Lecture 13
EE 421 / CS 425
Digital System Design

Fall 2025

Shahid Masud

Topics

- Binary Array Multipliers – Quick Recap
- Operation of Sequential Multiplier
- Control Circuits for Multipliers
- Reducing Registers in Sequential Multipliers
- Taking care of sign in Signed Multiplication
- Fractional Binary numbers
- **QUIZ 3 NEXT LECTURE**

Decimal Multiplication using Pencil and paper

$$
\begin{array}{rrrr}
 & 9 & 4 & 3 \\
\times & & 1 & 8 \\
\hline
 & 7 & 5 & 4 & 4 \\
+ & 9 & 4 & 3 & 0 \\
\hline
1 & 6 & 9 & 7 & 4
\end{array}
$$

Keep shifting right

Keep shifting left

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?
Identify longest Carry 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

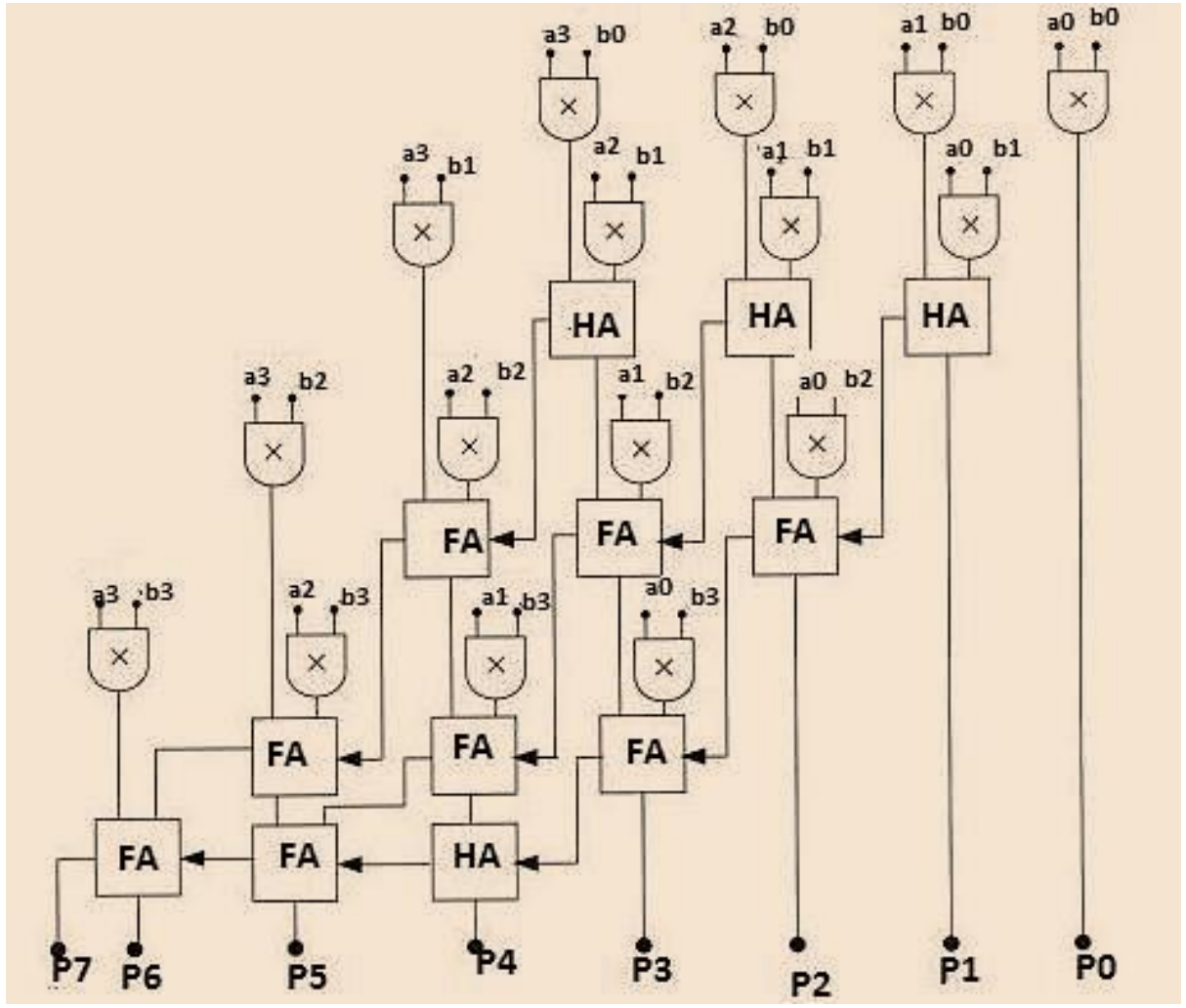
Worst Case Delay is (2n+1) Ƈ
Where Ƈ is the worst adder delay

Designing An Array Multiplier Cell

				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

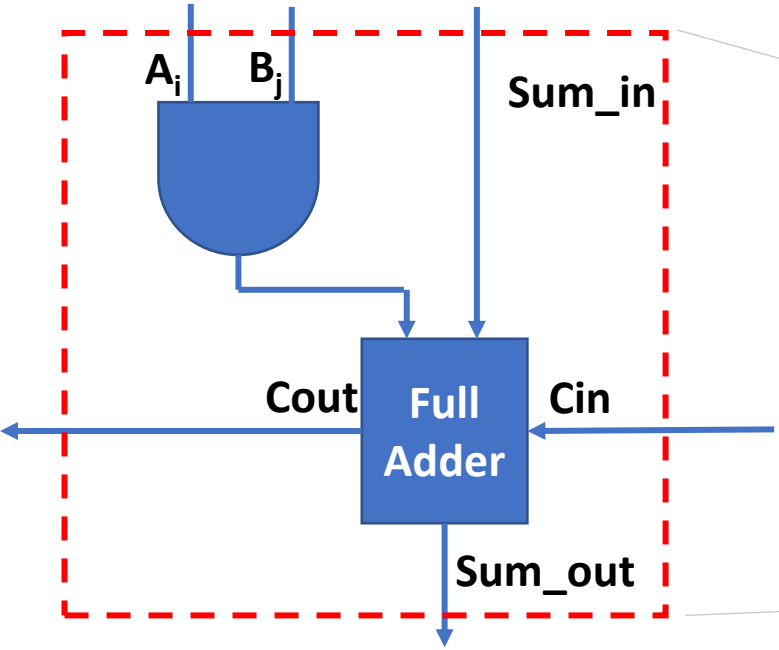
4-Bit Array Multiplier connected as AND and ADD

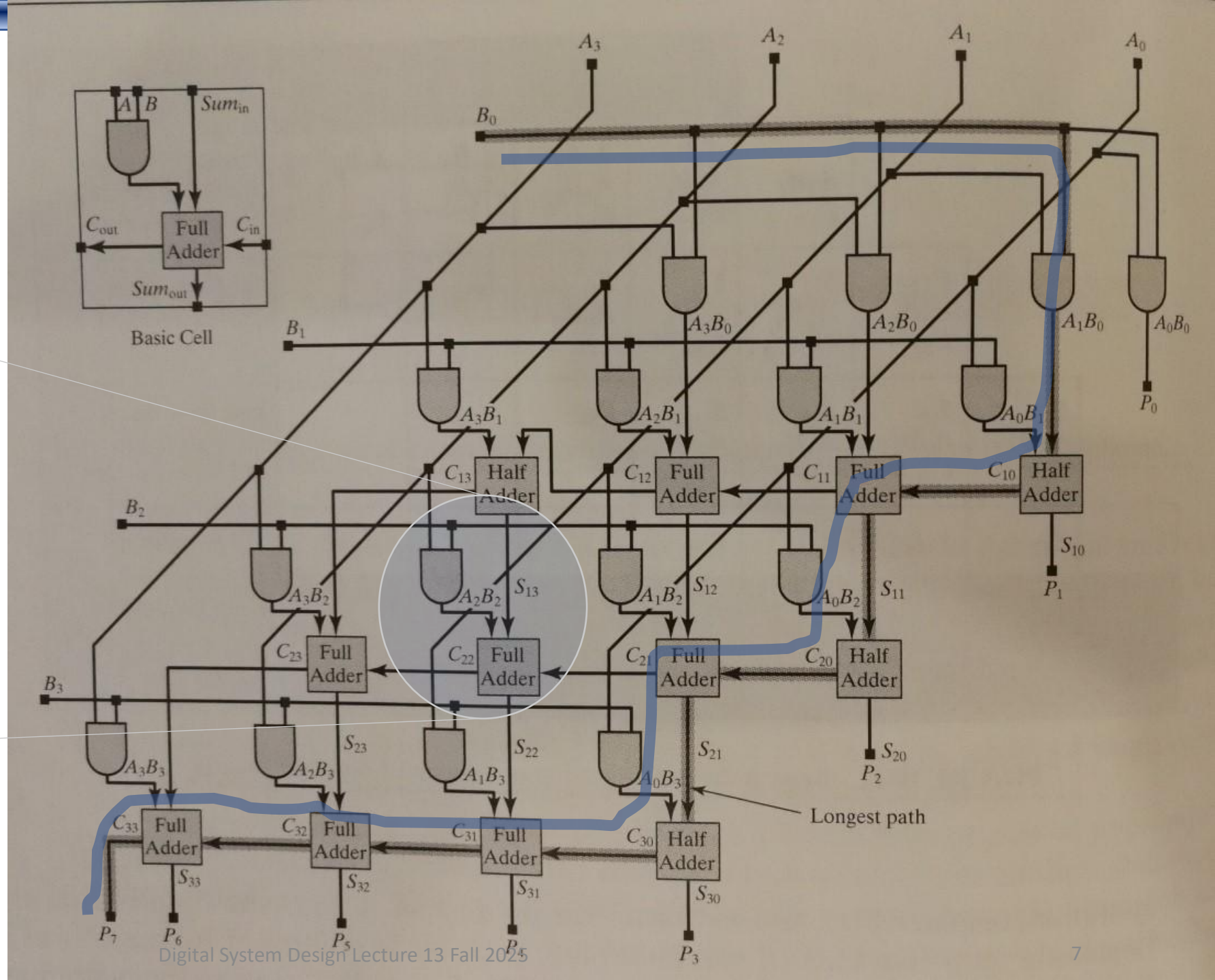
				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	0
		A_3B_2	A_2B_2	A_1B_2	A_0B_2	0	0
	A_3B_3	A_2B_3	A_1B_3	A_0B_3	0	0	0
Cout	P_6	P_5	P_4	P_3	P_2	P_1	P_0



Array Multiplier Circuit Delays

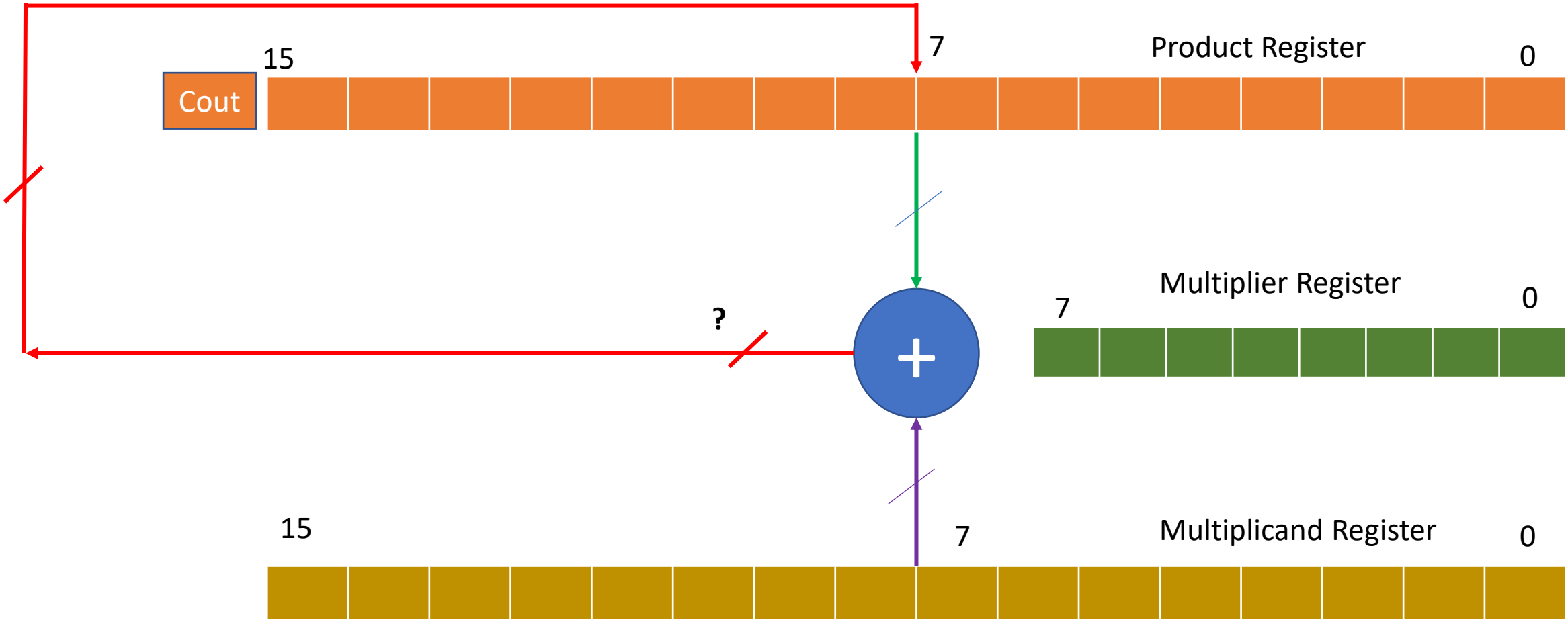
Building Block





Worst case delay path is shaded
This is Critical Path

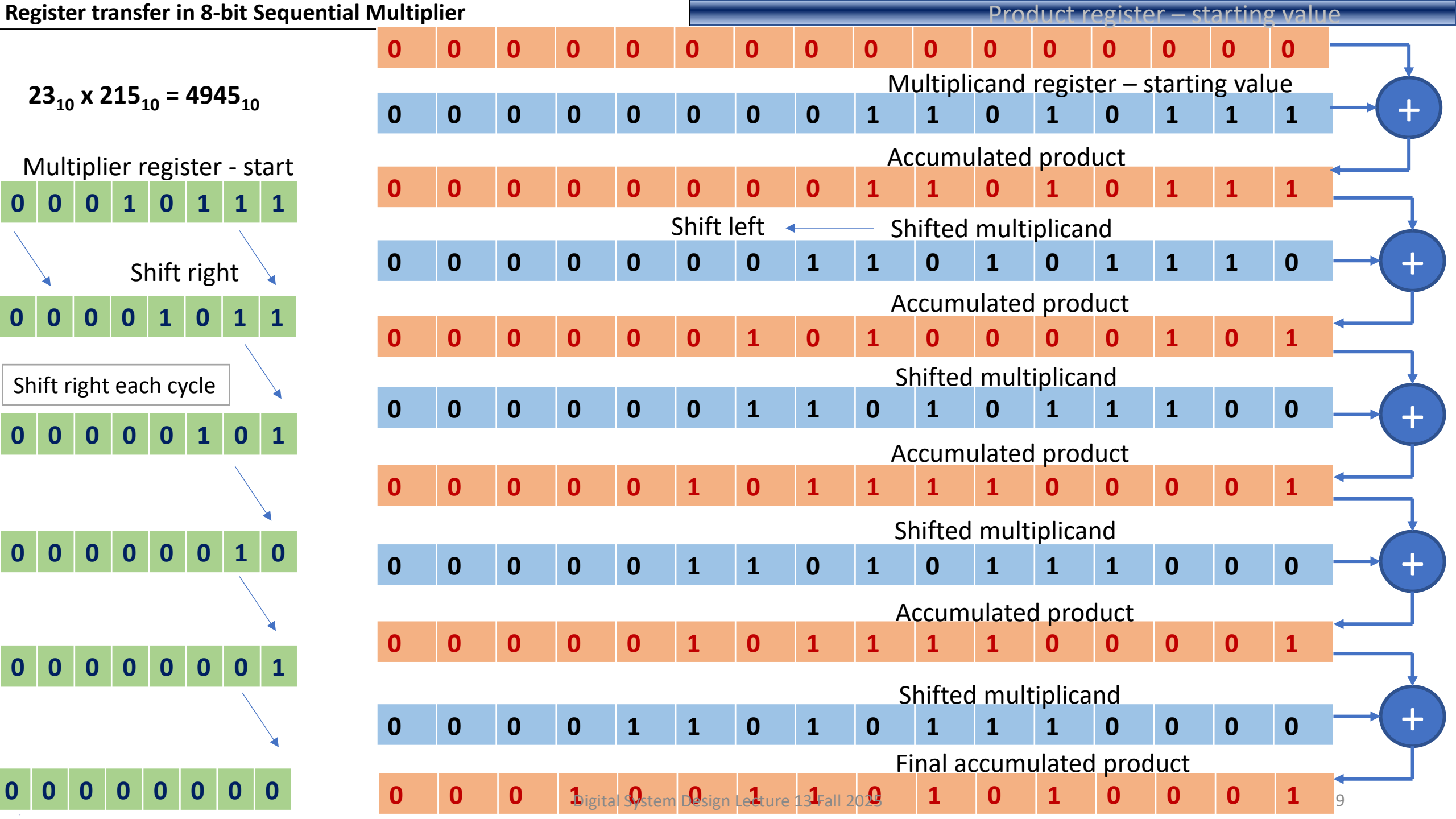
Operation of Sequential Multiplier



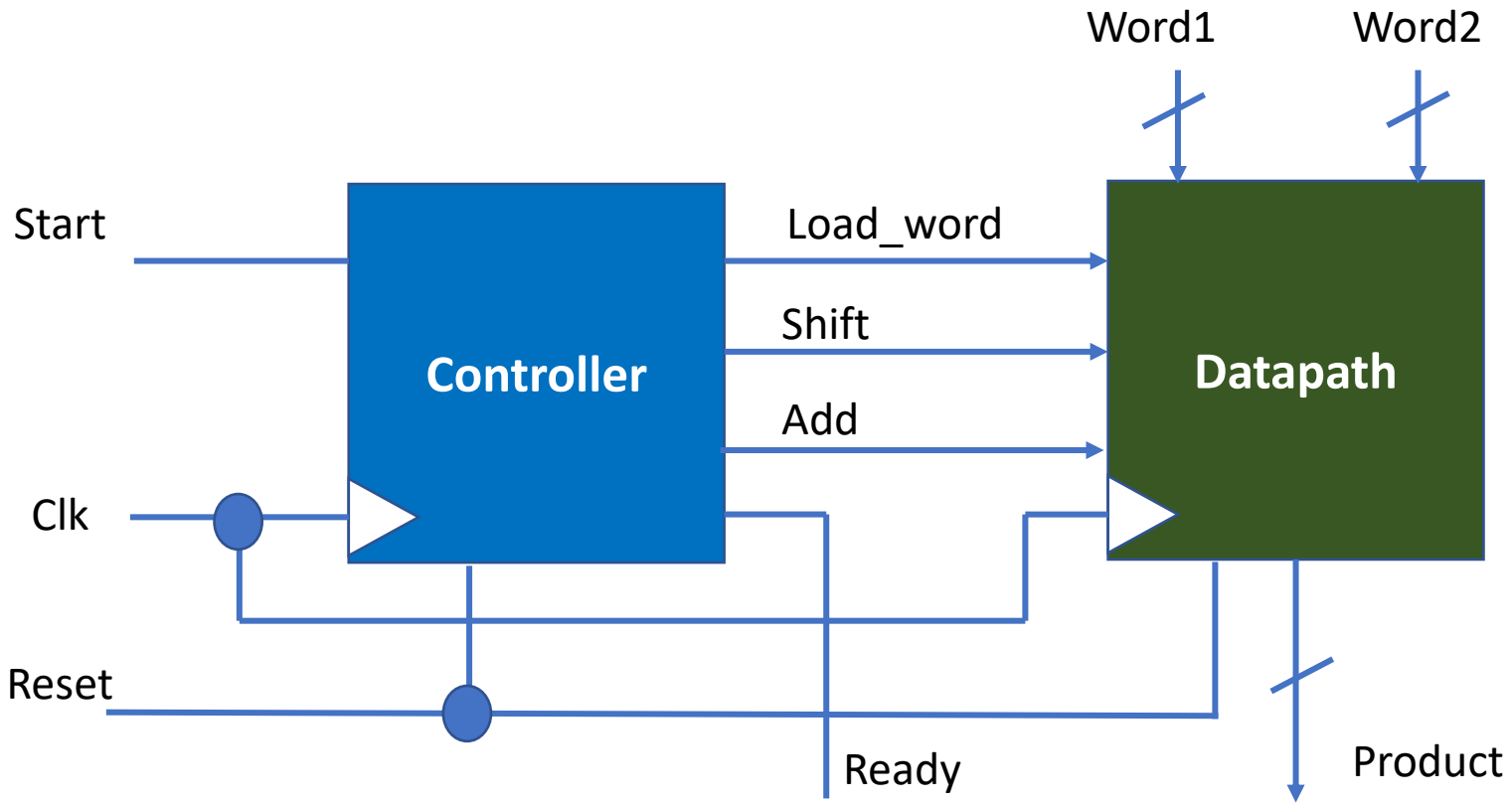
Datapath of a Sequential Multiplier

Register transfer in 8-bit Sequential Multiplier

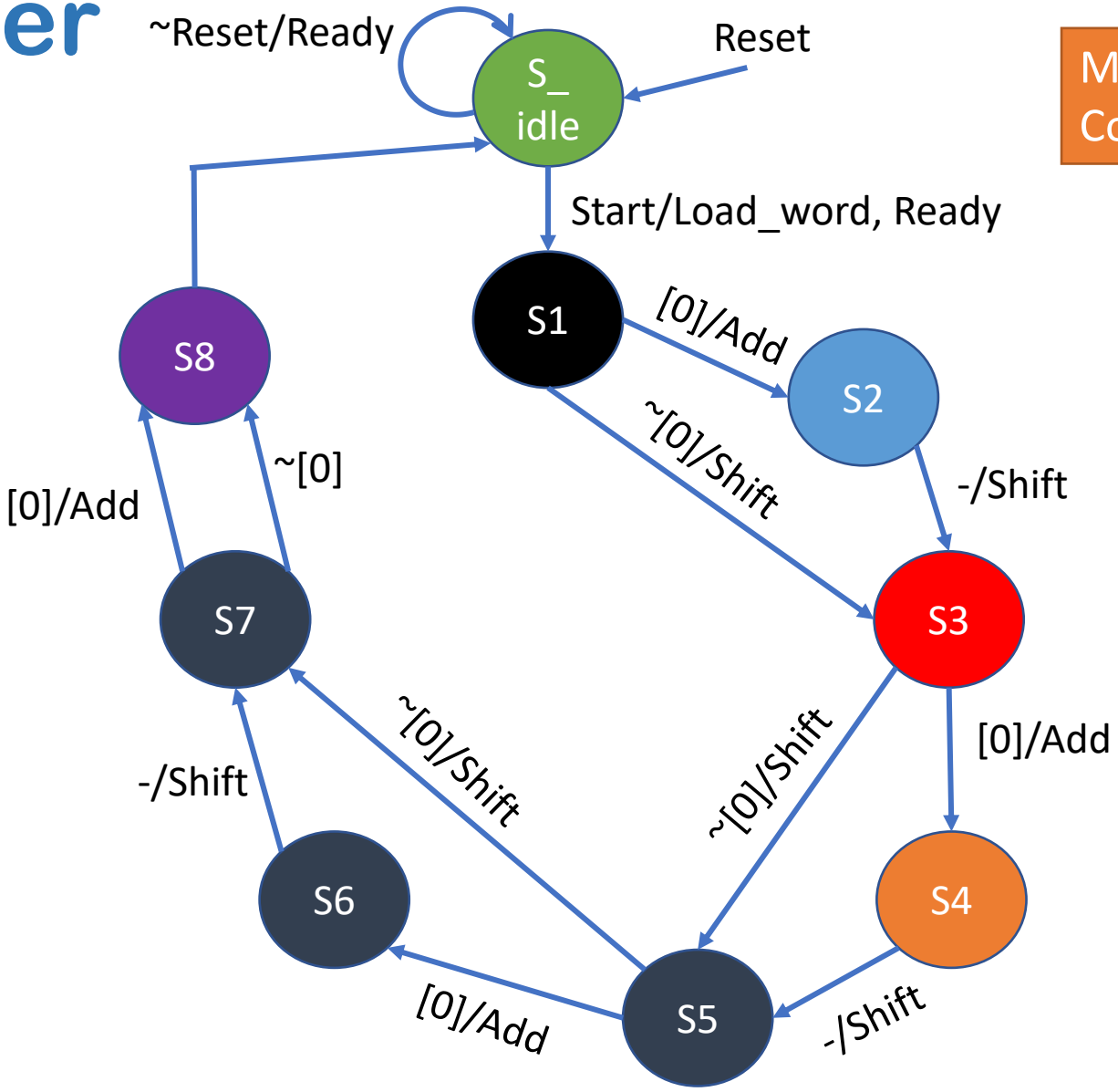
$23_{10} \times 215_{10} = 4945_{10}$



Data Path Architecture of Sequential Mult

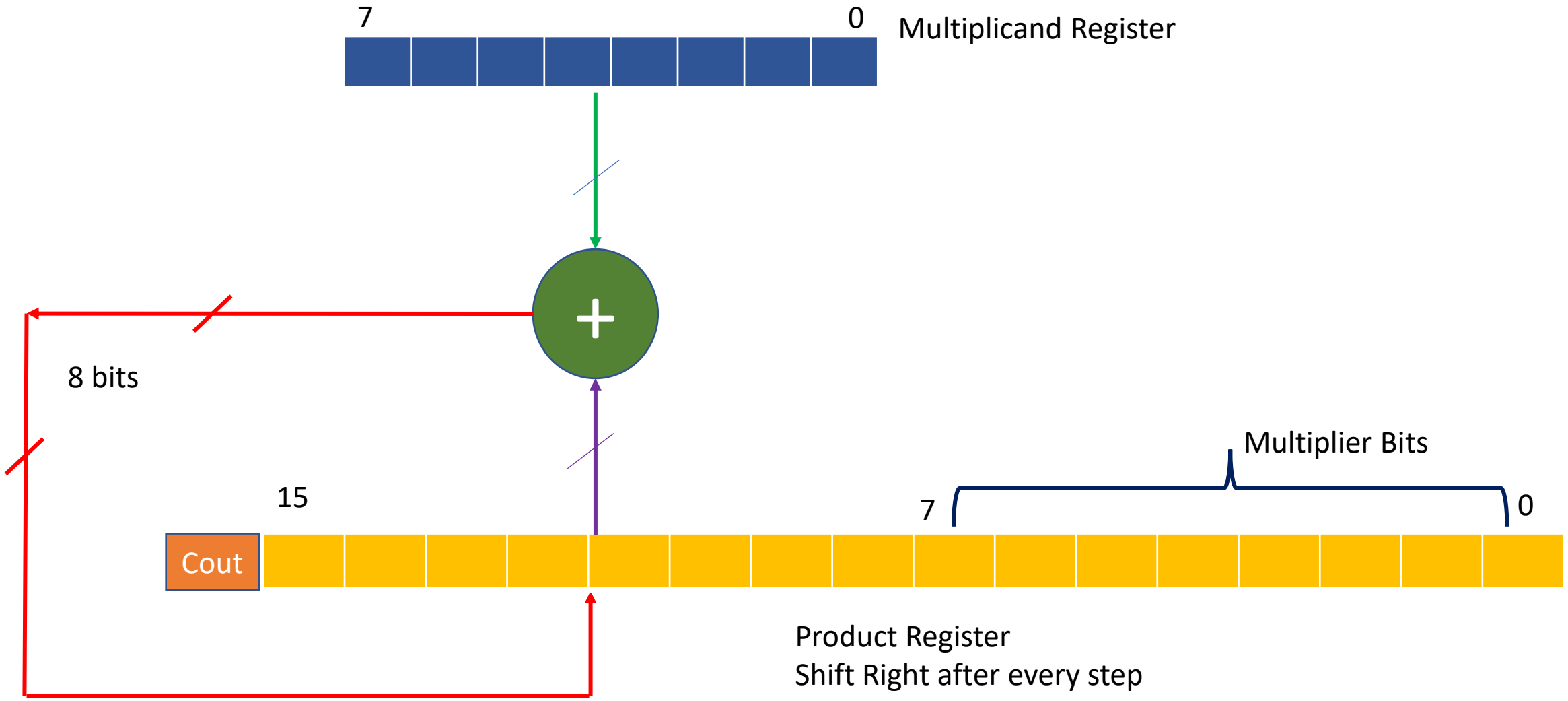


STG for a 4 Bit Sequential Binary Multiplier

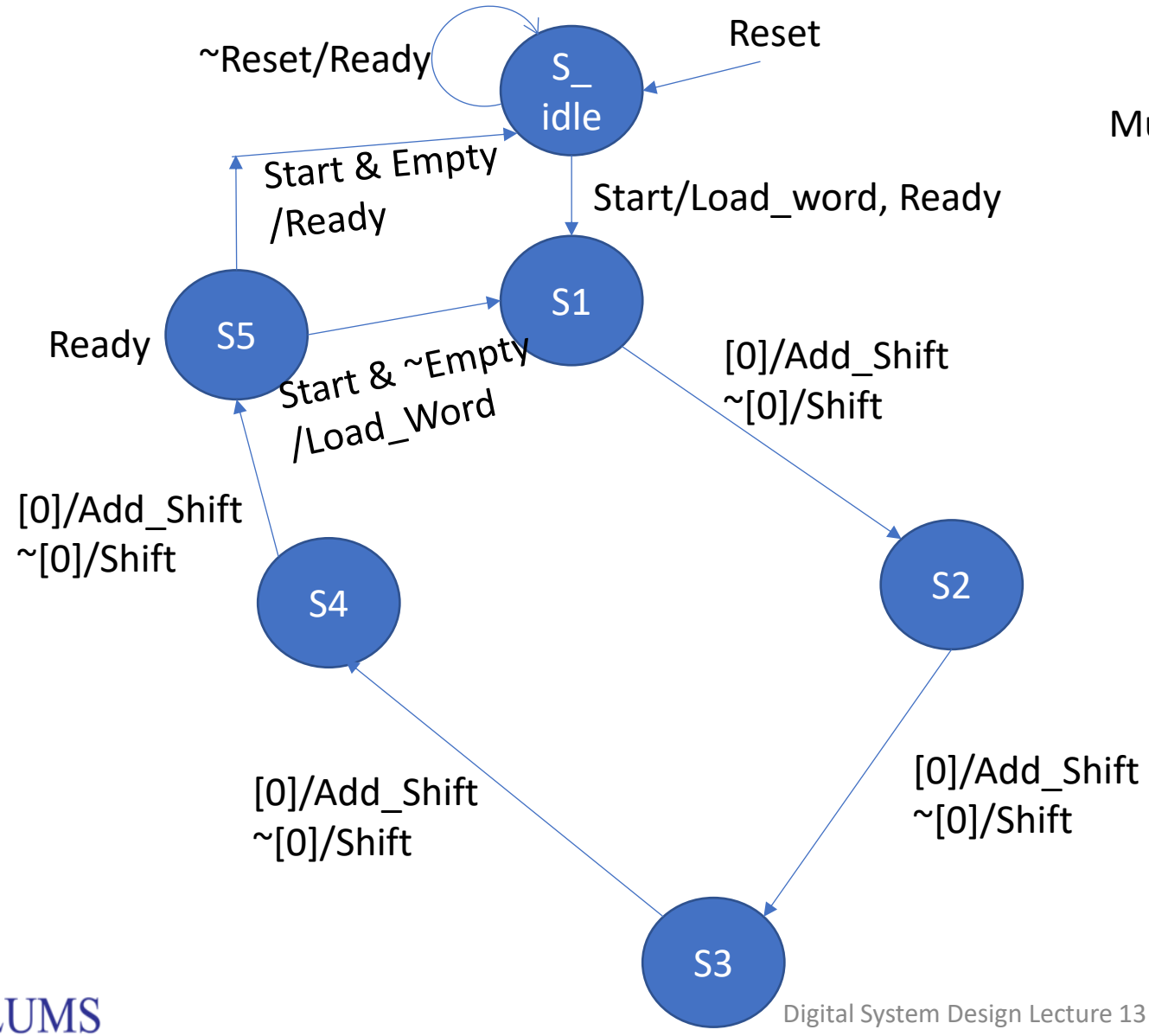


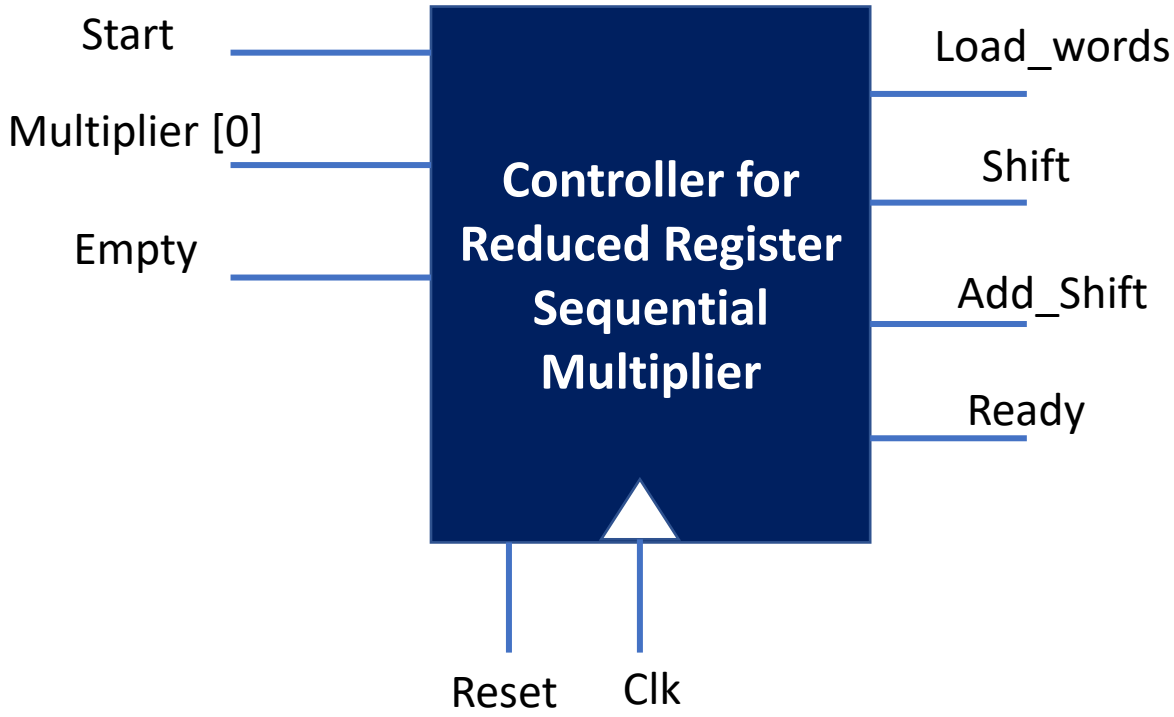
Machine returns to Idle state after Completion of 4 bit multiplication

Improvement - Sequential Multiplier with Reduced Registers

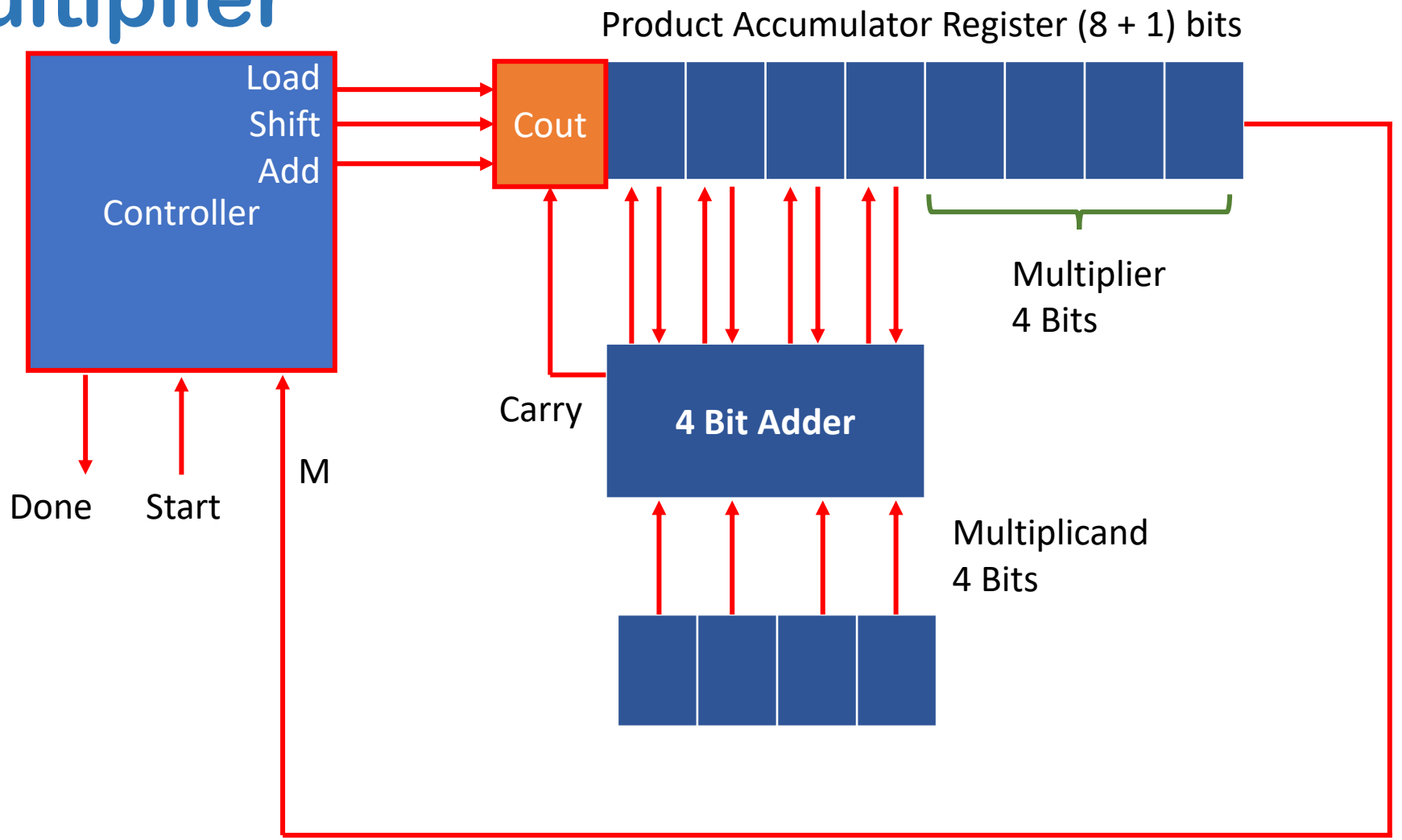


STG of Reduced Register Sequential Multiplier





Example of a 4-bit Serial Parallel Multiplier



Shift the contents
To the right after
Every step

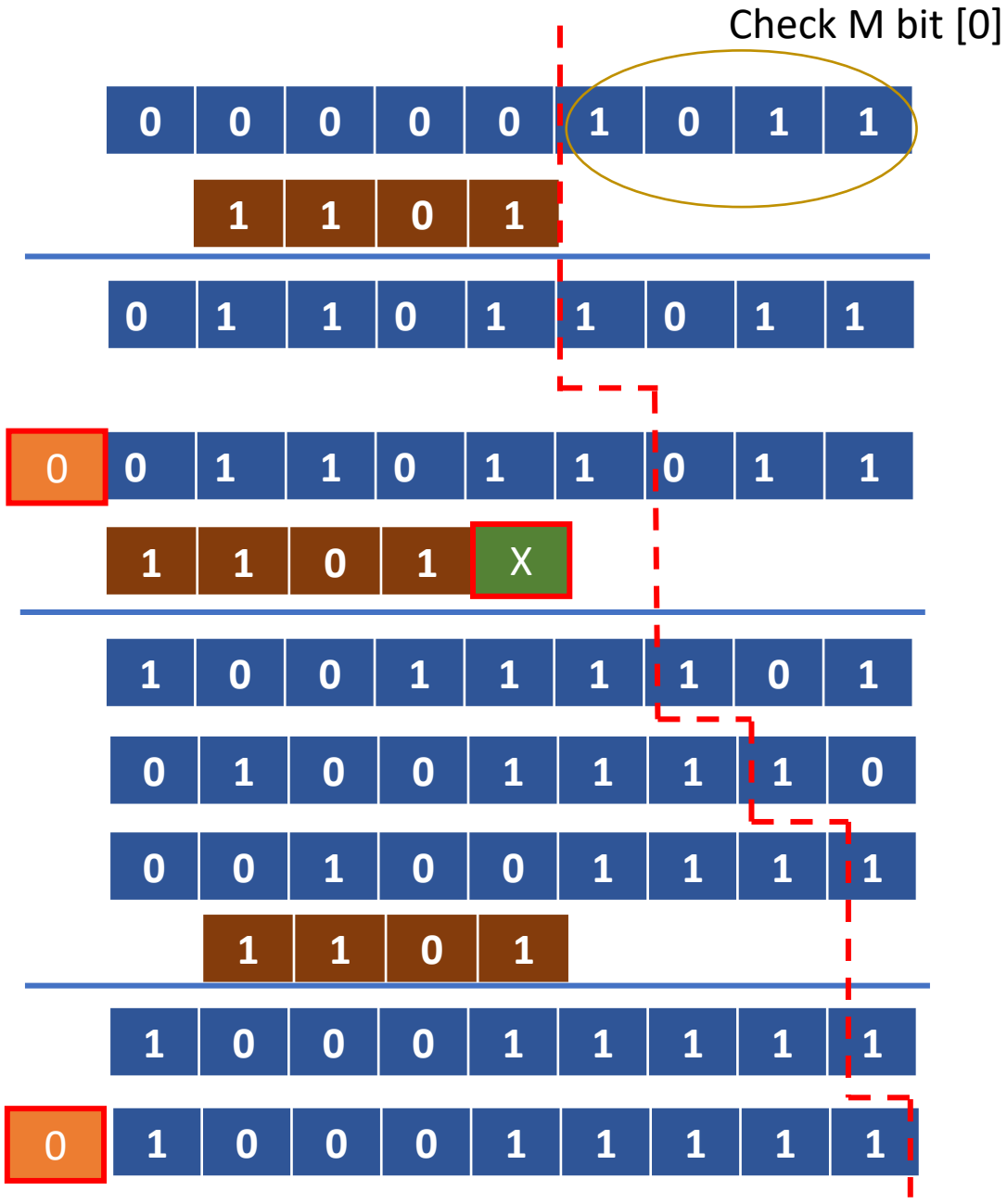
Example continued

				1	1	0	1	Multiplicand
			x	1	0	1	1	Multiplier
				1	1	0	1	
			1	1	0	1	X	Shift Left by one
		1	0	0	1	1	1	Partial product after first step
		0	0	0	0	X	X	Another shift left
		1	0	0	1	1	1	Partial product after second step
	1	1	0	1	X	X	X	Another shift left
1	0	0	0	1	1	1	1	Partial product after final step

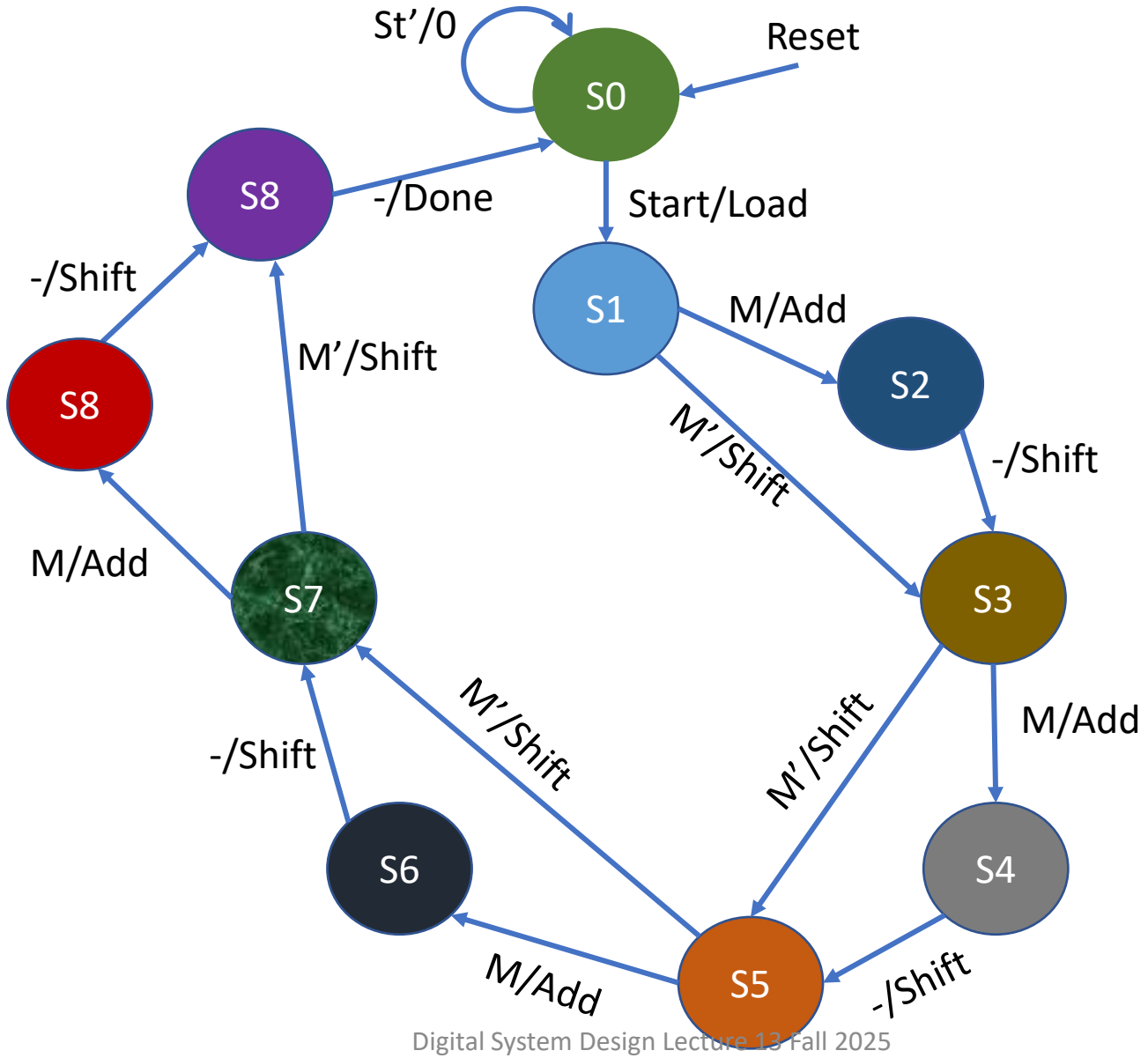
Answer = $(10001111)_2 = (143)_{10}$

Multiplier Register Operation

Initial Contents of Accumulator
Multiplicand bit [0] is ‘1’
After Add operation
After Shift Right
Next bit M = 1 hence Add
After Add
After Shift Right
Next bit M=0, hence Skip Add operation
After Shift Right
Next bit M=1, hence Add
After Addition
After Shift Right, final answer

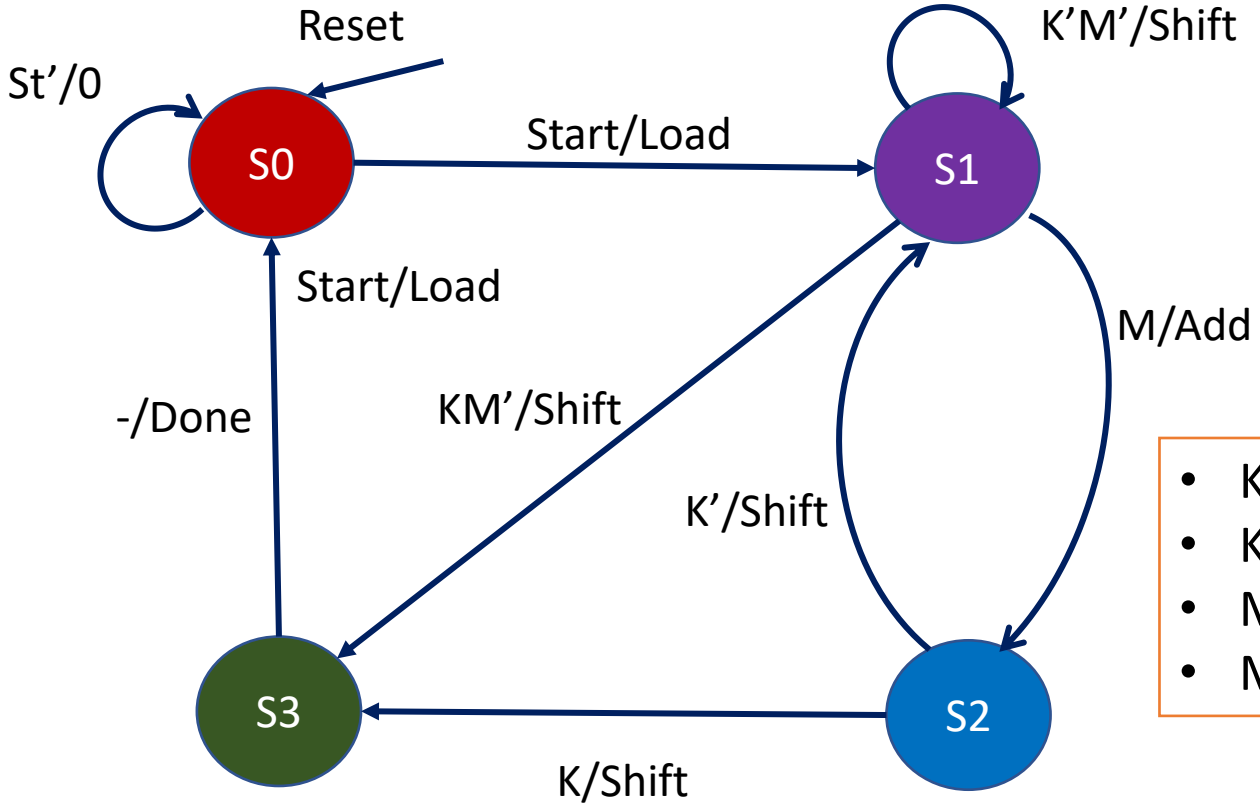


STG Control Diagram for this Multiplier



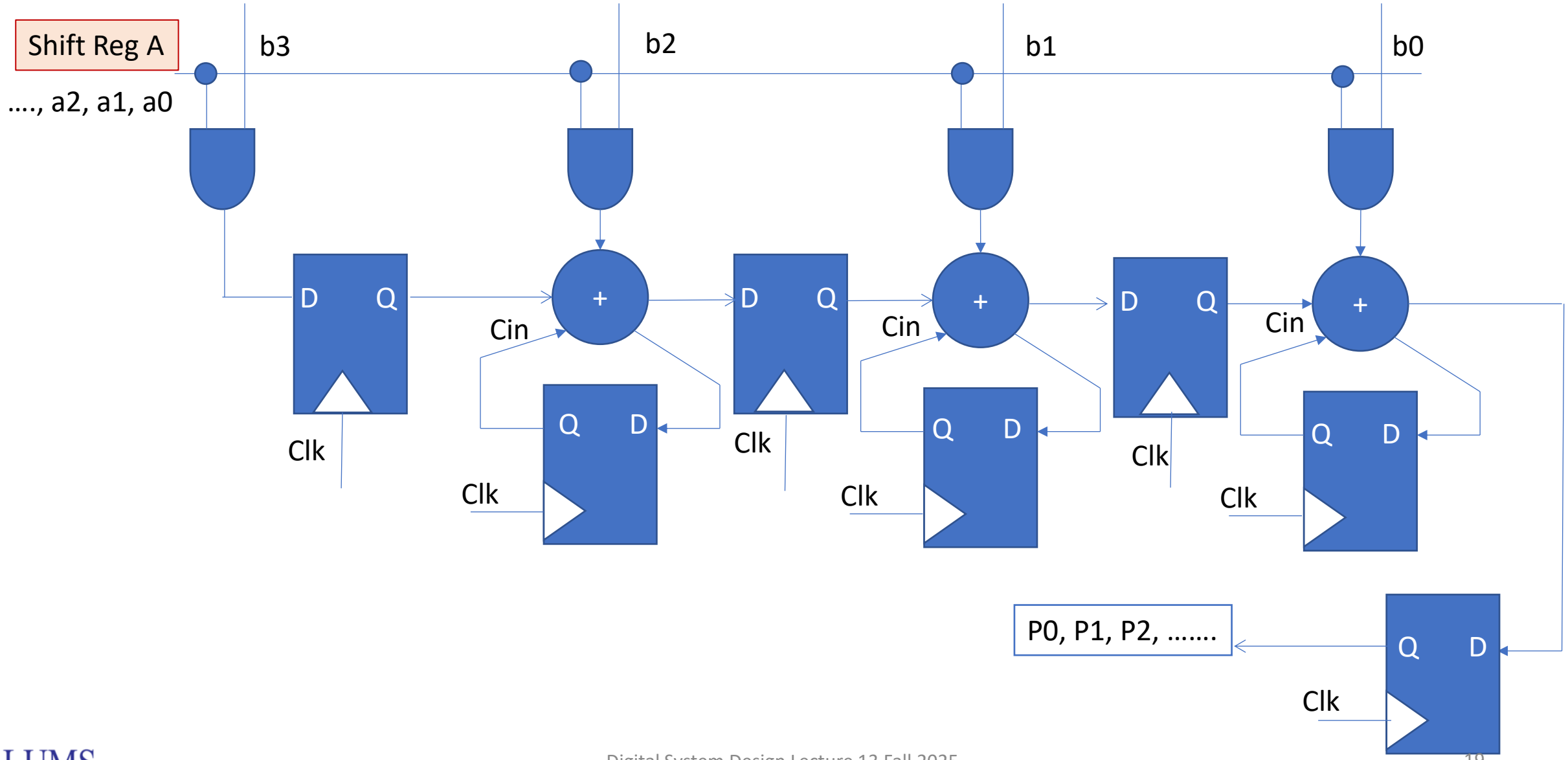
M = relevant bit of multiplier

Flexible STG for any no. of multiplicand bits



- K = Completion signal generated after (n-1) shifts
- K=1 means one more add (if needed) and shift operation
- M=0 means do the final shift
- M=1 means add before shift and go to S2

Parallel-Serial Multiplier - Concept



Multiplication of Signed Binary Numbers

Case I: Negative Multiplicand, Positive Multiplier

Example: $-3_{10} \times 6_{10}$

Sign-bit of the multiplicand must be extended to the word length of the final product before Operating on the 2's Complement words.

This sign-extended multiplicand is used when forming Partial products and accumulated sums.

The result of the multiplication is the 2's Complement of the Product. The final magnitude is found by taking 2's Complement.

Bit Assignment: We assign 8 bits to both numbers. The product will thus be 16 bits.
+3 = 0000 0011
Thus 2's Complement = -3 = 1111 1101
+6 = 0000 0110

1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
							x	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	X
1	1	1	1	1	1	1	1	1	1	1	1	0	1	X	X
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0

Remember: Sign Extension to maximum number of bits in datapath

Answer = (1111 1111 1111 10 1110)
Take 2's Complement, Answer = -(010010) = -18_{10}

Multiplication of Fractions

Convert from decimal to binary $(\frac{3}{4})$

= 0.75

0.75 x 2 = 1.5, keep 1

0.5 x 2 = 1.0, keep 1

0 x 2 = 0 keep 0

And only zeros afterwards

= $2^{-1} + 2^{-2} + 0 + 0$

= 0.1100; assigning four fractional bits

2's Complement of Binary Fractional Nos.

- **Given binary fractional number = (0.1100)**

- **Method 1:**
- **Decide on the number of total bits, eg. 5 bits; Invert all bits; add +1 to LSB**

- **2's Complement = 1.0011**
- **+ 1**
- **_____**
- **= 1.0100**
- **Method 2:**
- **Look from right to left; when you Encounter first 1; invert all bits to the left**

- **For (0.1100), the 2's Complement = (1.0100)**

Question?

- Represent 9/16 using five fractional bits:

- Hint: 9/16 = 0.xxxxx
- Keep multiplying by 2; if answer is greater than 1, keep 1

- Solve: 0.562 x 2 = 1.125, keep 1
- 0.125 x 2 = 0.25, keep 0
- 0.25 x 2 = 0.5, keep 0
- 0.5 x 2 = 1.0, keep 1
- 0 x 2 = 0, keep 0, and same for more terms
- Answer = 0.10010 in binary

Finally, 2's Complement of "0.10010" is (look right to left) "1.01110" that is (-9/16)

Convert Fraction Number to Binary

- Represent 9/16 using five fractional bits:
- Hint: 9/16 = 0.5625
- Keep multiplying by 2; if answer is greater than 1, keep 1

- Solve: 0.5625 x 2 = **1**.125, keep 1
- 0.125 x 2 = **0**.25, keep 0
- 0.25 x 2 = **0**.5, keep 0
- 0.5 x 2 = **1**.0, keep 1
- 0 x 2 = **0**, keep 0, and same for more terms
- Answer = 0.10010 in binary

Finally, 2's Complement of "0.10010" is (look right to left) "1.01110" that is (-9/16)

Multiplication of Signed Fractions

- Fractions are multiplied like whole numbers, but overflow is not possible
- A 4-bit fractional number is represented as minimum 5-bit fixed point number with MSB holding the sign bit in 2's Complement format
- The product of two 5-bit numbers will produce 10-bit result
- MSB will be sign-extended (bit replication) for negative multiplicand

Example 1: Positive multiplicand, positive multiplier, fraction multiplication

Show binary multiplication of $(3/4)_{10}$ x $(1/2)_{10}$
Use 5-bits to represent each number

3/4 = 0.1100

1/2 = 0.1000

					0.	**1**	**1**	**0**	**0**	Positive multiplicand
				x	**0.**	**1**	**0**	**0**	**0**	Positive multiplier
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	X	0
0	0	0	0	0	0	0	0	X	X	0
0	0	0	1	1	0	0	X	X	X	1
0	0.	0	1	1	0	0	0	0	0	

Check Multiplier bits one at a time

Insert decimal, count how many bits after decimal in both numbers (4 + 4 = 8 bits in both numbers)

Answer = (0.0110000000) = +(3/8)

Example 2: Negative multiplicand, positive multiplier, fraction multiplication

Show binary multiplication of $(-3/4)_{10}$ x $(3/8)_{10}$
Use 5-bits to represent each number

3/4 = 0.1100
-3/4 = -(0.1100)
= 1.0100

3/8 = 0.0110

Sign bit replication

		1	**1**	**1**	**1**	**1**	**1.**	**0**	**1**	**0**	**0**
						x	**0.**	**0**	**1**	**1**	**0**
	1	1	1	1	1	1	0	1	0	0	X
1	1	1	1	1	1	0	1	0	0	X	X
Extra	1	1.	1	0	1	1	1	0	0	0	

Negative multiplicand

Negative multiplier

(first 0 only shift, then 1, Add multiplicand)

(next 1, Add Multiplicand, then 0 shifts only)

Insert decimal, count how many bits after decimal in both numbers (4 + 4 = 8 bits in both numbers)

Answer = (11.10111000) , take 2's complement = -(0.01001000) = (-9/32)

Example 3: Positive multiplicand, negative multiplier, fraction multiplication

Show binary multiplication of $(3/4)_{10}$ x $(-3/8)_{10}$
Use 5-bits to represent each number

3/4 = 0.1100

3/8 = 0.0110
-3/8 = -(0.0110)
= (1.1010)

				0.	**1**	**1**	**0**	**0**	
				x	**1.**	**1**	**0**	**1**	**0**
				0	0	0	0	0	
			0	1	1	0	0	X	
		0	0	0	0	0	X	X	
	0	1	1	0	0	X	X	X	
1	0	1	0	0	X	X	X	X	
1.	1	0	1	1	1	0	0	0	

Positive multiplicand

Negative multiplier

Check Multiplier bits one at a time

2's Complement of Multiplicand due to 1 MSB

Insert decimal, count how many bits after decimal in both numbers (4 + 4 = 8 bits in both numbers)

Answer = (1.10111000), take 2's Complement = -(0.01001000) = -(9/32)

Example 4: Negative multiplicand, negative multiplier, fraction multiplication

Show binary multiplication of $(-3/4)_{10}$ x $(-3/8)_{10}$
Use 5-bits to represent each number

3/4 = 0.1100
-3/4 = -(0.1100)
= 1.0100

3/8 = 0.0110
-3/8 = -(0.0110)
= 1.1010

Sign bit replication

					1	1	1	1	1.	0	1	0	0	Negative multiplicand
								x	1.	1	0	1	0	Negative multiplier
				Extra ←	0	0	0	0	0	0	0	0	0	0
				1	1	1	1	1	1	0	1	0	0	X
			1	0	0	0	0	0	0	0	0	0	X	X
			1	1	1	1	1	1	0	1	0	0	X	X	X
			0	0	0	0	0	1	1	0	0	X	X	X	X
		1	0	1	0	0	0.	0	1	0	0	1	0	0	0

2's Complement of Multiplicand due to 1 MSB

Insert decimal, count how many bits after decimal in both numbers (4 + 4 = 8 bits in both numbers)

Answer = $+(0.01001000) = 2^{-2} + 2^{-5} = 0.25 + 0.03125 = 0.28125_{10}$

To Remember in Signed Multiplication

- When Multiplicand is Negative, do a sign extension to cover the possible bit-width of Answer
- When Multiplier is Negative, there is a final 2's Complement Addition Step corresponding to the MSB of multiplier

Algorithmic Improvement in Multipliers

Booth Encoding

Booth Multiplication Process

Booth Encoded Multipliers

Object: To reduce the number of 'Add' steps required in complete multiplication cycle

MSB '1' shows negative number

2's Complement of $7_{10} = (1\ 0\ 0\ 1)_2$

$1 \times 2^0 = 1$

$0 \times 2^1 = 0$

$0 \times 2^2 = 0$

$-1 \times 2^3 = -8$

Allow both +ive and –ive signs to be used in conversion

Decimal value of $(1001)_2 = (-8+1) = -7 =$ **(1 0 0 1) or (-1 0 0 1)**

Booth's algorithm is valid for both positive and negative numbers in 2's complement format

Booth Recoding of a 2's Complement Number

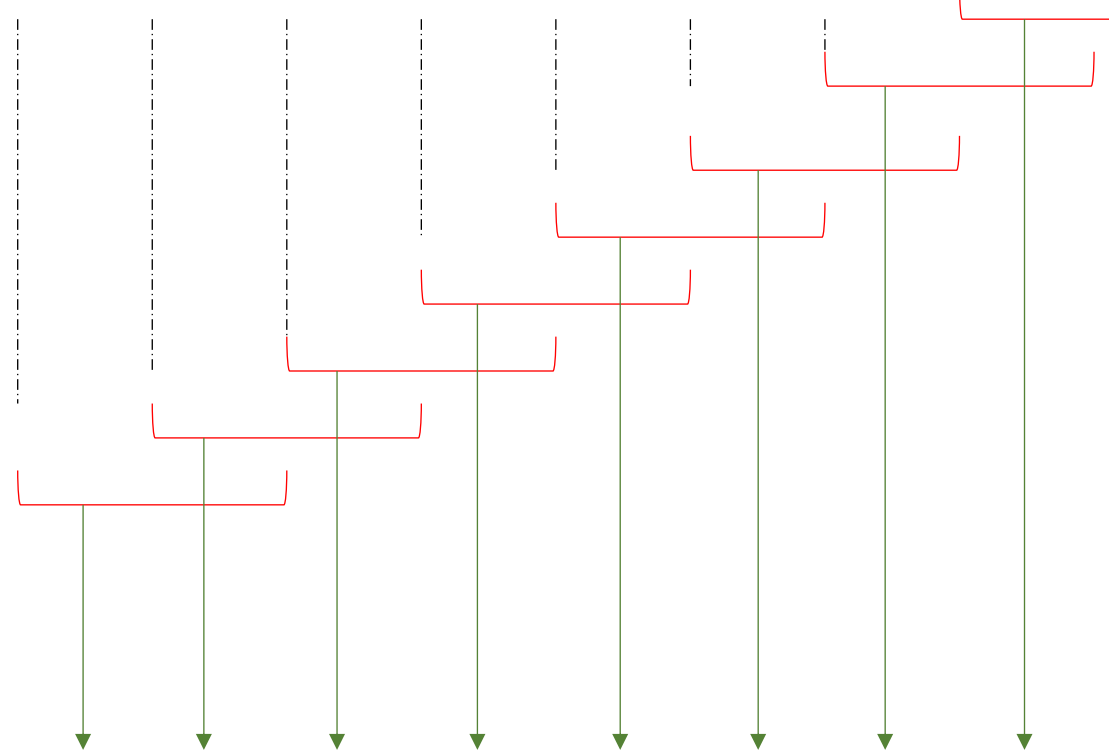
m_i	m_{i-1}	Booth Recoded C_i	Value	Status
0	0	0	0	String of 0s
0	1	1	+1	End of string of 1s
1	0	$\bar{1}$	-1 or $\bar{1}$	Begin string of 1s
1	1	0	0	Midstring of 1s

Booth Recoding of -65_{10}

Append '0' on right, if LSB=1

-65_{10} = | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 2's Complement notation

+65 = (01000001)
2's Complement
-65 = (10111111)



m_i	m_{i-1}	Booth Recoded Ci
0	0	0
0	1	1
1	0	$\underline{1}$
1	1	0

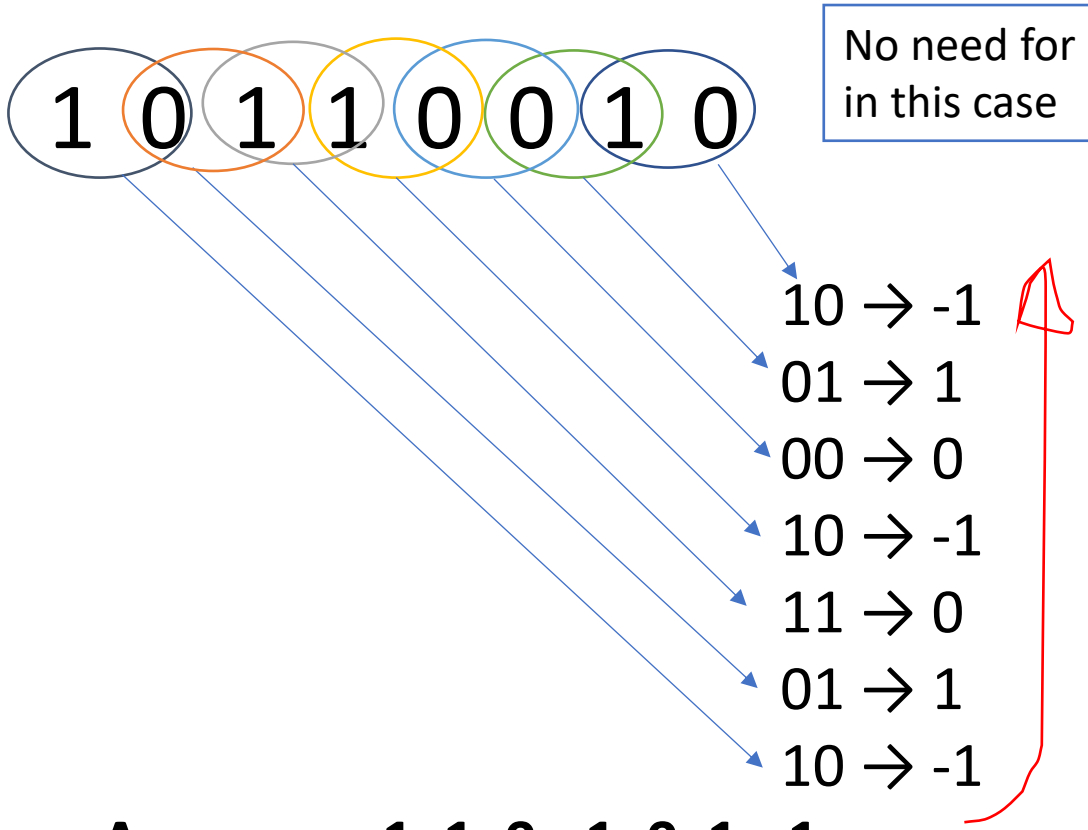
-65_{10} = Or  Booth Recoded notation

Question?

Convert decimal number –78 to Booth Encoded format using 8 binary bits

+78 = 01001110
Take 2's Complement
-78 = 10110010

m_i	m_{i-1}	Booth Recoded Ci
0	0	0
0	1	1
1	0	$\bar{1}$
1	1	0



Answer = -1 1 0 -1 0 1 -1

After Booth Encoding