Computer Organization & Architecture

# Chapter 9 – Unsigned Multiplication

Zhang Yang 张杨

cszyang@scut.edu.cn

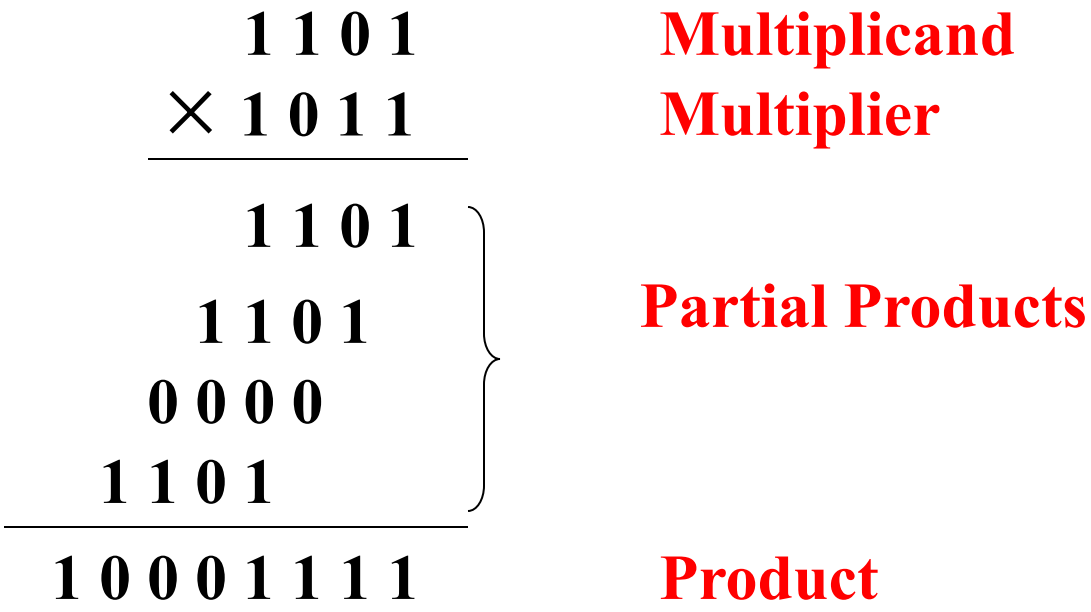
Autumn 2025

# Content of this lecture

- 9.3 Multiplication of Unsigned Numbers
  - Manual Multiplication
  - Array Multiplier
  - Sequential Multiplier
  - Summary

# Manual Multiplication (1)

- Unsigned & Positive Signed Numbers
  - Example: M= 1101, Q= 1011, calculate P= M×Q



# Manual Multiplication (2)

- Manual Multiplication Algorithm
  - Multiplication of the multiplicand by one bit of the multiplier
    - If the multiplier bit is 1, the multiplicand is entered in the appropriate position to be added to added to the partial product. If the multiplier bit is 0, then 0s are entered.
  - Note
    - The multiplication product of two n-bit binary integers results in a product of up to 2n bits in length.

# Content of this lecture

- 9.3 Multiplication of Unsigned Numbers
  - Manual Multiplication
  - Array Multiplier
  - Sequential Multiplier
  - Summary

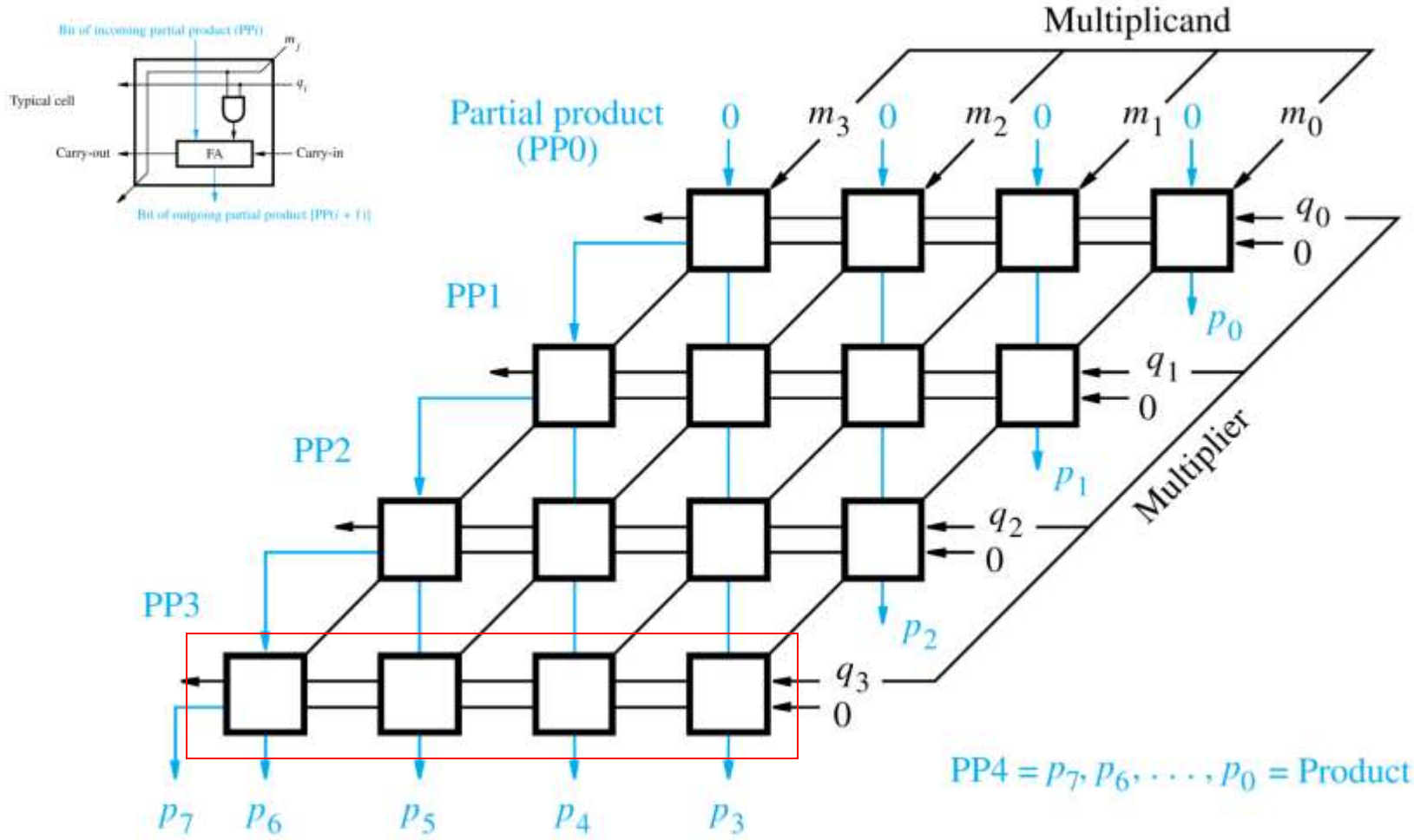
# Array Multiplier (1)

- Assume that
  - $M = m_3m_2m_1m_0 \quad Q = q_3q_2q_1q_0$
  - $P = M \times Q = p_7p_6p_5p_4p_3p_2p_1p_0$

$$
\begin{array}{cccccccc}
 & & m_3 & m_2 & m_1 & m_0 & & \\
 & \times & q_3 & q_2 & q_1 & q_0 & & \\
\hline
 & & m_3q_0 & m_2q_0 & m_1q_0 & m_0q_0 & & \\
 & m_3q_1 & m_2q_1 & m_1q_1 & m_0q_1 & & & \\
m_3q_2 & m_2q_2 & m_1q_2 & m_0q_2 & & & & \\
m_3q_3 & m_2q_3 & m_1q_3 & m_0q_3 & & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

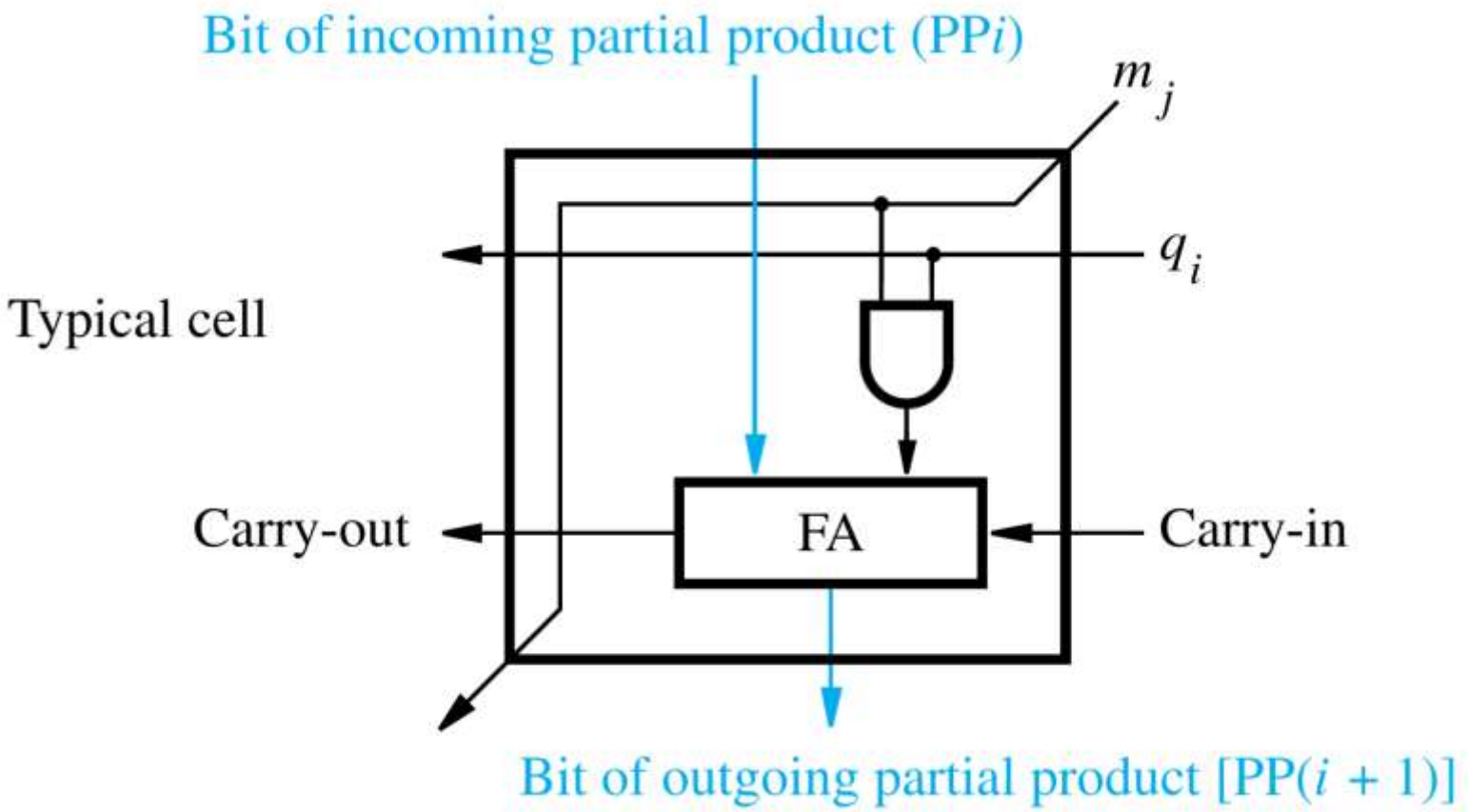
# Array Multiplier (2)

- Array Implementation



# Array Multiplier (3)

- Array Implementation (ctd.)



# Array Multiplier (4)

- Question: When calculating P = M × Q (M and Q are all n bit unsigned binary numbers) , how many FAs and AND gates do we need (using n × n array multiplier)?

A:

FAs : n(n-1)

AND gates : $n^2$

n= 32, 992 FAs, 1024 AND gates

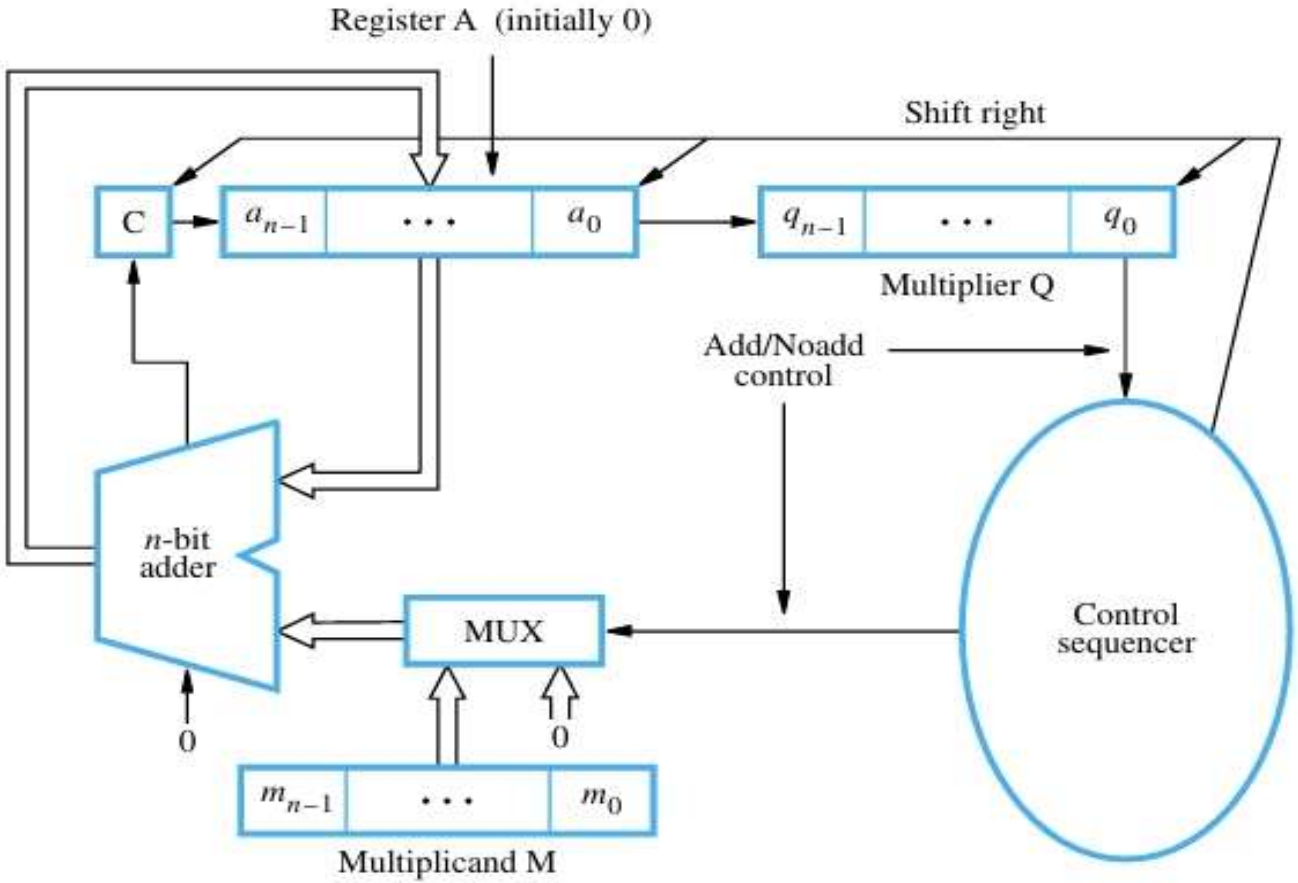
n= 64, 4032 FAs, 4096 AND gates

# Content of this lecture

- 9.3 Multiplication of Unsigned Numbers
  - Manual Multiplication
  - Array Multiplier
  - Sequential Multiplier
  - Summary

# Sequential Multiplier (1)

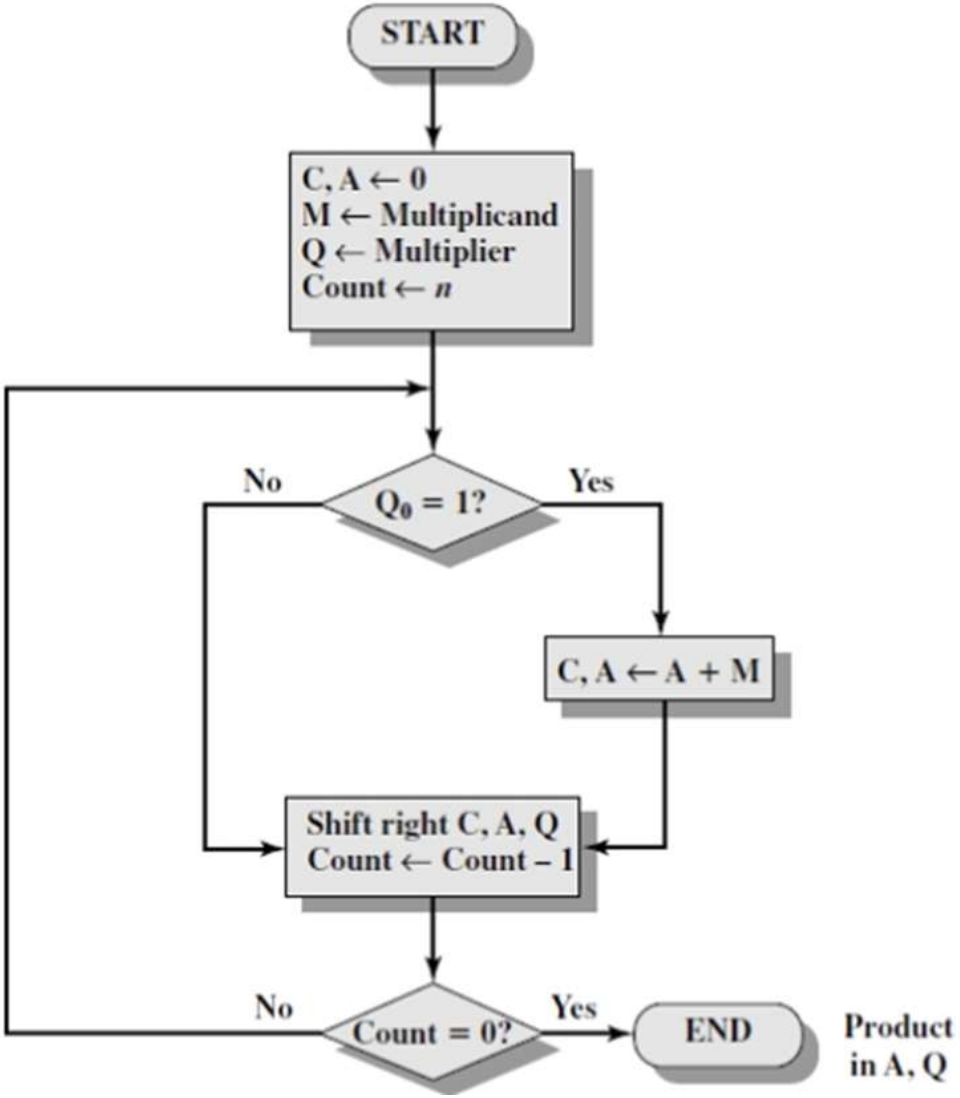
- Register Configuration




(a) Register configuration

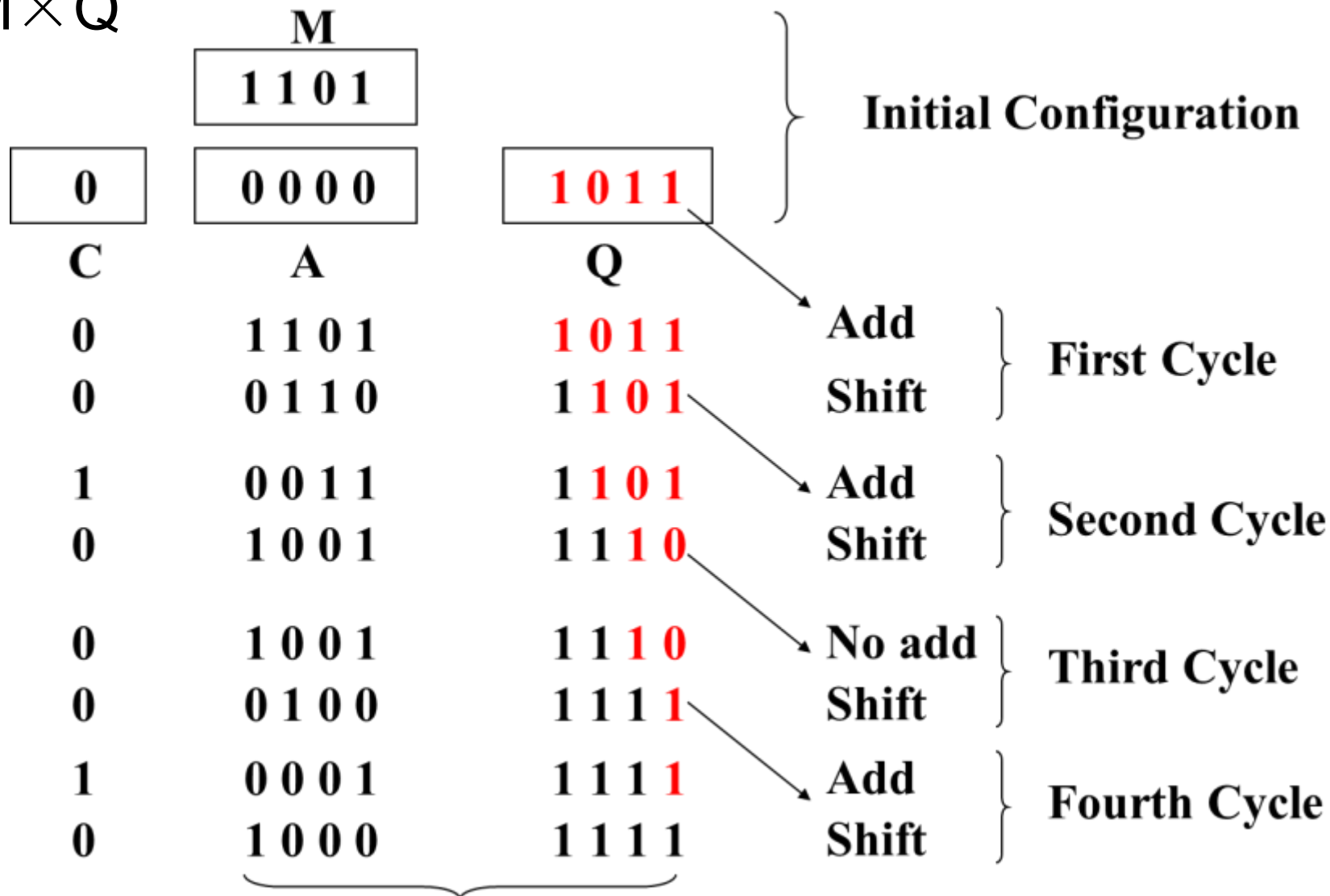
# Sequential Multiplier (2)

- Flowchart for Multiplication



# Sequential Multiplier (3)

- Multiplication Example: M= 1101, Q= 1011, calculate P= M×Q



# Summary

- 知识点Sequential Multiplier
- 掌握程度
  - 给定两个整数，能够用机器算法计算出结果，写出整个计算步骤。

Computer Organization & Architecture

# Chapter 9 – Signed Multiplication

Zhang Yang
cszyang@scut.edu.cn

Autumn 2024

# Content of this lecture

- 9.4 Multiplication of Signed Numbers
  - Signed-Operand Multiplication
  - Booth Algorithm
  - Multiplication Summary
  - Solved Problem Example 9.2 (P374)
  - Summary

# How do we handle signed numbers ?

- Convert to unsigned, multiply, then convert back to signed.
  - Works but  is slow and a bit cumbersome.
- Directly handle the multiplication with the signed numbers.
  - Booth algorithm recodes the bits to allow for such a computation.
  - Works with 2’s complement numbers directly.

# Signed-Operand Multiplication (1)

- If the multiplicand or the multiplier is negative, how should we do multiplication straightly?
- Case1: a negative multiplicand and a positive multiplier.
  - Example: M = 1001, Q = 0011, calculate P = M×Q
    - P = 00011011
    - If M, Q, and P are unsigned non-negative numbers, then: M = 9, Q = 3, P=27
    - If M, Q, and P are signed 2's complement numbers, then: M = -7, Q = 3, P = 27

# Signed-Operand Multiplication (2)

- Case 1: a negative multiplicand and a positive multiplier. (ctd.)
  - Example: M = 1001, Q =0011, calculate P = M×Q

$$
\begin{array}{r}
1\ 0\ 0\ 1 \\
\times\ 0\ 0\ 1\ 1 \\
\hline
1\ 1\ 1\ 1\ 1\ 0\ 0\ 1 \\
1\ 1\ 1\ 1\ 0\ 0\ 1 \\
0\ 0\ 0\ 0\ 0\ 0 \\
0\ \ 0\ 0\ 0\ 0 \\
\hline
1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \quad = -21
\end{array}
$$

# Signed-Operand Multiplication (3)

- Case 2: a negative multiplicand and a negative multiplier
  - Example: M = 1101, Q = 1011,  calculate P = M×Q
    - Straightforward multiplication P = 10001111
    - If M, Q, and P are unsigned non-negative numbers, then: M = 13, Q = 11, P = 143
    - If M, Q, and P are signed 2's complement numbers, then: M = - 3, Q = -5, P =  - 113

# Signed-Operand Multiplication (4)

- Case 2: a negative multiplier and a negative multiplicand (ctd.)
  - Example: M = 1101, Q = 1011, calculate P = M×Q

```
        0 0 1 1
      × 0 1 0 1
  ---------------   M = 0011, 2's complement of M
  0 0 0 0 0 0 1 1   N = 0101, 2's complement of N
  0 0 0 0 0 0 0 0
  0 0 0 0 1 1
  0 0 0 0 0
-------------------
  0 0 0 0 1 1 1 1   =15
```

# Content of this lecture

- 9.4 Multiplication of Signed Numbers
  - Signed-Operand Multiplication
  - Booth Algorithm
  - Multiplication Summary
  - Solved Problem Example 9.2 (P374)
  - Summary

# Booth Algorithm (1)

- Booth Algorithm
  - Notice the following equality (Booth did)
    - $2^J + 2^{J-1} + 2^{J-2} + \ldots + 2^K = 2^{J+1} - 2^K$
    - Example: 0111 = 1000 – 0001 (decimal notation: 7 = 8 – 1)
    - Example: 0011110 = 0100000 – 0000010 (decimal notation: 30 = 32 – 2)
    - We can exploit this to create a faster multiplier.
  - How?
    - Sequence of N 1s in the multiplier yields sequence of N additions.
    - Replace with one addition and one subtraction.

# Booth Algorithm (2)

- Booth Algorithm (ctd.)
  - Reducing number of partial products.
  - Fewer partial products generated for groups of consecutive 0's and 1's.
  - Group of consecutive 0's in multiplier - no new partial product - only shift partial product right one bit position for every 0.
  - Group of m consecutive 1's in multiplier - less than m partial products generated
    - Example: 0011110 = 0100000 – 0000010 (decimal notation: 30 = 32 – 2)

# Booth Algorithm (3)

- Recoding of a multiplier

| Multiplier | | Version of Multiplicand Selected by bit i |
|---|---|---|
| Bit i | Bit i-1 | |
| 0 | 0 | 0×M |
| 0 | 1 | +1×M |
| 1 | 0 | - 1×M |
| 1 | 1 | 0×M |

# Booth Algorithm (4)

- Recoding of a multiplier (ctd.)
  - Why it works?
    - $b \times (a_{31}a_{30}\ldots a_0)$

      $= a_0 \times b \times 2^0 + a_1 \times b \times 2^1 + \ldots + a_{31} \times b \times 2^{31}$

      $= (0 - a_0) \times b \times 2^0 + (a_0 - a_1) \times b \times 2^1 + \ldots$

      $+(a_{30} - a_{31}) \times b \times 2^{31} + a_{31} \times b \times 2^{32}$

# Booth Algorithm (5)

- Examples of Recoding of a Multiplier
  - Example 1
    - 0 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 0 0
    - 0+1 -1+1 0-1 0+1 0 0-1+1-1+1 0 -1 0 0
  - Example 2

$$\begin{array}{r} 0\ 1\ 1\ 0\ 1\ (+13) \\ \times\ 1\ 1\ 0\ 1\ 0\ (-6) \\ \hline \end{array}$$

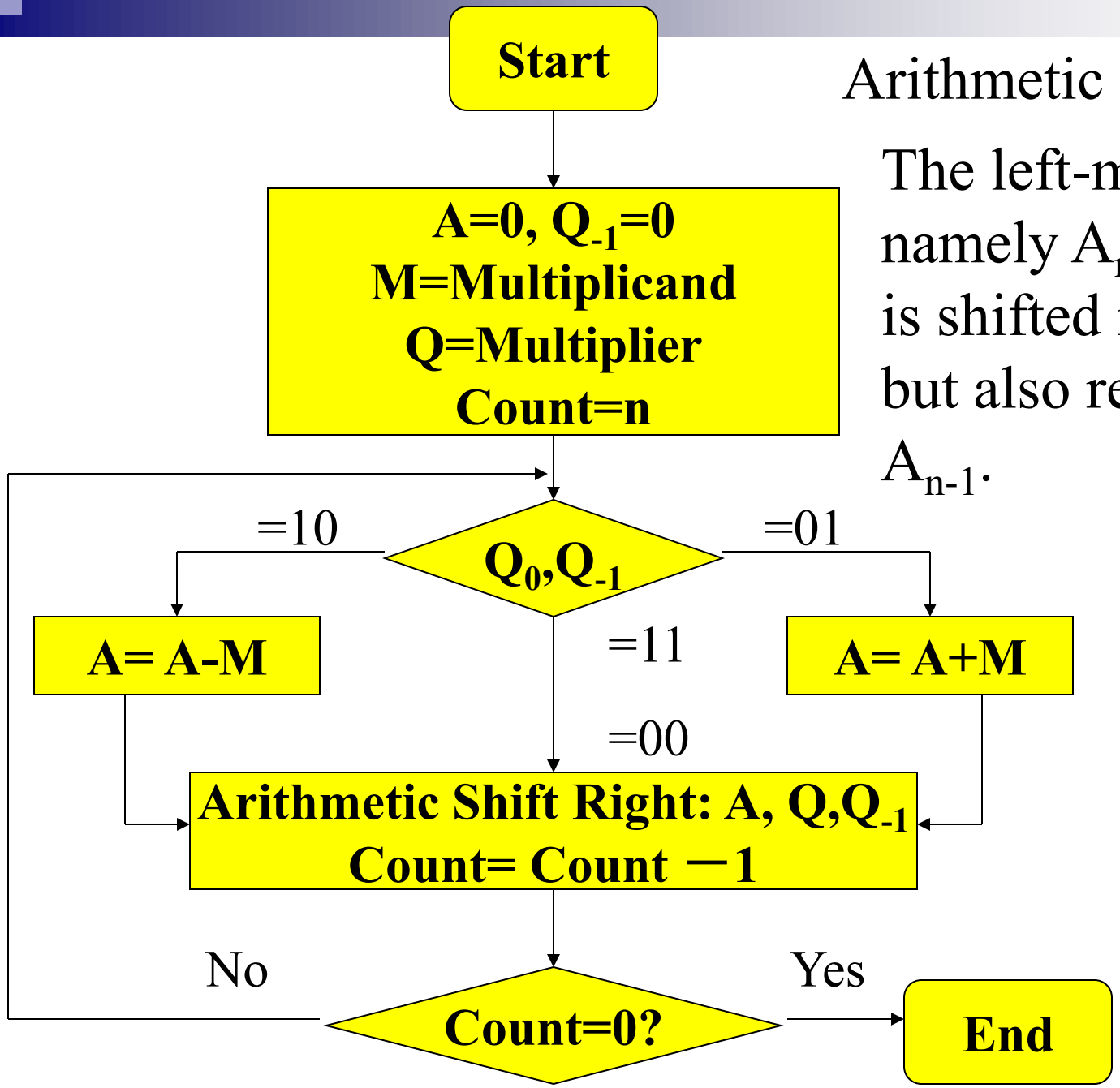
# Booth Algorithm (6)

- Examples of Recoding of a multiplier(ctd.)
  - Example 2 (ctd.)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 1 | 1 | 0 | 1 | |
| | | | | 0 | -1 | +1 | –1 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | | |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

# Booth Algorithm (7)

- Hardware Organization
  - As before, the multiplicand and multiplier are placed in the Q and M registers, respectively.
  - There is a 1-bit register placed logically to the right of the least significant bit (Q0) of the Q register and designated Q-1.
  - The results of the multiplication will appear in the A and Q registers.
  - Control logic scans the bits of the multiplier one at a time. Now, as each bit is examined, the bit to its right is also examined.




Arithmetic Shift Right:

The left-most of A, namely $A_{n-1}$, not only is shifted into $A_{n-2}$, but also remains in $A_{n-1}$.

# Booth Algorithm (8)

- Example

| A | Q | $Q_{-1}$ | M | | |
|---|---|---|---|---|---|
| 0000 | 0011 | 0 | 0111 | Initial Values | |
| 1001 | 0011 | 0 | 0111 | A ← A - M | First Cycle |
| 1100 | 1001 | 1 | 0111 | Shift | |
| 1110 | 0100 | 1 | 0111 | Shift | Second Cycle |
| 0101 | 0100 | 1 | 0111 | A ← A + M | Third Cycle |
| 0010 | 1010 | 0 | 0111 | Shift | |
| 0001 | 0101 | 0 | 0111 | Shift | Fourth Cycle |

# Booth Algorithm (9)

- Advantages
  - Handle both positive and negative multipliers uniformly.
  - Achieve some efficiency in the number of additions required when the multiplier has a few large blocks of 1s.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Good | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| multiplier | 0 | 0 | 0 | +1 | 0 | 0 | 0 | 0 | –1 | 0 | 0 | 0 | +1 | 0 | 0 | –1 |
| Ordinary | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| multiplier | 0 | –1 | 0 | 0 | +1 | –1 | +1 | 0 | – 1 | +1 | 0 | 0 | 0 | –1 | 0 | 0 |

# Booth Algorithm (10)

- Advantages
  - On average, the speed of doing multiplication with the Booth algorithm is the same as with the normal algorithm.

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Worst-case | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| multiplier | +1 | –1 | +1 | –1 | +1 | –1 | +1 | –1 | +1 | –1 | +1 | –1 | +1 | –1 | +1 | –1 |

# Content of this lecture

- 9.4 Multiplication of Signed Numbers
  - Signed-Operand Multiplication
  - Booth Algorithm
  - Multiplication Summary
  - Solved Problem Example 9.2 (P374)
  - Summary

# Multiplication Summary (1)

- Multiplication involves 2 basic operations: generation of partial products + their accumulation
- 2 ways to speed up
  - Reducing number of partial products and/or
  - Accelerating accumulation
- 3 types of high-speed multipliers
  - Sequential multiplier - generates partial products sequentially and adds each newly generated product to previously accumulated partial product

# Multiplication Summary (2)

- 3 types of high-speed multipliers (ctd.)
  - Parallel multiplier - generates partial products in parallel, accumulates using a fast multi-operand adder
  - Array multiplier - array of identical cells generating new partial products; accumulating them simultaneously
    - No separate circuits for generation and accumulation.
    - Reduced execution time but increased hardware complexity.

# Solved Problems

- Example 9.2

**Problem:** Assuming 6-bit 2's-complement number representation, multiply the multiplicand A = 110101 by the multiplier B = 011011 using both the normal Booth algorithm and the bit-pair recoding Booth algorithm, following the pattern used in Figure 9.15.

**Solution:** The multiplications are performed as follows:

(*a*) Normal Booth algorithm

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 1 | 0 | 1 | 0 | 1 |
| | | | | | × | +1 | 0 | −1 | +1 | 0 | −1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| | | | | | | | | | | 0 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | | | |
| | | | | | | | 0 | | | | |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | | | | | |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |

# Summary

- 知识点Booth Algorithm
  - Recoding of multiplier
  - Using recoded multiplier to multiply
- 掌握程度
  - 使用布斯算法熟练转换乘数
  - 使用手工算法将转换后的乘数与被乘数相乘

# Exercise (1)

- 1.In the hardware circuit of implementing Booth algorithm, the multiplier is placed in the Q register. And a 1-bit register $Q_{-1}$ is placed logically to the right of the least significant bit ($Q_0$) of the Q register. The results of the multiplication will appear in the A and Q registers. At the end of each cycle of computing the product, A, Q, $Q_{-1}$ are _____ one bit position.
  - shift left
  - shift right
  - arithmetically shift left
  - arithmetically shift right

# Exercise (2)

- 2. About Booth algorithm, which of the following is not true?
  - It can handle both positive and negative multipliers uniformly.
  - It achieves some efficiency in the number of additions required when the multiplier has a few large blocks of 1s.
  - Its speed of doing multiplication is always faster than the normal algorithm.
  - The multiplier 111100110 will generate three partial products.

# Exercise (3)

- 3. What are the main advantages of the Booth algorithm?
- Solution:
  - 1. Directly handle the multiplication with the signed numbers.
  - 2. Fewer partial products generated for groups of consecutive 0's and 1's.

# Homework

- P378 9.8