Eletrônica Digital

Introdução

- Eletrônica Digital
- Dois níveis de tensão
  - Alto e Baixo (*High* and *Low*)
  - 1 e 0
- Álgebra de Boole
  - Sistema de Numeração
  - Funções Lógicas
  - Teoremas

Sistemas de Numeração

Sistemas de Numeração

- Sistema de numeração Decimal
  - Sistema usual de numeração
  - Dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Organização posicional:

  $2003 = 2x10^3 + 0x10^2 + 0x10^1 + 3x10^0$
  - Números são expressos como somas de potências de 10 (a **base** do sistema decimal)

Sistemas de Numeração

- Sistema de numeração Binário:
  - Dígitos: 0, 1
  - Organização posicional:

  $10101 = 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$
  - Números são expressos como somas de potências de 2 (a **base** do sistema binário)
  - Típico dos sistemas computacionais devido a correspondência entre estados inativo e ativo (*on* e *off*)

Sistemas de Numeração

- Conversão Binário - Decimal
  - Basta efetuar soma das potências de 2 equivalentes:

  $11101 = 1x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 29$
- Conversão Decimal - Binário
  - Basta efetuar sucessivas divisões inteiras por 2 até que o resultado seja 0. O valor binário é a justaposição dos restos (o primeiro resto é o dígito menos significativo e o último resto o dígito mais significativo:

Sistemas de Numeração

- Conversão Decimal - Binário (divisão inteira)

```
        29 ⌊ 2
dígito menos
significativo  1  14 ⌊ 2
                  0   7 ⌊ 2
                      1   3 ⌊ 2
                          1   1 ⌊ 2
                  dígito mais
                  significativo  1   0
```

$(11101)_2 = (29)_{10}$

Sistemas de Numeração

- Conversão binário → decimal → binário

1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Álgebra de Boole

Álgebra de Boole (ou Boleana)

- Desenvolvida pelo matemático George Boole para estudo da lógica.
- Definida sobre um conjunto de dois elementos:

(falso, verdadeiro) (0, 1) (baixo, alto)

- Seus elementos, a princípio, não tem significado numérico.
- Postulados: se x é uma variável boleana então:
  - Se $x \neq 0 \Rightarrow x = 1$
  - Se $x \neq 1 \Rightarrow x = 0$

Álgebra de Boole: operações

- São definidas algumas operações elementares na álgebra boleana:
  - Operação “Não” (NOT)
  - Operação “E” (AND)
  - Operação “Ou” (OR)
  - Operação “Ou-Exclusivo” (Exclusive-Or ou XOR)

Álgebra de Boole: operações

- Operação “Não” (NOT)
  - operador barra –
  - $\overline{0} = 1$
  - $\overline{1} = 0$
- Operação “E” (AND)
  - operador ponto .
  - 0 . 0 = 0
  - 0 . 1 = 0
  - 1 . 0 = 0
  - 1 . 1 = 1

Álgebra de Boole: operações

- Operação “Ou” (OR)
  - operador +
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 1
- Operação “Ou-Exclusivo” (XOR)
  - operador ⊕
  - 0 ⊕ 0 = 0
  - 0 ⊕ 1 = 1
  - 1 ⊕ 0 = 1
  - 1 ⊕ 1 = 0

Álgebra de Boole: funções

- Uma variável boleana só pode assumir apenas um dos valores possíveis (0 e 1)
- Uma ou mais variáveis e operadores podem ser combinados formando uma função lógica
  - $Z_1(A) = f(A) = ...$ (expressão usando var. A)
  - $Z_2(A,B) = f(A,B) = ...$ (expr. usando var. A e B)
- Resultados de uma função lógica podem ser expressos numa tabela relacionando todas as combinações possíveis dos valores que suas variáveis podem assumir e seus resultados correspondentes: a Tabela-Verdade.

Álgebra de Boole: Tabela Verdade

Variáveis | Função Lógica

Lista das combinações possíveis dos estados das variáveis de entrada

A	B	Z=f(A,B)
0	0	0
0	1	1
1	0	1
1	1	1

Resultados da função lógica para cada combinação dos estados de entrada

- Tabela-Verdade relaciona os resultados (saída) de uma função lógica para todas as combinações possíveis de suas variáveis (entrada).
- Na Tabela-Verdade acima a função lógica Z possui duas variáveis A e B, sendo Z = f(A, B) = A + B

Funções Lógicas

Funções Lógicas

- Expressam um conjunto relacionado de condições através de:
  - variáveis lógicas
  - operadores lógicos
- Podem possuir uma ou mais variáveis lógicas.
- Podem ser simplificadas utilizando as identidades e teoremas da álgebra boleana.

Funções Lógicas: Formas Padrão

- Funções lógicas podem ser reduzidas a duas "formas padrão":
  - forma padrão de soma de produtos
    expressão é uma soma (OR) de produtos (AND) de variáveis e variáveis complementadas
  - forma padrão de produto de somas
    expressão é um produto (AND) de somas (OR) de variáveis e variáveis complementadas

Forma Padrão: soma de produtos

- Dadas as funções lógicas, as mesmas poder ser reduzidas para:

f(A,B,C,D) = (A' + BC)(B + C'D)
= A'B + A'C'D + BC

f(A,B,C,D,E) = (A + (BC)')(D+BE)'
= AB'D' + AD'E' + B'D' +
B'D'E' + B'C'D' + C'D'E'

Mintermos

- Na soma padrão de produtos, cada termo correspondente a um produto é denominado mintermo.
- Embora as formas padrão não sejam as formas mais simplificadas (e por vezes mais complexas que as formas originais) se prestam a sistematização da simplificação.

Mintermos

- Cada mintermo se associa a uma possibilidade de entrada de uma função lógica. Por exemplo Y=f(A,B)=(A.B)'

Mintermo		A	B	Y
A'.B'		0	0	1
A'.B		0	1	1
A.B'		1	0	1
A.B		1	1	0

Mintermos

- A partir da tabela verdade é possível se escrever a função lógica:
  - tomando-se os mintermo correspondentes a 1
    - Y = A'.B' + A'.B + A.B'

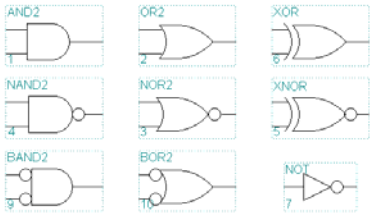
Mintermo	A	B	Y
A'.B'	0	0	1
A'.B	0	1	1
A.B'	1	0	1
A.B	1	1	0

Mintermos

- Numerando as entradas da tabela verdade é possível se identificar os mintermos e maxtermos genericamente:
  - mintermos: 0 equivale variável complementada
    1 equivale variável
- Assim a entrada 0, que equivale a A=0 e B=0:
  - mintermo: A'.B'

Diagramas Lógicos

- Simbologia das portas lógicas



Portas lógicas

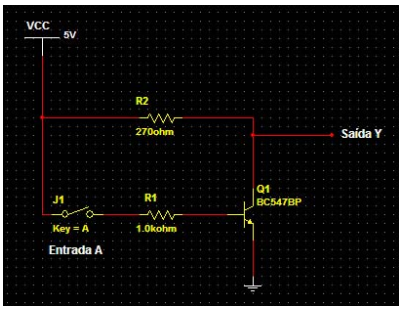
- Na prática
  - Modelos de portas lógicas
  - Datasheet

Implementação de Portas Lógicas

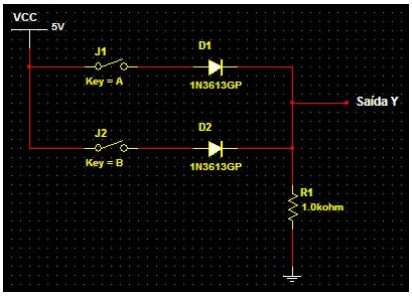
Portas lógicas

- Podemos implementar portas lógicas a partir de componentes discretos que já conhecemos
- Vamos propor a construção de três portas lógicas:
  - Inversora
  - Ou (OR)
  - E (AND)

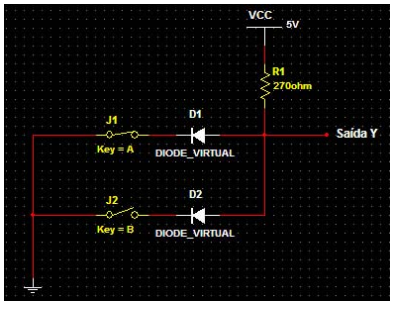
Porta Inversora



Porta OU (OR)



Porta E



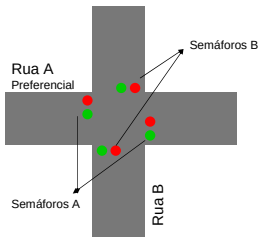
Circuitos combinacionais

Circuitos Combinacionais

- São os circuitos cujas saídas são válidas apenas enquanto houverem sinais presentes em suas entradas. Retirando tais sinais, as saídas tornam-se inválidas, mesmo que nelas existam sinais.
- São muito comuns e importantes dentro da eletrônica digital. Configurações comuns são:
  - codificadores/decodificadores
  - multiplexadores/demultiplexadores
  - somadores/subtratores
  - geradores/verificadores

Exemplo: semáforo

- Automatizar os semáforos
  - A partir da presença de carros nas ruas
  - Rua A é preferencial
  - Dois semáforos em cada rua



Exercícios

- Seletor automático por prioridade
  - 1ª toca-discos
  - 2ª toca-fitas
  - 3ª rádio
- O sistema deve identificar, através das entradas, qual aparelho enviar para o amplificador

Exercícios

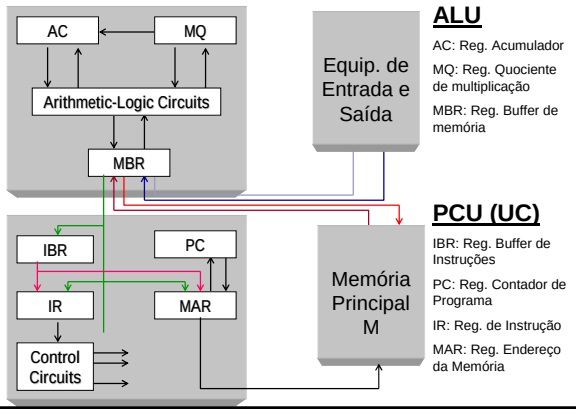
- Obtenha um circuito combinacional que funcione como uma chave seletora digital com 2 entradas e 1 saída digital.
- O circuito, em função do nível lógico aplicado a uma entrada de seleção, deve comutar à saída os sinais aplicados às entradas digitais.

Exercícios

- Implemente um circuito que simule o funcionamento do controle de saída do produto em uma máquina de doces hipotética, onde o doce será fornecido ao usuário caso o total de dinheiro inserido (moedas apenas) seja igual a R$ 0,15. Para coletar as moedas (inseridas através de uma fenda), a máquina possui três bandejas, sendo que:
  - Bandeja A: moedas de R$ 0,10.
  - Bandeja B: moedas de R$ 0,05.
  - Bandeja C: moedas de R$ 0,05, porém só existirão moedas nesta bandeja se houver moeda na bandeja B
  - Bandeja D: moedas de R$ 0,05, porém só existirão moedas nesta bandeja se houver moeda na bandeja C

Circuitos de Propósito Geral e Circuitos Dedicados

- As portas lógicas são o que chamamos de circuitos de propósito geral, pois podem ser empregadas na solução de qualquer problema envolvendo circuitos digitais.
- Já alguns problemas freqüentes encontrados no projeto de circuitos digitais levou os projetistas a estudarem implementações especiais de soluções que podem ser empregadas em diversos problemas.
- Circuitos implementados com este propósito são chamados de circuitos dedicados.



ALU

AC: Reg. Acumulador

MQ: Reg. Quociente de multiplicação

MBR: Reg. Buffer de memória

PCU (UC)

IBR: Reg. Buffer de Instruções

PC: Reg. Contador de Programa

IR: Reg. de Instrução

MAR: Reg. Endereço da Memória

Circuitos Aritméticos

Circuitos Aritméticos

- Circuitos lógicos capazes de realizar operações aritméticas são conhecidos como circuitos aritméticos.
- Existem inúmeras aplicações onde existe a necessidade de realizarmos operações aritméticas, tais como os processadores existentes em computadores, controladores de diversas naturezas ou simples calculadoras.

Somadores

- São circuitos destinados a realização de somas de 2 números binários.
- Supondo que tenhamos 2 números binários de n dígitos:

$$\begin{array}{rl} A = & A_{n-1}\ A_{n-2}\ \ldots\ A_1\ A_0 \\ B = & B_{n-1}\ B_{n-2}\ \ldots\ B_1\ B_0 \quad + \\ \hline S = S_n & S_{n-1}\ S_{n-2}\ \ldots\ S_1\ S_0 \end{array}$$

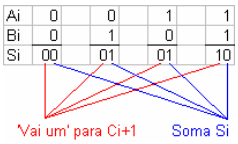
- Cada dígito Si é determinado pela soma dos dígitos Ai e Bi e também pelo excesso ocorrido na soma do dígito anterior (o 'vai um' = Ci).

Somadores

- Podemos elaborar uma tabela verdade com todas as combinações possíveis da soma de A e B, para depois sintetizarmos um circuito digital.
- Mas com esta estratégia, para **2 números de 4 bits** teríamos uma tabela verdade **com 8 entradas e 256 combinações.**
- Como a soma de cada dígito, independente de sua posição, obedece os mesmos princípios, podemos criar um circuito capaz de somar apenas a parcela de um 1 bit de dois números.

Somadores

- Observando a ilustração ao lado, vemos que a soma de um dígito binário Ai e Bi possui apenas quatro combinações.
- Note também que, conforme o valor dos dígitos somados, ocorre um excesso que deve ser transportado para a soma do próximo dígito (o 'vai um') que denominamos Ci (de *carry* ou transporte).

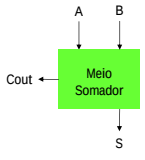


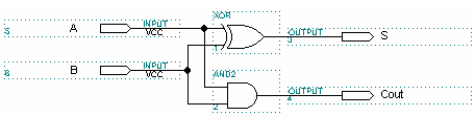
Somadores

- Assim:
  - Um '**meio somador**' (*half adder*) realiza a soma dos dígitos menos significativos (dígitos mais a direita) envolvidos.
  - Um ou mais '**somadores completos**' (*full adders*) realizam a soma dos demais dígitos, considerando o *carry* recebido da soma dos dígitos anteriores.
- Combinando-se um circuito 'meio somador' a outros circuitos 'somadores completos' podemos implementar facilmente um somador de n-bits.

Meio Somador

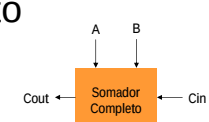
A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

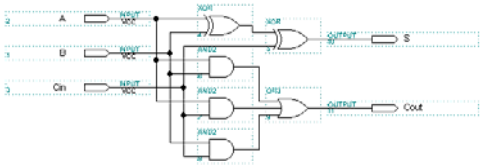




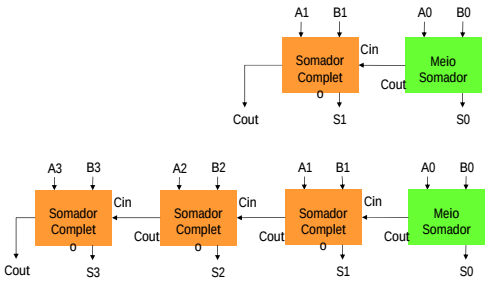
Somador Completo

A	B	Cin	S	Cou
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





Somador de 2 e 4 bits

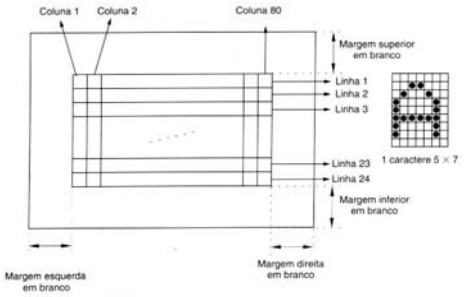


Códigos

Códigos

- Máquinas necessariamente precisam de códigos para se comunicarem
  - BCD 8421
  - Código Gray
  - Código 0123456789
  - Código ASCII
- Protocolos de comunicação

Representação textual



Codificadores e Decodificadores

- Codificador: circuito que recebe as informações de uma forma produzindo uma saída correspondente em outra forma equivalente, mais conveniente para utilização, processamento ou transmissão.
- Decodificador: circuito capaz de “desfazer” a conversão realizada por codificadores.
- Exemplos: codificadores decimal-binário, binário-decimal, octal-binário, binário-octal, bcd-decimal, bcd-7segmentos.

Codificador Octal-Binário

- Oito entradas discretas podem ter seu acionamento codificado em binário por um circuito especial: codificador Octal-Binário
- Um teclado com 7 teclas poderia ser "codificado" assim (7 estados ativos correspondendo ao acionamento das teclas e um estado inativo).

Tabela Verdade

A	B	C	D	E	F	G		Y2	Y1	Y0
1	0	0	0	0	0	0		0	0	1
0	1	0	0	0	0	0		0	1	0
0	0	1	0	0	0	0		0	1	1
0	0	0	1	0	0	0		1	0	0
0	0	0	0	1	0	0		1	0	1
0	0	0	0	0	1	0		1	1	0
0	0	0	0	0	0	1		1	1	1
X	X	X	X	X	X	X		0	0	0

Codificador Octal-Binário

- Determinação da função lógica do codificador:
  - mapa K muito grande (7 entradas)
  - soma padrão de produtos (mintermos) permite solução mais rápida.
  - Para Y0 podemos obter:

Y0 = A B'C'D'E'F'G' + A'B'C D'E'F'G' +
A'B'C'D'E F'G' + A'B'C'D'E'F'G

Y0 = B'D'F'(AC'E'G' + A'CE'G' + A'C'EG' + A'C'E'G)

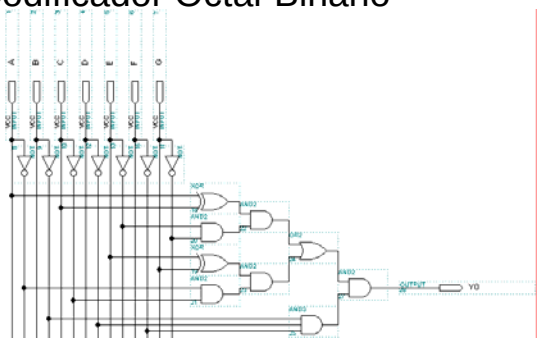
Y0 = B'D'F'(E'G'(AC' + A'C) + A'C'(EG' + E'G))

- Para Y1 e Y2 obtemos o mesmo:

Y1 = A'D'E'(F'G'(BC' + B'C) + B'C'(FG' + F'G))

Y2 = A'B'C'(F'G'(DE' + D'E) + D'E'(FG' + F'G))

Codificador Octal-Binário



Decodificador Binário- Octal

- Resultado binário muitas vezes deve ser convertido em outra base numérica, p.e., octal.
- Cada resultado binário deve ser indicado por uma saída independente

Tabela Verdade

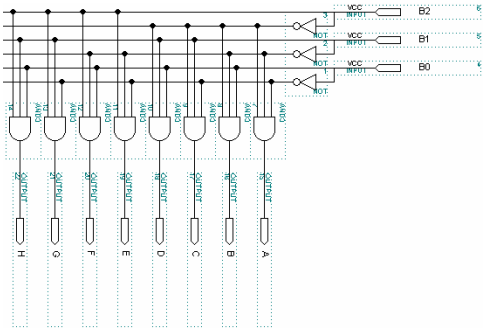
B2	B1	B0	A	B	C	D	E	F	G	H
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Decodificador Binário- Octal

- Determinação da função lógica do decodificador:
  - por observação, percebe-se que cada saída é representada por um único mintermos, assim:

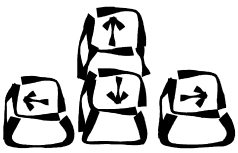
A = B2'.B1'.B0'
B = B2'.B1'.B0
C = B2'.B1.B0'
D = B2'.B1.B0
E = B2.B1'.B0'
F = B2.B1'.B0
G = B2.B1.B0'
H = B2.B1.B0

Decodificador Binário- Octal



Exercício

- Display de “flexas”
  - Controle de aparelho de som
    - Play
    - Reverse Play
    - Stop
    - Eject

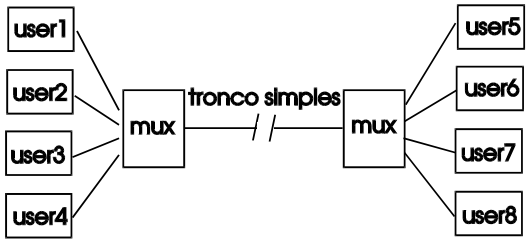


Display de 7 segmentos

- Método de representação hexadecimal
  - 0..9
  - A..F

Mux e Demux

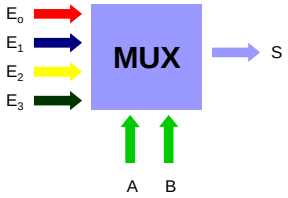
Multiplexação e Demultiplexação



Multiplexador (Mux)

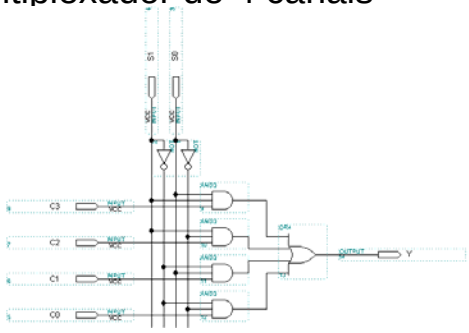
- Muitas vezes existe a necessidade de selecionarmos um “canal” de dados dentre vários para que suas informações sejam processadas ou transmitidas por outras partes de um sistema.
- Os multiplexadores são circuitos destinados a prover tal mecanismo de seleção utilizando log2(canais) entradas de controle (ou seleção).

Mux



A	B	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

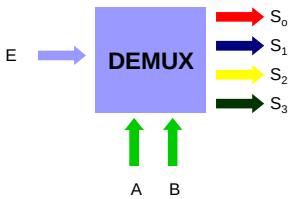
Multiplexador de 4 canais



Demultiplexador (Demux)

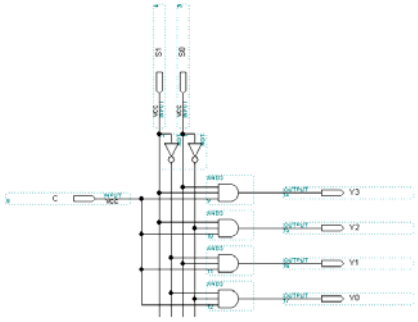
- Em outras situação precisamos distribuir os dados recebidas através de um canal de dados entre vários outros canais de modo para tais informações sejam processadas ou transmitidas para outras partes adequadas de um sistema.
- Os demultiplexadores são circuitos destinados a prover tal mecanismo de seleção utilizando log2(canais) entradas de controle (ou seleção).

Demux



A	**B**	S_0	S_1	S_2	S_3
0	**0**	E	0	0	0
0	**1**	0	E	0	0
1	**0**	0	0	E	0
1	**1**	0	0	0	E

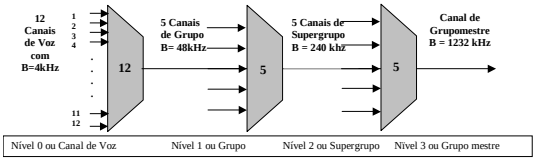
Demultiplexador de 4 canais



Técnicas de Multiplexação Exemplo

- FDM – Sistema de **Telefonia**
- Trajeto desconhecido
  - Diversos meios físicos – par/fibra/microondas/satélite
- Sucessivas multiplexações e reconstituições
  - Canal Grupo – 12 canais de voz
  - Canal Supergrupo – 5 canais de Grupo (60)
  - Canal Grupo Mestre – 5 canais de Supergrupo (300)
  - Canal de Super Grupo Mestre – 3 Grupo Meste (900)

Técnicas de Multiplexação Exemplo



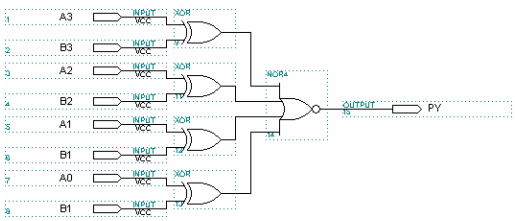
Outros circuitos

Úteis à detecção de erros em transmissões de dados

Detetor de Igualdade

- Muitas vezes é preciso comparar informações binárias para verificar sua igualdade.
- A operação lógica “Ou-Exclusivo (XOR) permite comparar dois bits de informação:
  - resultando 0 quando bits são iguais
  - resultando 1 quando bits são diferentes
- Associando duas ou mais portas XOR através de uma porta “Não-Ou” (NOR), obtemos um detetor de igualdade onde:
  - resultando 0 quando informação são diferentes
  - resultando 1 quando informação são iguais

Detetor de Igualdade



Gerador de Bit de Paridade

- Um problema grave na transmissão de informação digital é a possibilidade de interferência externa (ruído), provocando o aparecimento de erros de transmissão.
- Uma das formas de detecção de erros desta natureza é obtida através do controle de paridade.
- A paridade é um bit extra acrescentado ao conjunto de bits da informação transmitida, de modo que:
  - bit de paridade = 1 quando número total de bits 1 é par ou ímpar (conforme implementação).

Gerador de Bit de Paridade

- Paridade **par** é quando indicamos com 1 a ocorrência de um número par de 1's na informação.
- Paridade **ímpar** é quando indicamos com 1 a ocorrência de um número ímpar de 1's na informação.
- Na tabela verdade dada indicamos paridade ímpar.

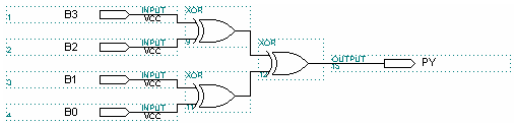
B3	B2	B1	B0	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Gerador de Bit de Paridade

- Observando o mapa de *Karnaugh* vemos não existir simplificação aparente, mas sem trabalha algebricamente podemos notar que entre quaisquer dois bits da entrada a saída corresponde a uma operação XOR.

B3B2 \ B1B0	00	01	11	10
00	0 0	4 1	12 0	8 1
01	1 1	5 0	13 1	9 0
11	3 0	7 1	15 0	11 1
10	2 1	6 0	14 1	10 0

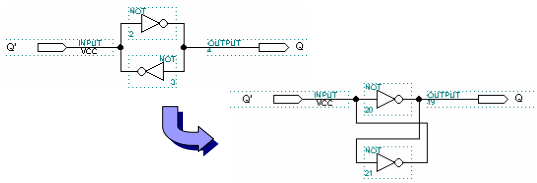
Gerador de Bit de Paridade



Circuitos sequenciais

Latches

- Circuito lógico especial capaz de manter um nível lógico em sua saída sem qualquer intervenção externa (i.e., sem a dependência da presença de valores em suas entradas).



Latches

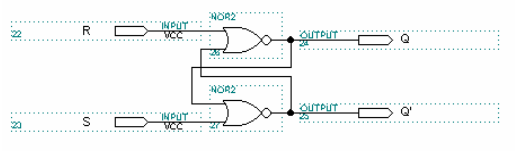
- Devido sua independência das entradas, um latch pode servir para 'armazenar' (registrar ou memorizar) um bit lógico.
- Assim, um conjunto de n latches pode constituir um registrador de n bits.
- Os dois estados do latch são chamados de:
  - set: aquele que o terminal chamado Q está em 1
  - reset: aquele que o terminal chamado Q está em 0 (este estado também é chamado de clear)
  - estado de Q' sempre complementar ao estado de Q

Latches

- Para armazenar um bit lógico num latch basta conectar (temporariamente) a entrada Q a um ponto externo que esteja no estado desejado.
- Quando a conexão é removida o latch *permanece* no estado no estado ajustado.

Latch com portas NOR

- É conveniente substituirmos os inversores por portas NOR.
- Os terminais adicionais de entrada servirão como entradas adicionais de controle que permitem novas formas de acesso ao latch.



Latch com portas NOR

Este latch se comporta assim:

- Se as entradas S e R são ajustadas em 0, o estado das saídas Q e Q' se mantém (0 ou 1) conforme o estado anterior;
- Se S = 1, então Q = 1 e Q' = 0, assim esta entrada é chamada *set*;
- Se R = 1, então Q = 0 e Q' = 1, assim esta entrada é chamada *reset*;
- Se R = S = 1, o estado de Q e Q' passam a depender da velocidade das portas, permitindo que Q e Q' se tornem não complementares (inconsistente = erro lógico).

S	R	Q	Q'
0	0	0/1	0/1
0	1	0	1
1	0	1	0
1	1	?	?

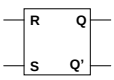
Flip-Flops

- São circuitos que possuem dois estados estáveis, ou seja, são circuitos biestáveis:
  - Flip = atirar ao alto ou movimento rápido
    Circuito assume estado lógico alto
  - Flop = queda brusca ou repentina
    Circuito assume estado lógico baixo
- Constituem os elementos básicos dos circuitos seqüênciais, ou seja, com eles podemos implementar registradores, contadores etc.
- Podem ser obtidos através da ligação adequada de portas lógicas

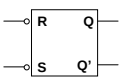
Flip-Flops RS

- Os latches construídos com portas lógicas NOR e NAND também são conhecidos como flip-flops RS devido ao comportamento de suas saídas.
- Símbolos:

FlipFlop RS (NOR)

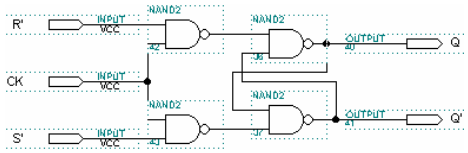


FlipFlop RS (NAND)



Flip-Flops RS Síncrono

- Tem como característica um terceira entrada denominada pulso de controle (*clock* ou CK) agregada a um estágio de entrada adicional.
- O clock faz com que o flip-flop RS atualize seus estados.

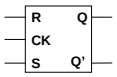


Flip-Flops RS Síncrono

- Com o clock em nível zero (CK=0), as saídas anteriores são mantidas.
- Com o clock em nível um (CK=1), o flip-flop RS síncrono opera como um flip-flop RS.

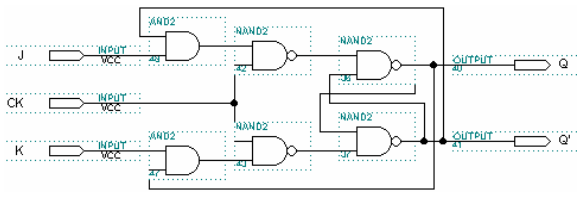
CK	S	R	Q	Q'
0	X	X	Qa	Qa
1	0	0	0/1	0/1
1	0	1	0	1
1	1	0	1	0
1	1	1	?	?

FlipFlop RS Sinc



Flip-Flop JK

- Tipo de flip-flop RS aprimorado, onde o erro lógico foi eliminado.



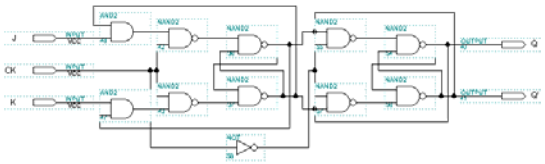
Flip-Flop JK

- Enquanto CK=1 e J=K=1, a complementação da saída e a realimentação provocarão sucessivas complementações (oscilações) da saída.
- Uso deste estado ainda inviável.

CK	J	K	Q	Q'
0	X	X	Qa	Qa'
1	0	0	Qa	Qa'
1	0	1	0	1
1	1	0	1	0
1	1	1	*	*

Flip-Flop JK Master-Slave

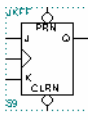
- Para eliminar a oscilação do flip-flop JK, foram combinados dois flip-flops RS como no circuito a seguir, denominado flip-flop JK Master-Slave (Mestre-Escravo).



Flip-Flop JK Master-Slave

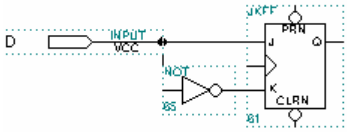
- O flip-flop JK M-S é um circuito sensível a borda de descida do pulso de clock CK, ou seja, somente na transição de 1 para 0 verifica-se sua tabela verdade.

CK	J	K	Q	Q'
0	X	X	Qa	Qa'
1->0	0	0	Qa	Qa'
1->0	0	1	0	1
1->0	1	0	1	0
1->0	1	1	Qa'	Qa



Flip-Flop D (JK tipo D)

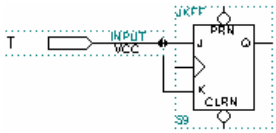
- A partir de um flip-flop JK, podemos construir um tipo particular de flip-flop através da conexão ilustrada abaixo, obtendo um flip-flop tipo D.



CK	D	Q	Q'
0	X	Qa	Qa'
1->0	0	0	1
1->0	1	1	1

Flip-Flop T (JK tipo T)

- A partir de um flip-flop JK, podemos construir um outro tipo particular de flip-flop através da união de suas entradas J e K (ilustrada abaixo), obtendo um flip-flop tipo T.



CK	T	Q	Q'
0	X	Qa	Qa'
1->0	0	Qa	Qa'
1->0	1	Qa'	Qa

Aplicações dos Flip-Flops

- Com a utilização dos flip-flops, podemo construir circuitos:
  - divisores de freqüência;
  - registradores de deslocamento unidirecionais e bidirecionais e
  - contadores assíncronos e síncronos.
- Comercialmente temos os CIs:
  - TTL 7476 (dual JK FF /sensível a borda 1->0)
  - CMOS 4027 (dual JK FF/sensível a borda 0->1)

Contadores e Registradores

- No simulador!

Referências Bibliográficas

- IDOETA, Ivan Valeije; CAPUANO, Francisco Gabriel. **Elementos de Eletrônica Digital**. São Paulo: Editora Érica, 1998.
- JANDL, Peter Jr. **Slides de aulas**. São Paulo: Universidade São Francisco. Disponível em http://docente.saofrancisco.edu.br/peter/aulas/cd/ .
- NETTO, Luiz Ferraz. **Feira de Ciências – O Imperdível**. Disponível em www.feiradeciencias.com.br.
- **Sebenta Multimédia – Análise de Circuitos Eléctricos**. Instituto Superior Técnico (material eletrônico).
- SILVA, Ricardo Pereira e. **Eletrônica Básica: um enfoque voltado à Informática**. Florianópolis: Ed. da UFSC, 1995.