

01

수의 체계와 변환

01. 수의 체계와 변환

I. 진법의 개념

- **진법** : 사용할 수 있는 숫자 개수와 각 숫자의 위치 값을 정의한 수 체계.
- 사람은 주로 10진법을, 컴퓨터는 2진법을 사용함.

하나 더 알기

진법을 알아야 하는 이유

- 컴퓨터는 0과 1, 디지털 형식으로 정보를 표현하는데, 이를 실생활에 적용하려면 아날로그 데이터를 디지털 데이터로 변환해야 함.
- 아날로그 데이터는 연속적 데이터, 디지털 데이터는 비연속적인 데이터를 의미.



그림 2-1 아날로그 시계와 디지털 시계

01. 수의 체계와 변환

II. 10진법과 2진법

- **10진법** : 0부터 9까지 10개의 숫자를 한 묶음으로 하여 10이 될 때마다 1자리씩 자리올림을 하는 방법.
- [그림 2-2]에서 자릿수의 의미
 - 4는 432를 100으로 나누었을 때의 '몫'
 - 32는 앞서 100으로 나누었을 때의 '나머지'
 - 2는 10으로 나누었을 때의 '나머지'

$$\begin{aligned} 432_{10} &= 4 \times 10^2 + 3 \times 10^1 + 2 \times 10^0 \\ &= 4 \times 100 + 3 \times 10 + 2 \times 1 \\ &= 432 \end{aligned}$$

그림 2-2 10진수 표현 방식

01. 수의 체계와 변환

II. 10진법과 2진법

- 컴퓨터는 0과 1, 두 가지 숫자로만 수를 표현하는 2진법을 사용.
- **컴퓨터가 2진법을 사용하는 이유** : 최초의 컴퓨터가 진공관을 사용했기 때문인데, 진공관은 켜고 끄는(on/off) 기능만 있었기 때문에 진공관이 꺼지면 0, 진공관이 켜지면 1로 인식.



그림 2-3 2진법 개념

01. 수의 체계와 변환

III. 진법 변환

▪ 10진수 → 2진수

- 정수 계산 방식 : 10진수를 계속 2로 나누면서 몫은 아래에, 나머지는 오른쪽에 기입하면 됨. 몫이 더 이상 2로 나누어지지 않을 때 아래에서부터 순서대로 나머지를 나열하면 2진수가 됨.

2)	41	(나머지)
2)	20	... 1
2)	10	... 0
2)	5	... 0
2)	2	... 1
	1	... 0

결과: 101001₂

그림 2-4 10진수 → 2진수 변환: 정수 부분

01. 수의 체계와 변환

III. 진법 변환

■ 10진수 → 2진수

- **소수 계산 방식** : 정수 부분은 앞서 말한 방식대로 변환. 소수점 아래의 소수 부분은 2를 계속 곱하면서 정수로 자리올림이 발생하는지 기록하고 이를 2진수 변환하면 됨.

$$\begin{array}{r} 0.6875 \\ \times 2 \\ \hline 1.3750 \text{ (} \rightarrow 0.1 \text{)} \\ 0.3750 \\ \times 2 \\ \hline 0.7500 \text{ (} \rightarrow 0.10 \text{)} \\ 0.7500 \\ \times 2 \\ \hline 1.5000 \text{ (} \rightarrow 0.101 \text{)} \\ 0.5000 \\ \times 2 \\ \hline 1.0000 \text{ (} \rightarrow 0.1011 \text{)} \end{array}$$

결과: 0.1011₂

그림 2-5 10진수 → 2진수 변환: 소수 부분

01. 수의 체계와 변환

III. 진법 변환

▪ 2진수 → 10진수

- 정수 계산 방식 : 2진수의 0과 1을 각 자릿수만큼의 2의 지수 승으로 곱한 후 모두 더하면 됨.

$$\begin{aligned} 101001_2 &= 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^0 \\ &= 1 \times 32 + 1 \times 8 + 1 \times 1 \\ &= 41_{10} \end{aligned}$$

그림 2-6 2진수 → 10진수 변환 : 정수 부분

01. 수의 체계와 변환

III. 진법 변환

▪ 2진수 → 10진수

- **소수 계산 방식** : 정수의 변환 방식과 동일하게 각 자릿수를 고려해 계산하면 됨. 단, 정수와는 반대로 소수점 아래로 내려갈수록 자릿수가 커지고 마이너스를 붙여 계산해야 함.

$$\begin{aligned} 0.1011_2 &= 1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-4} \\ &= 1 \times 0.5 + 1 \times 0.125 + 1 \times 0.0625 \\ &= 0.6875_{10} \end{aligned}$$

그림 2-7 2진수 → 10진수 변환 : 소수 부분

02

데이터의 표현

02. 데이터의 표현

I. 정수의 표현

- 정수(Integer) : 셀 수 있는 수를 의미.
- 정수의 구성 : 음의 정수, 0, 양의 정수

■ 부호 없는 정수

- 부호 없는 정수(Undsigned Integer) : 부호를 생략한다는 의미로, 모든 숫자는 0 또는 양의 정수.
- 8비트로 부호 없는 정수를 표현하면 $0_{10} \sim 255_{10}$ 까지 나타낼 수 있음.
- N비트를 이용한 부호 없는 정수 표현은 0부터 $(2^n - 1)$ 까지 가능.

$$\begin{aligned} 0000\ 0000_2 &\leftrightarrow 0_{10} \\ 1111\ 1111_2 &\leftrightarrow 255_{10} \end{aligned}$$

그림 2-8 8비트의 2진수 값을 10진수로 변환

02. 데이터의 표현

I. 정수의 표현

▪ 부호 없는 정수

- 부호 없는 정수에서 2진법을 이용한 사칙연산 방법은 기본적으로 10진법에서의 연산 방법과 동일.
- **덧셈 연산** : 2진법의 가장 오른쪽 비트인 최소유효비트(LSB)부터 시작해 각 비트의 수를 더하고, 1+1로 자리올림이 발생하면 상위 자리로 1을 올리면 됨.

$$\begin{array}{r} 0000\ 1010_2 = 10_{10} \\ + 1000\ 1010_2 = 138_{10} \\ \hline 1001\ 0100_2 = 148_{10} \end{array}$$

(a) $00001010_2 + 10001010_2$

$$\begin{array}{r} 0110\ 1010_2 = 106_{10} \\ + 1011\ 0011_2 = 179_{10} \\ \hline 1\ 0001\ 1101_2 = 285_{10} \end{array}$$

(b) $01101010_2 + 10110011_2$

그림 2-9 2진수의 덧셈 연산

02. 데이터의 표현

I. 정수의 표현

하나 더 알기

오버플로우

- **오버플로우(overflow)** : 덧셈의 결과가 8비트로 표현할 수 있는 범위를 넘어선 상황을 의미함.
- 컴퓨터 내부에서는 논리적으로 정확한 결과를 냈다 하더라도 표현 가능한 범위를 벗어났기 때문에 실제 수행 결과는 옳은 값이 아님.
- 즉, 계산 결과를 정확히 표현할 수 없음.
- 실제로 컴퓨터 프로그래밍 시 아주 큰 수끼리의 연산에서 오버플로우가 발생함.
- 문제는 연산 결과에 오류가 나더라도 프로그램은 계속 실행된다는 점.
- 틀린 값을 정확히 체크하지 않으면 최종 결과에 영향이 미침.

02. 데이터의 표현

I. 진법의 개념

▪ 부호 있는 정수

• 부호화 절댓값(Signed-magnitude) 표현

- 가장 왼쪽에 위치한 비트인 최대유효비트가 0이면 양의 정수(+0 포함)로, 최대 유효비트가 1이면 음의 정수(-0 포함)로 표현하는 방식.

$$0000\ 0101_2 \rightarrow +5_{10}$$

$$1000\ 0101_2 \rightarrow -5_{10}$$

그림 2-10 2진수의 부호화 절댓값 표현

- 8비트인 경우 절대값 계산을 위해 7비트만 사용할 수 있으므로 표현할 수 있는 값의 범위는 -127~127이다.
- n비트를 이용한 부호화 절댓값 표현은 $-(2^{n-1}-1) \sim +(2^{n-1}-1)$ 까지 가능.

02. 데이터의 표현

I. 진법의 개념

▪ 부호 있는 정수

• 보수(Complement) 표현

– 보수 : 두 수의 합이 진법의 밑수(N)가 되게 하는 수.

ex) 10진수 2_{10} 의 10의 보수는 8_{10} 이고, 10진수 6_{10} 의 10의 보수는 4_{10}

– 2의 보수는 절댓값이 같고 부호가 다른 두 수.

$$\begin{aligned} & \text{NOT } 0000\ 1010_2 + 1 \\ &= 1111\ 0101_2 + 1 \\ &= +10_{10} \text{의 2의 보수} \\ &= -10_{10} \end{aligned}$$

그림 2-13 NOT을 이용한 2의 보수 구하기

02. 데이터의 표현

II. 실수의 표현

- 실수(Real Number) : 유리수와 무리수를 총칭하여 확장한 수로, 수직선 위에 나타낼 수 있는 모든 수를 의미함.
- 컴퓨터 내부에서 실수를 표현하는 방법 : 고정소수점 표현, 부동소수점 표현

■ 고정소수점 표현

- [그림 2-14]와 같이 16비트를 사용하는 경우, 앞의 8비트는 정수 부분을 표현하고 나머지 8비트는 소수 부분으로 할당됨.



$$5.34_{10} = 101.01010111_2$$

그림 2-14 고정소수점 표현법

02. 데이터의 표현

II. 실수의 표현

▪ 고정소수점 표현

- 고정소수점 표현(Fixed-point Representation) : 소수점의 위치를 고정시켜 표현한다는 의미.
- 고정소수점 표현은 숫자 표현이 간단하기 때문에 연산 속도가 빠르다는 장점이 있지만, 일반적으로 컴퓨터에서는 이런 방식을 사용하지 않음.

02. 데이터의 표현

II. 실수의 표현

▪ 부동소수점 표현

- **부동소수점 표현(Floating-point Representation)** : 소수점의 위치를 고정시키지 않고 가수(유효숫자)와 지수(소수점의 위치)를 사용해 실수를 표현한다는 의미.
- 부동소수점으로 표현할 때는 정수 부분에 '0 아닌 수를 하나'만 남기는 정규형으로 먼저 바꿔야 함.
- [그림 2-15]를 보면 2진수 비트 열에서 지수는 가수 앞에 위치하며, 부호는 가수의 부호를 나타냄.

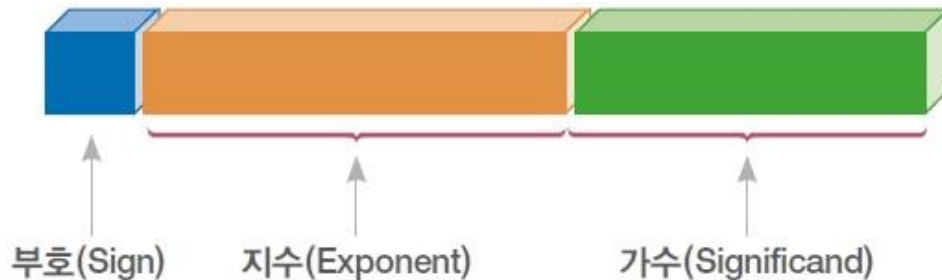


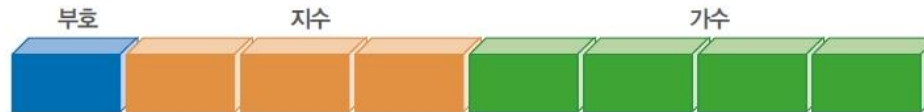
그림 2-15 부동소수점 표현 구조

02. 데이터의 표현

II. 실수의 표현

▪ 부동소수점 표현

- 8비트로 실수를 표현할 때 부호 1비트, 지수 3비트, 가수 4비트를 할당하면?



(a) 8비트를 사용한 실수 표현



(b) 1.0101110×2^{-2} 표현

그림 2-16 부동소수점 표현법

- 지수에 3비트를 할당할 때는 3초과표현을 사용.
- 정규형으로 표현된 1.0101110×2^{-2} 은 지수가 -2가 되기 때문에 지수 영역에는 3을 초과하여 $(-2+3)$ 1을 적는 것.
- 가수는 1.0101110 에서 4비트밖에 사용하지 못하므로, 앞에서부터 4비트를 잘라 1010이 됨.
- 0.34_{10} 는 컴퓨터 내부에서 00011010_2 이라는 2진수로 저장됨.

02. 데이터의 표현

III. 문자의 표현

■ 아스키코드

- **아스키코드(ASCII Code)** : 초창기 컴퓨터로 문자를 표현하는 과정에서 여러 문제가 발생해 문자를 표현할 약속 체계를 만들었는데, 그중 가장 많이 사용하는 코드 체계가 미국표준협회에서 만든 아스키코드임.
- 아스키코드는 7비트로 구성되어 있으며, 표현할 수 있는 문자 개수는 $128(2^7)$ 임.

02. 데이터의 표현

III. 문자의 표현

■ 아스키코드

표 2-2 아스키코드

10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호
0	0	NUL	32	20	SP	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	\$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(72	48	H	104	68	h
9	9	HT	41	29)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p

02. 데이터의 표현

III. 문자의 표현

■ 아스키코드

10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호	10 진수	16 진수	부호
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	
29	1D	GS	61	3D	=	93	5D]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

02. 데이터의 표현

III. 문자의 표현

▪ 유니코드

- **유니코드(Unicode)** : 아스키코드는 영어 문화권에만 사용 가능해, 다양한 나라의 언어를 표현하고자 만든 코드 체계임.
- 언어의 종류와 상관없이 모든 문자를 16비트(2바이트)로 구성해 65,536(2^{16})개의 문자 표현이 가능함.

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가 AC00	감 AC10	감 AC20	갸 AC30	갈 AC40	각 AC50	갹 AC60	거 AC70	검 AC80	겐 AC90	갸 ACA0	결 ACB0	격 ACC0	겹 ACD0	고 ACE0	곰 ACF0
1	각 AC01	갑 AC11	갸 AC21	갹 AC31	갈 AC41	갹 AC51	갹 AC61	걱 AC71	겁 AC81	겹 AC91	겹 ACA1	겹 ACB1	겹 ACC1	겹 ACD1	곡 ACE1	곱 ACF1
2	갹 AC02	갹 AC12	갹 AC22	갹 AC32	갹 AC42	갹 AC52	갹 AC62	긔 AC72	긔 AC82	겹 AC92	겹 ACA2	겹 ACB2	겹 ACC2	겹 ACD2	곡 ACE2	긔 ACF2

그림 2-17 한글 유니코드 일부

03

논리회로

03. 논리회로

I. 논리회로의 개념

- **논리회로(Logic Circuit)** : 논리합(OR), 논리곱(AND), 부정(NOT), 부정논리곱(NAND), 부정논리합(NOR) 등의 논리연산을 수행하는 회로.
- 컴퓨터는 전자 소자들의 집합체로, 이 전자 소자들이 0과 1을 표현하는데, 이 소자들의 연산 수행을 가능케 하는 것이 바로 논리회로임.



그림 2-18 전기적 신호의 개념적 형태 : 전압

03. 논리회로

I. 논리회로의 개념

- 논리회로의 동작이나 기능을 기술하는 방식
 - 논리식(Boolean Expression) : 논리연산자(AND, OR, NOT 등)로 구성된 식.
 - 논리도(Logic Diagram) : 논리식을 그림으로 표현해 시각적으로 구현한 방법.
 - 진리표(Truth Table) : 논리회로에 입력 가능한 모든 경우의 수에 대응하는 출력 값을 정의한 표.

03. 논리회로

I. 논리회로의 개념

▪ 논리연산자

- 논리회로에서의 '논리' : 참(True) 또는 거짓(False)을 다룬다는 의미.
- 불 대수(Boolean Algebra) : 컴퓨터가 사용하는 2진수 값과 논리연산을 다루는 분야를 의미하며, 논리연산이라고도 함.
- 논리연산자(Logical Operator) : 논리연산에 필요한 연산자로, 참 또는 거짓, 1 또는 0을 피연산자로 사용함.

03. 논리회로

I. 논리회로의 개념

▪ 논리연산자

• 논리연산자의 종류





- AND 연산자 : 두 조건이 모두 참일 때만 결과가 참이 되는 논리연산자.
- OR 연산자 : 두 조건이 모두 거짓일 때만 결과가 거짓이 되는 논리연산자.
- NOT 연산자 : 참과 거짓이 반대되는 논리연산자.
- XOR 연산자 : 두 조건이 서로 다를 때만 결과가 참이 되는 논리연산자.
- NAND 연산자 : AND 연산자와 NOT 연산자를 결합한 것으로, AND 연산자의 연산 결과를 부정하는 논리연산자.
- NOR 연산자 : OR 연산자와 NOT 연산자를 결합한 것으로, 양쪽의 변수가 거짓일 때만 진리표의 참값이 되는 논리연산자.

03. 논리회로

I. 논리회로의 개념

■ 논리게이트

표 2-3 논리연산자의 논리도, 논리식, 진리표

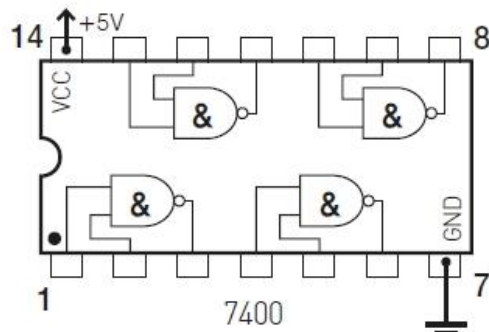
종류	논리도(심벌)	논리식(기호)	진리표																		
AND		$A \cdot B$	<table><tr><th colspan="2">입력</th><th>출력</th></tr><tr><th>A</th><th>B</th><th>A AND B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	입력		출력	A	B	A AND B	0	0	0	0	1	0	1	0	0	1	1	1
입력		출력																			
A	B	A AND B																			
0	0	0																			
0	1	0																			
1	0	0																			
1	1	1																			
OR		$A + B$	<table><tr><th colspan="2">입력</th><th>출력</th></tr><tr><th>A</th><th>B</th><th>A OR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	입력		출력	A	B	A OR B	0	0	0	0	1	1	1	0	1	1	1	1
입력		출력																			
A	B	A OR B																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	1																			
NOT		\overline{A}	<table><tr><th>입력</th><th>출력</th></tr><tr><th>A</th><th>NOT A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	입력	출력	A	NOT A	0	1	1	0										
입력	출력																				
A	NOT A																				
0	1																				
1	0																				
XOR		$A \oplus B$	<table><tr><th colspan="2">입력</th><th>출력</th></tr><tr><th>A</th><th>B</th><th>A XOR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	입력		출력	A	B	A XOR B	0	0	0	0	1	1	1	0	1	1	1	0
입력		출력																			
A	B	A XOR B																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	0																			

03. 논리회로

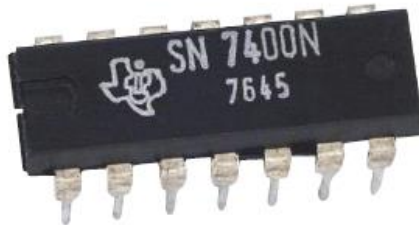
I. 논리회로의 개념

■ 논리연산자

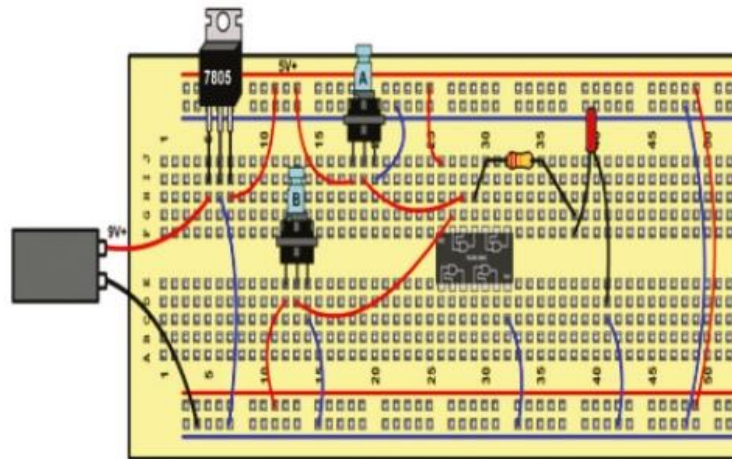
- 게이트는 [그림 2-19]와 같이 반도체 칩으로 구성되어 실제 회로 구성에 사용됨.



(a) 논리회로



(b) IC칩



(c) 브레드보드

그림 2-19 논리회로 실습 구현

03. 논리회로

II. 논리회로의 표현 방식

- 일반적인 논리회로는 [그림 2-20]처럼 입력과 출력의 개수, 이름만 표시되어 있음.
그래서 내부에서 어떤 동작이 일어나는지 알 수 없음.

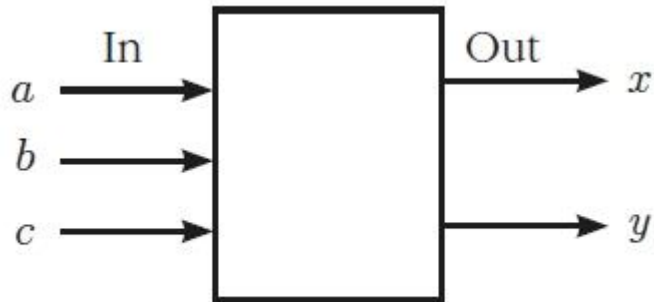


그림 2-20 입출력 관련 내용만 표시된 회로

03. 논리회로

II. 논리회로의 표현 방식

- 회로가 수행하는 기능을 설명할 때 가장 많이 사용하는 방법에는 진리표가 있음.
- [그림 2-21]은 입력이 3개 (a,b,c), 출력이 2개(x,y)인 회로인데, 이때 출력 x와 y는 독립적으로 동작한다고 생각하면 됨.
 - 출력 x : 빨간색 지시선 2개와 같이 입력 (a,b,c)가 (0,0,1) 또는 (0,1,1)인 경우 x가 1을 출력하는 회로. 그 외 입력에 대해선 0이 출력됨.
 - 출력 y : 초록색 지시선과 같이 입력이 (0,1,1) 또는 (1,0,0)인 경우 y는 1을 출력. 그 외의 경우는 0이 출력됨.

a	b	c	x	y
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

그림 2-21 입력 3개, 출력 2개인 회로의 진리표

03. 논리회로

II. 논리회로의 표현 방식

- x 를 출력하려면 $a'b'c$ 또는 $a'bc$ 인 경우에만 가능.
- 논리식(Logic Expression, Boolean Expression)으로 표현하면 다음과 같음.

$$x = a'b'c + a'bc$$

그림 2-22 출력 x 의 논리식

- ✓ **TIP.** a' 라는 점(Apostrophe) 표시는 NOT을 의미하는 것으로, 즉 $a'b'c$ 는 (a,b,c) 각각의 값이 $(0,0,1)$ 인 경우. 이때 a, b, c 각각은 AND 조건임.
- ✓ **TIP.** $+$ 기호는 OR를 의미. $a'b'c$ 이거나 $a'bc$ 인 경우.

03. 논리회로

II. 논리회로의 표현 방식

- [그림 2-22] 논리식을 좀 더 간단히 정리하면 $x = a'c$ 로 간단해짐.
- 전등(출력) x 는 스위치 a 가 off이고, 스위치 c 가 on일 때만 켜진다는 말.

$$\begin{aligned}x &= a'b'c + a'bc \\&= a'(b'c + bc) \\&= a'c(b' + b) \\&= a'c(1) \\&= a'c\end{aligned}$$

그림 2-23 출력 x 의 논리식 정리

- 출력 y 의 논리식은 다음과 같음.

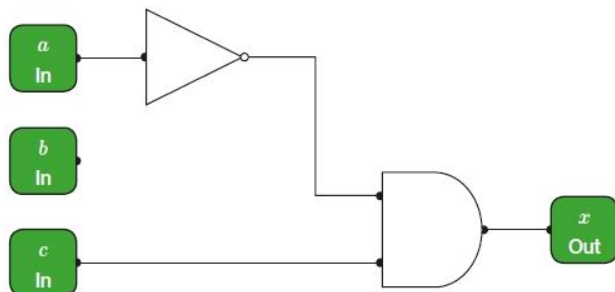
$$y = a'bc + ab'c'$$

그림 2-24 출력 y 의 논리식

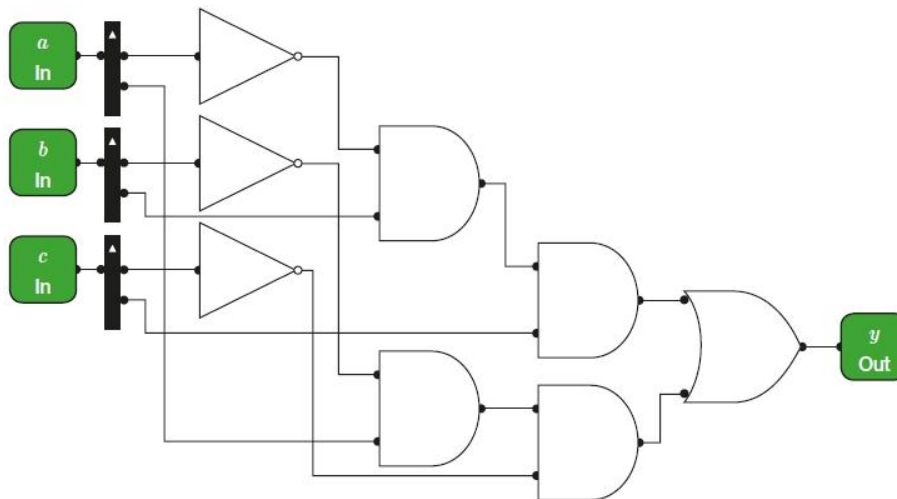
03. 논리회로

II. 논리회로의 표현 방식

- 논리식은 논리도(Logic Diagram) 또는 회로도(Circuit Diagram)로 표현 가능.



(a) $x = a'c$ 회로도



(b) $y = a'bc + ab'c'$ 회로도

그림 2-25 논리식을 정리한 회로도