# ECE/CS 3700
# Digital System Design

## Chapter 3: Fast Adders and Multipliers


***Priyank Kalla***
Professor
Electrical & Computer Engineering

# Problems with Ripple Carry Adders

- Ripple carry adders are compact, simple to design and implement, and there is uniformity of design fabrication
- Ripple carry adders can be slow, when the carry ripples all the way across the carry chain

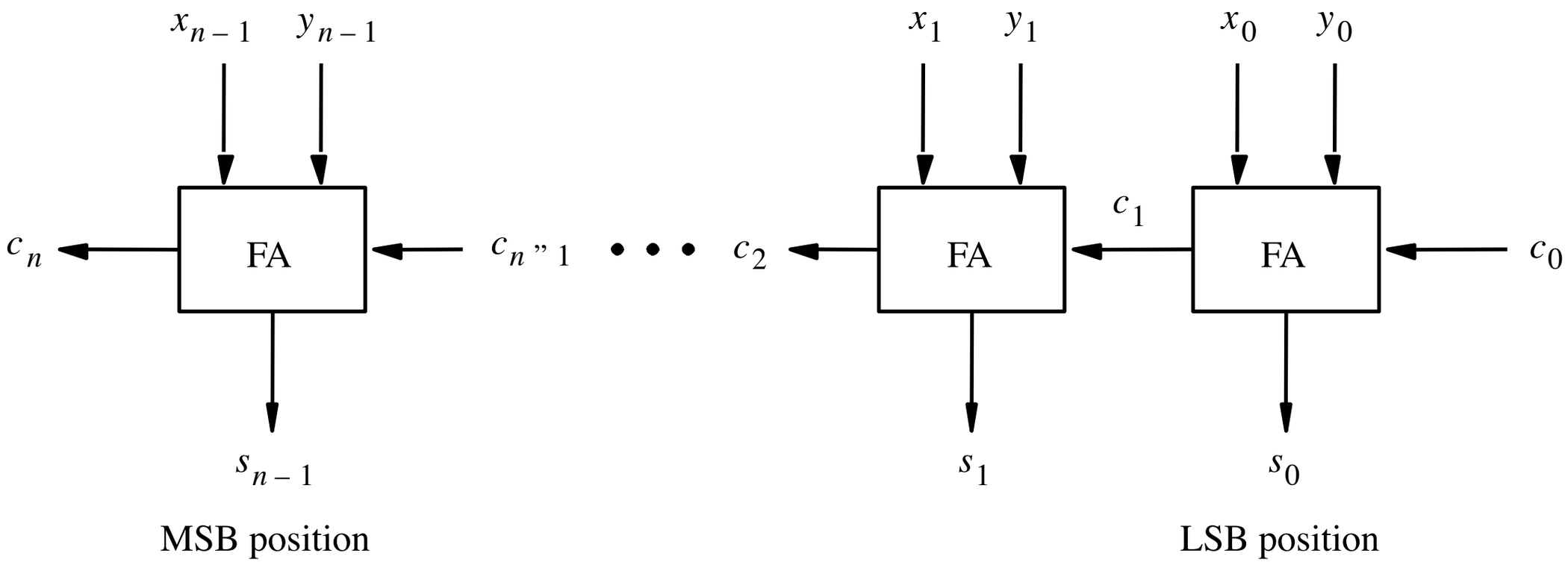
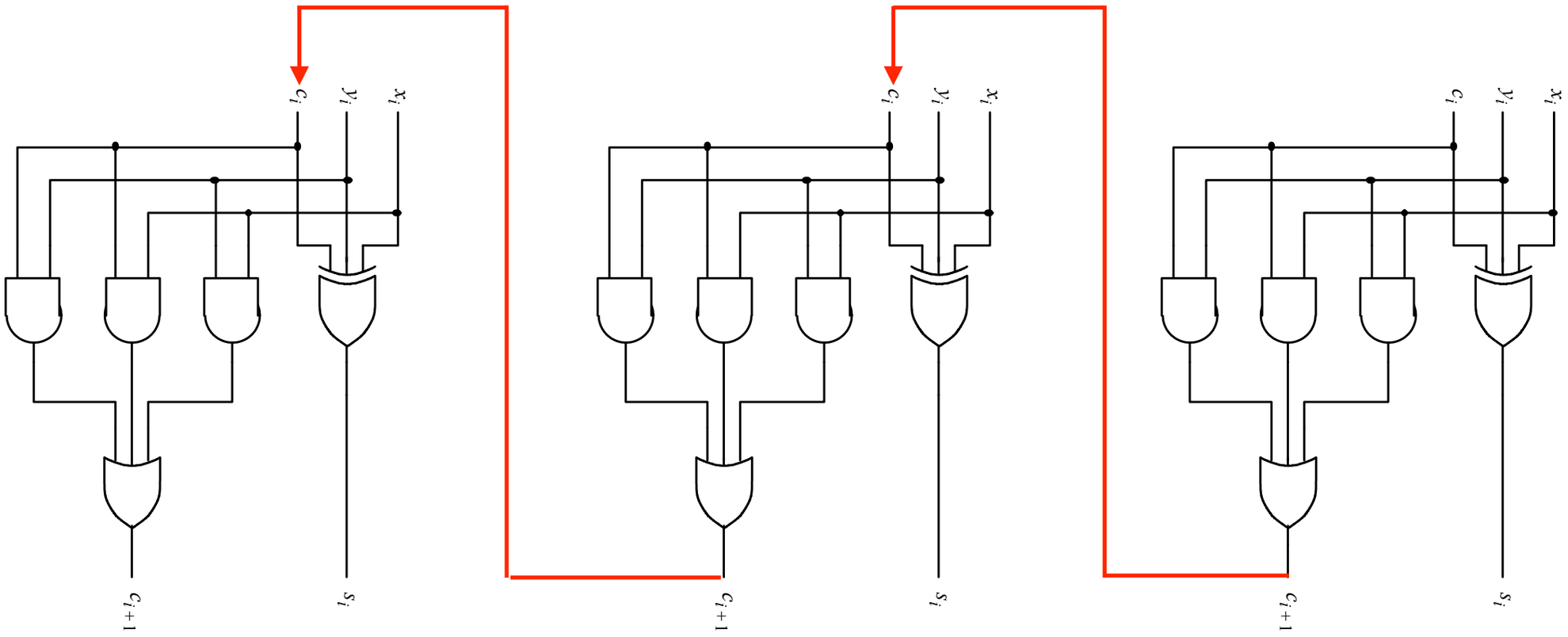



Figure 3.5. An *n*-bit ripple-carry adder.




- Observation:
- Carry-out of each stage relies on carry-in of the previous stage
  - This creates a **multi-level** logic circuit, where **levels** = **topological depth** = **gate delay**

- Objective to Speed-up the circuits:
- Can we have the carry-out of each stage rely mostly on the primary inputs of previous stages?
- Example: can we have *carry-out(stage 2) = F(x2, y2, x1, y1, x0, y0, c0)?*
- Then, $c_{i+1}$ does not have to wait for $c_i$ (cause of the delay)

# Towards a Faster Circuit

- Since carry-chains are a culprit, target the $c_{i+1}$ signal:

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

If we factor this expression as

$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$

then it can be written as

$$c_{i+1} = g_i + p_i c_i$$

where

$$g_i = x_i y_i$$
$$p_i = x_i + y_i$$

**Not a fast adder just yet, still a ripple-carry adder**

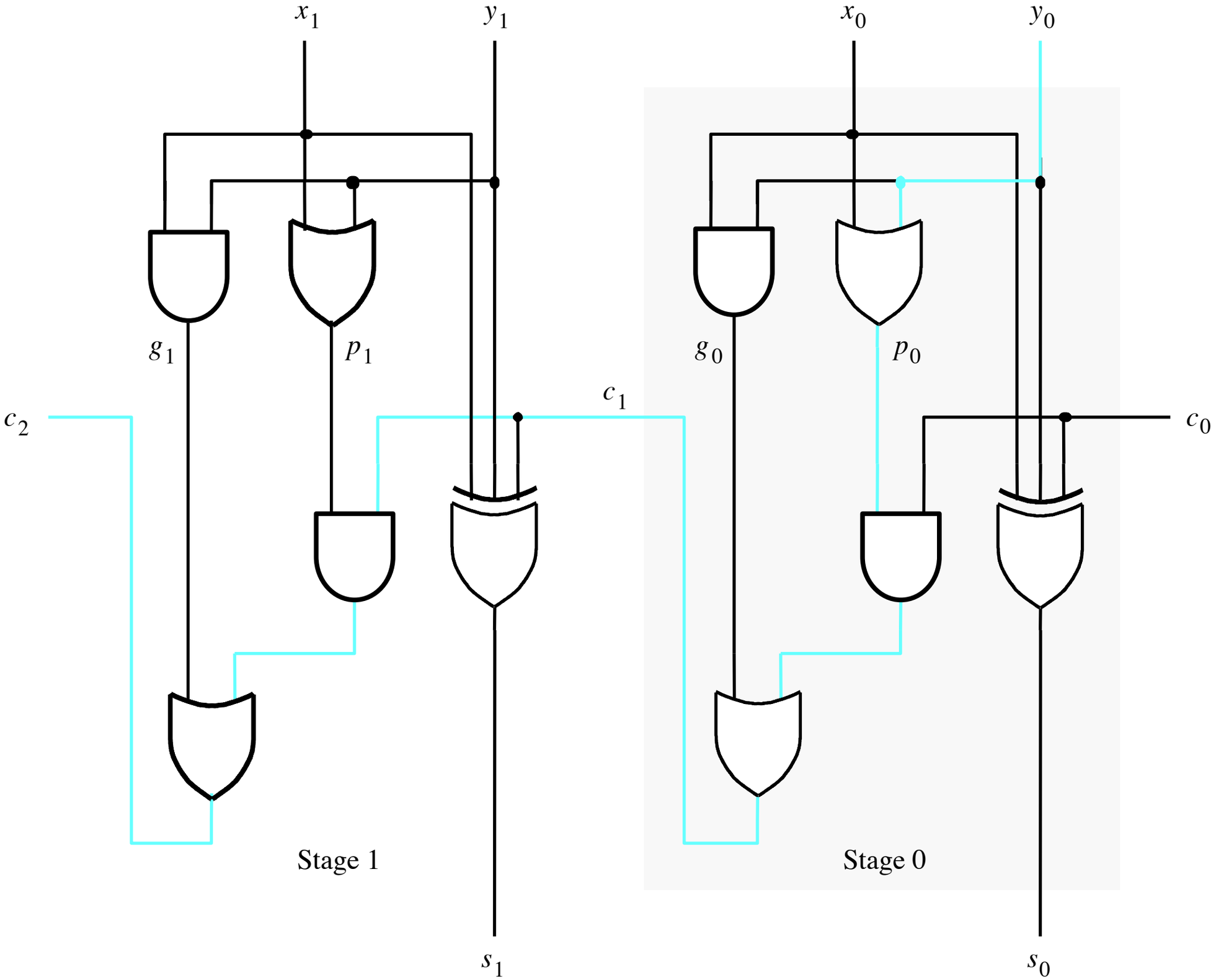



Figure 3.14. A ripple-carry adder based on Expression 3.3.

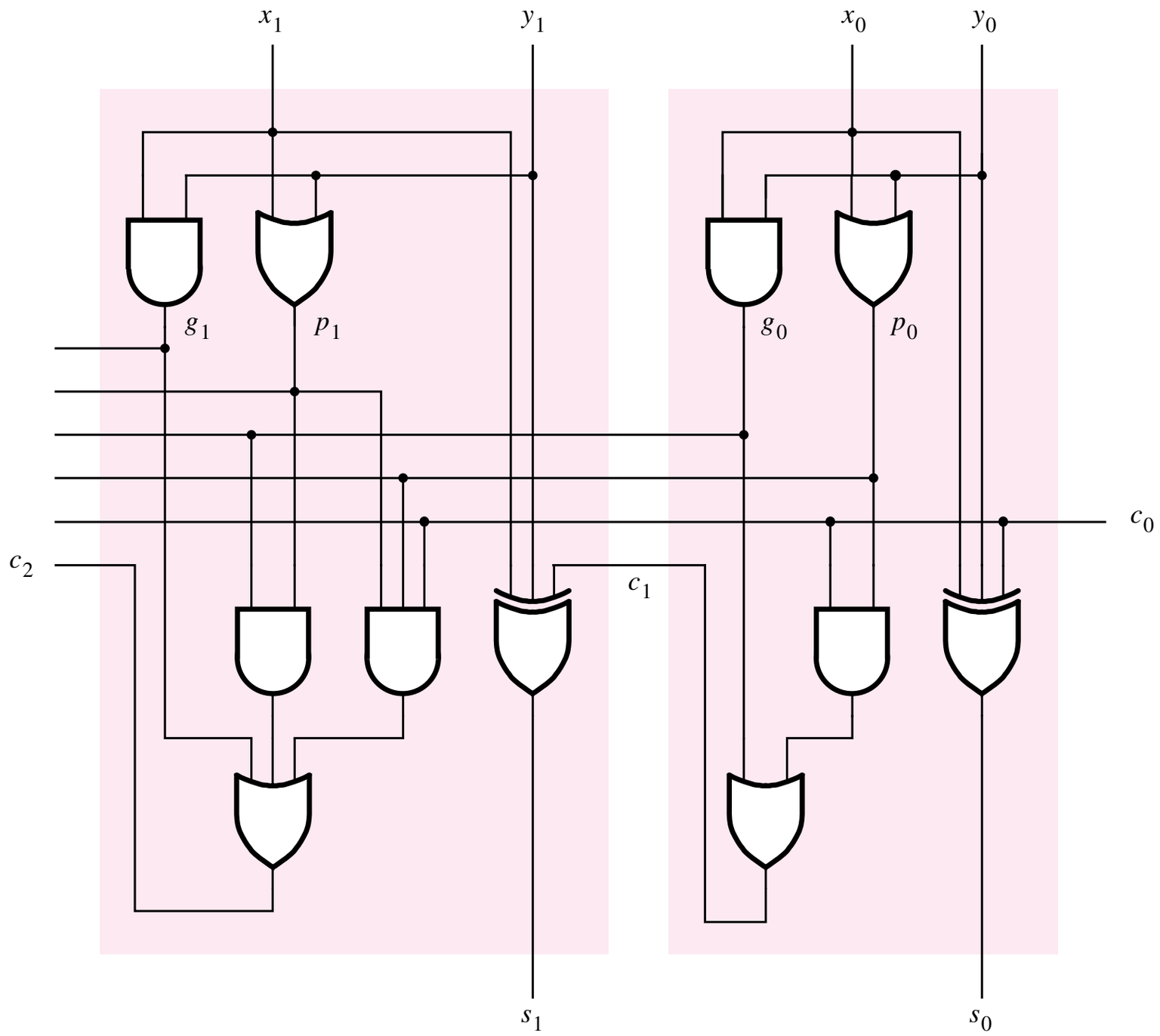
# Towards a Faster Circuit

- Since carry-chains are a culprit, target the $c_{i+1}$ signal:

$$c_{i+1} = g_i + p_i c_i$$
$$c_{i+1} = g_i + p_i(g_{i-1} + p_{i-1} c_{i-1})$$
$$= g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-1}$$

**Specifically, for 3 stage ripple-carry adder:**

$$c_1 = g_0 + p_0 \cdot c_0$$
$$c_2 = g_1 + g_0 \cdot p_1 + p_1 \cdot p_0 \cdot c_0$$
$$c_3 = g_2 + g_1 \cdot p_2 + g_0 \cdot p_1 \cdot p_2 + p_2 \cdot p_1 \cdot p_0 \cdot c_0$$




$$c_1 = g_0 + p_0 \cdot c_0$$

$$c_2 = g_1 + g_0 \cdot p_1 + p_1 \cdot p_0 \cdot c_0$$

$$c_3 = g_2 + g_1 \cdot p_2 + g_0 \cdot p_1 \cdot p_2 + p_2 \cdot p_1 \cdot p_0 \cdot c_0$$

# Carry Lookahead Adder Design for Lab 2

- Design the logic in such a way that the topological depth for each carry-out is 3-levels
- Use assign statements:

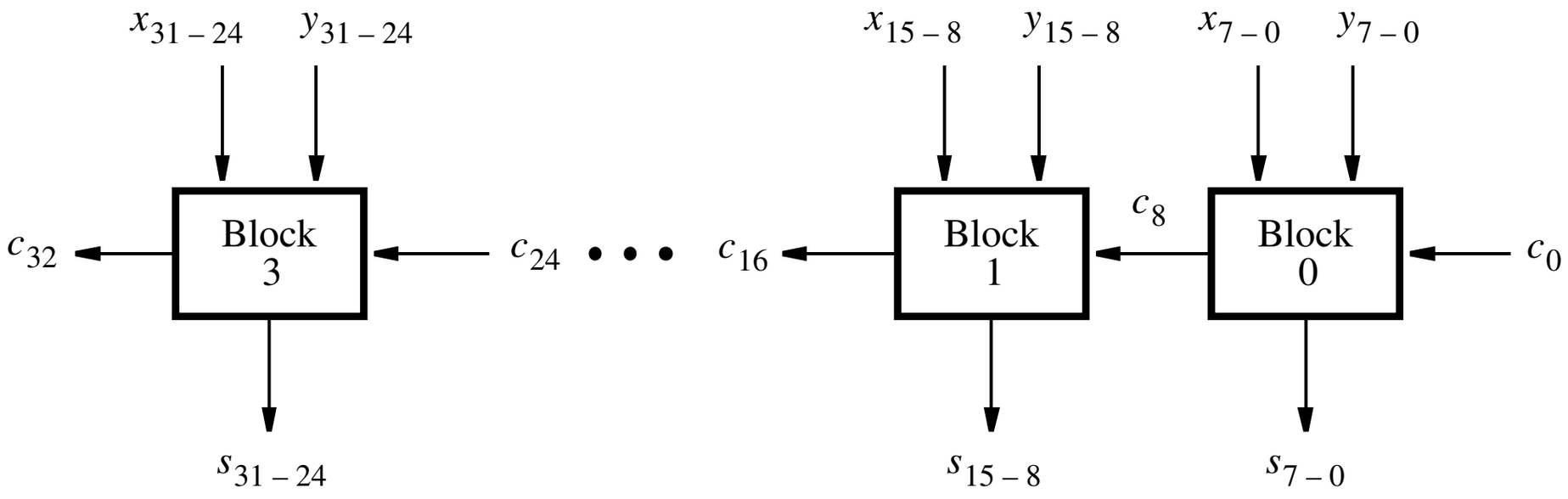



**Figure 3.16** A hierarchical carry-lookahead adder with ripple-carry between blocks.

- $c_8$ is a look-ahead carry
- $c_8 = g_7 + g_6p_7 + g_5p_6p_7 + \ldots + p_7 \cdots p_0 \cdot c_0$
- But $c_8$ ripples to the next block
- $c_{16} = g_{15} + g_{14}p_{15} + \ldots + p_{15} \cdot p_8 \cdot c_8$
- And so on..

| | | |
|---|---|---|
| Multiplicand M | (14) | 1 1 1 0 |
| Multiplier Q | (11) | x 1 0 1 1 |
| | | 1 1 1 0 |
| | | 1 1 1 0 |
| | | 0 0 0 0 |
| | | 1 1 1 0 |
| Product P | (154) | 1 0 0 1 1 0 1 0 |

(a) Multiplication by hand

| | | |
|---|---|---|
| Multiplicand M | (14) | 1 1 1 0 |
| Multiplier Q | (11) | x 1 0 1 1 |
| Partial product 0 | | 1 1 1 0 |
| | | + 1 1 1 0 |
| Partial product 1 | | 1 0 1 0 1 |
| | | + 0 0 0 0 |
| Partial product 2 | | 0 1 0 1 0 |
| | | + 1 1 1 0 |
| Product P | (154) | 1 0 0 1 1 0 1 0 |

(b) Using multiple adders

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | $m_3$ | $m_2$ | $m_1$ | $m_0$ | |
| | | x | $q_3$ | $q_2$ | $q_1$ | $q_0$ | |
| Partial product 0 | | | $m_3q_0$ | $m_2q_0$ | $m_1q_0$ | $m_0q_0$ | |
| | | + $m_3q_1$ | $m_2q_1$ | $m_1q_1$ | $m_0q_1$ | | |
| Partial product 1 | | $PP1_5$ | $PP1_4$ | $PP1_3$ | $PP1_2$ | $PP1_1$ | |
| | + $m_3q_2$ | $m_2q_2$ | $m_1q_2$ | $m_0q_2$ | | | |
| Partial product 2 | $PP2_6$ | $PP2_5$ | $PP2_4$ | $PP2_3$ | $PP2_2$ | | |
| + $m_3q_3$ | $m_2q_3$ | $m_1q_3$ | $m_0q_3$ | | | | |
| Product P | $p_7$ | $p_6$ | $p_5$ | $p_4$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ |

c) Hardware implementation

Figure 3.34. Multiplication of unsigned numbers.

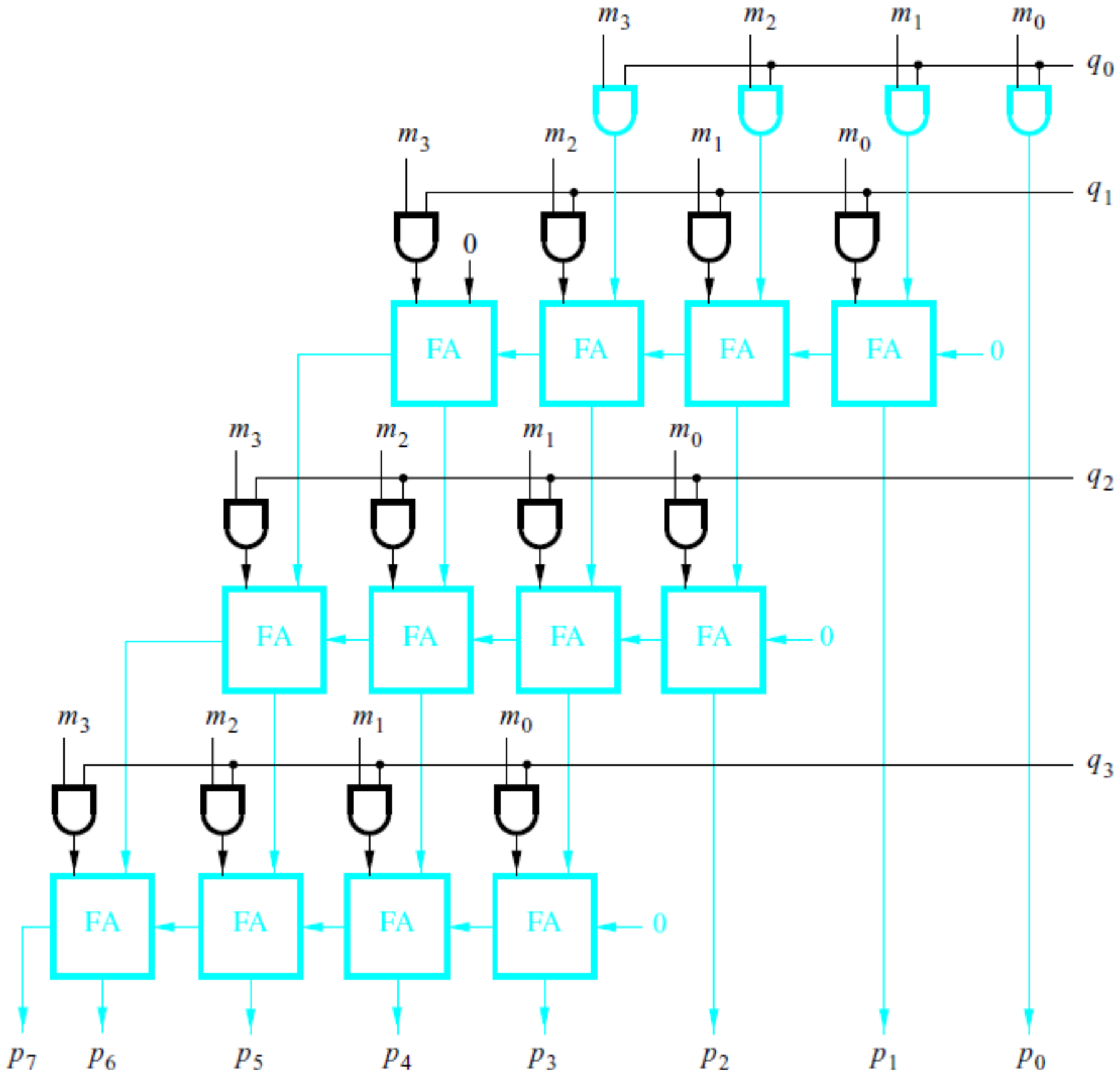



Figure 3.35. A 4x4 multiplier circuit.