Lecture 11
EE 421 / CS 425
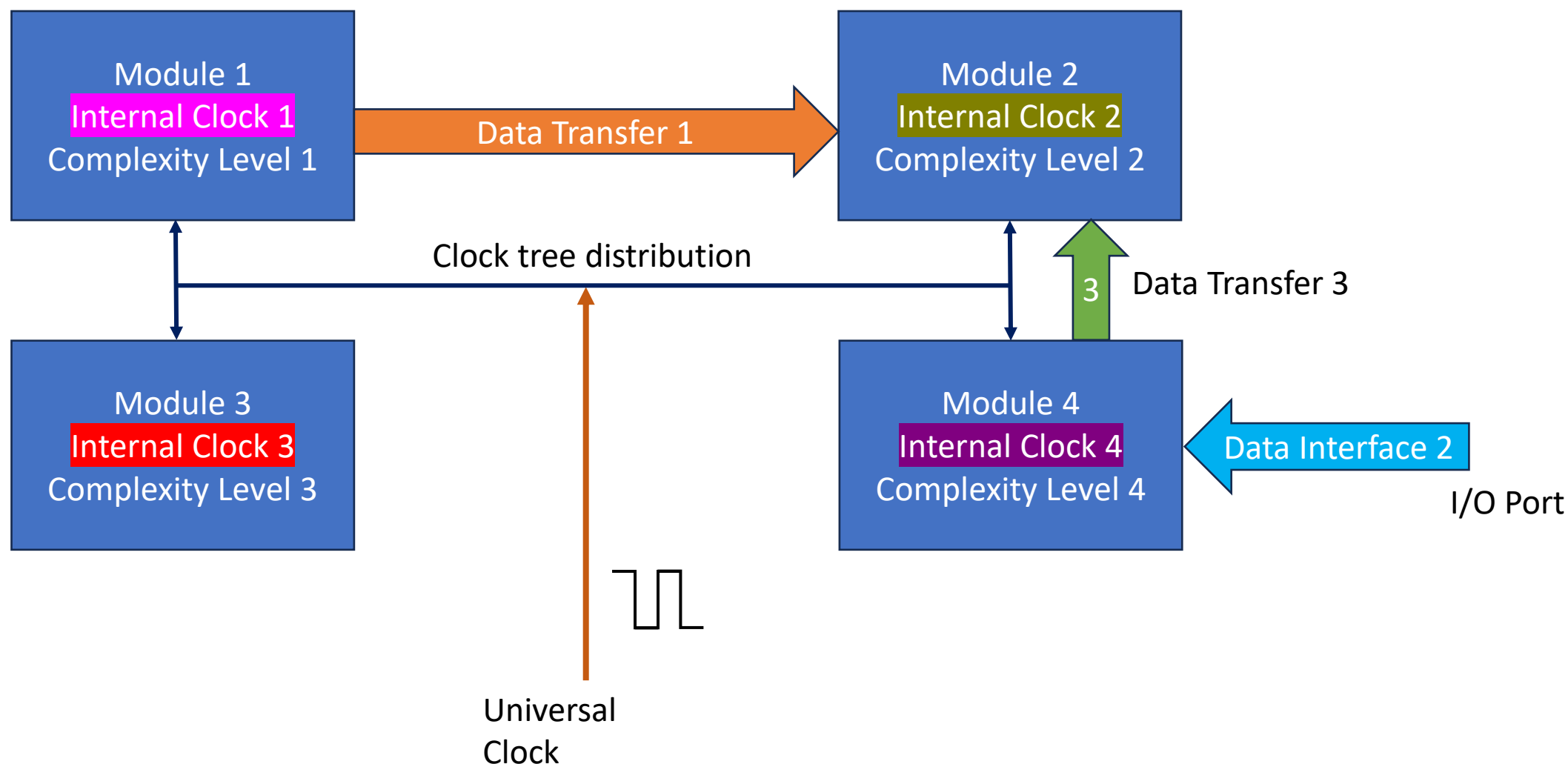# Digital System Design

Fall 2025

Shahid Masud

LUMS

Topics

- Communication Signaling across modules at different clock speeds (Revision – How clock is good and can be bad?) Quick Review

- ---

- Computer Arithmetic Circuits

- Basic Element – Full Adder

- Ripple Carry Adder – Timing Issues

- Bit Serial Adder – Concept of Throughput and Latency

- Carry Look Ahead Adder

- Carry Select Adder

Quiz 3 Next Week

LUMS
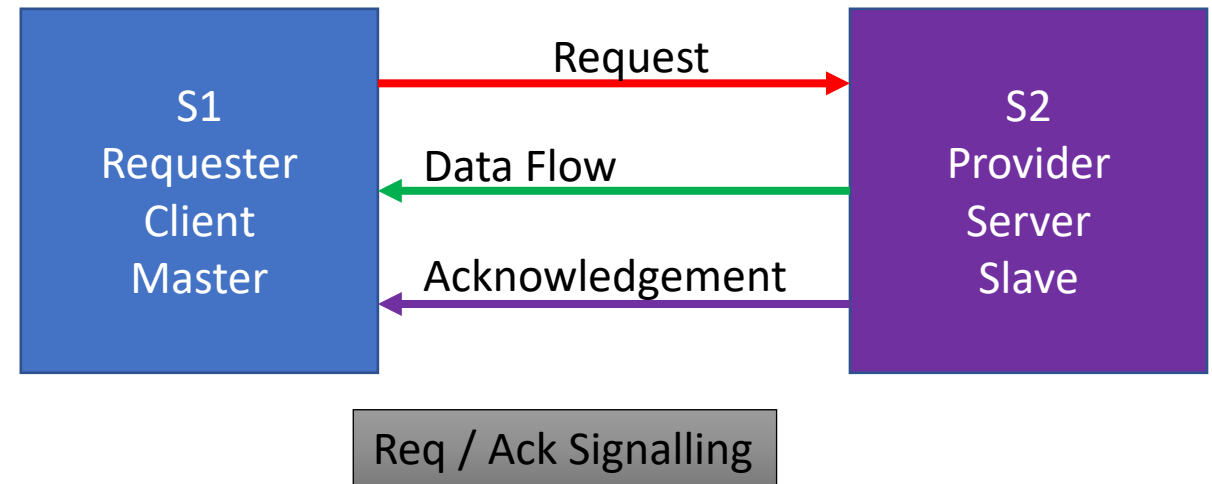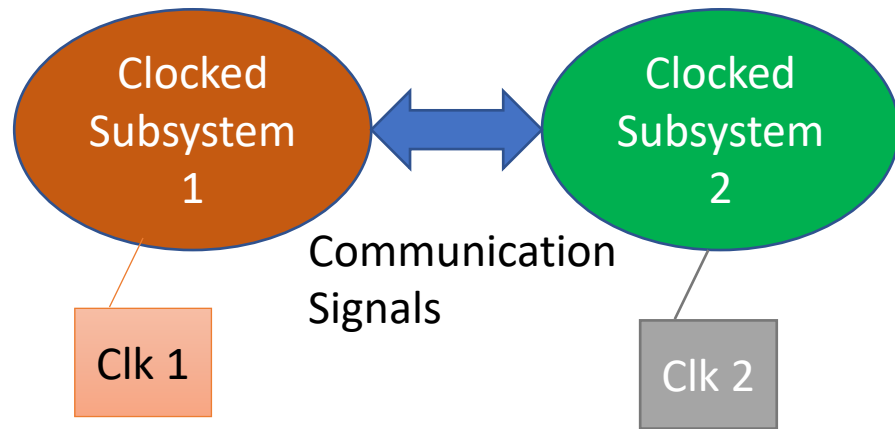
Self-Timed and Speed Independent Circuits

- Having a completed Synchronous system is at times too challenging for a complex and fast digital system

- The limiting problem becomes how to distribute a single global clock without introducing intolerable clock skew

- The alternate is to partition the digital system into locally clocked pieces that communicate with each other using delay-insensitive signaling techniques (i.e. local clock for local communication)

- Each block proceeds at its own speed without the need for a global clock, synchronizing local communication whenever needed

- Usually a Request-Response Signaling method is employed

LUMS

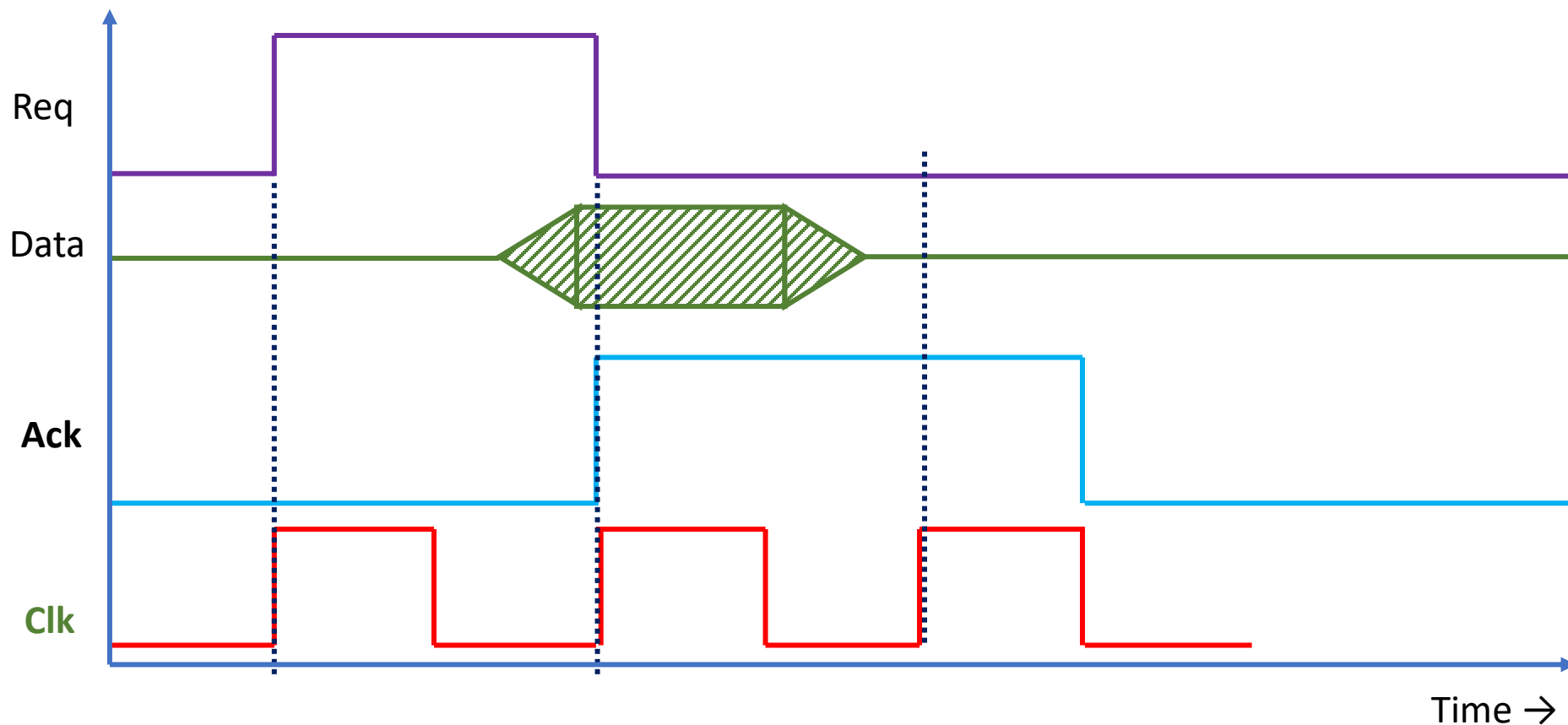# Data Transfer in Complex Modular Design

Request / Acknowledge Signaling
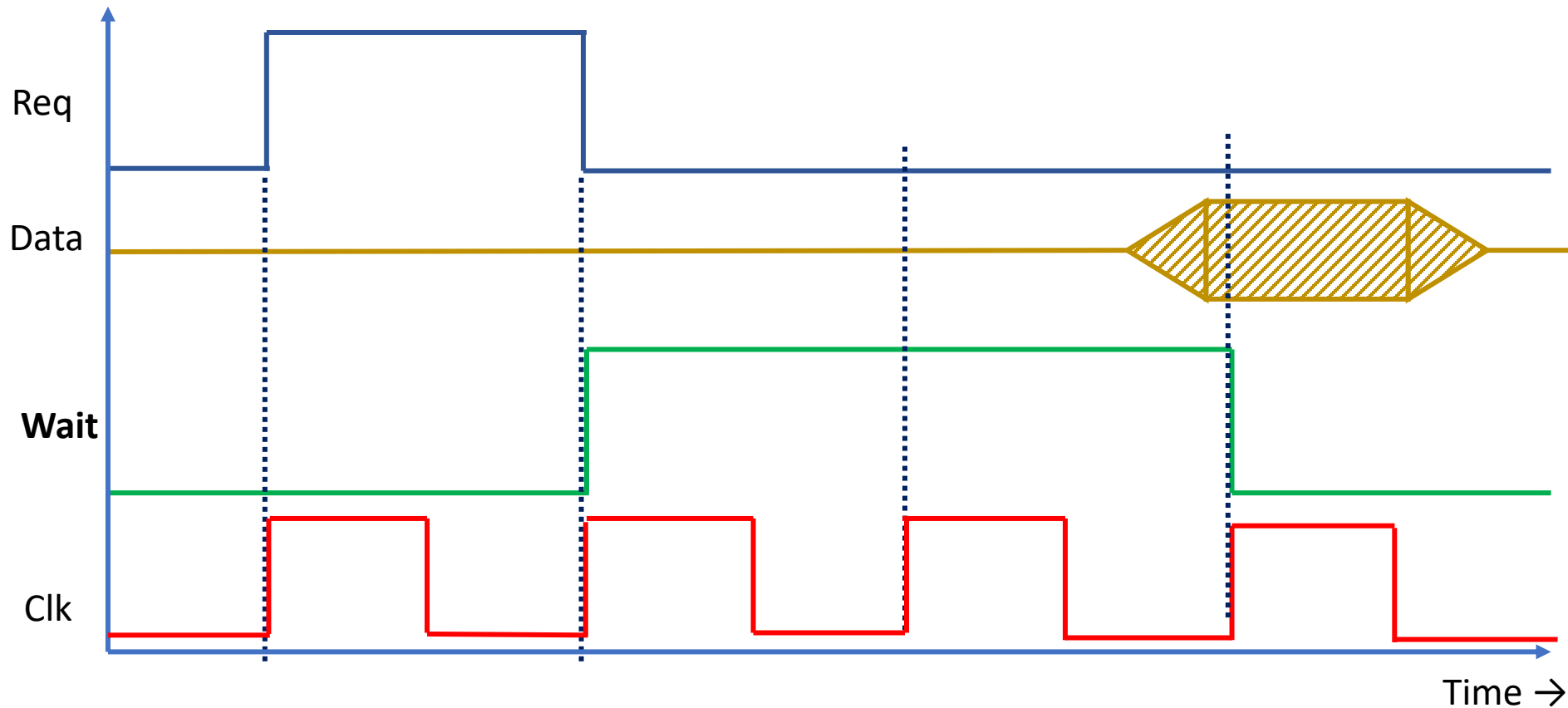
Independently clocked Subsystems

Clocked Subsystem 1

Clocked Subsystem 2

Communication Signals

Clk 1

Clk 2

S1
Requester
Client
Master

S2
Provider
Server
Slave

Request

Data Flow

Acknowledgement

Req / Ack Signalling

LUMS

Synchronous Req / Ack Signaling

Req

Data

Ack

Clk

Time →

Req and Ack signals are synchronized to **Clk**

LUMS

Synchronous Req with Wait Signaling

Slave can delay the Master by asserting Wait signal as it prepares the data and needs more clock cycles
When the slave un-asserts Wait signal, it acknowledges that data is now available for the Master to read
All interface signals are synchronized with Clock edge

Four Cycle Asynchronous Signaling

RTZ – Return to Zero Signaling with No Clock

Two Cycle Asynchronous Signaling

Non-Return-to-Zero Signalling

Req

Data

Data 1

Data 2

Ack

1

2

3

Time →
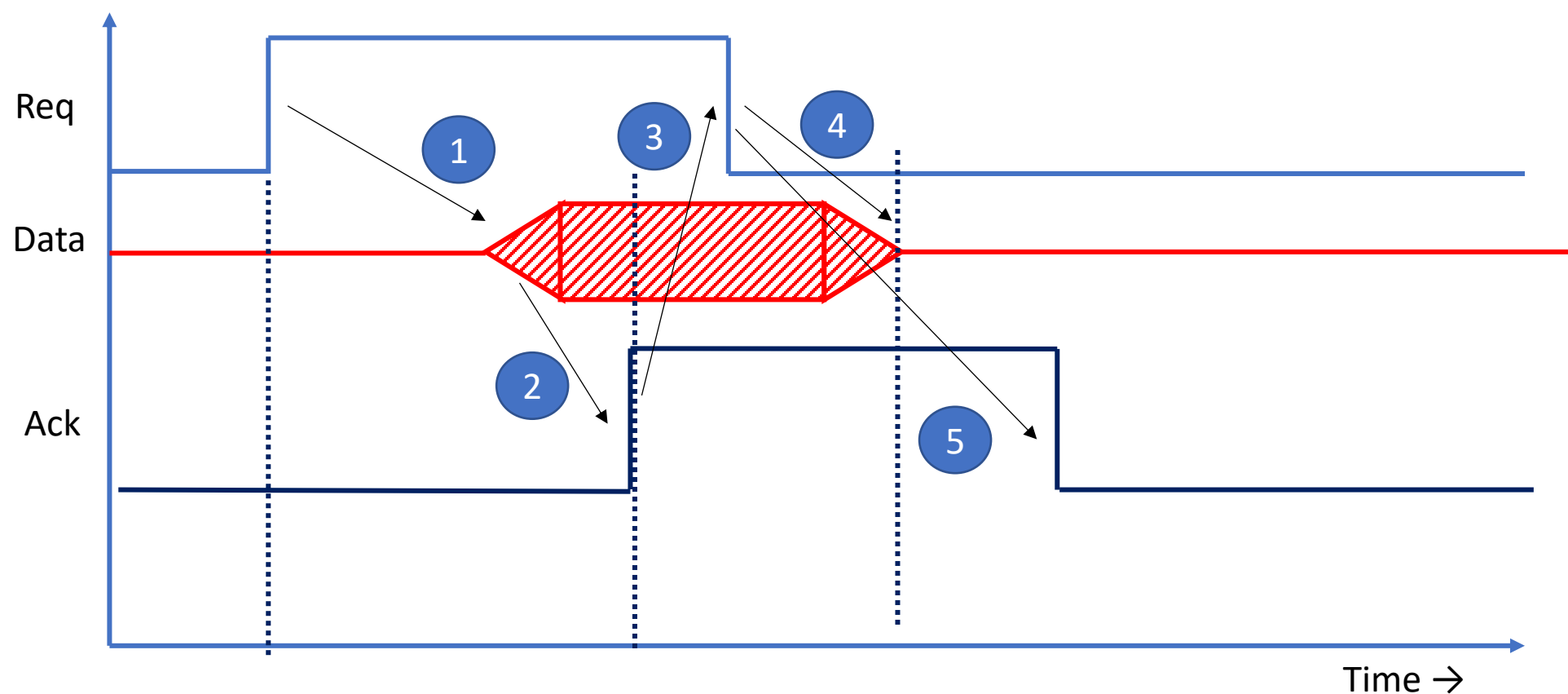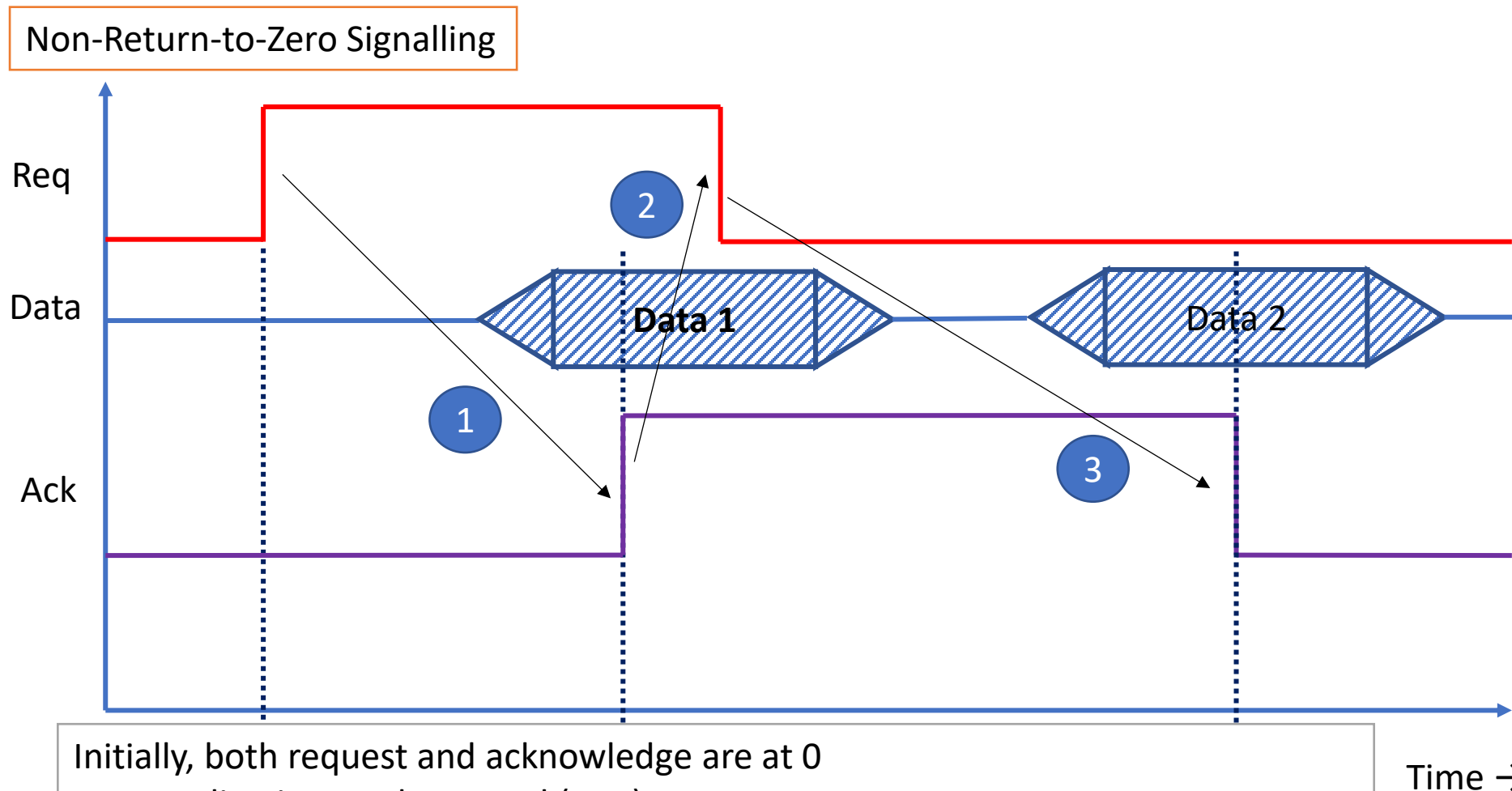
Initially, both request and acknowledge are at 0
Request line is complemented (to 1)
Slave notices change in Req, provides data, and complements the Ack line (to 1)
Further request and acknowledgement is by complementing the respective state

LUMS

Self-Timed Circuits

- A self-timed circuit can determine on its own when a request has been serviced by mimicking the worst-case propagation delay path by using special logic to delay the request signal.

- This guarantees sufficient time to compute the correct output.

Computer Arithmetic

LUMS
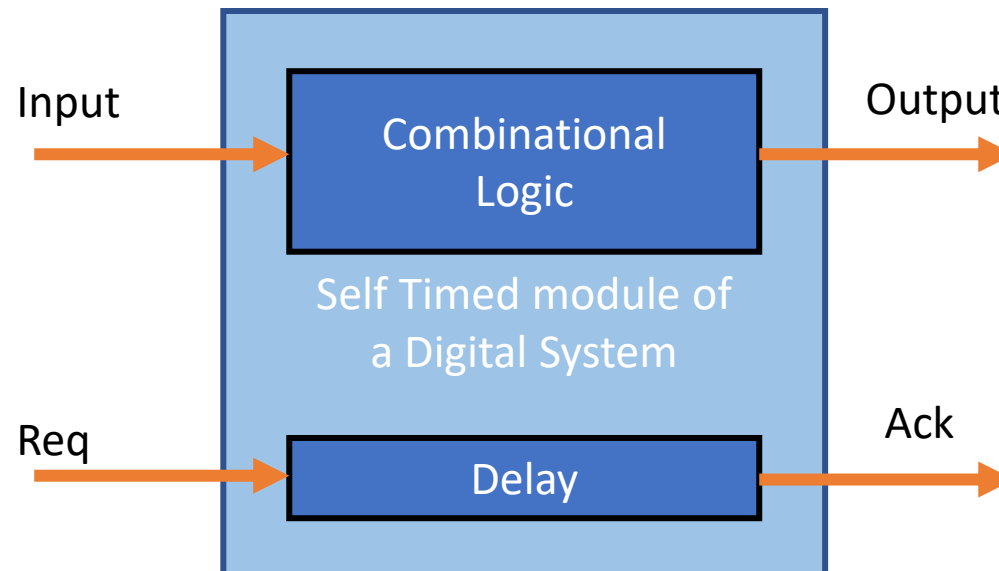
Computer Arithmetic

Study Data path and Control path of a binary hardware arithmetic unit

Computer Arithmetic Circuits

- **Generally, the digital system design falls into one of the following categories:**
  - **Data handling** **– acquisition, storage, retrieval, compression, search, etc.**
  - **Data processing** **– filtering, de-noising, object detection, pattern recognition**
- **Computer Arithmetic is an important functional block of both aspects, especially plays an important part in Data Processing**
- **Efficient arithmetic circuits can improve the speed – area – energy performance nexus in a complex digital system**

LUMS

Basic Functional Block – Full Adder

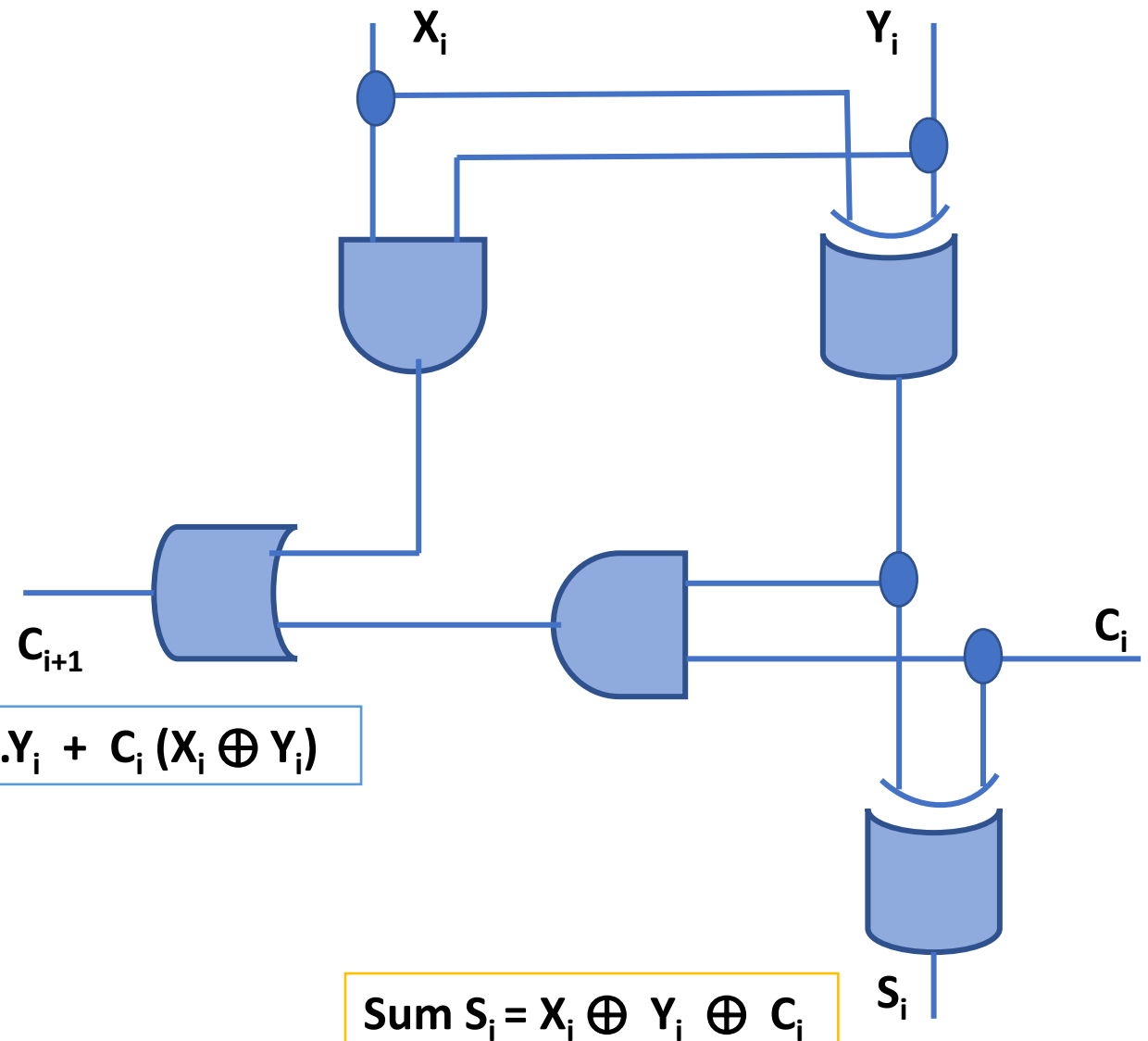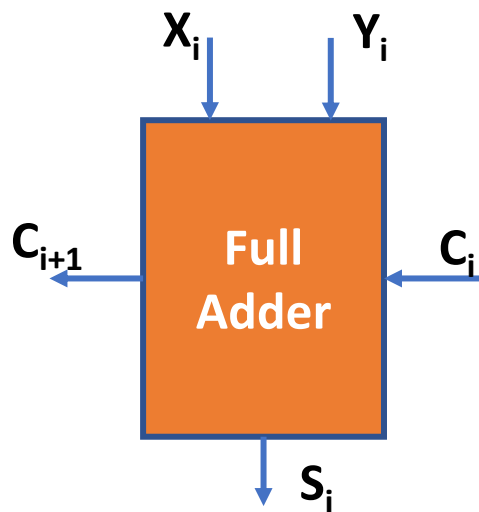| X_i	Y_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\text{Sum } S_i = X_i \oplus Y_i \oplus C_i$$

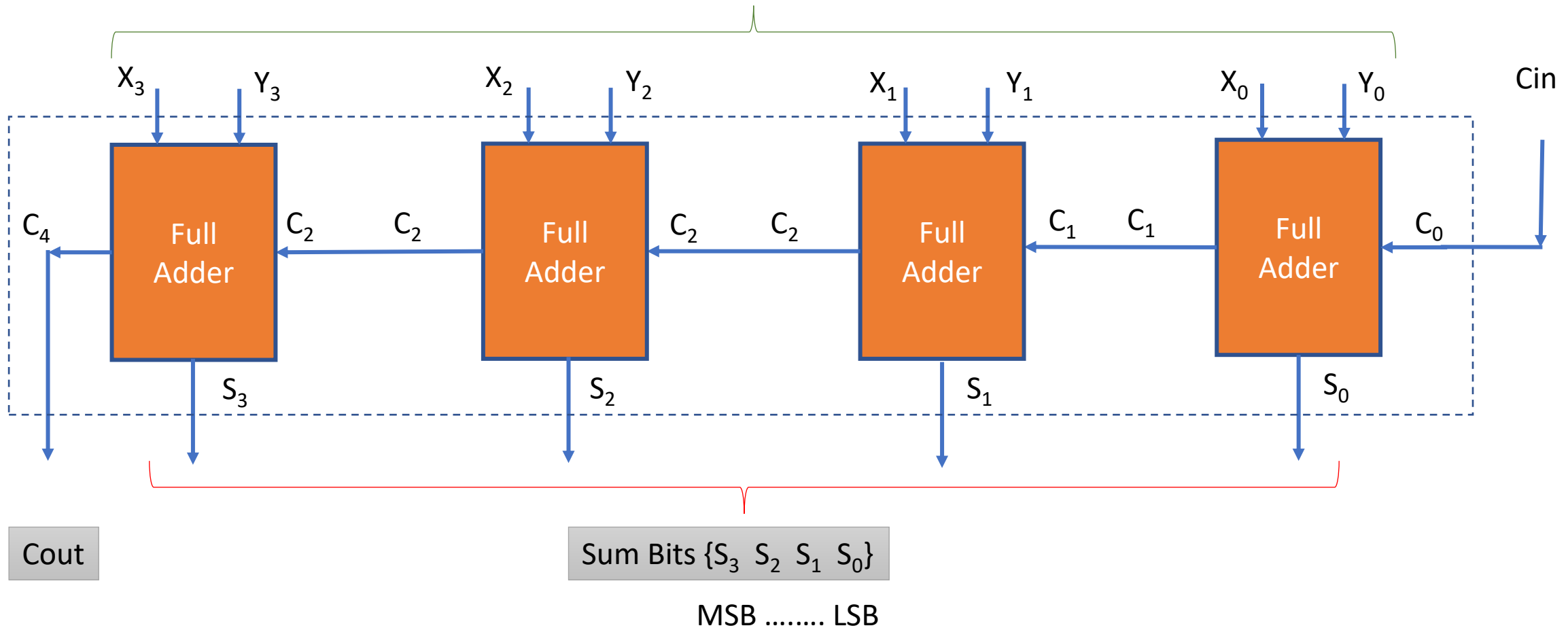$$C_{i+1} = X_i \cdot Y_i + C_i(X_i \oplus Y_i)$$
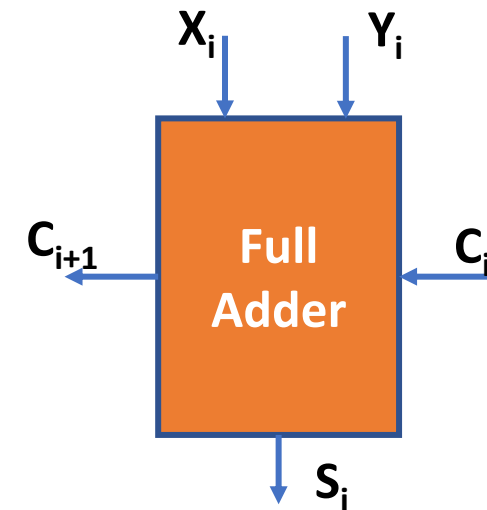
Full Adder Circuit

Sum $S_i = X_i \oplus Y_i \oplus C_i$

$C_{i+1} = X_i . Y_i + C_i (X_i \oplus Y_i)$

X_i Y_i

Full Adder

C_{i+1} C_i

S_i

$C_{i+1} = X_i.Y_i + C_i (X_i \oplus Y_i)$

Sum $S_i = X_i \oplus Y_i \oplus C_i$

LUMS

Multiple Bits – Ripple Carry Adder

Input Numbers $\{X_3\ X_2\ X_1\ X_0\}$ and $\{Y_3\ Y_2\ Y_1\ Y_0\}$, Carry Input at Cin

Cout

Sum Bits $\{S_3\ S_2\ S_1\ S_0\}$

MSB …….. LSB

LUMS

Timing Issues in Ripple Carry Adders

❖ **Delay between Yi and Ci+1**
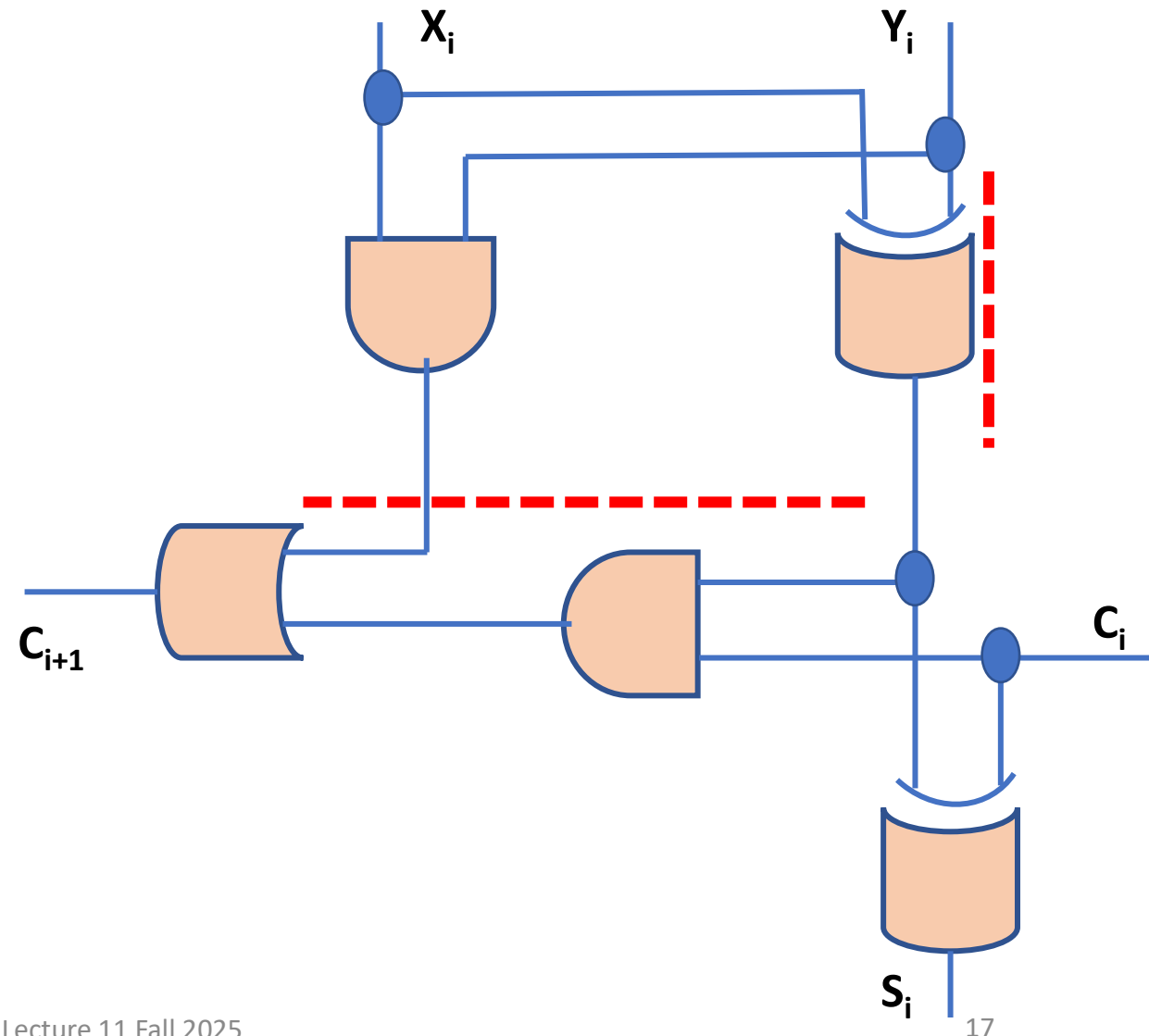**= 1 EXOR + 1 AND + 1 OR (= Critical Path)**

❖ **Suppose delays through gates are:**
EXOR = 20ns, AND = 10ns, OR = 10ns

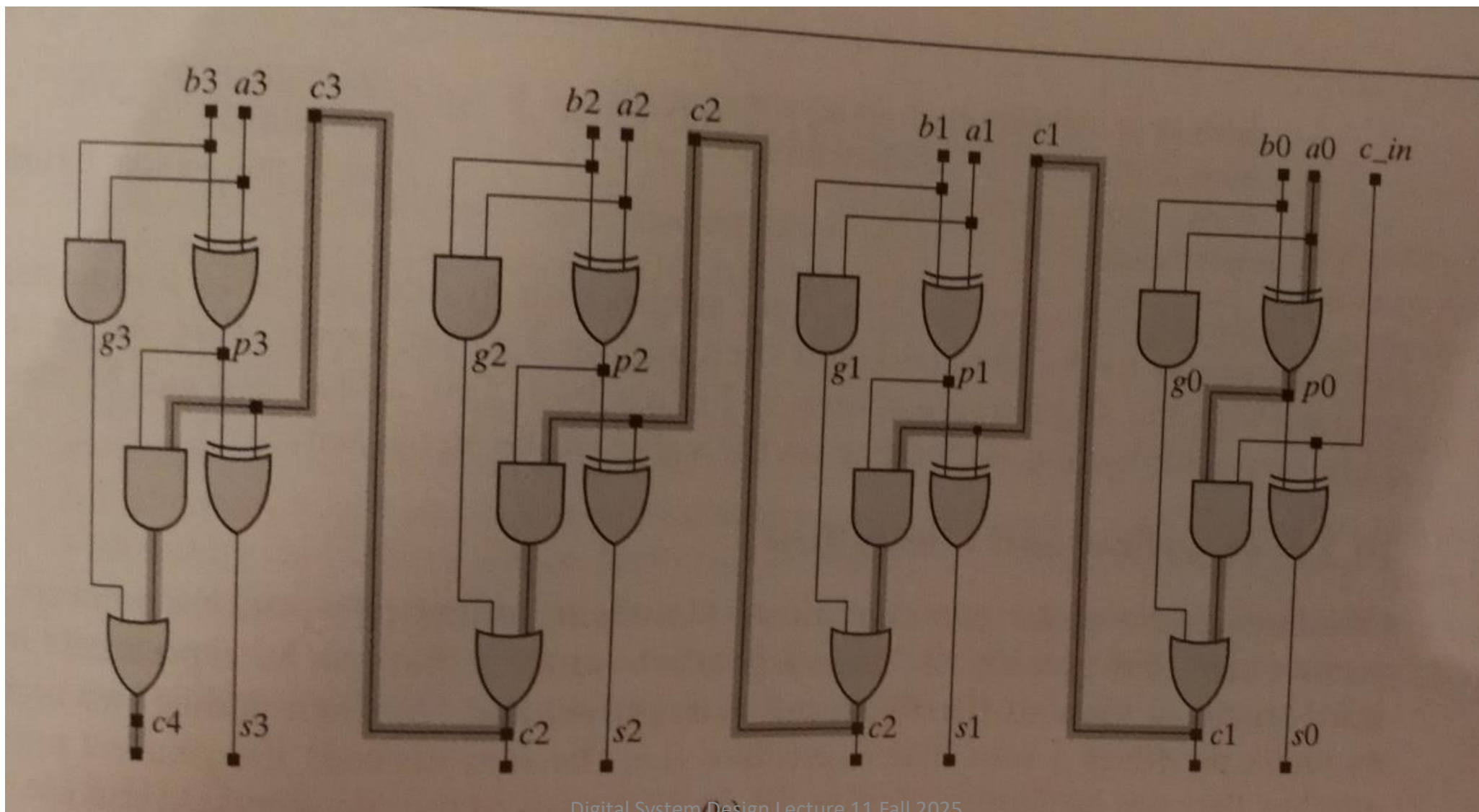❖ **Then Delay between Yi and Ci+1**
= 20ns + 10ns + 10ns = 40ns

❖ **Worst case delay through N adders**
= N × 40ns

❖ **For 32-bit data;**
Delay through 32-bit RCA =
32 x 40ns = 1280ns = 1.28us

❖ **Corresponds to Maximum Data Throughput**
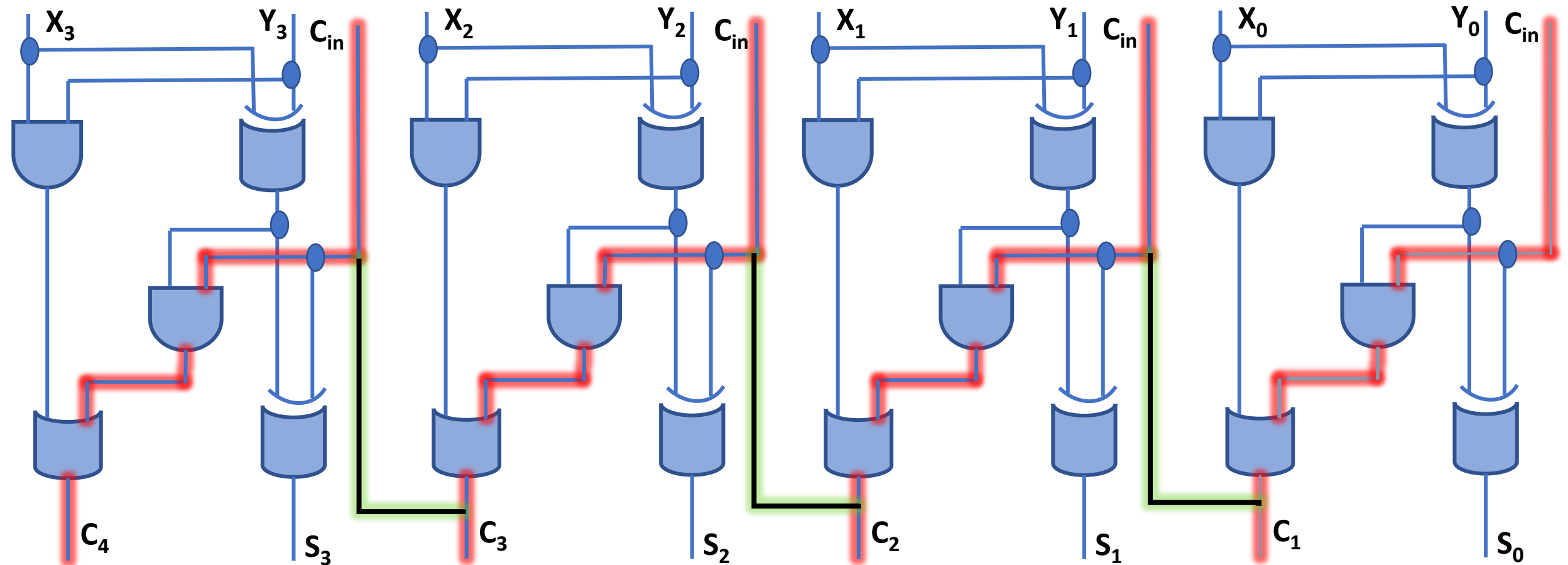**= 1/(1.28μs) = 0.78 MHz (too slow in today's terms**
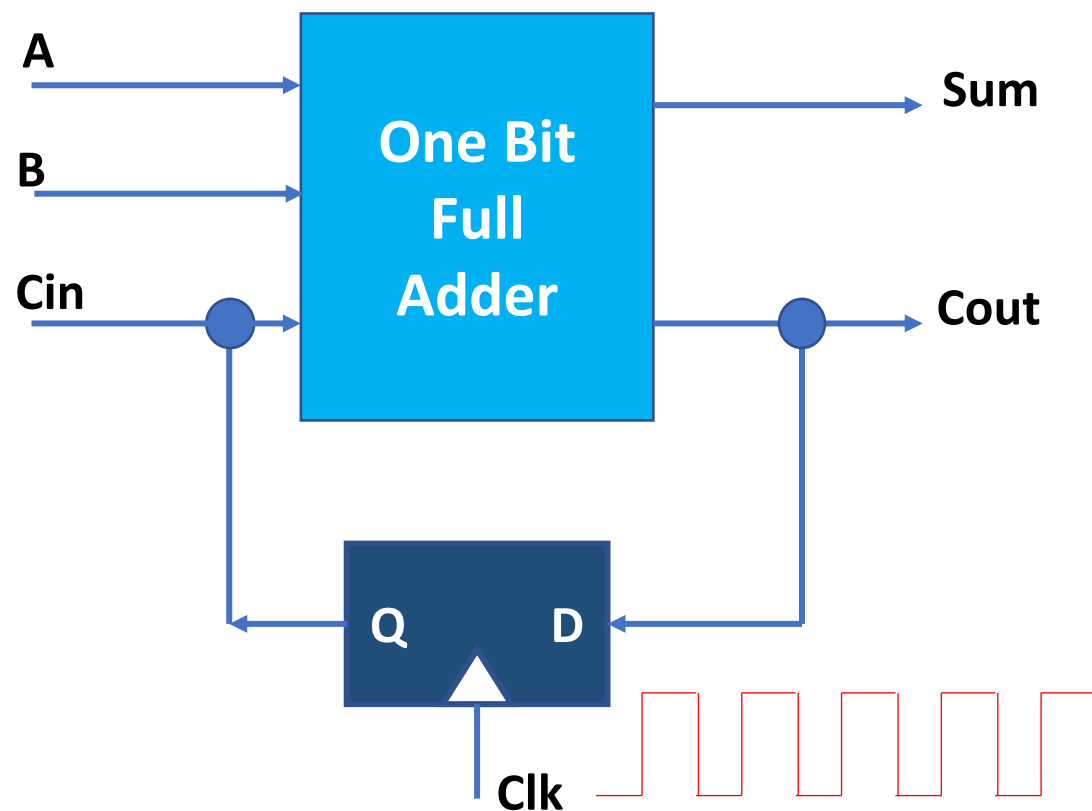
Delay of Ripple Carry Full Adder

Compare with Critical Path of Full Adder

REVIEW

LUMS

Multi-bit Adder – Using 1-Bit Serial Adder

All inputs and outputs are one-bit

What is the **Critical Path**?
And **Maximum Delay**?
And **Maximum Throughput**?

LUMS

8-Bit, Bit-Serial Full Adder
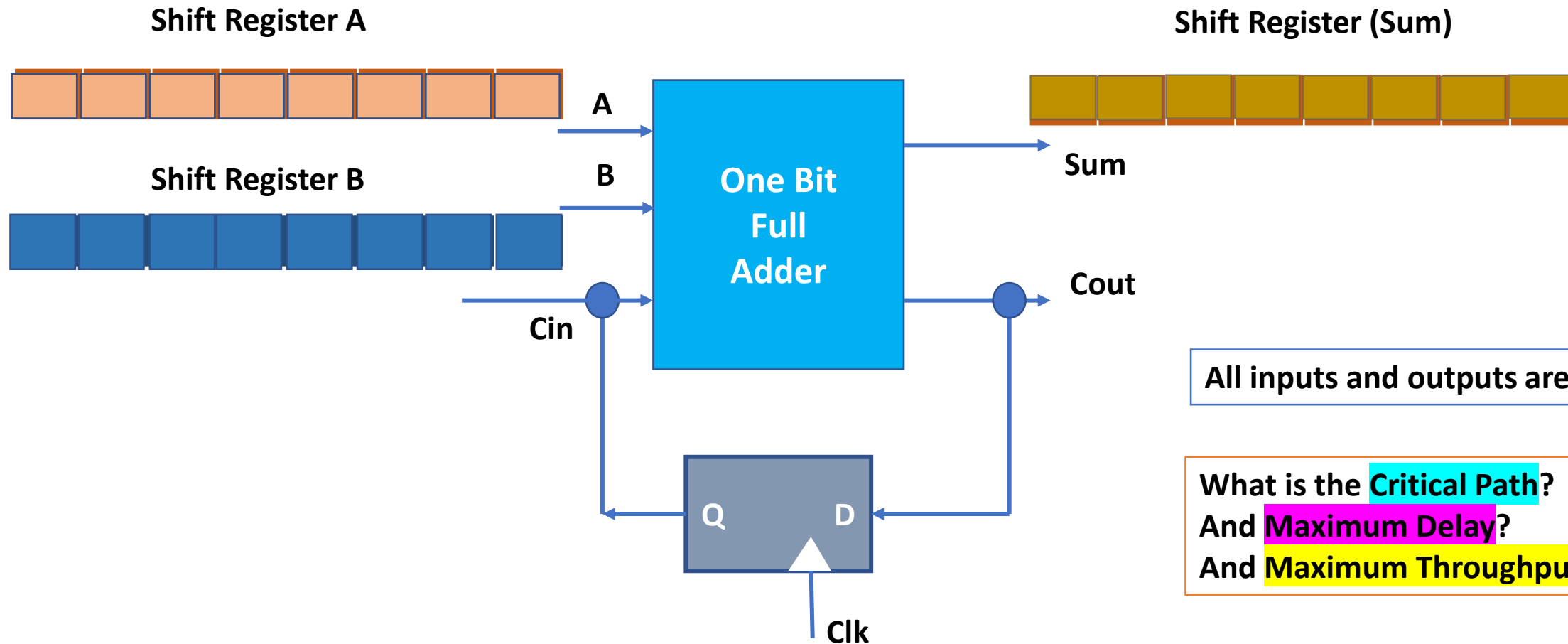
Shift Register A

Shift Register (Sum)

A

B

One Bit Full Adder

Sum

Cout

Cin

Shift Register B

Q   D

Clk

All inputs and outputs are one-bit

What is the Critical Path?
And Maximum Delay?
And Maximum Throughput?

LUMS

Timing Considerations in Bit-Serial Design

- **It is an extreme example of a pipelined design**

- **What is Critical Path?**

- **What is Latency?**

- **What is Data Throughput?**

- **What is maximum clock speed achievable?**

Definitions related to timing performance

- **Latency:** Amount of time before the appearance of first correct output

- **Throughput:** Amount of processed data available at the output port, per unit time

- **Critical Path:** The combinational logic path with maximum number of circuit elements that the input travels before reaching the output

- **Maximum Achievable Clock Speed:** Limited by Critical Path

Carry Lookahead Adder

Carry Lookahead Adder – A type of fast adder

- **In Ripple Carry Adder, the Carry Input Cin propagates through all Adder circuits before reaching the final Carry Out, Cout. Thus, there is <mark>a Long carry chain that makes the Critical Path worse</mark>.**

- **In Carry Lookahead Adder (CLA), the <mark>Carry Circuit is separated</mark> from the Sum circuit and both work independently. This reduces the number of gates in Critical Path and the Adder can work faster.**

LUMS

Basic Functional Block – Full Adder

Truth Table of 1-Bit Full Adder

A_i	B_i	C_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

Boolean Expressions for Sum and Carry Out

$$\text{Sum } S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i \cdot B_i + C_i (A_i \oplus B_i)$$

LUMS

Boolean Algebra formulation of Sum and Cout in Carry Lookahead Adder

Define bits as follows:
A_i and B_i are data bits at i^{th} cell of the adder
C_i is the carry in into the ith cell
S_i is Sum output bit of the ith cell
C_{i+1} is the Carry Out of the ith cell

Define two signals generate G_i and propagate P_i :
$G_i = A_i . B_i$
$P_i = A_i \oplus B_i$
$S_i = (A_i \oplus B_i) \oplus C_i$
Also
$S_i = P_i \oplus C_i$
Carry Out = C_{i+1}
$C_{i+1} = ((A_i \oplus B_i) \oplus C_i) + (A_i . B_i)$
Also
$C_{i+1} = (P_i . C_i) + G_i$

Sum $S_i = A_i \oplus B_i \oplus C_i$

$C_{i+1} = A_i . B_i + C_i (A_i \oplus B_i)$

Algorithmically:
$S_i = P_i \oplus C_i$
And
$C_{i+1} = (P_i . C_i) + G_i$

Carry Look Ahead Boolean Equations – S0, S1

$S_0 = P_0 \oplus C_0$

$C_1 = (P_0 . C_0) + G_0$

$S_1 = (P_1 \oplus C_1)$

$\quad = P_1 \oplus ((P_0 . C_0) + G_0)$

$C_2 = (P_1 . C_1) + G_1$

$C_2 = (P_1 . P_0 . C_0) + (P_1 . G_0) + G_1$

$\quad$ where $C_1 = (P_0 . C_0) + G_0$

Carry Lookahead Boolean – S2, C3

$S_2 = P_2 \oplus C_2$

$S_2 = P_2 \oplus [P_1 . P_0 . C_0 + P_1 . G_0 + G_1]$

$C_3 = P_2 . C_2 + G_2$

$\quad = P_2 . [P_1 . P_0 . C_0 + P_1 . G_0 + G_1] + G_2$

$C_3 = P_2 . P_1 . P_0 . C_0 + P_2 . P_1 . G_0 + P_2 . G_1 + G_2$

Carry Lookahead Boolean – S3 and C4

$S_3 = P_3 \oplus C_3$

$S_3 = P_3 \oplus [P_2 . P_1 . P_0 . C_0 + P_2 . P_1 . G_0 + P_2 . G_1 + G_2]$

$C_4 = (P_3 . C_3) + G_3$

$C_4 = P_3 . [P_2 . P_1 . P_0 . C_0 + P_2 . P_1 . G_0 + P_2 . G_1 + G_2] + G_3$

Recap:

Define two signals generate G_i and propagate P_i :

$G_i = A_i . B_i$

$P_i = A_i \oplus B_i$

$S_i = (A_i \oplus B_i) \oplus C_i$

Also

$S_i = P_i \oplus C_i$

Carry Out = C_{i+1}

$C_{i+1} = ((A_i \oplus B_i) . C_i) + (A_i . B_i)$

Also

$C_{i+1} = (P_i . C_i) + G_i$

Since $C_{i+1} = G_i + P_i.C_i$, therefore:

$C_1 = G_0 + P_0.C_0$

$C_2 = G_1 + P_1.C_1$

$C_3 = G_2 + P_2.C_2$

$C_4 = G_3 + P_3.C_3$

The Carry Out at any stage can be generated from Pi, Gi and C0;
Without waiting for carry to ripple through all the previous stages

LUMS

Block Diagram of 4-Bit CLA

A_3 B_3

Evaluate G, P
as per
Boolean
Expressions

A_2 B_2

Evaluate G, P
as per
Boolean
Expressions

A_1 B_1

Evaluate G, P
as per
Boolean
Expressions

A_0 B_0

Evaluate G, P
as per
Boolean
Expressions

C_0

G_3 P_3 C_3 G_2 P_2 C_2 G_1 P_1 C_1 G_0 P_0

Carry Lookahead Logic as per Boolean Expression

C4 C_0

Cin

S_3 S_2 S_1 S_0

LUMS

Delay of 4-Bit CLA Adder

E 10-8 Gate-level circuits of 4-bit slice
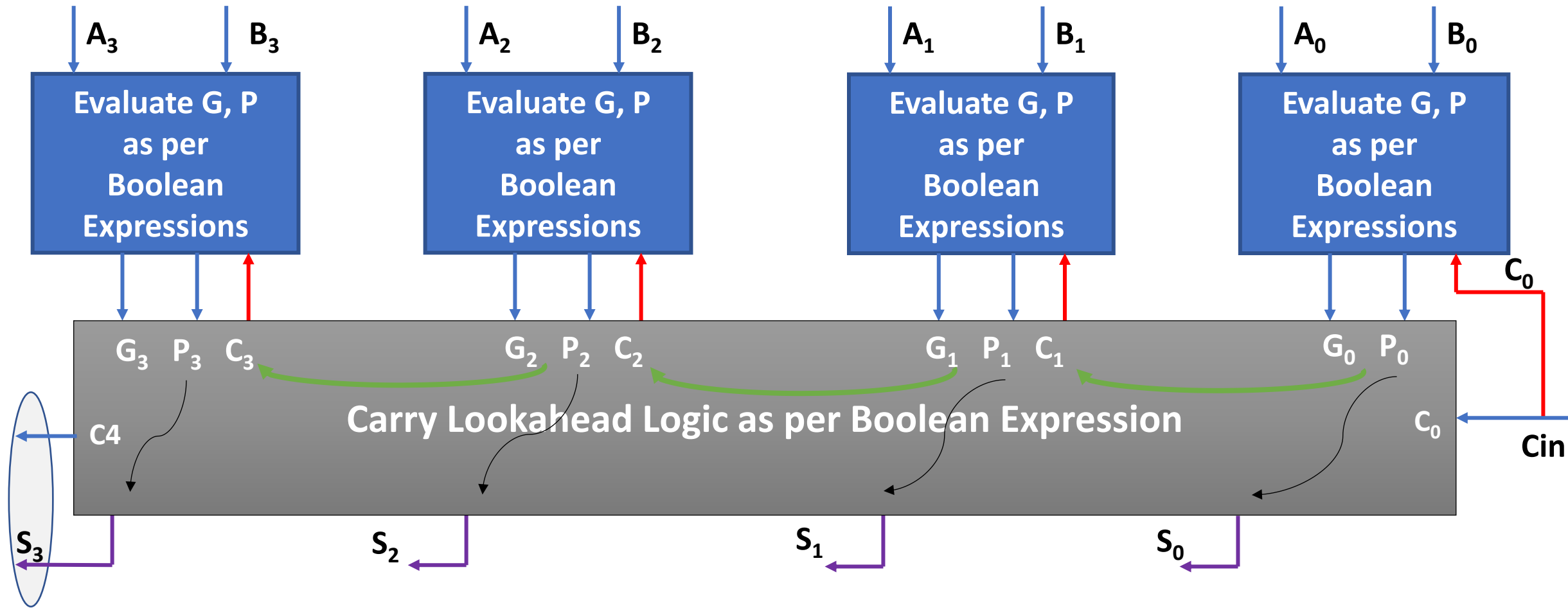
4-Bit CLA Adder
Showing Critical Path

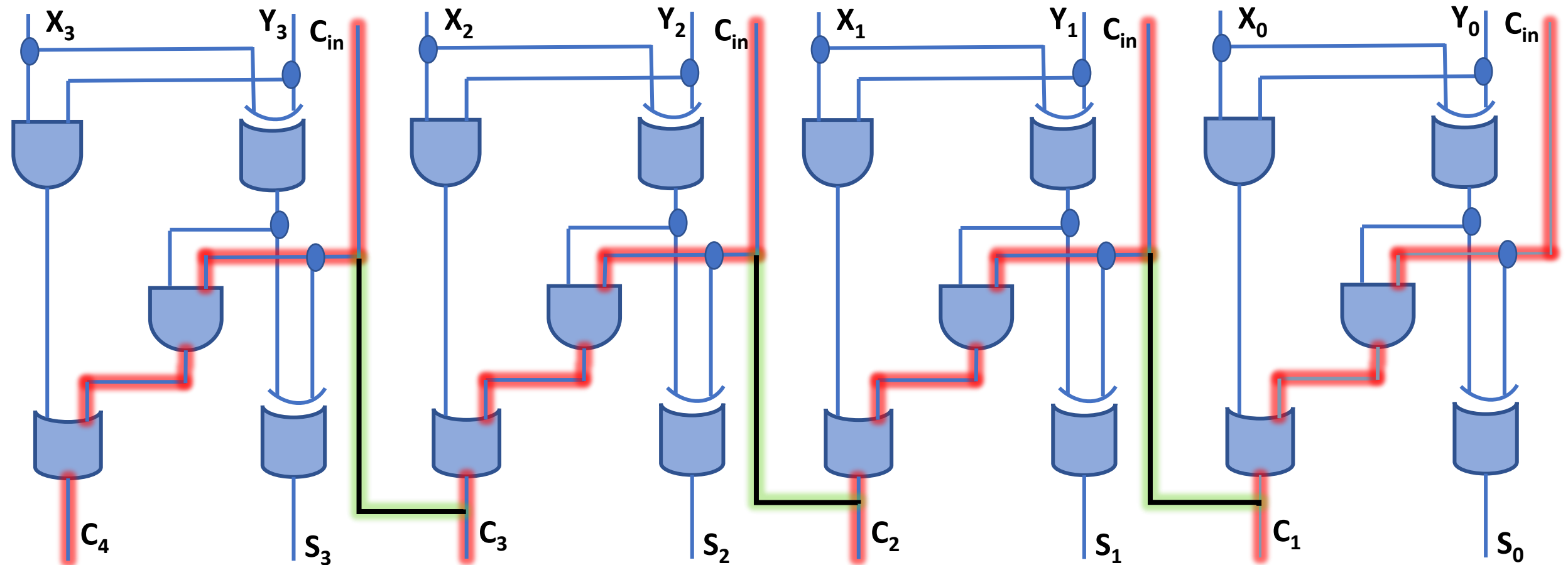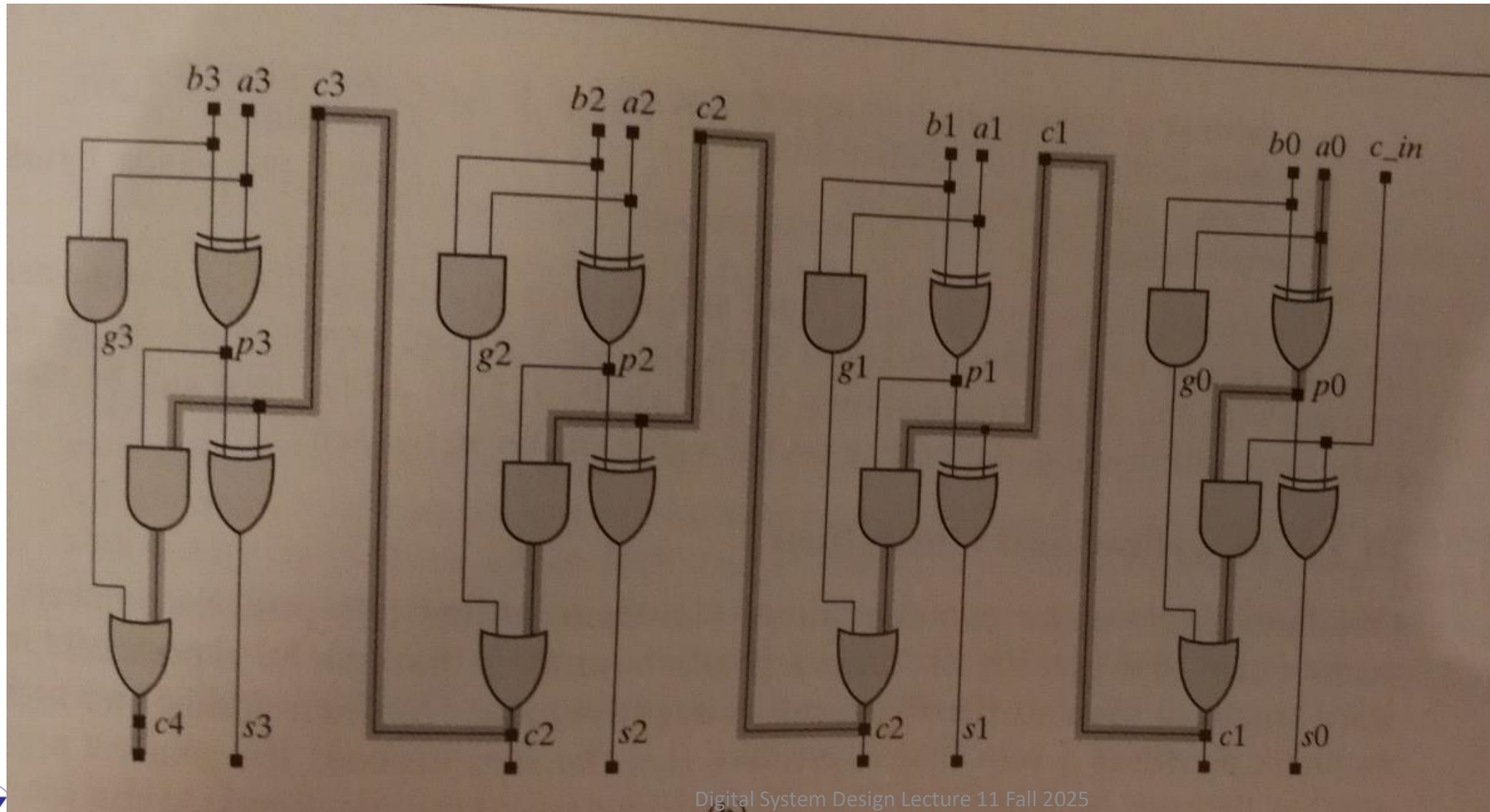Estimate Maximum Clock Speed?
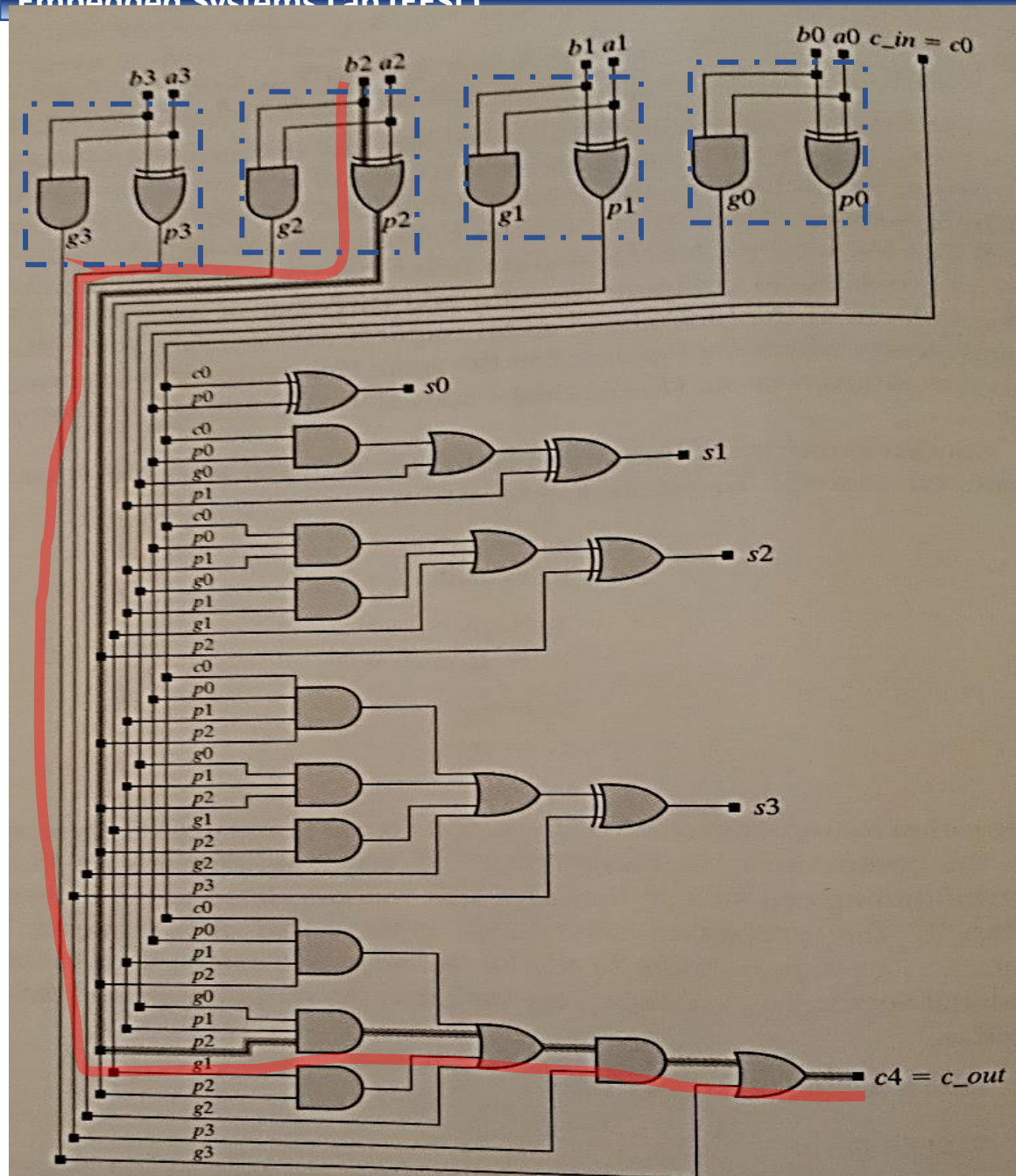
Diagram from Ciletti course text book

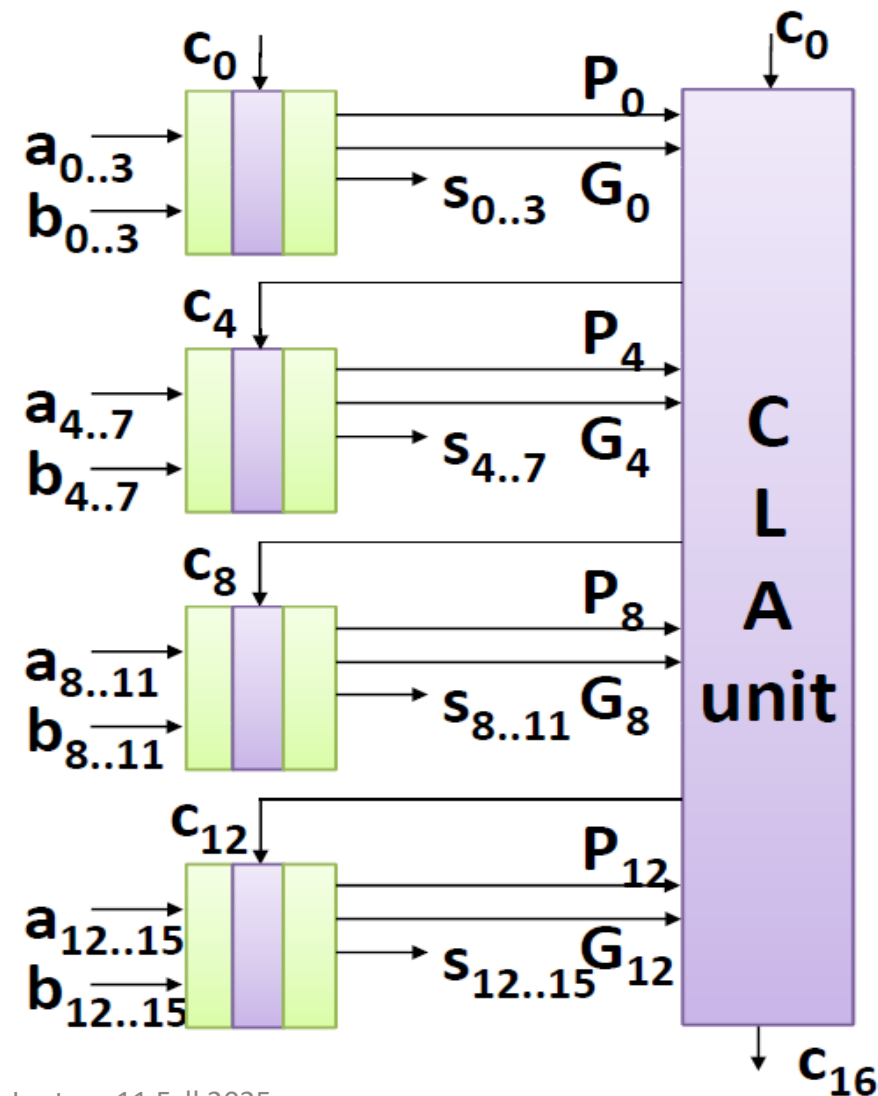# Compare with Critical Path of Full Adder

REVIEW

LUMS

Compare with Critical Path of Full Adder

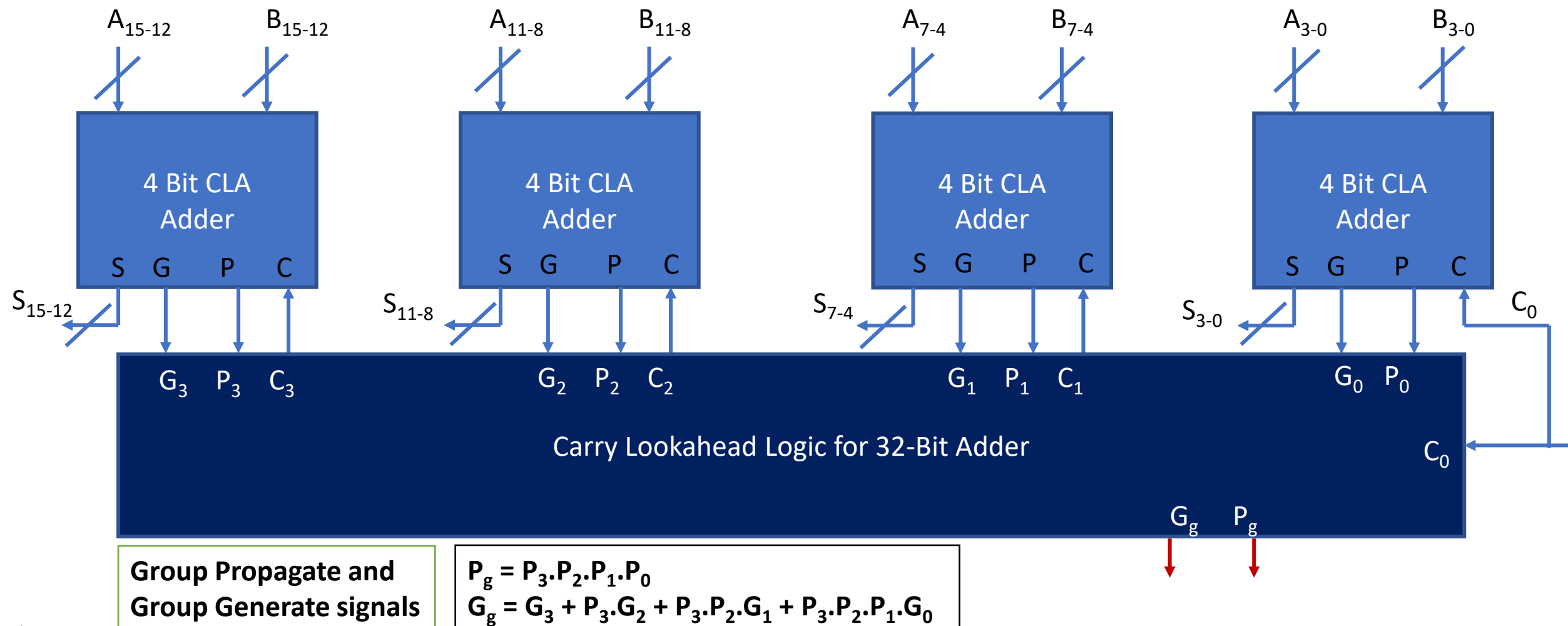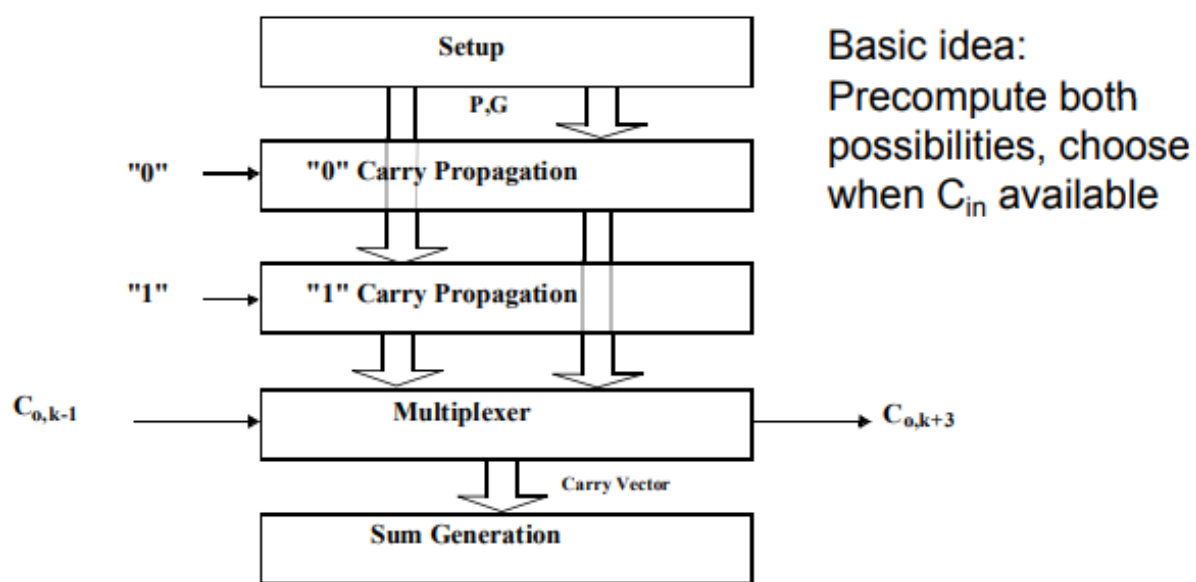# 2 Level CLA Generation

2 Levels of look ahead

no rripling of carry

16-Bit CLA – Gates with many inputs needed?

A_{15-12} B_{15-12} A_{11-8} B_{11-8} A_{7-4} B_{7-4} A_{3-0} B_{3-0}

4 Bit CLA Adder

S G P C

S_{15-12} S_{11-8} S_{7-4} S_{3-0} C_0

G_3 P_3 C_3 G_2 P_2 C_2 G_1 P_1 C_1 G_0 P_0

Carry Lookahead Logic for 32-Bit Adder

C_0

G_g P_g

Group Propagate and Group Generate signals

$$P_g = P_3.P_2.P_1.P_0$$
$$G_g = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0$$

Carry Select Adder

Carry Select Adder - Principle

Carry-Select Adder

Basic idea:
Precompute both possibilities, choose when C_{in} available
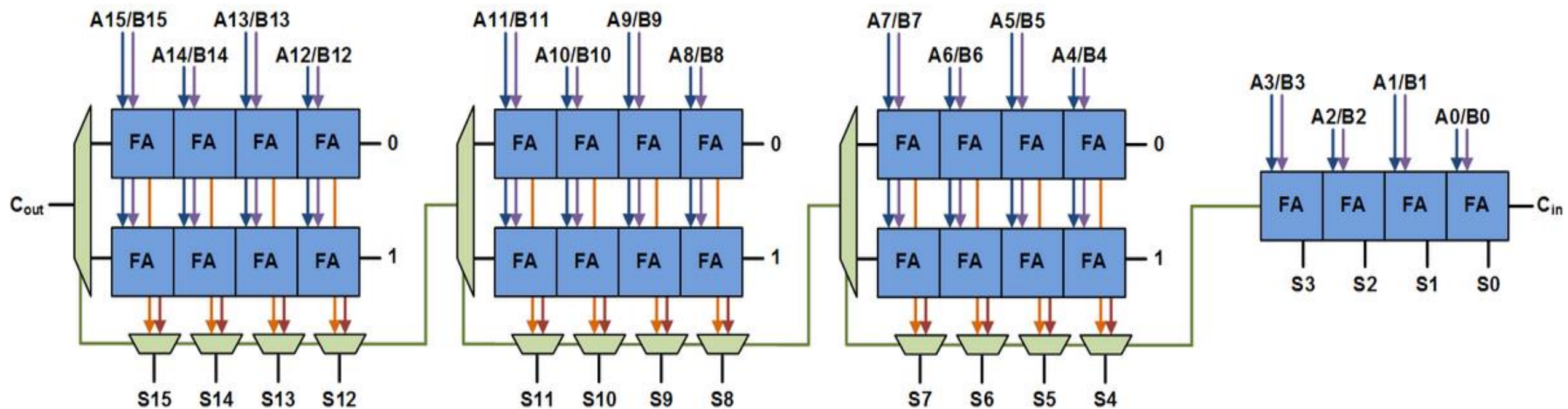
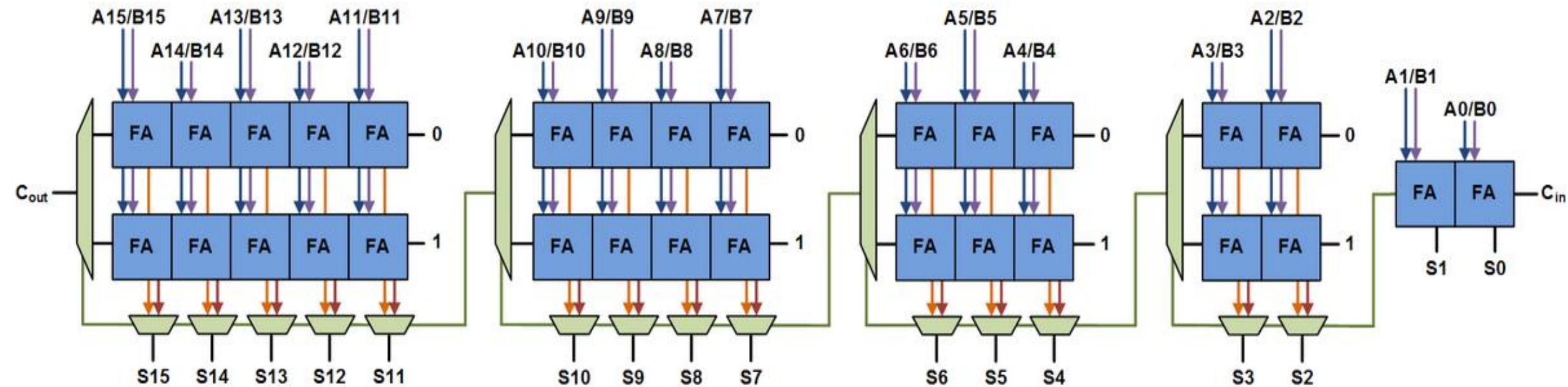EECS 427 F08   Lecture 7   24
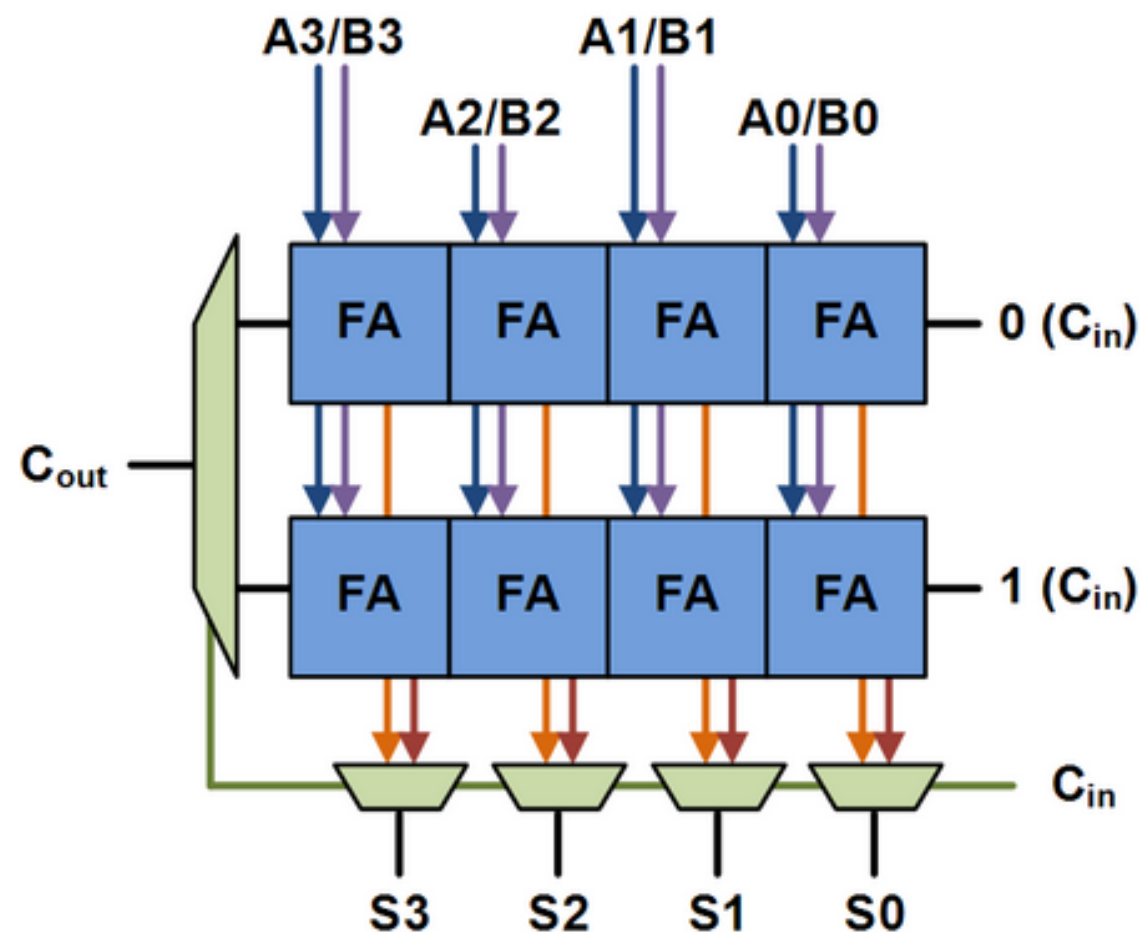
Ref: https://en.wikipedia.org/wiki/Carry-select_adder

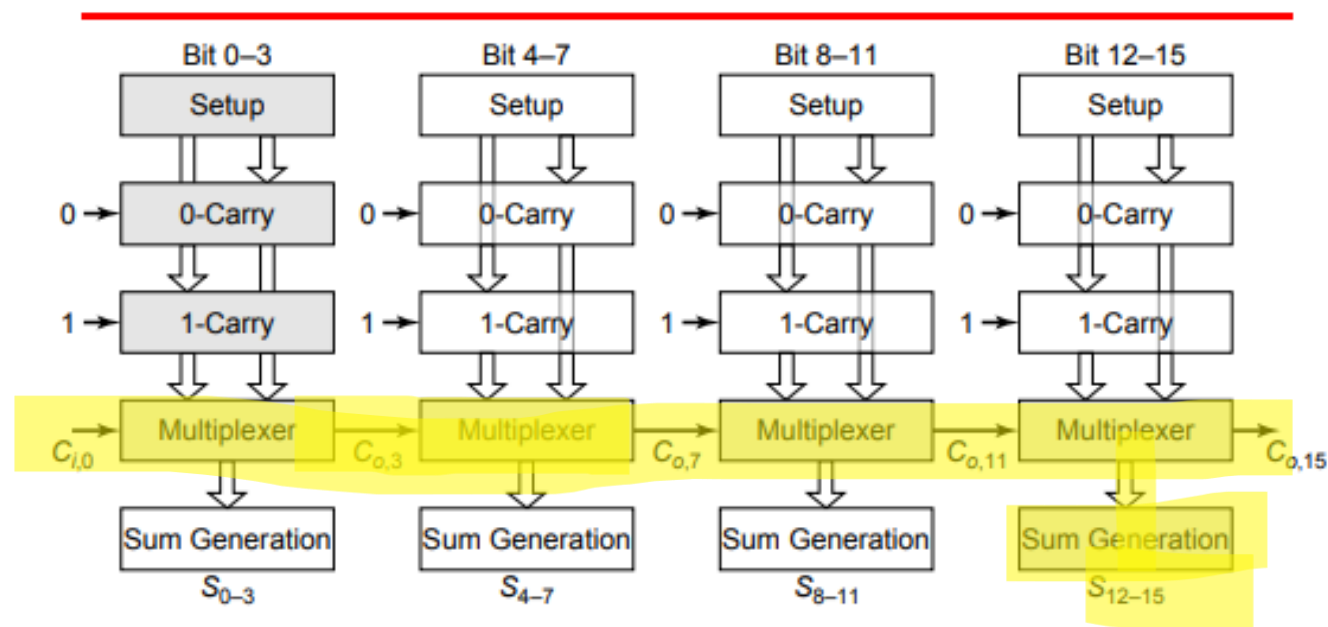# Carry Select Adder – Uniform Size Adders

Carry Select Adder – Variable Size Adders

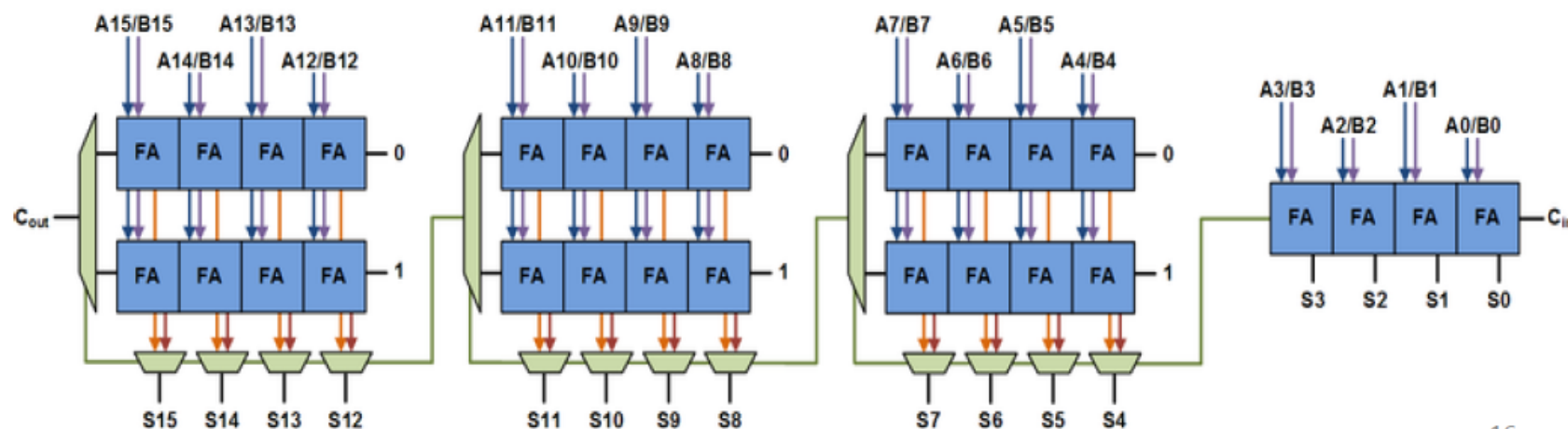# Carry Select Adder Critical Path

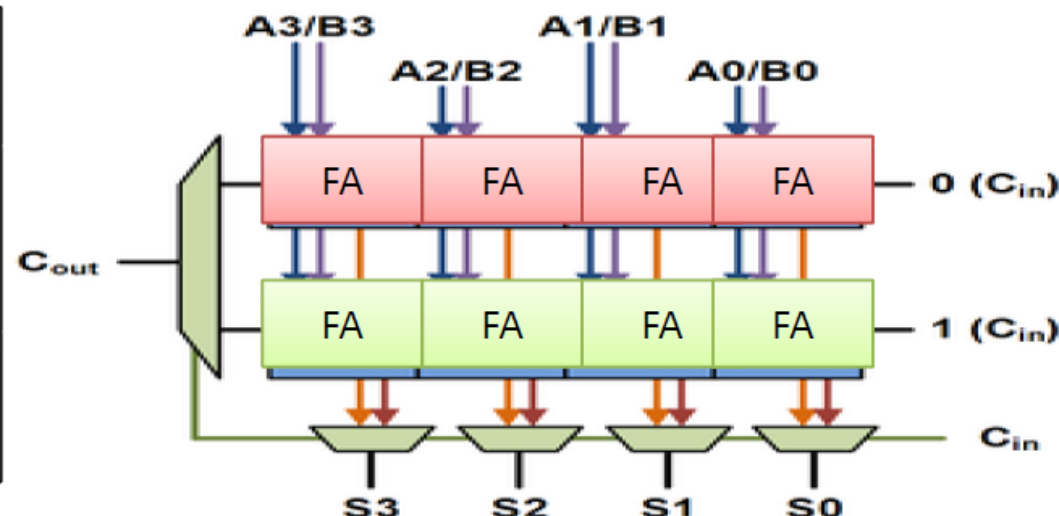

Carry Select Adder: Critical Path

$$t_{add} = t_{setup} + Mt_{carry} + (N/M)t_{mux} + t_{sum}$$

Carry Select Adder

Carry Select Adder

For a group Sum & Carry is already calculated

Simply select based on carry

16