이론, 실습, 시뮬레이션

디지털 논리회로

개정 3판

Chapter 03. 디지털 코드

학습목표 및 목차

- 다양한 디지털 코드를 구분하여 이해할 수 있다.
- 문자와 숫자를 나타내는 코드를 이해할 수 있다.
- 가중치 코드와 비가중치 코드를 이해하고 이를 활용할 수 있다.
- 에러 검출 코드를 이해하고 이를 활용할 수 있다.

01 BCD 코드와 3초과 코드

BCD 코드(Binary Coded Decimal Code : 2진화 10진 코드, 8421코드)

- BCD코드는 10진수 0(0000)부터 9(1001)까지를 2진화한 코드
- 표기는 2진수이지만 의미는 10진수
- 1010부터 1111까지 6개는 사용하지 않음

10진수	BCD 코드	10진수	BCD 코드	10진수	BCD 코드
0	0000	10	0001 0000	20	0010 0000
1	0001	11	0001 0001	31	0011 0001
2	0010	12	0001 0010	42	0100 0010
3	0011	13	0001 0011	53	0101 0011
4	0100	14	0001 0100	64	0110 0100
5	0101	15	0001 0101	75	0111 0101
6	0110	16	0001 0110	86	1000 0110
7	0111	17	0001 0111	97	1001 0111
8	1000	18	0001 1000	196	0001 1001 0110
9	1001	19	0001 1001	237	0010 0011 0111

01 BCD 코드와 3초과 코드

BCD 코드의 연산

10진 덧셈 (6+3=9)

```
   0110
+  0011
------
   1001
```

10진 덧셈 (42+27=69)

```
   0100 0010
+  0010 0111
-------------
   0110 1001
```

- 계산 결과가 BCD코드를 벗어나는 즉, 9(1001)를 초과하는 경우에는 계산 결과에 6(0110)을 더해준다.

(8+7=15)

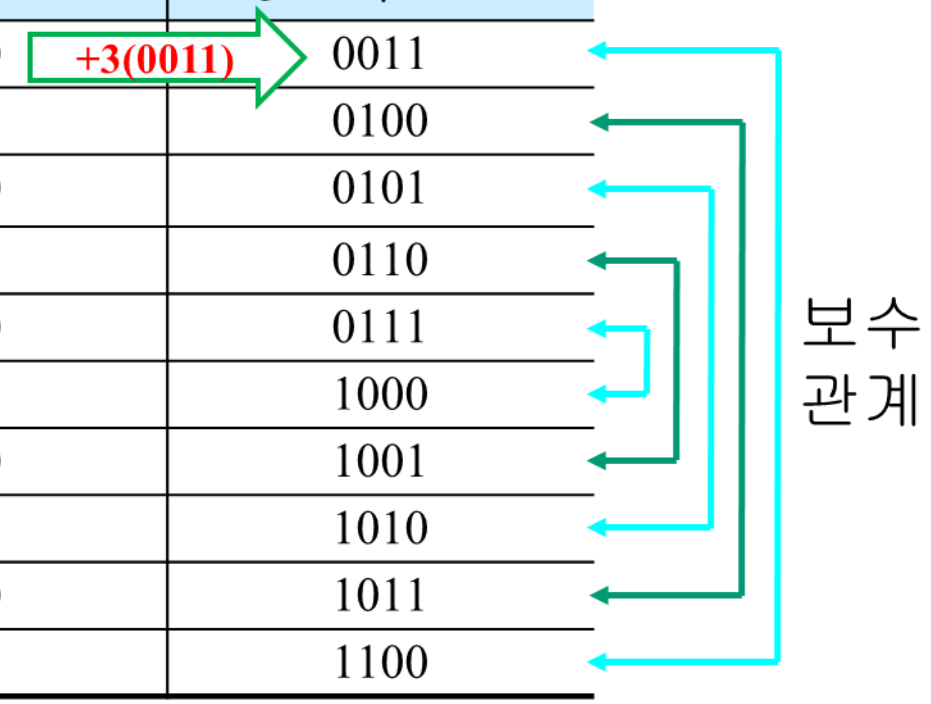
```
   1000
+  0111
------
   1111
+  0110   <- 6
------
0001 0101
```

01 BCD 코드와 3초과 코드

3초과 코드

- BCD코드(8421코드)로 표현된 값에 3을 더해 준 값으로 나타내는 코드
- 자기 보수의 성질

10진수	BCD 코드	3-초과 코드
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100



02 다양한 2진 코드들

가중치 코드(Weighted Code)

- 그 위치에 따라 정해진 값을 갖는 코드

10진수	8421코드 (BCD)	2421 코드	5421 코드	84-2-1 코드	51111 코드	바이퀴너리코드 (Biquinary Code) 5043210	링 카운터 (ring counter) 9876543210
0	0000	0000	0000	0000	00000	0100001	0000000001
1	0001	0001	0001	0111	00001	0100010	0000000010
2	0010	0010	0010	0110	00011	0100100	0000000100
3	0011	0011	0011	0101	00111	0101000	0000001000
4	0100	0100	0100	0100	01111	0110000	0000010000
5	0101	1011	1000	1011	10000	1000001	0000100000
6	0110	1100	1001	1010	11000	1000010	0001000000
7	0111	1101	1010	1001	11100	1000100	0010000000
8	1000	1110	1011	1000	11110	1001000	0100000000
9	1001	1111	1100	1111	11111	1010000	1000000000

02 다양한 2진 코드들

❖ 8421 코드(BCD 코드)

10진수	8421코드	10진수 계산
0	0000	$8\times0+4\times0+2\times0+1\times0=0$
1	0001	$8\times0+4\times0+2\times0+1\times1=1$
2	0010	$8\times0+4\times0+2\times1+1\times0=2$
3	0011	$8\times0+4\times0+2\times1+1\times1=3$
4	0100	$8\times0+4\times1+2\times0+1\times0=4$
5	0101	$8\times0+4\times1+2\times0+1\times1=5$
6	0110	$8\times0+4\times1+2\times1+1\times0=6$
7	0111	$8\times0+4\times1+2\times1+1\times1=7$
8	1000	$8\times1+4\times0+2\times0+1\times0=8$
9	1001	$8\times1+4\times0+2\times0+1\times1=9$

☞ 자기보수 성질 없음

02 다양한 2진 코드들

❖ 2421 코드

10진수	2421 코드	10진수 계산	2421 코드	10진수 계산
0	0000	2×0+4×0+2×0+1×0=0	0000	2×0+4×0+2×0+1×0=0
1	0001	2×0+4×0+2×0+1×1=1	0001	2×0+4×0+2×0+1×1=1
2	0010	2×0+4×0+2×1+1×0=2	1000	2×1+4×0+2×0+1×0=2
3	0011	2×0+4×0+2×1+1×1=3	1001	2×1+4×0+2×0+1×1=3
4	0100	2×0+4×1+2×0+1×0=4	1010	2×1+4×0+2×1+1×0=4
5	1011	2×1+4×0+2×1+1×1=5	0101	2×0+4×1+2×0+1×1=5
6	1100	2×1+4×1+2×0+1×0=6	0110	2×0+4×1+2×1+1×0=6
7	1101	2×1+4×1+2×0+1×1=7	0111	2×0+4×1+2×1+1×1=7
8	1110	2×1+4×1+2×1+1×0=8	1110	2×1+4×1+2×1+1×0=8
9	1111	2×1+4×1+2×1+1×1=9	1111	2×1+4×1+2×1+1×1=9

☞ 자기보수 성질을 가짐

02 다양한 2진 코드들

❖ 5421 코드

10진수	5421 코드	10진수 계산	5421 코드	10진수 계산
0	0000	5×0+4×0+2×0+1×0=0	0000	5×0+4×0+2×0+1×0=0
1	0001	5×0+4×0+2×0+1×1=1	0001	5×0+4×0+2×0+1×1=1
2	0010	5×0+4×0+2×1+1×0=2	0010	5×0+4×0+2×1+1×0=2
3	0011	5×0+4×0+2×1+1×1=3	0011	5×0+4×0+2×1+1×1=3
4	0100	5×0+4×1+2×0+1×0=4	0100	5×0+4×1+2×0+1×0=4
5	1000	5×1+4×0+2×0+1×0=5	0101	5×0+4×1+2×0+1×1=5
6	1001	5×1+4×0+2×0+1×1=6	0110	5×0+4×1+2×1+1×0=6
7	1010	5×1+4×0+2×1+1×0=7	0111	5×0+4×1+2×1+1×1=7
8	1011	5×1+4×0+2×1+1×1=8	1011	5×1+4×0+2×1+1×1=8
9	1100	5×1+4×1+2×0+1×0=9	1100	5×1+4×1+2×0+1×0=9

☞ 자기보수 성질 없음

02 다양한 2진 코드들

❖ 84-2-1 코드

10진수	84-2-1코드	10진수 계산
0	0000	8×0+4×0−2×0−1×0=0
1	0111	8×0+4×1−2×1−1×1=1
2	0110	8×0+4×1−2×1−1×0=2
3	0101	8×0+4×1−2×0−1×1=3
4	0100	8×0+4×1−2×0−1×0=4
5	1011	8×1+4×0−2×1−1×1=5
6	1010	8×1+4×0−2×1−1×0=6
7	1001	8×1+4×0−2×0−1×1=7
8	1000	8×1+4×0−2×0−1×0=8
9	1111	8×1+4×1−2×1−1×1=9

☞ 자기보수 성질을 가짐

02 다양한 2진 코드들

비가중치코드(non-weighted code)

- 각각의 위치에 해당하는 값이 없는 코드
- 데이터 변환과 같은 특수한 용도로 사용되기 위한 코드 (2-out-of-5)

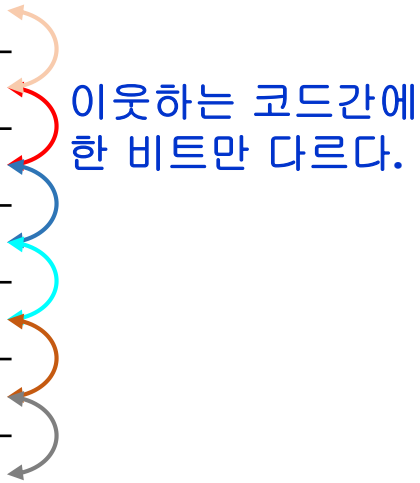
10진수	3-초과 코드	5중 2코드 (2-out-of-5)	shift counter	그레이코드
0	0011	11000	00000	0000
1	0100	00011	00001	0001
2	0101	00101	00011	0011
3	0110	00110	00111	0010
4	0111	01001	01111	0110
5	1000	01010	11111	0111
6	1001	01100	11110	0101
7	1010	10001	11100	0100
8	1011	10010	11000	1100
9	1100	10100	10000	1101

03 그레이 코드

그레이 코드(Gray Code)

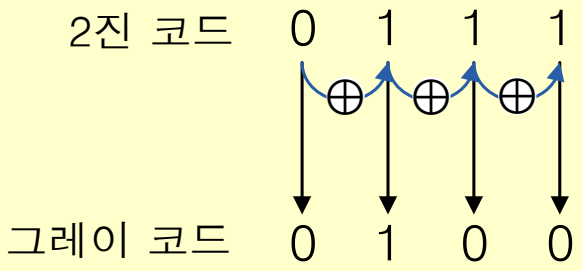
- 가중치가 없는 코드이기 때문에 연산에는 부적당하지만, 아날로그-디지털 변환기나 입출력 장치 코드로 주로 쓰인다.
- 연속되는 코드들 간에 하나의 비트만 변화하여 새로운 코드가 된다.

10진수	2진 코드	그레이 코드	10진수	2진 코드	그레이 코드
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

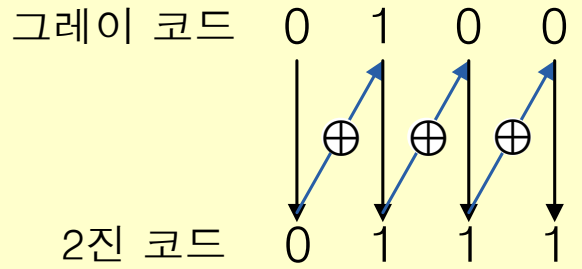


03 그레이 코드

2진 코드를 그레이 코드로 변환하는 방법



그레이 코드를 2진 코드로 변환하는 방법



<XOR 진리표>

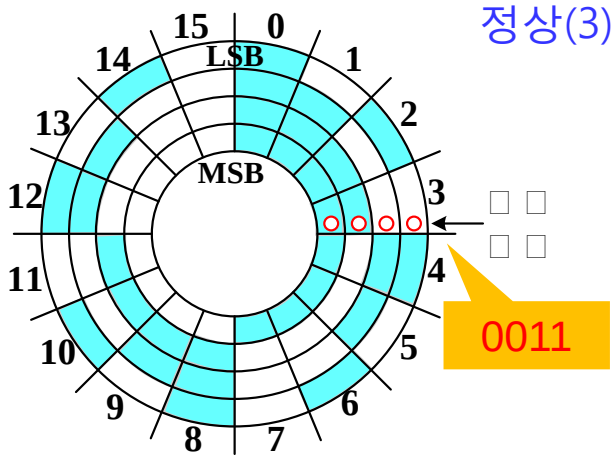
입력		출력
A	*B*	*F*
0	0	0
0	1	1
1	0	1
1	1	0

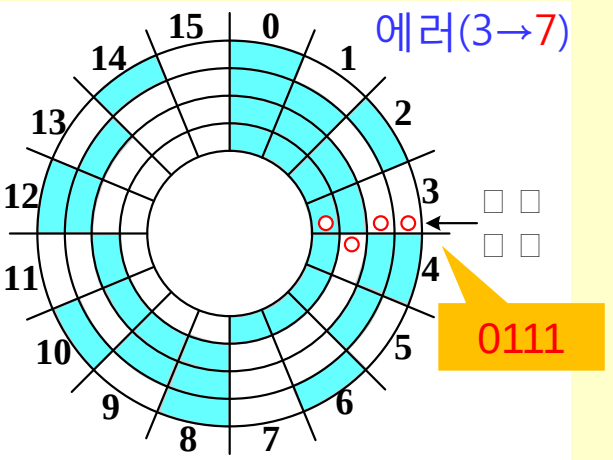
$F=A \oplus B$

03 그레이 코드

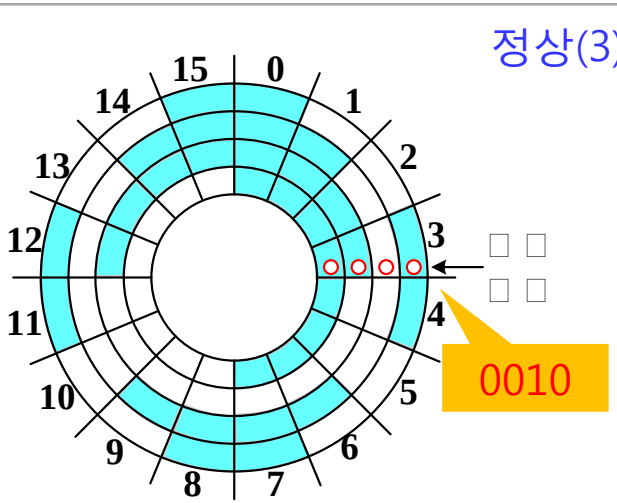
그레이 코드 입력장치 적용 예

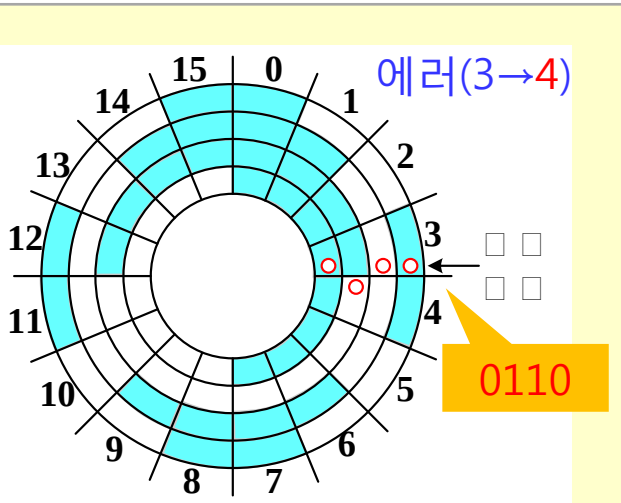
2진코드





그레이코드





0

1

그레이 코드는 오차가 적음

04 에러 검출 코드

1. 패리티 비트

- 짝수패리티(even parity) : 데이터에서 1의 개수를 짝수 개로 맞춤
- 홀수패리티(odd parity) : 1의 개수를 홀수 개로 맞춤
- 패리티 비트는 데이터 전송과정에서 에러 검사를 위한 추가 비트
  패리티는 단지 에러 검출만 가능하며, 여러 비트에 에러가 발생할 경우에는 검출이 안될 수도 있음

7비트 ASCII 코드에 패리티 비트를 추가한 코드

데이터	짝수패리티	홀수패리티
…	…	…
A	0 1000001	1 1000001
B	0 1000010	1 1000010
C	1 1000011	0 1000011
D	0 1000100	1 1000100
…	…	…

04 에러 검출 코드

병렬 패리티(parallel parity)

- 패리티를 블록 데이터에 적용해서 가로와 세로 데이터들에 대해서 패리티를 적용하면 에러를 검출하여 그 위치를 찾아 정정할 수 있다.

1	0	1	0	1	1	1	1	*0*
1	0	0	0	0	0	1	1	*1*
0	1	0	0	0	0	0	0	*1*
1	1	1	1	0	0	0	0	*0*
1	0	1	1	1	0	0	1	*1*
0	0	0	0	0	1	1	1	*1*
1	1	1	1	1	1	1	1	*0*
0	1	1	1	1	0	0	0	*0*
1	*0*	*1*	*0*	*0*	*1*	*0*	*1*	*0*

원래 데이터 블록

가로세로 모두 1의 개수가 짝수임

1	0	1	0	1	1	1	1	*0*
1	0	0	0	0	0	1	1	*1*
0	1	0	0	0	0	0	0	*1*
1	1	1	1	0	0	0	0	*0*
1	0	1	1	0	0	0	1	*1*
0	0	0	0	0	1	1	1	*1*
1	1	1	1	1	1	1	1	*0*
0	1	1	1	1	0	0	0	*0*
1	*0*	*1*	*0*	*0*	*1*	*0*	*1*	*0*

에러가 발생한 블록

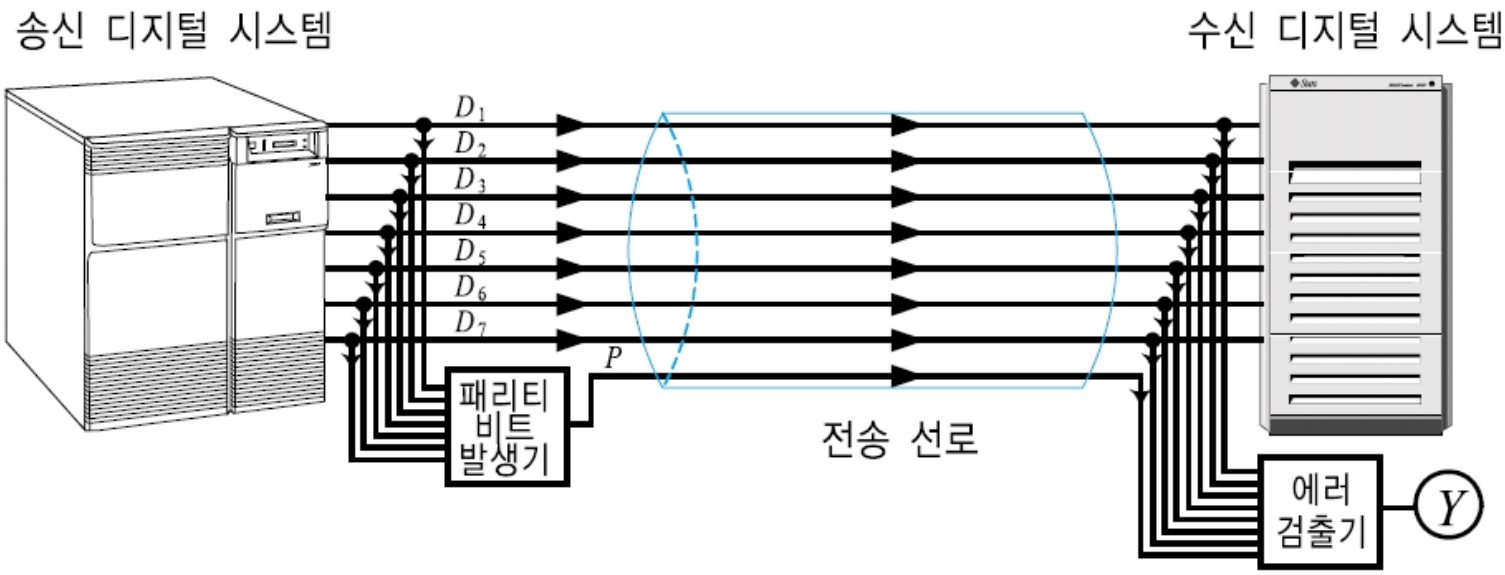
가로세로 회색 부분에 1의 개수가 홀수임 : 겹치는 부분 에러

04 에러 검출 코드

■ 데이터 전송 시스템에서 패리티 비트를 사용한 에러 검출

- 에러를 검출하기 위하여 송신측에 패리티 발생기를 구성하고 수신측에는 패리티 검출기를 구성하여 그 출력을 보고 에러 발생 여부를 판단

짝수 패리티	Y=0(**에러 없음**), Y=1(**에러 발생**)
홀수 패리티	Y=1(**에러 없음**), Y=0(**에러 발생**)



04 에러 검출 코드

2. 에러 정정 코드: 해밍코드(Hamming Code)

- 에러를 정정할 수 있는 코드
- 추가적으로 많은 비트가 필요하므로 많은 양의 데이터 전달이 필요
- 데이터 비트와 패리티 비트와의 관계

$$2^{p-1} - p + 1 \le d \le 2^{p} - p - 1$$

p는 패리티 비트의 수, d는 데이터 비트의 수, $p \ge 2$

- p=4일 때, $2^{4-1} - 4 + 1 \le d \le 2^{4} - 4 - 1$이므로 $5 \le d \le 11$이다.
- 따라서 데이터 비트수가 5개 이상 11개 이하일 때 패리티는 4개가 필요하다.
- 패리티 비트의 위치는 앞에서 부터 $2^0, 2^1, 2^2, 2^3, 2^4, \ldots$번째, 즉 1, 2, 4, 8, 16, … 번째이다.
- 데이터 비트는 나머지 위치에 순서대로 들어간다.

04 에러 검출 코드

- 해밍코드에서는 짝수 패리티를 사용

비트 위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
P_1 영역	✓		✓		✓		✓		✓		✓	
P_2 영역		✓	✓			✓	✓			✓	✓	
P_4 영역				✓	✓	✓	✓					✓
P_8 영역								✓	✓	✓	✓	✓

04 에러 검출 코드

8비트 데이터의 에러 정정 코드

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11}$$
$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11}$$
$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12}$$
$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12}$$

- For Example

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
		0		0	1	0		1	1	1	0

$$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$$
$$P_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$
$$P_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$
$$P_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 = 1$$

04 에러 검출 코드

해밍코드에서 패리티 비트 생성 과정

비트위치	1	2	3	4	5	6	7	8	9	10	11	12
기호	P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
원본 데이터			0		0	1	0		1	1	1	0
P_1 영역	0		0		0		0		1		1	
P_2 영역		1	0			1	0			1	1	
P_4 영역				1	0	1	0					0
P_8 영역								1	1	1	1	0
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
생성된 코드	0	1	0	1	0	1	0	1	1	1	1	0

생성된 패리티

04 에러 검출 코드

해밍코드에서 패리티 비트 검사 과정

전송된 데이터 : 010111011110

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
0	1	0	1	1	1	0	1	1	1	1	0

☞ 패리티들을 포함하여 검사

$$P_1 = P_1 \oplus D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} = 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$P_2 = P_2 \oplus D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$P_4 = P_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$$

$$P_8 = P_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$

- 검사된 패리티를 $P_8\ P_4\ P_2\ P_1$ 순서대로 정렬
- 모든 패리티가 0이면 에러 없음
- 하나라도 1이 있으면 에러 발생: 결과가 0101이므로 에러 있음
- 0101을 10진수로 바꾸면 5이며, 수신된 데이터에서 앞에서 5번째 비트 010**1**1011110에 에러가 발생한 것이므로 010**0**1011110으로 바꾸어 주면 에러가 정정된다.

04 에러 검출 코드

■ 해밍코드에서 에러가 발생한 경우 교정

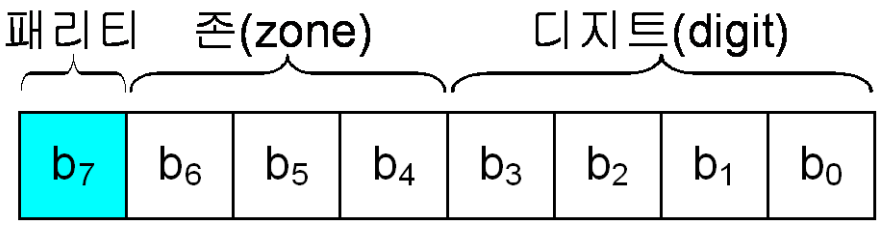
비트위치		1	2	3	4	5	6	7	8	9	10	11	12
기호		P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
Error 해밍코드		0	1	0	1	1	1	0	1	1	1	1	0
P_1 계산	1	0		0		1		0		1		1	
P_2 계산	0		1	0			1	0			1	1	
P_4 계산	1				1	1	1	0					0
P_8 계산	0								1	1	1	1	0

$P_8P_4P_2P_1$ =0101= ***5*** : 5번째 비트에 에러 발생, $1 \rightarrow 0$으로 교정

05 영숫자 코드

1. ASCII(American Standard Code for Information Interchange) 코드

- 미국 국립 표준 연구소(ANSI)가 제정한 정보 교환용 미국 표준 코드
- 128가지의 문자를 표현 가능



■ ASCII 코드의 구성

parity	zone bit			digit bit			
7	6	5	4	3	2	1	0
C	1	0	0	영문자 A~O(0001~1111)			
	1	0	1	영문자 P~Z(0000~1010)			
	0	1	1	숫자 0~9(0000~1001)			

05 영숫자 코드

표준 ASCII 코드표

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;		=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	|	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

05 영숫자 코드

확장 ASCII 코드표

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	Pt	f
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌜	⌟	½	¼	¡	«	»
B	▒	▓	▓	┃	┤	╡	╢	╖	╕	╣	╗	╝	╜	╛	┘	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▮	▄	▌	▐	▀
E	α	β	Γ	π	∑	σ	µ	τ	Φ	Θ	Ω	δ	∞	Ø	ε	∩
F	≡	±	≥	≤	⌠	⌡	÷	≈	◦	•	·	√	n	2	■	

05 영숫자 코드

2. 표준 BCD 코드

- 6비트로 하나의 문자를 표현
- 최대 64문자까지 표현 가능한 코드

코드의 구성

parity	zone bit		digit bit			
6	5	4	3	2	1	0
C	1	1	영문자 A~I(0001~1001)			
	1	0	영문자 J~R(0001~1001)			
	0	1	영문자 S~Z(0010~1001)			
	0	0	숫자 0~9(0001~1010)			
	혼용		특수문자 및 기타문자			

05 영숫자 코드

표준 BCD 코드표

문자	C ZZ8421	문자	C ZZ8421	문자	C ZZ8421	문자	C ZZ8421	문자	C ZZ8421
A	0 110001	J	1 100001	S	1 010010	1	0 000001	=	0 001011
B	0 110010	K	1 100010	T	0 010011	2	0 000010	>	1 001100
C	1 110011	L	0 100011	U	1 010100	3	1 000011	+	0 010000
D	0 110100	M	1 100100	V	0 010101	4	0 000100	,	1 011011
E	1 110101	N	0 100101	W	0 010110	5	1 000101	)	0 011100
F	1 110110	O	0 100110	X	1 010111	6	1 000110	%	1 011101
G	0 110111	P	1 100111	Y	1 011000	7	0 000111	?	0 011111
H	0 111000	Q	1 101000	Z	0 011001	8	0 001000	-	1 100001
I	1 111001	R	0 101001			9	1 001001	@	1 111010
						0	1 001010	$	1 111111

05 영숫자 코드

3. EBCDIC(Extended Binary Coded Decimal Interchange Code) 코드

- 대형 컴퓨터와 IBM 계열 컴퓨터에서 많이 사용되고 있는 8비트 코드(IBM에서 개발)
- 256종류의 문자 코드를 표현할 수 있는 영숫자 코드

코드의 구성

b_9	$b_8\ b_7\ b_6\ b_5$	$b_4\ b_3\ b_2\ b_1$
패리티	존(zone)	디지트(digit)
1	4	4

$b_8\ b_7$		$b_6\ b_5$	
0 0	통신제어문자		
0 1	특수문자		
1 0	소문자	0 0	a ~i
		0 1	j~r
		1 0	s~z
		1 1	
1 1	대문자/숫자	0 0	A~I
		0 1	J~R
		1 0	S~Z
		1 1	0~9

05 영숫자 코드

EBCDIC 코드표

16진		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	2진	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL	SOH	STX	ETX		HT		DEL				VT	FF	CR	SO	SI
1	0001	DLE						BS		CAN	EM			IFS	IGS	IRS	IUS
2	0010						LF	ETB	ESC						ENQ	ACK	BEL
3	0011			SYN					EOT						NAK		SUB
4	0100	space										[	.		(	+	|
5	0101	&										!	$	*	)		^
6	0110	-	/									|	,	%	_	>	?
7	0111										`	:	#	@	‘	=	"
8	1000		a	b	c	d	e	f	g	h	i						
9	1001		j	k	l	m	n	o	p	q	r						
A	1010		~	s	t	u	v	w	x	y	z						
B	1011																
C	1100	{	A	B	C	D	E	F	G	H	I						
D	1101	}	J	K	L	M	N	O	P	Q	R						
E	1110	\		S	T	U	V	W	X	Y	Z						
F	1111	0	1	2	3	4	5	6	7	8	9						

05 영숫자 코드

4. 유니코드(Unicode)

- ASCII 코드의 한계성을 극복하기 위하여 개발된 인터넷 시대의 표준
- 유니코드 컨소시엄(IBM, Novell, Microsoft, DEC, Apple 등)에 의해서 32(UTF-32), 16(UTF-16), 8bit(UTF-8)의 세 가지 기본 코드
- 미국, 유럽, 동아시아, 아프리카, 아시아 태평양 지역 등의 주요 언어들에 적용될 수 있다.
- 유니코드는 유럽, 중동, 아시아 등 거의 대부분의 문자를 포함하고 있으며, 10만 개 이상의 문자로 구성되어 있다.
- 특히 아시아의 중국, 일본, 한국, 타이완, 베트남, 싱가포르에서 사용하는 표의 문자(한자) 70,207개를 나타낼 수 있다.
- 구두표시, 수학기호, 전문기호, 기하학적 모양, 딩벳 기호 등을 포함
- 앞으로도 계속해서 산업계의 요구나 새로운 문자들을 추가하여 나갈 것이다.

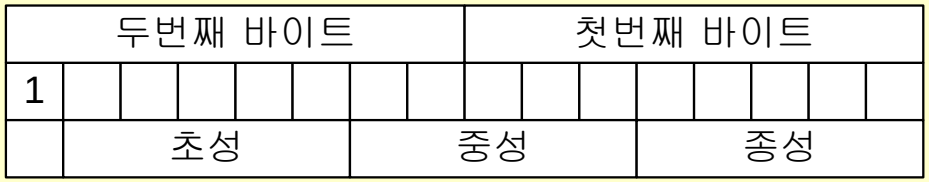
05 영숫자 코드

5. 한글코드

- 한글은 ASCII코드를 기반으로 16비트를 사용하여 하나의 문자를 표현
- 조합형과 완성형으로 분류

조합형

- **조합형으로 표현된 한글은 때에 따라서 다른 응용프로그램에서는 사용할 수 없는 문자들이 많다.**
- **조합형은 자음과 모음으로 조합 가능한 모든 한글을 사용할 수 있으며, 심지어 우리나라 고어(古語)까지 취급할 수 있는 장점이 있으나, 출력 시 다시 모아 써야 하는 불편이 있다는 것이 단점이다.**



완성형

- **1987년 정부가 한국표준으로 정한 것으로 가장 많이 사용되는 한글 음절을 2바이트의 2진수와 1 대 1로 대응하여 표현하는 방법**

감사합니다 ☺