*** RELATÓRIO TÉNICO ***

PROPOSTA DE UMA ARQUITETURA CONFIGURÁVEL POR SOFTWARE

Maria Cristina do Nascimento
A. Mesquita
Edil S. Fernandes

NCE 17/91
Outubro/91

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

Este artigo foi apresentado no VI Simpósio Brasileiro de Concepção de Circuitos Integrados, realizado de 23 a 25 de outubro.

PROPOSTA DE UMA ARQUITETURA CONFIGURÁVEL POR SOFTWARE

RESUMO

Este artigo apresenta um novo paradigma para computação: arquitetura configurável por *software*, na qual a função implementada é definida por um programa de configuração. Esta estrutura é composta de várias células lógicas programáveis capazes de implementar funções lógicas e estruturas de roteamento programáveis, que realizam a comunicação entre células. O artigo inclui uma descrição da implementação em tecnologia CMOS - 2.0 μm.

A PROPOSAL OF A SOFTWARE CONFIGURABLE ARCHITECTURE

ABSTRACT

This paper presents a new paradigm for computation: software configurable hardware, in which the function implemented is dependent on a control store. This structure is composed of several programmable logic cells capable of implementing one of a set of logic level functions, and a programmable routing area which allows inter-cell communication. It is also presented the implementation of such a structure in CMOS - 2.0 μm.

Proposta de uma Arquitetura Configurável por Software

Maria Cristina do Nascimento [*]
A. Mesquita [**]
Edil S. Fernandes [***]

ABSTRACT

This paper presents a new paradigm for computation: Configurable Hardware, in which the function implemented is dependent on a control store. This structure is composed of several programmable logic cells capable of implementing one of a set of logic level functions and a programmable routing area which allows inter-cell communication.

1. Introdução

Apesar das inúmeras vantagens de desempenho, os projetos dedicados (*custom*) e semi-dedicados (*semi-custom*) de circuitos integrados para uma aplicação especifica (*ASIC - Application Specific Integrated Circuit*) apresentam algumas desvantagens, como o elevado custo para a fabricação de um número pequeno de CI's e a dificuldade para a realização de alterações no projeto. Qualquer alteração realizada, por menor que seja, implica na repetição de todos os testes e verificações para a confecção de novas máscaras de fabricação.

Uma alternativa para evitar os problemas acima mencionados é a utilização de dispositivos tais como FPLA's (*Field Programmable Logic Arrays*) e EPLD's (*Electrically Programmable Logic Devices*). Uma FPLA tradicional é composta por células lógicas programáveis através da queima de fusíveis, em posições adequadas no circuito. As EPLD's são dispositivos programáveis onde as conexões podem ser apagadas através de radiação UV.

Entretanto, o máximo de flexibilidade de uso é atingido através de arquiteturas compostas por células programáveis que permitem realizar o projeto através de configuração por software. Neste caso, as alterações no projeto se traduzem por simples mudanças nas linhas do programa de configuração. Esta idéia deu origem a um novo modelo de computação conhecido como *hardware configurável dinamicamente*, que surge como uma evolução natural dos dispositivos programáveis eletricamente.

O hardware configurável é uma estrutura composta de células lógicas programáveis capazes de implementar um conjunto de funções e de estruturas programáveis para a conexão destas células, além de células de I/O programáveis, que tanto podem funcionar como células de entrada como de saída. Existem, ainda, decodificadores que realizam o endereçamento das células, durante a configuração. A figura 1 ilustra um esquema geral para este tipo de estrutura.

(*) NCE/UFRJ e-mail: NCD99070@UFRJ.BITNET
(**) Prog. de Eng. Elétrica COPPE/UFRJ e-mail:COE10003@UFRJ.BITNET
(***) Prog. de Eng. de Sistemas COPPE/UFRJ e-mail:COS05001@UFRJ.BITNET

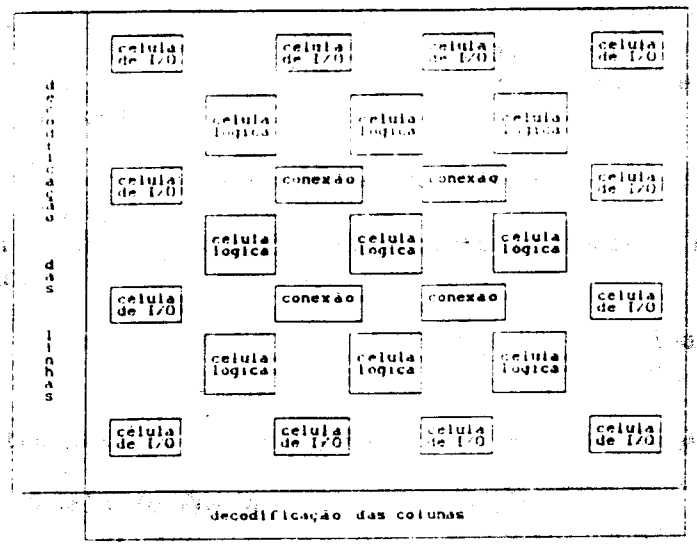


Figura 1 - Estrutura Geral do Hardware Configurável

Atualmente já existem estruturas programaveis por software comerciais, tais como o Logic Cell Array, da Xilinx [1] e o Multiple Array Matrix, da Altera [2].

Recentemente, T. Kean [3] propôs uma estrutura configurável por software, baseada em uma matriz de células lógicas simples compostas de multiplexadores, que permite um melhor aproveitamento dos recursos da estrutura.

O objetivo deste trabalho é apresentar um estudo das estruturas que podem ser usadas para implementar um hardware configurável e propor uma estrutura configurável baseada na proposta de T. Kean. Neste trabalho, são examinados os problemas de uma tal implementação, sendo proposto um *layout* para a célula lógica com sua caracterização elétrica, bem como uma avaliação das limitações de *performance* que afetam este tipo de estrutura (velocidade e área).

2. Estruturas Básicas para o Projeto de um Hardware Configurável

Um hardware configurável é composto, basicamente, de três tipos de células:

-células lógicas programáveis, que implementam funções lógicas de suas entradas,

-células de conexão programáveis, que permitem conectar entradas e saídas das células lógicas para formar a rede desejada e,

-células programáveis de I/O, que realizam a comunicação do hardware configurável com o mundo exterior.

2.1 - Células Lógicas Programáveis:

A célula lógica é composta de uma unidade funcional e uma estrutura programável que permite rotear os sinais para a unidade funcional. A unidade funcional é a estrutura responsável pela implementação da função lógica realizada pela célula.

O circuito de roteamento de sinais para a unidade funcional pode ser interno, constituindo parte integrante da célula lógica, ou externo. No primeiro caso, as conexões externas entre celulas lógicas podem ser fixas, simplificando consideravelmente o *layout*.

As várias possibilidades de implementação da unidade funcional são discutidas a seguir.

2.1 - Alternativas de Projeto da Unidade Funcional

2.1.1 - Tabela de Busca

Neste caso, utiliza-se uma RAM como uma tabela de busca. Uma função de n entradas pode ser descrita usando-se uma tabela verdade de 2^n colunas e pode ser implementada utilizando-se uma tabela de busca de 2^n bits. Os dados de entrada (mintermos) acionam um seletor que escolhe a célula de RAM que fornecerá a saída. Esta estrutura é utilizada pelo *Gate Array Programável* da Xilinx, e está ilustrada na figura 2.

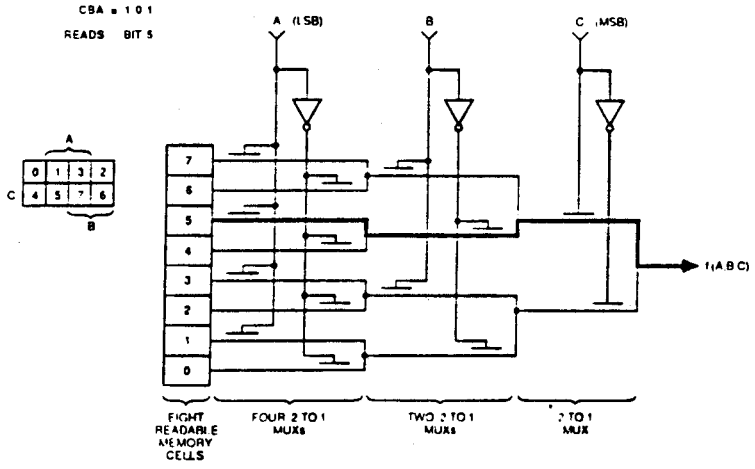


Fig.2 - Estrutura da Tabela de Busca

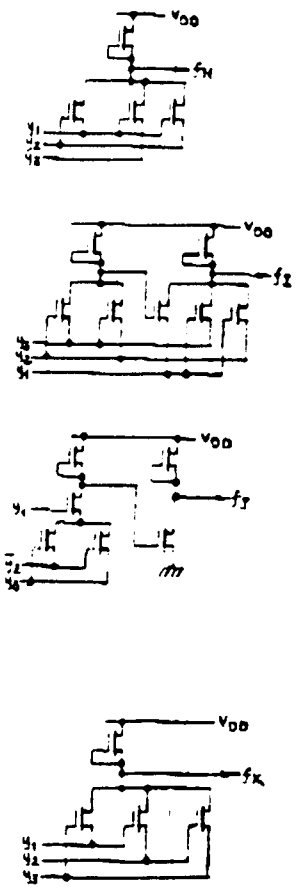
A estrutura apresentada na figura 2 é capaz de implementar funções de até 8 mintermos. Caso seja necessário implementar funções com menos de 8 mintermos, esta estrutura tem parte de seus recursos desperdiçados.

2.1.2 - Módulos Lógicos Universais

Um elemento lógico universal é um circuito capaz de realizar todas as possíveis 2^{2^n} funções de n variáveis de entrada binárias independentes $x_1, x_2, \ldots, x_n$, $n \geq 2$. A função implementada pelo MLU depende somente da ordem de conexões das entradas, pertencentes ao conjunto $\{0, 1, x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$.

Trabalhos de Yau e Orsic [4] e Preparata [5] apresentam várias propostas para a implementação de elementos lógicos universais.

Para tornar o projeto mais simples e eficiente no sentido do aproveitamento de recursos, é melhor utilizar unidades com, no maximo, duas entradas e uma saída. Estas unidades são capazes de implementar apenas uma operação lógica de cada vez. Funções mais complexas podem ser obtidas a partir da conexão destas unidades simples. Esta escolha evita o desperdício de recursos decorrentes da sub-utilização de estruturas maiores. Neste sentido, pesquisas recentes [6] mostraram que existem somente seis configurações possiveis de MLU com o número minimo de três terminais (capazes de implementar funções booleanas de duas variáveis). A figura 3 apresenta estas configurações bem como a tabela de programação.

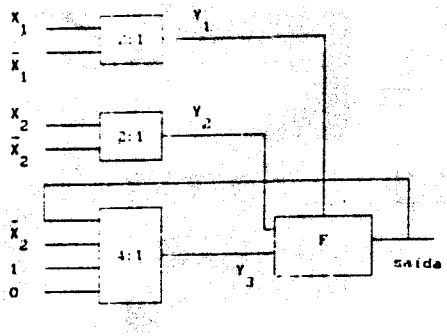


n ≤ 2 função	f_h			f_i			f_j			f_k			f_l			f_m		
	y_1	y_2	y_3	y_1	y_2	y_3	y_1	y_2	y_3	y_1	y_2	y_3	y_1	y_2	y_3	y_1	y_2	y_3
0	1	0	x2	1	x2	0	0	x2	dc	1	x2	0	x1	0	1	0	x2	1
1	0	0	x2	x2	x2	1	1	x2	x2	x2	x2	x2	x1	1	0	0	x2	0
x1	$\bar{x}1$	0	x2	$\bar{x}1$	x2	$\bar{x}2$	x1	x2	x2	1	x1	1	x1	1	1	0	x2	$\bar{x}1$
$\bar{x}1$	x1	0	x2	x1	x2	$\bar{x}2$	$\bar{x}1$	x2	x2	0	x1	0	x1	0	0	0	x2	x1
x2	$\bar{x}2$	0	x2	0	x2	0	1	x2	1	1	x2	1	x1	x2	$\bar{x}2$	1	x2	1
$\bar{x}2$	x2	0	x2	0	x2	1	1	x2	0	0	x2	0	x1	$\bar{x}2$	x2	1	x2	0
x1.x2	x1	1	x2	1	x2	x1	x1	x2	1	1	x2	x1	x1	x2	1	x1	x2	1
x1.$\bar{x}2$	x1	1	$\bar{x}2$	$\bar{x}1$	x2	x1	x1	x2	0	0	x2	$\bar{x}1$	x1	$\bar{x}2$	1	x1	$\bar{x}2$	$\bar{x}1$
$\bar{x}1$.x2	$\bar{x}1$	1	x2	1	x2	$\bar{x}1$	$\bar{x}1$	x2	1	1	x2	$\bar{x}1$	x1	0	$\bar{x}2$	$\bar{x}1$	x2	1
$\bar{x}1$.$\bar{x}2$	$\bar{x}1$	1	$\bar{x}2$	x1	x2	$\bar{x}1$	$\bar{x}1$	x2	0	0	x2	x1	x1	0	x2	$\bar{x}1$	x2	x1
x1+x2	$\bar{x}1$	$\bar{x}1$	x2	x1	x2	$\bar{x}1$	$\bar{x}1$	x2	0	0	x2	x1	x1	1	$\bar{x}2$	$\bar{x}1$	x2	$\bar{x}1$
x1+$\bar{x}2$	$\bar{x}1$	$\bar{x}1$	$\bar{x}2$	1	x2	$\bar{x}1$	$\bar{x}1$	x2	1	1	x2	$\bar{x}1$	x1	1	x2	$\bar{x}1$	x2	0
$\bar{x}1$+x2	x1	x1	x2	$\bar{x}1$	x2	x1	x1	x2	0	0	x2	$\bar{x}1$	x1	x2	0	x1	x2	x1
$\bar{x}1$+$\bar{x}2$	x1	x1	$\bar{x}2$	1	x2	x1	x1	x2	1	1	x2	x1	x1	$\bar{x}2$	0	x1	x2	0
x1⊕x2	$\bar{x}1$	x2	x2	0	x2	$\bar{x}1$	1	x2	$\bar{x}1$	x2	x2	$\bar{x}1$	x1	$\bar{x}2$	$\bar{x}2$	1	x2	$\bar{x}1$
$\overline{x1⊕x2}$	x1	x2	x2	0	x2	x1	1	x2	x1	x2	x2	x1	x1	x2	x2	1	x2	x1

Fig.3 - Configurações para MLU
e Tabela de Programação

2.1.3 - Circuito Composto por Multiplexadores

Todas as funções lógicas de duas variáveis de entrada (X_1 e X_2) podem ser computadas utilizando-se um multiplexador 2:1 (indicado por F na figura 4), que seleciona Y_2 se $Y_1 = 0$ e Y_3 se $Y_1 = 1$. Este circuito também só é capaz de implementar apenas uma operação lógica de cada vez, não apresentando, como conseqüência, o problema do desperdício de recursos.



Nº	Função	Y1	Y2	Y3
0	ZERO	$X_1 = X_2$	X_2	0
1	UM	$X_1 = X_2$	$\overline{X_2}$	1
2	X_1	X_1	$X_2 = X_1$	1
3	$\overline{X_1}$	$\overline{X_1}$	$\overline{X_1} = \overline{X_2}$	1
4	X_2	$X_1 = X_2$	X_2	1
5	$\overline{X_2}$	X_1	$\overline{X_2}$	$\overline{X_2}$
6	$X_1 \cdot X_2$	$\overline{X_1}$	X_2	0
7	$X_1 \cdot \overline{X_2}$	$\overline{X_1}$	$\overline{X_2}$	0
8	$\overline{X_1} \cdot X_2$	X_1	X_2	0
9	$\overline{X_1} \cdot \overline{X_2}$	X_1	$\overline{X_2}$	0

Nº	Função	Y1	Y2	Y3
10	$X_1 + X_2$	X_1	X_2	1
11	$X_1 + \overline{X_2}$	X_1	$\overline{X_2}$	1
12	$\overline{X_1} + X_2$	$\overline{X_1}$	X_2	1
13	$\overline{X_1} + \overline{X_2}$	$\overline{X_1}$	$\overline{X_2}$	1
14	$X1 \oplus X_2$	$\overline{X_1}$	X_2	$\overline{X_2}$
15	$\overline{X1 \oplus X_2}$	X_1	X_2	$\overline{X_2}$
16	D Latch	X_1 = Clk	X_2 = D	Saída
17	$\overline{\text{D Latch}}$	X_1 = Clk	$X_2 = \overline{D}$	Saída
18	D $\overline{\text{Clk}}$ Latch	X_1 = Clk	X_2 = D	Saída
19	$\overline{\text{D CLK}}$ latch	X_1 = Clk	$X_2 = \overline{D}$	Saída

Fig. 4 - Circuito Composto por Multiplexadores

2.2 - Conexão entre Células Lógicas

2.2.1 - Topologias para a Conexão de Células Lógicas

As alternativas mais interessantes, do ponto de vista de paralelismo, para se conectar células lógicas são:

-conexão de cada célula diretamente com todas as outras
-conexão de cada célula apenas com os vizinhos mais próximos
-conexão das células na forma de um cubo

A conexão de cada célula diretamente com todas as outras pode ser implementada através de barramentos. Neste caso, há um sinal do barramento conectado a cada entrada e saída de cada célula lógica. É fácil perceber que a desvantagem deste tipo de conexão é o elevado consumo de área decorrente da dimensão do barramento.

A conexão de cada célula apenas com os vizinhos mais próximos pode ser realizada facilmente através de circuitos tipo *crossbar switch* ou multiplexadores.

A conexão das células na forma de cubo pode ser realizada através da rede cubo [7], que é uma rede multi-estágio particularmente útil quando se deseja uma estrutura multi-planar a nível lógico.

2.2.2 Roteamento de Sinais Intra-Célula x Entre-Células

O roteamento de sinais entre-células exige que as conexões externas entre células lógicas sejam programáveis. Já o roteamento de sinais intra-célula permite que as conexões externas entre células sejam fixas, e a escolha dos sinais ocorre dentro da própria célula.

No roteamento entre-células, as conexões externas entre saídas e entradas de células lógicas podem ser feitas através de chaves, que habilitam ou não a conexão, dependendo da programação efetuada. A figura 5 mostra um exemplo, onde cada célula lógica possui duas entradas e duas saídas. O sistema de chaveamento está indicado pela letra S.

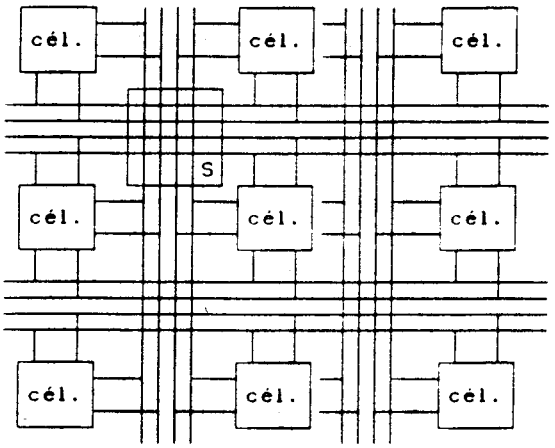


Fig.5 - Conexões Externas Programáveis

No caso do roteamento de sinais feito pela própria celula, circuitos seletores internos realizam a escolha de sinais. Estes circuitos seletores são construídos a partir de multiplexadores, que definem os sinais que devem ser usados como entrada para a unidade funcional e os que devem ser fornecidos como saída da célula. A figura 6 ilustra um circuito deste tipo. Neste exemplo, considera-se uma celula logica com 4 entradas (norte, sul, leste e oeste) e 4 saídas (norte, sul, leste e oeste).

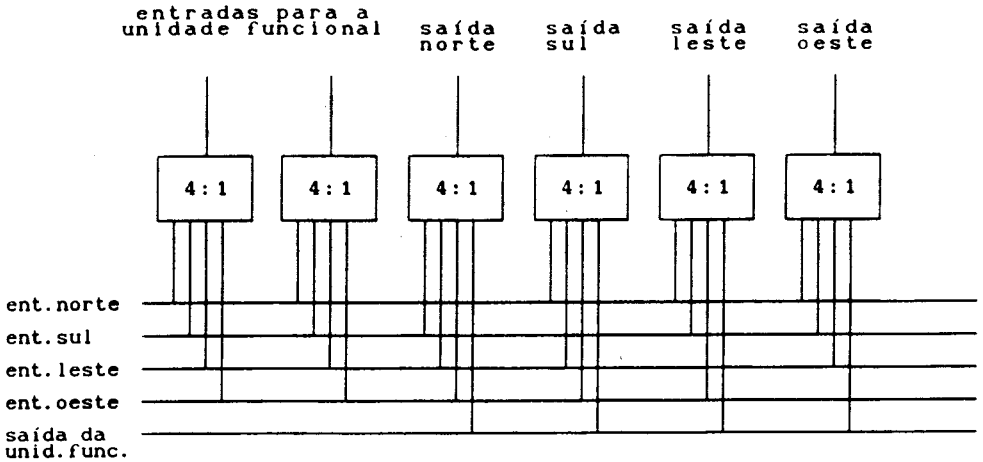


Fig. 6 - Circuito de Roteamento

3. Célula de I/O

As células de I/O devem ser programáveis de modo a poderem funcionar tanto como célula de entrada quanto de saída.
A.estrutura usual neste caso é o PAD bidirecional [8].

4. Descrição do Projeto

O plano de lógica configurável proposto compõe-se de uma matriz de células lógicas cercada por um anel de células de I/O. As células lógicas são estruturas simples, capazes de implementar funções lógicas de duas variáveis. A filosofia de conexão entre células lógicas adotada é a conexão com o vizinho mais próximo, ou seja, uma célula só é capaz de se comunicar diretamente com as células imediatamente acima, abaixo, a direita e a esquerda. As conexões externas entre as células são fixas, e todo o roteamento de sinais é feito pela própria célula lógica (fig. 7).

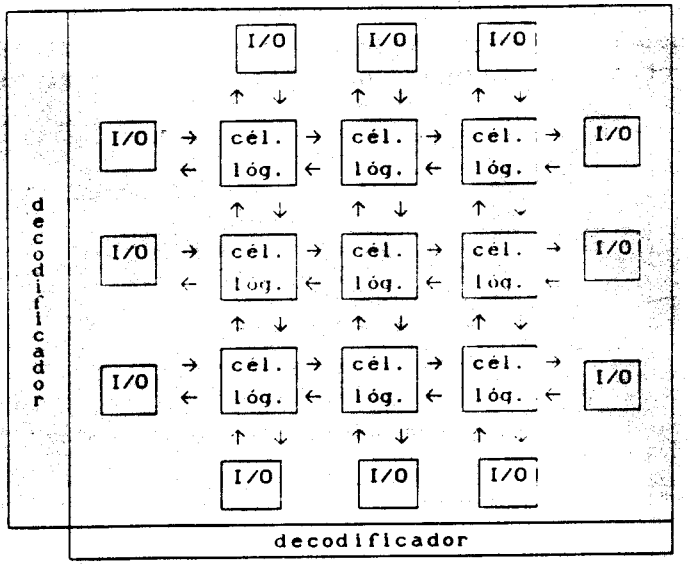


Fig. 7 - Plano

A programação das células é feita por 2 decodificadores: um para endereçar as linhas e outro para as colunas em que estão localizadas as células. Cada decodificador apresenta somente uma saída ativa por vez e o sinal de habilita escrita é gerado a partir de um AND dos sinais de linha e coluna. A palavra de programação tem o formato apresentado na figura 8.

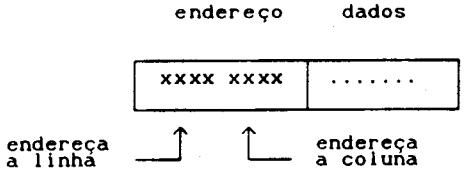


Fig.8 - Palavra de Programação

4.1 - Estrutura da Célula Lógica

A célula lógica possui 4 entradas (norte,sul,leste,oeste) e 4 saídas (norte,sul, leste,oeste) para realizar a conexão com os vizinhos mais próximos. Ela possui ainda duas entradas globais (G1 e G2) para sinais de *clock*.

A célula lógica é composta de uma unidade funcional e um circuito interno de roteamento de sinais. A unidade funcional, por seu turno, é composta de multiplexadores, como apresentado em 2.1.3. O circuito interno

de roteamento também é construído a partir de multiplexadores, como discutido em 2.2.2. A opção pelos multiplexadores tem como objetivo tornar o *layout* o mais regular possível. Os problemas de degradação de sinal são evitados pelo uso de inversores na saída de cada multiplexador.

4.2 - Célula de I/O

As células lógicas na periferia do array podem fornecer os sinais de controle para as células de I/O, que nada mais são do que *PADS* bi-direcionais. Segundo esta filosofia, as células lógicas são agrupadas em pares: a saída da primeira célula aciona a linha de dados e a saída da segunda habilita a linha para o *PAD*. A entrada do *PAD* é conectada às linhas de entrada de ambas as células. Este esquema está ilustrado na figura 9.

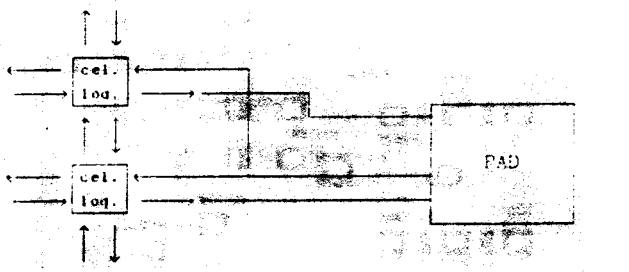


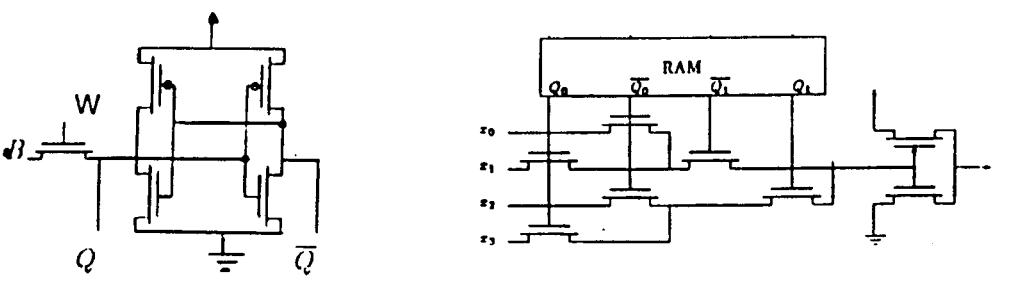
Fig.9 - Esquema de Conexão das Células de I/O

5. Implementação em VLSI e Caracterização Elétrica

O *layout* das células do plano foi editado no TEDMOS/NCE [9] e as regras de projeto adotadas foram as da ES2 2μm. Estas células foram simuladas eletricamente com o auxílio do PSPICE [10].

A célula lógica é construída a partir do conjunto multiplexador + célula de RAM, e é este conjunto que determina o tamanho do *chip*. A célula é composta de multiplexadores 2:1,4:1 e 6:1. A título de ilustração, será apresentada a seguir a estrutura do multiplexador 4:1, sendo que os demais podem ser facilmente obtidos a partir deste.

A célula de RAM utilizada é estática, composta de 5 transitores (fig.10.a). Esta célula oferece a linha de bit e a linha de bit barrada, sendo adequada para se usar em conjunto com o multiplexador (fig.10.b).



(a) (b)

Fig.10 - Célula de RAM e Multiplexador

A estrutura de multiplexador usada é compacta e a saída *bufferizada* evita problemas de degradação do sinal.

O conjunto apresentado na figura 10 foi projetado para trabalhar a uma freqüência de 25MHz.O atraso introduzido por este conjunto é da ordem de 3ns. O *layout* é mostrado na figura 11.

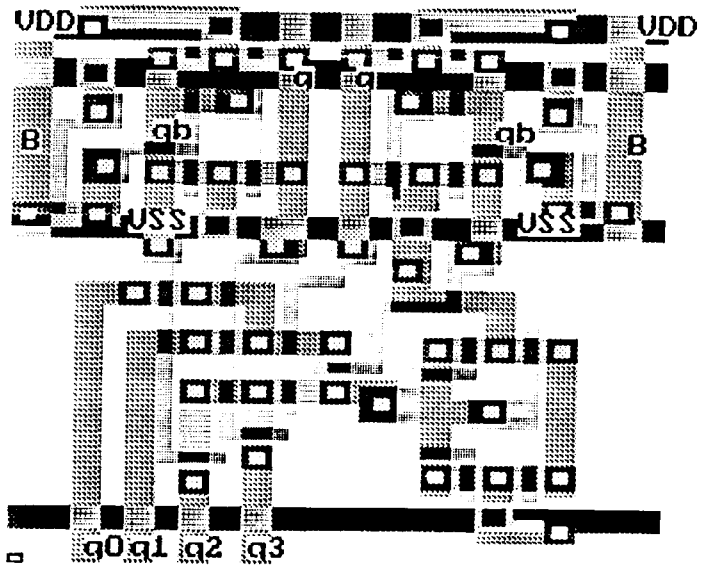


Fig. 11 - Layout do Multiplexador 4:1

O *layout* da célula lógica é apresentado na figura 12. As linhas de programação das células de RAM correm em metal 2, aproveitando o espaço interno da célula. Cada célula lógica possui dimensão 377μm x 259μm.

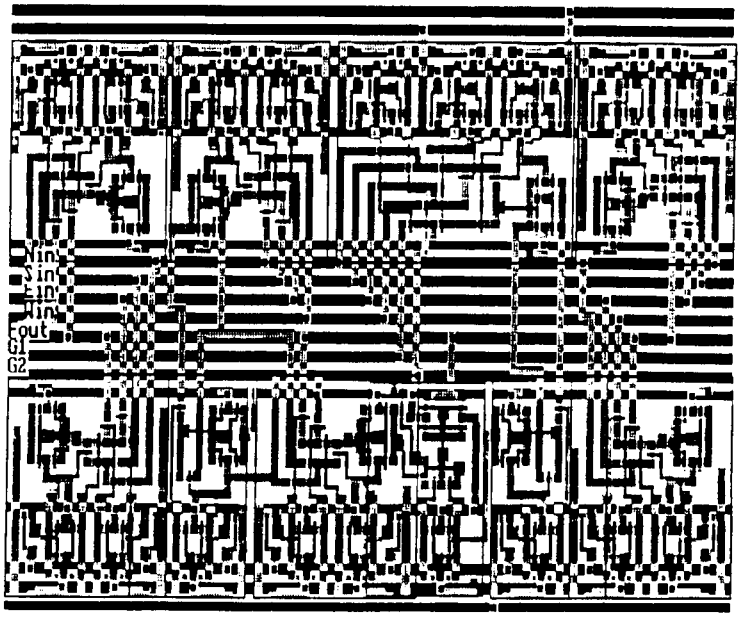


Fig.12 - Layout da Célula Lógica

O atraso introduzido, em média, por cada célula lógica ao executar uma operação lógica, é da ordem de 10 ns.

6. Conclusão

O *hardware* configurável por *software* é particularmente útil para a pesquisa nas universidades e centros de desenvolvimento, uma vez que projetos de arquiteturas digitais podem ser testados e modificados através de programação, evitando o problema de confeccionar máscaras para cada projeto testado. As alterações são implementadas através do programa de configuração.

A estrutura configurável consome um pouco mais de área e apresenta uma pequena diminuição de velocidade, devido à introdução de chaves, porém, estas desvantagens são plenamante compensadas pela flexibilidade fornecida pela estrutura.

O projeto apresentado neste trabalho utilizou as regras 2μm da ES2. Trabalhos futuros têm como objetivo implementar estas células utilizando-se regras 1.5μm, visando uma futura integração no PMU (Projeto Multi-Usuário).

Referências

[1] Xilinx Inc., *The Programmable Gate Array Design Handbook.* San Jose, CA., 1986.

[2] Altera Corporation. *Altera Data Book 1987.* Altera Corporation, Santa Clara, California, 1987.

[3] Thomas Kean, *Configurable Logic: A Dynamically Programmable Cellular Architecture and its VLSI Implementation*, University of Edinburgh, 1989.

[4] S.S. Yau & M. Orsic, *Universal Logic Modules*, in *Proc. 3rd. Princeton Conf. Inform. Sci. Syst.*, Março 1969, pp.498-502.

[5] F.P. Preparata, *On the Design of Universal Boolean Functions*, in *IEEE Transactions on Computers*, Abril 1971, pp.418-423.

[6] X. Chen & S.L. Hurst, *A consideration of the Minimum Number of Input Terminals on Universal Logic Gates and Their Realization*, in *Int. J. Electron.*, vol.50, pp. 1-13, 1981.

[7] Kai Hwang & Fay A. Briggs, *Computer Architecture and Parallel Processing*, Mcgraw-Hill, 1985.

[8] A.M. Meslin ET AL, *Um Estudo dos Problemas de Projeto de Matrizes de Portas com Tecnologia CMOS*, Relatório Técnico NCE-30/90, 70pp, Novembro 1990.

[9] Eber Schmidt, *Manual do Tedmos*, NCE/UFRJ,1991.

[10] *Manual do PSPICE.*