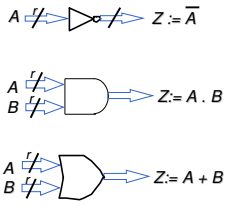
Operações Lógicas

Aula 8
(Operações e Aplicações c/MUX e DEMUX)
CIn-UFPE

Aplicações lógicas

- Operações lógicas aplicadas a vetores são simplesmente uma extensão das operações lógicas como aquelas aplicadas a escalares.
- Exemplo:



Operações Lógicas com vetores

- Possíveis formas de Operação binárias

Oper. 1	Oper. 2	Resultado	Comentário
– Escalar	Escalar	Escalar	Operação padrão
– Escalar	vetor	Escalar	Não existe
– Vetor	Escalar	Escalar	Não existe
– Escalar	Vetor	Vetor	Modo Misto
– Vetor	Escalar	Vetor	
– Vetor	Vetor	Escalar	Operação relacional
– Vetor	Vetor	Vetor	Operação vetorial

Operações Lógicas com vetores

- Operações escalares podem ser estendidas a vetores. estas operações são importantes no nível de sistema.
- 1. Operação lógica com vetores
- 2. Operações relacionais
- 3. Operações no modo misto

- **Operações lógicas com vetores**

Operação	Representação	significado
NO	Z:= $\overline{X}$	$z_i = \overline{x_i}$
AND	Z:= X .Y	$z_i = x_i . y_i$
OR	Z:= X +Y	$z_i = x_i + y_i$
OR-Exclusivo	Z:= X ⊕ Y	$z_i = x_i \oplus y_i$
Coincidência	Z:= X⊙ Y	$z_i = x_i \odot y_i$
NAND	Z:= $\overline{X . Y}$	$z_i = \overline{x_i . y_i}$
NOR	Z:= $\overline{X + Y}$	$z_i = \overline{x_i + y_i}$

Operações relacionais com vetores

- Em várias tarefas de processamento é necessário verificar ou melhor compara dois tipos de informação e tomar alguma decisão lógica. Se a decisão é verdadeira (true) ou falsa(false), em função da comparação solicitada.
- Matematicamente isto é feito através de um operador relacional. Estes operadores operam sobre vetores e produzem um resultado escalar.

  z := X <relação> Y
- Operações relacionais básicas:

< menor que	> maior que
≤ menor ou igual a	≥ maior ou igual a
= igual	≠ não igual

Aplicações - operações relacionais

- Na realização de operações relacionais deve-se tomar cuidado com o domínio dos operandos. Os operandos devem pertencer ao mesmo domínio (inteiros, ASCII, EBCDIC, ...)
- Exemplo

  x := A < B

  Para realizarmos tal operação devemos decidir qual código (domínio) será usado para representar a informação correspondente aos vetores A e B. Por exemplo A e B são números binários positivos.

  Considere: **A:= [a_{r-1}, a_{r-2}, a_0] e B:= [b_{r-1}, b_{r-2}, b_0]**

  A idéia é começar a avaliar os dois vetores da direita para esquerda até encontrarmos dois bits diferentes. Se a_i = 0 e b_i = 1, então sabemos que A <B e fazemos um certo d_i =1. Este d_i funciona como um bit auxiliar para identificar a relação A < B.

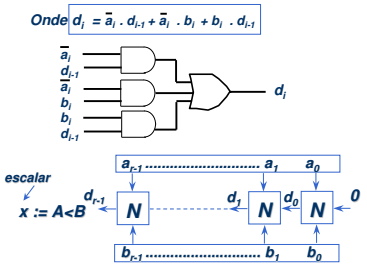
Aplicações - operações relacionais

- A tabela verdade para o sub-circuito que gera esta função pode ser dada por:

$x := A<B$

Onde $d_i = \bar{a}_i \cdot d_{i-1} + \bar{a}_i \cdot b_i + b_i \cdot d_{i-1}$

a_i	b_i	d_{i-1}	d_i
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



Operações no modo misto

- Existem operações onde nós desejamos combinar um escalar x com um vetor para formar um outro vetor.
- Para tal convencionamos que sendo x um escalar e Y e Z vetores, façamos:
  <operação> no modo misto se
  1. <operação> é uma das operações básicas
  2. Z := x <operação> Y implica que $z_i := x$ <operação> y_i, para i = 1,2,...,r
- Algumas operações no modo misto

AND	Z:= x.Y	$z_i := x \cdot y_i$ i = 1,2, ...r	Z:= [0] se x=0; Y se x=1
OR	Z := x+Y	$z_i := x + y_i$ i = 1,2, ...r	Z:= Y se x=0; [1,1,...] se x=1

Aplicações - modo misto

- Modo Misto

  Modo misto é importante quando queremos selecionar um vetor de um grupo de valores.

  Suponha um circuito com três entradas W, X e Y e desejamos controlar qual vetor nós devemos usar na computação. Este é o princípio da multiplexação.

  $Z := (a_1 \cdot W) + (a_2 \cdot X) + (a_3 \cdot Y)$

Onde:

1. se $a_1 = 1; a_2 = 0; a_3 = 0;$ $Z := (1 \cdot W) + (0 \cdot X) + (0 \cdot Y) := W$
2. se $a_1 = 0; a_2 = 1; a_3 = 0;$ $Z := (0 \cdot W) + (1 \cdot X) + (0 \cdot Y) := X$
3. se $a_1 = 0; a_2 = 0; a_3 = 1;$ $Z := (0 \cdot W) + (0 \cdot X) + (1 \cdot Y) := Y$

Estas idéias podem ser expandidas em expressões relacionais ou expressões lógicas para computar valores de a nas expressões acima. Estes escalares (a's) são chamados variáveis de controle.

Aplicações

Multiplexadores/Decodificadores

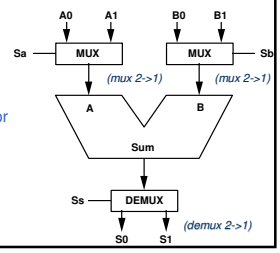
Multiplexadores e Decodificadores

- **Exemplo de conexão entre Multiplexadores e Demultiplexadores**

Múltiplas fontes de entrada

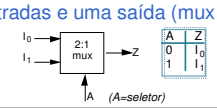
Somador

Múltiplas saídas



Multiplexadores/Seletores

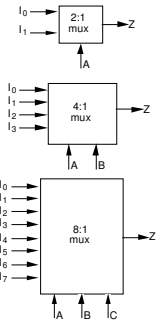
- Conceito geral
  - A seleção de 2^n entradas é feitas através de n linhas de controle que endereçam cada uma destas entradas para a saída.
  - Cada entrada possui um endereço determinado, o qual é, em geral, igual a um minitermo.
  - A saída recebe o valor da entrada correspondente ao endereço escolhido.
- Exemplo de um multiplexador/selector de duas entradas e uma saída (mux 2->1)



A	Z
0	I_0
1	I_1

$Z = A' I_0 + A I_1$

Multiplexadores/Seletores



$Z = A' I_0 + A I_1$

$Z = A' B' I_0 + A' B I_1 + A B' I_2 + A B I_3$

$Z = A' B' C' I_0 + A' B' C I_1 + A' B C' I_2 + A' B C I_3 + A B' C' I_4 + A B' C I_5 + A B C' I_6 + A B C I_7$

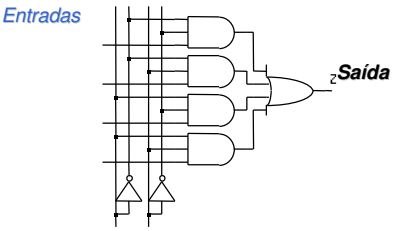
A seleção em um multiplexador pode ser dado em geral por:

$Z = \sum_{k=0}^{2^n-1} m_k I_k$

m = minitermo
I = entrada

Multiplexadores/Seletores

- Implementação de um multiplexador 4:1



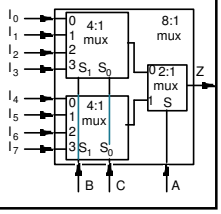
Multiplexadores/Seletores

- Implementação de grandes multiplexadores a partir de pequenos multiplexadores
- Implementação de um multiplexador 8:1 a partir de multiplexadores 4:1 e 2:1

Os controles B e C escolhem uma das entradas de I_0 a I_3 e ao mesmo tempo de I_4 a I_7 entre os muxs 4:1.

O controle A estabelece a saída Z através da seleção no mux 2:1, cujas entradas são saídas dos muxs 4:1

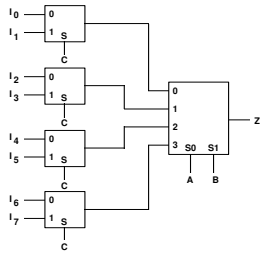
A	B	C	Z
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



Multiplexadores/Seletores

- Exemplo de multiplexador 8:1 a partir de muxs 2:1 e 4:1

A	B	C	Z
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



Multiplexadores/Seletores

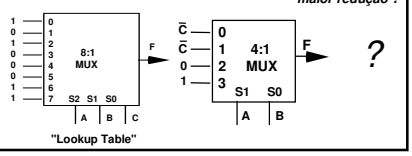
- Multiplexadores 2^n:1 podem implementar qualquer função de n variáveis, com n-1 variáveis de controle . As demais variáveis serão entrada para o mux.

$F(A,B,C) = m0 + m2 + m6 + m7$
$= A' B' C' + A' B C' + A B C' + A B C$
$= A' B' (C') + A' B (C') + A B' (0) + A B (1)$

Possíveis implementações

É possível maior redução ?

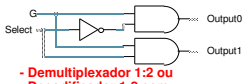
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



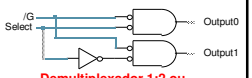
"Lookup Table"

Decodificadores/Demultiplexadores

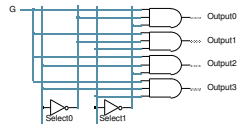
- Alternativas de Implementação



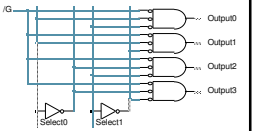
- Demultiplexador 1:2 ou
- Decodificador 1:2 com habilitação ativa em alta ("1")



- Demultiplexador 1:2 ou
- Decodificador 1:2 com habilitação ativa em baixa ("0")



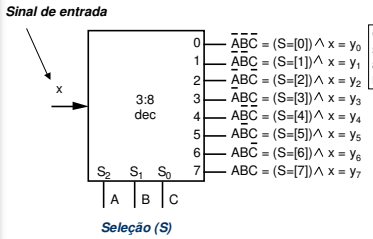
- Demultiplexador 2:4 ou
- Decodificador 2:4 com habilitação em alta ("1")



- Demultiplexador 2:4 ou
- Decodificador 2:4 com habilitação ativa em baixa ("0")

Decodificadores/Demultiplexadores

- Demultiplexador/Decodificador 3:8 como bloco lógico

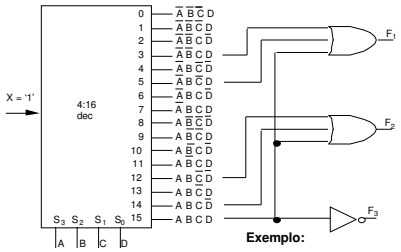


O decodificador gera saídas equivalentes a codificação dos minitermos

$y_i = m_i \wedge x$

Decodificadores/demultiplexadores

- Decodificadores como gerador de funções



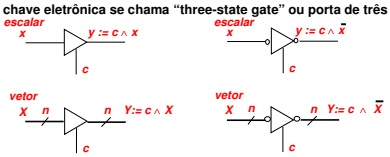
Exemplo:

F1 = A' B C' D + A' B' C D + A B C D
F2 = A B C' D' + A B C
F3 = (A' + B' + C' + D')

Elementos lógicos Three-state

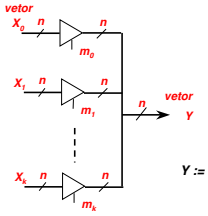
- **Em um multiplexador as porta ANDs são usada para controlar o sinal a ser transmitido para a saída.**
- **A porta OR garante que as portas AND ficarão isoladas entre si como se tivéssemos uma chave.**

- **Este chaveamento seria simplificado se nós substituíssemos as portas ANDs e ORs exatamente por uma chave eletrônica equivalente à mecânica acima.**
- **Esta chave eletrônica se chama "three-state gate" ou porta de três estado.**



Elementos lógicos Three-state

- Uso de elementos three-state para implementar um multiplexador



A diferença entre uma porta 3-state e uma porta AND é que o sinal na porta 3-state é eletronicamente desconectado quando o controle desabilita a porta.

$Y := (m_0 \wedge X_0) \quad (m_0 \wedge X_0) \ldots\ldots\ldots \quad (m_k \wedge X_k)$