life.augmented

# Computer Arithmetic

## Adders

Ahmed Abdelsalam

# Agenda

0 Overview

1 Digital IC Design Flow

2 Computer Arithmetic Intro

3 Basic Adder and Carry Analysis

4 Variations of Fast Adders

5 Multi-Operands Adders

life.augmented

# Training Overview

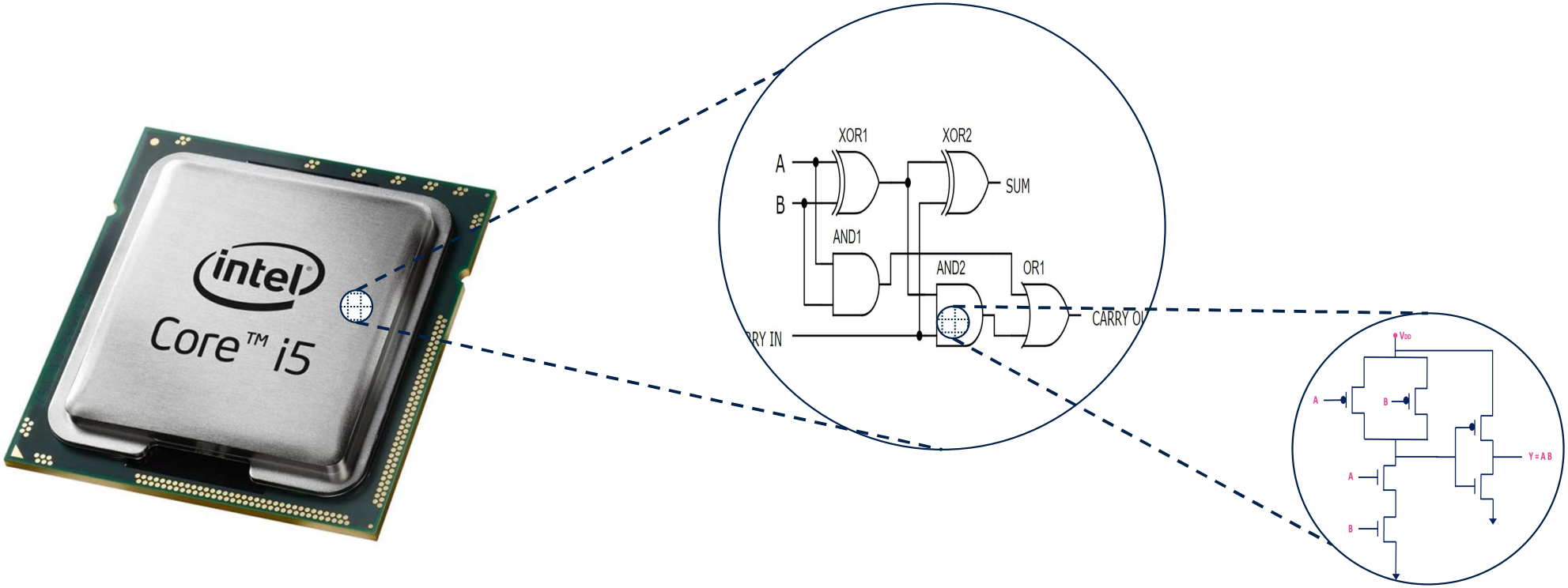
## Rules to be defined

- Online attendance is not allowed for this training; it is only conducted **face-to-face**.
- Assignments are very restricted, and it is not acceptable to miss them. Only a 12-hour extension from the deadline is allowed, and further extensions require a valid excuse.
- Honesty and transparency are essential; group solutions are not allowed.
- We have three out of four sessions in part one of the training before the first checkpoint, which may result in some participants being asked to leave (we hope this will not be necessary).
- Training certificates are based on your performance throughout the journey.

# 1- Digital IC Design Flow

life.augmented

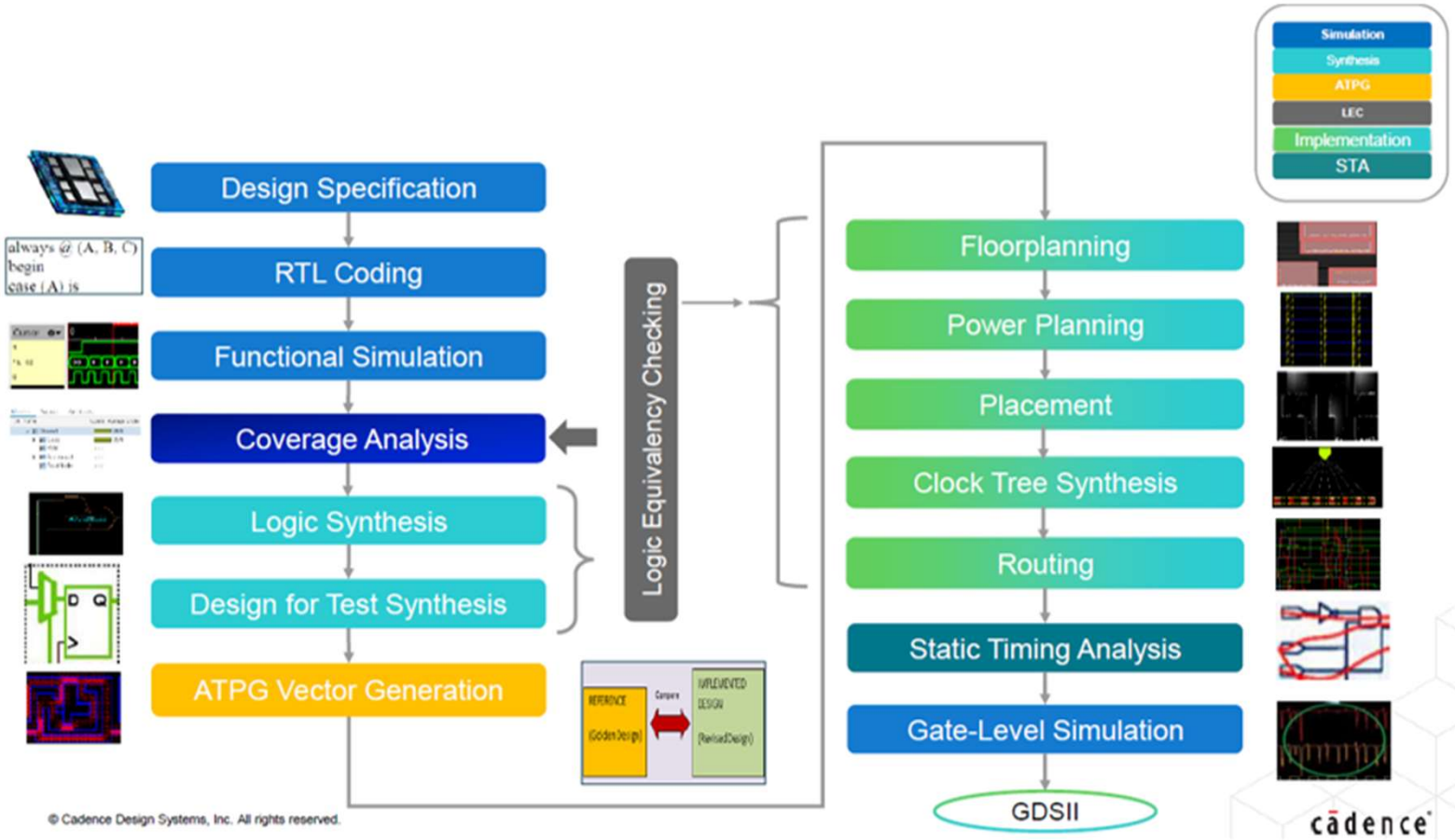
# Digital Design Flow




life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Digital Design Flow



# 2- Computer Arithmetic Intro

# Binary System

## Binary

| | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| (123) | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

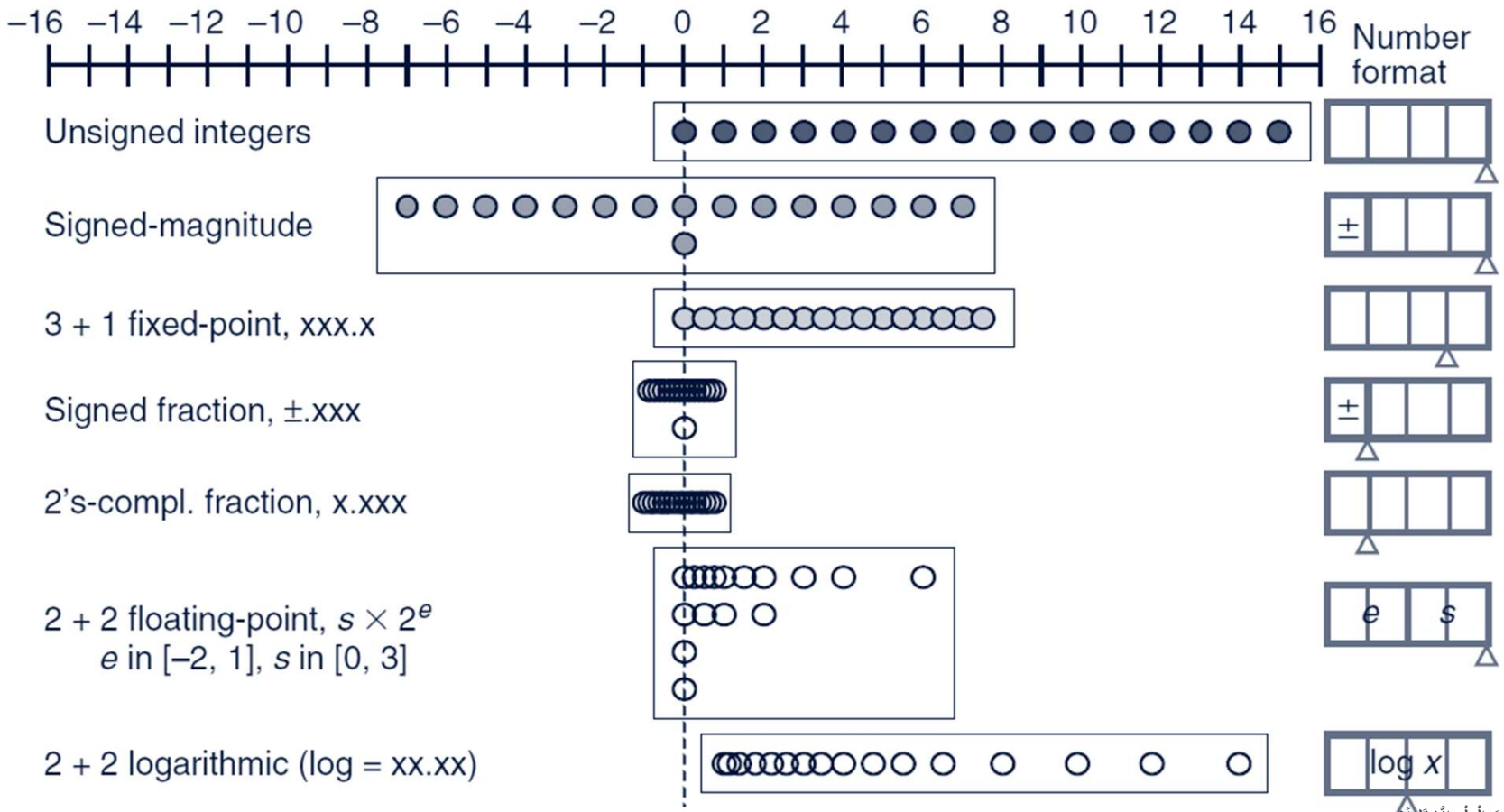
## Decimal

| 1000 | 100 | 10 | 1 |
|---|---|---|---|
| 0 | 1 | 2 | 3 |

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Number Representation



# 3- Basic Adder and Carry Analysis

# Binary Addition

**Binary**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 1 | | 1 | 1 | 1 | |
| (21) | | 0 | 1 | 0 | 1 | 0 | 1 |
| (-9) | | 1 | 1 | 0 | 1 | 1 | 1 |
| (12) | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

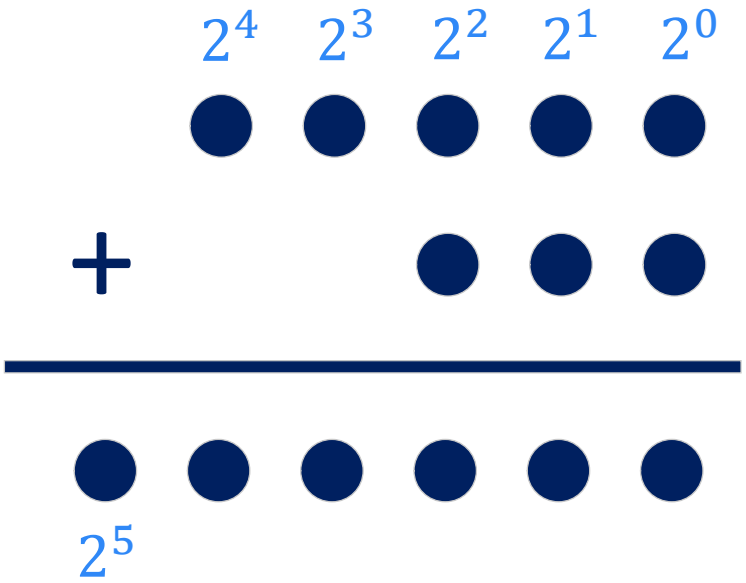
**Decimal**

| | | | |
|---|---|---|---|
| | 1 | 1 | |
| 4 | 6 | 5 | 3 |
| 2 | 1 | 6 | 7 |
| 6 | 8 | 2 | 0 |

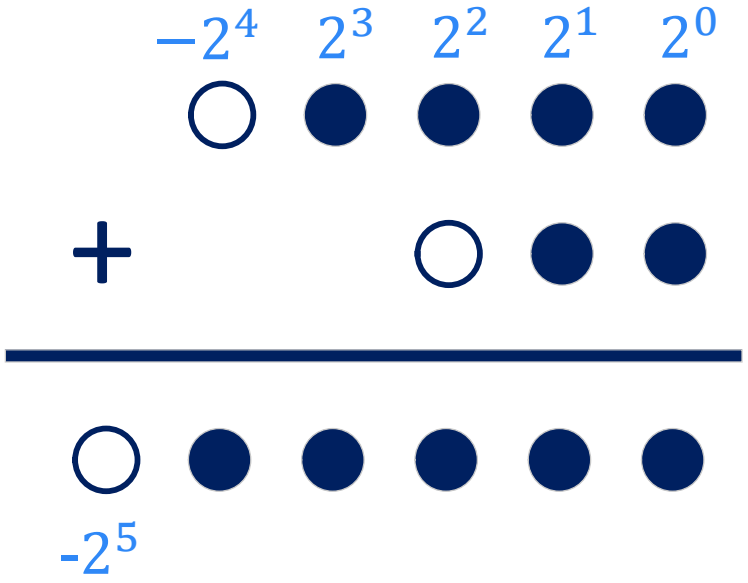
life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

## Dot-Notation



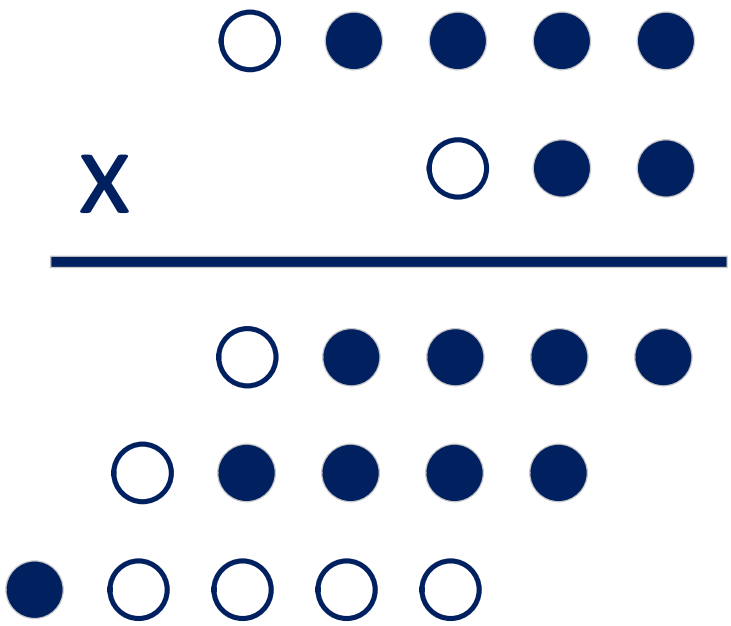

## Extended DN



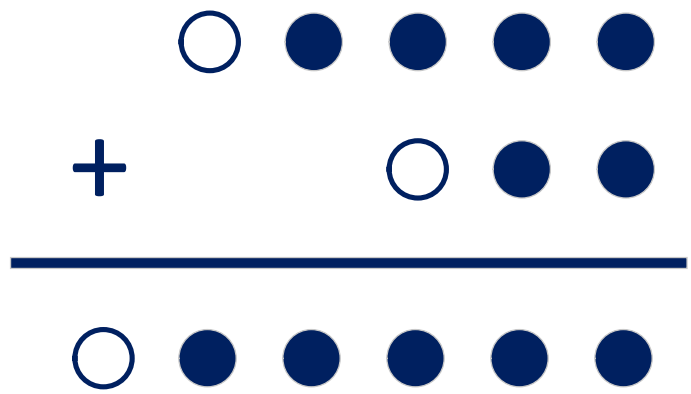

life.augmented

# Extended Dot-Notation

## Multiplication



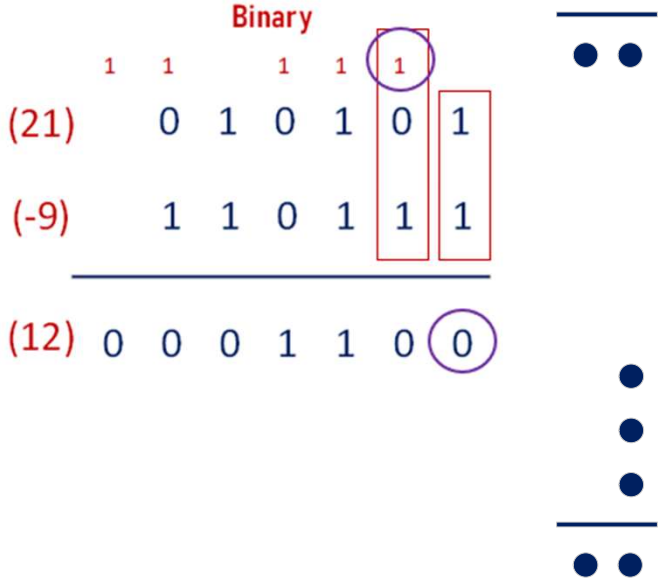
## Addition



life.augmented

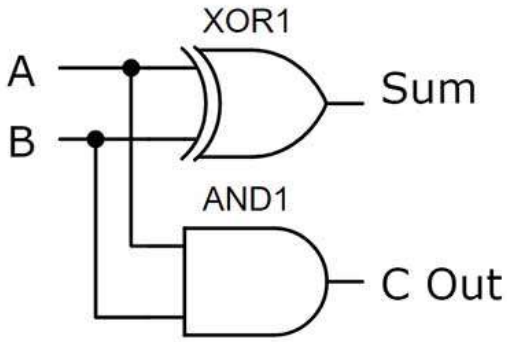
وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# One-Bit width Adder



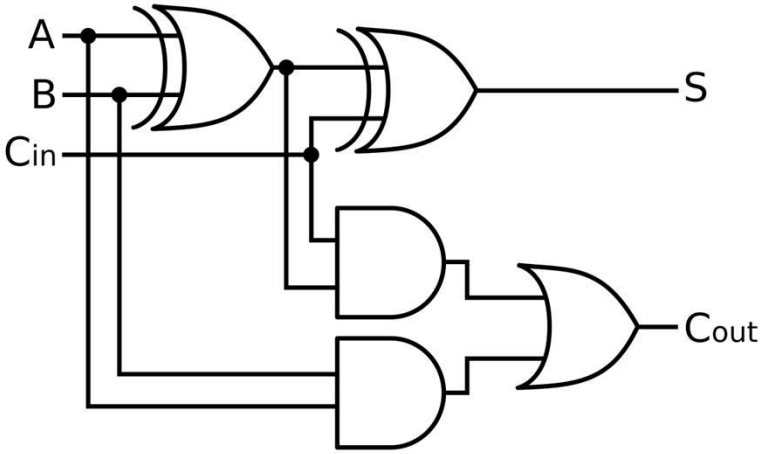

Half Adder

| A | B | C Out | Sum |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |




Full Adder

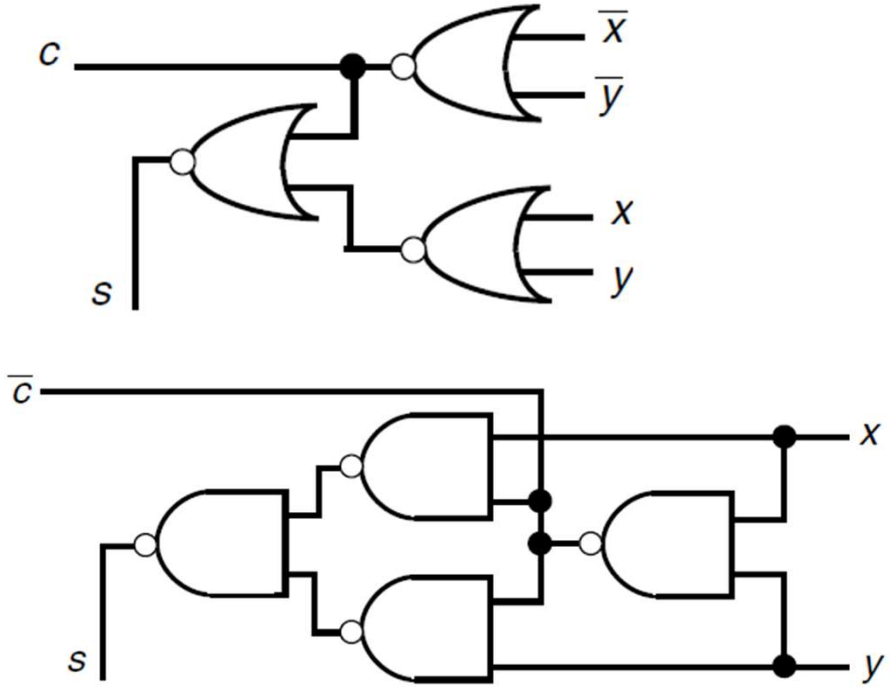
| A | B | C | C Out | Sum |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



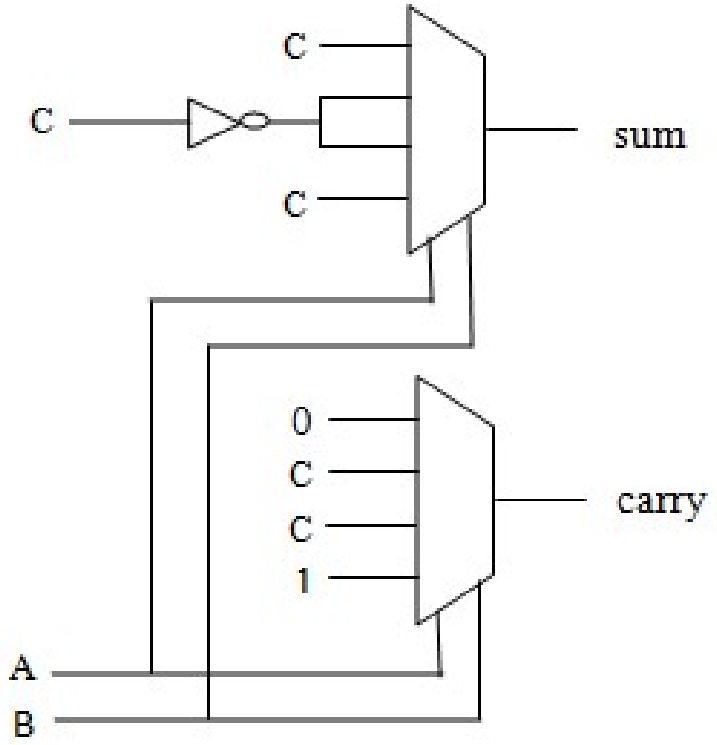

life.augmented

وما أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# One-Bit width Adder




Half Adder variants

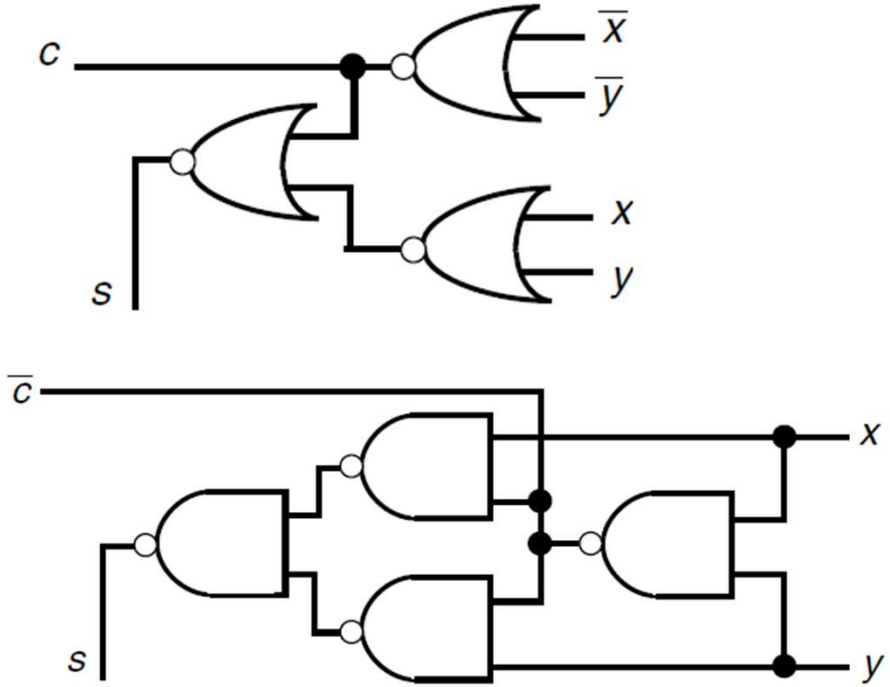



Full Adder (CMOS Suitable)

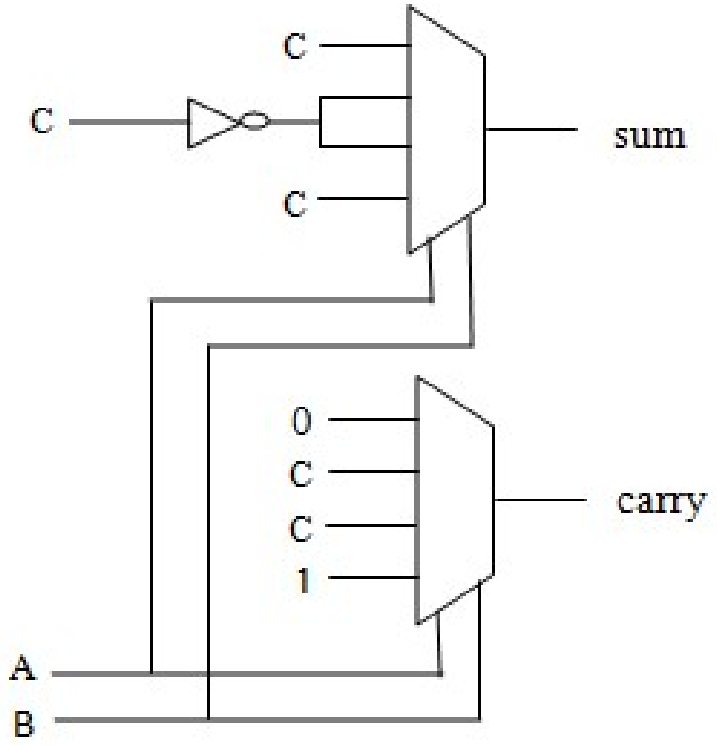
life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# One-Bit width Adder




Half Adder variants

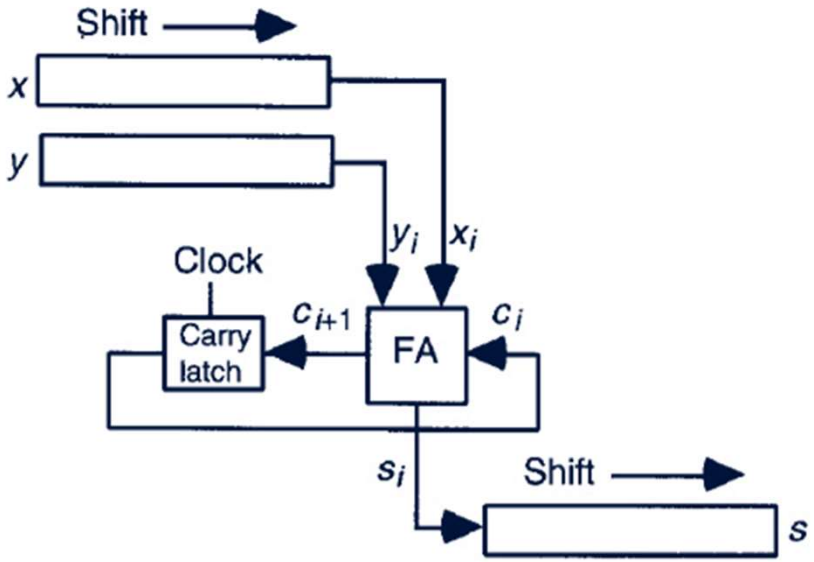


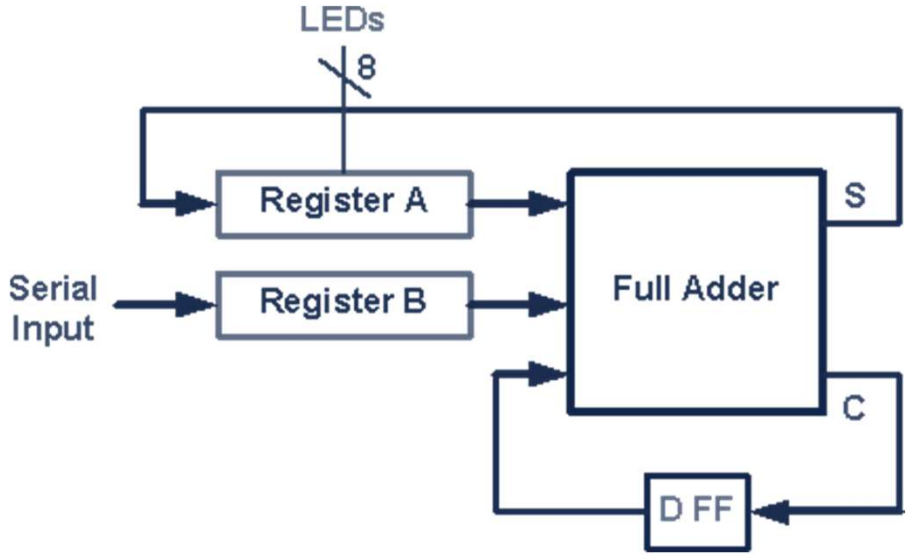

Full Adder (CMOS Suitable)

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Bit-Serial Adder




$$T_{\text{bit-serial}} = n * T_{FA}(c_{in} \rightarrow s)$$

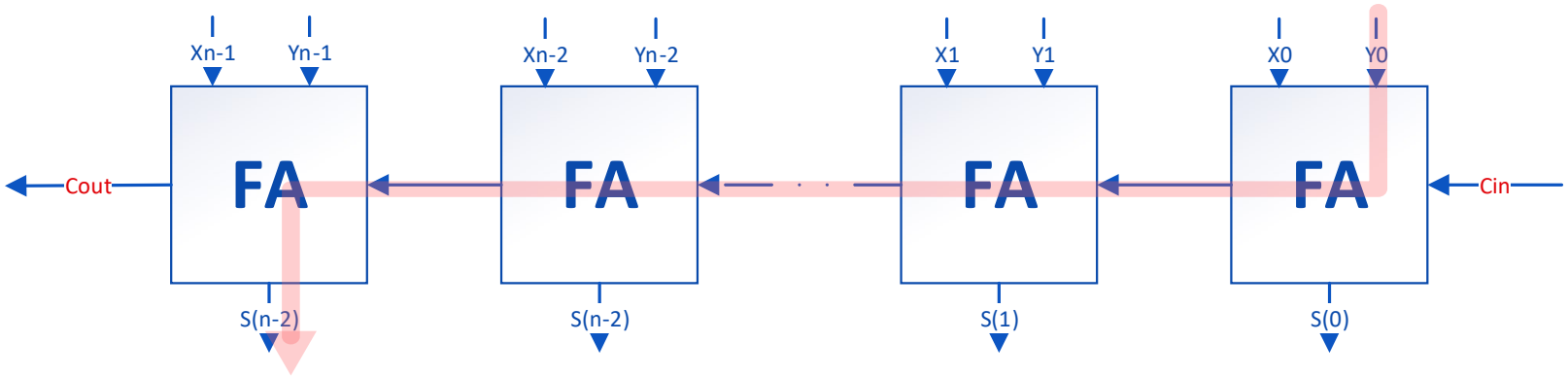
$$Thruput_{\text{pipeline}} = 1/T_{FA}(c_{in} \rightarrow s)$$

$$Area_{\text{bit-serial}} = A_{FA} + (3n+1) * A_{FF}$$

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Carry Ripple Adder



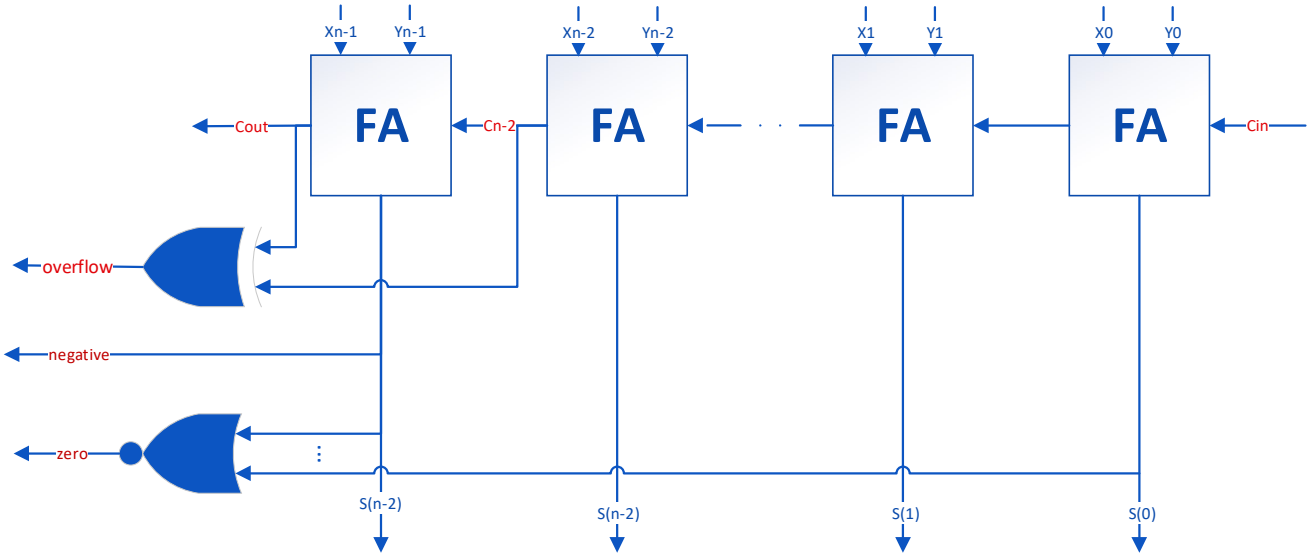

$$T_{\text{ripple-add}} = T_{FA}(x,y \rightarrow c_{out}) + (k-2) \times T_{FA}(c_{in} \rightarrow cout) + T_{FA}(c_{in} \rightarrow s)$$

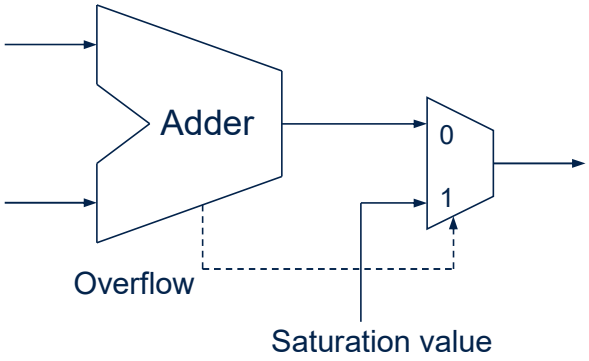
$$Area_{\text{bit-serial}} = n * A_{FA}$$

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Exceptions and Saturating Adder

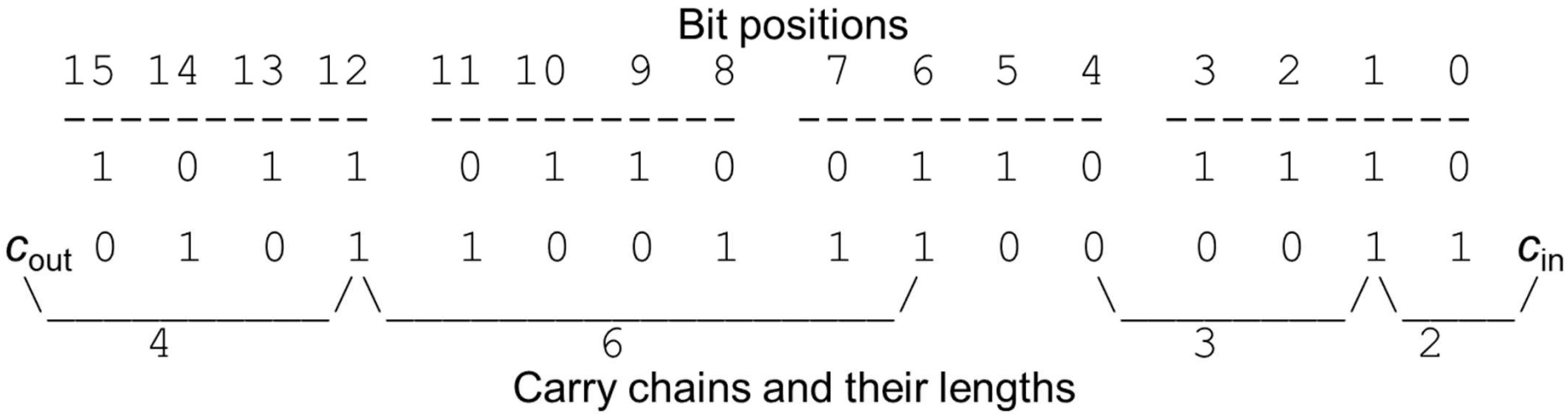



life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Carry Length




life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Carry Length Expectation

Given binary numbers with random bits, for each position $i$ we have

- Probability of carry generation = ¼ (both 1s)
- Probability of carry annihilation = ¼ (both 0s)
- Probability of carry propagation = ½ (different)

Probability that carry generated at position $i$ propagates through position $j - 1$ and stops at position $j$ ($j > i$)

$$2^{-(j-1-i)} \times 1/2 = 2^{-(j-i)}$$

Expected length of the carry chain that starts at position $i$

$$2 - 2^{-(k-i-1)}$$

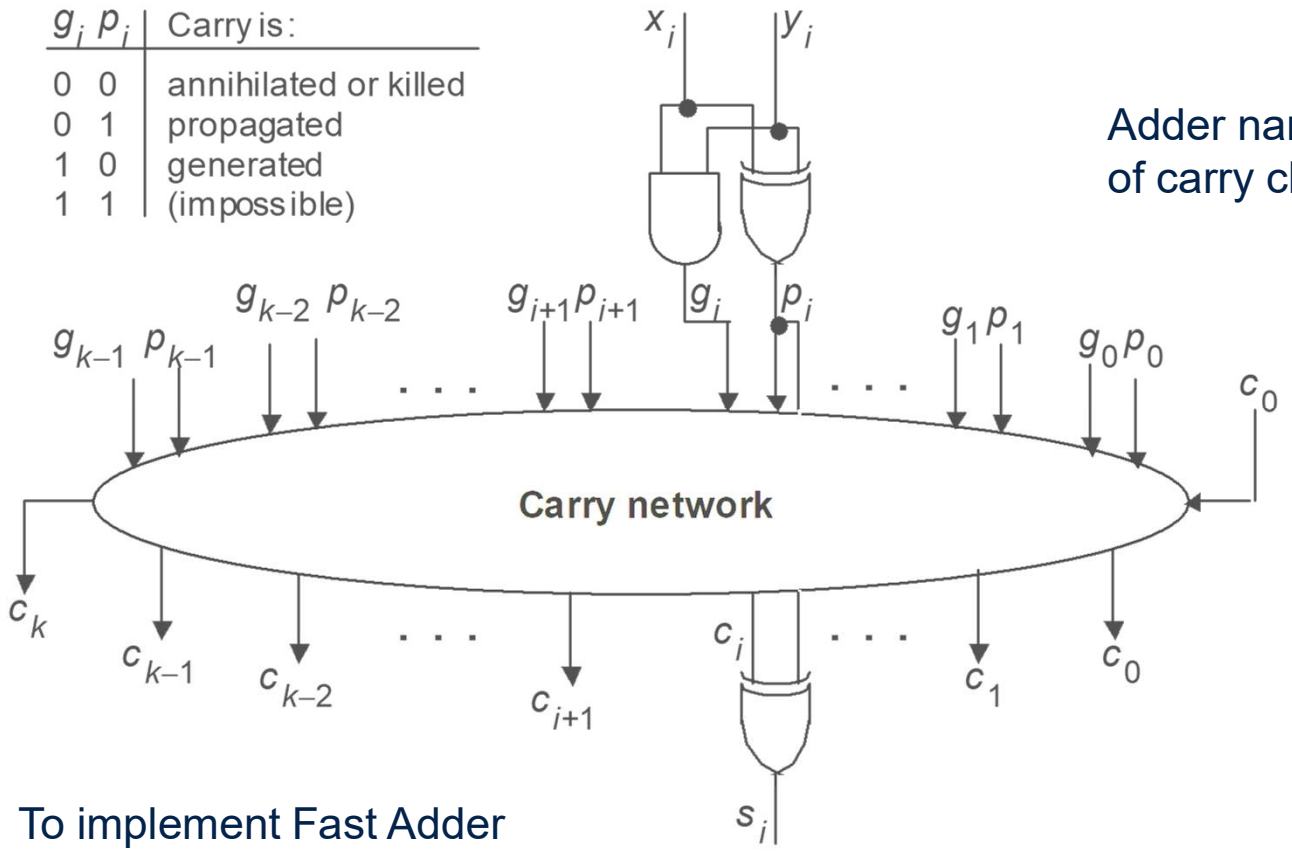
Average length of the longest carry chain in $k$-bit addition is strictly less than $\log_2 k$; it is $\log_2(1.25k)$ per experimental results

Analogy: Expected number when rolling one die is 3.5; if one rolls many dice, the expected value of the largest number shown grows

# 4- Variations of Fast Adders

# Carry Network the Key

| $g_i$ | $p_i$ | Carry is: |
|---|---|---|
| 0 | 0 | annihilated or killed |
| 0 | 1 | propagated |
| 1 | 0 | generated |
| 1 | 1 | (impossible) |




Adder named by type of carry chain

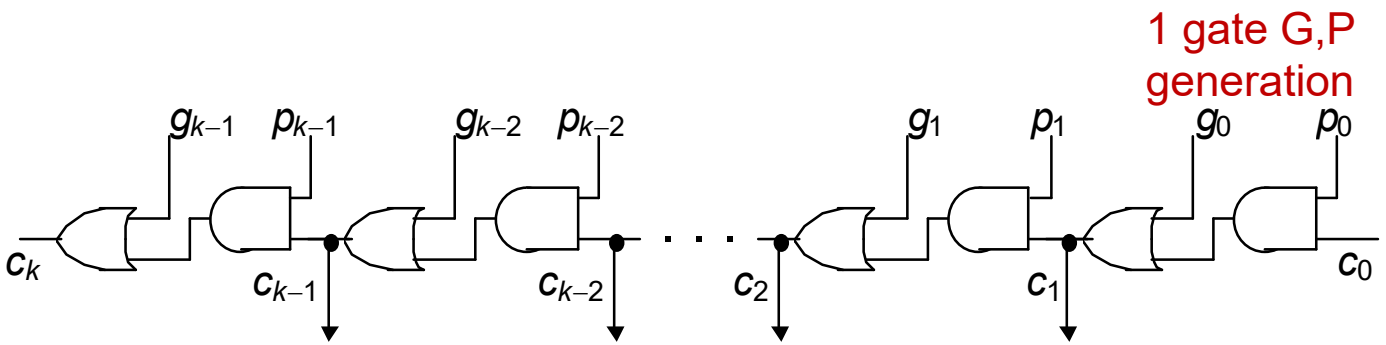
To implement Fast Adder need to optimize carry chain

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Revisit carry Ripple Adders

The carry recurrence: $c_{i+1} = g_i \vee p_i\, c_i$




life.augmented

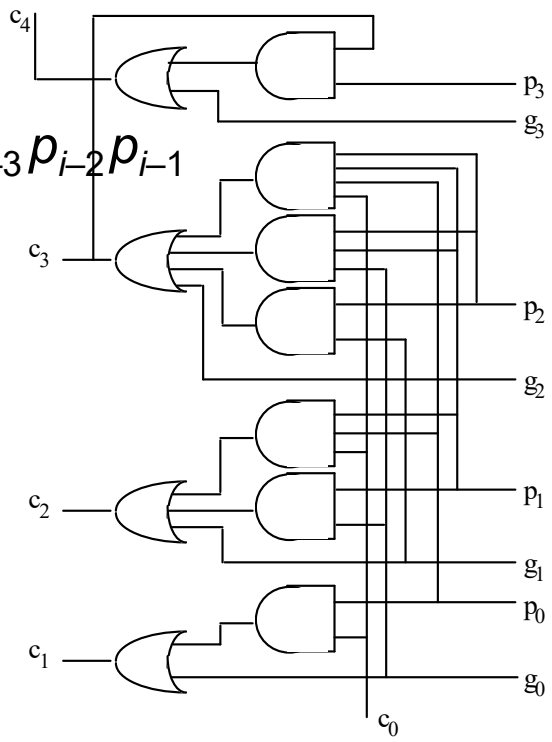
وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Unrolling Carry

$$
\begin{aligned}
c_i &= g_{i-1} \vee c_{i-1} p_{i-1} \\
&= g_{i-1} \vee (g_{i-2} \vee c_{i-2} p_{i-2}) p_{i-1} \\
&= g_{i-1} \vee g_{i-2} p_{i-1} \vee c_{i-2} p_{i-2} p_{i-1} \\
&= g_{i-1} \vee g_{i-2} p_{i-1} \vee g_{i-3} p_{i-2} p_{i-1} \vee c_{i-3} p_{i-3} p_{i-2} p_{i-1} \\
&= g_{i-1} \vee g_{i-2} p_{i-1} \vee g_{i-3} p_{i-2} p_{i-1} \vee g_{i-4} p_{i-3} p_{i-2} p_{i-1} \vee c_{i-4} p_{i-4} p_{i-3} p_{i-2} p_{i-1} \\
&= \ldots
\end{aligned}
$$

Full carry lookahead is quite practical for a 4-bit adder

$$
\begin{aligned}
c_1 &= g_0 \vee c_0 p_0 \\
c_2 &= g_1 \vee g_0 p_1 \vee c_0 p_0 p_1 \\
c_3 &= g_2 \vee g_1 p_2 \vee g_0 p_1 p_2 \vee c_0 p_0 p_1 p_2 \\
c_4 &= g_3 \vee g_2 p_3 \vee g_1 p_2 p_3 \vee g_0 p_1 p_2 p_3 \\
&\qquad \vee c_0 p_0 p_1 p_2 p_3
\end{aligned}
$$




life.augmented

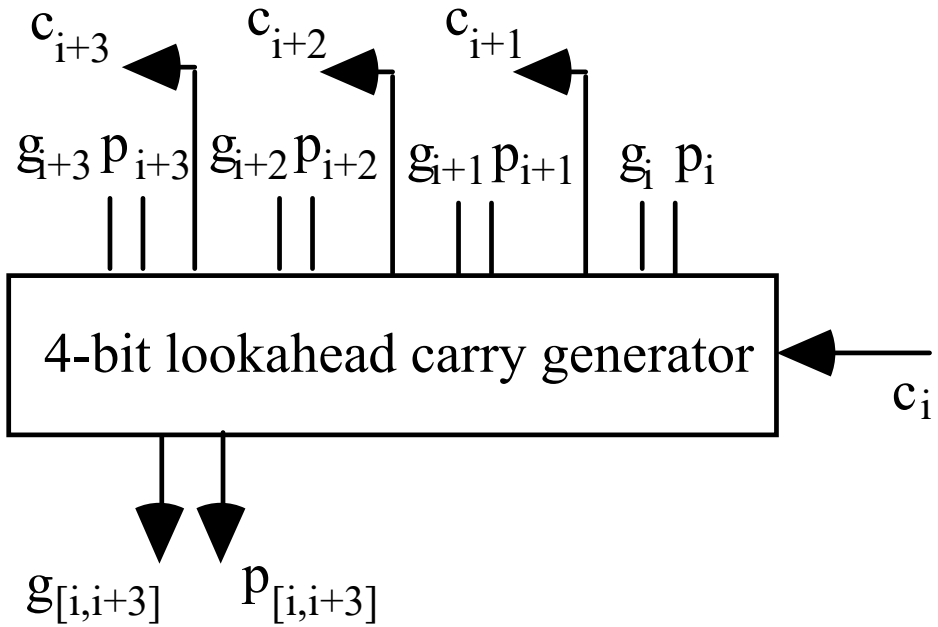
وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Carry-Lookahead Adder

Block *generate* and *propagate* signals

$$g_{[i,i+3]} = g_{i+3} \vee g_{i+2}p_{i+3} \vee g_{i+1}p_{i+2}p_{i+3} \vee g_i\, p_{i+1}p_{i+2}p_{i+3}$$

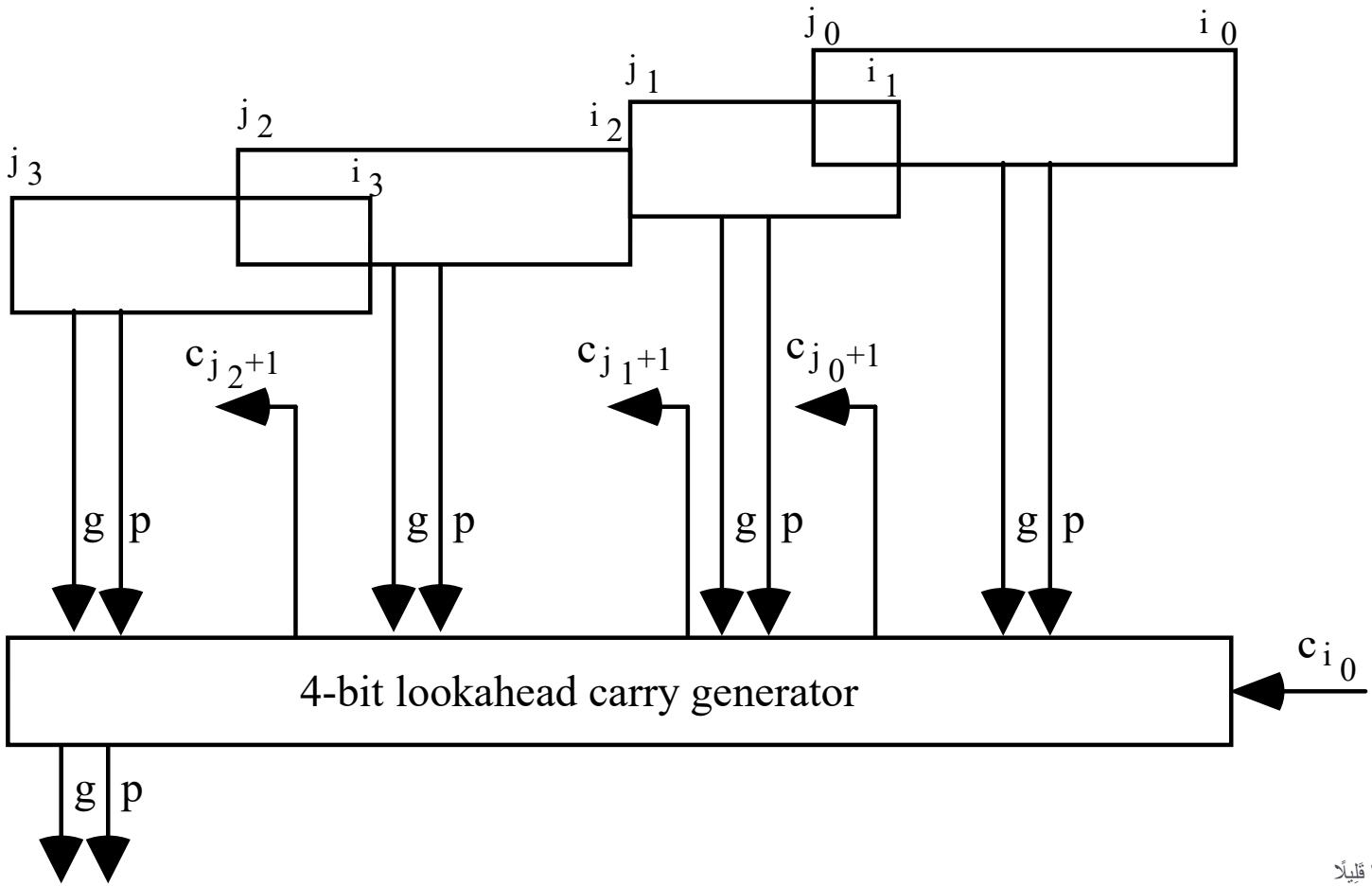
$$p_{[i,i+3]} = p_i\, p_{i+1}p_{i+2}p_{i+3}$$




life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Combine G,P Signals

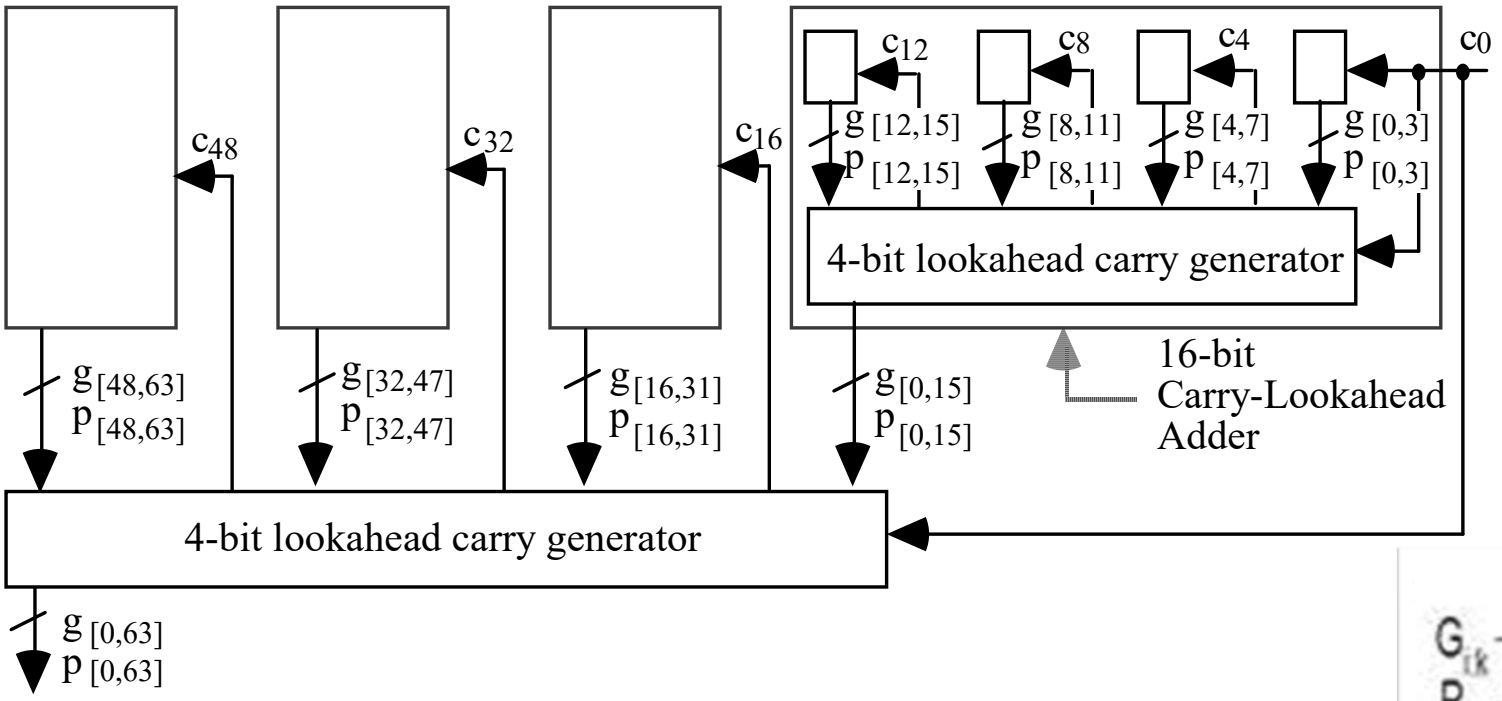


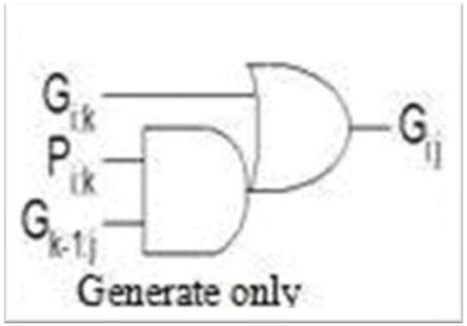

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Multilevel Carry Look Ahead

$T_{lookahead\text{-}add}$ = 4 log4k + 1 gate levels




- 1 Gate for G,P Generation
- 2 Gates for CLA computation (for each level)
- 2 Gates for Combining (for each level)
- 1 Gate for Sum Calculation

life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Ling Adder

Consider the carry recurrence and its unrolling by 4 steps:

$$c_i = g_{i-1} \vee c_{i-1} t_{i-1}$$
$$= g_{i-1} \vee g_{i-2} t_{i-1} \vee g_{i-3} t_{i-2} t_{i-1} \vee g_{i-4} t_{i-3} t_{i-2} t_{i-1} \vee c_{i-4} t_{i-4} t_{i-3} t_{i-2} t_{i-1}$$

Ling's modification: Propagate $h_i = c_i \vee c_{i-1}$ instead of $c_i$

$$h_i = g_{i-1} \vee h_{i-1} t_{i-2}$$
$$= g_{i-1} \vee g_{i-2} \vee g_{i-3} t_{i-2} \vee g_{i-4} t_{i-3} t_{i-2} \vee h_{i-4} t_{i-4} t_{i-3} t_{i-2}$$

| | | | |
|---|---|---|---|
| CLA: | 5 gates | max 5 inputs | 19 gate inputs |
| Ling: | 4 gates | max 5 inputs | 14 gate inputs |

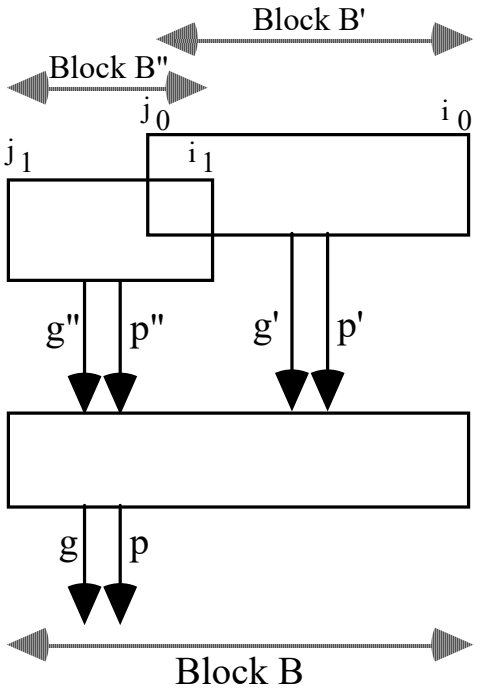
The advantage of $h_i$ over $c_i$ is even greater with wired-OR:

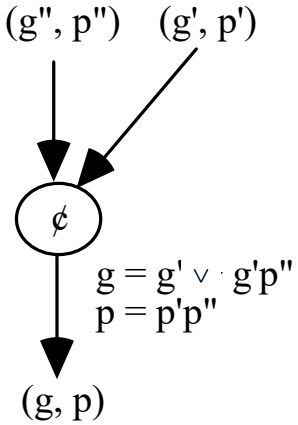
| | | | |
|---|---|---|---|
| CLA: | 4 gates | max 5 inputs | 14 gate inputs |
| Ling: | 3 gates | max 4 inputs | 9 gate inputs |

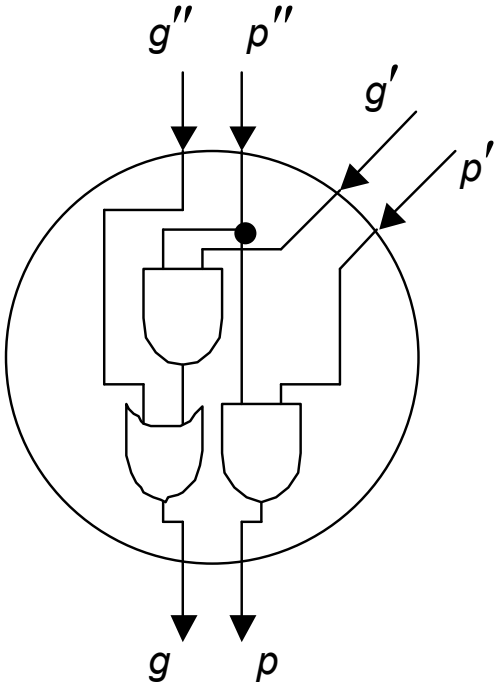
Once $h_i$ is known, however, the sum is obtained by a slightly more complex expression compared with $s_i = p_i \oplus c_i$

$$s_i = p_i \oplus h_i \, t_{i-1}$$

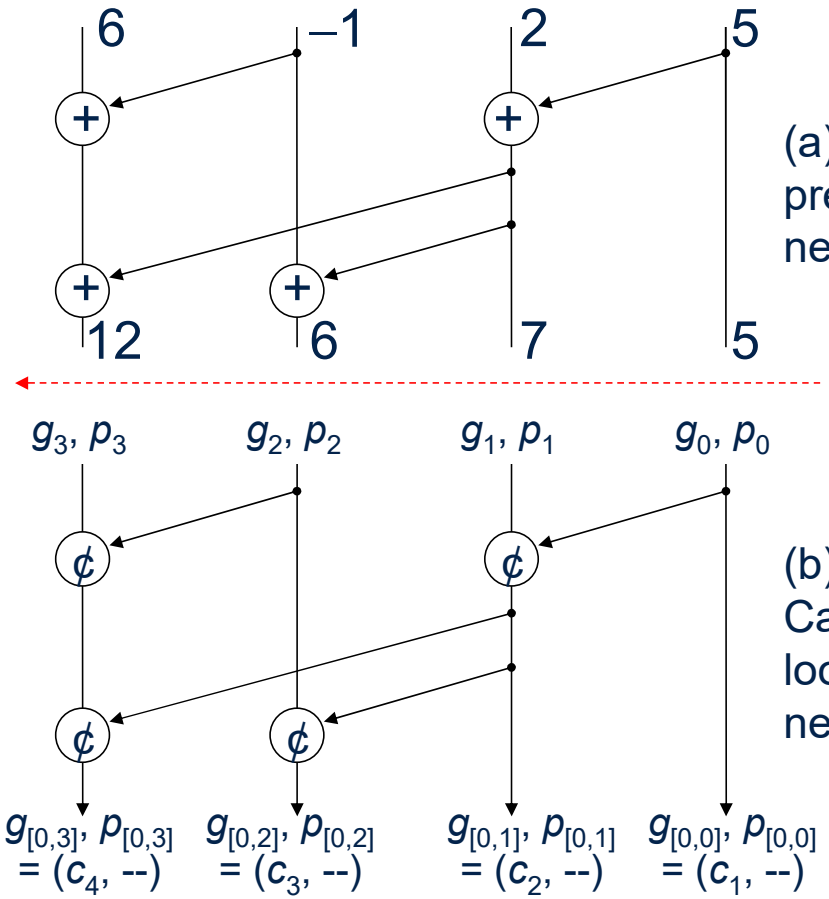
# Carry Determination as Prefix Computation



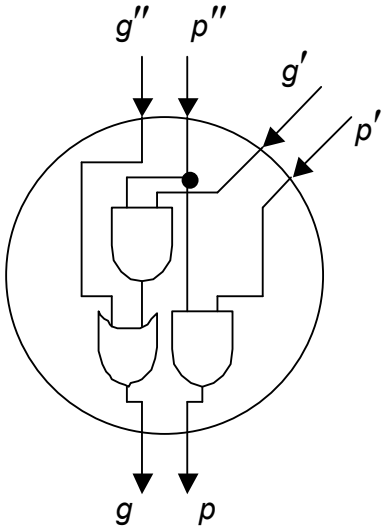



# Prefix-Based Carry Network




(a) A 4-input prefix sums network

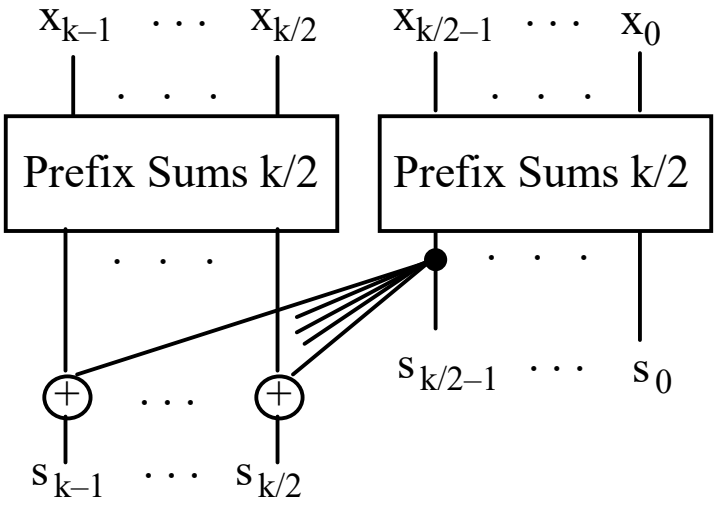
(b) A 4-bit Carry lookahead network




life.augmented

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

# Parallel Prefix Networks

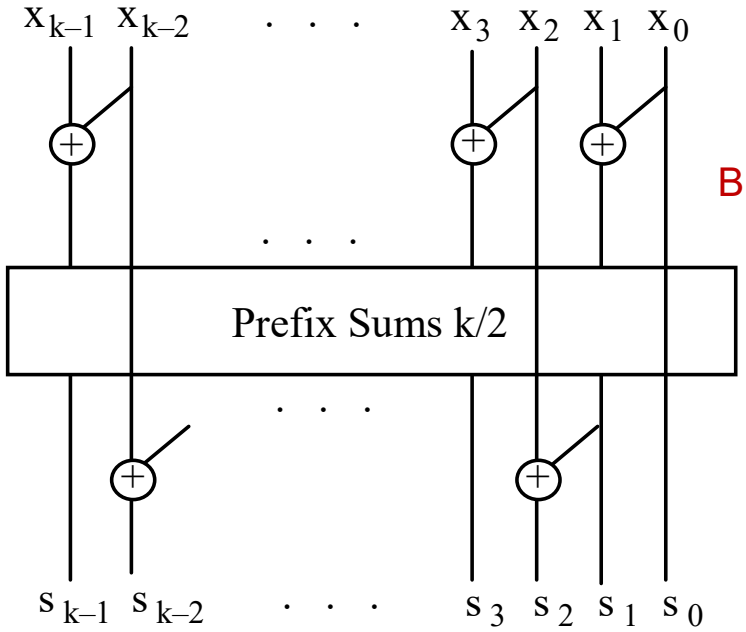



Delay recurrence

$$D(k) = D(k/2) + 1 = \log_2 k$$

Cost recurrence

$$C(k) = 2C(k/2) + k/2 = (k/2)\log_2 k$$




Brent-Kung

Delay recurrence

$$D(k) = D(k/2) + 2 = 2\log_2 k - 1 \quad (-2 \text{ really})$$

Cost recurrence

$$C(k) = C(k/2) + k - 1 = 2k - 2 - \log_2 k$$

وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا

life.augmented