

---

# **EE434**

# **ASIC & Digital Systems**

## **Arithmetic Circuits**

Spring 2015  
Dae Hyun Kim  
daehyun@eecs.wsu.edu

# Arithmetic Circuits

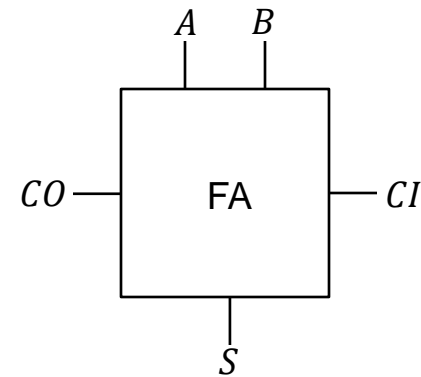
---

- What we will learn
  - Adders
    - Basic
    - High-speed

# Adder

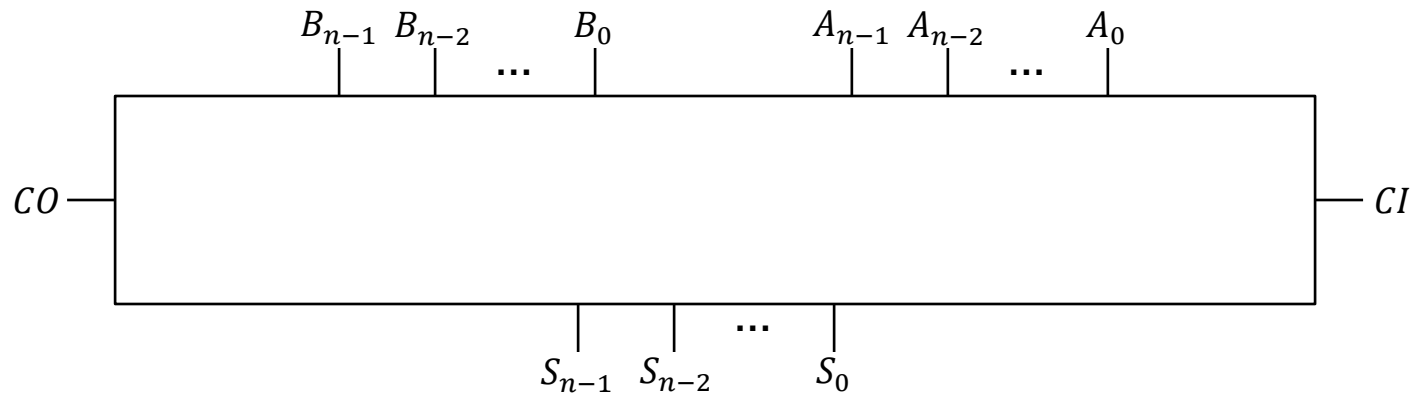
- 1-bit adder
  - $Sum = A \oplus B \oplus CI$
  - $CO = A \cdot B + B \cdot CI + A \cdot CI$

A	B	CI		Sum	CO
0	0	0		0	0
0	0	1		1	0
0	1	0		1	0
0	1	1		0	1
1	0	0		1	0
1	0	1		0	1
1	1	0		0	1
1	1	1		1	1



# Adder

- n-bit Adder



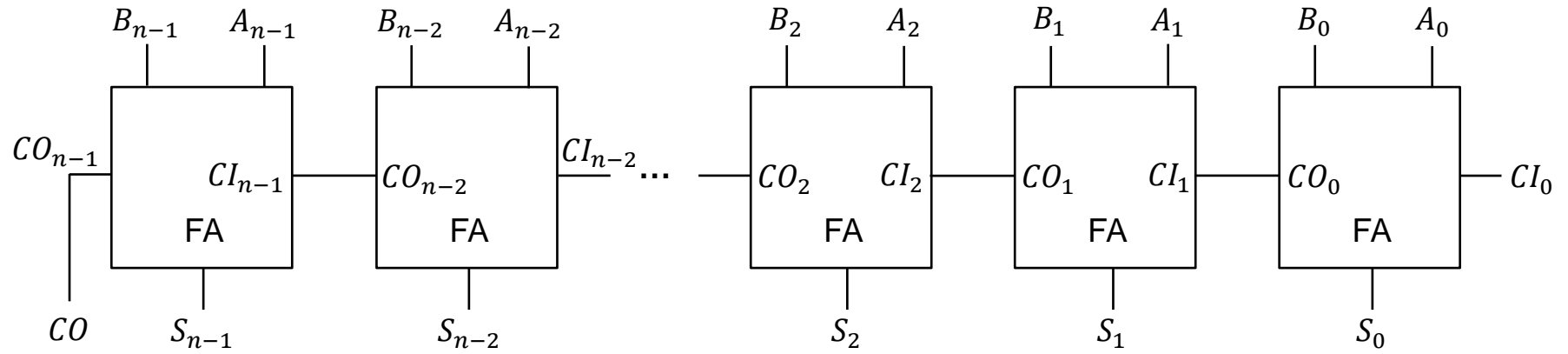
# Adder

---

- Two-operand adders
  - Ripple carry adder
  - Carry select adder
  - Conditional sum adder
  - Carry skip adder
  - Prefix adders
  - Carry lookahead adder
  - Ling adder

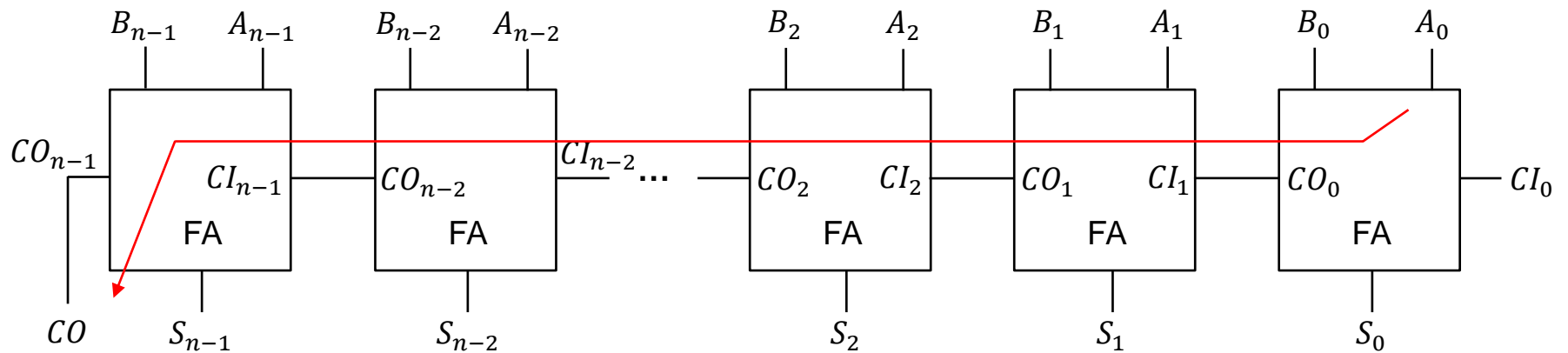
# Ripple Carry Adder

---



# Ripple Carry Adder

- Delay computation
  - FA delay:  $\Delta_{FA}$
  - n-bit RCA delay:  $n \cdot \Delta_{FA}$



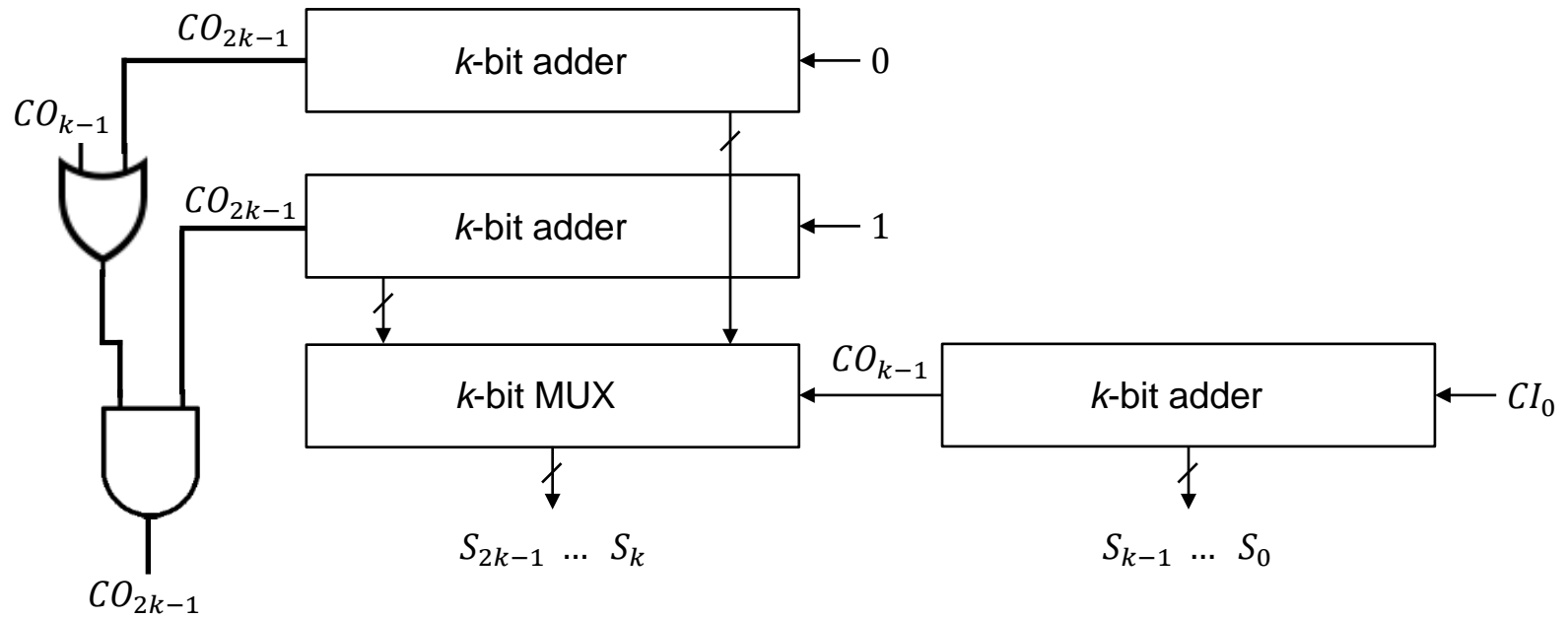
# Adder

---

- Bottleneck
  - Carry propagation
- How to improve
  - Parallelization



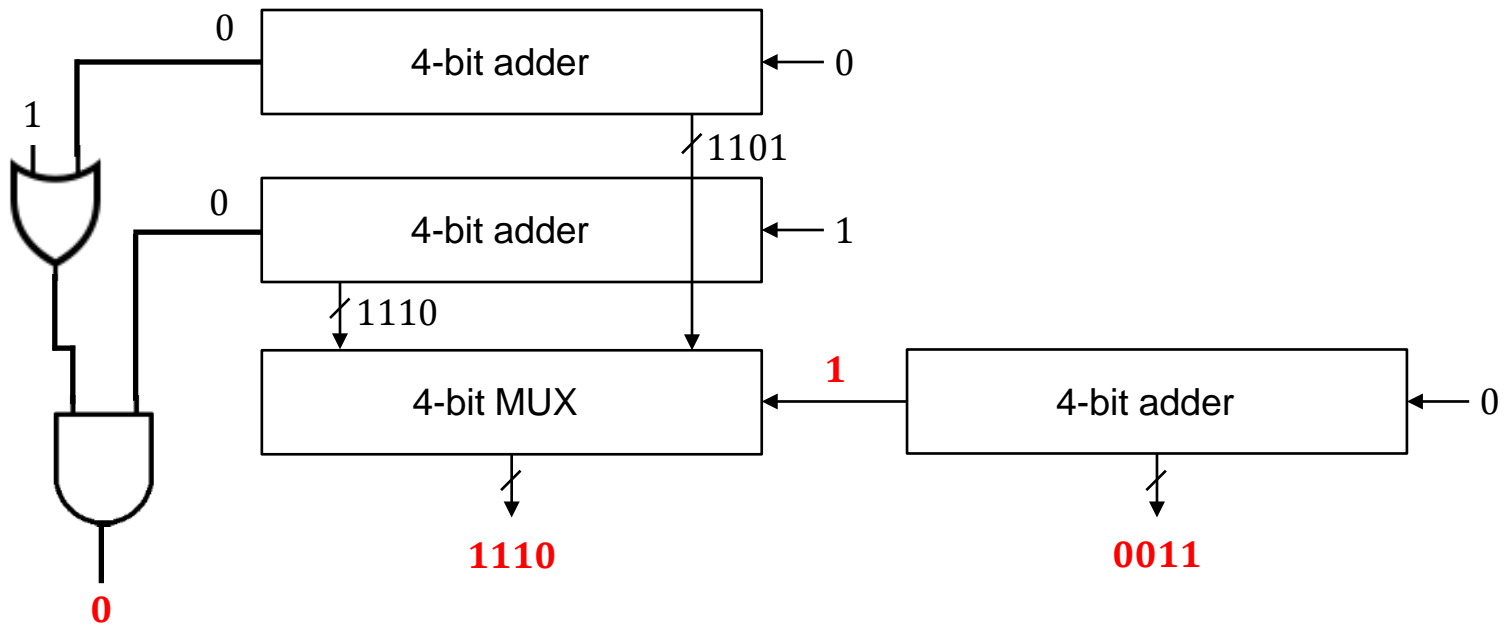
# Carry Select Adder



# Carry Select Adder

- Example

$i:$	7	6	5	4		3	2	1	0	$CI_0 = 0$
$A_i:$	1	0	1	1		0	1	1	0	
$B_i:$	0	0	1	0		1	1	0	1	



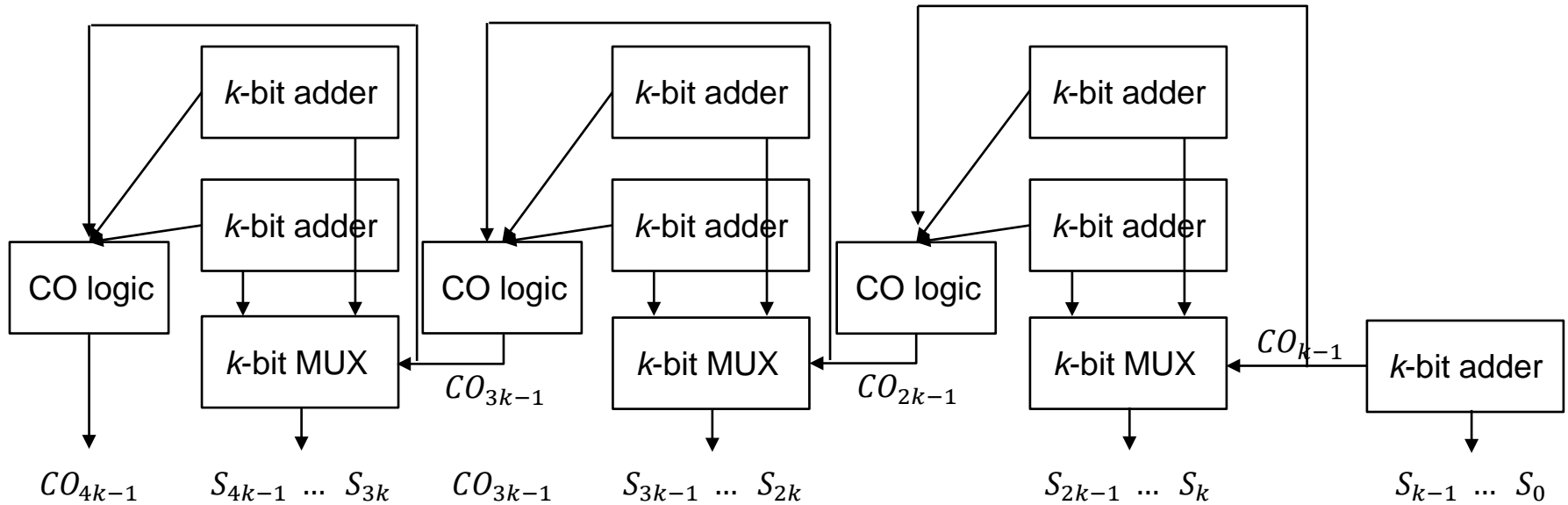
# Carry Select Adder

---

- Optimization example
  - 64-bit carry select adder
- Assumptions
  - The k-bit adder is a ripple-carry adder.
  - Delay of a k-bit adder:  $k \cdot \Delta_{FA}$
  - Delay of a carry-out logic or a MUX:  $\varepsilon$

# Carry Select Adder

- Optimization



- Delay:  $k \cdot \Delta_{FA} + 3 \cdot \varepsilon$

# Carry Select Adder

---

- Optimization (N-bit adder)
  - $k$ -bit adder
  - # groups:  $N/k$
  - # Carry-out logic stages:  $N/k - 1$
  - Delay:  $\tau = k \cdot \Delta_{FA} + \varepsilon \left( \frac{N}{k} - 1 \right)$

$$\frac{d\tau}{dk} = \Delta_{FA} - \frac{N \cdot \varepsilon}{k^2} = 0$$

$$k = \sqrt{\frac{N \cdot \varepsilon}{\Delta_{FA}}}$$

# Carry Select Adder

---

- Delay:  $k = \sqrt{\frac{N \cdot \varepsilon}{\Delta_{FA}}}$
- Example
  - 64-bit adder,  $\Delta_{FA} = 200ps$ ,  $\varepsilon = 40ps$ 
    - $k = \sqrt{\frac{64 \cdot 40}{200}} = 3.57 \rightarrow k = 4.$
    - Delay:  $\tau = 4 \cdot 200ps + 40ps \cdot \left(\frac{64}{4} - 1\right) = 1400ps$

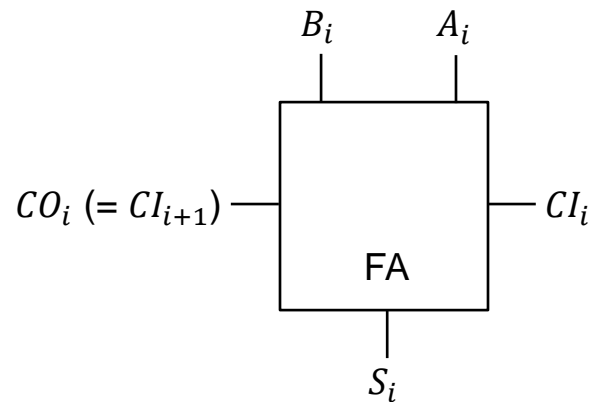
# Conditional Sum Adder

---

- Generate two sets of outputs for a given group of operand bits.
  - Set 1: assumes the incoming carry is zero.
  - Set 2: assumes the incoming carry is one.
- Once the incoming carry is known, select the correct set of outputs.

# Conditional Sum Adder

- Generate two sets of outputs for a given group of operand bits.



Assuming incoming carry ( $CI_i$ ) = 0

$$S_i = S_i^0 = A \oplus B \oplus 0 \rightarrow \text{Delay} = \Delta_{FA}$$

$$CO_i = CO_i^0 = A \cdot B + 0 \cdot (A + B) \rightarrow \text{Delay} = \Delta_{FA}$$

Assuming incoming carry ( $CI_i$ ) = 1

$$S_i = S_i^1 = A \oplus B \oplus 1 \rightarrow \text{Delay} = \Delta_{FA}$$

$$CO_i = CO_i^1 = A \cdot B + 1 \cdot (A + B) \rightarrow \text{Delay} = \Delta_{FA}$$



# Conditional Sum Adder

- Generate two sets of outputs

	$i:$	7	6	5	4	3	2	1	0
	$A_i:$	1	0	1	1	0	1	1	0
	$B_i:$	0	0	1	0	1	1	0	1
Step 1	$S_i^0:$	1	0	0	1	1	0	1	1
	$CO_i^0:$	0	0	1	0	0	1	0	0
	$S_i^1:$	0	1	1	0	0	1	0	
	$CO_i^1:$	1	0	1	1	1	1	1	

$$CI_0 = 0$$

# Conditional Sum Adder

- Merge two bits

	$i:$	7	6	5	4	3	2	1	0
	$A_i:$	1	0	1	1	0	1	1	0
	$B_i:$	0	0	1	0	1	1	0	1
Step 1	$S_i^0:$	1	0	0	1	1	0	1	1
	$CO_i^0:$	0	0	1	0	0	1	0	0
Step 1	$S_i^1:$	0	1	1	0	0	1	0	
	$CO_i^1:$	1	0	1	1	1	1	1	
Step 2	$S_i^0:$	1	0	0	1	0	0	1	1
	$CO_i^0:$	0		1		1		0	
Step 2	$S_i^1:$	1	1	1	0	0	1		
	$CO_i^1:$	0		1		1			

$CI_0 = 0$

# Conditional Sum Adder

- Merge four bits

	$i:$	7	6	5	4	3	2	1	0
	$A_i:$	1	0	1	1	0	1	1	0
	$B_i:$	0	0	1	0	1	1	0	1
Step 1	$S_i^0:$	1	0	0	1	1	0	1	1
	$CO_i^0:$	0	0	1	0	0	1	0	0
Step 2	$S_i^1:$	0	1	1	0	0	1	0	
	$CO_i^1:$	1	0	1	1	1	1	1	
Step 3	$S_i^0:$	1	0	0	1	0	0	1	1
	$CO_i^0:$	0				1			
Step 3	$S_i^1:$	1	1	1	0	0	1		
	$CO_i^1:$	0				1			

$$CI_0 = 0$$

# Conditional Sum Adder

- Final

	$i:$	7	6	5	4	3	2	1	0
	$A_i:$	1	0	1	1	0	1	1	0
	$B_i:$	0	0	1	0	1	1	0	1
Step 1	$S_i^0:$	1	0	0	1	1	0	1	1
	$CO_i^0:$	0	0	1	0	0	1	0	0
Step 2	$S_i^1:$	0	1	1	0	0	1	0	
	$CO_i^1:$	1	0	1	1	1	1	1	
Step 3	$S_i^0:$	1	0	0	1	0	0	1	1
	$CO_i^0:$	0		1		1		0	
Step 3	$S_i^1:$	1	1	1	0	0	1		
	$CO_i^1:$	0		1		1			
Result		1	1	1	0	0	0	1	1

$$CI_0 = 0$$

# Conditional Sum Adder

- Exercise

	$i:$	7	6	5	4	3	2	1	0
	$A_i:$	1	0	1	1	0	1	1	0
	$B_i:$	0	0	1	0	1	1	0	1
Step 1	$S_i^0:$								
	$CO_i^0:$								
Step 2	$S_i^1:$								
	$CO_i^1:$								
Step 3	$S_i^0:$								
	$CO_i^0:$								
Step 3	$S_i^1:$								
	$CO_i^1:$								
Result									

$$CI_0 = 1$$

# Conditional Sum Adder

- Analysis

	$i:$	7	6	5	4	3	2	1	0
	$A_i:$	1	0	1	1	0	1	1	0
	$B_i:$	0	0	1	0	1	1	0	1
Step 1	$S_i^0:$	1	0	0	1	1	0	1	1
	$CO_i^0:$	0	0	1	0	0	1	0	0
	$S_i^1:$	0	1	1	0	0	1	0	
	$CO_i^1:$	1	0	1	1	1	1	1	
Step 2	$S_i^0:$	1	0	0	1	0	0	1	1
	$CO_i^0:$	0		1		1			
	$S_i^1:$	1	1	1	0	0	1		
	$CO_i^1:$	0		1		1			
Step 3	$S_i^0:$	1	1	0	1	0	0	1	1
	$CO_i^0:$	0				1			
	$S_i^1:$	1	1	1	0				
	$CO_i^1:$	0							

$$\tau = \Delta_{FA}$$

$$\tau = \Delta_{FA} + \Delta_{MUX}$$

$$\tau = \Delta_{FA} + 2\Delta_{MUX}$$

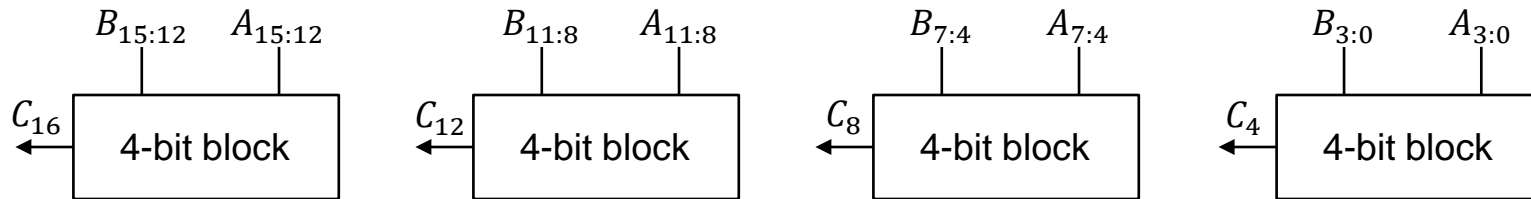
# Conditional Sum Adder

---

- Analysis
  - N-bit conditional sum adder
  - Delay of a 1-bit adder:  $\Delta_{FA}$
  - Delay of a MUX:  $\Delta_{MUX}$ 
    - Delay:  $\tau = \Delta_{FA} + (\log_2 N - 1) \cdot \Delta_{MUX}$ 
      - $\Delta_{FA} = 200ps, N = 64, \Delta_{MUX} = 80ps \rightarrow \tau = 600ps$

# Carry Skip Adder

- Split the input into a few groups.

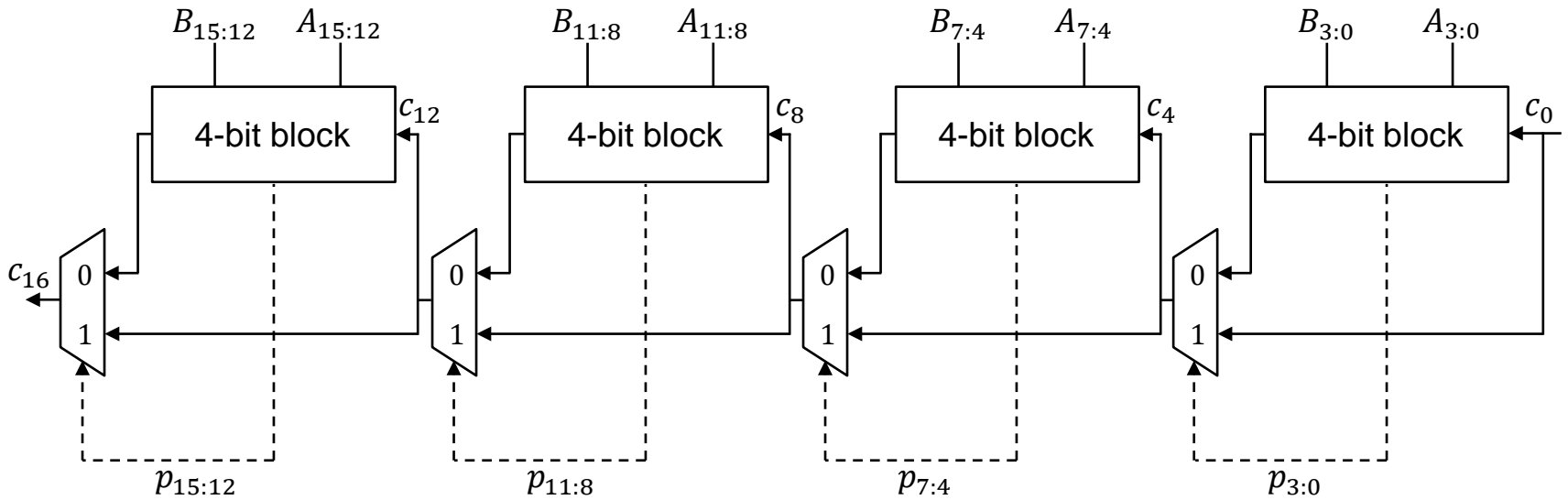


- Carry generation ( $C_4$ ,  $C_8$ ,  $C_{12}$ ,  $C_{16}$ , ...)
  - Generated internally
- Carry propagation
  - Propagated only when  $p$  is 1.
  - $p_i = A_i \oplus B_i$
  - $p_{i+3:i} = p_{i+3} \cdot p_{i+2} \cdot p_{i+1} \cdot p_i$



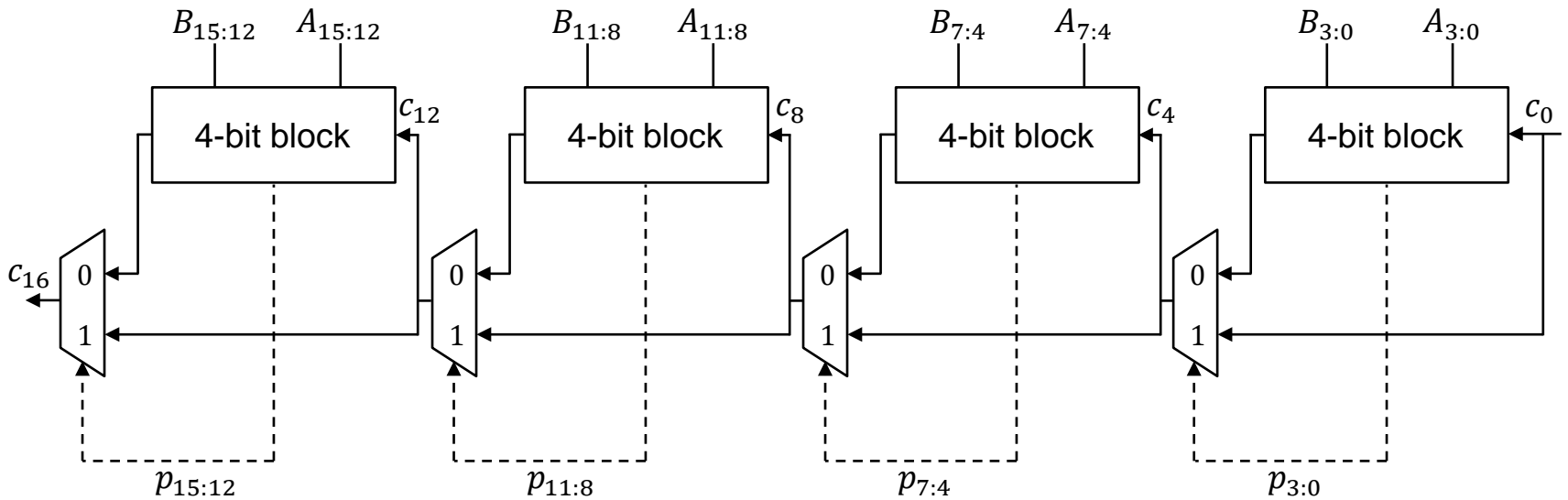
# Carry Skip Adder

- Split the input into a few groups.
- Propagate carries.



# Carry Skip Adder

- Exercise
  - $A = 1101101101111010$
  - $B = 0101010111010101$
  - $C_{I_0} = 1$



# Carry Skip Adder

---

- Analysis
  - N-bit carry skip adder
- Assumptions
  - The k-bit adder is a ripple-carry adder.
  - Delay of a k-bit adder:  $k \cdot \Delta_{FA}$
  - Delay of a MUX:  $\varepsilon$
- Delay:  $\tau = k \cdot \Delta_{FA} + \frac{N}{k} \cdot \varepsilon$

# Prefix Adder

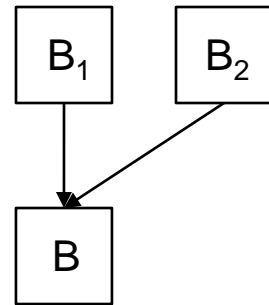
---

- Carry generate
  - $G_i = A_i \cdot B_i$
  - $G_i = 1$  only when  $(A_i, B_i) = (1,1)$
- Carry propagate
  - $P_i = A_i \oplus B_i$  (conceptually  $A_i \oplus B_i$ , practically  $A_i + B_i$ )
  - $P_i = 1$  only when  $(A_i, B_i) = (1,0)$  or  $(0,1)$
- Carry out
  - $C_{i+1} = A_i \cdot B_i + A_i \cdot C_i + B_i \cdot C_i = A_i \cdot B_i + C_i \cdot (A_i + B_i) = A_i \cdot B_i + C_i \cdot (A_i \oplus B_i) = G_i + C_i \cdot P_i$
- Sum
  - $S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$

# Prefix Adder

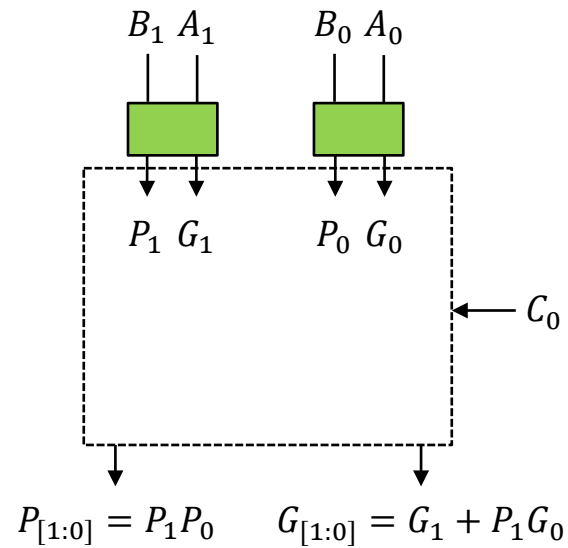
---


- Two blocks,  $B_1$  and  $B_2$  overlap.
- (generate, propagate) of  $B_1$ :  $(g_1, p_1)$
- (generate, propagate) of  $B_2$ :  $(g_2, p_2)$
  
- Merge  $B_1$  and  $B_2$  into  $B$ 
  - (generate, propagate) of  $B$ :  $(g, p)$
  - $g = g_1 + p_1 \cdot g_2$
  - $p = p_1 \cdot p_2$



# Prefix Adder

- Example



  $P_i$  and  $G_i$   
generation

# Prefix Adder

---

- Basic principle
  - Carry determination

Given

$$(g_0, p_0) \quad (g_1, p_1) \quad \dots \quad (g_{i-2}, p_{i-2}) \quad (g_{i-1}, p_{i-1})$$

Find

$$(g_{[0:0]}, p_{[0:0]}) \quad (g_{[1:0]}, p_{[1:0]}) \quad \dots \quad (g_{[i-2:0]}, p_{[i-2:0]}) \quad (g_{[i-1:0]}, p_{[i-1:0]})$$

- Sum

Given

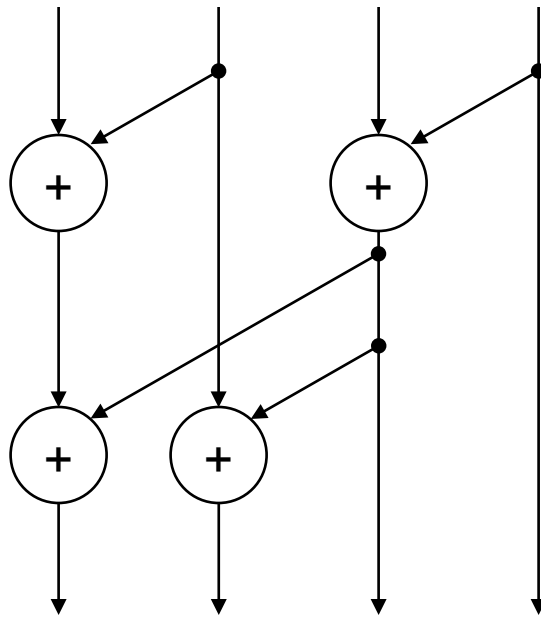
$$x_0 \quad x_1 \quad \dots \quad x_{i-2} \quad x_{i-1}$$

Find

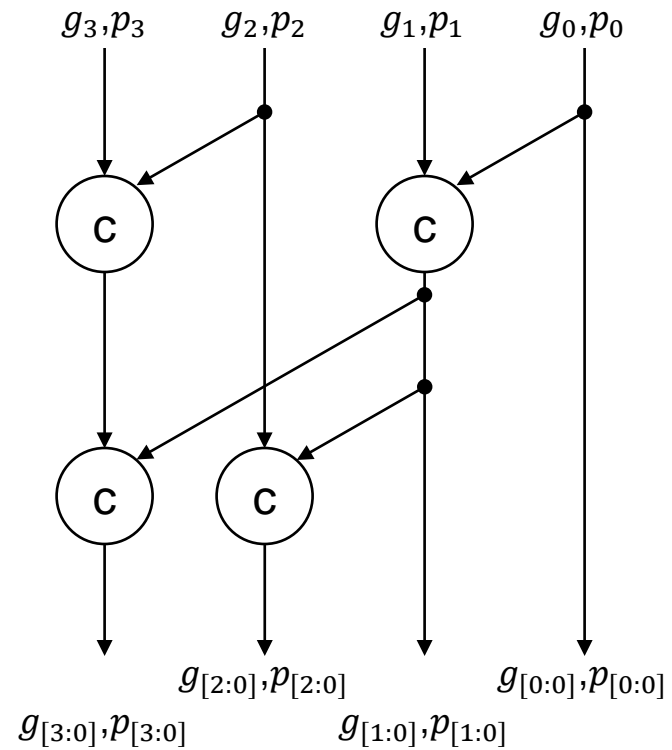
$$x_0 \quad x_0 + x_1 \quad \dots \quad x_0 + x_1 + \dots + x_{i-2} \quad x_0 + x_1 + \dots + x_{i-1}$$

# Prefix Adder

- Basic principle



Sum network



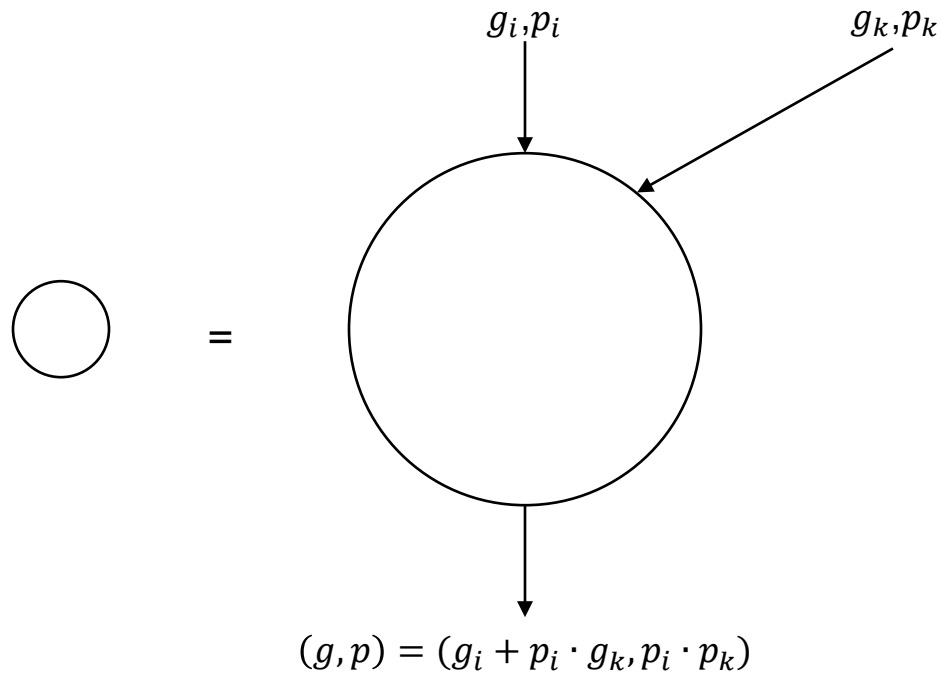
Carry network



# Prefix Adder

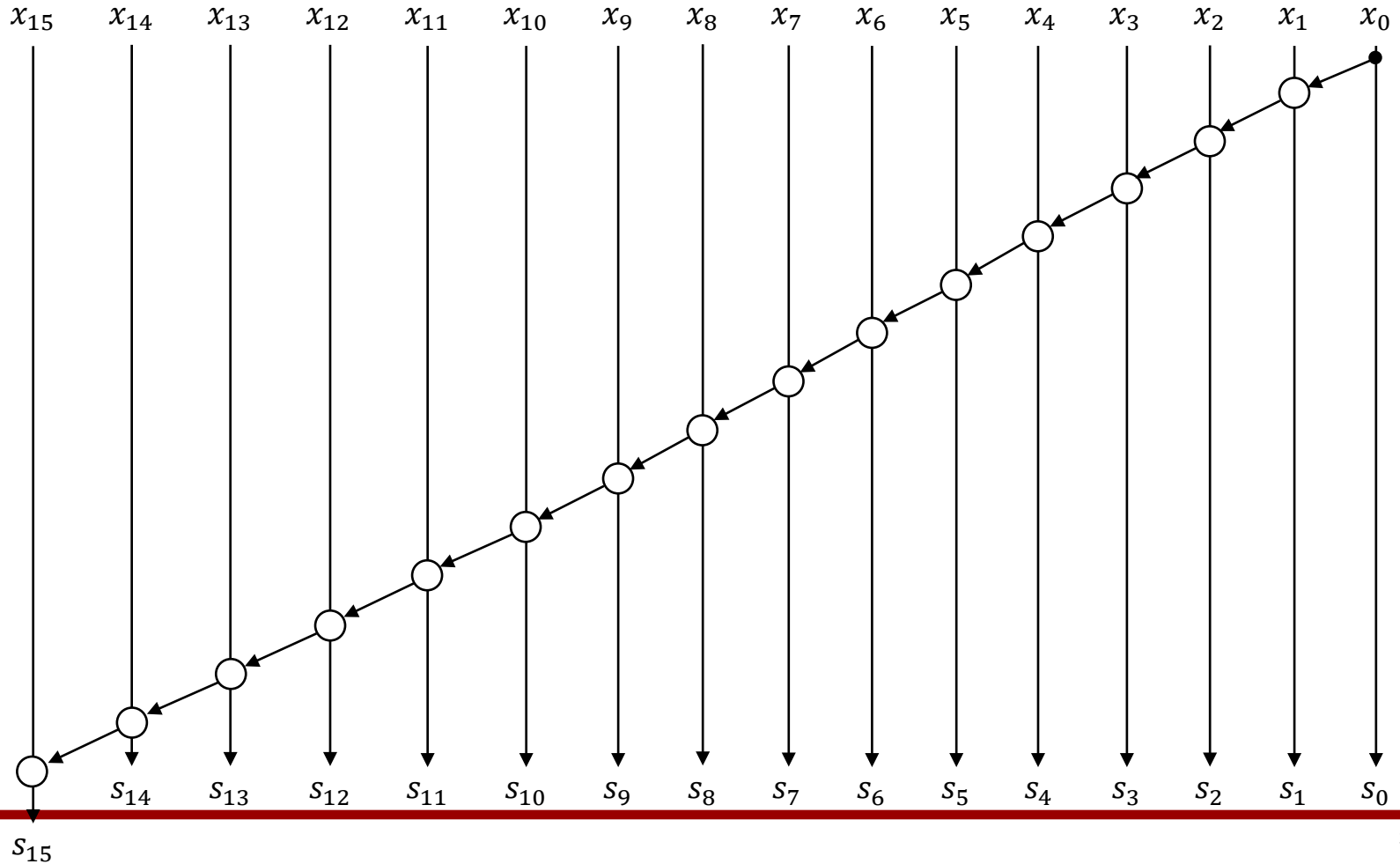
- $S_i = A_i \oplus B_i \oplus C_i$
- Propagate carries

$$\begin{aligned} G_i &= A_i \cdot B_i \\ P_i &= A_i \oplus B_i \\ S_i &= P_i \oplus C_i \\ C_{i+1} &= G_i + C_i \cdot P_i \end{aligned}$$



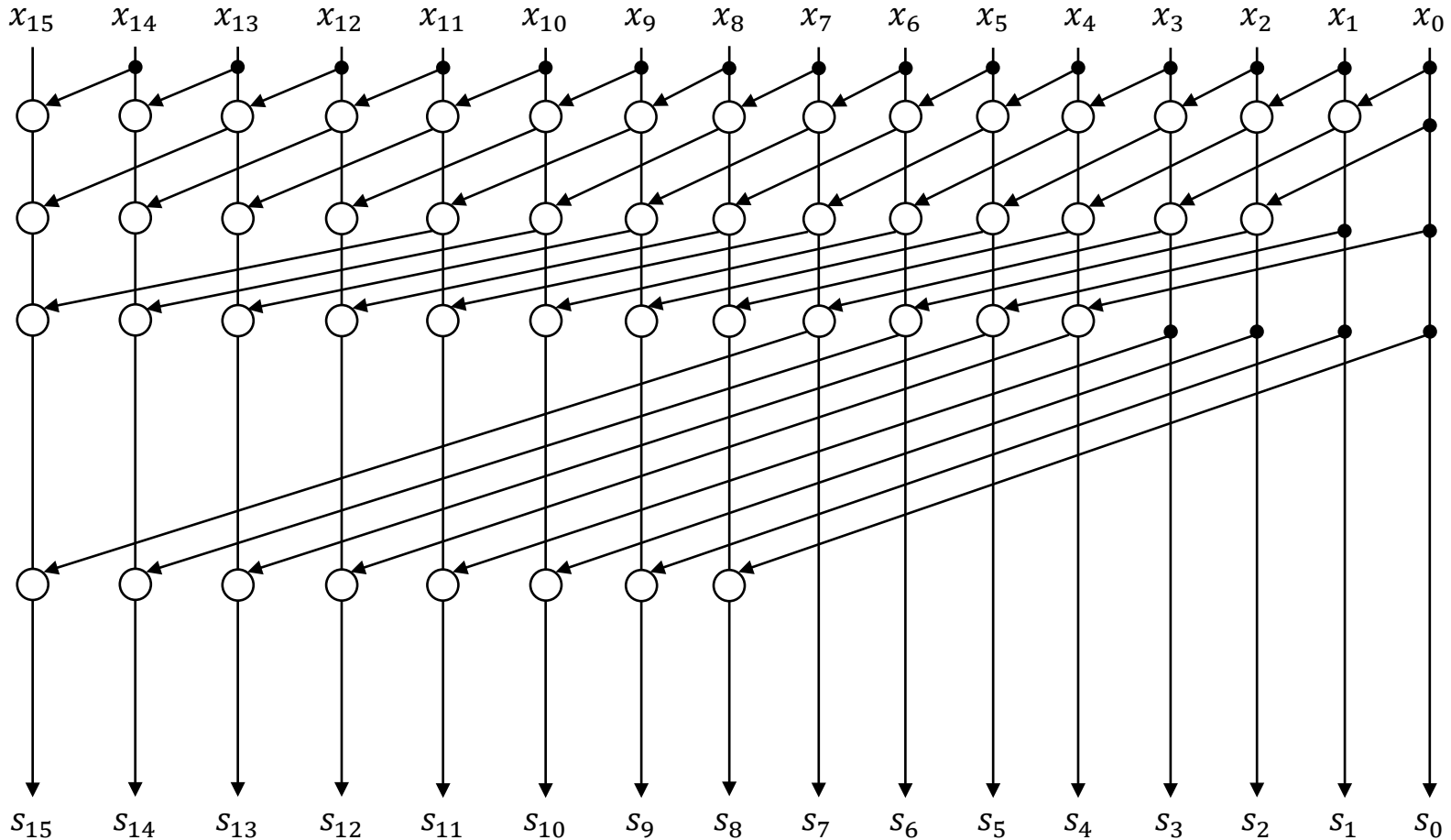
# Prefix Adder

- Example (ripple carry adder)



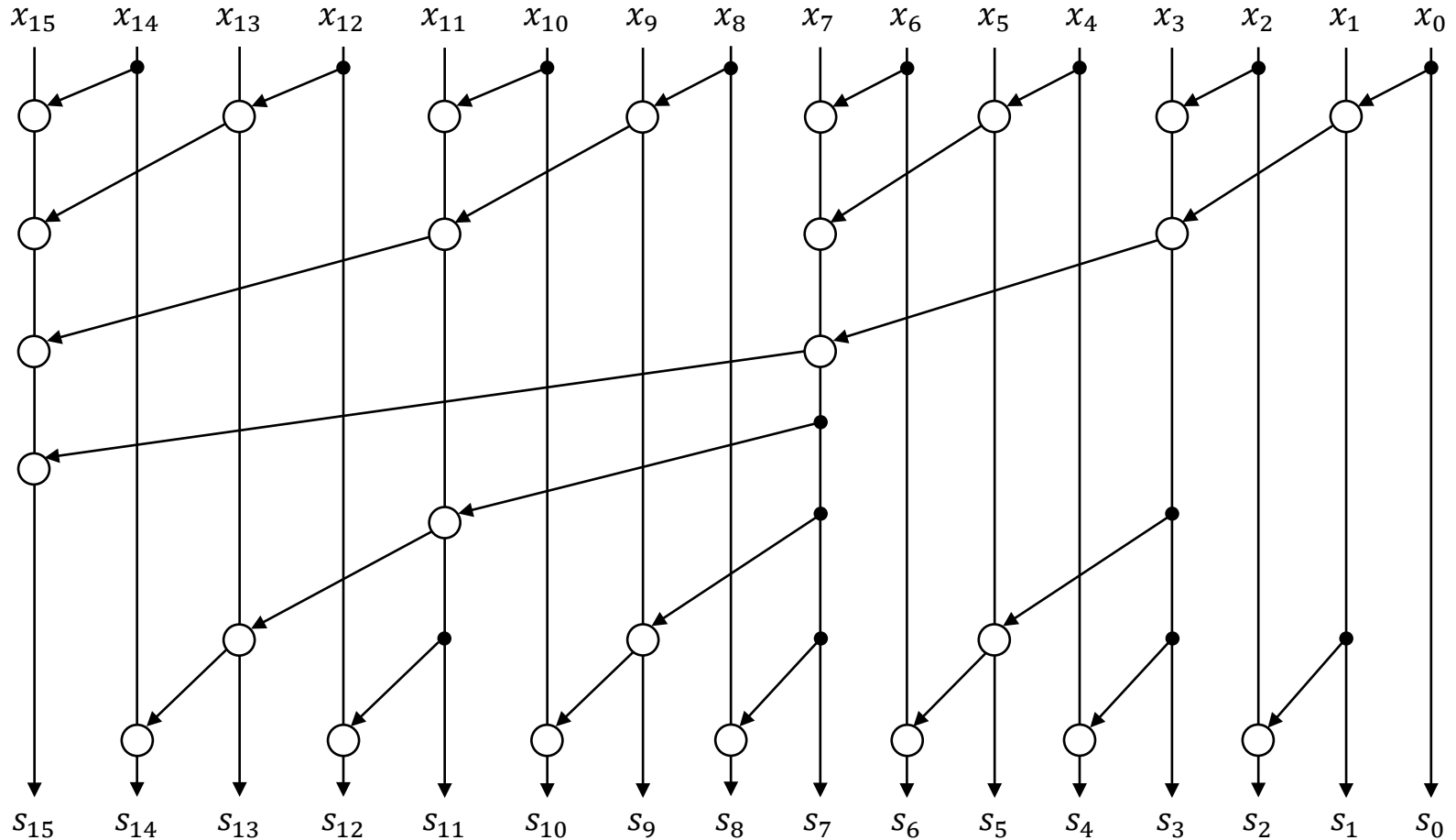
# Prefix Adder

- Kogge-Stone (min. logic depth, min. fanout, large area, routing)



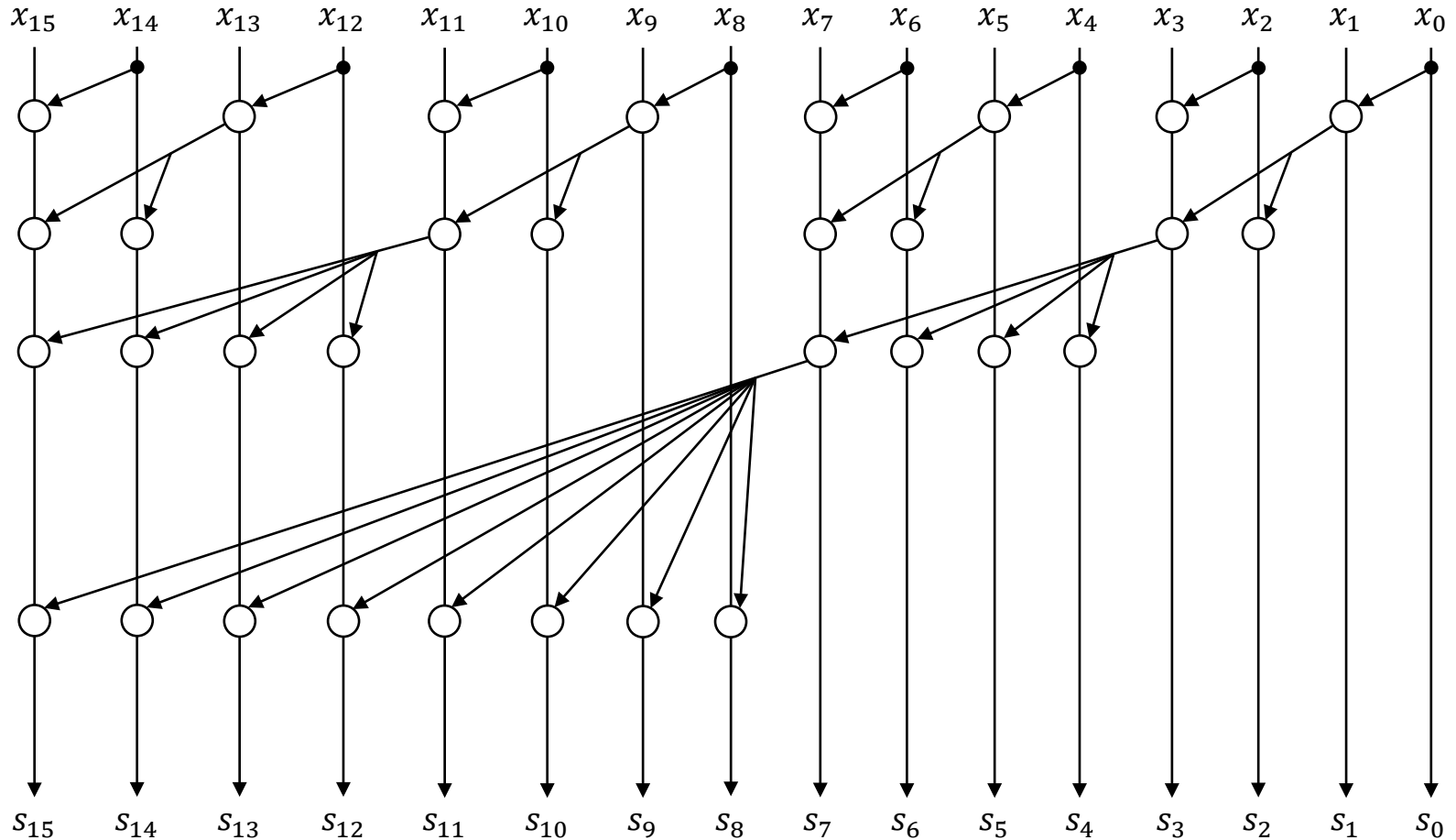
# Prefix Adder

- Brent-Kung (max. logic depth, min. area)



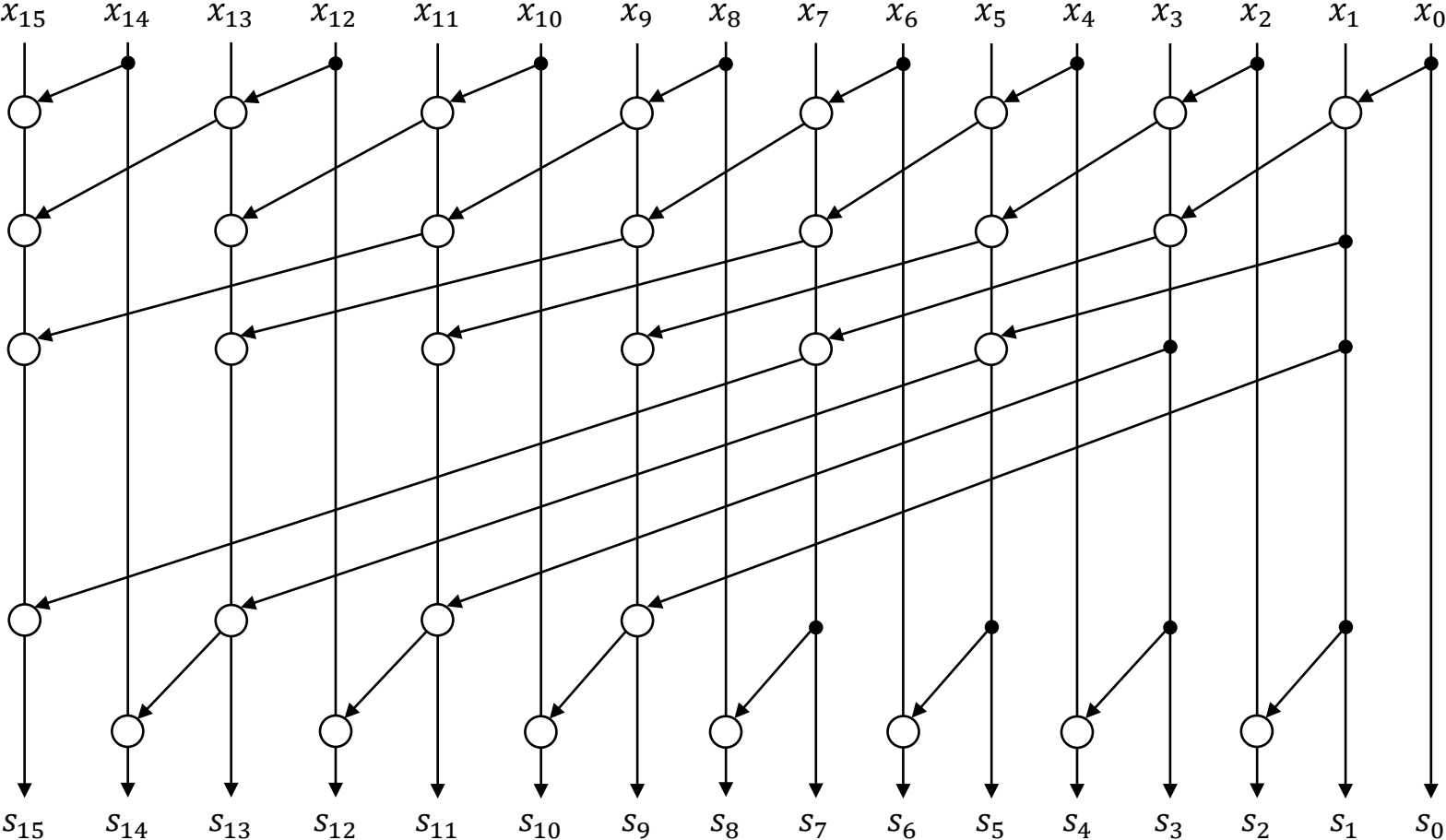
# Prefix Adder

- Ladner-Fischer (min. logic depth, large fanout)




# Prefix Adder

- Han-Carlson (Brent-Kung + Kogge-Stone)



# Prefix Adder

- Analysis
  - Generation of  $P_i$  and  $G_i$ :  $\Delta_c$
  - Merging:  $\Delta_m$  
  - Sum:  $\Delta_s$
  - Delay:  $\Delta_c + k \cdot \Delta_m + \Delta_s$
- Example (16-bit adder)
  - Ripple carry:  $\Delta_c + 15 \cdot \Delta_m + \Delta_s$
  - Kogge-Stone:  $\Delta_c + \log_2 16 \cdot \Delta_m + \Delta_s = \Delta_c + 4 \cdot \Delta_m + \Delta_s$
  - Brent-Kung:  $\Delta_c + 6 \cdot \Delta_m + \Delta_s$
  - Ladner-Fischer:  $\Delta_c + \log_2 16 \cdot \Delta_m + \Delta_s = \Delta_c + 4 \cdot \Delta_m + \Delta_s$
  - Han-Carlson:  $\Delta_c + 5 \cdot \Delta_m + \Delta_s$

# Carry Lookahead Adder

---

- Carry generate
  - $G_i = A_i \cdot B_i$
  - $G_i = 1$  only when  $(A_i, B_i) = (1,1)$
- Carry propagate
  - $P_i = A_i \oplus B_i$
  - $P_i = 1$  only when  $(A_i, B_i) = (1,0)$  or  $(0,1)$
- Sum
  - $S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$
- Carry out
  - $C_{i+1} = A_i \cdot B_i + A_i \cdot C_i + B_i \cdot C_i = A_i \cdot B_i + C_i \cdot (A_i + B_i) = A_i \cdot B_i + C_i \cdot (A_i \oplus B_i) = G_i + C_i \cdot P_i$



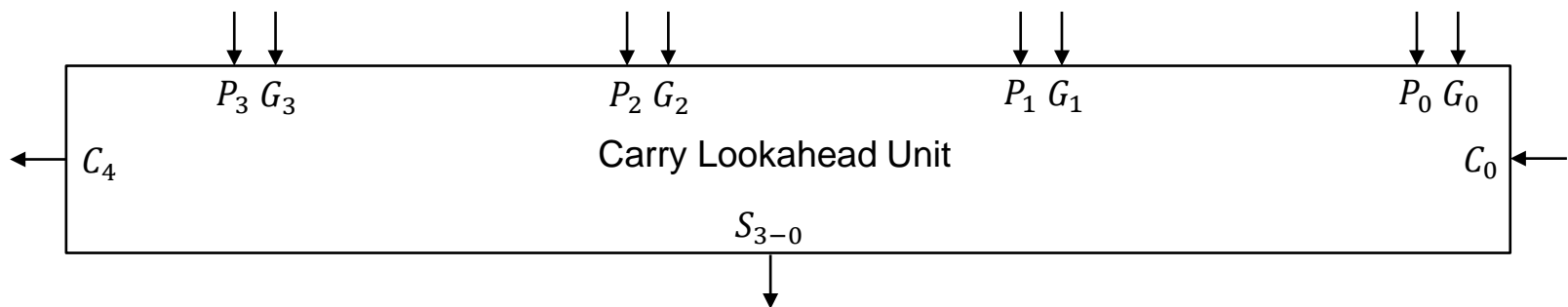
# Carry Lookahead Adder

- Carry-out logic

- $C_1 = G_0 + P_0 C_0$
- $C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$
- $C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
- $C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

$$\begin{aligned} G_i &= A_i \cdot B_i \\ P_i &= A_i \oplus B_i \\ S_i &= P_i \oplus C_i \\ C_{i+1} &= G_i + C_i \cdot P_i \end{aligned}$$

- Carry lookahead unit



# Carry Lookahead Adder

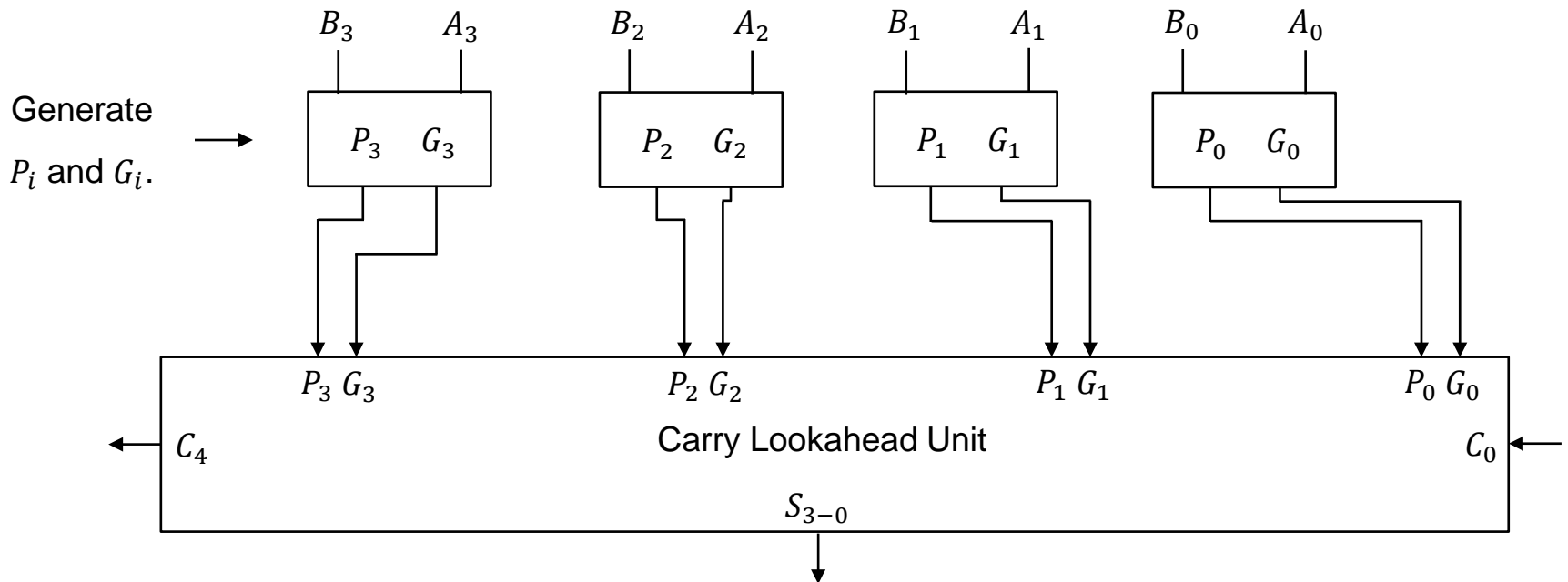
- Delay computation

- $G_i, P_i$ : 1 gate delay
- $S_0$ : 2 gate delays
- $C_1 = G_0 + P_0C_0$ : 3 gate delays
- $C_2 = G_1 + P_1G_0 + P_1P_0C_0$ : 3 gate delays
- $C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$ : 3 gate delays
- $C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$ : 3 gate delays
- $S_1, S_2, S_3$ : 4 gate delays

$$\begin{aligned}G_i &= A_i \cdot B_i \\P_i &= A_i \oplus B_i \\S_i &= P_i \oplus C_i \\C_{i+1} &= G_i + C_i \cdot P_i\end{aligned}$$

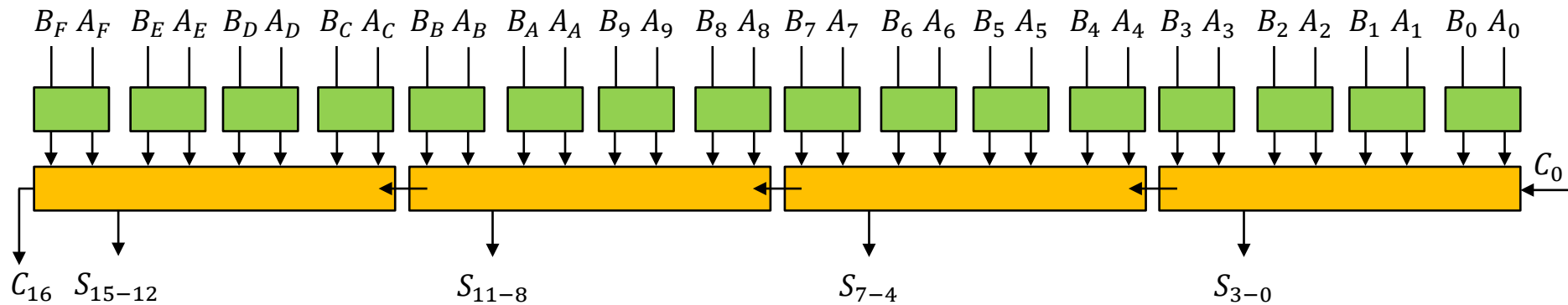
# Carry Lookahead Adder


- A 4-bit carry lookahead adder




# Carry Lookahead Adder

- A 16-bit carry lookahead adder

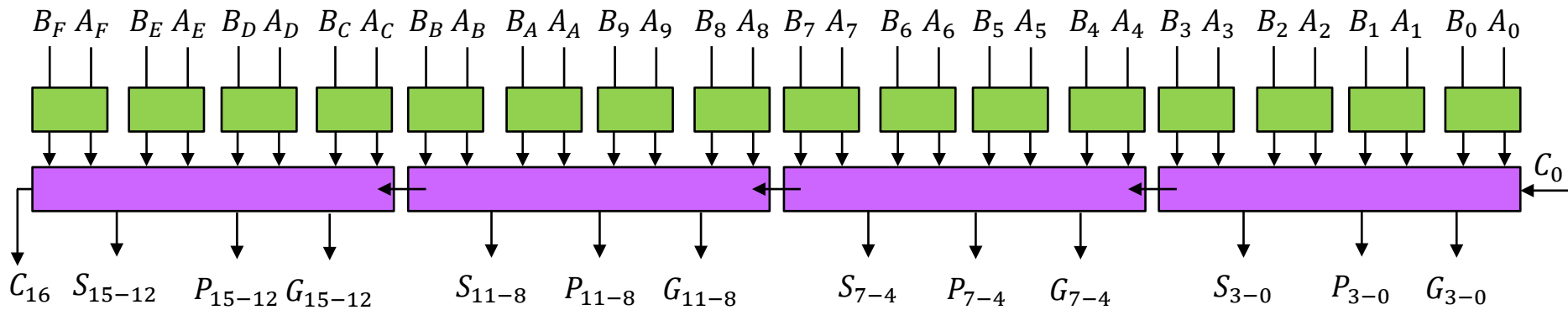



  $P_i$  and  $G_i$  generation


 4-bit carry lookahead unit

# Carry Lookahead Adder

- Multi-level carry lookahead adder



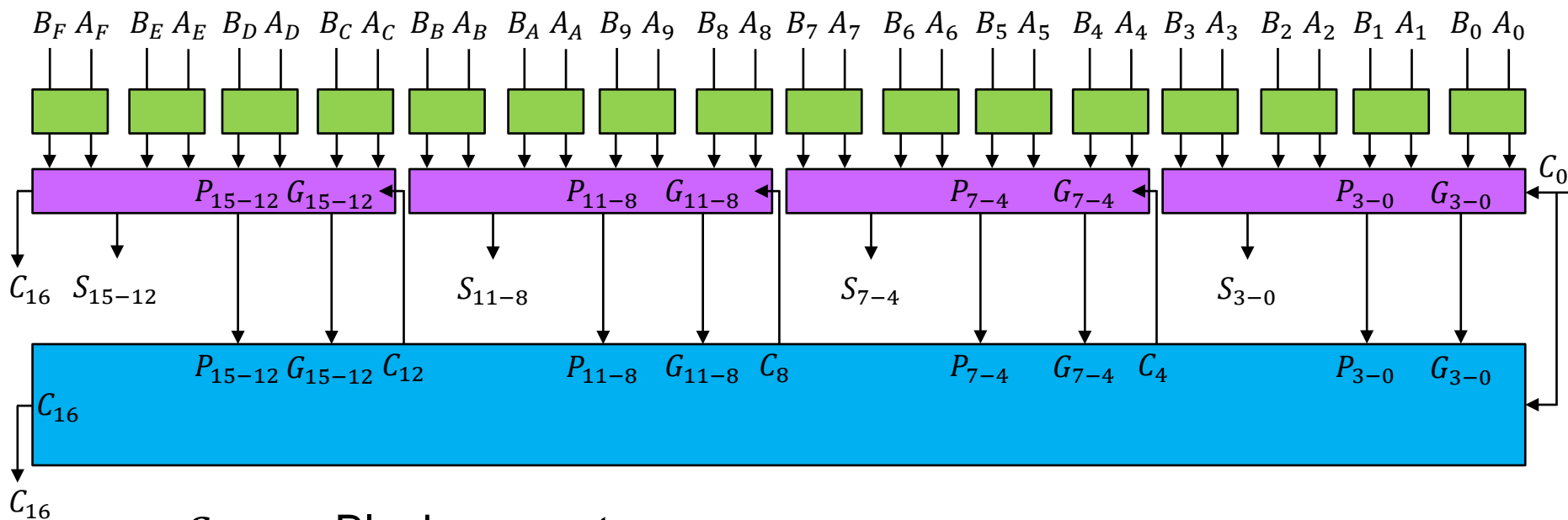
  $P_i$  and  $G_i$  generation

 4-bit carry lookahead unit with  $P_i'$  and  $G_i'$ .


 Multi-level carry lookahead unit


# Carry Lookahead Adder

- Multi-level carry lookahead adder



- $G_{[i+3,i]}$ : Block generate
- $P_{[i+3,i]}$ : Block propagate

  $P_i$  and  $G_i$  generation

 4-bit carry lookahead unit with  $P_i'$  and  $G_i'$ .

 Multi-level carry lookahead unit

# Carry Lookahead Adder

---

- Carry recurrence formula

- $C_{i+1} = G_i + P_i \cdot C_i$

- $C_{i+4} = G_{i+3} + P_{i+3} \cdot C_{i+3}$

- $= G_{i+3} + P_{i+3} \cdot (G_{i+2} + P_{i+2} \cdot C_{i+2}) = G_{i+3} + P_{i+3} \cdot G_{i+2} + P_{i+3} \cdot P_{i+2} \cdot C_{i+2}$

- $= \dots$

- $= G_{i+3} + P_{i+3} \cdot G_{i+2} + P_{i+3} \cdot P_{i+2} \cdot G_{i+1} +$

- $P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot G_i + P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot P_i \cdot C_i$

- Block generate

- $G_{[i+3,i]} = G_{i+3} + P_{i+3} \cdot G_{i+2} + P_{i+3} \cdot P_{i+2} \cdot G_{i+1} + P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot G_i$

- Block propagate

- $P_{[i+3,i]} = P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot P_i$

- Carry out

- $C_{i+4} = G_{[i+3,i]} + C_i \cdot P_{[i+3,i]}$

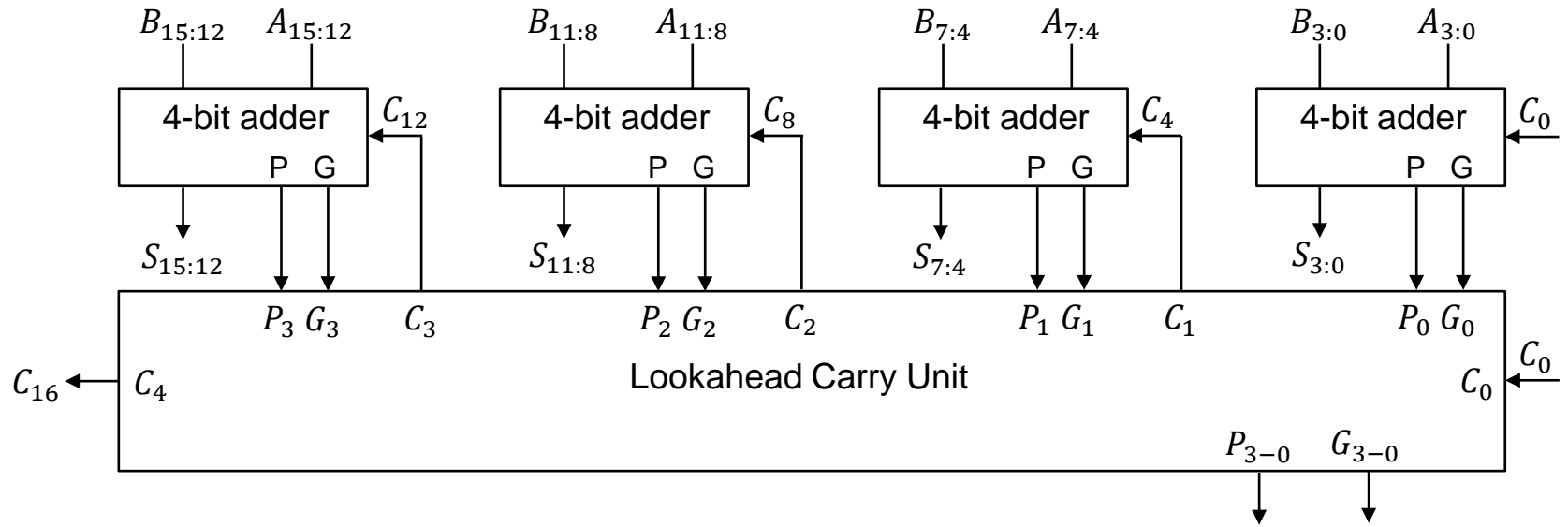
# Carry Lookahead Adder

---

- Delay computation
  - $G_i, P_i$ : 1 gate delay
  - Block propagate
    - $P_{[i+3,i]} = P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot P_i$ : 2 gate delays
  - Block generate
    - $G_{[i+3,i]} = G_{i+3} + P_{i+3} \cdot G_{i+2} + P_{i+3} \cdot P_{i+2} \cdot G_{i+1} + P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot G_i$ : 3 gate delays
  - Carry out
    - $C_4 = G_{[3,0]} + C_0 \cdot P_{[3,0]}$ : 4 gate delays
    - $C_8 = G_{[7,4]} + C_4 \cdot P_{[7,4]} = G_{[7,4]} + P_{[7,4]} \cdot G_{[3,0]} + P_{[7,4]} \cdot P_{[3,0]} \cdot C_0$ : 5 gate delays
    - $C_{12}$ : 5 gate delays
    - $C_{16}$ : 5 gate delays
  - Sum
    - $S_{[7,4]}$ : 7 gate delays
    - $S_{[11,8]}$ : 8 gate delays
    - $S_{[15,12]}$ : 8 gate delays



# Carry Lookahead Adder



# Ling Adder

- A type of carry lookahead adder
- Achieves significant hardware savings.
- Modifications

$$- P_i = A_i \oplus B_i \rightarrow t_i = A_i + B_i$$

$$- h_i = C_i + C_{i-1}$$

$$- C_i = G_{i-1} + C_{i-1} \cdot P_{i-1}$$

$$\begin{aligned} - C_{i-1} \cdot P_{i-1} &= C_{i-1} \cdot P_{i-1} + \overset{=0}{G_{i-1} \cdot P_{i-1}} + \overset{\text{repeated}}{P_{i-1} \cdot C_{i-1} \cdot P_{i-1}} \\ &= C_{i-1} \cdot P_{i-1} + (G_{i-1} + P_{i-1} \cdot C_{i-1}) \cdot P_{i-1} \\ &= (C_{i-1} + C_i) \cdot P_{i-1} \quad = C_i \\ &= h_i \cdot P_{i-1} \end{aligned}$$

# Ling Adder

- Modifications

- $t_i = A_i + B_i$

- $h_i = C_i + C_{i-1}$

- $G_{i-1} \Rightarrow h_i \cdot G_{i-1}$

- $C_i = G_{i-1} + C_{i-1} \cdot P_{i-1} = G_{i-1} + h_i \cdot P_{i-1} = h_i \cdot G_{i-1} + h_i \cdot P_{i-1}$   
 $= h_i \cdot (G_{i-1} + P_{i-1}) = h_i \cdot t_{i-1}$

- $h_i = C_i + C_{i-1} = (G_{i-1} + C_{i-1} \cdot P_{i-1}) + C_{i-1} = G_{i-1} + C_{i-1}$   
 $= G_{i-1} + h_{i-1} \cdot t_{i-2}$

- $h_i = G_{i-1} + h_{i-1} \cdot t_{i-2}$   
 $= G_{i-1} + t_{i-2} \cdot (G_{i-2} + h_{i-2} \cdot t_{i-3})$

- $= G_{i-1} + G_{i-2} + h_{i-2} \cdot t_{i-2} \cdot t_{i-3} \quad \longleftarrow \boxed{t_{i-2} \cdot G_{i-2} = G_{i-2}}$

- $= G_{i-1} + G_{i-2} + G_{i-3} \cdot t_{i-2} + G_{i-4} \cdot t_{i-2} \cdot t_{i-3} + h_{i-4} \cdot t_{i-2} \cdot t_{i-3} \cdot t_{i-4}$

# Ling Adder

---

- Modifications

- $C_{i+4} = G_{i+3} + G_{i+2} \cdot t_{i+3} + G_{i+1} \cdot t_{i+3} \cdot t_{i+2} + G_i \cdot t_{i+3} \cdot t_{i+2} \cdot t_{i+1} + C_i \cdot t_{i+3} \cdot t_{i+2} \cdot t_{i+1} \cdot t_i$
- $S_i = t_i \oplus h_{i+1} + G_i \cdot t_{i-1} \cdot h_i$

- Advantages

- Lower fan-in
- Suitable for some specific technologies.