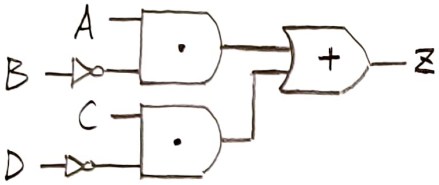
Ch.4 : Applications of Boolean Algebra / Minterm and Maxterm Expansions.

4.1 Conversion of English Sentences to Boolean Equations.

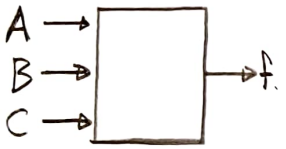
- Steps in designing a single-output combinational switching circuit.
  1. Find switching function which specifies the desired behavior of the circuit.
  2. Find a simplified algebraic expression for the function.
  3. Realize the simplified function using available logic elements.

- Example.
  1. The alarm will ring (Z) iff the alarm switch is turned on (A) "and" the door is not closed (B), "or" it is after 6PM (C) "and" window is not closed (D')
  2. Boolean Equation → $Z = AB' + CD'$
  3. Circuit realization



4.2 Combinational Logic Design Using a Truth Table.

- Combinational Circuit with Truth Table : 예를 들어 f를 구해보자.

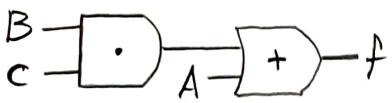


A	B	C	f	f'	
0	0	0	0	1	··· ①
0	0	1	0	1	··· ②
0	1	0	0	1	··· ③
0	1	1	1	0	··· ④
1	0	0	1	0	··· ⑤
1	0	1	1	0	··· ⑥
1	1	0	1	0	··· ⑦
1	1	1	1	0	··· ⑧

- 1이 되는 조건을 골라 f 식을 구하는 방법 (minterm)

$f = A'BC + ABC' + A'B'C + ABC' + ABC$ (SOP 형태)
④ ⑤ ⑥ ⑦ ⑧ 중 하나만 1을 만족해도 성립.

→ Simplified equation : $f = A + BC$
Circuit realization :



- 0이 되는 조건을 골라 f 식을 구하는 방법 (Max-term)

$f = (A + B + C)(A + B + C')(A + B' + C)$ (POS 형태)
① ② ③ 중 하나만 0을 만족해도 성립.

→ Simplified equation : $f = A + BC$

• f의 maxterm을 얻는 다른 방법 - f'을 이용.

f'의 minterm = $A'B'C' + A'B'C + A'BC'$

→ f'의 Complement를 취함.

$(f')' = f = (A+B+C)(A+B+C')(A+B'+C)$ by 드모르간 법칙

4.3 Minterm and Maxterm Expansions.

• Minterm, Maxterm for three variables.

Row No.	A	B	C	Minterms	Maxterms
0	0	0	0	$A'B'C' = m_0$	$A+B+C = M_0$
1	0	0	1	$A'B'C = m_1$	$A+B+C' = M_1$
2	0	1	0	$A'BC' = m_2$	$A+B'+C = M_2$
3	0	1	1	$A'BC = m_3$	$A+B'+C' = M_3$
4	1	0	0	$AB'C' = m_4$	$A'+B+C = M_4$
5	1	0	1	$AB'C = m_5$	$A'+B+C' = M_5$
6	1	1	0	$ABC' = m_6$	$A'+B'+C = M_6$
7	1	1	1	$ABC = m_7$	$A'+B'+C' = M_7$

• Minterm

Minterm of n variables is a product of n literals in which each variable appears exactly once in either true (A) or Complemented form (A'), but not both.

→ 즉 A와 A'중에 하나만 존재하며, 곱의 형태라는 소리.

• Minterm expansion, Standard Sum of Product

$f = A'BC + AB'C' + ABC + ABC' + ABC$

$f(A,B,C) = m_3 + m_4 + m_5 + m_6 + m_7$

$= \sum m(3, 4, 5, 6, 7)$

• Maxterm

Maxterm of n variables is a sum of n literals in which each variable appears exactly once in either true (A) or complemented form (A'), but not both.

• Maxterm, Standard Product of Sum

$f = (A+B+C)(A+B'+C)(A+B'+C)$

$f(A,B,C) = M_0 M_1 M_2$

$= \prod M(0, 1, 2)$

• Minterm and Maxterm expansions are complement each other

$f' = (m_3 + m_4 + m_5 + m_6 + m_7)' = m_3' m_4' m_5' m_6' m_7' = M_3 M_4 M_5 M_6 M_7$

$f' = (M_0 M_1 M_2)' = M_0' + M_1' + M_2' = m_0 + m_1 + m_2$

4.4 General Minterm and Maxterm Expansions

- General truth table for 3 variables

A	B	C	F
0	0	0	a_0
0	0	1	a_1
0	1	0	a_2
0	1	1	a_3
1	0	0	a_4
1	0	1	a_5
1	1	0	a_6
1	1	1	a_7

→ a_i is either 0 or 1

- Minterm expansion for general function

$$F = a_0m_0 + a_1m_1 + \cdots + a_7m_7 = \sum_{i=0}^{7} a_i m_i$$

$\begin{cases} a_i = 1 \rightarrow m_i \text{ present} \\ a_i = 0 \rightarrow m_i \text{ not present} \end{cases}$

- Maxterm expansion for general function

$$F = (a_0 + M_0)(a_1 + M_1)\cdots(a_7 + M_7) = \prod_{i=0}^{7} (a_i + M_i)$$

$\begin{cases} a_i = 1 \rightarrow a_i + M_i = 1 \quad \therefore M_i \text{ not present} \\ a_i = 0 \rightarrow M_i \text{ present} \end{cases}$

- Complement of F

$$F' = \left[\prod_{i=0}^{7} (a_i + M_i)\right]' = \sum_{i=0}^{7} a_i' M_i' = \sum_{i=0}^{7} a_i' m_i \rightarrow F\text{에서 포함 안된 minterm 모두 포함됨}$$

$$= \left[\sum_{i=0}^{7} a_i m_i\right]' = \prod_{i=0}^{7} (a_i' + m_i') = \prod_{i=0}^{7} (a_i + M_i) \rightarrow F\text{에서 포함 안된 maxterm 모두 포함됨}$$

- Minterm expansion의 특성

If i and j are different, $m_i m_j = 0$ → 서로 다른 minterm 사이의 곱은 0이다

ABC 와 같이 리터럴의 모든 종류가 포함되는 동시에, 그 자신 혹은 Complement로 이뤄져야 하므로.
13 페이지를 보면 어느 하나는 적어도 (자기 자신) × (Complement) 로 나타날 수밖에 없음. → 무조건 0.

- Maxterm expansion의 특성

If i and j are different, $M_i + M_j = 0$ → 서로 다른 maxterm 사이의 합은 0이다.
위와 마찬가지. 어느 하나는 적어도 (자기 자신) + (Complement) 로 나타날 수밖에 없음.

- Minterm expansion 특성 이용.

n개의 변수 존재 시.

$$f_1 = \sum_{i=0}^{2^n-1} a_i m_i, \quad f_2 = \sum_{j=0}^{2^n-1} b_j m_j$$

$$\rightarrow f_1 f_2 = \left(\sum_{i=0}^{2^n-1} a_i m_i\right)\left(\sum_{j=0}^{2^n-1} b_j m_j\right) = \sum_{i=0}^{2^n-1}\sum_{j=0}^{2^n-1} a_i b_j m_i m_j = \sum_{i=0}^{2^n-1} (a_i b_i) m_i$$

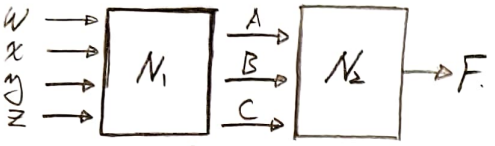
특성 덕에 double Sumol Single Sum으로. → 둘 다 1이어야 minterm에 포함 가능함 (공통 minterm만 남)

• Converse between minterm and maxterm expansions of F and F′ (정리 표)

	Minterm Expansion of F	Maxterm Expansion of F	Minterm Expansion of F′	Maxterm Expansion of F′
Minterm Expansion of F	—	F의 Maxterm numbers는 F의 minterm numbers 리스트에 있지 않은 수이다.	F에 포함되지 않은 minterm 리스트	F에 나타난 maxterm numbers는 F′에 나타난 minterm numbers와 동일함.
Maxterm Expansion of F	F의 minterm numbers는 F의 maxterm numbers 리스트에 있지 않은 수이다.	—	F에 나타난 minterm numbers는 F′에 나타난 maxterm numbers와 동일함.	F에 포함되지 않은 maxterm 리스트.

4.5 Incompletely Specified Functions.

• Don't care Conditions.



→ N_1 구조에 따라 ABC의 모든 조합이 나오지 않는 경우가 발생할.
N_2의 입장에선 일어나지 않는 케이스가 존재함
→ 신경 쓰지 않아도 되는 것.

A	B	C	F
0	0	0	1
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1

→ X 값이 Don't Care values.

• Finding function.

Case 1 : X를 0으로 보고 처리함.
Case 2 : X의 일부는 1로, 나머지는 0으로 보고 처리함.
Case 3 : X를 1로 보고 처리함.

• Don't Care가 있을 때의 함수 표현.

$F = \Sigma m(0,3,7) + \Sigma d(1,6)$
$F = \Pi M(2,4,5)\,\Pi D(1,6)$
→ Don't Cares

4.6 Examples of Truth Table Construction

• Binary Adder.

a	b	Sum
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

→

A	B	X	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$X = AB$

$Y = A'B + AB' = A \oplus B$

4.7 Design of Binary Adders and Subtractors

- Half Adder : bit 단위에서의 합과 Carry만을 고려함.

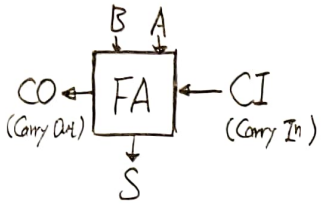
X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = X \oplus Y$ (Sum)

$C = XY$ (Carry)

회로도 표현하는 방식은? → 직접 그려보고 강의 PDF 확인

- Full Adder : 아랫단에서 올라오는 Carry까지 고려함.



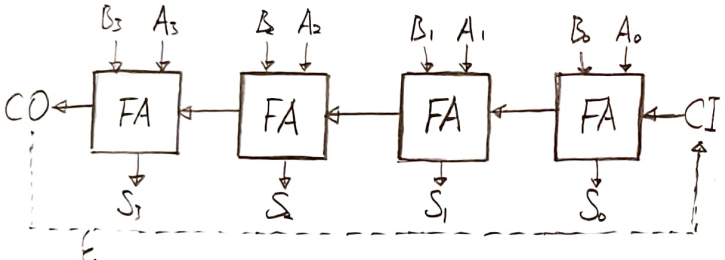
CI	B	A	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = A'B'CI + AB'CI' + ABCI + A'BCI'$

$= A \oplus B \oplus CI$

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

회로도 표현하는 방식? → 직접 그려보고 강의 PDF 확인.

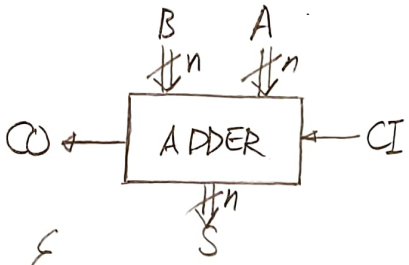


맨 처음 CI는 0으로 할당하고, LSB부터 시작해서 직렬 연결.

Carry가 물결처럼 넘어가는 것 같다 하여 Carry ripple adder라 함.

집선은 1의 보수에서 나오는 end-around carry의 기술적 표현. ⇒ 음수 덧셈 또한 처리 가능.

간략 ⟹



n: 비트 수, 간략하게 표현

- Sum & Carry. in FA.

$Sum = X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in}$

$= X \oplus Y \oplus C_{in}$

$C_{out} = X'YC_{in} + XY'C_{in} + XYC'_{in} + XYC_{in}$

$= YC_{in} + XC_{in} + XY$

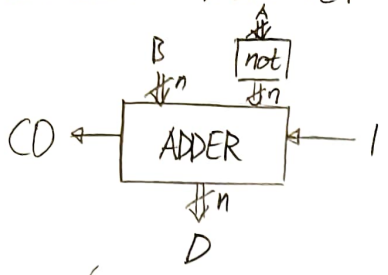
- Overflow (V) when adding two signed binary number.

충분한 자릿수를 사용하지 않을 때 발생하는 현상. → 연산이 정확히 이루어졌는지 확인하기 위해 체크 함수 필요.

$V = A_3'B_3'S + A_3B_3S'$ (4비트 기준)

⇒ 음수 & 음수 들어왔는데 양수 결과 OR 양수 & 양수 들어왔는데 음수 결과

• Subtracters : 덧셈으로 충분히 구현 가능하다만 굳이 하고싶다면.

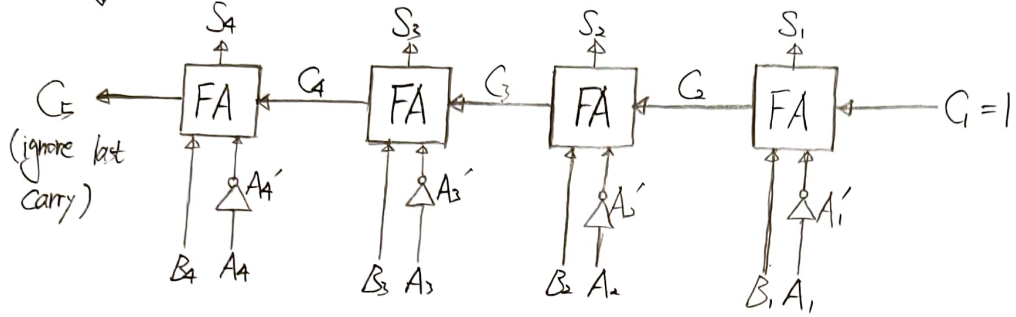


→ 2's Complemented number

: 양수의 1은 0으로, 0은 1로 바꾼 후 1을 더함.

기존의 덧셈기로도 뺄셈 작동이 수행 가능. (빼는 수를 inverting 해서)

↓ 확대



⇒ B-A ; 전부 Complement 취한 후 1 더함.

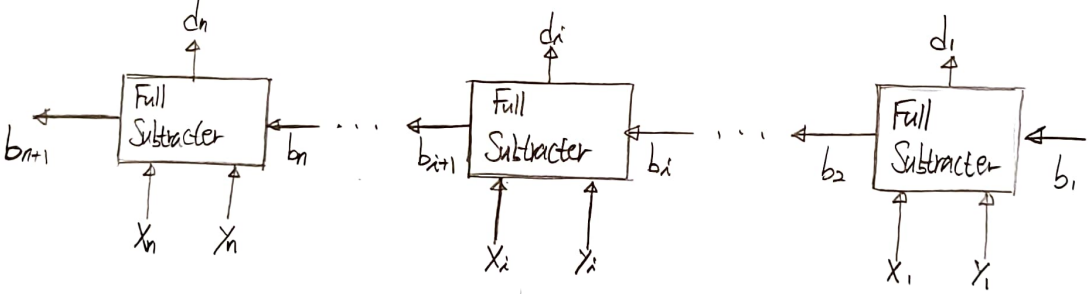
(2's complement $N^* = 2^n - N = \underbrace{(2^n - 1) - N}_{\text{이 과정이 inverting}} + 1$)

ex) 0101 (N) →

$$\begin{array}{r} 1111\ (2^n-1) \\ 0101\ (=N) \\ \hline 1010\ (\text{inverted}) \\ +\ 0001 \\ \hline 1011\ (N^*) \end{array}$$

• Alternative Subtracters : Parallel Subtracter.

Full Adder 처럼 복수 개를 연속 일반 기본 게이트를 써서 구성하고 싶다면.



Truth table

X_i	Y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

b : borrow, 즉 아랫단에 빌려준 수를 의미함.