Lecture 12
EE 421 / CS 425
Digital System Design

Fall 2024

Shahid Masud

Topics

- Quick Recap – Full Adder and Multi-bit Adder Designs
- --
- Overflow Detection
- 2's Complement Arithmetic for Signed Numbers
- Binary Multipliers – timing issues
- Parallel Array Multiplier

QUIZ 3
Next Monday
14 October 2024

Carry Lookahead Adder – A type of fast adder

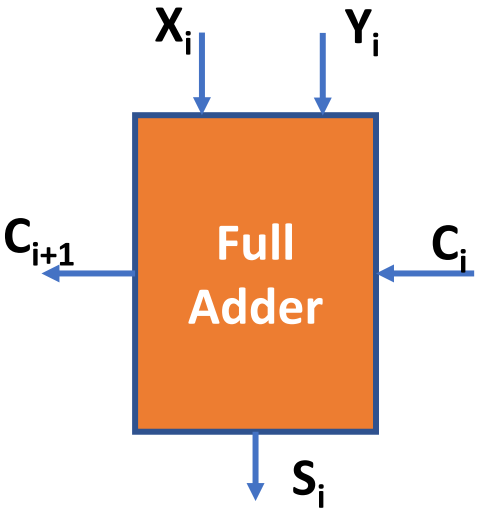
- In Ripple Carry Adder, the Carry Input Cin propagates through all Adder circuits before reaching the final Carry Out, Cout. Thus there is Long carry chain that makes the Critical Path worse.
- In Carry Lookahead Adder (CLA), the carry Circuit is separated from the Sum circuit and both work independently. This reduces the number of gates in Critical Path and the Adder can work faster.

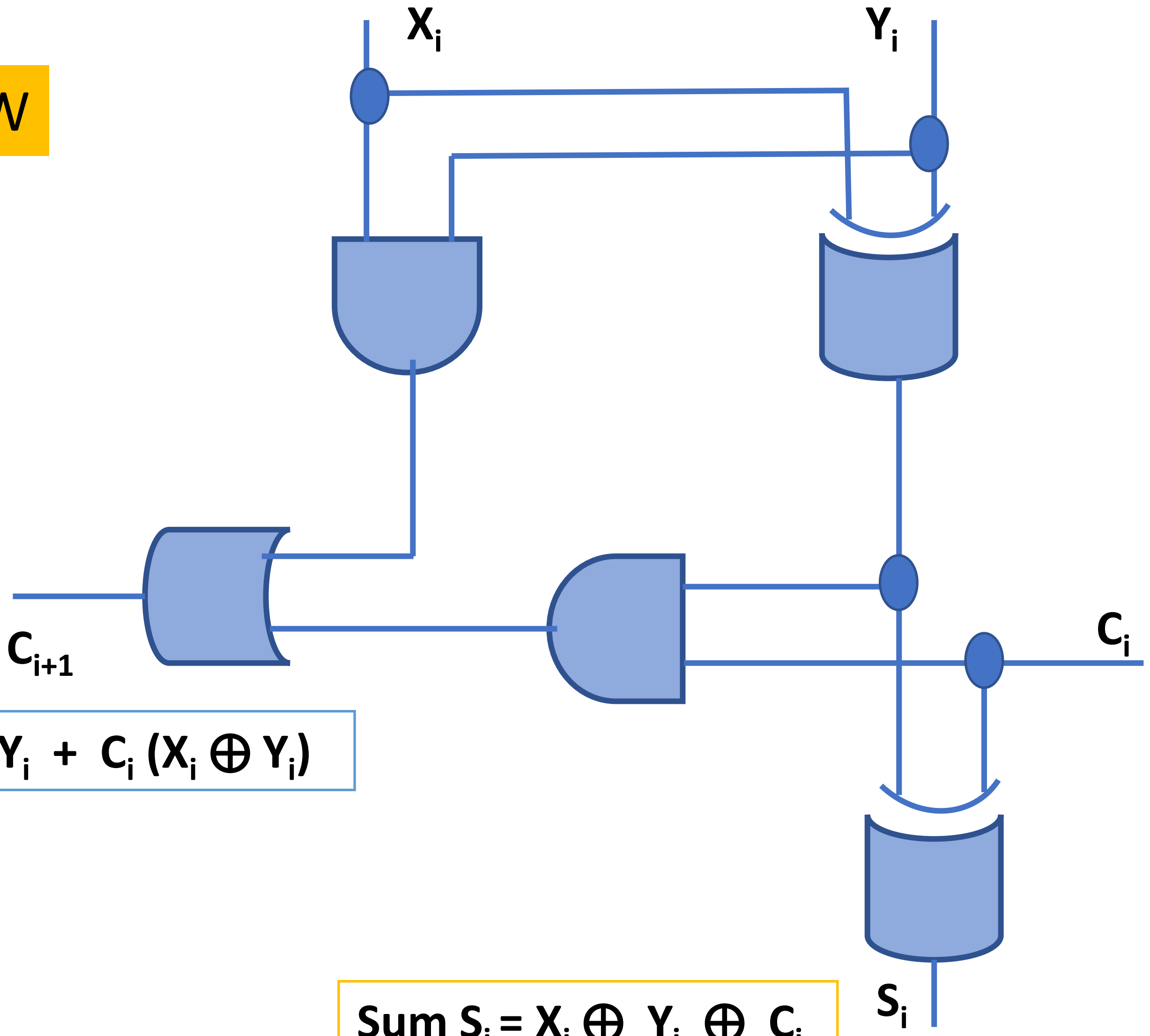
Full Adder Circuit

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

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

REVIEW



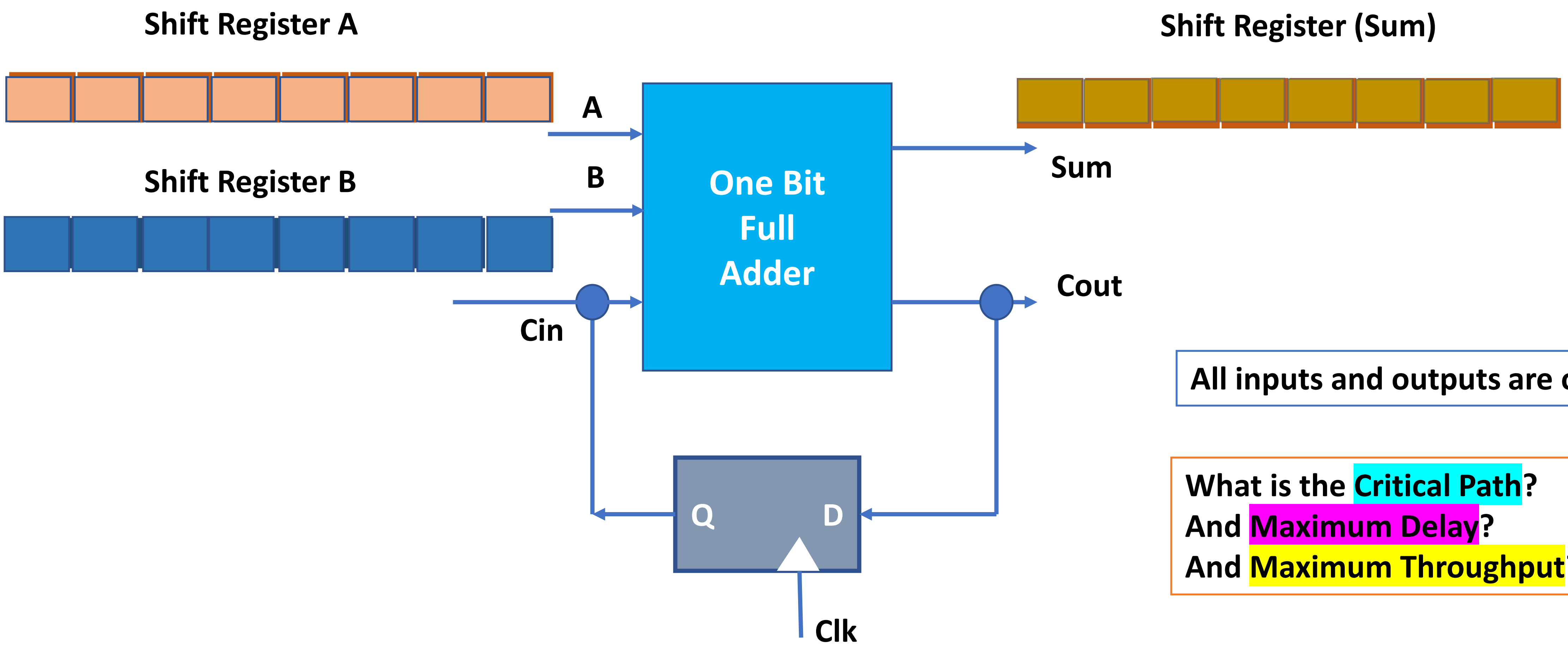


$C_{i+1} = X_i.Y_i + C_i (X_i \oplus Y_i)$

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

8-Bit, Bit-Serial Full Adder

REVIEW

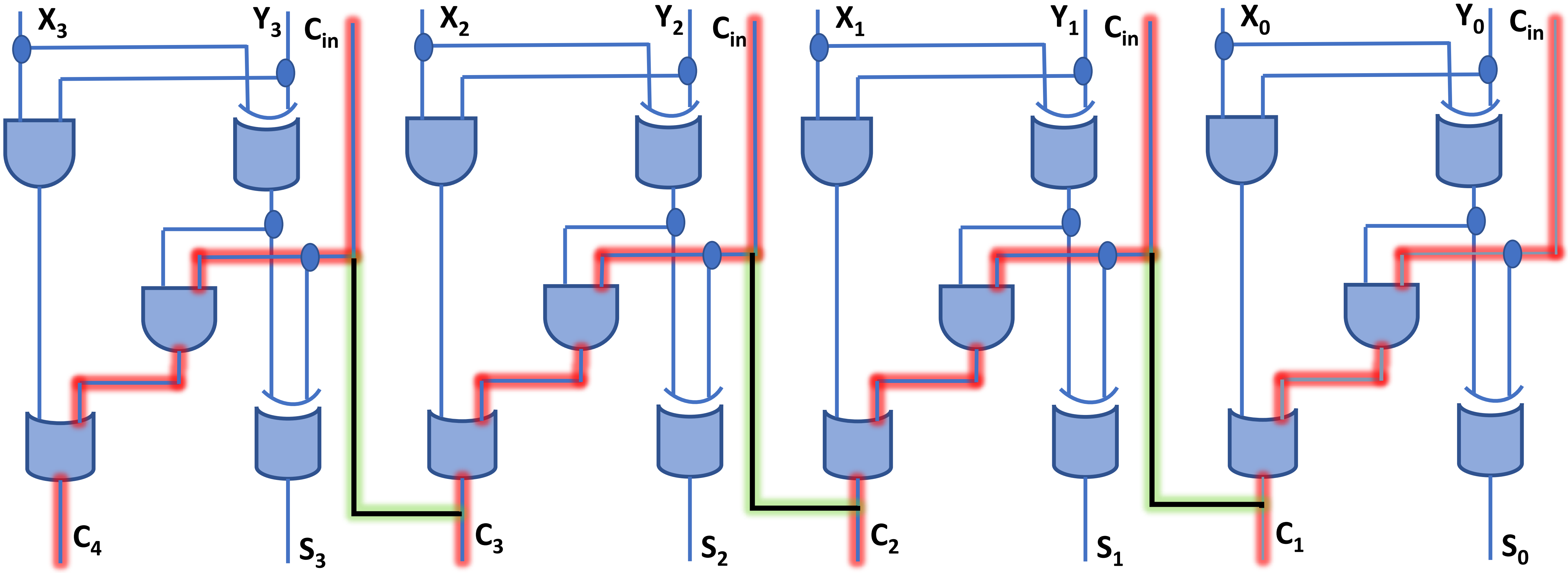


All inputs and outputs are one-bit

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

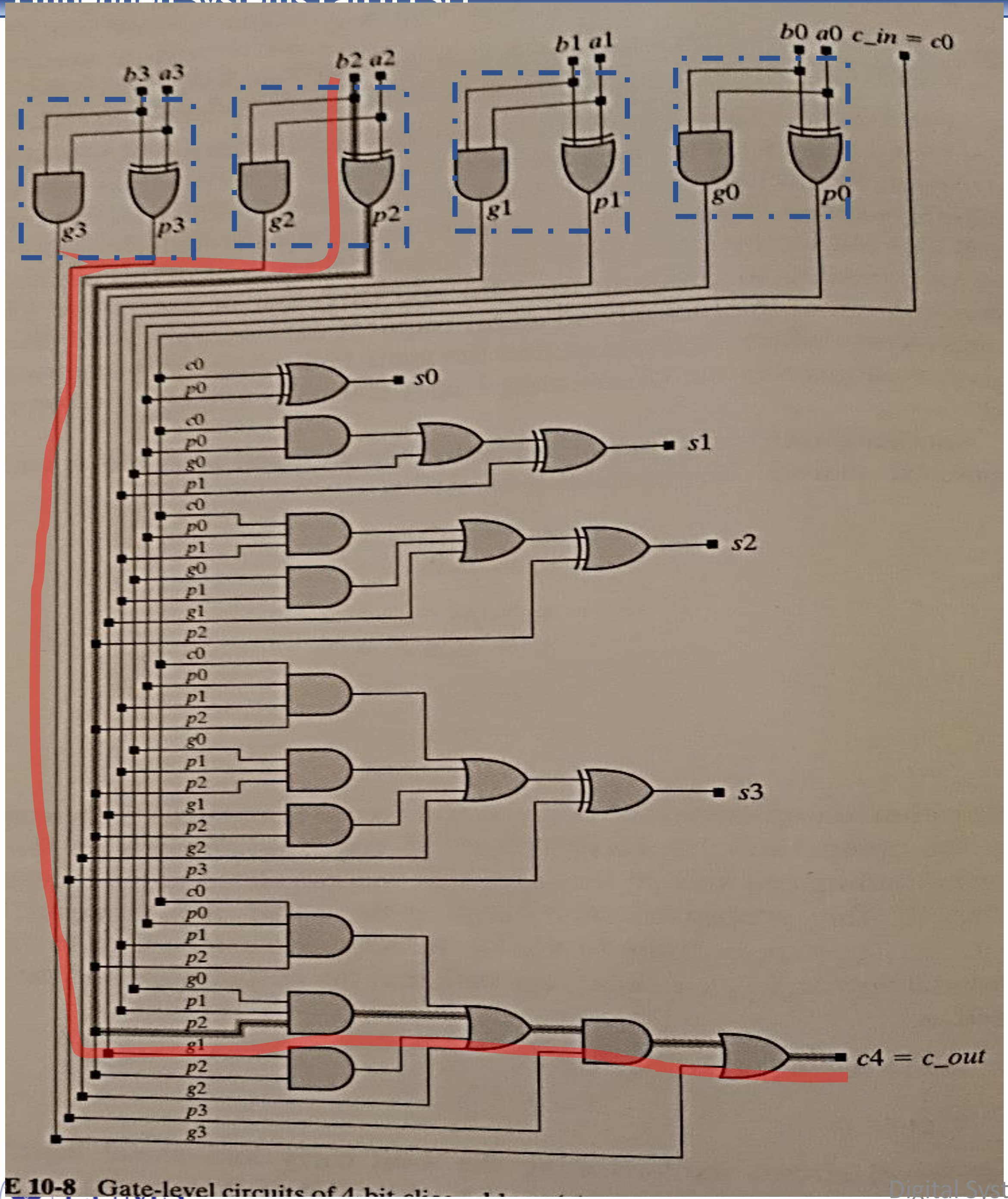
Compare with Critical Path of Full Adder

REVIEW



Delay of 4-Bit CLA Adder

REVIEW



4-Bit CLA Adder
Showing Critical Path

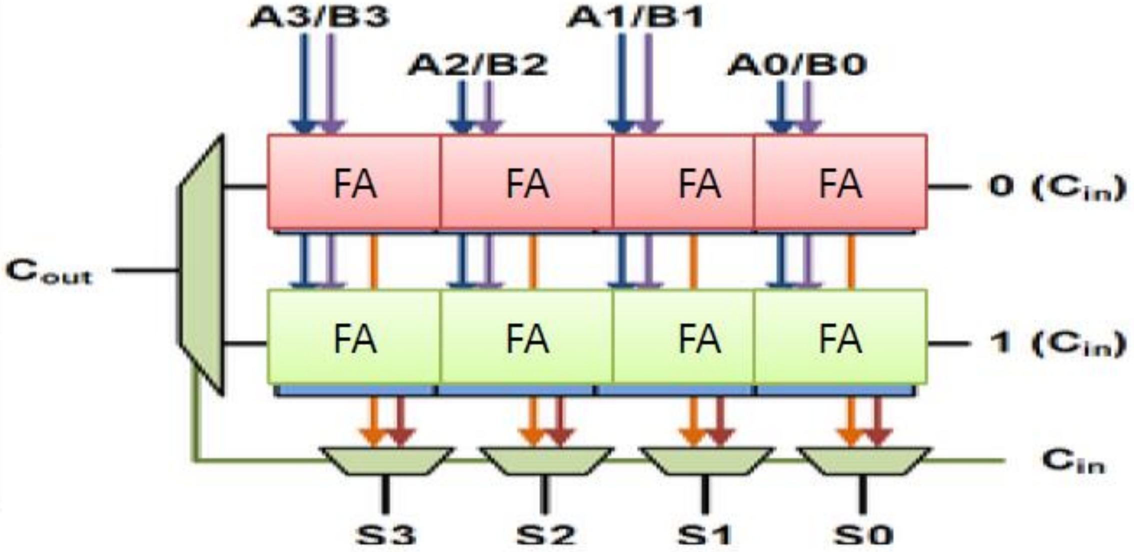
Estimate Maximum Clock Speed?

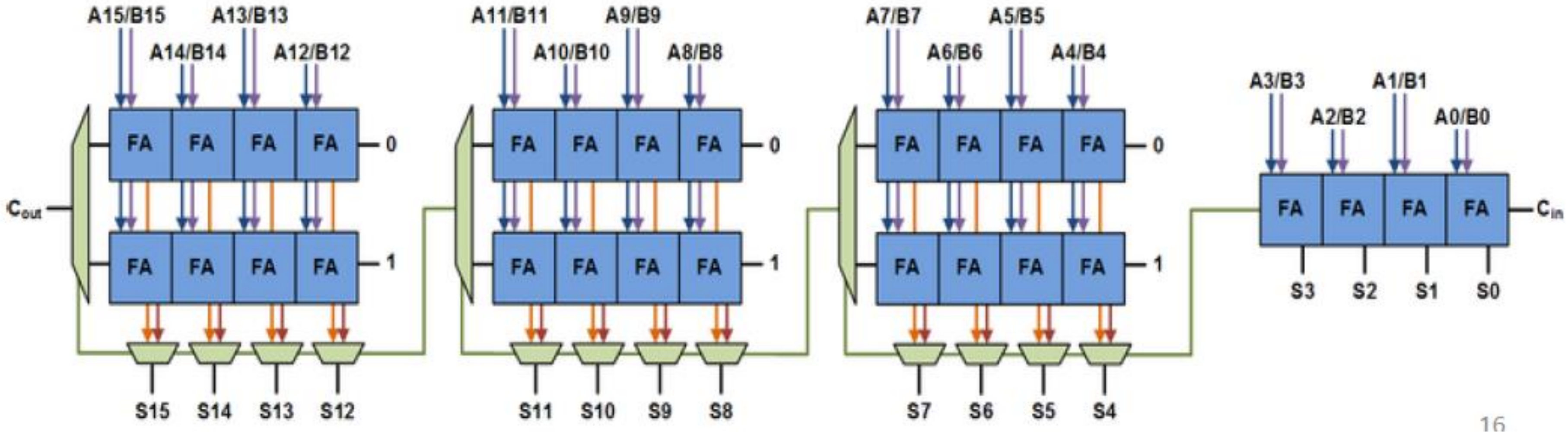
Diagram from Ciletti course text book

Carry Select Adder

For a group Sum & Carry is already calculated

Simply select based on carry





Signed Numbers

Representing Signed Numbers

**When we work in a system where both positive and negative numbers can be processed,
We need to do something about the 'sign' of the numbers**

Given two signed numbers A and B

A6	A5	A4	A3	A2	A1	A0
B6	B5	B4	B3	B2	B1	B0

**MSB is the 'sign' bit in Signed binary representation of numbers, remaining bits represent the magnitude
'0' sign bit means positive number and '1' sign bit means negative number**

**In arithmetic processing, the magnitudes are processed (eg, +/-/x operations) separately and
Correct sign is determined and inserted at the end of the processing**

2’s Complement Representation

- Sign-bit becomes a part of the number after taking 2’s Complement
- The arithmetic circuit is based on one sign (eg. positive numbers) only
- The sign-bit of the answer guides if the number is positive or negative
- 2’S Complement of the answer is taken at the end to get a positive number

2’s Complement of a binary number is processed as follows:

- ❖ Add One or More Extra bits to take care of sign-extension
- ❖ All ‘0’ in extended bits indicate positive number
- ❖ All ‘1’ in extended bits indicate negative number
- ❖ If extended bits are ‘1’; first take 1’s complement, i.e. invert all bits
- ❖ Then add “+1” to the number
- ❖ The answer is 2’s Complement of the number

Example of 2's Complement

Eg. Take 2's Complement of decimal number -45

Binary representation of "+45" is "101101"
Take minimum one extra bit for sign, negative means '1'
So minimum signed binary width of "-45" is "1101101"

Signed Binary number	**1**	**1**	**0**	**1**	**1**	**0**	**1**
Take 1's Complement (invert all bits)	**0**	**0**	**1**	**0**	**0**	**1**	**0**
Add "1" at LSB						**+**	**1**
	0	**0**	**1**	**0**	**0**	**1**	**1**

2's Complement of "-45" is a 7-bit binary number "0010011"

2's Complement of +45

Eg. Take 2's Complement of decimal number +45

Binary representation of 45 is "101101"
Take minimum extra bit for sign, positive means '0'
So minimum signed binary width of "+45" is "0101101"

Signed Binary number	**0**	**1**	**0**	**1**	**1**	**0**	**1**
Take 1's Complement (invert all bits)	**1**	**0**	**1**	**0**	**0**	**1**	**0**
Add "1" at LSB						**+**	**1**
Result	**1**	**0**	**1**	**0**	**0**	**1**	**1**

2's Complement of "+45" is a 7-bit binary number "1010011"
The '1' at MSB indicates it is now a negative number, after taking 2's Complement of a positive number

Another quick method for 2's Complement

Decimal number = +46

Extra sign bit

Given Binary number	**0**	**1**	**0**	**1**	**1**	**1**	**0**
Examine from Right to Left (LSB to MSB)	**1**	**0**	**1**	**0**	**0**	**1**	**0**
When you encounter first '1' , invert all bits on the left	**0**	**1**	**0**	**1**	**1** **Invert this bit and all to left**	**1** **First '1' seen**	**0** **LSB is '0', go left**
2's Complement Answer	**1**	**0**	**1**	**0**	**0**	**1**	**0**

Extra bits for sign have to be specified before conversion
All extra bits occupy same value, '0' for positive and '1' for negative, bit replication

2's Complement Addition Examples

Case 1: Both numbers positive
4-Bit numbers with one bit for sign

Add "+9" and "+4"

+9	**0**	**1**	**0**	**0**	**1**
+4	0	0	1	0	0
	0	1	1	0	1

Sign bit

Answer = "1101" = "+13"

Case 2: Positive numbers and smaller negative number
4-Bit numbers with one bit for sign

Add "+9", "-4"; using 5 bits total

2's Complement of 4 (00100) is (11100)

+9	**0 , 1 Cin**	**1**	**0**	**0**	**1**
-4	1	1	1	0	0
1 Cout	0	0	1	0	1

Extra bit, Sign bit
Carry out

Answer = "00101" = "+5"

Cout is beyond 5 bits, hence discarded

2’s Complement more addition cases

Case 3: Positive number and a larger negative number
4-Bit numbers with one bit for sign

Add “+4”, “-9”; using 5 bits total
2’s Complement of 9 (01001) is (10111)

+4	0	0, Cin 1	1	0	0
-9	1	0	1	1	1
	1	1	0	1	1

Sign bit

Answer = “11011”
Take 2’s Complement = “00101” = “-5”

Case 4: Both negative numbers
4-Bit numbers with one bit for sign

Add “-9”, “-4”; using 5 bits total
2’s Complement of 4 (00100) is (11100)
2’s Complement of 9 (01001) is (10111)

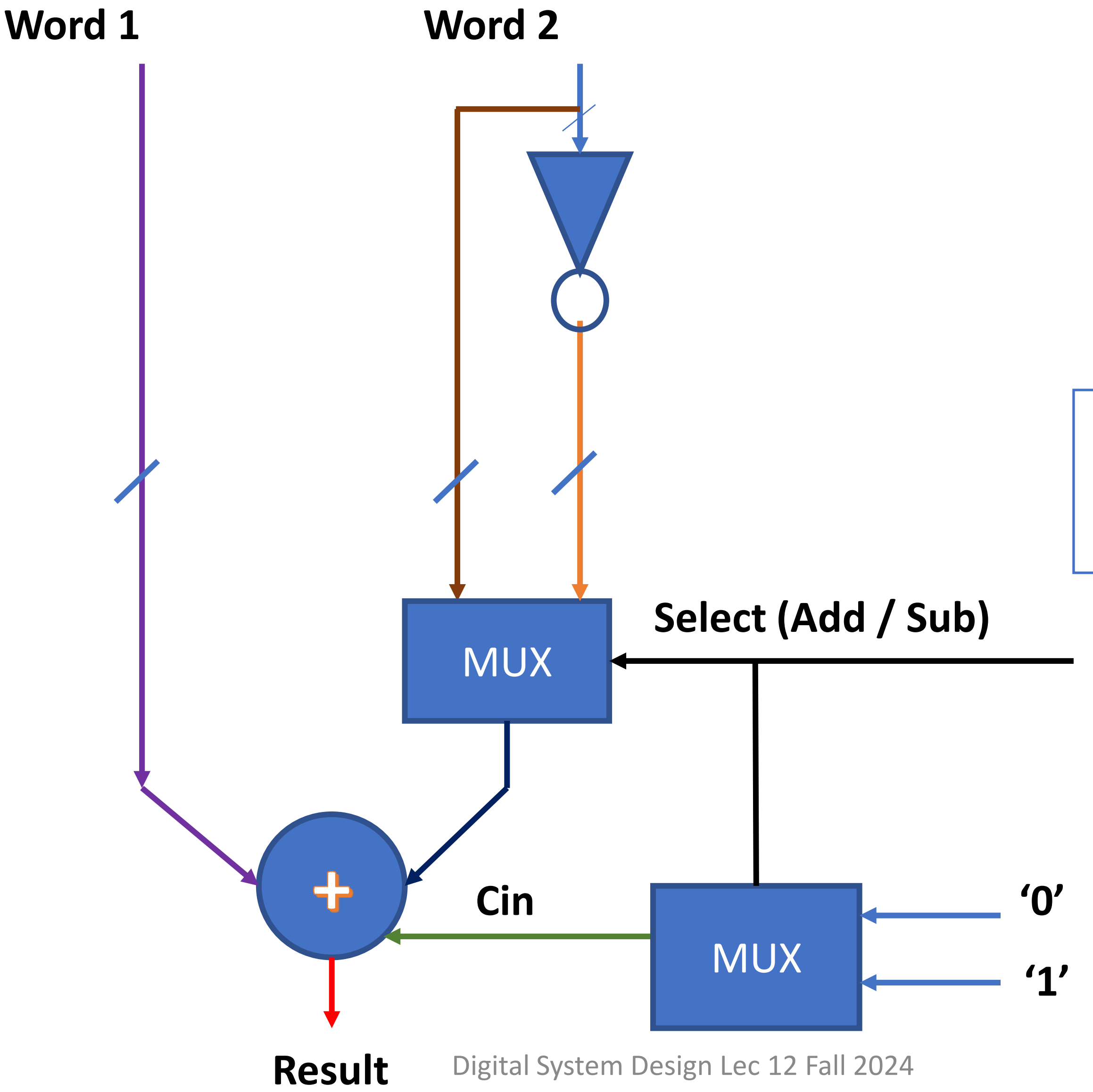
-9	1, Cin 1	0, Cin 1	1	1	1
-4	1	1	1	0	0
1 Cout	1	0	0	1	1

Extra bit, Sign bit
Carry out

Answer = “10011”
Take 2’Complement = “01101” = “-13”

Cout is beyond 5 bits, hence discarded

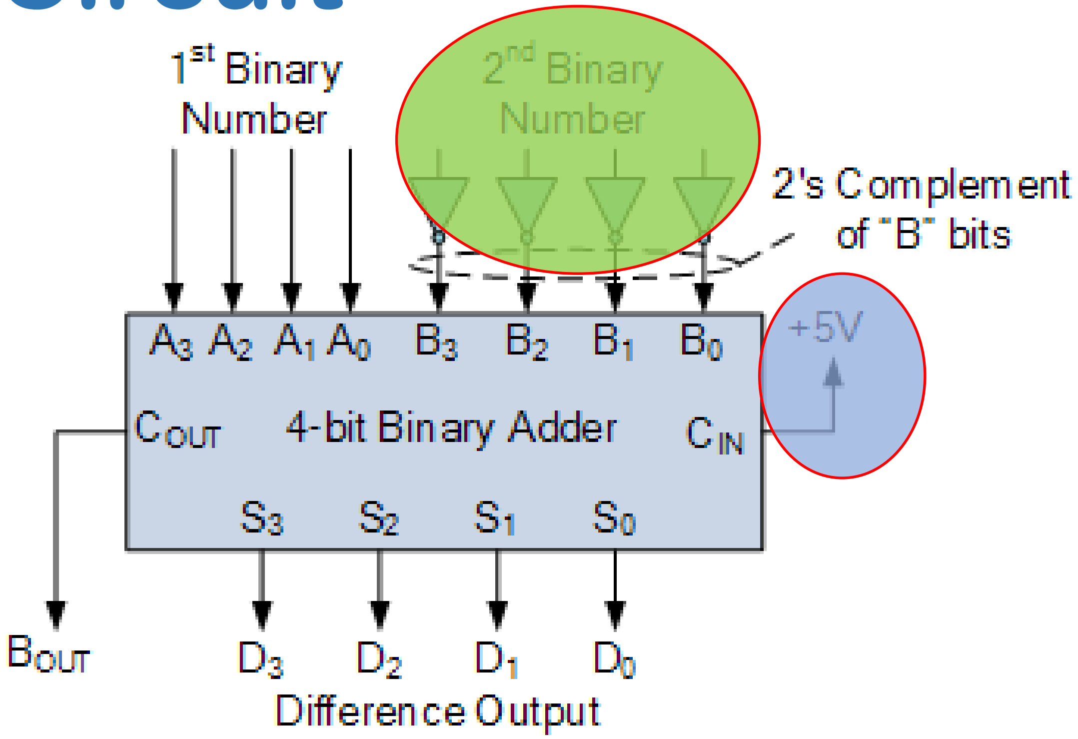
Combined Adder and 2's Complement Subtraction

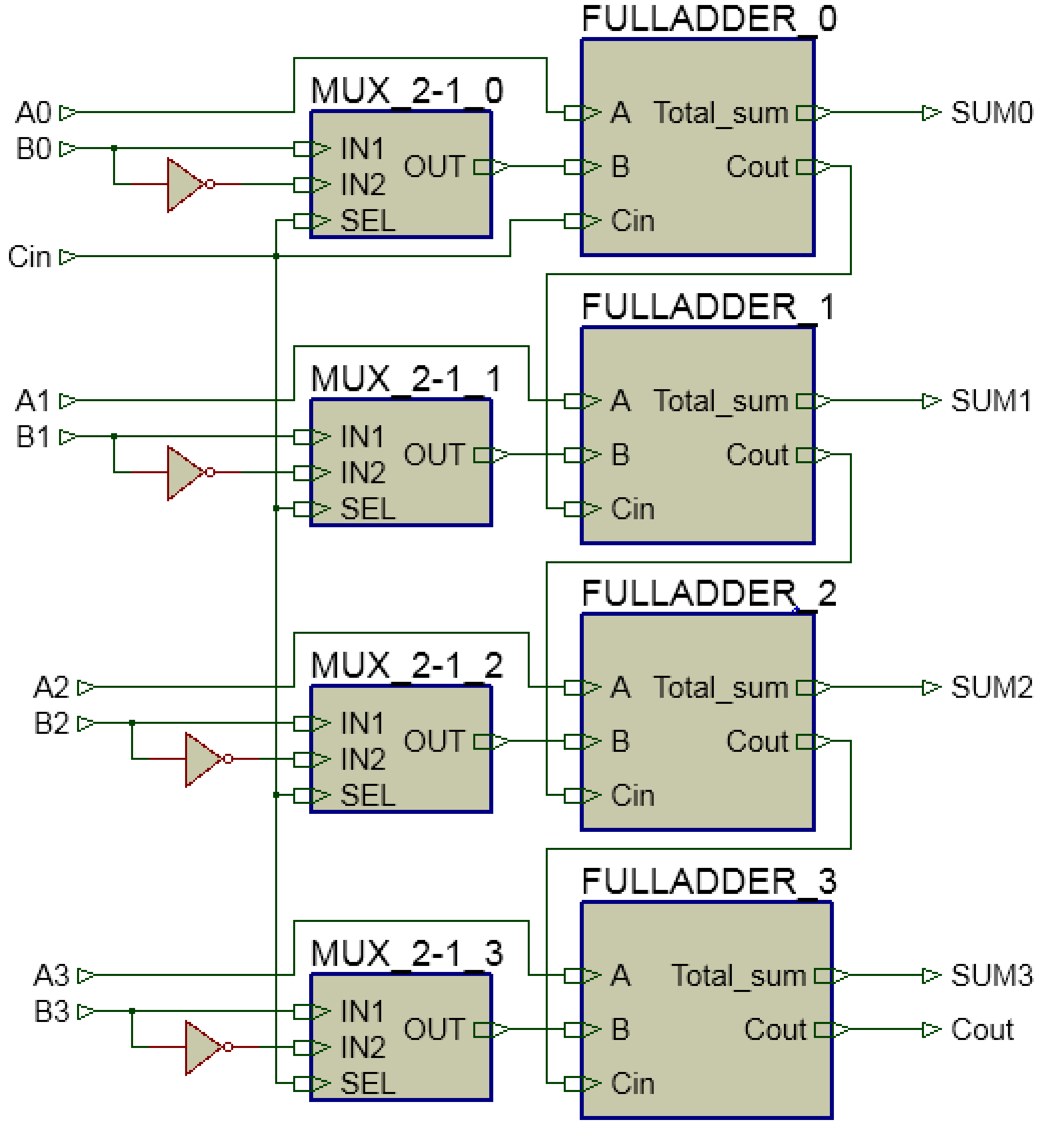


For 2's Complement,
Select the path through inverter and Cin = '1'

Select '1' for 2's Complement

4-Bit Adder/Subtractor Circuit





https://www.electronics-tutorials.ws/combination/binary-subtractor.html

Ref: https://www.electricaltechnology.org/2018/05/binary-adder-subtractor.html

Overflow Condition

Add "+9" and "+8", using 5-bit binary arithmetic circuit

+9	0, Cin 1	1	0	0	1
+8	0	1	0	0	0
	1	0	0	0	1

Sign bit

Answer = "10001" = "-1"

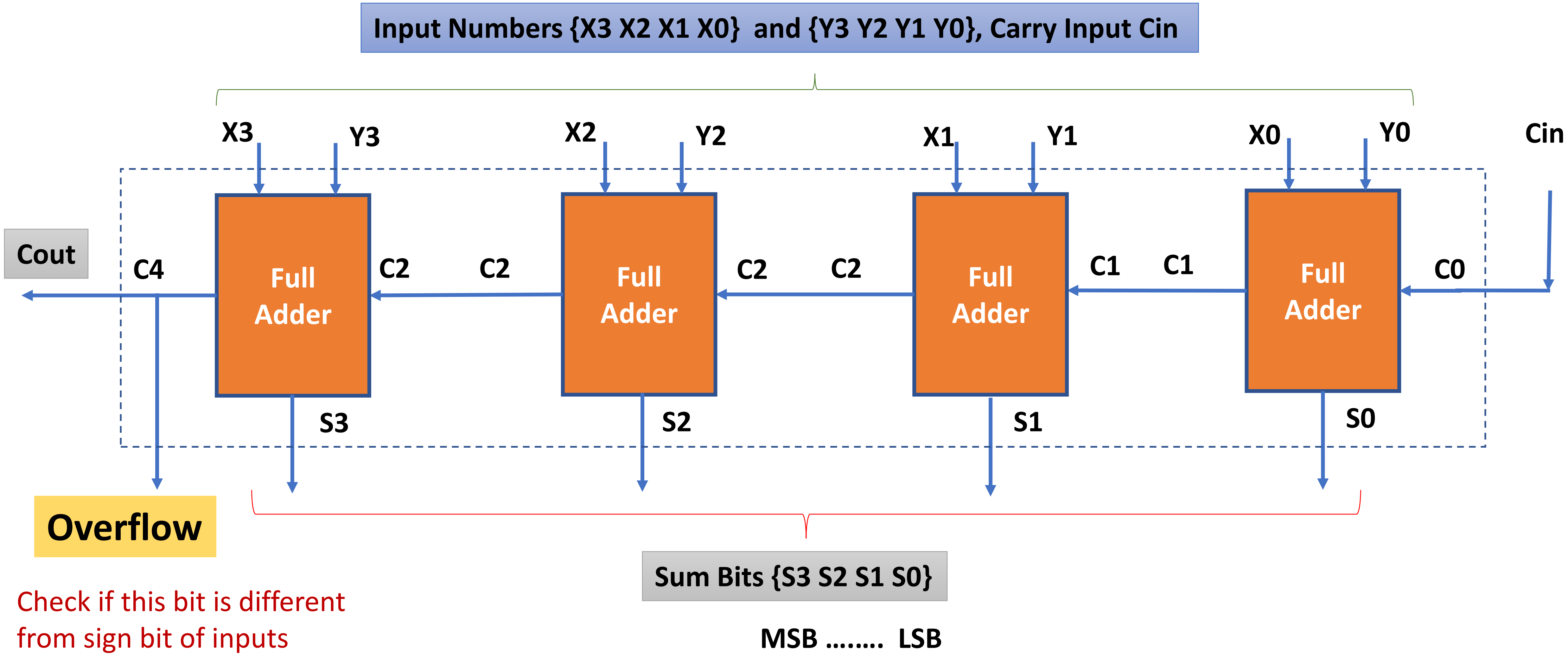
Both numbers are positive with '0' sign bit
The answer should be positive too with '0' sign bit
The above is the condition of 'Overflow'. Shows that bits allocated for this arithmetic operation are in-sufficient

The electronic circuits of a processor can easily detect overflow of unsigned binary addition by checking if the carry-out of the leftmost column is a zero or a one. A program might branch to an error handling routine when overflow is detected.

Overflow in 2's Complement

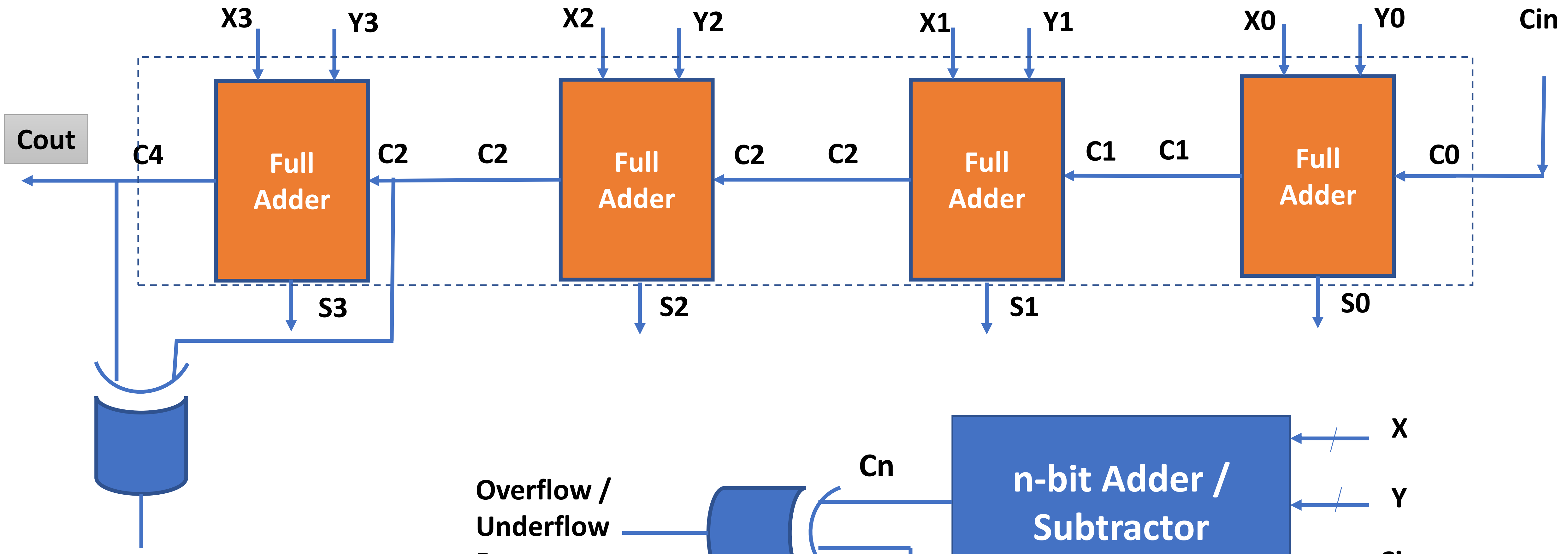
- If x and y have opposite signs (one is negative, the other is non-negative), then the sum will never overflow. The result will either be x or y or somewhere in between.
- Thus, overflow can only occur when x and y have the same sign.
- One way to detect overflow is to check the sign bit of the sum. If the sign bit of the sum does not match the sign bit of x and y, then there's overflow.
- Suppose x and y both have sign bits with value 1. That means, both representations represent negative numbers. If the sum has sign bit 0, then the result of adding two negative numbers has resulted in a non-negative result, which is clearly wrong. Overflow has occurred.
- Suppose x and y both have sign bits with value 0. That means, both representations represent non-negative numbers. If the sum has sign bit 1, then the result of adding two non-negative numbers has resulted in a negative result, which is clearly wrong. Overflow has occurred.

Overflow in Un-signed in Binary Adder / Sub

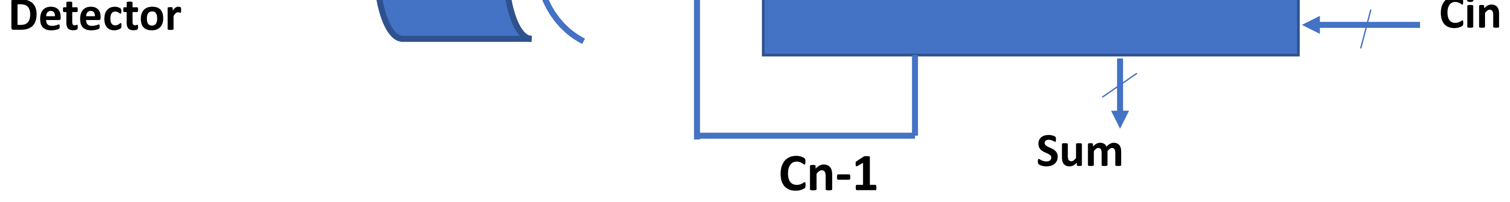


Check if this bit is different from sign bit of inputs

Overflow in Signed Binary Add / Sub



Overflow Detector



Binary Multiplication

Decimal Multiplication using Pencil and paper

```
      9 4 3
  x     1 8
  ---------
    7 5 4 4
  + 9 4 3 0
  ---------
  1 6 9 7 4
```

Keep shifting right

Keep shifting left

Array Multipliers – Parallel and Serial forms

$$X = \sum_{i=0}^{m-1} X_i.2^i$$

$$Y = \sum_{j=0}^{n-1} Y_j\,2^j$$

$$P = X.Y = \sum_{i=0}^{m-1} X_i\ 2^i.\ \sum_{j=0}^{n-1} Yj, 2^j$$

$$P = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(X_iY_j)2^{i+j}$$

$$P = \sum_{k=0}^{m+n-1} P_k 2^k$$

				X_3	X_2	X_1	X_0
				Y_3	Y_2	Y_1	Y_0
				X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0
			X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1	Nil
		X_3Y2	X_2Y_2	X_1Y_2	X_0Y_2	Nil	Nil
	X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3	Nil	Nil	Nil
Cout	P_6	P_5	P_4	P_3	P_2	P_1	P_0

Complexity of Binary Array Multiplier

				X_3	X_2	X_1	X_0
				Y_3	Y_2	Y_1	Y_0
				X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0
			X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1	0
		X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2	0	0
	X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3	0	0	0
Cout	P_6	P_5	P_4	P_3	P_2	P_1	P_0

How many AND gates?
How many Adders?
Propagation Delay – Longest Carry Ripple Path

Complexity and Timing

For an n-bit x n-bit multiplier;
We need:

n(n-2) full adders

n half adders

n^2 AND Gates

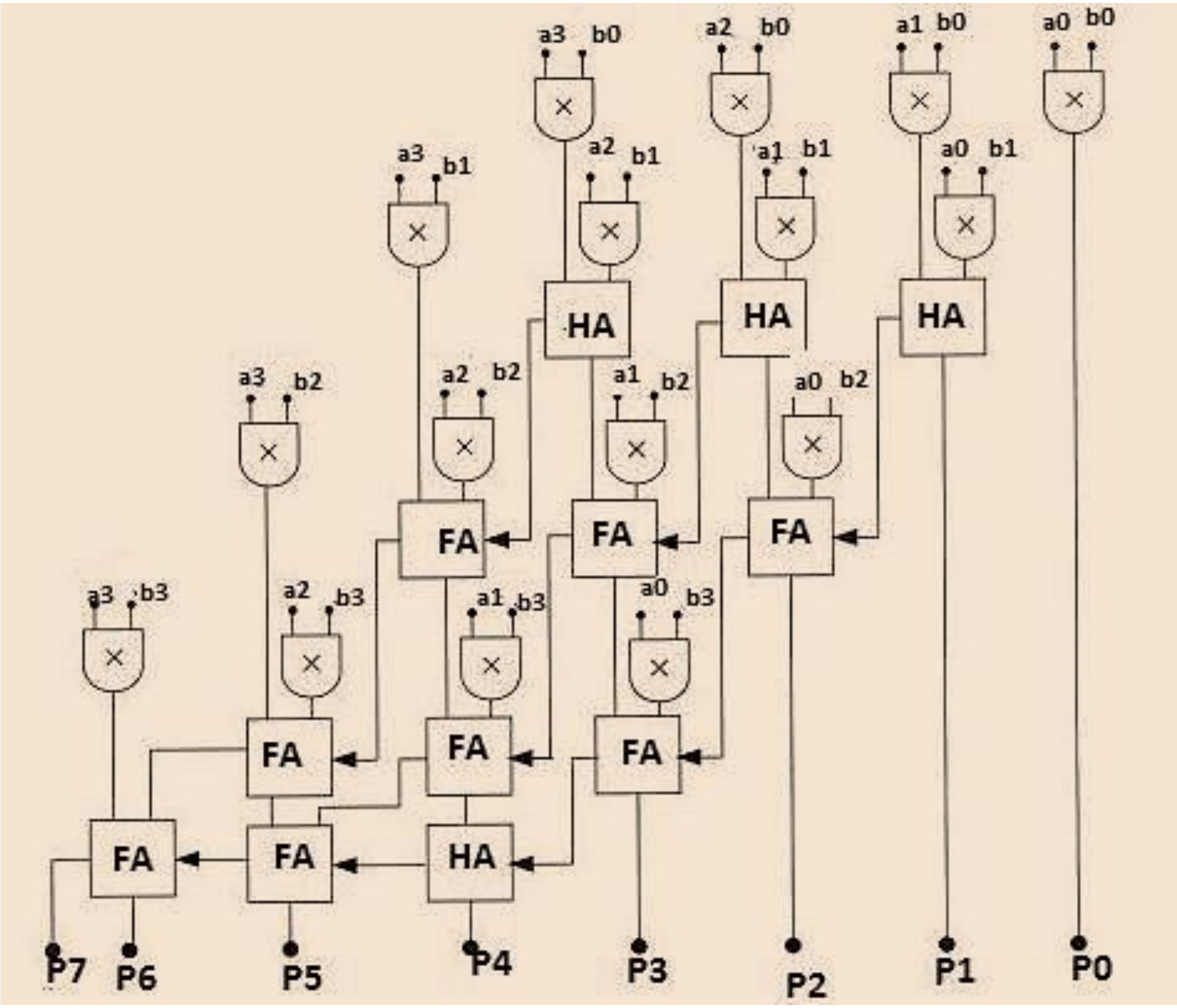
Worst Case Delay is (2n+1) Ꚍ
Where Ꚍ is the worst adder delay

Question: Delay estimation in Array Multiplier

- **Assume, delay through AND Gate is 10ns**
- **Delay through half-adder is 30ns**
- **Delay through full-adder is 50ns**
- **Calculate Critical path delay of a 4-bit array multiplier**
- **Calculate maximum clock speed for this 4-bit array multiplier**

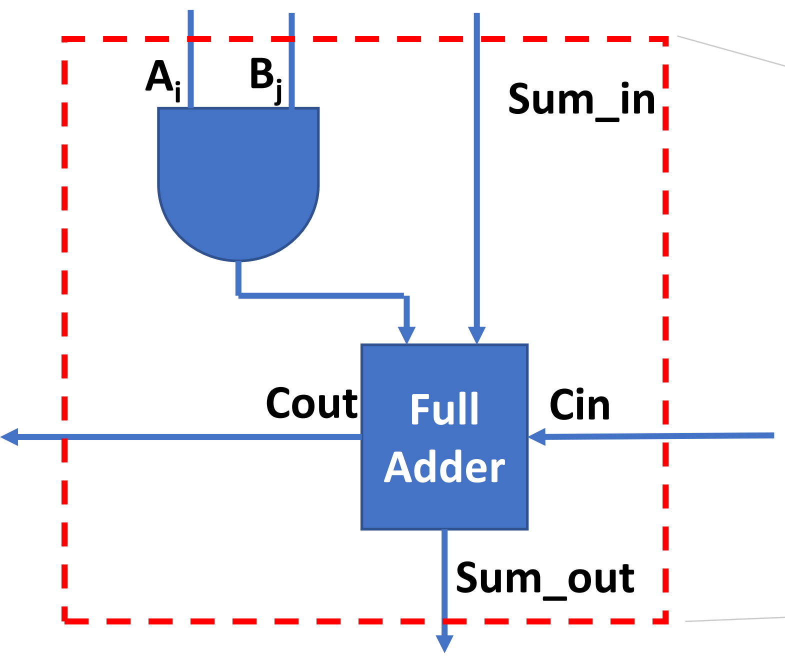
4-Bit Array Multiplier connected as AND and ADD

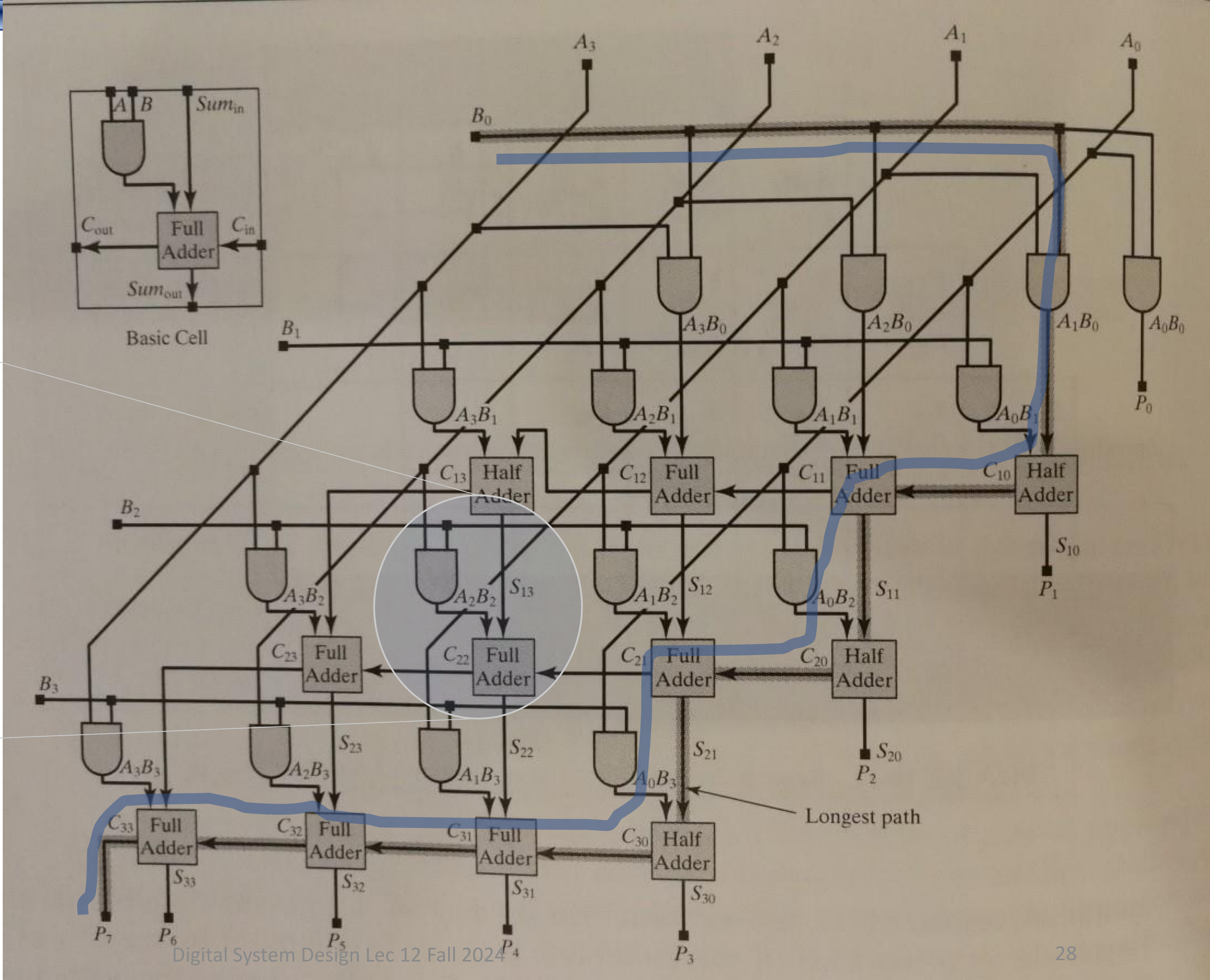
				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	0
		A_3B_2	A_2B_2	A_1B_2	A_0B_2	0	0
	A_3B_3	A_2B_3	A_1B_3	A_0B_3	0	0	0
Cout	P_6	P_5	P_4	P_3	P_2	P_1	P_0



Array Multiplier Circuit Delays

Building Block

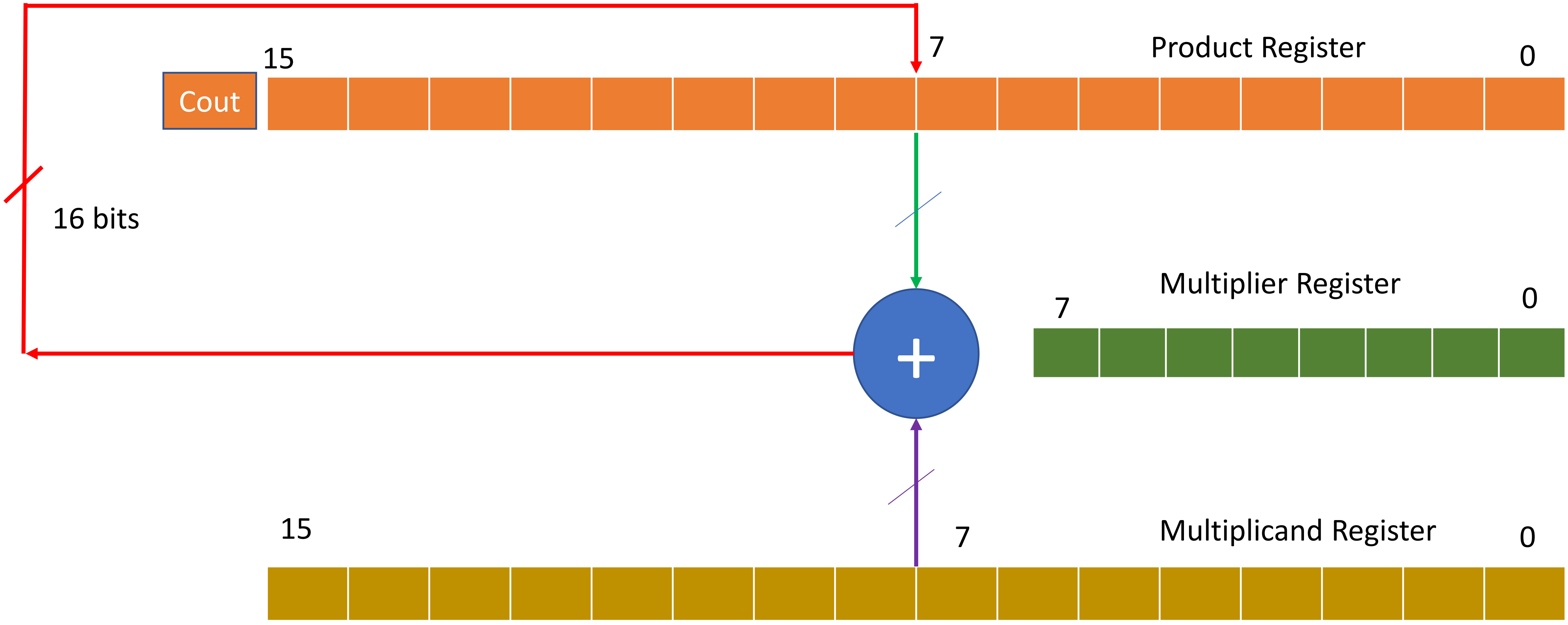




Worst case delay path is shaded

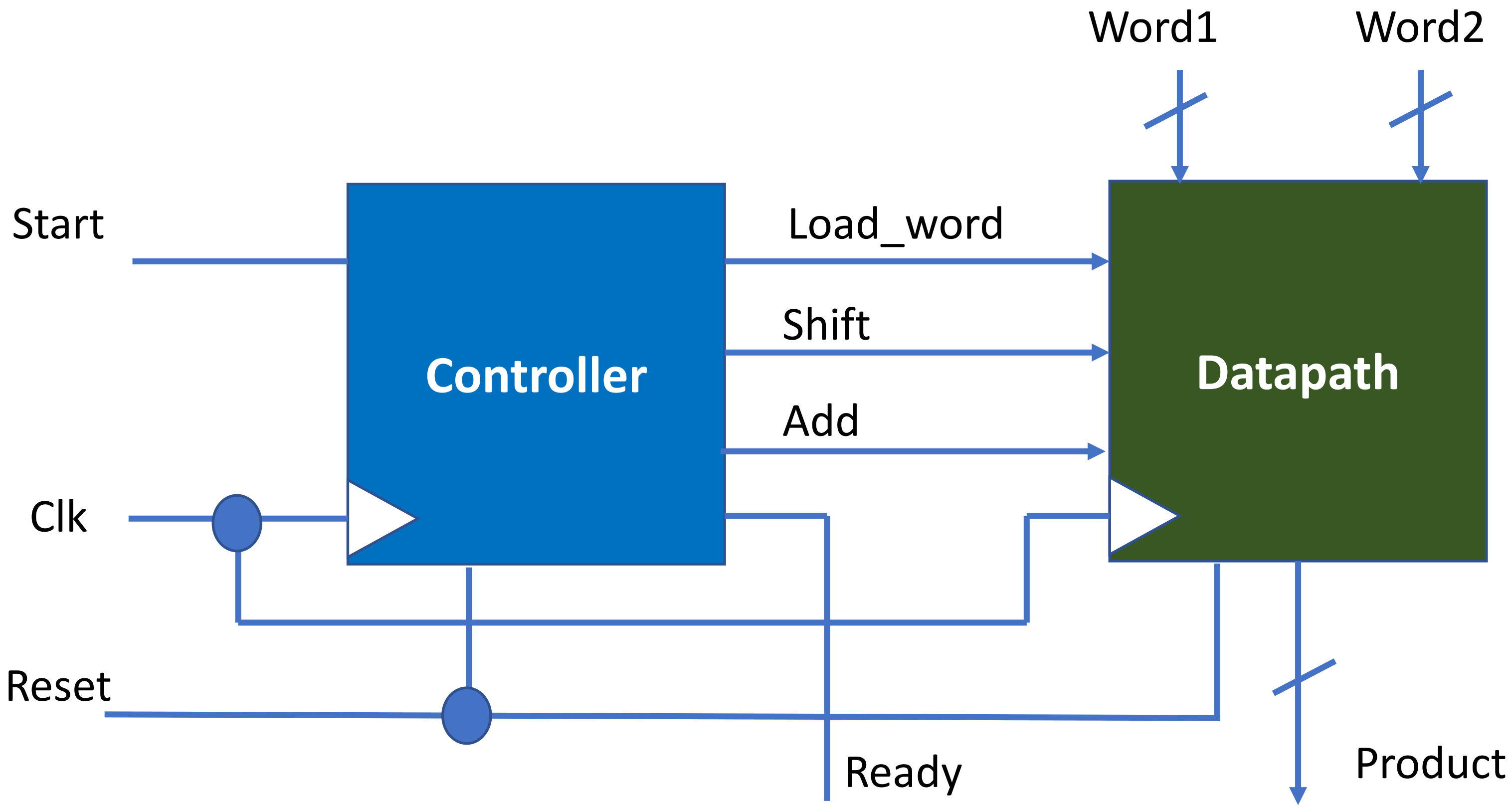
This is Critical Path

Operation of Sequential Multiplier



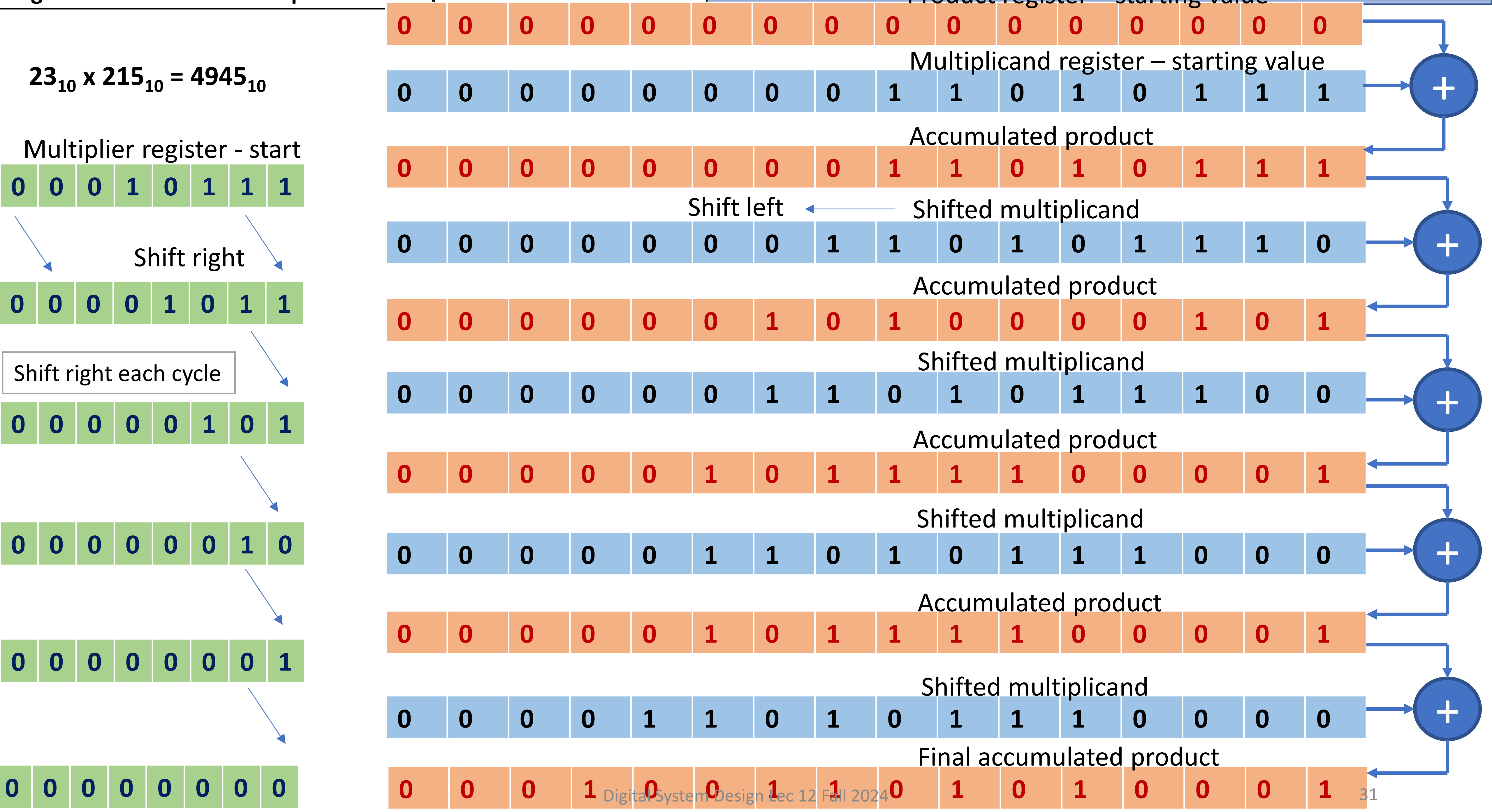
Datapath of a Sequential Multiplier

Data Path Architecture of Sequential Mult

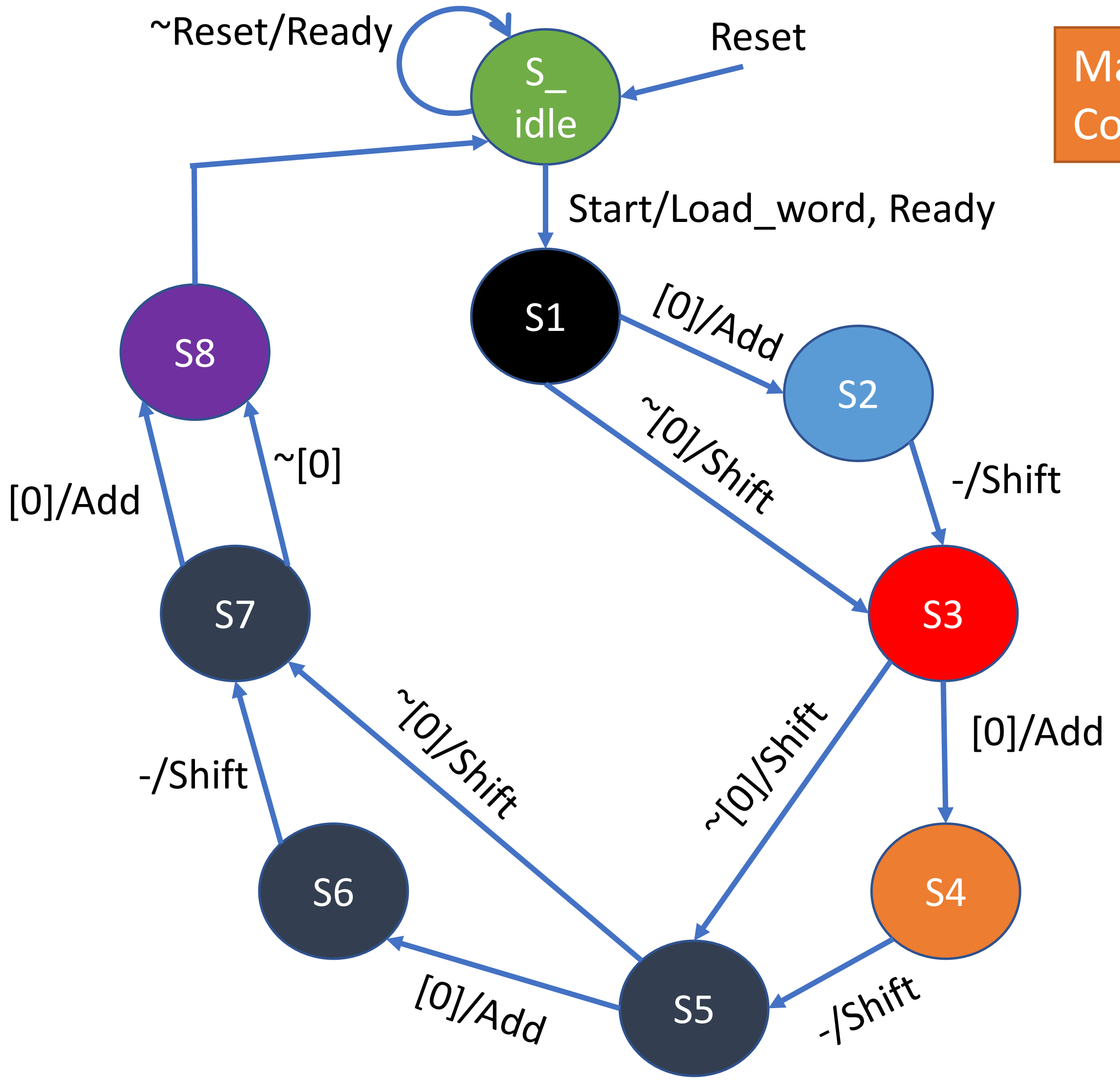


Register transfer in 8-bit Sequential Multiplier

$23_{10} \times 215_{10} = 4945_{10}$



STG for a 4 Bit Sequential Binary Multiplier



Machine returns to Idle state after Completion of 4 bit multiplication