# Lecture 13

# EE 421 / CS 425

# Digital System Design

Fall 2024

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 Today**

# Decimal Multiplication using Pencil and paper


$$\begin{array}{r} 943 \\ \times\ 18 \\ \hline 7544 \\ +9430 \\ \hline 16974 \end{array}$$

Keep shifting right

Keep shifting left

# Array Multipliers – Parallel and Serial forms

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

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

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

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

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

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | $X_3$ | $X_2$ | $X_1$ | $X_0$ |
| | | | | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| | | | | $X_3Y_0$ | $X_2Y_0$ | $X_1Y_0$ | $X_0Y_0$ |
| | | | $X_3Y_1$ | $X_2Y_1$ | $X_1Y_1$ | $X_0Y_1$ | 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?
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

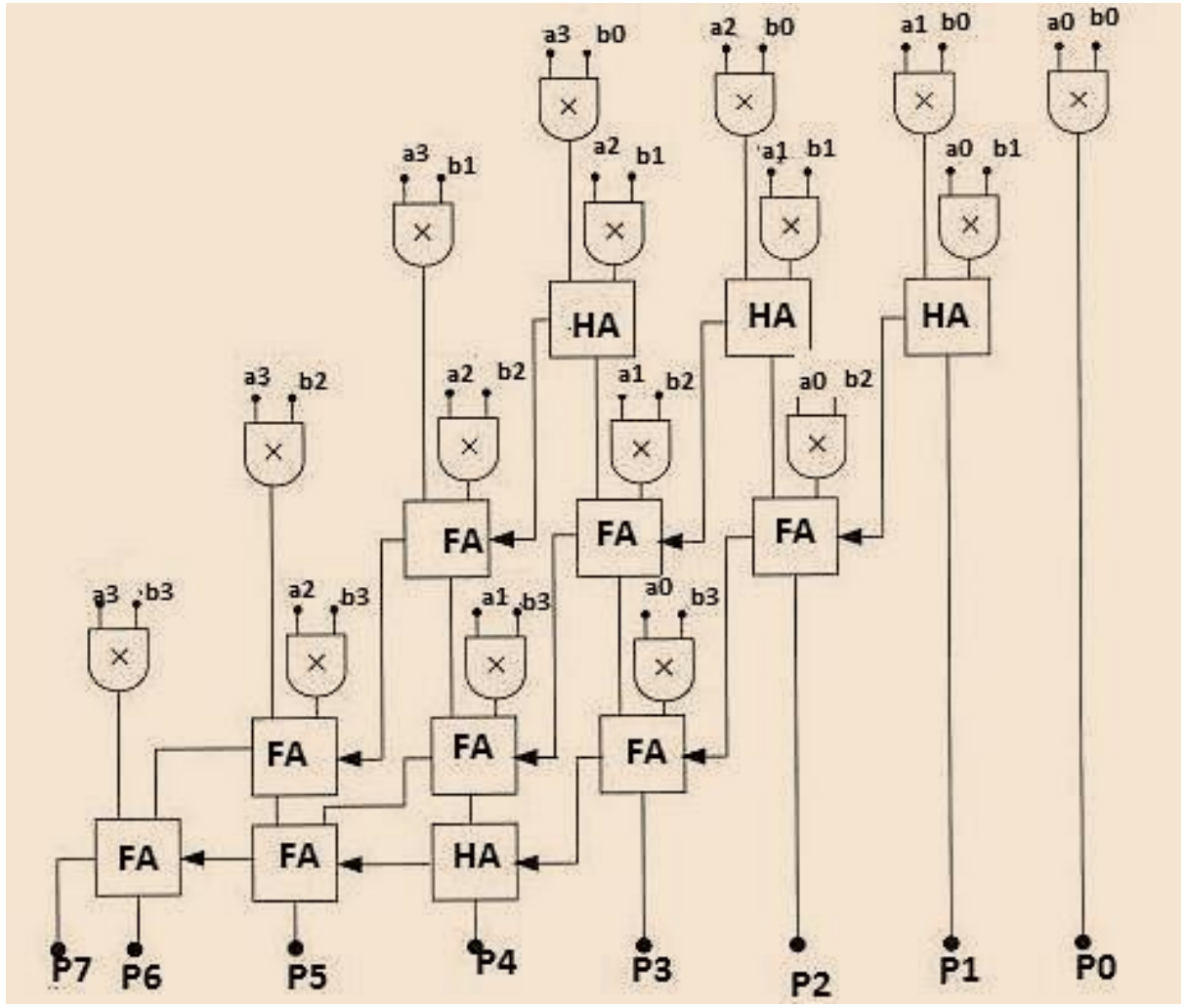
Worst Case Delay is (2n+1) Ƈ
Where Ƈ is the worst adder delay

# 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 | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ |

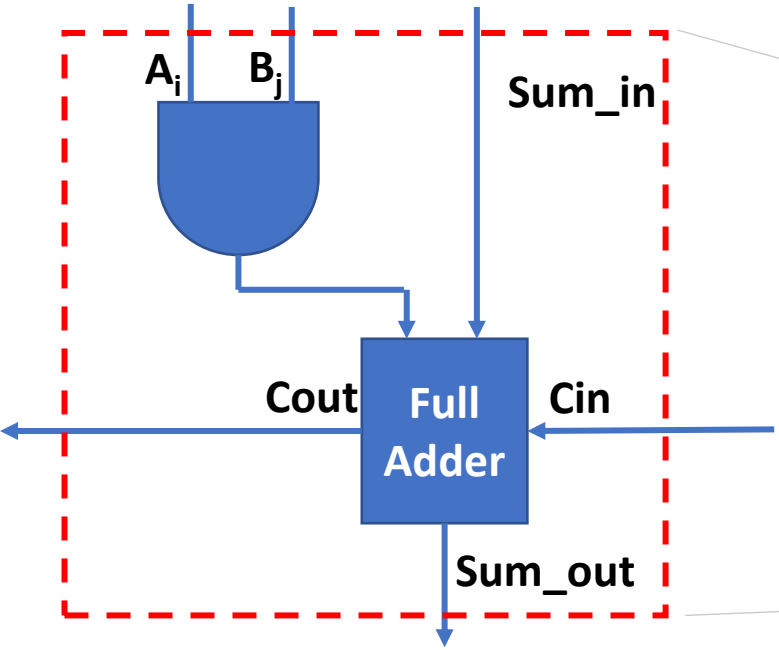
# 4-Bit Array Multiplier connected as AND and ADD

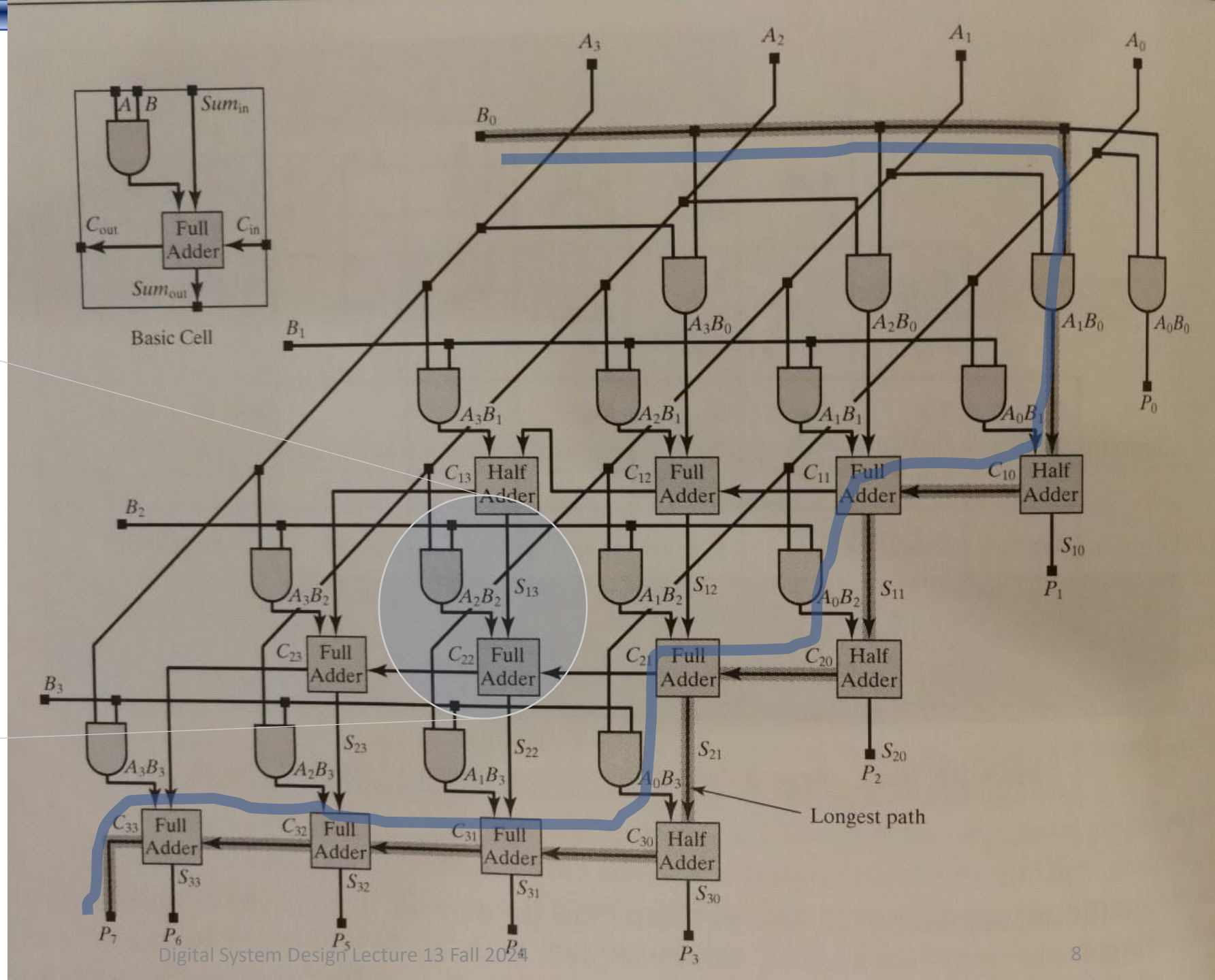
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
| | | | | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| | | | | $A_3B_0$ | $A_2B_0$ | $A_1B_0$ | $A_0B_0$ |
| | | | $A_3B_1$ | $A_2B_1$ | $A_1B_1$ | $A_0B_1$ | 0 |
| | | $A_3B_2$ | $A_2B_2$ | $A_1B_2$ | $A_0B_2$ | 0 | 0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



# Array Multiplier Circuit Delays

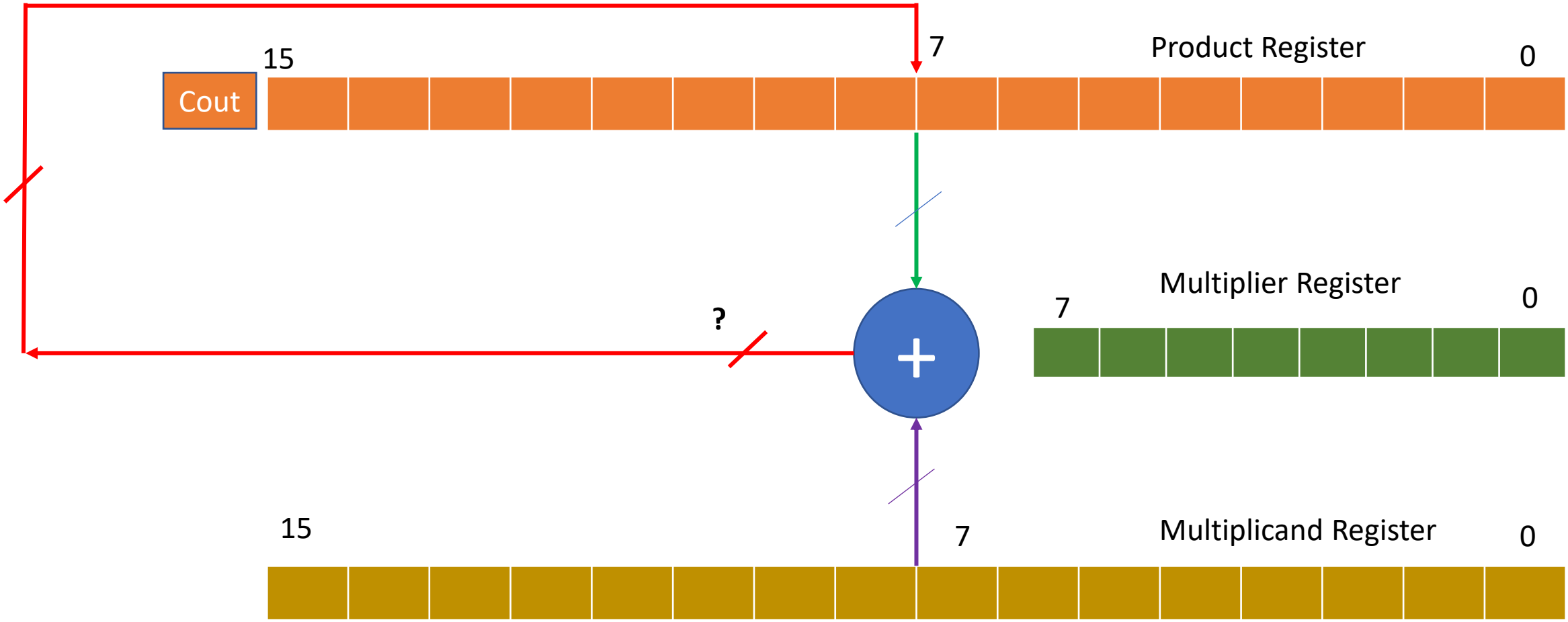
**Building Block**




**Worst case delay path is shaded**
**This is Critical Path**

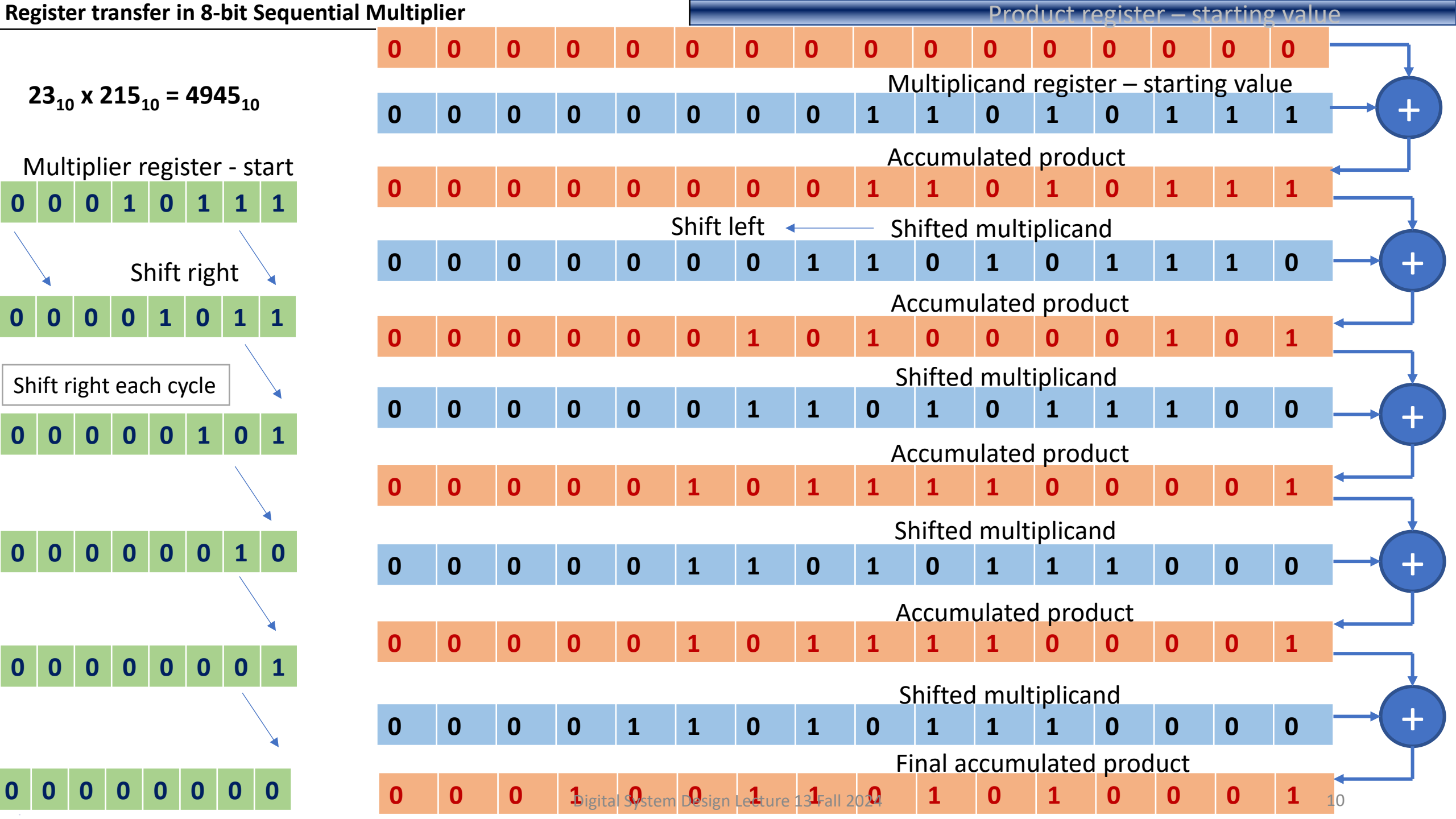
# Operation of Sequential Multiplier



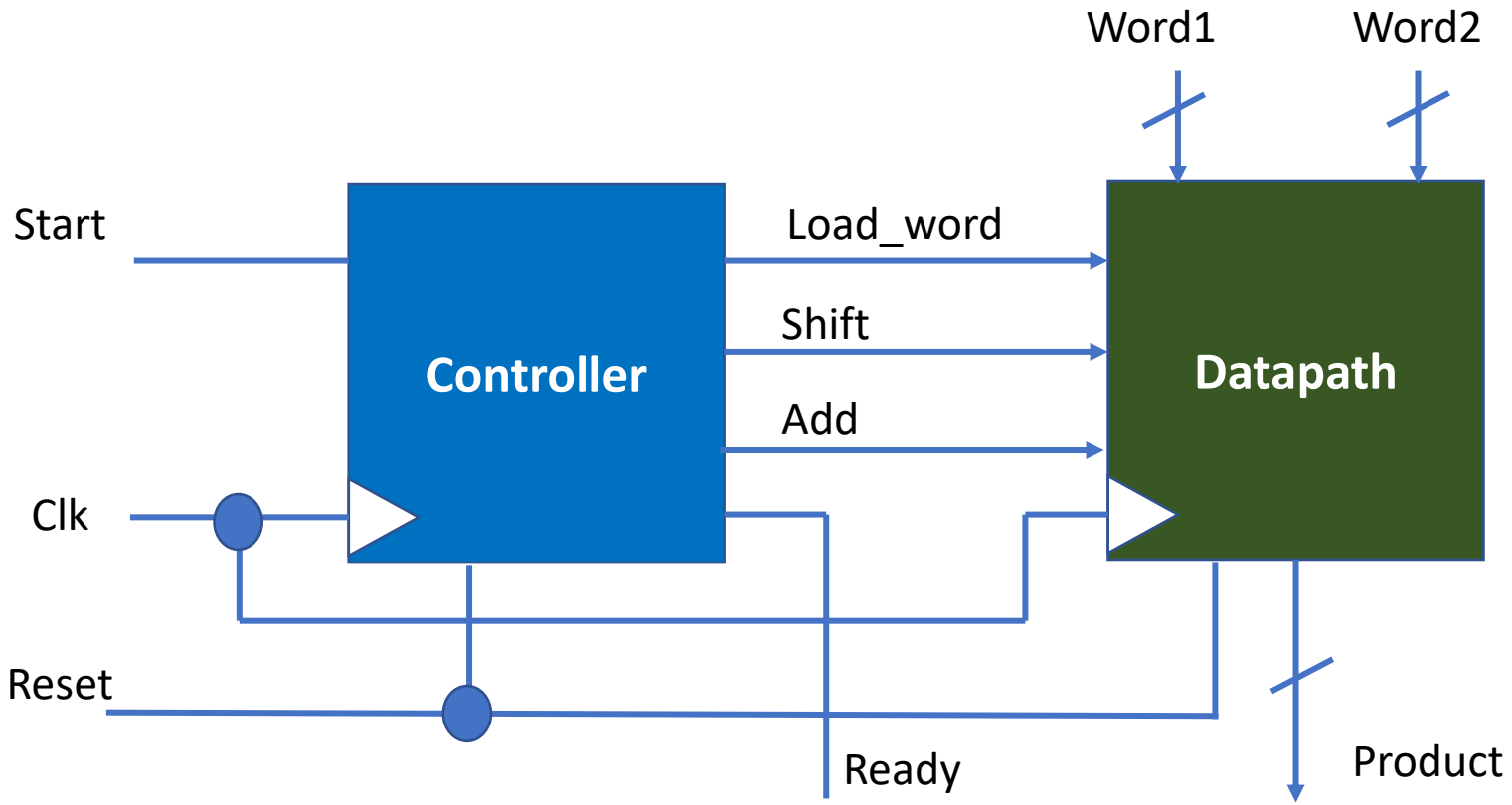

Datapath of a Sequential Multiplier

# Register transfer in 8-bit Sequential Multiplier

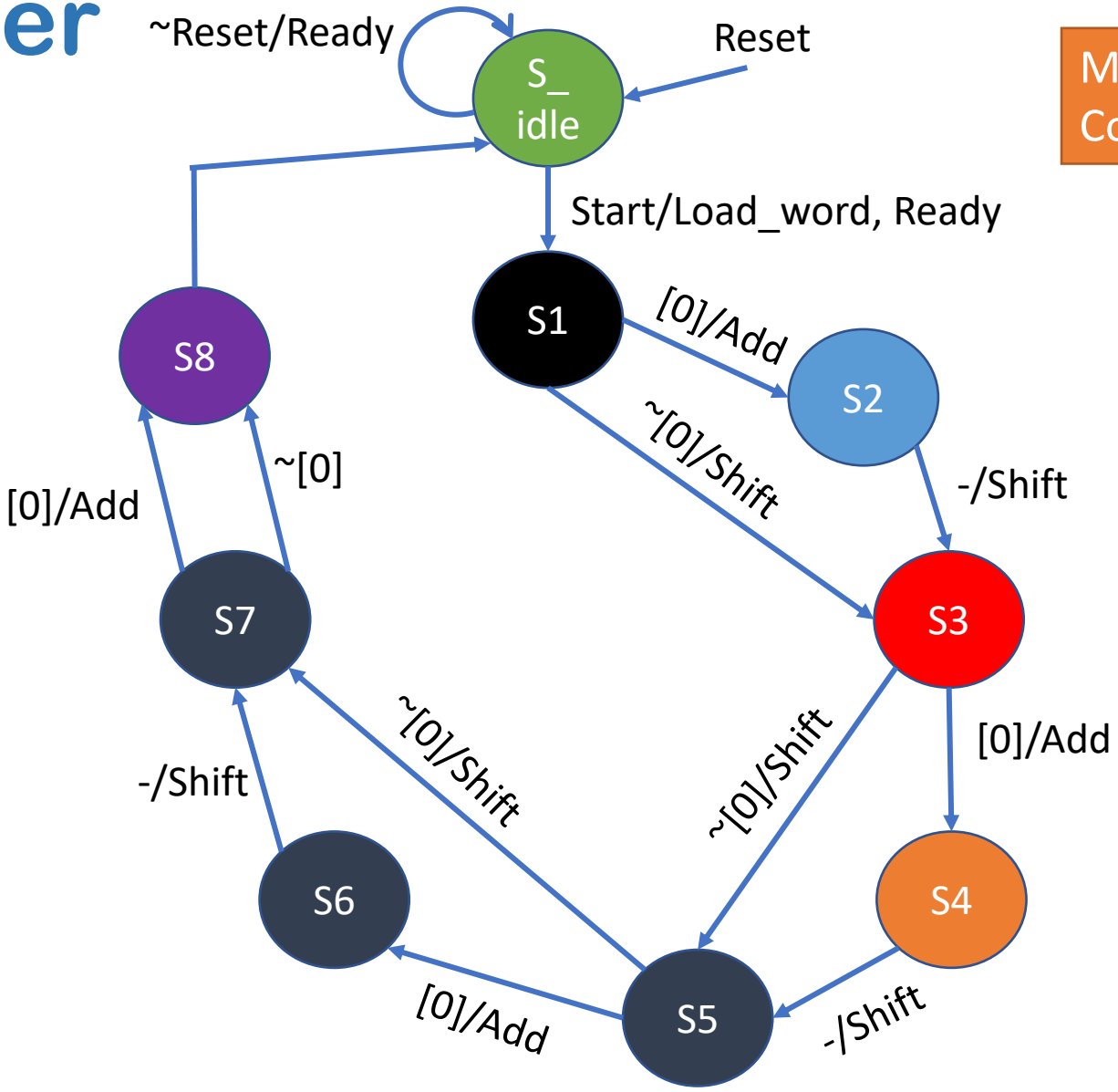
$23_{10} \times 215_{10} = 4945_{10}$



# Data Path Architecture of Sequential Mult

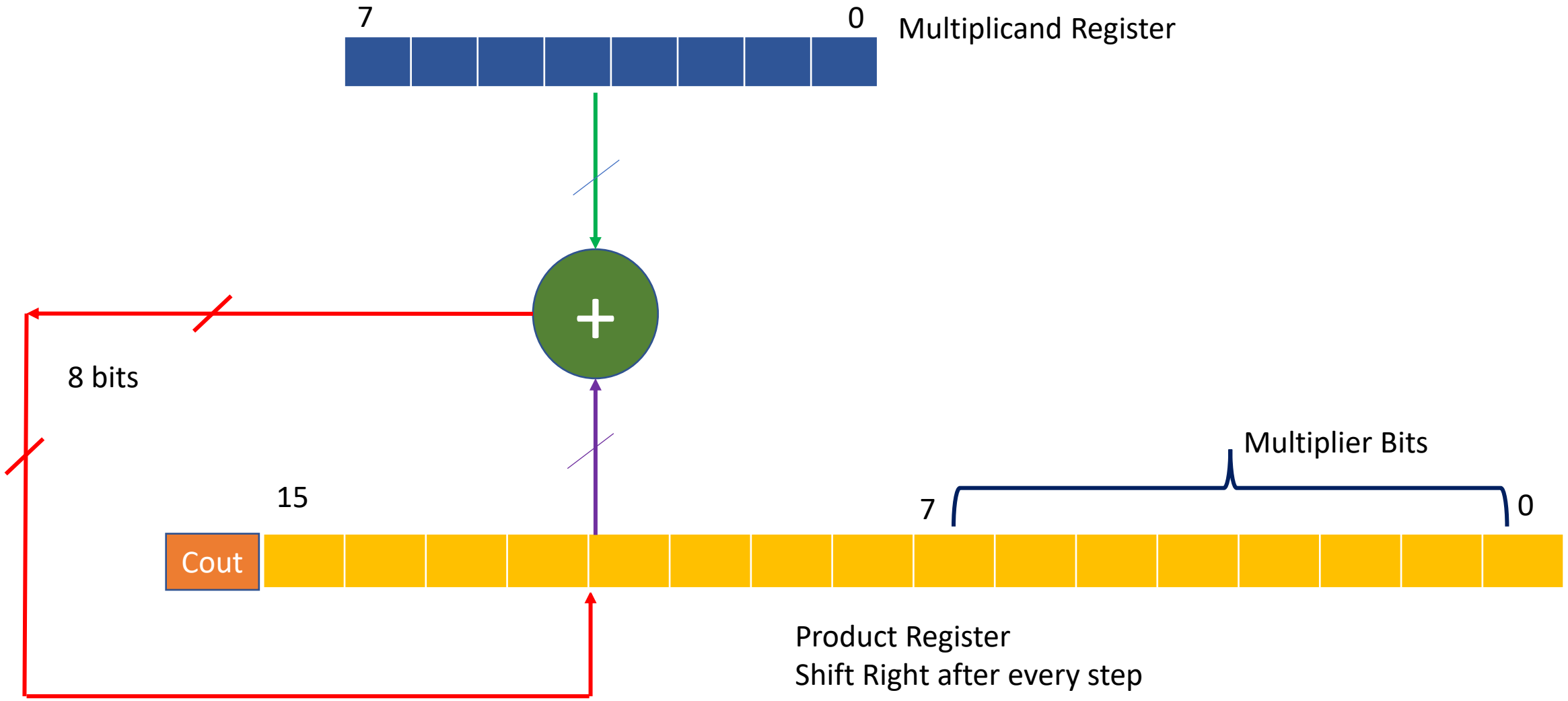


# STG for a 4 Bit Sequential Binary Multiplier

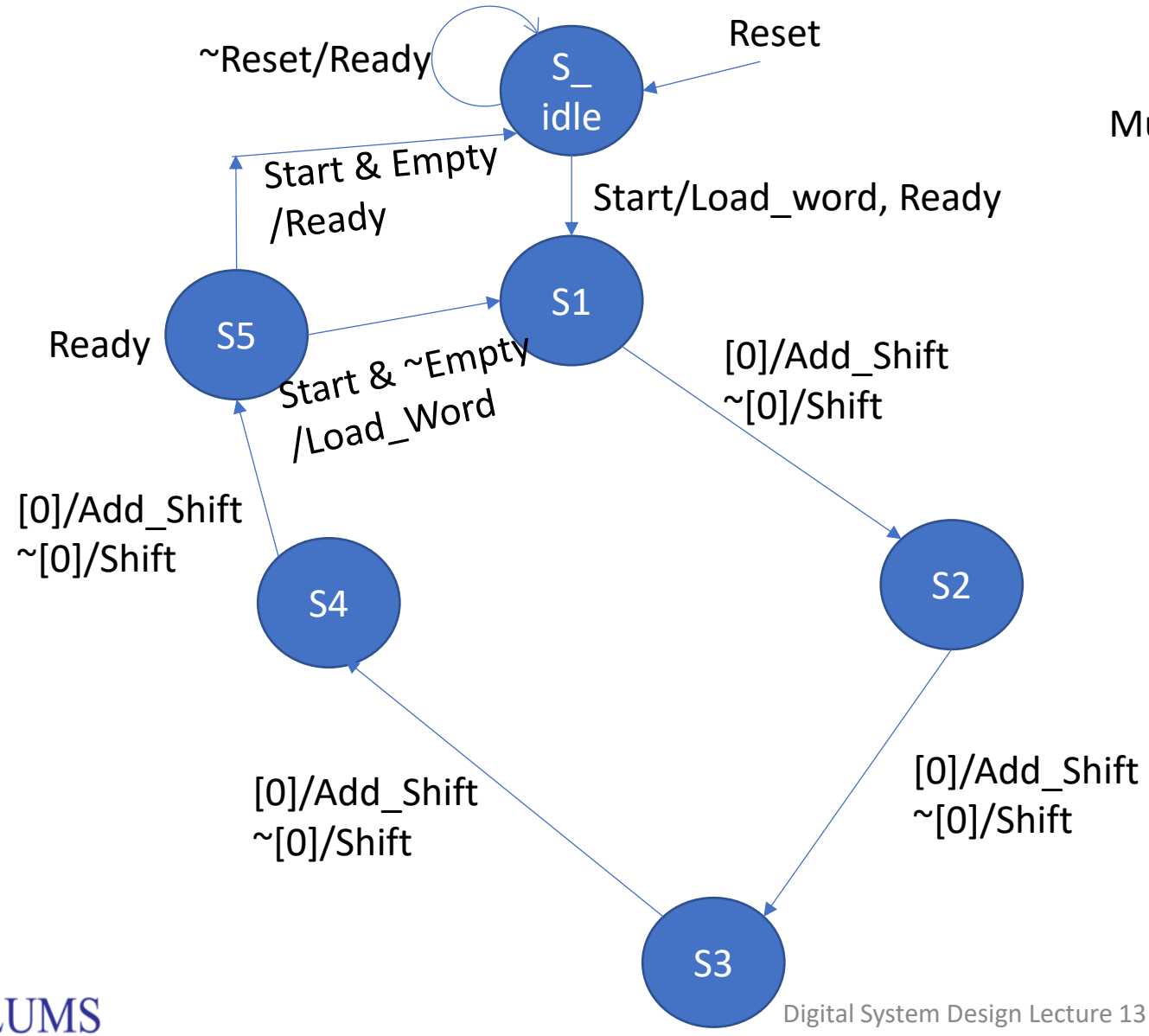


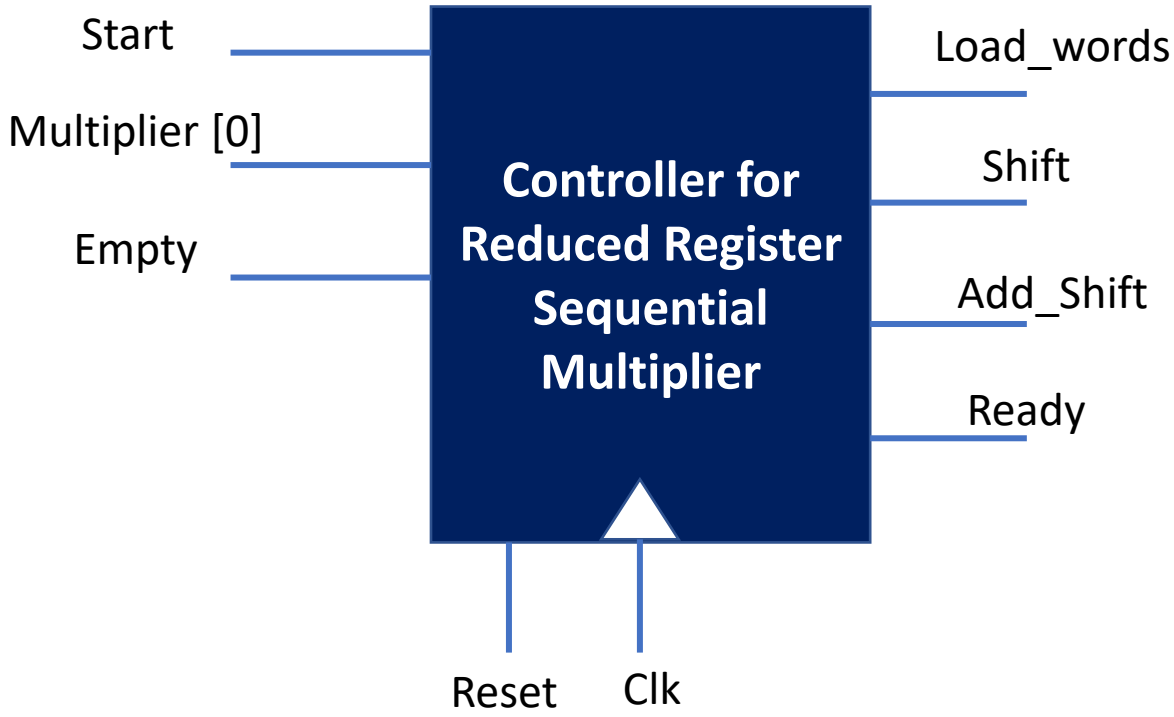

Machine returns to Idle state after Completion of 4 bit multiplication

# Improvement - Sequential Multiplier with Reduced Registers

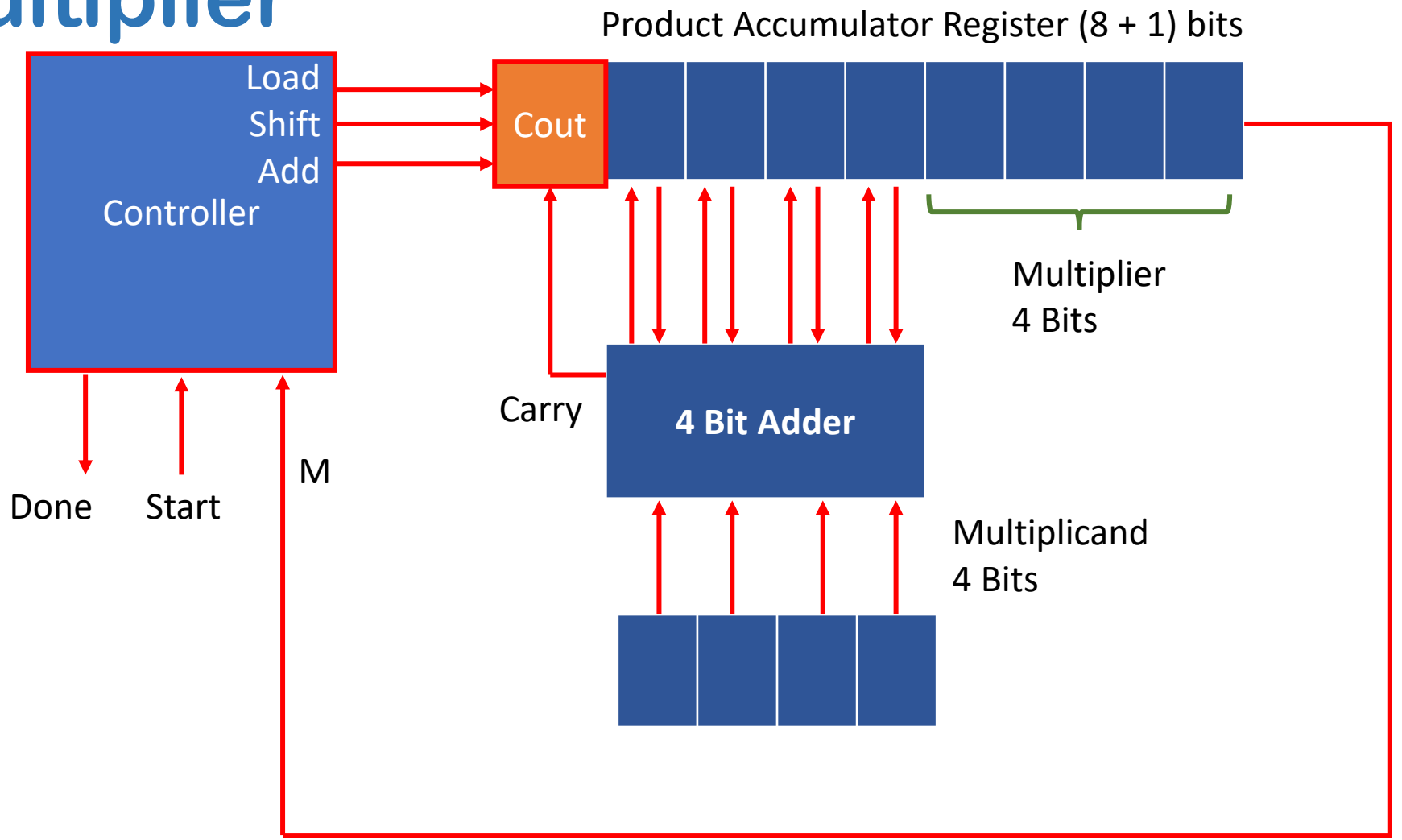


# STG of Reduced Register Sequential Multiplier



# Example of a 4-bit Serial Parallel Multiplier




Shift the contents
To the right after
Every step

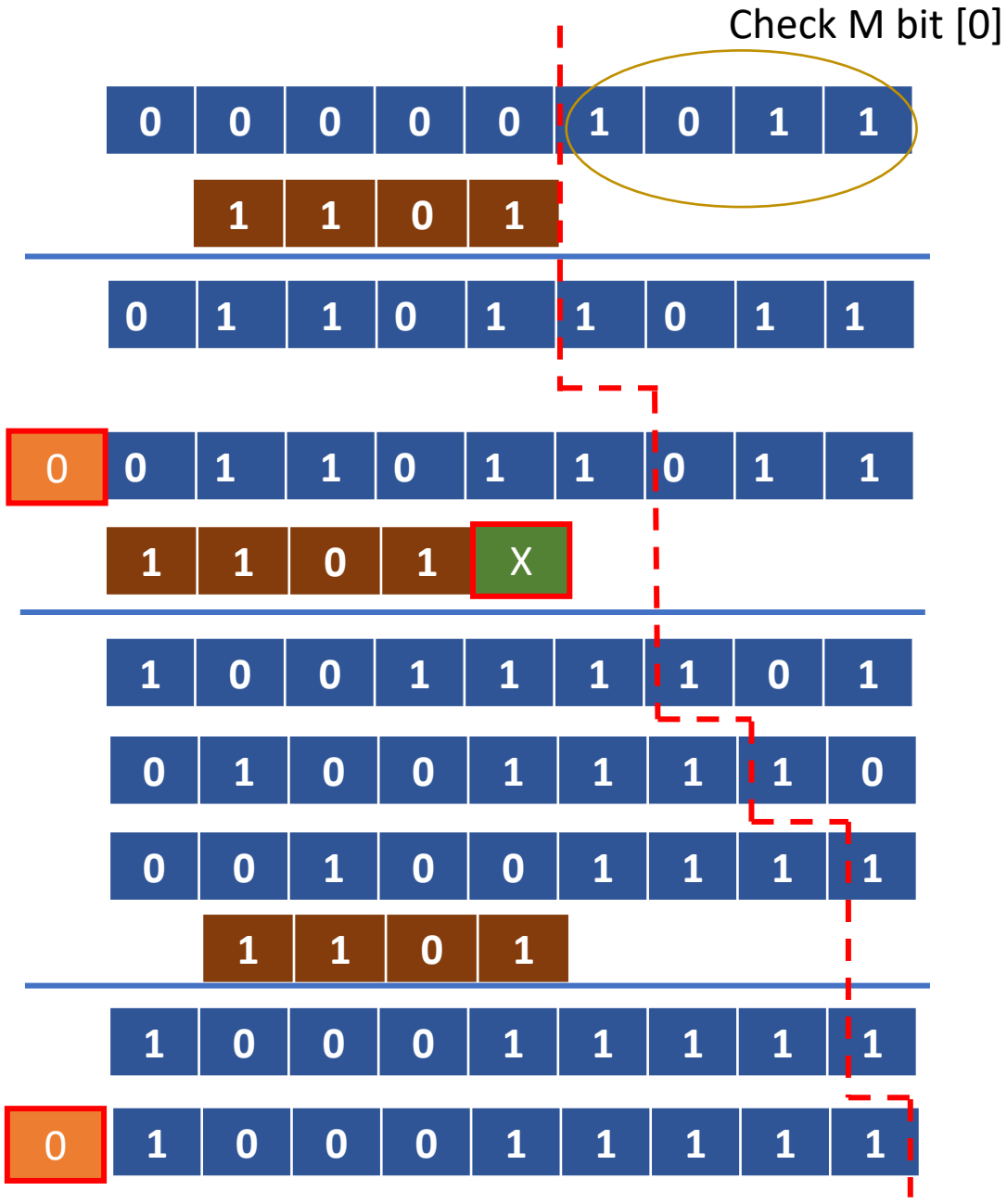
# Example continued

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 |

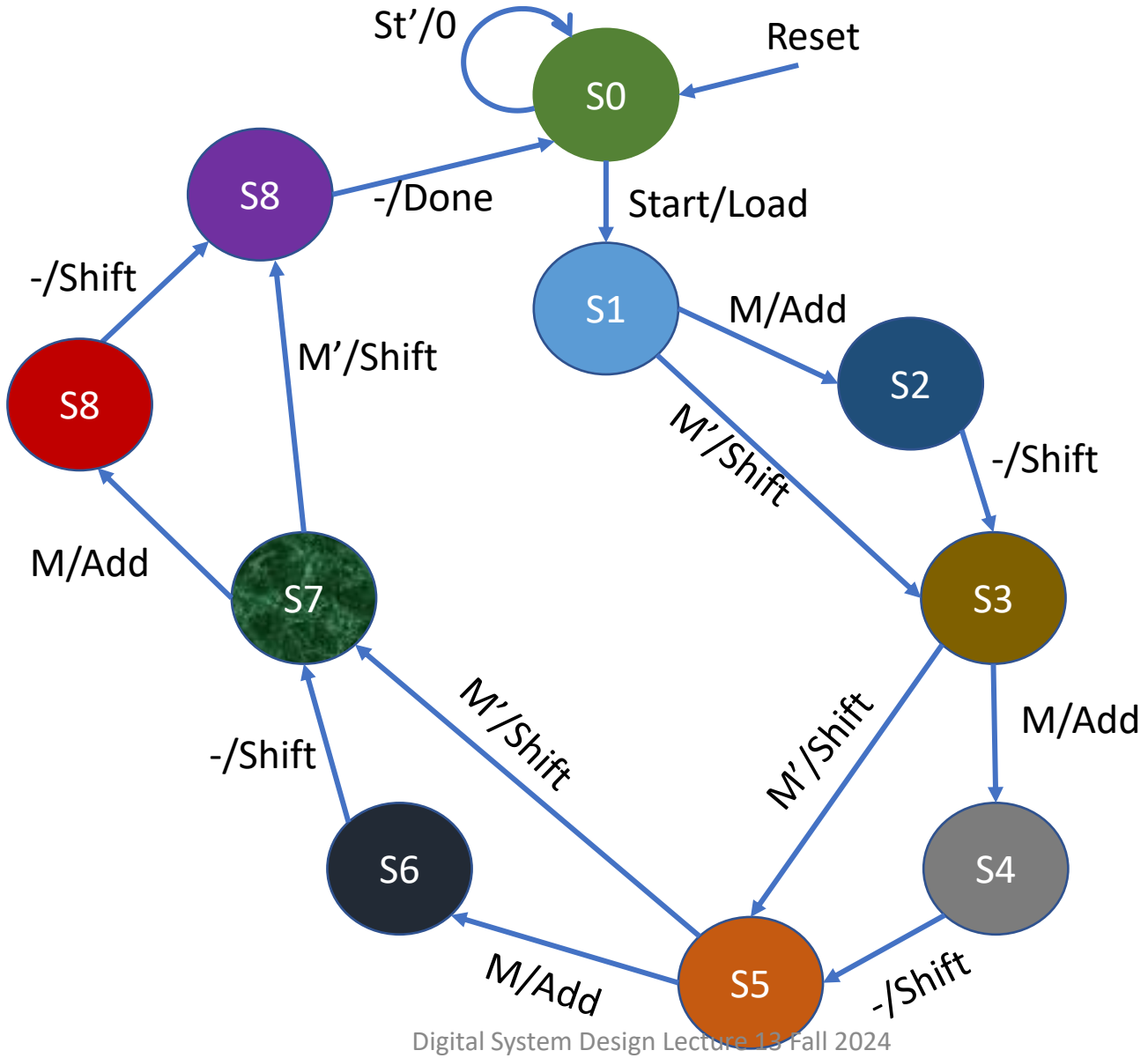
**Answer = $(10001111)_2 = (143)_{10}$**

# 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 |

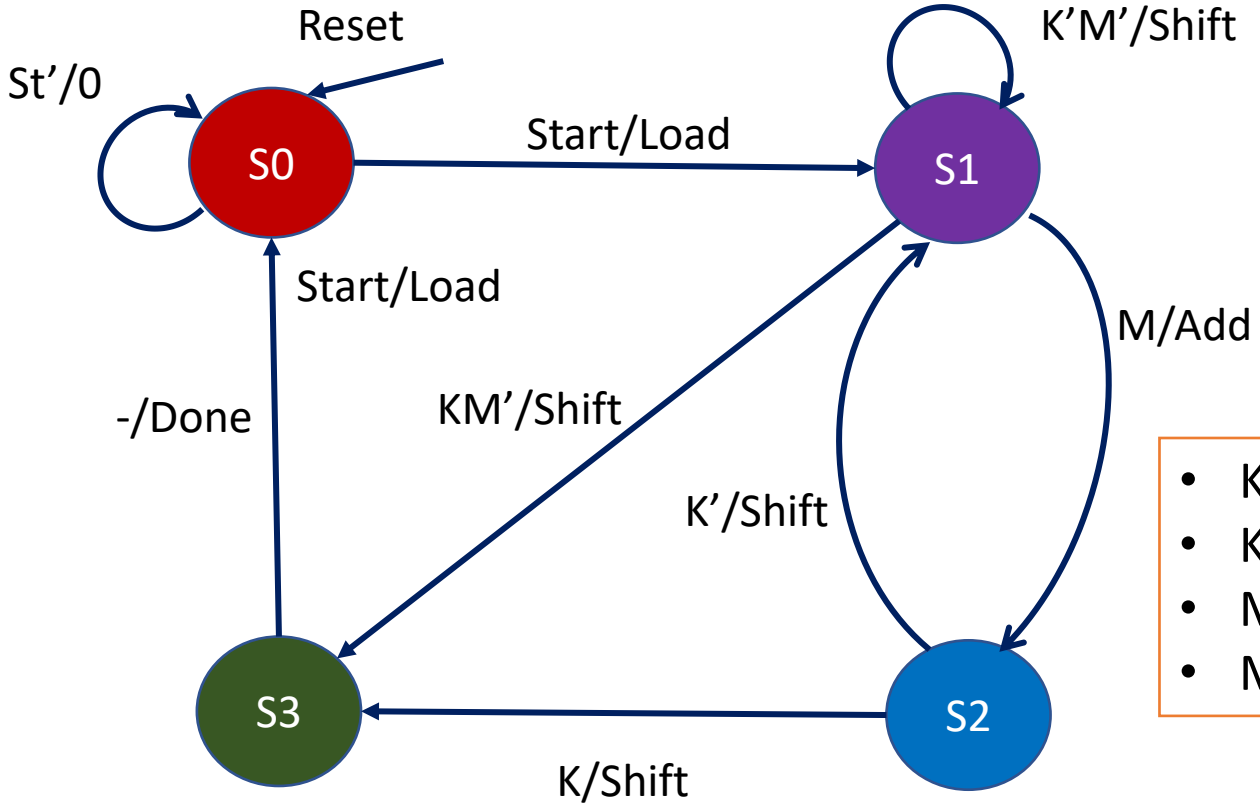


# STG Control Diagram for this Multiplier



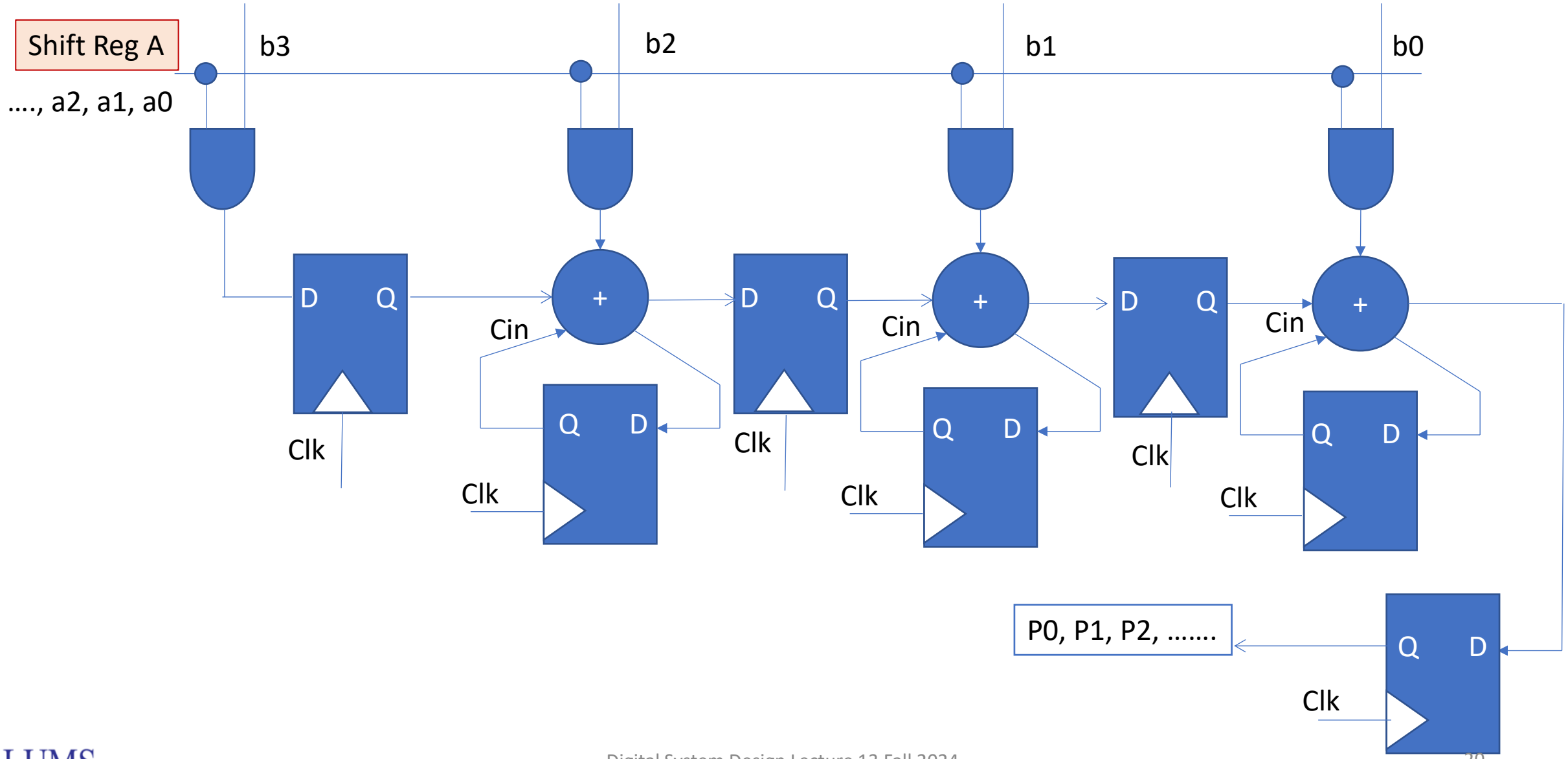

M = relevant bit of multiplier

# 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)**