

MC542

Organização de Computadores
Teoria e Prática

2007

Prof. Paulo Cesar Centoducatte
ducatte@ic.unicamp.br
www.ic.unicamp.br/~ducatte

MC542
11.1

MC542

Circuitos Lógicos

ULA, Operações de Ponto Flutuante, Memórias

"DDCA" - (Capítulo 5)
"FDL" - (Capítulo)
"COD" - (Capítulo)

MC542
11.2

Sumário

- Comparador
 - Igualdade
 - Menor Que
 -
- Arithmetic Logic Unit (ALU)
- Set on Less Than (SLT)
- Shifters
 - Shifters como Multiplicador e Divisor
- Multiplicação
- Divisão
- Sistemas Numéricos
 - Números Fracionários
 - Ponto Flutuante - Arredondamento
 - Ponto Flutuante: Adição
- Memórias

MC542
11.3

Comparador: Igual

Símbolo

Implementação

MC542
11.4

Comparador: Menor Que

$A < B \iff A - B < 0$

MC542
11.5

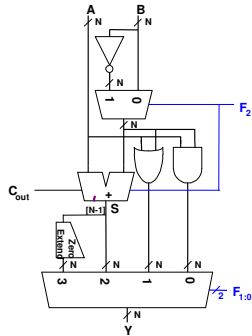
Arithmetic Logic Unit (ALU)

F _{2,0}	Function
000	A & B
001	A B
010	A + B
011	not used
100	A & ~B
101	A ~B
110	A - B
111	SLT

• SLT - Set on Less Than

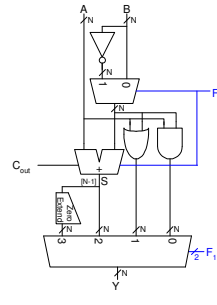
MC542
11.6

Projeto de uma ALU



MCS42
11.7

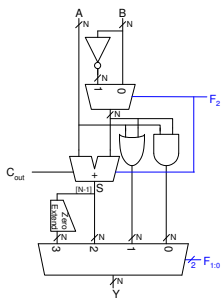
ALU - Funções



F _{2:0}	Function
000	A & B
001	A B
010	A + B
011	not used
100	A & B
101	A ~B
110	A - B
111	SLT

MCS42
11.8

Set Less Than (SLT)



• Configuração da ALU 32-bit para a operação SLT. Suponha que $A = 25$ e $B = 32$.

- Como A é menor que B, Y deve ser a representação de 1 em 32-bit (0x00000001).
- Para SLT, $F_{2:0} = 111$.
- $F_3 = 1$ configura a unidade adder como um subtrator ($25 - 32 = -7$).
- A representação complemento de dois de -7 tem um 1 no bit mais significativo, assim, $S_{31} = 1$.
- Com $F_{1:0} = 11$, o último mux seleciona $Y = 1$.

MCS42
11.9

Shifters

• Shifter lógico:

- Ex: $11001 \gg 2 = 00110$
- Ex: $11001 \ll 2 = 00100$

• Shifter Aritmético:

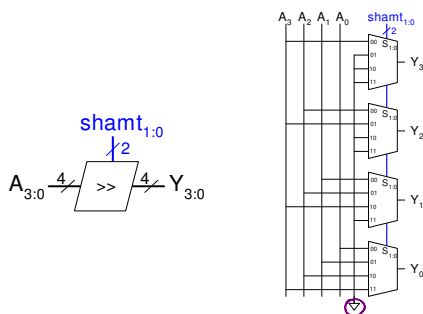
- Ex: $11001 \gg 2 = 11110$
- Ex: $11001 \ll 2 = 00100$

• Rotação:

- Ex: $11001 \text{ ROR } 2 = 01110$
- Ex: $11001 \text{ ROL } 2 = 00111$

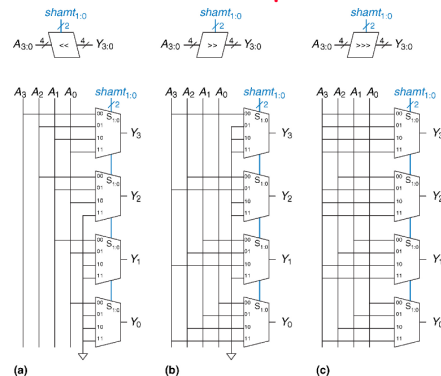
MCS42
11.10

Shifter Rápido



MCS42
11.11

Shifter Rápido



MCS42
11.12

Shifters como Multiplicador e Divisor

Um shift left de N bits multiplica o número por 2^N

- Ex: $00001 \ll 2 = 00100$ ($1 \times 2^2 = 4$)
- Ex: $11101 \ll 2 = 10100$ ($-3 \times 2^2 = -12$)

Um shift right aritmético de N divide o número por 2^N

- Ex: $01000 \ggg 2 = 00010$ ($8 \div 2^2 = 2$)
- Ex: $10000 \ggg 2 = 11100$ ($-16 \div 2^2 = -4$)

MCS42
11.13

Multiplicação

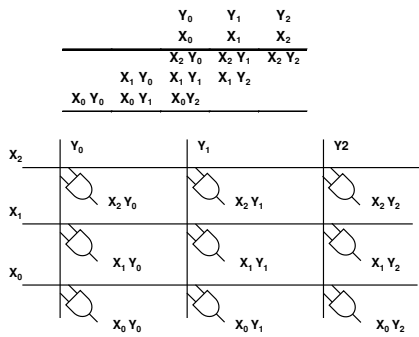
```

810 1000 multiplicando
910 1001 multiplicador
-----
1000
0000
0000 produtos parciais
1000
-----
7210 1001000
    
```

$\max = (2^8 - 1) * (2^8 - 1) = 225$
 $225 > 128 \Rightarrow 8$ bits
 $32 * 32$ bits $\Rightarrow 64$ bits

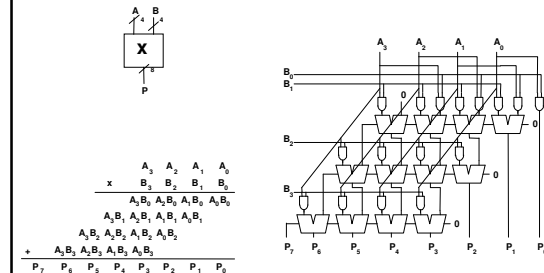
MCS42
11.14

Multiplicação Geração Rápida dos Produtos Parciais



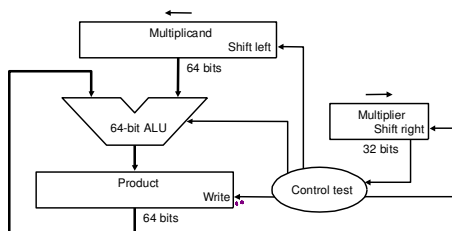
MCS42
11.15

Multiplicador 4x4



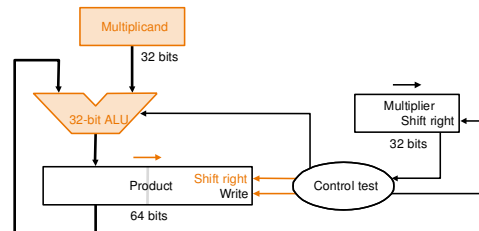
MCS42
11.16

Multiplicador - Primeira Versão

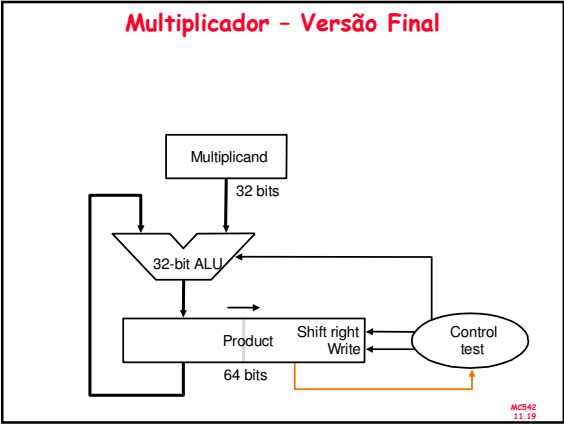


MCS42
11.17

Multiplicador - Segunda Versão



MCS42
11.18



Divisão

$29 + 3 \Rightarrow 29 = 3 * Q + R = 3 * 9 + 2$
dividendo divisor quociente resto

$29_{10} = 011101 \quad 3_{10} = 11$

0 1 1 1 0 1	1 1	Q = 9	R = 2
1 1	0 1 0 0 1		
0 0 1 0 1	1 1		
1 1	1 0		

Como implementar em hardware?

MCS42 11.20

Alternativa 1: Divisão com Restauração

- hardware não sabe se "vai caber ou não"
- registrador para guardar resto parcial
- verificação do sinal do resto parcial
- caso negativo \Rightarrow restauração

$29 - 3 * 2^4 = -19$	$q_4 = 1$	
$-19 + 3 * 2^4 = 29$	$q_4 = 0$	Restauração
$29 - 3 * 2^3 = 5$	$q_3 = 1$	
$5 - 3 * 2^3 = -7$	$q_3 = 0$	Restauração
$-7 + 3 * 2^3 = 5$	$q_2 = 1$	
$5 - 3 * 2^2 = -1$	$q_2 = 0$	Restauração
$-1 + 3 * 2^2 = 5$	$q_1 = 1$	
$5 - 3 * 2^1 = -1$	$q_1 = 0$	Restauração
$-1 + 3 * 2^1 = 5$	$q_0 = 1$	

$R = 10 = 2 \quad q_4 q_3 q_2 q_1 q_0 = 01001 = 9$

MCS42 11.21

Alternativa 2: Divisão sem Restauração

Regras

se resto parcial > 0	próxima operação	subtração	objetivo R \rightarrow 0
se resto parcial < 0	próxima operação	soma	

se operação corrente +	$q_i \neq 1$
se operação corrente -	$q_i = 1$

$29 - 3 * 2^4 = -19 < 0$	próx = SOMA	$q_4 = 1$
$-19 + 3 * 2^4 = 5 > 0$	próx = SUB	$q_3 = 0$
$5 - 3 * 2^3 = -7 < 0$	próx = SOMA	$q_2 = 1$
$-7 + 3 * 2^3 = 5 > 0$	próx = SUB	$q_1 = 0$
$5 - 3 * 2^2 = -1 < 0$	próx = SOMA	$q_0 = 1$
$-1 + 3 * 2^2 = 5 > 0$		

Resto = 2 Quociente = 1001 = 9

MCS42 11.22

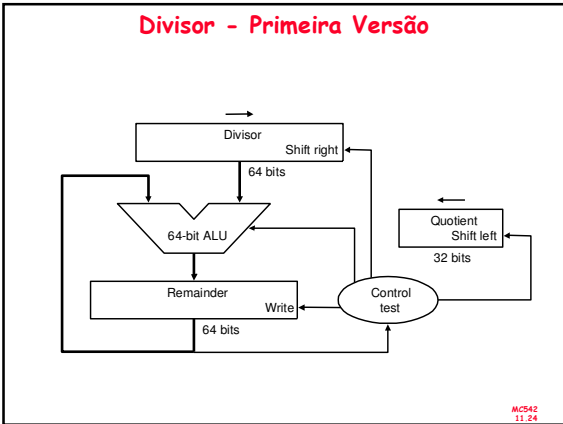
Alternativa 2: Conversão do Resultado

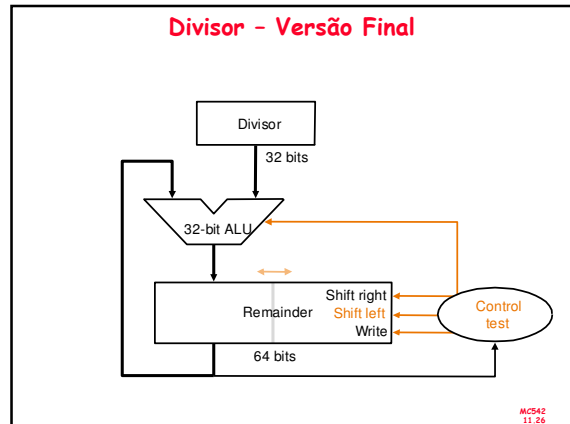
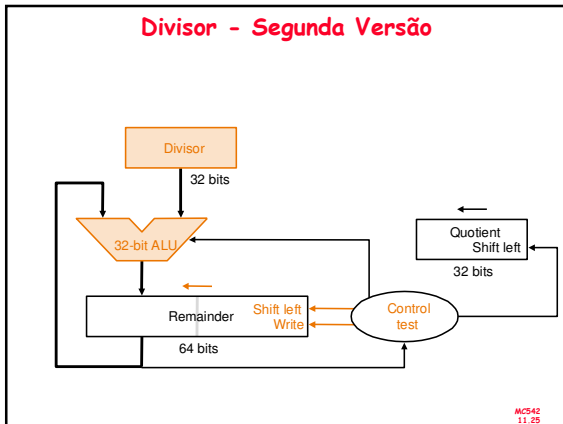
$1 \bar{1} \bar{1} \bar{1} \bar{1} = (2^4 - 2^3 + 2^2 - 2^1 - 2^0)$
 $16 - 8 + 4 - 2 - 1$

$\dots 1 \bar{1} \dots = 2^n - 2^{(n-1)} = 2^{(n-1)}(2 - 1) = 2^{(n-1)}$

1 1 1 1 1	<ul style="list-style-type: none"> • Nº de somas: 3 • Nº de subtrações: 2 • Total: 5 • OBS: se resto < 0 deve haver correção de um divisor para que resto > 0
0 1 1 1 1	
0 1 0 1 1	
0 1	

MCS42 11.23





- ### Sistemas Numéricos
- Que tipo de números queremos representar usando uma representação binária?
 - Números Positivos
 - » Unsigned binary
 - Números Negativos
 - » Complemento de dois
 - » Sinal/magnitude
 - E frações?
- MCS42 11.27

- ### Números Fracionários
- Duas notações são comuns:
 - Ponto Fixo (Fixed-point):
 - » o ponto binário é fixo
 - Ponto Flutuante (Floating-point):
 - » O ponto binário flutua a direita do bit mais significativo
 - Já visto
- MCS42 11.28

- ### Ponto Flutuante - Arredondamento
- Overflow: quando o valor é muito grande para ser representado
 - Underflow: quando o valor é muito pequeno para ser representado
 - Modos de Arredondamento:
 - Down
 - Up
 - Toward zero
 - To nearest

Internamente, o processador armazena os números de ponto flutuante com ao menos 2 bits a mais: *guard e round*. Um terceiro bit, o *sticky* indica se algum conteúdo significativo foi perdido em arredondamento
 - Exemplos: arredondar 1.100101 (1.578125) de forma ser representado com 3 bits de fração.
 - Down: 1.100
 - Up: 1.101
 - Toward zero: 1.100
 - To nearest: 1.101 (1.625 é mais próximo de 1.578125 do que 1.5)
- MCS42 11.29

- ### Ponto Flutuante: Adição
1. Extrair os bits do expoente e da fração
 2. Adicionar o bit 1 à mantissa (prefixo)
 3. Compar os expoentes
 4. Deslocar a menor mantissa, se necessário
 5. Somar as mantissas
 6. Normalizar a mantissa e ajustar o expoente, se necessário
 7. Arredondar o resultado
 8. Montar o expoente e a fração no formato de ponto flutuante
- MCS42 11.30

Ponto Flutuante: Exemplo de Adição

Somar os seguintes números em ponto flutuante :
 0x3FC00000
 0x40500000

1. Extrair o expoente e a fração

1 bit	8 bits	23 bits
0	01111111	100 0000 0000 0000 0000
Sign	Exponent	Fraction
1 bit	8 bits	23 bits
0	10000000	101 0000 0000 0000 0000
Sign	Exponent	Fraction

Para o primeiro número (N1): S = 0, E = 127, F = .1
 Para o segundo número (N2): S = 0, E = 128, F = .101

MCS42
11.31

Ponto Flutuante: Exemplo de Adição

2. Adicionar o bit 1 à mantissa (prefixo)

N1: 1.1
 N2: 1.101

3. Comparar os expoentes

127 - 128 = -1, shift N1 para a direita de 1 bit

4. Deslocar a menor mantissa, se necessário

shift mantissa de N1: 1.1 >> 1 = 0.11 ($\times 2^1$)

MCS42
11.32

Ponto Flutuante: Exemplo de Adição

5. Somar as mantissas

$$\begin{array}{r} 0.11 \times 2^1 \\ + 1.101 \times 2^1 \\ \hline 10.011 \times 2^1 \end{array}$$

6. Normalizar a mantissa e ajustar o expoente, se necessário

$$10.011 \times 2^1 = 1.0011 \times 2^2$$

7. Arredondar o resultado

Não é necessário (cabe em 23 bits)

8. Montar o expoente e a fração no formato de ponto flutuante

S = 0, E = 2 + 127 = 129 = 10000001₂, F = 001100..

MCS42
11.33

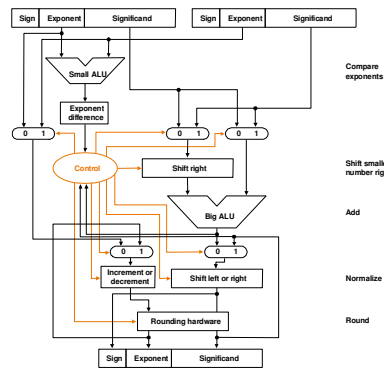
Ponto Flutuante: Exemplo de Adição

1 bit	8 bits	23 bits
0	10000001	001 1000 0000 0000 0000
Sign	Exponent	Fraction

Em hexadecimal: 0x40980000

MCS42
11.34

Hardware de Adição de Ponto Flutuante



MCS42
11.35

Floating-point numbers

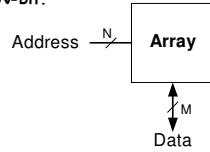
	0	10000001	111 1100 0000 0000 0000
	0	01111100	100 0000 0000 0000 0000
	Exponent	Fraction	
Step 1	10000001	111 1100 0000 0000 0000	
	01111100	100 0000 0000 0000 0000	
Step 2	10000001	1.111 1100 0000 0000 0000	
	01111100	1.100 0000 0000 0000 0000	
Step 3	10000001	1.111 1100 0000 0000 0000	
	01111100	1.100 0000 0000 0000 0000	-101 (shift amount)
Step 4	10000001	1.111 1100 0000 0000 0000	
	10000001	0.000 0110 0000 0000 0000	00000
Step 5	10000001	1.111 1100 0000 0000 0000	
	10000001	0.000 0110 0000 0000 0000	+
		10.000 0010 0000 0000 0000	
Step 6	10000001	10.000 0010 0000 0000 0000	>> 1
		1.000 0001 0000 0000 0000	
Step 7			(No rounding necessary)
Step 8	0	10000010	000 0001 0000 0000 0000

© 2007 Elsevier Inc. All rights reserved.

MCS42
11.36

Memórias

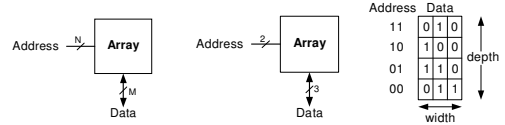
- Memory arrays - armazena eficientemente grande quantidade de dados.
- Três tipos mais comuns de memory arrays:
 - Dynamic random access memory (DRAM)
 - Static random access memory (SRAM)
 - Read only memory (ROM)
- Um dado de valor de M -bit pode ser lido ou escrito por vez em um endereço de N -bit.



MCS42 11.37

Memórias

- As memórias são organizadas como um array de duas dimensões de células de bits. Cada célula armazena um bit.
- Um array com N bits de endereço e dados de M bits tem 2^N linhas e M colunas. Each row of data is called a word.
 - Depth: número de linhas do array
 - Width: número de colunas no array (word size)
 - Array size: dado por $\text{depth} \times \text{width}$

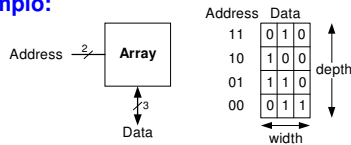


MCS42 11.38

Memória : Exemplo

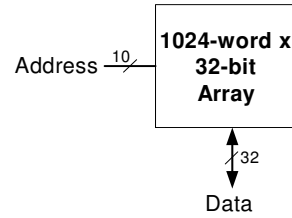
- O array abaixo é um array $2^2 \times 3$ -bit.
- Word size de 3-bits.

Exemplo:



MCS42 11.39

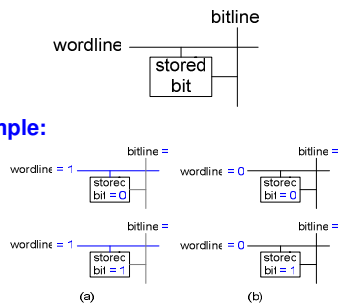
Memória : Exemplo



MCS42 11.40

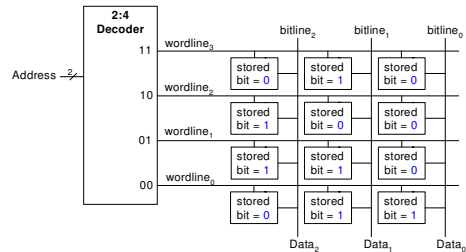
Memória: Célula de bit

Exemplo:



MCS42 11.41

Memória: 4x3



MCS42 11.42

Tipos de Memórias

- Random access memory (RAM): **volátil**
- Read only memory (ROM): **não volátil**

MCS42
11.43

RAM

- Random access memory
 - Volátil: perde o dado quando a alimentação é desligada
 - Pode ser lida ou escrita rapidamente
 - A memória principal do seu computador é RAM (specificamente, DRAM)
 - Historicamente denominada de *random access memory* porque qualquer palavra de dado pode ser acessada como qualquer outra (em contraste com sequential access memories como fita magnética).

MCS42
11.44

ROM

- Read only memory (ROM)
 - Não volátil: não perdem seus dados quando a alimentação é desligada
 - Pode ser lida rapidamente, porém a escrita é lenta
 - Memórias em câmeras digitais, pen drives são ROMs
 - Historicamente denominadas de *read only memory* porque ROMs eram escritas queimando-se fusíveis. Uma vez que os fusíveis eram escritos não podiam ser escritos novamente.

MCS42
11.45

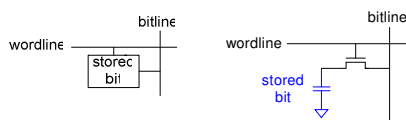
Tipos de RAM

- Os dois tipos de RAM são:
 - Dynamic random access memory (DRAM)
 - Static random access memory (SRAM)
- A diferença é como armazenam os dados:
 - DRAM usa um capacitor
 - SRAM usa *cross-coupled inverters* ("latch")

MCS42
11.46

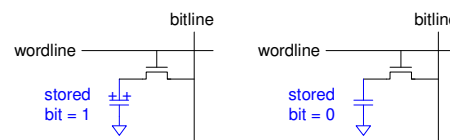
DRAM

- Data bits são armazenados em um capacitor.
- DRAM denominado de *dynamic* porque os valores necessitam ser reescritos (refreshed) periodicamente e após serem lidos por que:
 - A corrente de fuga do capacitor degrada o valor
 - A leitura destrói o valor armazenado

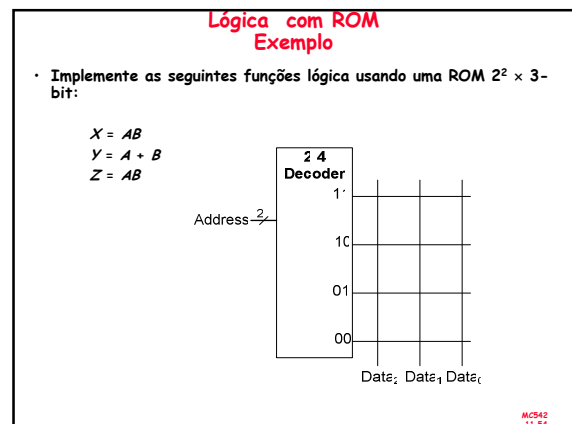
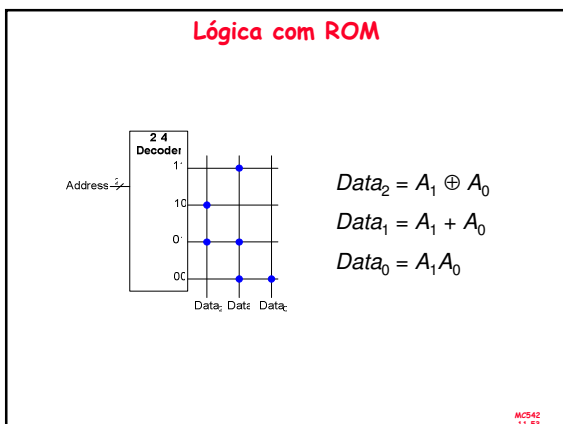
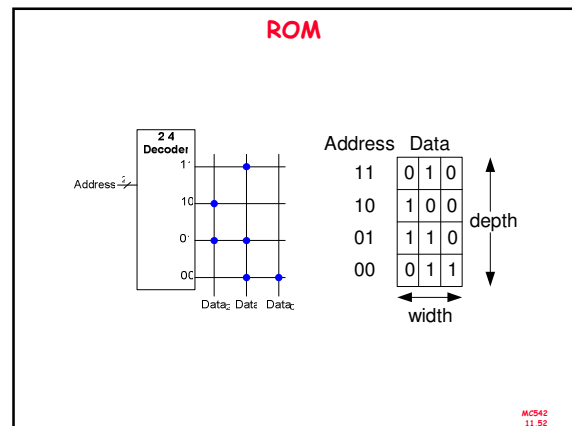
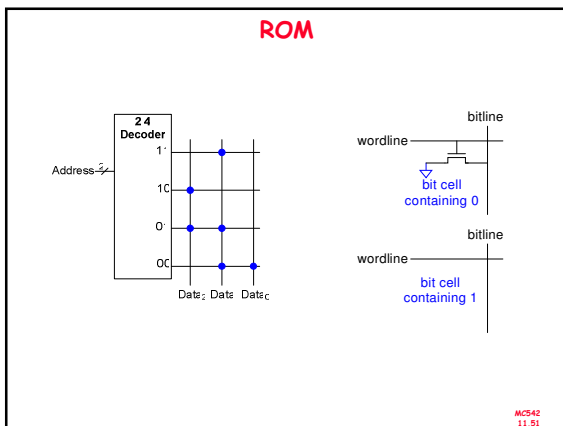
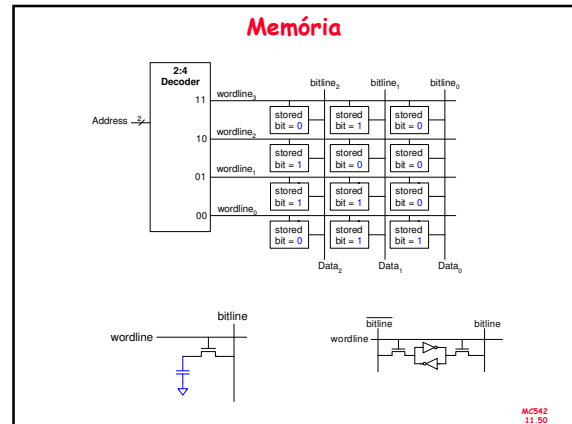
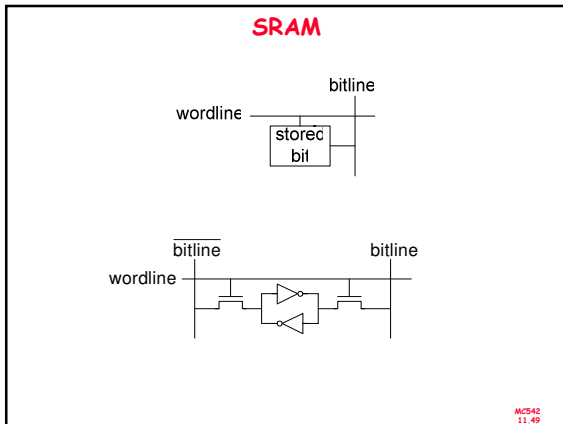


MCS42
11.47

DRAM



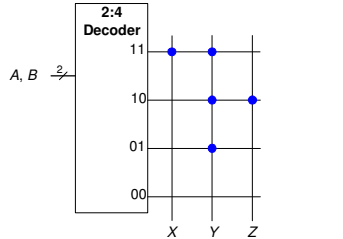
MCS42
11.48



Lógica com ROM Exemplo

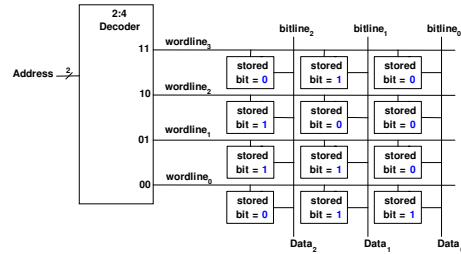
- Implemente as seguintes funções lógicas usando uma ROM $2^2 \times 3$ -bit:

$$\begin{aligned} X &= AB \\ Y &= A + B \\ Z &= AB \end{aligned}$$



MCS42 11.55

Lógica com Memória



$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = A_1 + A_0$$

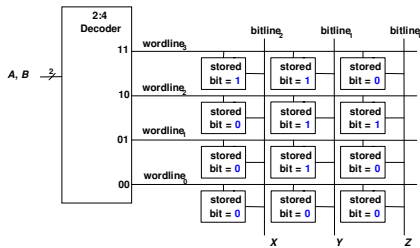
$$Data_0 = A_1 A_0$$

MCS42 11.56

Lógica com Memória

- Implemente as seguintes funções lógicas com uma memória $2^2 \times 3$ -bit:

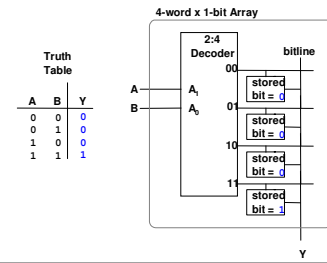
$$\begin{aligned} X &= AB \\ Y &= A + B \\ Z &= AB \end{aligned}$$



MCS42 11.57

Lógica com Memória

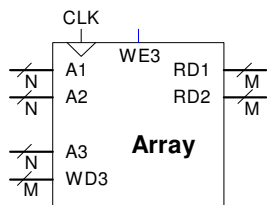
- A memória usada para executar funções lógicas é denominada *lookup tables (LUT)*.
- O usuário tem o valor de saída para cada combinação das entradas (address).



MCS42 11.58

Memórias Multi-Portas

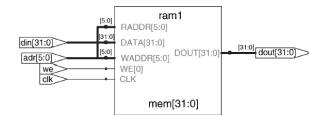
- Porta: par endereço/dado (address/data)
- Memória 3-portas
 - 2 portas de leitura (A1/RD1, A2/RD2)
 - 1 porta de escrita (A3/WD3, WE3 enables writing)



MCS42 11.59

Memória - VHDL

Library IEEE;
Use IEEE.STD_LOGIC_ARITH.ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;



Entity ram Is
Generic (N: Integer := 6; M: Integer := 32);
Port (clk,
we : In STD_LOGIC;
addr : In STD_LOGIC_VECTOR(N-1 Downto 0);
din : In STD_LOGIC_VECTOR(M-1 Downto 0);
dout : In STD_LOGIC_VECTOR(M-1 Downto 0));
End;

MCS42 11.60

Memória - VHDL

```

Architecture synth Of ram Is
  Type mem_ram Is Array ((2**N-1) Downto 0)
    Of STD_LOGIC_VECTOR(M-1 Downto 0);
  Signal mem: mem_ram;

Begin
  Process (clk)
  Begin
    If clk'event and clk = '1' Then
      If we = '1' Then
        mem(CONV_INTEGER(addr)) <= din;
      End If;
    End If;
  End Process;

  dout <= mem (CONV_INTEGER(addr));
End;
  
```

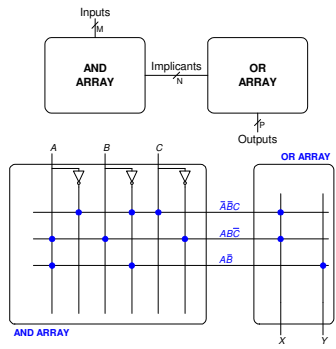
MCS42
11.61

Logic Arrays

- Programmable Logic Arrays (PLAs)
 - Array de ANDs seguido por um array de ORs
 - Executa somente lógica combinacional
 - Conexões internas "fixas"
- Field Programmable Gate Arrays (FPGAs)
 - Array de blocos lógicos configuráveis (CLBs)
 - Executa lógica combinacional e seqüencial
 - Conexões internas programáveis

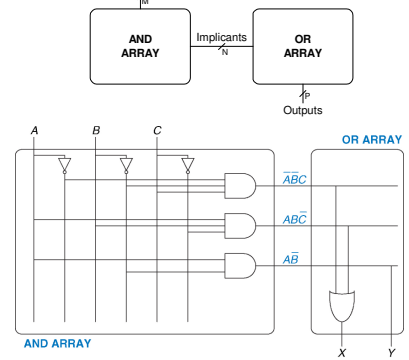
MCS42
11.62

PLAs



MCS42
11.63

PLAs



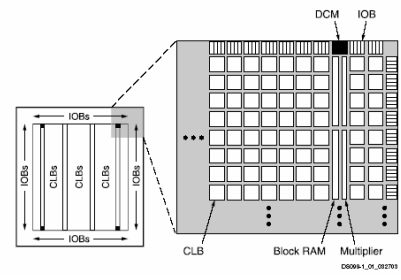
MCS42
11.64

FPGAs

- Composto por:
 - CLBs (Configurable logic blocks): executa a lógica
 - IOBs (Input/output buffers): interface com o mundo exterior
 - Programmable interconnection: usado para conectar CLBs e IOBs
 - Algumas FPGAs incluem outros blocos funcionais como somadores, multiplicadores e RAMs

MCS42
11.65

FPGA Xilinx Spartan 3



MCS42
11.66

