Arithmetic Circuits

High-Speed Adders

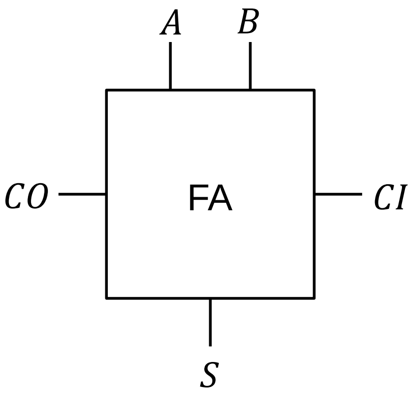
Dae Hyun Kim

EECS
Washington State University

Full Adder

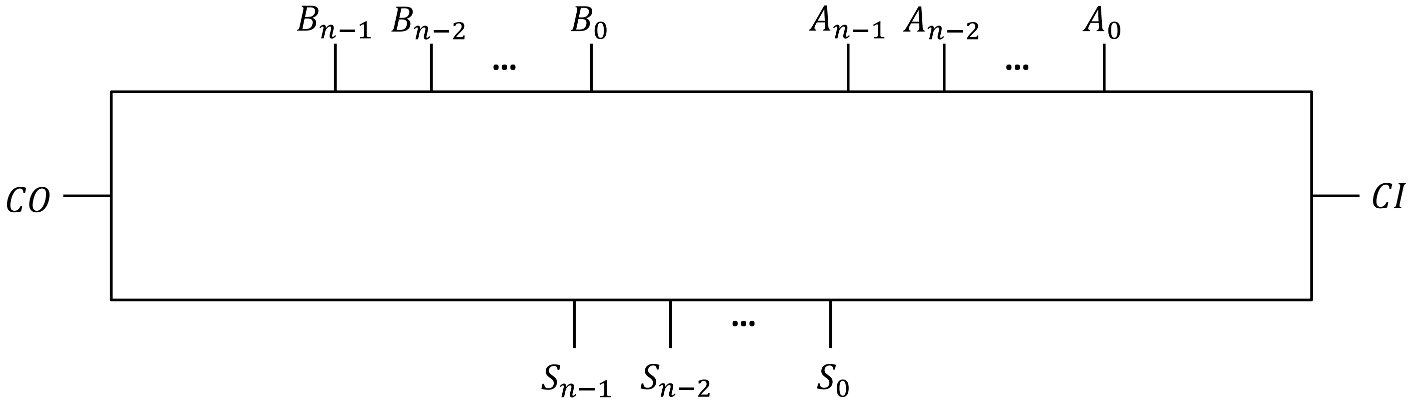
- Input: A, B, CI (Carry-in)
- Output: S, CO (Carry-out)
  - $S = A \oplus B \oplus CI$
  - $CO = A \cdot B + B \cdot CI + CI \cdot A = A \cdot B + CI \cdot (A + B)$

A	B	CI	S	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



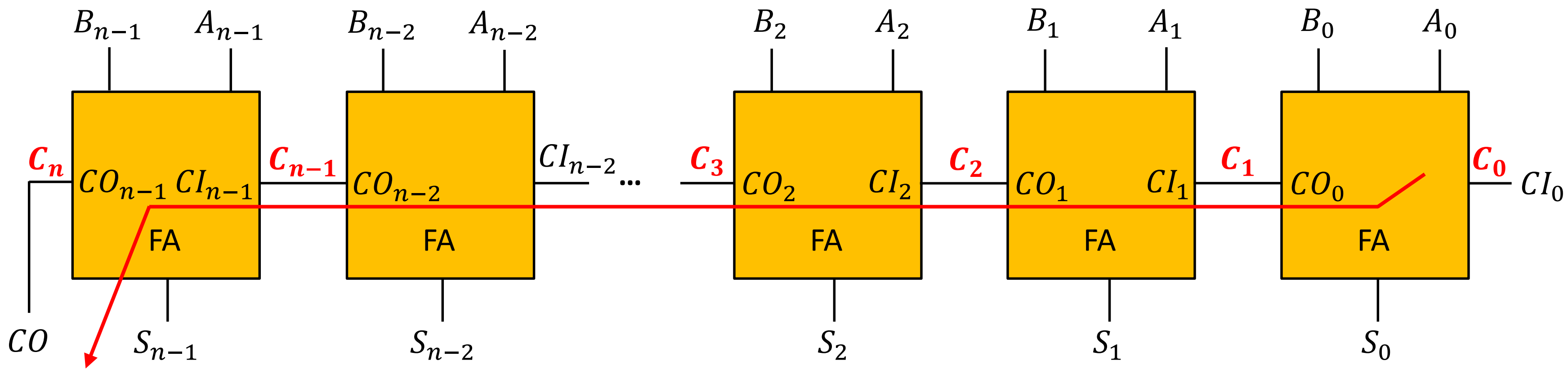
Two-Operand n-bit Adder

- Input: $A[n-1:0], B[n-1:0], CI$
- Output: $S[n-1:0], CO$
- For two's complement subtraction, there could be a control signal s.
  - $s = 0$: addition ($A + B$)
    - In this case, we compute $A + B$, so $CI = 0$.
  - $s = 1$: subtraction ($A - B$)
    - In this case, we compute $A + \bar{B} + 1$, so B is inverted and $CI = 1$.

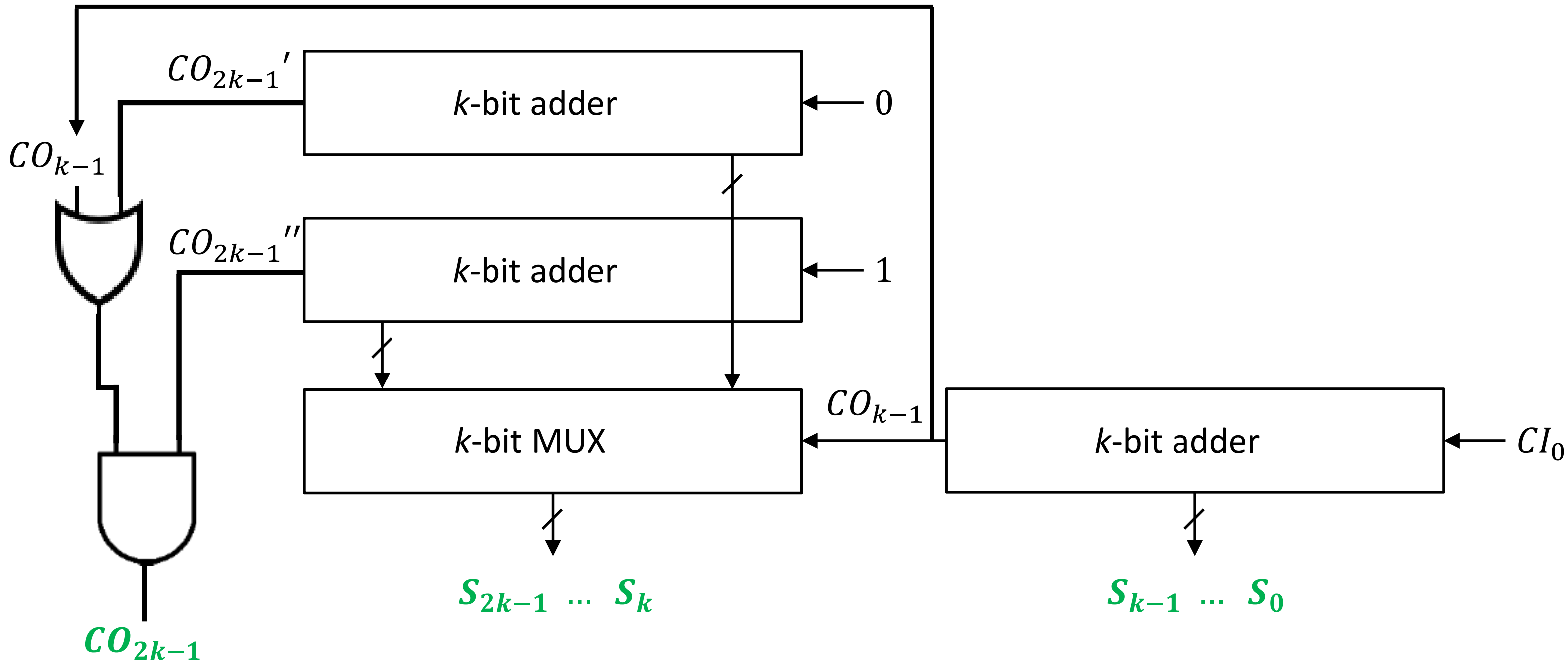


n-bit Ripple Carry Adder (delay: $O(n)$)

- C_k: Carry
- Delay computation
  - FA delay: d
  - n-bit RCA delay: $n \cdot d$
    - $O(n)$
  - Example
    - $n: 64, d: 50ps$ (45nm) → delay = 3.2ns (Clock: 312.5MHz)
- Bottleneck: Carry propagation
- Solution: Parallelization



$2k$-bit Carry Select Adder

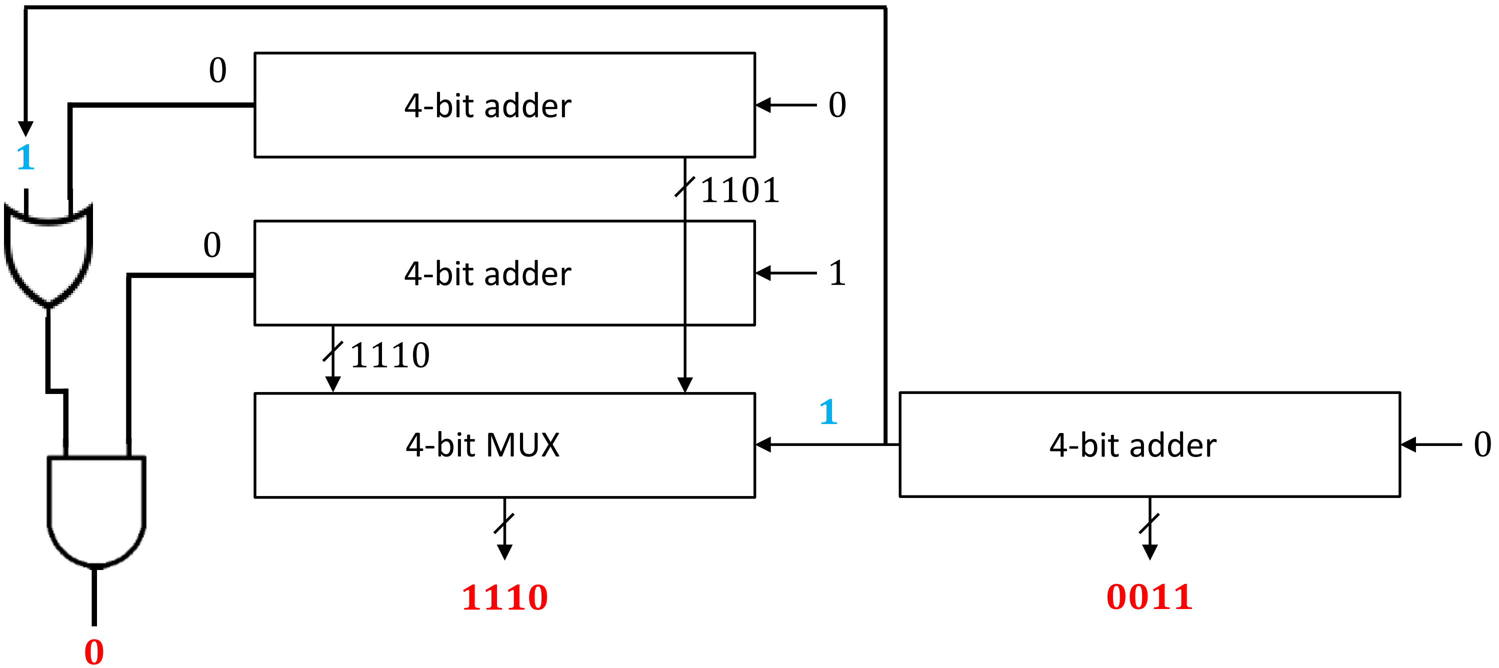


$2k$-bit Carry Select Adder

- Example

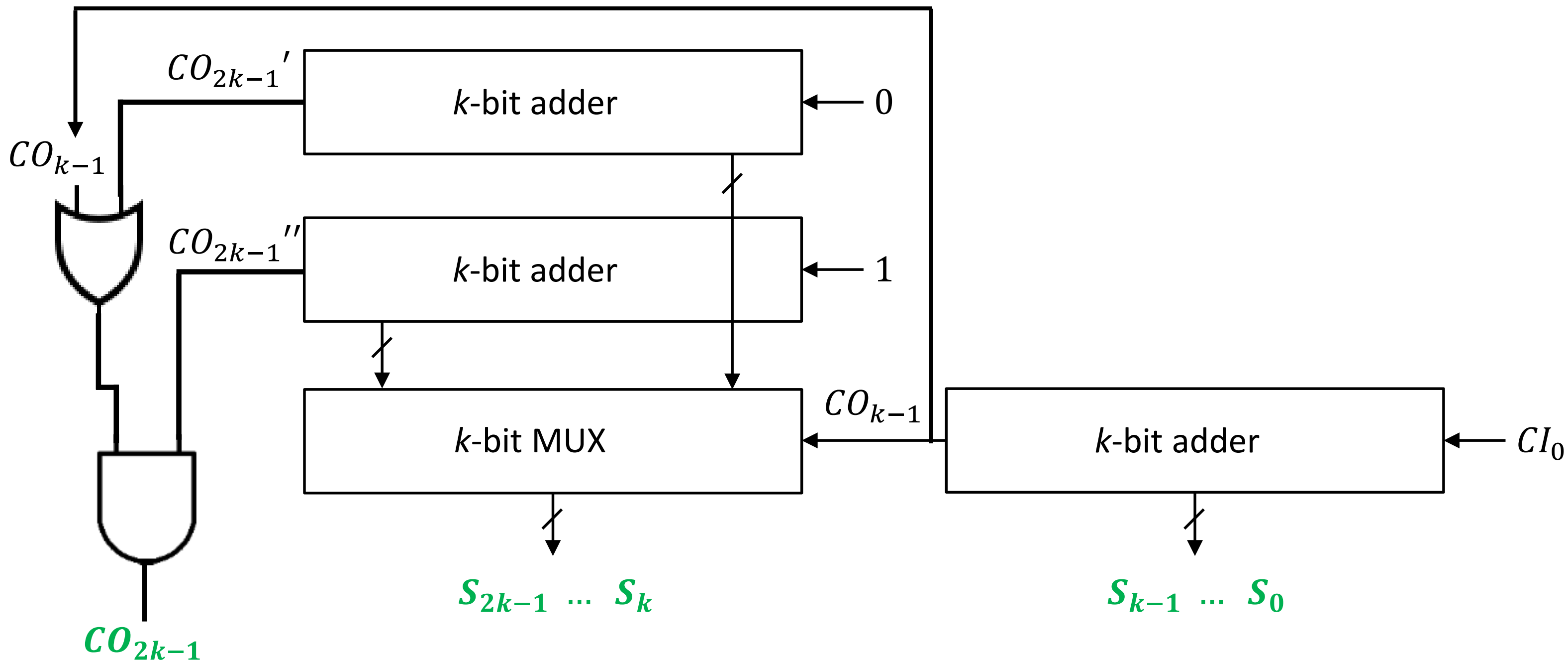
i:	7	6	5	4	3	2	1	0
A_i:	1	0	1	1	0	1	1	0
B_i:	0	0	1	0	1	1	0	1

$CI_0 = 0$



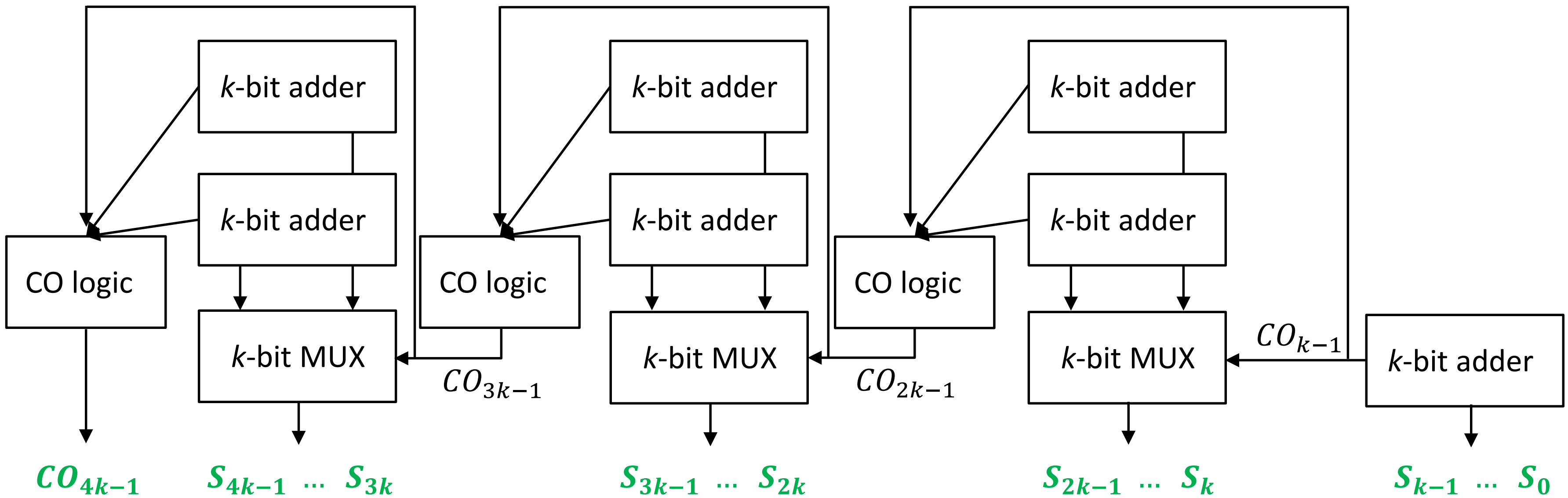
$2k$-bit Carry Select Adder

- Delay analysis
  - Delay of a k-bit adder + delay of (a MUX or two gates)



Carry Select Adder

- Parallelization
  - Example: $4k$-bit carry select adder
    - Delay: Delay of a k-bit adder + 2*(CO logic delay) + (CO logic delay or a MUX delay)



$\boldsymbol{n}$-bit Carry Select Adder (delay: $\boldsymbol{O(\sqrt{n})}$)

- Optimization
  - Find the optimal # stages minimizing the total delay.
- Assumption
  - Delay of a full-adder (FA): d
  - The k-bit adder is a ripple-carry adder.
    - Delay: $k \cdot d$
  - Delay of a carry-out logic or a MUX: e
- # stages: $\frac{n}{k}$ (find k minimizing the delay)
- Delay: $\tau = k \cdot d + e \cdot \left(\frac{n}{k} - 1\right)$
  - $\frac{d\tau}{dk} = d - \frac{e \cdot n}{k^2} = 0$
  - $k = \sqrt{\frac{e \cdot n}{d}}$ (so the optimal # stages is $\frac{n}{k} = \sqrt{\frac{d \cdot n}{e}}$)
  - Delay: $\tau = 2\sqrt{d \cdot e \cdot n} - e \rightarrow O(\sqrt{n})$

$\boldsymbol{n}$-bit Carry Select Adder (delay: $\boldsymbol{O}(\sqrt{\boldsymbol{n}})$)

- Example
  - $n: 64, d: 50ps, e: 50ps$
  - $k = \sqrt{\frac{50 \cdot 64}{50}} = 8$ (8-bit ripple-carry adder)
  - Delay: $\tau = 2\sqrt{50 \cdot 50 \cdot 64} - 50 = 750ps$ (4.3X improvement)
  - Delay of a 64-bit RCA: $64 \cdot 50 = 3{,}200ps$

Carry Signals

- $C_k = (A_{k-1} \cdot B_{k-1}) + (A_{k-1} \oplus B_{k-1}) \cdot C_{k-1}$

- $C_1 = (A_0 \cdot B_0) + (A_0 \oplus B_0) \cdot C_0$
- $C_2 = (A_1 \cdot B_1) + (A_1 \oplus B_1) \cdot C_1 = [(A_1 \cdot B_1) + (A_1 \oplus B_1) \cdot (A_0 \cdot B_0)] + [(A_1 \oplus B_1) \cdot (A_0 \oplus B_0) \cdot C_0]$
- $C_3 = (A_2 \cdot B_2) + (A_2 \oplus B_2) \cdot C_2 =$
  $[(A_2 \cdot B_2) + (A_2 \oplus B_2) \cdot (A_1 \cdot B_1) + (A_2 \oplus B_2) \cdot (A_1 \oplus B_1) \cdot (A_0 \cdot B_0)] +$
  $[(A_2 \oplus B_2) \cdot (A_1 \oplus B_1) \cdot (A_0 \oplus B_0) \cdot C_0]$
- $C_4 = (A_3 \cdot B_3) + (A_3 \oplus B_3) \cdot C_3 =$
  $[(A_3 \cdot B_3) + (A_3 \oplus B_3) \cdot (A_2 \cdot B_2) + (A_3 \oplus B_3) \cdot (A_2 \oplus B_2) \cdot (A_1 \cdot B_1) + (A_3 \oplus B_3) \cdot (A_2 \oplus B_2) \cdot (A_1 \oplus B_1) \cdot (A_0 \cdot B_0)] +$
  $[(A_3 \oplus B_3) \cdot (A_2 \oplus B_2) \cdot (A_1 \oplus B_1) \cdot (A_0 \oplus B_0) \cdot C_0]$

- Black: generated only by A_k and B_k.
- Green: generated by A_k, B_k, and C_0.

- Generation: $g_i = A_i \cdot B_i$
- Propagation: $p_i = A_i \oplus B_i$

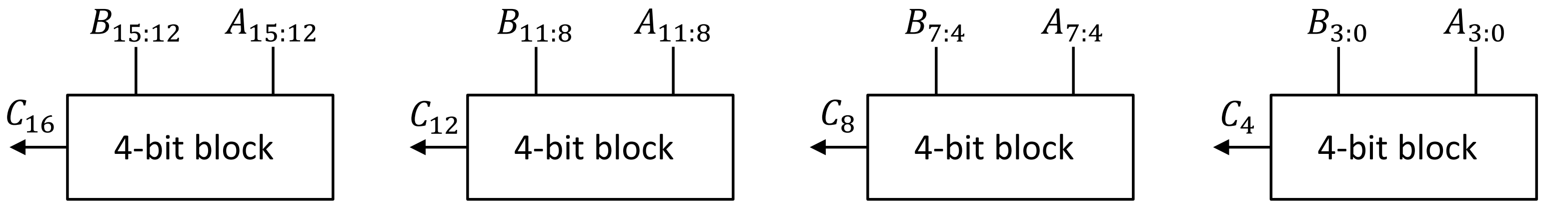
- $C_1 = g_0 + p_0 \cdot C_0$
- $C_2 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0$
- $C_3 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0$
- $C_4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0$

Carry Skip Adder

- Carry (C_4)
  - $C_4 = \boxed{g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0} + \boxed{p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0}$

  Group carry generation — Group carry propagation

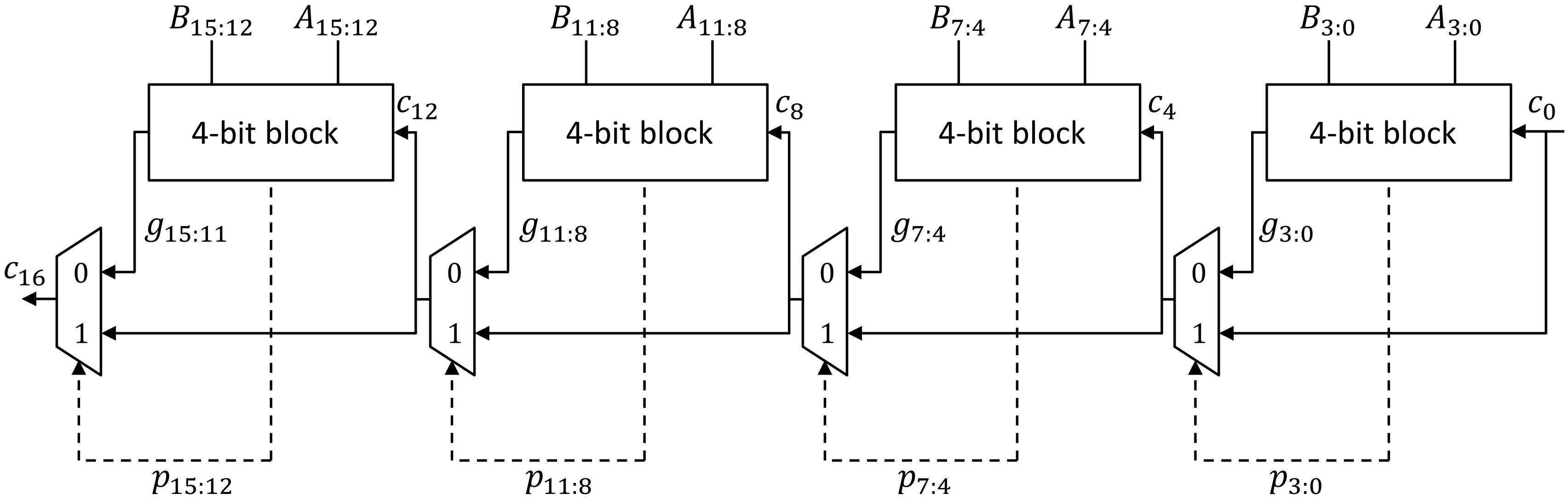
- Split the input into a few groups.



- Group carry generation
  - $g_{3:0} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0.$
- Group carry propagation
  - $p_{3:0} = p_3 \cdot p_2 \cdot p_1 \cdot p_0$
- Carry generation
  - $C_4 = g_{3:0} + p_{3:0} \cdot C_0$

Carry Skip Adder

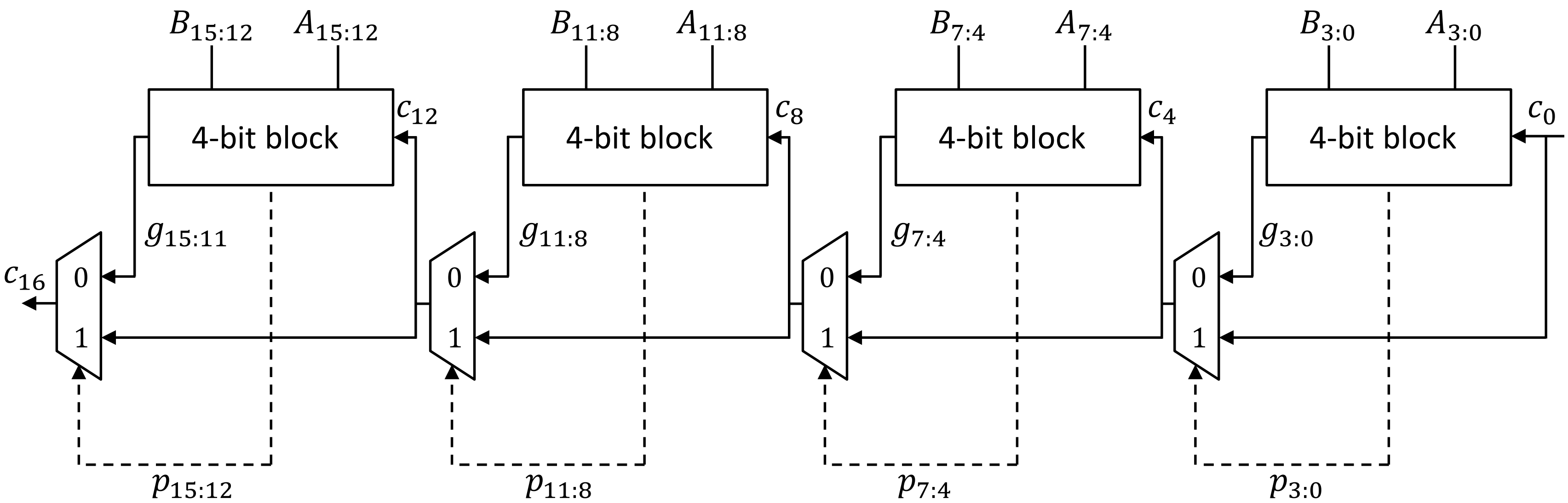
- 4-bit blocks
  - Generates $g_{k:k-3}$ and $p_{k:k-3}$ all in parallel.
  - Propagate C_0 to the last stage quickly.



- If $g_{k:k-3}$ is 1, $p_{k:k-3}$ is 0, so $g_{k:k-3}$ is passed to the next stage (generated).
- If $g_{k:k-3}$ is 0 and $p_{k:k-3}$ is 1, C_k is passed to the next stage (propagated).
- If $g_{k:k-3}$ is 0 and $p_{k:k-3}$ is 0, 0 is passed to the next stage (killed).

Carry Skip Adder

- Example
  - $A = 1101\ 1011\ 0111\ 1010$
  - $B = 0101\ 0101\ 1101\ 0101$
  - $C_0 = 1$



$\boldsymbol{n}$-bit Carry Skip Adder (delay: $\boldsymbol{O(\sqrt{n})}$)

- Optimization
  - Find the optimal # stages minimizing the total delay.
- Assumption
  - Delay of a full-adder (FA): d
  - The k-bit adder is a ripple-carry adder.
    - Delay: $k \cdot d$
  - Delay of a gate or a MUX: e
- # stages: $\frac{n}{k}$ (find k minimizing the delay)
- Delay
  - g_i, p_i: e
  - $p_{i:i-3}$: $2e$
  - $g_{i:i-3}$: $3e$ (two-stage logic, AND-OR)
  - Delay: $\tau = 3e + \left(\frac{n}{k} - 1\right) \cdot e + k \cdot d$
  - $\frac{d\tau}{dk} = d - \frac{e \cdot n}{k^2} = 0$
  - $k = \sqrt{\frac{e \cdot n}{d}}$ (so the optimal # stages is $\frac{n}{k} = \sqrt{\frac{d \cdot n}{e}}$)
  - Delay: $\tau = 2\sqrt{d \cdot e \cdot n} + 2e \rightarrow O(\sqrt{n})$
  - Example: 900ps for $n = 64, d = e = 50ps$

n-bit Conditional Sum Adder (Delay: $O(\log_2 n)$)

- Main idea
  - Generate two sets of outputs for a given group of operand bits.
    - Set 1: assuming the incoming carry is zero.
    - Set 2: assuming 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

	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

		7	6	5	4	3	2	1	0
	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		0	
	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
	${S_i}^1$:	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	
	${S_i}^1$:	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							

$CI_0 = 0$

Conditional Sum Adder

- Merge eight bits (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
	${S_i}^1$:	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	
	${S_i}^1$:	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							
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:								
	S_i^1:								
	CO_i^1:								
Step 2	S_i^0:								
	CO_i^0:								
	S_i^1:								
	CO_i^1:								
Step 3	S_i^0:								
	CO_i^0:								
	S_i^1:								
	CO_i^1:								
Result									

$CI_0 = 1$

Conditional Sum Adder

- Delay analysis

		7	6	5	4	3	2	1	0	
	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		$\tau = \Delta_{FA}$
	${CO_i}^1$:	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			$\tau = \Delta_{FA} + \Delta_{MUX}$
	${CO_i}^1$:	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								$\tau = \Delta_{FA} + 2\Delta_{MUX}$

n-bit Conditional Sum Adder

- Delay analysis
  - Delay of a 1-bit adder: d
  - Delay of a MUX: m
  - Delay: $\tau = d + (\log_2 n - 1) \cdot m$ ➡ $O(\log_2 n)$
    - $d = 50ps, m = 50ps, n = 64$
    - $\tau = 300ps$

	RCA	Carry Select	Carry Skip	Conditional Sum
Delay	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\log_2 n)$
64-bit	3,200ps	750ps	900ps	300ps

Prefix Adder

- Generation: $g_i = A_i \cdot B_i$
  - $g_i = 1$ only when $(A_i, B_i) = (1,1)$

- Propagation: $p_i = A_i \oplus B_i$ (conceptually $A_i \oplus B_i$, practically $A_i + B_i$)
  - $p_i = 1$ only when $(A_i, B_i) = (1,0)$ or $(0,1)$

- $C_1 = g_0 + p_0 \cdot C_0$
- $C_2 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0 = g_1 + p_1 \cdot C_1$
- $C_3 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0 = g_2 + p_2 \cdot C_2$
- $C_4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0 = g_3 + p_3 \cdot C_3$

- In general
  - $C_{i+1} = g_i + p_i \cdot C_i$
  - $S_i = A_i \oplus B_i \oplus C_i = p_i \oplus C_i$

- Question: How can we quickly generate all the carry signals?

Prefix Adder

- $C_1 = \underbrace{g_0}_{g_{0:0}} + \underbrace{p_0}_{p_{0:0}} \cdot C_0$

- $C_2 = g_1 + p_1 \cdot C_1 = \underbrace{g_1 + p_1 \cdot g_0}_{g_{1:0}} + \underbrace{p_1 \cdot p_0}_{p_{1:0}} \cdot C_0 = g_{1:0} + p_{1:0} \cdot C_0$

- $C_3 = g_2 + p_2 \cdot C_2 = \underbrace{g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0}_{g_{2:0}} + \underbrace{p_2 \cdot p_1 \cdot p_0}_{p_{2:0}} \cdot C_0 = g_{2:0} + p_{2:0} \cdot C_0$

- $C_4 = g_3 + p_3 \cdot C_3 = \underbrace{g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0}_{g_{3:0}} + \underbrace{p_3 \cdot p_2 \cdot p_1 \cdot p_0}_{p_{3:0}} \cdot C_0 =$ $g_{3:0} + p_{3:0} \cdot C_0$

- In general
  - $C_{i+1} = g_i + p_i \cdot C_i = g_{i:0} + p_{i:0} \cdot C_0$
  - Thus, the question is how to generate $g_{i:0}$ and $p_{i:0}$ quickly and not with an excessive use of gates.

Prefix Adder

- Observation
  - $p_{i:0} = p_i \cdot p_{i-1} \cdot \cdots \cdot p_1 \cdot p_0 = (p_i \cdot \cdots \cdot p_{i-k}) \cdot (p_{i-k-1} \cdot \cdots \cdot p_0) = p_{i:i-k} \cdot p_{i-k-1:0}$

  - $g_{i:i-k}$

    $g_{i:0} = g_i + (p_i \cdot g_{i-1}) + (p_i \cdot p_{i-1} \cdot g_{i-2}) + \cdots + (p_i \cdot p_{i-1} \cdot \cdots \cdot p_{i-k+1} \cdot g_{i-k}) + (p_i \cdot \cdots \cdot p_{i-k} \cdot g_{i-k-1}) + (p_i \cdot \cdots \cdot p_{i-k-1} \cdot g_{i-k-2}) + \cdots + (p_i \cdot \cdots \cdot p_0 \cdot g_0)$

    $(p_i \cdot p_{i-1} \cdot \cdots \cdot p_{i-k}) \cdot (g_{i-k-1} + p_{i-k-1} \cdot g_{i-k-2} + \cdots + p_{i-k-1} \cdot \cdots \cdot p_0 \cdot g_0) = p_{i:i-k} \cdot g_{i-k-1:0}$

  - Thus, we can break down $p_{i:0}$ and $g_{i:0}$ into two components.
    - $p_{i:0} = p_{i:i-k} \cdot p_{i-k-1:0}$
    - $g_{i:0} = g_{i:i-k} + p_{i:i-k} \cdot g_{i-k-1:0}$

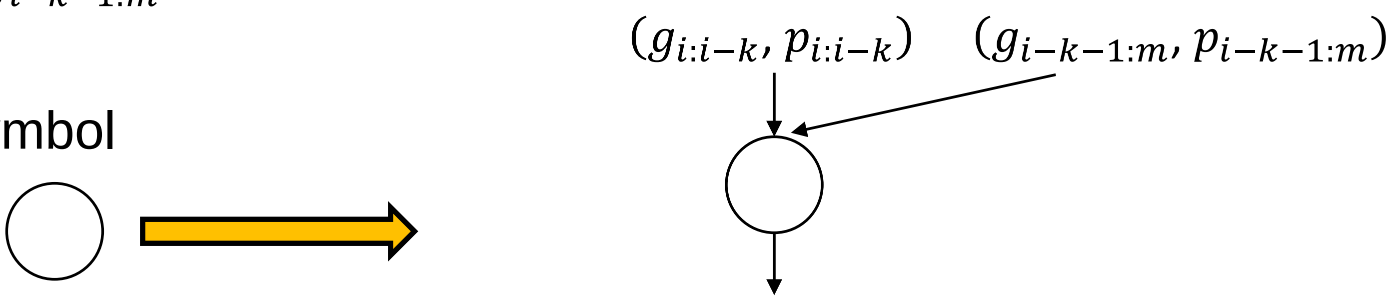
  - In general
    - $p_{i:m} = p_{i:i-k} \cdot p_{i-k-1:m}$
    - $g_{i:m} = g_{i:i-k} + p_{i:i-k} \cdot g_{i-k-1:m}$

Prefix Adder

- How to obtain $p_{i:m}$ and $g_{i:m}$ systematically
  - $p_{i:m} = p_{i:i-k} \cdot p_{i-k-1:m}$
  - $g_{i:m} = g_{i:i-k} + p_{i:i-k} \cdot g_{i-k-1:m}$

- In other words, we need
  - $p_{i:i-k}$
  - $g_{i:i-k}$
  - $p_{i-k-1:m}$
  - $g_{i-k-1:m}$

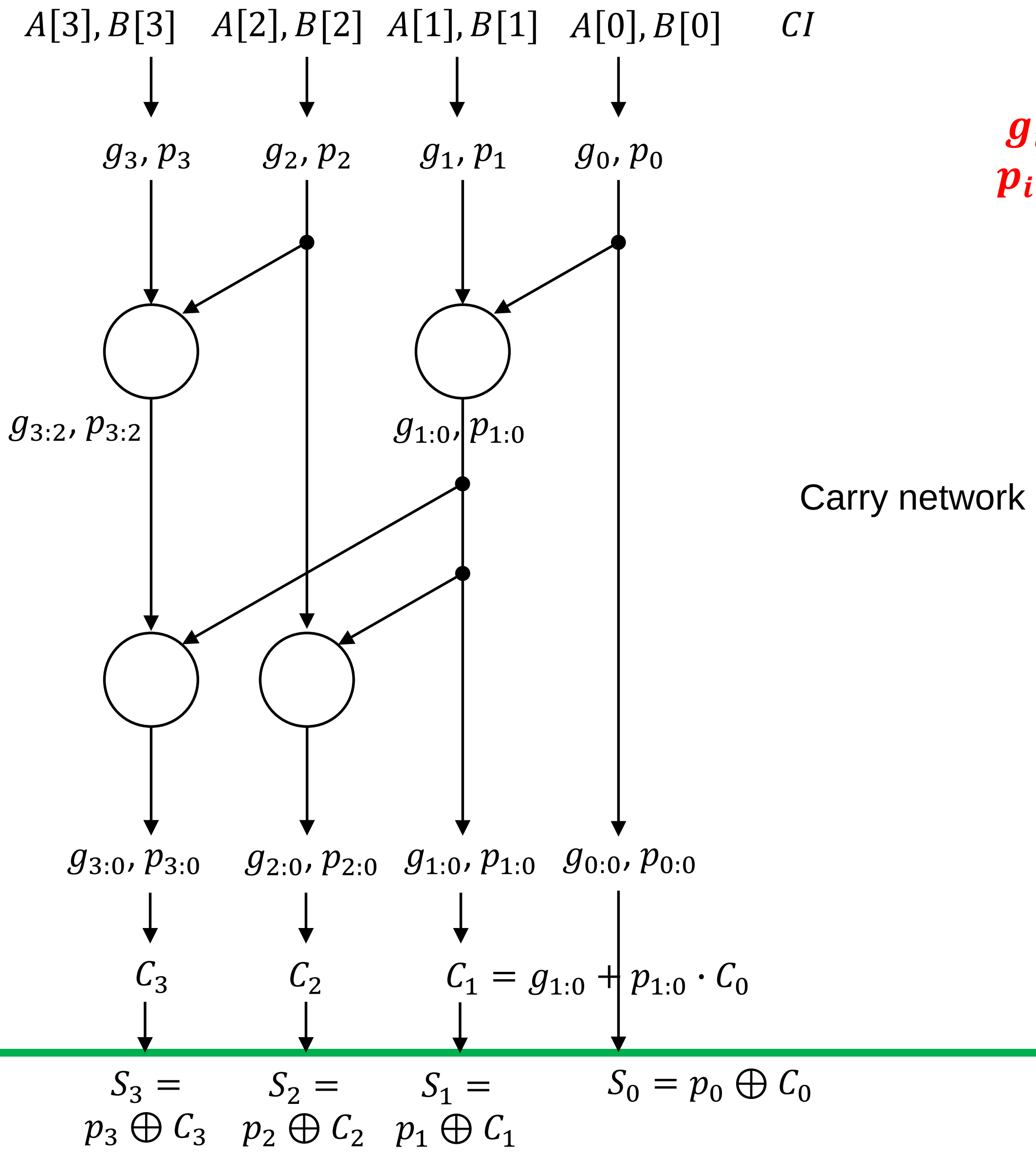
- Symbol



$$(\boldsymbol{g}_{i:m}, \boldsymbol{p}_{i:m}) = (\boldsymbol{g}_{i:i-k} + \boldsymbol{p}_{i:i-k} \cdot \boldsymbol{g}_{i-k-1:m}, \boldsymbol{p}_{i:i-k} \cdot \boldsymbol{p}_{i-k-1:m})$$

Prefix Adder

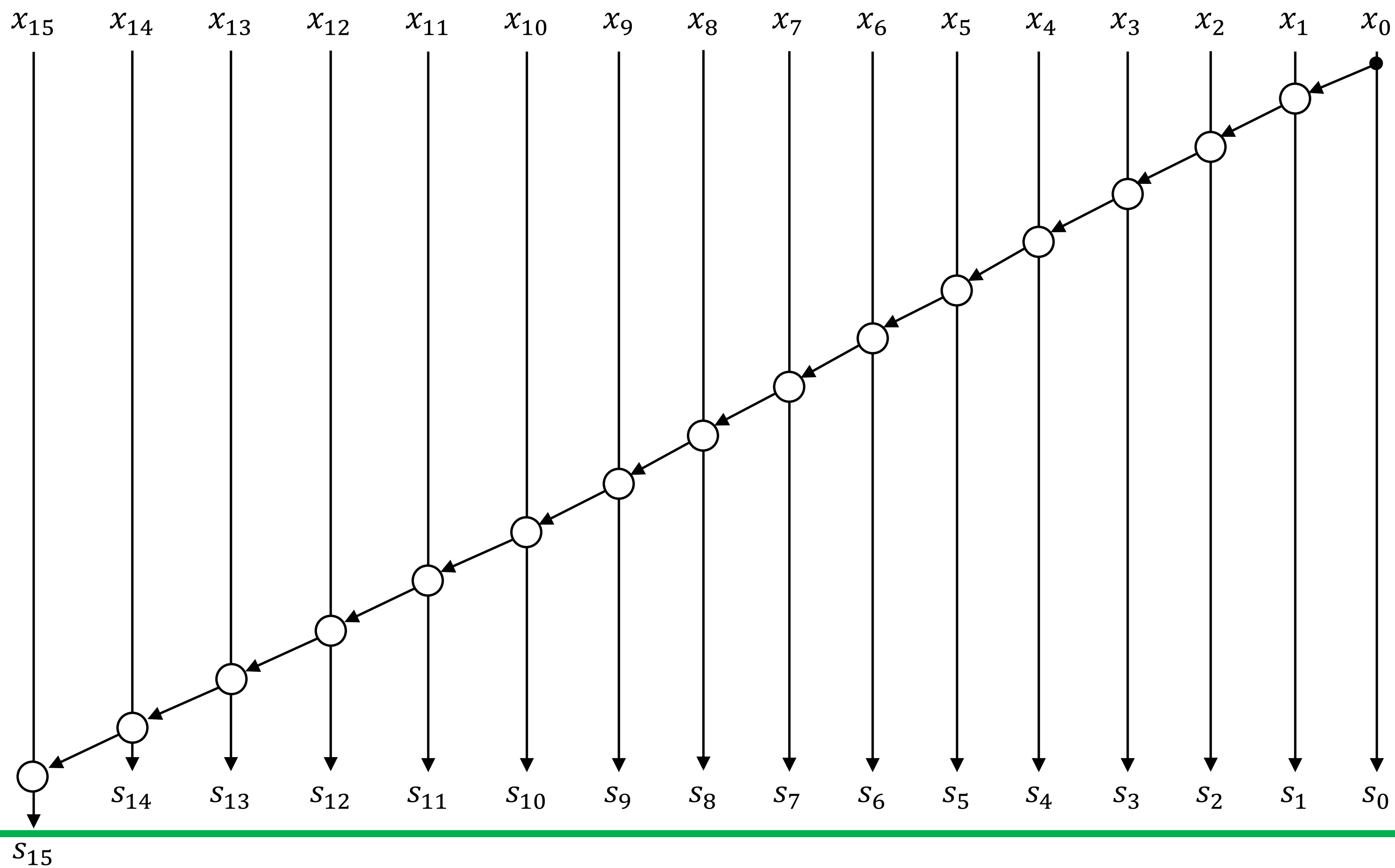
- Example



$$\boldsymbol{g_i = A_i \cdot B_i}$$
$$\boldsymbol{p_i = A_i + B_i}$$

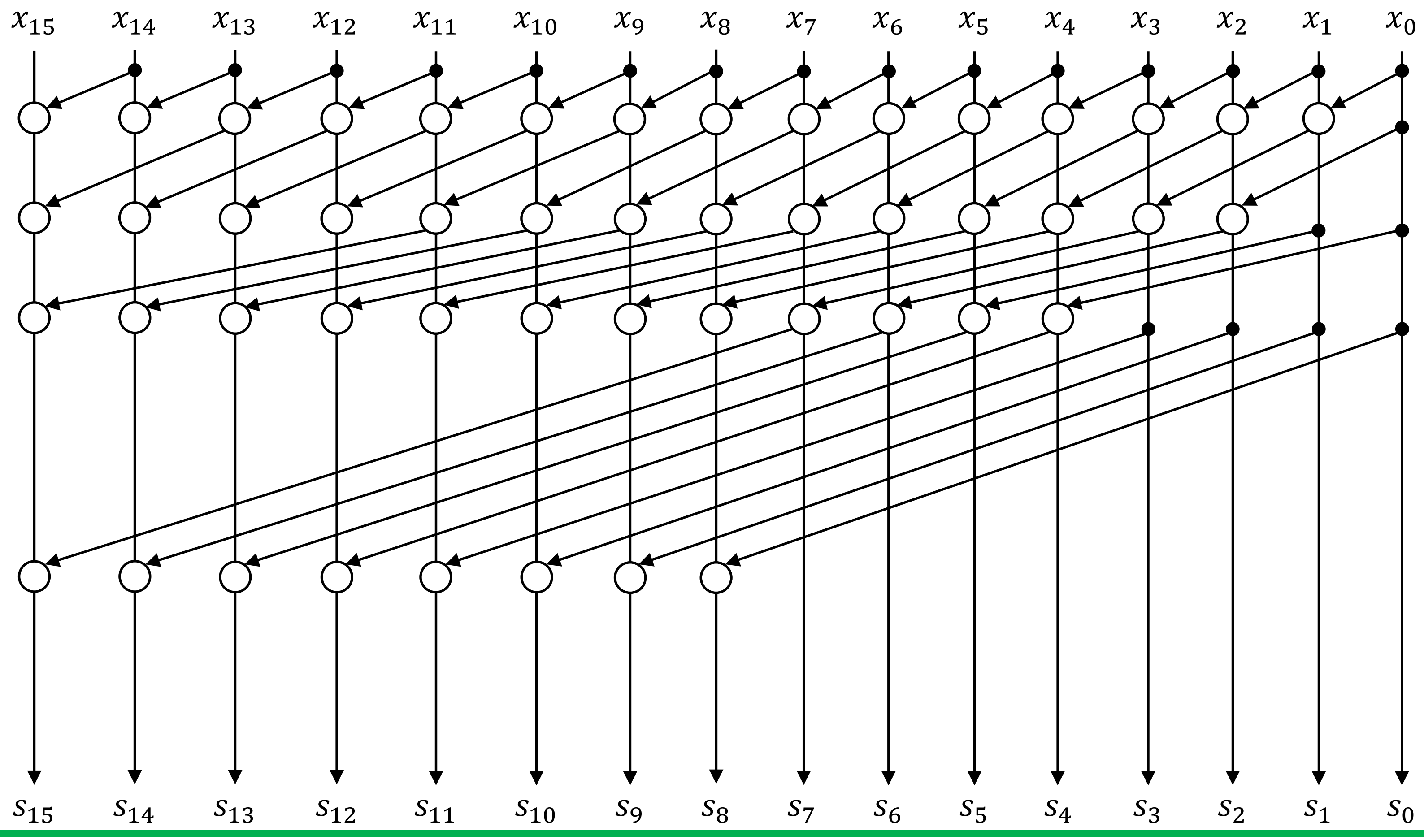
Prefix Adder

- Ripple carry adder



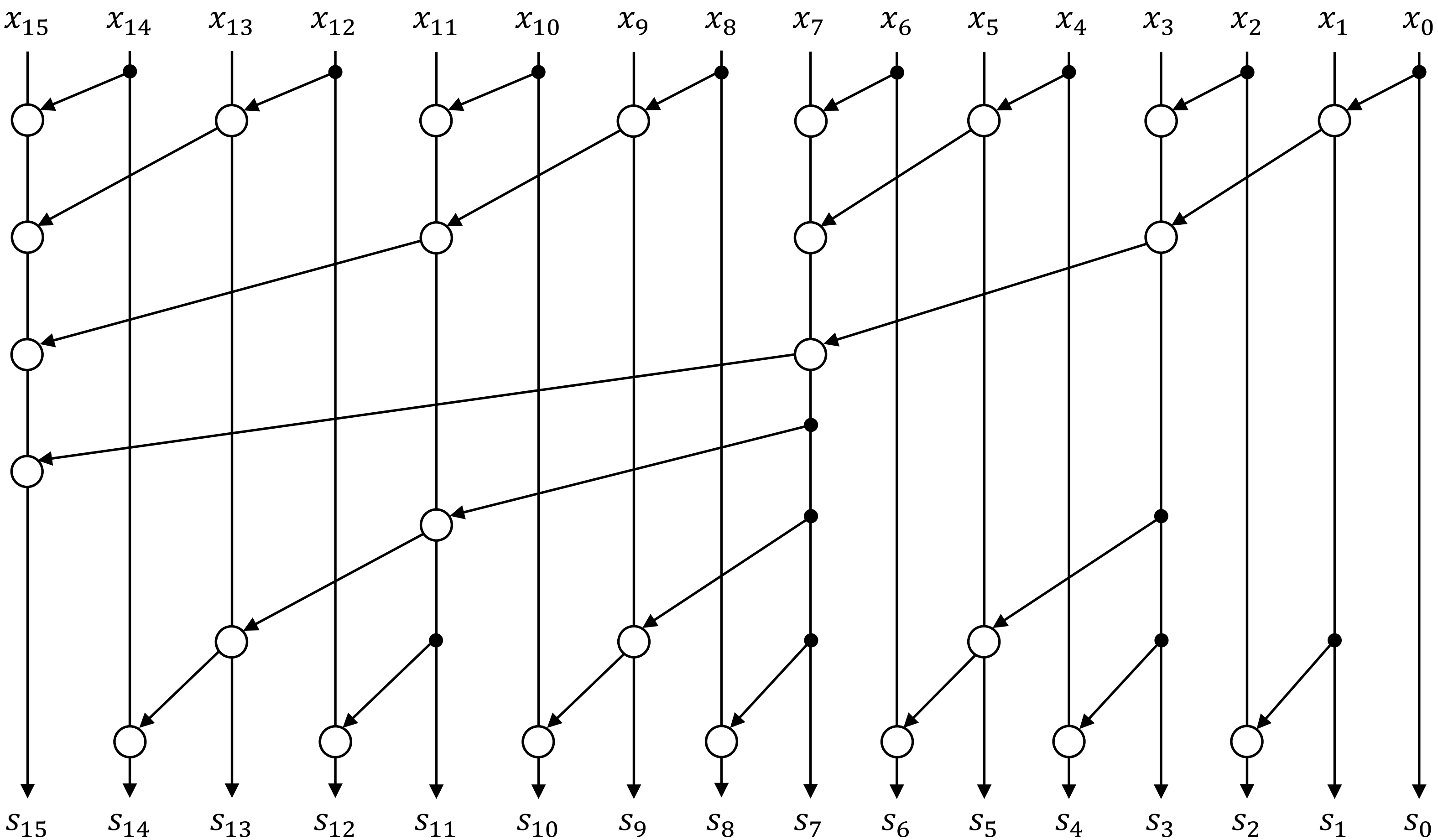
Prefix Adder

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



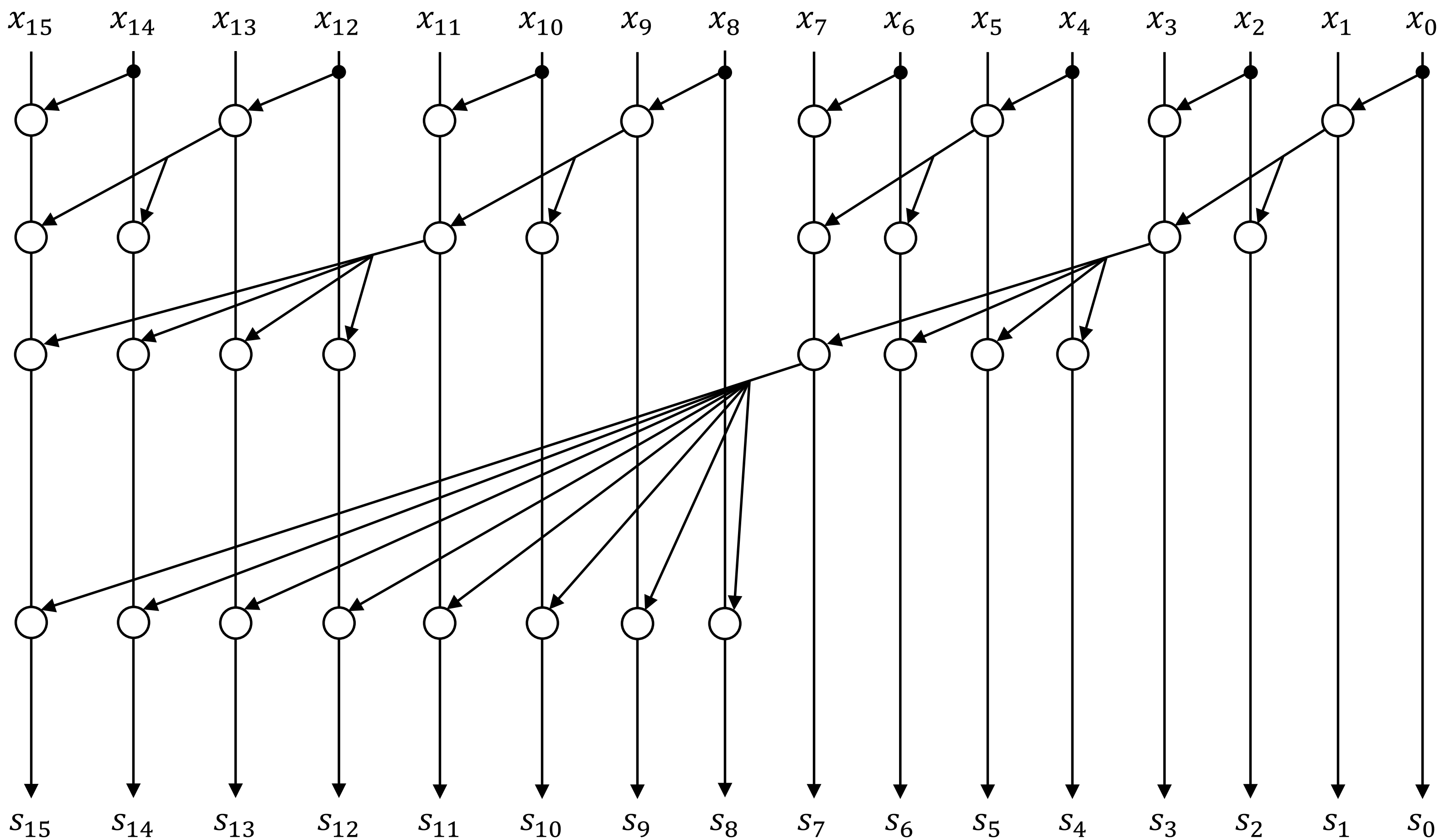
Prefix Adder

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



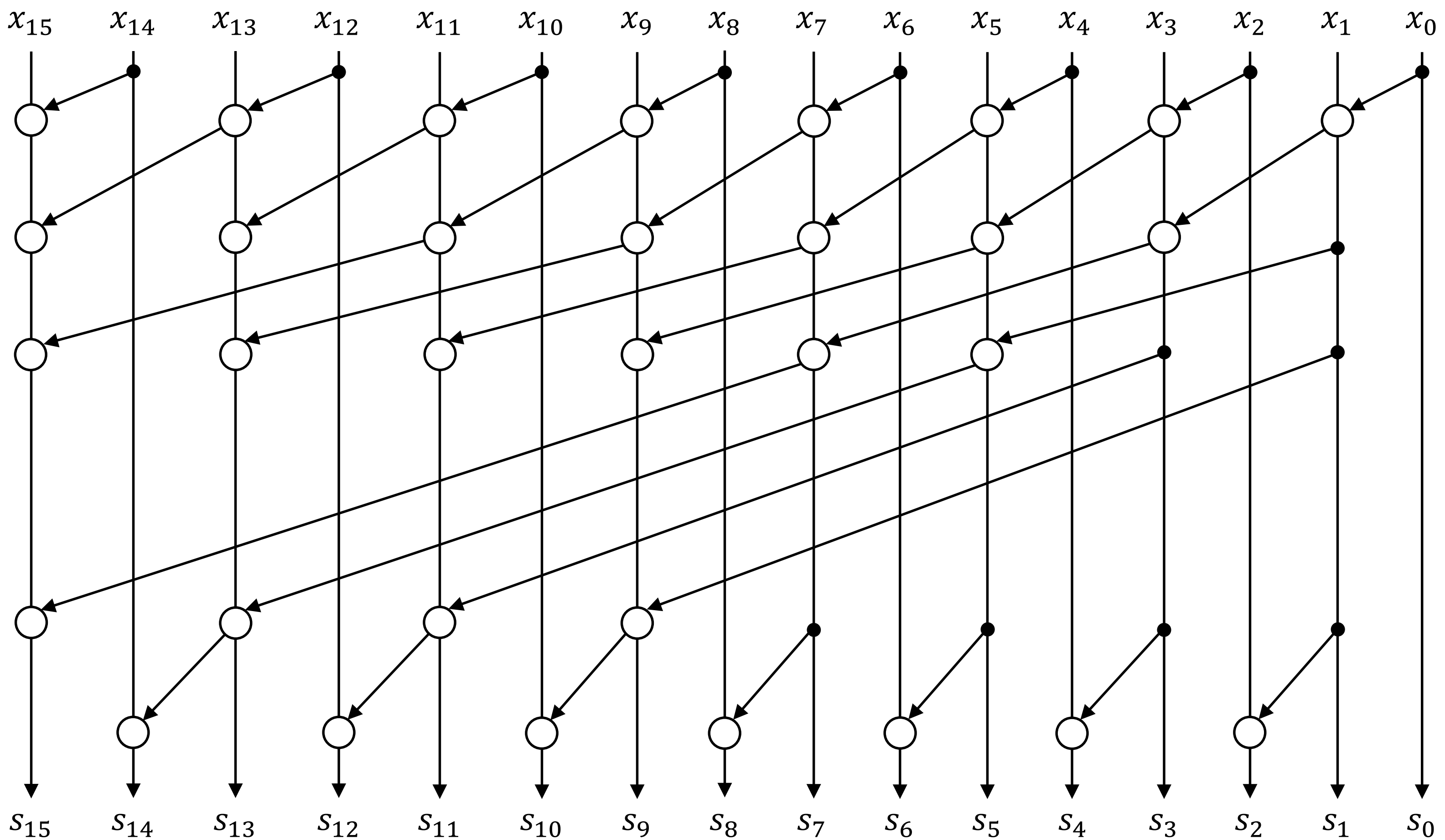
Prefix Adder

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



Prefix Adder

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



Prefix Adder

- Delay analysis
  - Generation of p_i and g_i: d
  - Merging logic $\bigcirc$: $2d$
  - Sum: $2d$
  - Delay: $d + k \cdot (2d) + 2d + 2d$
    - d: Generation of p_i and g_i
    - $k \cdot (2d)$: Generation of $p_{i:0}$ and $g_{i:0}$ (k: logic depth)
    - $2d$: Calculation of C_i
    - $2d$: Calculation of S_i

  - Example (16-bit adder)
    - Ripple carry: $d + 15d + 2d + 2d = 20d$
    - Kogge-Stone: $d + 4d + 2d + 2d = 9d$ (k is $\log_2 n$)
    - Brent-Kung: $d + 6d + 2d + 2d = 11d$
    - Ladner-Fischer: $d + 4d + 2d + 2d = 9d$
    - Han-Carlson: $d + 5d + 2d + 2d = 10d$

Carry-Lookahead Adder

- To calculate C_{i+1}
  - Use $g_{i:0}$ and $p_{i:0}$: Prefix adder
  - Use some C_k: Carry-lookahead adder

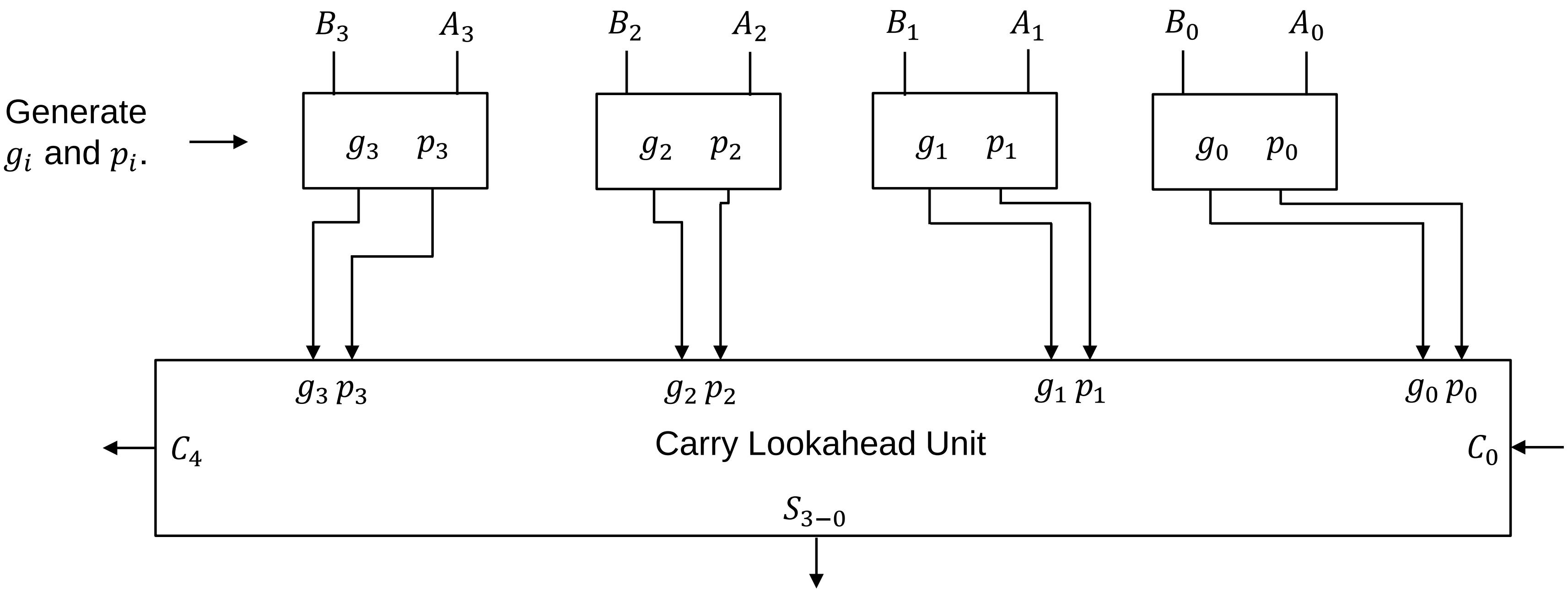
- Observation
  - $C_1 = g_0 + p_0 \cdot C_0$
  - $C_2 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0$
  - $C_3 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0$
  - $C_4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0$

  - $C_5 = g_4 + p_4 \cdot C_4$
  - $C_6 = g_5 + p_5 \cdot g_4 + p_5 \cdot p_4 \cdot C_4$
  - $C_7 = g_6 + p_6 \cdot g_5 + p_6 \cdot p_5 \cdot g_4 + p_6 \cdot p_5 \cdot p_4 \cdot C_4$
  - $C_8 = g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + p_7 \cdot p_6 \cdot p_5 \cdot g_4 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot C_4$

  - …

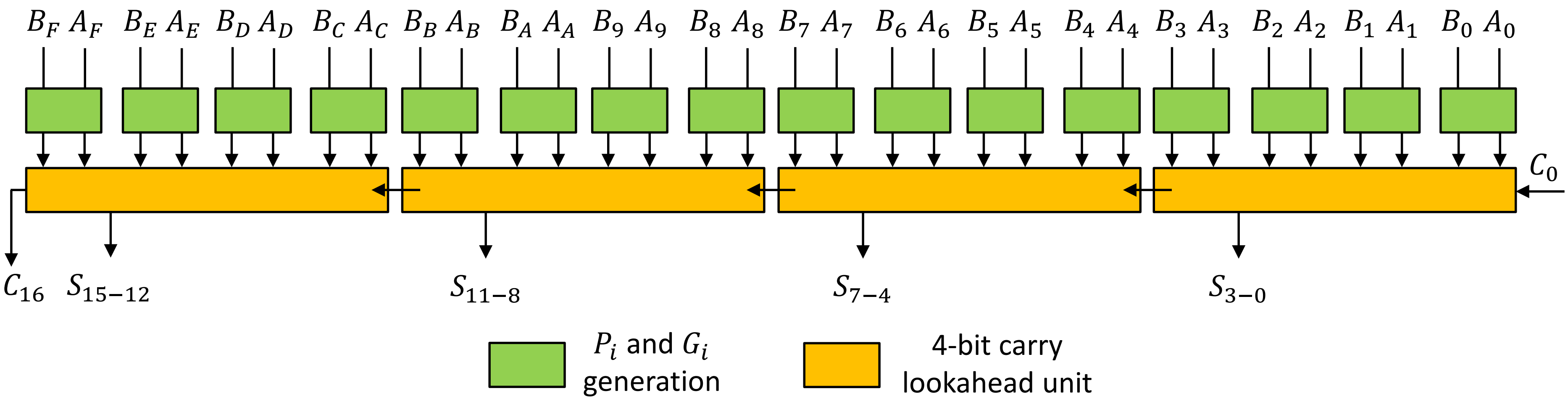
Carry-Lookahead Adder

- A 4-bit carry lookahead logic



Carry-Lookahead Adder

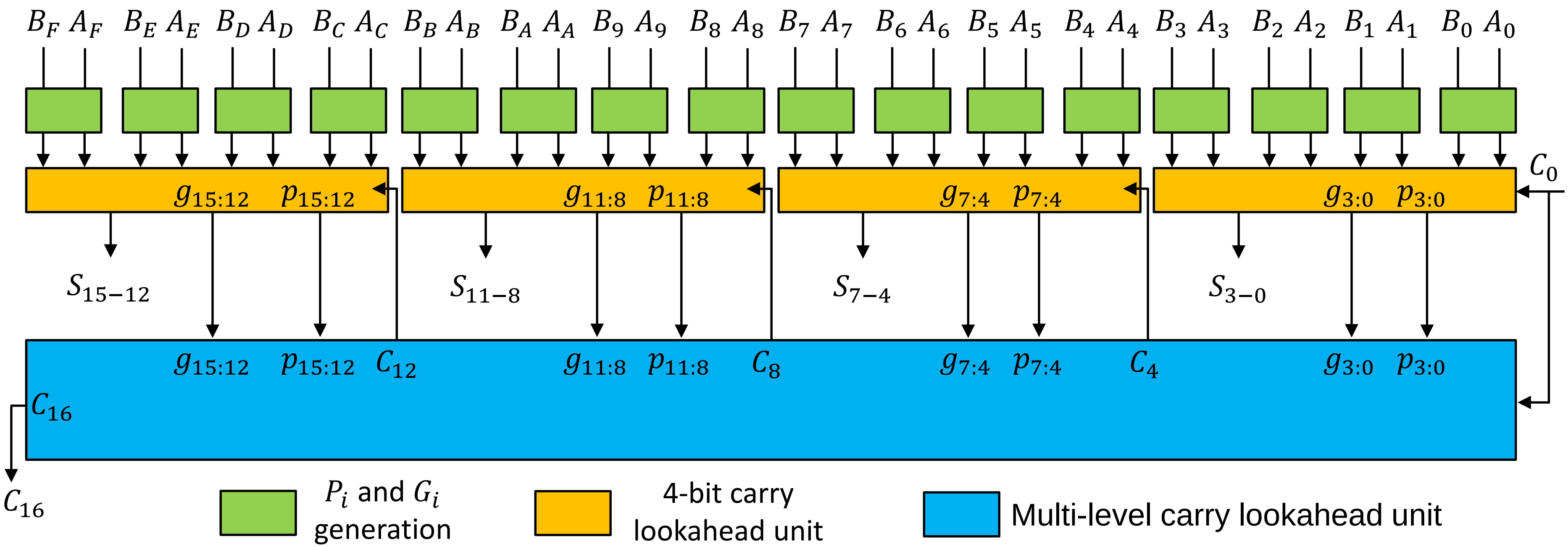
- A one-level 16-bit carry lookahead adder



- How to compute C_i
  - $C_4 = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot C_0$
  - $C_5 = g_4 + p_4 \cdot C_4$
  - $C_8 = g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + p_7 \cdot p_6 \cdot p_5 \cdot g_4 + p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot C_4$
  - $C_9 = g_8 + p_8 \cdot C_8$
  - $C_{16} = g_{15} + p_{15} \cdot g_{14} + p_{15} \cdot p_{14} \cdot g_{13} + p_{15} \cdot p_{14} \cdot p_{13} \cdot g_{12} + p_{15} \cdot p_{14} \cdot p_{13} \cdot p_{12} \cdot C_{12}$

Carry-Lookahead Adder

- Two-level carry lookahead adder



- How to generate C_i
  - $C_4 = g_{3:0} + p_{3:0} \cdot C_0$
  - $C_5 = g_4 + p_4 \cdot C_4$
  - $C_8 = g_{7:4} + p_{7:4} \cdot g_{3:0} + p_{7:4} \cdot p_{3:0} \cdot C_0$

Carry-Lookahead Adder

- How to generate C_i in 2-level CLA
  - $C_1 = g_0 + p_0 \cdot C_0$
  - $C_2 = g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot C_0$
  - $C_3 = g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot C_0$

    (In the 1-level carry lookahead unit)
  - $C_4 = g_{3:0} + p_{3:0} \cdot C_0$ → In the 2-level carry lookahead unit

  - $C_5 = g_4 + p_4 \cdot C_4$
  - $C_6 = g_5 + p_5 \cdot g_4 + p_5 \cdot p_4 \cdot C_4$
  - $C_7 = g_6 + p_6 \cdot g_5 + p_6 \cdot p_5 \cdot g_4 + p_6 \cdot p_5 \cdot p_4 \cdot C_4$

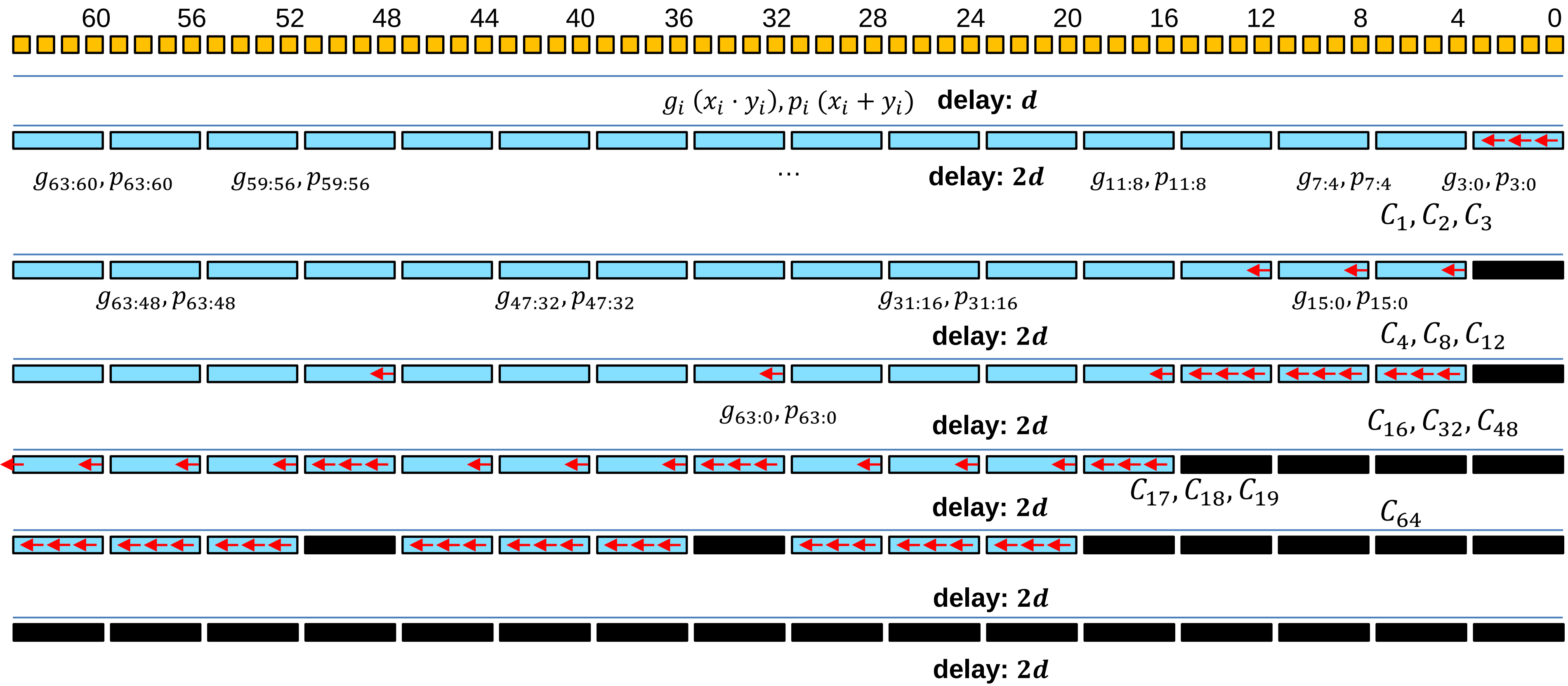
    (In the 1-level carry lookahead unit)
  - $C_8 = g_{7:4} + p_{7:4} \cdot g_{3:0} + p_{7:4} \cdot p_{3:0} \cdot C_0$ → In the 2-level carry lookahead unit

  - $C_9 = g_8 + p_8 \cdot C_8$
  - $C_{10} = g_9 + p_9 \cdot g_8 + p_9 \cdot p_8 \cdot C_8$
  - $C_{11} = g_{10} + p_{10} \cdot g_9 + p_{10} \cdot p_9 \cdot g_8 + p_{10} \cdot p_9 \cdot p_8 \cdot C_8$
  - $C_{12} = g_{11:8} + p_{11:8} \cdot g_{7:4} + p_{11:8} \cdot p_{7:4} \cdot g_{3:0} + p_{11:8} \cdot p_{7:4} \cdot p_{3:0} \cdot C_0$

  - $C_{16} = g_{15:12} + p_{15:12} \cdot g_{11:8} + p_{15:12} \cdot p_{11:8} \cdot g_{7:4} + p_{15:12} \cdot p_{11:8} \cdot p_{7:4} \cdot g_{3:0} + p_{15:12} \cdot p_{11:8} \cdot p_{7:4} \cdot p_{3:0} \cdot C_0$
    - Notice that C_{16} should be generated by a 3-level carry lookahead unit.

Carry-Lookahead Adder

- 64-bit CLA



Delay: $13d$

Carry-Lookahead Adder

- Example: How is S_{53} is calculated and what is the delay of the calculation?
  - Sum
    - $S_{53} = p_{53} \oplus C_{53}$ (C_{53} is the delay bottleneck)
  - Calculate the group that C_{53} belongs to.
    - $53 \; mod \; 4 = 1$
  - Represent C_{53} w.r.t. the incoming carry C_{52} in the group
    - $C_{53} = g_{52} + p_{52} \cdot C_{52}$ (level 1)
  - Calculate the logic that generates C_{52}
    - $52/16 = 3.\text{xxx}$
    - $C_{52} = g_{51:48} + p_{51:48} \cdot C_{48}$ (level 2)
  - Calculate the logic that generates C_{48}
    - $48/64 = 0.\text{xxx}$
    - $C_{48} = g_{47:32} + p_{47:32} \cdot g_{31:16} + p_{47:32} \cdot p_{31:16} \cdot g_{15:0} + p_{47:32} \cdot p_{31:16} \cdot p_{15:0} \cdot C_0$ (level 3)
  - Calculation of $g_{i:i-15}$ (or $p_{i:i-15}$)
    - $g_{15:0} = g_{15:12} + p_{15:12} \cdot g_{11:8} + p_{15:12} \cdot p_{11:8} \cdot g_{7:4} + p_{15:12} \cdot p_{11:8} \cdot p_{7:4} \cdot g_{3:0}$
  - Calculation of $g_{i:i-3}$
    - $g_{3:0} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$
  - Calculation of g_i
    - $g_0 = A_0 \cdot B_0$
  - Delay calculation (backtrace)
    - $d + 2d + 2d + 2d + 2d + 2d + 2d = 13d$

Carry-Lookahead Adder

- Example: How is S_{64} is calculated and what is the delay of the calculation?
  - Sum
    - $S_{64} = p_{64} \oplus C_{64}$ (C_{64} is the delay bottleneck)
  - Calculate the group that C_{64} belongs to.
    - $64\ mod\ 4 = 0$
    - $64\ mod\ 16 = 0$
    - $64\ mod\ 64 = 0$ (this means C_{64} is calculated in the level-3 carry lookahead unit)
  - Represent C_{64} w.r.t. the incoming carry C_0 in the group
    - $C_{64} = g_{63:0} + p_{63:0} \cdot C_0$ (level 3)
  - Calculation of $g_{i:i-63}$ (or $p_{i:i-63}$)
    - $g_{63:0} = g_{63:48} + p_{63:48} \cdot g_{47:32} + p_{63:48} \cdot p_{47:32} \cdot g_{31:0} + p_{63:48} \cdot p_{47:32} \cdot p_{31:16} \cdot g_{15:0}$
  - Calculation of $g_{i:i-15}$ (or $p_{i:i-15}$)
    - $g_{15:0} = g_{15:12} + p_{15:12} \cdot g_{11:8} + p_{15:12} \cdot p_{11:8} \cdot g_{7:4} + p_{15:12} \cdot p_{11:8} \cdot p_{7:4} \cdot g_{3:0}$
  - Calculation of $g_{i:i-3}$
    - $g_{3:0} = g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$
  - Calculation of g_i
    - $g_0 = A_0 \cdot B_0$
  - Delay calculation (backtrace)
    - $d + 2d + 2d + 2d + 2d + 2d = 11d$