Arithmetic Functions p.145

산술 기능을 수행하는 기능 블록

블록의 반복 배열

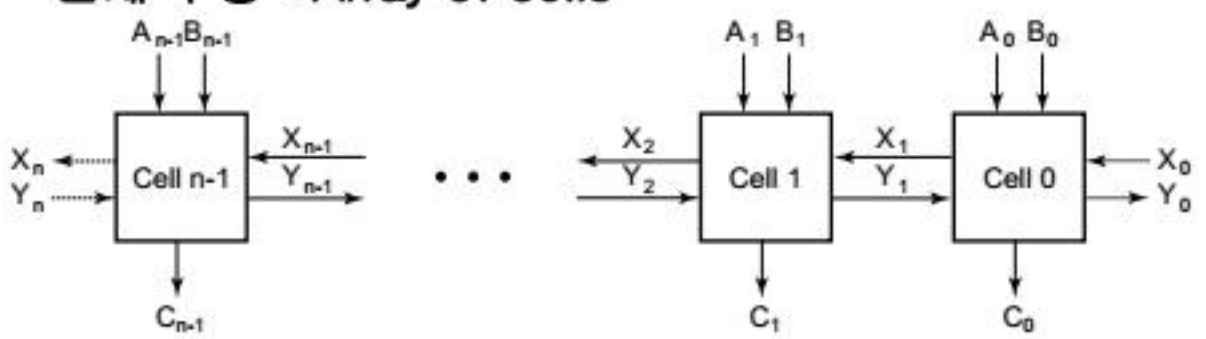
1

1

Iterative combinational circuits p.146

- Arithmetic blocks
  - 2진 벡터 입력 ➔ 2진 벡터 출력
  - 각 비트마다 적용되는 기본 블록이 반복 사용된다.
  - 기본 블록 : cell
  - 전체 구성 : Array of cells

A: (0001 0011)
B: (1101 0110)



2

2

Binary Adders p.147

- 2진 가산
  - 0+0=00 0+1=01 1+0=01 1+1=10
- 2진 가산기
  - 산술 연산 회로의 기본 블록
  - Half adder
    (두 개의 이진수의 합)
    - F(X,Y) ➔ S, C

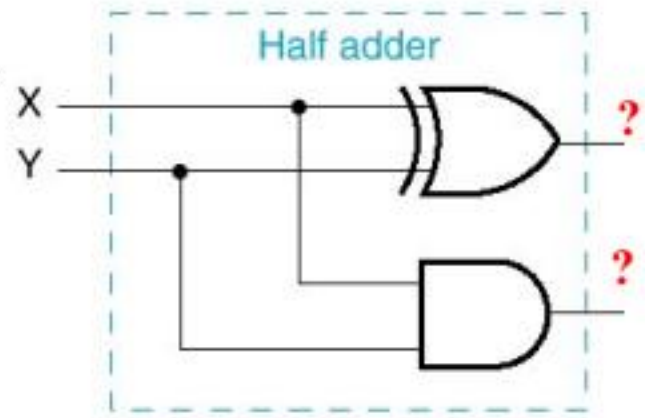
Inputs		Outputs	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

3

Half Adder p.147

- 반가산기(half adder)
  - 입력변수 : 2 개의 입력 X Y
  - 출력변수 : 합과 캐리 S C
  - S = (X'Y + XY') = X⊕Y
  - C = XY

입력		출력	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



4

Full Adders p.147

- Full adder
  (세 개의 이진수의 합)
- F(X,Y,Z) ➔ S,C

A_i + B_i
+ (i−1) 비트로부터의 carry
➔ C, S

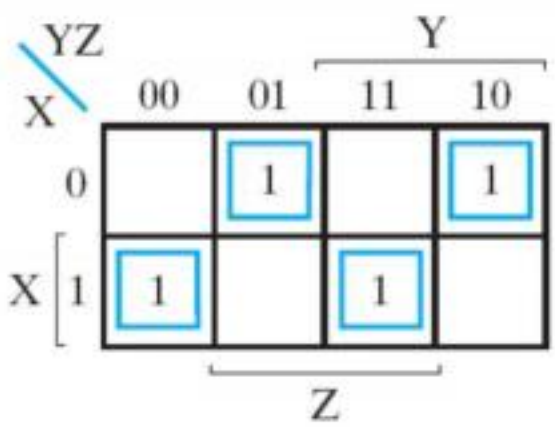
입력			출력	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

5

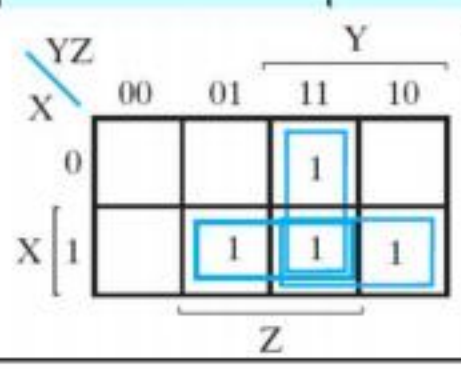
5

Full Adder p.148

- 전가산기(full adder)
  - S = X'Y'Z + X'YZ'
    + XY'Z' + XYZ
  - C = XY+XZ+YZ



입력			출력	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



6

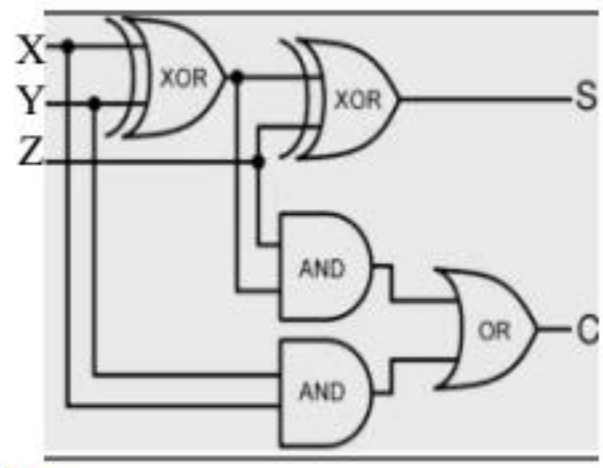
6

Full adder의 설계

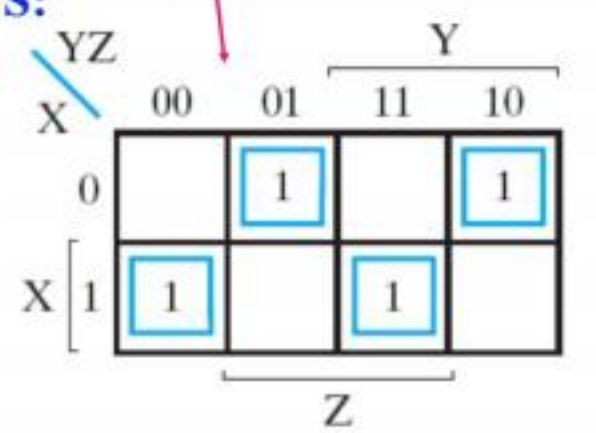
- S는 **홀수함수**로 표현 가능

$$S = (X \oplus Y) \oplus Z$$

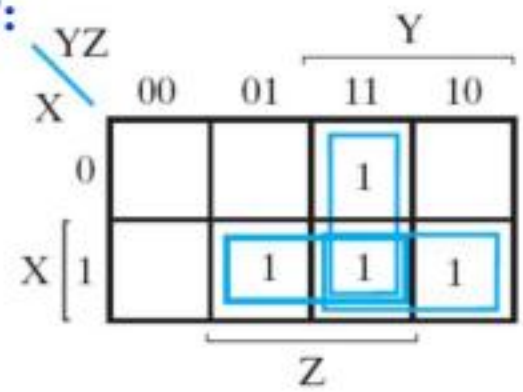
$$C = XY + XZ + YZ$$



S:



C:



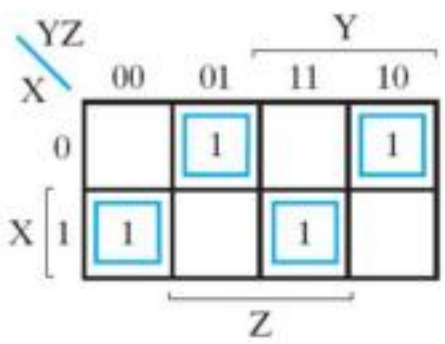
7

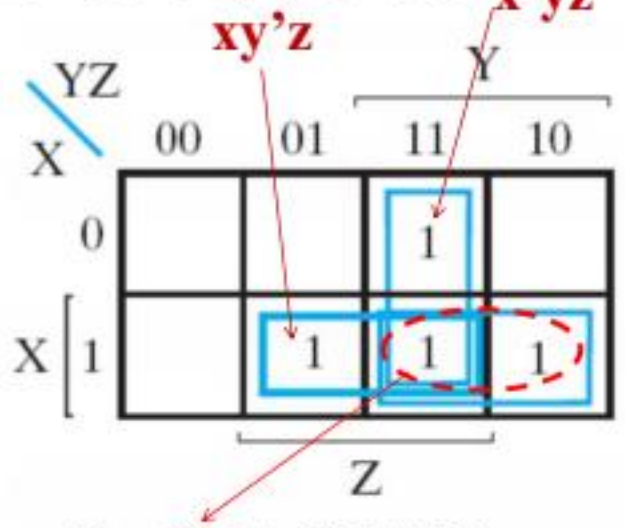
7

Full Adder p.148

$S = X \oplus Y$
$C = XY$

- 다른 구현 ➔Half Adder를 이용하여 구현가능
- $S = (X \oplus Y) \oplus Z$





$C = XY + \mathbf{XY'Z + X'YZ}$
$= XY + Z(XY' + X'Y)$
$= XY + Z(X \oplus Y)$

$C = XY + XZ + YZ$
$= XY + Z(X\overline{Y} + \overline{X}Y)$
$= XY + Z(X \oplus Y)$

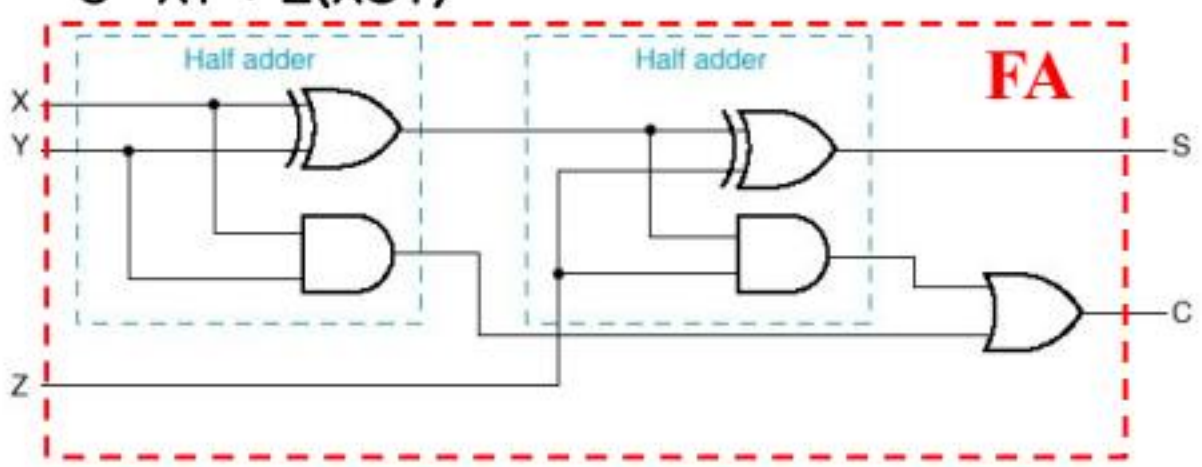
8

8

Full Adder (FA) p.148

- 다른 구현(cont'd) **(half adder 2개 + OR gate 1개)**
  - S = (X⊕Y)⊕Z
  - C를 S의 게이트 (X⊕Y)를 이용하도록 수정

    C= XY + Z(X⊕Y)



9

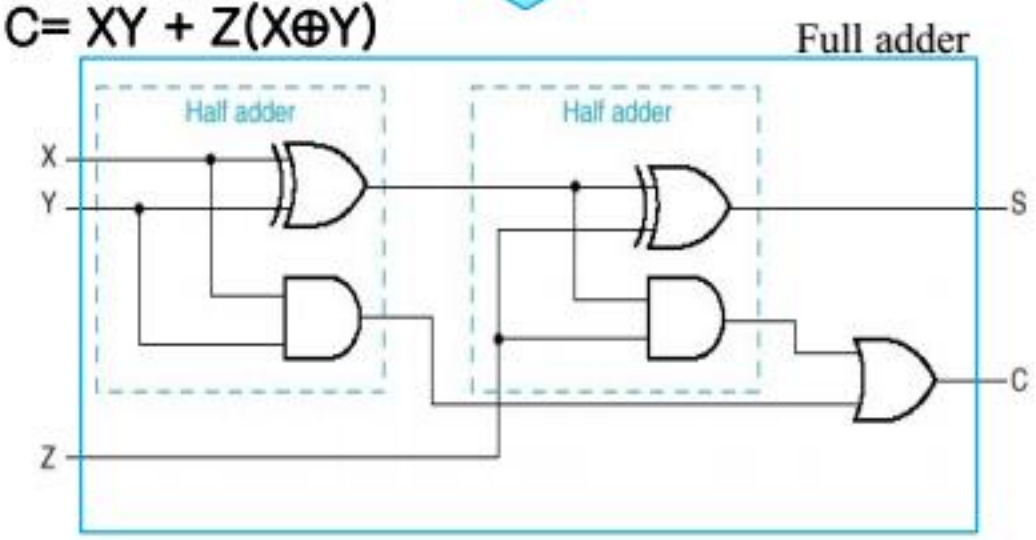
9

S = X'Y'Z + X'YZ' + XY'Z' + XYZ

C = XY+XZ+YZ

복잡도를 낮추다.

- S = (X⊕Y)⊕Z
- C= XY + Z(X⊕Y)



10

10

FA의 배열(조합회로의 반복) p.149

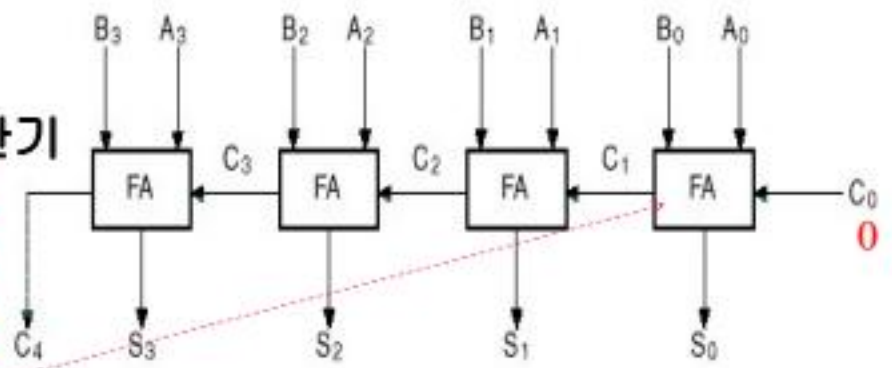
예) 4비트 가산기

C : 0 0110

A : 1011

B : 0011

S : 1110



계층적 설계
기본 블록(FA)의 재사용

11

11

Binary Ripple Carry Adder p.149

- 2진 리플 캐리 가산기
  (n개의 전가산기를 이용한 n비트 가산기)
  - 병렬2진 가산기
  - 캐리의 전파 : 잔물결(ripple) 모양

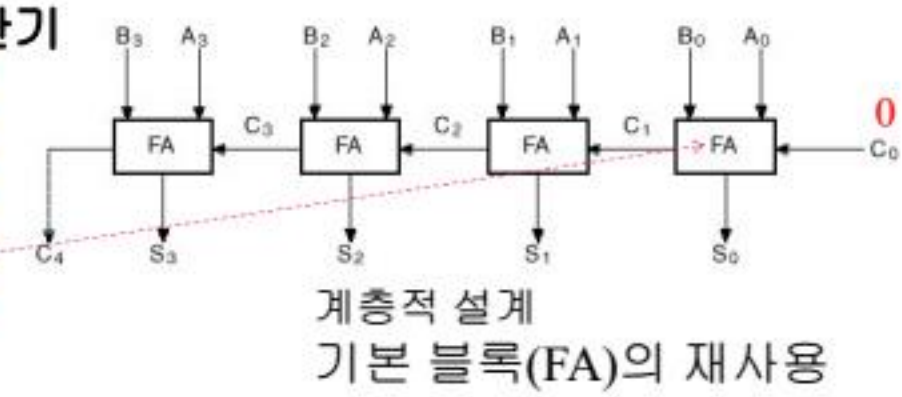
예) 4비트 가산기

C : 0 0110

A : 1011

B : 0011

S : 1110

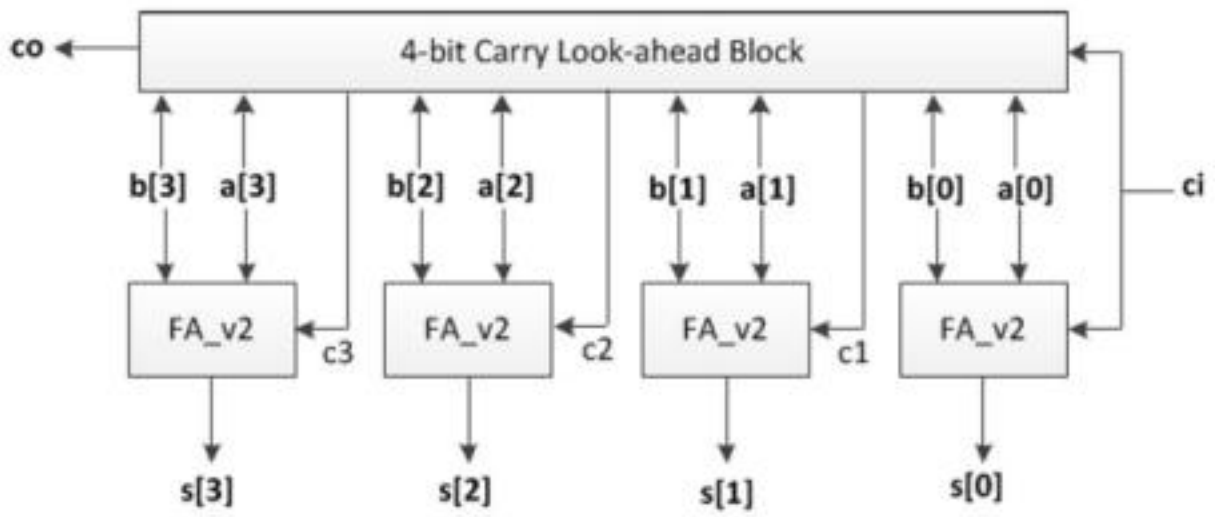


계층적 설계
기본 블록(FA)의 재사용

12

12

Carry Look-ahead



13

13

$C = XY+XZ+YZ=XY+Z(X+Y)$ ➔ $AB+C(A+B)$

Carry Look-ahead

$C_1 = A_0B_0+C_0(A_0+B_0)$

$C_0 = 0$

$C_1 = A_0B_0+C_0(A_0+B_0)$
$= G_0 + C_0(P_0)$

$C_2 = A_1B_1+C_1(A_1+B_1)$
$= G_1 + C_1(P_1) = G_1 + (G_0 + C_0(P_0))(P_1)$
$= G_1 + P_1G0 + P_1P_0C_0$

$C_3 = A_2B_2+C_2(A_2+B_2)$
$= G_2 + C_2(P_2)$
$= G_2 + (G_1 + P_1G0 + P_1P_0C_0)(P_2)$
$= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$
:
:

G,P정의

$G_i = A_iB_i$

$P_i = A_i + B_i$

C : 0
A : 1010
B : 0011

look-carry ahead

0100

14

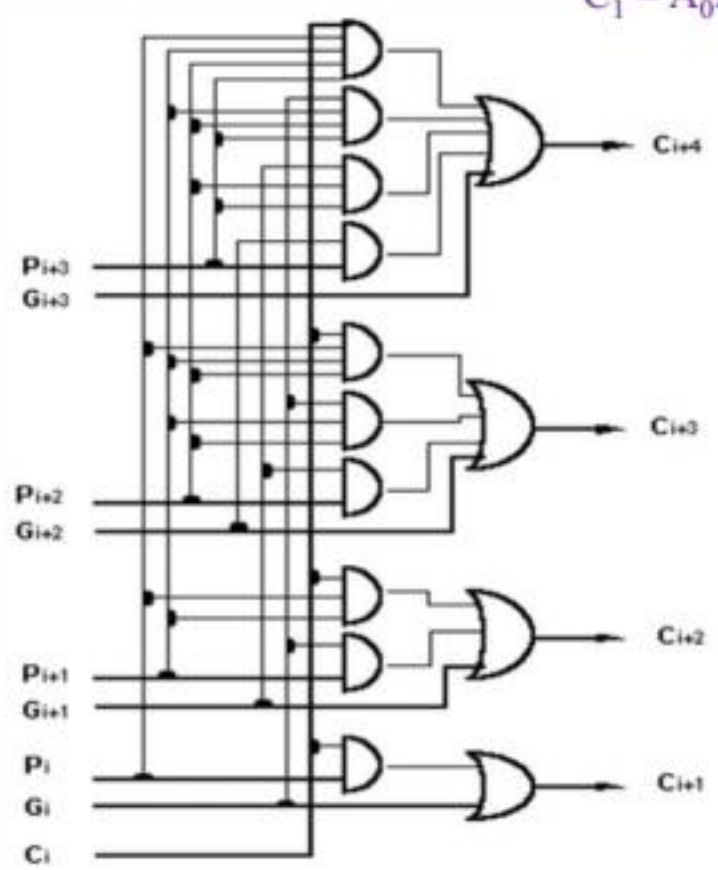
14

$C_1 = A_0B_0 + C_0(A_0 + B_0)$

G,P정의

$$G_i = A_iB_i$$

$$P_i = A_i + B_i$$



$C4 = G3 + P3 \cdot G2 + P3 \cdot P2 \cdot G1 + P3 \cdot P2 \cdot P1 \cdot G0 + P3 \cdot P2 \cdot P1 \cdot P0 \cdot C_0$

$C3 = G2 + P2 \cdot G1 + P2 \cdot P1 \cdot G0 + P2 \cdot P1 \cdot P0 \cdot C_0$

$C2 = G1 + P1 \cdot G0 + P1 \cdot P0 \cdot C_0$

$C1 = G0 + P0 \cdot C_0$

$C0 = 0$

15

15

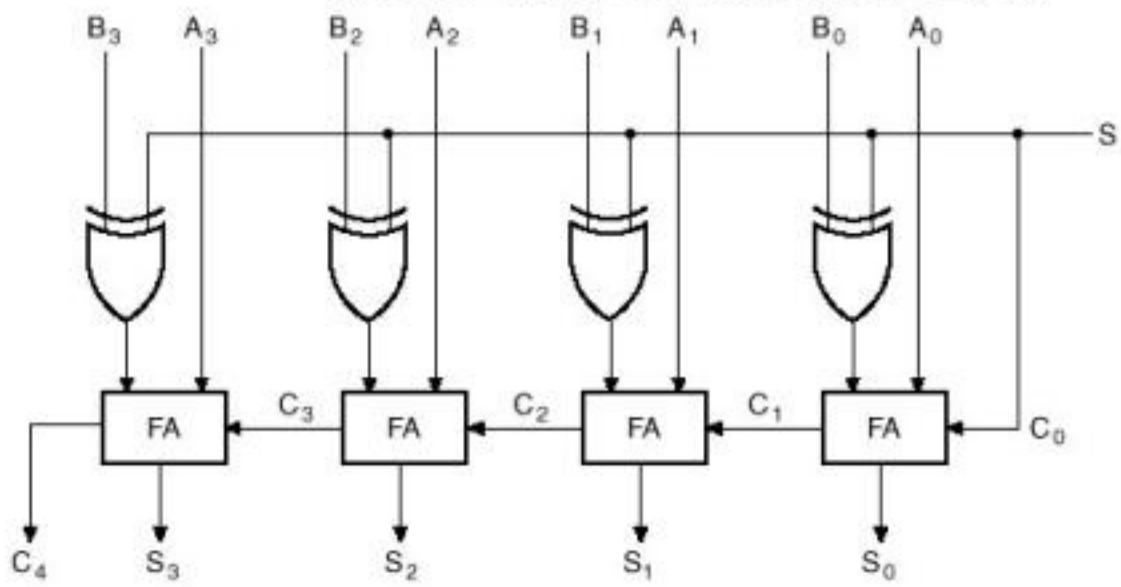
CRA(carry ripple adder)
+ CLA(carry lookahead adder)
➔ 다양한 변형이 제안되고 사용되고 있다.

16

16

2진 가산기-감산기 p.155

- Signed integer 가산기/감산기
- 두 연산 대상 A와 B의 부호를 전혀 신경쓰지 않는다.
  부호비트까지 포함하여 계산한다.

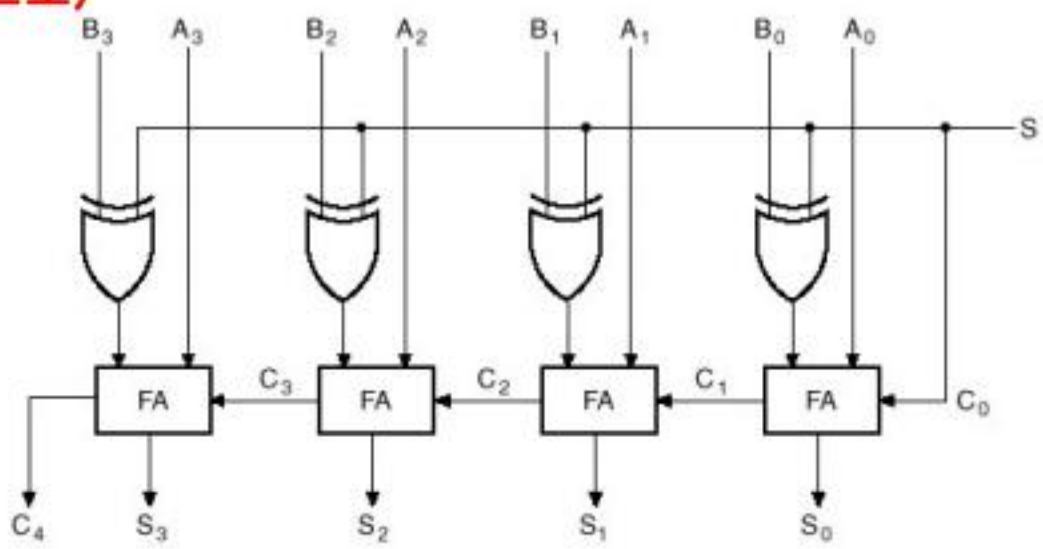


17

17

감산기 = 가산기 + 보수기 p.155

- If A + B, S = 0
- If A − B, S = 1
- ★★결과가 음수라면 그대로 해석 (2의 보수 시스템 이므로)

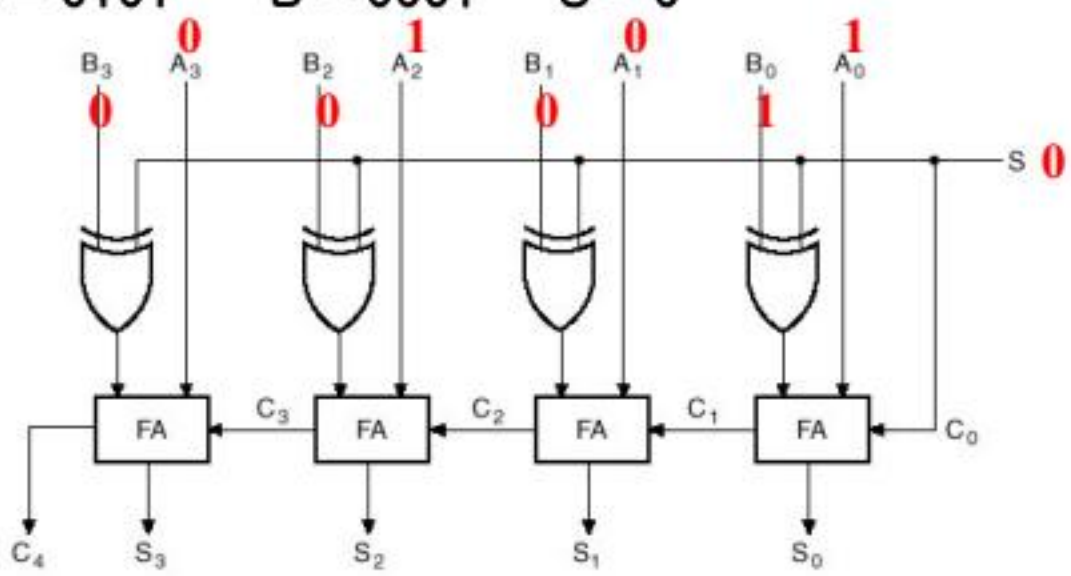


18

18

Binary Adder-Subtractors p.155

- 해석하자 (1)
- A = 0101 B = 0001 S = 0

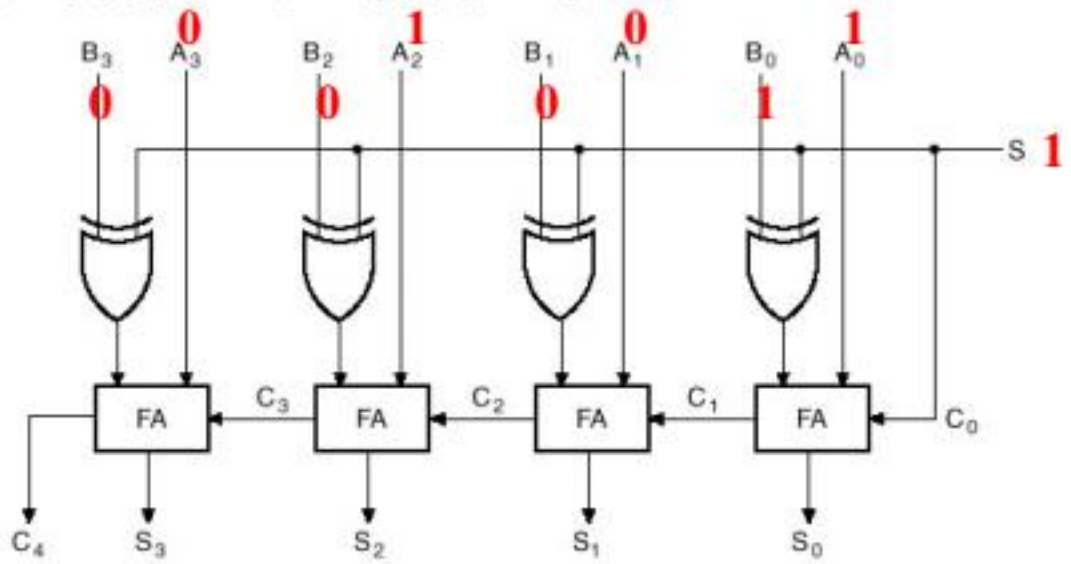


19

19

Binary Adder-Subtractors p.155

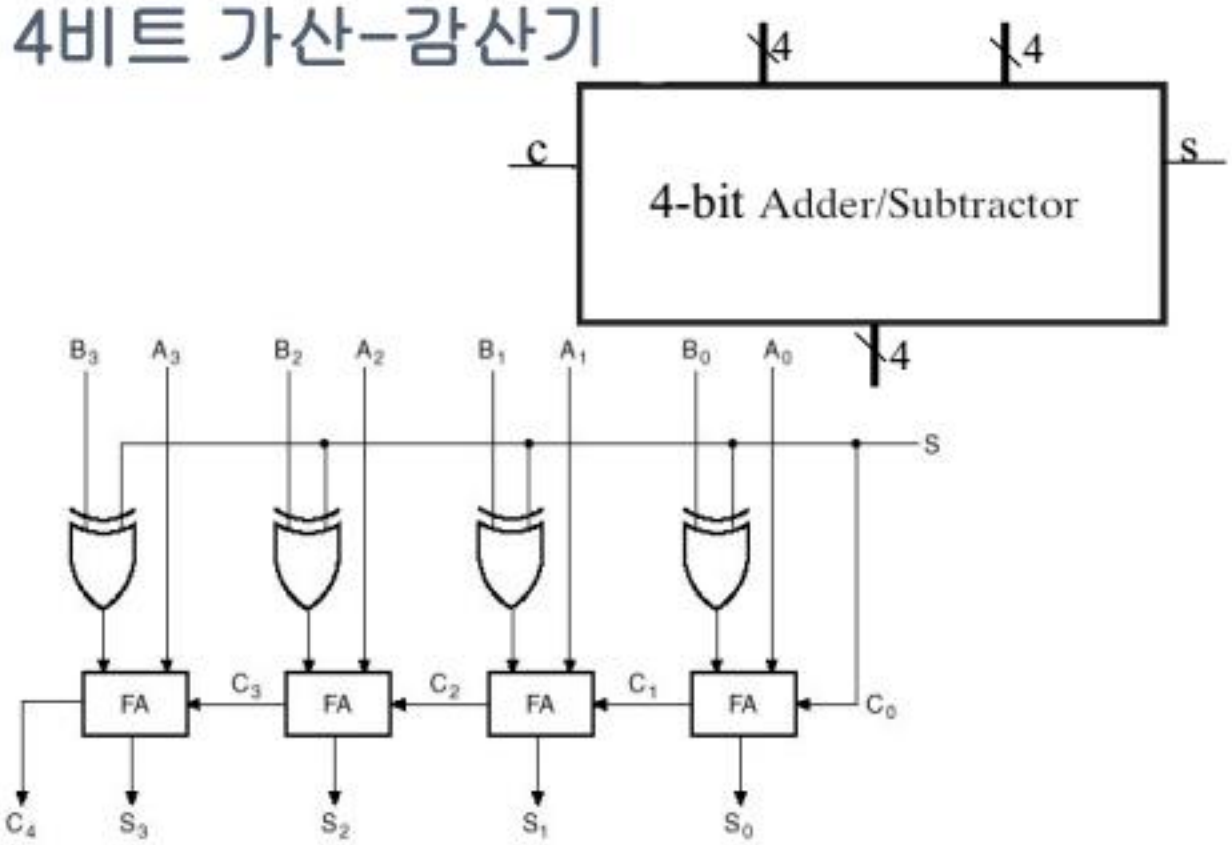
- 해석하자 (2)
- A = 0101 B = 0001 S = 1



20

20

4비트 가산-감산기



21

21

Overflow p.159,160

- 오버플로(overflow)
  - n 비트 연산결과가 n+1 비트를 차지할 때
    - 결과 값이 너무 커서 주어진 비트로 표현할 수 없다.
- 해결 : 오버플로 검출
- Overflow 예 (검출 특징은 무엇인가?)

8 비트 signed 정수의 표현범위 : + 127 ~ -128

01 *000 0000*	***10*** *110 0000*	*11 110 0000*	*00 000 0000*
+ 70 : 0 100 0110	- 70 : 1 011 1010	- 70 : 1 011 1010	+ 70 : 0 100 0110
+ 80 : 0 101 0000	- 80 : 1 011 0000	+ 80 : 0 101 0000	- 80 : 1 011 0000
+150 : 1 001 0110	-150 : 0 110 1010	+ 10 : 0 000 1010	- 10 : 1 111 0110

+106으로 해석 (오류)

-106으로 해석 (오류)

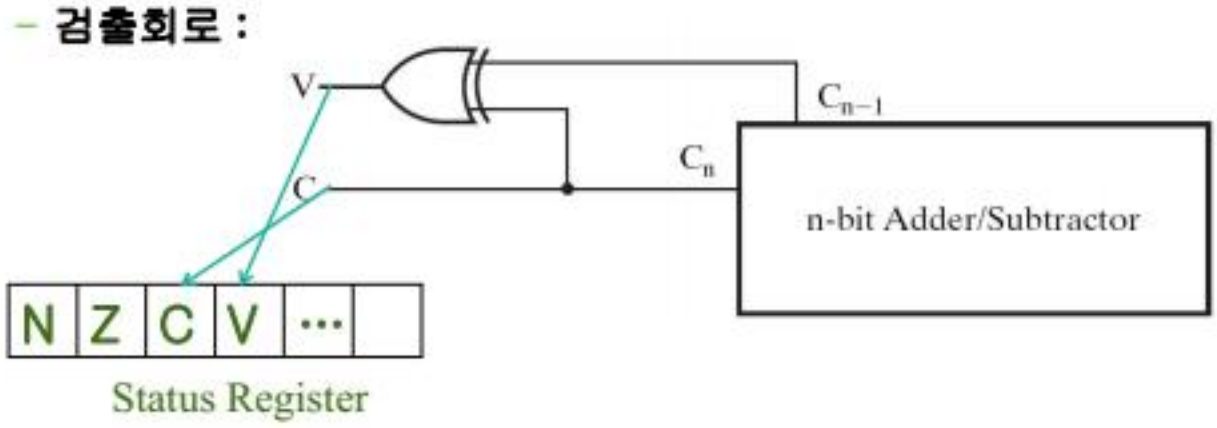
22

22

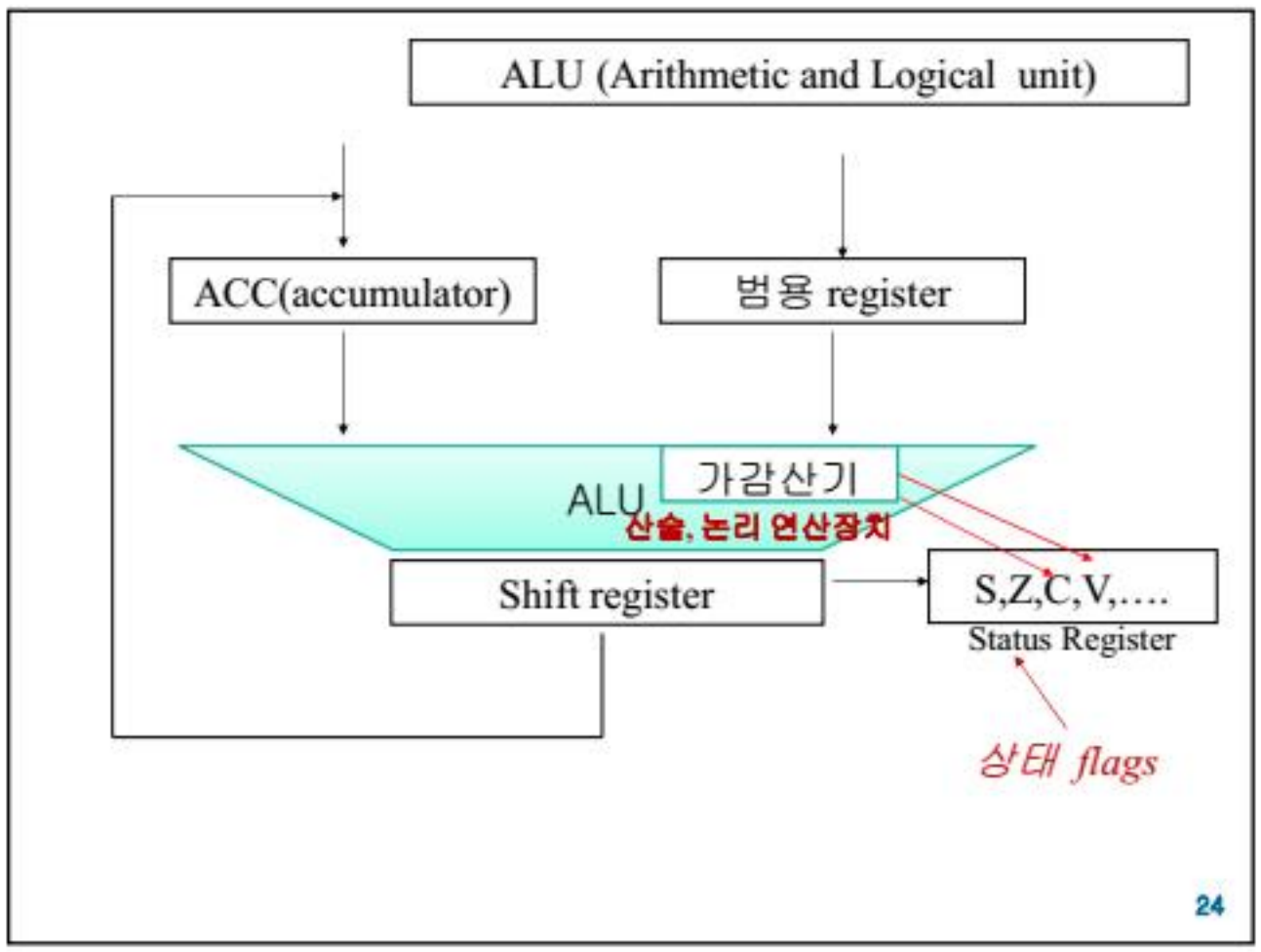
Overflow Detection Logic (signed)

p.176

- 부호화된 숫자의 가산-감산에서의 오버플로 검출
  - Overflow 감지 가정 : 8비트 정수(부호 포함)
    - C8 ≠ C7일때 오버플로
    - Overflow 검출 회로 : XOR로 구현
  - 검출회로 :



23



정리

- Half Adder
- Full Adder
- Ripple carry adder
- Binary adder-subtractor (가산기 + 보수기)
- overflow

25