REPORT

인하대학교
INHA UNIVERSITY

과목명 | 논리회로

담당교수 | 최성용

학과 | 컴퓨터공학과

학년 | 2

학번 | 12191661

이름 | 윤혁

제출일 | 2020. 12. 07

6-3) *2의 보수 체계 이므로 맨 앞 비트는 부호 비트로 사용된다
만약 0이면 양수 이므로 순수한 2진수로 읽으면 되고
만약 1이면 음수 이므로 2의 보수 형태 이전을 찾아서 (-)부호를 붙여준다.
*2의 보수를 취하는 방법은 LSB부터 최초 1을 만날 때 까지는 그대로 써주고
그 이후부터는 뒤집는다

(a) 01101 → 양수이므로 $8+4+1 = 13_{10}$

(b) 11101 → 음수이므로 11101
00011 → 3) $= -3_{10}$

(c) 0111011 → 양수이므로 $64+32+16+8+2+1 = 123_{10}$

(d) 1001100 → 음수이므로 1001100
01100111 → $64+32+4+2+1=103$) $= -103_{10}$

(e) 0111111 → 양수이므로 $64+32+16+8+4+2+1 = 127_{10}$

(f) 1000000 → 음수이므로 10000000
10000000 → 128) $= -128_{10}$

(g) 111111 → 음수이므로 1111111
00000001 → 1) $= -1_{10}$

(h) 1000001 → 음수이므로 1000001
0111111 → $64+32+16+8+4+2+1 = 127$) $= -127_{10}$

(i) 0110011 → 양수이므로 $64+32+2+1 = 99_{10}$

(j) 1101100 → 음수이므로 1101100
00100111 → $32+4+2+1=39$) -39_{10}

6-6)

(a) $+73_{10}$ → 01001001_2 → 10110111_2 (-73_{10})

(b) -12_{10} → 00001100_2
11110100_2 → 00001100_2 $(+12_{10})$

(c) $+15_{10}$ → 00001111_2 → 11110001_2 (-15_{10})

(d) -1_{10} → 00000001_2
11111111_2 → 00000001_2 $(+1_{10})$

(e) -128_{10} → 10000000_2 → 010000000 $(+128_{10})$
10000000_2 (8bit는 -128~127 범위의 수를 표현 가능하다. +128은 9bit 필요)

(f) $+127_{10}$ → 01111111_2 → 10000001_2 (-127_{10})

6-9)

(a) +14 더하기 +9

00001110

+00001001

00010111 → 16+4+2+1 = 23 = 14+9

(b) +15 더하기 -19

00001111

+11101101

11111100

2의 보수 00000100 → -4 = 15+(-19)

(c) +21 더하기 -30

00010101

+11100010

11110111

2의 보수 00001001 → -9 = 21+(-30)

(d) -38 더하기 -75

11011010

+10110101

~~1~~10001111

2의 보수 01110001 → -113 = (-38)+(-75)

(e) +22 빼기 +19

= +22 더하기 -19

00010110

+11101101

~~1~~00000011 → 3 = (+22)-(+19).

(f) -21 빼기 +31

= -21 더하기 -31

11101011

+11100001

~~1~~11001100

2의 보수 00110100 → -52 = (-21)-(+31)

(g) +15 빼기 51

= 15 더하기 -51

00001111

+11001101

11011100

2의 보수 00100100 → -36 = (+15)-51

(h) -12 빼기 -12

= -12 더하기 12

11110100

+00001100

~~1~~00000000 → 0 = -12-(-12)

(i) +85 더하기 -85

01010101

+10101011

~~1~~00000000 → 0 = 85+(-85)

6-9)

(j) -26 빼기 -26 = -26 더하기 26

  11100110
+ 00011010
~~1~~00000000

→ 0 = -26 - (-26)

(k) 20 더하기 20

  00010100
+ 00010100
  00101000

→ 40 = 20 + 20

(l) 50 빼기 -25 = 50 더하기 25

  00110010
+ 00011001
  01001011

→ 75 = 50 - (-25)

6-10)

(a) 45 더하기 89

  00101101
+ 01011001
  10000110

→ 양수 + 양수 이므로 양수가 나와야 하는데 덧셈 결과값의 맨 앞자리가 '1' 이라는 것은 음수라는 것이므로 overflow 발생!

(c) -45 더하기 -89

문제 순서 b↔c

  11010011
+ 10100111
~~1~~01111010

→ 음수 + 음수 이므로 음수가 나와야 하는데 덧셈 결과값의 맨 앞자리가 '0' 이라는 것은 양수라는 것이므로 overflow 발생!

(b) -89 빼기 +45 = -89 더하기 -45

  10100111
+ 11010011
~~1~~01111010

→ (c)번과 같다

(d) 89 빼기 -45 = 89 더하기 45

  01011001
+ 00101101
  10000110

→ (a)번과 같다

6-13)

(a) 74+23

```
  0111  0100
+ 0010  0011
------------
  1001  0111 BCD = 97₁₀
```

(b) 58+37

```
  0101 1000
+ 0011 0111
-----------
  1000 1111
+      0110   (보정수 6을 더함)
-----------
  1001 0101 BCD = 95₁₀
```

(c) 147+380

```
  0001 0100 0111
+ 0011 1000 0000
----------------
  0100 1100 0111
+      0110
----------------
  0101 0010 0111 BCD = 527₁₀
```

(d) 385+118

```
  0011 1000 0101
+ 0001 0001 1000
----------------
  0100 1001 1101
+           0110
----------------
  0100 1010 0011
+      0110
----------------
  0101 0000 0011 BCD = 503₁₀
```

(e) 998+003

```
  1001 1001 1000
+ 0000 0000 0011
----------------
  1001 1001 1011
+           0110
----------------
  1001 1010 0001
+      0110
----------------
  1010 0000 0001
+ 0110
----------------
0001 0000 0000 0001 BCD = 1001₁₀
```

(f) 623+599

```
  0110 0010 0011
+ 0101 1001 1001
----------------
  1011 1011 1100
+ 0110 0110 0110
----------------
0001 0010 0010 0010 BCD = 1222₁₀
```

(g) 555+274

```
  0101 0101 0101
+ 0010 0111 0100
----------------
  0111 1100 1001
+      0110
----------------
  1000 0010 1001 BCD = 829₁₀
```

(h) 487+116

```
  0100 1000 0111
+ 0001 0001 0110
----------------
  0101 1001 1101
+           0110
----------------
  0101 1010 0011
+      0110
----------------
  0110 0000 0011 BCD = 603₁₀
```

6-15)

(a) 3E91−2F93 = 3E91 + $\underline{(-2F43)}$ → 3E91 + D06D

$$\begin{array}{r} FFFF \\ -\ 2F93 \\ \hline D06C \\ +\quad\ \ 1 \\ \hline D06D \end{array} \qquad = D06D \qquad \begin{array}{r} 3E91 \\ +\ D06D \\ \hline \cancel{1}\ 0EFE \end{array}$$

∴ 0EFE

(b) 91B − 6F2 = 91B + $\underline{(-6F2)}$

$$\begin{array}{r} FFF \\ -\ 6F2 \\ \hline 90D \\ +\quad\ 1 \\ \hline 90E \end{array} \qquad = 90E \qquad \begin{array}{r} 91B \\ +\ 90E \\ \hline \cancel{1}\,229 \end{array}$$

∴ 229

(c) 0300 − 005A = 0300 + $\underline{(-005A)}$

$$\begin{array}{r} FFFF \\ -\ 005A \\ \hline FFA5 \\ +\quad\ \ 1 \\ \hline FFA6 \end{array} \qquad = FFA6 \qquad \begin{array}{r} 0300 \\ +\ FFA6 \\ \hline \cancel{1}\ 02A6 \end{array}$$

∴ 02A6

(d) 0200 − 0003 = 0200 + $\underline{(-0003)}$

$$\begin{array}{r} FFFF \\ -\ 0003 \\ \hline FFFC \\ +\quad\ \ 1 \\ \hline FFFD \end{array} \qquad = FFFD \qquad \begin{array}{r} 0200 \\ +\ FFFD \\ \hline \cancel{1}\ 01FD \end{array}$$

∴ 01FD

(e) F000 − EFFF = F000 + $\underline{(-EFFF)}$

$$\begin{array}{r} FFFF \\ -\ EFFF \\ \hline 1000 \\ +\quad\ \ 1 \\ \hline 1001 \end{array} \qquad = 1001 \qquad \begin{array}{r} F000 \\ +\ 1001 \\ \hline \cancel{1}\ 0001 \end{array}$$

∴ 0001

6-15)

(f) 2F00 − 4000 = 2F00 + (−4000) → 2F00 + C000

```
  FFFF
- 4000
------
  BFFF
+    1
------
  C000
```

(−4000) = C000

```
  2F00
+ C000
------
  EF00
```

∴ EF00

(g) 9AE5 − C01D = 9AE5 + (−C01D) → 9AE5 + 3FE3

```
  FFFF
- C01D
------
  3FE2
+    1
------
  3FE3
```

(−C01D) = 3FE3

```
   1 1
  9AE5
+ 3FE3
------
  DAC8
```

∴ DAC8

(h) 4321 − F165 = 4321 + (−F165) → 4321 + 0E9B

```
  FFFF
- F165
------
  0E9A
+    1
------
  0E9B
```

(−F165) = 0E9B

```
  1
  4321
+ 0E9B
------
  51BC
```

∴ 51BC

6-20)

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$SUM = A \oplus B$
$CARRY = AB$
] 첫 번째 반가산기 표현식

이를 새로운 Input C_{IN} 과 한번더 반복하고 OR게이트로 더하면

$SUM = [A \oplus B] \oplus C_{IN}$, $CARRY = [A \oplus B] \cdot C_{IN}$

$CARRY_{out} = AB + [A \oplus B] \cdot C_{IN}$

$= AB + AC_{IN} + BC_{IN}$ 로 표현된다. 이는

전가산기 에서의 S, C_{out} 과 표현식이 같으므로

∴ 그림 6-27은 전가산기로 동작한다

6-22) * 각 FF : $t_{PLH} = t_{PHL} = 50ns$, $T_s = 15ns$, FA (전가산기) propagation delay = 65ns.

1. B레지스터에서 LOAD 펄스의 PGT 발생 후 50ns 후에 B값이 바뀌고 동시에 FA로 들어감

2. 각 C(캐리)는 FA에서 65ns 후에 출력된다. 전가산기가 총 4개이므로 $(65 \times 4)ns = 260ns$

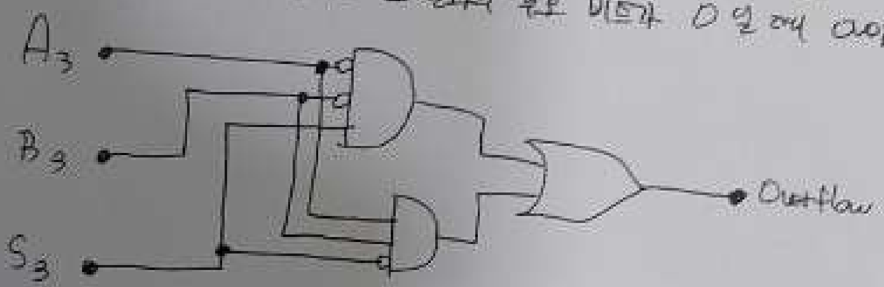
3. SUM 값을 A레지스터에 넣는데 FF의 setup time 이 15ns 이므로 최소 15ns 후에 PGT를 발생시켜야 제대로 동작한다.

→ $(50 + 65 \times 4 + 15)ns = 325ns$

6-23) * A_3 : A의 부호 비트, B_3 : B의 부호 비트, S_3 : A+B의 부호 비트

1. $\overline{A_3}\,\overline{B_3}\,S_3 + A_3 B_3 \overline{S_3}$ = overflow

(더해지는 수가 모두 양수이고 그 결과의 부호 비트가 1이거나, 더해지는 수가 모두 음수이고 그 결과의 부호 비트가 0일 때 overflow 발생)



6-25)

$C_1 = C_0 \cdot (A_0 \oplus B_0) + A_0 B_0$

$C_2 = C_1 (A_1 \oplus B_1) + A_1 B_1$

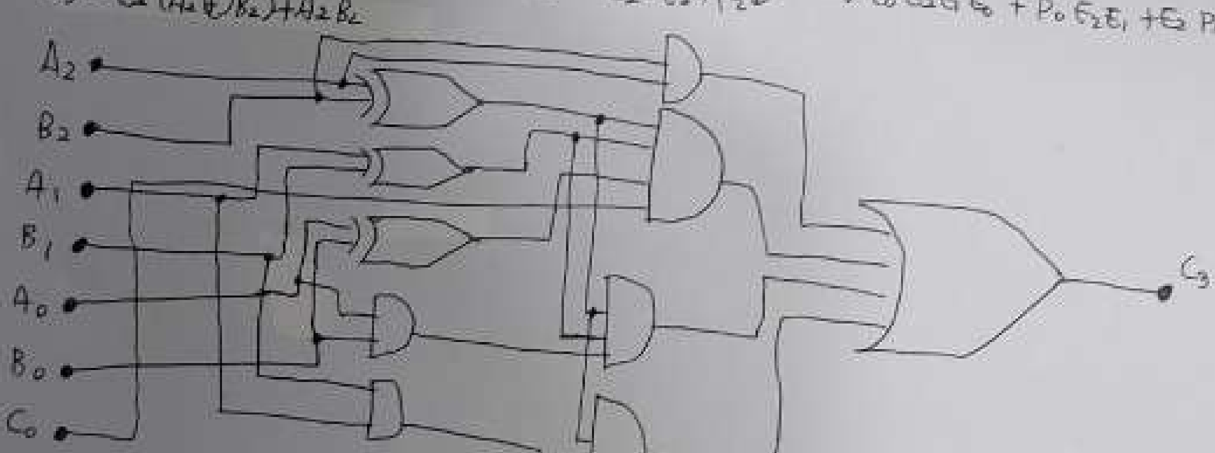
$C_3 = C_2 (A_2 \oplus B_2) + A_2 B_2$

Let $E_i = A_i \oplus B_i$, $P_i = A_i \cdot B_i$

$C_1 = C_0 \cdot E_0 + P_0$

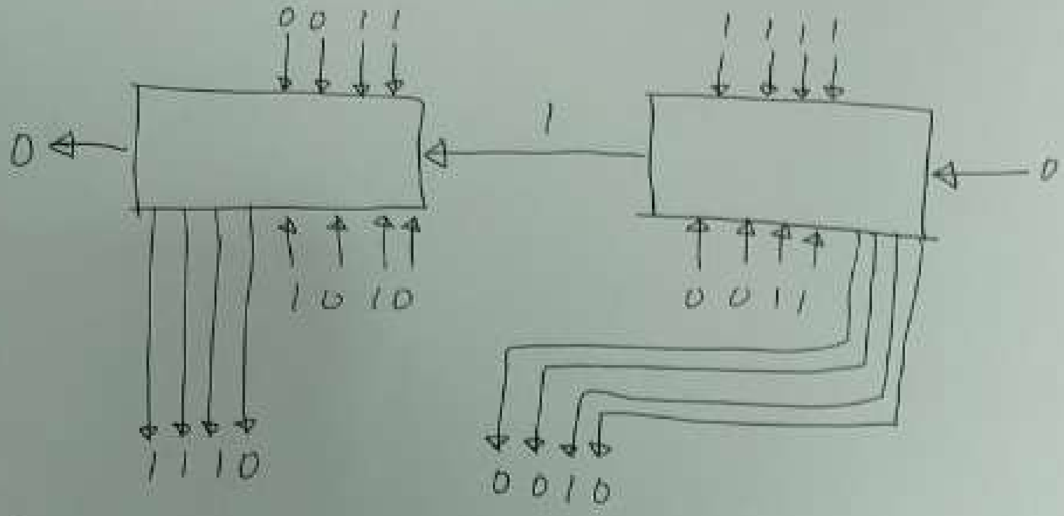
$C_2 = C_1 \cdot E_1 + P_1$ → $C_0 E_1 E_0 + P_0 E_1 + P_1$

$C_3 = C_2 \cdot E_2 + P_2$ → $C_0 E_2 E_1 E_0 + P_0 E_2 E_1 + E_2 P_1 + P_2$



6-26)

$3F_{16} = 0011\ 1111$, $A3_{16} = 1010\ 0011$



6-28)

*ADD가 1이고 SUB이 0이면 B를 그대로 넣고 $C_0=0$ 상태로 연산을 하고
ADD가 0이고 SUB이 1이면 $\bar{B}$를 넣고 $C_0=1$ 상태(2의 보수 of B)로 연산한다.

(a) A: 1010 (-6) , B: 0001 (+1) ; SUB=1, ADD=0

```
   1010
 + 1110
 ------
  11001
2의보수 0111
```

→ -7 = A-B
= -6+(-1)

(b) A: 1001 (-7), B: 0110 (+6) ; SUB=1, ADD=0

```
   1001
 + 1001
 ------
  10011
```

→ 음수+음수를 했는데 부호 비트가 0이므로 양수이다.
overflow 발생!

(c) A: 1110 (-2) , B: 0101 (+5) ; SUB=0, ADD=1

```
   1110
 + 0101
 ------
  10011
```

→ 3= A+B
= (-2)+5

6-30) (a) 음수 + 음수 = 음수 . overflow 발생하지 않았다.

(b) 음수 + 음수 = 양수 . 음수가 나와야 하지만 양수 이므로 overflow 발생.

(c) 음수 + 양수 이므로 overflow 발생하지 않는다.

6-33)

(a) [S]=011 (덧셈), [A]=0110, [B]=0011, $C_N=0$

$$\begin{array}{ccccc} C_{N+4} & & & & C_N \\ 0 & & 1 & 1 & 0 \\ & 0 & 1 & 1 & 0 \\ + & 0 & 0 & 1 & 1 \\ \hline & 1 & 0 & 0 & 1 \end{array}$$

$\rightarrow$ F=1001

$C_{N+4}=0$

OVR=1 (양수+양수≠음수)

(b) [S]=001 (B−A), [A]=0110, [B]=0011, $C_N=1$

$$\begin{array}{ccccc} C_{N+4} & & & & C_N \\ 0 & & 1 & 1 & 1 \\ & 0 & 0 & 1 & 1 \\ + & 1 & 0 & 0 & 1 \\ \hline & 1 & 1 & 0 & 1 \end{array}$$

$\rightarrow$ F=1101

$C_{N+4}=0$

OVR=0

(c) [S]=010 (A−B), [A]=0110, [B]=0011, $C_N=1$

$$\begin{array}{ccccc} C_{N+4} & & & & C_N \\ 1 & 1 & & & 1 \\ & 0 & 1 & 1 & 0 \\ + & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 1 \end{array}$$

$\rightarrow$ F=0011

$C_{N+4}=1$

OVR=0

6-34)

(a) Increment A

[S]=011 (ADD), [B]=0001, $C_N=0$

$\therefore A=A+1$

(b) decrement A

[S]=010 (A minus B), [B]=0001, $C_N=1$

$\therefore A=A-1$

6-35)

(a) [S]=011 (A+B), [A]=11010010, [B]=00110011

$$
\begin{array}{r}
11010010 \\
+\ 00110011 \\
\hline
\cancel{1}00001101
\end{array}
\quad \rightarrow \Sigma = 00001101
$$

(b) [S]=101 (OR), [A]=11001110, [B]=10101010

$$
\begin{array}{lr}
A+B & 11001110 \\
\text{(OR)} & 10101010 \\
\hline
 & 11101110
\end{array}
\quad \rightarrow \Sigma = 11101110
$$

6-38)

B register 가 XOR gate 에 의해 병렬 가산기에 입력되어도 문제가 생기지 않는다.

case1. ADD=0, SUB=0

ADD, SUB 모두 0 이면 B register 에서 입력하는 B 또는 $\bar{B}$ 값은 없다 ($0 \oplus 0 = 0$) 결과 값은 A값 그대로이다. 이는 기존 OR gate (0+0=0)과 마찬가지이다.

case2. ADD=1, SUB=0

ADD=1, SUB=0 이면 B값이 내려오고 SUB=0 이기 때문에 2,4,6,8 gate 들은 항상 0 이다. $X \oplus 0 = X$ 이므로 B값을 그대로 통과 시킨다. 이는 기존 ORgate (0+B=B)와 마찬가지이다.

case3. ADD=0, SUB=0

ADD=0, SUB=1 이면 $\bar{B}$값이 내려오고 ADD=0 이기 때문에 1,3,5,7 gate들은 항상 0이다. $X \oplus 0 = X$ 이므로 B값을 그대로 통과 시킨다 이는 기존 OR gate ($0+\bar{B}=\bar{B}$)와 마찬가지이다.