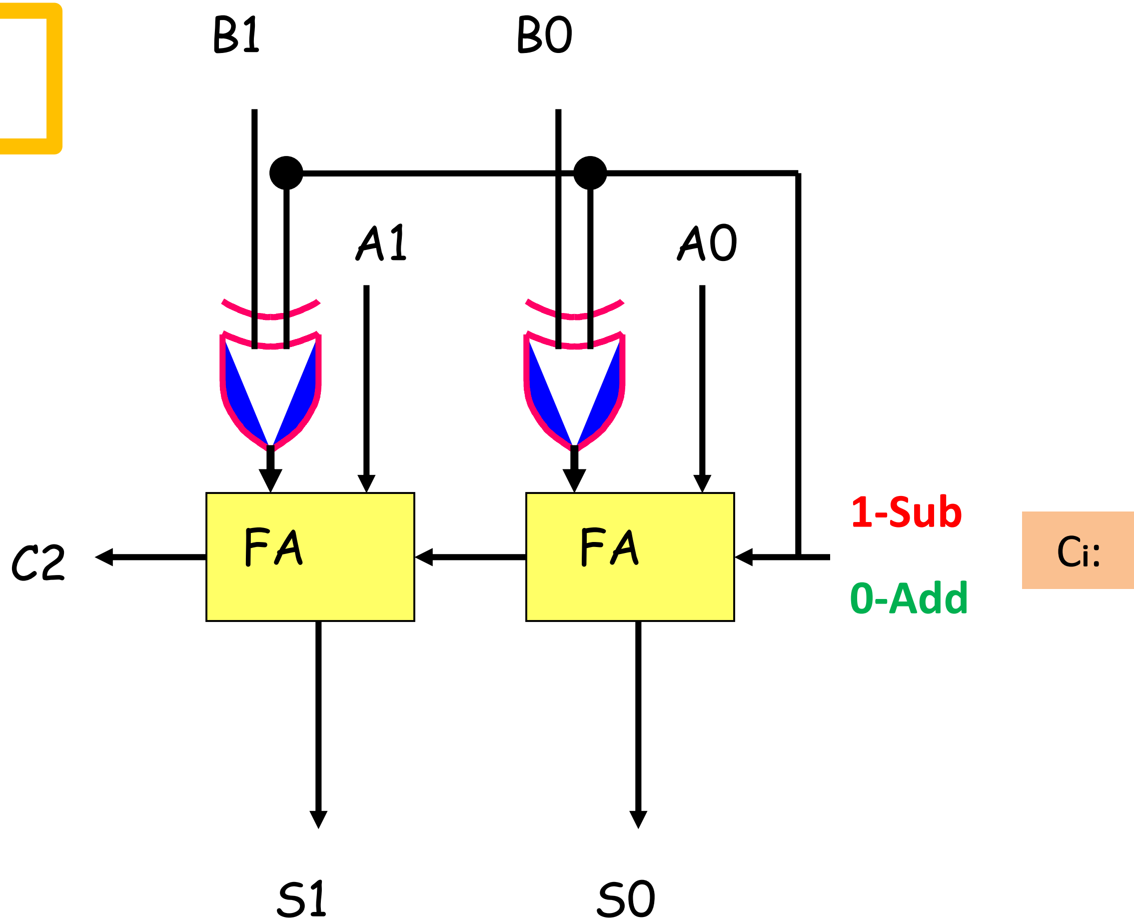
If 1 (**Subtration**); then the Bi = Negated as should be
If 0 (**Addition**); then the Bi is the same as should be

Combined 2's(Subtractor/Adder)

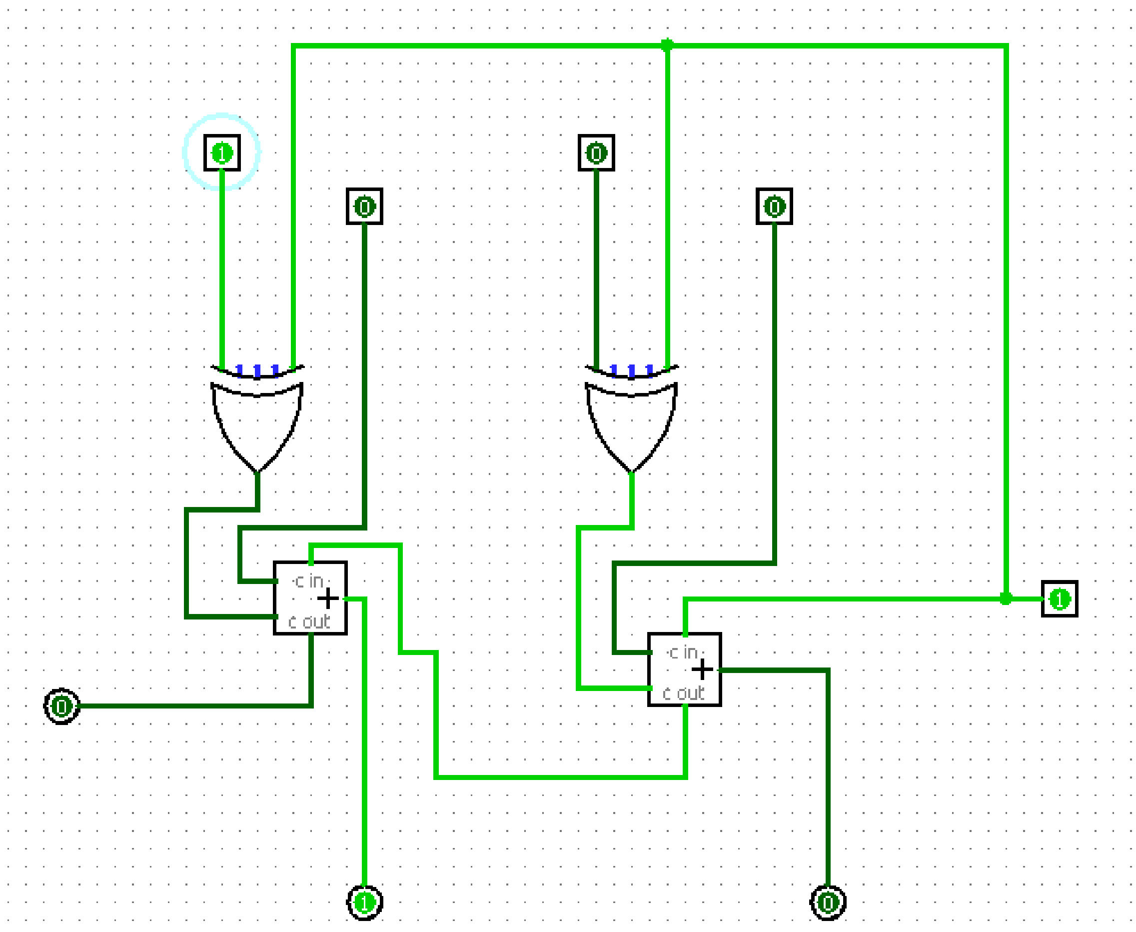


Combined 2's(Subtractor/Adder)

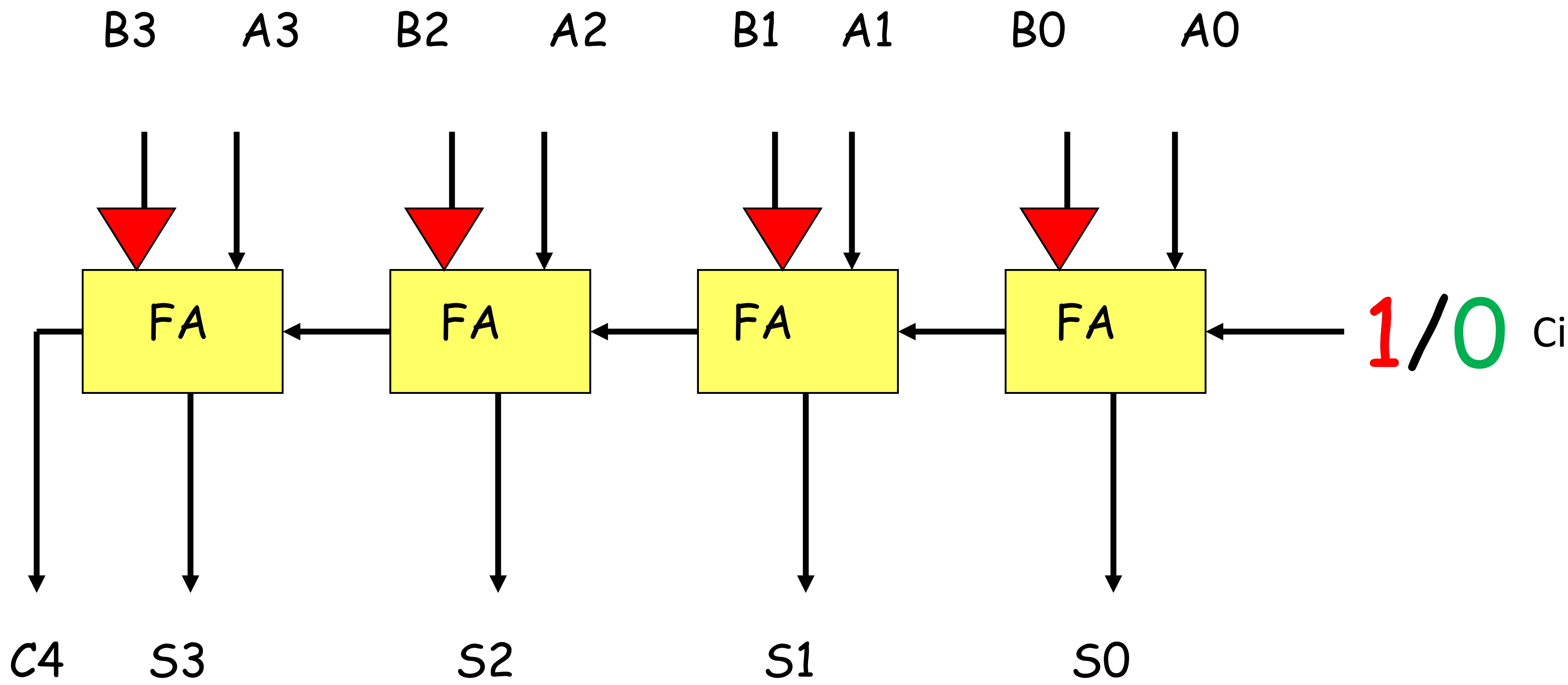
Carry input (C_i) = 1; for subtraction
Carry input (C_i) = 0, for addition



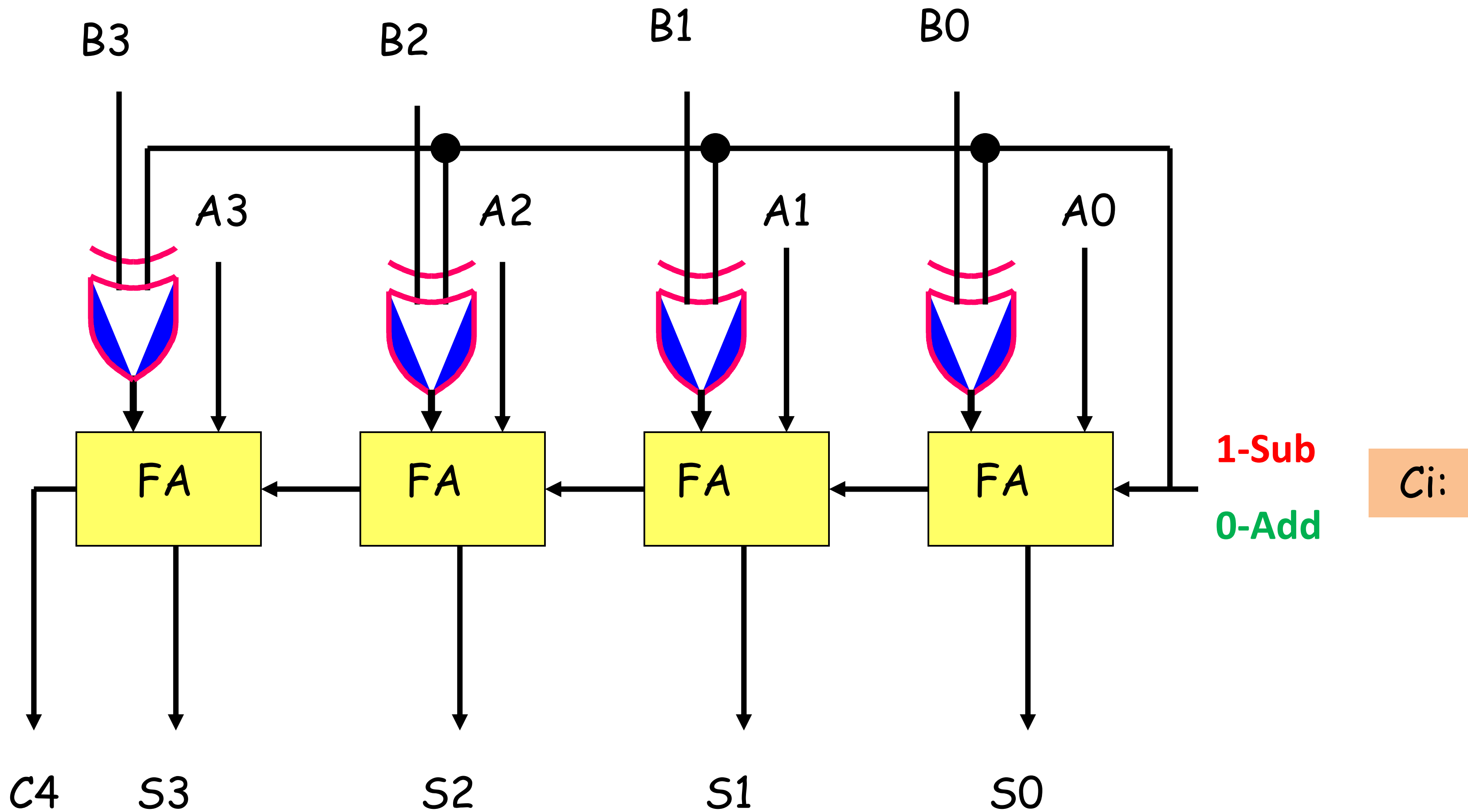
Live proof! (LogiSim freeware)



Combined 2's(Subtractor/Adder)



Combined 2's(Subtractor/Adder)



Design a 2-bit Multiplier

$$\begin{array}{ccc} & B_1 & B_0 \\ \times & A_1 & A_0 \\ \hline \end{array}$$

2-bit Multiplication

		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0

$C_0 = A_0B_0$

$C_1 = A_1B_0 + A_0B_1$ (carry)

$C_2 = A_1B_1$ (carry)

$C_3 =$ carry

Design a 2-bit Multiplier using Adders and gates

2-bit Multiplier using: Adders+Gates

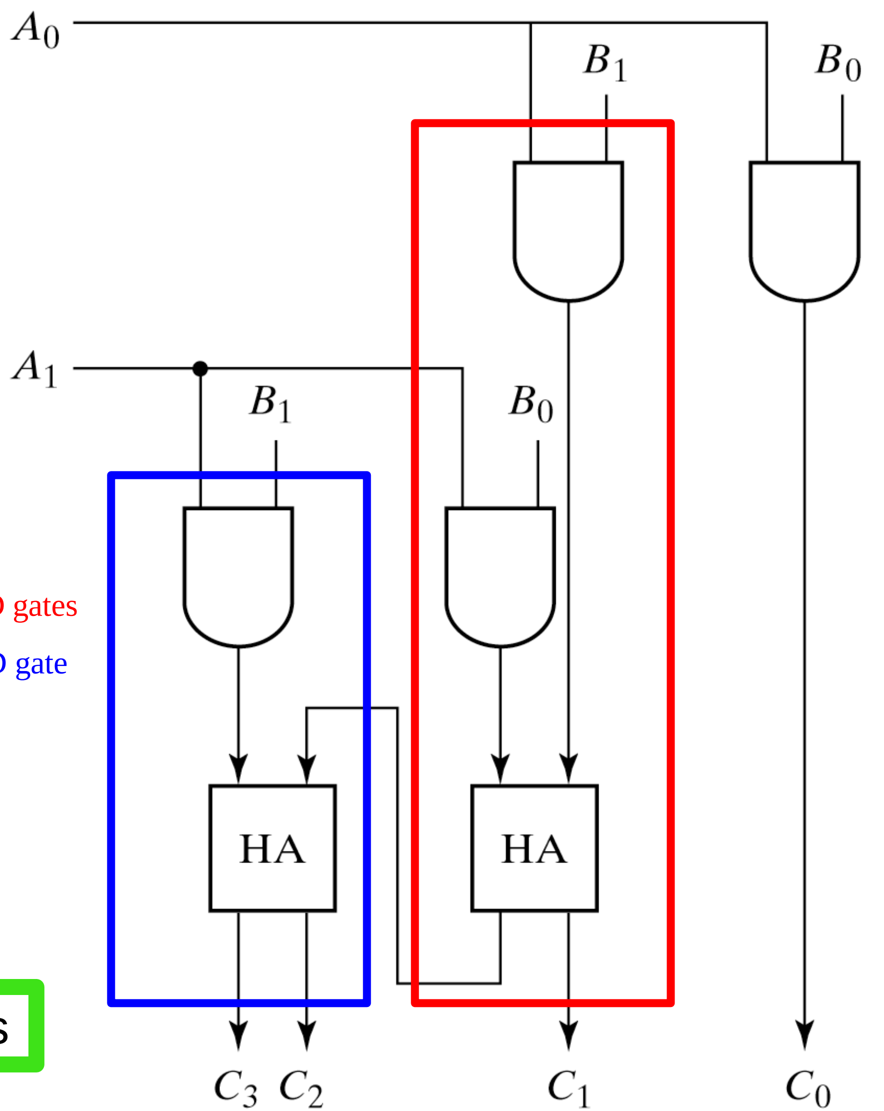
$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$

$C_0 = A_0B_0$ [AND gate]

$C_1 = A_1B_0 + A_0B_1$ (carry) [HALF ADDER] and AND gates

$C_2 = A_1B_1$ (carry) [HALF ADDER] and AND gate

$C_3 =$ carry



Only use Half-Adders and AND gates

Divider (Division Operation)

Division

- Division is the fourth mathematical operation. The binary division operation consists of multiplications and subtractions. Circuits that perform binary division are more complex than adders. Division and multiplication algorithms is an important subject of computer organization.

http://www.wikihow.com/Divide-Binary-Numbers

David Patterson and John L. Hennessy, "Computer Organization and Design: the Hardware/Software Interface", Morgan Kaufmann Publishers, 2017.