Sistemas Digitais

Dep.Armas e Electrónica- Escola Naval

V.1.9 V.Lobo 2013

SISTEMAS DE NUMERAÇÃO

Sistemas Digitais

- **DECIMAL**
  - SÍMBOLOS 0,1 .. 9
  - 1842 ⇒ $1x10^3 + 8x10^2 + 4x10^1 + 2x10^0$
- **OCTAL**
  - SÍMBOLOS 0..7
  - 1634 ⇒ $1x8^3 + 6x8^2 + 3x8^1 + 4x8^0$
- **HEXADECIMAL**
  - SÍMBOLOS 0.. 9,A,B,C,D,E,F
  - 5F1A0 ⇒ $5x16^4 + 15x16^3 + 1x16^2 + 10x16^1 + 0x16^0$
- **BINÁRIO**
  - SÍMBOLOS 0,1
  - 10110 ⇒ $1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0$

A **POSIÇÃO** é que dá importância ou **PESO** ao dígito.
O dígito **MAIS SIGNIFICATIVO** é o que está mais à esquerda **(MSB)**
O dígito **MENOS SIGNIFICATIVO** é o que está mais à direita **(LSB)**

1

SISTEMA BINÁRIO

Sistemas Digitais

- **IMPORTÂNCIA DO SISTEMA BINÁRIO**
  - Fácil implementação física
  - Implementação com sistemas hidráulicos, eléctricos, luminosos, etc.
- **CONVERSÕES:**
  - DECIMAL → BINÁRIO
  - BINÁRIO → DECIMAL

$26_d = 11010_b$

26_d | 2
0 ← 13 | 2
1 ← 6 | 2
0 ← 3 | 2
1 ← 1 → MSB

LSB

$1\,0\,1\,0\,0\,1\,1\,0_b = 166_d$

1	0	1	0	0	1	1	0
128	64	32	16	8	4	2	1

+ + + → 128+32+4+2=166

2

BASES POTÊNCIAS DE 2

Sistemas Digitais

- **As bases que são potências de 2 são facilmente convertidas em binário e vice-versa**
  - Octal — 1 dígito octal = 3 dígitos binários
  - Hexadecimal — 1 dígito hexa = 4 dígitos binários

$$\overbrace{10}^{2}\,\overbrace{1101}^{D}\,\overbrace{0011}^{3}$$

1 0 1 1 0 1 0 0 1 1

$$\underbrace{1}_{1}\,\underbrace{011}_{3}\,\underbrace{010}_{2}\,\underbrace{011}_{3}$$

$2D3_H = 1011010011_b = 1323_{Oct.}$

- **Vantagens**
  - Usam menos dígitos para representar um dado número
  - São mais facilmente entendidas por humanos
  - São muito usadas

3

Aritmética binária

Sistemas Digitais

- **Basicamente as mesmas regras que a aritmética decimal !**
  - Somam-se os números dígito a dígito
  - De um dígito para o seguinte (mais significativo), pode "ir um", ou seja pode haver "CARRY"
  - 1 e 1 são dois (ou seja 10_b)
  - Exemplo:

Adição ⟶

```
   (11011)₂        (647)₁₀
 + (10011)₂      + (537)₁₀
  (101110)₂       (1184)₁₀
```

Multiplicação ⟶

```
    1101          152
  ×  101        × 231
    1101          152
   0000          456
  1101          304
  1000001       35012
```

São apenas deslocamentos e somas !!!

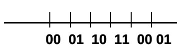
4

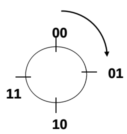
Aritmética binária

Sistemas Digitais

- **Numa máquina, o número de dígitos é FINITO**
  - Não posso usar todos os dígitos que quiser
  - Há um número MÁXIMO que se pode representar:

00
01
10
11
00
01





- **Consequência da representação com um número FINITO de dígitos**
  - Os números não são representados por uma recta, mas sim por uma circunferência !

5

REPRESENTAÇÃO DE NÚMEROS NEGATIVOS

Sistemas Digitais

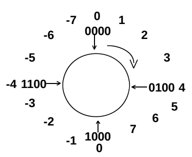
- **Problema:**
  - Como indicar que um número é negativo, sem usar o símbolo "-" (usando apenas 0 e 1)
  - Solução: usar uma das posições para representar o sinal
- **SINAL E MÓDULO** (signed integer)
  - O bit mais significativo representa o sinal, e os restantes a magnitude
  - Sinal = 0 => Positivo (representação normal)
  - Sinal = 1 => Negativo
  - Exemplos:

0100 = 4
1100 = -4
0010 = 2
1011 = -3

0 1 1 0

MAGNITUDE

SINAL



6

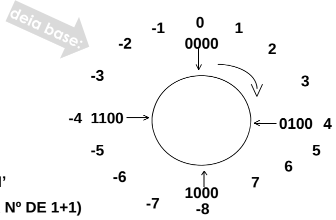
Sistemas Digitais

Dep.Armas e Electrónica- Escola Naval
V.1.9 V.Lobo 2013

COMPLEMENTO PARA 2

Sistemas Digitais

- **Ideia Base**
  - Facilitar somas e subtrações



$N^{(2)} = M - N$

- NÚMERO 'N'
- MÓDULO (MÁX Nº DE 1+1)
- NÚMERO 'N' EM COMPLEMENTO PARA 2

Regra para fazer as conversões:
COMPLEMENTAR TODOS OS DÍGITOS A PARTIR DO 1º '1'

```
 100000
-00110
 111010
```

00110 (6_{10}) $\Rightarrow$ 11010 (-6_{10})

7

COMPLEMENTO PARA 2

Sistemas Digitais

- **Complemento para 2**
  - Usa o bit mais significativo para representar o sinal (tal como anteriormente)
  - Os restantes bits são calculados de acordo com o algoritmo apresentado
  - Vantagens
    - →Permite ver rapidamente se um número é positivo ou negativo
    - →Não existem números repetidos (com 2 representações)
    - →O número -1 está imediatamente antes do 0
    - →As operações se doma e subtracção podem ser feitas usando os algoritmos usuais
- **Algoritmos para a conversão positivo/negativo em complemento p/2**
    - →Subtrair o número positivo ao número 1000.... (2^N)
    - →Começar do lado direito, e deixar na mesma todos os dígitos até ao primeiro 1 (inclusive). Complementar todos os dígitos a partir desse ponto.

8

Números não-inteiros

Sistemas Digitais

- **Representação em vírgula fixa**
  - Igual a sinal e módulo
  - Número pré-determinado de dígitos são para a parte fraccionária
  - Exemplo:
    - →Pré-fixado que há dois dígitos binários para a parte fraccionária
    - →2.5 = 1010
- **Coeficientes da parte fráccionária**
  - Potências negativas da base
  - 2^{-1} (=0.5) , 2^{-2} (=0.25), ...
- **Representação em vírgula flutuante**

9

Representação em vírgula flutuante

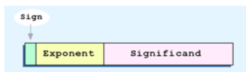
Sistemas Digitais

- **Permitem uma maior GAMA DE VARIAÇÃO**
- **Têm menos precisão que vírgula fixa**
- **Representação:**

$$+\,0.43 \times 10^{+6} = 430000$$

Expoente (c/sinal)
Base
Sinal
Mantissa

- **Norma IEEE-754 (single precision floating point)**
  - 32 bits – 1 bit de sinal, 8 bits de expoente, 23 bits de mantissa



10

CÓDIGOS BINÁRIOS - numéricos

Sistemas Digitais

- **Para representação de números, sem ser em binário natural**
  - Para simplificar as convenções binário / decimal
  - BCD - Binary coded decimal (natural, ou 8421)
    - →Usam-se 4 dígitos binários para cada dígito decimal
    - →Perdem-se 6 posições em cada 16
  - Aiken (ou 2421)
    - →Os bits têm peso 2421
    - →Os números desperdiçados são os "do meio"
    - →Permite destinguir facilmente os números maiores que 5
    - →É autocomplementar
  - Excesso 3 (não ponderado)
    - →Usa os 10 números "do meio" - 3 a 13
    - →É autocomplementar
  - 7421 - Minimiza o consumo

11

CÓDIGOS BINÁRIOS - numéricos

Sistemas Digitais

Dec.	BCD	AIKEN	EXC.3	7421
0	0000	0000	0011	0000
1	0001	0001	0100	0001
2	0010	0010	0101	0010
3	0011	0011	0110	0011
4	0100	0100	0111	0100
5	0101	1011	1000	0101
6	0110	1100	1001	0110
7	0111	1101	1010	1000
8	1000	1110	1011	1001
9	1001	1111	1100	1010

Gray
0000
0001
0011
0010
0110
0111
0101
... ...
... ...

- **Código Gray (binário reflectido)**
  - Serve para minimizar transições
  - Pode resolver problemas de estados transitórios nas mudanças
    - →Conversores físicos
  - É um código cíclico
  - Fácil passagem para binário

12

Sistemas Digitais

Dep.Armas e Electrónica- Escola Naval

V.1.9 V.Lobo 2013

Representar o mundo com 0's e 1's

- **Consigo representar números**
  - Binário natural, complemento para 2, vírgla fixa, vírgula flutuante
- **Com números posso representar:**
  - TEXTO
  - IMAGEM
  - SOM
  - Outros
    - →Vibração
    - →Cheiro
    - →....o que quiser....

- **Para trocar informação**
  - Tem que haver NORMAS para interpretar os 0s e 1s
  - Formato PDF, DOCX, JPG, XLS, MP3, etc,etc

CÓDIGOS BINÁRIOS - Alfanuméricos

- **Para representação de caracteres**
  - Código ASCII
    - →*A*merican *S*tandard *C*ode for *I*nformation *I*nterchange
    - →Define caracteres normais, símbolos, e caracteres de controlo.
    - →Extensões para 8 bits para caracteres especiais
  - Código ebcdic (usado apenas na IBM)
  - Unicode (16 bits, extensão do ASCII que inclui caracteres orientais)

0		16		32		48	0	64	@	80	P	96	`	112	p
1		17		33	!	49	1	65	A	81	Q	97	a	113	q
2		18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3		19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4		20	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5		21		37	%	53	5	69	E	85	U	101	e	117	u
6		22		38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23		39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24		40	(	56	8	72	H	88	X	104	h	120	x
9		25		41	)	57	9	73	I	89	Y	105	i	121	y
10	LF	26		42	*	58	:	74	J	90	Z	106	j	122	z
11		27	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	FF	28		44	,	60	<	76	L	92	\	108	l	124	|
13	CR	29		45	-	61	=	77	M	93	]	109	m	125	}
14	SO	30		46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31		47	/	63	?	79	O	95	_	111	o	127	

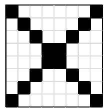
Imagens

- **Ideia geral**
  - Dividir a imagem em "quadradrinhos", ou "Picture Elements"
    - →PIXEL
  - Cada PIXEL pode ocupar 1 bit (0,1) ou vários, para ter diferentes cores, intensidades, etc
- **Formatos raster**
  - BMP (Windows Bitmap)
    - →24 bits (3 Bytes), equivalente a R,G,B, por pixel.
    - →Não tem compressão
  - TIFF (Tagged Image File Format)
    - →24 ou 32 bits
    - →Usa compressão sem perdas (LZW)
  - JPEG (Joint Photographic Experts Group)
    - →Compressão com perdas, RFC 1341
  - (RAW) – Sem cabeçalho, formatação, ou compressão
- **Outros formatos**
  - Formato Vectorial (p.ex. PCX)
  - Outros (GIF, PNG, CGM, SVG, JPG(2000), TGA, PDF, CDR, EPS, ODG, WMF, XPS, VML, XPS, DXF, PIC)

Exemplo de um formato de imagem

- **Consideremos a seguinte imagem:**
- **A codificação (raw) seria:**

```
1 0 0 0 0 0 0 1
0 1 0 0 0 0 1 0
0 0 1 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
1 0 0 0 0 0 0 1
```



- **Representado de outra maneira**
  - Em bits:
    - →1000000101000010001001000001100000011000001001000100001010000001
  - Em números de 8 bits representados em:
    - →Decimal: 129,66,36,24,24,36,66,129
    - →Hexadecimal: 81,42,24,18,18,24,42,81
  - Com cabeçalho: (3 bytes para profundidade de cor, 2 para largura, 2 para altura: 00 00 01 00 08 00 08 81 42 24 18 18 24 43 81.

Sons

- **Ideia geral**
  - O som é um sinal que representa variações de pressão
  - Em cada instante verifica-se qual o valore (relativo) da pressão
    - →Quantificação do sinal determina erro quantificação
  - Periodicamente vai-se repetir o processo
    - →Frequencia de amostragem determina a gama de frequências que se consegue observar.
- **Formatos de som**
  - WAV
    - →Guarda o sinal de pressão
  - MP3
    - →Cumprime a informação
    - →Calcula a transformada de fourier, e guarda apenas os coeficientes mais importantes

ERROS

- **O que é um erro**
  - É um 1 passar a 0, ou vice-versa
- **Erros de transmissão**
- **Degradação do meio magnético**
- **Soluções**
  - Mandar informar redundante para confirmação
  - Utilização de BITS DE PARIDADE
    - →1 bit permite detectar se houve um número impar de erros
    - →Paridade Par, Ímpar, Mark, e Space
    - →Paridade byte a byte, e paridade vertical
  - Utilização de códigos correctores
    - →Códigos de Hamming 5/3
  - Utilização de checksums

ALGEBRA DE BOOLE

Regras matemáticas para manipular os 0's e 1's com que representamos o mundo

Álgebra DE BOOLE

- **Definição FORMAL**

$\left\{ U , + , . \right\}$

U = Conjunto finito

+,· = Operações (soma , produto)

1 → $a + b \in U$; $a . b \in U$

2 → $a + b = b + a$; $a . b = b . a$

3 → $a + 0 = a$; $a . 1 = a$

4 → $a (b + c) = a b + a c$; $a + b c = (a + b) (a + c)$

5 → $a + X = 1$; $a . X = 0$

$X \equiv \overline{a}$ (complemento)

UTILIDADE EM SISTEMAS LÓGICOS

- **Consideramos U = {0,1}**
  - o conjunto U é apenas os 2 valores binários
  - podemos implementar facilmente este tipo sistemas com: lâmpadas, relés, transístores, actuadores mecânicos e hidráulicos, etc.
- **Operação adição**
  - Corresponde ao OU lógico
- **Operação de multiplicação**
  - Corresponde ao E lógico
- **Operação complemento**
  - É a simples negação

Usamos binário porque é fácil fazer máquinas que tenham 2 estados possíveis

$U = \left\{ 0 , 1 \right\}$

+ = "OR" (operação OU)

, = "AND" (operação E)

Complemento = "NOT" (operação NEGAÇÃO)

TEOREMAS

- **Vão ser as ferramentas para toda a manipulação de dados que vamos fazer...**
- **PRINCÍPIO DA DUALIDADE**
  - Se uma dada proposição é verdadeira, então, substituindo os E com OU e os 1 com 0 , obtenho também uma proposição verdadeira
- **1 - ELEMENTO ABSORVENTE**
  - $A . 0 = 0$ $\quad$ $A + 1 = 1$
- **2 - ELEMENTO NEUTRO**
  - $A . 1 = A$ $\quad$ $A + 0 = A$
- **3 - IDEMPOTÊNCIA**
  - $A . A = A$ $\quad$ $A + A = A$

TEOREMAS

- **4 - COMPLEMENTARIDADE**
  - $A . \overline{A} = 0$ $\quad$ $A + \overline{A} = 1$
- **5 - INVOLUÇÃO**
  - $A = \overline{\overline{A}}$
- **6 - COMUTATIVIDADE**
  - $A . B = B . A$ $\quad$ $A + B = B + A$
- **7 - ASSOCIATIVIDADE**
  - $A . B . C = (A . B) C = A . (B . C)$
  - $A + B + C = (A + B) + C = A + (B + C)$
- **8 - LEIS DE DeMORGAN**

$\overline{A . B} = \overline{A} + \overline{B}$ $\quad$ $\overline{A + B} = \overline{A} . \overline{B}$

TEOREMAS

- **9 - DISTRIBUTIVIDADE**
  - $A . (B + C) = A . B + A . C$
  - $A + B C = (A + B) . (A + C)$
- **10 - ABSORÇÃO**
  - $A + A B = A$ $\quad$ $A (A + B) = A$
- **11 -**
  - $A B + A \overline{B} = A$ $\quad$ $(A + B) . (A + \overline{B}) = A$
- **12 -**
  - $A + \overline{A} B = A + B$ $\quad$ $A . (\overline{A} + B) = A . B$
- **13 - TEOREMA DO TERMO INCLUÍDO**
  - $A B + \overline{A} C + B C = A B + \overline{A} C$
  - $(A + B) (\overline{A} + C) (B + C) = (A + B) (\overline{A} + C)$

DEMONSTRAÇÕES

- **USANDO TABELAS DE VERDADE**
  - Demonstra-se para TODOS os casos possíveis.
  - Tabela de verdade das funções AND e OR

A	B	S = A.B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	S = A+B
0	0	0
0	1	1
1	0	1
1	1	1

EXEMPLO:

- **Provar que $A \cdot (\bar{A} + B) = A \cdot B$**

A	B	$\bar{A}$	$\bar{A}+B$	$A \cdot (\bar{A}+B)$	A . B
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	1	0	1	1	1

Funções de 2 variáveis

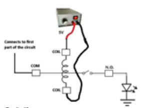
- **Quantas funções existem de 2 variáveis ?**
  - É um número finito.
  - 2 variáveis ⇒ 4 combinações de entrada ⇒ 2^4=16 funções
  - 3 delas decorrem imediatamente da definição da álgebra
    - →AND (E, .)
    - →OR (OU, +)
    - →NOT (NEG, ¯)
  - Há outras funções que são muito usadas: XOR, NAND, NOR
- **Implementação física**
  - Sistemas mecânicos (alavancas, rodas dentadas)
  - Sistemas hidráulicos (usados em certos ambientes perigosos)
  - Sistemas eléctricos (relés)
  - Sistemas electrónicos (transístores, díodos, circuitos integrados)
    - →De longe o mais eficiente, logo mais usado !

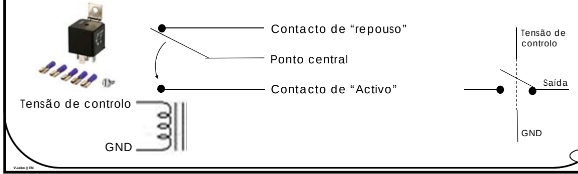
Implementação FÍSICA da Álgebra de Boole

Montar circuitos ou máquinas que façam as operações previstas na ágebra de boole (and, or, not, ...)

Implementação física para servir de exemplo

- **Objectivo:**
  - Construir um **dispositivo** (uma máquina) que consegue fazer as operações necessárias para realizar as **funções lógicas** da álgebra de Boole
- Exemplo
  - Podemos usar circuitos eléctricos, com relés.
  - São **Interruptores** controlados electricamente
  - O “0” da álgebra pode ser representado por **GND** (**0 V**)
  - O “1” da álgebra pode ser representado por V_{cc} (por ex. **5 V**)





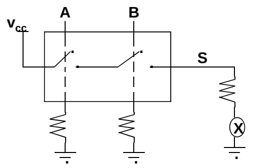
REALIZAÇÃO FÍSICA COM INTERRUPTORES

- **PORTA “AND” C/ RELÉS**

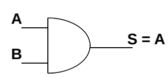
Tabela de verdade

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Implementação física



Símbolo Lógico



Sistemas Digitais

Dep.Armas e Electrónica- Escola Naval
V.1.9 V.Lobo 2013

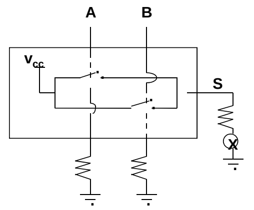
REALIZAÇÃO FÍSICA COM INTERRUPTORES

- PORTA “OU” C/ RELÉS

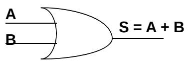
Tabela de verdade

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Implementação física



Símbolo Lógico



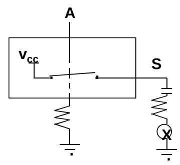
REALIZAÇÃO FÍSICA COM INTERRUPTORES

- PORTA “NOT” C/ RELÉS

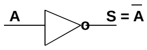
Tabela de verdade

A	$\overline{A}$
0	1
1	0

Implementação física



Símbolo Lógico



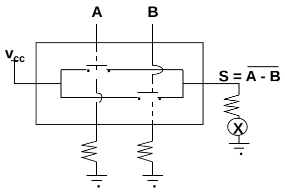
REALIZAÇÃO FÍSICA COM INTERRUPTORES

- PORTA “NAND” C/ RELÉS

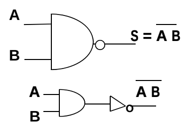
Tabela de verdade

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Implementação física



Símbolo Lógico



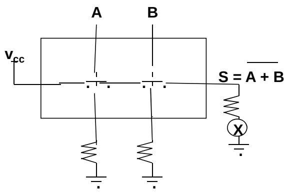
REALIZAÇÃO FÍSICA COM INTERRUPTORES

- PORTA “NOR” C/ RELÉS

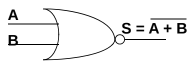
Tabela de verdade

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Implementação física



Símbolo Lógico



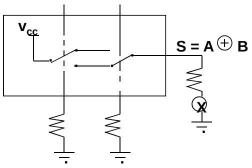
REALIZAÇÃO FÍSICA COM INTERRUPTORES

- PORTA “XOR” C/ RELÉS

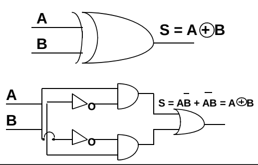
Tabela de verdade

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Implementação física

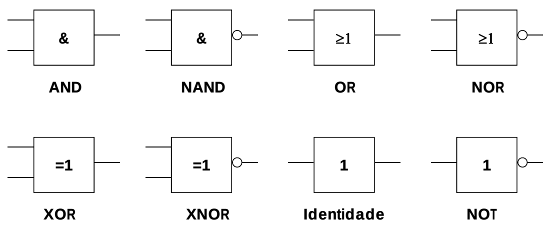


Símbolo Lógico



Outras simbologias

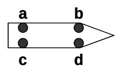
- **Norma ANSI Y.32.14**
  - Simplifica a representação das portas lógicas
  - É menos “bonita”, mas mais eficiente



Exercícios

Sistemas Digitais

- **Problema do alarme de segurança**
  - Suponha que existem dois sensores de incêndio, e uma lâmpada que deverá acender quando um deles fôr activado. Projecte o circuito que actua sobre a lâmpada.
- **Problema da segurança do navio**
  - Suponha num dado navio existem 4 pontos onde devem estar sentinelas quando o navio está fundeado: dois em cada bordo, um na alheta, outro na amura. Em cada um desses pontos está um sensor que envia um sinal 1 quando aí se encontra um sentinela, e 0 em caso contrário. Na câmara de oficiais deverão existir duas lâmpadas: uma amarela, outra vermelha. A vermelha deverá acender sempre que não há qualquer sentinela num dos bordos. A amarela deverá acender quando há apenas 2 sentinelas nos seus postos. Projecte o circuito que resolve este problema.

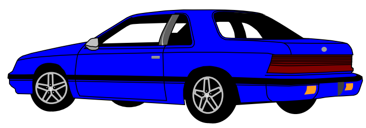


37

Exercícios

Sistemas Digitais

- **PROBLEMA DO SEMÁFORO "POR PEDIDO"**
  - Imagine que num dado local existe um estrangulamento numa estrada onde só passa um automóvel. Existem uns sensores para indicar que há um automóvel em cada lado do estrangulamento, e dois semáforos (verde/vermelho) que controlam o acesso a essa área. Se apenas houver automóveis de um dos lados, o semáforo deverá deixar passar esses automóveis. Se não houver automóveis em qualquer dos lados, os semáforos deverão estar ambos vermelhos. Caso contrário, o semáforo deverá estar verde para apenas um dos lados (à sua escolha).



38

SUFICIÊNCIA DO NAND

Sistemas Digitais

- **Quantas portas diferentes são necessárias para gerar uma função booleana ?**
  - A álgebra é definida com três operações (que por definição geram todas as funções possíveis):
    - →AND, OR, NOT
  - Se eu conseguir realizar essas funções com uma só gate, poderei gerar qualquer outra função com essa gate
- **Suficiência do NAND**
  - NOT(A) = A NAND A
  - A AND B = (A NAND B) NAND (A NAND B)
  - A OR B = (A NAND A) NAND (B NAND B)

39

REALIZAÇÕES FÍSICAS

Sistemas Digitais

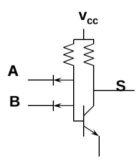
- **FAMÍLIAS LÓGICAS**
  - Permitem ligações directas entre as diversas portas lógicas
  - Exemplos: interruptores, relés, sistema mecânico e hidráulico
- **FAMÍLIAS ELECTRÓNICAS**
  - DTL ,RTL
    - →Fácil compreensão
  - ECL
    - →Muito rápida, consome bastante
  - CMOS
    - →Consumo muito baixo, tolerância a diversos níveis de tensão, grande integração (integrados da família 4000)
  - I^2L
    - →Mais uma alternativa...
  - TTL
    - →Barato, simples de usar, compromisso bastante bom de características. É a mais usada (integrados da família 74xx, 54xx)

40

DTL

Sistemas Digitais

- **Diode-Transistor Logic**
  - Usa díodos e transístores
  - Exemplo: gate NAND
  - Hipótese 1: A=0v ou B=0v
    - →Os díodos conduzem
    - →A tensão na base do transístor é aprox. =0v
    - →O transístor não conduz
    - →A resistência de saída faz de pull-up: S=5v
  - Hipótese 2: A=B=1
    - →Os díodos não conduzem
    - →A resistência de entrada faz com que a tensão na base do transístor seja aproximadamente = 5v
    - →O transístor conduz
    - →A tensão de saída é aprox. =0v

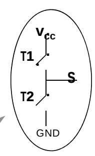


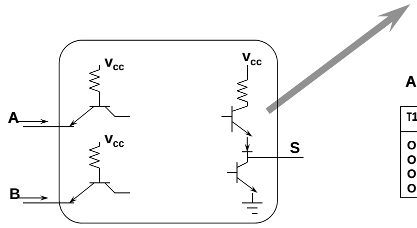
41

TTL

Sistemas Digitais

- **Transistor-Transistor Logic**
  - Transístores de junção bipolar
  - Vamos estudar apenas
    - →Andar de entrada (díodos)
    - →Andar de saída (totem-pole)





Andar de saída

T1	T2	Saida
ON	OFF	V_{cc}
OFF	ON	GND
OFF	OFF	Tri-State
ON	ON	Bumm!

42

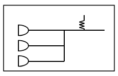
Sistemas Digitais

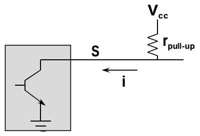
Dep.Armas e Electrónica- Escola Naval
V.1.9 V.Lobo 2013

TTL

Sistemas Digitais

- **Gates Open-Collector**
  - O andar de saída só tem um transístor (ligado à massa)
  - A gate pode forçar o valor lógico ZERO
  - Tem que haver uma resistência externa de PULL-UP para forçar o nível lógico 1
  - Posso implementar um WIRED-AND, ligando várias saídas O.C.





TRANSISTOR LIGADO
A tensão de saída é 0, e a sendo a corrente $I=V_{cc}/R_{pull-up}$

TRANSISTOR DESLIGADO
A corrente é 0, logo a tensão de saída é V_{cc}

43

CARACTERÍSTICAS

Sistemas Digitais

- **TTL**
  - FACILIDADE DE FABRICO, E DISPONIBILIDADE
  - ROBUSTEZ E FIABILIDADE
  - BAIXO CUSTO
  - CONSUMO MODERADO (LOGO DISSIPAÇÃO MODERADA)
  - FAMÍLIA 74xxxy e 54xxy
    - →54xx tem especificações militares: grande amplitude te temperaturas/humidade/vibração, distribuição optimizada dos pinos
    - →VARIAÇÕES 74S , 74LS , 74L , 74H (consumo, velocidade)
- **CMOS, NMOS e PMOS**
  - TRANSISTORES DE EFEITO DE CAMPO
  - CONSUMO MUITO BOM
  - LENTIDÃO , E PROBLEMAS C/ ESTÁTICA
  - MAIOR FLEXIBILIDADE NOS NÍVEIS DE TENSÃO
  - FAMÍLIA 40xx

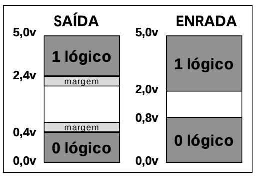
44

CARACTERÍSTICAS

Sistemas Digitais

- **FAN-OUT**
  - Nº de portas que podem ser ligadas à saída
  - Pode ser especificado em número de gates que consegue alimentar (da mesma família lógica) ou em corrente máxima de saída (em mA).
- **FAN-IN**
  - Corrente que injecta/consome na entrada
- **MARGEM DE RUÍDO**
  - Tolerância entre níveis
  - 0 lógico não é 0v

Nota:
O que é ruído ?
Quais os seus efeitos ?
Quais são as fontes de ruído ?
Como pode ser diminuido ?



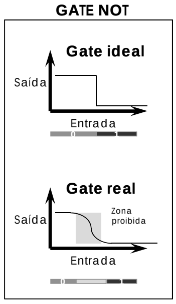
45

CARACTERÍSTICAS

Sistemas Digitais

- **FUNÇÃO DE TRANSFERÊNCIA**
  - A transição de 0 lógico para 1 lógico não é perfeita
  - Exemplo: gate NOT
- **TEMPO DE PROPAGAÇÃO**
  - Uma gate leva um certo tempo até que as saídas reflictam o estado das entradas
  - O tempo de propagação quando as saídas têm que passar de 0 para 1 é normalmente diferente de 1 para 0.
- **DISSIPAÇÃO**
  - As gates consomem corrente que provoca aquecimento
  - O aquecimento é normalmente proporcional à velocidade de processamento

GATE NOT

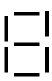


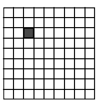
46

DISPLAYS

Sistemas Digitais

- **Quanto à tecnologia física**
  - Indicadores de descarga de gás
    - →são válvulas
  - Leds - diodos emissores de luz
    - →Baixo consumo
    - →Interface muito simples
    - →Grande variedade
  - Cristais Líquidos (lcd)
    - →Consumo muitíssimo baixo
    - →Mudanças na polorização (provocados pela aplicação de campo eléctrico) fazem com que a luz não seja reflectida
- **Quanto à disposição gráfica**
  - Displays de 7 segmentos
  - Matriz de pontos





47

FUNÇÕES BOOLEANAS

Sistemas Digitais

- **S = F (A) S = F (A,B,C, ...)**
  - Onde A,B,C... podem assumir os valores 0 e 1
- **PARA UM DADO NÚMERO DE VARIÁVEIS, O Nª DE FUNÇÕES POSSÍVEIS É LIMITADO**
  - Exemplo: FUNÇÕES DE 1 VARIÁVEL:

FUNÇÃO	ENTRADAS 0	1	DESIGNAÇÃO	EXPRESSÃO DE BOOLE
S0	0	0	Zero	0
S1	0	1	Igualdade	A
S2	1	0	Negação	!A (ou $\bar{A}$)
S3	1	1	Identidade	1

48

FUNÇÃO DUAL E COMPLEMENTO

- **FUNÇÃO DUAL**
  - G é função dual de F sse G(A)= (F(A*))*
  - (X* é o dual de X se em X trocar 1 por 0, + por . , e vice-versa)
  - Exemplos
    - →O dual da função AND é a função OR
    - →O dual da negação é a própria negação
- **FUNÇÃO COMPLEMENTO**
  - G é função complemento de F sse G(A)=!F(A)
    - →O complemento da função AND é a função NAND
    - →O complemento da negação é a igualdade

FUNÇÕES DE 2 VARIÁVEIS

FUNÇÃO	ENTRADAS 00 ,01,10,11	DESIGNAÇÃO	EXPRESSÃO DE BOOLE	NOTAÇÃO	DUAL	COMPLE MENTO
S0	0 0 0 0	Zero	0		15	15
S1	0 0 0 1	And	A.B	A.B	7	14
S2	0 0 1 0	Inibição ou Nix	A.B*		11	13
S3	0 0 1 1	Iguldade	A		3	12
S4	0 1 0 0	Inibição ou Nix	A*.B		13	11
S5	0 1 0 1	Igualdade	B		5	10
S6	0 1 1 0	Or Exclusivo ou Dilema	A*.B+A.B*	A ⊕ B	9	9
S7	0 1 1 1	Or (Inclusivo)	A+B	A+B	1	8
S8	1 0 0 0	Nor ou Função Dagger	(A+B)*	$\overline{A+B}$	14	7
S9	1 0 0 1	Equivalência	A.B+A*.B*	A ≡ B	6	6
S10	1 0 1 0	Not (Negação)	B*		10*	5
S11	1 0 1 1	Implicação Material	A+B*	B ⇒ A	2	4
S12	1 1 0 0	Not (Negação)	A*		12	3
S13	1 1 0 1	Implicação Material	A*+B	A ⇒ B	4	2
S14	1 1 1 0	Nand ou Função Stroke	(A.B)*	$\overline{A\ .\ B}$	8	1
S15	1 1 1 1	Unidade ou Identidade	1		0	0

FORMAS CANÓNICAS

- **Como identificar de forma unívoca e normalizada uma dada função?**
  - Expressões analíticas podem ter várias formas
  - Tabelas de verdade são muito extensas
  - Formas canónicas: são a solução ideal
    - →A tabela de verdade tem na coluna de resultados 0 ou 1
    - →Posso identificar a função dizendo que entradas da tablea de verdade são 1 (ou 0)
    - →A tabela de verdade tem que ter as entradas por uma determinada ordem

$A.\bar{B}+\bar{A}.B$=A.!B+!A.B=A⊕B

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

As linhas 1 e 2 têm 1

As entradas estão por ordem (0,1,2,3)

$F_{1,2}$

FORMAS CANÓNICAS

- **Como identificar as linhas da tabela de verdade ?**
  - Cada linha corresponde a um produto de todas as variáveis
- **MINTERMOS**
  - Produtos que englobam todas as variáveis independentes
  - Correspondem às linhas da tabela de verdade, se esta for escrita de modo a que as variáveis formem o código binário
  - São numeradas, atribuindo 0 às variáveis negadas , e 1 às afirmadas
- **MAXTERMOS**
  - Somatórios que englobam todas as variáveis independentes
  - Podem-se obter a partir dos mintermos, e vice-versa
  - $M_i=m_{2^n-1-i}$

FORMAS CANÓNICAS

- **1ª. FORMA CANÓNICA**
  - Soma de mintermos
  - Exemplo: função XOR
    - →XOR(A,B) = A.!B+!A.B = m_1+m_2 = Σ (1,2)
  - Problemas
    - →Qual a tabela de verdade da função de 3 variáveis Σ (0,5,7) ?
    - →Qual a 1ª forma canónica da função OR de 3 variáveis
- **2ª. FORMA CANÓNICA**
  - Produto de maxtermos
  - Exemplo: função XOR
    - →XOR(A,B) = (!A+!B).(A+B) = M_0 . M_3 = Π (0,3)
  - Problemas
    - →Qual a tabela de verdade da função de 3 variáveis Π (0,5,7) ?
    - →Qual a 2ª forma canónica da função OR de 3 variáveis

RESOLUÇÃO DE PROBLEMAS

1) OBTENÇÃO DE UMA FUNÇÃO QUE RESOLVA O PROBLEMA POSTO
- Métodos analíticos
- Especificar o problema numa tabela de verdade
  - →Obter os mintermos

2) SIMPLIFICAR A EXPRESSÃO
- Métodos analíticos
- Mapas de Karnaugh

3) IMPLEMENTAR O CIRCUITO
- Escolher os integrados que implementam as gates
  - →Pode ser necessário alterar a função obtida em 2 para minimizr o número de integrados usado
- Desenhar o logigrama (com pinout) do circuito

Exemplo

- **Passo 1 para o problema dos vigias do navio:**
  - Método analítico: L= a.b.!c.!d + a.!b.!c.!d + !a.b.!c.d + ...
  - Tabela de verdade:

Mintermos:
0,1,2,3,4,6,8,12

A(a,b,c,d) = Σ (0,1,2,3,4,6,8,12)

A(a,b,c,d) = !a.!b.!c.!d + !a....

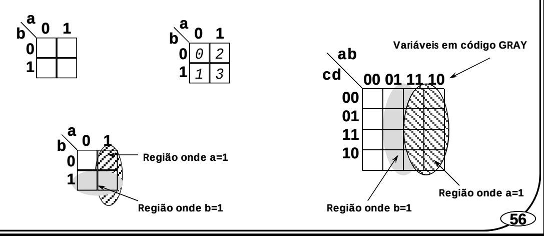
a b c d	A
0 0 0 0	1
0 0 0 1	1
0 0 1 0	1
0 0 1 1	1
0 1 0 0	1
0 1 0 1	0
0 1 1 0	1
0 1 1 1	0
1 0 0 0	1
1 0 0 1	0
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	0
1 1 1 0	0
1 1 1 1	0

$\bar{a}.\bar{b}.\bar{c}.\bar{d}$

$a.b.\bar{c}.\bar{d}$

MAPAS DE KARNAUGH

- **Um mapa de karnaugh é um modo de escrever a tabela de verdade**
- **Cada quadrícula tem apenas 1 bit diferente dos vizinhos (distância de Hamming=1)**



MAPAS DE KARNAUGH

- **Método gráfico, baseado nos diagramas de Venn, que permite detectar adjacências**
  - 1) Escrever o mapa usando código reflectido, de modo a que 2 quadrículas contíguas diferem em apenas 1 byte.
  - 2) Cada quadrado corresponde a uma linha da tabela de verdade => corresponde a um mintermo da expressão se for 1
  - 3) Como na tabela 2 quadrados contíguos diferem apenas numa das variáveis, podemos escrevê-los como $\Pi x_i y$ e $\Pi x_i !y$
  - 4) Se dois quadrados contíguos forem 1, podemos representá-los como $\Pi x_i y + \Pi x_i !y = \Pi x_i(!y+y) = \Pi x_i$, de onde se conclui que podemos ignorar a variável que troca de valor
- **REGRA:**
  - 1) Formar quadrados ou rectângulos com 2^m quadrículas
  - 2) Pôr na expressão só as variáveis que se mantêm constantes

MAPAS DE KARNAUGH

- **Os grupos resultantes da junção de mintermos chamam-se IMPLICANTES**
  - Implicante PRIMO
    - →Implicante que não pode ser mais alargado
  - Implicante ESSENCIAL
    - →Implicante que seja o único (dos primos) que "cobre" um dado mintermo

Problemas:

1. Vigias
2. Descodificador de 7 Segmentos para BCD
3. Semáforos
4. Segurança para as portas da cidadela

	00	01	11	10
00				
01			1	1
11			1	1
10	1	1	1	1

Implicante Não Primo

Implicante Primo Não Essencial

Implicantes Primo Essenciais

MAPAS DE KARNAUGH

- **Indeterminações**
  - Correspondem a casos onde "tanto faz" que a resposta seja 1 ou 0 (pode por exemplo ser uma combinação de entrada que nunca ocorre
  - Representam-se nos mapas de Karnaugh por X
  - Podemos simplificar os X como 1 ou como 0, conforme nos dê mais jeito
  - Exemplo: descodificador de 7 segmentos BCD (traço do meio)

	00	01	11	10
00	0	1	x	1
01	0	1	x	1
11	1	0	x	x
10	1	1	x	x

Alguns X são interpretados como 1 outros como 0