# Lecture 13
# EE 421 / CS 425
# Digital System Design

**Spring 2023**

**Shahid Masud**

LUMS

# Topics

- Binary Array Multipliers
- 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



```
          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

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

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

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

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

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

| | | | $X_3$ | $X_2$ | $X_1$ | $X_0$ |
|---|---|---|---|---|---|---|
| | | | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
| | | | $X_3 Y_0$ | $X_2 Y_0$ | $X_1 Y_0$ | $X_0 Y_0$ |
| | | $X_3 Y_1$ | $X_2 Y_1$ | $X_1 Y_1$ | $X_0 Y_1$ | Nil |
| | $X_3 Y2$ | $X_2 Y_2$ | $X_1 Y_2$ | $X_0 Y_2$ | Nil | Nil |
| $X_3 Y_3$ | $X_2 Y_3$ | $X_1 Y_3$ | $X_0 Y_3$ | Nil | Nil | Nil |
| Cout | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ |

LUMS

# 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) Ʈ**
**Where Ʈ is the worst adder delay**

LUMS

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

LUMS

# 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 | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ |
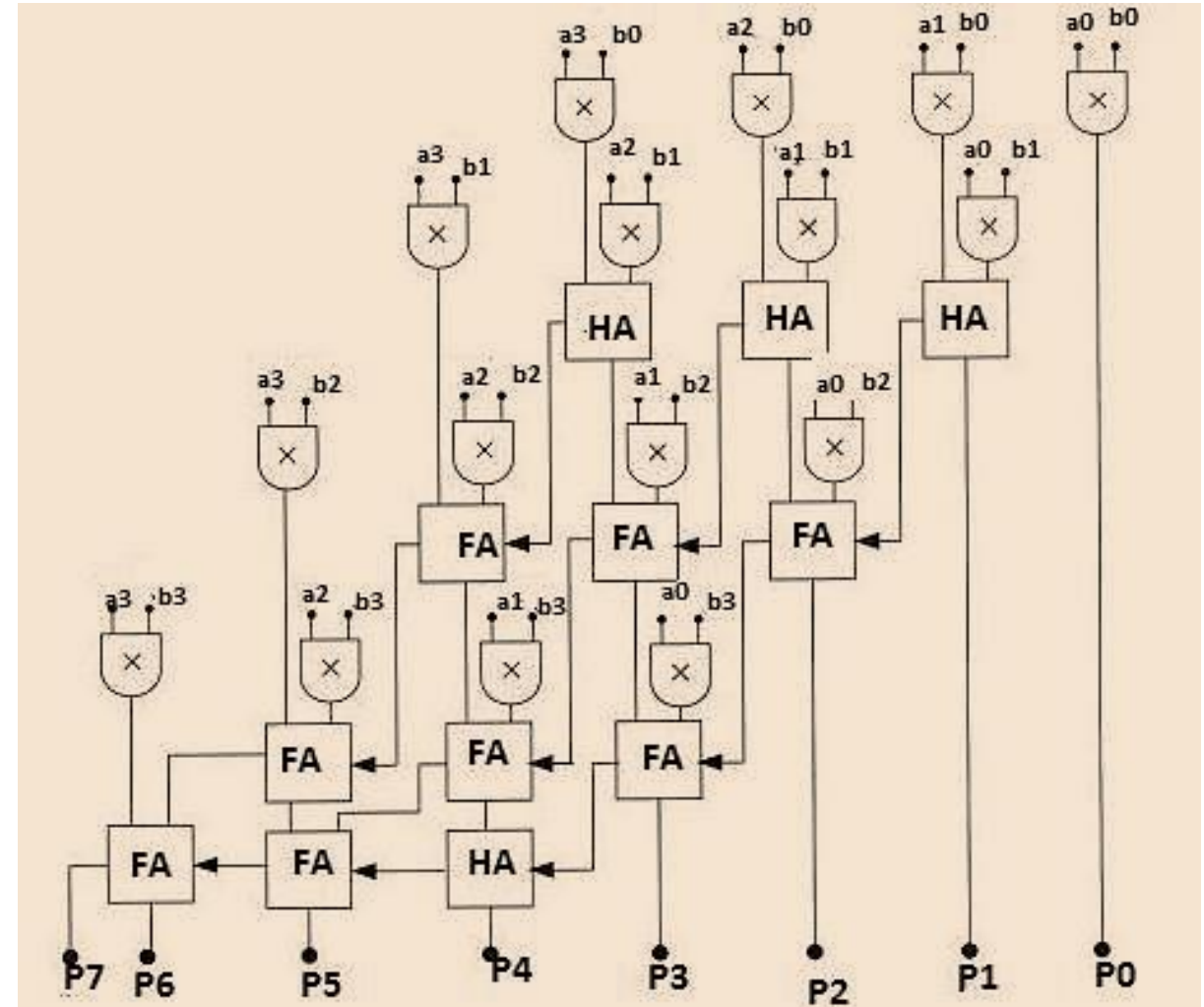
# Array Multiplier Circuit Delays

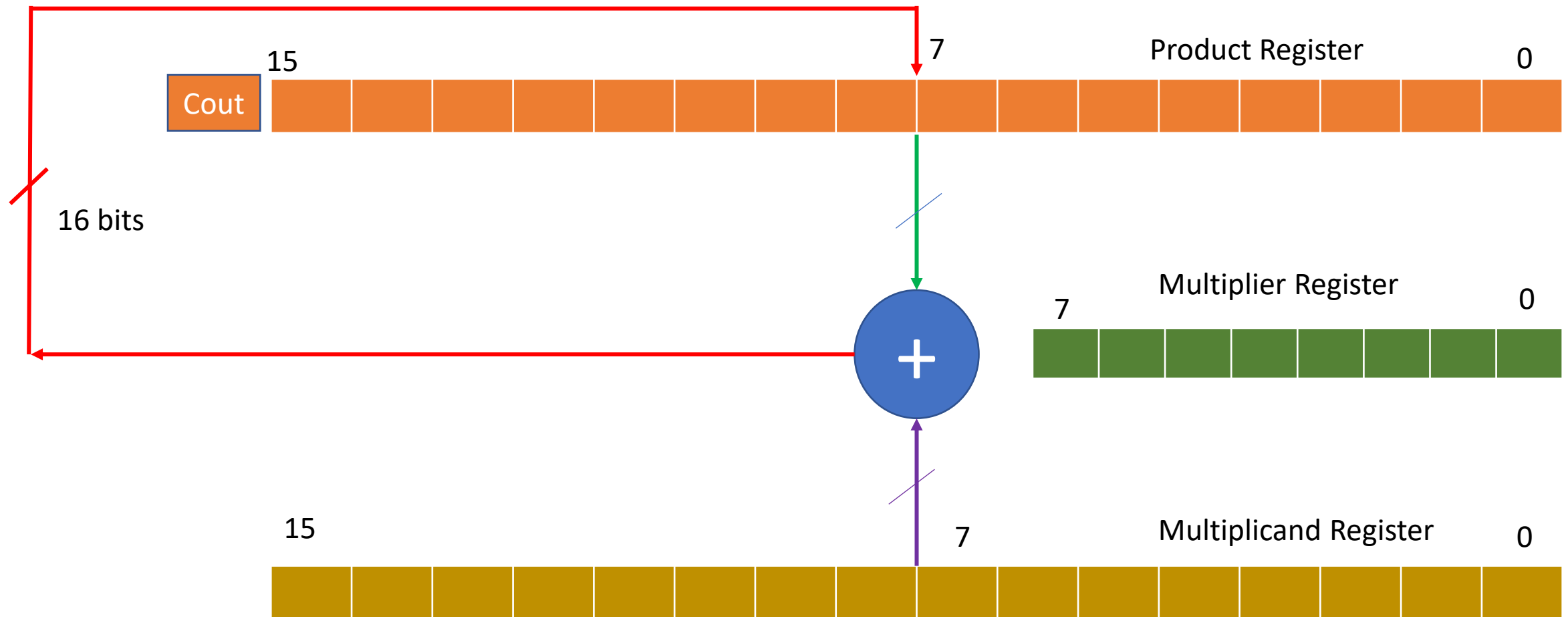**Building Block**



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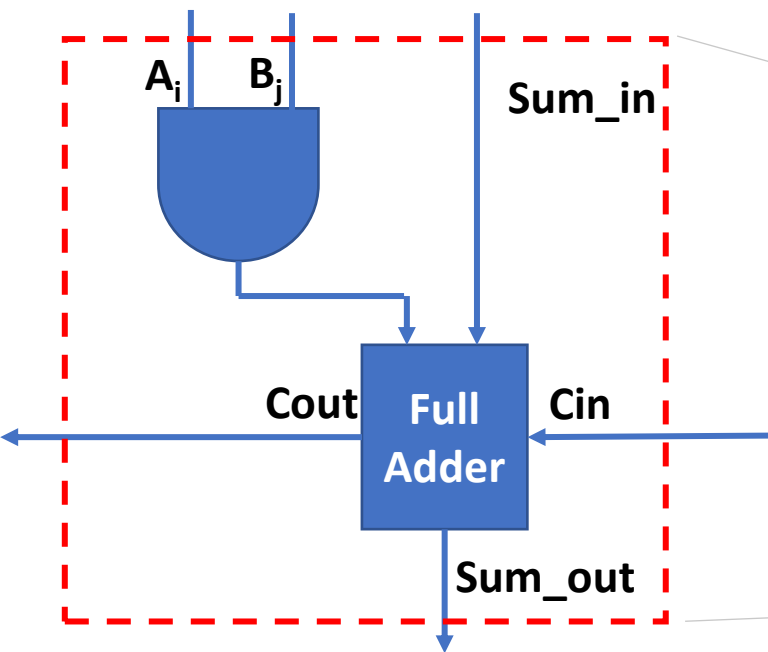
**Worst case delay path is shaded**
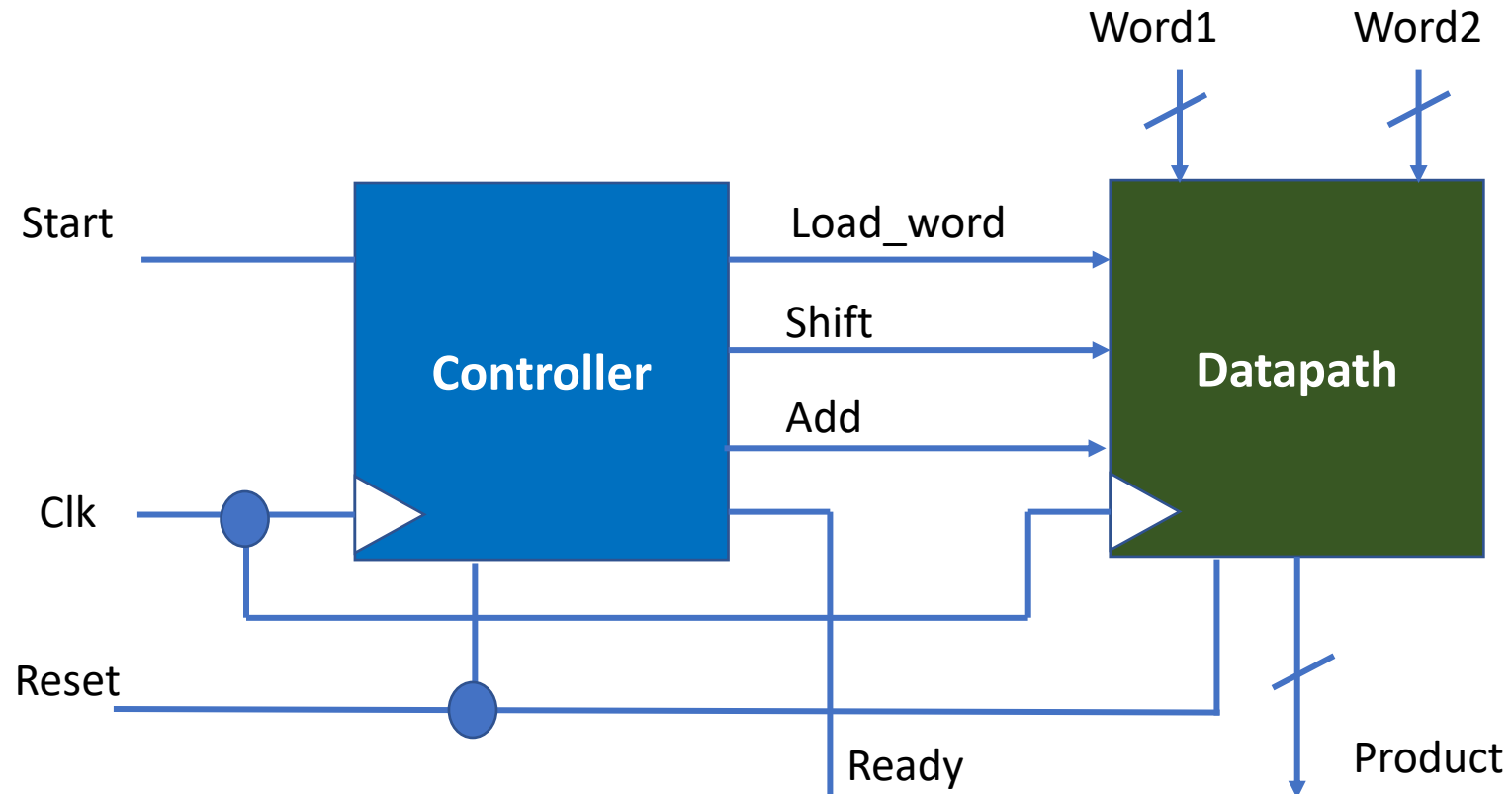**This is** Critical Path

LUMS
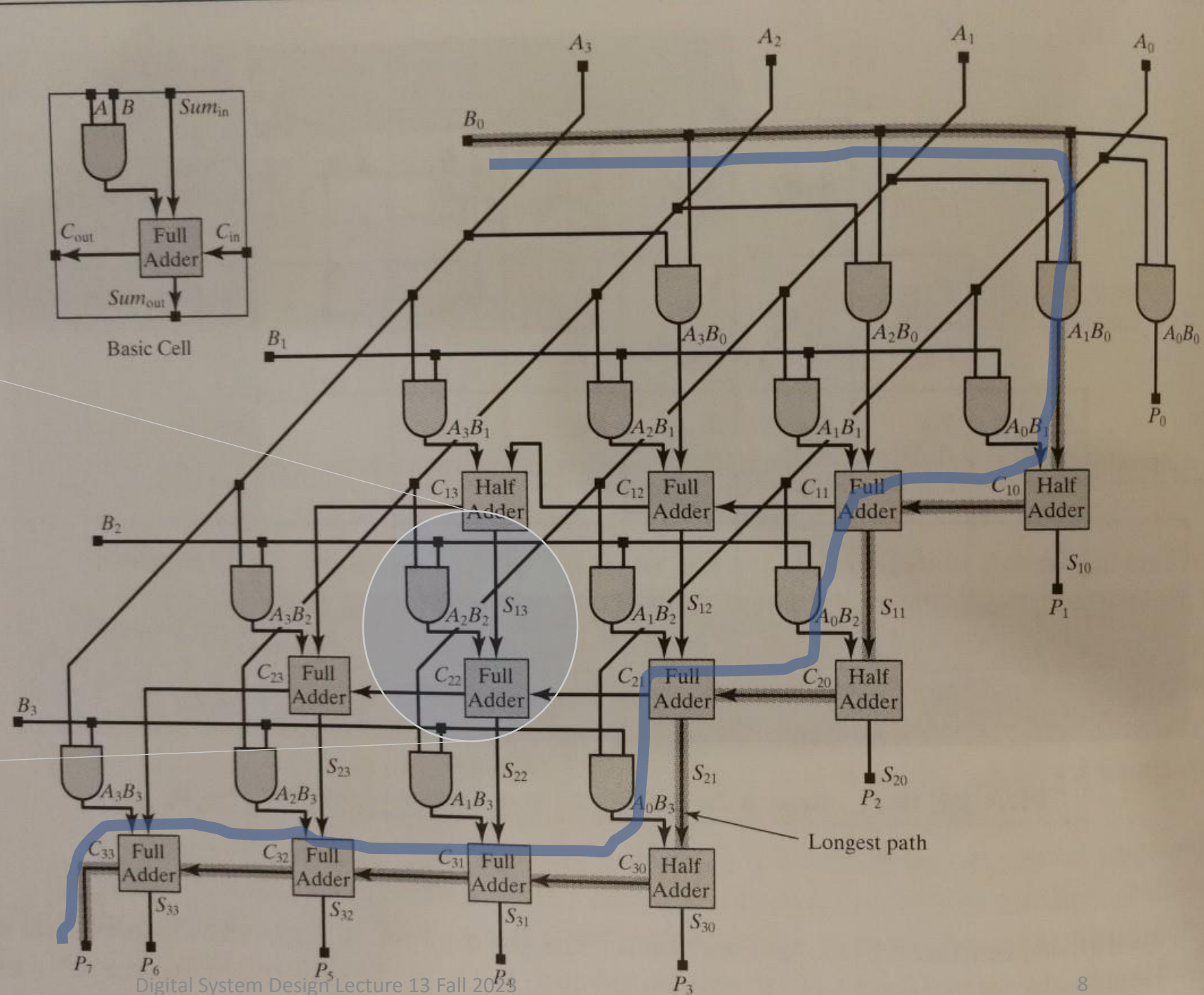
# Operation of Sequential Multiplier



Datapath of a Sequential Multiplier

# Data Path Architecture of Sequential Mult

LUMS

Product register – starting value

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Multiplicand register – starting value

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

Multiplier register - start

| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |

Accumulated product

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

Shift left ← Shifted multiplicand

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

Shift right

| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

Accumulated product

| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

Shift right each cycle

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

Shifted multiplicand

| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

Accumulated product

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Shifted multiplicand

| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Accumulated product

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Shifted multiplicand

| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Final accumulated product

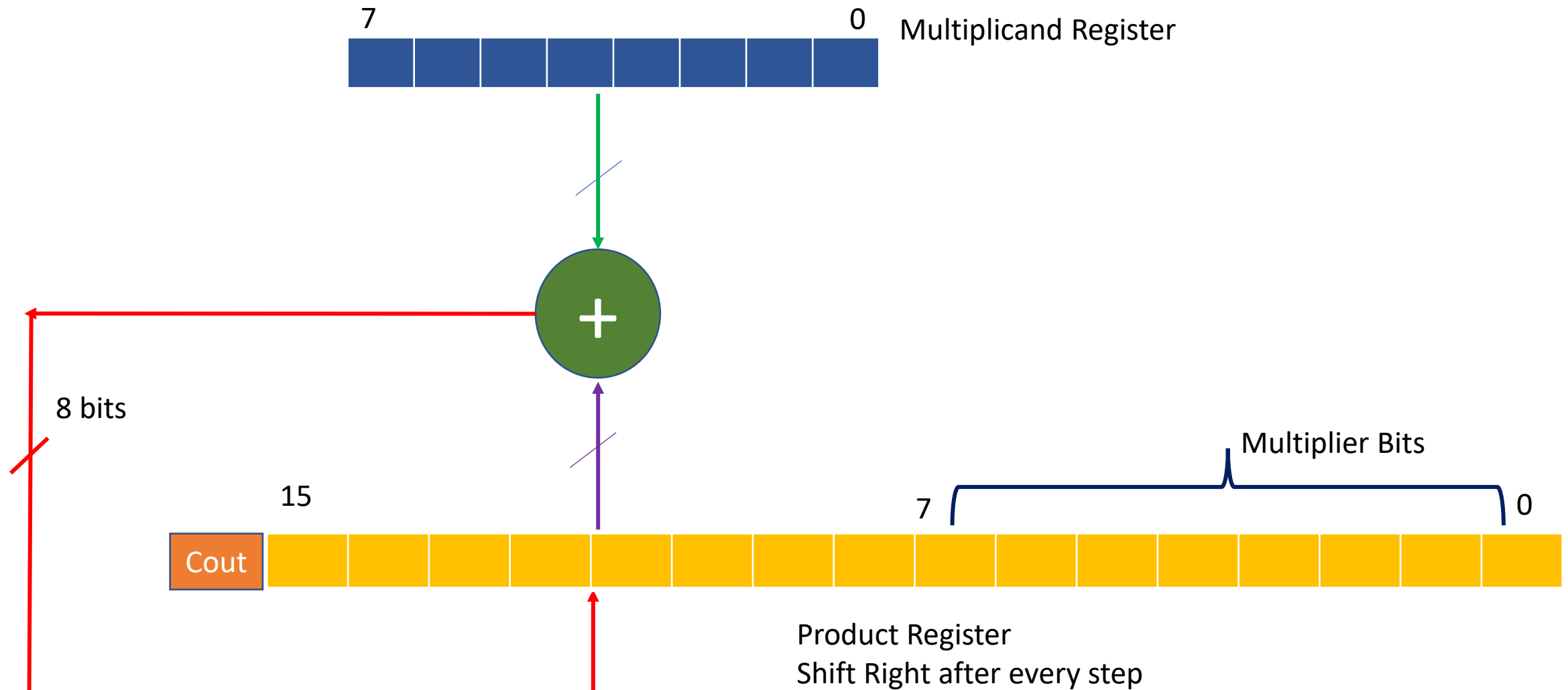| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
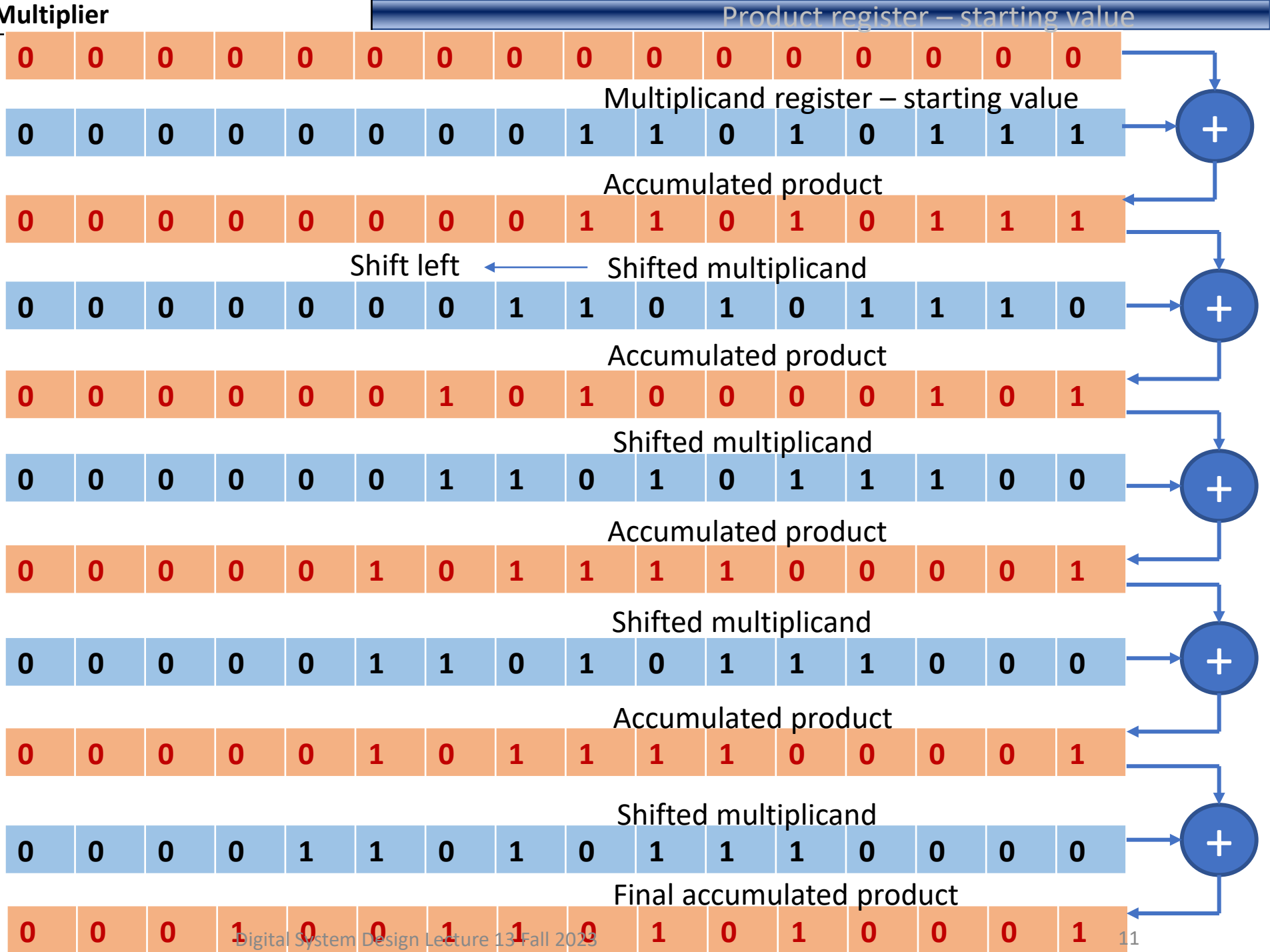
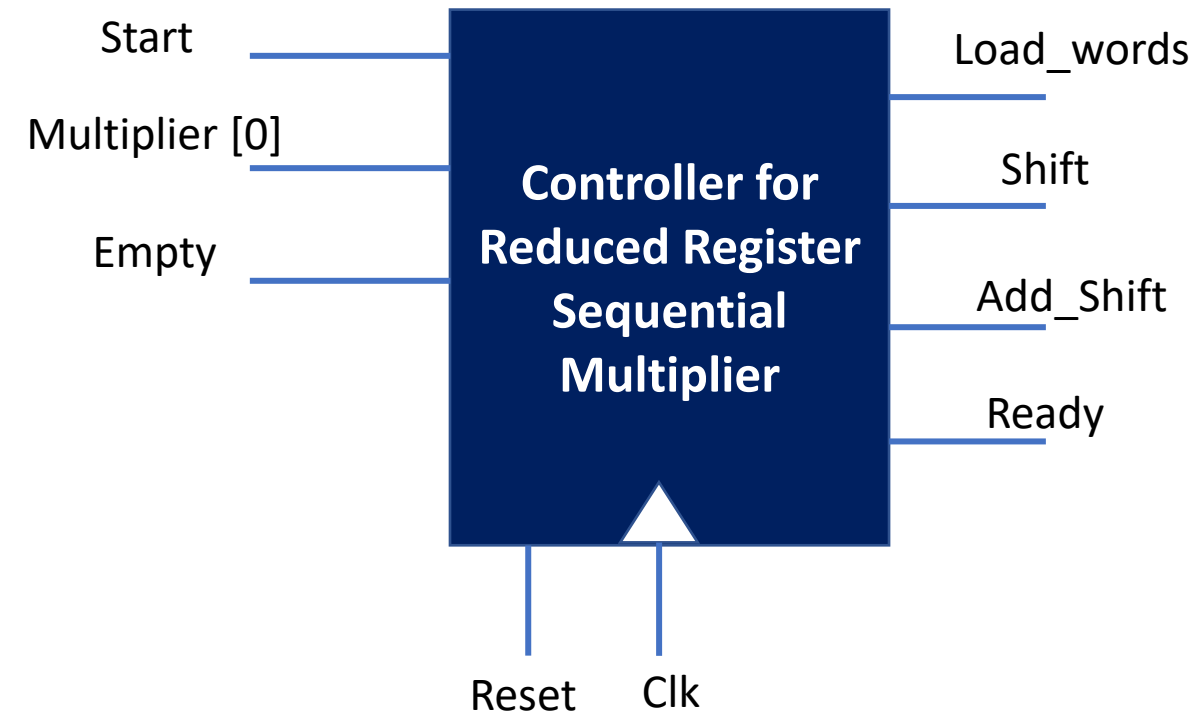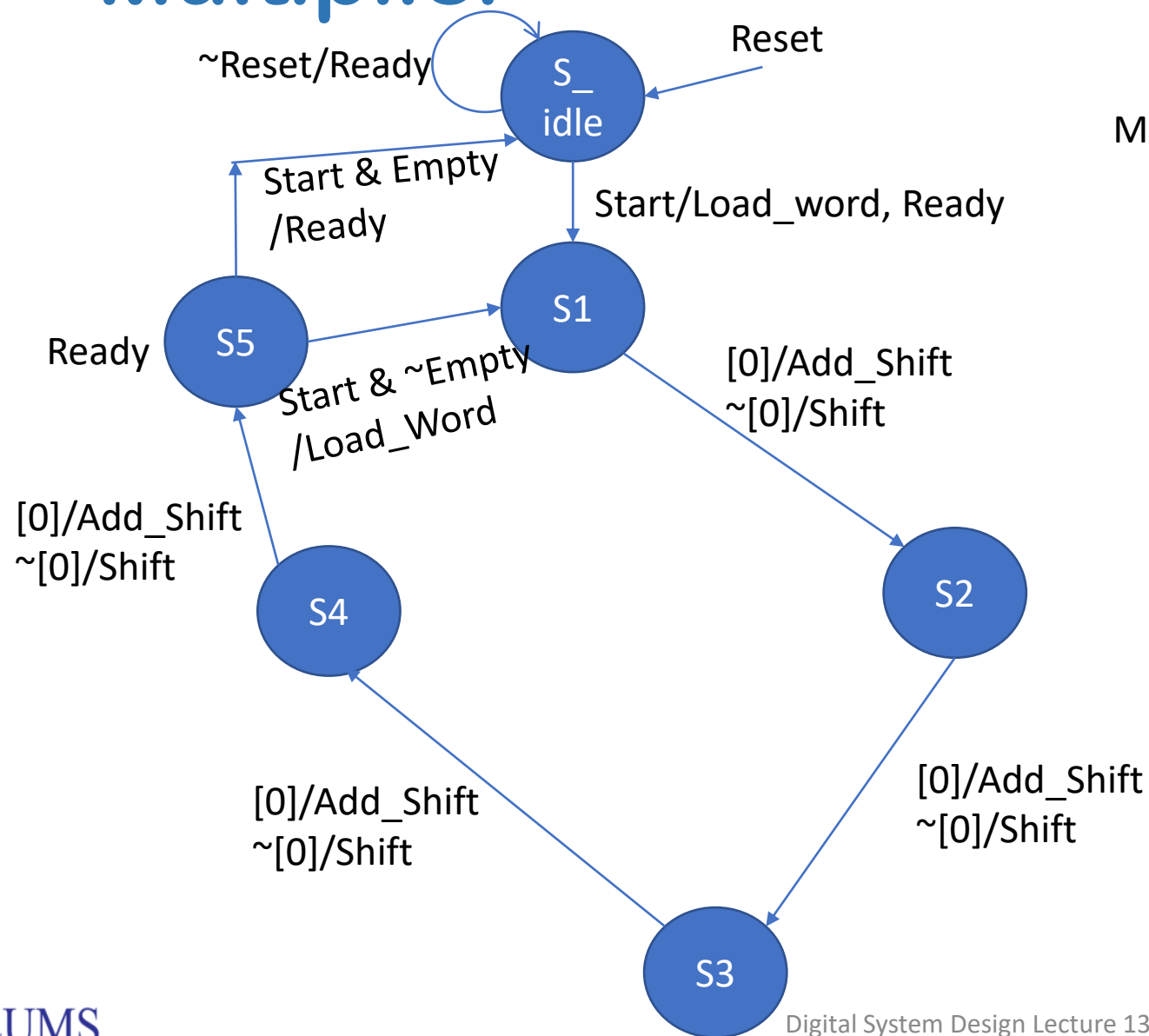# STG for a 4 Bit Sequential Binary Multiplier
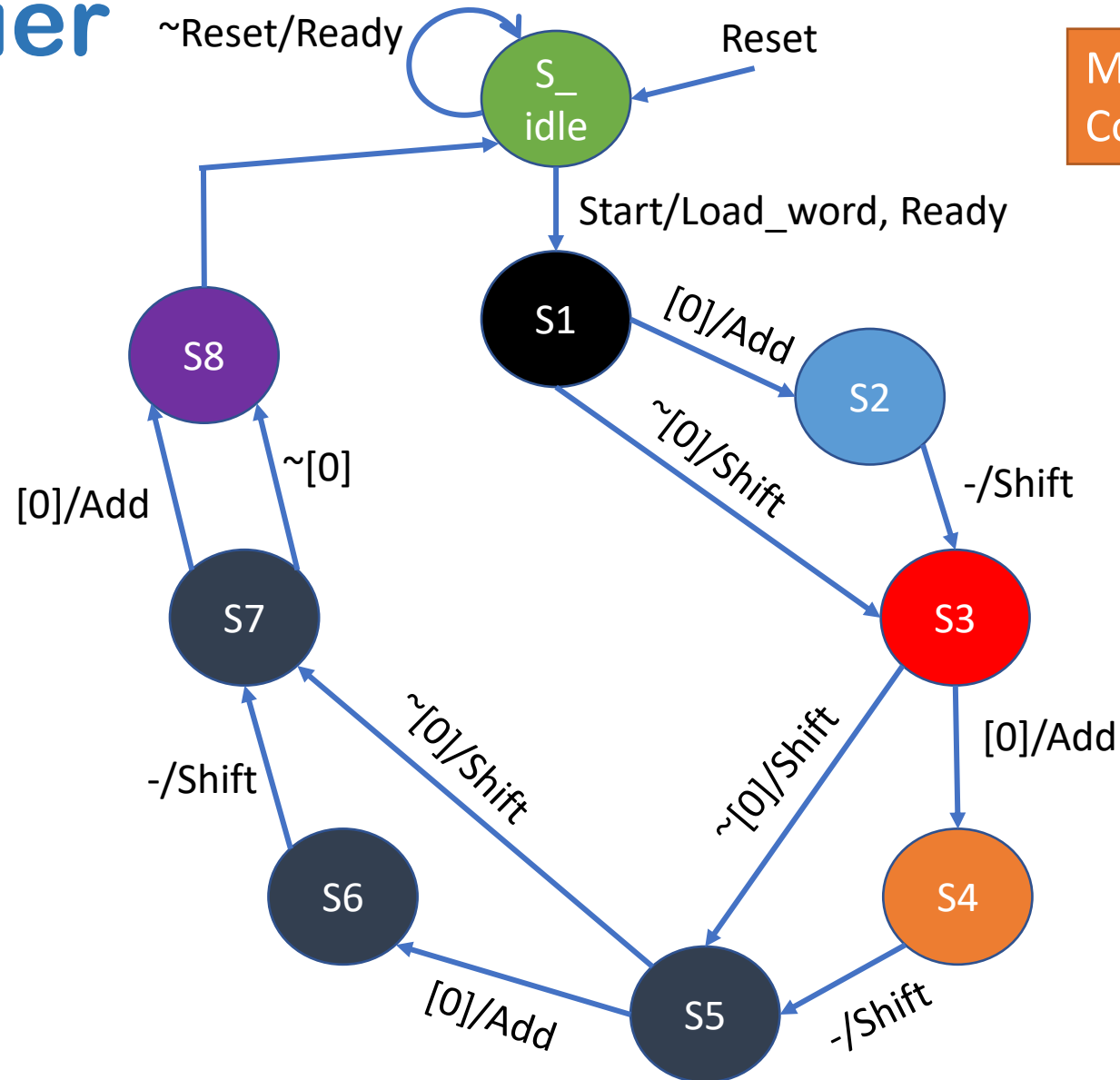


Machine returns to Idle state after Completion of 4 bit multiplication

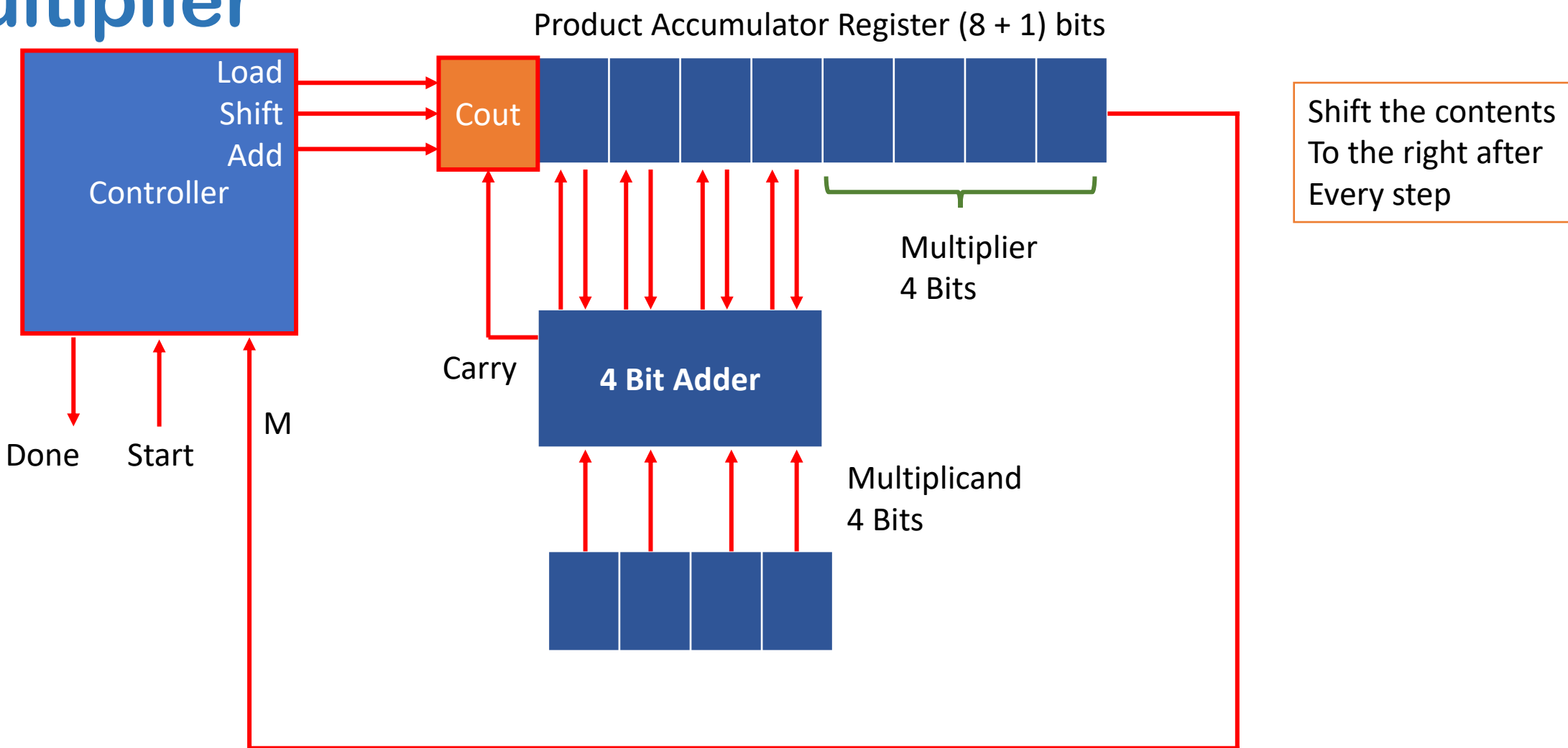# Sequential Multiplier with Reduced Registers



7        0    Multiplicand Register

8 bits

Multiplier Bits

15       7       0

Cout

Product Register
Shift Right after every step

LUMS

# STG of Reduced Register Sequential Multiplier

# Example of a 4-bit Serial Parallel Multiplier



Product Accumulator Register (8 + 1) bits

Load
Shift
Add
Controller
Cout

Shift the contents
To the right after
Every step

Multiplier
4 Bits

Carry

**4 Bit Adder**

Done    Start    M

Multiplicand
4 Bits

LUMS

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

LUMS

# Multiplier Register Operation

Check M bit [0]

| | |
|---|---|
| **Initial Contents of Accumulator** | 0 0 0 0 0 1 0 1 1 |
| **Multiplicand bit [0] is '1'** | 1 1 0 1 |
| **After Add operation** | 0 1 1 0 1 1 0 1 1 |
| **After Shift Right** | 0   0 1 1 0 1 1 0 1 1 |
| **Next bit M = 1 hence Add** | 1 1 0 1 X |
| **After Add** | 1 0 0 1 1 1 1 0 1 |
| **After Shift Right** | 0 1 0 0 1 1 1 1 0 |
| **Next bit M=0, hence Skip Add operation** | 0 0 1 0 0 1 1 1 1 |
| **After Shift Right** | 1 1 0 1 |
| **Next bit M=1, hence Add** | |
| **After Addition** | 1 0 0 0 1 1 1 1 1 |
| **After Shift Right, final answer** | 0   1 0 0 0 1 1 1 1 1 |

LUMS
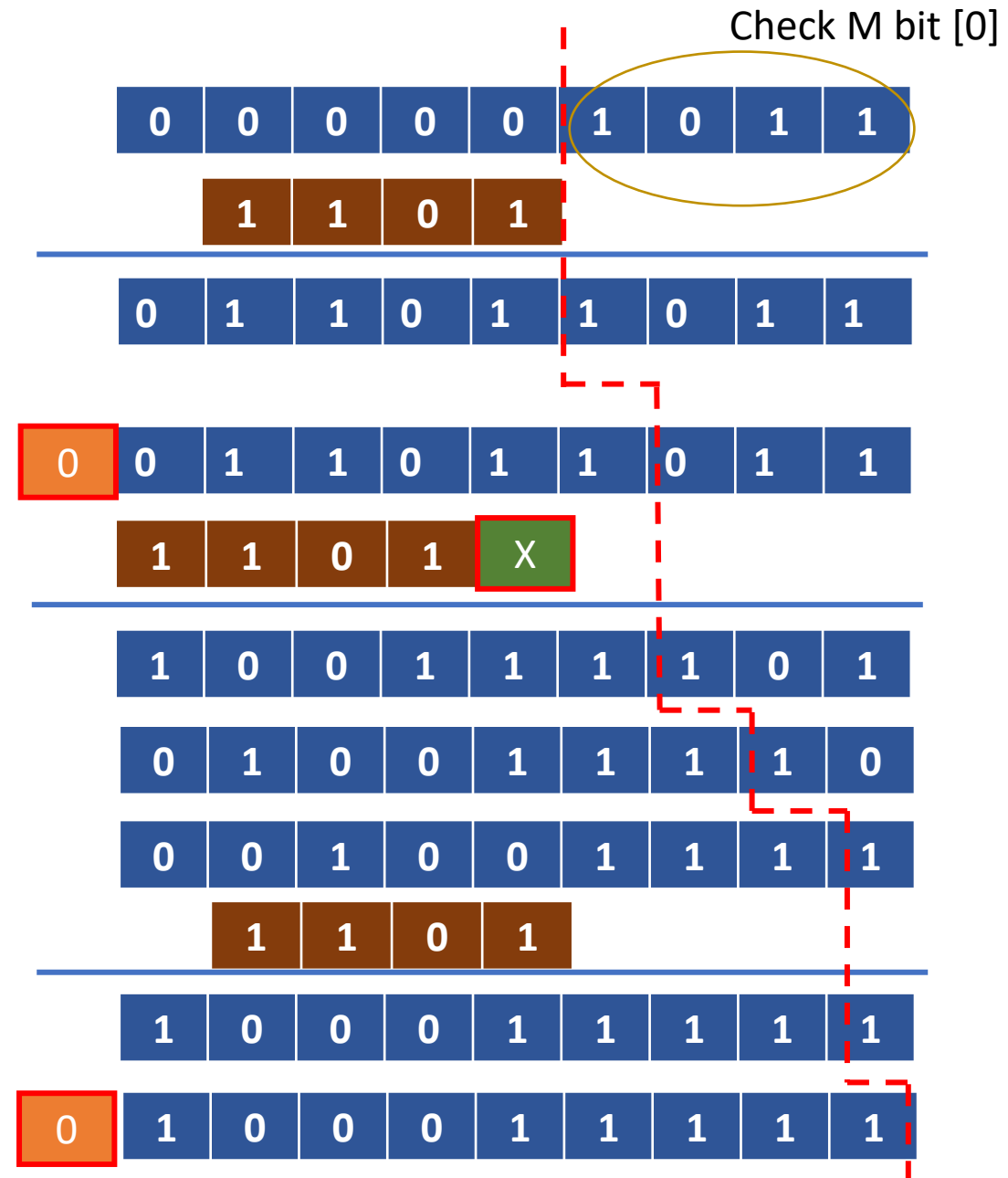
# STG Control Diagram for this Multiplier



M = relevant bit of multiplier
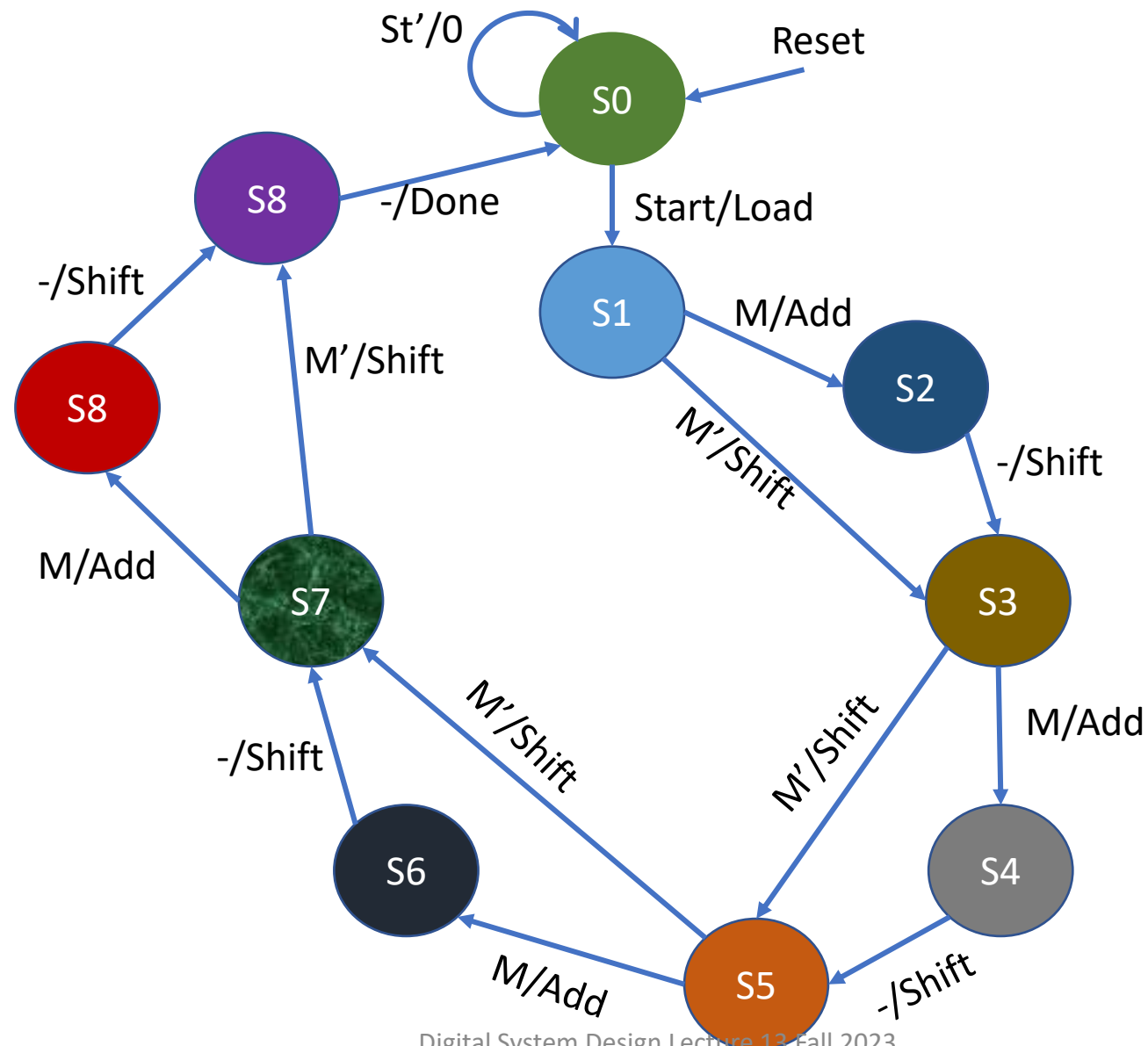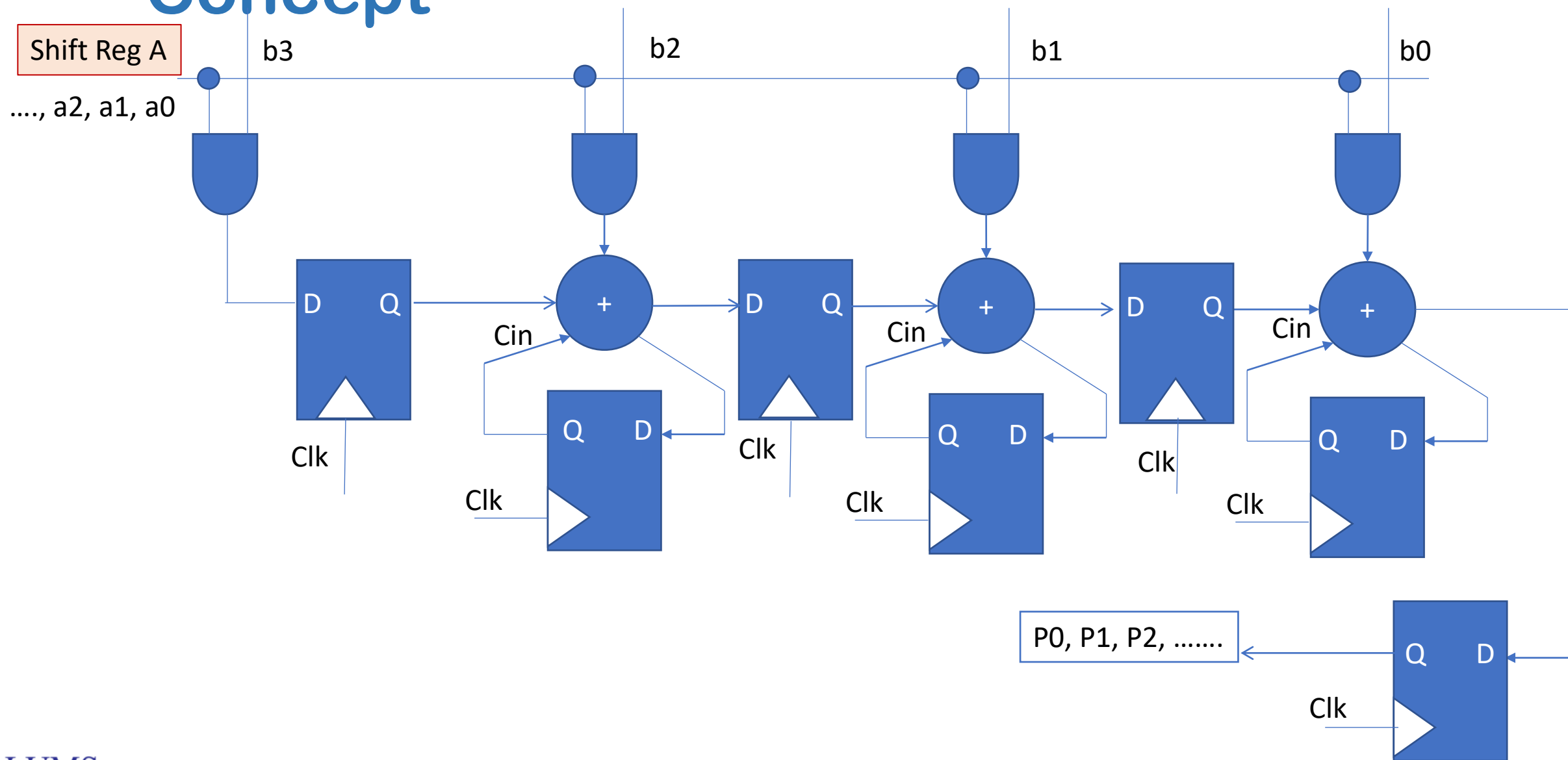
# Flexible STG for any no. of multiplicand bits
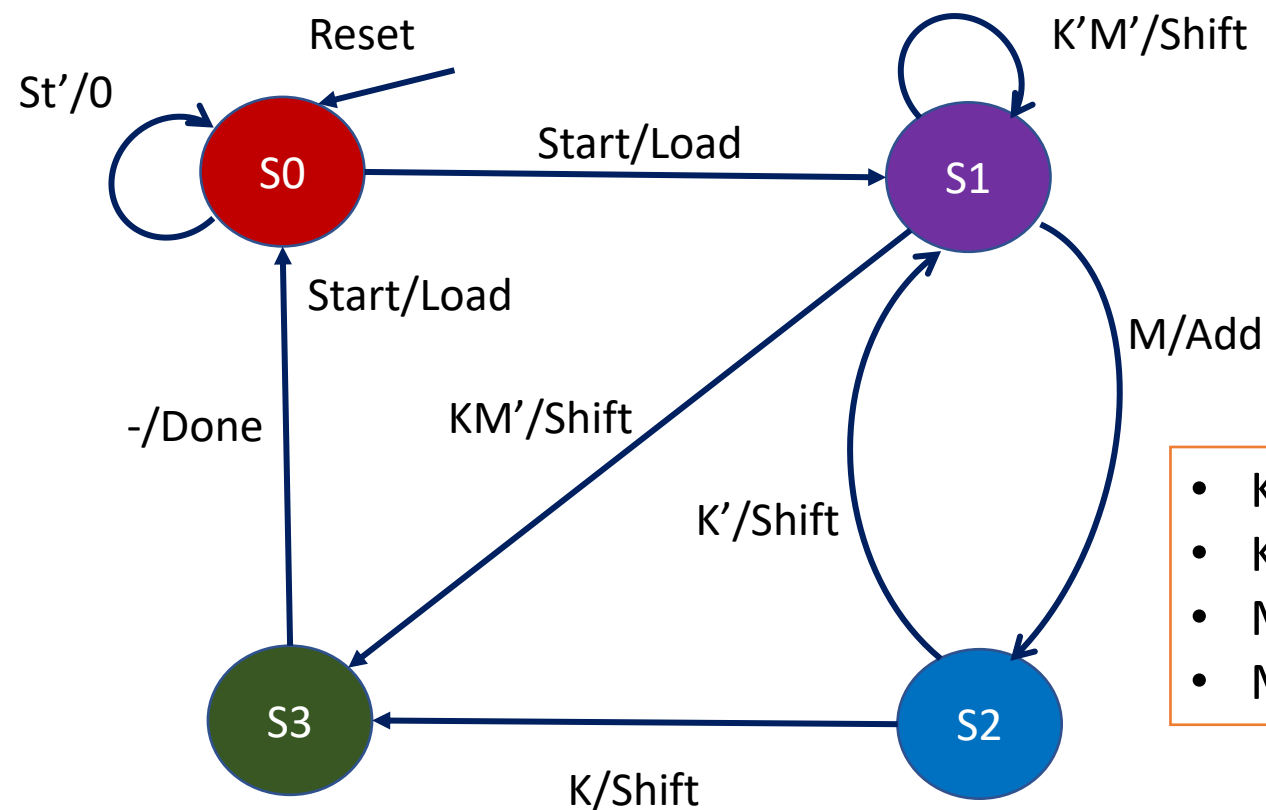


- K = Completion signal generated after (n-1) shifts
- K=1 means one more add (if needed) and shift operation
- M=0 means do the final shift
- M=1 means add before shift and go to S2

LUMS

# Parallel-Serial Multiplier - Concept

Shift Reg A

...., a2, a1, a0

b3

b2

b1

b0



P0, P1, P2, .......

LUMS

# Multiplication of <mark>Signed</mark> Binary Numbers

**Case I: Negative Multiplicand, Positive Multiplier**

**Example: $-3_{10}$ x $6_{10}$**

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) = $-18_{10}$**

LUMS

# Multiplication of Fractions

**Convert from decimal to binary** $\left(\dfrac{3}{4}\right)$

= 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

LUMS