Sistemas de Procesamiento de Datos – Unidad 3

- Sistemas numéricos
- Algebra de Boole
- Compuertas
- Transcodificadores

Profesor: Fabio Bruschetti
Ayudante: Pedro Iriso
Ver 2020

Sistemas numéricos

- Sistema numéricos basados (base = b)
  - La base define el conjunto de símbolos
  - Un número = es una secuencia de símbolos
  - Reglas
    - Los dígitos (d) se enumeran de derecha a izquierda desde 0 (d_3 d_2 d_1 d_0)
    - La ubicación de cada dígito tiene un “peso” definido por la base
      - peso = base $^{\text{posición}}$
    - El valor del dígito depende del símbolo y de su ubicación
      - Valor = dígito * peso
    - El valor del número es la suma de los valores de sus dígitos

$$Valor = \sum_0^n d_i * b^i$$

Sistemas numéricos

- Sistema Decimal
  - Base = 10
  - Símbolos {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Notación: NNN_d – NNN_{10}
  - Ejemplo de 4 dígitos
    - $d_3\ d_2\ d_1\ d_0 = 1436_d$
      - $= d_3 \times 10^3 + d_2 \times 10^2 + d_1 \times 10^1 + d_0 \times 10^0$
      - $= 1 \times 10^3 + 4 \times 10^2 + 3 \times 10^1 + 6 \times 10^0$
- Nota:
  - Sumas y Restas: Me “llevo” potencias de 10 (10, 100,1000, etc.)
  - Multiplicación por 10: Agrego un cero por la derecha desplazando los dígitos 1 lugar hacia la izquierda.
  - División por 10: Agrego un cero por la izquierda desplazando los dígitos 1 lugar hacia la derecha y desecho el dígito menos significativo..
  - Desplazar una cifra agregando ceros por derecha o izquierda → (“Shift”)

Números binarios enteros positivos

- Sistema Binario
  - Base = 2
  - Símbolos = {0, 1}
  - Notación: NNN_b – NNN_2
  - Ejemplo de 4 dígitos
    - $d_3\ d_2\ d_1\ d_0 = 1010_b$
      - $= d_3 \times 2^3 + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0$
      - $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
      - $= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$
      - $= 8 + 4$
      - $= 10_d$
- Nota:
  - Sumas y Restas: Me llevo potencias de 2
  - Multiplicación y División por 2: Ídem decimal

Números binarios enteros positivos

Suma

111_b	1010_b
$+ \ 10_b$	$+ \ 111_b$
---	---
1001_b	10001_b

Resta

101_b	1001_b
$- \ 10_b$	$- \ 111_b$
---	---
011_b	010_b

Multiplicación por potencias de 2

Por 2^1 : $100_b * 10_b = 1000_b$

Por 2^2 : $11_b * 100_b = 1100_b$

Por 2^3 : $100_b * 1000_b = 100000_b$

División por potencias de 2

Por 2^1 : $100_b / 10_b = 10_b$

Por 2^2 : $1100_b / 100_b = 11_b$

Por 2^3 : $100000_b / 1000_b = 100_b$

Números binarios enteros positivos

- Dentro del computador, todos los números son representados sobre una cantidad fija de bits.
- Rango de representación
  - Con n bits se pueden formar 2^n combinaciones binarias distintos. Cada una de ellas corresponde a su respectivo número decimal
  - Si se comienza la representación en 0, entonces el número más grande representable es $2^n - 1$
  - Se pueden usar n bits para representar los números decimales
  - Ejemplo con 4 bits

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

Números hexadecimales enteros positivos

- Sistema Hexadecimal
  - Base = 16 = 2^4
  - Permite manejar mejor los números binarios grandes
  - Notación: NNN_h – NNN_{16}
  - Símbolos = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
  - Ejemplo de 4 dígitos
    - $d_3\ d_2\ d_1\ d_0 = 12AF_h$
      - $= d_3 \times 16^3 + d_2 \times 16^2 + d_1 \times 16^1 + d_0 \times 16^0$
      - $= 1 \times 16^3 + 2 \times 16^2 + A \times 16^1 + F \times 16^0$
      - $= 1 \times 4096 + 2 \times 256 + 10 \times 16 + 15 \times 1$
      - $= 4096 + 512 + 160 + 15$
      - $= 4783_d$
- Nota:
  - Sumas y Restas: Pueden llevar o quitar 16's
  - Multiplicación y División por 16: "Shift" a Izquierda o Derecha

Números hexadecimales enteros positivos

Suma

181_h	1510_h
$+ \ 89_h$	$+ \ E11_h$
---	---
$20A_h$	2321_h

Resta

$1E1_h$	1001_h
$- \ 1F_h$	$- \ 111_h$
---	---
$1C2_h$	$EF0_h$

Multiplicación por potencias de 16

Por 16^1 : $20_h * 10_h = 200_h$

Por 16^2 : $3_h * 100_h = 300_h$

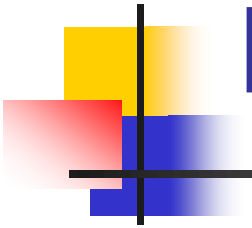
División por potencias de 16

Por 16^1 : $105_h / 10_h = 10_h$

Por 16^2 : $10E0_h / 100_h = 10_h$

Conversiones

- Binario a Hexadecimal
  - 16 bits: 1000101001101110_2 (= 35468_d)
    - Agrupando en 4: 1000 1010 0110 1110
    - Reemplazo con Hex: 8 A 6 E_h
  - 10 bits: 1001011001_2 (= 601_d)
    - Agrupando en 4: 0010 0101 1001
    - Reemplazo con Hex: 2 5 9_h
- Hexadecimal a Binario
  - Binario: Reemplazar cada dígito hexa por 4 dígitos binario
    - Ejemplo: 134_h = 0001 0011 0100_b



Preguntas

- ¿Qué significa cuando se dice “diez hex”?
- ¿Es verdad que $1010\ 0010_b = A2_h$?
- ¿Cuántos bits tienen las siguientes cifras?
  - 0010010_b
  - 2_h
  - 22_h
  - 1010_h
  - 1010_b

Unidades de Medidas

- Algunas abreviaturas:
  - Nibble = 4 bits
  - Byte = 8 Bits
  - Word (palabra)
    - 8 bits, 16 bits, 32 bits, 64 bits +
  - DWord (palabra doble)

Sistema de unidades (SI)			Sistema Binario			
Factor	Nombre	Símbolo	Factor	Nombre	Símbolo	Cantidad de bytes
10^3	kilobyte	KB	2^{10}	kibibyte	KiB	1,024
10^6	megabyte	MB	2^{20}	mebibyte	MiB	1,048,576
10^9	gigabyte	GB	2^{30}	gibibyte	GiB	1,073,741,824
10^{12}	terabyte	TB	2^{40}	tebibyte	TiB	1,099,511,627,776
10^{15}	petabyte	PB	2^{50}	pebibyte	PiB	1,125,899,906,842,624
10^{18}	exabyte	EB	2^{60}	exbibyte	EiB	1,152,921,504,606,846,976
10^{21}	zettabyte	ZB	2^{70}	zebibyte	ZiB	1,180,591,620,717,411,303,424
10^{24}	yottabyte	YB	2^{80}	yobibyte	YiB	1,208,925,819,614,629,174,706,176

Operaciones lógicas

- Algebra de Boole
  - Sean A y B variables binarias
    - Suma lógica (unión)
      - $A + 1 = 1$ y $A + 0 = A$
    - Producto lógico (intersección)
      - $A \cdot 1 = A$ y $A \cdot 0 = 0$
    - Negación (complemento)
      - Si $A = 1$, $\overline{A} = 0$ y si $A = 0$, $\overline{A} = 1$
      - $A + \overline{A} = 1$ y $A \cdot \overline{A} = 0$
    - Conmutatividad y distributividad
    - De Morgan
      - $\overline{A + B} = \overline{A} \cdot \overline{B}$
      - $\overline{A \cdot B} = \overline{A} + \overline{B}$

Compuertas lógicas

- Implementación con Compuertas
  - Unión → OR u "O"

$s = a + b$

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

  - Intersección → AND o "Y"

$s = a \cdot b$

a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

  - Negación → NOT

$s = \bar{a}$

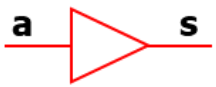
a	s
0	1
1	0

  - Suma binaria → XOR u "O Exclusivo"

$s = a.\bar{b} + \bar{a}.b$

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

  - Buffer

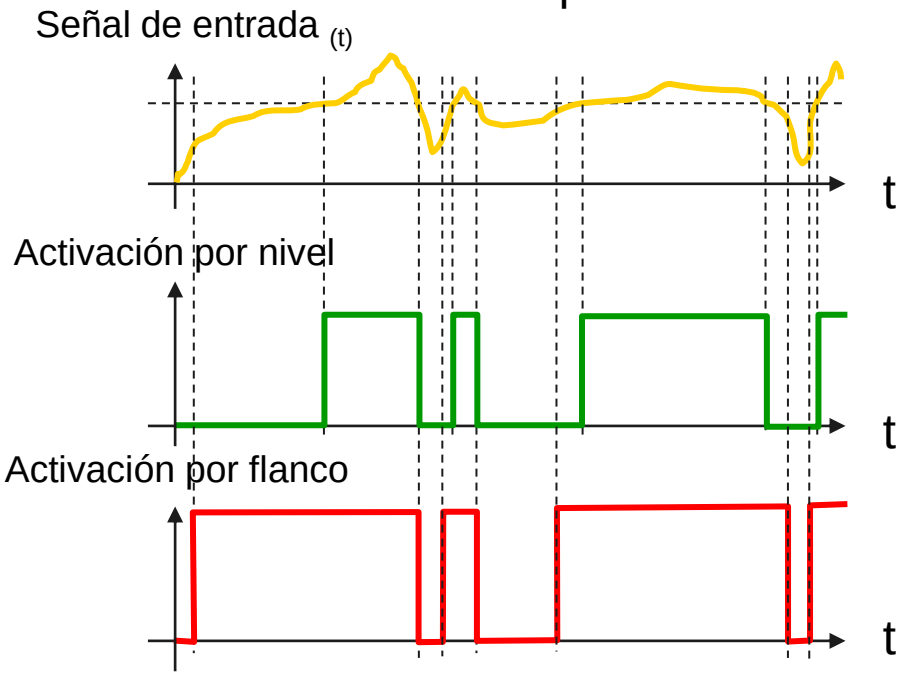


$s = a$

a	s
0	0
1	1

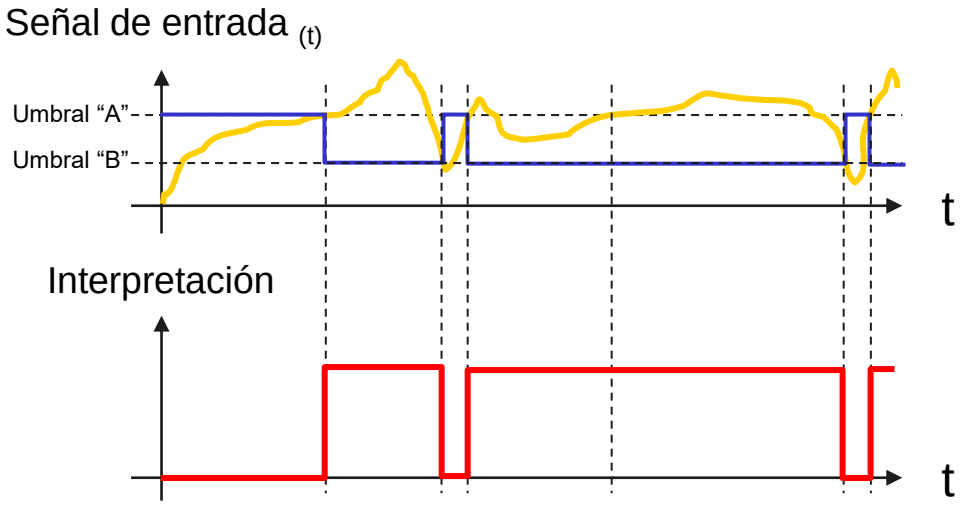
Compuertas lógicas – Activación

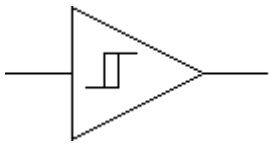
- Activación por nivel
  - Cuando la señal de entrada a una compuerta alcanza un cierto valor (umbral), se interpreta un cambio de estado en la compuerta
- Activación por flanco
  - Cuando la pendiente de una señal alcanza un determinado valor, se interpreta un cambio de estado en la compuerta



Compuertas lógicas – Entrada

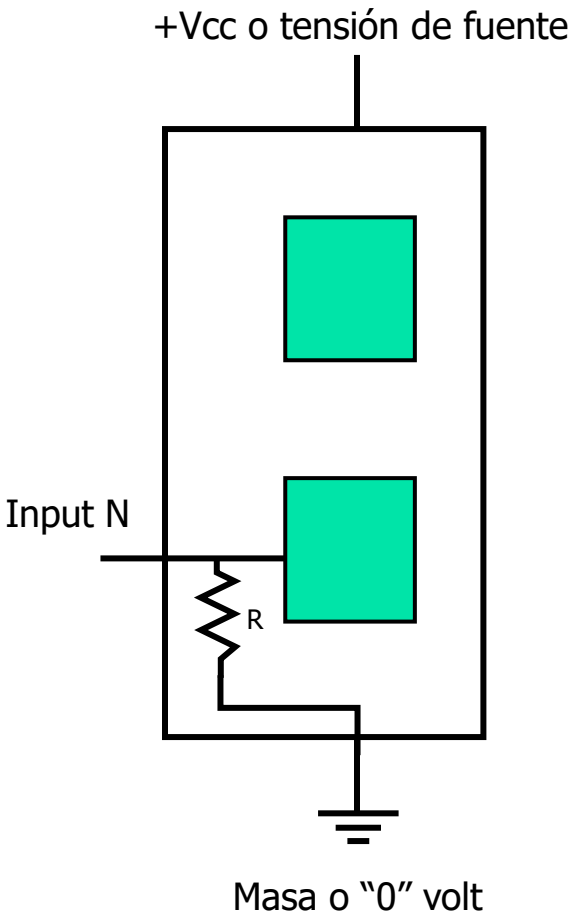
- Entrada Schmitt-Trigger
  - Cambia su estado de salida cuando la tensión en su entrada sobrepasa un determinado umbral “A”
  - Una vez sobrepasado, el umbral se posiciona en un valor “B” menor que “A”
  - La salida vuelve a cambiar cuando se atraviesa por debajo del umbral “B”
  - En esa situación, el umbral vuelve al valor “A”
  - Se la utiliza para prevenir el ruido que podría solaparse a la señal original y que causaría falsos cambios de estado si los niveles de referencia y entrada son parecidos





Compuertas lógicas – Entrada

- Entradas “a masa” (grounded inputs)
  - Como los circuitos digitales manejan señales muy bajas, son muy susceptibles al ruido
  - Si la entrada de un dispositivos no está conectada “a nada”, por allí se puede inyectar ruido (como si fuese una antena…)
  - Por error puede desconectarse la entrada de un dispositivo con lo cual quedará “flotando”
  - Por seguridad, algunos dispositivos, cada entrada digital es conectada a masa, potencial de referencia, o “a tierra” mediante una resistencia de alto valor
  - Esto previene una eventual desconexión u olvido y elimina la necesidad de conectar físicamente una resistencia en la plaqueta o circuito impreso en donde se encuentra el dispositivo conectado

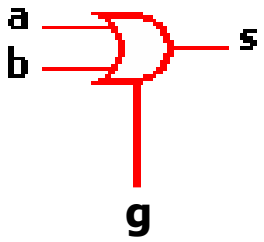


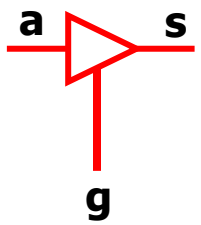
Compuertas lógicas - Salida

- Salida 3 estados (3-state u Open Collector)
  - Existe un tercer estado que no es ni 1 (uno) ni 0 (cero)
  - El tercer estado se llama Alta Impedancia o HZ (por High Z)
  - Que exista alta impedancia se puede interpretar como que la salida de la compuerta está virtualmente desconectada, como si hubiese "desoldado" físicamente el conector con el exterior
  - Enormemente utilizada para relevar o aislar circuitos entre si
  - La señal "g" (gate) coloca a la salida de la compuerta en el estado HZ
  - Ejemplo compuerta OR 3-state

g	a	b	s = a+b
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	X	X	HZ

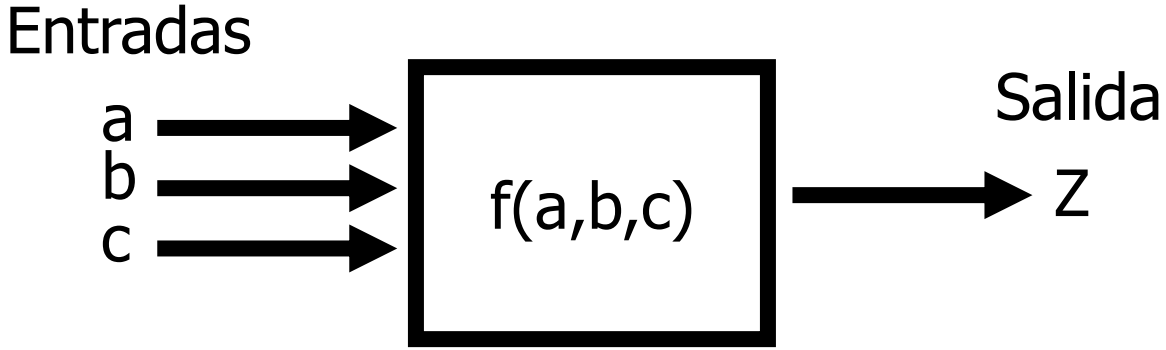
X = Don´t care





Funciones lógicas

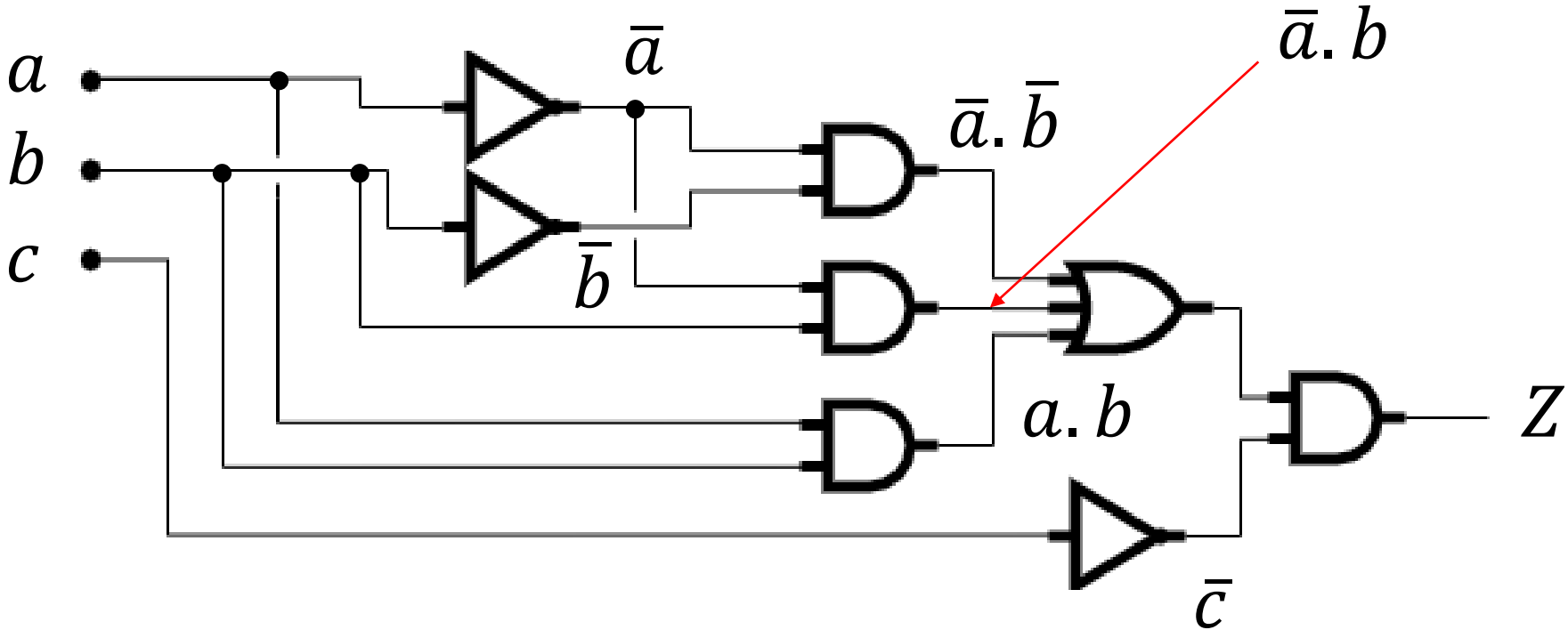
- Función lógica combinacional (FLC)
  - Se tiene una FLC cuando **depende únicamente de sus variables de entrada**
- $Z = f(a, b, c) = \bar{c}.[\bar{a}.\bar{b} + \bar{a}.b + a.b]$



- Realizar el circuito de la función Z

Funciones lógicas

$$Z = f(a, b, c) = \bar{c}.[\bar{a}.\bar{b} + \bar{a}.b + a.b]$$



Codificación de Caracteres

- Representación de caracteres mostrables:
  - Caracteres: { A, B , . . . , Y, Z }
  - 26 ×2 = 52 (mayúsculas y minúsculas)
  - Dígitos (10) decimales: {0, 1, . . . , 8, 9 }
  - Puntuación: ! ”’, . –? / : ;
  - Símbolos matemáticos: + * = (– y /)
  - Paréntesis: ( ) [ ] { } < >
  - Otros: @ # $ % ^ & \| ~
  - Espacio en blanco: “ ”
  - 90+ Símbolos (??)
  - Varios esquemas de codificación han sido usados

Codificación ASCII (7-bits)

- ASCII = **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - 7 bits para codificar cada caracter (128 códigos)
  - Se extiende a 8 bits (byte) poniendo el bit más significativo = 0
  - 2 dígitos hexadecimales
- Ejemplo
  - "Hola Mundo!" codificado en ASCII (en hexa)

  48_h $6F_h$ $6C_h$ 61_h 20_h $4D_h$ 75_h $6E_h$ 64_h $6F_h$ 21_h
- Otros esquemas de codificación:
  - IBM estandarizó un esquema de 8-bits (256 caracteres) por defecto (PC's !)
  - Java: esquema unicode–16–bits (65,536 caracteres) – Conjunto de caracteres multi-lenguaje

Codificación ASCII (7-bits)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Codificación ASCII Extendida (8-bits)

Extended ASCII Codes

As people gradually required computers to understand additional characters and non-printing characters the ASCII set became restrictive. As with most technology, it took a while to get a single standard for these extra characters and hence there are few varying 'extended' sets. The most popular is presented below.

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	⁻
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	₧	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Códigos de paridad

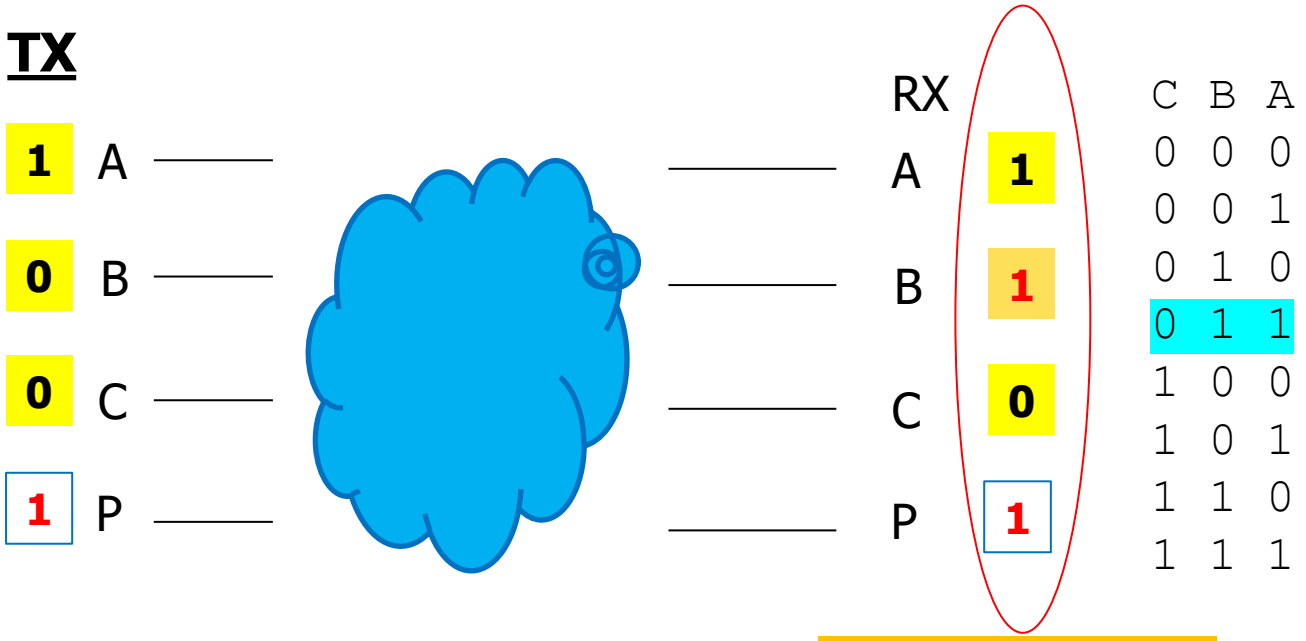
- Una combinación binaria puede contener una cantidad para o impar de unos (por ende de ceros también)
  - Paridad PAR = cantidad par de "unos"
  - Paridad IMPAR = cantidad impar de "unos"
  - Cualquier combinación binaria puede convertirse en una combinación de paridad par o impar, agregando un bit más que cumpla con las condiciones de paridad
  - Ejemplo:
    - `0001010100b` → Es de paridad IMPAR
    - `1`0001010100b → Es de paridad PAR
  - Se los utiliza para detectar errores de transmisión de datos (una cantidad par o impar de errores)
    - Si transmito una combinación de paridad PAR y recibo una de paridad IMPAR, puedo asegurar que hay **error**
    - Si transmito PAR y recibo PAR, no puedo asegurar nada, pero… la tasa de error en más de 1 bit es muy baja!!

Decimal N°	Códigos de paridad Código Z	Código W
0	0 0 0 0 0	0 0 0 0 1
1	0 0 0 1 1	0 0 0 1 0
2	0 0 1 0 1	0 0 1 0 0
3	0 0 1 1 0	0 0 1 1 1
4	0 1 0 0 1	0 1 0 0 0
5	0 1 0 1 0	0 1 0 1 1
6	0 1 1 0 0	0 1 1 0 1
7	0 1 1 1 1	0 1 1 1 0
8	1 0 0 0 1	1 0 0 0 0
9	1 0 0 1 0	1 0 0 1 1

Códigos de paridad

Código de paridad par

P	C	B	A
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1



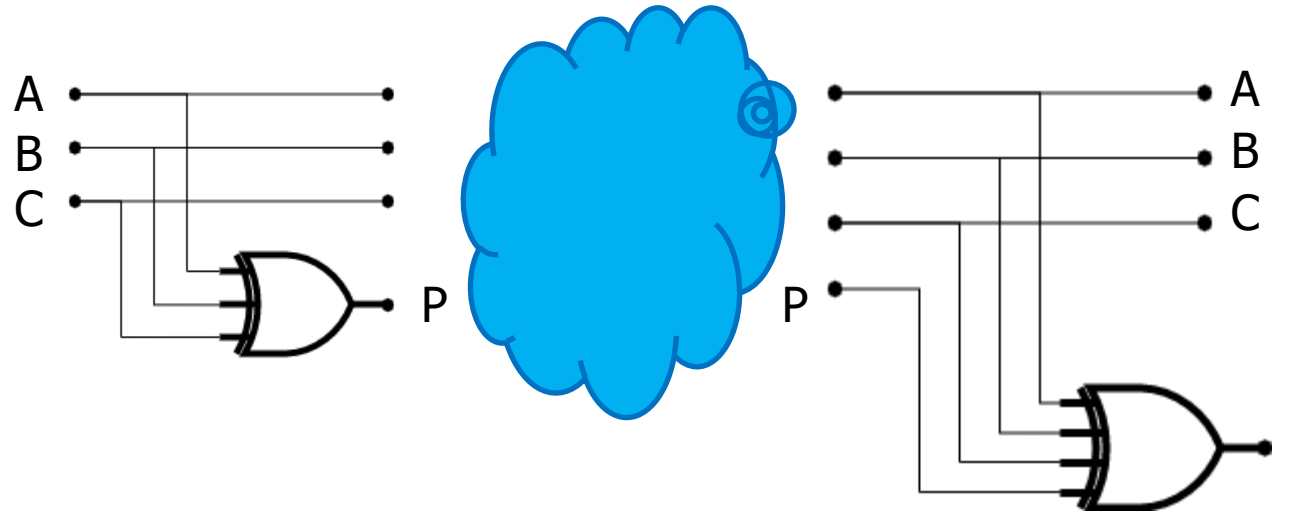
Bit de paridad = A xor B xor C
ERROR = A xor B xor C xor P

300 bits → 400 bits → Overhead 33%
Tasa de 1ue 2 bits fallen simultáneamente es muuuuy baja

Códigos de paridad

Código de paridad par

P	C	B	A
0	0	0	0
1	0	0	1
1	0	1	0
0	0	1	1
1	1	0	0
0	1	0	1
0	1	1	0
1	1	1	1



C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

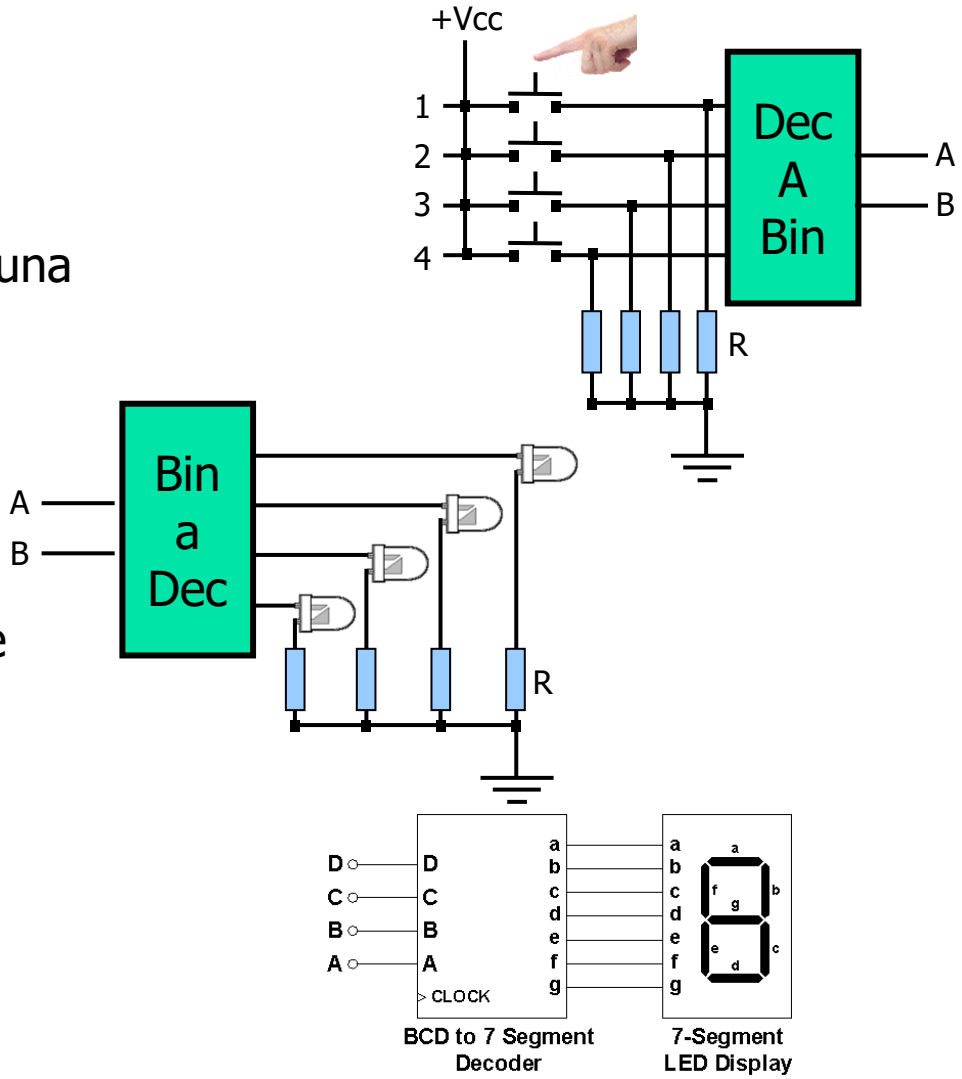
Código BCD Natural

- Representa los dígitos decimales del 0 al 9 con una combinación de 4 bits para cada uno de ellos
- Ejemplo
  - 348_D
    - = $0011|0100|1000_{BCD}$
- Los números decimales no se representan por su correspondiente combinación binaria
- Ejemplo
  - 348_D
    - = 101011100_B

Decimal	BCD Natural			
N°	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

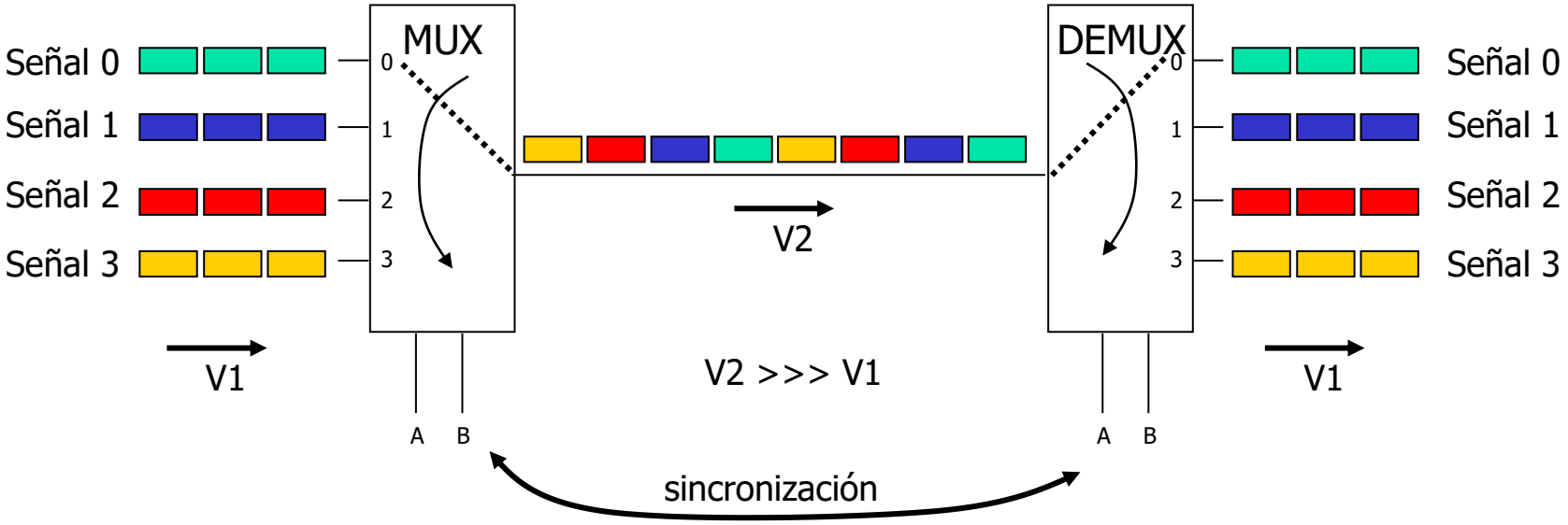
Transcodificadores

- Codificadores
  - Entrada: no codificada
  - Salida: código
  - Ejemplo: Sensores de una alarma
- Decodificadores
  - Entrada: código
  - Salida: no codificada
  - Ejemplo: Encendido de luces a distancia
- Conversores de código
  - Entrada: código "A"
  - Salida: código "B"
  - Ejemplo: BCD a "7 segmentos"



Multiplexores y demultiplexores

- Multiplexar (en el tiempo)
  - Compartir un único canal transmitiendo más de una señal en forma "simultánea"
  - Es una llave rotativa controlada por un código binario
- Demultiplexar
  - Operación inversa a la multiplexación



Funciones lógicas

$$Z = \bar{c}.[\bar{a}.\bar{b} + \bar{a}.b + a.b]$$

c	b	a	z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- Con decodificadores