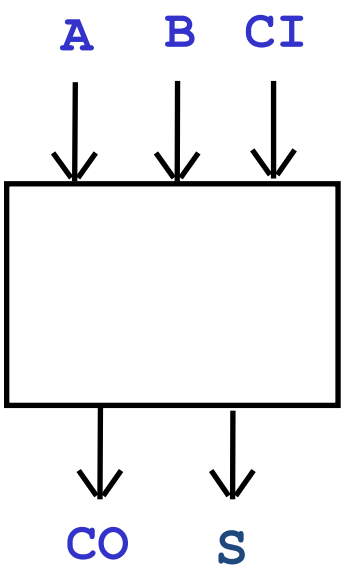
FA Based Adders – Ripple-carry adder

Binary addition

- This is the primitive of almost all arithmetic computation.



Truth Table:

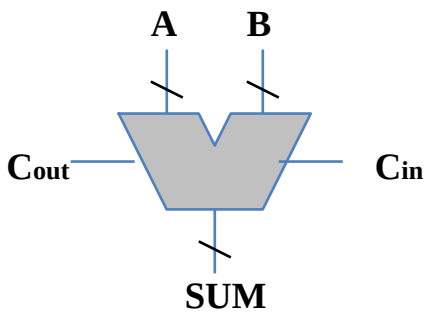
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

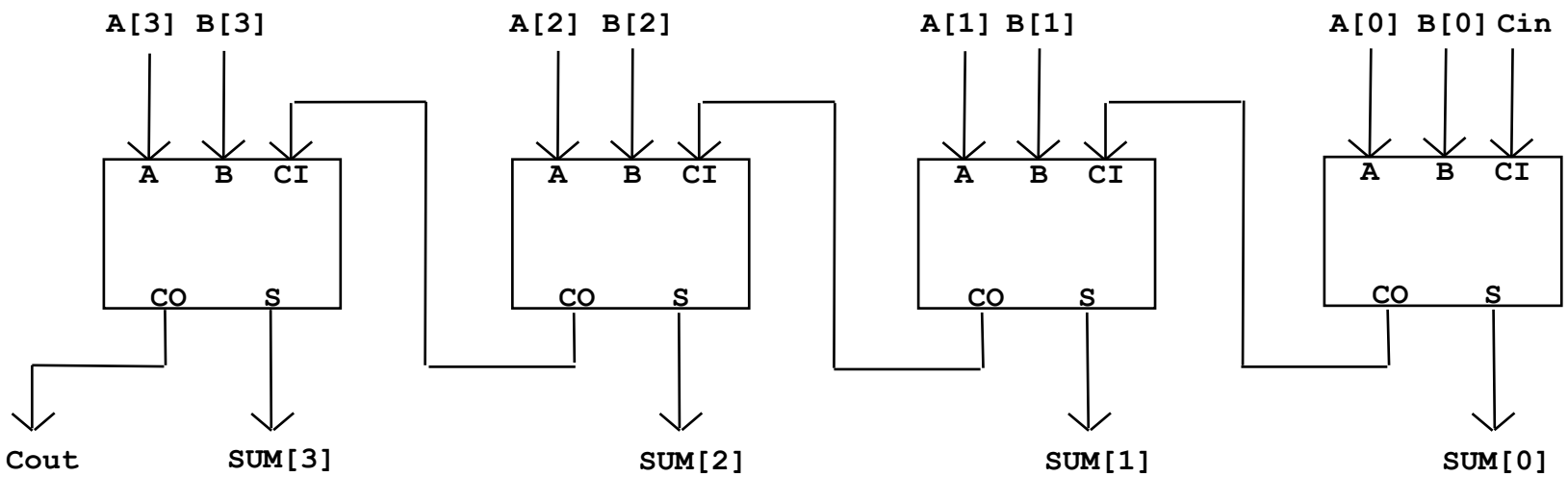
$$CO = A \cdot B + B \cdot CI + CI \cdot A$$

$$S = A \oplus B \oplus CI$$

4-Bit Ripple-Carry Adder (RCA)

- The carry chain ripples from the least to the most significant bit (LSB to MSB).

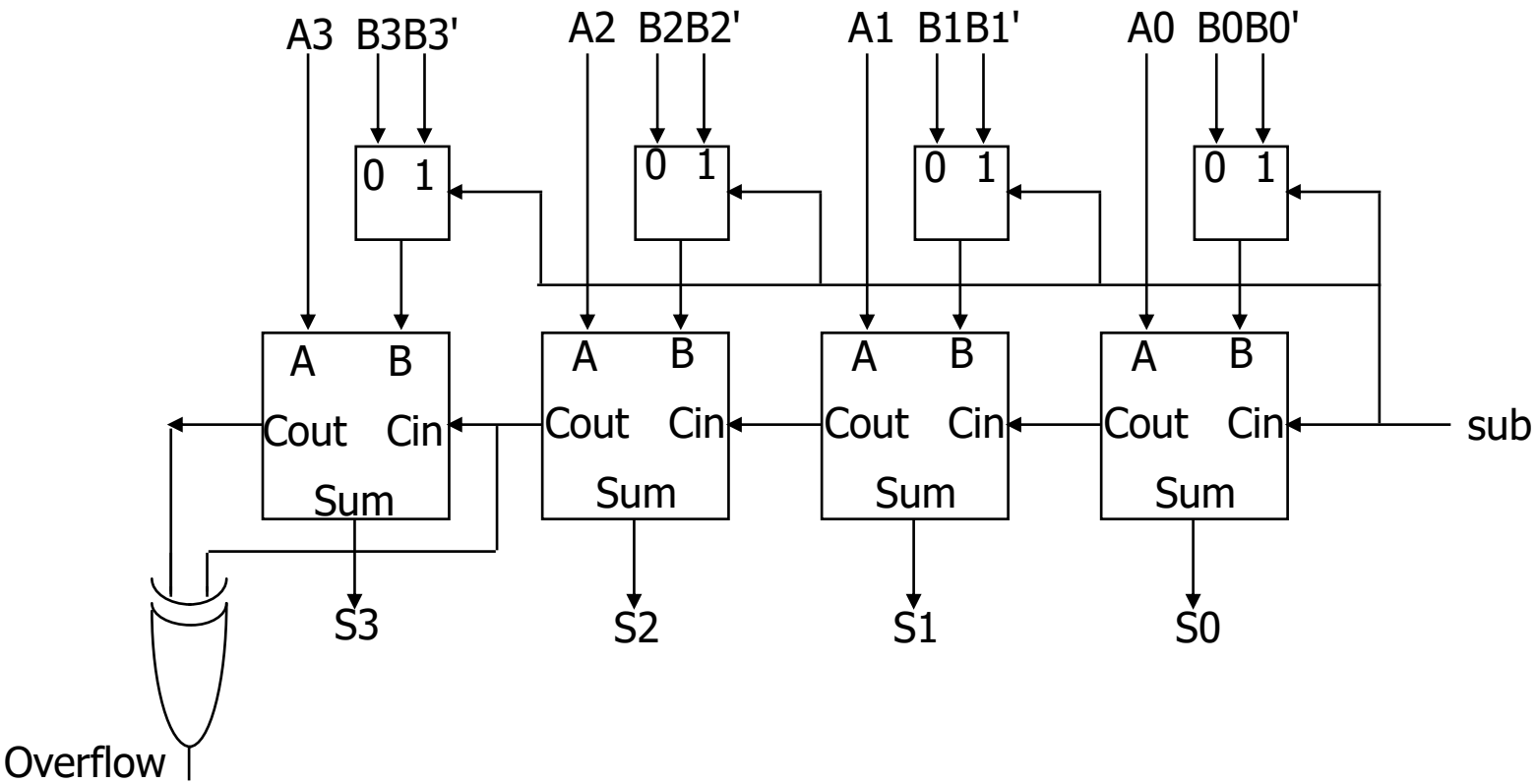




Area = O(bit-width)
Delay = O(bit-width)

Adder/subtractor

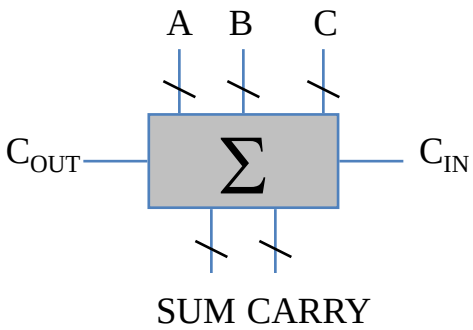
- Use an adder to do subtraction thanks to 2s complement representation
  - A – B = A + (– B) = A + B' + 1
  - control signal selects B or 2s complement of B

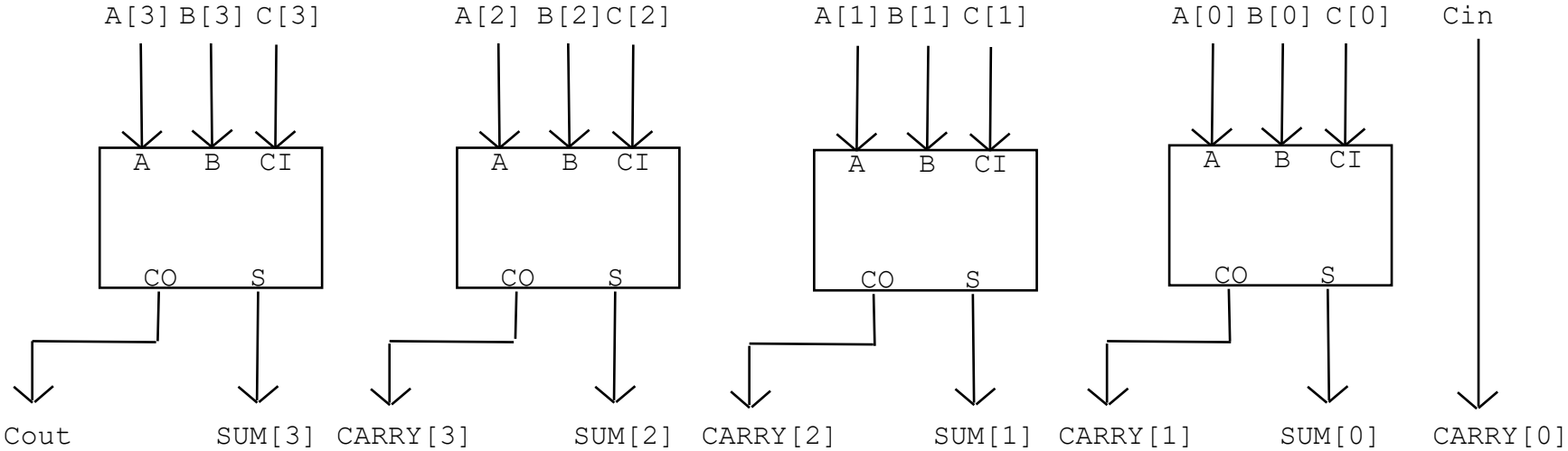


FA Based Adders – Carry-save adder

Carry-save Adder (CSA)

- There is no carry chain rippling from LSB to MSB
- Each CO pin goes to the next higher bit column

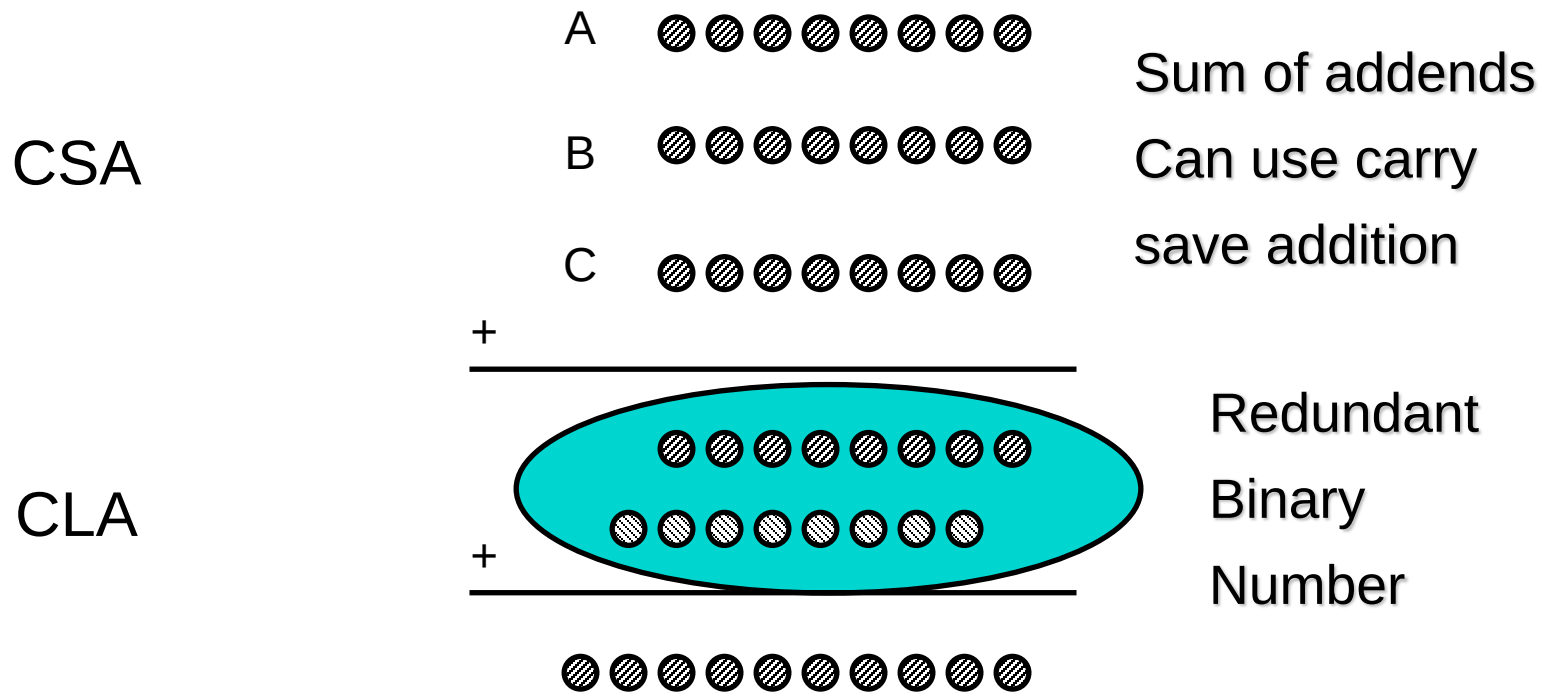




Area = O(bit-width)
Delay = O(1)

Carry Save Addition

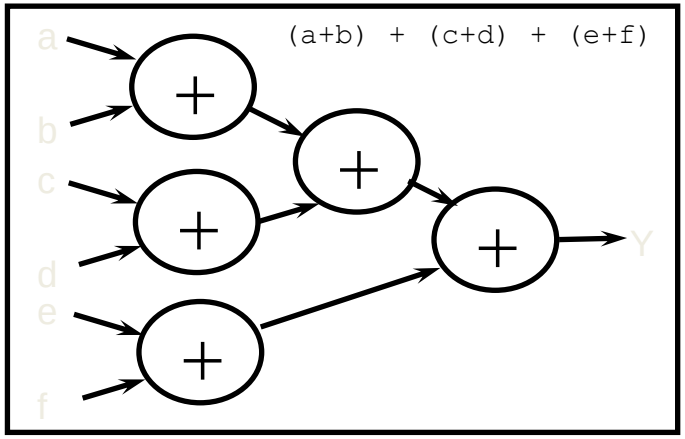
- Sum of 3 addends : A+B+C



- With Carry save addition, A+B+C can be computed with a single CLA adder for the final sum in binary. Conventional addition (fast) would require 2 CLAs.

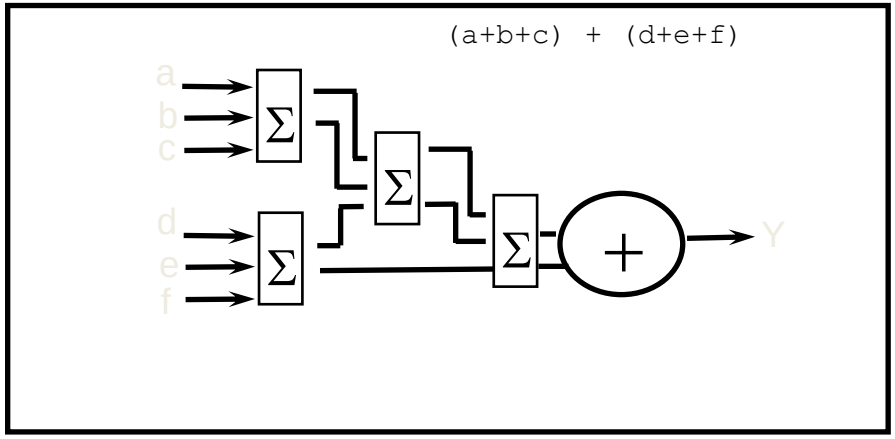
Designing Optimized Arithmetic

Conventional Arithmetic



- Additions in groups of 2
- Complete computation at the end of each operator

Carry Save Addition (CSA) Arithmetic

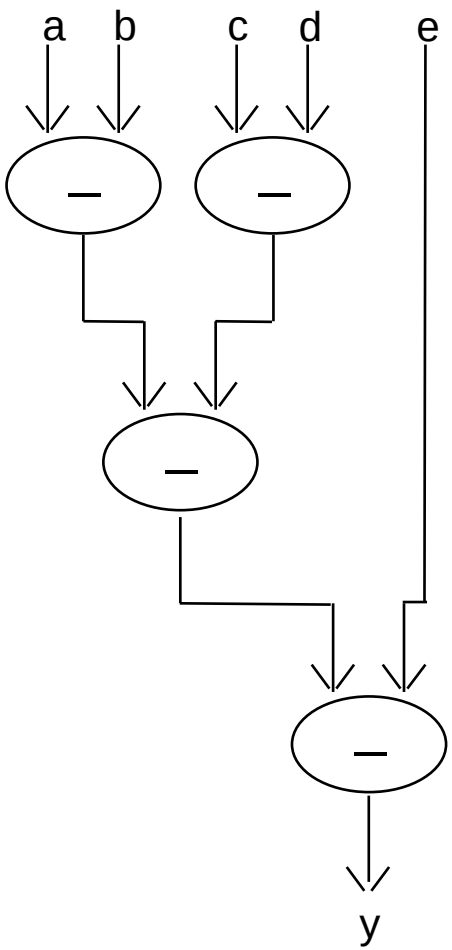


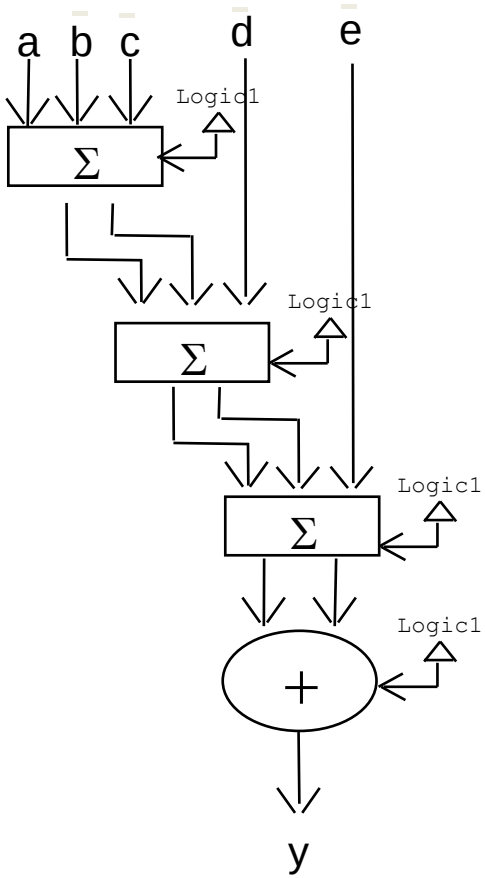
- Additions in groups of 3
- Partial intermediate computation (saves area and improves speed)

CAS Transformation for Subtractions

$a - b - c - d - e$

$a + \bar{b} + \bar{c} + \bar{d} + \bar{e} + 4$





Summary

- ✓ FA (half adder) 는 3개의 bit 입력에 대해 logic-1의 입력 개수를 (2진법 형태로) 출력으로 내기 때문에 _____counter 라고도 부른다.

- ✓ FA based adder의 대표적인 종류로 _____ 와 _____ 을 들 수 있다.

- ✓ N-bit RCA의 Area 와 Delay는 모두 N에 _____적으로 비례한다.

- ✓ N-bit CSA (carry-save-adder)의 Area는 N에 _____적으로 비례하며. Delay는 N에 _____적으로 비례한다.

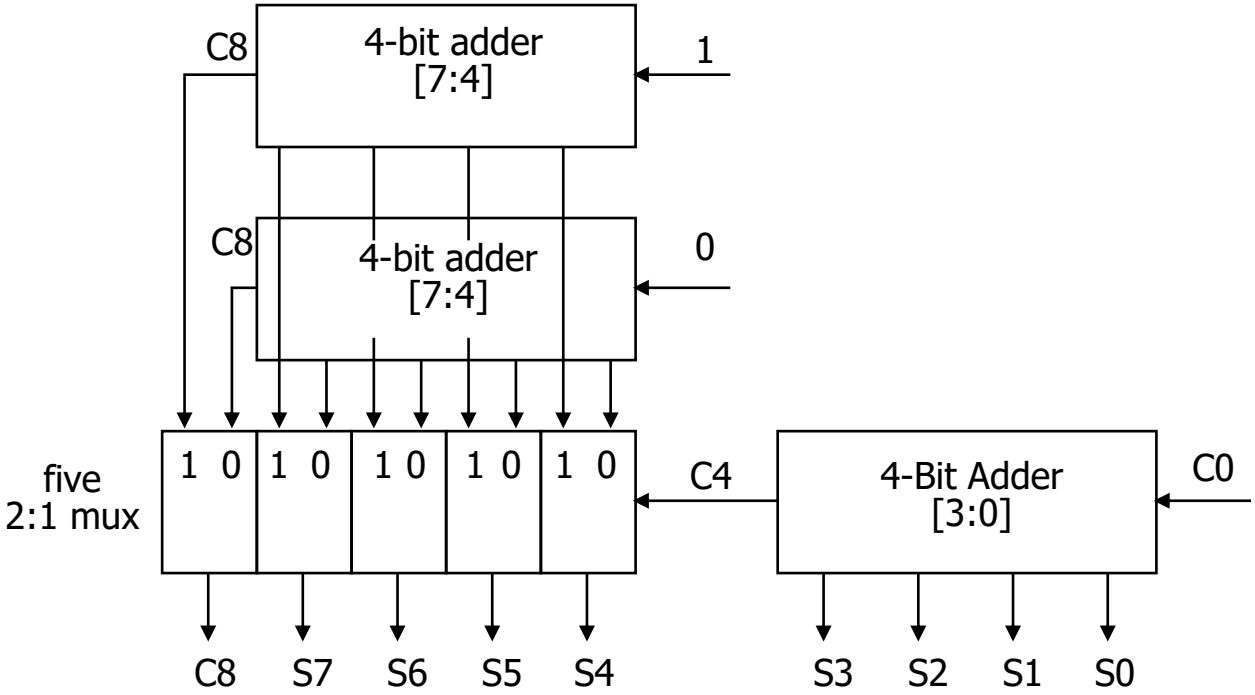
- ✓ CSA (carry-save-adder)는 Addition expression 뿐만 아니라 _____ 와 _____을 포함하는 expression 에서도 사용 가능하다.

- ✓ CSA (carry-save-adder) 사용의 주요 목적은 _____을 줄이기 위함이다.

FA Based Adders – Carry-select adder

Carry-Select Adder

- Redundant hardware to make carry calculation go faster
  - compute two high-order sums in parallel while waiting for carry-in.
  - one assuming carry-in is 0 and another assuming carry-in is 1.
  - select correct result once carry-in is finally computed.



P&G Based Adders – Carry-lookhead adder

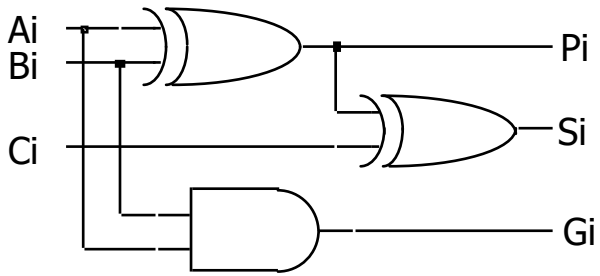
P&G Logic

- Carry **G**enerate: **Gi** = Ai Bi
  - must generate carry when Gi = 1, which means A = B = 1
- Carry **P**ropagate: **Pi** = Ai xor Bi
  - carryin will equal carryout when Pi = 1
- Sum and Cout can be re-expressed in terms of generate/propagate:
  - C_{i+1} = **Gi** + Ci **Pi**
  - Si = Ai xor Bi xor Ci
    = **Pi** xor Ci

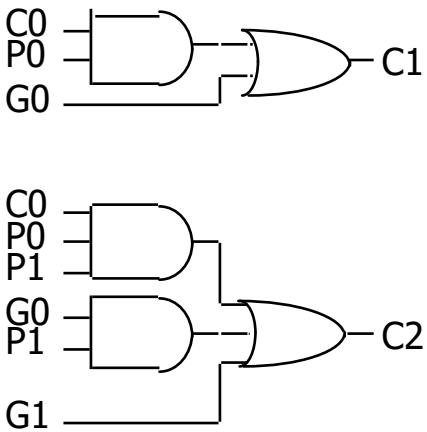
- Carryout logic:
  - $C1 = G0 + P0\ C0$
  - $C2 = G1 + P1\ C1 = G1 + P1\ G0 + P1\ P0\ C0$
  - $C3 = G2 + P2\ C2 = G2 + P2\ G1 + P2\ P1\ G0 + P2\ P1\ P0\ C0$
  - $C4 = G3 + P3\ C3 = G3 + P3\ G2 + P3\ P2\ G1 + P3\ P2\ P1\ G0 + P3\ P2\ P1\ P0\ C0$

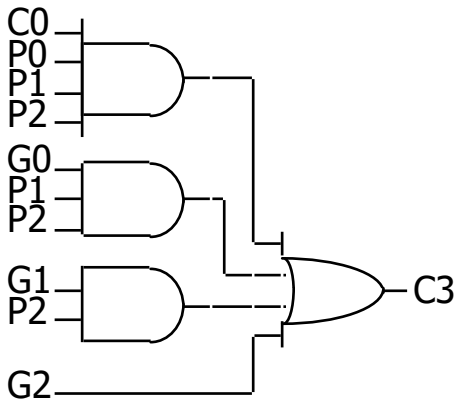
- Each of the carryout equations can be **independently** implemented with two-level logic.
  - all carryout values are now **directly derived from data inputs** and not from intermediate carries.
  - this allows computation of **all sum outputs to proceed in parallel.**

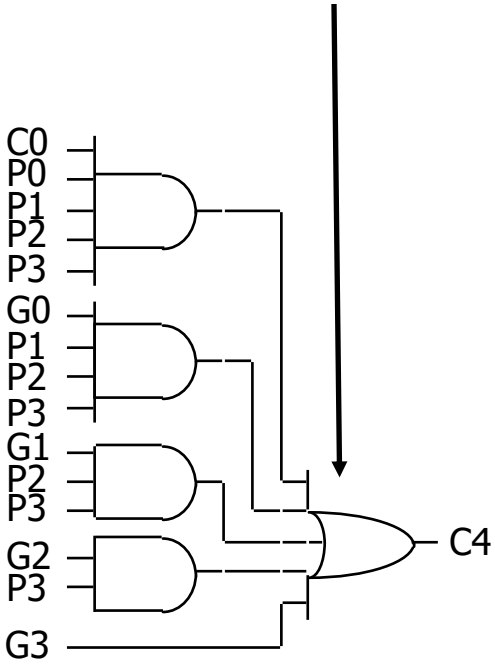
Carryout Implementation (a.k.a Carry Look-ahead Logic)



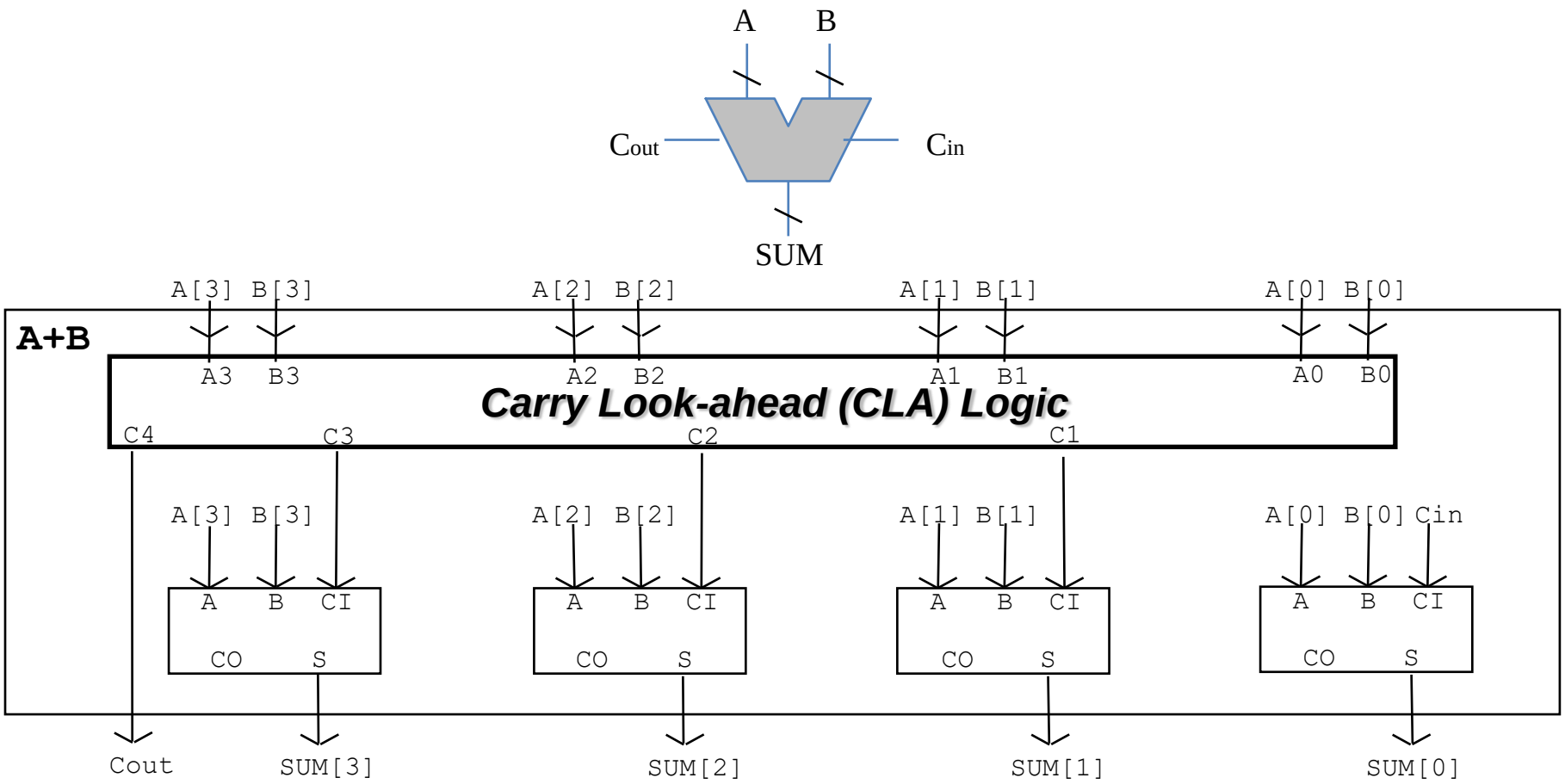
#fanins _____ → delay _____







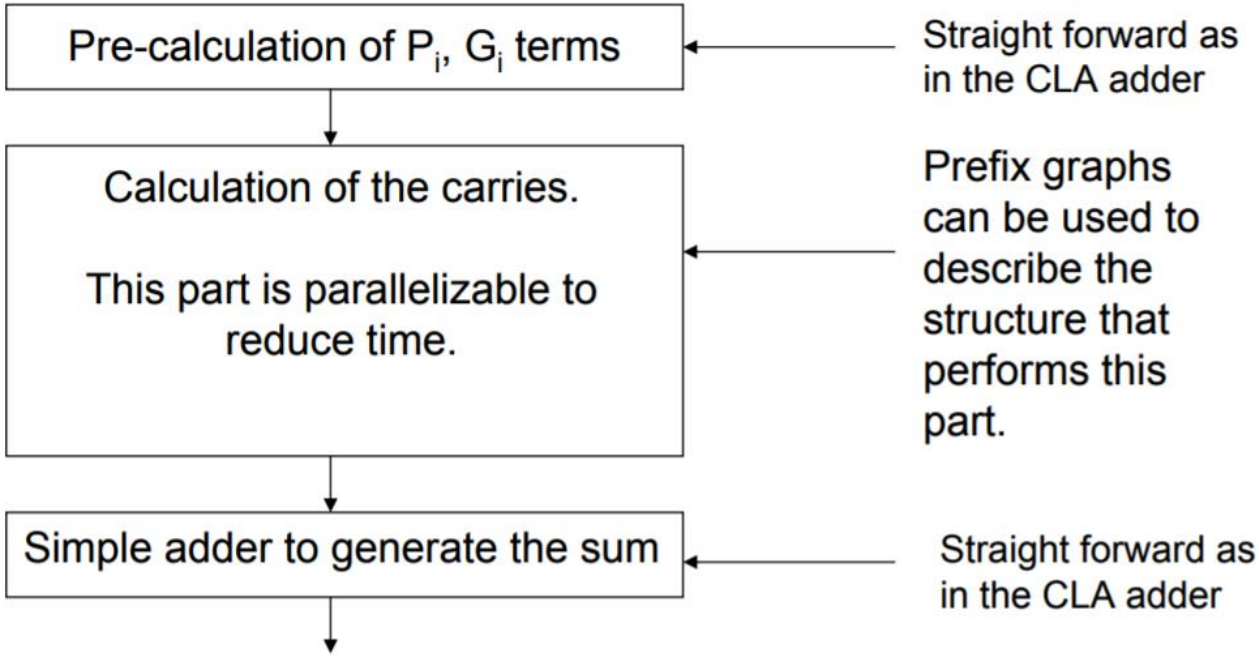
4-Bit Carry Look-ahead (CLA) Adder



P&G Based Adders – Parallel-prefix adder

Parallel Prefix Adder (PPA)

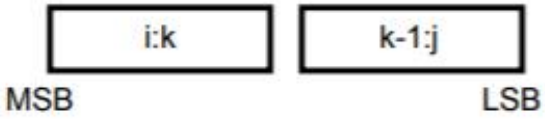
- The parallel prefix adder employs the 3-stage structure of the CLA adder. The improvement is in the carry generation stage which is the most intensive one:

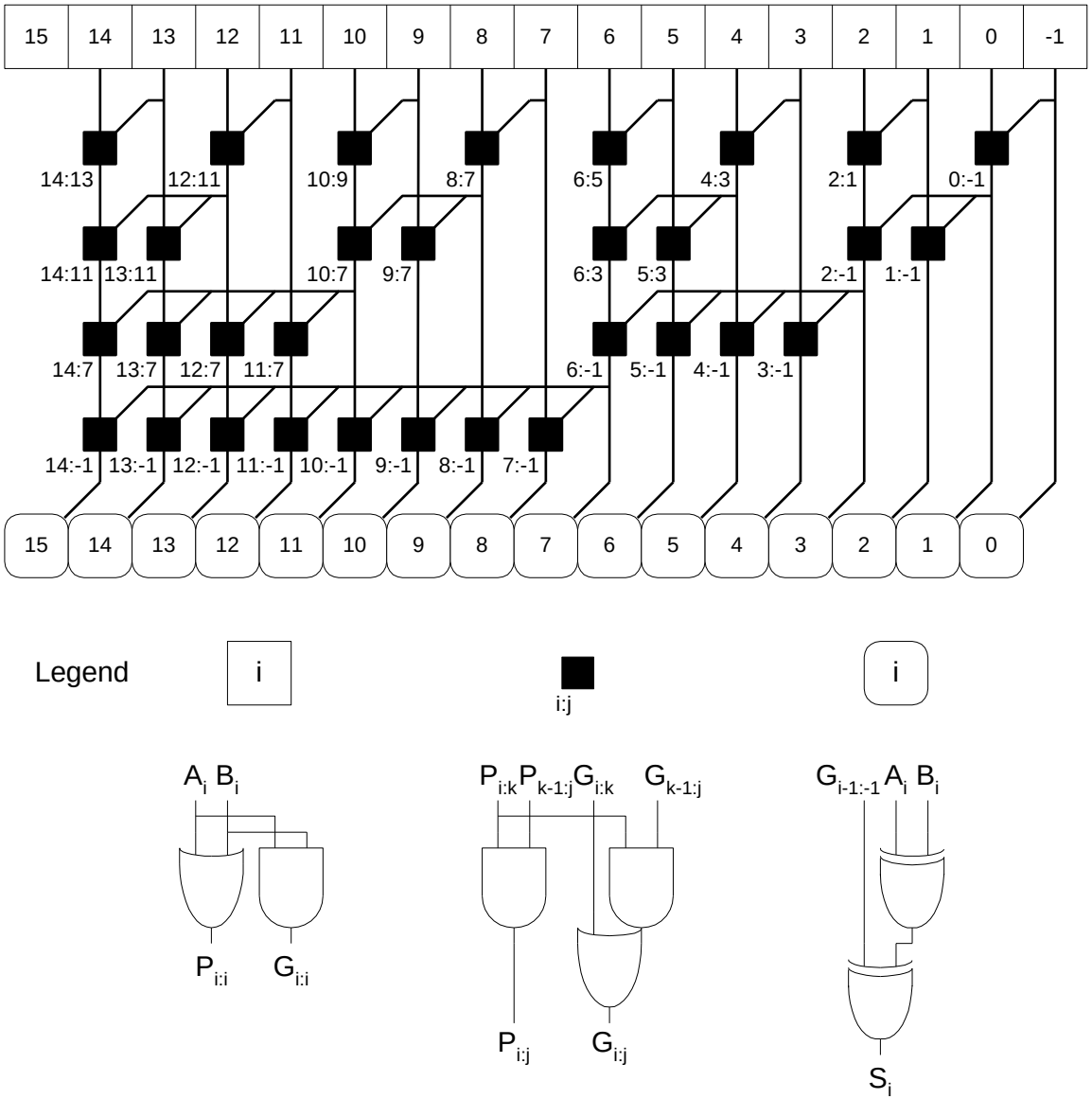


- 1960: J. Sklansky – conditional adder
- 1973: Kogge-Stone adder
- 1980: Ladner-Fisher adder
- 1982: Brent-Kung adder
- 1987: Han Carlson adder
- 1999: S. Knowles

Using P & G

- We can combine G_i and P_i into larger blocks
  - Call these "group generate" and "group propagate" terms
- When does a group Generate a carry out? (*e.g.*, 4 bits)
  - $G_{3:0} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$
- When does a group Propagate a carry in to the carry out?
  - $P_{3:0} = P_3P_2P_1P_0$
- We can also combine groups of groups
  - $G_{i:j} = G_{i:k} + P_{i:k}G_{k-1:j}$
  - $P_{i:j} = P_{i:k}P_{k-1:j}$




15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 -1
14:13 12:11 10:9 8:7 6:5 4:3 2:1 0:-1
14:11 13:11 10:7 9:7 6:3 5:3 2:-1 1:-1
14:7 13:7 12:7 11:7 6:-1 5:-1 4:-1 3:-1
14:-1 13:-1 12:-1 11:-1 10:-1 9:-1 8:-1 7:-1
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Legend
i
i:j
i
A_i B_i
$P_{i:i}$ $G_{i:i}$
$P_{i:k}$ $P_{k-1:j}$ $G_{i:k}$ $G_{k-1:j}$
$P_{i:j}$ $G_{i:j}$
$G_{i-1:-1}$ A_i B_i
S_i

Summary

- ✓ Prefix adder 와 Carry-skip adder는 _____ based adder 의 대표적인 분류이다.

- ✓ Prefix adder는 3단계의 구조를 지니며 첫단계는 각 bit position에 해당하는 _____ 을 만들어 내고 두번째 단계에서는 각 bit position 에 해당하는 _____ 을 만들어 내며, 마지막 단계에서는 각 bit position 에 해당하는 _____ 을 구한다.

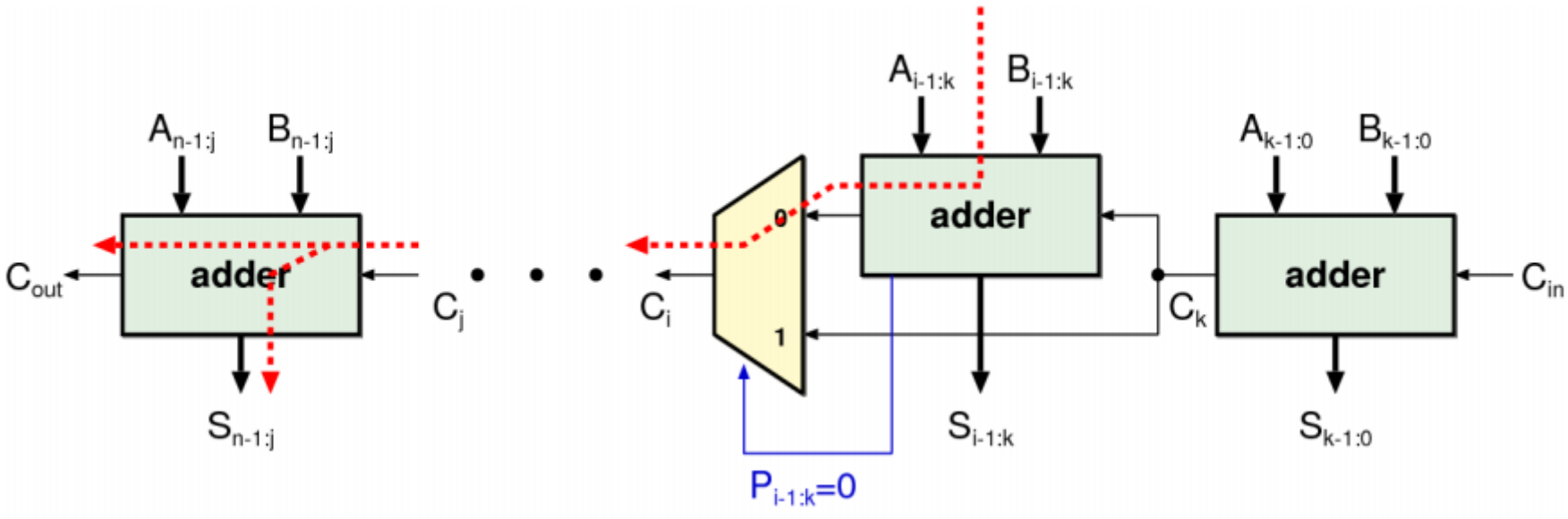
- ✓ Prefix 의 _____ 번째 단계의 구성 구조에 따라 adder 의 이름을 달리 부른다 (예: Kogge-Stone, Brent-Kung)

- ✓ 이론적으로 N-bit Adder를 Prefix adder로 구현할 때 delay는 _____ 적으로 비례하도록 할 수 있다.

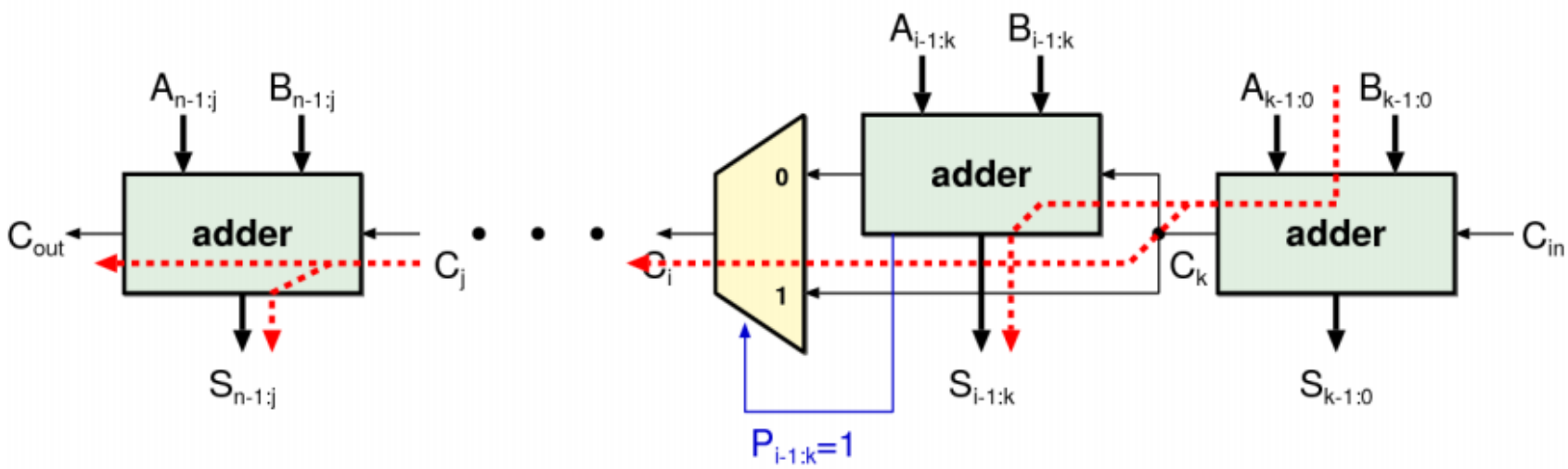
P&G Based Adders – Carry-skip adder

Carry-Skip Adder

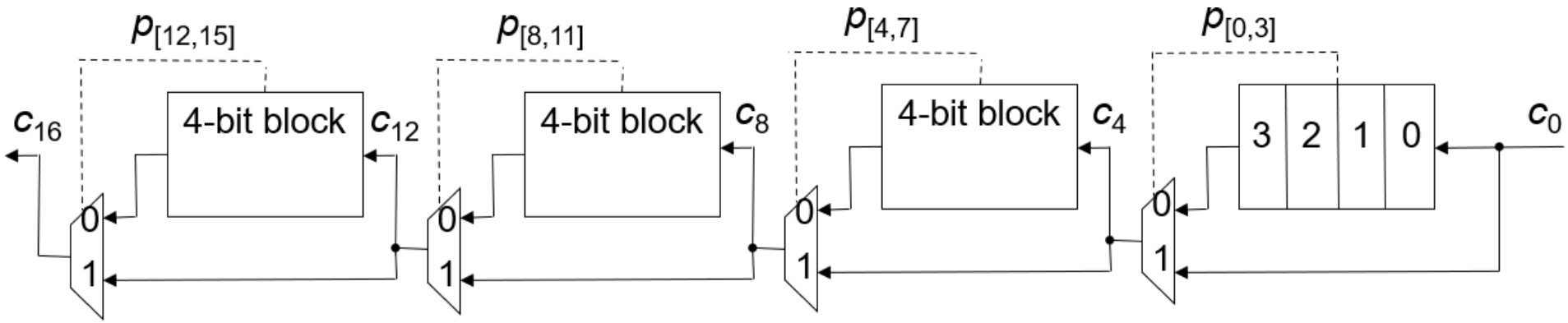
- Utilize the fact that the delay corresponding to the adder's longest path rarely happens.

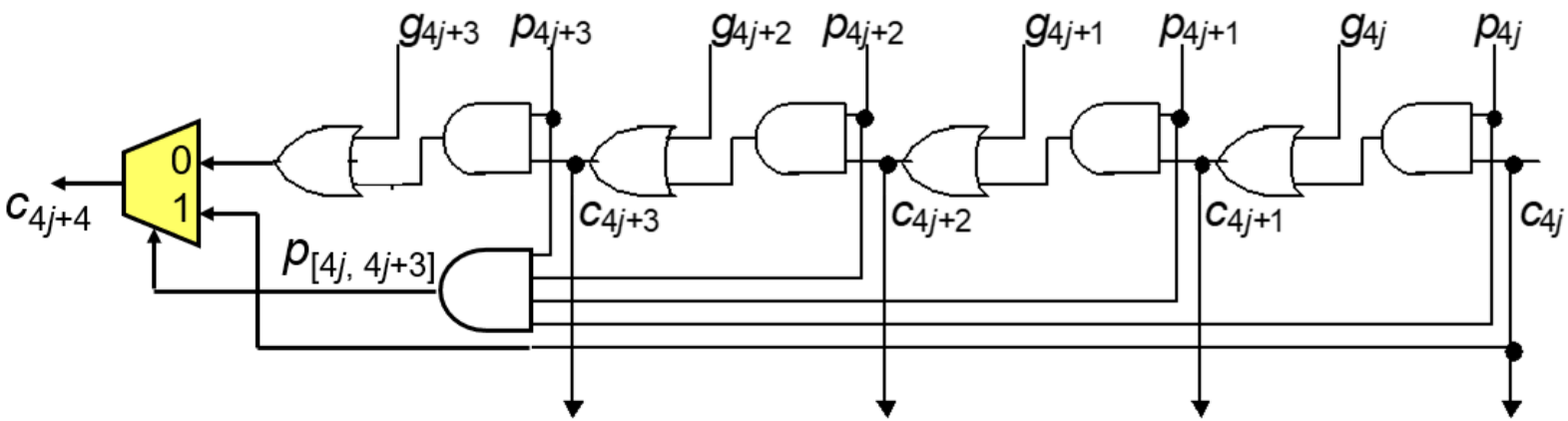


- If Pi-1:k = 0 (it means not all P values from bit i-1 to bit k are 1), the result of the carry is generated within this block.



- If Pi-1:k = 1, the carry of the previous block is propagated.


$p_{[12,15]}$
$p_{[8,11]}$
$p_{[4,7]}$
$p_{[0,3]}$
c_{16}
4-bit block
c_{12}
4-bit block
c_8
4-bit block
c_4
3
2
1
0
c_0
0
1
0
1
0
1
0
1


g_{4j+3}
p_{4j+3}
g_{4j+2}
p_{4j+2}
g_{4j+1}
p_{4j+1}
g_{4j}
p_{4j}
0
1
c_{4j+4}
c_{4j+3}
c_{4j+2}
c_{4j+1}
c_{4j}
$p_{[4j,\ 4j+3]}$

Multiplier – Partial products

multiplicand	1101	(13)
multiplier	* 1011	(11)
Partial products	1101	
	1101	
	0000	
	1101	
	10001111	(143)

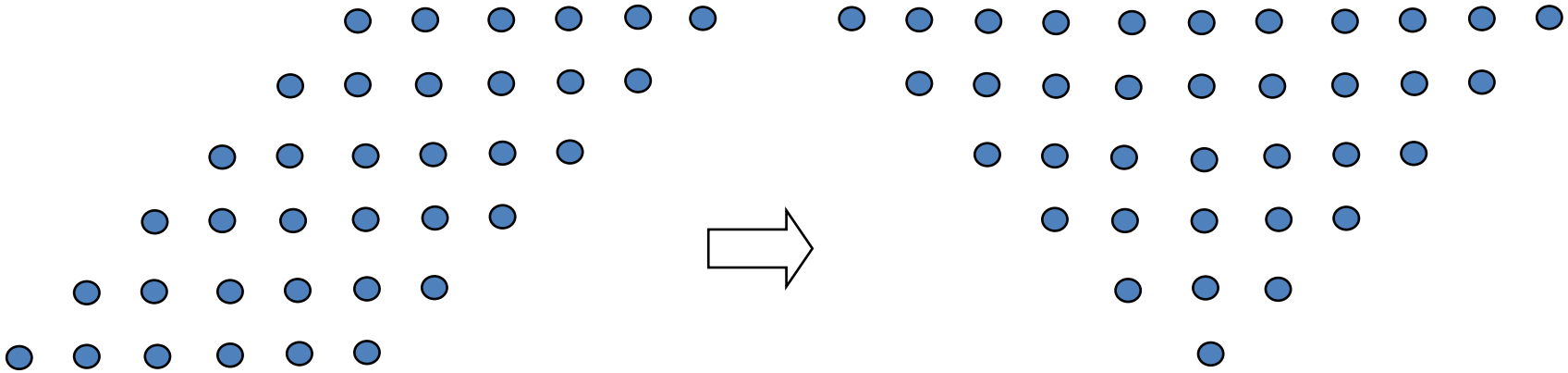
product of 2 4-bit numbers
is an 8-bit number

Partition Products

Partial Product Accumulation

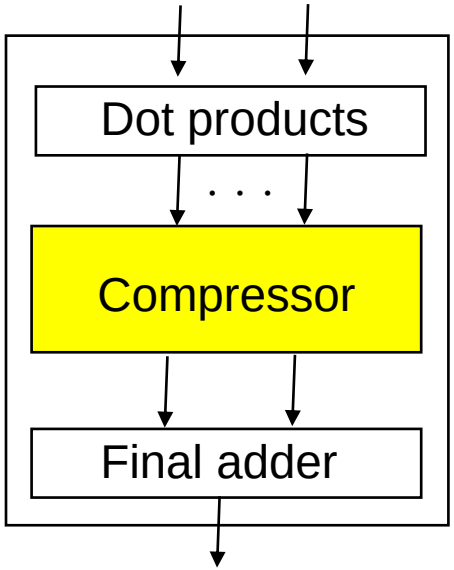
				A3	A2	A1	A0
				B3	B2	B1	B0
				A2 B0	A2 B0	A1 B0	A0 B0
			A3 B1	A2 B1	A1 B1	A0 B1	
		A3 B2	A2 B2	A1 B2	A0 B2		
	A3 B3	A2 B3	A1 B3	A0 B3			
S7	S6	S5	S4	S3	S2	S1	S0

Multiplier - Compressor



Original matrix of 36-bits

Reorganized matrix of bits



Parallel multiplier

FA based compressor

CSA based compressor

Multiplier – Recoding

Partial Product Generation using Canonical Recoding

Partial Product Generation using Booth Recoding

Bit	b7	b6	b5	b4	b3	b2	b1	b0	b-1
weight	-128	64	32	16	8	4	2	1	n/a (0)
d_3	-128	64	64						
d_2			-32	16	16				
d_1					-8	4	4		
d_0							-2	1	1

Summary

- Sequential multiplier
  회로 ____, 시간 ____ 걸림
- Parallel multiplier
  회로 ____, 시간 _____ 걸림
- Partial product reduction (compression)
  CSAs, FAs
- Partial product generation
  Canonical recoding, Booth recoding