

01 - Sistemas Numéricos

Sistema Decimal

Composto por dez símbolos (0,1,2,3,4,5,6,7,8,9). Todos os números podem ser escritos como potências de dez (base dez).

Exemplo 1.1:

$$37,75 = 37,75_{(10)} = 3 \times 10^1 + 7 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}$$

Sistema Binário

Composto por dois símbolos (0,1). Todos os números podem ser escritos como potências de dois (base dois).

Exemplo 1.2:

$$100101,11 = 100101,11_{(2)} = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

Sistema Octal

Composto por oito símbolos (0,1,2,3,4,5,6,7). Todos os números podem ser escritos como potências de oito (base oito).

Exemplo 1.3:

$$45,6 = 45,6_{(8)} = 4 \times 8^1 + 5 \times 8^0 + 6 \times 8^{-1}$$

Sistema Hexadecimal

Composto por dezesseis símbolos (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). Todos os números podem ser escritos como potências de dezesseis (base dezesseis). As letras assumem os valores de 10 a 15 para as conversões entre sistemas numéricos ou conversões de base.

Conversão entre sistemas ou Conversão de Base

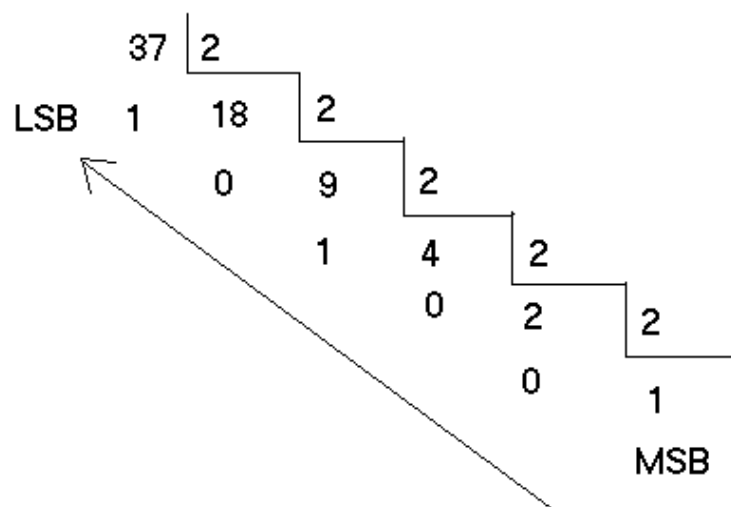
Decimal - Binário

Para a parte inteira do número, devemos realizar divisões sucessivas por 2, até encontrarmos quociente 1, este quociente é o MSD (Most Significant Digit), os restos formam o restante do n^o, sendo o 1^o resto o LSD (Least Significant Digit), como ilustrado no exemplo a seguir.

Exemplo 1.4:

$37,75_{(10)}$

Conversão da parte inteira 37



Conversão da parte fracionária 0,75

Efetua-se multiplicações sucessivas por 2 retirando os um's ou zeros na seqüência em que aparecem.

$$0,75 \times 2 = 1,50 \rightarrow 1 \rightarrow 0,1$$

$$\rightarrow 0,50 \times 2 = 1,00 \rightarrow 1 \rightarrow 0,01$$

$$\text{Portanto } 37,75_{(10)} = 100101,11_{(2)}$$

Binário - Decimal

Consiste em somar os produtos na série de potências que representam o nº binário.

Exemplo 1.5:

$$100101,11 =$$

$$100101,11_{(2)} =$$

$$1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} =$$

$$2^5 + 2^2 + 2^0 + 2^{-1} + 2^{-2} =$$

$$32 + 4 + 1 + 0,5 + 0,25 =$$

$$37,75_{(10)}$$

Octal - Binário e Binário - Octal

A conversão fica muito facilitada pela relação de potências de 2 entre as bases. A cada grupo de três algarismos ou bits no sistema binário, corresponde um algarismo no sistema octal.

Exemplo 1.5:

45,6₍₈₎ para base 2

4 5 , 6

100 101 110

Por tanto :

$$45,6_{(8)} = 100101,11_{(2)}$$

11010,01₍₂₎ para base 8

011 010 , 010

3 2 2

Por tanto :

$$11010,01_{(2)} = 32,2_{(8)}$$

Hexadecimal - Binário e Binário - Hexadecimal

A cada quatro algarismos ou bits do sistema binário corresponde 1 algarismo do sistema hexadecimal.

Exemplo 1.8:

$25, C_{(16)}$ para base 2

2 5 , C
0010 0101 1100

Por tanto :

$$25, C_{(16)} = 100101,11_{(2)}$$

$111101001,101_{(2)}$ para base 16

1 1110 1001 , 1010
1 F 9 A

Por tanto :

$$111101001,101_{(2)} = 1F9, A_{(16)}$$

Exercícios Propostos

- 1.1) Converter de binário para decimal o nº $1101011_{(2)} = \text{_____}_{(10)}$
- 1.2) Qual binário é o consecutivo ao nº $10111_{(2)}$?
- 1.3) Qual o maior nº decimal que se pode representar com 12 bits?
- 1.4) Quantos bits são no mínimo necessários para representar os dez algarismos decimais?
- 1.5) Converter todos os algarismos do sistema Hexadecimal (de 0 a F) para binário.

02 - Códigos Numéricos

A um conjunto de símbolos que representam palavras, números e letras (informações) damos o nome de código. Um exemplo de código bastante conhecido é o código Morse.

Código BCD.

O código BCD (Binary Coded Decimal) decimal codificado em binário é muito utilizado nas conversões entre o sistema decimal e o binário. Cada algarismo do sistema decimal é representado por seu correspondente binário de quatro bits.

Exemplo:

4 7 9 8 , 0 6 (decimal)
0100 0111 1001 1000 , 0000 0110 (BCD)
10010101111110,000011110... (binário)

Comparando a representação BCD e binária, fica evidente a vantagem do BCD, principalmente para a parte fracionária do nº em questão.

Código Excesso-3

Semelhante ao BCD exceto por somar-se 3 a cada algarismo decimal antes de fazer a conversão como no código BCD. Para certas operações aritméticas oferece vantagem sobre o BCD.

Exemplo:

59 5 + 3 = 8 9 + 3 = 12
 8 12
 1000 1100 (código Excesso - 3)
 59 (decimal)

Código de GRAY

Somente um bit muda quando se passa de um n° para o seu consecutivo.

Para obtenção do código partimos do binário mantendo o mesmo MSD. O bit seguinte é somado com o bit à esquerda sem considerar qualquer transporte.

Exemplo:

$$0111_{(2)} \rightarrow (MSD) = 0 \quad 1+0=1 \quad 1+1=0 \quad 1+1=0 \quad \rightarrow 0100_{(GRAY)}$$

$$1000_{(2)} \rightarrow (MSD) = 1 \quad 0+1=1 \quad 0+0=0 \quad 0+0=0 \quad \rightarrow 1100_{(GRAY)}$$

$$0111_{(2)} \rightarrow 0100_{(2)} \quad (\text{Nesta transição todos os bits mudaram})$$

$$0100_{(GRAY)} \rightarrow 1100_{(GRAY)} \quad (\text{Apenas o 1º bit mudou})$$

Código de Paridade

Código de Paridade ou bit de paridade tem por função a detecção de erros que podem ocorrer durante a transmissão dos dados. Acrescenta-se um bit extra aos dados conforme a paridade escolhida:

- Paridade par - n° de 1's for ímpar, acrescenta-se o bit 1.
- Paridade par - n° de 1's for par, acrescenta-se o bit 0.
- Paridade ímpar - n° de 1's for par, acrescenta-se o bit 1.
- Paridade ímpar - n° de 1's for ímpar, acrescenta-se o bit 0

Na recepção dos dados é feita a verificação se a paridade foi mantida, caso haja erro um sinal retorna ao transmissor para que os dados sejam retransmitidos.

Exemplo:

Supondo que o sistema tenha adotado paridade ímpar e os dados para serem transmitidos são: 1001011. No transmissor será acrescentado um bit 1 para que o n° de um's seja ímpar. Teremos portanto 111001011 onde o um mais a esquerda é o bit de paridade.

Códigos Alfanuméricos

Os códigos alfanuméricos mais conhecidos são o ASCII (American Standard Code for Information Interchange) e o EBCDIC (Extended Binary-Coded-Decimal Interchange Code).

O ASCII Trabalha com 7 bits podendo representar $2^7 = 128$ caracteres diferentes, incluindo funções de controle.

O EBCDIC Usa o formato BCD e trabalha com 8 bits, podendo representar $2^8 = 256$ caracteres diferentes.

O UNICODE Dito universal por representar todos os caracteres de todas as línguas, trabalha com 16 bits, o que possibilita $2^{16} = 65536$ caracteres incluindo funções de controle.

3. Funções e Portas Lógicas

Os circuitos digitais são projetados para operarem no modo binário (0,1) executando funções ou operações, que são mais bem entendidas com o uso da teoria dos conjuntos. Para relacionar as várias funções em circuitos lógicos usa-se a Álgebra Booleana cujas variáveis são as do sistema binário.

A tabela verdade é a representação gráfica da função booleana entre as entradas e a saída, facilitando a simplificação da solução dos problemas lógicos.

Função Inversora NOT (NÃO).

Símbolo da porta NOT

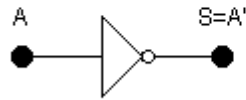


Tabela Verdade

A	S
0	1
1	0

A função executada é $S = \bar{A}$ ou $S = A'$, a saída é o complemento da entrada ou seja, a saída é 0 se a entrada for 1 e vice versa.

Função AND (E).

Símbolo da porta AND

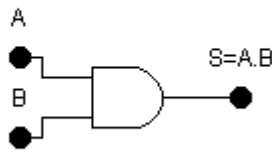


Tabela Verdade

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

A função executada, $S = A \cdot B$, é equivalente a intersecção de conjuntos, a saída será igual às duas entradas simultaneamente,

Função OR (OU).

Símbolo da porta OR

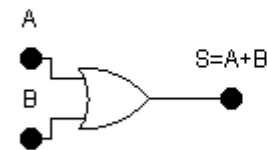


Tabela Verdade

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

A função executada, $S = A + B$, é equivalente a união de conjuntos, a saída será igual a um, desde que uma das entradas seja um.

Função NAND (NÃO E).

Símbolo da porta NAND

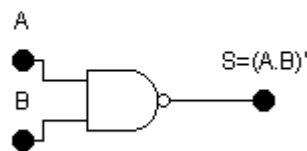
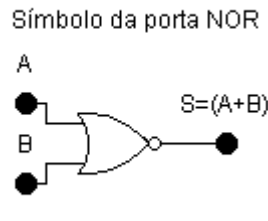


Tabela Verdade

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

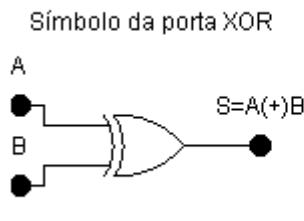
A operação, $S = (A \cdot B)'$ ou $S = \overline{A \cdot B}$, equivale a operação AND seguida por um inversor (NOT), ou seja, é o complemento ou a negação da AND.

Função NOR (NÃO OU).

Símbolo da porta NOR	Tabela verdade															
	<table border="0"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</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>0</td> </tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0
A	B	S														
0	0	1														
0	1	0														
1	0	0														
1	1	0														

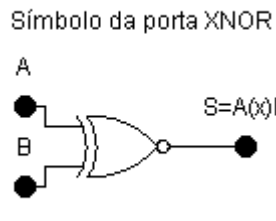
A operação, $S = (A + B)'$ ou $S = \overline{A + B}$, é o complemento ou a negação da operação OR e equivale a operação OR seguida por um inversor (NOT).

Função - XOR- EXCLUSIVE OR (OU EXCLUSIVO).

Símbolo da porta XOR	Tabela verdade															
	<table border="0"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <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> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	0
A	B	S														
0	0	0														
0	1	1														
1	0	1														
1	1	0														

A função, $S = A \oplus B$ ou $A(+)B$, é a soma Booleana das variáveis de entrada (como veremos mais adiante), ou como se percebe pela tabela verdade, a saída será zero se as entradas estiverem simultaneamente em nível alto ou baixo.

3.7 Função -XNOR- EXCLUSIVE NOR (NÃO OU EXCLUSIVO).

Símbolo da porta XNOR	Tabela verdade															
	<table border="0"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</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> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	1
A	B	S														
0	0	1														
0	1	0														
1	0	0														
1	1	1														

A operação, $S = \overline{A \oplus B}$ ou $S = A \otimes B$ ou ainda $S = A (\times) B$, é o complemento ou a negação da função XOR, ou seja, é a função XOR seguida por um inversor.

4. Simplificação de Equações Lógicas

Circuitos lógicos digitais são projetados para executar funções específicas, estes projetos normalmente permitem simplificações das equações lógicas obtidas, a seguir serão vistas ferramentas utilizadas para efetuar estas simplificações.

4.1 Álgebra de Boole

A álgebra booleana é baseada no sistema binário (somente algarismos 0 e 1) e permite a simplificação de expressões lógicas (simplificação algébrica).

4.1 Postulados da álgebra de Boole

(a) $A + B =$ operação OR (b) $A.B =$ operação AND

4.2 Teoremas Booleanos

(T.1) $A.0 = 0$ (T.2) $A.1 = A$ (T.3) $A.A = A$ (T.4) $A.\bar{A} = 0$

(T.5) $A+0 = A$ (T.6) $A+1 = 1$ (T.7) $A+A = A$ (T.8) $A+\bar{A} = 1$

(T.9) $A+B = B+A$ (T.10) $A.B = B.A$

(T.11) $A+(B+C) = (A+B)+C = A+B+C$ (T.12) $A(BC) = (AB)C = ABC$

(T.13a) $A(B+C) = AB+AC$ (T.13b) $(D+A)(B+C) = DB+DC+AB+AC$

(T.14) $A+AB = A$ (T.15) $A+\bar{A}B = A+B$

O teorema (T.14) pode ser comprovado:

$$A+AB = A(1+B) \text{ como } 1+B=1 \text{ (6)} \Rightarrow A+AB = A.1 \text{ (2)} = A \quad \therefore A+AB =$$

O teorema (T.15) pode ser comprovado como abaixo ou pelas tabelas verdade:

$$A+B = (A+\bar{A})(A+B) = AA+AB+\bar{A}A+\bar{A}B = \quad \text{já que } A+\bar{A}=1 \text{ (8)}$$

$$AA = A \text{ (3)} \quad \bar{A}A = 0 \text{ (4)} \quad A+\bar{A}B = A \text{ (14)}$$

$$\Rightarrow \quad A+B \quad = \quad A+\bar{A}B$$

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

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

Exercícios resolvidos:

Simplificação de Expressões

4.1.1 $S = AB + AC + BC + ABC$

Solução: colocando A em evidência

$$S = A(B+C+BC) + BC; \quad C+BC = C \text{ (T.15)}$$

$$S = A(B+C) + BC$$

4.1.2 $S = AB\bar{C}\bar{D} + ABCD$ (colocando AB em evidência) \Rightarrow

$$S = AB(\bar{C}\bar{D} + CD) \Rightarrow AB(C \otimes D)$$

4.2 Teoremas de DeMorgan.

$$(T.16) \quad \overline{(A+B)} = \bar{A}.\bar{B} \quad (T.17) \quad \overline{(A.B)} = \bar{A}+\bar{B}$$

Dupla inversão $\overline{\bar{A}} = A$

Exercícios resolvidos:

Simplificações usando teoremas de DeMorgan

$$4.2.1 \quad S = \overline{A\bar{B} + C} \quad \text{solução:}$$

$$S = \overline{A\bar{B} + C} \quad \text{usando(T.16)} \Rightarrow S = \overline{(A\bar{B}) \cdot \bar{C}} \quad \text{usando(T.17)} \Rightarrow$$

$$\Rightarrow S = \overline{(A + \bar{B}) \cdot \bar{C}} = \overline{(A + B) \cdot \bar{C}} \Rightarrow$$

$$S = \overline{A\bar{C}} + \overline{B\bar{C}}$$

$$4.2.2 \quad S = \overline{\overline{AB} + \overline{AB}} \quad \text{solução:}$$

$$S = \overline{\overline{AB} + \overline{AB}} \quad (T.16) \Rightarrow S = \overline{\overline{AB} \cdot \overline{AB}} = \overline{(\bar{A} + \bar{B}) \cdot (\bar{A} + \bar{B})} = \overline{(A + B) \cdot (\bar{A} + B)} \Rightarrow$$

$$S = A\bar{A} + AB + \bar{B}A + \bar{B}B; \quad \text{como } A\bar{A} = 0, \quad \bar{B}B = 0 \Rightarrow$$

$$S = AB + \bar{A}B$$

4.3 Exercícios propostos:

4.3.1 Mostrar a veracidade das igualdades usando tabelas verdade:

$$a) \quad \bar{A} \cdot B + A \cdot \bar{B} = A \oplus B$$

$$b) \quad \bar{A} \cdot \bar{B} + AB = A \otimes B$$

4.3.2 Usando o exercício anterior e os teoremas aprendidos, mostrar a veracidade da igualdade

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

4.4 Mapa de Karnaugh

O mapa de Karnaugh é um processo gráfico para a simplificação de equações lógicas e consiste basicamente no agrupamento de 1's numa tabela verdade convenientemente modificada. Chama-se de mapa K a esta tabela, e de células os espaços para as saídas.

O mapa é construído de maneira que haja apenas uma transição de bits quando se passa de uma célula para outra.

Agrupando-se os um's dois a dois, ou quatro a quatro, ou oito a oito, ou ainda em maior ordem possível. As variáveis que serão mantidas são aquelas em que os bits não sofrem transição quando as células são percorridas.

O mapa pode ter suas extremidades unidas, formando um cilindro, e desta forma os um's nas extremidades podem ser agrupados como descrito anteriormente.

4.4.1 Mapa de Karnaugh para 2 variáveis de entrada.

O N° de células é igual a dois elevado ao n° de entradas, temos então quatro células para o mapa de duas entradas conforme exemplo da figura 4.4.1

fig.4.4.1) Tabela verdade Mapa K p/ 2 Variáveis

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

A \ B	0	1
0	0	1
1	0	1

$S = A$

4.4.2 Mapa de Karnaugh para 3 variáveis de entrada. O N° de células é igual a $2^3 = 8$ conforme exemplo na figura 4.4.2

fig.4.4.2) Tabela verdade Mapa K p/ 3 Variáveis

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

AB \ C	00	01	11	10
0	0	1	1	1
1	1	1	1	0

$S = \bar{A}C + B + A\bar{C} = A \oplus C + B$

fig.4.4.3) Tabela verdade Mapa K

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

AB \ C	00	01	11	10
0	0	1	1	1
1	1	1	0	1

$\Rightarrow A + B \oplus C$

Exercícios resolvidos:

- 1) Simplificar e desenhar os circuitos de $S = \bar{A}B + A\bar{B} + \bar{A}\bar{B}$.
- 2) Projetar um circuito lógico de três entradas cuja saída esteja no nível lógico alto sempre que a maioria de suas entradas estiver no nível lógico alto. Simplificar (se possível) as equações algebricamente e pelo mapa K e desenhar os circuitos.
- 3) Projetar um circuito lógico de três entradas que gera paridade ímpar. Simplificar (se possível) as equações algebricamente e pelo mapa K e desenhar os circuitos.

1) $S = \bar{A}B + A\bar{B} + \bar{A}\bar{B}$

Solução: $\bar{A}B(1) + A\bar{B}(2) + \bar{A}\bar{B}(3)$

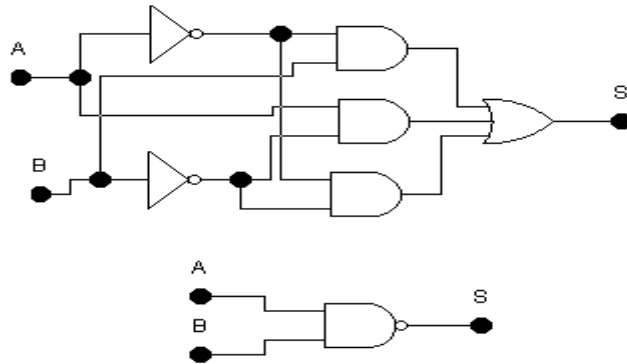
(1) - (3) $\rightarrow \bar{A}(B + \bar{B}) = \bar{A}$ (2) - (3) $\rightarrow \bar{B}(A + \bar{A}) = \bar{B} \Rightarrow S = \bar{A} + \bar{B}$

Equação	A	B	S
(3)	0	0	1
(1)	0	1	1
(2)	1	0	1
	1	1	0

A \ B	0	1
0	1	1
1	1	0

$\Leftrightarrow \bar{A} + \bar{B}$

$S = \bar{A} + \bar{B} \Rightarrow S = \overline{A \cdot B} = \overline{A \cdot B} \Rightarrow S = \bar{A} + \bar{B} \text{ ou } S = \overline{A \cdot B}$



Circuitos do ex 1 - sem simplificação e simplificado

2) Tabela verdade

Mapa K

A	B	C	S	Equações
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	0	
1	0	1	1	$A\bar{B}C$
1	1	0	1	$ABC\bar{C}$
1	1	1	1	ABC

A \ B	00	01	11	10
0	0	0	1	0
1	0	1	1	1

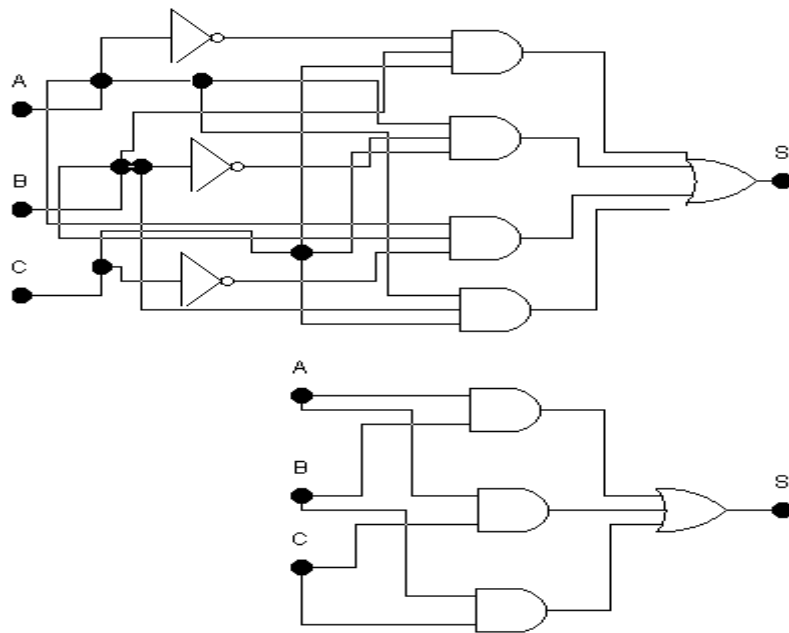
$\Rightarrow AB + AC + BC$

Simplificação algébrica

$S = \bar{A}BC(1) + A\bar{B}C(2) + ABC\bar{C}(3) + ABC(4)$

(4) - (1) $\rightarrow BC(\bar{A} + A) = BC$ (4) - (2) $\rightarrow AC(\bar{B} + B) = AC$ (4) - (3) $\rightarrow AB(\bar{C} + C) = AB$

$S = AB + AC + BC$



Circuitos Ex.2 sem simplificação e simplificado

3) Tabela verdade

Mapa K

A	B	C	P	Equações
0	0	0	1	\overline{ABC}
0	0	1	0	
0	1	0	0	
0	1	1	1	\overline{ABC}
1	0	0	0	
1	0	1	1	$A\overline{BC}$
1	1	0	1	$AB\overline{C}$
1	1	1	0	

$$\Rightarrow$$

A \ B	00	01	11	10
0	1	0	1	0
1	0	1	0	1

O mapa K não permite simplificação

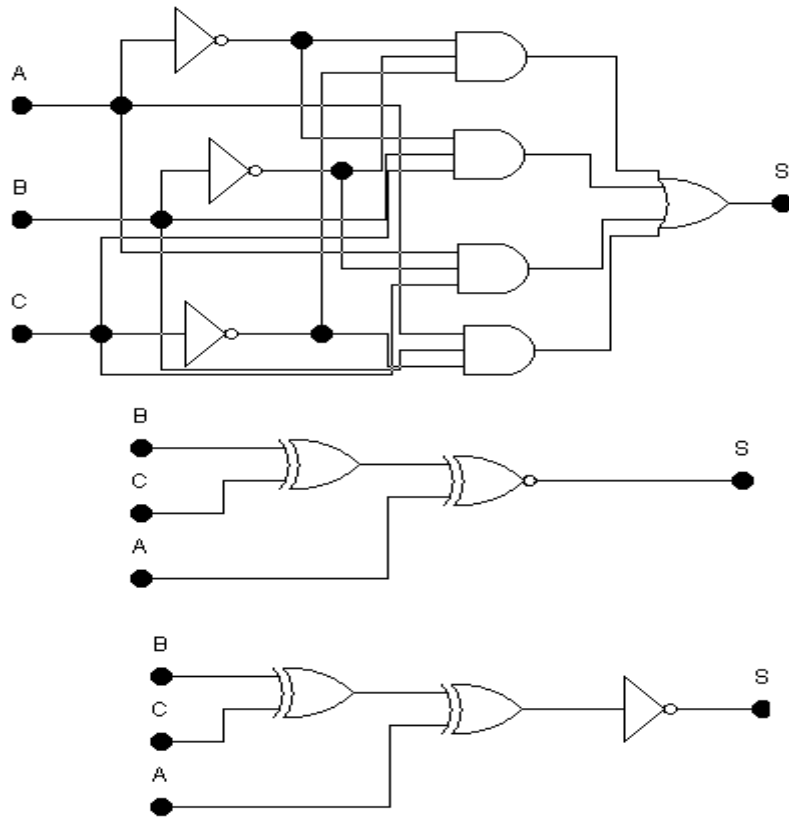
Simplificação Algébrica

$$S = \overline{ABC}(1) + \overline{ABC}(2) + A\overline{BC}(3) + AB\overline{C}(4)$$

$$(1) - (2) \rightarrow \overline{A}(\overline{BC} + BC) = \overline{A}(B \oplus C) \quad (3) - (4) \rightarrow A(\overline{BC} + BC) = A(B \oplus C)$$

$$S = \overline{A}(B \oplus C) + A(B \oplus C) \quad \text{chamando } B \oplus C = X$$

$$S = \overline{A}X + AX = \overline{A} \oplus X \Rightarrow S = \overline{A} \oplus (B \oplus C)$$



Circuitos Ex 3 sem simplificação e com simplificações

4.5 Exercícios

1) Simplificar as expressões abaixo, pelo mapa e algebricamente, desenhar o circuito lógico

$$1-a) S = \overline{A}\overline{B} + A\overline{B} + AB$$

Tabela Verdade Mapa K

A	B	S
0	0	
0	1	
1	0	
1	1	

 \Leftrightarrow

A \ B	0	1
0		
1		

$$1-b) S = \overline{A}\overline{B}\overline{C} + AD + BD + CD$$

1-b) Tabela Verdade

Mapa K

A	B	C	D	S
0	0	0	0	
0	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	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

 \Leftrightarrow

AB \ CD	00	01	11	10
00				
01				
11				
10				

 $\Rightarrow S =$

4.5.1-c) $S = \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}BC$

A	B	C	D	S
0	0	0	0	
0	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	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

⇔

$\begin{matrix} AB \\ \backslash \\ CD \end{matrix}$	00	01	11	10
00				
01				
11				
10				

S =

4.5.1-d) $S = \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D}$

A	B	C	D	S
0	0	0	0	
0	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	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

⇔

$\begin{matrix} AB \\ \backslash \\ CD \end{matrix}$	00	01	11	10
00				
01				
11				
10				

S =

4.5.2) Projete um circuito lógico de quatro entradas cuja saída esteja no nível lógico alto sempre que a maioria de suas entradas estiver no nível lógico baixo. Simplificar (se possível) as equações algebricamente e pelo mapa K.

4.5.2) Saída nível alto quando a maioria das entradas em nível baixo

D	C	B	A	S	$EQUAÇÕES$
0	0	0	0		
0	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		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

$DC \backslash BA$	00	01	11	10
00				
01				
11				
10				

$\Leftrightarrow S =$

4.5.3) Projete um circuito lógico de quatro entradas que gera paridade par. Simplificar (se possível) as equações algebricamente e pelo mapa K.

<i>D</i>	<i>C</i>	<i>B</i>	<i>A</i>	<i>S</i>	<i>EQUAÇÕES</i>
0	0	0	0		
0	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		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

⇔

$\begin{matrix} DC \\ \backslash \\ BA \end{matrix}$	00	01	11	10
00				
01				
11				
10				

S =

4.5.4) Projetar um circuito verificador de paridade para quatro entradas.

<i>D</i>	<i>C</i>	<i>B</i>	<i>A</i>	<i>S</i>	<i>EQUAÇÕES</i>
0	0	0	0		
0	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		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

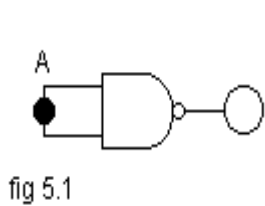
<i>DC</i> \ <i>BA</i>	00	01	11	10
00				
01				
11				
10				

05 - Universalidade das portas NAND e NOR

As portas NAND ou as portas NOR quando convenientemente combinadas podem substituir quaisquer outras portas e por isto são chamadas de portas universais.

Exercícios

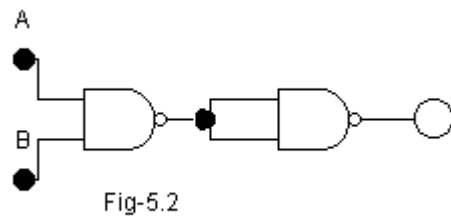
5.1) Para cada circuito a seguir: montar a tabela verdade, a equação booleana correspondente e o nome da função lógica, mostrando as equações em todos os nós.



<i>A</i>	<i>S</i>
0	
1	

↔ *S* =

Porta:



<i>A</i>	<i>B</i>	<i>S</i>
0	0	
0	1	
1	0	
1	1	

↔ *S* =

Porta:

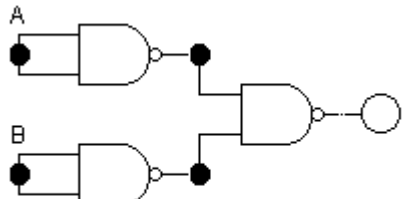


Fig-5.3

A	B	S
0	0	
0	1	
1	0	
1	1	

$\Leftrightarrow S =$

Porta:

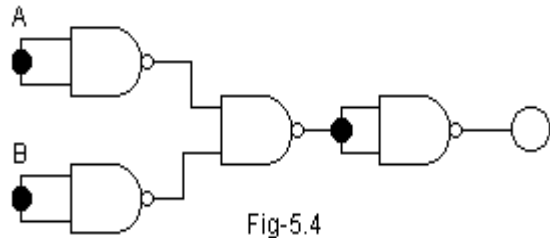


Fig-5.4

A	B	S
0	0	
0	1	
1	0	
1	1	

$\Leftrightarrow S =$

Porta:

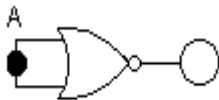


Fig-5.5

A	S
0	
1	

$\Leftrightarrow S =$

Porta:

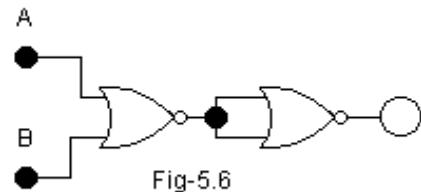


Fig-5.6

A	B	S
0	0	
0	1	
1	0	
1	1	

$\Leftrightarrow S =$

Porta:

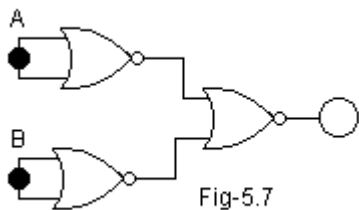


Fig-5.7

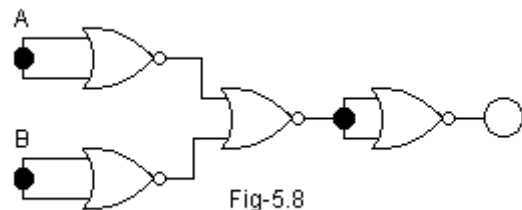


Fig-5.8

A	B	S
0	0	
0	1	
1	0	
1	1	

$\Leftrightarrow S =$

Porta:

A	B	S
0	0	
0	1	
1	0	
1	1	

$\Leftrightarrow S =$

Porta:

5.2) Construir uma porta EXOR utilizando somente portas NAND

Sugestão: a partir de $S = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$, utilizar os teoremas de DeMorgan.

5.3) Construir uma porta EXNOR utilizando somente portas NOR

Sugestão: A partir de $S = A \otimes B = \overline{A \cdot B} = \overline{A} \cdot \overline{B} + A \cdot B$, utilizar os teoremas de DeMorgan.

06 - Circuitos seqüenciais

Os circuitos seqüenciais são assim chamados por terem a saída não só dependente das entradas, mas também de uma realimentação da saída, ou seja, de uma condição anterior de sua saída. Podemos dividir os circuitos seqüências em dois grupos: Os latches e os flip-flops.

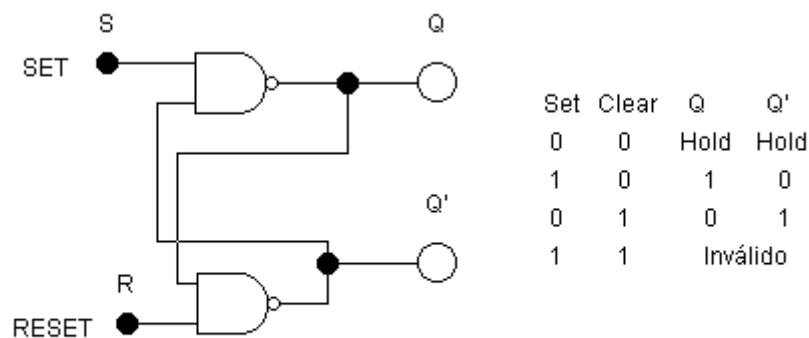


Fig.6.1 LATCH com NAND

Na figura 6.1 temos o latch RS construído com portas NAND e a respectiva tabela verdade, onde notamos que o RS possui um estado indesejável, se R e S forem iguais a 1 teremos saída inválida. na figura 6.2 o latch RS está construído com portas NOR
Na figura 6.4 está representado o símbolo do latch RS.

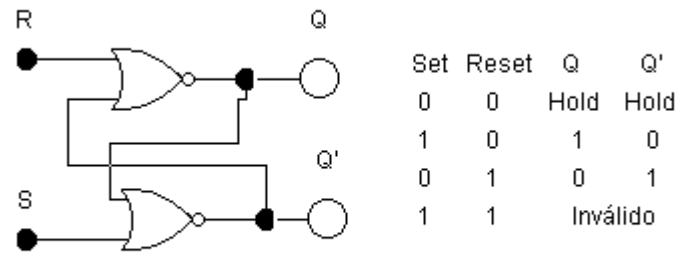
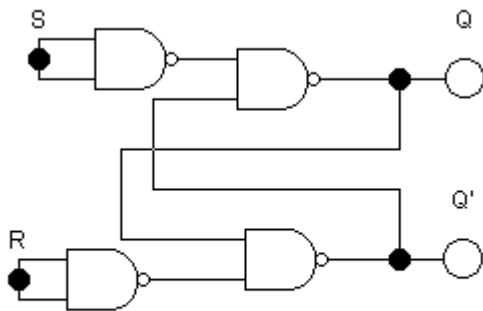


Fig. 6.2 LATCH com NOR

(Ex 6.1 a) Para o circuito da fig. 6.3 montar a respectiva tabela verdade.



S	R	saída Q
0	0	
0	1	
1	0	
1	1	

Fig. 6.3 LATCH com NAND prático

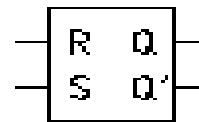


Fig. 6.4 Símbolo do RS

O Latch RS pode melhor ser aplicado em circuitos seqüenciais se tiver uma entrada de controle síncrona, esta entrada síncrona funciona como uma entrada de habilitação. Normalmente o sinal de comando é uma onda quadrada, este sinal é chamado de CLOCK.

Quando a ativação síncrona é feita pelo nível do clock, o circuito é chamado de latch. Quando a ativação é feita pela transição entre os níveis, o circuito é chamado de flip-flop.

A figura 6.5 mostra um circuito com duas portas AND que funciona como controle, a saída só responde as entradas R ou S quando a entrada E estiver em nível alto. Substituindo a chave por um gerador de clock, a saída acompanha as entradas somente quando o pulso de clock estiver no nível alto.

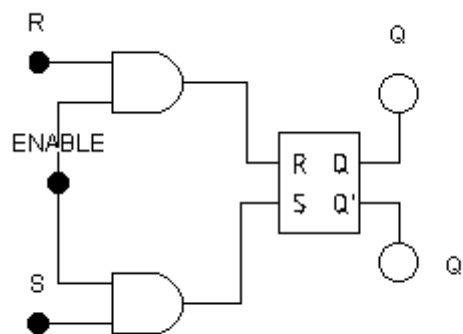
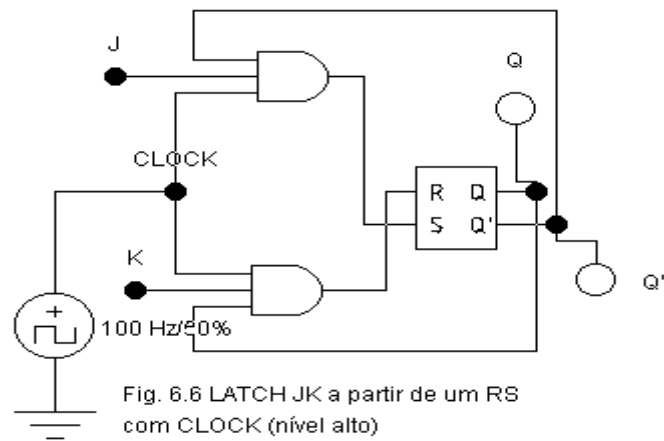


Fig. 6.5 LATCH RS com ativação por nível



O flip-flop ou latch JK é feito a partir do flip-flop ou latch RS, mas com o circuito adicional visto na figura 6.6, o estado indesejável que ocorre no RS não ocorre no JK. Caso as entradas síncronas J e K estejam simultaneamente no nível alto o flip-flop fica oscilando, ou seja, as saídas ficam alternando (toggle) entre nível alto e baixo sucessivamente. Estando as entradas JK simultaneamente no nível baixo o flip-flop mantém as saídas travadas (hold) no estado em que se encontravam antes da mudança destas mesmas entradas.

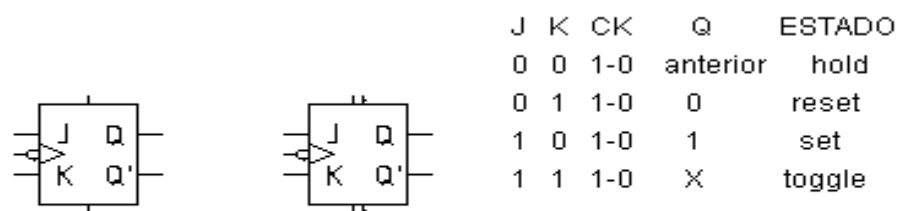


Fig. 6.7 FFs JK ativados na transição do clock de alto para baixo e tabela verdade

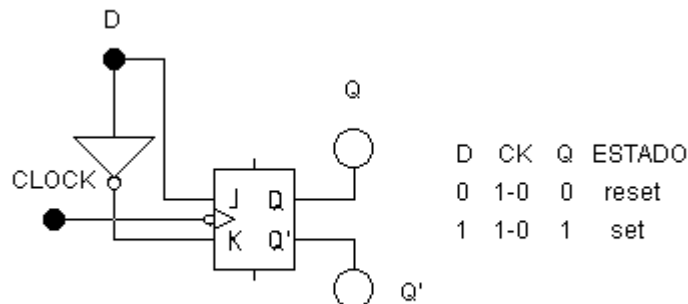
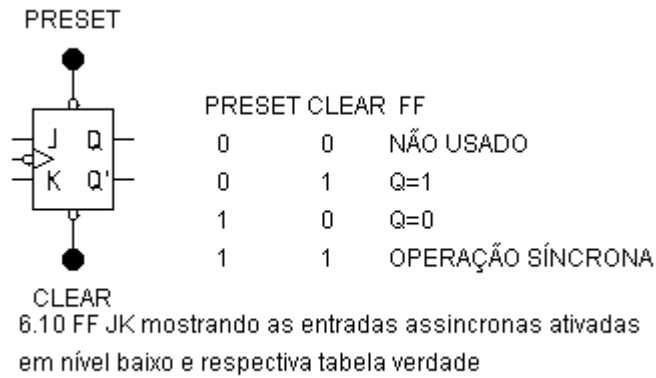
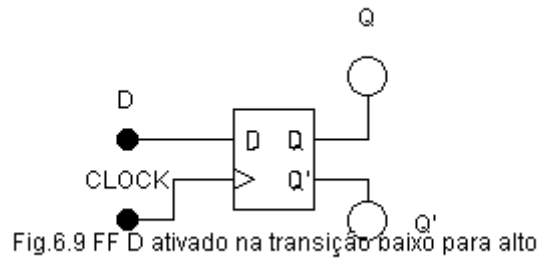


Fig. 6.8 FF D a partir de um JK e tabela verdade

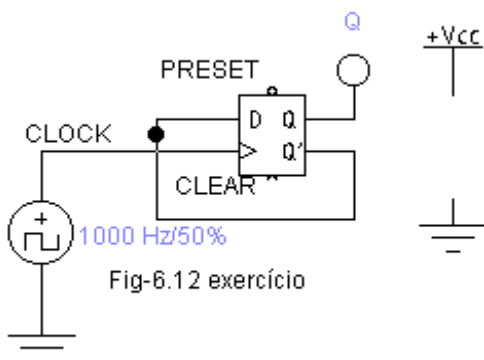
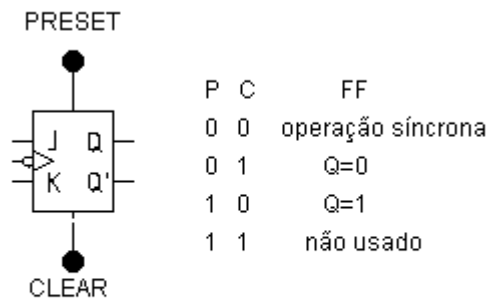
A figura 6.7 mostra os símbolos de flip-flops JK ativados por clock na transição alto para baixo. a diferença entre os dois flip-flops está nas entradas assíncronas.

O flip-flop D feito a partir de um JK mostra que a saída acompanha a entrada sempre que o clock estiver presente, conforme a fig. 6.8 e na fig 6.9 está mostrado o símbolo do FF D para simulação de seu funcionamento.



Os flip-flops JK e D normalmente possuem entradas adicionais chamadas assíncronas, são ativadas por nível de tensão, independem do clock, por isto são chamadas assíncronas, e enquanto ativas mantêm as entradas síncronas desativadas (hold).

As entradas assíncronas podem ser ativas por nível alto como representado na fig. 6.11 ou ativas por nível baixo como na figura 6.10. Estas entradas possuem um estado não usual na prática. este estado varia conforme a escolha do circuito, conforme pode ser verificado pelas tabelas verdade que acompanham os circuitos.



- 6.1.1 Exercício
- Ligar o Preset e Clear para que o circuito funcione
 - Em qual dos quatro pontos da forma de onda do clock a saída comuta?
 - Tabela verdade
- | D | Clock | Q |
|---|-------|---|
|---|-------|---|

07 - Circuitos seqüenciais Contadores

Contadores Assíncronos

Os contadores assíncronos possuem os flip-flops em estado de toggle e os pulsos de clock comandam geralmente o primeiro flip-flop e os flip-flops subseqüentes são comandados pelas saídas dos anteriores. Na figura 7.1 temos um contador assíncrono e a respectiva tabela verdade para contagem crescente de 0 a 3.

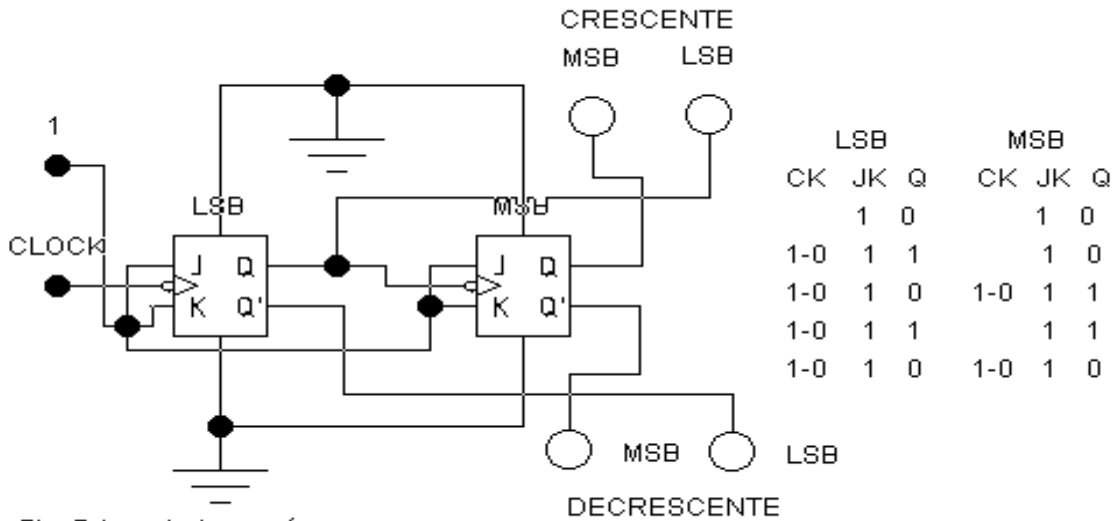


Fig. 7.1 contador assíncrono

Contador Síncrono

Contador síncrono é aquele no qual os flip-flops recebem os pulsos de comando de clock simultaneamente. Na figura 7.2 está representado um contador síncrono de 2 bits e a tabela verdade para o contador crescente de 0 a 3.

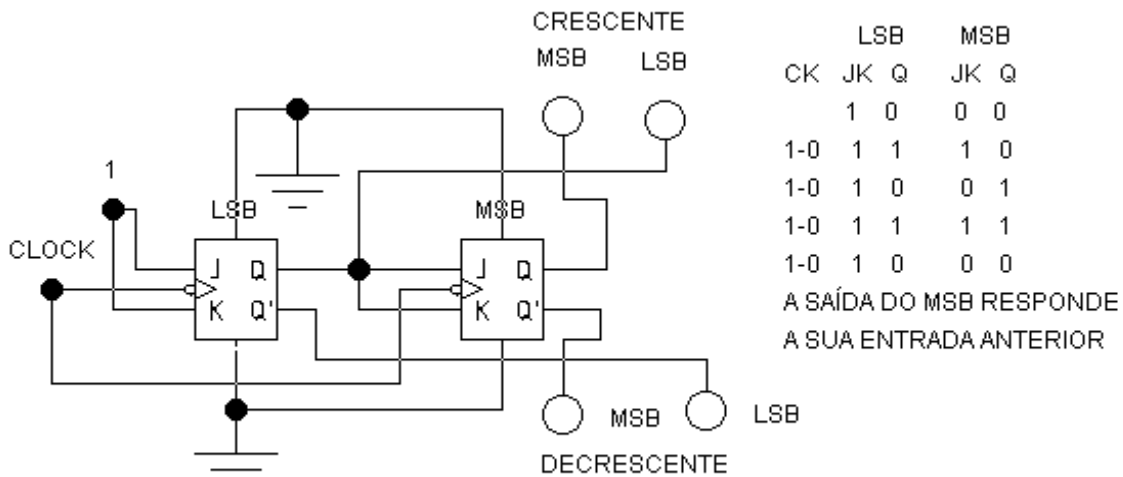


Fig. 7.2 Contador Síncrono

08 - Circuitos Seqüenciais Registradores de Deslocamento

Conversor Série Paralelo

O conversor série paralelo converte a informação que é transmitida bit a bit numa única linha, ou seja, de uma forma seqüencial, em uma transmissão simultânea ou paralela dos dados. Na figura 8.1 está representado um conversor série paralelo de 4 bits, onde a cada pulso de clock a informação é deslocada do flip-flop da esquerda para o flip-flop a direita até montar um nibble que pode agora ser transmitido de forma paralela.

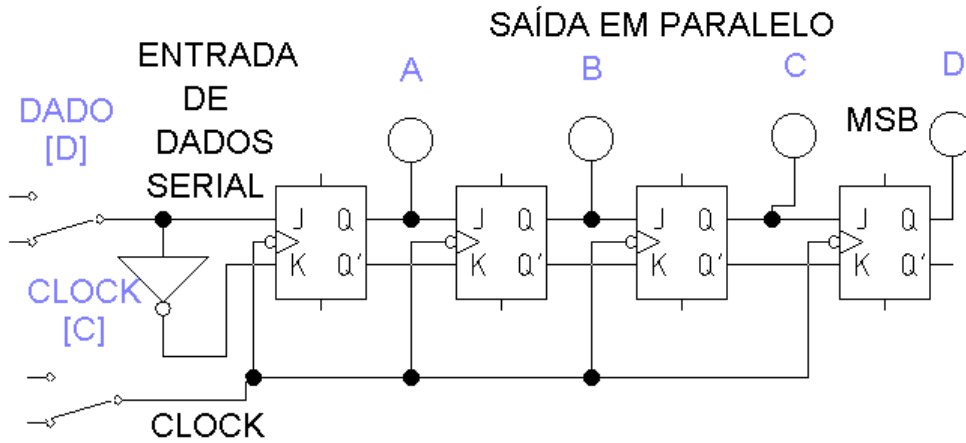


Figura 8.1 Conversor série-paralelo

Conversor Paralelo Série

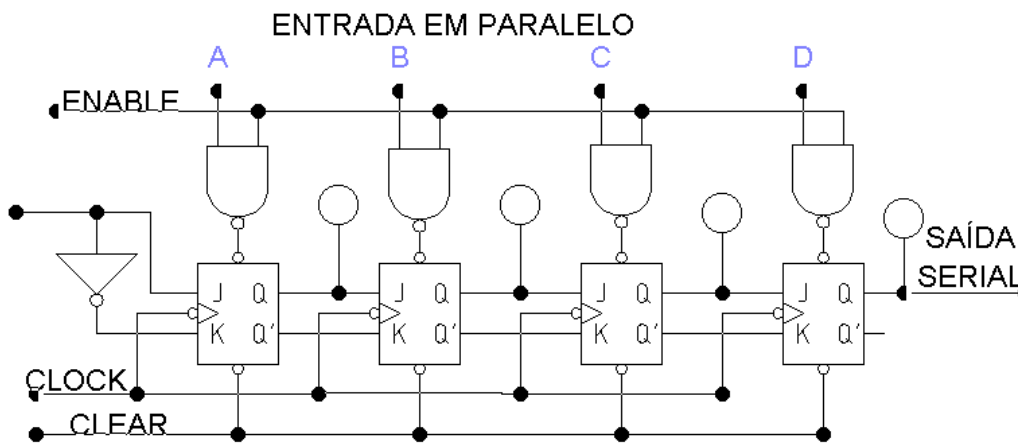


Figura 8.2 Conversor paralelo série

Na figura 8.2 temos a representação de um conversor paralelo série. Estando a entrada (enable) de habilitação em nível alto as entradas estarão refletidas nas saídas dos flip-flops, os flip-flops devem ser preparados antes de receberem as entradas, através de um pulso de clear que limpa qualquer dado armazenado nos flip-flops. A cada pulso de clock os dados serão deslocados para a saída serial.

09 - Circuitos de Processamento de Dados

Multiplexadores

Um circuito multiplexador possui várias entradas e apenas um única saída, sendo o seu princípio de funcionamento parecido com o conversor série paralelo, com a ressalva das entradas terem suas habilitações feitas de modo independente e na ordem escolhida.

Na figura 9.1 está representado um multiplexador 4x1, de 4 entradas, uma saída e as duas entradas de controle X e Y, que conforme as suas combinações permitem na saída, os valores de uma das entradas por vez, conforme tabela verdade que acompanha a figura.

Demultiplexadores

Um circuito demultiplexador possui uma entrada e várias saídas. Através de um controle adequado pode-se escolher qual saída será conectada a entrada num certo instante de tempo.

Na figura 9.2 temos a representação de um demultiplexador 1x4, possui uma entrada, 4 saídas e duas entradas de controle X e Y, responsáveis por definir qual das saídas estará recebendo o bit de entrada em certo instante.

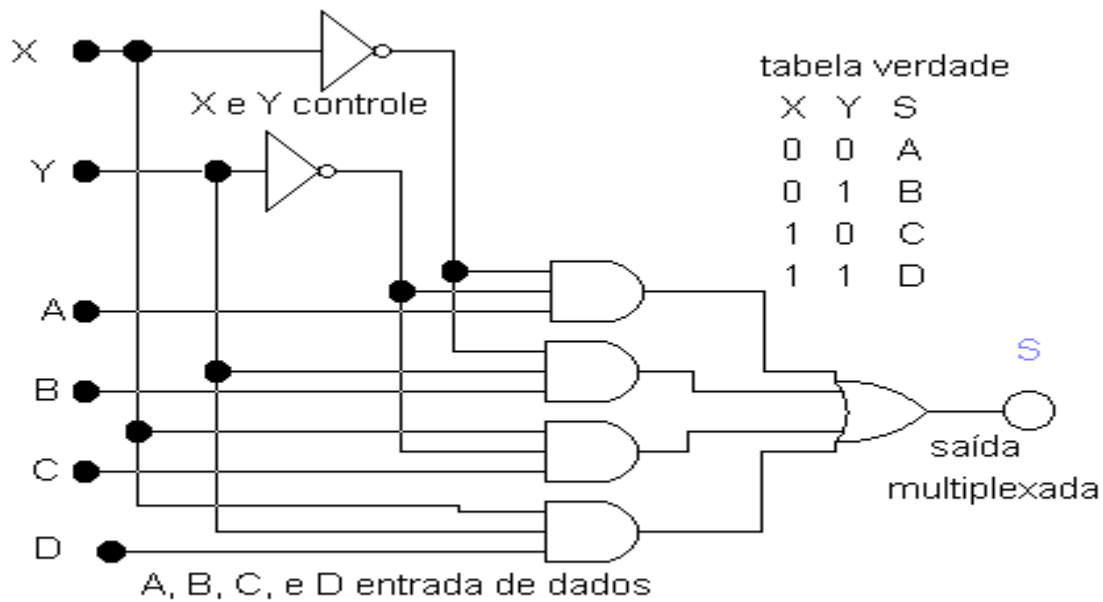


Figura 9.1 circuito multiplexador de quatro entradas

Codificadores

Circuitos codificadores convertem informações de entrada de um tipo em sinais ou informações codificadas de outro tipo. Como exemplo podemos codificar informações ou dados decimais em BCD.

Na figura 9.3 temos um circuito que faz esta codificação decimal BCD

Decodificadores

Circuitos decodificadores convertem sinais de entrada (códigos) em sinais que podem ser lidos por exemplo num display de sete segmentos, caso o código de entrada seja o BCD, teremos um decodificador acionador de sete segmentos BCD decimal. Um exemplo de um circuito ou chip da família TTL capaz de receber nas entradas os códigos BCD, converte-los em sinais que acionam um display de sete segmentos é o 7448

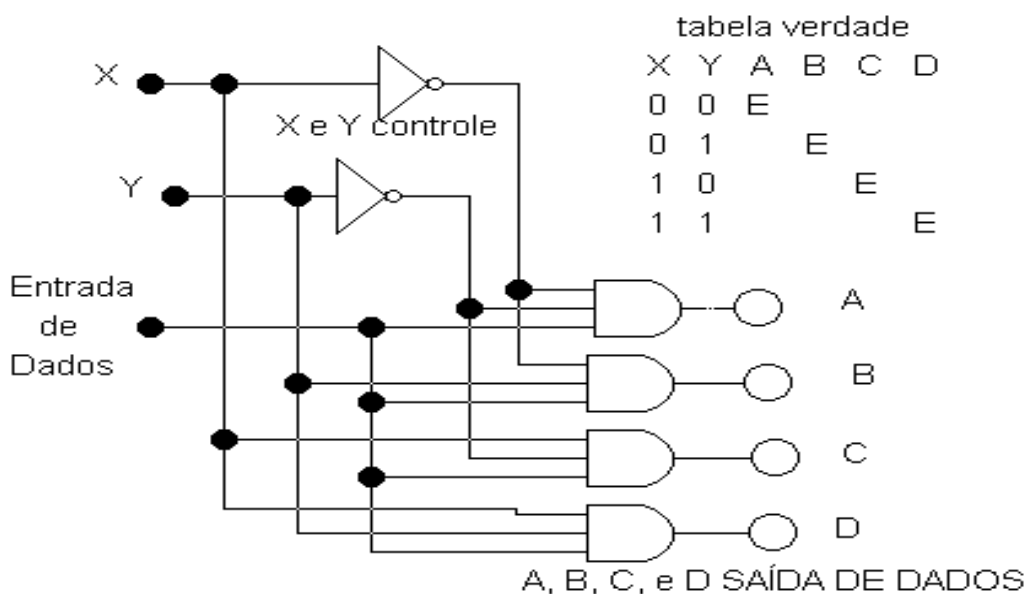


Figura 9.2 demultiplexador de um para quatro

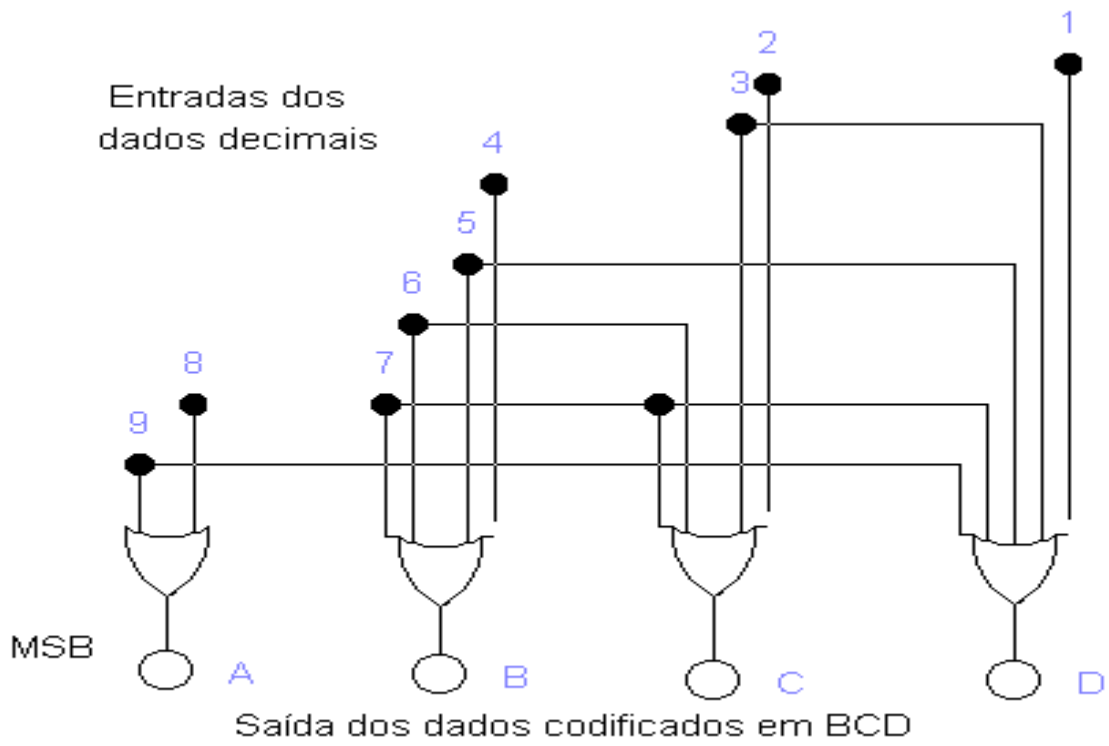


Fig. 9.3 Codificador BCD