MSI/LSI 연산회로

10주차 발표

20191048 김도솔 | 20211600 지소현 | 20221595 이선우

MSI/LSI 연산회로

Index

01 4-Bit Binary Parallel Adder

4비트 이진 병렬 가산기

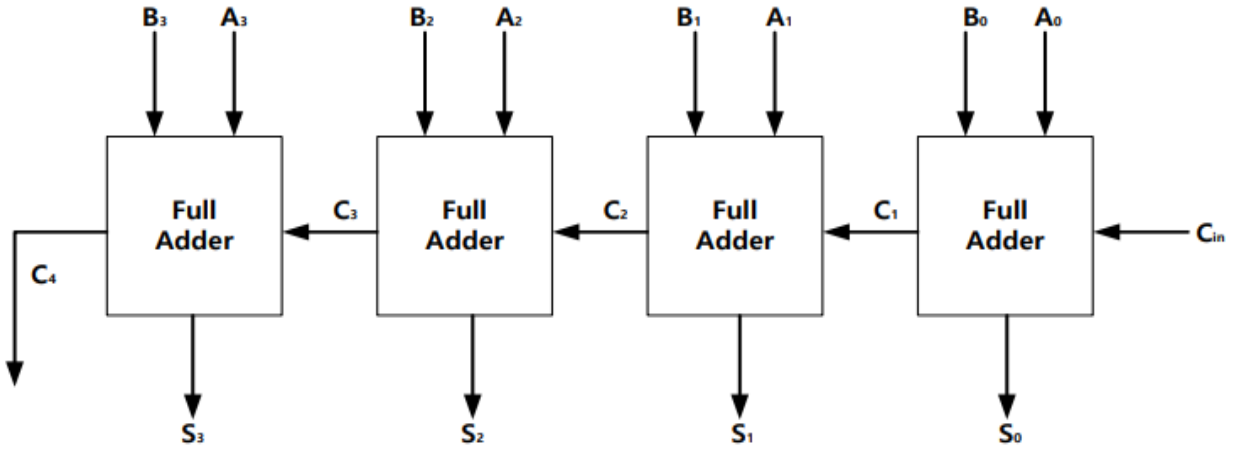
Parallel Adder (병렬 가산기)

: 전가산기 여러 개를 병렬로 연결한 가산기

계산 결과

(합) = $S_3S_2S_1S_0$

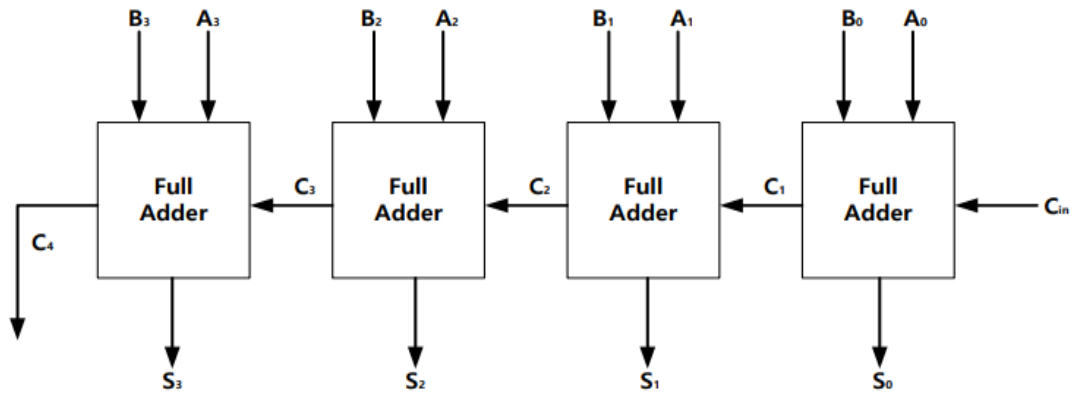
(캐리) = C_4



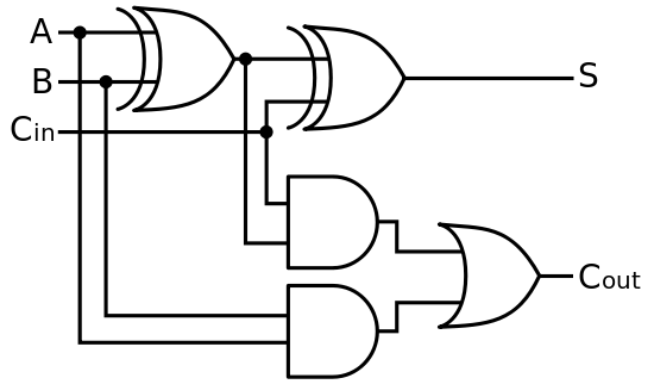
- 각 전가산기의 출력 캐리가 그 다음 차수의 전가산기의 캐리 입력에 연결된 전가산기가 각 비트 쌍에 대해 병렬로 연쇄적으로 연결되어 있다.
- 일반적으로 연속적인 가산 단계 간의 캐리 전파가 가산 속도를 제한하지 않도록 하기 위해 캐리 전파 루틴을 포함한다.

01 4-Bit Binary Parallel Adder

Full Adder(전가산기) vs N-bit Binary Adder



[N-bit adder]

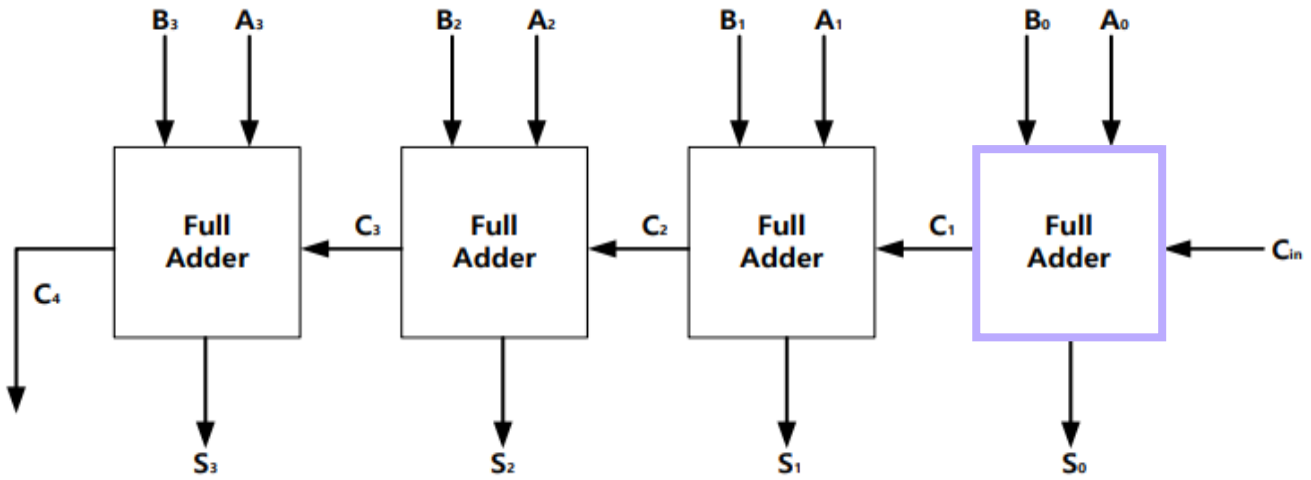


[Full Adder]

- **Full Adder:** 한 비트 숫자와 입력 carry를 더하는 작업
- **N-bit adder:** 산술 합을 병렬로 처리하여 한 비트 이상의 이진 숫자의 합을 찾는 디지털 회로. 작업을 수행하기 위해 n개의 전가산기가 필요.

01 4-Bit Binary Parallel Adder

동작 원리

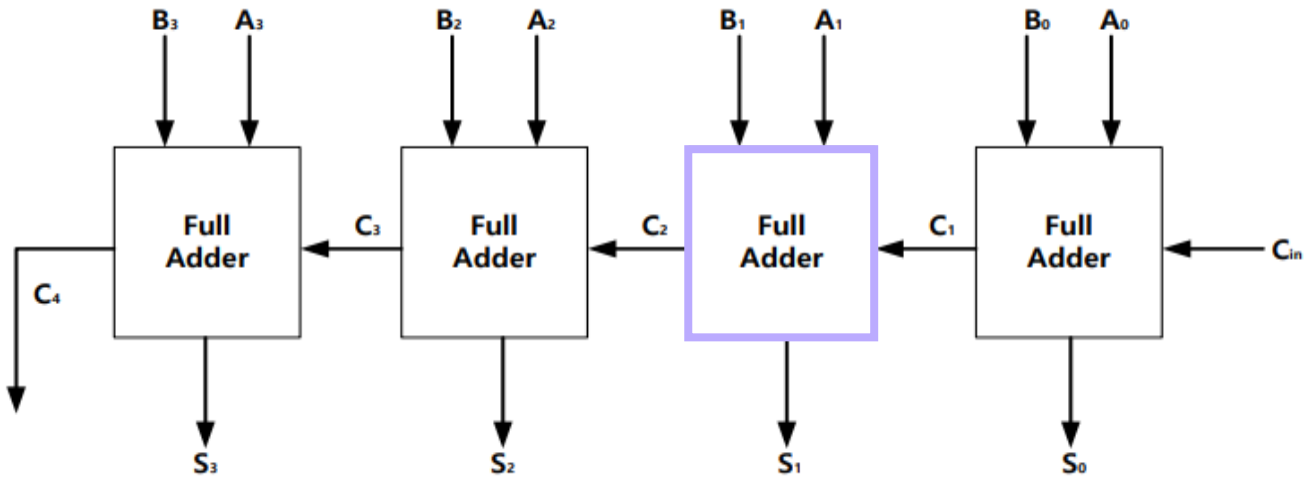


1) 첫 번째 전가산기: A_0, B_0, 캐리 C_{in}을 입력으로 받고, 출력 합 S_0와 캐리 C_1을 생성한다.

이때 S_0는 출력 합의 첫 번째 비트를 나타내고, C_1은 다음 전가산기에 연결된다.

01 4-Bit Binary Parallel Adder

동작 원리

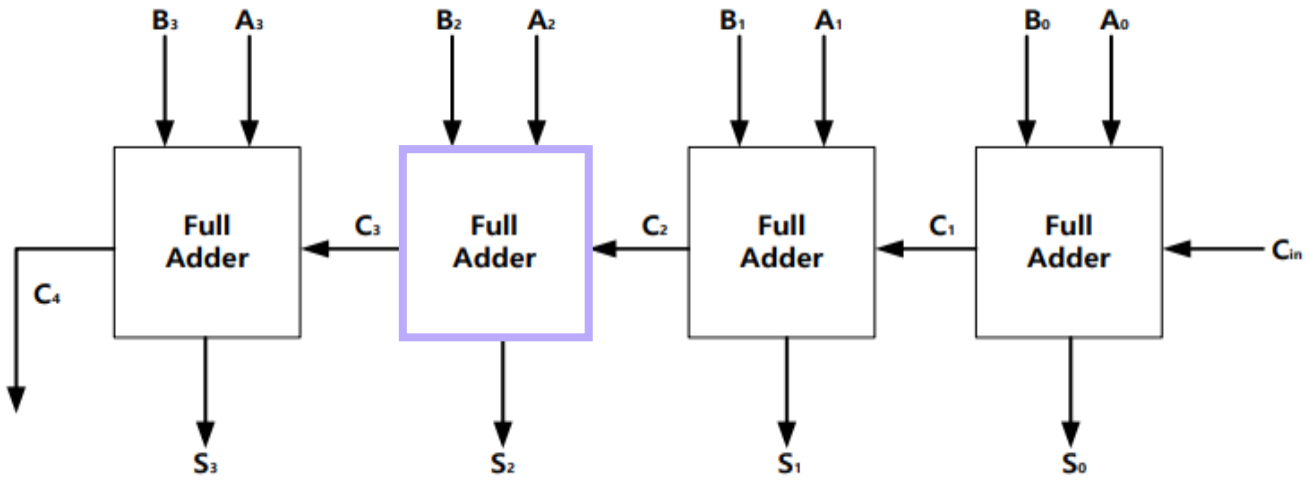


2) 두 번째 전가산기: A_1, B_1, 캐리 C_1을 입력으로 받고, 출력 합 S_1와 캐리 C_2를 생성한다.

이때 S_1은 출력 합의 두 번째 비트를 나타내고, C_2는 다음 전가산기에 연결된다.

01 4-Bit Binary Parallel Adder

동작 원리

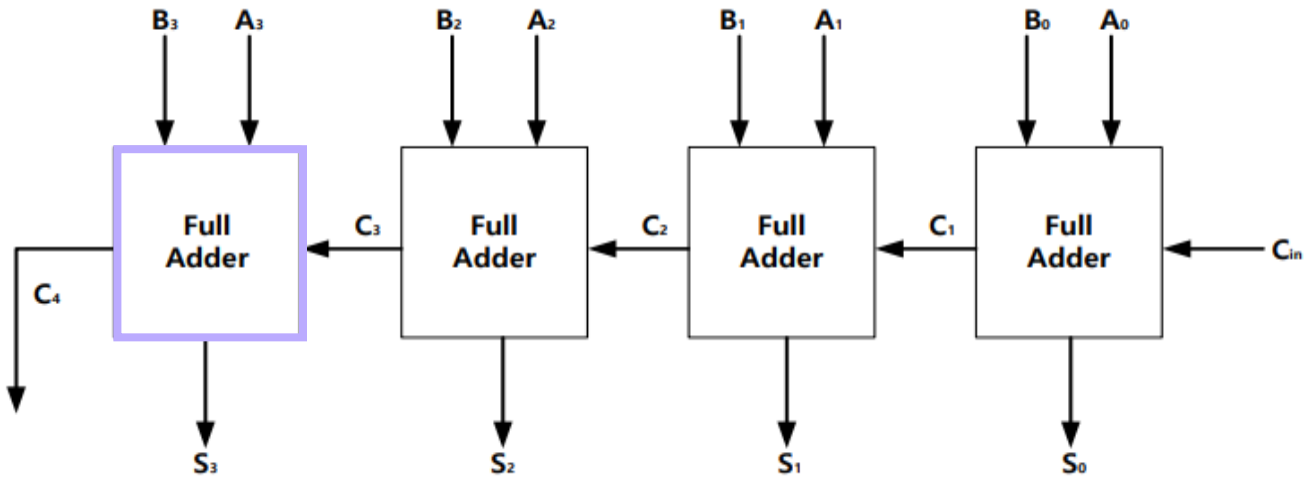


3) 세 번째 전가산기: A_2, B_2, 캐리 C_2을 입력으로 받고, 출력 합 S_2와 캐리 C_3를 생성한다.

이때 S_2는 출력 합의 세 번째 비트를 나타내고, C_3는 다음 전가산기에 연결된다.

01 4-Bit Binary Parallel Adder

동작 원리



4) 네 번째 전가산기: A_3, B_3, 캐리 C_3을 입력으로 받고, 출력 합 S_3와 캐리 C_4를 생성한다.

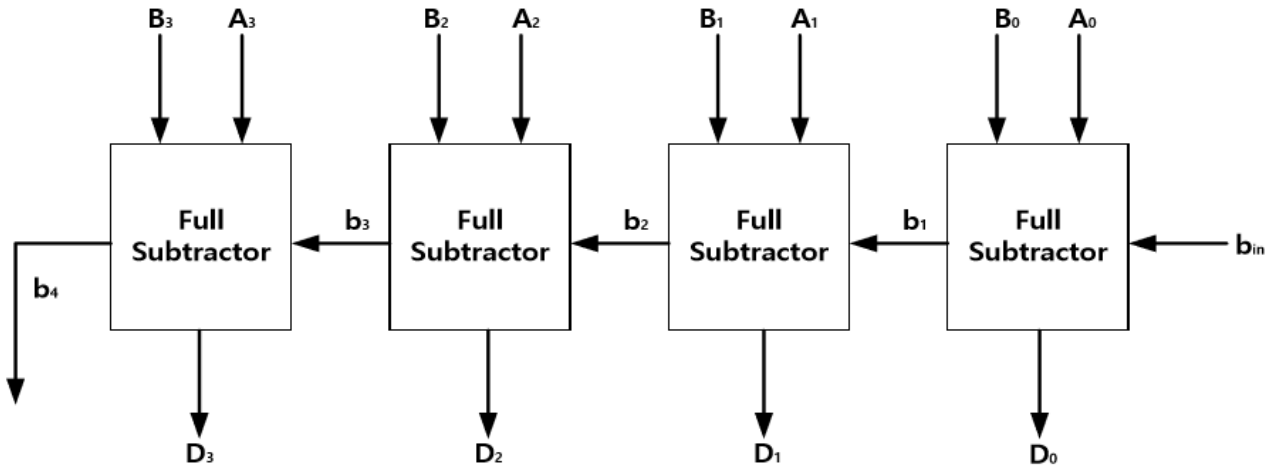
이때 S_3는 출력 합의 네 번째 비트를 나타내고, C_4는 최종 출력 캐리가 된다.

02 4-Bit Binary Parallel Subtractor

4비트 이진 병렬 감산기

Parallel Subtractor (병렬 감산기)

: 병렬로 동작하여 두 이진 숫자의 산술 차이를 찾을 수 있는 감산기



계산 결과

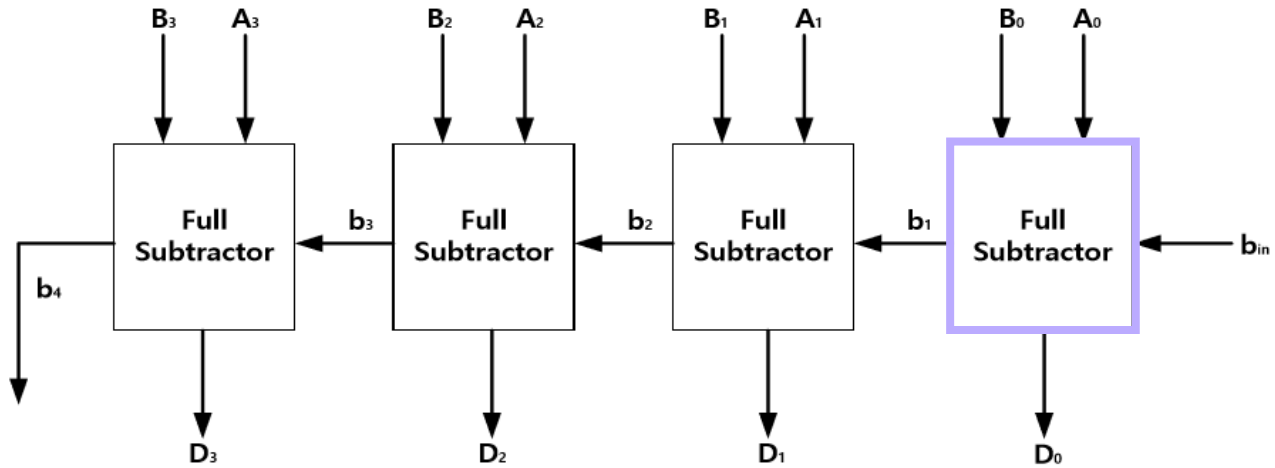
(차이) = $D_3D_2D_1D_0$

(빌림) = b_4

- 2진수의 길이가 1비트 이상인 경우에 사용된다.
- 반감산기와 전감산기의 조합, 또는 감수의 보수 입력을 사용하는 전가산기의 조합 등 다양하게 설계할 수 있다.

02 4-Bit Binary Parallel Subtractor

동작 원리

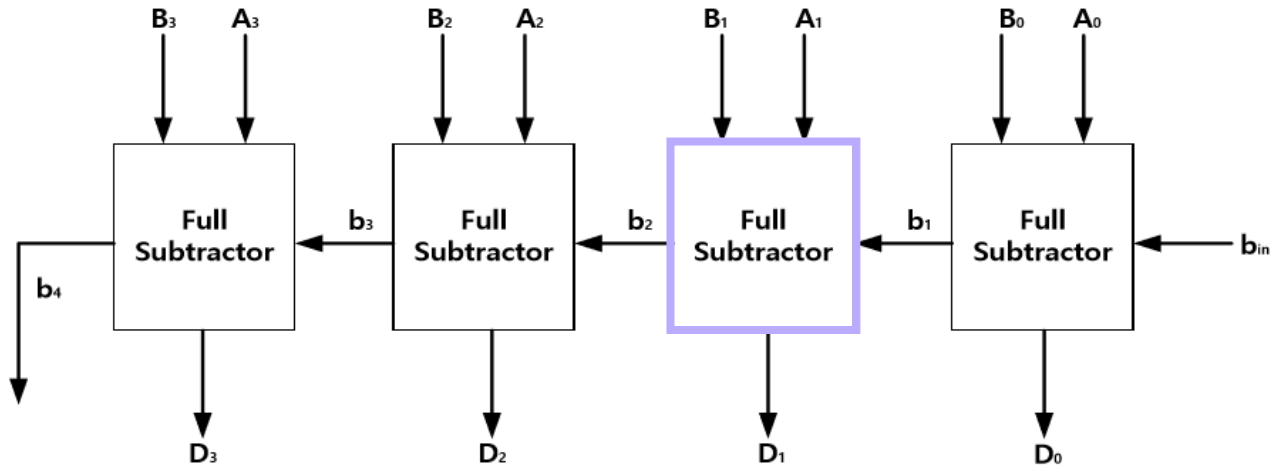


1) 첫 번째 전감산기: A_0, B_0, 빌림 b_{in}을 입력으로 받고, 차이 D_0와 빌림 b_1을 생성한다.

이때 D_0는 차이 출력의 첫 번째 비트를 나타내고, b_1은 다음 전감산기에 연결된다.

02 4-Bit Binary Parallel Subtractor

동작 원리

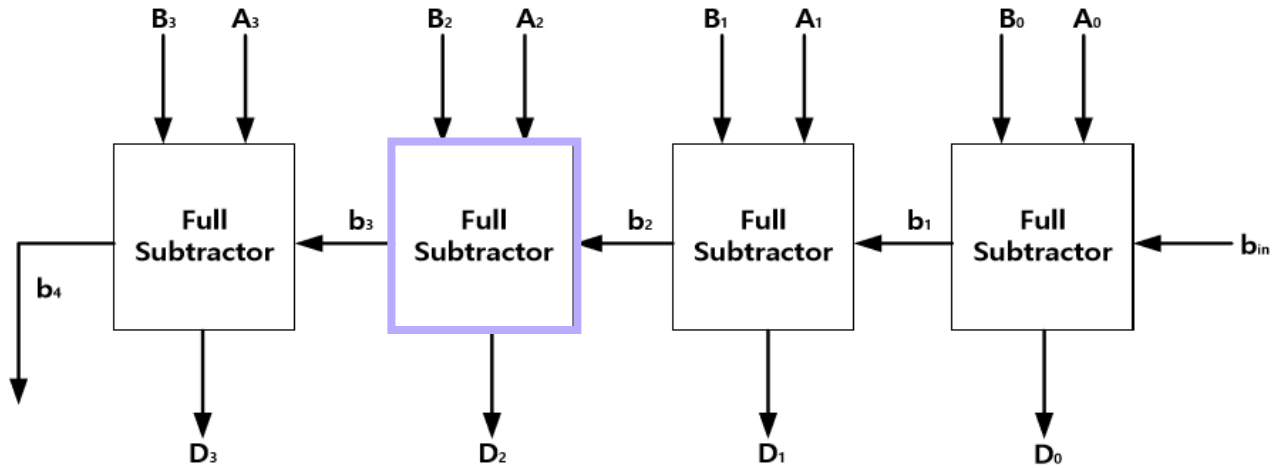


2) 두 번째 전감산기: A_1, B_1, 빌림 b_1을 입력으로 받고, 차이 D_1와 빌림 b_2를 생성한다.

이때 D_1은 차이 출력의 두 번째 비트를 나타내고, b_2는 다음 전감산기에 연결된다.

02 4-Bit Binary Parallel Subtractor

동작 원리

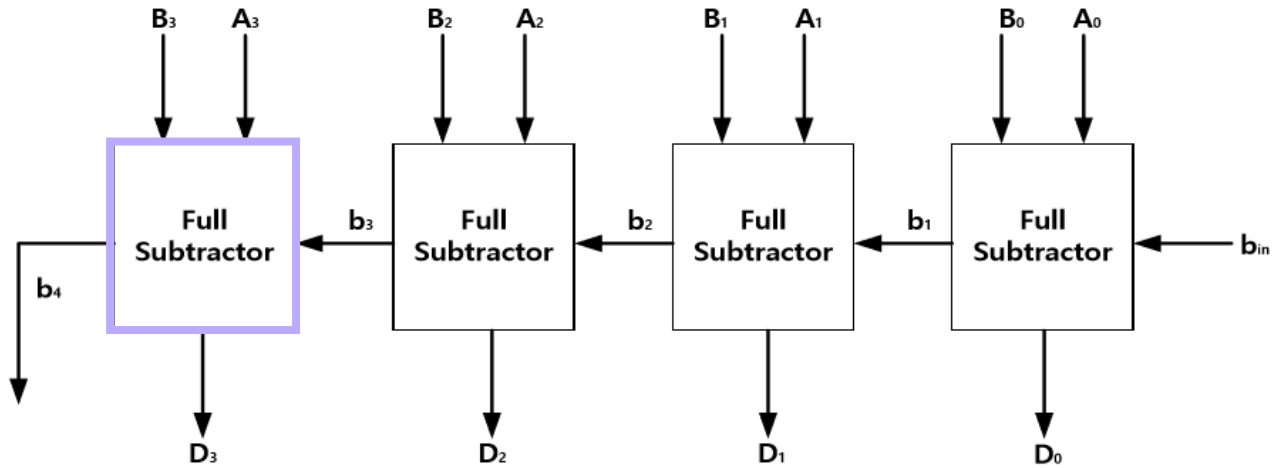


3) 세 번째 전감산기: A_2, B_2, 빌림 b_2을 입력으로 받고, 차이 D_2와 빌림 b_3를 생성한다.

이때 D_2는 차이 출력의 세 번째 비트를 나타내고, b_3는 다음 전감산기에 연결된다.

02 4-Bit Binary Parallel Subtractor

동작 원리



4) 네 번째 전감산기: A_3, B_3, 빌림 b_3을 입력으로 받고, 차이 D_3와 빌림 b_4를 생성한다.

이때 D_3는 차이 출력의 네 번째 비트를 나타내고, b_4는 최종 빌림 수가 된다.

02 4-Bit Binary Parallel Subtractor

Two's Complement

< 양수 (Positive) >

$+5_{10}$ = 000101

$+21_{10}$ = 010101

→ MSB에 0을 붙여 양수임을 표시
Unsigned integer와 동일한 표기

< 음수 (Negative) >

-5_{10} = 111011

-21_{10} = 101011

000101
→ 111010 (1의 보수)
→ 111011 (LSB에 +1)

∴ -5_{10} = 111011

→ MSB가 1이면 음수
양수 표현에서 1과 0을 서로 바꾸고 (1의 보수), 1을 더함

02 4-Bit Binary Parallel Subtractor

Two's Complement

< 덧셈 (Addition) >

(1) 5 + (-3) = 2

```
  0101 (+5)
+ 1101 (-3)
------------
1 0010 (+2)
```

(2) 5 + 2 = 7

```
  0101 (+5)
+ 0010 (+2)
------------
  0110 (+7)
```

→ 두 n-bit 입력 A, B가 있을 때
$A + B$를 n-bit Adder로 계산

< 뺄셈 (Subtraction) >

(1) 5 - 3 = 5 + (-3) = 2

```
  0101 (+5)
-  0011 (+3)
```

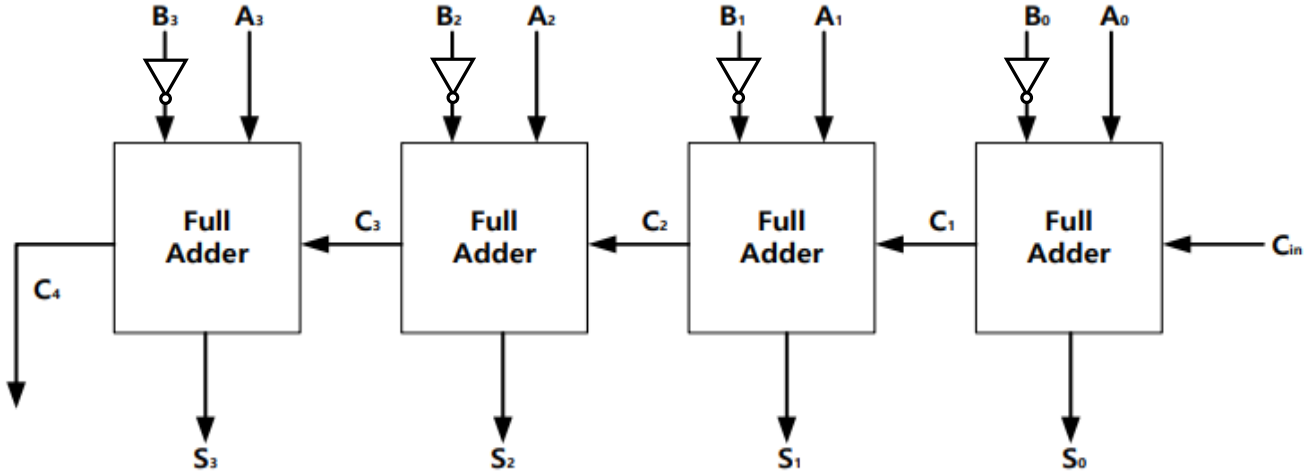
➡

```
  0101 (+5)
+ 1101 (-3)
-----------
1 0010 (+2)
```

→ 두 n-bit 입력 A, B가 있을 때
$A + (\overline{B} + 1)$를 n-bit Adder로 계산

02 4-Bit Binary Parallel Subtractor

병렬 가산기를 이용한 설계



- 4개의 전가산기를 연결해 4비트 이진 병렬 감산기를 설계
- 이 회로는 피감수에 감수의 2의 보수를 더하는 것이 기존 뺄셈 과정과 동일하다는 원리를 이용해 뺄셈 연산을 수행한다.
- NOT gate를 통해 B의 1의 보수를 얻어내고, 입력 캐리를 통해 1을 더해 B의 2의 보수를 구한다.

03 4-Bit Binary Parallel Adder/Subtractor

XOR 게이트, 2의 보수법을 활용한 4비트 이진 병렬 가감산기

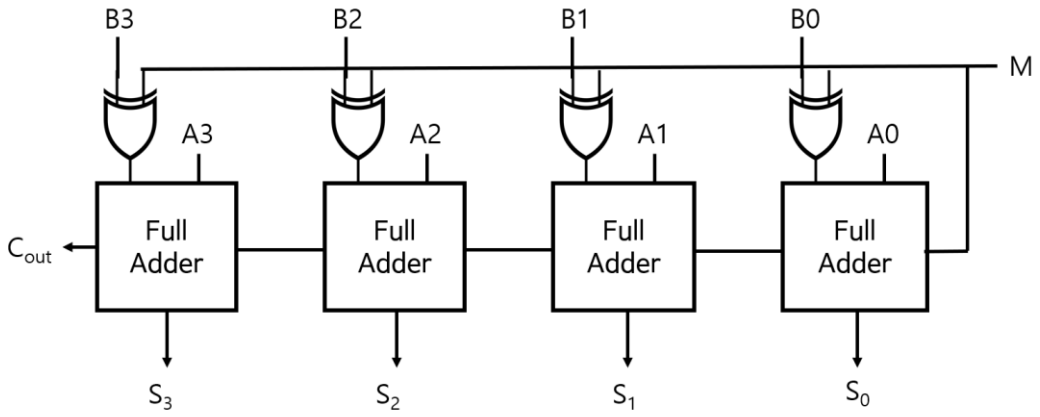
Parallel Adder/Subtractor (병렬 가감산기)

: 가산과 감산 동작을 전부 수행할 수 있는 회로

계산 결과

(합 또는 차) = $S_3S_2S_1S_0$

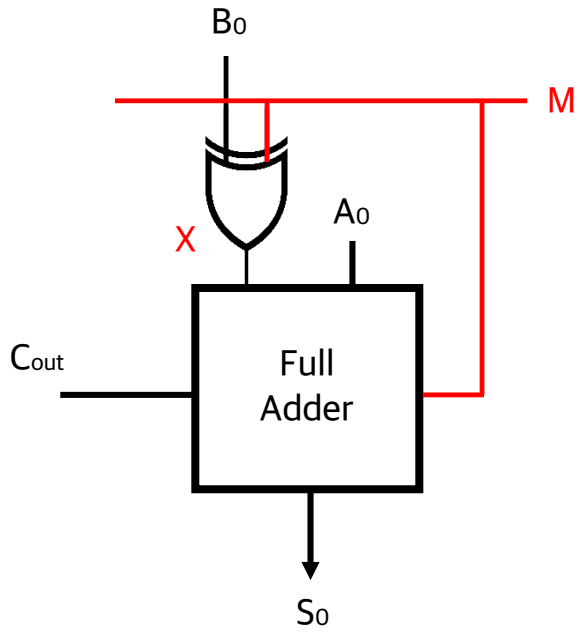
(캐리) = C_{out}



XOR 게이트를 이용해 이진수의 덧셈, 뺄셈을 한 디지털 회로에서 같이 처리할 수 있다.

03 4-Bit Binary Parallel Adder/Subtractor

동작 원리



Case 1) M = 0

$$X = M \oplus \mathrm{B}_0 = \mathrm{B}_0$$

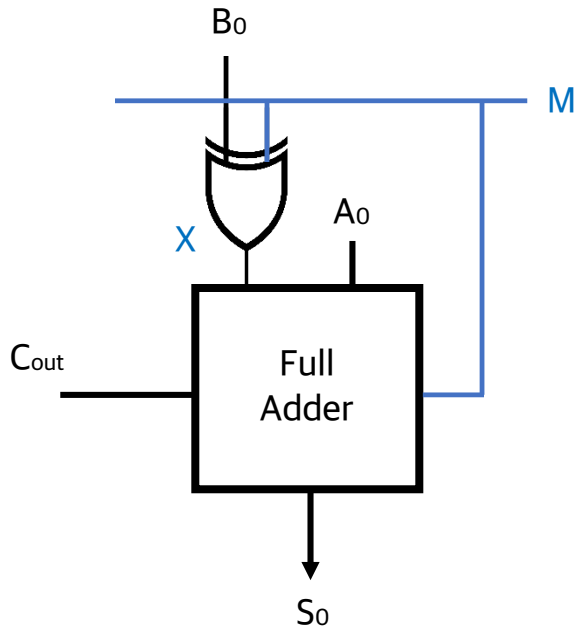
$$S_0 = A_0 \oplus X \oplus \mathrm{M} = \mathrm{A}_0 \oplus \mathrm{B}_0$$

$$C_{out} = M(A_0 \oplus X) + A_0X = A_0B_0$$

→ A_0과 B_0의 덧셈 연산

03 4-Bit Binary Parallel Adder/Subtractor

동작 원리



Case 1) M = 1

$$X = M \oplus \mathrm{B}_0 = \overline{B_0}$$

$$S_0 = A_0 \oplus X \oplus \mathrm{M} = \mathrm{A}_0 \oplus B_0$$

$$C_{out} = M(A_0 \oplus X) + A_0 X = A_0 \oplus \overline{B_0} + A_0\,\overline{B_0}$$

→ A_0과 B_0의 뺄셈 연산

03 4-Bit Binary Parallel Adder/Subtractor

장단점

< 장점 >

- serial adder/subtractor에 비해 비교적 빠른 연산 속도
- 덧셈에 필요한 시간은 비트의 수에 의존하지 않음
- 출력은 병렬 형식으로 나타나며, 모든 비트가 동시에 연산됨
- 비용이 더 저렴함

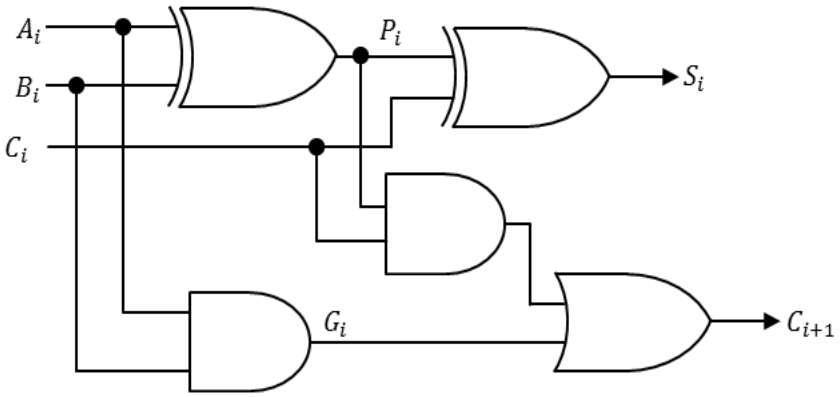
< 단점 >

- 각 가산기는 연쇄적으로 이어진 이전 가산기에서 생성되어야 하는 캐리를 기다려야 함
- 더해야 하는 비트 수가 증가함에 따라 전파 지연 증가

➡ **Look Ahead Carry(예견 올림수)를 활용하자 !**

04 4-bit Carry Look Ahead Adder

Look Ahead Carry(예견 올림수)란?

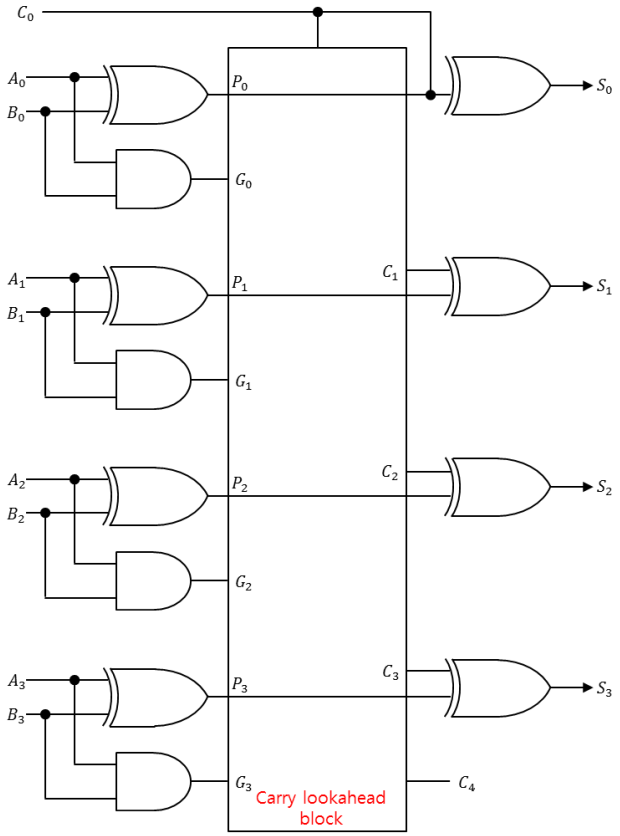


[그림] A_i와 B_i를 더하는 1 bit 논리 회로

- C: 캐리(carry), 받아올림
- S: 연산 결과
- G: carry generate, $G_i = A_i \cdot B_i$
- P: carry propagate, $P_i = A_i \oplus B_i$
- G_i = 1 ⟺ carry-out = 1 , P_i = 1 ⟺ carry-in = carry-out
- 출력 $S_i = P_i \oplus C_i$, 캐리 $C_{i+1} = G_i + P_i C_i$

04 4-bit Carry Look Ahead Adder

동작 원리



$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$$

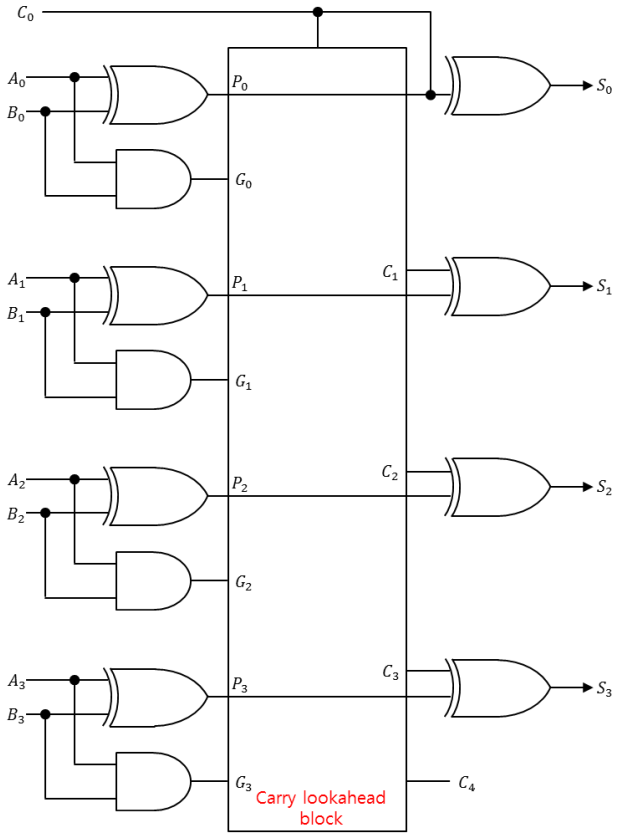
$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

$$C_4 = G_3 + P_3C_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

→ C_1~C_4가 모두 P_i, G_i, C_0로 구성되어 있으므로 한 번에 C_1~C_4까지 생성할 수 있다

04 4-bit Carry Look Ahead Adder

vs Ripple Carry Adder

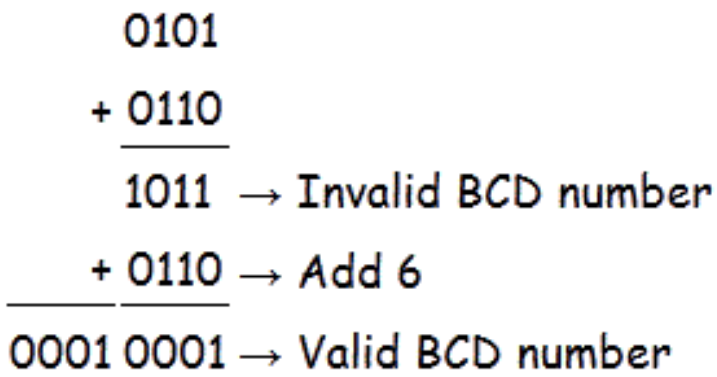


Ripple Carry Adder에 비해

- 회로가 훨씬 더 복잡함
- bit 수가 커질 수록 연산 속도가 훨씬 빨라짐

05 BCD 연산

개념



BCD 연산이란?

BCD 코드의 연산을 말한다.

- 각 자릿수에 대하여 이진수 연산 시행
- BCD 연산의 결과값 역시 BCD 코드여야 한다.
- 결과값이 BCD 코드에서 사용되지 않는 1010(=10_{10}) ~ 1111(=15_{10})에 해당하는 경우 0110(=6_{10})을 더해 값을 수정

05 BCD 연산

연산 과정

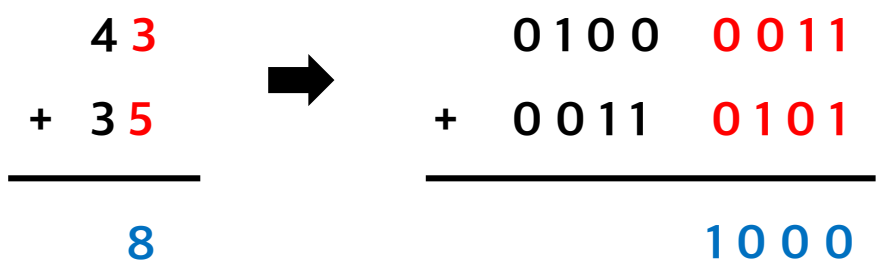
Case 1) 캐리가 발생하지 않는 경우

```
   4 3          0100  0011
 + 3 5        + 0011  0101
-------      ------------------
```

05 BCD 연산

연산 과정

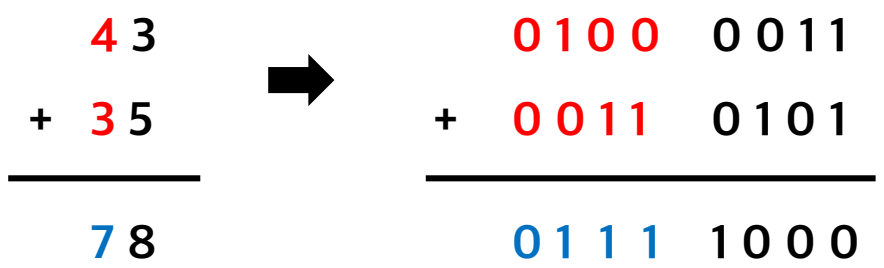
Case 1) 캐리가 발생하지 않는 경우



05 BCD 연산

연산 과정

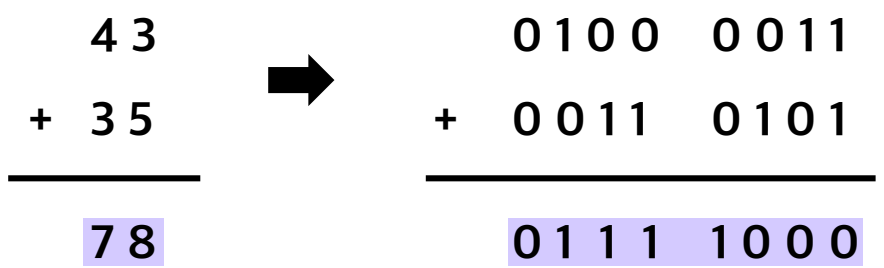
Case 1) 캐리가 발생하지 않는 경우



05 BCD 연산

연산 과정

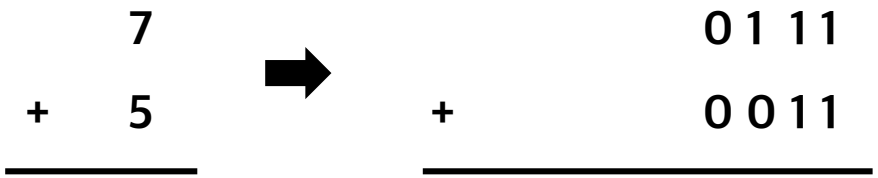
Case 1) 캐리가 발생하지 않는 경우



05 BCD 연산

연산 과정

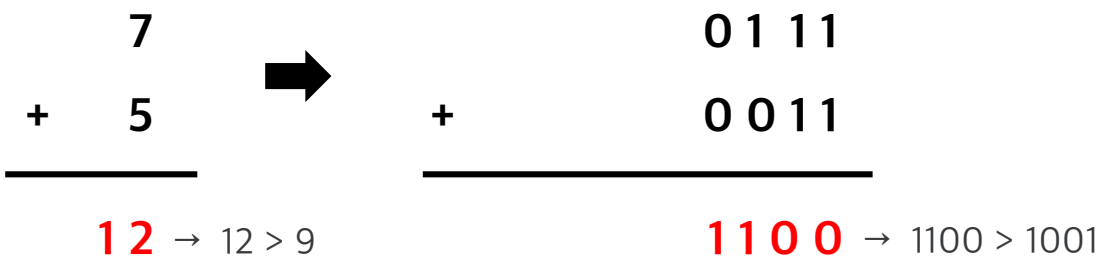
Case 2) 캐리가 발생하는 경우



05 BCD 연산

연산 과정

Case 2) 캐리가 발생하는 경우



05 BCD 연산

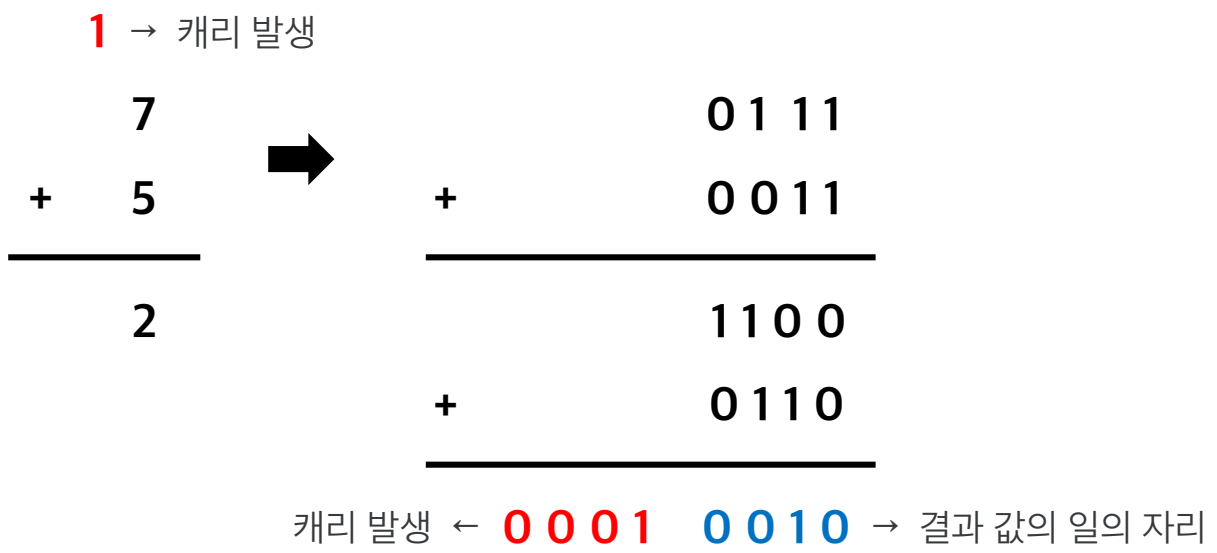
연산 과정

Case 2) 캐리가 발생하는 경우

05 BCD 연산

연산 과정

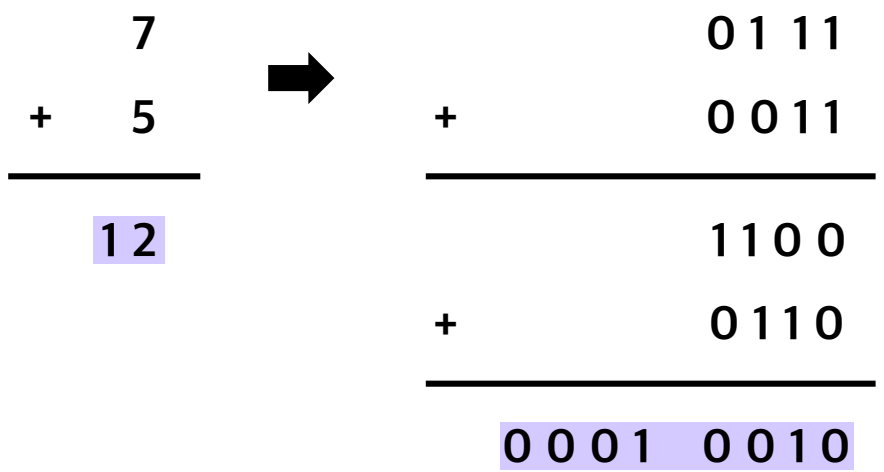
Case 2) 캐리가 발생하는 경우



05 BCD 연산

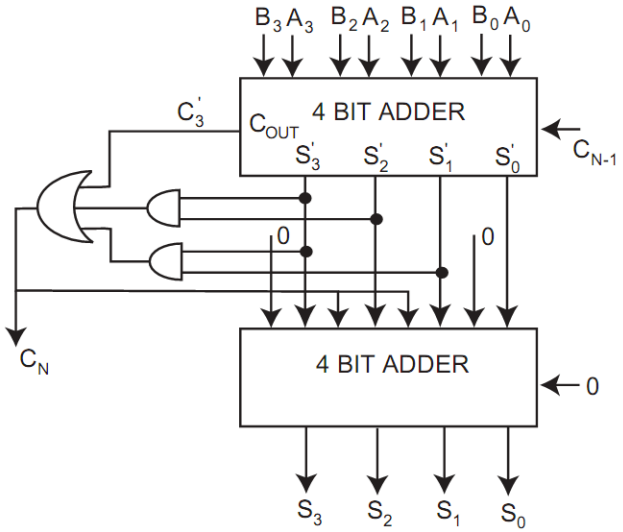
연산 과정

Case 2) 캐리가 발생하는 경우

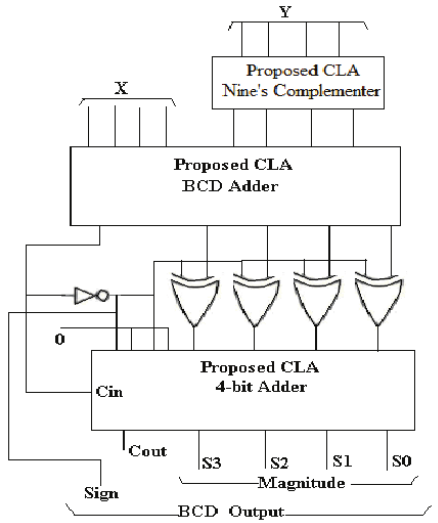


05 BCD 연산

BCD Adder와 Subtractor



[BCD Adder]

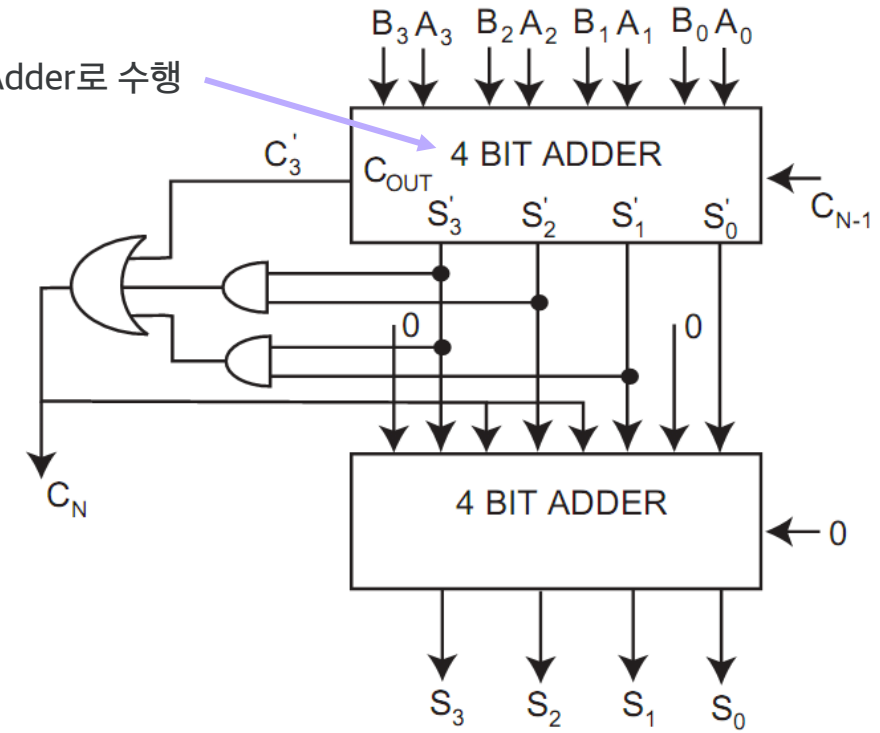


[BCD Subtractor]

05 BCD 연산

BCD Adder 동작 원리

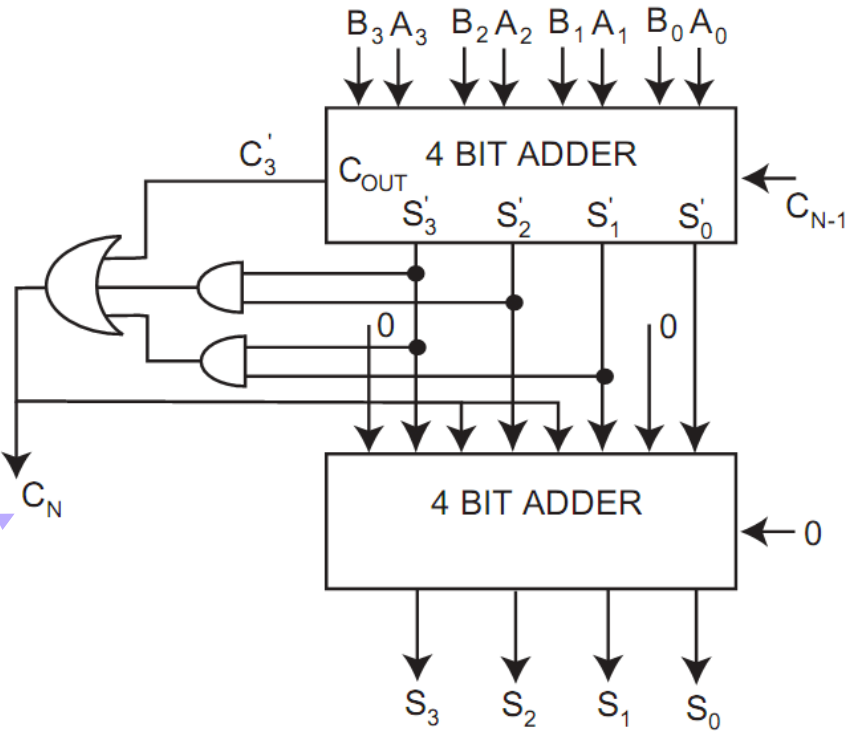
(1) A와 B의 덧셈 연산을 4-bit Adder로 수행



[BCD Adder]

05 BCD 연산

BCD Adder 동작 원리

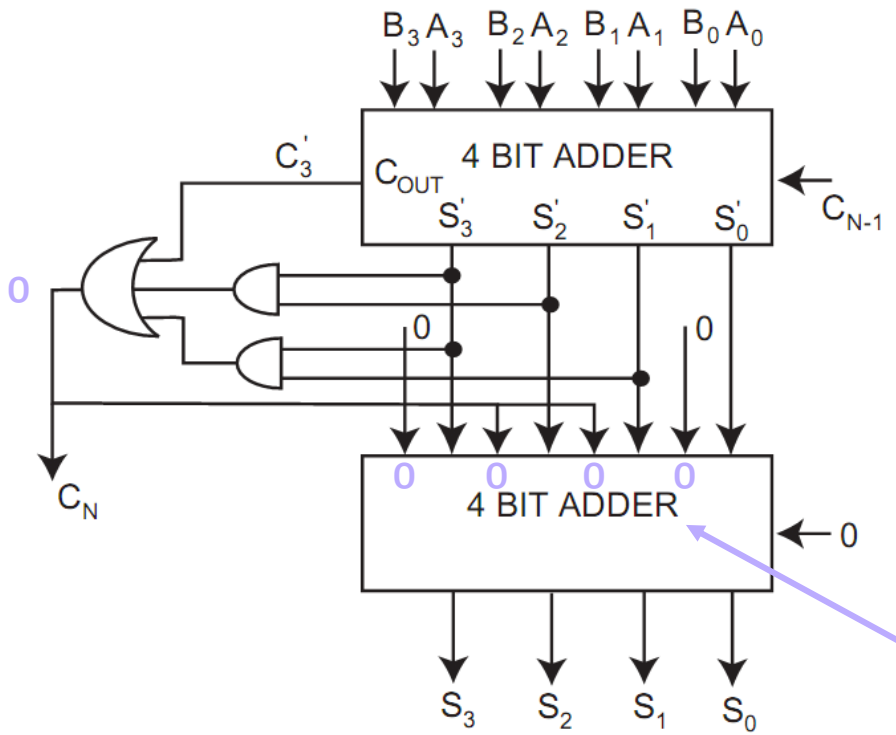


[BCD Adder]

(2) 합 S'3 S'2 S'1 S'0이 1001을 초과하는 경우 캐리 1 발생

05 BCD 연산

BCD Adder 동작 원리



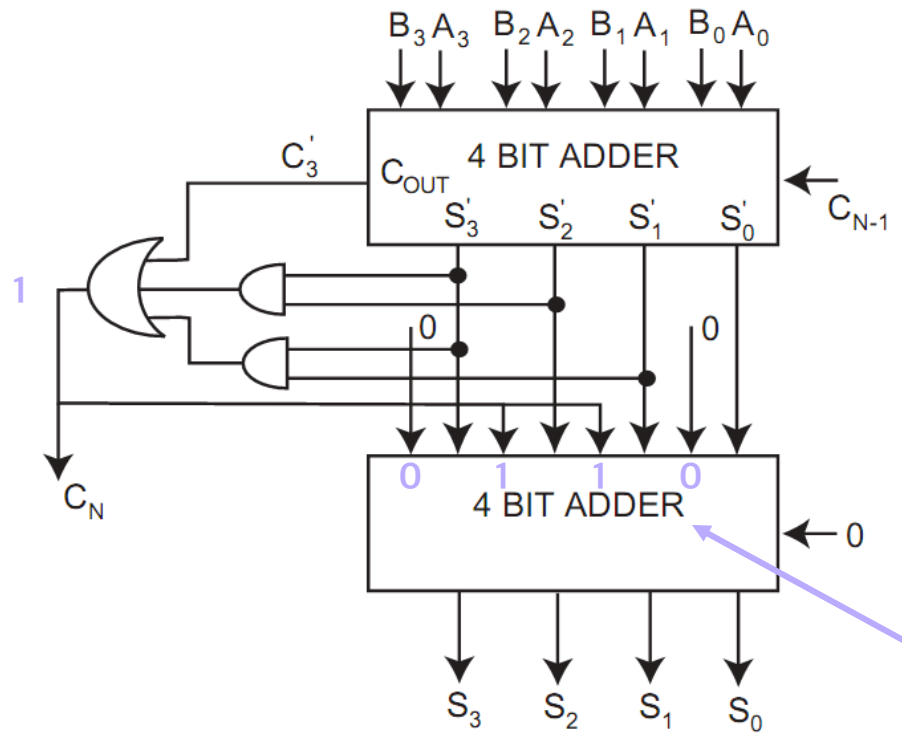
[BCD Adder]

(3) 합 $S'_3\ S'_2\ S'_1\ S'_0$이 1001 이하인 경우 0000을 덧셈함.

→ 이 합을 그대로 최종 합으로 출력

05 BCD 연산

BCD Adder 동작 원리

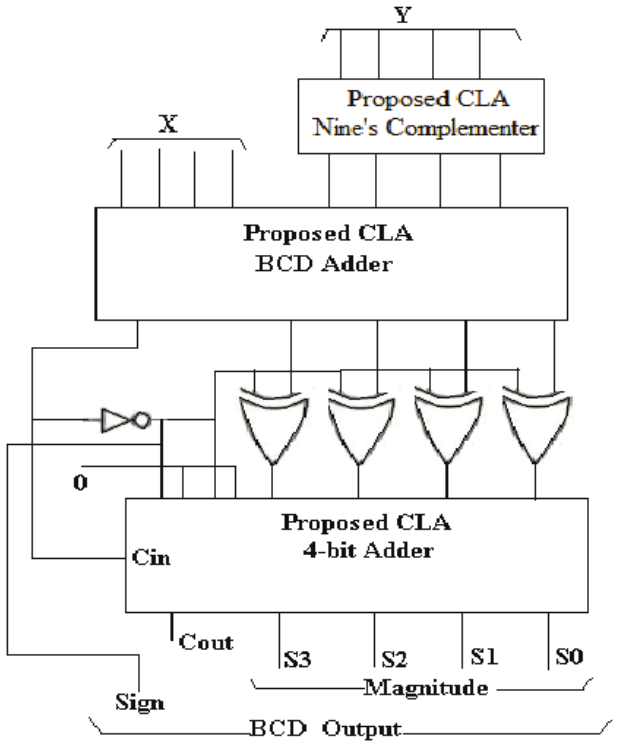


[BCD Adder]

(4) 합 $S'_3\ S'_2\ S'_1\ S'_0$이 1001 초과인 경우 0110과 덧셈한 뒤 캐리는 버리고 C_N과 합 $S_3\ S_2\ S_1\ S_0$을 출력

05 BCD 연산

BCD Subtractor 동작 원리



→ 9의 보수를 이용하여 뺄셈 연산

4자리 Y의 9의 보수 => 9999 - Y

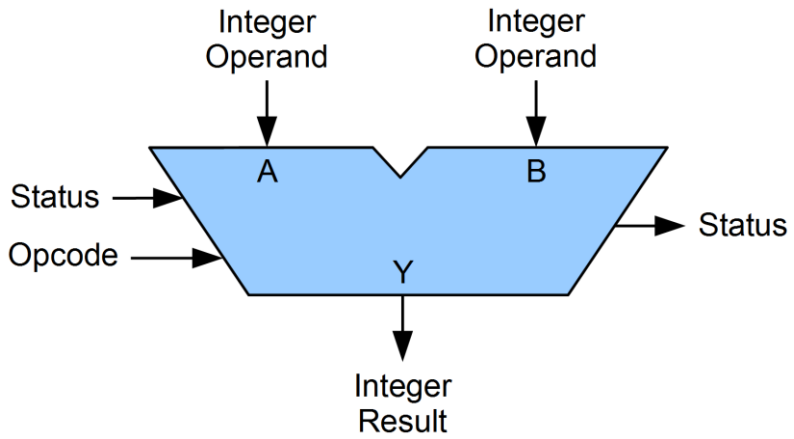
① Y의 9의 보수를 구함 … A

② X와 A의 BCD 덧셈 연산 수행 … B

③ X - Y = B + 캐리

06 [심화] ALU

개념



ALU란?

산술 논리 장치(Arithmetic Logic Unit)의 줄임말로, 덧셈, 뺄셈 같은 두 숫자의 논리 연산과 배타적 논리합, 논리곱, 논리합 등의 논리 연산을 계산하는 디지털 회로를 말한다. 컴퓨터 중앙처리장치(CPU) 의 기본 설계 블록으로 사용되며, 디지털 시계와 같은 작은 회로에도 작은 ALU가 포함된다.

06 [심화] ALU

주요 연산 기능

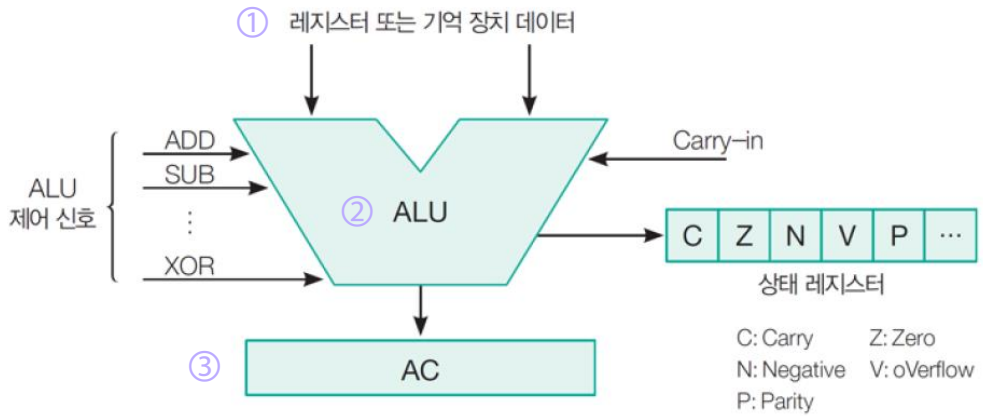
ALU의 주요 연산 기능

1) 정수형 산술 연산 (사칙연산 등)
2) 비트 논리 연산 (AND, OR, NOT, XOR)
3) 비트 시프트 연산

수의 제곱근과 같은 복잡한 연산은 여러 간단한 ALU의 조합으로 나누어 처리한다.

06 [심화] ALU

연산 과정



① 레지스터 또는 기억 장치에서 데이터를 가져온다.

② ALU에서 명령어에 따른 연산을 수행한다.

③ 연산의 결과값을 누적기(Accumulator)에 저장한다.

Presentation

출처 및 기여도

https://velog.io/@underlier12/%EC%BB%B4%ED%93%A8%ED%84%B0%EA%B5%AC%EC%A1%B0-11-%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C-%EB%AA%85%EB%A0%B9%EA%B3%BC-ALU

https://www.geeksforgeeks.org/parallel-adder-and-parallel-subtractor/

https://www.researchgate.net/figure/Block-Diagram-of-BCD-Adder_fig2_266486454

https://www.researchgate.net/figure/Proposed-CLA-BCD-Subtractor_fig6_220110221

Introduction to Logic and Computer Design. International edition, 2008 Alan B.Marcovitx McGraw-Hill

임석구, 홍성호,『디지털 논리회로』, 한빛아카데미(2016)

서강대학교 컴퓨터공학실험2 강의자료

20191048 김도솔 (34%) | 20211600 지소현 (33%) | 20221595 이선우 (33%)

END OF PRESENTATION

감사합니다