Revisão Eletrônica Digital

Prof. Roberto de Matos

roberto.matos@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Câmpus
São José

Objetivo

Objetivo

- Relembrar conceitos de eletrônica digital.
- Relembrar blocos de construção digital (combinacional e sequencial).
- Fazer uma implementação utilizando os blocos de construção digital.

Sistemas de Numeração

Decimal

- Base 10: Possibilidade de variar 10 algarismos diferentes $(0, ..., 9)$ em cada posição.
- Da direita para a esquerda, os pesos de cada coluna são $1, 10, 100, 1000$, ...
- $9742_{10} = 9 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$

Binário

- Base 2: Possibilidade de variar 2 algarismos diferentes $(0, 1)$ em cada posição.
- Da direita para a esquerda, os pesos de cada coluna são 1, 2, 4, 8, 16, ...
- $10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$
- Para representar a mesma quantidade de informações, precisamos usar mais posições (bits) em binário.

Binário

- Base 2: Possibilidade de variar 2 algarismos diferentes (0, 1) em cada posição.
- Da direita para a esquerda, os pesos de cada coluna são 1, 2, 4, 8, 16, ...
- $10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$
- Para representar a mesma quantidade de informações, precisamos usar mais posições (bits) em binário.

Dicas:

- Um número binário de N bits pode representar 2^N possibilidades, ou seja, $0, 1, 2, 3, ..., 2^N - 1$.
- Para saber exatamente o número de bits (N) necessários para representar um determinado valor decimal (D) use a seguinte fórmula: $N = \lfloor log_2(D) \rfloor + 1$

Hexadecimal

- Base 16: Possibilidade de variar 16 algarismos $(0, ..., 9, A, ..., F)$ diferentes em cada posição.
- Da direita para a esquerda, os pesos de cada coluna são $1, 16, 256, 4096, 65.536, ...$
- $2ED_{16} = 2 \times 16^2 + E \times 16^1 + D \times 16^0 = 749_{10}$, sendo $E = 14_{10}$ e $D = 13_{10}$

Hexadecimal

- Base 16: Possibilidade de variar 16 algarismos $(0, ..., 9, A, ..., F)$ diferentes em cada posição.
- Da direita para a esquerda, os pesos de cada coluna são $1, 16, 256, 4096, 65.536$, ...
- $2ED_{16} = 2 \times 16^2 + E \times 16^1 + D \times 16^0 = 749_{10}$, sendo $E = 14_{10}$ e $D = 13_{10}$

Dicas:

- Facilita a escrita de binários longos.
- 4ª potência da base 2 ($2^4 = 16$), ou seja, cada hexadecimal representa 4 bits.
- Colocar $0x$ na frente de um hexadecimal é uma notação muito utilizada (ex.: $0x2ED$).

Representação de números negativos

Código complemento de dois

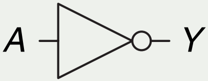
- Esta é a opção adotada para representar números negativos em praticamento todos os computadores e outros sistema digitais.
- A representação binária de um número negativo é obtida da seguinte forma:
  1. Converta a sua representação positiva (magnitude) para binário
  2. Inverta todos os bits (complemento de um)
  3. Some um (1)
- Exemplo: Converter -7_{10} considerando uma representação de 5 bits.

Código complemento de dois

- Esta é a opção adotada para representar números negativos em praticamento todos os computadores e outros sistema digitais.
- A representação binária de um número negativo é obtida da seguinte forma:
  1. Converta a sua representação positiva (magnitude) para binário
  2. Inverta todos os bits (complemento de um)
  3. Some um (1)
- Exemplo: Converter -7_{10} considerando uma representação de 5 bits.
  1. $+7_{10} = 00111_2$
  2. 11000_2
  3. $11000_2 + 1 = 11001_2$
- Assim, $-7_{10} = 11001_2$

Portas lógicas

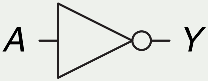
NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

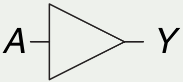
NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

BUF

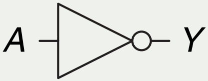


$$Y = A$$

A	Y
0	0
1	1

Portas lógicas

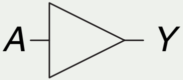
NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

BUF



$$Y = A$$

A	Y
0	0
1	1

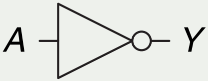
AND

$$Y = AB$$

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

Portas lógicas

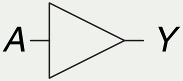
NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

BUF



$$Y = A$$

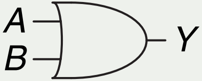
A	Y
0	0
1	1

AND

$$Y = AB$$

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

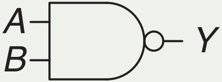
OR



$$Y = A + B$$

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

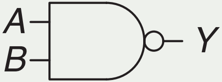
NAND



$$Y = \overline{AB}$$

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

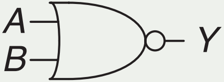
NAND



$$Y = \overline{AB}$$

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

NOR

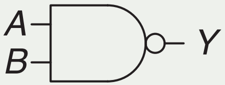


$$Y = \overline{A + B}$$

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

Portas lógicas

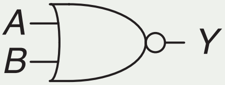
NAND



$$Y = \overline{AB}$$

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

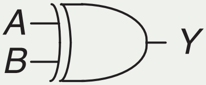
NOR



$$Y = \overline{A + B}$$

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

XOR

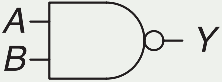


$$Y = A \oplus B$$

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

Portas lógicas

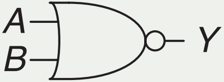
NAND



$$Y = \overline{AB}$$

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

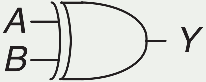
NOR



$$Y = \overline{A + B}$$

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

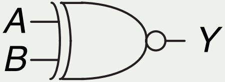
XOR



$$Y = A \oplus B$$

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

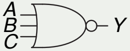
XNOR



$$Y = \overline{A \oplus B}$$

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

NOR3

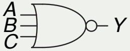


$$Y = \overline{A + B + C}$$

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

Portas lógicas

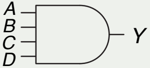
NOR3



$$Y = \overline{A + B + C}$$

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

AND4

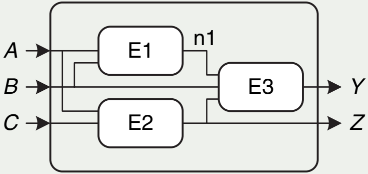


$$Y = ABCD$$

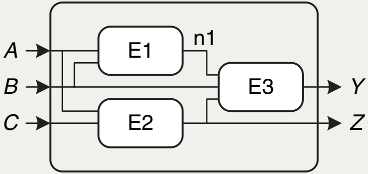
A	C	B	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Lógica Combinacional

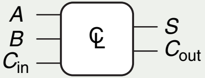
- Saída tem relação só com entradas:



- Saída tem relação só com entradas:

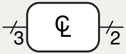


- Somador Completo:



$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

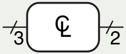
- Vários sinais agrupados para simplificar o circuito.
- Exemplo:



(a)

(b)

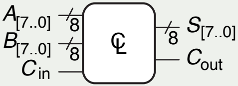
- Vários sinais agrupados para simplificar o circuito.
- Exemplo:



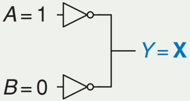
(a)

(b)

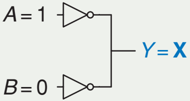
- Somador 8 bits:



- Valor ***X*** pode significar:
  - Circuito: Valor ilegal ou desconhecido

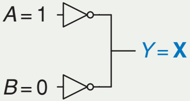


- Valor ***X*** pode significar:
  - Circuito: Valor ilegal ou desconhecido



  - Simulador: Valor não inicializado

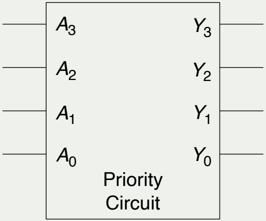
- Valor ***X*** pode significar:
  - Circuito: Valor ilegal ou desconhecido



  - Simulador: Valor não inicializado
  - Tabela Verdade: Não importa (*don't care*)

Exemplo do uso do ***X***

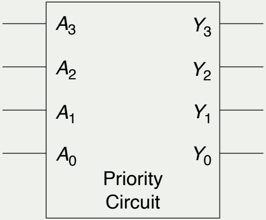
- Tabela completa:



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Exemplo do uso do ***X***

- Tabela completa:



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

- Tabela Simplificada:

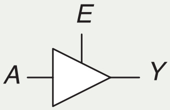
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

Além de 0's e 1's

- Valor ***Z*** significa que o nodo do circuito está flutuando ou alta impedância.
- Associado a um terceiro estado (*tristate*).

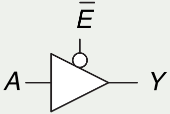
Além de 0’s e 1’s

- Valor ***Z*** significa que o nodo do circuito está flutuando ou alta impedância.
- Associado a um terceiro estado (*tristate*).
- Buffer *tristate*:



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Ativo alto



$\overline{E}$	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

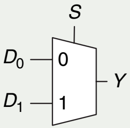
Ativo baixo

Blocos de Construção Combinacional

Multiplexador

- Roteamento em circuitos.
- Implementação de lógica.

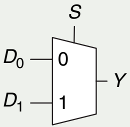
- Roteamento em circuitos.
- Implementação de lógica.



S	D_1	D_0	Y
0	0	0	0
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	1

Multiplexador

- Roteamento em circuitos.
- Implementação de lógica.



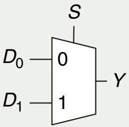
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

➔

S	Y
0	D_0
1	D_1

Multiplexador

- Roteamento em circuitos.
- Implementação de lógica.

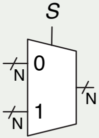


S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

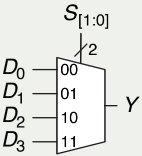
➔

S	Y
0	D_0
1	D_1

- Barramentos :



- Multiplexador 4x1 :

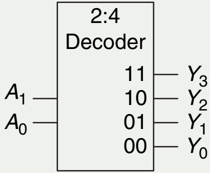


Dica:

Um multiplexador $N:1$ precisar de $log_2 N$ linhas de seleção.

Decodificador

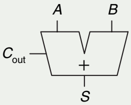
- Decodifica um padrão para outro. Exemplos:
  - BCD $\rightarrow$ Display 7 segmentos;
  - Código de operação $\rightarrow$ sinais de controle;
  - Binário $\rightarrow$ *one hot*. N entradas para 2^N saídas.



A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Somadores

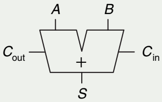
Half Adder



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$
$$C_{out} = AB$$

Full Adder

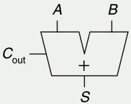


C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Somadores

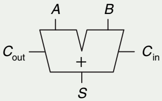
Half Adder



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

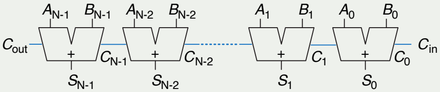
$$S = A \oplus B$$
$$C_{out} = AB$$

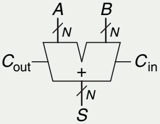
Full Adder



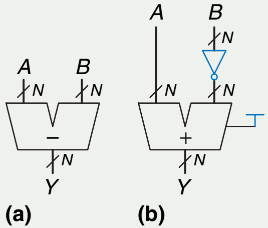
C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$





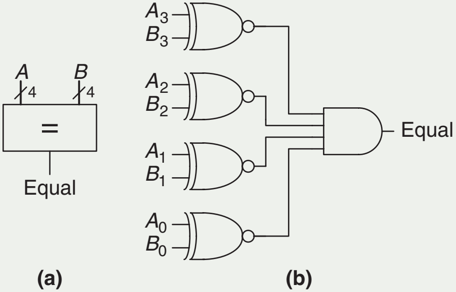
Subtrator



(a): Símbolo

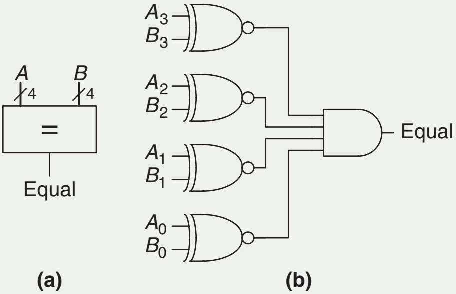
(b): Implementação ($Y = A + \overline{B} + 1 = A - B$)

- Igualdade:

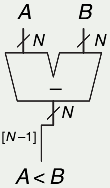


Comparadores

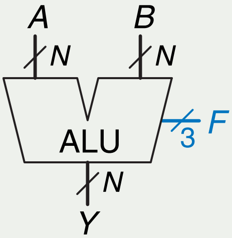
- Igualdade:



- Menor que:



Unidade Lógica Aritmética - ULA



$F_{2:0}$	Operação
000	A AND B
001	A OR B
010	A + B
011	not used
100	A AND $\overline{B}$
101	A OR $\overline{B}$
110	A − B
111	not used

FIM!