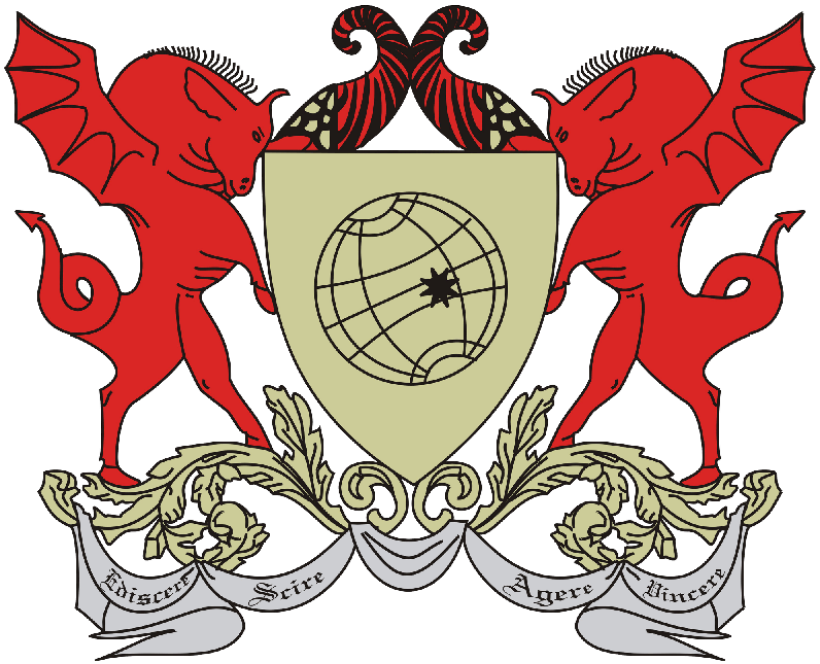


CCF 251 – Introdução aos Sistemas Lógicos

Aula 04 – Tecnologias lógica combinacional

Prof. José Augusto Nacif – jnacif@ufv.br

Tecnologias lógica combinacional

- Portas padrão
  - Circuitos integrados com poucas portas (10-100)
  - Disponíveis a partir da década de 60
  - Utilizados na prática de laboratório (74xx)
- ROMs e PLAs/PALs
  - ROMs para representar tabelas verdade, mas lentas
  - PLAs e PALs são soluções de menor custo
    - Fusíveis queimados para definir ligações de portas AND/OR
- Circuitos integrados de aplicação específica
  - *Gate array* generalizou PLAs e PALs
  - Células padrão permitiu desenvolvimento de circuitos mais complexos

Lógica aleatória

- Transistores integrados dentro de portas lógicas (1960s)
- Catálogo das portas comuns (1970s)
  - Livro Texas Instruments Logic Data – A bíblia amarela
  - Todos os pacotes comuns listados e caracterizados (atrasos, energia)
  - Pacotes típicos:
    - IC de 14-pin: 6-inversores, 4 portas NAND, 4 portas XOR
- Hoje, poucas partes são ainda utilizadas
- Contudo, partes das bibliotecas existem para projetos de chip
  - Projetistas já reutilizam portas lógicas caracterizadas em chips
  - Mesma razão que antes
  - A diferença está nas partes que não existem um inventário físico – criado se necessário

Lógica aleatória

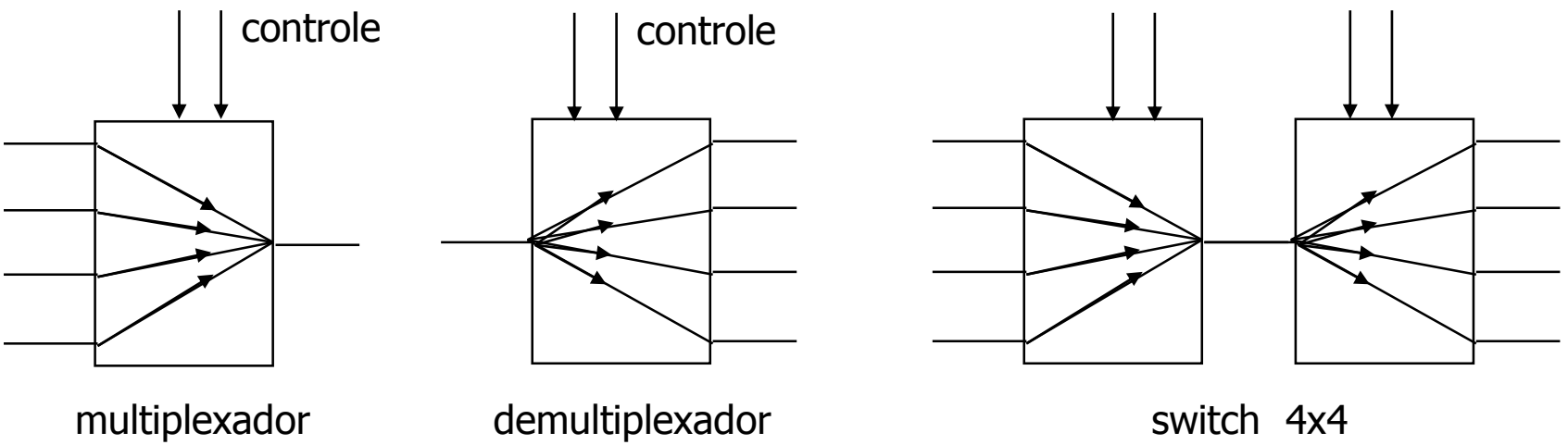
- Também é difícil imaginar exatamente a saída de qual porta usar
  - Mapa a partir da lógica de redes NAND/NOR
  - Determinar o número mínimo de pacotes
    - Leve modificação nas funções lógicas, pode diminuir o custo
- Alterações dificultam a implementação
  - Precisa religar as partes
  - Pode precisar de partes novas
  - Projeto com partes sobressalentes (menos inversores extras e portas em toda placa)

Lógica regular

- Necessário fazer projetos mais rápidos
- Necessário fazer alterações na engenharia de modo mais fácil
- Simplificar para projetistas entenderem e mapear as funcionalidades
  - Difícil pensar em termos de portas específicas
  - Melhor pensar em termos de um grande bloco de multi propósitos

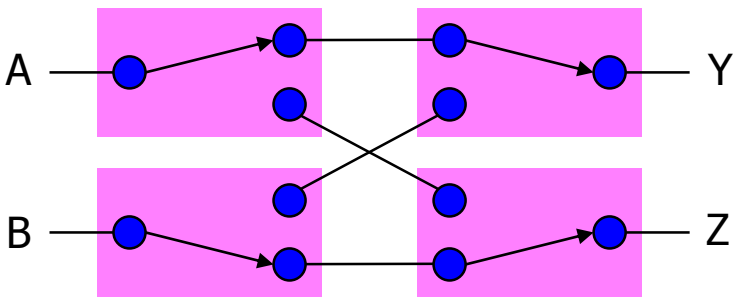
Fazendo conexões

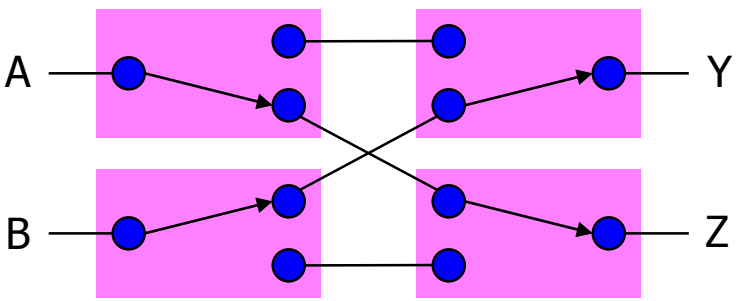
- Conexões diretas ponto a ponto entre portas
  - Wires (fios) são vistos de modo distante
- Rotear uma de várias entradas em uma única saída --- Multiplexador
- Rotear uma única entrada para uma ou várias saídas --- demultiplexador



Mux e demux

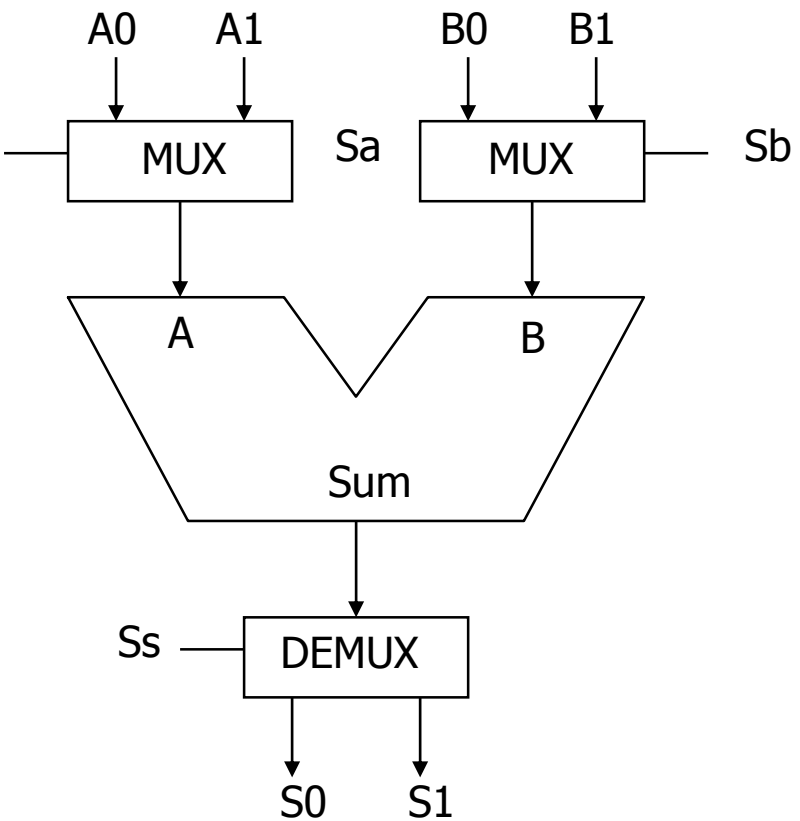
- Implementação chave dos multiplexadores e demultiplexadores
  - Podem ser compostos para se fazer um tamanho arbitrário de redes chaveadas
  - Usado para implementar interconexões de múltiplas origens/ múltiplos destinos





Mux e demux

- Uso de multiplexadores/demultiplixadores em conexões multi pontos



Múltiplas entradas de origem

Múltiplas saídas de destino

Multiplexadores/seletores

- Multiplexadores/seletores: Conceito geral
  - 2^n entradas de dados, n entradas de controle (chamadas “seletores"), 1 saída
  - Usado para conectar 2^n pontos para um único ponto
  - O padrão de sinal de controle constitui um índice binário de entrada ligada à saída

$$Z = A' I_0 + A I_1$$

A	Z
0	I_0
1	I_1

Forma funcional

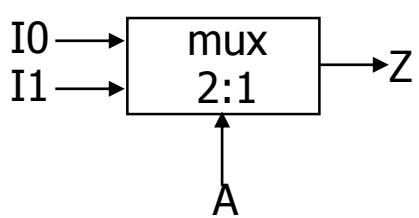
Forma lógica

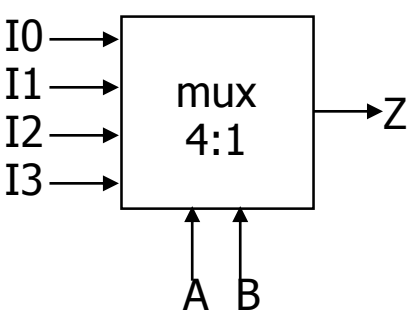
I_1	I_0	A	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

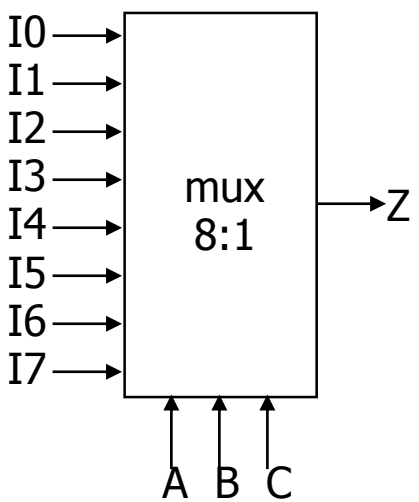
Duas formas alternativas
Tabela verdade para um Mux 2:1

Multiplexadores/seletores

- mux 2:1: $Z = A'I_0 + AI_1$
- mux 4:1: $Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$
- mux 8:1: $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$
- Em geral: $Z = \sum_{k=0}^{2^n - 1} (m_k I_k)$
  - Mintermo forma curta para um Mux 2^n:1

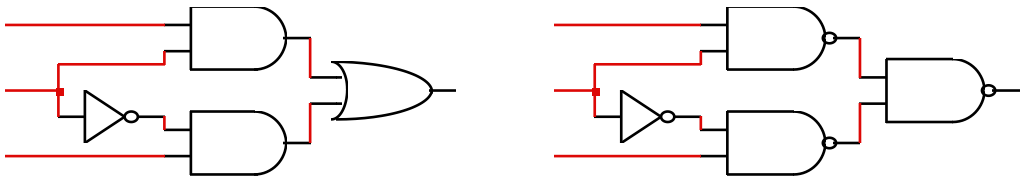




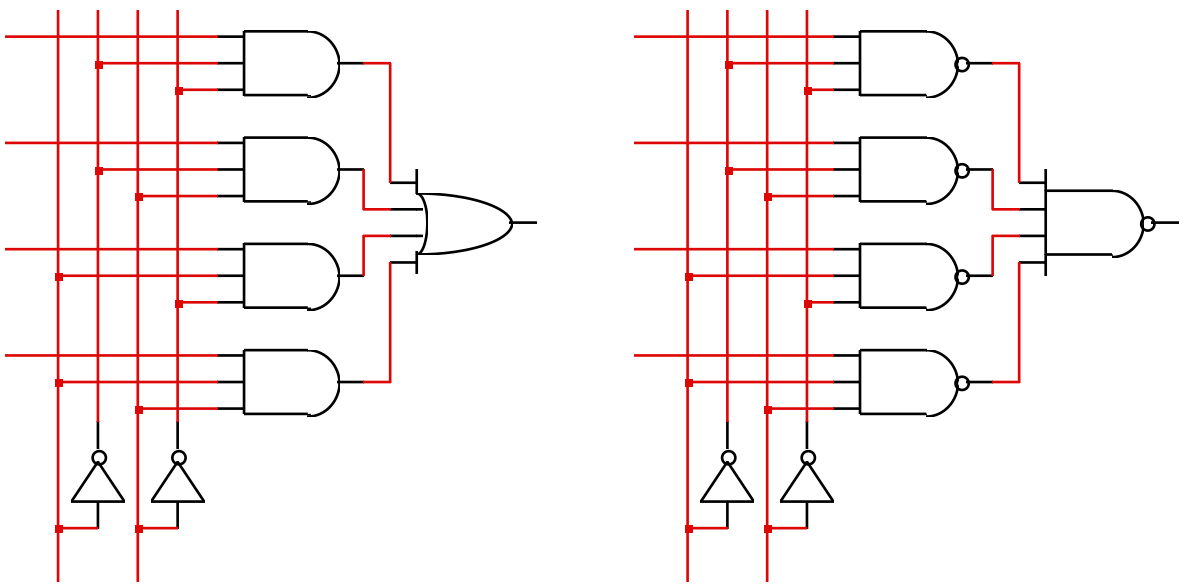


Implementação porta de nível multiplexadores

- mux 2:1

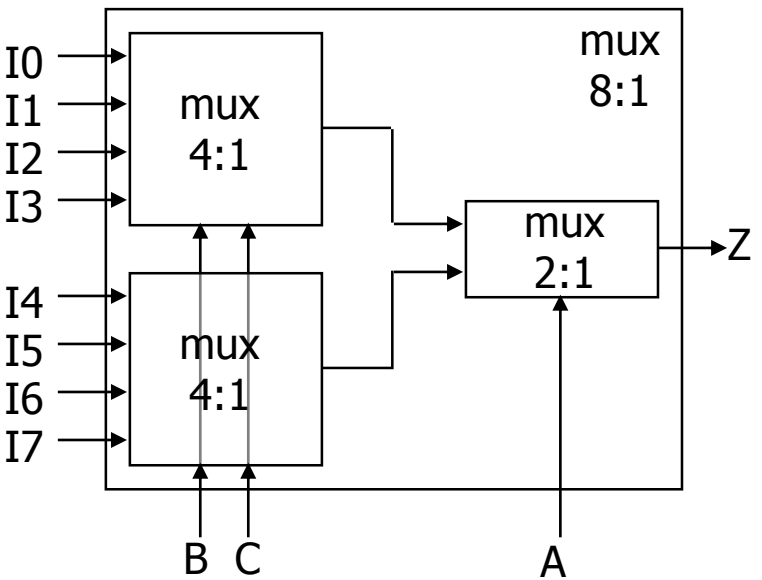


- mux 4:1



Cascateamento multiplexadores

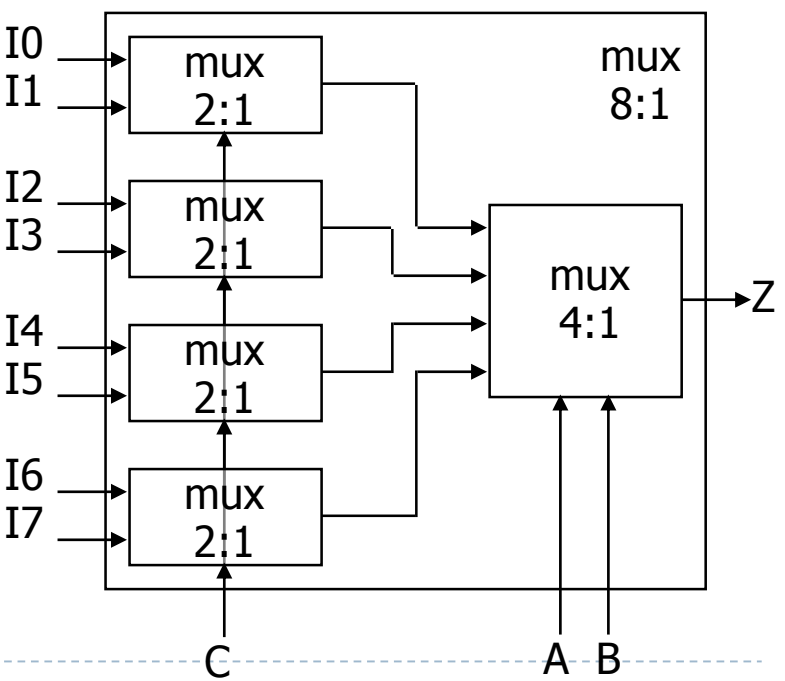
- Grandes multiplexadores podem ser construídos com cascateamentos menores



Sinais de controle B e C. Simultaneamente se escolhe um dos I0, I1, I2, I3 e um dos I4, I5, I6, I7

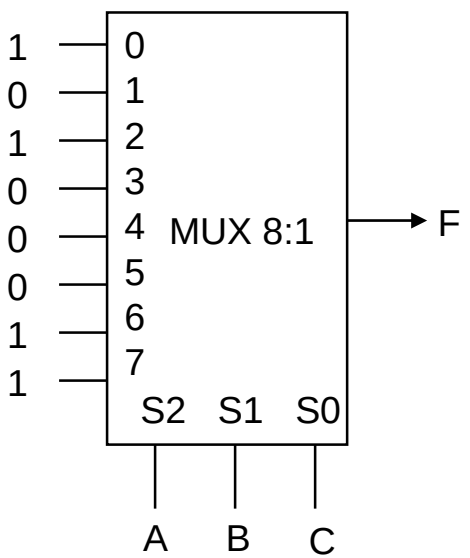
Sinal de controle A escolhe qual das saídas do Mux's alta ou baixa para a porta e para Z

Implementação alternativa



Multiplexadores como lógica de próposito geral

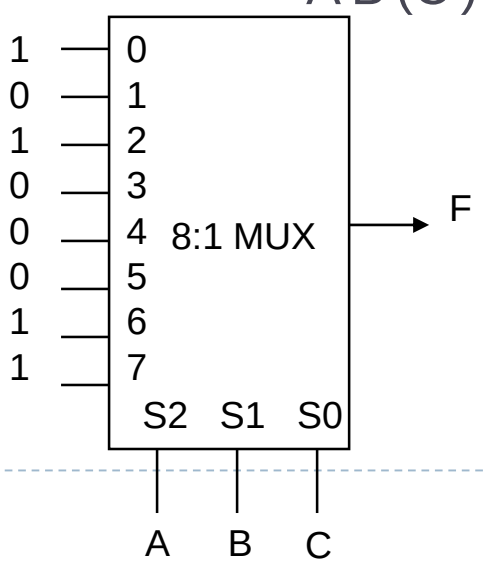
- Um multiplexador 2^n:1 pode implementar qualquer função de n variáveis
  - Com as variáveis utilizadas como entradas de controle e
  - As entradas de dados vinculadas em 0 ou 1
  - Em essência, uma tabela de pesquisa
- Exemplo:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC
    = A'B'C'(1) + A'B'C(0)
    + A'BC'(1) + A'BC(0)
    + AB'C'(0) + AB'C(0)
    + ABC'(1) + ABC(1)



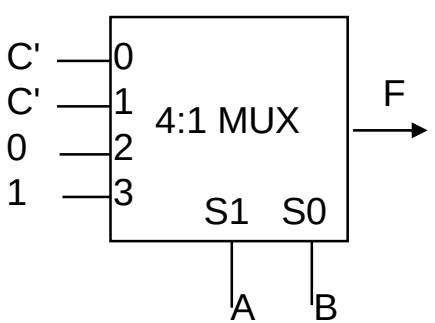
$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

Multiplexadores como lógica de próposito geral

- Um multiplexador 2^n:1 pode implementar qualquer função de n variáveis
  - Com n-1 variáveis utilizadas como entradas de controle e
  - As entradas de dados vinculadas à última variável ou seu complemento
- Exemplo:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC
    = A'B'(C') + A'B(C') + AB'(0) + AB(1)

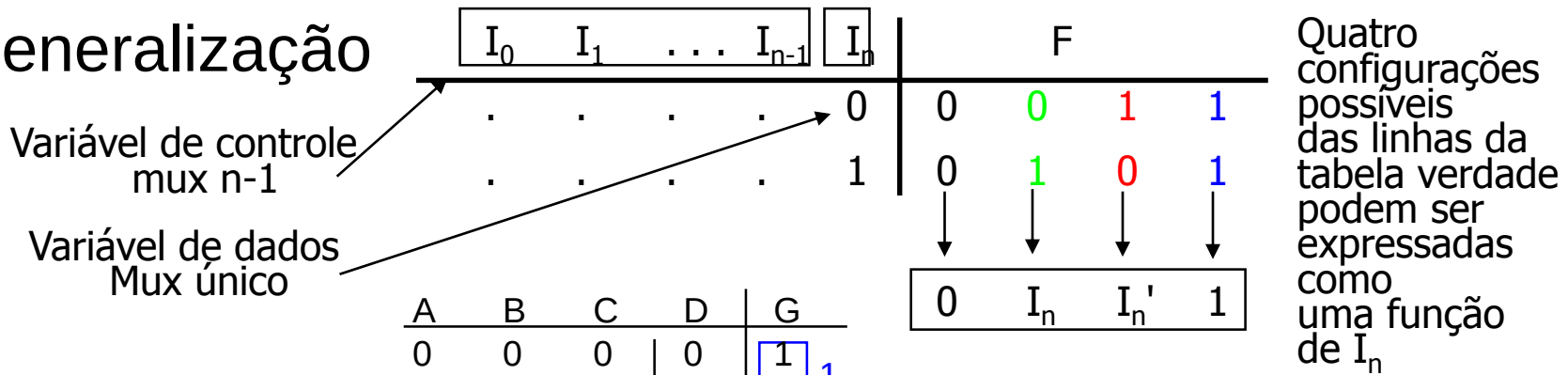


A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	



Multiplexadores como lógica de próposito geral

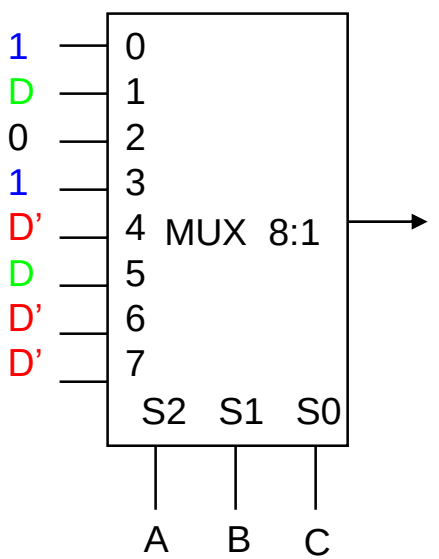
- Generalização



- Exemplo: G(A,B,C,D) pode ser realizada por um MUX 8:1

Escolher A,B,C como variáveis de controle

A	B	C	D	G	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	D
0	0	1	1	1	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	1	D'
1	0	0	1	0	
1	0	1	0	0	D
1	0	1	1	1	
1	1	0	0	1	D'
1	1	0	1	0	
1	1	1	0	1	D'
1	1	1	1	0	



Atividade

- Perceba F = B’CD’ + ABC’ com um multiplexador 4:1 e um mínimo de outras portas:

Demultiplexadores/decodificadores

- Decodificadores/demultiplicadores: conceito geral
  - Entrada de dados única, n entradas de controle, 2^n saídas
  - Entradas de controle (chamadas “seletores” (S)) representam um índice binário de saída para qual entrada está conectada
  - Entrada de dados usualmente chamada de “enable” (G)

Decodificador 1:2

O0 = G • S′
O1 = G • S

Decodificador 2:4

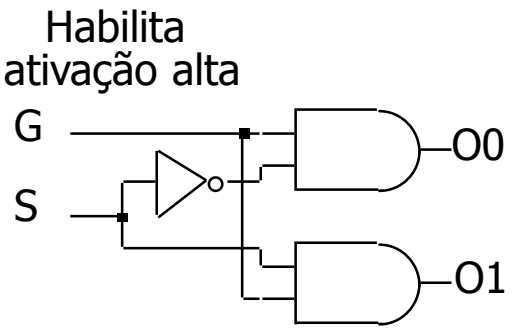
O0 = G • S1′ • S0′
O1 = G • S1′ • S0
O2 = G • S1 • S0′
O3 = G • S1 • S0

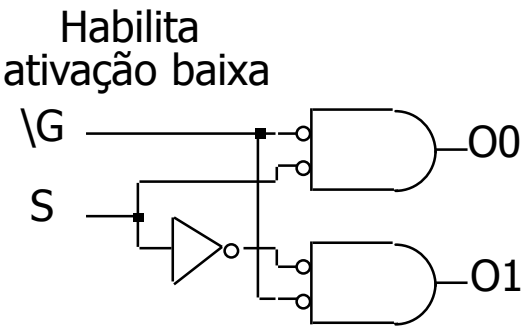
Decodificador 3:8

O0 = G • S2′ • S1′ • S0′
O1 = G • S2′ • S1′ • S0
O2 = G • S2′ • S1 • S0′
O3 = G • S2′ • S1 • S0
O4 = G • S2 • S1′ • S0′
O5 = G • S2 • S1′ • S0
O6 = G • S2 • S1 • S0′
O7 = G • S2 • S1 • S0

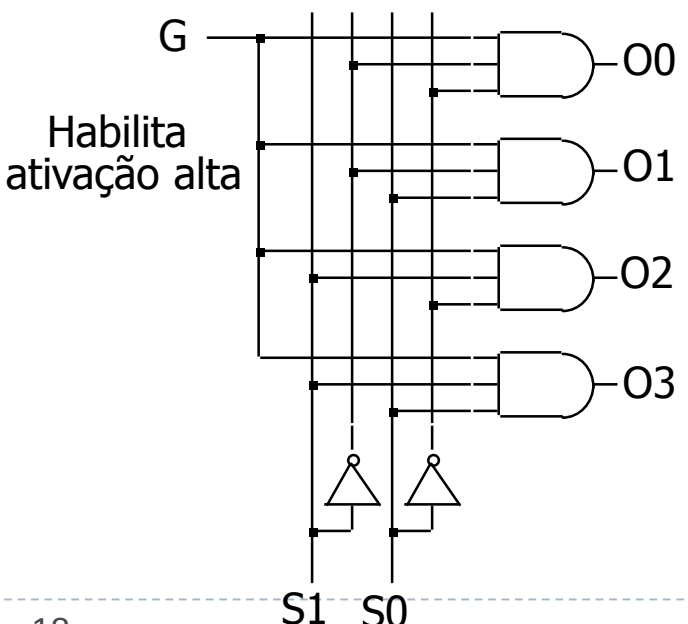
Implementação nível de porta de demultiplexadores

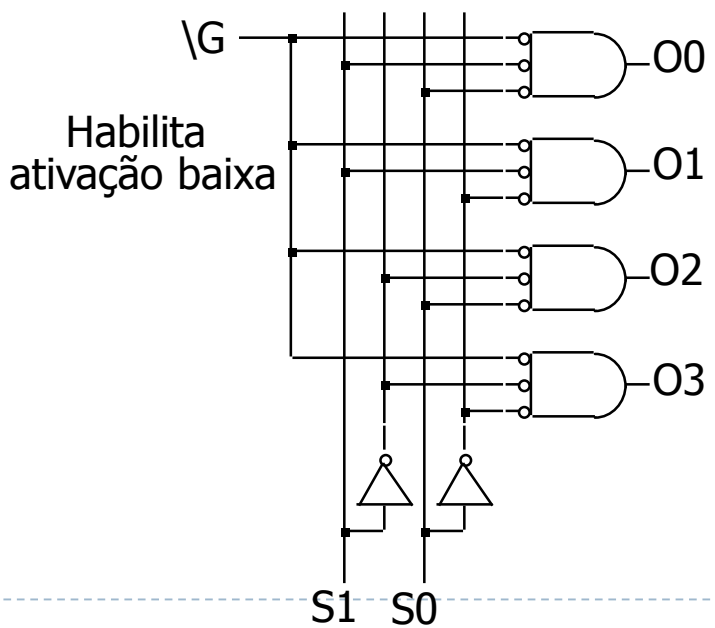
- Decodificador 1:2





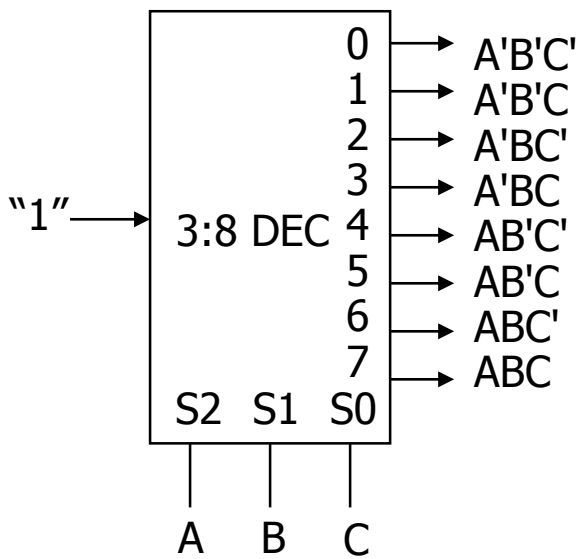
- Decodificador 2:4





Demultiplexadores como lógica de propósito geral

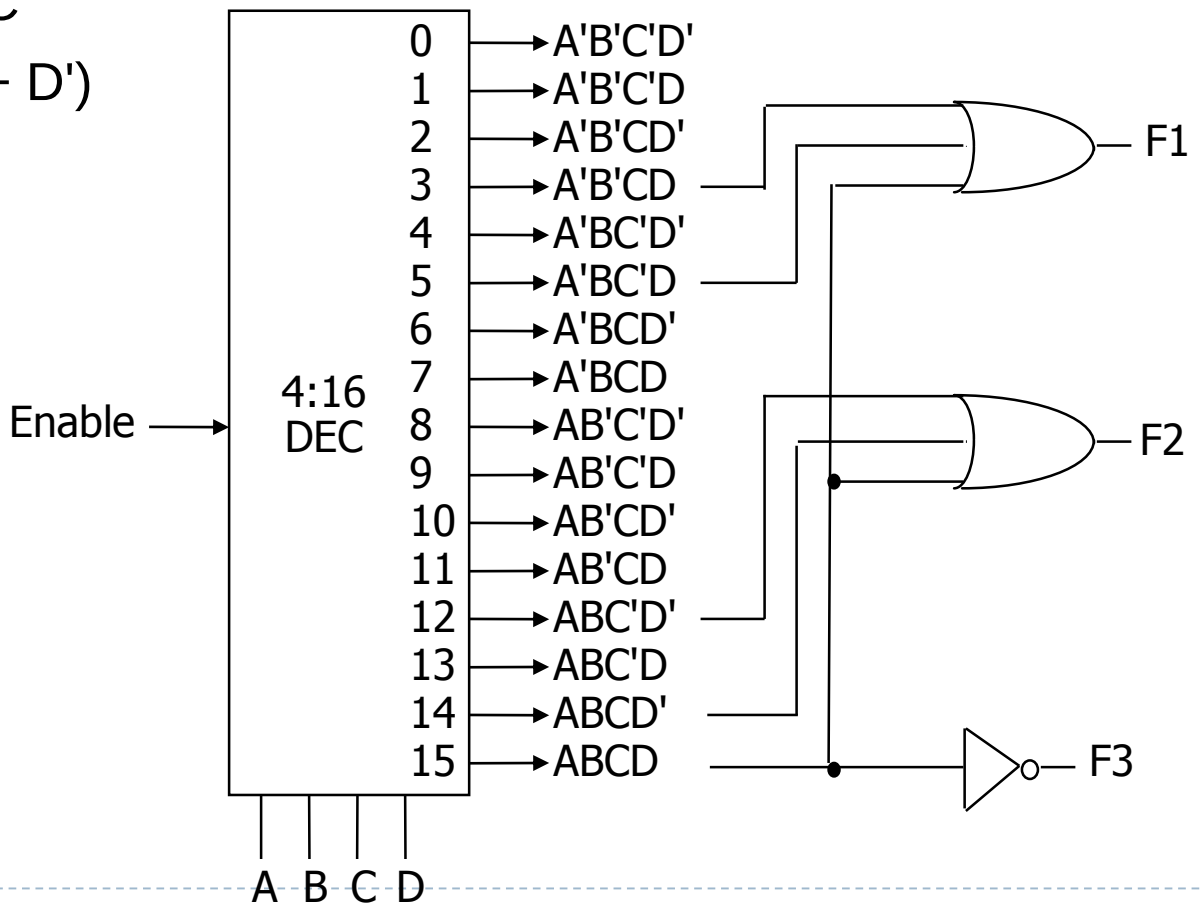
- Um decodificador n:2^n pode implementar qualquer função de n variáveis
  - Com as variáveis utilizadas como entradas de controle
  - E entradas habilitadas e vinculadas em 1
  - As abordagens mintermos resumidas para formar a função



Demultiplexador gera abordagens mintermo baseado em sinais de controle ("decodifica" sinais de controle)

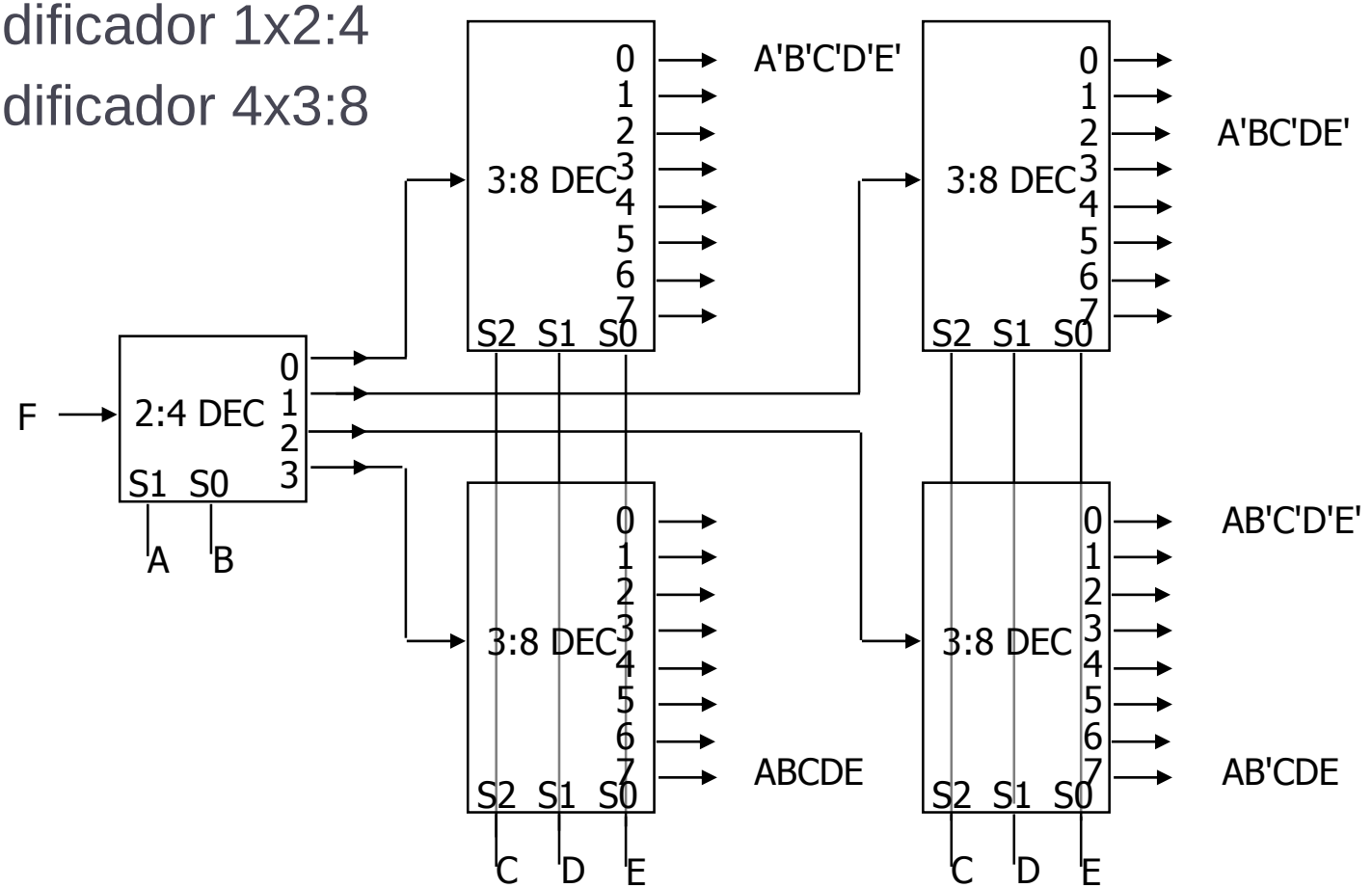
Demultiplexadores como lógica de propósito geral

- F1 = A'BC'D + A'B'CD + ABCD
- F2 = ABC'D' + ABC
- F3 = (A' + B' + C' + D')



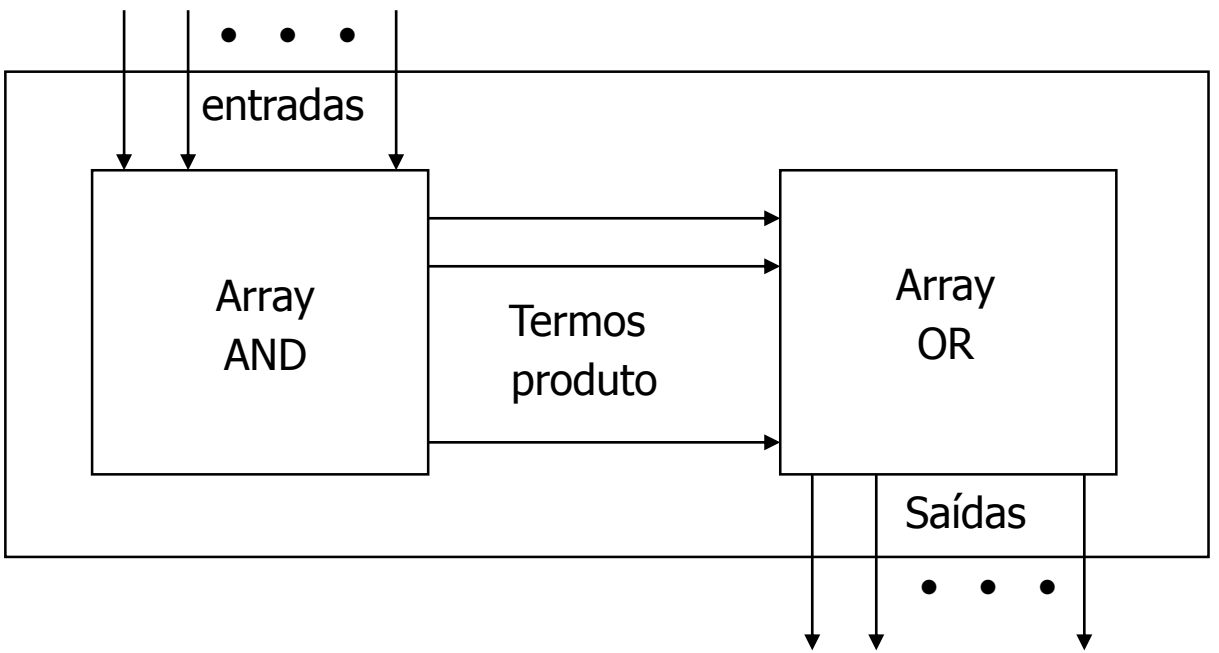
Cascateando decodificadores

- Decodificador 5:32
  - Decodificador 1x2:4
  - Decodificador 4x3:8



Array lógico programável

- Bloco pré-fabricado construído de muitas portas AND/OR
  - Na realidade NOR ou NAND
  - "personalizado" fazendo/dividindo conexões entre as portas
  - Diagrama de bloco *array* programável para formar a soma de produtos



Conceito

- Compartilhar termos produto entre saídas

Exemplo:

F0 = A + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A

Matriz personalizada

Termo produto	entradas A	B	C	saídas F0	F1	F2	F3
AB	1	1	–	0	1	1	0
B'C	–	0	1	0	0	0	1
AC'	1	–	0	0	1	0	0
B'C'	–	0	0	1	0	1	0
A	1	–	–	1	0	0	1

Lado entrada:

1 = não complementa o termo
0 = complementa o termo
– = Não participa

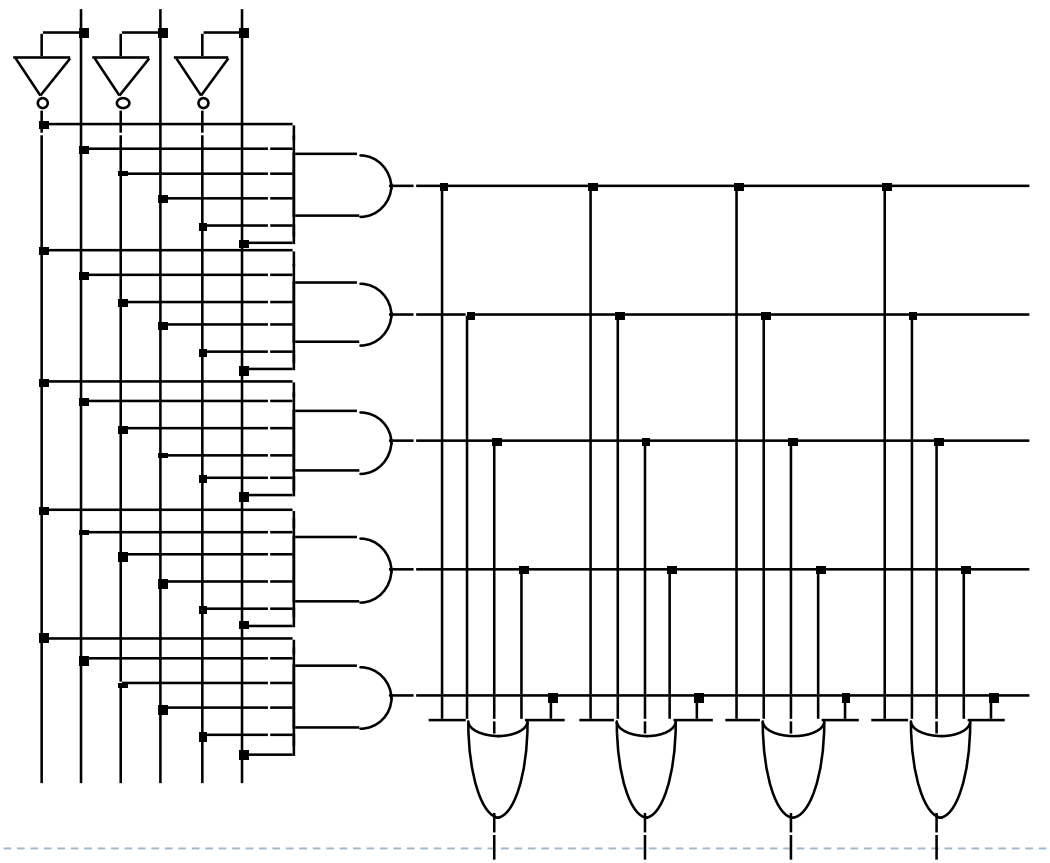
Lado saída:

1 = termo conectado com a saída
0 = nenhuma conexão com a saída

Reutilização de termos

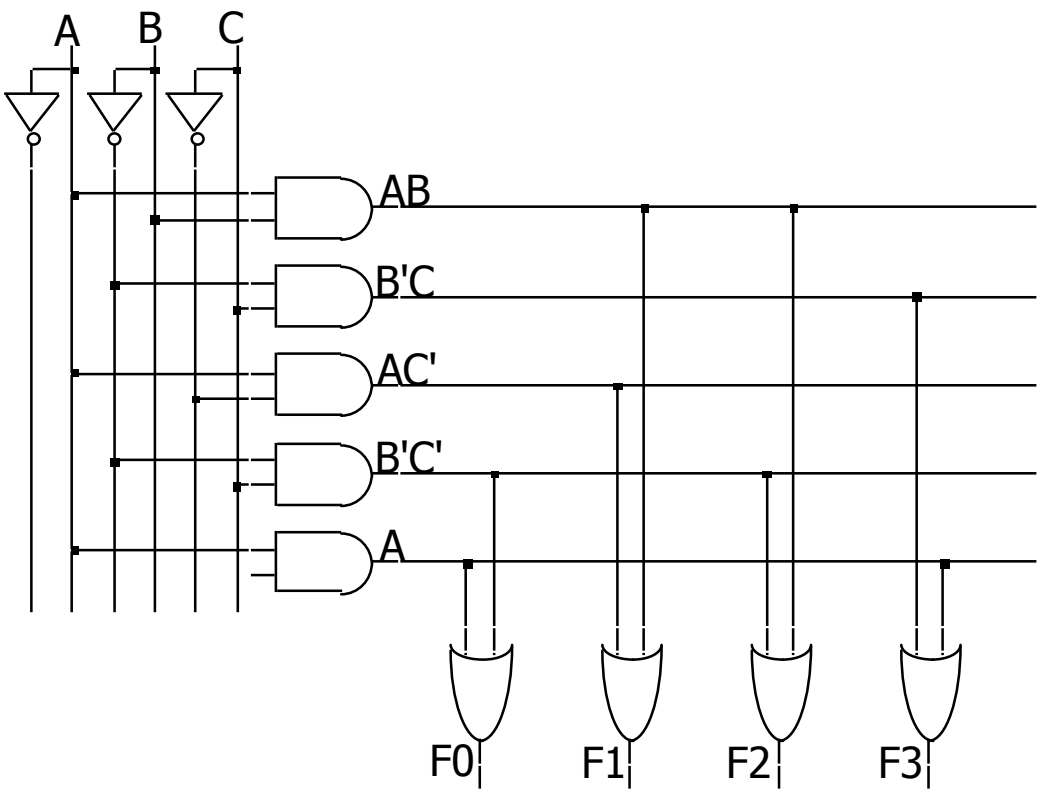
Antes da programação

- Todas as possíveis conexões estão disponíveis antes da “programação”
  - Na realidade, todas as portas AND e OR são NANDs



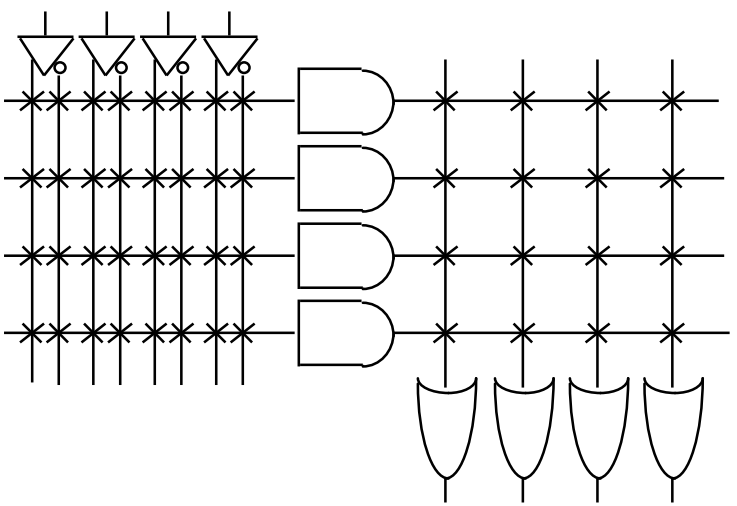
Depois da programação

- Conexões indesejadas são queimadas
  - fuse (normalmente conectado, quebra os indesejados)
  - anti-fuse (normalmente desconectado, faz conexões desejadas)



Representação alternativa para estruturas com fan-in alto

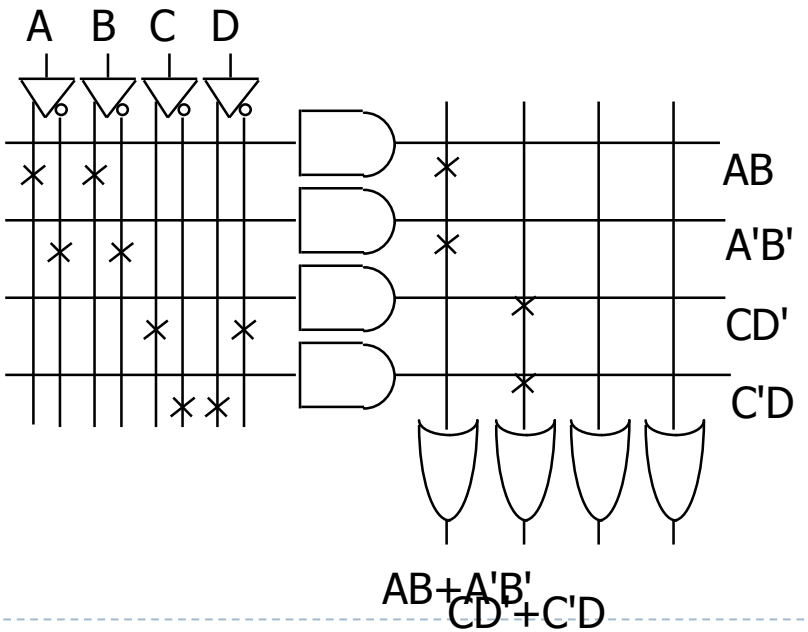
- Notação simplificada
  - Não é necessário desenhar todos os fios
  - × significa que uma conexão está presente e o sinal perpendicular está em uma entrada da porta



Notação para implementação

F0 = A B + A' B'

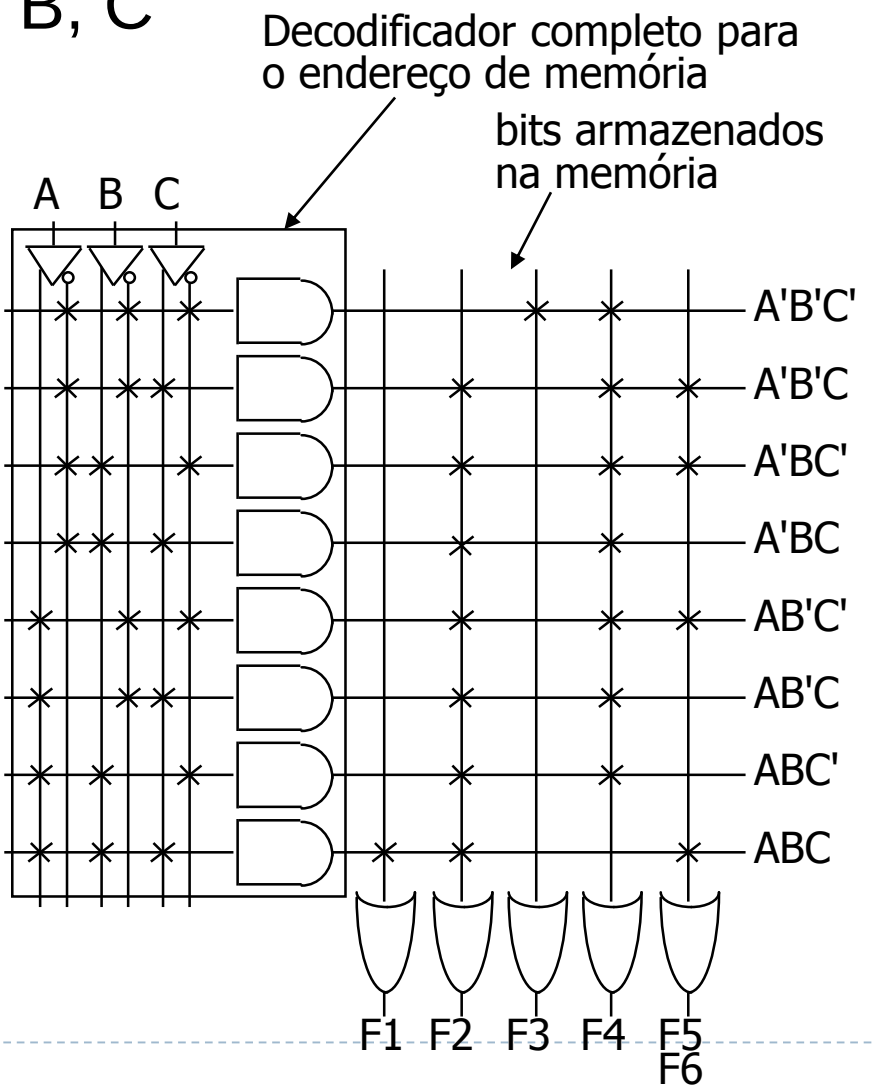
F1 = C D' + C' D



Array lógico programável Exemplo

- Funções múltiplas de A, B, C
  - F1 = A B C
  - F2 = A + B + C
  - F3 = A' B' C'
  - F4 = A' + B' + C'
  - F5 = A xor B xor C
  - F6 = A xnor B xnor C

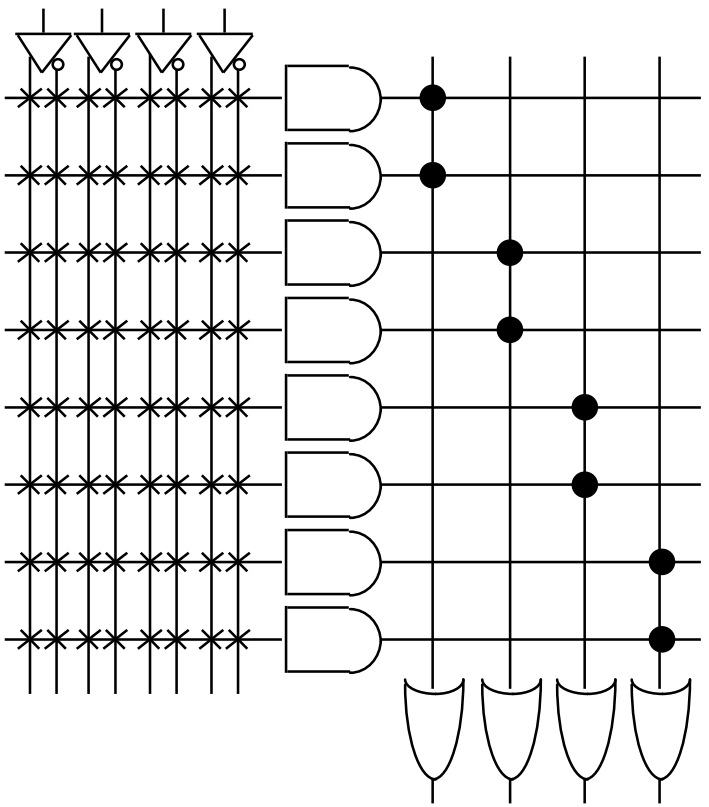
A	B	C	F1	F2	F3	F4	F5	F6
0	0	0	0	0	1	1	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	1	1



PALs e PLAs

- Array lógico programável (PLA)
  - O que vimos até agora
  - Arrays AND e OR são genéricos e totalmente irrestritos
- Array lógico programável (PAL)
  - Topologia restrita do array OR
  - Inovação por memórias monolíticas
  - Plano OR mais rápido e menor

Uma dada coluna do array OR tem acesso apenas a um subconjunto dos possíveis termos do produto



PALs e PLAs: Exemplo projeto

- Conversor BCD para código Gray

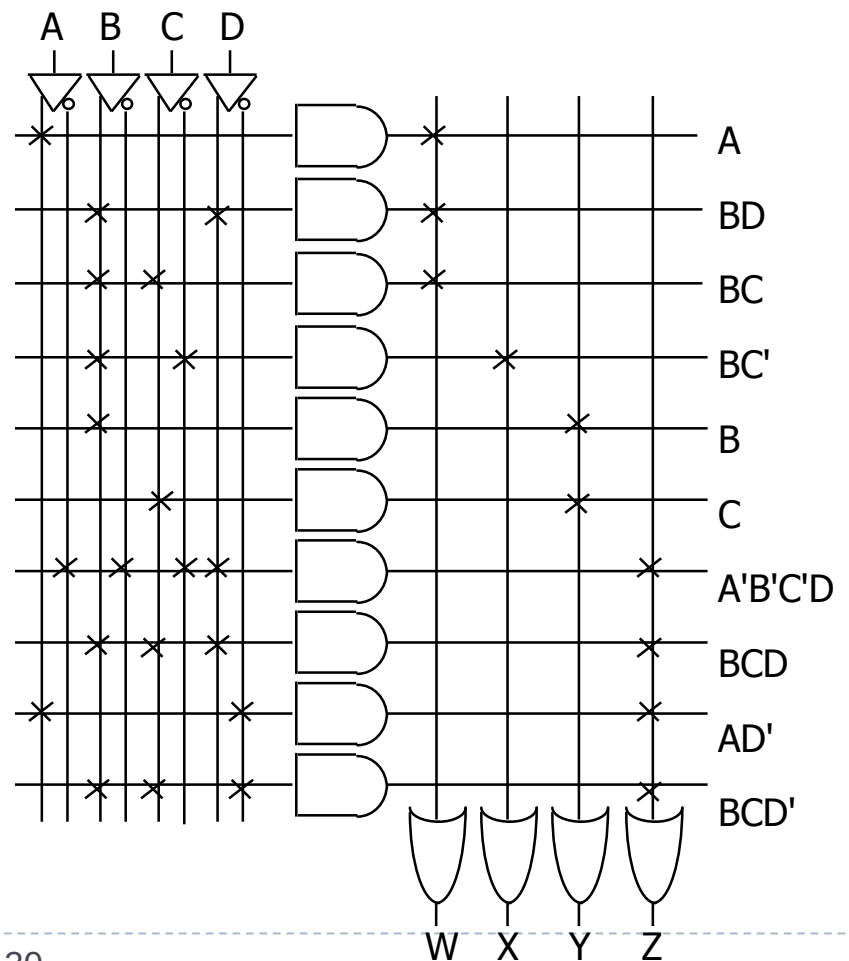
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	–	–	–	–	–
1	1	–	–	–	–	–	–

Função minimizada:

W = A + BD + BC
X = BC'
Y = B + C
Z = A'B'C'D + BCD + AD' + B'CD'

PALs e PLAs: Exemplo projeto

- Código conversor: PLA programado



Funções minimizadas:

W = A + BD + BC
X = B C'
Y = B + C
Z = A'B'C'D + BCD + AD' + B'CD'

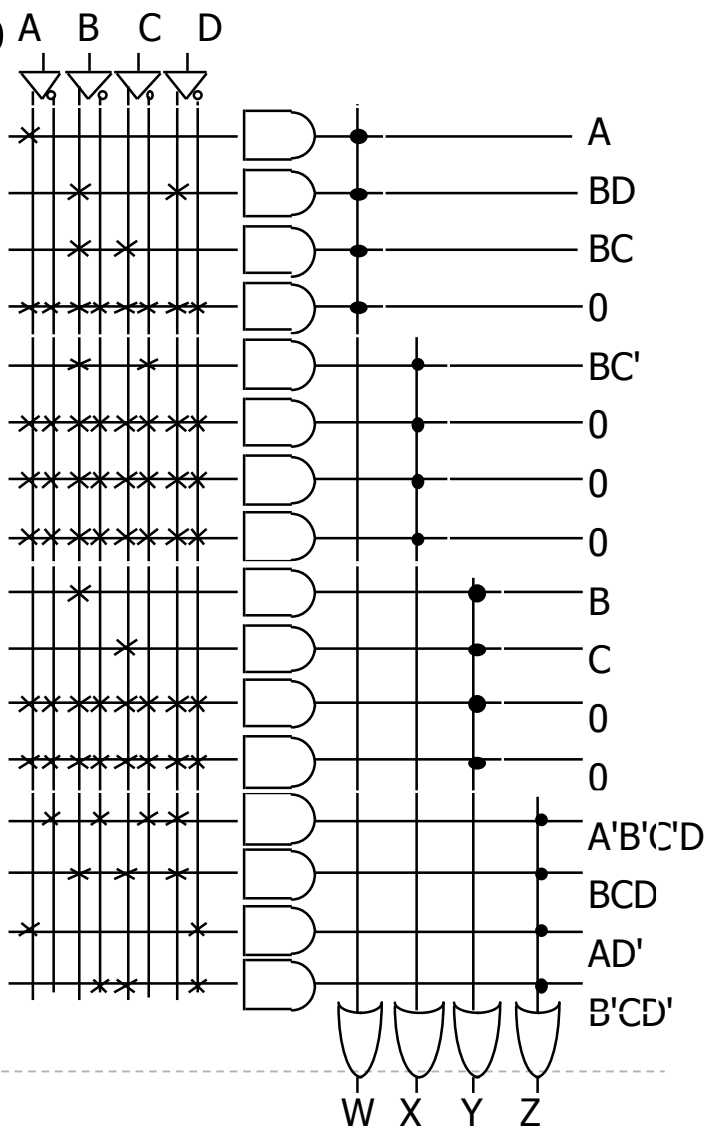
Não é um bom candidato particulamente para implementações PAL/PLA, pois, nenhum termo é compartilhado entre as saídas

No entanto, é uma implementação muito mais compacta e regular quando comparado com portas AND e OR discretas

PALs e PLAs: Exemplo projeto

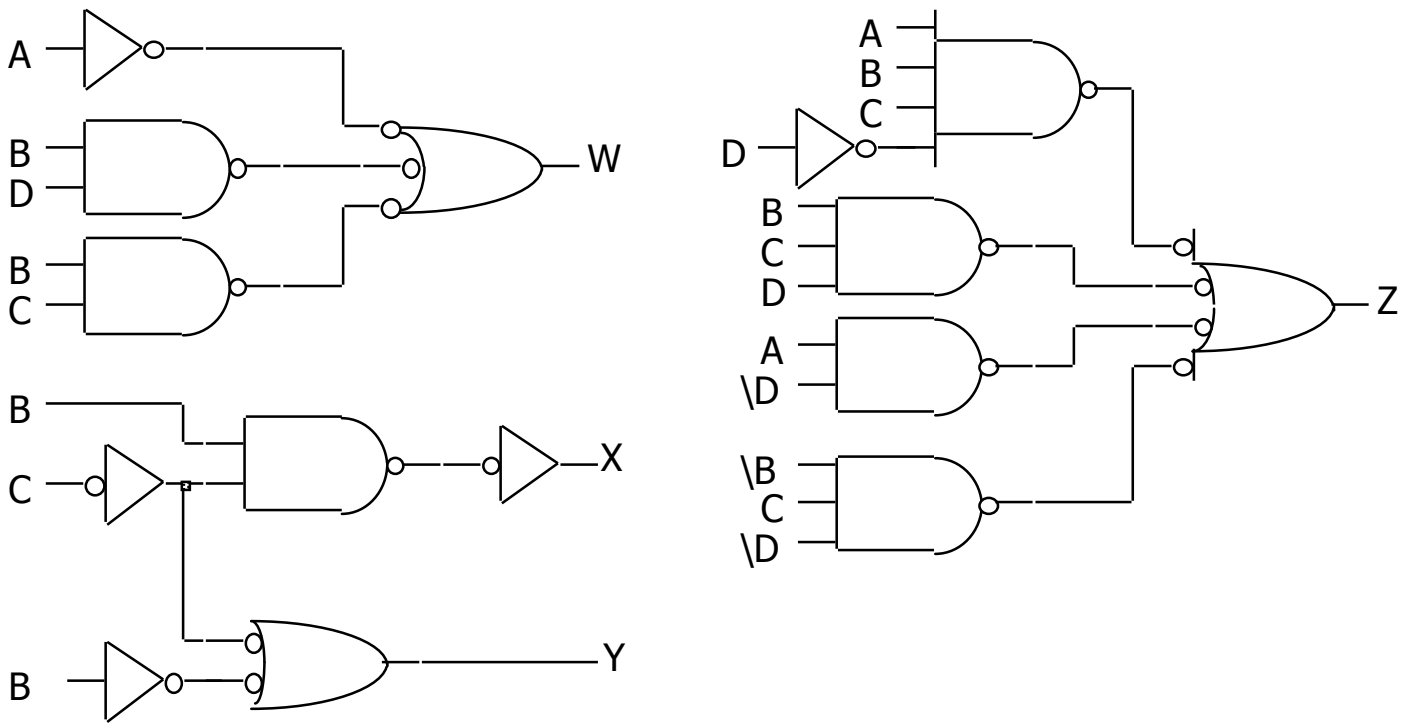
- Código conversor: PAL programado

4 termos de produto
por porta OR



PALs e PLAs: Exemplo projeto

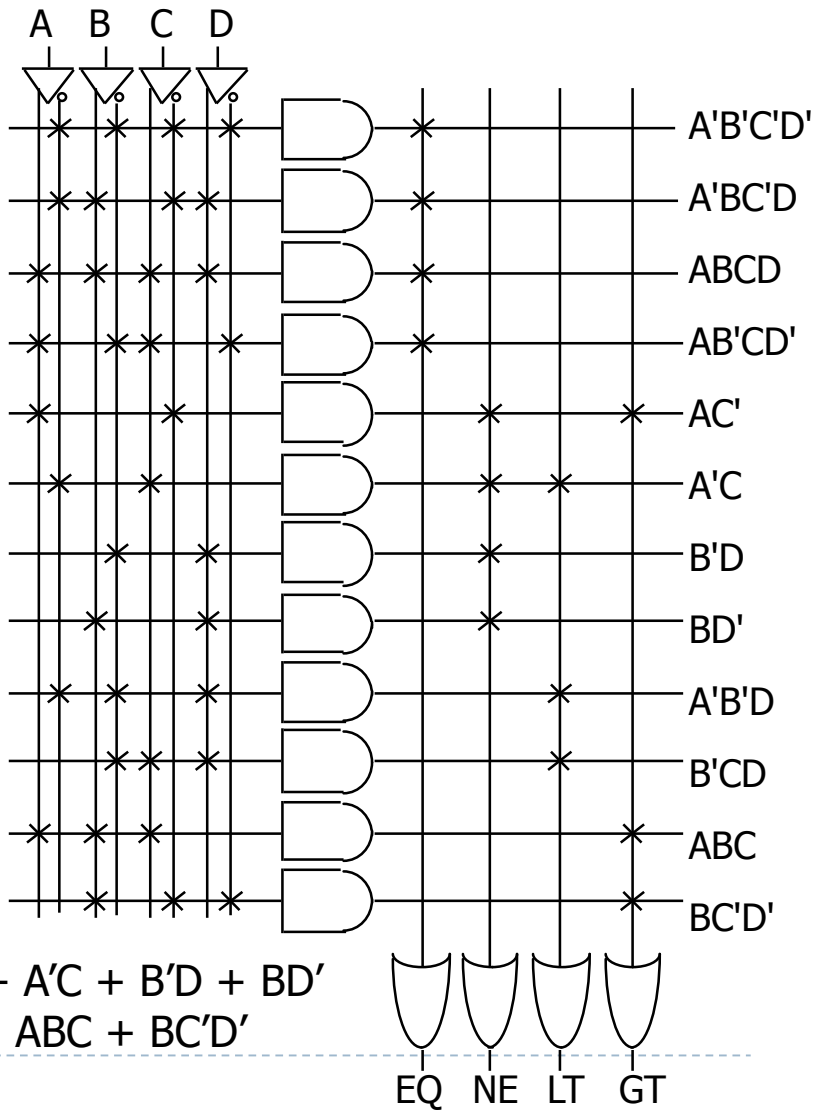
- Código conversor: Implementação porta NAND
  - Perda ou regulariedade, mais difícil para entender
  - Mais difícil para fazer alterações



PALs e PLAs: Outro exemplo de projeto

- Comparador de magnitude

A	B	C	D	EQ	NE	LT	GT
0	0	0	0	1	0	0	0
0	0	0	1	0	1	1	0
0	0	1	0	0	1	1	0
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	1	0	0	0
0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	0	1
1	0	1	0	1	0	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	1
1	1	0	1	0	1	0	1
1	1	1	0	0	1	0	1
1	1	1	1	1	0	0	0



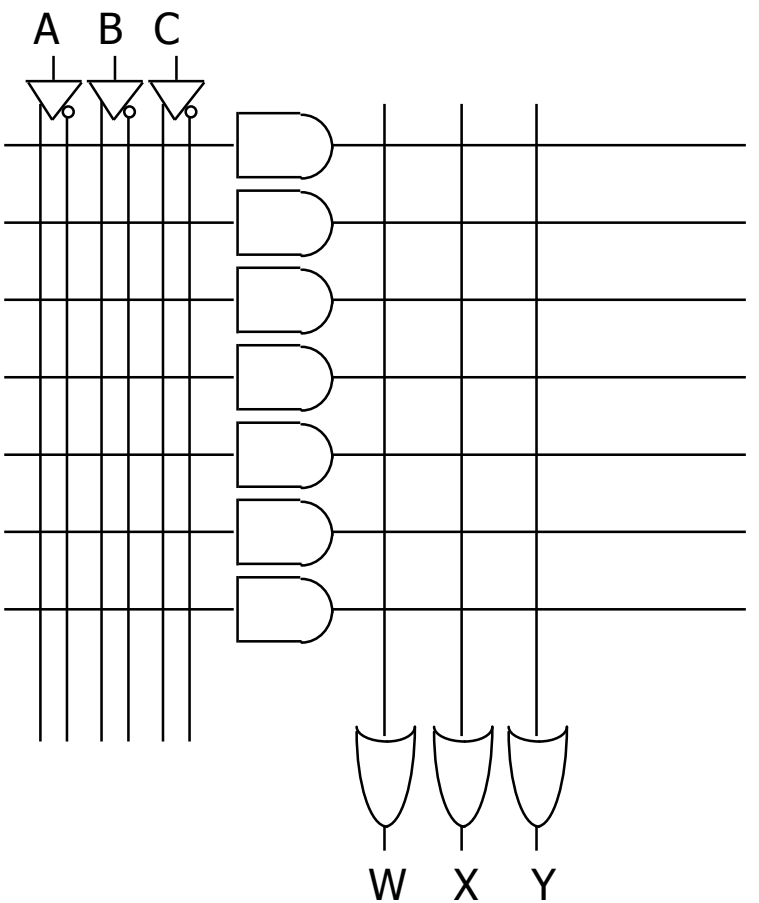
Funções minimizadas:

EQ = A'B'C'D' + A'BC'D + ABCD + AB'CD' NE = AC' + A'C + B'D + BD'

LT = A'C + A'B'D + B'CD GT = AC' + ABC + BC'D'

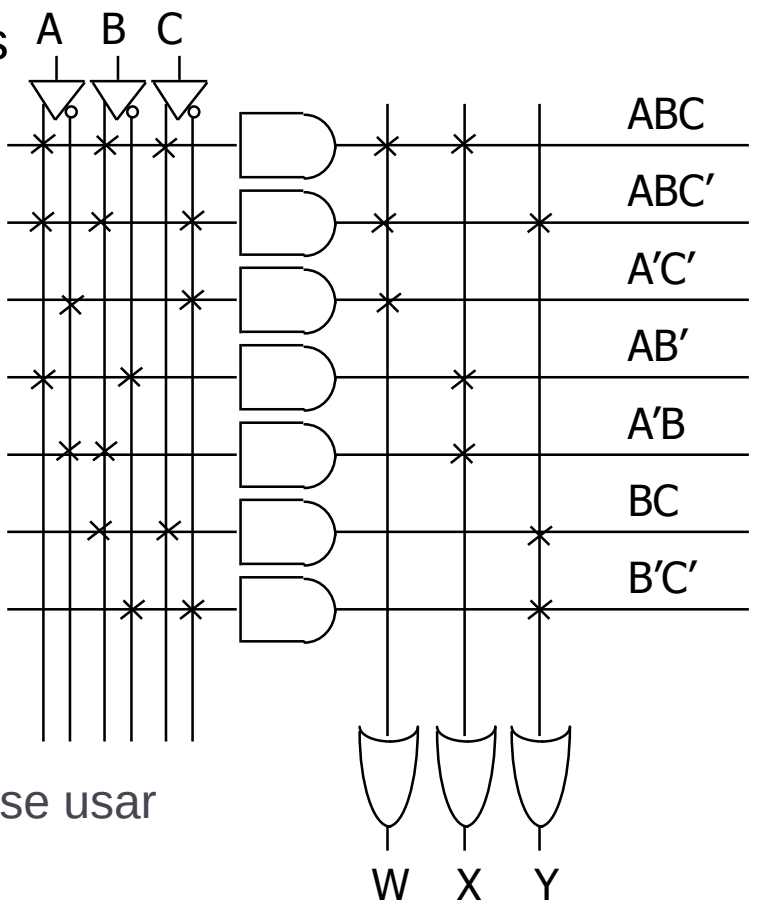
Atividade

- Mapear as seguintes funções para o PLA abaixo:
  - W = AB + A'C' + BC'
  - X = ABC + AB' + A'B
  - Y = ABC' + BC + B'C'



Atividade

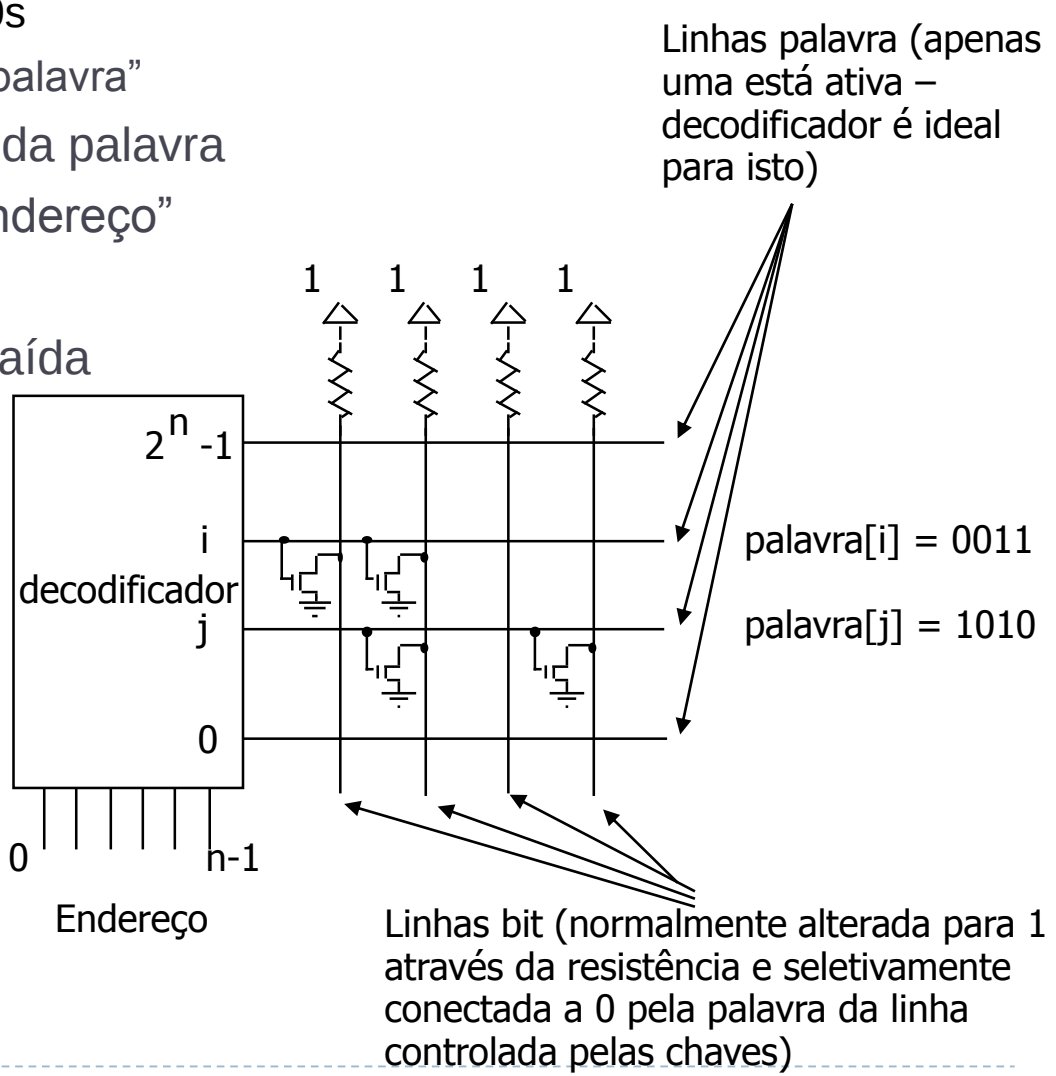
- 9 termos não irão caber em 7 termos do PLA
  - pode aplicar o teorema concensus para simplificar:
    W = AB + A'C'
- 8 termos não irão caber em 7 termos do PLA
  - Observe que AB = ABC + ABC'
  - Pode rescrever W para reutilizar os termos:
    W = ABC + ABC' + A'C'
- Agora se encaixa
  - W = ABC + ABC' + A'C'
  - X = ABC + AB' + A'B
  - Y = ABC' + BC + B'C'
- Isto é chamado de tecnologia de mapeamento
  - manipulação de funções lógicas para se usar os recursos disponíveis



Memórias somente leitura

- Dois arrays dimensionais de 1s e 0s
  - Entrada (linha) é chamada de “palavra”
  - Largura da linha = tamanho da palavra
  - Indíce é chamado de um “endereço”
  - Endereço é uma entrada
  - selecionar palavras é uma saída

Organização interna



ROMs e lógica combinacional

- Implementação de lógica combinacional (forma canônica dois níveis) usando uma ROM

F0 = A' B' C + A B' C' + A B' C

F1 = A' B' C + A' B C' + A B C

F2 = A' B' C' + A' B' C + A B' C'

F3 = A' B C + A B' C' + A B C'

A	B	C	F0	F1	F2	F3
0	0	0	0	0	1	0
0	0	1	1	1	1	0
0	1	0	0	1	0	0
0	1	1	0	0	0	1
1	0	0	1	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	0	1
1	1	1	0	1	0	0

Tabela verdade

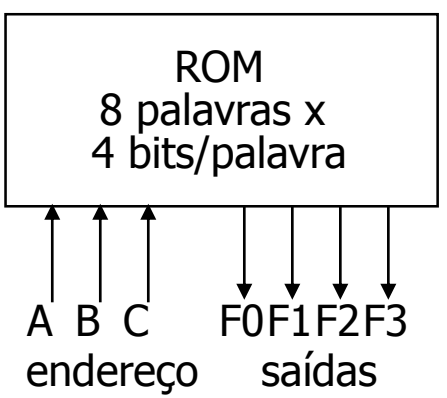
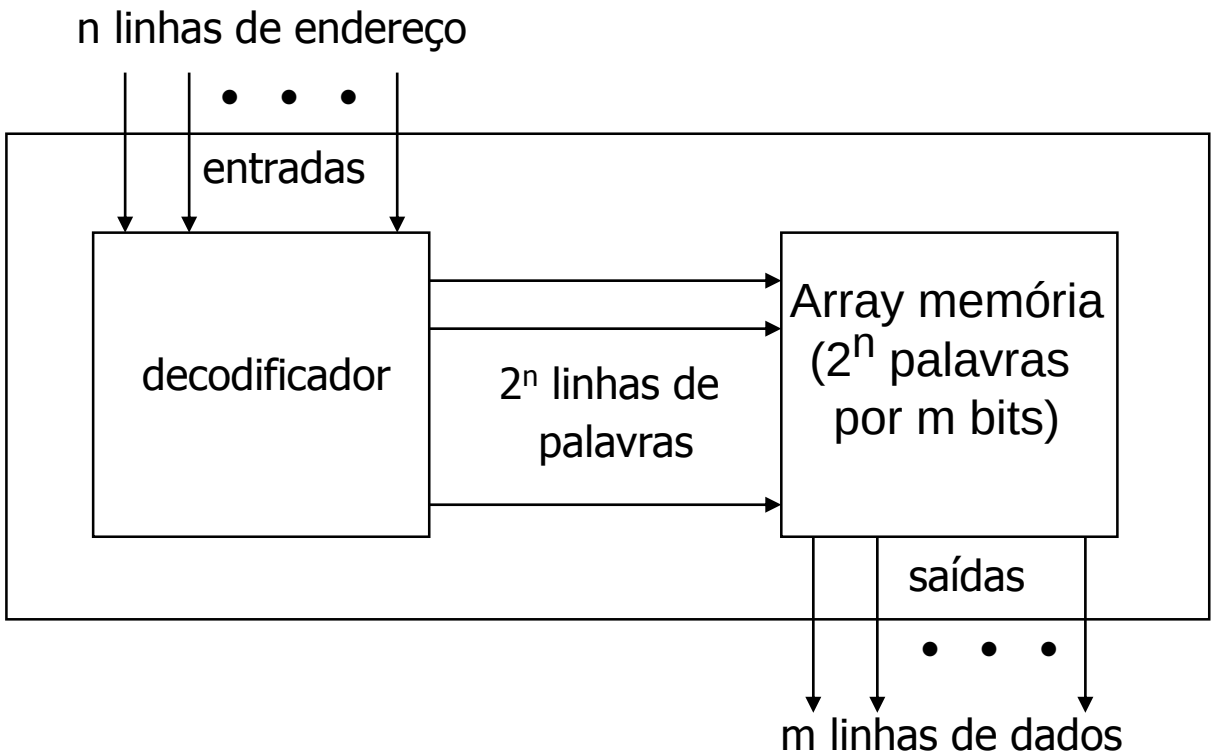


Diagrama de bloco

Estrutura ROM

- Similar a estrutura do PLA, mas com um array AND completamente decodificado
  - Array OR completamente flexisível (ao contrário do PAL)



ROM vs. PLA

- Abordagem ROM vantajosa quando
  - Tempo do projeto é curto (não precisa minimizar as funções da saída)
  - A maioria das entradas combinacionais são necessárias (e.g., código conversor)
  - Pouco compartilhamento de termos de produto entre as funções de saída

- Problemas ROM
  - Tamanho dobra para cada entrada adicional
  - Não pode explorar don't cares

ROM vs. PLA

- Abordagem PLA vantajosa quando
  - Ferramentas de projeto estão disponíveis para minimização de multi saídas
  - Existem relativamente menos combinações mintermo únicas
  - Muitos mintermos são compartilhados entre as funções de saída
- Problemas PAL
  - Restrições fan-ins em plano OR

Estrutura lógica regular para lógica dois níveis

- ROM – Plano AND completo, Plano OR Genérico
  - Barato (componente de alto volume)
  - Pode implementar qualquer função de n entradas
  - Velocidade média
- PAL – Plano AND programável, Plano OR fixo
  - Custo intermediário
  - Pode implementar funções limitadas pelo número de termos
  - Velocidade alta (apenas um plano programável que é muito menor que o decodificador da ROM)
- PLA – Planos AND e OR programáveis
  - Muito caro (mais complexo o projeto, precisa de mais ferramentas sofisticadas)
  - Pode implementar qualquer função para um limite do termo do produto
  - Lento (dois planos programáveis)

Estrutura lógica regular para lógica dois níveis

- Difícil de encontrar uma estrutura regular para conexões arbitrárias entre um grande conjunto de diferentes tipos de portas
  - Preocupações de eficiência/velocidade para cada estrutura
  - Em 467 você irá aprender sobre arranjo de portas programáveis de campo que são apenas estruturas multi níveis programáveis
    - Multiplexadores programáveis para wiring
    - Tabelas de buscas para funções lógicas (programação de preenchimento da tabela)
    - Células multi propósitos (utilização é uma grande questão)
- Uso de múltiplos níveis de PALs/PLAs/ROMs
  - Resultado da saída intermediária
  - Faz uma entrada ser utilizada em mais lógica

Resumo tecnologias lógica combinacional

- Lógica aleatória
  - Portas únicas ou em grupos
  - Conversão para redes NAND-NAND e NOR-NOR
  - Transição a partir de portas simples para portas mais complexas construídas com blocos
  - Redução quantidade portas, fan-ins, potencialmente mais rápida
  - Mais níveis, projeto mais difíceis
- Tempo de resposta em redes combinacionais
  - Atraso porta e tempo formas de onda
  - hazards/glitches (O que são e porque eles acontecem)
- Lógica Regular
  - Multiplexadores/Decodificadores
  - ROMs
  - PLAs/PALs
  - Vantagens/Desvantagens de cada