# SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# COURSE NAME :19IT301 COMPUTER ORGANIZATION AND ARCHITECTURE

II YEAR /III SEMESTER

## Unit 2- ARITHMETIC OPERATIONS

Topic 2 :  – Design of fast adders

# Addition/subtraction of signed numbers

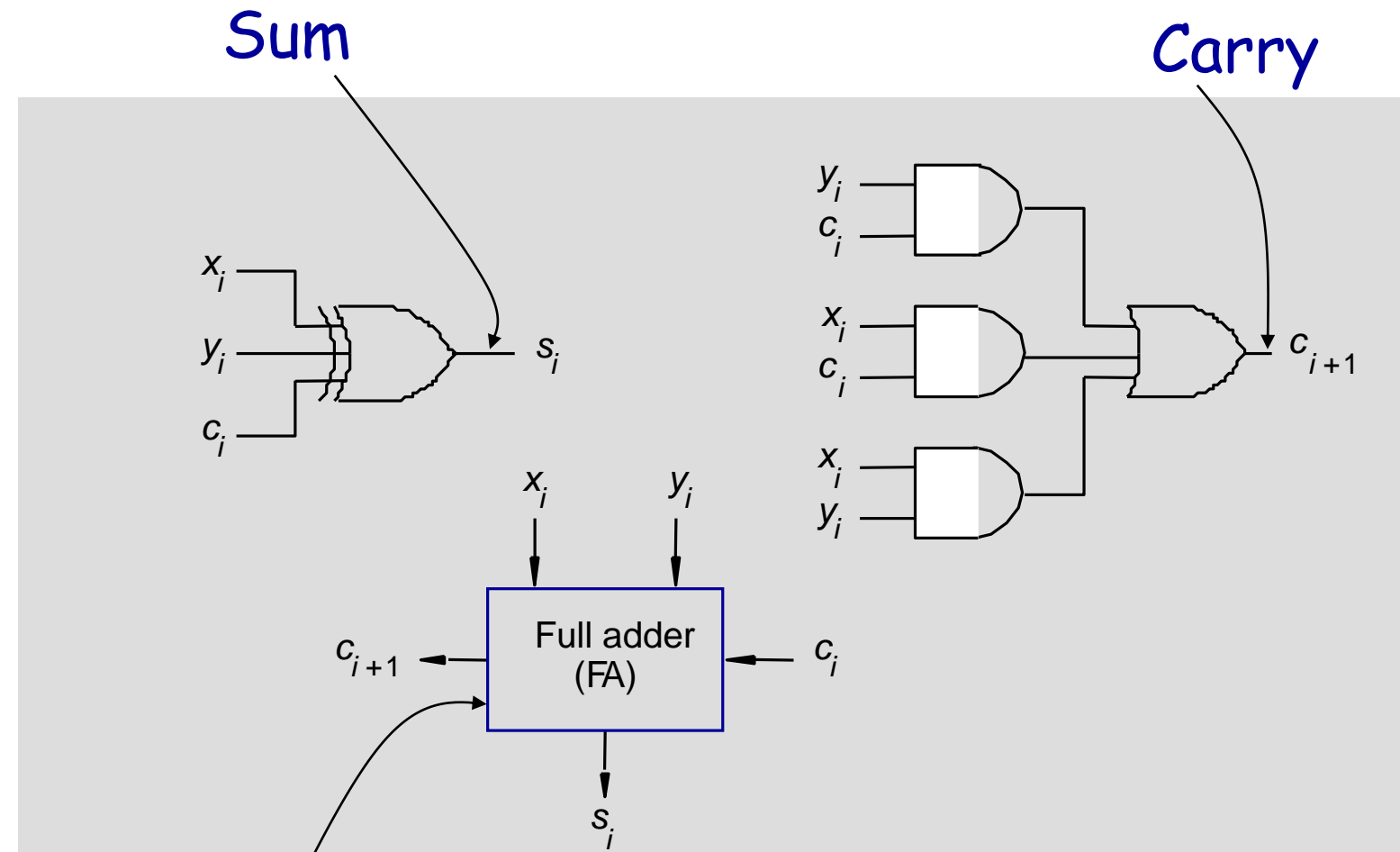| $x_i$ | $y_i$ | Carry-in $c_i$ | Sum $s_i$ | Carry-out $c_{i+1}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

$$s_i = \overline{x_i}\,\overline{y_i}\,c_i + \overline{x_i}\,y_i\,\overline{c_i} + x_i\,\overline{y_i}\,\overline{c_i} + x_i\,y_i\,c_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = y_i c_i + x_i c_i + x_i y_i$$

At the $i^{th}$ stage:

<u>Input:</u>
$c_i$ is the carry-in

<u>Output:</u>
$s_i$ is the sum
$c_{i+1}$ carry-out to $(i+1)^{st}$ state

Example:

$$
\begin{array}{ccc}
X & 7 & \quad 0\ 1\ 1\ 1 \\
+\ Y\ = & +\ 6\ = & +\ _0 0\ _1 1\ _1 1\ _0 0\ _0 \\
\hline
Z & 13 & \quad 1\ 1\ 0\ 1
\end{array}
$$

Carry-out $c_{i+1}$ ← $x_i$, $y_i$ → Carry-in $c_i$, $s_i$

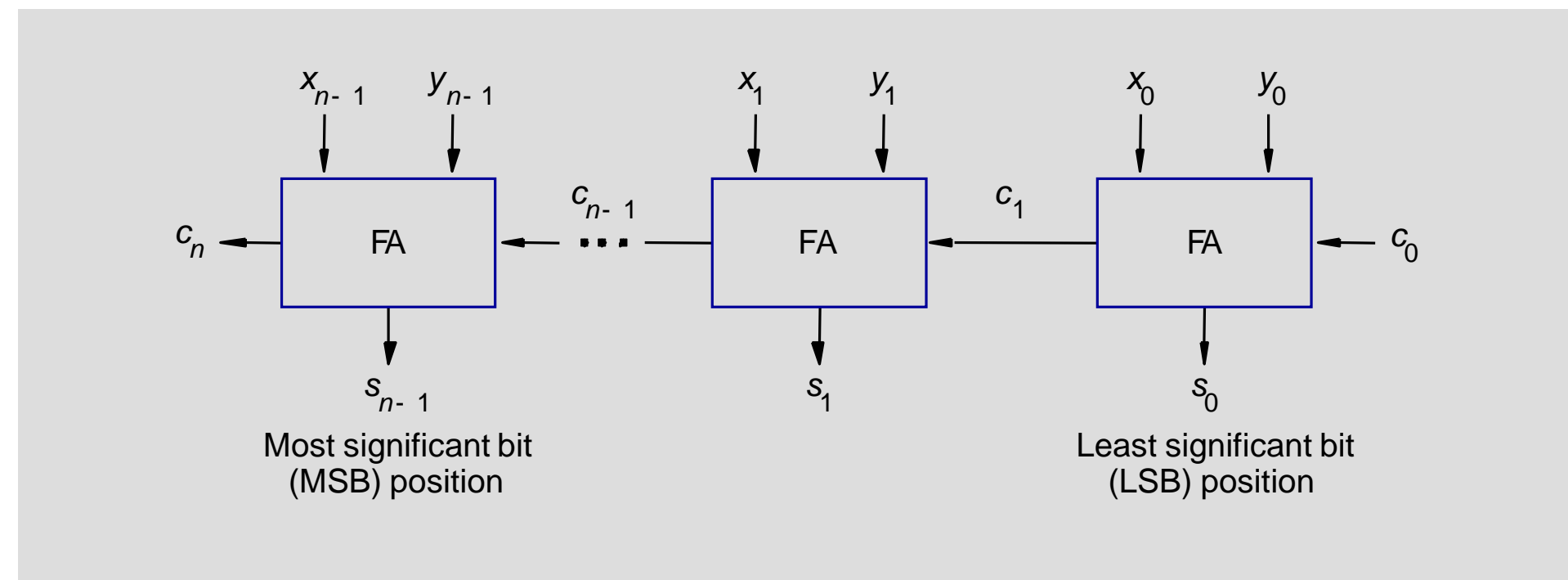Legend for stage $i$
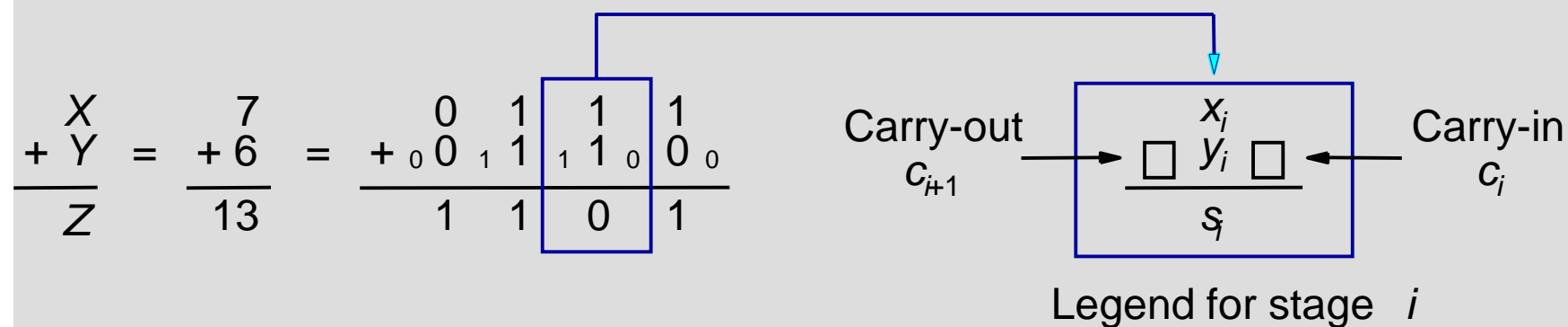
# Addition logic for a single stage



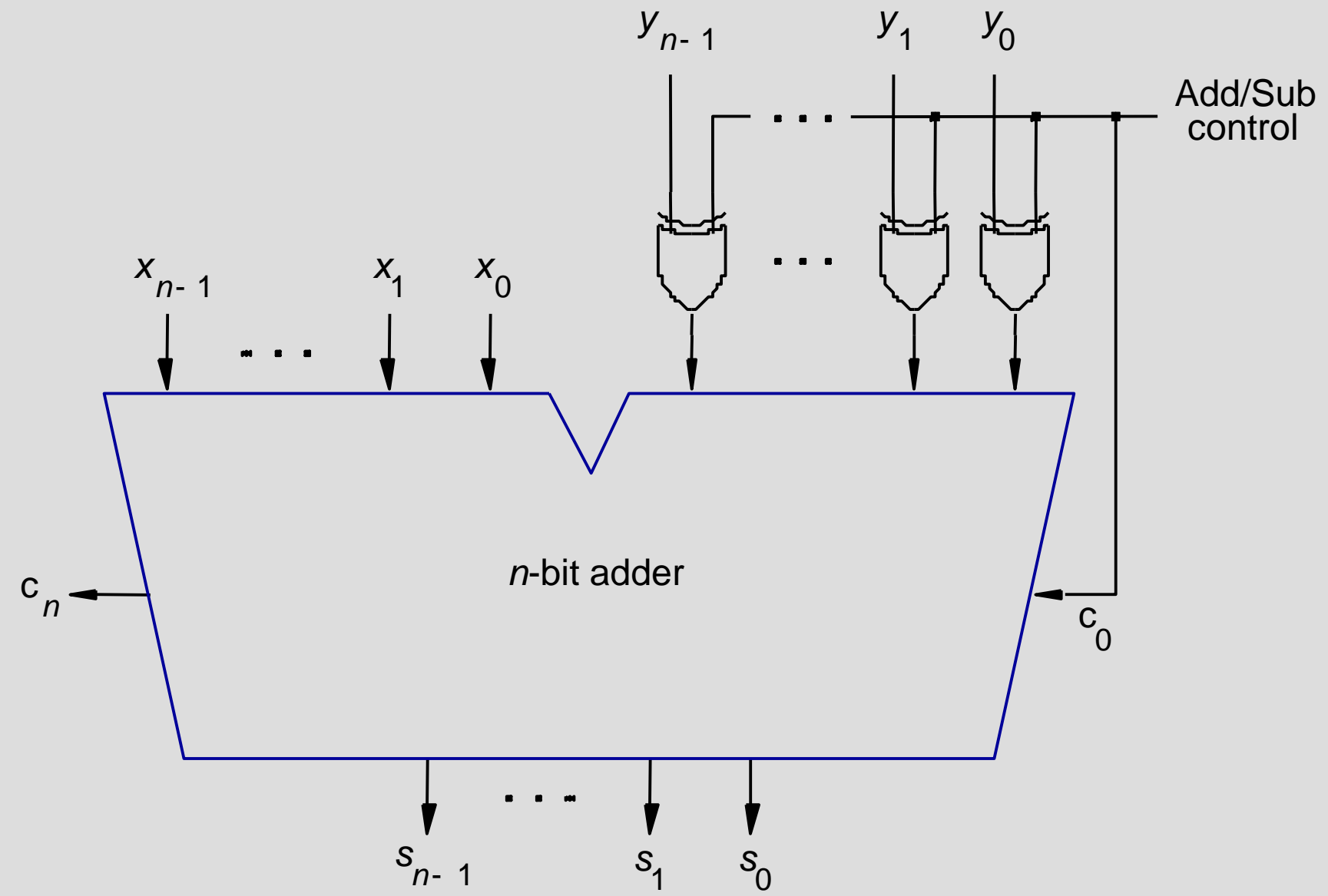Full Adder (FA): Symbol for the complete circuit
for a single stage of addition.

# $n$-bit adder

- Cascade $n$ full adder (FA) blocks to form a $n$-bit adder.
- Carries propagate or ripple through this cascade, _$n$-bit ripple carry adder._

# *n*-bit adder/subtractor (contd..)



- Add/sub control = 0, addition.
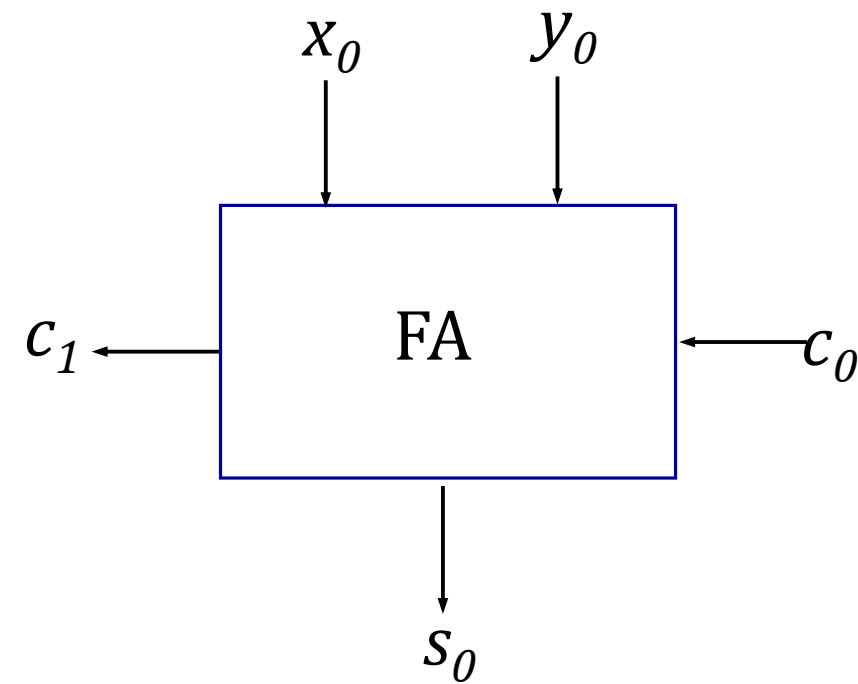- Add/sub control = 1, subtraction.

# Detecting overflows

- Overflows can only occur when the sign of the two operands is the same.
- Overflow occurs if the sign of the result is different from the sign of the operands.
- Recall that the MSB represents the sign.
  - $x_{n-1}$, $y_{n-1}$, $s_{n-1}$ represent the sign of operand $x$, operand $y$ and result $s$ respectively.
- Circuit to detect overflow can be implemented by the following logic expressions:

$$Overflow = x_{n-1} y_{n-1} \overline{s}_{n-1} + \overline{x}_{n-1} \overline{y}_{n-1} s_{n-1}$$

$$Overflow = c_n \oplus c_{n-1}$$

Design of fast adder/Computer organization and architecture/Dr.K.Periyakaruppan/CSE/SNSCE

# Computing the add time



$x_0$   $y_0$

FA

$c_1$          $c_0$

$s_0$

Consider $0^{th}$ stage:
- $c_1$ is available after 2 gate delays.
- $s_1$ is available after 1 gate delay.



Sum                                  Carry

$x_i$
$y_i$          $s_i$
$c_i$

$y_i$
$c_i$

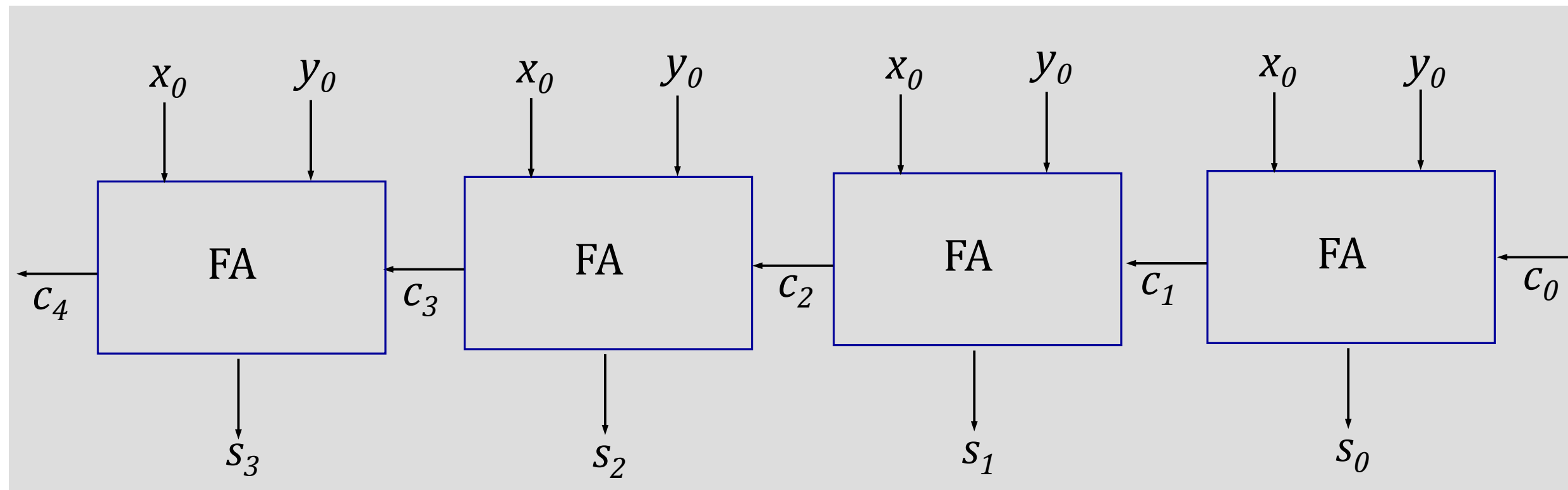$x_i$                    $c_{i+1}$
$c_i$

$x_i$
$y_i$
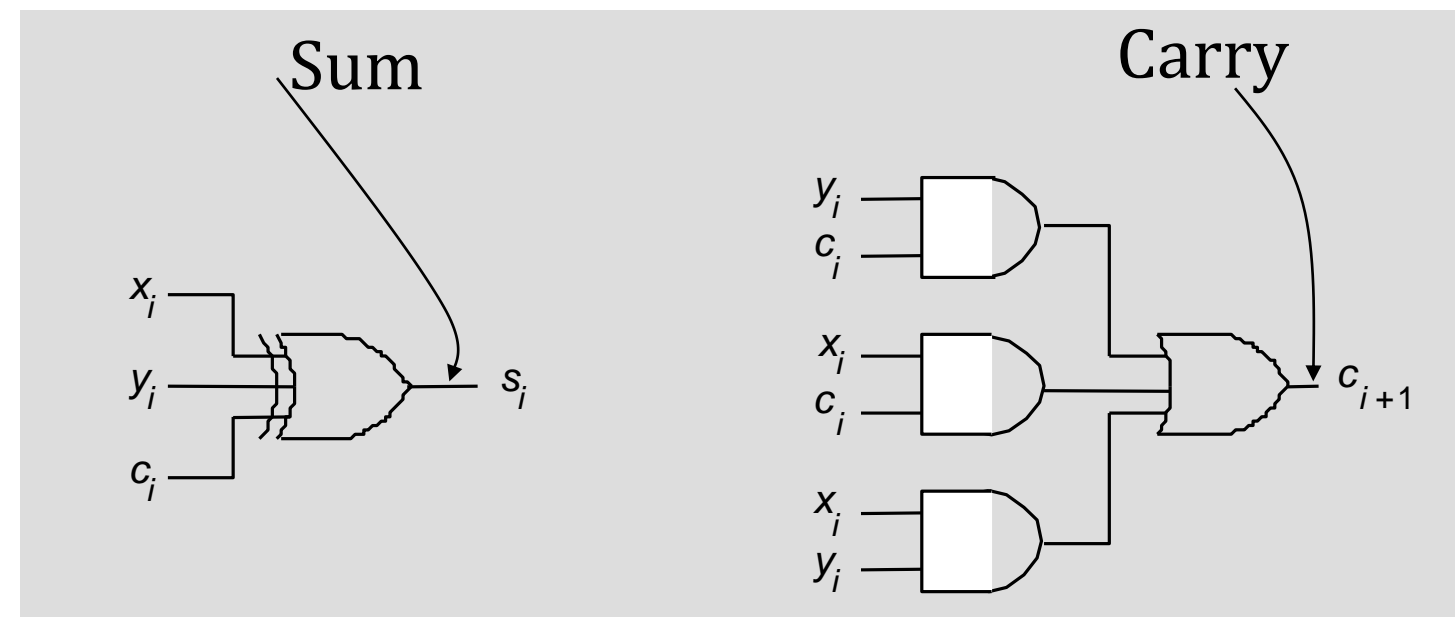
# Computing the add time (contd..)

Cascade of 4 Full Adders, or a 4-bit adder



- $s_0$ available after 1 gate delays, $c_1$ available after 2 gate delays.
- $s_1$ available after 3 gate delays, $c_2$ available after 4 gate delays.
- $s_2$ available after 5 gate delays, $c_3$ available after 6 gate delays.
- $s_3$ available after 7 gate delays, $c_4$ available after 8 gate delays.

For an $n$-bit adder, $s_{n-1}$ is available after *2n-1* gate delays
$c_n$ is available after *2n* gate delays.

# Fast addition

Recall the equations:

$$s_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

Second equation can be written as:

$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$

We can write:

$$c_{i+1} = G_i + P_i c_i$$

$$where \; G_i = x_i y_i \; and \; P_i = x_i + y_i$$

- $G_i$ is called generate function and $P_i$ is called propagate function
- $G_i$ and $P_i$ are computed only from $x_i$ and $y_i$ and not $c_i$, thus they can be computed in one gate delay after $X$ and $Y$ are applied to the inputs of an $n$-bit adder.

# Carry lookahead

$$c_{i+1} = G_i + P_i c_i$$

$$c_i = G_{i-1} + P_{i-1} c_{i-1}$$

$$\Rightarrow c_{i+1} = G_i + P_i(G_{i-1} + P_{i-1} c_{i-1})$$

*continuing*

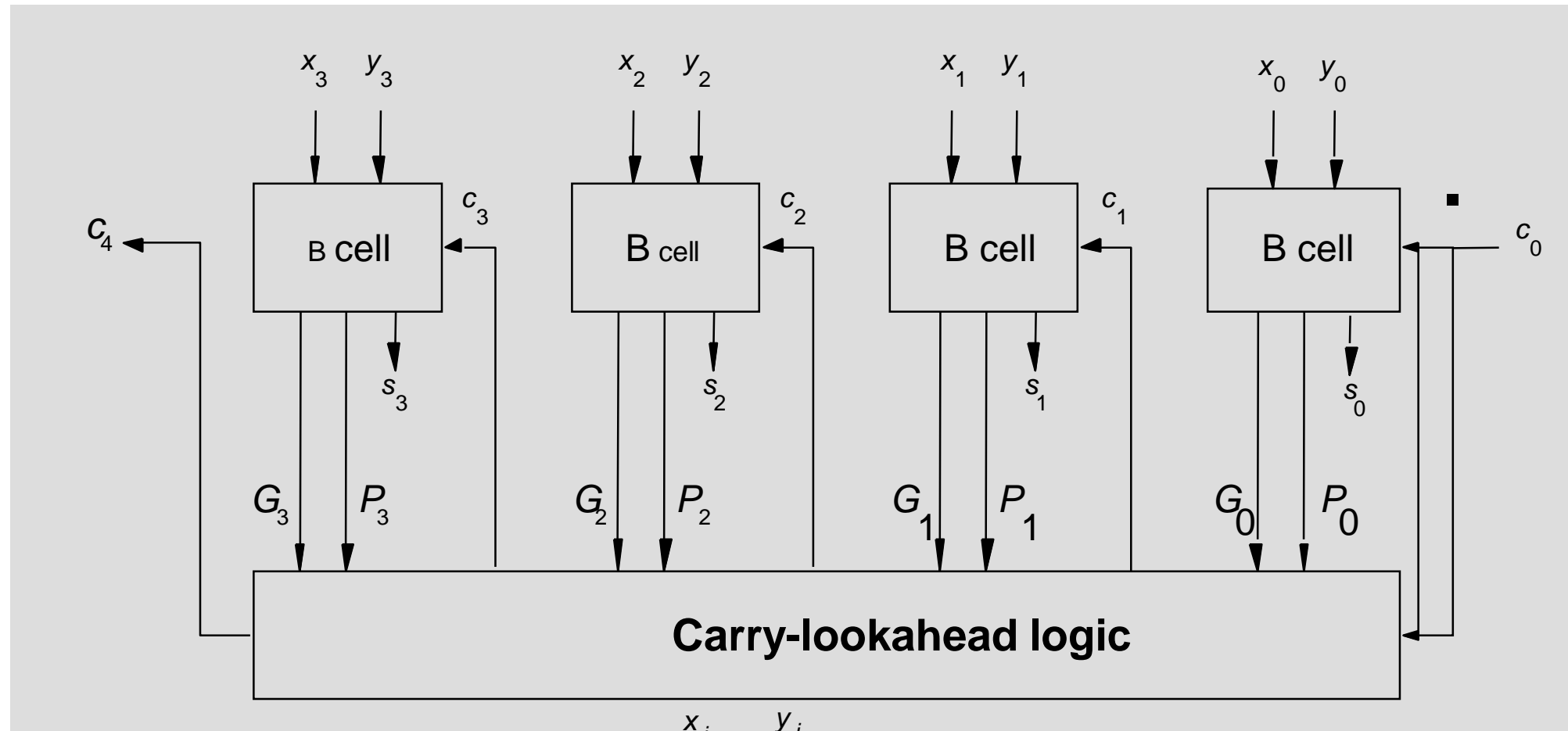$$\Rightarrow c_{i+1} = G_i + P_i(G_{i-1} + P_{i-1}(G_{i-2} + P_{i-2} c_{i-2}))$$

*until*

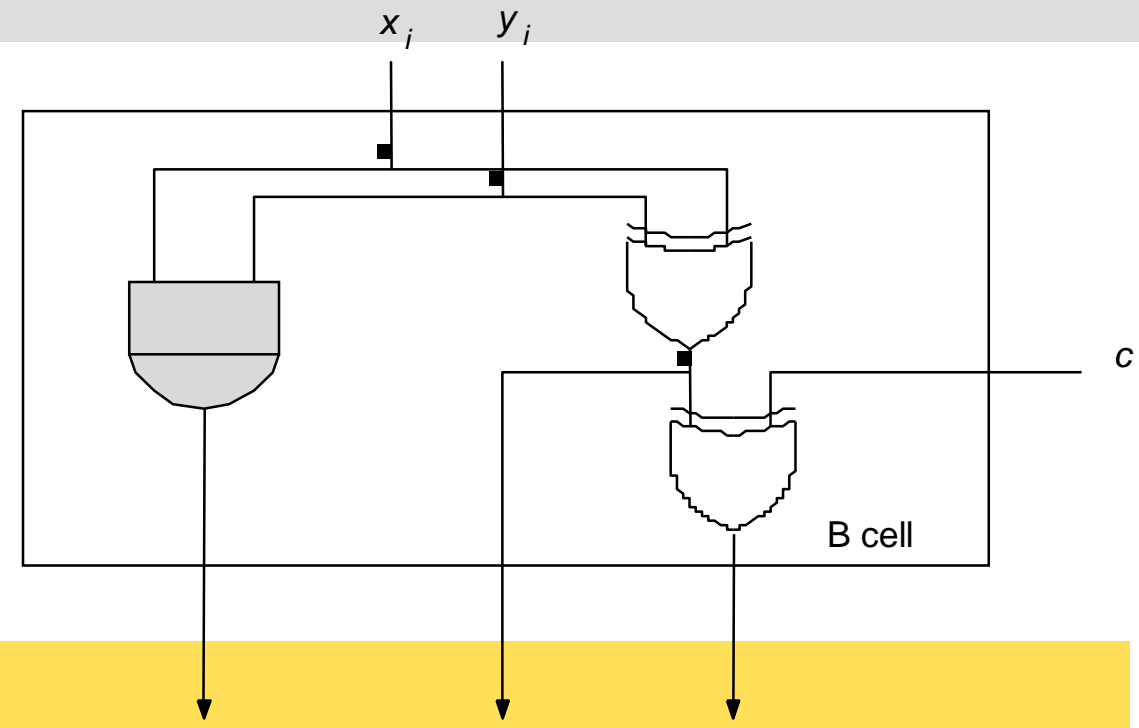$$c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + .. + P_i P_{i-1}..P_1 G_0 + P_i P_{i-1}...P_0 c_0$$

- All carries can be obtained 3 gate delays after $X, Y$ and $c_0$ are applied.
   - One gate delay for $P_i$ and $G_i$
   - Two gate delays in the AND-OR circuit for $c_{i+1}$
- All sums can be obtained 1 gate delay after the carries are computed.
- Independent of $n$, $n$-bit addition requires only 4 gate delays.
- This is called Carry Lookahead adder.

Design of fast adder/Computer organization and architecture/Dr.K.Periyakaruppan/CSE/SNSCE

# Carry-lookahead adder



4-bit carry-lookahead adder

B-cell for a single stage

# Carry lookahead adder (contd..)

- Performing $n$-bit addition in 4 gate delays independent of $n$ is good only theoretically because of fan-in constraints.

- Last AND gate and OR gate require a fan-in of (n+1) for a n-bit adder.
  - For a 4-bit adder ($n=4$) fan-in of 5 is required.
  - Practical limit for most gates.
- In order to add operands longer than 4 bits, we can cascade 4-bit Carry-Lookahead adders. Cascade of Carry-Lookahead adders is called Blocked Carry-Lookahead adder.

$$c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + .. + P_i P_{i-1}..P_1 G_0 + P_i P_{i-1}...P_0 c_0$$

# Assessment

What is

1.Generate function_____

2. Propagate function____

3.Gate delay _____

4.Overflow_____

# Reference

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", McGraw-Hill, 6$^{th}$ Edition 2012.

Design of fast adder/Computer organization and architecture/Dr.K.Periyakaruppan/CSE/SNSCE