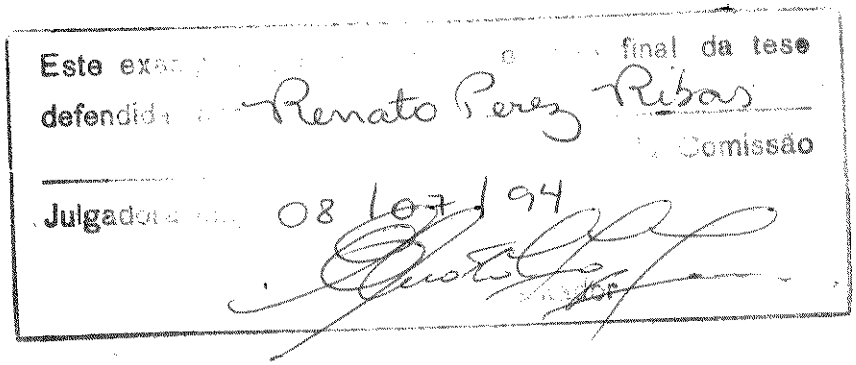
UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ELETRÔNICA E MICROELETRÔNICA



Estudo e uso de componentes de lógica programável (PLDs) em um ambiente de prototipagem rápida genérico

por

Renato Perez Ribas

Dissertação submetida como requisito
parcial para a obtenção do grau de
Mestre em Engenharia Elétrica

Prof. Dr. Elnatan Chagas Ferreira
Orientador

Campinas, Julho de 1994.

Sumário

Agradecimentos

Inicialmente, gostaria de agradecer ao Instituto de Microeletrônica da Fundação Centro Tecnológico para Informática (CTI), em especial ao Prof. Carlos I.Z. Mammana, que me deu a oportunidade de realizar este trabalho como uma extensão do projeto de prototipagem rápida em desenvolvimento neste centro de pesquisa.

Às empresas Hicad Sistema Ltda e União Digital Comércio e Representações Ltda, na pessoa dos Engs. Vanderlei Perez Sanchez e José Carlos Stagni, respectivamente, que empresentaram os sofwares de Xilinx e da Altera para realizar o estudo comparativo apresentado no capítulo 4.

Aos Engs. Luiz Manoel Madureira e Carlos Krüger, ambos do CPqD (TELEBRÁS), e Rivaldo de Oliveira Ferreira, da Escola Politécnica da USP, que participaram do desenvolvimento deste trabalho.

Ao meu orientador Elnatan Ferreira que contribuiu fortemente para que esta dissertação fosse concluída.

Ao meu chefe, orientador e grande amigo Frank Behrens, a quem devo muito do meu crescimento pessoal e profissional nos últimos três anos. Fernando Chavez, Paulo Anteguini Filho, Joao Adalberto Pereira, Saulo Finco, Walter De Marco, Vera Lídia, Susete e a todos com quem convivi nestes últimos anos.

Aos Profs. Ricardo Reis e Sérgio Bampi da Universidade Federal do Rio Grande do Sul, onde este trabalho teve início. Aos amigos e colegas José Luis Güntzel, André Reis e Luigi Carro, pelas nossas divagações e investidas.

E por fim, a meus pais, irmãos e principalmente a mais amável, paciente, companheira e amiga de todas as esposas, Nilseia Lapresa Ribas, pelo árduo caminho que trilhamos juntos para chegarmos onde estamos e que, tenho certeza, iremos muito mais longe.

À minha esposa, meus pais, irmãos e aos verdadeiros amigos que encontrei e cativei nestes anos de vida.

"A felicidade de uma vida deve-se medir pelo progresso do indivíduo no caminho da perfeição moral, e não pela soma das cobiças satisfeitas ou pela ausência de desejos!"

Lista de abreviaturas

ASIC	*Application Specific Integrated Circuit*
ASSP	*Application Specific Standard Product*
CAD	*Computer-Aided Design*
CI	circuito integrado
CL	célula lógica
CLB	*Configurable Logic Block*
CMOS	*Complementary Metal-Oxide-Semiconductor*
CPqD	Centro de Pesquisa e Desenvolvimento (TELEBRÁS)
CTI	Fundação Centro Tecnológico para Informática
DRC	*Design Rules Check*
EPLD	*Erasable Programmable Logic Device*
EPROM	*Erasable Programmable Read-Only Memory*
E2PROM	*Electrical Erasable Programmable Read-Only Memory*
ERC	*Electrical Rules Check*
ES2	*European Silicon Structures*
FPGA	*Field Programmable Gate Array*
FPLA	*Field Programmable Logic Array*
HAL	*Hardwire Array Logic*
HDL	*Hardware Description Language*
I/O	*Input / Output*
LAB	*Logic Array Block*
LCA	*Logic Cell Array*
LPCI	*Laboratório de Projetos de Circuitos Integrados (CTI)*
LSI	*Large Scale Integrated*
LVS	*Layout Versus Schematic*
MOS	*Metal-Oxide-Semiconductor*
MSI	*Medium Scale Integrated*
NRE	*Non-Recurring Engineer*
PAL	*Programmable Array Logic*
PC	*Personal Computer*
PCB	*Printed Circuit Board*
PIA	*Programmable Interconect Array*
PLA	*Pragrammable Logic Array*
PLD	*Programmable Logic Device*
PLE	*Programmable Logic Element*
PLS	*Programmable Logic Sequencer*
PML	*Programmable Macro Logic*
PREP	*Programmable Electronics Performance Corporation*
PROM	*Programmable Read-Only Memory*
RAM	*Read-Access Memory*
ROM	*Read-Only Memory*
SRAM	*Static Read-Access Memory*
SSI	*Small Scale Integrated*
ULSI	*Ultra-Large Scale Integrated*
VHDL	*VLSI Hardware Description Language*
VLSI	*Very Large Scale Integrated*

Lista de figuras

Lista de tabelas

Resumo

Este trabalho apresenta um estudo sobre os componentes de lógica programável (PLDs) e o seu uso em um ambiente de prototipagem rápida baseado na descrição esquemática genérica de circuitos digitais. O ambiente proposto visa facilitar a migração de ASICs através das formas de implementação disponíveis.

Para atingir tal objetivo foi necessário realizar um estudo aprofundado sobre os PLDs de última geração e suas ferramentas de CAD, a fim de possibilitar a implantação do ambiente proposto e a análise da aplicabilidade desta tecnologia frente aos tradicionais *gate arrays* e *standard cells*.

Este ambiente encontra-se especificado, as incompatibilidade para sua implantação foram cuidadosamente avaliadas, realizou-se uma análise detalhada dos fabricantes de PLDs (seus componentes e ferramentas de projeto) e o programa para a conversão dos esquemáticos encontra-se operacional. Resta, atualmente, apenas aguardar a aquisição da ferramenta MaxPlus2 por parte do Instituto de Microeletrônica da Fundação Centro Tecnológico para Informática, onde o ambiente será instalado, para que ele esteja realmente em uso.

Abstract

This work presents a study on Programmable Logic Devices (PLDs) and their use in a fast prototyping environment, based on a generic schematic description of digital circuits. The proposed environment looks an easy migration of ASICs through the available implementation styles.

In order to reach the goals it was necessary to carry out a deep study about the last generation PLDs together with their CAD tools; this study makes possible the implantation of the proposed environment and realize an analisys about the applicability of this technology against traditional gate arrays and standard cells.

Nowadays, this environment is specified. Uncomptibilities concerning with its implantation were carefully studies. Also, it was carried out an detailed analisys about PLDs (their components and design tools) and the software to schematic conversion is ready. Finally, the Microelectronic Institut of the Fundação Centro Tecnológico para Informática (CTI), where the environment will be implanted, is buying the sofware MaxPlus2 (Altera).

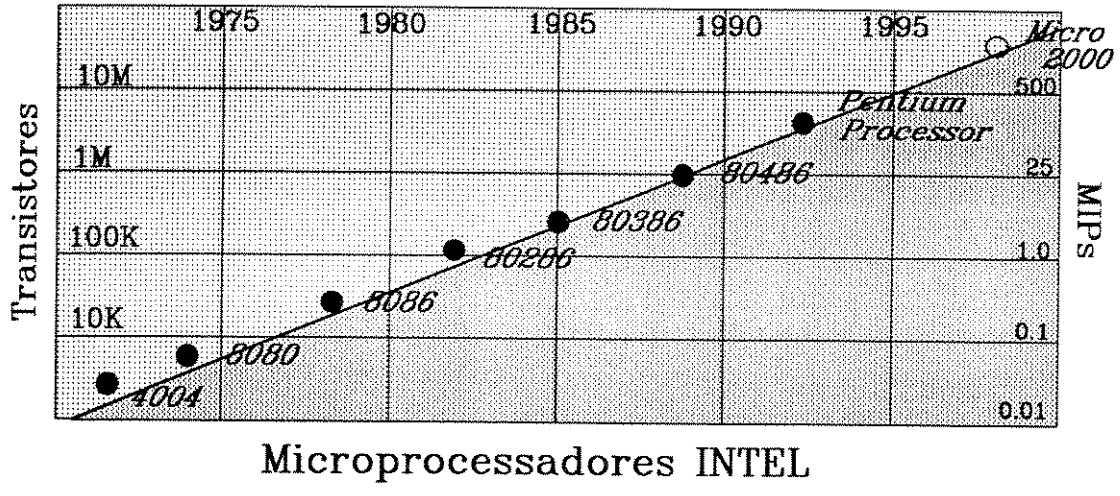
Capítulo 1

Introdução

O desenvolvimento dos dispositivos semicondutores foi o principal responsável pela popularização da eletrônica que começou por volta dos anos 60. Este processo ganhou grande impulso a partir do momento em que vários transistores passaram a ser integrados numa mesma pastilha semicondutora (de silício, por exemplo), originando o que se convencionou chamar de circuito integrado (**CI**).

A primeira demonstração de um CI foi analógica, em 1958, por Jack Kilby, embora os primeiros produtos comerciais tenham sido os componentes digitais da série 51, da Texas Instruments [1] [2]. Logo notou-se um perfeito casamento entre a tecnologia de circuitos integrados e a lógica digital, principalmente pelo fato desta última necessitar grande quantidade de componentes, porém com funções básicas bastante simples.

A partir dos anos 60, a capacidade de integração foi impulsionada pela demanda gerada com a utilização da lógica digital nos mais diversos produtos eletrônicos: de pouco mais de 100 transistores em 1970, passou-se para 10.000 em 1980 e para 1 milhão em 1990 [3] [4]. Na figura 1.1 pode ser observada a evolução dos microprocessadores da Intel Corp., segundo o seu número de transistores e o número de instruções por segundo (MIPs).



Microprocessadores INTEL

Figura 1.1 - Evolução dos microprocessadores da Intel Corp.

Para que este aumento da capacidade de integração fosse possível, foram decisivos os avanços na área de processos de fabricação, principalmente no sentido de se obter transistores menores (mais rápidos, conseqüentemente) e com menor consumo de potência. Neste sentido, a adoção do processo **CMOS**

(*Complementary Metal-Oxide-Semiconductor*) viabilizou a integração de microprocessadores inteiros, o que era impossível anteriormente devido aos problemas de dissipação de calor e rendimento de fabricação (*yield*).

Com o intuito de facilitar a classificação dos CIs em termos de quantidade de transistores, são geralmente utilizadas as siglas **SSI**, **MSI**, **LSI, VLSI** e **ULSI**. As fronteiras entre tais faixas de integração não são rígidas.

Uma possível divisão, segundo a literatura [5] [6], poderia ser:

- SSI (*Small Scale Integration*) - designa circuitos com até 500 transistores;
- MSI (*Medium Scale Integration*) - designa circuitos que possuem entre 500 e 5.000 transistores;
- LSI (*Large Scale Integration*) - designa circuitos que possuem entre 5.000 e 50.000 transistores;
- VLSI (*Very Large Scale Integration*) - designa circuitos que possuem entre 50.000 e 1.000.000 transistores;
- ULSI (*Ultra-Large Scale Integration*) - designa circuitos que possuem mais de 1.000.000 transistores.

Esta rápida evolução da tecnologia dos circuitos integrados é o resultado de um esforço concentrado que envolve centenas de milhares de profissionais nas diferentes áreas relacionadas com a microeletrônica e tem causado um grande impacto sobre o mercado de produtos eletrônicos.

Inicialmente, um sistema digital era composto por uma ou várias placas de circuito impresso (**PCB** - *printed circuit board*), contendo inúmeros CIs de baixa complexidade. Com a facilidade de se realizarem mais componentes numa mesma pastilha, passou a ser viável a integração de partes de sistemas ou até mesmo sistemas inteiros sob encomenda, para circuitos a serem utilizados em produtos com grande demanda. Por outro lado, a utilização em massa de CIs tais como memórias e microprocessadores, e o lançamento contínuo de circuitos cada vez mais complexos resultaram na queda dos preços no mercado dos produtos de microletrônica.

A queda dos custos de desenvolvimento e fabricação dos CIs, aliada à crescente necessidade do mercado de implementar funções bastante específicas para os produtos de uma única empresa, propiciaram o surgimento dos chamados **circutos integrados de aplicação específica (ASICs)**, por volta dos anos 70.

Além do custo final do produto, a concorrência acirrada do mercado de eletrônica fez com que fatores como desempenho, confiabilidade e sigilo de projeto passassem a pesar no momento de se decidir pelo uso da microeletrônica, mesmo para produtos que não apresentavam grande demanda.

1.1 ASICs, ASSPs e componentes *standard*

Durante muito tempo, projetos de circuitos eletrônicos eram realizados apenas com componentes *standard* ou *off-the-shelf*, como também são referenciados na literatura. Nos últimos anos, os produtos eletrônicos têm cada vez mais incorporado circuitos integrados desenvolvidos especialmente para eles. Estes circuitos podem simplesmente substituir alguns dos componentes *standard* utilizados ou mesmo implementar o sistema completo. Tais circuitos dedicados são denominados **ASSPs** (*Application Specific Standard Products*) e **ASICs** (*Application Specific Integrated Circuits*). Neste trabalho serão considerados apenas circuitos digitais, apesar dos conceitos de componentes *standard*, ASSPs e ASICs serem estendidos também à eletrônica analógica.

Dentro deste contexto, definem-se **componentes *standard*** como sendo aqueles cujas funções lógicas são de uso geral para todas as áreas da eletrônica digital, por exemplo a família 74 TTL [7] [8]. Estes componentes estão catalogados nos *data books* dos fabricantes, onde o usuário obtém suas especificações funcionais e elétricas, e são facilmente encontrados no mercado. Seu preço unitário é relativamente baixo devido a sua produção em grande escala.

Os **ASSPs** são componentes utilizados para aplicações específicas em sistemas digitais, projetados por alguns fabricantes de semicondutores e estão disponíveis para qualquer usuário, sem sigilo de funcionamento, ou seja, eles são apresentados nos *data books* destes fabricantes [9].

A diferença dos ASSPs para os componentes *standard*, é que os últimos são produzidos por muitos fabricantes de semicondutores e servem para variadas aplicações digitais, por representarem circuitos básicos como portas lógicas NANDs, NORs, inversores, *latches*, *flip-flops*, entre outras. Já os ASSPs são componentes lançados no mercado, por um certo fabricante, devido ao intenso uso de um determinado circuito digital de média complexidade, justificando o seu projeto em uma única pastilha semicondutora, ao invés de implementá-lo sobre uma placa de circuito impresso, com componentes *standard* [10].

Os ASSPs não são facilmente encontrados no mercado, como ocorre com os componentes *standard*, mas qualquer usuário pode adquiri-los diretamente do fabricante, sem a existência de sigilo de projeto.

Os **ASICs**, por sua vez, são componentes cujas funções lógicas também estão direcionadas para uma aplicação específica, porém são implementados para o uso de um único encomendante. Estes componentes surgiram da necessidade de circuitos mais complexos e de funções cada vez mais específicas para determinadas aplicações ou produtos, além da necessidade de sigilo sobre o funcionamento dos circuitos, devido à crescente concorrência do mercado eletrônico. Os ASICs são produzidos na quantidade requerida pelo encomendante, que é o único praticamente a conhecer seu funcionamento lógico, além do próprio fabricante.

Desde a sua introdução, os ASICs têm apresentado uma crescente participação no mercado da eletrônica digital. Em 1982, menos de 25% dos dólares gastos em circuitos lógicos foram em ASICs. Já em 1988, 40% do mercado de circuitos integrados correspondeu à implementação destes componentes, devendo ter atingido cerca de 60% em 1993 [3].

A utilização de ASICs na elaboração de um produto eletrônico resulta numa série de vantagens, tanto sob o ponto de vista do fabricante como da qualidade do produto final:

- **Melhor desempenho elétrico do sistema**: isto decorre da diminuição dos atrasos de propagação dos sinais devido ao fato de que um maior número de conexões são realizadas no nível do circuito integrado, e não em trilhas sobre a placa. Também, a carga capacitiva representada pelos pinos dos CIs é diminuída por causa da redução do número de componentes utilizados. Tudo isto resulta em aumento da freqüência de funcionamento do sistema;

- **Menor área de placa para implementação do sistema**: esta característica é o resultado da redução do número de CIs utilizados, permitindo que o produto final seja mais compacto e mais leve;

- **Menor consumo de energia**: com menos CIs na placa, a tendência é que o consumo diminua bastante (a menos que o ASIC incorpore muitas funções adicionais que o circuito original não dispunha);

- **Maior confiabilidade do sistema**: com menos pontos de solda, menor número de trilhas na placa de circuito impresso e menor número de CIs, diminui a possibilidade de ocorrer defeitos na montagem final do produto ou mesmo durante sua vida útil;

- **Aumento do sigilo sobre o projeto**: a utilização de ASICs dificulta muito a realização de cópias de um produto, pois exige que seja feita engenharia reversa sobre o leiaute do ASIC, o que é muito custoso e demorado;

- **Facilidade em adicionar funções ou alterar o sistema**: o uso da microeletrônica proporciona a liberdade de alterar o circuito, adicionando-lhe novas funções ou características que seriam difíceis de serem implementadas com o uso de componentes *standard*.

Porém, existem várias formas de se implementar um ASIC, cada uma sendo mais apropriada para uma determinada relação entre demanda e desempenho. De uma forma genérica, existem os custos relacionados com o desenvolvimento (especificação, análise de viabilidade, projeto lógico, simulações, leiautes de máscaras, fabricação de protótipos, testes, possíveis reprojetos e entrada em produção) e os custos de fabricação durante a produção, que estão relacionados com a demanda do produto, com seu tempo de obsolescência e com o tempo de amortização do custo de desenvolvimento.

Todos estes fatores devem ser considerados na decisão de se fazer ou não um ASIC e principalmente na escolha da forma de implementação a ser utilizada.

1.2 Classificação das formas de implementação

Há diversas maneiras de se implementar um ASIC digital: o circuito pode ser contruído sobre uma pastilha de silício, ou outro semicondutor, totalmente "limpa", quer dizer, cada transistor do circuito deve ser definido e contruído sobre a pastilha; ou então, o circuito pode ser implementado sobre uma matriz de transistores já pré-difundidos sobre a pastilha, restando apenas utilizar os níveis de metalização para ligar estes transistores, uns nos outros, conforme a lógica desejada; e, numa terceira possibilidade, o circuito pode ser criado com o uso de componentes programáveis, através da queima de fusíveis ou da configuração de pontos de memória, por exemplo, sem a necessidade de interação com o processo de fabricação do CI.

A literatura tem procurado classificar as formas de implementação de ASICs utilizando os mais variados critérios como a maneira de realizar o leiaute ou as etapas de processo de fabricação já realizadas (*custom* e *semi-custom*).

Mas com o surgimento de novas formas de implementação, diferentes das tradicionalmente utilizadas, tornou-se difícil manter a classificação clássica dos ASICs, ainda amplamente utilizada. Esta forma clássica baseia-se na análise simultânea das etapas de projeto e de fabricação já realizadas (ou nas faltantes) e classifica os circuitos como *full custom*, *standard cells* e *gate arrays*. A diferenciação entre *full custom* e *standard cells* deve-se às etapas de projeto já realizadas (bibliotecas de células prontas, ou não), porém a diferenciação entre *gate arrays* e as outras duas deve-se principalmente às etapas de processo de fabricação por realizar.

Talvez uma maneira mais adequada seja classificar as formas de implementação de ASICs segundo o momento no qual o CI é diferenciado (personalizado), conforme sugere-se em [11] e [12] (figura 1.2):

- **CIs personalizáveis por todas as máscaras**;
- **CIs personalizáveis pela metalização**;
- **CIs personalizáveis após o encapsulamento**.

Os ASICs **personalizáveis por todas as máscaras** têm seu processo de fabricação diferenciado desde as primeiras etapas (processo inicial). Esta situação faz com que o cliente seja responsável pelo custo total do desenvolvimento, o qual deverá ser amortizado durante a produção do produto que utiliza o CI projetado. Como todas as etapas devem ser realizadas, o tempo de fabricação é mais longo do que nas demais formas de implementação. Por outro lado, há total liberdade na elaboração do leiaute, possibilitando o máximo aproveitamento possível da área da pastilha.

As definições clássicas de ***standard cell*** e ***full custom*** estão incluídos neste primeiro grupo das formas de implementação de ASICs. A abordagem *standard cell* consiste no uso de uma biblioteca de células pré-projetadas e pré-caracterizadas que reduz o tempo de geração do leiaute do circuito. A abordagem *full custom*, por sua vez, corresponde a elaboração manual do leiaute, sem a existência de sub-circuitos ou células já realizadas.

Para a classificação de circuitos mistos que utilizam os dois estilos na elaboração do seu leiaute, define-se como circuito *standard cell* aquele que apresenta mais de 50% de seu leiaute formado por sub-blocos proveniente de uma biblioteca de células. No caso de mais da metade deste leiaute ter sido gerado manualmente, durante o projeto do ASIC, este será classificado como um circuito *full custom* [3].

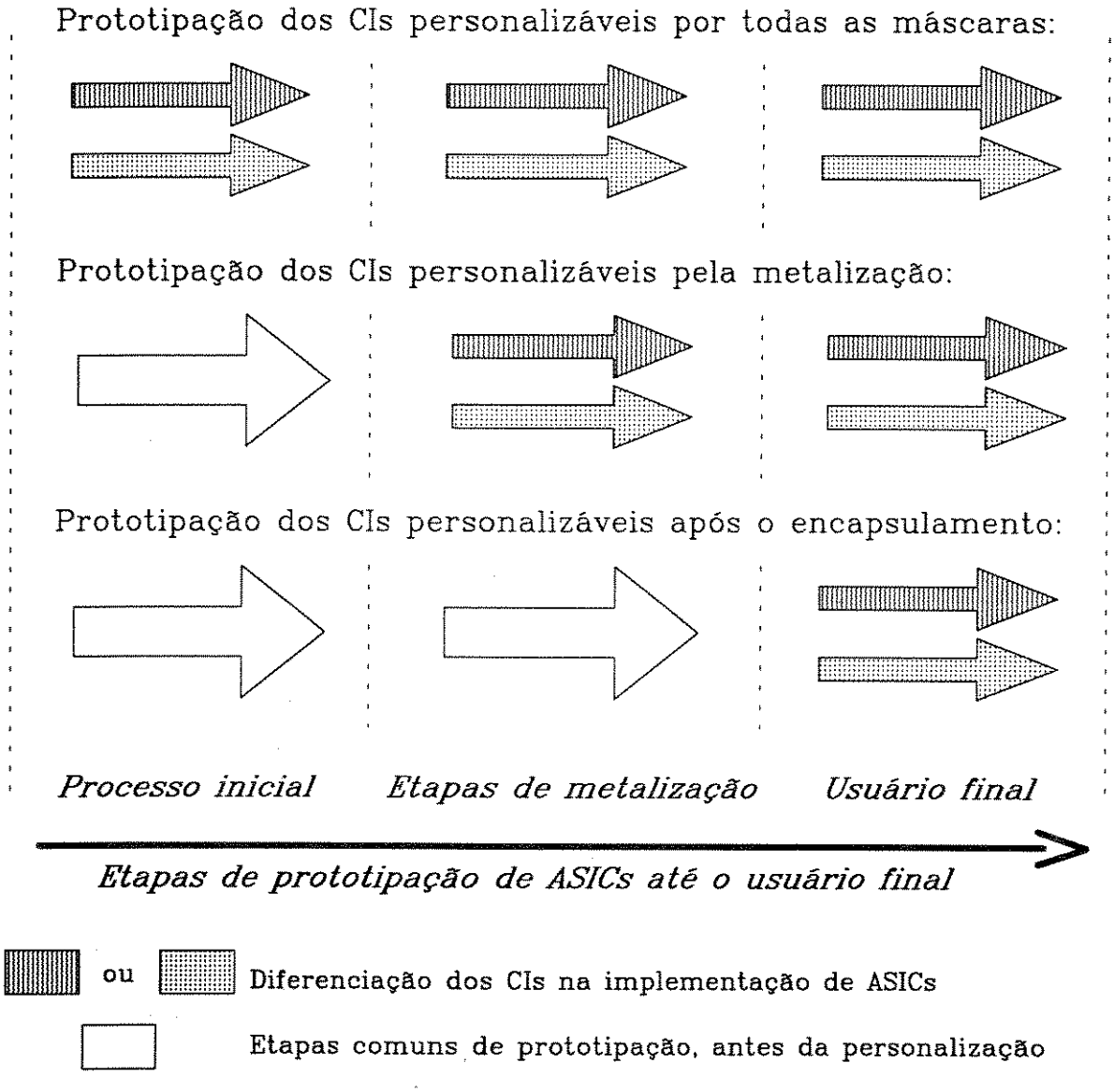


Figura 1.2 - Classificação para o projeto de ASICs segundo [11] e [12].

Porém, o leiaute dos CIs personalizáveis por todas as máscaras não são necessariamente provenientes de uma biblioteca *standard cell* ou elaborados manualmente. Podem ser utilizados geradores automáticos de leiaute (síntese automática) como geradores de módulos aleatórios, geradores automáticos de PLAs, ROMs e RAMs. Uma classificação das abordagens existentes nesta forma de implementação de ASICs pode ser encontrada em [11] ou em [13].

Para os ASICs **personalizáveis pela metalização**, ou pré-difundidos, o circuito já se encontra pré-fabricado e a definição de suas funções se dá a partir das camadas de metalização. Com isso, reduz-se significativamente o tempo de fabricação e os custos associados, pois uma mesma família de pré-difundidos pode ser utilizada para personalizar CIs de diversos clientes. Porém, nesta forma de implementação, o projetista perde em flexibilidade devido ao compromisso com as características elétricas e geométricas inerentes à etapa já realizada. Isto significa que nesta forma de implementação nem sempre é possível atingir o máximo desempenho elétrico permitido pela tecnologia e o melhor aproveitamento da área da pastilha.

Dentre as abordagens que fazem parte deste segundo grupo estão os *gate arrays*, *sea-of-gates*, *compacted arrays*, *sea-of-cells* [12]. Nesta forma de implementação de ASICs é comum também o uso de bibliotecas de células para reduzir o tempo de elaboração do leiaute (neste caso, somente os níveis de metalização).

Já para os ASICs **personalizáveis após o encapsulamento**, ou PLDs (*Programmable Logic Devices*), a implementação é realizada estando o componente já fabricado, em condições de comercialização, bastando apenas a sua programação. Sendo que esta programação, por sua vez, pode ser realizada pelo próprio usuário. Conseqüentemente, o tempo de prototipação é reduzido ao projeto lógico, uma vez que todo o leiaute já se encontra definido. Porém, desta forma há uma dependência muito grande entre as especificações de desempenho do ASIC e as arquiteturas das famílias de PLDs.

Quanto às ferramentas de CAD (*Computer-Aided Design*) necessárias em cada uma das formas de implementação apresentadas acima, as duas primeiras, personalizáveis por todas as máscaras e pela metalização, necessitam de editores e verificadores de leiaute, simuladores lógicos e elétricos, e muitas vezes, posicionadores e roteadores automáticos. Estas ferramentas normalmente exigem computadores de médio e grande porte (*workstations*, por exemplo) para serem utilizadas com eficiência.

No caso dos ASICs personalizáveis após o encapsulamento, não há necessidade de elaboração de leiautes. O projetista descreve a lógica do circuito e gera um arquivo de configuração ou programação do componente. Para isto, basta que ele disponha de editores de texto ou simbólicos, para descrever o circuito, e simuladores lógicos, para avaliar o seu desempenho funcional. Praticamente todas as ferramentas de CAD para uso desta forma de implementação de ASICs funcionam eficientemente em computadores PC (*Personal Computer*).

É muito importante escolher a forma mais eficaz para a implementação do ASIC. Nesta escolha devem ser considerados fatores como complexidade do circuito, volume de produção, desempenho elétrico especificado, custo de projeto e fabricação, tempo para mercado ou *time-to-market* (tempo de projeto + tempo de fabricação + tempo de avaliação e teste dos protótipos + tempo para possíveis reprojetos), riscos de projeto e a disponibilidade das formas de implementação.

1.3 CIs personalizáveis após o encapsulamento

O surgimento dos CIs personalizáveis após o encapsulamento só foi possível devido à evolução tecnológica dos semicondutores. Com a criação dos fusíveis, em tecnologia bipolar, no início dos anos 70, foram lançados os componentes PROMs, para sua utilização como circuitos armazenadores de dados (memórias) ou como circuitos de lógica aleatória. Nos anos 80 vieram os anti-fusíveis e com eles os componentes FPGAs. Nesta época lançou-se também os componentes EPLDs, com tecnologia EPROM e E^2PROM.

Os CIs personalizáveis após o encapsulamento possuem uma história recente, apesar de já existirem os PROMs, PLAs e PALs, desde os anos 70. Desde o final da década de 80, com o lançamento dos FPGAs e EPLDs, tem havido uma evolução muito rápida dos CIs personalizáveis após o encapsulamento. Por este motivo que ainda hoje há tantas dúvidas com relação a taxonomia destes componentes e à própria nomenclatura, utilizada pela literatura e pelas empresas, para descrevê-los. Entre as denominações utilizadas para referenciá-los, de maneira genérica, estão as siglas FPGA (*Field Programmable Gate Array*), e EPLD (*Erasable Programmable Logic Device*). Porém, estas duas nomenclaturas, juntamente com tantas outras, ajudam a diferenciar tecnologias e arquiteturas.

Talvez o termo mais apropriado seja PLD (*Programmable Logic Device*) como referência genérica para os CIs personalizáveis após o encapsulamento, ou componentes de lógica programável pelo usuário, como também são chamados. O problema de nomenclaturas e taxonomia, assim como as arquiteturas e os elementos de programação (fusíveis, anti-fusíveis, SRAM,...) serão discutidos ao longo deste trabalho.

As principais características dos PLDs são a sua rapidez na implementação do ASIC, a inexistência de protótipos de teste e a possibilidade de configuração ou personalização do componente na própria bancada de trabalho pelo projetista. A rapidez de implementação deve-se ao fato de não haver interação com o processo de fabricação do CI. O PLD encontra-se totalmente difundido e já encapsulado, restando ao projetista apenas programar o componente através da queima de fusíveis ou do carregamento de células de memória, definindo a sua lógica.

A utilização de PLDs para a implementação de ASICs evita existência de protótipos de teste, ao contrário das demais formas de implementação citadas. A lógica do circuito é testada antes do componente ser configurado. E além disso,

caso o ASICs não apresente um correto funcionamento, quando colocado no produto final, devido às variáveis externas do sistema, alguns PLDs permitem a reprogramação de sua lógica, evitando o desperdício de componentes.

Outra vantagem do uso dos PLDs é a simplicidade do ambiente de trabalho e o fácil uso do *software* de programação. A maioria dos fabricantes de CIs personalizáveis após o encapsulamento fornecem ferramentas de CAD de baixo custo (entre 5 e 10 mil dólares) e que rodam em computadores PC. Além disso, o projetista (usuário) não precisa ser especialista em microeletrônica para trabalhar com estes CIs.

Porém, o uso de componentes já fabricados reduz a flexibilidade do projetista na alteração das características de *timing* do circuito, pois estas estão diretamente ligadas às arquiteturas e elementos de programação dos PLDs. Portanto, circuitos com exigentes especificações elétricas talvez não possam ser implementados devido às limitações dos componentes.

Além do desempenho elétrico, outro fator dependente da arquitetura pré-definida do PLD é a sua capacidade lógica. Se o componente permite a implementação máxima de 5.000 portas equivalentes, por exemplo, este é o limite, qualquer circuito maior não poderá utilizar este PLD. O mesmo ocorre com os CIs personalizáveis pela metalização, onde o ASIC é implementado sobre matrizes de transistores ou células lógicas pré-fabricadas. Já nos CIs personalizáveis por todas as máscaras esta limitação praticamente não existe.

1.4 Ambientes de prototipagem rápida

Nos últimos anos, a concorrência entre os fabricantes de produtos eletrônicos tem crescido muito rapidamente. São americanos, europeus, japoneses e outras nações asiáticas disputando um mercado muito rico e inesgotável. A evolução tecnológica nestes países é tal que a todo momento o mundo é surpreendido com novos lançamentos na área de computadores, componentes, aparelhos eletrodomésticos e até mesmo brinquedos.

Tudo isso faz com que o sigilo sobre os projetos seja um dos principais fatores para enfrentar a concorrência. Mas apenas o sigilo tecnológico não garante a conquista do mercado, pois todos estão evoluíndo, e quem lançar primeiro seu produto estará atraindo a atenção do consumidor para algo inédito que carrega o nome da sua empresa. Os que vierem depois não terão o mérito da criatividade e serão vistos como "imitadores", mesmo que isso não seja uma verdade.

A redução do *time-to-market* tem um compromisso direto com as especificações do circuito [14]. Quando este apresenta especificações de desempenho elétrico bastante restritas muitas vezes não é possível utilizar qualquer uma das formas de implementação de ASIC apresentadas, pois as com menor *time-to-market*, muitas vezes, apresentam dependências com relação as suas arquiteturas e matrizes pré-fabricadas, que limitam as características elétricas do

circuito. Este compromisso com as especificações do ASIC deve ser muito bem analisado pelo projetista, pois o seu tempo para lançamento no mercado acaba definindo o tempo de vida do produto e a quantidade máxima de unidades a serem vendidas (figura 1.3) [3].

Pelo gráfico da figura 1.3 observamos que o produto de uma empresa, lançado após seu concorrente, não atingirá o volume de vendas do produto adversário, além de ser retirado mais cedo do mercado.

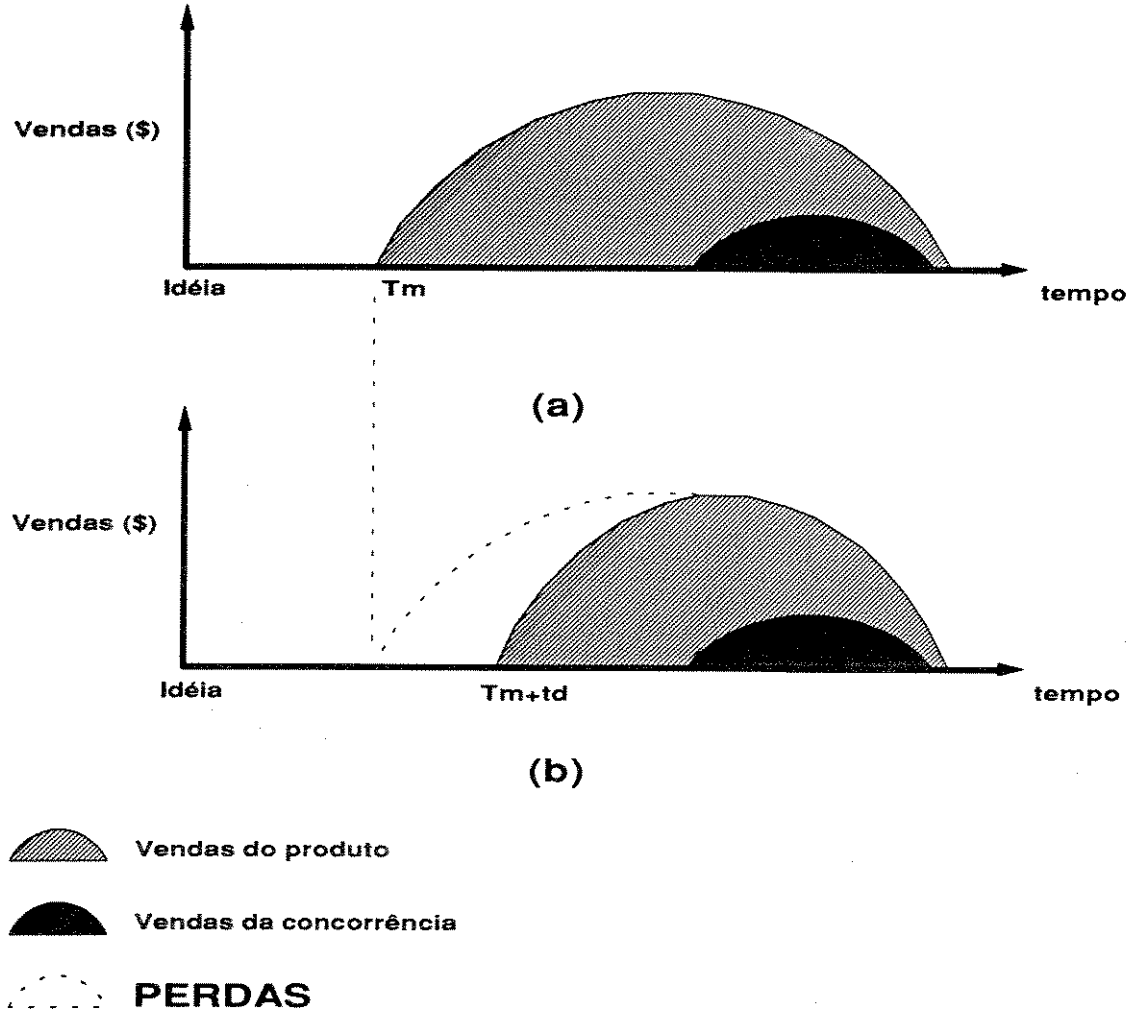


Figura 1.3 - Curva de permanência de um produto no mercado.

Visando reduzir o *time-to-market* dos produtos, os fabricantes têm procurado diminuir o **tempo de projeto e de fabricação dos ASICs**. Para que o **tempo de projeto** seja reduzido, há um investimento no sentido de desenvolver programas de computador ou ferramentas de CAD, como são mais conhecidos os *softwares* que auxiliam na descrição e análise de projetos, e na elaboração dos leiautes necessários para a fabricação dos CIs personalizáveis por máscaras. Estas ferramentas de CAD são editores esquemáticos, editores de leiautes, simuladores lógicos e elétricos, roteadores e posicionadores automáticos, verificadores de leiautes.

Uma outra forma de reduzir o tempo de projeto é utilizar pequenos circuitos previamente desenhados e caracterizados, guardados numa biblioteca de células, para compor circuitos maiores. Toda vez que for realizado um novo circuito pode-se utilizar esta biblioteca de células para otimizar o tempo de geração do leiaute.

E uma terceira forma de redução do tempo para elaboração do leiaute é através do uso de ferramentas de CAD que realizam sínteses automáticas, ou seja, ferramentas que geram automaticamente o leiaute do circuito a partir de sua descrição textual ou simbólica.

O **tempo de fabricação**, por outro lado, pode ser reduzido utilizando-se pastilhas pré-difundidas, contendo matrizes de transistores já fabricados. O circuito é implementado sobre estas matrizes através da conexão dos transistores, utilizando as etapas de metalização do processo de fabricação. Outra maneira possível é implementar o circuito sobre componentes já fabricados que contém elementos de programação (fusíveis ou anti-fusíveis, por exemplo) para definição da sua lógica. Neste segundo caso, o tempo de fabricação praticamente não existe pois ele corresponde apenas ao tempo de configuração do componente programável, normalmente da ordem de milisegundos.

Quando se utiliza ferramentas de CAD, biblioteca de células ou síntese automática de leiaute juntamente com opções rápidas de fabrição de CIs obtém-se o que se costuma chamar de **ambientes de prototipagem rápida**.

O uso destes ambientes de prototipagem rápida, a fim de reduzir o tempo de obtenção de protótipos, já está muito difundido dentro da comunidade de microeletrônica. Estes ambientes podem ser encontrados utilizando as mais diversas formas de implementação de ASICs.

Eles têm por objetivo diminuir ao máximo o tempo para obtenção dos protótipos de teste, e conseqüentemente o tempo para lançamento do produto no mercado. Porém, estes ambientes não interferem no período de desenvolvimento da idéia para definição do produto, até atingir-se um descrição simbólica ou textual do ASIC. O tempo de implementação de uma idéia dentro de um circuito digital pode demorar dias, semanas, ou até meses, e nesta etapa as ferramentas de CAD têm uma participação discreta.

A partir do circuito descrito, os protótipos podem ser obtidos rapidamente (dias ou semanas). Para então serem levados à bancada de testes, onde não é possível mensurar o tempo de trabalho desta fase, havendo inclusive o risco de reprojeto do circuito. Somente após a validação dos protótipos é que o ASIC vai para produção.

Portanto, o uso de um ambiente de prototipagem rápida não significa que uma nova idéia estará sendo lançada no mercado em poucos dias. Isto dependerá muito do tempo gasto na etapa de definição do circuito e da complexidade dos testes realizados sobre os protótipos.

Como já foi dito, as mais diversas formas de implementação de ASICs podem ser encontradas nestes ambientes. A seguir serão descritos suscintamente três ambientes de prototipagem rápida que fazem parte da realidade brasileira, cada um deles baseado em uma das formas de implementação definidas em [11]:

- **SOLO 1400** - uso de componentes personalizáveis por todas as máscaras;
- **GA2500** - uso de componentes personalizáveis pela metalização;
- **MAXPLUS2** - uso de componentes personalizáveis após o encapsulamento.

1.4.1 SOLO 1400

O ambiente de prototipagem rápida **SOLO 1400**, fornecido pela *European Silicon Structures* (ES2) - França, consiste basicamente de um pacote de ferramentas de CAD para otimização do tempo de projeto de ASICs, através de CIs personalizáveis por todas as máscaras.

A descrição dos circuitos pode ser por esquemáticos ou por arquivo textual HDL (*Hardware Description Language*). A ES2 fornece, além da biblioteca de células *standard cell*, geradores automáticos de leiaute para estruturas regulares como PLA, ROM e RAM. Durante a captura esquemática o projetista utiliza os símbolos lógicos da biblioteca ou pode descrever blocos funcionais que representam estas estruturas geradas automaticamente.

A partir da descrição esquemática podem ser realizadas simulações lógicas funcionais e temporais, utilizando as características de *timing* da biblioteca, fornecida pela *foundry*. O posicionamento e roteamento das portas lógicas no leiaute do circuito são realizados automaticamente, segundo as regras de projeto utilizadas, apesar de serem permitidos pré-posicionamentos e pré-roteamentos manuais.

É possível a realização de projetos mistos utilizando-se circuitos de lógica aleatória, gerados com o auxílio de bibliotecas de células, e circuitos como PLAs, RAMs e ROMs, gerados automaticamente. O ambiente SOLO 1400 fornece ainda uma biblioteca de *pads* analógicos, incluindo conversores digital-analógico e analógico-digital, amplificadores operacionais, multiplexadores e comparadores.

Pode-se realizar simulações após a geração do leiaute utilizando-se informações dos elementos parasitas existentes extraídos por um dos *softwares* do SOLO 1400. Concluído e verificado, o leiaute do circuito é enviado diretamente para a ES2, para a fabricação dos protótipos. Este ambiente auxilia também na escolha do melhor encapsulamento (*package*) para o ASIC.

Observa-se que todas as etapas de fabricação devem ser realizadas sobre o silício, segundo as informações do leiaute. Não há nada pré-fabricado como ocorre nos CIs personalizáveis pela metalização. A ES2 demora entre 4 e 8 semanas para entregar os primeiros protótipos do ASIC. Estes, por sua vez, são testados pelo

projetista e, se o seu funcionamento estiver dentro das especificações, o CI poderá entrar em produção. Caso contrário, deve-se reprojetar as partes deficientes.

A aquisição do sistema SOLO 1400 representa um investimento que poderá não apresentar um retorno imediato, além da necessidade da disposição de estações de trabalho SUN, Apollo ou DEC (este ambiente não roda em plataforma PC). No Brasil, caso o cliente não queira adquirir toda esta estrutura de *software* e *hardware*, ele pode utilizar os serviços de empresas de projeto, como a Quickchip Engenharia e Projetos Ltda, para implementar o seu ASIC na ES2, com o uso deste ambiente de projeto.

1.4.2 GA2500

O ambiente de prototipagem rápida que está sendo desenvolvido na Fundação Centro Tecnológico para Informática (CTI) encontra-se em fase final de implantação. Este ambiente consiste na implementação de ASICs sobre uma matriz *gate array* de dois níveis de metal, com a personalização somente do segundo nível (metal2).

Esta matriz *gate array* de propriedade do CTI, chamada de **matriz GA2500**, tem a capacidade lógica de aproximadamente 2.500 portas equivalentes, 112 *pads* configuráveis de I/O (entrada e saída), 8 *pads* de alimentação (4 Vdd e 4 Gnd) e 2 *pads* diretos (somente com diodos de proteção) [15]. O *core* (núcleo de transistores) da matriz contém 20 linhas com 313 pares de transistores CMOS (Wp = 16μm, Wn = 7μm e Lp = Ln = 1,6μm).

A tecnologia utilizada neste ambiente é CMOS 1.5μm, dois níveis de metal, da *foundry* ES2 (França), onde serão fabricadas as matrizes para personalização final no CTI. Esta personalização corresponde apenas ao *etching* do nível mais alto de metalização, já depositado sobre a lâmina. Portanto, na matriz GA2500, além dos transistores (como todo *gate array* tradicional), o primeiro nível de metal (metal1) e as vias (conexão entre os dois níveis de metalização) já encontram-se definidos. No CTI serão retiradas as áreas de metal2 indesejáveis, para a definição das conexões nas células lógicas, nos *pads* de I/O e entre eles. Por fim, é depositada a camada de passivação e abertas as janelas sobre esta (abertura de janela dos *pads*), para a soldagem dos *pads*.

A parte de projeto das máscaras faltantes para a **personalização da matriz (metal2 e depassivação)** é realizada com o auxílio de ferramentas de CAD da Mentor Graphics. Para otimizar ainda mais a etapa de projeto, utiliza-se uma biblioteca de células lógicas e *pads*, previamente projetados e caracterizados [16]. A única forma de descrição dos circuitos é através de esquemáticos. A partir desta descrição, são realizadas simulações lógicas utilizando-se as características de *timing* da biblioteca de células. A interação entre a descrição esquemática e as simulações lógicas ocorrem até que o desempenho funcional e elétrico previsto para o ASIC esteja correto.

O tempo gasto nesta definição do circuito, com o auxílio das simulações lógicas, é reduzido com o uso das ferramentas de CAD. Mas quem realmente determina a duração desta fase são conjuntamente o projetista e o cliente. Todas as variáveis externas ao ASIC, ou seja, as condições de contorno do sistema onde o circuito será colocado devem ser consideradas.

A partir da definição do esquemático final, o circuito é então posicionado e roteado sobre a matriz. Este passo é praticamente automático, assim como a verificação final do leiaute (DRC, ERC e LVS) e a geração do arquivo contendo a descrição das máscaras do metal2 e da passivação, normalmente em linguagem CIF ou GDSII.

Pode-se ainda realizar uma simulação pós-roteamento, considerando as capacitâncias decorrentes das trilhas de metalização do próprio roteamento, a fim de aumentar a confiabilidade sobre o projeto do ASIC.

Todo este ambiente de projeto, suportado pelo *software* da Mentor e instalado em *workstations* Apollo (estações de trabalho), praticamente não é possível ser repassado ao usuário devido à complexidade de dados da biblioteca de células e da matriz. Isto seria quase que um novo desenvolvimento deste ambiente, sem considerar os elevados custos das ferramentas de CAD e das estações de trabalho.

O **ambiente de prototipagem rápida GA2500** é um serviço que o CTI estará oferecendo em breve. O cliente provavelmente poderá adquirir um ambiente de baixo custo (em plataforma PC) onde será possível realizar a descrição do circuito e a sua simulação lógica, utilizando os dados da biblioteca de células. Esta descrição é enviada ao CTI e dentro de duas a três semanas o cliente receberá os primeiros protótipos do seu ASIC.

1.4.3 MaxPlus2

O **ambiente de prototipagem rápida MaxPlus2**, da Altera Corp., baseia-se no uso de PLDs (EPLDs e FPGAs). Estes componentes permitem a implementação de ASICs, já descritos e validados por simulações lógicas, em poucos minutos. Para isso, basta configurar corretamente os elementos de programação (no caso da Altera são dispositivos EPROM, E^2PROM e células SRAM) que os PLDs executam as funções lógicas desejadas.

Porém, para programar estes componentes é necessário a utilização de uma ferramenta de CAD apropriada, fornecida pelo fabricante, onde o circuito é descrito, simulado e compilado dentro do componente programável. Sem este *software* é praticamente impossível configurá-los.

O ambiente MaxPlus2 foi desenvolvido para trabalhar sobre plataforma PC. Ele permite diversas formas de descrição do circuito: captura esquemática, descrição textual HDL, descrição das formas de onda dos sinais de entrada e saída,

tabelas verdades e equações booleanas [17].

A partir da descrição do circuito, o projetista pode realizar simulações, considerando nulos os tempos de propagação de sinais, a fim de verificar o correto funcionamento lógico do circuito (simulações lógicas funcionais). Até este momento não é possível garantir que o ASIC funcionará corretamente devido às características de *timing*, ignoradas nestas simulações lógicas: sincronismo de sinais, atrasos de propagação de sinais através de caminhos críticos, e freqüências limites de funcionamento do circuito.

Após validada a lógica descrita, o circuito é então compilado (mapeado) dentro de um componente programável. O **mapeamento** consiste na adaptação da lógica descrita para as células lógicas existentes dentro dos PLDs. Nesta etapa de projeto, pode-se definir previamente o componente a ser utilizado para a implementação do ASIC ou deixar que a ferramenta faça uma escolha automática, de forma a utilizar o menor componente possível.

A Altera possui três famílias de **EPLDs** (Clássica, Max 5000 e Max 7000) e uma de **FPGA** (Flex 8000) [18]. A diferença entre componentes EPLDs e FPGAs será apresentada mais adiante. Caso o ASIC não caiba em nenhum dos componentes existentes, a ferramenta pode realizar o particionamento automático da lógica em dois ou mais componentes.

Quando o circuito é compilado dentro de um PLD o *software* gera um arquivo de descrição (*report*) onde encontram-se os números de utilização do componente, sua pinagem e as equações booleanas implementadas. É gerado também um arquivo utilizado pelo *hardware* de programação, acoplado ao computador, para configuração da lógica do circuito no PLD. Por fim, um terceiro arquivo gerado na compilação é aquele que contém as características de *timing* do componente escolhido para a implementação do ASIC.

Com este arquivo o projetista pode então realizar simulações temporais para verificar se o componente, a ser configurado com a lógica descrita, apresentará problemas de caminhos críticos de propagação de sinais ou para conhecer os limites de frequência de funcionamento do ASIC implementado. Caso venham a surgir problemas de *timing* o componente deverá ser trocado por outro mais rápido ou a lógica do circuito deverá ser alterada, pois não há como mexer na arquitetura e na tecnologia de um PLD.

Realizada a simulação de *timing*, o componente PLD pode então ser configurado com a lógica do circuito, utilizando-se o equipamento de programação (mais conhecido como "*hardware* de programação"). Após o processo de configuração, que não demora mais do que alguns milisegundos, ele pode ser testado, através do próprio *hardware* de programação, fazendo-se uso dos vetores utilizados nas simulações funcionais e temporais.

Apesar do ambiente de prototipagem da Altera ser totalmente independente, ou seja, não requerer ferramentas de CAD de outros fabricantes para realizar a

configuração de seus componentes, ele permite a interface de descrições esquemáticas com outros editores como o Orcad, Neted (Mentor Graphics) e View Logic.

O pacote completo, com todos os módulos do *software* com o *hardware* de programação, não custam mais do que 10 mil dólares e não é imprecindível que o usuário (projetista) tenha conhecimentos de microeletrônica para trabalhar com este ambiente de prototipagem rápida.

Outros ambientes de prototipagem baseados em PLDs, como da Xilinx Inc. e da Actel Corp., se assemelham muito ao da empresa Altera, com pequenas diferenças devido à própria arquitetura e tecnologia de cada fabricante. Detalhes das arquiteturas e tecnologias (elementos de programação) utilizadas pelas principais empresas na área de PLDs serão apresentadas no capítulo 3.

Capítulo 2

Proposta de um ambiente de prototipagem rápida genérico

Conforme apresentado anteriormente, as formas de implementação de ASICs são classificadas em três categorias, segundo a etapa de processo de fabricação em que o circuito é personalizado:

a) **CIs personalizáveis por todas as máscaras**;
b) **CIs personalizáveis pela metalização**, ou **pré-difundidos**;
c) **CIs personalizáveis após o encapsulamento**, ou **PLDs**.

Os CIs **personalizáveis por todas as máscaras** são diferenciados desde as primeiras etapas de fabricação. A fim de simplificar a análise comparativa entre as formas de implementação, será adotada a terminologia usualmente citada na literatura, que divide os CIs personalizáveis por todas as máscaras em dois grandes grupos: ***full custom*** e ***standard cell***. Porém, esta generalização desconsidera os CIs cujos leiautes foram realizados através de geradores automáticos de PLAs, RAMs e ROMs, e sintetizadores de lógica aleatória [13].

Portanto, dentro deste contexto, a definição para *full custom* engloba os CIs que apresentem mais de 50% do leiaute gerado de maneira manual [3]. Por outro lado, neste trabalho a definição para *standard cell* será expandida para os circuitos que possuam mais de 50% do leiaute gerado a partir de primitivas geométricas (bibliotecas de células) e/ou gerado automaticamente por ferramentas específicas. Isto significa que os circuitos que utilizem células padrão e/ou outras formas de implementação automáticas, tais como geração *gate-matrix* e de lógica regular [11], passam a fazer parte de tal classificação.

As formas **personalizáveis pela metalização**, ou pré-difundidos, que apresentam as abordagens *gate array, sea-of-gates, compacted array* e *structured array* [12], são tratados na literatura pelo termo *gate array*, sem distinção, quando se trata de comparação com as demais formas de implementação.

A principal característica dos pré-difundidos é a utilização de matrizes de transistores pré-fabricados e pré-caracterizados, sobre as quais o projetista define as trilhas de roteamento, por meio das máscaras de metalização, de forma a realizar as conexões necessárias à implementação de determinado circuito. Com isso, as características elétricas, geométricas e posicionais dos transistores não podem ser alteradas. Esta inflexibilidade resulta num fator de

aproveitamento de área da pastilha geralmente bem abaixo daquele apresentado pelos CIs personalizáveis por todas as máscaras.

Nos pré-difundidos, como ocorre com os *standard cells*, utilizam-se bibliotecas com células pré-projetadas, a fim de otimizar o tempo de projeto. Portanto, a perda de flexibilidade para realização de projetos nesta forma de implementação é maior do que nos *standard cells* pois, além da restrição de se projetar com base numa biblioteca de células, há também a limitação dos parâmentros elétricos dos dispositivos devido às características físicas da matriz utilizada para a implementação do circuito.

A categoria dos **CIs personalizáveis após o encapsulamento** descreve os componentes personalizados pelo próprio usuário, após o processo de fabricação e encapsulamento. Até chegar a este usuário, os CIs não apresentam função lógica definida, mas uma matriz de células lógicas e trilhas de roteamento fixas, contendo elementos de programação (PROM, EPROM, E^2PROM, anti-fusíveis e SRAM) para sua configuração.

Tal configuração é realizada pelo usuário apenas no momento de implementação do circuito, havendo a possibilidade de reprogramação de alguns destes componentes, dependendo do dispositivo de programação utilizado. Tanto a programação quanto a reprogramação dos componentes é realizada em alguns segundos, permitindo a rápida obtenção do circuito implementado. Porém, a perda de flexibilidade nesta categoria é ainda maior do que nos pré-difundidos, devido à dependência da arquitetura do dispositivo e do *software* de programação, específico para cada fornecedor.

2.1 Escolha da forma de implementação

Tendo conhecimento das formas de implementação de ASICs, uma das tarefas mais difíceis para o projetista é escolher qual delas utilizar. As formas de implementação apresentam características bem distintas, com inúmeras vantagens e desvantagens a serem consideradas. A escolha é tão importante quanto o projeto em si, pois ela definirá as características de desempenho e custo, como também o tempo necessário para lançamento de um produto no mercado [19].

Inúmeras são as variáveis que interferem na escolha da forma a ser usada, como custo por CI fabricado, tempo de lançamento do produto no mercado, tamanho da pastilha, número de pinos (sinais de entrada e saída), tempo necessário para projeto e fabricação, disponibilidade e eficiência do suporte computacional existente, riscos de projeto, entre outros. A análise das variáveis consideradas e, conseqüentemente, a escolha da forma, não seguem regras rígidas e bem definidas. Isso faz com que tal escolha, na maioria das vezes, torne-se subjetiva e dependente da experiência (*feeling*) do projetista.

Na análise a seguir, destacam-se quatro variáveis principais: volume de produção do componente a ser fabricado, complexidade do circuito implementado, desempenho e o tempo necessário para projeto e fabricação do circuito até o seu lançamento no mercado (*time-to-market*). Estas variáveis são o ponto de partida e, muitas vezes, suficientes para a escolha da melhor forma de implementação do ASIC.

2.1.1 Volume de produção

O volume de produção ou número de unidades produzidas interfere diretamente na escolha da forma apropriada para a implementação do ASIC por estar relacionado com o custo de fabricação. A variação do número de CIs fabricados em cada lote de produção altera seu custo unitário, fazendo com que a conveniência do uso de determinada forma de implementação dependa do volume produzido.

O custo total de projeto e fabricação do CI, sem considerar o sistema onde este será utilizado (placa de circuito impresso e interfaces necessárias), pode ser dividido em duas parcelas, quais sejam, custo fixo e custo variável, multiplicado pelo volume de produção, onde:

custo total = custo fixo + (custo variável x volume)

O **custo fixo** considera todos os custos que independem do volume produzido, envolvendo basicamente o **custo NRE** (*non-recuring engineering*) e **custos adicionais** de engenharia [3] [19] [20].

NRE representa os custos associados ao desenvolvimento do ASIC, que são repassados pelo fornecedor de tecnologia e agregados pelo trabalho do projetista. Está relacionado às atividades de projeto propriamente ditas, tais como a análise de viabilidade técnico-econômica, especificação e descrição do circuito, geração do leiaute, simulações lógicas e elétricas pré e pós leiaute, verificação do leiaute (DRC, ERC e LVS), criação de vetores de teste, preparo e envio dos dados de projeto ao fornecedor de tecnologia, acompanhamento da validação e aprovação dos protótipos.

O **custo NRE** engloba genericamente:

- custo de projeto;
- custo de utilização das ferramentas de projeto (*software* e *hardware*);
- recursos humanos necessários;
- custo de fabricação de protótipos e
- custo de suporte do fornecedor de tecnologia.

Os **custos adicionais**, eventualmente necessários, referem-se ao tempo adicional de projeto e criação dos vetores de teste visando testabilidade (se necessário), custo adicional das ferramentas de suporte computacional necessárias (*hardware* e *software*), tempo de utilização de equipamentos eventuais e recursos humanos adicionais para todas estas etapas, além do custo de um eventual reprojeto.

Em certos casos, existem também custo adicional de simulação (para aferição final do projeto antes da fabricação) e o custo de engenharia para elaboração do programa de teste, ambos realizados pelo fornecedor, com ferramentas de CAD, equipamentos e recursos humanos do próprio fornecedor. A tabela 2.1 apresenta valores típicos para o custo fixo segundo a forma de implementação utilizada [3].

Formas de implementação	Custo fixo (US$)
full custom	100.000
standard cell	60.000
pré-difundido	30.000
PLD	1.500

Tabela 2.1 - Comparação de custos fixos típicos para as formas de implementação de ASICs, supondo CIs com complexidade de 2.000 portas equivalentes.

Os CIs *full custom* apresentam custo fixo mais elevado que os demais, principalmente devido à geração manual do leiaute (tempo de projeto), ao maior risco de projeto (custo de reprojeto) e à necessidade de verificação de toda a lógica implementada, uma vez que não são utilizadas bibliotecas de células e geradores automáticos de leiaute.

Os CIs pré-difundidos e os *standard cells* apresentam grande semelhança no que se refere ao custo de projeto e ao custo de utilização de ferramentas de CAD e de recursos humanos. Eles diferem quanto ao tempo e recursos humanos gastos para a fabricação, pois nos pré-difundidos, são realizadas apenas as etapas de metalização, enquanto que nos *standard cells,* são realizadas todas as etapas de processo, do mesmo modo que nos *full custom.*

O custo fixo relacionado com a fabricação é menor para os pré-difundidos do que para os *full custom* e *standard cells*, devido ao melhor aproveitamento de equipamentos e recursos humanos referentes ao processo inicial. Isto ocorre porque as matrizes pré-difundidas são produzidas em grandes quantidades, reduzindo seu custo unitário, enquanto que para as formas personalizáveis por

todas as máscaras, dificilmente se explora a máxima capacidade dos equipamentos de processo.

Tomando-se a análise acima, observa-se que o custo fixo dos CIs *standard cells* é menor que dos CIs *full custom* devido ao custo de projeto, porém é maior em relação aos pré-difundidos, por causa da parcela do custo de fabricação de protótipos.

Os PLDs, por sua vez, apresentam custo fixo muito baixo em relação às demais metodologias, pois não há necessidade de geração de leiaute nem fabricação de protótipos específicos para teste. O circuito é descrito, simulado e validado funcionalmente, compilado para configurar o componente, e testado no próprio ambiente de programação. Devido, ainda, à possibilidade de reprogramação de algumas famílias, seu custo de reprojeto é bastante baixo. Além disso, o custo de utilização das ferramentas de *software* para personalização do CI é menor do que das ferramentas necessárias para projetos nas demais formas apresentadas anteriormente.

A taxa de variação do custo total em relação ao volume de produção é dada pelo **custo variável**. Este custo é calculado pela equação:

custos variáveis = (preço por *gate*) x (total de *gate*),

entendendo-se por *gate*, *gates* equivalentes ou portas equivalentes, o número de transistores necessários para a implementação de uma porta lógica nand de 2 entradas.

O preço por *gate* depende da tecnologia e do processo utilizado, como também da densidade de integração da forma de implementação adotada.

Dentre as formas de implementação de ASICs citadas, os *full custom* apresentam maior **densidade de integração**, expressa em *gates* por mm^2, resultando num menor preço por *gate*, seguidos pelos *standard cells*, pré-difundidos e PLDs, respectivamente. A tabela 2.2 resume valores de preço por *gate* para as formas de implementação consideradas. Costuma-se caracterizar os pré-difundidos e PLDs não pela densisdade de integração, mas sim pela **capacidade** em termos do número máximo de *gates* disponíveis no componente.

A densisdade de integração dos pré-difundidos e PLDs é menor do que para os *full custom* e *standard cell*, devido ao fato de terem arquitetura fixa quanto à disposição das células internas, canais de roteamento, número e posição dos *pads*, etc. Num *gate array*, por exemplo, existe um número fixo de *pads* já presentes na estrutura, quer se utilize todos ou não. Nos PLDs encontram-se, ainda, estruturas ou dispositivos de programação não

considerados na medida da capacidade do componente, mas que são necessários para permitir a configuração do circuito.

Formas de implementação	Preço por *gate* (US$)
full custom	0.0005
standard cell	0.0012
pré-difundido	0.0020
PLD	0.0035

Tabela 2.2 - Preço por *gate* segundo a forma de implementação.

A porcentagem máxima da capacidade utilizável em um CI pré-difundido é dada pelo **fator de utilização**, geralmente entre 80 a 95% nos *gate arrays* de canal, chegando a menos de 50% nos *sea-of-gates* roteados por ferramentas de *routing* para canal.

Quanto ao número de total de *gates* implementáveis nos PLDs, sua capacidade lógica deve ser sempre maior do que a necessária para a implementação do ASIC, devido ao mapeamento de funções lógicas nem sempre otimizado. O **fator de utilização** destes componentes situa-se normalmente em torno de 50 a 80%. No caso das formas personalizáveis por todas as máscaras, constroe-se sobre a pastilha apenas os transistores efetivamente necessários, tendo, portanto, 100% de utilização.

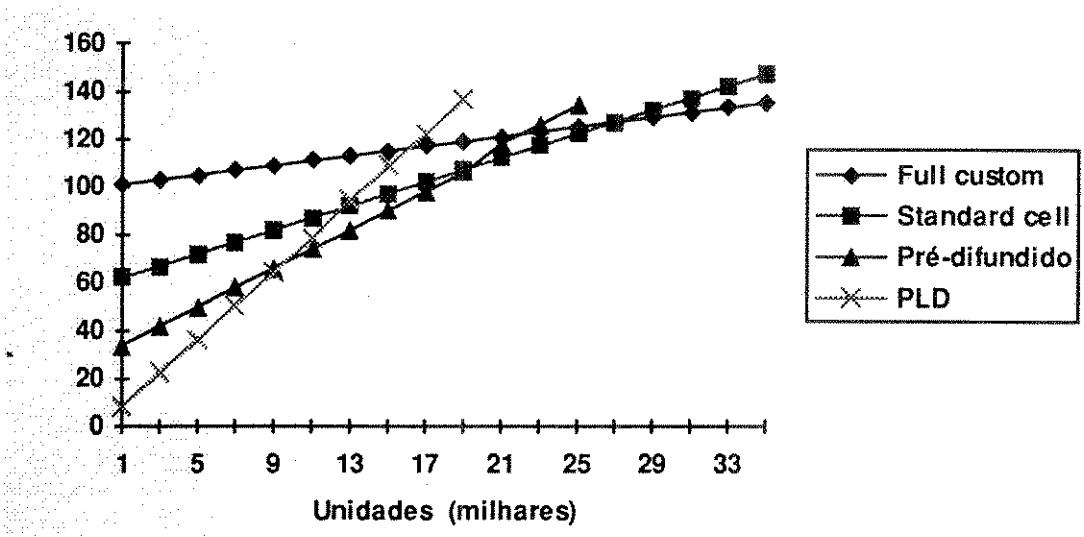


Figura 2.1 - Curva típica de custo total de desenvolvimento em função do volume de produção, considerando 2.000 portas equivalentes.

A figura 2.1 ilustra como varia o custo total em função do volume de produção, numa situação típica de um ASIC de 2.000 portas equivalentes, segundo as quatro formas de implementação consideradas. Nota-se que os PLDs são econômicamente viáveis para a faixa de 1 a 9.000 peças. Os pré-difundidos (*gate arrays*) constituem vantagem de 9.000 a cerca de 20.000 peças, cedendo posteriormente lugar aos *standard cells*, de 20.000 a 30.000 peças. A partir de 30.000 peças, torna-se viável desenvolver um CI *full custom*.

2.1.2 Complexidade

A medida normalmente utilizada para avaliação da complexidade de circuitos digitais é o número de ***gates*** ou **portas equivalentes**. Relembrando que uma porta equivalente corresponde a implementação de uma função lógica nand de duas entradas, ou seja, quatro transistores. Esta medida permite uma boa estimativa do número de transistores necessário para a implementação do circuito e, conseqüentemente, da área utilizada sobre a pastilha.

A análise desta variável está diretamente ligada ao tamanho final da pastilha para as metodologias *standard cell* e *full custom*, e à viabilidade de utilização de determinada matriz de pré-difundidos ou componente PLD, nos quais a área e a capacidade já estão pré-definidas.

A área de silício ocupada influi no custo variável. Portanto, a complexidade do circuito, nos estilos personalizáveis por todas as máscaras (*full custom* e *standard cell*), tem relação direta com a área final do CI e, conseqüentemente, com o preço pago pelo silício utilizado. A tabela 2.3 relaciona os parâmetros densidade, capacidade e área de silício assumindo projetos típicos de 2.000 portas equivalentes e 48 pinos.

Formas de implementação	**Densidade (portas/mm²)**	**Capacidade (portas)**	**Área de silício (mm²)**
full custom	300	-	8
standard cell	240	-	12
pré-difundido	-	2500	28
PLD	-	3000	-

Tabela 2.3 - Comparação de complexidade: ASICs de 2.000 portas equivalentes.

O conceito de um projeto *pad-limited* significa que a área do CI é determinada pelo anel de *pads* (número de pinos exigidos pelo circuito), e não pela área utilizada para implementação das funções lógicas. Isto ocorre com

circuitos de baixa complexidade mas com grande número de sinais de entrada e saída. Por outro lado, *core-limited* conceitua a situação em que se tem um CI de alta complexidade de funções lógicas envolvendo poucos *pads*, sendo portanto a área determinada pela densidade da metodologia empregada.

Para os pré-difundidos e PLDs, a viabilidade de implementação de um ASIC está ligada à capacidade da matriz ou do componente programável, como também ao número de pinos (*pads*) disponível. Os PLDs da primeira geração (PLAs, PALs e PROMs) possuem uma capacidade de implementação em torno de 1.000 portas equivalentes. Já os mais recentes chegam a atingir 20.000 portas equivalentes, porém com um fator de utilização da ordem de 60%.

Há matrizes de pré-difundidos cobrindo a faixa de 1.000 a 30.000 *gates*, existindo atualmente alguns fornecedores anunciando produtos para até 200K portas equivalentes [21]. Existe um compromisso entre área do CI, custo unitário em função da forma de implementação utilizada e tempo de projeto. Por exemplo, consegue-se pastilhas menores com os *full custom*, porém com maior tempo de projeto.

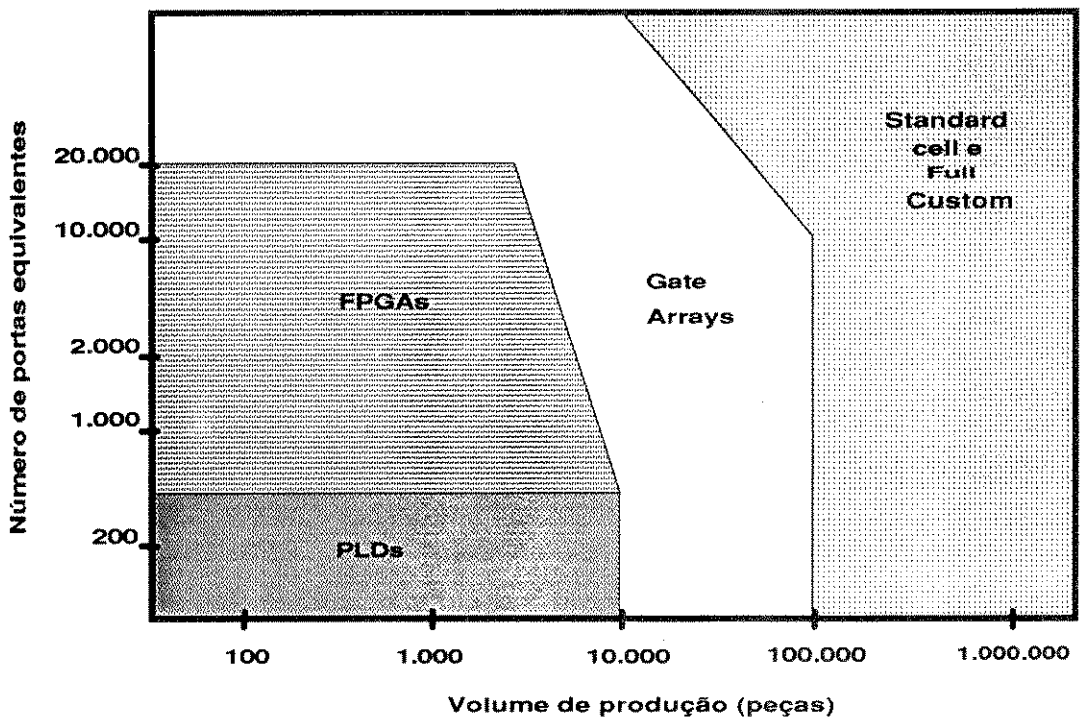


Figura 2.2 - Adequação das formas de implementação segundo a complexidade do ASIC e seu volume de produção [3] [20].

A figura 2.2 ilustra uma proposição de repartição de mercado das formas de implementação segundo a complexidade do ASIC projetado [3] [20]. Nota-se que com o advento dos PLDs, a faixa de mercado de poucas peças a 10.000

unidades encontra melhor solução de implementação com os componentes PALs, EPLDs e FPGAs. *Gate arrays* só se viabilizam acima de 2.000 portas, para volumes acima de 10K, devido a disponibilidade de FPGAs competitivos.

2.1.3 Desempenho

A diferença de desempenho, ou seja, as características de *timing* e consumo de potência entre as formas de implementação apresentadas é bastante acentuada. A especificação da velocidade de funcionamento de um circuito digital pode ser o fator preponderante na escolha da forma de implementação do ASIC.

No *full custom,* o projetista tem a liberdade de desenhar (dimensionar) cada transistor a ser usado no circuito, a fim do otimizar seu desempenho. Porém, isto acarreta em maior tempo para geração do leiaute e simulações lógicas ou elétricas. A limitação em desempenho para esta forma de implementação é determinada pela própria tecnologia utilizada para fabricação do ASIC.

Nos *standard cell,* estas características de funcionamento estão diretamente relacionadas com a biblioteca de células disponível. Chega-se muito próximo do desempenho dos *full custom*, porém com menor tempo de projeto. Para os pré-difundidos a dependência em relação a biblioteca é semelhante, porém, a biblioteca de células desta categoria não consegue ser tão variável em desempenho quanto a dos CIs *standard cell,* devido ao tamanho fixo dos transistores.

Os PLDs, por sua vez, são os componentes mais lentos entre as formas de implementação abordadas. Além da dependência da arquitetura fixa (transistores, *pads*, trilhas de roteamento), a existência de elementos de programação, necessários para a configuração do componente, implica no aumento dos tempos de propagação de sinais dentro do circuito.

Observa-se, pela análise acima, que o ganho de desempenho corresponde a uma maior flexibilidade de projeto, mas também a um maior tempo para obtenção do circuito integrado.

2.1.4 Tempo para o mercado (*Time-to-market*)

O **tempo para o mercado** consiste no tempo decorrido desde a decisão de se iniciar o desenvolvimento do ASIC, concretizada quando o cliente contata um agente conceptor (projetista), até a aplicação do componente final, já em

fase de produção, apto portanto a integrar um sistema ou produto pronto para comercialização [14].

Esta variável corresponde a junção dos tempos de análise de viabilidade, projeto, fabricação, avaliação de protótipos, aprovação do projeto, incluindo ainda o tempo para entrega do primeiro lote de produção do ASIC e as ações de fabricação do produto que o contenha.

Esta é a principal variável considerada no lançamento de um novo produto no mercado. A figura 1.3 demonstra como a demora no lançamento de um produto reduz o faturamento. Um certo produto, previsto para ser lançado no mercado no tempo **Tm** (figura 1.3a), produzirá o faturamento dado pela área hachurada. Havendo um atraso **td**, mesmo que o produto atinja o mesmo volume de vendas previsto, não produzirá o mesmo faturamento. A área tracejada da figura 1.3b corresponde às perdas pelo atraso.

Portanto, o *time-to-market* está relacionado à expectativa de lucro com a comercialização do produto, sendo assim um objetivo importante a ser perseguido.

Atualmente, com a rapidez com que evolui o mercado de bens eletrônicos, projetos demorados podem tornar-se obsoletos, do ponto de vista de vendas, antes mesmo de serem postos em produção. Porém, na escolha da forma de implementação do ASIC deve-se levar em conta, quando for avaliado o tempo para lançamento do produto, os riscos de projeto relacionados a cada uma das formas.

O conceito de **risco de projeto** está relacionado à probabilidade de sucesso do ASIC, este funcionando perfeitamente na sua primeira prototipação. **Baixo risco** significa probabilidades de sucesso próximas a 100%. Por outro lado, **alto risco** representa casos em que se tenha probabilidades de sucesso abaixo de 80%. Tais índices são característicos de cada fornecedor e dependentes da capacitação do projetista e da disponibilidade de ferramentas de CAD confiáveis.

A tabela 2.4 resume alguns índices típicos para o risco de projeto. Não se considera probabilidades inferiores a 80%, uma vez que tais valores não seriam típicos de tecnologias maduras para aplicações comerciais. Tecnologias em desenvolvimento, por sua vez, implicariam em maior risco, devido à instabilidade do processo produtivo.

O risco de projeto, as dificuldades para avaliação dos protótipos, o custo de um eventual reprojeto e o atraso dele resultante devem ser considerados durante a análise da viabilidade do CI, para se determinar o respectivo impacto sobre o tempo para o mercado. Um outro fator importante a ser considerado nesta análise é a disponibilidade das ferramentas de auxílio ao projeto (CAD), o

seu grau de automatização, sua rapidez, eficiência e confiabilidade na obtenção dos resultados.

Formas de implementação	Probabilidade de sucesso na primeira prototipação do ASIC (%)
PLD	100
pré-difundido	95
standard Cell	95
full custom	95 a 80

Tabela 2.4 - Comparação do risco de projeto para as diversas formas de implementação de ASICs.

Muitas vezes, perde-se em desempenho para reduzir ao máximo o tempo de lançamento do produto e conquistar uma determinada faixa do mercado consumidor. À medida que este mercado é conquistado, pode-se realizar uma conversão ou reprojeto sobre uma forma de implementação com melhores características de desempenho e custo, em contrapartida ao possível aumento no volume de produção.

CIs *full custom*, por apresentarem a etapa de projeto quase totalmente manual em relação aos demais e não haver nenhuma etapa de fabricação pré-processada, possuem o maior tempo para mercado e o maior risco de projeto dentre as metodologias apresentadas. CIs *standard cell*, por sua vez, possuem o mesmo tempo de fabricação que os *full custom*, porém seu tempo de projeto é reduzido pelo auxílio de bibliotecas de células e geração automática de leiaute, além da redução do risco de projeto pelo mesmo motivo.

Os pré-difundidos apresentam um tempo de projeto semelhante aos *standard cells*, diferenciando-se no tempo de fabricação devido ao pré-processamento da matriz, restando apenas as etapas de metalização para configuração do ASIC. Logo, o tempo para mercado para esta forma de implementação é menor do que para os CIs personalizáveiss por todas as máscaras, além do risco de projeto ser o mesmo que no *standard cells*.

Por fim, os componentes personalizáveis após o encapsulamento, PLDs, conquistaram seu espaço no mercado de ASICs devido exclusivamente as suas características de *time-to-market*. Estes componentes podem ser obtidos (personalizados), a partir da descrição de um circuito simulado e validado, em menos de uma hora, enquanto que as formas de implementação que dependam do processo de fabricação para personalização do CI levam de 8 a 12 semanas para estarem disponíveis. Além disso, o circuito é testado na própria bancada de projeto, havendo possibilidade de reprogramação em algumas famílias de

componentes. Desta forma, praticamente não há riscos de projeto para os CIs personalizáveis após o encapsulamento.

2.1.5 Comparação das formas de implementação de ASICs

Na tabela 2.5 é mostrado um quadro comparativo simplificado onde são apresentadas as quatro variáveis principais para a escolha da forma de implementação do ASIC. Conforme dito anteriormente, a análise das variáveis não segue regras rígidas e bem definidas, tornando esta uma tarefa bastante subjetiva. Além do mais, as variáveis estão relacionadas entre si não podendo ser consideradas isoladamente.

Formas de implementação	Volume de produção	Complexidade	Desempenho	Tempo para o mercado
full custom	alto	alta	alto	alto
standard cell	médio-alto	alta-média	médio-alto	médio
pré-difundido	médio	alta-média	médio	médio
PLD	baixo ou muito baixo	média-baixa	baixo-médio	baixo ou nulo

Tabela 2.5 - Quadro comparativo das formas de implementação de ASICs.

2.2 Conversão de formas de implementação

A eficiência de emprego das formas de implementação depende muito das condições de mercado. Estas, por sua vez, estão em constante mudança, de modo que uma determinada metodologia pode ser a mais eficiente em um dado instante e tempos depois deixar de ser. Um exemplo clássico disso é a variação do volume de compras pelo mercado consumidor em um país como o Brasil, onde a economia não é estável, alterando constantemente o volume de produção comprometido entre empresas e seus fornecedores, no seguimento de componentes semicondutores.

Muitas vezes, torna-se viável e importante a troca de metodologia na implementação de um ASIC principalmente para garantir o rápido lançamento do produto e um preço competitivo para o mesmo. Porém, esta troca não ocorre automaticamente, com o auxílio de ferramentas computacionais. É necessário um reprojeto, com novas simulações lógicas e/ou elétricas, geração de leiautes, obtenção de protótipos e teste dos mesmos. Este reprojeto deve considerar toda a análise de variáveis realizada originalmente na escolha da forma apropriada de implementação inicial do circuito.

Tem-se estudado, nos últimos anos, a possibilidade de conversão automática de PLDs para componentes personalizáveis por máscaras, sem a necessidade de reprojeto [22] [23] [24]. Para os PLDs da primeira geração (PALs, PLAs e PROMs) esta conversão é possível pela simples troca dos elementos de programação pela metalização dos pontos de configuração, como, por exemplo, PAL (*Programmable Array Logic*) para HAL (*Hardwired Array Logic*) [25]. Como estes componentes não apresentam caminhos críticos de interconexão, devido à simples implementação de soma-de-produtos nos *arrays* AND e OR (a ser visto no capítulo 3), não há necessidade de ressimulação do circuito. Este apenas terá capacidade de operar a uma velocidade maior.

No caso dos PLDs mais recentes, como FPGAs e ELPDs, a simples mudança dos dispositivos de programação por conexões de metal, através de máscaras de processo, altera os tempos de propagação de sinais entre as células lógicas configuráveis. Assim, o circuito precisa ser novamente simulado e testado, realizando-se praticamente um reprojeto do CI.

A empresa Altera Corp., por fornecer EPLDs com *array* de interconexão bastante regular, tenta reduzir o problema de mudança das características de *timing* dos sinais internos adicionando capacitâncias parasitas, de modo a manter os tempos de propagação [26]. Para os FPGAs, onde o roteamento é bastante irregular e definido no momento da configuração do ASIC, esta solução não é possível.

Alguns especialistas afirmam que, mesmo com a possibilidade de conversão de PLDs para pré-difundidos, ela se viabiliza apenas em 5% dos projetos [23]. Existem três casos em que poderia ocorrer uma passagem de um componente programável pelo usuário para um pré-difundido:

a) converter um CI PLD em um CI pré-difundido;
b) colocar vários CIs PLDs em um único CI pré-difundido;
c) realizar o projeto do CI personalizável pela metalização, em paralelo com o CI PLD, lançando-o mais tarde em substituição a este PLD.

As duas primeiras opções, apesar de viáveis, são ineficientes. Na passagem de um projeto, inicialmente validado com componentes de lógica programável pelo usuário (PLD), deve ocorrer compatibilidade de pinagem de encapsulamento e semelhança nas características de *timing* dos sinais, correndo o risco de circuitos com lógica síncrona necessitarem novas simulações, o que envolve especialistas e tempo de projeto. Tudo isso se reflete em custos adicionais.

Outra questão a ser considerada é que, ao invés de se realizar uma conversão direta, após a validação e sucesso de um sistema com PLDs, realize-

se o mesmo investimento em um novo projeto do circuito, com características e funções adicionais.

A realização de projetos em paralelo faz sentido por diversas razões. Uma delas é que certas condições funcionais são difíceis de serem simuladas, como, por exemplo, a parte mecânica de determinados sistemas. Neste caso, a simulação pode ser realizada no próprio ambiente de funcionamento definitivo com o uso de PLDs, enquanto que o componente pré-difundido, mais rápido e viável economicamente para escala industrial, está sendo projetado e fabricado.

Outra razão para realizar os projetos em paralelo é a redução do tempo de lançamento do produto (*time-to-market*), analisado anteriormente: lança-se o produto com componentes PLDs, ocupando-se inicialmente o mercado, enquanto conclui-se o projeto do CI personalizado pela metalização, que trará redução de custo e/ou melhora de desempenho.

A idéia de não se realizar conversões diretas, por incompatibilidade ou por ser mais atraente o reprojeto, como também a possibilidade de se realizar dois desenvolvimentos em paralelo, podem ser estendida aos demais estilos de implementação de ASICs, *full custom* e *standard cell*.

A conversão entre as outras formas de implementação, embora possível, por vezes não se viabiliza pelos mesmo motivos discutidos para o caso de PLDs. A conversão de pré-difundidos para *standard cells*, disponível em alguns fornecedores, também não é automática, implicando sempre em custos adicionais de ressimulação lógica, leiaute, testes. Qualquer conversão para *full custom*, devido a sua própria natureza, caracteriza-se por ser praticamente um projeto novo.

2.3 Ambiente proposto

Os ambientes de prototipagem rápida apresentados no primeiro capítulo possuem uma característica semelhante: eles praticamente trabalham com uma forma apenas de implementação de ASICs:

- **SOLO 1400 - CIs personalizáveis por todas as máscaras**
- **GA2500 - CIs personalizáveis pela metalização**
- **MaxPlus2 - CIs personalizáveis após o encapsulamento**

Porém alguns destes ambientes fornecem a possibilidade de conversão para outras formas de implementação conforme discutido acima. Mas esta espécie de conversão nem sempre é muito clara e simples para o projetista.

Um dos objetivos principais deste trabalho é apresentar um ambiente de prototipagem rápida genérico que integre as três formas de implementação de

ASICs citadas, permitindo ao usuário a escolha de uma destas formas depois que o circuito já apresenta sua descrição esquemática concluída. Trabalho semelhante é apresentado em [83].

Conforme foi visto ao longo deste capítulo, esta opção de escolha é muito importante pois cada uma das formas de implementação apresentam características econômicas e de desempenho elétrico bastantes distintas, fazendo com que cada uma delas tenha o seu nicho de mercado onde se apresentam como a opção mais eficiente.

O ambiente de prototipagem rápida proposto tem como característica principal permitir a elaboração e descrição do circuito de maneira genérica, independente de tecnologia ou forma de implementação. Isto ocorre quando utiliza-se a linguagem HDL para desenvolvimento de circuitos, onde o projetista descreve as funções lógicas desejadas através de linhas de programas (de forma textual) sem se preocupar de que forma o circuito será fabricado.

Porém, este ambiente proposto trabalhará num nível mais baixo de abstração: a descrição genérica do circuito será feita através de diagramas esquemáticos, utilizando-se símbolos genéricos dos quais o projetista conhece apenas a função lógica associada, mas não as suas características temporais.

A partir da descrição esquemática poderão ser realizadas simulações lógicas, do mesmo modo como ocorre no ambiente MaxPlus2, sem considerar os tempos de atrasos de propagação de sinais através das portas lógicas. Acredita-se que a definição e validação funcional da lógica do circuito seja uma das etapas mais demoradas do desenvolvimento do ASIC, além do mais qualquer erro de especificação inicial resultará no fracasso do projeto. Esta etapa, muitas vezes, é realizada pelo próprio cliente, previamente, ou em conjunto com o projetista.

Depois do circuito descrito e validado logicamente, realiza-se a escolha da forma de implementação do ASIC, tomando-se em consideração tanto critérios técnicos (desempenho elétrico, complexidade) quanto econômicos (volume de produção, *time-to-market*, riscos de projeto). O circuito poderá também ser implementado inicialmente com uma forma de implementação que garanta um curto *time-to-market,* para depois ser desenvolvido com outra que otimize seu desempenho elétrico, por exemplo. Para isso, não haverá a necessidade de descrever novamente o circuito nem elaborar seus vetores de simulação e teste, apenas ajustar as novas características de *timing*.

A escolha da descrição genérica do circuito através de esquemáticos objetiva quebrar as barreiras contra o uso de ASICs nas indústrias brasileiras. O uso de ferramentas de CAD como Orcad, Tango e Pcad, sobre plataformas PC são muito difundidos em nosso meio quando se refere à área da eletrônica digital. São poucas as empresas que investem em estação de trabalho

equipadas com ferramentas de síntese lógica, que permitem o desenvolvimento dos ASICs independentes da tecnologia de fabricação, como o AutoLogic da Mentor Graphics, por exemplo [27].

O ambiente proposto evita o choque cultural, técnico e financeiro imposto por estas ferramentas de CAD de última geração. O cliente continuará desenvolvendo seu esquemático, como vem sendo feito para as placas de circuito impresso que utilizam componentes *standard*, e o validará funcionalmente (apenas sua lógica). Dentro deste ambiente ele receberá o ASIC, configurado em um componente PLD, provavelmente no mesmo dia enquanto que a versão com CIs personalizáveis por máscaras já estará em fase de implementação.

Todavia, não é inviabilizada a possibilidade de uso de ferramentas para descrição textual (HDL) de ASICs pois pode-se gerar um esquemático, resultante da descrição HDL, utilizando-se a biblioteca genérica do ambiente proposto. A partir disso, o projetista conseguirá trabalhar dentro deste ambiente.

Capítulo 3

Tecnologias e arquiteturas dos PLDs

As formas de implementação de ASICs **personalizáveis após o encapsulamento**, ou **PLDs**, surgiram em meados da década de 70, com o desenvolvimento da tecnologia PROM (fusíveis).

O componente PLD é um circuito integrado que pode ser configurado pelo próprio usuário, em bancada de trabalho, para desempenhar uma determinada função lógica. Diferentemente dos CIs personalizáveis por máscaras, todos os PLDs de uma mesma família não apresentam distinções até a conclusão das etapas de fabricação de encapsulamento (*package*).

A rapidez na prototipação de circuitos integrados de aplicação específica através do uso de PLDs é uma de suas principais características, porém resulta em perda de flexibilidade de projeto devido à dependência com as arquiteturas pré-fabricadas e de características de *timing* bem definidas.

O fator de utilização, ou seja, a quantidade de funções lógicas implementadas em relação a capacidade lógica do componente, tem aumentado com o desenvolvimento de ferramentas de CAD cada vez mais eficientes. A perda de capacidade lógica disponível e, conseqüentemente, de área de pastilha não utilizada é inevitável. Porém, esta redução da flexibilidade e do fator de utilização, ou taxa de ocupação, é compensada com o baixo custo de desenvolvimento do ASIC e a sua rápida obtenção.

O fato do usuário personalizar PLDs na própria bancada de trabalho deve-se à utilização de **elementos** ou **dispositivos de programação**, existentes na arquitetura destes componentes, para definir a funcionalidade das células lógicas configuráveis, como também o roteamento de sinais internos.

Os elementos de programação comumente encontrados são os dispositivos EPROM e E^2PROM, células SRAM, fusíveis e anti-fusíveis [28]. Alguns destes dispositivos permitem reprogramabilidade, de forma que há PLDs que podem ser configurados inúmeras vezes e outros cuja programação é inalterável, conforme será visto adiante.

Outras características, como volatilidade e tamanho dos dispositivos, serão apresentados neste capítulo. Serão citadas, também, as inúmeras denominações que os fornecedores e a própria literatura utilizam para referenciar os PLDs. É comum encontrar nomes como EPLDs, FPGAs, FPLAs, PLAs, PALs, PMLs e tantos outros, referindo-se aos componentes de lógica programável, de maneira genérica.

Este capítulo tentará esclarecer estas denominações, discutindo o problema da taxonomia ainda inexistente para os CIs personalizáveis após o encapsulamento.

Por enquanto, a maneira mais ditática de apresentar os PLDs é dividi-los em dois grupos: os componentes com **arquiteturas básicas**, que foram os primeiros a surgir por volta dos anos 70 (**primeira geração**), e os componentes com **arquiteturas avançadas**, ou **segunda geração**, surgida no final dos anos 80, que tornam possível a implementação de ASICs com mais de 10.000 portas equivalentes [29].

3.1 Elementos de programação

Os elementos de programação, utilizados para personalizar os PLDs, são dispositivos (chaves) que permitem conectar, ou não, dois pontos estratégicos, definindo os caminhos de roteamento de sinais e as funções dos blocos lógicos configuráveis. Estes dispositivos são a base para a construção de componentes personalizáveis após o encapsulamento pois, através da sua programação, configura-se o PLD a fim de que ele realize determinadas funções lógicas.

Para melhor entendimento do funcionamento e da eficiência destes elementos, há várias características elétricas e físicas que devem ser consideradas como resistividade do dispositivo (maior causador da degradação de sinais internos), tamanho da área ocupada na pastilha, potência necessária para programação dos dispositivos. Estas características muitas vezes são transparentes para o usuário, que nem toma conhecimento das mesmas ao trabalhar como os PLDs. Porém duas são de grande importância, principalmente para a compreensão das vantagens de certas famílias de componentes e sua aplicabilidade:

- **volatilidade**: corresponde a dependência da configuração dos elementos de programação em relação a sua tensão de alimentação; componentes formados por elementos voláteis perdem a sua configuração na ausência do sinal de alimentação;

- **reprogramabilidade**: corresponde a possibilidade de reconfiguração do elemento de programação e, conseqüentemente, do componente depois que ele já tenha sido programado alguma vez.

A característica de reprogramabilidade, por si mesma, já declara sua importância. Utilizar componentes reprogramáveis para implementar ASICs significa tornar praticamente inexistentes os riscos de projeto. Além disso, permite a possibilidade de futuras alterações no funcionamento do circuito praticamente sem modificar o ambiente externo onde o ASIC está colocado.

A volatilidade, por outro lado, é uma característica que pode ser tanto benéfica como prejudicial ao sistema implementado. De uma maneira geral é mais vantajoso utilizar um componente que não precise ter sua configuração reestabelecida a cada ausência de tensão de alimentação. Este carregamento de

configuração acarreta no uso de um sistema computacional permanentemente ligado ao PLD ou de uma memória não-volátil auxiliar.

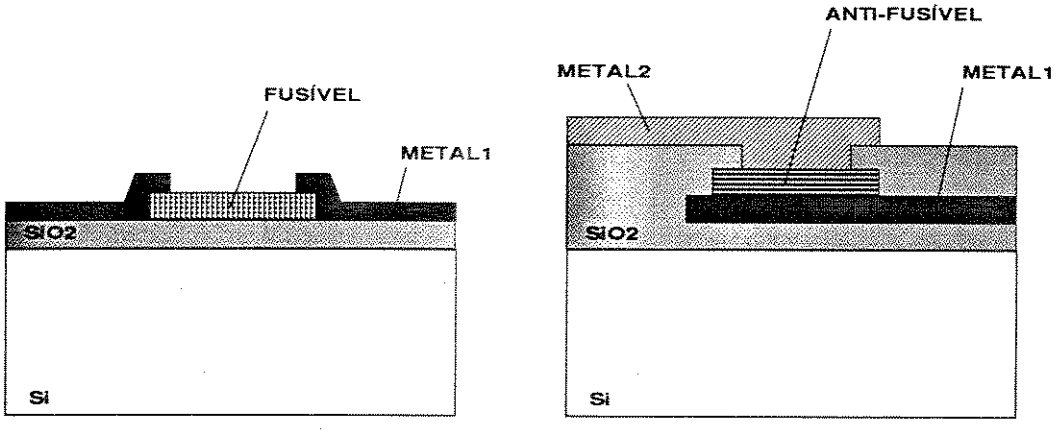
Entretanto, os componentes voláteis possuem áreas de aplicabilidade incontestáveis: para tarefas não simultâneas realizadas por uma mesma máquina é muito mais útil o uso destes componentes e diversas memórias de carregamento de configuração, havendo um ganho de área em relação a implementação de vários circuitos para cada uma das tarefas.

Outra aplicação dos PLDs com elementos de programação voláteis é a implementação de algoritmos de *software* em *hardware*. Há estudos para utilizar estes componentes *on-line* para realizar tarefas, previamente descritas em linhas de programas, ao invés de compilá-las em linguagens como Pascal ou C++, por exemplo. Alguns resultados apresentaram um ganho de até 100 vezes em desempenho destes algoritmos [30] [31].

Os PLDs disponíveis comercialmente apresentam como elementos de programação **fusíveis** (**PROM**), **anti-fusíveis**, células **SRAM**, dispositivos **EPROM** ou **E²PROM** [28] [32].

3.1.1 Fusível

Os **fusíveis**, ou dispositivos **PROM** - *Programmable Read-Only Memory*, foram os primeiros elementos de programação desenvolvidos e utilizados. Eles são formados por material condutivo de baixo ponto de fusão e mantêm conectados dois pontos (duas trilhas de metal, por exemplo) a menos que sejam queimados, provocando seu rompimento, e isolando estes pontos.



(a) (b)

Figura 3.1 - Corte transversal do fusível (a) e do anti-fusível (b).

Os fusíveis são elementos não-voláteis, não permitem reprogramabilidade e ocupam pequena área de pastilha, suficiente para colocar o material a ser queimado e garantir a isolação dos pontos após a programação (figura 3.1a).

Porém, eles são construídos em tecnologia bipolar que por sua vez não permite a alta integração de transistores como ocorre com a tecnologia MOS. Não se tem conhecimento de componentes comerciais que fazem uso de fusíveis MOS.

Por terem sido os primeiros elementos de programação criados e dominados, os fusíveis foram durante muitos anos os responsáveis pela existência de PLDs, no caso os componentes da primeira geração como os PALs. Porém, o fato de serem não reprogramáveis e serem fabricados em tecnologia bipolar, de menor densidade de transistores do que o MOS, faz com que os estes elementos não acompanhem a evolução dos componentes programáveis.

3.1.2 Anti-fusível

Os **anti-fusíveis** apresentam um princípio de funcionamento contrário aos dos fusíveis [33] [34]. Um material de baixa condutividade é colocado entre dois pontos a serem programados, isolando-os. Com a queima destes elementos, através da passagem de corrente, o material anti-fusível tem sua resistividade reduzida a valores muito baixos, passando de alguns mega-ohms para algumas centenas de ohms, permitindo desta forma a conectividade dos pontos citados (figura 3.1b).

Os anti-fusíveis, do mesmo modo que os fusíveis, são elementos não-voláteis, não reprogramáveis e que utilizam pequena área de silício (aproximadamente a área de um ponto conexão entre os níveis de metalização). A vantagem dos anti-fusíveis com relação aos fusíveis é que eles são fabricados com tecnologia MOS, que é a principal responsável pela integração de sistemas digitais cada vez maiores e de melhor desempenho.

3.1.3 Célula SRAM

As **células SRAM** (*Static Random Access Memory*) são usadas para controlar transistores de passagem, *transmission gates* ou multiplexadores que, por sua vez, realizam as conexões dos pontos estratégicos para configuração dos PLDs (figura 3.2) [30]. Os transistores de passagem e *transmission gates* permitem conectar ou isolar dois pontos, de acordo com o dado armazenado na célula SRAM, enquanto que o uso de multiplexadores permite inúmeras combinações possíveis entre pontos estratégicos. Esta última forma é útil para selecionar, dentre diversos sinais que chegam a uma célula lógica configurável, qual sinal será utilizado como variável de entrada.

A célula SRAM consiste de dois inversores realimentados sendo, portanto, um dispositivo volátil, facilmente reprogramável e que ocupa a área equivalente de aproximadamente seis transistores MOS: a célula SRAM, propriamente dita, mais a chave de conexão (transistor de passagem, *transmission gate* ou multiplexador), conforme a figura 3.3.

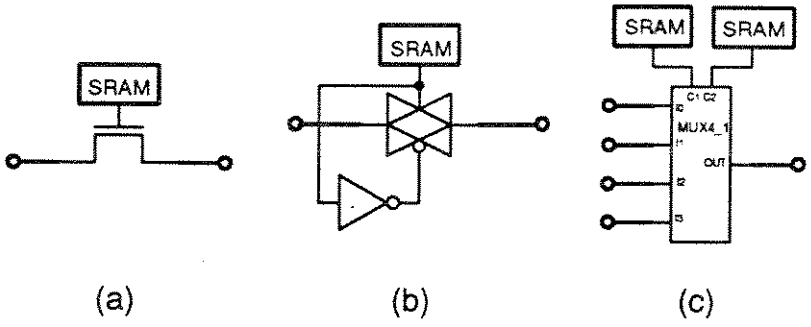


Figura 3.2 - Formas de programação com SRAM: (a) *pass transistor*, (b) *transmission gate* e (c) multiplexador.

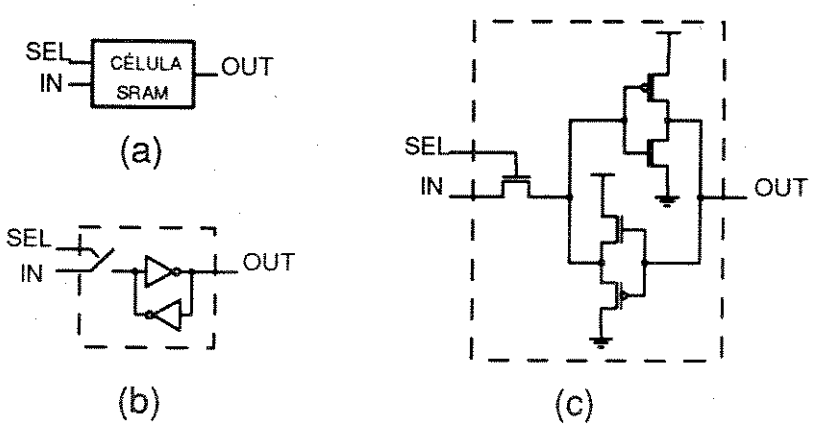


Figura 3.3 - Célula SRAM CMOS: (a) representação simbólica, (b) lógica e (c) elétrica.

3.1.4 Dispositivos EPROM e E²PROM

O dispositivo **EPROM** (*Erasable Programmable Read-Only Memory*) é um transistor MOS com dois *gates*: além do convencional ou principal, para controle do canal do transistor, há um *gate* flutuante, completamente isolado por óxido, que pode armazenar uma certa carga [35].

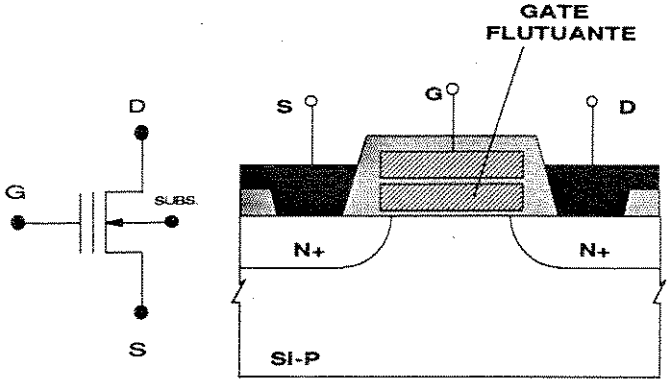
O *gate* flutuante situa-se entre o *gate* principal e o canal do transistor (figura 3.4). Se o *gate* flutuante esta descarregado, o estado do transistor é definido pelo *gate* principal, tal como o transistor MOS usual. Caso o *gate* flutuante esteja carregado o canal é mantido cortado (chave desligada), pois qualquer sinal colocado no *gate* principal é neutralizado por esta carga.

Para descarregar a carga armazenada no *gate* flutuante deve-se expor o componente à luz ultravioleta. Esta luz altera as propriedades do óxido que envolve o *gate* flutuante permitindo a passagem de elétrons através dele. Logo, o dispositivo EPROM é um elemento reprogramável, não-volátil e ocupa a área de aproximadamente um transistor MOS.

O dispositivo **E²PROM** (*Electrically Erasable Programmable Read-Only Memory*) apresenta um princípio de funcionamento semelhante ao EPROM, porém a descarga do *gate* flutuante é feita eletricamente [36] [46]. Assim, o dispositivo E²PROM também é reprogramável, não-volátil e ocupa praticamente o dobro da área de pastilha correspondente ao EPROM.

Um resumo das características dos elementos de programação apresentados pode ser visto na tabela 3.1. Quanto a reprogramabilidade, os elementos SRAM e E²PROM são reprogramáveis *in-circuit*, ou seja, não há necessidade de retirá-los do ambiente de trabalho (placa de circuito impresso) para reprogramá-los, enquanto que o elemento EPROM deve ser exposto à luz ultravioleta para apagar a sua configuração.

Quanto ao tamanho dos dispositivos de programação, os fusíveis e anti-fusíveis são aproximadamente do tamanho de uma conexão entre níveis de metalização, porém o fusível é realizado com tecnologia bipolar cujos transistores são muito maiores do que os transistores MOS, de modo que o PLD com fusíveis não possuem grande capacidade de implementação lógica.



GATE	GATE FLUTUANTE	CANAL
X	CARREGADO	OFF
0	DESCARREGADO	OFF
1	DESCARREGADO	ON

Figura 3.4 - Dispositivo de programação EPROM.

Elementos	volátil	reprogramável	tamanho
fusível (PROM)	não	não	pequeno (bipolar)
anti-fusível	não	não	pequeno (= via - MOS)
SRAM	sim	sim	6 trans. MOS
EPROM	não	sim	1 trans. MOS
E²PROM	não	sim	2 trans. MOS

Tabela 3.1 - Quadro comparativo dos elementos de programação.

3.2 Arquiteturas básicas

Qualquer função, seja combinacional ou seqüencial, pode ser representada em forma de **soma-de-produtos**, usando as leis de De Morgan e teoremas da álgebra booleana, para manipulação lógica (figura 3.5).

As arquiteturas da **primeira geração** de PLDs estão baseadas neste conceito de soma-de-produtos. Estes componentes são constituídos basicamente por um *array* de portas lógicas AND (*array* AND), conectado aos sinais de entrada do CI, seguido por um *array* de portas lógicas OR (*array* OR), que fornecem os sinais de saída do circuito. As entradas do CI são combinadas no *array* AND, onde são gerados os termos produtos, e estes, por sua vez, são enviados ao *array* OR, onde são somados, resultando nas somas-de-produtos responsáveis pela implementação da lógica requerida.

Um ou ambos os *arrays* AND e OR podem ser programáveis. Os primeiros PLDs, lançados no mercado, utilizavam fusíveis (PROM) como dispositivos de programação. Atualmente, os componentes da primeira geração podem ser encontrados com dispositivos EPROM ou E^2PROM.

De acordo com a possível programação dos *arrays* AND e/ou OR, os PLDs com arquitetura básica podem ser classificados em três categorias (figura 3.6) [37] [38]:

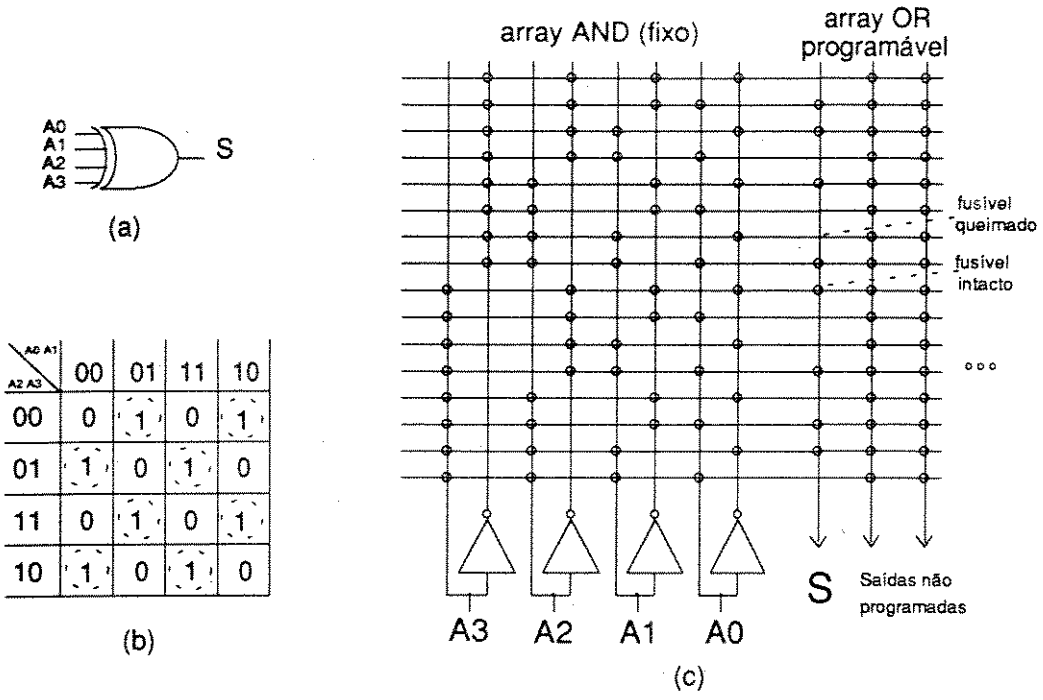
- **PROM**: apresenta o *array* AND fixo e o *array* OR programável;
- **PAL**: apresenta o *array* AND programável e o *array* OR fixo;
- **PLA**: ambos os *arrays* AND e OR são programáveis.

Para melhor representar os *arrays* AND e OR foi criado o conceito de "linha de produto" e "linha de soma", onde todas as entradas das portas lógicas AND e OR são representadas por uma linha apenas (figura 3.7).

Os componentes da primeira geração permitem implementar circuitos de baixa complexidade, em torno de 1.000 portas equivalentes. Quando o circuito é mais complexo, não havendo lógica suficiente disponível em um único PLD, este circuito pode ser particionado em vários componentes. Características adicionais para registro de dados e outras facilidades podem ser encontrados nestes PLDs, porém sua capacidade lógica está limitada devido ao tamanho dos *arrays* AND e OR. Para cada entrada ou saída adicionada ao componente os *arrays* AND e OR praticamente duplicam suas dimensões, aumentando excessivamente os tempos de propagação dos sinais através dos *arrays* e a quantidade de dispositivos de programação necessários.

Logo, os fabricantes estão restritos ao desenvolvimento de ferramentas de CAD para simulação e programação destes PLDs, e dependentes da evolução tecnológica do processo de fabricação, para aumentar a capacidade lógica dos componentes. Para circuitos com até 500 portas equivalentes, pequeno volume de

produção ou para desenvolvimento de protótipos, os PLDs da primeira geração dominam o mercado consumidor.



$$S = A0 \oplus A1 \oplus A2 \oplus A3$$

$$S = A0 \bullet \overline{A1} \bullet \overline{A2} \bullet \overline{A3} + \overline{A0} \bullet A1 \bullet \overline{A2} \bullet \overline{A3} + \overline{A0} \bullet \overline{A1} \bullet A2 \bullet \overline{A3} + \overline{A0} \bullet \overline{A1} \bullet \overline{A2} \bullet A3$$
$$+ A0 \bullet A1 \bullet A2 \bullet \overline{A3} + A0 \bullet A1 \bullet \overline{A2} \bullet A3 + A0 \bullet \overline{A1} \bullet A2 \bullet A3 + \overline{A0} \bullet A1 \bullet A2 \bullet A3$$

(d)

Figura 3.5 Representação em soma-de-produtos de uma função lógica:
(a) representação simbólica, (b) mapa de Karnaugh,
(c) implementação PROM e (d) equação booleana.

3.2.1 PROM (*Programmable Read-Only Memory*)

Os componentes **PROMs** têm tido um excelente desenvolvimento em tamanho e velocidade desde a sua introdução pela Monolithic Memories, com o lançamento da primeira memória PROM bipolar de 1K bit no mundo, em meados dos anos 70. O PROM pode ser utilizado eficientemente tanto como memória de alta velocidade, para armazenagem de dados, como para substituir circuitos lógicos combinacionais ou seqüenciais.

Não se deve confundir PLDs de arquitetura PROM com elementos de programação PROM (fusíveis). O componente PROM, ou melhor, o PLD com arquitetura básica PROM pode fazer uso de elementos de programação PROM, EPROM ou E^2PROM, apresentados anteriormente.

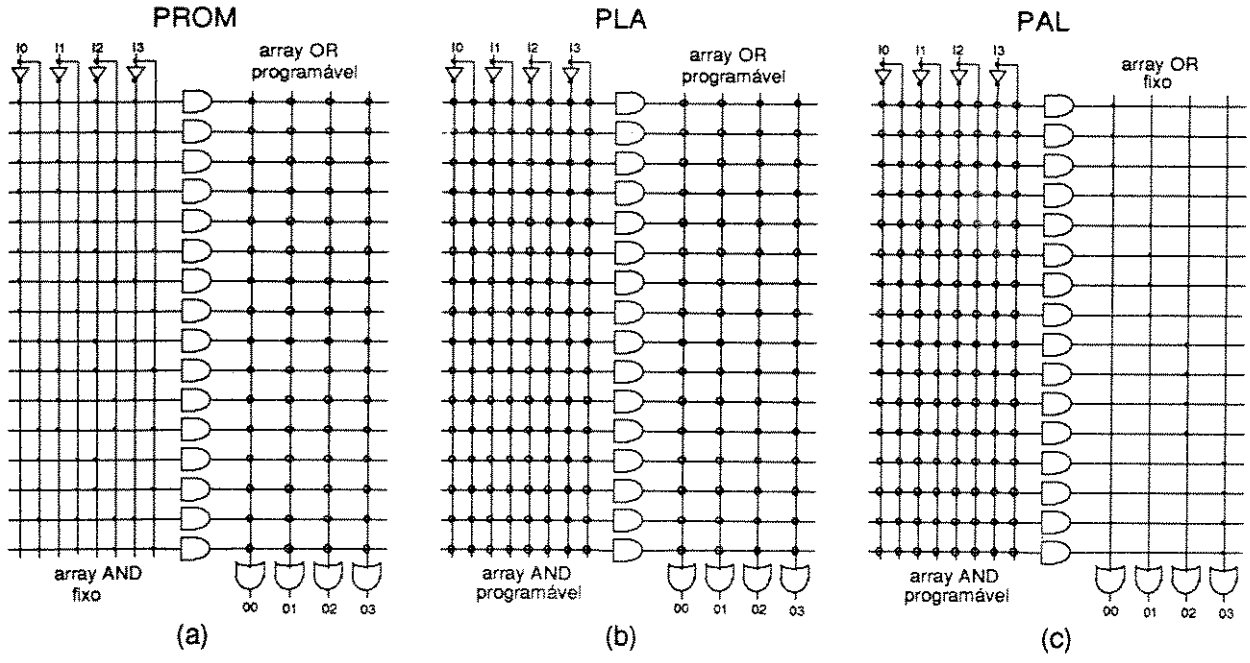


Figura 3.6 - Arquiteturas básicas de PLDs: (a) PROM, (b) PLA e (c) PAL.

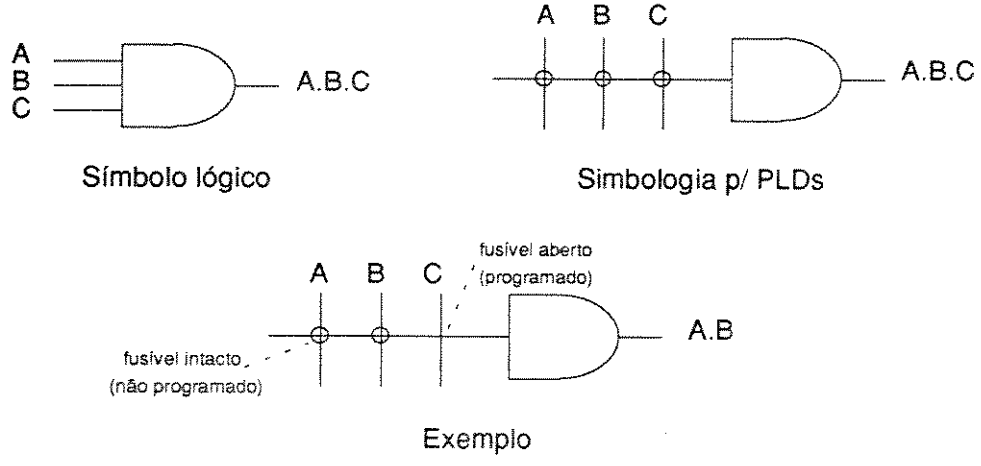


Figura 3.7 - Representação simbólica utilizada em PLDs.

Quando utilizado como memória, o *array* AND fixo do componente PROM decodifica o endereço da memória, enquanto que o *array* OR programável fornece o valor na saída definido pelo usuário. O componente PROM é chamado também de **PLE** (*Programmable Logic Element*) quando for utilizado para substituição de lógica aleatória [37]. Nesta aplicação cada porta lógica do *array* AND corresponde a um termo produto, com variáveis fixas, previamente decodificadas neste *array*. As saídas são as somas dos produtos definidas pelo usuário no *array* OR programável.

Desta forma os PLEs, ou PROMs, devem apresentar todos os termos produtos possíveis de serem gerados com as variáveis de entrada (*array* AND fixo). Logo, a estrutura destes PLDs é eficiente para implementar circuitos de lógica

aleatória que requeiram um grande número de combinações de entradas e de termos produtos por saída. Há, também, PLEs com registradores para facilitar a implementação de circuitos seqüenciais e máquinas de estados que requeiram um grande número de variáveis nas equações de cada estado.

Apesar da principal vantagem dos componentes PROMs ser esta decodificação prévia de todas as combinações de entrada já definidas no *array* AND, isto faz com que o número de variáveis de entrada permitido seja limitado, pois o tamanho dos *arrays* AND e OR devem ser dobrados a cada adição de nova entrada na arquitetura. A aritmética funciona assim: um PROM com X entradas e Y saídas requer um *array* OR de Y portas lógicas OR com 2^X entradas cada.

Por exemplo, um PROM de 10 entradas e oito saídas exige um *array* OR com 8 portas lógicas OR, cada uma contendo 2^{10} entradas, o que corresponde a 8.192 posições de fusíveis. Para aumentar uma entrada no componente PROM o *array* OR dobraria de tamanho passando para 16.384 posições de fusíveis de interconexão.

Estatisticamente, há um número limitado de termos produto nas equações booleanas que descrevem o funcionamento de um circuito. Assim, a flexibilidade do *array* OR programável existente nos componentes PROMs, decorrentes da disponibilidade de todos os termos produto possíveis, é normalmente desnecessária.

Porém, a implementação de certas funções que exigem um grande número destes termos produto, tais como funções OR exclusivo (XOR), é mais eficiente com este componente (figura 3.5). Por exemplo, um XOR de cinco entradas deve ser implementado utilizando 16 termos produto. Isto pode exceder o número de termos produto disponíveis em um PAL ou utilizar muitos termos produto em um PLA, que são maiores e mais lentos que o PROM. Este compromisso entre flexibilidade, custo e desempenho sempre deve ser considerado na escolha do componente adequado a uma determinada aplicação.

Os componentes PROM apresentam, atualmente, maior desenvolvimento para aplicação em memórias. Os PLEs tendem a desaparecer do mercado mundial, devido à maior eficiência do uso de arquiteturas PAL para implementação de lógica aleatória de pequena complexidade, conforme será visto mais adiante.

3.2.2 PLA (*Programmable Logic Array*)

Por apresentar ambos os *arrays* AND e OR programáveis, o **PLA** é o componente que permite maior flexibilidade na implementação das somas-de-produtos.

A arquitetura do PLA contém um dispositivo de programação por função lógica a mais do que as arquiteturas PAL e PROM, correspondendo a um acréscimo em área de pastilha e uma diminuição da velocidade de funcionamento.

Porém, o PLA é muito eficiente para circuitos seqüenciais, onde sua flexibilidade permite implementar máquinas de estado maiores.

Por causa desta característica, há uma família de PLAs, chamada **PLS** (*Programmable Logic-based Sequencer*), cujos componentes apresentam algumas características próprias que facilitam a implementação de circuitos para aplicações específicas em seqüenciadores (figura 3.8) [39]. Para permitir a implementação de máquinas de estado mais complexas, os PLSs perdem um pouco da sua generalidade, tornando-se ineficientes para aplicações gerais.

A possibilidade de programação do *array* AND evita a restrição encontrada nos PROMs, onde este *array* deve ser suficiente para permitir todas as possíveis combinações de entrada. Isto ocorre, como já foi dito, porque apenas um número limitado de termos produto é requerido nas equações lógicas de um circuito. Eliminando combinações redundantes com técnicas de minimização lógica, tais como **mapas de Karnaugh** e **álgebra booleana**, pode-se reduzir ainda mais o número de termos produto em certas aplicações. Desta forma quase todas as combinações de entrada podem ser decodificadas em um PLA.

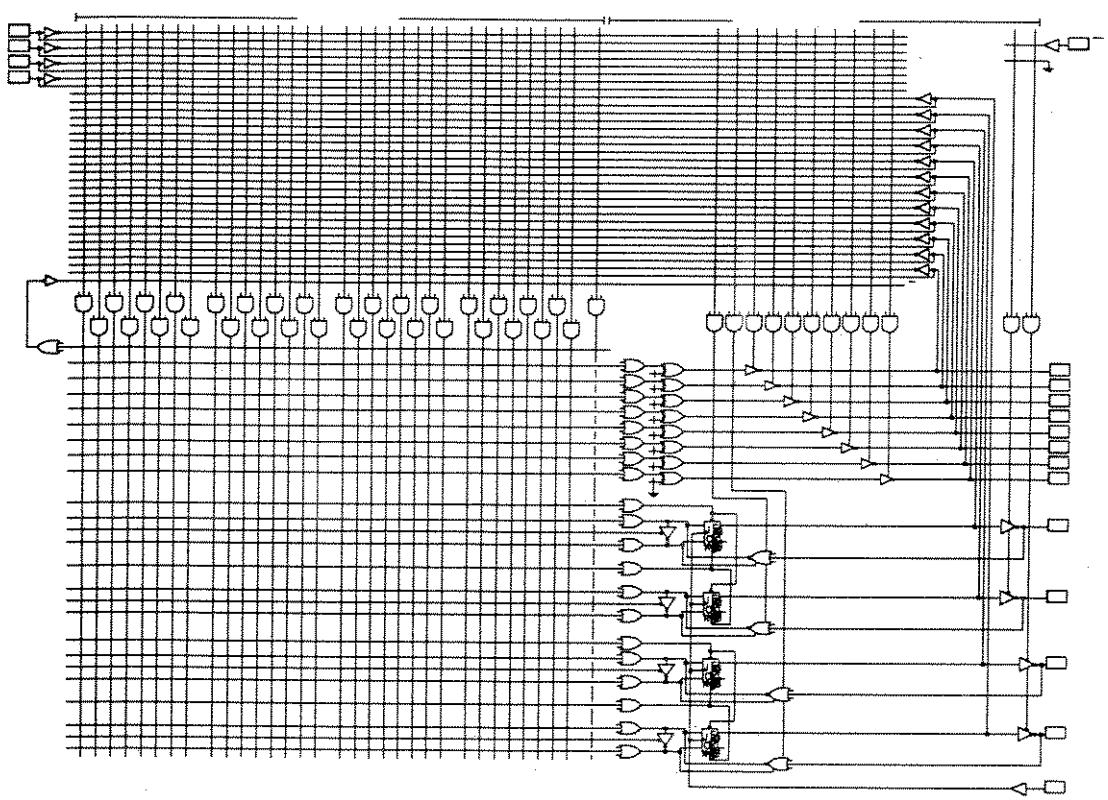


Figura 3.8 - PLS: um PLA específico para implementação de máquinas de estado.

Devido à longa história dos PLAs, sua nomenclatura pode gerar pequenas confusões. Os primeiros fabricantes de PLAs programáveis pelo usuário chamaram seus produtos de FPLA (*Field Programmable Logic Array*) para distinguir dos PLAs personalizados por máscaras [25].

Porém, os fabricantes acabaram simplificando a nomenclatura e, assim, tanto os PLAs mascaráveis quanto os que são programáveis pelo usuário recebem o mesmo nome. Normalmente, PLAs personalizáveis por máscaras são gerados automaticamente com ferramentas de CAD.

Os PLAs programáveis pelo usuário tendem a perder seu nicho de mercado para os PALs da mesma forma que os PROMs, devido a maior eficiência dos PALs para implementação de ASICs com menos de 1.000 portas equivalentes. A única área de aplicação não concorrida com demais PLDs de baixa complexidade (primeira geração) é aquela para a qual estão voltados os PLSs.

3.2.3 PAL (*Programmable Array Logic*)

Os componentes de maior sucesso entre os PLDs da primeira geração são os **PALs** [40] [41]. Para aplicações de baixa complexidade, onde os PLDs de arquitetura básica encontram seu nicho de mercado, os PALs apresentam-se mais eficientes na utilização da área da pastilha, velocidade de funcionamento e custos de projeto e implementação. Até o surgimento da segunda geração de PLDs, os PALs também foram os mais eficientes na implementação de circuitos de média complexidade e baixo volume de produção. Neste caso, particionava-se automaticamente o circuito em diversos PALs devido à limitação de funções lógicas implementáveis nestes componentes.

O fato do *array* AND ser programável em um PAL torna possível para o componente ter muitas entradas, enquanto que o *array* OR fixo o mantém pequeno e rápido, porém com número limitado de termos produto por saída (figuras 3.9 e 3.10) [38].

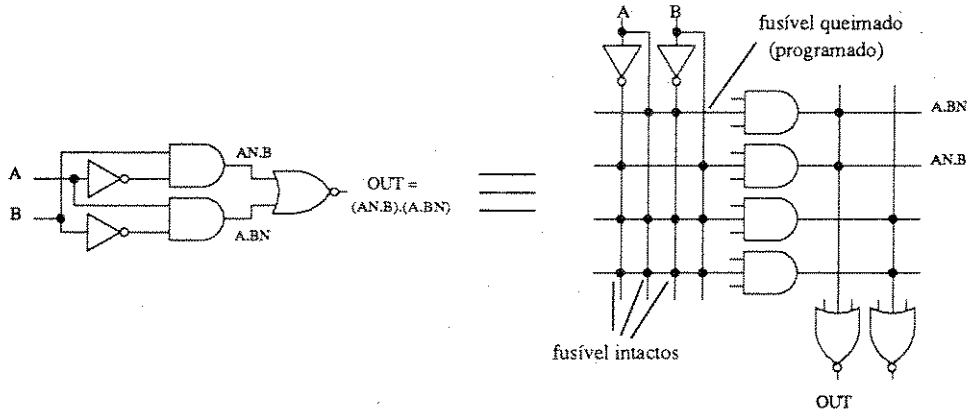


Figura 3.9 - Exemplo de programação de um PAL.

Os PALs são encontrados em diferentes tecnologias, tais como TTL, CMOS e ECL. Os componentes CMOS permitem as mesmas funções dos TTLs e podem ser utilizados com a mesma pinagem, com benefício de menor consumo de potência. Eles ainda permitem a reprogramabilidade pois podem ser configurados

com dispositivos EPROM ou E²PROM. Os PALs TTL são mais rápidos e estão disponíveis apenas em tecnologia PROM. Os componentes ECL, por sua vez, destinam-se a aplicações de alta velocidade e requerem considerações de projeto completamente diferentes de PALs CMOS e TTL, não podendo ser substituídos por componentes destas tecnologias.

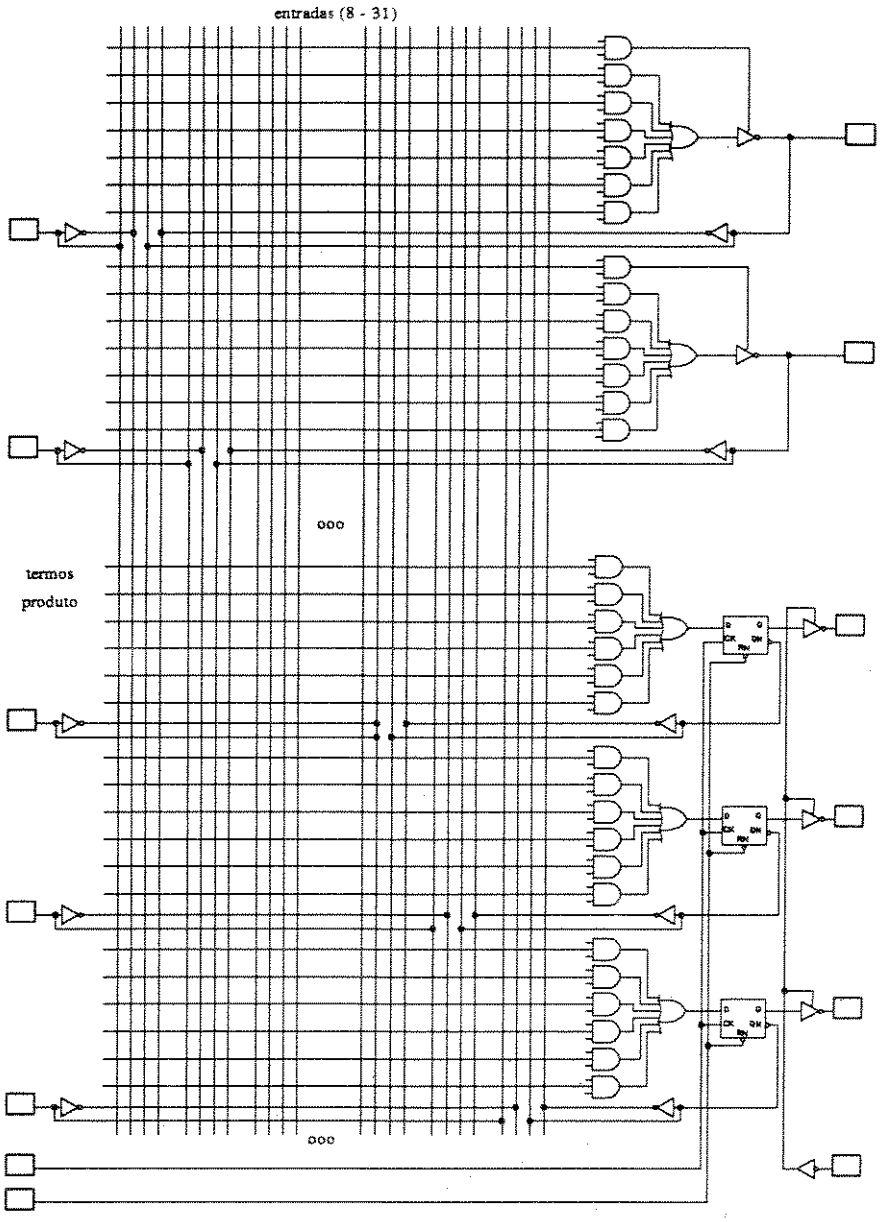


Figura 3.10 - PAL: *array* AND programável, *array* OR fixo.

De acordo com o consumo podemos classificar os PALs CMOS em dois tipos: *zero power* e *low power*. O componente *zero power* corresponde àquele que, quando em estado quiescente, praticamente não consomem potência. O componente *low power* é mais rápido e, mesmo em estado quiescente, drena uma quantidade de corrente nominal, porém menor do que os bipolares equivalentes. A diferença básica é que, a cada variação de sinal no componente *zero power,* existe

um consumo apenas durante a transição e em seguida o componente retorna a condição estática onde, praticamente, não há consumo de potência. Isto torna o circuito mais lento. Existem componentes que possuem um termo produto específico para permitir a escolha de um dos modos de funcionamento.

Devido ao seu sucesso, muitos projetistas passaram a usar as denominações PAL e PLD como sinônimos. Enquanto existia apenas a primeira geração de componentes programáveis pelo usuário, esta troca de nomenclatura não causava muita confusão pois os componentes PLAs e PROMs eram inexpressivos para implementação de lógica aleatória. Porém, devido ao surgimento da segunda geração e suas inúmeras denominações, aconselha-se destinar o termo PLD para referir-se, genericamente, aos componentes personalizáveis após o encapsulamento.

Para grandes volumes de produção, o PAL pode ser substituído por um componente personalizado por máscaras, chamado HAL (*Hardwired Array Logic*) [25]. O componente HAL troca os fusíveis do PAL por uma etapa final de máscara, que metaliza permanentemente as conexões lógicas. Esta alternativa aproxima o PAL de aplicações que são atraentes para os pré-difundidos. O HAL combina o mais baixo custo variável para grandes volumes de produção dos pré-difundidos, discutidos anteriormente, com as facilidades de prototipação dos PALs. Esta conversão para componentes mascaráveis pode ser apropriada uma vez que o circuito já encontra-se validado com PALs.

3.3 Arquiteturas avançadas

Conforme apresentado, a primeira geração de CIs personalizáveis após o encapsulamento baseia-se na implementação de funções lógicas através de equações booleanas na forma de soma-de-produtos. Mostrou-se também que os componentes PROMs apresentam número limitado de entradas, enquanto que os PALs estão restritos a um pequeno número de termos produto por saída. Os PLAs, por sua vez, são mais flexíveis na implementação das equações em soma-de-produtos, porém mais lentos devido à propagação dos sinais através de dois *arrays* programáveis. Apesar destes componentes se complementarem, tanto estrutural quanto funcionalmente, os PALs apresentam-se mais eficientes dentre os PLDs com arquitetura básica.

Todavia, os fabricantes de PALs, PLAs e PROMs dependem da evolução do processo de fabricação para conseguir aumentar a capacidade de integração destes componentes, sem perder em desempenho elétrico. Por conseqüência, é necessário limitar o tamanho dos *arrays* AND e OR e a quantidade de elementos de programação, a fim de manter os atrasos de propagação de sinais em ordens de grandeza aceitáveis.

Objetivando aumentar a capacidade de implementação lógica dos PLDs, surgiram novas arquiteturas, cujas princípios de configuração diferem do simples uso de equações na forma de soma-de-produtos para descrição do funcionamento

dos circuitos [29]. Esta **segunda geração** de componentes personalizáveis após o encapsulamento mostra-se eficiente na implementação de circuitos com mais de 1.000 portas equivalentes, quando o volume de produção e o tempo para obtenção do ASIC inviabilizam o uso de CIs personalizáveis por máscaras.

Dentre os componentes da segunda geração, a quantidade de denominações recebidas pelos PLDs é ainda maior do que na primeira. Inicialmente, os componentes serão divididos em três categorias segundo as denominações utilizadas pelos próprios fabricantes: **EPLDs**, **FPGAs** e ***folded-arrays***. Isto não quer dizer que esta seja a melhor forma de classificá-los. Após a descrição suscinta destas categorias, será mais fácil discutir e propor uma melhor taxonomia para os PLDs.

Os **EPLDs** (*Erasable Programmable Logic Devices*) são uma evolução natural da arquitetura dos tradicionais PALs, com um *array* de portas lógicas AND programável e outro com portas lógicas OR fixas. Os primeiros componentes foram lançados por volta de 1988 [81] [82].

Estes componentes diferem dos PALs por apresentarem *arrays* de interconexão programáveis, diversas realimentações de sinais, macrocélulas com grande capacidade de manipulação lógica e armazenagem de dados, blocos de I/O (entradas e saídas do CI) independentes das macrocélulas e *arrays* de expansão de termos produto. Os EPLDs são encontrados com tecnologias EPROM e E^2PROM, sendo a empresa Altera Corp. o seu principal fabricante [18] [42] [43].

Os **FPGAs** (*Field Programmable Gate Arrays*), referenciados também por alguns fabricantes como **LCAs** (*Logic Cell Arrays*), utilizam como elementos de programação células de memória RAM estática (SRAM) e anti-fusíveis [BUR92c]. Os FPGAs recebem esta denominação por assemelharem-se muito com a arquitetura *gate array* dos pré-difundidos, com áreas de configuração lógica e canais de roteamento [REI92] [GÜN93]. A Actel Corp. é a principal fornecedora dos componentes com anti-fusíveis, enquanto que a Xilinx, a Cuncurrent Logic e a Algotronix são fabricantes que utilizam células SRAM em seus produtos [44] [45].

Finalmente, os componentes ***folded-arrays***, fornecidos pela Signetics Corp. e pela Exel Corp., são PLDs que utilizam *arrays* programáveis contendo apenas portas lógicas OR ou AND, realimentadas, para implementação da lógica do circuito. Nos *folded-arrays*, ao invés de somas-de-produtos, o funcionamento lógico é descrito apenas com somas (funções OR) ou produtos (funções AND). Para saída de sinais, ainda podem ser encontradas algumas características adicionais, como registro de dados ou sinais *tri-state*. Estes componentes utilizam tecnologias PROM e EPROM [39] [46].

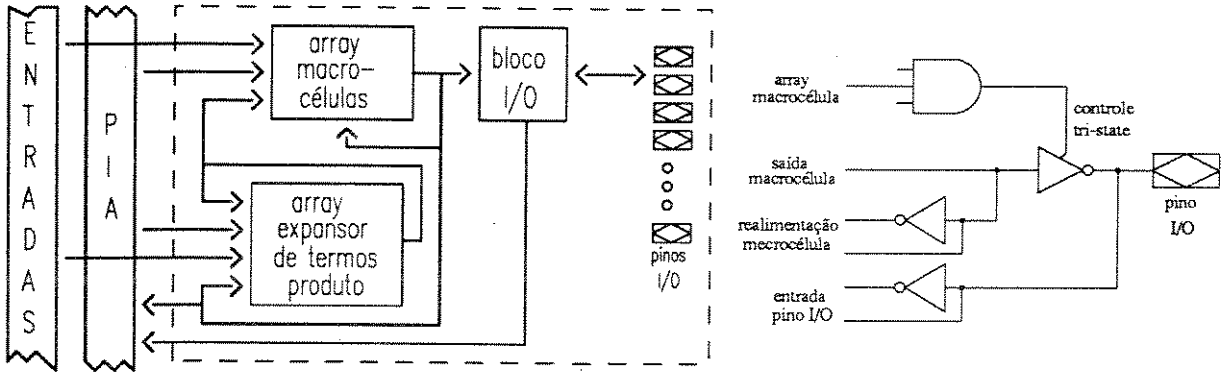
Da mesma forma que as arquiteturas evoluíram para aumentar a capacidade de integração dos PLDs, as ferramentas de suporte necessárias para a implementação de ASICs sobre os componentes programáveis tiveram que acompanhar esta evolução, a fim de permitir um bom fator de utilização dos PLDs [47].

3.3.1 EPLD (*Erasable Programmable Logic Device*)

O princípio de funcionamento dos componentes EPLDs tem origem nos PALs, com *arrays* ANDs programáveis e *arrays* OR fixos. Porém, o que diferencia um EPLD de um PAL é que, enquanto no PAL existe apenas um *array* AND programável, no EPLD existem vários sub-circuitos com arquitetura e capacidade semelhante a um PAL, interconectados por um *array* programável. Com isso, os sub-circuitos podem funcionar independentemente ou serem interligados dentro do componente. Pode-se fazer a analogia de que um EPLD corresponde a uma placa de circuito impresso, contendo vários PALs interligados pelas trilhas de roteamento da placa.

A tecnologia de programação utilizada nestes componentes é EPROM ou E^{2}PROM. Logo, é permitida a reprogramabilidade, apagando-se a configuração anterior por luz ultravioleta ou eletricamente. O principal fabricante é a empresa Altera Corp., que faz uso do processo CMOS 0,8 micra da Cypress Semiconductor Corp. para construir os dispositivos de programação. A arquitetura descrita neste trabalho baseia-se nos componentes da Altera [18], observando-se que os EPLDs de outros fabricantes possuem um princípio de funcionamento semelhante [42] [43].

A implementação das funções lógicas é realizada em sub-circuitos chamados de *Logic Array Blocks* (**LABs**) [48]. Cada LAB contém uma ou mais **macrocélulas**, **blocos de I/O** e um **circuito de expansão de termos produto** (figura 3.11a).



(a) (b)

Figura 3.11 - EPLD da Altera: (a) *Logic Array Block*; (b) bloco de I/O.

Uma **macrocélula** (célula lógica) é formada por um *array* de portas lógicas AND conectadas por portas OR para a implementação de soma-de-produtos (figura 3.12). O sinal da macrocélula pode ser armazenado em *flip-flops* ou enviados novamente ao *array* AND, permitindo realimentação de sinais. Os sinais de controle dos *flip-flops* permitem inúmeras facilidades para implementação do circuito no EPLD.

Os **blocos de I/O** funcionam independentemente da macrocélula, ao contrário do que ocorre com os PALs, onde o uso de um bloco de I/O para entrada de sinais impede o uso lógica disponível na macrocélula. Isto é conseguido com a existência de uma célula *tri-state* entre o bloco de I/O e a macrocélula (figura 3.11b).

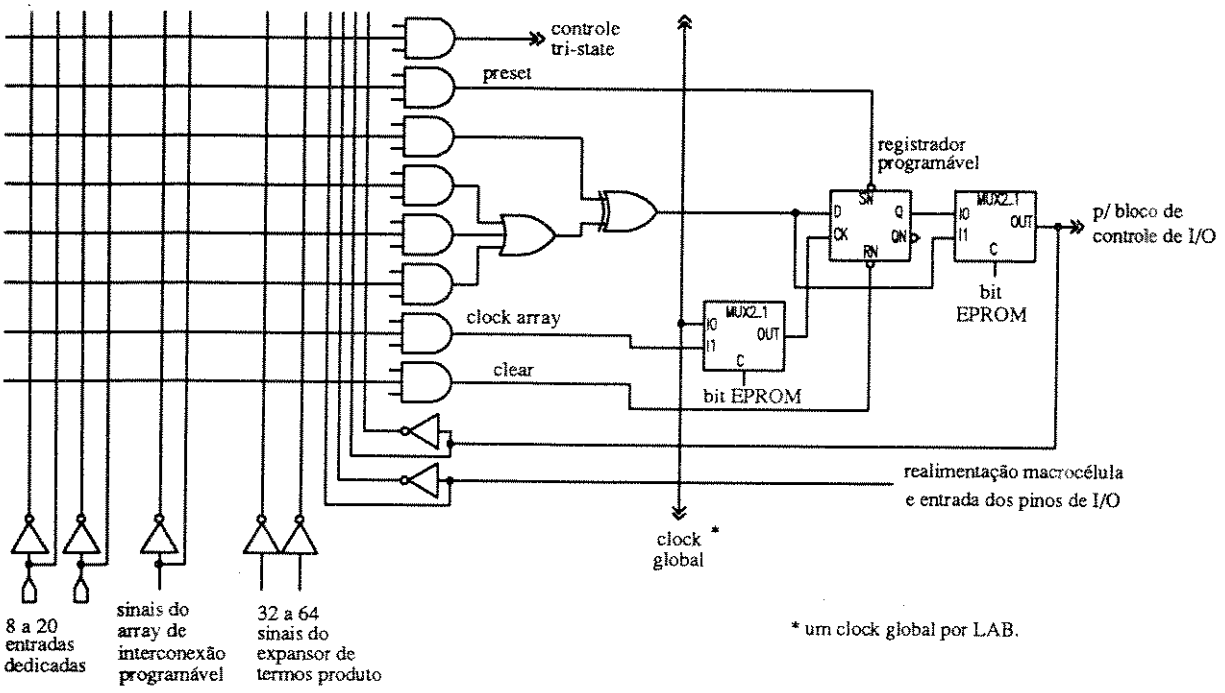


Figura 3.12 - Macrocélula (célula lógica) de um EPLD.

O **circuito de expansão de termos produto** é encontrado somente nos EPLDs e corresponde a um *array* extra de portas lógicas AND, que pode ser utilizado para suprir a falta de termos produto no *array* da macrocélula ou para implementar *flip-flops* para armazenagem de dados. Com isso, o *array* AND utilizado na macrocélula é otimizado para conter um número médio de termos produto existentes em uma equação booleana, definidos estatisticamente.

Os LABs estão interligados entre si através do *array* de interconexão programável (**PIA** - *Programmable Interconection Array*), formado por dispositivos EPROM ou E²PROM e trilhas de metal fixas. Este *array* de interconexão é o principal responsável pela alta capacidade de integração permitida pelos EPLDs (figura 3.13). O fabricante garante, ainda, que o atraso de propagação de sinais através do PIA são fixos e bem definidos, devido à capacitância equivalente das trilhas de metal do *array* serem semelhantes. Desta forma, simplifica-se a análise de sincronismo de sinais dentro do componente.

A Altera possui várias famílias de EPLDs que diferem entre si na disposição interna dos LABs e do PIA. A família mais utilizada é a MAX5000 que integra até 384 *flip-flops* e possui até 84 pinos de I/O configuráveis, equivalendo a mais de 10.000 portas equivalentes utilizáveis [17] [18] [48].

É importante observar os tipos de encapsulamentos para os EPLDs pois eles interferem na reprogramabilidade: o encapsulamento cerâmico apresenta janela para exposição à luz ultravioleta, e o encapsulamento plástico não possui esta janela, impedindo a reprogramação do componente. O encapsulamento plástico é utilizado quando o circuito já encontra-se validado, a fim de reduzir o preço do componente.

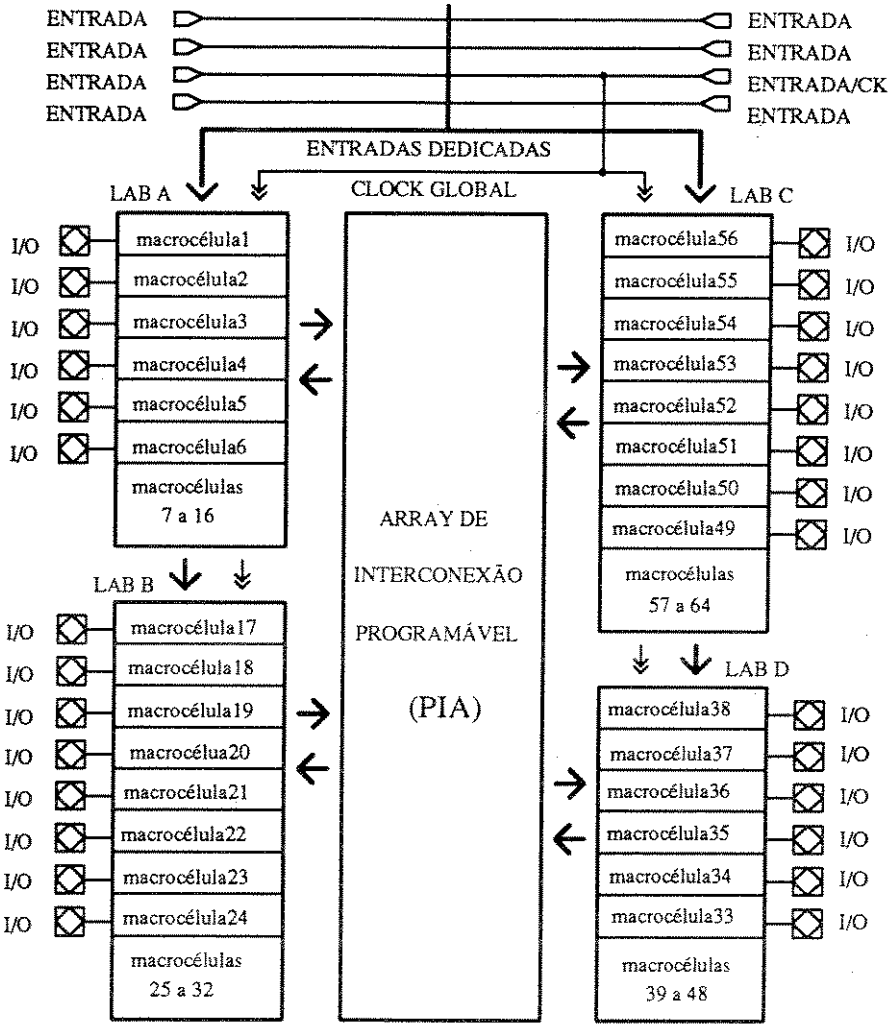


Figura 3.13 - Arquitetura de um EPLD da família MAX 5000 da Altera.

3.3.2 FPGA (*Field Programmable Gate Array*)

Os **FPGAs**, ou **LCAs**, são componentes personalizáveis após o encapsulamento que tiveram sua origem nas arquiteturas da abordagem *gate array* (CIs personalizáveis pela metalização) [49], com áreas de configuração lógica separadas por canais de roteamento e blocos de I/O na periferia do circuito.

O princípio da arquitetura do FPGA é dispor os elementos lógicos em áreas específicas e interligá-los por trilhas de metal fixas, enquanto que na abordagem

gate array estas trilhas de metal são definidas pelas máscaras de processo de fabricação do circuito. O que possibilita a roteabilidade nos FPGAs são as chaves de passagem, que conectam as trilhas de metal fixas entre si e/ou nas células lógicas configuráveis. Estas chaves podem ser anti-fusíveis ou transistores controlados por células de memória SRAM.

Devido a sua origem, FPGAs e EPLDs apresentam características bem distintas de funcionamento e desempenho elétrico. Nos FPGAs a célula lógica configurável é mais genérica, com a possibilidade de configuração de inúmeras funções lógicas com até 5 variáveis de entrada, enquanto que nos EPLDs as macrocélulas são específicas para a implementação de equações booleanas na forma de soma-de-produtos, com a possibilidade de registro do sinal de saída.

Outra diferença entre EPLDs e FPGAs é a forma de interligação dos blocos lógicos. Nos FPGAs as chaves de passagem permitem o uso de tantas trilhas quantas forem necessárias para o roteamento do sinal, enquanto que nos EPLDs utiliza-se o *array* de interconexão programável (PIA). Diferentemente do caso dos FPGAs, este *array* permite qualquer roteamento de sinal entre os LABs, utilizando-se, por exemplo, para a família MAX 5000 da Altera, sempre três trilhas fixas de metal: a de saída de um LAB, a de entrada de outro, e a terceira para ligar as duas anteriores. Isto faz com que os atrasos de propagação dos sinais, através dos canais de roteamento nos FPGAs, sejam variáveis e dependentes do próprio roteamento gerado pelo *software* de programação, enquanto que nos EPLDs estes atrasos são sempre fixos e previsíveis, simplificando as simulações lógicas [17].

Os FPGAs disponíveis comercialmente podem ser divididos em dois grupos, segundo os elementos de programação utilizados: **FPGAs com anti-fusíveis** e **FPGAs com células SRAM**.

a) FPGAs com anti-fusíveis

Os anti-fusíveis surgiram na década de 80, revolucionando os elementos de programação utilizados em PLDs por serem construídos em tecnologia MOS [50]. Estes dispositivos assemelham-se em tamanho com os fusíveis, sendo, portanto, menores do que os EPROMs, E^2PROMs e células SRAM, possibilitando maior densidade de integração. Porém, da mesma forma que o dispositivo PROM (fusível), os anti-fusíveis apresentam a desvantagem de não permitirem reprogramação.

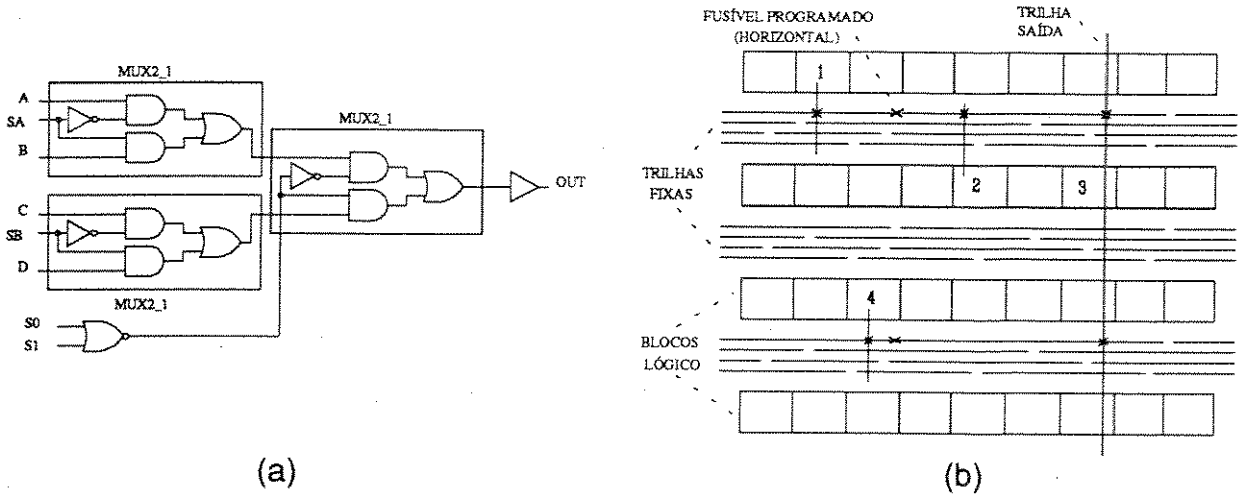
Os **FPGAs com anti-fusíveis** são fabricados pelas empresas Crosspoint, Actel e QuickLogic [51] [52]. A Actel foi a primeira empresa a tirar proveito destes elementos de programação, utilizando uma arquitetura praticamente idêntica a abordagem *gate array* dos pré-difundidos, com linhas de células lógicas (ou **módulos lógicos**, como são denominados pela empresa) separadas por **canais de roteamento**.

Cada **módulo lógico** do FPGA da Actel corresponde a dois multiplexadores 2x1 ligados na entrada de um terceiro multiplexador 2x1. Com isso, tem-se cerca de sete entradas e saída única em cada módulo. De acordo com a escolha das entradas e a fixação das restantes nos níveis lógicos 0 ou 1, o bloco lógico realiza todas as funções lógicas de duas entradas, a maioria das funções com três entradas e muitas funções com quatro entradas (figura 3.14a). Por exemplo, para implementar um *flip-flop* tipo D com *set* e *reset*, que utiliza cerca de 14 pares de transistores MOS em um *gate array*, são necessários dois módulos lógicos deste FPGA [53].

Os módulos lógicos estão dispostos um ao lado do outro, compondo linhas de configuração lógica. Entre estas linhas lógicas encontram-se trilhas de metal pré-difundidas e com variadas extensões. Estes espaços da arquitetura, entre as linhas lógicas, são definidos como **canais de roteamento**, da mesma forma que nos *gate arrays* (figura 3.14b).

As entradas e saídas das linhas lógicas estão disponíveis em trilhas verticais, implementadas com um segundo nível de metalização. As entradas estão conectadas a trilhas verticais que se prolongam apenas até o canal de roteamento adjacente, enquanto que as saídas estão disponíveis em trilhas verticais que cruzam todos esses canais. Há também trilhas verticais de metal2 cruzando os módulos lógicos com a finalidade de permitir a passagem do sinal de um canal de roteamento para outro.

Os **anti-fusíveis** são dispositivos bidirecionais e encontram-se situados nos cruzamentos de trilhas verticais e horizontais de roteamento, permitindo a troca dos níveis de metal para propagação de sinais.



(a) (b)

Figura 3.14 - FPGA da Actel: (a) módulo lógico; (b) arquitetura básica.

Há anti-fusíveis também entre trilhas adjacentes, horizontais ou verticais, a fim de permitir longos caminhos em um mesmo sentido, como também para evitar capacitâncias parasitas e desperdícios de trilhas para a ligação de pontos próximos (figura 3.15). Devido à resistência destes dispositivos de programação, evita-se a

passagem de um determinado sinal por mais de três anti-fusíveis, de forma a evitar a degradação excessiva do nível de tensão deste sinal.

Os blocos para entrada e saída de sinais (**blocos de I/O**) também são configuráveis por anti-fusíveis, assumindo as funções de circuitos de entrada, saída ou bidirecionais, e estão dispostos na periferia do componente.

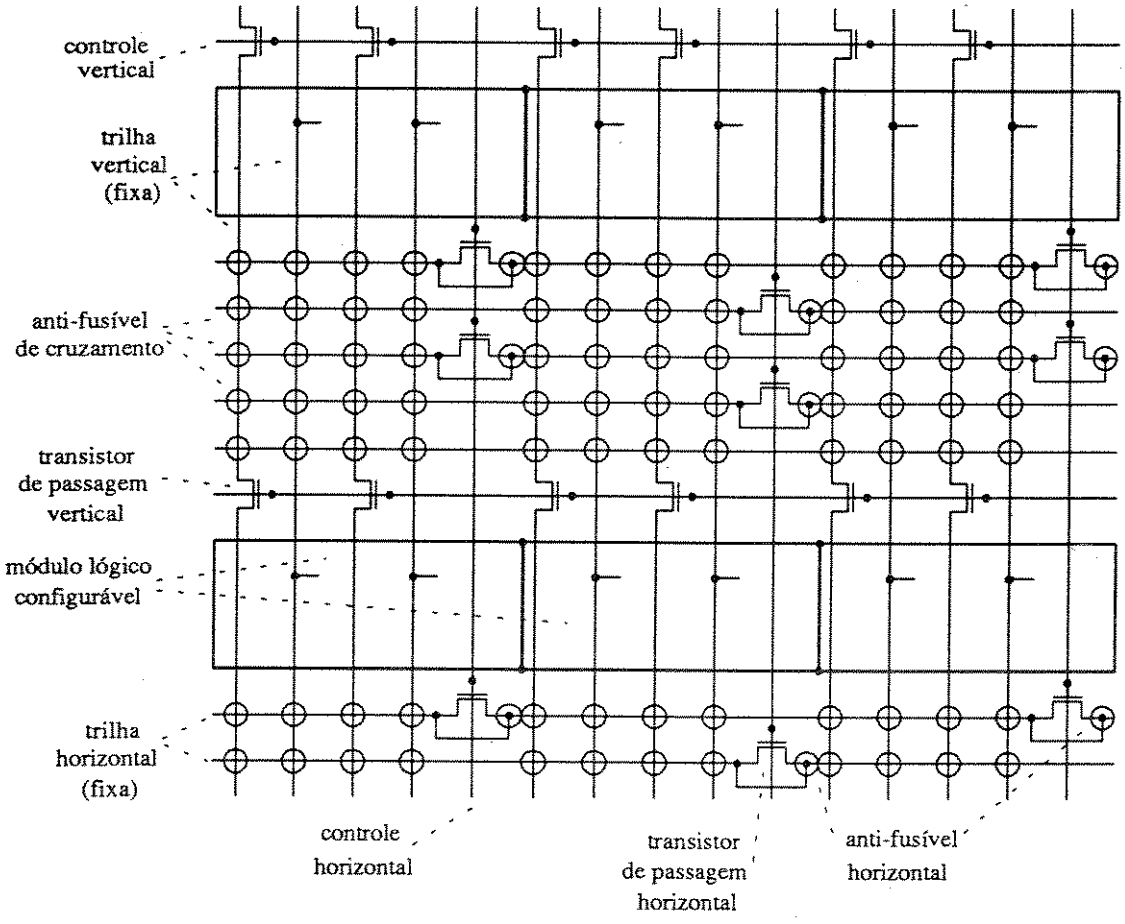


Figura 3.15 - Detalhe dos anti-fusíveis do FPGAs da Actel.

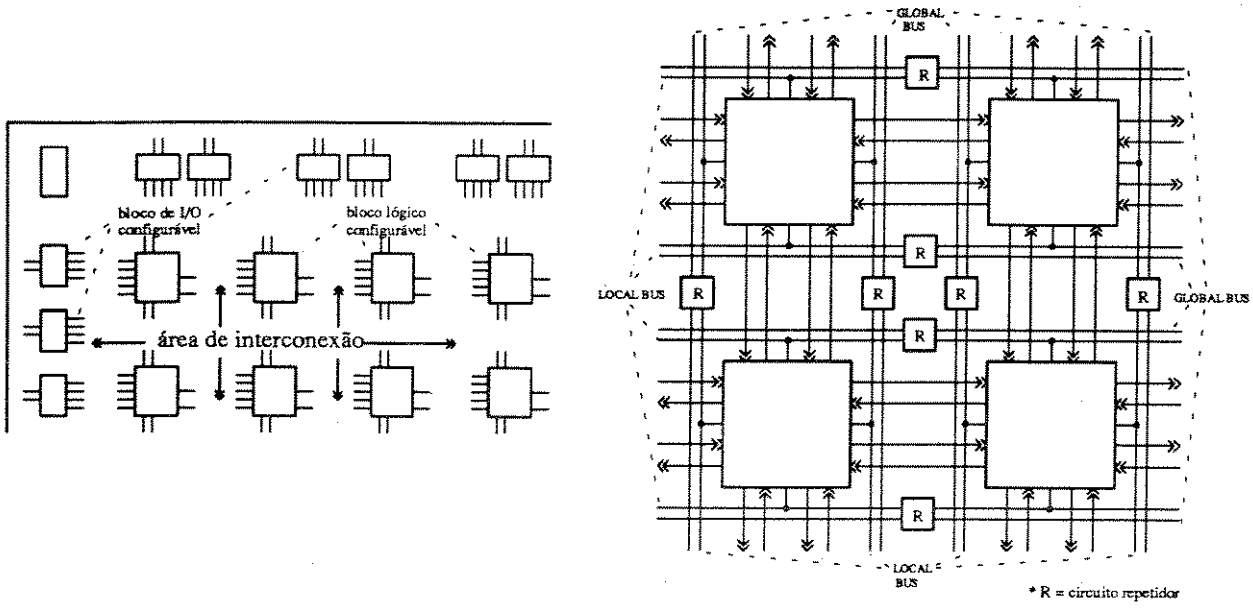
b) FPGAs com SRAM

O primeiro FPGA a utilizar células SRAM para sua configuração foi lançado em 1986. Este componente, da empresa Xilinx possuía aproximadamente 1.000 portas equivalentes [54]. Atualmente, os **FPGAs com SRAM** podem chegar a 20.000 portas equivalentes de capacidade de implementação lógica [45] [55] [56].

A tecnologia SRAM, conforme visto anteriormente, é volátil, ou seja, na ausência de tensão de alimentação no componente os dados das células de memória e, conseqüentemente, da configuração do FPGA, são perdidos. Isto obriga ao usuário carregar, via circuito externo ou memória auxiliar, a configuração do componente toda vez que ele for ligado. Porém, devido a esta característica, é muito fácil utilizar estes FPGAs para implementar um circuito multi-tarefa, dentro de

um sistema onde tais tarefas não ocorram simultaneamente, bastando utilizar inúmeras memórias auxiliares e recarregar periodicamente o componente.

Entre os fabricantes de FPGAs com SRAM encontram-se a Concurrent Logic, a Algotronix, a Xilinx e até mesmo a Altera, principal fabricante de EPLDs [18] [45] [56] [57]. Os componentes da Xilinx são os que dominam atualmente o mercado dos PLDs com células de memória SRAM e, por isso, sua arquitetura será apresentada neste ítem [58]. Os demais FPGAs são compostos dos mesmos blocos básicos, diferenciando-se, praticamente, apenas na disposição destes blocos dentro do circuito e nas suas características funcionais (figuras 3.16a e 3.16b).



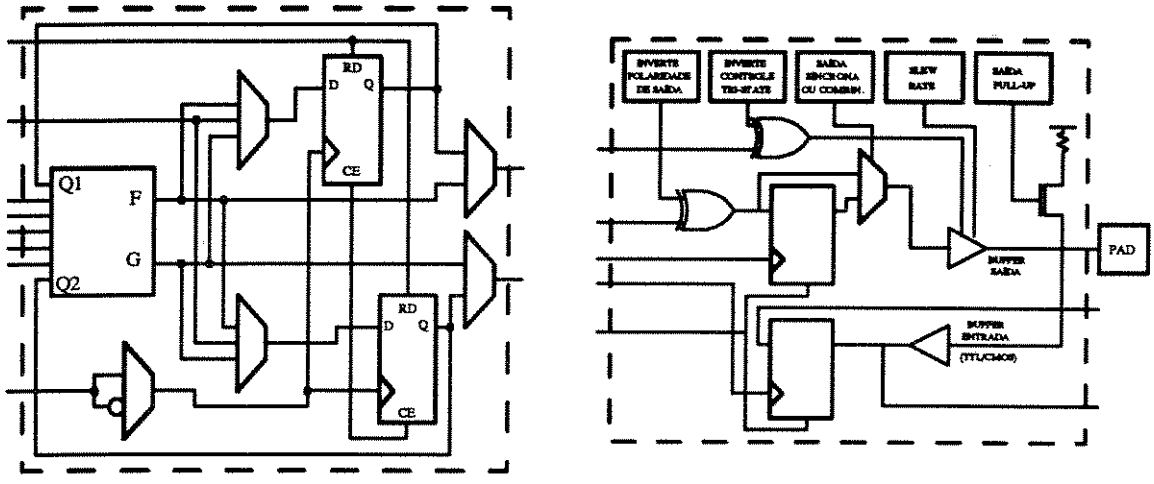
(a) (b)

Figura 3.16 - Arquitetura básica de FPGAs com SRAM:
(a) Xilinx; (b) Concurrent Logic.

A implementação das funções lógicas no componente é feita com células lógicas, ou **blocos lógicos configuráveis** (como são denominados pelo fabricante), posicionados em áreas estratégicas do circuito. Estes blocos apresentam uma parte combinacional programável, capaz de implementar qualquer função lógica com suas variáveis de entrada.

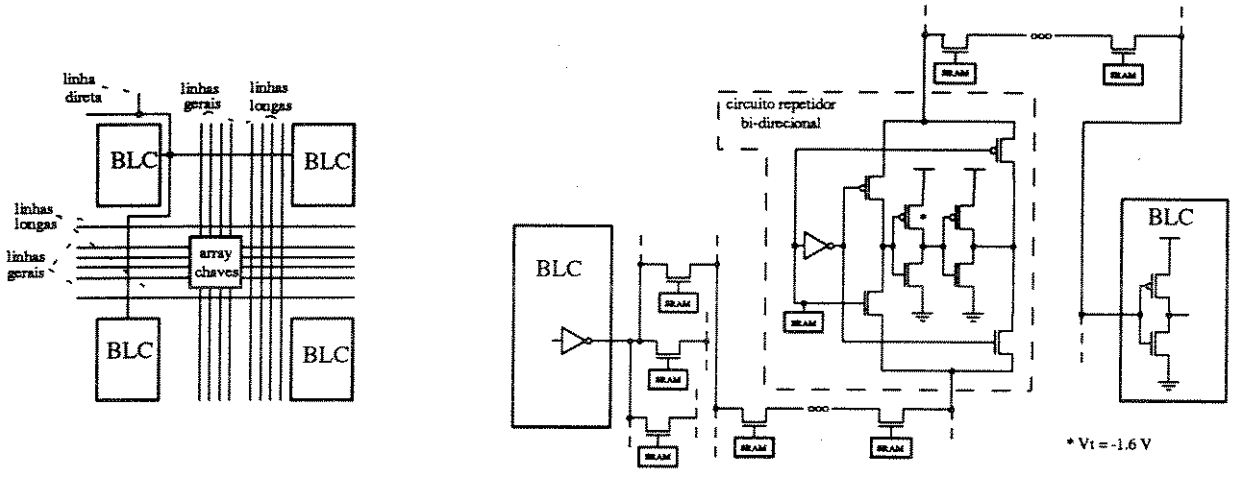
A configuração da parte combinacional é feita através de estruturas *look-up table*, utilizando células SRAM [31]. Nos blocos lógicos encontram-se também registradores e multiplexadores, com o objetivo de permitir a armazenagem de dados e a possibilidade de escolha de saída síncrona ou assíncrona de sinais (figura 3.17a). Estes blocos podem apresentar dois ou mais sinais de saída, diferentemente dos FPGAs com anti-fusíveis da Actel, que apresentam apenas um sinal de saída por bloco lógico (figura 3.14a).

Entre os blocos lógicos configuráveis encontram-se canais de roteamento, com trilhas de metal fixas, interligadas entre si e com os blocos lógicos por meio de transistores de passagem (figura 3.17a). Células SRAM definem o estado ON ou OFF dos transistores de passagem (chaves), formando o caminho percorrido pelos sinais. Estas chaves estão dispostas em forma de *arrays,* posicionadas estrategicamente nos cruzamentos de trilhas horizontais e verticais.



(a) (b)

Figura 3.17 - FPGA da Xilinx: (a) bloco lógico; (b) bloco de I/O.



(a) (b)

Figura 3.18 - FPGA da Xilinx: (a) detalhe do canal de roteamento; (b) representação de um roteamento com o uso de circuito "repetidor".

Uma deficiência encontrada nesta arquitetura é a degradação do nível de tensão dos sinais devido à utilização freqüente de muitos transistores de passagem por um mesmo sinal. Isto obriga este sinal, após percorrer determinadas distâncias, a passar por "repetidores" ou amplificadores a fim de reforçá-lo (figura 3.18b).

Porém, o uso de "repetidores" resulta no aumento dos atrasos de propagação deste sinal através do **canal de roteamento**.

Para interconexão de blocos lógicos vizinhos existem trilhas de metal cruzando os canais de roteamento e conectando diretamente estes blocos, sem utilizar os transistores de passagem. O mesmo ocorre com sinais roteados a longas distâncias, que podem utilizar trilhas de roteamento global, evitando a passagem por muitas chaves e a conseqüente degradação do seu nível de tensão (figura 3.18a).

Por fim, há os **blocos de I/O** configuráveis, posicionados na periferia do componente e utilizados para controle de entrada e saída de sinais, sejam eles síncronos ou assíncronos (figura 3.17b).

3.3.3 *Folded-arrays*

Os PLDs conhecidos por ***folded-arrays*** fazem uso de um *array* realimentado programável, formado apenas por portas lógicas AND ou OR para implementação das funções lógicas do circuito.

Apesar da semelhança do seu *array* programável com os *arrays* AND e OR dos PLDs da primeira geração, o princípio de implementação das funções lógicas é através de simples somas ou produtos, ao invés de soma-de-produtos. A álgebra booleana, da mesma forma que permite o equacionamento de qualquer função na forma de soma-de-produtos, possibilita a transformação de somas em produtos e vice-versa, de modo que qualquer equação lógica, resultante da soma-de-produtos, pode ser representada por simples somas ou produtos.

Os principais fabricantes de *folded-arrays* são a Exel Corp. e a Signetics Corp. A diferença entre eles está no uso de portas lógicas OR ou AND, no *array* programável (figuras 3.19a e 3.19b) [39] [46] [59]. Utiliza-se, atualmente, dispositivos PROM e EPROM para configurar estes componentes.

A arquitetura básica de um *folded-array* é formada apenas pelo *array* programável e por pinos de entrada, saída ou bidirecionais. Nos pontos de saída podem ser encontradas portas lógicas *tri-state*, para manter a saída em alta impedância, e portas para definição da polaridade do sinal de saída (portas lógicas OR exclusivo).

Existem componentes que possuem *flip-flops*, além do *array* programável, cujos sinais de saída podem ser enviados aos pinos de I/O ou realimentados para o *array* programável (figura 3.20). Desta forma, facilita-se a implementação de máquinas de estado e circuitos com lógica síncrona.

Um resumo das arquiteturas de CIs personalizáveis após o encapsulamento apresentadas pode ser visto na tabela 3.2 [19] [28] [30].

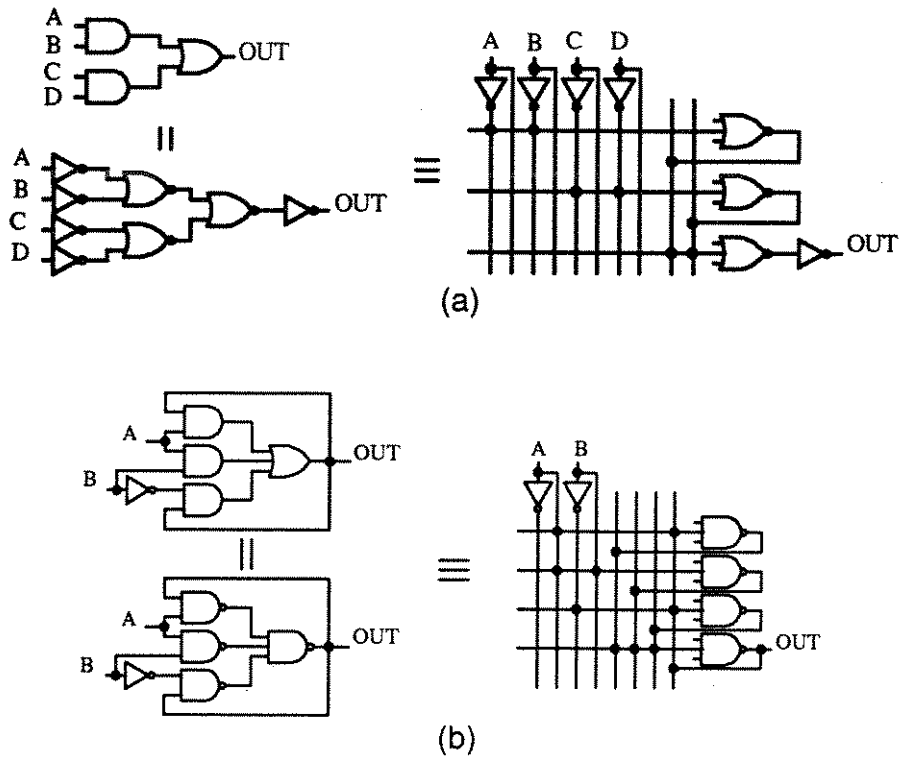


Figura 3.19 - Exemplo de implementação lógica utilizando-se apenas funções somas ou produtos: (a) Exel; (b) Signetics.

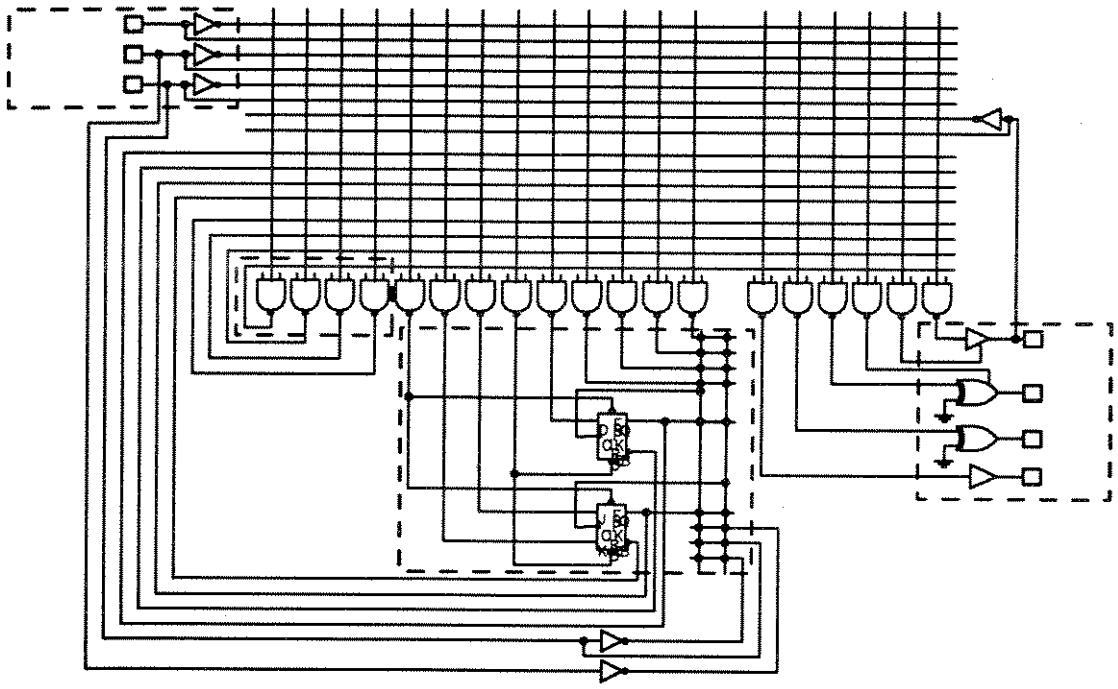


Figura 3.20 - Arquitetura básica de um *folded-array* da Signetics.

PLDs	arquitetura	elemento programação	alguns fabricantes
PROM	*array* AND fixo e *array* OR program.	fusível e EPROM	Motorola, National.
PLA	*arrays* AND e OR programáveis	fusível e EPROM	Signetics/Philips, Latice.
PAL	*array* AND program. e *array* OR fixo.	fusível, EPROM e E^2PROM	AMD/MMI, Signetics/Philips, Intel.
EPLD	macrocélula (PAL), *array* interconexão, expansor de produtos, blocos de I/O.	EPROM e E^2PROM	Altera, Plus Logic, AMD/MMI, Xilinx.
FPGA c/ antifusível	módulos lógicos, linhas de roteamento, blocos de I/O.	anti-fusível	Actel, Crosspoint, QuickLogic.
FPGA c/ SRAM	blocos lógicos, canais de roteamento, blocos de I/O, repetidores.	células SRAM	Xilinx, Concurrent Logic, Algotronix, Altera.
folded-array	*array* AND ou OR.	fusível e EPROM	Signetics/Philips, Exel.

Tabela 3.2 - Quadro comparativo de PLDs.

3.4 Taxonomia dos PLDs

Foi apresentado ao longo deste capítulo a variedade de nomenclaturas utilizadas pelos fabricantes e pela literatura para referenciar as componentes de lógica programável, ou CIs personalizáveis após o encapsulamento.

Para esta forma de implementação não existe ainda uma taxonomia definida, como ocorre com os CIs personalizáveis por máscaras [12] [13]. Os PLDs da primeira geração, devido ao seu tempo de existência, apresentam uma classificação, segundo a possibilidade de programação dos *arrays* AND e OR [38] [60], bastante utilizada pelos fabricantes e usuários, embora alguns insistem em utilizar o termo genérico PLD para citar os componentes PALs.

A segunda geração, por sua vez, tem uma história bastante recente e o lançamento de novos componentes, com arquiteturas e princípios de funcionamento inéditos, é ainda muito frequente. Isto dificulta encontrar uma classificação coerente onde possam ser encaixados os PLDs que estão para serem lançados no mercado.

Alguns autores simplificam esta discussão generalizando toda esta segunda geração como componentes FPGAs [19] [30]. Outros tentam encontrar critérios como a complexidade das células lógicas configuráveis, as características dos elementos de programação utilizados, ou mesmo os princípios de síntese e

mapeamento lógico adotados na utilização das CLs [29] [83]. Mas estas tentativas não têm sido adotadas pelos fabricantes que continuam fazendo uso de nomenclaturas próprias e confundindo o usuário no momento da escolha do componente mais adequado a sua aplicação.

Observa-se uma tendência de ser utilizada a sigla FPGA para aqueles PLDs que utilizam como elementos de programação células de memória SRAM ou anti-fusíveis, e a sigla EPLD para os componentes com dispositivos EPROM e E^2PROM. Outra divisão possível é segundo as características das CLs utilizadas. A primeira vista parece existir duas linhas para a compilação da lógica dos circuitos: através de equações booleanas na forma de soma-de-produtos, simples somas ou simples produtos, e através de células lógicas genéricas formadas por portas lógicas universais, multiplexadores ou estruturas *look-up table.*

Entretanto, o mais importante para o usuário, independente da taxonomia adotada por ele mesmo, é que no momento da adoção de determinado componente para implementação de seu ASIC sejam consideradas características básicas como a volatilidade e a reprogramabilidade dos elementos de programação, e as características dos canais de roteamento, que poderão permitir ao projetista a realização de conexões manuais e interferir fortemente na análise de *timing* do circuito.

Capítulo 4

Capacidade lógica dos PLDs

Até o início dos anos 80, as abordagens *gate array* e *standard cell* [11] constituiam as formas de implementação de ASICs mais usuais, úteis para prototipagem de CIs ou mesmo para produção em quantidades até 10K a 20K peças por ano.

Os *gate arrays* atuais chegam a atingir capacidades lógicas na faixa de 200K portas equivalentes [21], existindo, entretanto, famílias que cobrem a faixa de 2K a 20K portas de maneira eficiente em termos de desempenho e custo [10]. Enquanto que os PLDs da primeira geração (PALs, PLAs e PROMs) apresentam uma capacidade lógica estimada de até 1.000 portas equivalentes.

Com o surgimento da segunda geração de PLDs, em meados da década de 80, e a sua rápida evolução em capacidade lógica e desempenho, criou-se dúvidas para o usuário quanto a escolha da melhor alternativa para implementação de ASICs, principalmente pela diversidade de arquiteturas e elementos de programação (fusíveis, anti-fusíveis, células SRAM, EPROM e E^2PROM) [28].

Os primeiros PLDs da segunda geração já possibilitavam a implementação de ASICs com até 2.000 portas equivalentes (família Clássica da Altera e família XC2000 da Xilinx). A partir disso, o aumento da capacidade lógica dos componentes ocorreu quase "instantaneamente" (tabela 4.1) [18] [45] [61].

Componente	Fabricante	Capacidade lógica (portas equivalentes)
TCP1280	Actel / Texas Inst.	8.000
XC4013	Xilinx	13.000
EPF81500	Altera	15.500

Tabela 4.1 - Capacidade lógica de alguns PLDs.

O lançamento de PLDs cada vez maiores e mais sofisticados é decorrência não apenas da evolução tecnológica dos processos de fabricação e

das arquiteturas programáveis, mas principalmente da acirrada concorrência entre os fabricantes na conquista do mercado de ASICs [62].

4.1 Contagem em portas equivalentes

A medida da complexidade de circuitos digitais, expressa em portas equivalentes, tornou-se muito utilizada devido à possibilidade de se estimar o tamanho de placas de circuito impresso (PCB), de pastilhas de CIs *standard cell* ou mesmo a capacidade lógica de matrizes de transistores pré-difundidos (*gate arrays*).

Por definição, **1 porta equivalente** corresponde a implementação física de uma porta lógica NAND de duas entradas (nand2), ou seja, a porta nand2 é a referência para as demais funções lógicas, que são mensuradas em função desta segundo o espaço físico utilizado para a sua implementação sobre uma pastilha semicondutora. Por exemplo, um inversor corresponde a aproximadamente 1 porta, uma nor4 a 2 portas e um *flip-flop* tipo D a cerca de 6 portas equivalentes [63].

Quando um sistema digital é implementado sobre uma placa impressa com o uso de componentes *standard*, a contagem das portas equivalentes da descrição esquemática permite estimar a quantidade de componentes na placa e, conseqüentemente, as dimensões da mesma.

No caso do projeto de ASICs com formas de implementação personalizáveis por máscaras e que utilizam biblioteca de células (*gate arrays* e *standard cells*), esta estimativa é ainda melhor pois cada símbolo de primitivas existentes no esquemático possui um leiaute correspondente, que será efetivamente fabricado sobre a lâmina. Desta forma, é possível conhecer o tamanho do circuito integrado resultante em *standard cell*, ou se o circuito descrito poderá ser implementado sobre uma determinada matriz *gate array*, de capacidade lógica bem conhecida.

Esta análise para *standard cells* e *gate arrays* é válida pois estas formas de implementação de ASICs dificilmente realizam manipulações lógicas (minimizações ou otimizações) sobre o esquemático descrito. Normalmente, tudo o que é posto no esquemático será implementado com o leiaute correspondente. Porém, para os PLDs isto não ocorre, pois as ferramentas de CAD para configurar estes componentes não relacionam símbolos lógicos com leiautes, mas com as funções lógicas (equações booleanas) que deverão ser programadas dentro deles.

Logo, esta forma de definição de capacidade lógica não se apresenta tão precisa e adequada como no caso das matrizes *gate arrays*. Acredita-se que, devido à realização de manipulações lógicas pelos *softwares* de programação

dos PLDs, é difícil saber, a priori, qual a complexidade resultante de um circuito após a sua compilação num componente PLD, em relação ao seu diagrama esquemático.

O que se faz no caso dos PLDs, independentemente do critério utilizado para o cálculo da sua capacidade lógica máxima, é tomar a taxa de utilização do circuito digital implementado, sobre um determinado componente, e através do produto **capacidade x taxa de utilização** definir, a princípio, a complexidade resultante deste circuito após toda a manipulação lógica necessária na etapa de mapeamento [30].

Outras características que dificultam a obtenção da estimativa em portas equivalentes dos PLDs são:

a) **A utilização irregular das células lógicas (CLs)**: as células lógicas existentes nas arquiteturas (blocos lógicos configuráveis - Xilinx, macrocélulas - Altera, módulos lógicos - Actel ou qualquer outra denominação dada pelos fabricantes) podem ser utilizadas no limite da sua capacidade lógica, ou apenas para implementar equações booleanas muito simples, desperdiçando o restante da capacidade da CL. É muito difícil manter uma regularidade no fator de utilização destes blocos.

b) **Algumas portas lógicas básicas podem utilizar uma quantidade considerável da capacidade de uma CL**: uma porta OR exclusiva de 4 entradas, por exemplo, que corresponde a uma célula básica em bibliotecas *standard cell* e *gate arrays* (cerca de 2 a 3 portas equivalentes), utiliza toda uma macrocélula quando implementada nos componentes da Altera, devido a sua decomposição na forma de soma-de-produtos.

c) **O uso limitado de *flip-flops* e células *tri-state* devido à arquitetura fixa dos PLDs**: além de considerar a capacidade em portas equivalentes, deve-se observar também o número de *flip-flops* e células *tri-state* exigidos pelo ASIC, pois os componentes PLDs apresentam um número limitado destas estruturas em sua arquitetura.

Por mais eficiente que seja o compilador, este dificilmente conseguirá utilizar toda a lógica disponível em uma CL. Os fabricantes necessitam de um levantamento estatístico sobre o uso dos blocos configuráveis para estimar precisamente a capacidade de implementação lógica dos PLDs.

Além disso, conforme apresentado no capítulo anterior, cada fabricante fornece características bem distintas as suas CLs: a Actel utiliza multiplexadores, a Altera implementa soma-de-produtos, e assim por diante. Portanto, a eficiente utilização dos blocos lógicos configuráveis depende dos tipos de funções implementadas. Isto dificulta ainda mais a previsão do uso de

determinado componente apenas conhecendo-se a descrição esquemática do circuito digital.

A par destas considerações, foram feitos estudos para tentar relacionar o critério de cálculo de complexidade dos PLDs com as abordagens *standard cell* e *gate array*. Nas tabelas 4.2 e 4.3 são apresentados dados sobre os componentes da Altera e da Xilinx [18] [45], com ênfase na capacidade lógica máxima fornecida pelo respectivo fabricante.

Família	**Componentes**	**Células Lógicas* / nº *flip-flops***	**Capacidade ** (portas equiv.)**
Clássica	EP610	16 / 16	0,6K
	EP910	24 / 24	0,9K
	EP1810	48 / 48	1,8K
Max5000	EPM5016	16 / 16	0,6K
	EPM5032	32 / 32	1,2K
	EPM5064	64 / 64	2,5K
	EPM5128	128 / 128	5,0K
	EPM5130	128 / 128	5,0K
	EPM5192	192 / 192	7,5K
Max7000	EPM7032	32 / 32	1,2K
	EPM7064	64 / 64	2,5K
	EPM7096	96 / 96	3,6K
	EPM7128	128 / 128	5,0K
	EPM7160	160 / 160	6,4K
	EPM7192	192 / 192	7,5K
	EPM7256	256 / 256	10,0K
Flex8000	EPF8282	208 / 282	5,0K
	EPF8452	336 / 452	8,0K
	EPF8820	672 / 820	16,0K
	EPF81188	1.008 / 1.188	24,0K
	EPF81500	1.296 / 1.500	31,0K

(*) As células lógicas (CLs) são chamadas de Macrocélulas nas famílias Clássica, Max5000 e Max7000, e de Elementos Lógicos na família FLEX8000.
(**) A quantidade de portas equivalentes utilizáveis é de 50%.

Tabela 4.2 - Características de PLDs da Altera Corp.

Inicialmente, utilizou-se circuitos funcionais já existentes [64] [65], e numa segunda fase elaborou-se circuitos *benchmarks* específicos para este estudo comparativo.

Observa-se contudo que a restrição da quantidade de portas lógicas de armazenagem de dados (*flip-flops* e *latches*) e de saídas *tri-state* (*buffers* ou

inversores *tri-state*) impedem que a simples estimativa em portas equivalentes defina o componente PLD a ser utilizado.

Família	Compo-nentes	CLs*	Células *tri-state*	*Flip-flops*	Capacidade (portas equiv.)
XC2000	XC2064	64	0	122	0,8K - 1,0K
	XC2018	100	0	174	1,2K - 1,5K
XC3000	XC3020	64	16	256	1,3K - 1,8K
	XC3030	100	20	360	2,0K - 2,7K
	XC3042	144	24	480	2,5K - 3,7K
	XC3064	224	32	688	4,0K - 5,5K
	XC3090	320	40	928	5,0K - 7,5K
XC4000	XC4002	64	16	256	2,0K
	XC4003	100	20	360	3,0K
	XC4004	144	24	480	4,0K
	XC4005	196	28	616	5,0K
	XC4006	256	32	768	6,0K
	XC4008	324	36	936	8,0K
	XC4010	400	40	1.120	10,0K
	XC4013	576	48	1.536	13,0K
	XC4016**	676	-	1.768	16,0K
	XC4020**	900	-	2.280	20,0K

(*) As células lógicas (CLs) são chamadas de Blocos Lógicos Configuráveis (CLBs) pela Xilinx.
(**) Estes componentes ainda não foram lançados.

Tabela 4.3 - Características de PLDs da Xilinx, Inc.

4.2 Análise inicial: circuitos funcionais

A estratégia inicial deste estudo foi aproveitar circuitos já descritos, de complexidade lógica bem conhecida, e implementá-los em alguns PLDs (vide tabela 4.4). Foram utilizados os três circuitos funcionais que auxiliaram na caracterização e avaliação da matriz GA2500 [64]:

- **MUL_4x4** - multiplicador de 4 x 4 bits em arquitetura *pipeline*: realiza a multiplicação binária em quatro estágios de processamento (quatro ciclos de relógio), sendo portanto um circuito síncrono onde, a cada período de *clock* (CK), pode-se enviar um novo dado ao primeiro estágio da multiplicação, enquanto que os demais estágios processam outras multiplicações;

- **ULA_16** - unidade lógica e aritmética de 16 bits: realiza 6 operações básicas com dois vetores de 16 bits (soma, subtração, AND, OR, OR exclusivo e complemento) e apresenta células de armazenamento (*latches* ou *flip-flops*)

somente para sincronismo dos sinais de entrada, sendo portanto um circuito puramente combinacional;

- **Ouvidor** - circuito "observador" de barramentos que auxilia na testabilidade de sistemas digitais através da implementação de equações polinomiais: pode-se dizer que este é um circuito misto, com parte síncrona e combinacional.

O quarto circuito fucional utilizado nesta fase inicial provém de um projeto realizado na metodologia *standard cell* [65]:

- **QUIM** - parte digital de uma interface para um sistema de aquisição de dados de sensores químicos: a troca de dados do circuito com o sistema de controle é bidirecional, ou seja, há um barramento através do qual o sistema pode enviar o endereço do sensor requerido e receber seus dados; este circuito é composto basicamente de registradores.

Circuito	Complexidade (portas equiv.)	% lógica seqüencial	% lógica combinacional	*Flip-flops*	Células *tri-state*
QUIM	265	80,2	19,8	47	8
MUL_4x4	439	63,5	36,5	41	-
ULA_16	654	19,5	80,5	32	-
Ouvidor	1457	25,0	75,0	65	32

Tabela 4.4 - Circuitos funcionais para análise inicial.

Os resultados da compilação destes circuitos dentro dos componentes da Altera e da Xilinx são apresentados nas tabelas 4.5 e 4.6. A complexidade por portas equivalentes resultante dos circuitos, em cada uma das famílias sobre as quais eles foram implementados, foi obtida realizando-se o produto da capacidade do PLD pela taxa de ocupação das CLs (macrocélulas - Altera e CLBs - Xilinx).

A capacidade tomada neste cálculo para os componentes da Xilinx corresponde ao menor valor das faixas apresentadas na tabela 4.3, e para os componentes da Altera já foram considerados os 50% de utilização da lógica das macrocélulas, sugerido pelo fabricante (tabela 4.2).

Observa-se, primeiramente que, para ambos os fabricantes, **a utilização da capacidade lógica máxima dos PLDs** empregados não correponde a complexidade dos circuitos definida no ambiente GA2500 e *standard cell* (tabela 4.4), não se verificando nem mesmo uma proporção entre ambas.

Circuitos	Clássica	Max5000	Max7000
QUIM	EP ? - 1050 pe (56 CLs,47 ff, 8 tsta)	EPM5128 - 1075 pe (55 CLs,47 ff, 8 tsta)	EPM7096 - 1062 pe (57 CLs,47 ff, 8 tsta)
MUL_4x4	EP ? - 1013 pe (54 CLs, 41 ff)	EPM5064 - 913 pe (46 CLs, 41 ff)	EPM7096 - 936 pe (50 CLs, 41 ff)
ULA_16	EP ? - 2982 pe (159 CLs, 32 ff)	EPM5192 - 3113 pe (160 CLs, 32 ff)	EPM7160 - 2944 pe (147 CLs, 32 ff)
Ouvidor	-	EPM5130 - 1875 pe (96 CLs, 65 ff, 32 tsta)	EPM7128 - 1900 pe (97 CLs, 65 ff, 32 tsta)

Obs.: Abreviações - pe = portas equivalentes; CLs = células lógicas; ff = flip-flops; tsta = células tri-state.

Tabela 4.5 - Compilação dos circuitos funcionais nos PLDs da Altera.

Circuitos	XC2000	XC3000	XC4000
QUIM	-	XC3020 - 792 pe (39 CLs,47 ff, 8 tsta)	XC4002 - 719 pe (23 CLs,47 ff, 8 tsta)
MUL_4x4	XC2064 - 538 pe (43 CLs, 41 ff)	XC3020 - 833 pe (41 CLs, 41 ff)	XC4002 - 625 pe (20 CLs, 41 ff)
ULA_16	XC2018 - 960 pe (80 CLs, 32 ff)	XC3030 - 1.440 pe (72 CLs, 32 ff)	XC4002 - 1.250 pe (40 CLs, 32 ff)
Ouvidor	-	XC3064 - 2.375 pe (133 CLs, 65 ff, 32 tsta)	XC4002 - 1.688 pe (54 CLs, 65 ff, 32 tsta)

Obs.: Abreviações - pe = portas equivalentes; CLs = células lógicas; ff = flip-flops; tsta = células tri-state.

Tabela 4.6 - Compilação dos circuitos funcionais nos PLDs da Xilinx.

A título de exemplo, para os circuitos QUIM e MUL_4x4 não foi possível utilizar o componente EPM7064 que possui capacidade lógica utilizável estimada de 1.250 portas, embora este valor fosse numericamente suficiente. Também, os circuitos ULA_16 e Ouvidor utilizaram mais do que 50% dos CLs disponíveis nos componentes Xilinx XC3030 e XC3064, respectivamente, o que corresponde a complexidades estimadas superiores as da tabela 4.4.

Notou-se que o **aproveitamento dos PLDs** ocorre de acordo com as características funcionais dos circuitos (síncrono, assíncrono, combinacional) e com o tipo de portas lógicas utilizadas nos diagramas esquemáticos, tais como funções multiplexadoras baseadas em lógica combinacional ou *tri-state*. Como exemplo, o circuito ULA_16, que possui uma complexidade bem menor do que o Ouvidor, teve que utilizar componentes maiores quando compilado no ambiente da Altera. No caso dos circuitos MUL_4x4 e QUIM, de complexidade distinta, tiveram uma utilização bastante semelhante de CLs em ambos ambientes.

Deve-se chamar a atenção também quanto às **limitações do número de células *tri-state* e *flip-flops*** nestes componentes: os circuitos QUIM e Ouvidor não puderam ser compilados na família XC2000 (Xilinx) por esta não possibilitar o uso de células *tri-state*; a mesma limitação fez com que o Ouvidor não pudesse ser compilado para a família Clássica da Altera e tivesse que ser implementado no componente XC3064 da Xilinx, embora o XC3042 apresentasse um número suficiente de CLs para este circuito.

4.3 Análise complementar: *benchmarks* específicos

O uso de apenas quatros circuitos funcionais não permite uma amostragem significativa para esta comparação de capacidade lógica entre os CIs personalizáveis por máscaras (*gate arrays* e *standard cells*) e os PLDs. O fato também de serem circuitos funcionais em média pequenos e que não cobrem certas aplicações, como máquinas de estados, pode levar a resultados tendenciosos. Por isso, há necessidade de desenvolver *benchmarks* específicos para avaliar melhor a capacidade lógica dos PLDs, em relação a forma tradicional de portas equivalentes.

O conceito de circuito *benchmark*, neste estudo, se refere aos circuitos utilizados para avaliar e comparar tecnologias, arquiteturas ou formas de implementação de ASICs quanto ao desempenho elétrico, capacidade lógica, ou mesmo, custos de projeto e fabricação. Estes circuitos devem ser de aplicabilidade genérica e de média complexidade, para que os resultados destes estudos comparativos sejam os mais realistas possíveis.

Definir ou encontrar o circuito *benchmark* ideal não é fácil. A própria definição de aplicabilidade genérica não é muito clara. A priori, este circuito deve ter uma parte operativa, uma parte de controle, uma de armazenagem de dados (memória), quantidade semelhante de sinais de entrada, saída e bidirecional, entre outras características encontradas na maioria dos circuitos digitais.

Além de todas estas características de arquitetura, a complexidade ou o tamanho ideal do *banchmark* também é difícil de ser definido. Detalhes da topologia do circuito e da sua complexidade podem fazer grande diferença na análise comparativa realizada sobre determinado componente ou tecnologia.

O surgimento da segunda geração de PLDs, com suas variadas arquiteturas e elementos de programação, criou muitas dúvidas sobre a aplicabilidade destes componentes. Conforme foi apresentado no capítulo anterior, há características como volatilidade e reprogramabilidade associadas aos elementos de programação que podem decidir sobre o uso de um componente em determinado sistema digital. Além disso, arquiteturas com

atrasos de sinais internos fixos ou dependentes do seu roteamento também podem inteferir na escolha do componente para a implementação do ASIC.

Mas o ponto mais obscuro para o usuário é como comparar arquiteturas e tecnologias bastante distintas em desempenho elétrico e capacidade lógica. Considerando ainda que esta comparação pode apresentar resultados variados de acordo com a topologia do circuito.

Visando esclarecer estas dúvidas e indecisões, os fabricantes de PLDs criaram uma entidade ou, pode-se dizer, um grupo consultor formado por especialistas das empresas interessadas na área, com o objetivo de desenvolver circuitos *benchmarks* e utilizá-los para comparar os componentes programáveis disponíveis comercialmente.

Esta entidade, ou **PREP** (*Programmable Electronics Performance Corporation*) como é chamada, utiliza um conceito de *benchmark* um pouco diferente daquele comumente aplicado. Ao invés de desenvolver ou encontrar o *benchmark* ideal (complexidade média e de aplicação genérica), a filosofia do PREP é utilizar circuitos específicos, tanto combinacionais quanto síncronos ou mistos, como contadores, máquinas de estado, acumuladores, unidades aritméticas ou mesmo *data paths* [66] [67].

Acredita-se que desta forma a avaliação dos PLDs é mais realista, visto que desta forma é possível conhecer a eficiência da utilização dos PLDs, segundo algumas topologias de circuito de interesse, que expressam características operacionais típicas às existentes em aplicações reais.

Na avaliação da capacidade, toma-se um dos circuitos *benchmarks* (contador *up*, por exemplo) e o repete tantas vezes quantas for possível dentro do componente, de maneira serial (em cascata). Os *benchmarks* possuem uma complexidade baixa, entre 100 e 300 portas equivalentes. A capacidade lógica do PLD é medida pela contagem do número máximo de repetições possíveis do circuito. Comparações entre diferentes componentes podem ser feitas pela média dos resultados de implementação (repetições) de todos os *benchmarks*.

Para avaliação do desempenho elétrico, o PREP utiliza dois conceitos de freqüência:

- **freqüência interna e**
- **freqüência externa.**

A **freqüência interna** se refere a velocidade de operação do circuito dentro do componente. Ela é determinada tomando-se o inverso da medida do atraso de propagação do caminho crítico entre a entrada de um *benchmark* até a entrada do *benchmark* seguinte implementado dentro do mesmo PLD.

A **freqüência externa**, por sua vez, corresponde a velocidade de comunicação entre dois componentes. Para a medida da freqüência externa o procedimento é semelhante ao apresentado acima. Porém, toma-se o caminho crítico entre a entrada do último *benchmark,* repetido em um componente, e a entrada do primeiro *benchmark* posto em outro PLD. Logo, neste caminho está incluído o atraso de propagação de sinais de dois *pads* de I/O, devido a saída e entrada do sinal nos componentes. Este atraso adicional dos *pads* não era considerado na medida da freqüência interna.

Porém, a comparação realizada pelo PREP não questiona a medida de complexidade dos PLDs através da contagem de portas equivalentes, tal como apresentado nas tabelas 4.5 e 4.6, apesar dos fabricantes fornecerem através desta contagem o tamanho dos seus componentes, e também não compara a sua eficiência com as demais formas de implementação de ASICs (CIs personalizáveis por máscaras).

Benchmarks	**Complexidade * (portas equiv.- pe)**	**No. de repetições**	**Circuito resultante (complexidade)**
data path	160 pe (56% / 44%)	15	bench1 (2.400 pe)
timer	283 pe (47% / 53%)	8	bench2 (2.266 pe)
máquina de estado (8 bits)	94 pe (18% / 82%)	26	bench3 (2.444 pe)
máquina de estado (16 bits)	205 pe (11% / 89%)	12	bench4 (2.462 pe)
circuito aritmético	246 pe (18% / 82%)	10	bench5 (2.464 pe)
acumulador de 16 bits	90 pe (100% / 0%)	27	bench6 (2.419 pe)
contador *up*	237 pe (38% / 62%)	10	bench7 (2.372 pe)
contador *down*	225 pe (40% / 60%)	11	bench8 (2.477 pe)

(*) Os valores percentuais correspondem às porcentagens de lógica sequencial e combinacional do circuito, respectivamente.

Tabela 4.7 - *Benchmarks* específicos para comparação de capacidade lógica.

A estratégia adotada nesta segunda etapa de estudos foi utilizar o ambiente GA2500 como referência para definir os circuitos a serem implementados nos PLDs [15]. Desenvolveu-se 8 *benchmarks*, segundo as especificações do PREP, e os implementou na matriz GA2500, pelo princípio de

repetição, para então definir os circuitos a serem compilados nos PLDs (tabela 4.7). A descrição esquemática destes *benchmarks* estão no anexo I.

É importante enfatizar que não está sendo avaliada a capacidade de roteamento de cada ambiente. A definição dos circuitos dentro do ambiente GA2500 foi feita pelo maior número de repetições dos *benchmarks* permitido na etapa de particionamento. O mesmo procedimento foi adotado durante a compilação dentro dos ambientes de PLDs.

A avaliação da eficiência de roteamento de cada ambiente é um trabalho complexo e que depende muito do grau de especialização do projetista em relação as ferramentas de CAD utilizadas.

Os resultados das implementações dos circuitos bench1 a bench8 (circuitos resultantes da repetição dos *benchmarks* sobre a matriz GA2500) podem ser vistos nas tabelas 4.8 e 4.9. Nestas tabelas são apresentadas as quantidades de células lógicas e *flip-flops* utilizados em cada componentes.

A complexidade resultante mostrada corresponde ao produto entre a taxa de ocupação do componente (número de CLs) e a sua capacidade lógica, fornecida pelo fabricante. A descrição do circuito bench8 sofreu alguns contratempos e este não pode ser compilado nas famílias Flex8000 e XC4000.

Circuitos	Max5000	Max7000	Flex8000
bench1	EPM ? - 4.688 pe (240 CLs, 240 ff)	EPM7256 - 4.688 pe (240 CLs, 240 ff)	EPF8820 - 4.381 pe (368 CLs, 240ff)
bench2	EPM ? - 4.121 pe (211 CLs, 192 ff)	EPM ? - 5.391 pe (276 CLs, 192 ff)	EPF8820 - 4.095 pe (344 CLs, 192 ff)
bench3	EPM ? - 5.586 pe (286 CLs, 78 ff)	EPM ? - 5.586 pe (286 CLs, 78 ff)	EPF81188 - 9.595 pe (806 CLs, 78 ff)
bench4	EPM ? - 4.082 pe (209 CLs, 48 ff)	EPM ? - 5.078 pe (260 CLs, 48 ff)	EPF81188 - 9.595 pe (806 CLs, 48 ff)
bench5	EPM ? - 14.648 pe (750 CLs, 80 ff)	EPM ? - 15.684 pe (803 CLs, 80 ff)	EPF8820 - 6.667 pe (560 CLs, 80 ff)
bench6	EPM ? - 8.438 pe (432 CLs, 432 ff)	EPM ? - 8.438 pe (432 CLs, 432 ff)	EPF8820 - 5.143 pe (432 CLs, 432 ff)
bench7	EPM5192 - 3.125 pe (160 CLs, 160 ff)	EPM7192 - 3.223 pe (165 CLs, 160 ff)	EPF8820 - 7.024 pe (590 CLs, 160 ff)
bench8	EPM5192 - 3.438 pe (176 CLs, 176 ff)	EPM7192 - 3.555 pe (182 CLs, 176 ff)	*não disponível*

Obs.: O critério para contagem do número de portas equivalentes é o mesmo utilizado na tabela 4.5.

Tabela 4.8 - Resultados dos *benchmarks* nos PLDs da Altera.

O primeiro ponto a ser observado na análise destes resultados é que, do mesmo modo como ocorre para os circuitos funcionais apresentados, as complexidades resultantes dos circuitos *benchmarks* implementados não correspondem ao número de portas equivalentes obtidas no ambiente GA2500 (em torno de 2.500 portas). As complexidades resultantes destas implementações são as mais variadas possíveis.

Circuitos	XC2000	XC3000	XC4000
bench1	XC ? - 4.416 pe (368 CLs, 240 ff)	XC3090 - 3.969 pe (254 CLs, 240 ff)	XC4005 - 4.592 pe (180 CLs, 240 ff)
bench2	XC ? - 3.120 pe (260 CLs, 192 ff)	XC3090 - 3.672 pe (235 CLs, 192 ff)	XC4005 - 4.184 pe (164 CLs, 192 ff)
bench3	XC ? - 7.176 pe (598 CLs, 78 ff)	XC ? - 7.938 pe (508 CLs, 78 ff)	XC4008 - 8.000 pe (324 CLs, 78 ff)
bench4	XC ? - 9.000 pe (750 CLs, 48 ff)	XC ? - 7.672 pe (491 CLs, 48 ff)	XC4010 - 9.275 pe (371 CLs, 48 ff)
bench5	XC ? - 4.716 pe (393 CLs, 80 ff)	XC ? - 5.328 pe (341 CLs, 80 ff)	XC4008 - 6.617 pe (268 CLs, 80 ff)
bench6	XC ? - 5.184 pe (432 CLs, 432 ff)	XC3064 - 3.911 pe (219 CLs, 432 ff)	XC4006 - 5.063 pe (216 CLs, 432 ff)
bench7	XC ? - 3.600 pe (300 CLs, 160 ff)	XC3090 - 4.688 pe (300 CLs, 160 ff)	XC4006 - 5.039 pe (215 CLs, 160 ff)
bench8	XC ? - 3.960 pe (330 CLs, 176 ff)	XC ? - 5.141 pe (329 CLs, 176 ff)	*não disponível*

Obs.: O critério para contagem do número de portas equivalentes é o mesmo utilizado na tabela 4.6.

Tabela 4.9 - Resultados dos *benchmarks* nos PLDs da Xilinx.

Os circuitos com melhor aproveitamento das CLs nos EPLDs da Altera (famílias Max5000 e Max7000) e nos FPGAs da Xilinx (XC2000, XC3000 e XC4000), ou seja, que apresentaram o menor valor de portas equivalentes nestas implementações foram os *benchmarks data path*, *timer*, contadores *up* e *down* (circuitos bench1, bench2, bench7 e bench8, respectivamente).

Estes circuitos possuem percentuais semelhantes de lógica seqüencial e combinacional em suas arquiteturas. Porém, a tendência de que os circuitos mais equilibrados em relação a quantidade de lógica seqüencial e combinacional utilizam menor número de CLs não se confirma com os FPGAs da Altera (família Flex8000).

No caso dos circuitos bench3, bench4 e bench5 (máquinas de estado de 8 e 16 bits, e circuito aritmético, respectivamente), onde mais de 80% da sua lógica é combinacional, não se verifica tendência alguma.

As máquinas de estado apresentam melhor aproveitamento dos EPLDs (famílias Max5000 e Max7000) do que dos FPGAs (Flex8000) da Altera, sendo

inclusive os *benchmarks* que utilizaram os maiores componentes da família Flex8000 (EPF81188) dentre aqueles apresentados na tabela 4.8. No ambiente da Xilinx, estes dois circuitos foram os que utilizaram maior número de CLs.

O circuito aritmético (bench5), por sua vez, apresentou um aproveitamento médio de células lógicas na família Flex8000 e nos FPGAs da Xilinx, em relação aos demais *benchmarks*, mas teve uma utilização de CLs muito maior nas famílias Max5000 e Max7000.

O circuito bench6 (acumulador de 16 bits), formado exclusivamente por células de armazemento, ou seja, 100% da sua arquitetura corresponde a lógica seqüencial, praticamente avaliou a disponibilidade de *flip-flops* nas CLs de cada família de componentes.

Observe que estes circuitos *benchmarks*, a priori, possíveis de serem implementados sobre uma matriz *gate array* com capacidade lógica de 2.500 portas equivalentes, só não apresentaram dificuldades de implementação, pela falta de lógica disponível nos PLDs, nas maiores famílias de cada fabricante (Flex8000 e XC4000). Nas demais, os poucos circuitos que puderam ser implementados fizeram uso dos maiores componentes destas famílias.

4.4 Considerações finais

Neste capítulo foi proposta uma metodologia de análise da capacidade lógica dos PLDs, em relação a contagem de portas equivalentes utilizadas em *gate arrays*. Alguns projetos de circuitos digitais, já realizados segundo a metodologia *gate array*, para uma matriz de 2.500 portas equivalentes, foram implementados em EPLDs da Altera e FPGAs da Xilinx, tendo sido comparada a utilização dos componentes PLDs e a complexidade aparentes destes circuitos.

Os resultados preliminares indicam uma surpreendente discrepância entre a contagem de portas lógicas nas metodologias de projeto *gate array* e PLD, inclusive, não havendo uma proporção entre estes valores.

Circuitos digitais de cerca de 2.500 portas em *gate array* podem atingirem no ambiente Altera contagens estimadas na faixa de 3.125 a 15.684 portas equivalentes, correspondendo a um aumento de 25% a 527%. No ambiente Xilinx, obteve-se contagens estimadas na faixa de 3.120 a 9.275, ou seja, 24% a 271%.

Além destas faixas de variação das contagens, há alguns pontos referentes a capacidade dos PLDs que devem ser observados e são discutidos a seguir.

4.4.1 Arquivos *reports*

Há algumas inconsistências dos arquivos de reportagem da compilação (*reports*) em relação aos tamanhos dos componentes fornecidos nos *data books* dos fabricantes Altera e Xilinx.

Para a família XC4000 da Xilinx este arquivo de reportagem fornece uma estimativa da complexidade do circuito implementado e ele mesmo afirma que tal circuito não pode ser colocado em componentes que, segundo a própria documentação do fabricante, teriam capacidade lógica suficiente (figura 4.1).

Bench4

Preliminary evaluation of your selected part, 4006PG156:
*** Error: [ppr:USER_SELECTION_DOES_NOT_FIT]
The design does not fit in the 4006PG156.
Part has 512 function generators. The design uses 744.
11% utilization of io pins. (14 of 125)
145% utilization of function generators. (744 of 512)
9% utilization of clb flip-flops. (48 of 512)
The design apparently would fit (as follows) in a 4010CB196:
9% utilization of io pins. (14 of 160)
93% utilization of function generators. (744 of 800)
6% utilization of clb flip-flops. (48 of 800)
PPR Terminating due to error status from subtool.

Input XNF Design Statistics (Note 1)

Number of Logic Symbols:	1560	
Number of Flip Flops:	48	
Number of 3-State Buffers:	0	
Number of IO Pads:	14	
Number of Hard Macros:	0	
Number of Nets:	1633	
Number of Pins:	5502	
Equivalent "Gate Array" Gates:		**2548** (Note 2)
- From Logic:		2548
- From Random Access Memories:		0
- From Read Only Memories:		0

Note 2

The value listed for the number of "gate array" gates is an estimate which we provide for your reference. It is based on the symbols (AND, OR, FDRD, etc) in the internal net list file. The value is computed by adding the equivalent gate counts of each symbol. The symbol gate counts are derived from gate array data books.

Figura 4.1 - Detalhe do arquivo *report* gerado pelo ambiente da Xilinx.

Project Information d:\mad\ribas\bench1.rpt

MAX+plus II Compiler Report File
Version 3.20 7/14/93
Compiled: 03/28/94 13:15:54

***** Project compilation was successful

** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	LCs	Shareable Expanders	% Utilized
bench1	**EPM7256GC192**	**37**	**8**	**0**	**240**	**0**	**93 %**
User Pins:		37	8	0			

** RESOURCE USAGE **

	Logic Array Block	Logic Cells	I/O Pins	Shareable Expanders	External Interconnect
A:	LC1 - LC16	1/16(6%)	7/10(70%)	0/16(0%)	4/36(11%)
B:	LC17 - LC32	6/16(100%)	7/10(70%)	0/16(0%)	34/36(94%)
C:	LC33 - LC48	6/16(100%)	0/10(0%)	0/16(0%)	34/36(94%)
D:	LC49 - LC64	6/16(100%)	0/10(0%)	0/16(0%)	34/36(94%)
E:	LC65 - LC80	6/16(100%)	10/10(100%)	0/16(0%)	34/36(94%)
F:	LC81 - LC96	16/16(100%)	3/10(30%)	0/16(0%)	34/36(94%)
G:	LC97 - LC112	6/16(100%)	6/10(60%)	0/16(0%)	34/36(94%)
H:	LC113 - LC128	15/16(93%)	8/10(80%)	0/16(0%)	32/36(88%)
I:	LC129 - LC144	16/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)
J:	LC145 - LC160	16/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)
K:	LC161 - LC176	6/16(100%)	1/10(10%)	15/16(93%)	23/36(63%)
L:	LC177 - LC192	16/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)
M:	LC193 - LC208	16/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)
N:	LC209 - LC224	6/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)
O:	LC225 - LC240	6/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)
P:	LC241 - LC256	6/16(100%)	0/10(0%)	15/16(93%)	23/36(63%)

Total dedicated input pins used:	3/ 4 (75%)
Total I/O pins used:	42/ 160 (26%)
Total logic cells used:	240/ 256 (93%)
Total shareable expanders used:	0/ 256 (0%)
Total Turbo logic cells used:	240/ 256 (93%)
Total shareable expanders not available (n/a):	120/ 256 (46%)

Total input pins required:	37
Total output pins required:	8
Total bidirectional pins required:	0
Total logic cells required:	240
Total flipflops required:	240
Total shareable expanders in database:	0

Synthesized logic cells: 0/ 256 (0%)

Figura 4.2 - Detalhe do arquivo *report* gerado pelo ambiente da Altera.

Por outro lado, no caso dos arquivos de informação sobre a compilação de circuitos dentro do ambiente da Altera são apresentados valores percentuais da utilização dos componentes sobre os quais o projetista estima a complexidade destes circuitos (conforme foi feito nas tabelas 4.5 e 4.8). Porém estes valores correspondem a quantidade de células lógicas utilizadas e não ao aproveitamento interno da sua capacidade lógica (figura 4.2).

4.4.2 Uso de portas lógicas OR exclusivas

Os resultados das tabelas 4.5 (ULA_16) e 4.8 (bench5) mostram que os circuitos que apresentam muitas portas lógicas OR exclusivas (XOR) em sua descrição esquemática têm uma tendência de utilizar muitas células lógicas quando implementados nos componentes da Altera.

Isto ocorre por causa do elevado número de termos produtos quando equaciona-se a lógica da XOR na forma de soma-de-produtos. Pelas características das células lógicas dos PLDs da Altera, necessita-se praticamente uma destas células para cada porta lógica XOR implementada.

4.4.3 Caminho inverso de implementação

Um aspecto importante diz respeito ao caminho inverso de implementação, partindo-se de circuitos em PLDs para sua integração em *gate arrays*. Resultados preliminares, obtidos da análise de circuitos desenvolvidos pelo CPqD/TELEBRÁS para o sistema Trópico, que foram descritos em VHDL e prototipados em FPGAs da Xilinx, são reportados na tabela 4.10 abaixo [68] [69].

Circuito	Complexidade estimada no XC3042	Complexidade no GA2500	% de redução
TB45	3.267 pe	1.298 pe	60,3
TB46	2.780 pe	1.190 pe	57,2

Tabela 4.10 - Comparação de circuitos inicialmente implementados em PLDs.

Claramente, observa-se que a complexidade de tais circuitos, quando implementados no GA2500, sofre uma redução de aproximadamente 60%. Porém, tais resultados sobre apenas dois circuitos não permitem concluir que há uma tendência de redução desta ordem; para isto deve-se aumentar a amostragem de circuitos segundo esta metodologia de caminho inverso de implementação.

Com os dados apresentados neste capítulo, conclui-se que, do ponto de vista da engenharia de sistemas, implementar ASICs por meio de *gate arrays* ou PLDs resulta em avaliações de complexidade totalmente distintas e não coerentes entre si. Um sistema cuja complexidade seja avaliada por sua implementação em ambiente FPGA, por exemplo, pode ser mais facilmente realizável em *gate array*, ou mesmo *standard cell*, com benefícios óbvios de custo e desempenho.

Capítulo 5

Implementação do ambiente proposto e resultados

Após o estudo das arquiteturas e capacidades lógicas desta nova tecnologia de implementação de ASICs, ou seja, os CIs personalizáveis após o encapsulamento ou PLDs, adquiriu-se experiência suficiente para desenvolver o ambiente de prototipagem rápida genérico proposto no segundo capítulo (ítem 2.3), considerando que as abordagens *gate array* e *standard cell* já sejam bastante familiares por terem sido o alvo das principais pesquisas e desenvolvimentos na área de projetos de CIs nas duas últimas décadas.

5.1 Metodologia de projeto

A partir do interesse ou da necessidade de se fazer uso das vantagens, citadas anteriormente, em utilizar um circuito integrado de aplicação específica, do qual o usuário seja o único conhecedor das suas funções lógicas, há várias etapas a serem seguidas, desde o surgimento da idéia de integração de determinado circuito digital até a sua aplicação. Tais etapas podem ser observadas no fluxograma mostrado na figura 5.1.

Para melhor compreensão do texto que segue, deve-se distinguir algumas figuras importantes: o usuário da tecnologia de ASICs, encomendante ou **cliente** e o agente conceptor ou **projetista,** podendo ser o departamento de projetos da própria empresa interessada, um especialista ou consultor especialmente contratado, uma empresa de projetos que atue como *design house*, ou uma empresa que forneça o serviço completo de projeto e/ou fabricação, conhecida como *foundry*, fabricante ou **fornecedor de tecnologia**.

O projetista é o responsável por estabelecer o elo entre o cliente e o fornecedor de tecnologia, além de deter o conhecimento necessário ao desenvolvimento do ASIC, tanto do ponto de vista técnico como do relacionamento comercial, que se estabelecerá entre o cliente e o fornecedor.

Uma vez definida a **idéia**, devem ser elaboradas as **especificações** do CI, funcionais e paramétricas, de acordo com o sistema onde ele será colocado. Nas especificações são consideradas:

a) Descrição do ambiente:
- descrição básica da funcionalidade do sistema no qual o CI faz parte

- interfaces do CI para o sistema
- protocolos e normas do sistema

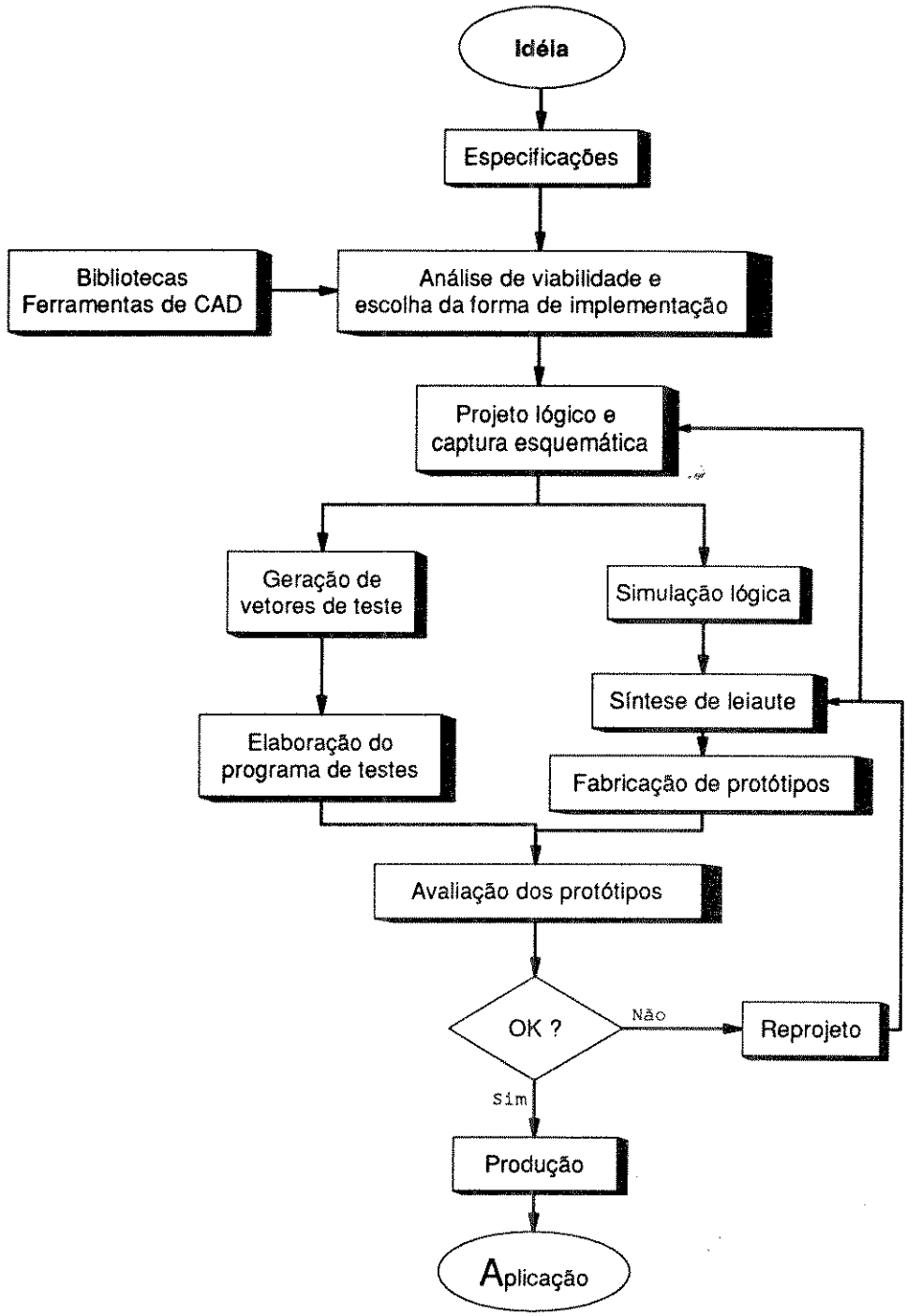


Figura 5.1 - Fluxograma de desenvolvimento de ASICs.

b) Descrição do próprio CI:
- descrição textual da funcionalidade do CI
- diagrama de blocos
- diagramas de tempos de todos os sinais de entrada e saída
- algoritmos internos

c) Detalhamentos funcionais preliminares:
- diagramas esquemáticos de circuito
- simulações preliminares
- existência de PCB inicial, realizado com componentes *standard*, se houver
- emulação do CI em *hardware* (*breadboard*), se houver

d) Parâmetros elétricos:
- níveis de tensão e de corrente nos pinos
- consumo de potência
- freqüência de operação
- tempos de atrasos de caminhos críticos

e) Mercado:
- volume de produção do CI
- tempo de vida do produto
- tempo de lançamento do produto no mercado
- preço alvo de compra do CI

A elaboração de um conjunto de especificações completo e consistente é primordial para o sucesso do projeto de um ASIC. É com base nas especificações que se avaliam protótipos e se valida o desenvolvimento do ASIC.

As especificações determinam, em última análise, o limite de responsabilidade entre o que um cliente deseja e o que se obtém do projetista, utilizando o fornecedor de tecnologia disponível para a realização do circuito.

Um mau funcionamento do CI em campo, que não seja previsível pela informação disponível no conjunto de especificações, implica em **erro de especificação**, resultando em perda para o cliente. Por outro lado, uma falha oriunda não do projeto em si, mas sim da tecnologia empregada, obrigaria o fornecedor a fabricar novamente o CI sem ônus ao cliente.

Com base em sua experiência no desenvolvimento de ASICs, o projetista tem condições de auxiliar o cliente a obter o melhor conjunto de especificações possível, que seja suficiente para conduzir a bom termo este desenvolvimento. Mesmo assim, continua sendo do cliente a responsabilidade de aprovar as especificações acordadas.

Definido o conjunto de especificações, o cliente pode submeter seu projeto à **análise de viabilidade técnica e econômica** de integração. Nesta etapa são realizadas, de uma forma geral, as seguintes atividades:

a) Análise técnica:
- análise das especificações funcionais
    - compreensão das funções do componente
    - tipo de arquitetura
    - factibilidade das funções pretendidas
    - necessidade de adaptações e seu impacto sobre as especificações
- análise da complexidade do CI
    - número de portas equivalentes
    - número de *pads*
    - estimativa do tamanho da pastilha
    - tipo de encapsulamento requerido
- análise dos requisitos de desempenho
    - freqüência de operação
    - rapidez de sinais específicos
    - *timing* de caminhos críticos
- deteção de requisitos funcionais específicos
    - níveis de tensão especiais
    - consumo de potência
    - condições ambientais: faixa de temperatura, estabilidade de parâmetros, e outros
- requisitos de qualidade e confiabilidade
    - classe de qualidade necessária à aplicação
    - ensaios específicos durante produção
    - testes específicos para avaliação de protótipos e para produção

b) Análise econômica
- avaliação dos recursos necessários (humanos e físicos)
- estimativa do tempo de desenvolvimento
- condições de fornecimento em produção
- estimativa de custos
    - custo de desenvolvimento
    - custo de produção

Nesta análise são consideradas as especificações originais de complexidade e desempenho do circuito, compatibilidade paramétrica, tempo de lançamento do produto no mercado, preço alvo, de forma que seja possível a **escolha da forma de implementação** a ser utilizada.

O estudo de viabilidade técnica e econômica é, então, submetido ao cliente, que decide sobre as condições de continuidade do projeto. No caso de inviabilidade, encerra-se as atividades normais de desenvolvimento do ASIC,

podendo, entretanto, o projeto seguir adiante sob forma de implementação com CIs *standard* comerciais.

Viabilizada a integração e escolhida a forma a ser utilizada para implementação do circuito, inicia-se a fase de **projeto lógico** propriamente dita. Inicialmente, o circuito é descrito detalhadamente através de **captura de diagramas esquemáticos**, tabelas verdades, equações booleanas, formas de onda dos sinais de entrada e saída ou linguagens de descrição de *hardware* (HDL).

Dois casos de projeto lógico se apresentam ao projetista:

a) projeto lógico detalhado somente até o nível de diagramas de blocos;
b) projeto lógico detalhado completamente até o nível de portas lógicas básicas (primitivas lógicas).

No caso (a), o projetista necessita detalhar cada bloco, hierarquicamente, até a obtenção de uma descrição a nível de portas básicas, de forma a poder simular logicamente utilizando os modelos disponíveis nas ferramentas de simulação. No caso (b), a captura do projeto no ambiente de ferramentas é imediata.

É, então, realizada uma **simulação lógica** funcional sobre esta descrição, para validação da lógica, e em seguida realiza-se uma simulação temporal utilizando-se as características de *timing* de cada função lógica implementada, segundo a metodologia adotada. Esta simulação temporal permite uma avaliação prévia do sincronismo e atrasos de propagação de sinais dentro do circuito, antes da elaboração do leiaute.

As características de *timing* das funções lógicas, utilizadas nas simulações temporais, são obtidas previamente através de simulações elétricas (SPICE) [70], que podem fornecer também os valores de consumo de potência por função. No caso de utilização de bibliotecas de células, tais valores já estão diponíveis para uso.

Como parte da atividade de simulação lógica, realiza-se em paralelo a **geração de vetores de teste.** Conforme o circuito é simulado, são naturalmente criados padrões de estímulos e respostas otimizados para demonstrar e validar a funcionalidade de cada bloco lógico. O conjunto de estímulos e respostas globais do circuito, aplicáveis e observáveis nos pinos de entrada e saída do CI, são mantidos a parte e constituirão os vetores de teste do CI [71].

Cabe ressaltar que existem diversas técnicas para obtenção do conjunto de vetores de teste, adequados a garantir a testabilidade do CI e permitir a deteção de falhas [72]. Técnicas de projeto visando testabilidade permitem

prover meios de se construir o circuito lógico e testá-lo, de forma que idealmente todas as entradas de portas lógicas tenham seus níveis lógicos externamente controlados, e que os níveis das saídas sejam também observáveis externamente, por meio dos pinos do CI. Durante a elaboração de vetores de teste, o projeto lógico é refinado, otimizando-se a testabilidade do CI.

Após a descrição e simulação do circuito, a etapa que se segue consiste na **síntese de leiaute**. O leiaute é a descrição da geometria dos diversos níveis de máscara, com sua topologia, dimensões e posições, que serão utilizadas no processo de fabricação de CIs.

Cada porta lógica ocupa uma área de pastilha bem definida, correspondente ao seu leiaute, que pode ser posicionado de forma automática ou manual, a fim de otimizar a área final do CI e facilitar o roteamento das interconexões.

Portanto, a fase de síntese de leiaute do circuito consiste no posicionamento e interconexão (*placement* e *routing*) de cada porta lógica, assim como das linhas de alimentação, Vdd (+) e Vss (-), e dos circuitos de entrada e saída de sinais (*pads*).

Durante o processo de síntese, são utilizados verificadores de regras elétricas e de desenho, tais como ERC (*electrical rule checker*), DRC (*design rule checker*) e LVS (*layout versus schematic*), que servem para garantir ao projetista que não há curto-circuitos, violação das regras de projeto, e incompatibilidade do leiaute final com o diagrama esquemático inicialmente descrito.

Finalizado o leiaute do CI, pode ser realizada uma nova simulação lógica, denominada simulação pós-leiaute, que inclui as capacitâncias parasitas extraídas das interconexões, para se verificar alguma alteração na temporização dos sinais. Caso se observe erro no funcionamento lógico, o projeto deve ser revisto, alterando-se o circuito esquemático ou o leiaute, de forma a corrigir o problema.

Juntamente com a fase de leiaute conclui-se o projeto do CI, propriamente dito, passando-se a etapa de **fabricação de protótipos**. As primeiras pastilhas processadas são utilizadas como protótipos para avaliação do projeto do ASIC.

Enquanto se espera pela fabricação, o conjunto de vetores de teste, as especificações paramétricas e de desempenho do CI são utilizados para a **elaboração do programa de testes**, visando a futura avaliação de protótipos, normalmente realizada em equipamentos de teste automáticos.

A **avaliação dos protótipos** corresponde a realização de testes funcionais, utilizando os vetores de teste, e paramétricos, onde se mede tensões, correntes, freqüências, consumo de potência, etc. É importante salientar que os protótipos devem ser avaliados, não somente segundo suas especificações funcionais e paramétricas, mas também dentro do sistema ou produto final onde será utilizado, com vistas à deteção de deficiências na especificação original do ASIC.

Uma vez validado os protótipos, o ASIC é posto em fase de **produção**. Caso contrário, há a necessidade de retorno às etapas anteriores de geração do leiaute, ou até mesmo de descrição e simulação do circuito, caracterizando a atividade de **reprojeto**.

Estando o CI em fase de produção, encerra-se o elenco de atividades típicas de desenvolvimento. Por parte do cliente, seguem-se agora todas as atividades referentes a **aplicação** do componente, ou seja, produção e comercialização do produto final no qual o CI se incere. Define-se, então, um relacionamento comercial mais estreito entre o cliente e o fornecedor de tecnologia, podendo ou não haver participação do projetista nesta transação.

Por fim, cabe ressaltar que o projetista detém completamente a informação técnica sobre o projeto, podendo auxiliar o cliente quanto a dúvidas e questões funcionais ao longo do uso do componente. É salutar, entretanto, que o cliente detenha em sua empresa ao menos um profissional capacitado a dominar igualmente todos os detalhes do desenvolvimento do ASIC, para desta forma melhor utilizá-lo.

A metodologia de projeto, descrita detalhadamente acima, foi desenvolvida e é utilizada pelas formas de implementação personalizáveis por máscaras, que praticamente consistiam nas únicas opções existentes para o projeto de ASICs até o final dos anos 80. Com o surgimento dos PLDs de maior capacidade lógica (segunda geração) esta metodologia de projeto sofreu algumas alterações devido à inexistência de elaboração de leiautes e obtenção de protótipos nesta nova tecnologia.

A figura 5.2 apresenta uma variação do fluxograma de desenvolvimento de ASICs, adaptado ao uso de CIs personalizáveis após o encapsulamento para implementação destes circuitos integrados de aplicação espacífica.

Neste novo fluxograma a etapa de síntese de leiaute é trocada pelas fases de compilação das funções lógicas desejadas dentro dos componentes programáveis: otimização lógica, mapeamento das equações booleanas, posicionamento e roteamento através da configuração dos elementos de programação [30].

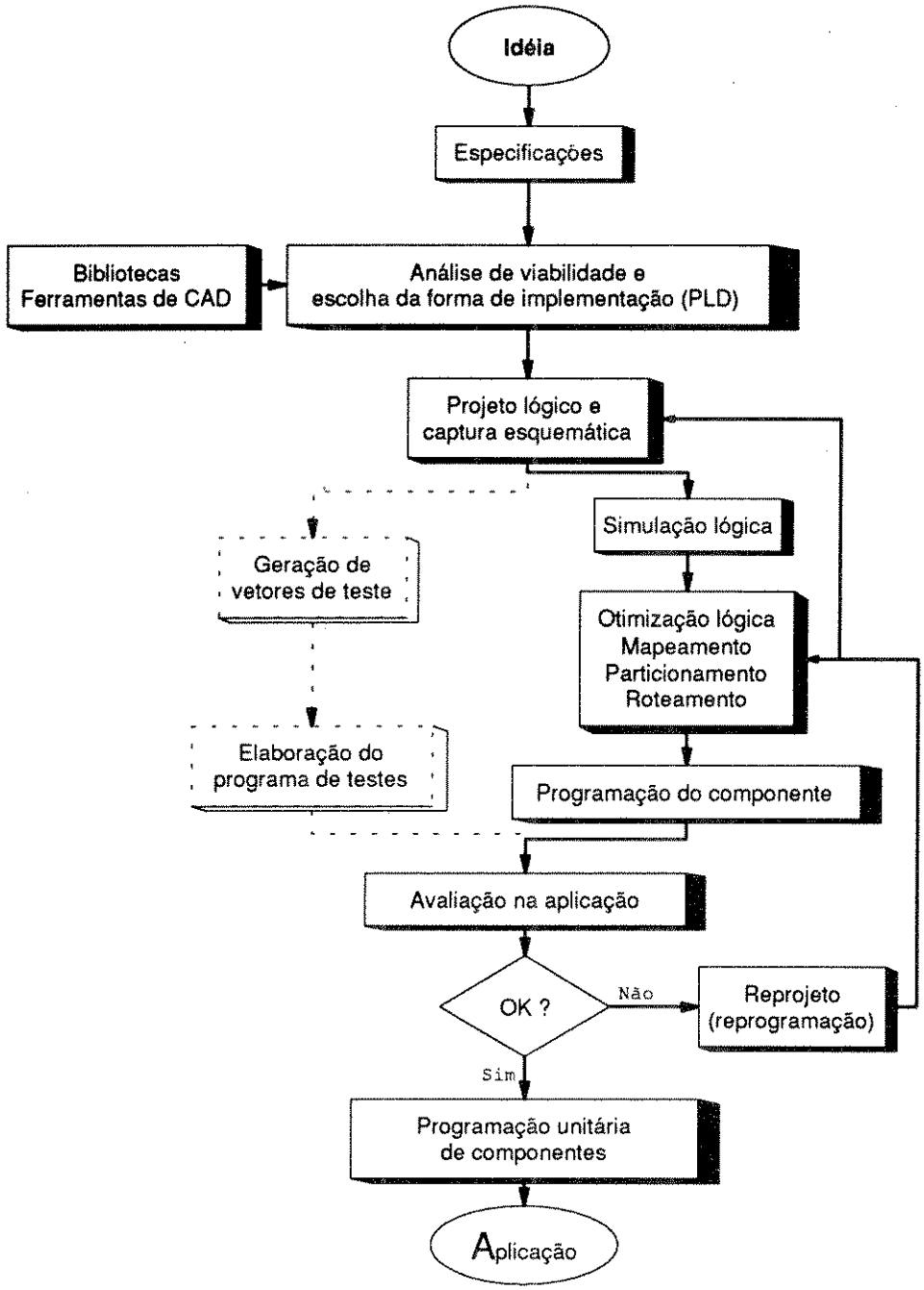


Figura 5.2 - Fluxograma de desènvolvimento de ASICs através de PLDs.

O incorreto funcionamento do ASIC é ocasionado por falha nas especificações preliminares ou na descrição do circuito, não detectado durante as simulações lógicas. Caso o circuito esteja de acordo com o esperado, ele é utilizado diretamente na aplicação, não havendo a etapa de produção em larga escala como ocorre com os CIs personalizáveis por máscaras; cada PLD deve ser programado individualmente.

Com o surgimento de ferramentas de CAD para desenvolvimento de CIs através de linguagens HDL, ao invés da captura esquemática, a etapa de **análise de viabilidade e escolha da forma de implementação**, apresentada no fluxograma da figura 5.1, foi dividida em duas. A análise de viabilidade de uso de ASICs continua sendo realizada após as especificações, porém a escolha da forma de implementação passou a ser feita somente depois do circuito descrito e validado por simulações lógicas funcionais, para então serem realizadas as simulações lógicas temporais, com os dados de *timing* da tecnologia escolhida.

5.2 Descrição e objetivos do ambiente proposto

É importante lembrar que o objetivo dos ambientes de prototipagem rápida, discutido no primeiro capítulo deste trabalho, é automatizar e otimizar as fases que compreendem desde o projeto lógico até a fabricação de protótipos, apresentadas nos fluxogramas das figuras 5.1 e 5.2. Ou seja, as etapas de especificação do circuito, análise de viabilidade, escolha da forma de implementação e avaliação dos protótipos, quando existentes, não são abordadas por estes ambientes.

Observa-se, também, que os ambientes encontrados no mercado brasileiro, e descritos no capítulo inicial, são específicos para cada uma das formas de implementação.

A possibilidade de migração de projetos entre estes três ambientes é remota e, por vezes, resulta num trabalho similar a um reprojeto do circuito. A conversão de ASICs para diferentes tecnologias é tratado por especialistas como pouco provável por ela não ser totalmente automática e exigir um certo esforço humano. Muitas vezes é compensatório realizar o reprojeto e permitir a adição de novas características ao circuito (vide ítem 2.3). O que eles defendem é que sejam realizados projetos paralelos, em diferentes formas de implementação, e postos em produção de acordo com o comportamento do mercado [23].

Deve-se considerar, entretanto, que nos países em desenvolvimento, como o Brasil, a Argentina e demais países latino-americanos, antes de defender a tese de projetos paralelos de circuitos integrados, é necessário vencer os obstáculos que distanciam a realidade dos empresários e os benefícios da microeletrônica. Basicamente, há três barreiras a serem derrubadas para que os ASICs tornem-se uma opção para o desenvolvimento de produtos eletrônicos nestes países:

- **cultural,**
- **tecnológica e**
- **econômica.**

A primeira dificuldade encontrada para o uso da microeletrônica é a **barreira cultural**. Os empresários e projetistas desconhecem esta área, não tendo condições de avaliar as suas vantagens e/ou desvantagens, além de não confiarem em componentes que eles mesmos possam desenvolver. Utilizar componentes *standard,* já familiares, para implementar seus sistemas é mais simples e seguro, mesmo que algumas vezes mais caros.

Quando se consegue mostrar a viabilidade dos ASICs e a conseqüente evolução dos produtos, defronta-se com o choque tecnológico criado - **barreira tecnológica**. A maioria dos engenheiros eletrônicos dos países do terceiro mundo não estão familiarizados a área de microeletrônica, e para que eles possam tirar proveito desta é necessário um aprendizado para compreender o que se passa sobre as pastilhas semicondutoras e para pilotar as ferramentas de CAD envolvidas. Isto causa um certo desconforto aos profissionais mais antigos, ao serem cobrados por sua atualização tecnológica, e cria a necessidade das empresas investirem no treinamento dos seus engenheiros.

Por fim, depois de conquistada a confiança do mercado e convencido os profissionais da área de eletrônica que eles devem se aprimorar e se manter em dia com os avanços da eletrônica, a etapa mais difícil a ser vencida é provar aos empresários que é viável economicamente utilizar ASICs, mesmo havendo a necessidade de alteração das placas de circuito impresso, de aquisição de equipamentos e correndo os risco de projeto inerentes. A **barreira econômica** é a mais difícil de ser vencida pois nestes países visa-se principalmente o lucro imediato, o que não é oferecido pela microeletrônica para quem está começando a utilizá-la.

O ambiente de prototipagem rápida genérico proposto não consiste em desenvolver ferramentas de CAD inéditas para auxílio nos projetos criar novos processos de fabricação de CIs. O objetivo deste trabalho é compatibilizar o interfaceamento de ambientes já existentes visando permitir ao projetista migrar de uma forma de implementação à outra de maneira rápida e segura. Desta maneira, o projetista estará trabalhando sempre nas versões mais recentes das ferramentas de CAD envolvidas.

A partir do circuito descrito e validado funcionalmente, é escolhida a forma de implementação adequada e são realizadas adaptações para compatibilizar as especificações do circuito com as características de *timing* correspondentes às tecnologias de fabricação. Com isso, pretende-se possibilitar projetos paralelos, conforme é defendido em [23], sem a necessidade de redescrever o ASIC.

O ambiente proposto visa também auxiliar na quebra das barreiras, citadas acima, que dificultam o crescimento da área de microeletrônica nos países em desenvolvimento. Após cumpridas as etapas de especificação,

descrição e validação funcional do circuito, o cliente recebe no mesmo dia a versão em PLDs do seu ASIC, que poderá ser posto na placa impressa e testado no próprio ambiente final de uso. Os PLDs evitam a espera de semanas para obtenção dos protótipos, para serem testados e, somente então, serem postos em produção, como acontece com os *gate arrays* e *standard cells*.

Esta versão inicial do ASIC implementada em componentes programáveis é reforçada pelo fato de que o volume inicial de vendas de produtos eletrônicos, nos referidos países, nem sempre é suficiente para justificar o uso de formas de implementação personalizáveis por máscaras.

Com o PLD configurado disponível ao cliente já foram praticamente vencidas as barreiras cultural, tecnológica e econômica. A partir disso, a passagem para as abordagens *standard cell* e *gate array* só depende da resposta do mercado: aumento do volume de vendas, melhoria do desempenho elétrico ou integração de um maior número de funções lógicas dentro do mesmo CI.

O fluxograma da figura 5.3 descreve o ambiente proposto. Observa-se que as etapas de desenvolvimento da idéia, de especificações e de análise de viabilidade de integração das funções lógicas em um único CI continuam sendo as atividades iniciais realizadas na metodologia de projetos de ASICs.

Optou-se pela descrição esquemática do circuito para amenizar o choque ou a barreira tecnológica, inevitável com a introdução de circuitos integrados de aplicação específica para substituição de componentes *standard.* Nota-se que o uso de esquemáticos para descrição e análise de circuitos eletrônicos é o mais difundido pelo grande número de usuários de ferramentas como o Tango e o Orcad.

Utilizar ambientes baseados em descrições VHDL resulta na aquisição de ferramentas de síntese lógica e *hardware* compatível (geralmente estações de trabalho - *workstations*) de elevado custo. No Brasil este tipo de instalação é encontrado somente em universidades e centros de pesquisa e desenvolvimento estatais, como o CPqD (TELEBRÁS), não se estendendo às empresas privadas da área.

Porém, o desenvolvimento de ASICs sobre ferramentas de VHDL não inviabiliza o uso do ambiente proposto pois a descrição textual pode facilmente ser traduzida para uma descrição esquemática.

Sobre o esquemático genérico são realizadas simulações lógicas funcionais. Paralelamente a estas simulações são gerados os vetores de teste para a avaliação dos protótipos, quando existentes, ou para a confirmação do correto funcionamento do ASIC implementado em componentes PLDs.

Somente após a validação funcional do esquemático é que realiza-se a escolha da forma de implementação mais adequada, segundo as especificações elétricas e as condições de mercado envolvidas (*time-to-market*, volume de produção).

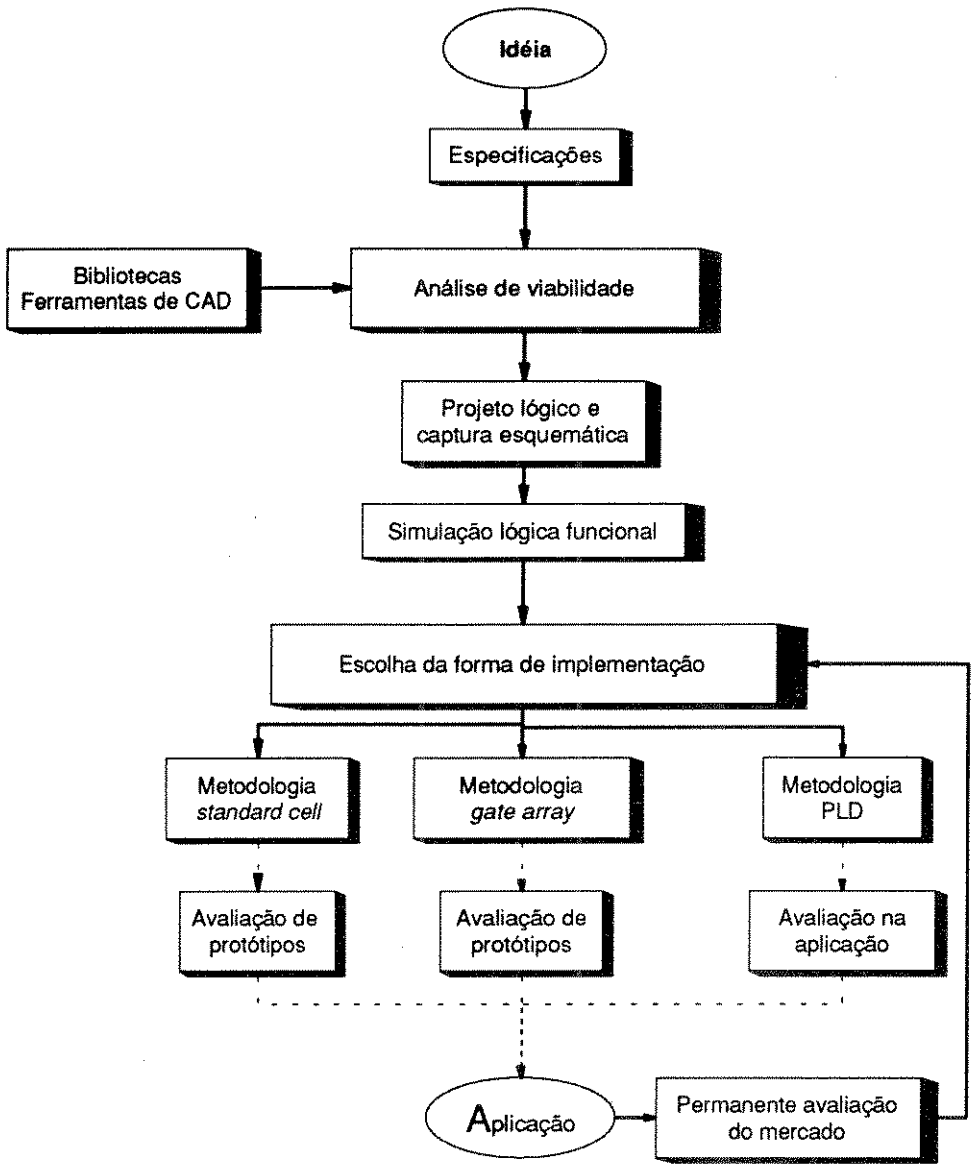


Figura 5.3 - Fluxograma do ambiente de prototipagem rápida genérico proposto.

A partir disso, o projeto segue o fluxo normal correspondente a cada tecnologia, seja com a geração de leiautes ou com a compilação de PLDs. Sugere-se, entretanto, que seja realizada a implementação do ASIC sobre os CIs personalizáveis após o encapsulamento, mesmo que esta não se apresente a forma mais eficaz, devido a sua rapidez na obtenção do circuito, a inexistência

de protótipos e a conseqüente possibilidade de avaliar o funcionamento do ASIC dentro do sistema final de trabalho.

Caso a evolução do mercado venha a exigir o reprojeto do circuito sobre uma forma de implementação diferente da original, a fim de reduzir o custo final do produto alterado por variações nas vendas, o projetista utilizará a descrição esquemática validada e os vetores de teste já desenvolvidos para adaptar as novas características temporais, chegando rapidamente ao novo ASIC. O esforço humano neste reprojeto é mínimo pois as etapas a serem realizadas estão praticamente automatizadas dentro dos ambientes de prototipagem rápida específicos.

5.3 Estudo de viabilidade

Para avaliar a viabilidade de implementação do ambiente proposto é necessário considerar as disponibilidades de equipamentos e ferramentas de CAD. A idealização deste ambiente é fruto da evolução natural e da experiência sobre o ambiente GA2500, em desenvolvimento na Fundação Centro Tecnológico para Informática (CTI).

Daqui em diante, será discutida a factibilidade da sua operacionalização em um meio real de projetos de CIs. Para isto, serão consideradas as condições de trabalho (*hardware* e *software*) existentes no Laboratório de Projetos de Circuitos Integrados (LPCI) do CTI, de onde o ambiente é originário e onde ele deverá ser efetivamente instalado.

O ambiente GA2500 foi desenvolvido sobre ferramentas de CAD da Mentor Graphics específicas para *standard cell*, adaptadas para sua utilização na implementação de circuitos sobre a matriz *gate array* configurável por um único nível de metal [15]. A parte de projetos do GA2500 encontra-se operacional. As matrizes pré-fabricadas são adquiridas da *foundry* ES2 (França) e personalizadas no CTI.

No caso da opção de projetos *standard cell*, o CTI trabalha até a geração dos leiautes e serve de interface entre o cliente e a ES2, para a fabricação dos protótipos ou produção dos ASICs. As bibliotecas de células destas duas abordagens foram desenvolvidas com características *timing* bastantes semelhantes a fim de simplificar as ressimulações lógicas temporais durante a migração de projetos de uma abordagem a outra.

A inclusão da opção das formas de implementação de ASICs em PLDs, dentro deste contexto, exige inicialmente um trabalho de avaliação dos fabricantes destes componentes. Estão presentes no mercado brasileiro as empresas Altera (através da União Digital Com. e Rep. Ltda), a Xilinx

(representada pea Hicad Sistemas Ltda) e a Actel / Texas Instruments. O resultado deste estudo pode ser dividido em duas categorias:

a) a adeqüabilidade dos *softwares* de programação, ou **características das ferramentas,** e

b) a eficiência das arquiteturas segundo os objetivos almejados - **características dos componentes**.

Quanto às **características das ferramentas** de cada ambiente de PLDs, o primeiro ponto analisado foi em relação ao *hardware* necessário para a instalação destas ferramentas. Dos três fabricantes citados, nenhum é compatível com as estações de trabalho disponíveis no LPCI / CTI, Apollo DN5500:

- **Actel / Texas** - trabalha somente em plataforma PC, não permitindo até o momento trabalhar sobre *workstations*.
- **Altera** - permite trabalhar em *workstations* SUN ou Apollo (porém, numa versão mais avançada que a DN5500), mas exige a interface com um computador PC para gravar os componentes.
- **Xilinx** - permite trabalhar em *workstations*, sem a necessidade de interface PC, porém seu ambiente também é incompatível com a Apollo DN5500.

Portanto, para incluir qualquer um destes fabricantes no ambiente proposto há necessidade de aquisição de novas estações de trabalho ou a alocação de um computador PC para projetos em PLDs.

O segundo ponto considerado na avaliação das ferramentas é quanto à possibilidade de transferência da descrição esquemática da Mentor Graphics para os ambientes de PLDs. Observa-se, inicialmente, que o único fabricante que possui um sistema "fechado" para projeto de ASICs sobre componentes programáveis é a Altera. Os demais fabricantes, Xilinx e Actel, necessitam interagir com editores esquemáticos de outras empresas para descreverem seus projetos.

A linguagem EDIF é a comumente usada para esta interface. Porém, as bibliotecas de células utilizadas nas descrições apresentam algumas particularidades:

- **Actel / Texas** - cada família de componentes possui uma biblioteca de símbolos lógicos específica, ou seja, a evolução da implementação de um esquemático para um componente maior obriga o projetista a recapturar novamente a lógica do circuito;
- **Altera** - utiliza um esquemático genérico para todas as suas famílias de componentes;
- **Xilinx** - também faz uso de bibliotecas específicas de cada família; o fabricante fornece conversores, mas a total compatibilidade da lógica

convertida não é garantida pois, por exemplo, células *latches* só são encontradas na família XC2000.

O uso de bibliotecas específicas entra em conflito com a proposta de descrição de esquemáticos genéricos dentro do ambiente descrito.

Quanto às **características dos componentes**, cabe salientar novamente que um dos objetivos do ambiente proposto é auxiliar na quebra das barreiras que dificultam o crescimento da microeletrônica em países como o Brasil. Por isso é importante que o ambiente de PLDs escolhido permita o uso de componentes reprogramáveis, para evitar o desperdício de peças, e não-voláteis, descartando a necessidade de carregamento da configuração do componentes a cada ausência de energia de alimentação.

Segundo estas características, descarta-se a possibilidade de utilização de FPGAs com anti-fusíveis (não reprogramáveis) e de FPGAs com SRAM (voláteis). Os EPLDs possuem o perfil descrito, e eles são fornecidos tanto pela Altera quanto pela Xilinx, apesar deste último ter entrado recentemente nesta tecnologia.

Portanto, através desta análise de ferramentas e componentes conclui-se que o único fabricante, dentre aqueles encontrados no Brasil, factível de ser incluído no ambiente proposto sem um custo adicional elevado para aquisição de equipamentos é a Altera. Estudo semelhante deve ser realizado sempre que for instalado um ambiente com as mesmas características, pois os resultados são dependentes das condições de trabalho disponíveis.

Por fim, um fator não abordado por este trabalho, mas que deve ser levado em conta por qualquer empresa que invista na aquisição de equipamentos e ferramentas de CAD, é o serviço de suporte técnico oferecido por representantes ou pelo próprio fabricante de PLDs.

5.4 Implementação do ambiente

Uma vez definidos claramente os objetivos do ambiente proposto, ou seja, não serão desenvolvidas ferramentas de projeto ou processos de fabricação inéditos, mas um trabalho de interfaceamento entre ambientes profissionais para prototipagem de CIs, pode-se partir para a implementação, propriamente dita, deste ambiente proposto.

Observa-se que o fato do LPCI / CTI estar suportado por ferramentas Mentor Graphics simplifica o trabalho de interfacear as abordagens *gate array* e *standard cell*. Foi dito anteriormente que a matriz GA2500 foi desenvolvida sobre ferramentas Mentor, específicas para projetos *standard cell*, realizando-se uma série de adaptações para trabalhar sobre uma matriz pré-difundida.

Portanto, para elaborar leiautes de CIs *standard cell* basta desenvolver uma biblioteca de células e seus modelos de simulação (descrição lógica das células e suas características temporais), e utilizar eficientemente as ferramentas disponíveis.

A adaptação de um fabricante de PLDs, segundo as discussões de *softwares* e componentes, apresentadas acima, sugere que seja utiizado um computador PC, com o ambiente MaxPlus2 instalado, conectado às estações de trabalho Apollo DN5500. A transferência de dados do editor de esquemáticos Neted (Mentor) para o compilador da Altera pode ser feita através da linguagem EDIF, desde que este esquemático apresente símbolos lógicos provenientes da biblioteca genérica da Mentor (*gen_lib*).

A idéia básica da descrição dos circuitos através de capturas esquemáticas genéricas é utilizar símbolos genéricos que permitam a avaliação funcional do circuito, considerando nulos os tempos de atraso de propagação de sinais em cada porta lógica. A migração deste esquemático para as diferentes metodologias de projetos *gate array*, *standard cell* e PLD, ocorre através da substituição dos símbolos lógicos genéricos pelos símbolos correspondentes em cada abordagem, agora com as suas características de *timing* associadas para a realização de simulações lógicas temporais.

Esta troca dos símbolos lógicos do esquemático é facilitada pela própria estrutura de arquivos geradas pelas ferramentas da Mentor Graphics: quando realiza-se uma captura esquemática não gera-se apenas um arquivo de descrição, como ocorre nas ferramentas Tango e Orcad, mas um diretório com vários arquivos específicos:

- *sheet1.nrel* : arquivo que indica a versão do *sheet1.nrel_$?* em uso;
- *sheet1.nrel_$1* : arquivo com dados da conectividade do esquemático;
- *sheet1.nrel_$1.ref* : arquivo de busca das células da biblioteca;
- *sheet1.pic* : arquivo que indica a versão do *sheet1.pic_$?* em uso;
- *sheet1.pic_$1* : arquivo responsável pela representação gráfica.

O arquivo *sheet1.nrel_$1.ref*, particularmente, é o único arquivo textual (ASCII) e responsável pela busca dos símbolos lógicos e sub-circuitos da hierarquia do esquemático. Basta alterar este arquivo textual para trocar a biblioteca de primitivas, em uso na descrição. Na figura 5.4 é visto um exemplo deste alteração. Este procedimento deve ser realizado para todos os níveis hierárquicos da descrição.

Porém, esta troca dos símbolos de uma biblioteca para outra, apenas modificando o arquivo *sheet1.nrel_$1.ref*, pode trazer alguns contratempos como a distorção das ligações dos diagramas esquemáticos ou a inconsistência dos símbolos entre si. Isto é causado por três fatores:

a) **inexistência de determinadas portas lógicas na biblioteca para a qual se quer migrar**: por exemplo, caso seja utilizada a função OR exclusiva de 4 entradas na descrição genérica, não haverá correspondente ao tentar utilizar a biblioteca GA2500 [16];

b) **símbolos que representam a mesma função lógica podem ser diferentes em cada biblioteca**: o fato dos símbolos apresentarem nomes ou representações gráficas diferentes, como tamanho e a posição dos pinos, pode trazer sérios problemas. Caso a mesma porta lógica apresente nomes distintos nas bibliotecas, como **ffdsr** e **ffd_sr**, o novo esquemático não encontrará a referida célula. E a existência de símbolos da mesma porta lógica com diferentes representações gráficas ocorre distorções no diagrama esquemático ou mesmo incompatibilidade dos seus pinos de entrada e saída (figura 5.5);

c) **diferentes pontos de referência das representações gráficas dos símbolos**: mesmo que o desenho de uma certa porta lógica seja idêntico nas bibliotecas disponíveis, também podem ocorrer distorções dos esquemáticos por estes símbolos possuírem diferentes pontos de referência para seu posicionamento (figura 5.6).

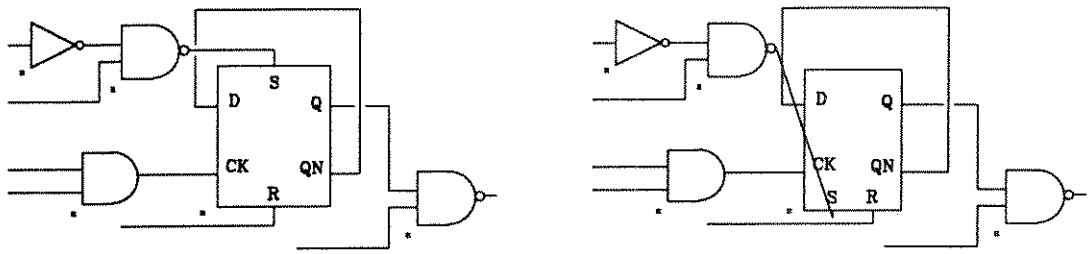
```
* arquivo sheet1.nrel_$1.ref original de um circuito:
        //mars/local_user/bibliotecas/genérica/and2
        //mars/local_user/bibliotecas/genérica/nand4
        //mars/local_user/bibliotecas/genérica/mux2_1
        //mars/local_user/bibliotecas/genérica/latch_r
        //mars/local_user/bibliotecas/genérica/ffjk
        //venus/local_user/projetos/multiplicador/half_adder

* arquivo sheet1.nrel_$1.ref modificado para a utilização da biblioteca GA2500:
        //mars/local_user/bibliotecas/GA2500/and2
        //mars/local_user/bibliotecas/GA2500/nand4
        //mars/local_user/bibliotecas/GA2500/mux2_1
        //mars/local_user/bibliotecas/GA2500/latch_r
        //mars/local_user/bibliotecas/GA2500/ffjk
        //venus/local_user/projetos/multiplicador/half_adder
```

Figura 5.4 - Exemplo da alteração do arquivo *sheet1.nrel_$1.ref* de um circuito para a troca da biblioteca de células utilizada.

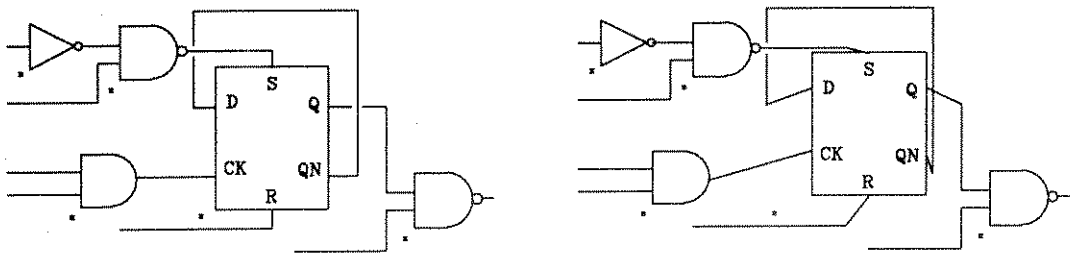
A solução encontrada para evitar estes três casos citados acima foi adotar símbolos hierárquicos para representar as primitivas. Estes símbolos serão copiados em cada biblioteca, de forma a garantir a existência e a semelhança das representações gráficas (formato e ponto de referência), e serão diferenciados em um nível mais baixo de hierarquia (figura 5.7). Desta forma, quando se troca a busca de uma biblioteca de células por outra o

desenho das portas lógicas praticamente não se altera. O que muda é um nível hierárquico abaixo, que pode ser representado por uma única porta lógica ou por um conjunto delas.



(a) (b)

Figura 5.5 - Distorção da descrição esquemática por causa da troca de símbolos com representações gráficas diferente: (a) arquivo original; (b) arquivo modificado.



(a) (b)

Figura 5.6 - Distorção da descrição esquemática causada pela diferença entre os pontos de referência dos símbolos de cada biblioteca: (a) arquivo original; (b) arquivo modificado.

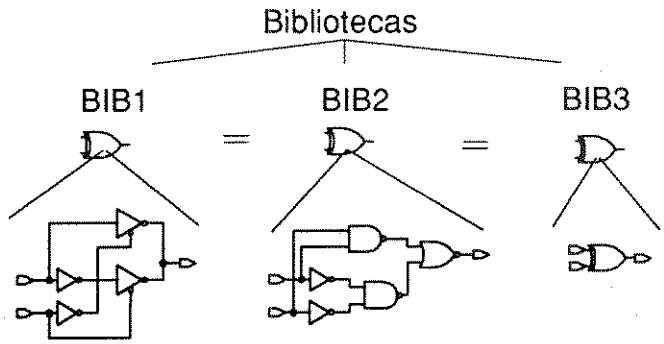


Figura 5.7 - Símbolos hierárquicos para representar as primitivas.

Porém, para que esta estratégia seja viável, sugere-se uma estrutura de diretórios das bibliotecas, dispostos lado a lado, dentro dos quais todas as portas lógicas apresentam o mesmo nome, a mesma representação gráfica, e são hierárquicas. Reinterando que dentro desta hierarquia será descrita a função lógica correspondente com as células da respectiva biblioteca.

O programa para alteração do arquivo textual *sheet1.nrel_$1.ref*, para trocar a biblioteca genérica por uma específica de um certo ambiente (*gate array*, *standard cell* ou mesmo PLD), foi implementado de forma a não modificar a descrição original. É gerado um novo esquemático com os símbolos da nova biblioteca, ou seja, copia-se o diretório do circuito original (genérico) e altera-se o seu arquivo de referências (*sheet1.nrel_$1.ref*). O mesmo procedimento é tomado para modificar os sub-circuitos, existentes na hierarquia, que fazem parte da descrição esquemática (ver anexo II).

Resumindo, primeiramente o projetista realiza a captura esquemática do seu ASIC com o uso de uma biblioteca genérica. A seguir o seu circuito é analisado e validado funcionalmente através de simulações lógicas que consideram nulos os tempos de atrasos. Somente após este etapa é que realiza-se a escolha da melhor forma de implementação deste ASIC, segundo os fatores já discutidos. Utiliza-se, então, o programa de conversão para gerar o novo esquemático, correspondente a forma de implementação escolhida, e ressimulá-lo, considerando agora as características temporais das portas lógicas. A partir desta etapa, segue-se normalmente a metodologia de projeto referente a cada forma de implementação.

Caso, se queira implementar o ASIC sobre os PLDs da Altera é necessário gerar uma descrição em linguagem EDIF do esquemático, composto por símbolos da *gen_lib* (Mentor), e transferir este arquivo (por rede, disquete, ou qualquer outra maneira permitida) para o computador PC, onde encontra-se instalado o ambiente MaxPlus2.

5.5 Estado atual e resultados

Atualmente, o ambiente de prototipagem rápida genérico não encontra-se operacional, pois depende-se da aquisição do sistema de desenvolvimento de PLDs MaxPlus2 da Altera, por parte do Instituto de Microeletrônica do CTI.

O ambiente de prototipagem GA2500 está com a sua parte de projetos finalizada, restando apenas que as instalações dos laboratório de processo, onde serão realizadas as etapas restantes para a personalização das matrizes *gate array*, sejam concluídas. A previsão é que para o início de 1995 este serviço esteja disponível no mercado brasileiro.

Quanto a possibilidade de projetos de CIs na abordagem *standard cell*, a ser utilizada também dentro do ambiente proposto, foi projetada uma biblioteca de células com características de *timing* bastante semelhantes à biblioteca GA2500. Desta forma, simplifica-se a fase de ressimulações temporais necessária devido à migração de circuitos de uma tecnologia para outra. Portanto, ASICs dentro da metodologia *standard cell* já podem ser projetados no CTI para serem fabricados na *foundry* ES2, em tecnologia CMOS 1.2μm ou 1.5μm, com dois níveis de metal.

Dentro deste panorama, observa-se que o ambiente proposto está bem especificado, as possíveis incompatibilidades para sua implantação foram cuidadosamente avaliadas, realizou-se um estudo aprofundado sobre os fabricantes (seus componentes e ferramentas de projeto) e o programa para conversão dos esquemáticos encontra-se operacional. Para que este ambiente esteja realmente em uso há inúmeras condições de contorno e fatores político-econômicos que estão fora do alcance dos profissionais envolvidos neste trabalho.

Capítulo 6

Conclusão

Este trabalho objetivou desenvolver um ambiente de projeto que permitisse aos projetistas realizarem uma única vez a descrição esquemática e a validação funcional do ASIC, para então escolher a melhor forma de implementação a ser utilizada e seguir a metodologia de projeto correspondente.

Inicialmente, procurou-se conceituar os diversos tipos de componentes utilizados na implementação de sistemas digitais, enfatizando-se as vantagens de uso dos circuitos integrados de aplicação específica para melhorar o desempenho dos sistemas e conseguir o sigilo de projeto necessário para enfrentar a acirrada concorrência do mercado eletrônico.

Foi visto também que há diversas formas de implementação de ASICs, com características bastante distintas segundo fatores como complexidade, volume de produção, *time-to-market*, desempenho elétrico e custos.

Ambientes de prototipagem rápida estão disponíveis comercialmente para reduzir o tempo de obtenção dos ASICs e reduzir os riscos de projeto existentes. Porém, normalmente eles são específicos para uma determinada forma de implementação de ASICs e dificilmente permitem uma fácil migração entre estas formas, muitas vezes necessária para manter um produto no mercado com preços competitivos. Isto ocorre, por exemplo, quando um ASIC é implementado em componentes PLDs, para o rápido lançamento do produto, e depois este mesmo ASIC é implementado sobre formas personalizáveis por máscaras, com melhor desempenho e menor custo, a partir de determinado volume de produção.

O problema de conversão entre as formas de implementação tem sido avaliado por especialistas que afirmam não ser eficiente conversões automáticas, mas projetos paralelos nas mais diversas metodologias [23]. O trabalho de descrição do circuito e sua validação funcional têm de ser refeitos toda vez que utiliza-se uma nova metodologia de projeto.

Atualmente estão sendo lançadas ferramentas de CAD, baseadas em descrições VHDL, como o AutoLogic da Mentor Graphics [27], que permitem a descrição textual e a validação funcional genéricas antes de escolher a forma final de implementação do ASIC. O ambiente proposto neste trabalho permite também esta descrição e validação genéricas, porém baseadas na captura

esquemática do circuito, a fim de incentivar o seu uso pelos projetistas, mais familiarizados com este tipo de descrição funcional.

O recente lançamente dos PLDs com arquiteturas avançadas (segunda geração), ao contrário do que ocorre com as abordagens *gate array* e *standard cell,* que já são bastantes conhecidos pelos projetistas na área de microeletrônica, levou a realizar um estudo bibliográfico sobre as arquiteturas e elementos de programação encontrados nos componentes programáveis. Este estudo mostrou a variedade de nomenclaturas envolvidas que confundem o usuário no momento de escolher o componentes mais adequado a sua aplicação.

Outro resultado do estudo bibliográfico foi a verificação de que não há ainda uma taxonomia aprovada pela literatura quando se refere a estes componentes. Este ponto é dificultado não apenas pela variedade de nomes utilizados pelos fabricantes, mas principalmente pelo estado evolutivo em que se encontram os PLDs. Devido ao surgimento de novas famílias de componentes, este levantamento bibliográfico deve estar em constante atualização.

Observou-se também neste estudo que os PLDs cresceram muito rapidamente em capacidade lógica e desempenho. Porém, o critério para estimativa desta capacidade lógica utilizado é através de portas equivalentes, que por si só já demonstra uma certa incoerência, uma vez que não há eleboração de leiautes e a manipulação lógica pelas ferramentas de projeto de PLDs impedem esta estimativa.

Foi realizado um trabalho comparativo entre a capacidade lógica da matriz GA2500 (com aproximadamente 2.500 portas equivalentes) e os componentes dos fabricantes Xilinx e Altera. Observou-se que realmente o valor em portas equivalentes fornecidos pelos fabricantes para cada componente não se apresenta como parâmetro mais adequado para estimar a capacidade lógica dos PLDs.

Trabalho semelhante comparando o desempenho elétrico da matriz GA2500 e os PLDs não pôde ser realizado por não ser possível ainda a personalização de ASICs sobre a matriz GA2500, devido à conclusão do laboratório de processo de fabricação onde esta será configurada.

A comparação dos PLDs entre si, em desempenho elétrico e capacidade lógica, já foi solucionada com criação do PREP pelos próprios fabricantes de componentes programáveis. Mas esta comparação não se estende as demais formas de implementação de ASICs.

É possivel implementar estruturas que meçam o desempenho de CIs *standard cell* e *gate array*, mas a avaliação da capacidade continua sendo uma

incógnita. A priori, parece difícil encontrar um critério genérico devido à manipulação lógica realizada pelas ferramentas de PLDs. Particularmente, no caso de matrizes *gate arrays*, estas poderiam participar do PREP implementando os *benchmarks* com as bibliotecas de células de cada fabricante e situando determinada matriz em meio aos PLDs.

A implantação do ambiente proposto encontra-se na dependência da compra da ferramenta MaxPlus2, por parte do Laboratório de Projetos de Circuitos Integrados do CTI, onde ele deverá ser utilizado. Os problemas de compatibilidade para interfacear as ferramentas já estão resolvidos, e o programa de conversão dos esquemáticos encontra-se implementado.

6.1 Trabalhos futuros

Alguns tópicos deixaram margem para desenvolvimento de trabalhos futuros como a avaliação da capacidade lógica da matriz GA2500 considerando a etapa de roteamento que foi ignorada neste trabalho. Para isto, talvez a estratégia mais eficiente fosse tentar obter dos participantes do PREP a descrição esquemática dos *benchmarks* que eles utilizam e implementá-los sobre a matriz, até a fase final de roteamento.

Os *benchmarks* utilizados na comparação apresentada no capítulo 4 são baseados nas descrições em blocos dos circuitos do PREP, mas isto não significa que sejam idênticos. Este fato obrigou a implementá-los também nos ambientes Altera e Xilinx para a realização deste trabalho. O que não aconteceria caso fossem utilizados exatamente os circuitos implementados pelos fabricantes de PLDs para esta comparação.

A obtenção da descrição destes *benchmarks* permite não apenas uma avaliação mais precisa da capacidade lógica da matriz GA2500 em relação aos PLDs, mas também irá permitir futuramente a sua avaliação em desempenho elétrico, situando a matriz sobre os gráficos comparativos divulgados pelo PREP.

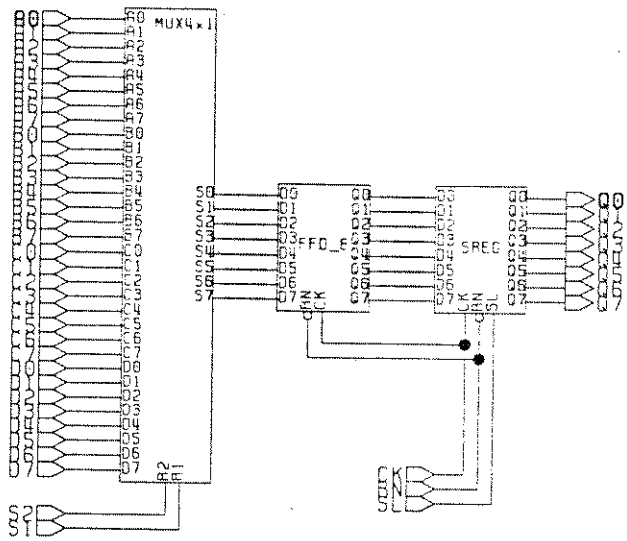
A nível mundial trabalhos de pesquisa têm sido desenvolvidos principalmente sobre a etapa de mapeamento lógico para a configuração dos PLDs [73]. Há trabalhos também para avaliar a estrutura ideal dos blocos lógicos configuráveis e a eficiência dos canais de roteamento [74] [75] [76] [77]. Os primeiros passos em direção a criação de componentes analógicos programáveis têm sido dados, mas ainda a nível acadêmico [78].

No âmbito brasileiro já são vistos trabalhos de pesquisa para desenvolver novas arquiteturas de PLDs [79] [80]. Mas é importante lembrar que para um arquitetura tornar-se viável comercialmente é de vital importância o desenvolvimento conjunto do *software* para sua configuração, caso contrário este trabalho apenas sobre a parte física do componente torna-se ineficiente.

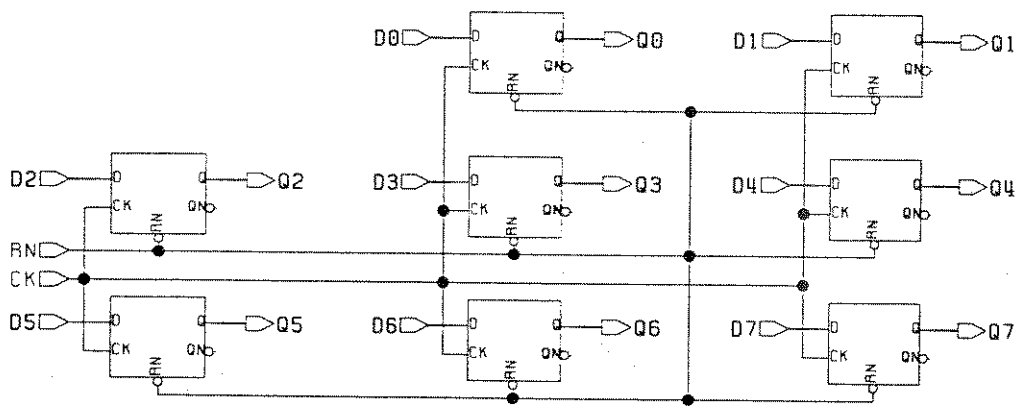
Anexo I

Decrição esquemática dos circuitos *benchmarks* utilizados na avaliação da capacidade lógica dos PLDs

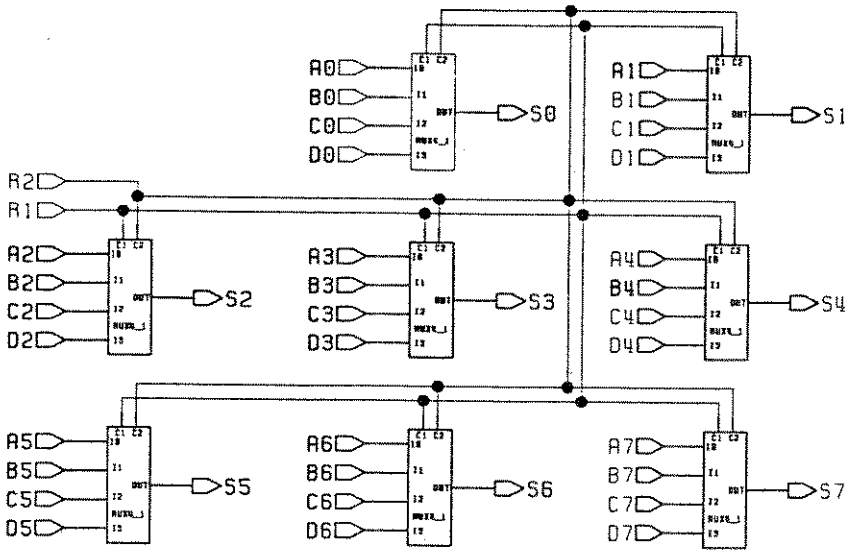
1) *Data Path*



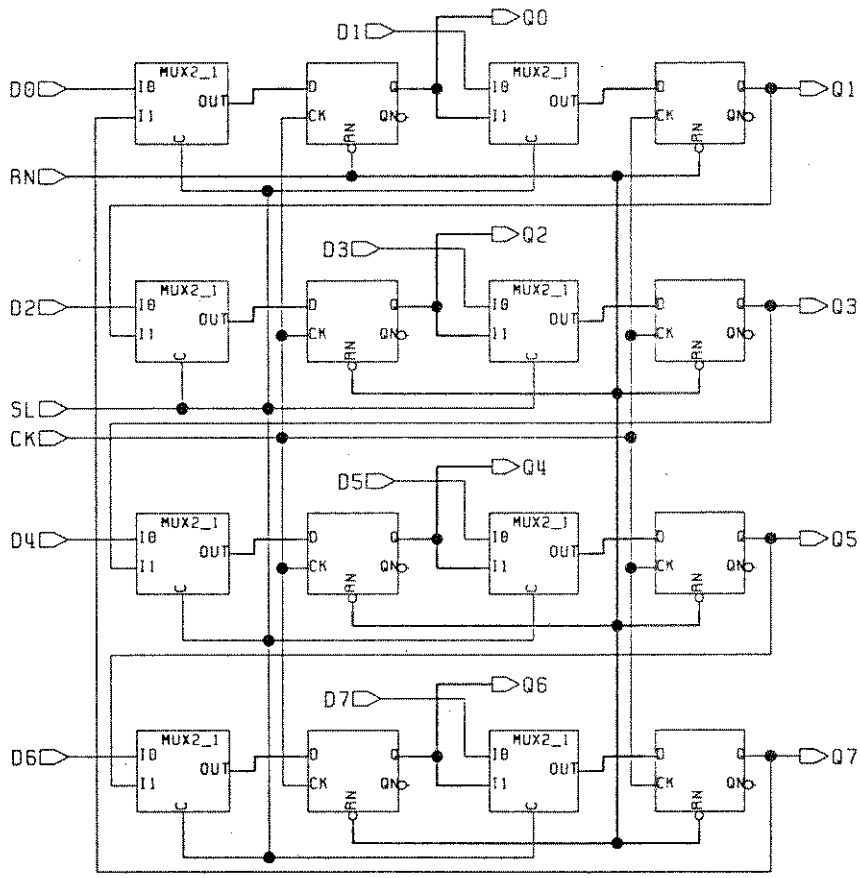
Sub-circuito FFD_8:



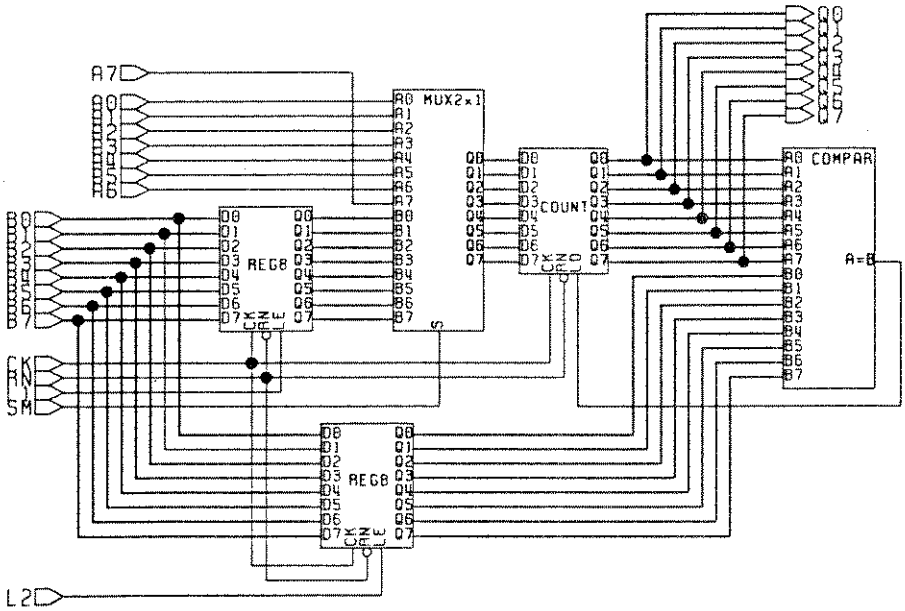
Sub-circuito MUX4x1:



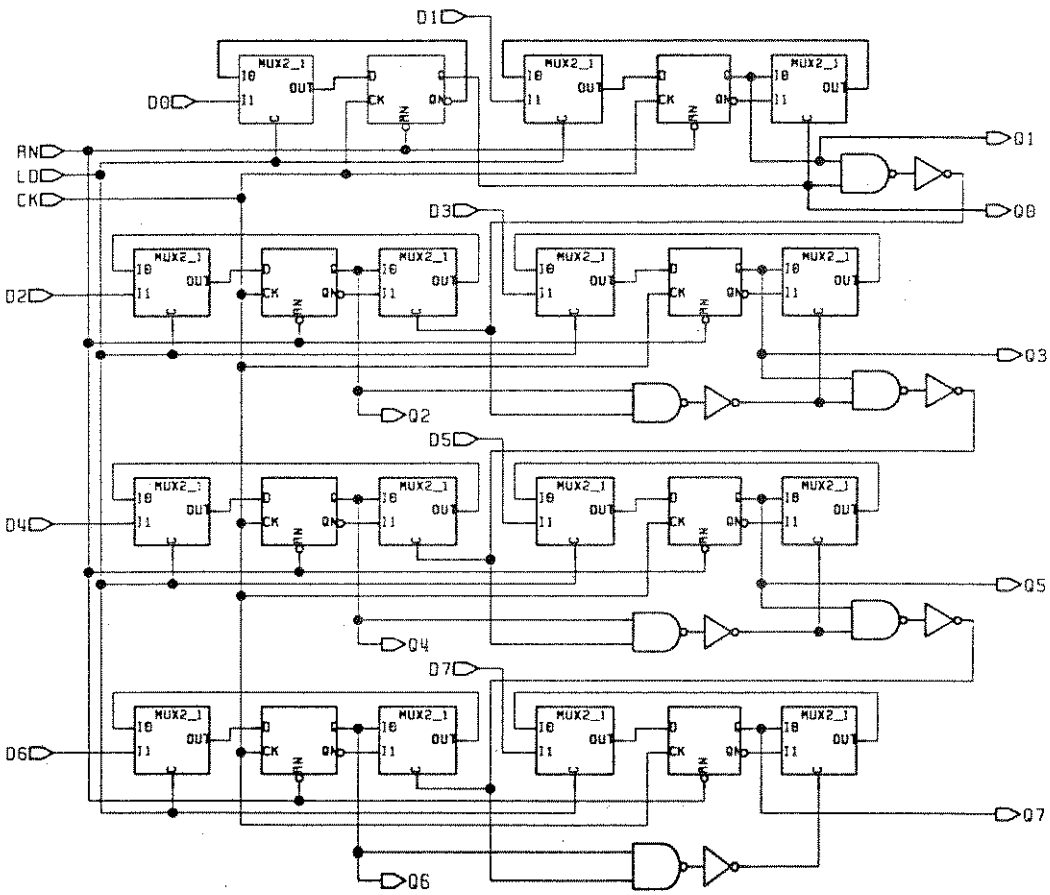
Sub-circuito SREG:



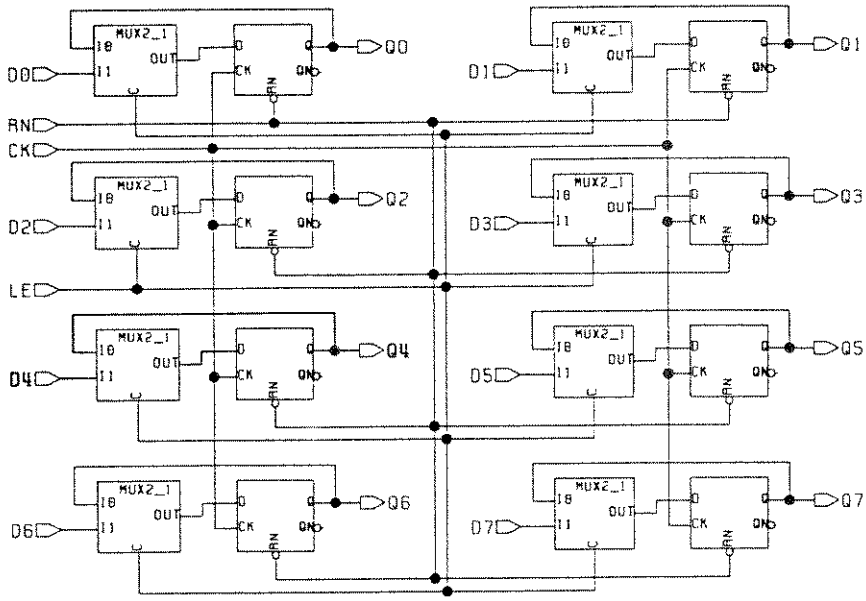
2) *Timer Counter*



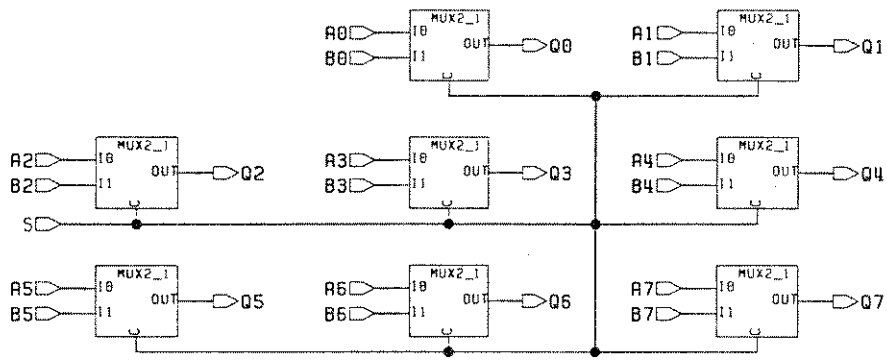
Sub-circuito COUNT:



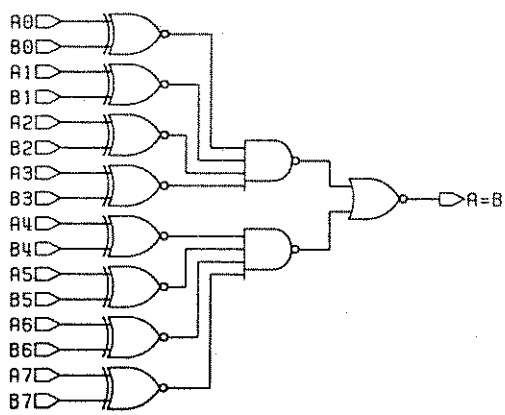
Sub-circuito REG8:



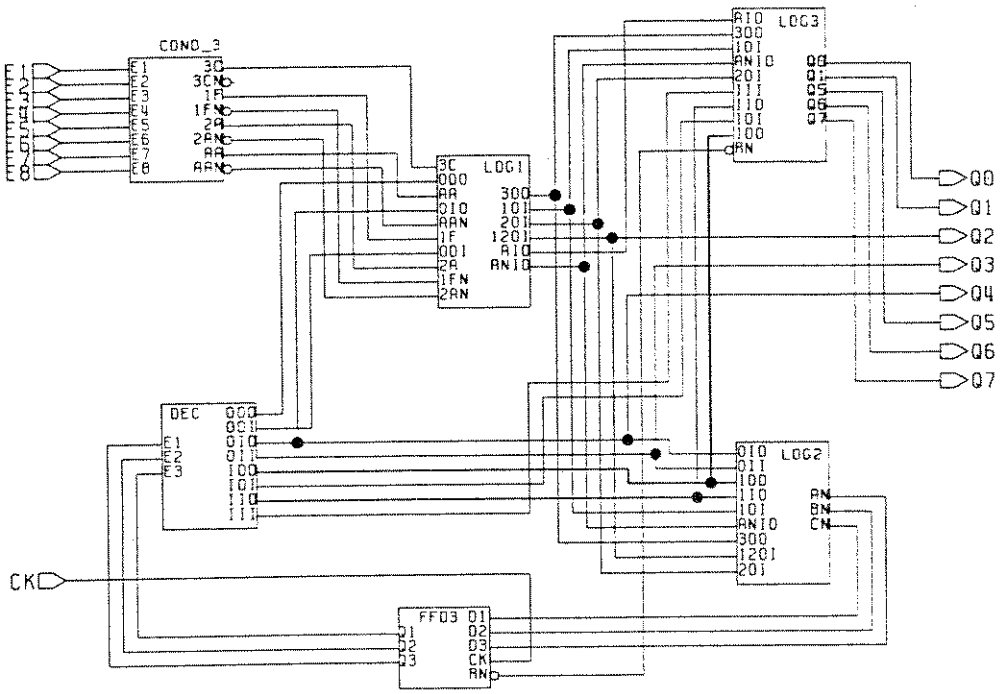
Sub-circuito MUX2x1:



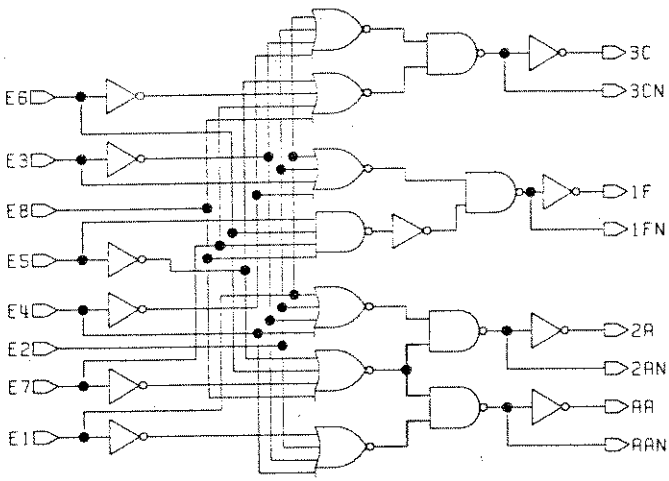
Sub-circuito COMPAR:



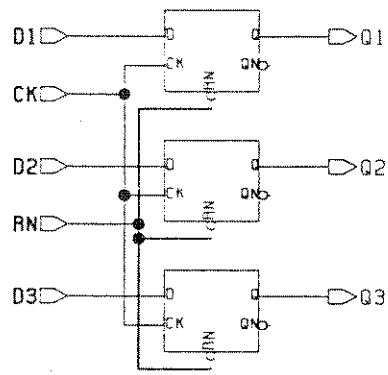
3) Máquina de estado - 8 bits



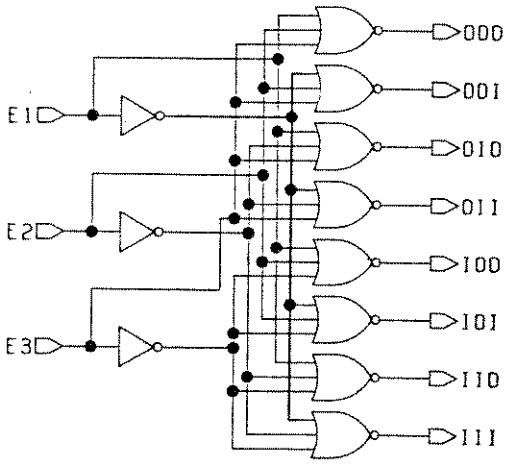
Sub-circuito COND_3:



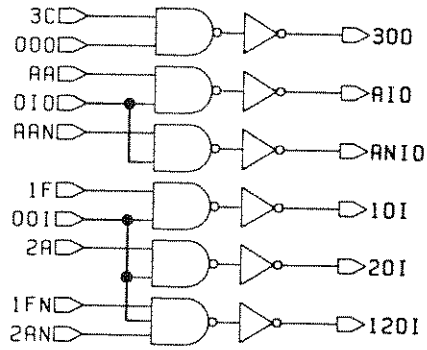
Sub-circuito FFD3:



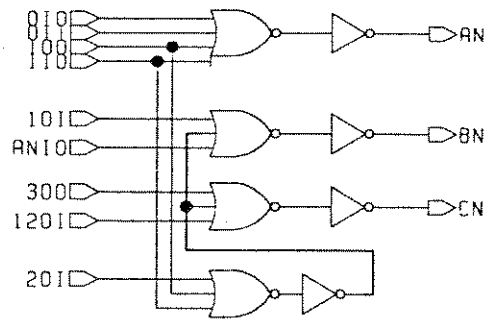
Sub-circuito DEC:



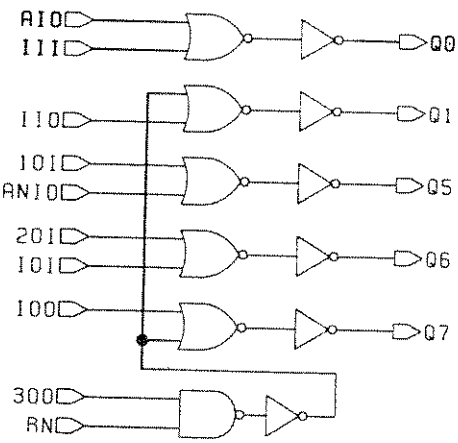
Sub-circuito LOG1:



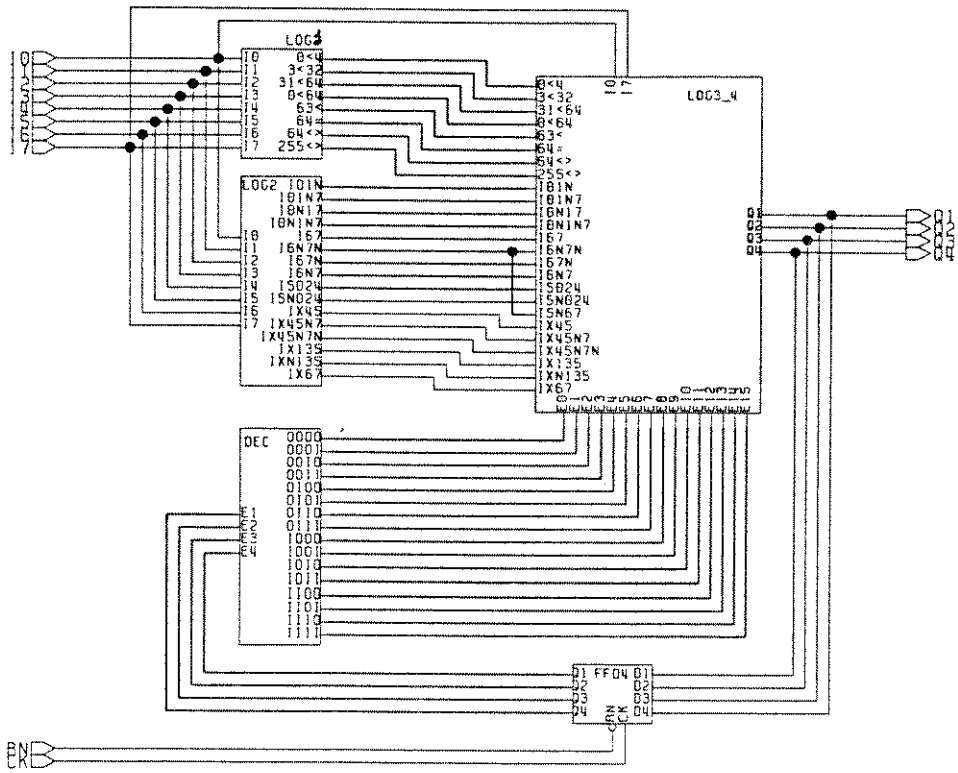
Sub-circuito LOG2:



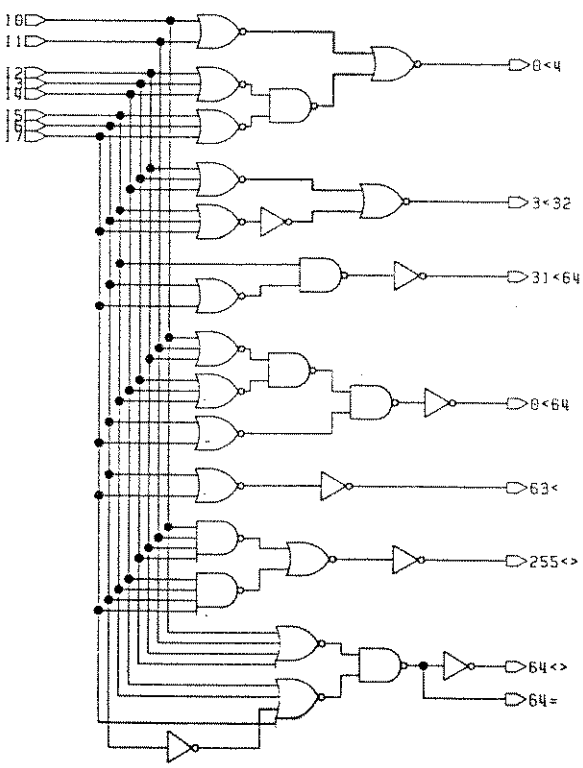
Sub-circuito LOG3:



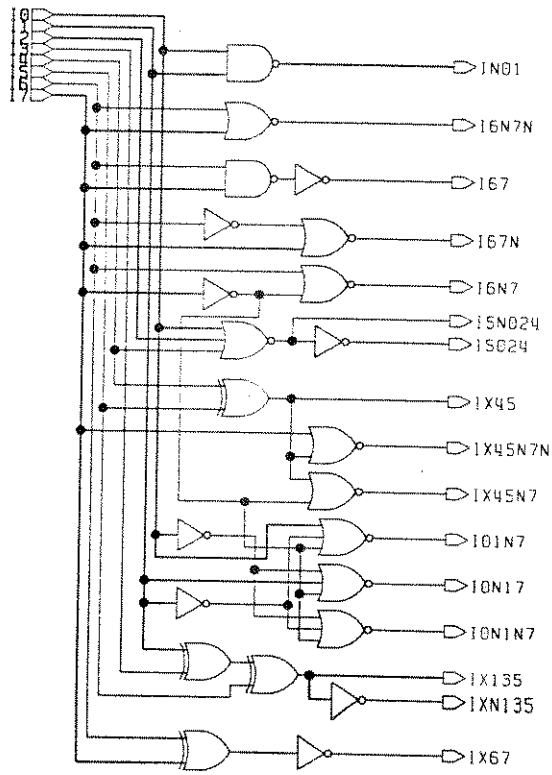
4) Máquina de estado - 16 bits



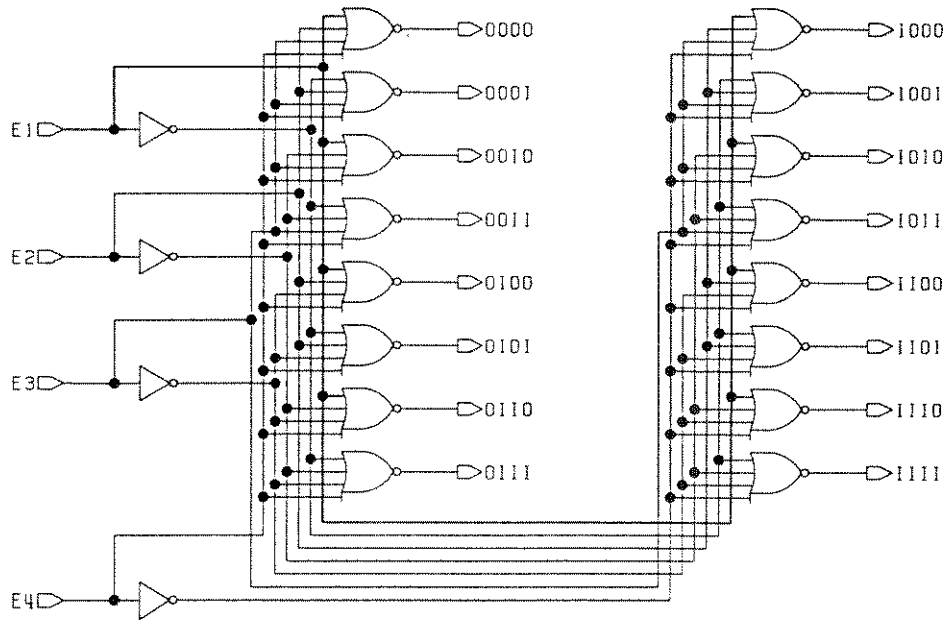
Sub-circuito LOG1:



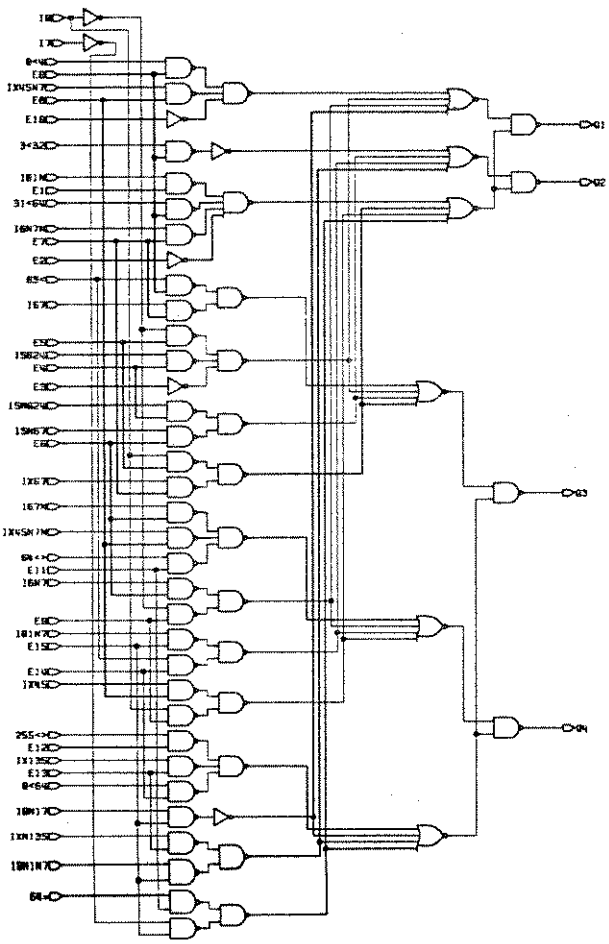
Sub-circuito LOG2:



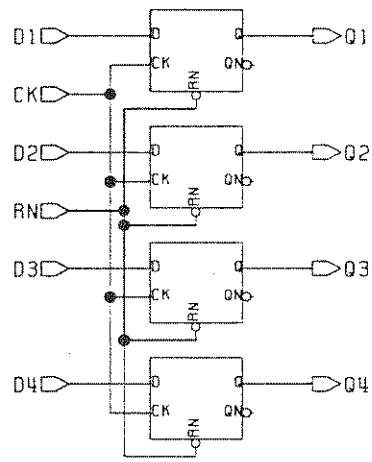
Sub-circuito DEC:



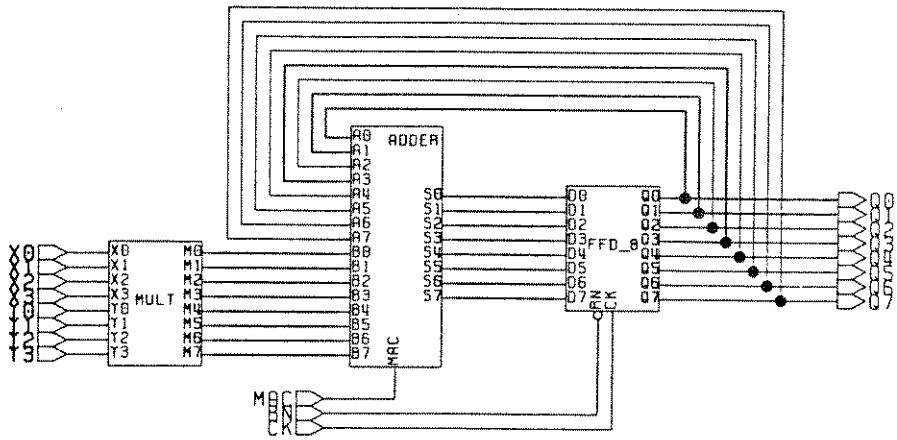
Sub-circuito LOG3_4:



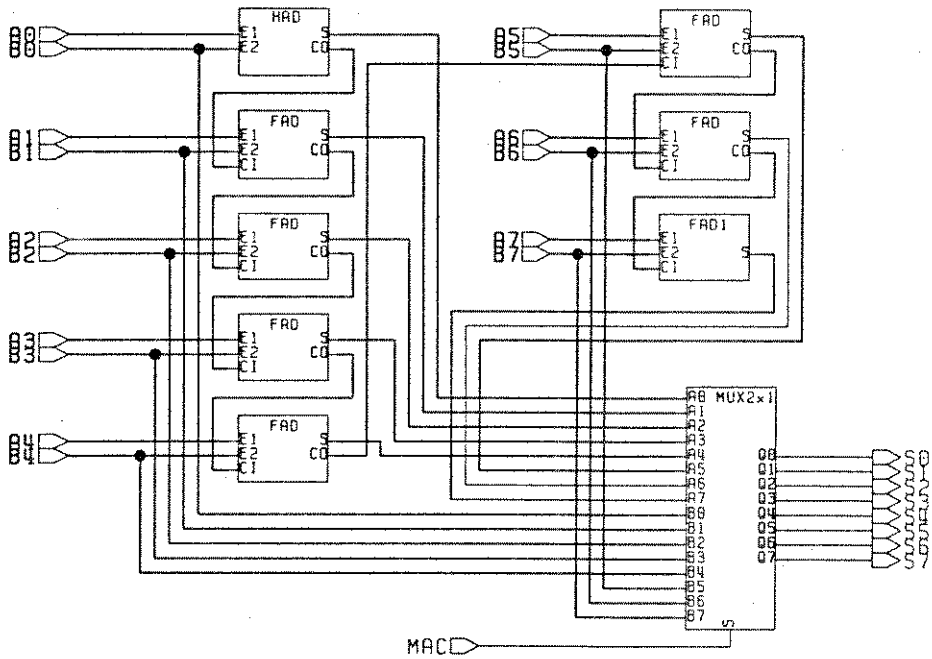
Sub-circuito FFD4:



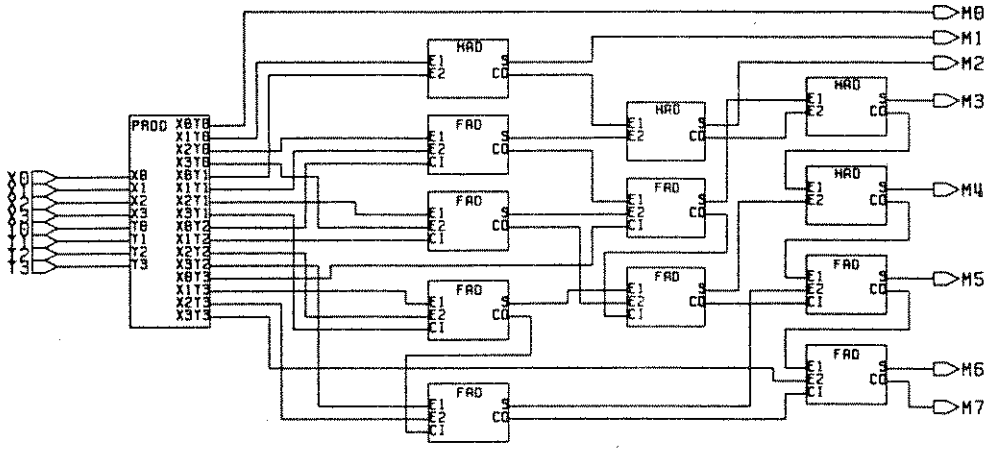
5) Circuito aritmético



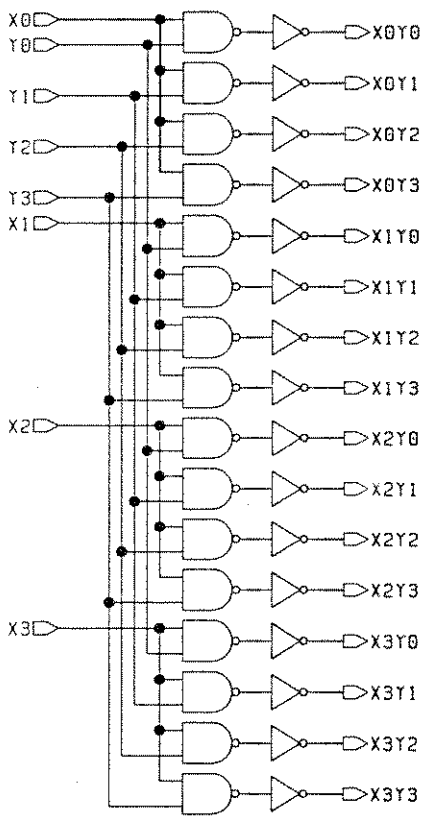
Sub-circuito ADDER:



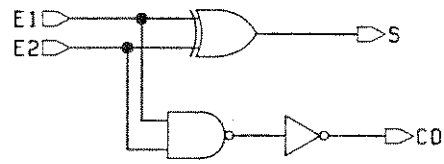
Sub-circuito MULT:



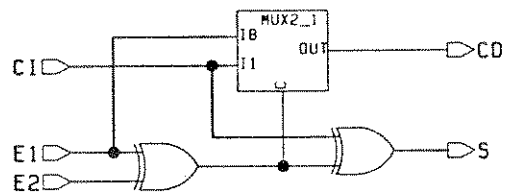
Sub-circuito PROD:



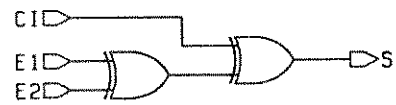
Sub-circuito HAD:



Sub-circuito FAD:

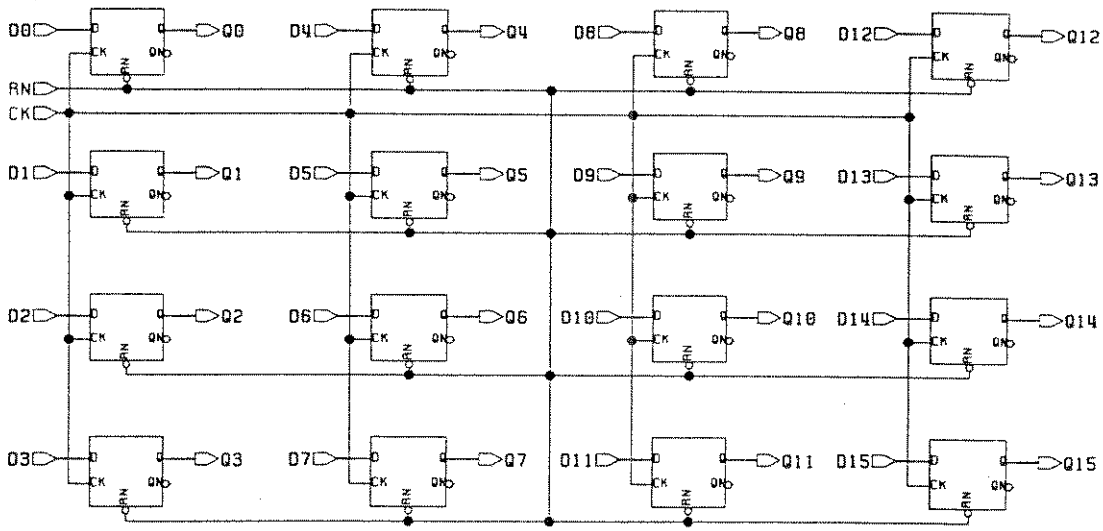


Sub-circuito FAD1:

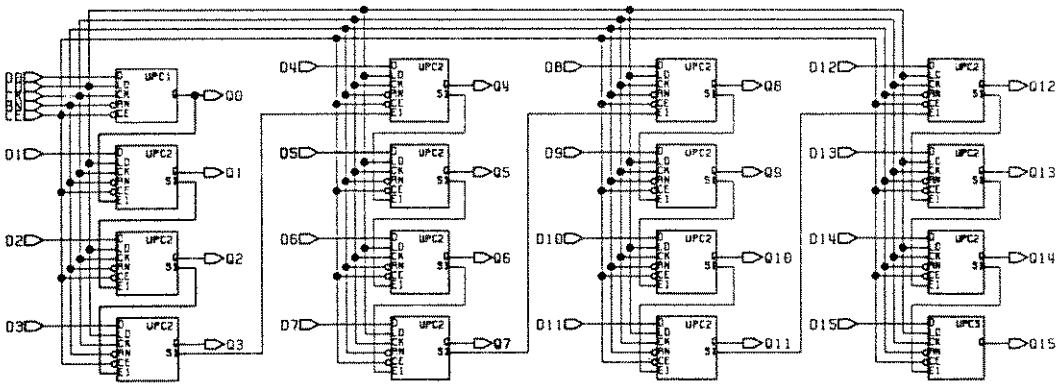


Obs.: O sub-circuito FFD_8 é o mesmo apresentado no *benchmarck Data Path* (1).

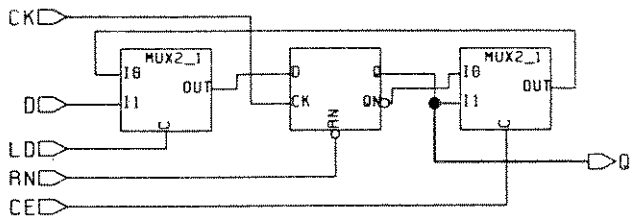
6) Circuito acumulador



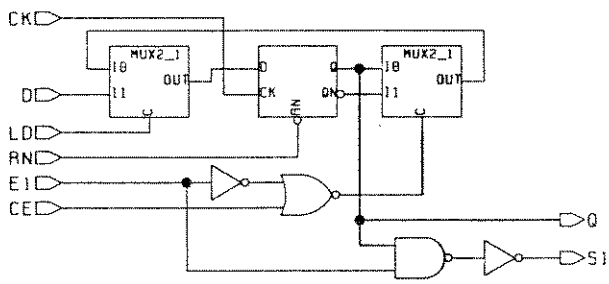
7) Contador *up* - 16 bits



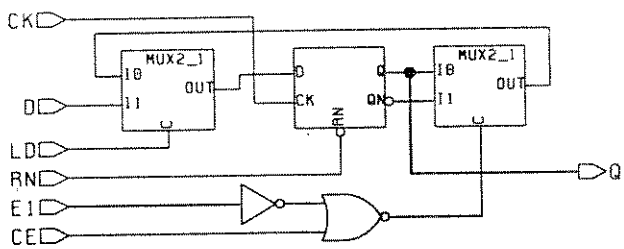
Sub-circuito UPC1:



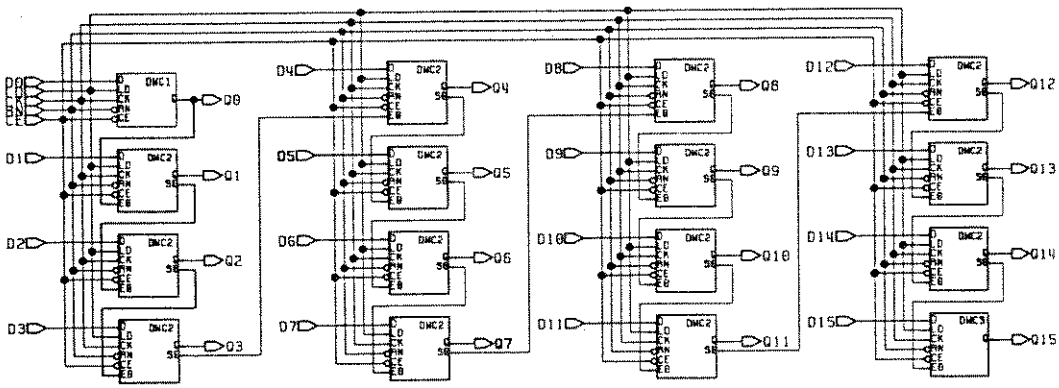
Sub-circuito UPC2:



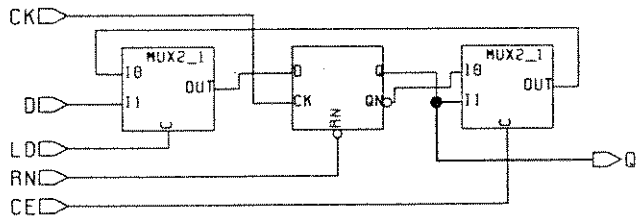
Sub-circuito UPC3:



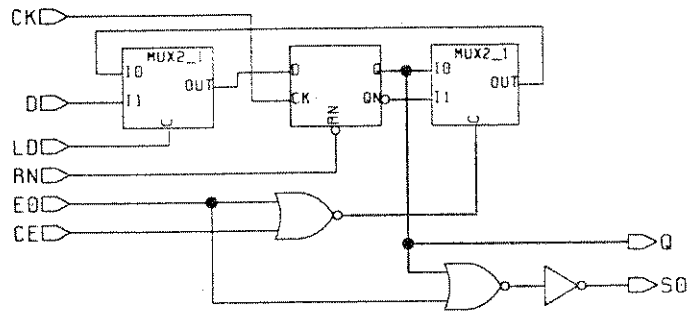
8) Contador *down* - 16 bits



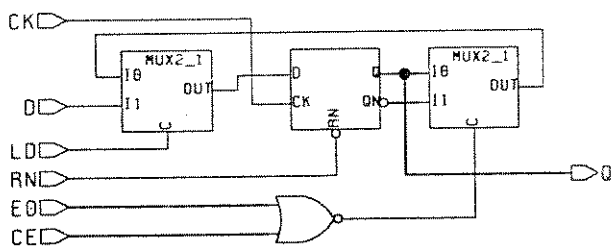
Sub-circuito DWC1:



Sub-circuito DWC2:



Sub-circuito DWC3:



Anexo II

Listagem do programa para alteração do arquivo *sheet1.nrel_$1.ref*

```
/*                                                              */
/* PROGRAMA PARA MODIFICACAO DE BIBLIOTECA UTILIZADA            */
/*                                                              */

# include <stdlib.h>
# include <stdio.h>
# include <string.h>

main (int argc, char *argv[ ])
{
   char aux[100];
   char nome1[100];
   char projeto[100];
   char dirproj[100];
   char extensao[20];
   int j,c,i,ch,termino;

/* verificacao da existencia do diretorio fonte */

   if (!checa_diretorio(argv[1]))
   {
      printf("\nProjeto original nao existente.\n");
      exit(0);
   }

/* separacao da extensao do diretorio fonte <projeto>-<extensao> */

   strcpy(nome1,argv[1]);
   i=strlen(nome1);
   c=0;
   while(nome1[i-1] != '-')
   {
      i--;
      c++;
   }
```

```
   for (j=0;j<c;j++)
      extensao[j] = nome1[i++];
   extensao[j] = '\0';

/* separacao do projeto do diretorio fonte <projeto>-<extensao> */

   termino = i-c-1;
   i = 0;
   while(((nome1[termino - i] != '/')) && ((termino-i) >=0))
      i++;
   for (j=0;j<(i-1);j++)
      projeto[j] = nome1[termino-i+j+1];
   projeto[j] = '\0';
   strncpy(dirproj,nome1,termino);

/* verificacao da existencia do diretorio destino */

   sprintf(aux,"'%s'-'%s'",dirproj,argv[2]);
   if (checa_diretorio(aux))
   {
      printf("\nATENCAO: VERSAO ANTERIOR DO PROJETO %s
ENCONTRADA!!!\n",aux);
      printf("\n(A)bandona ou (C)ontinua ?\n");
      if ((ch = getchar()) != 'c' && (ch != 'C'))
         exit(0);
      apaga_diretorio(aux);
   }

/* copia do projeto original <nome> para <nome>-<gar | std | pld | ...> */

   copia_diretorio(argv[1],aux);

/* apaga versoes anteriores do projeto */

   sprintf(aux,"/idea/com/dlv '%s'-'%s' 1\n",dirproj,argv[2]);
   system(aux);

/* verifica existencia do arquivo sheet*.*.ref no diretorio destino */

   sprintf(aux,"%s-%s",dirproj,argv[2]);
   if (!checa_ref(aux))
   {
      printf("\nNao existe arquivo de referencia no diretorio %s.\n",aux);
      exit(0);
   }
```

```
/* pega nome do arquivo de referencia */

   pega_nome(aux);

/* troca referencias da extensao para referencias do 2o argumento */

   troca_ref(aux,dirproj,projeto,extensao,argv[2]);
   exit(0);
}

int troca_ref(char arq[ ], char dirprojeto[ ], char projeto[ ], char cadeiavelha[ ], char
cadeianova[ ])
{
   char aux[100];
   char aux1[100];
   char novoproj[100];
   int a,i,j,k,l,c,fim,teste,troca,inicio,termino;
   FILE *in, *out;

/* abertura dos arquivos de entrada e saida */

   in = fopen(arq,"rt");
   sprintf(aux,"temporario.%s.txt",projeto);
   out = fopen(aux,"wt");
   fim = 0;
   while (!fim)
   {
/* monta linha do arquivo de entrada na variavel aux */

      i = 0;
      while (((c = fgetc(in)) != '\n') & (c != EOF))
      {
         aux[i] = c;
         i++;
      }
      aux[i] = '\0';

/* testa se chegou ao final do arquivo de entrada */

      if (c == EOF)
         fim = 1;
```

```
/* procura e troca da string cadeiavelha na variavel aux */

    troca = 0;
    for (i=0;i<strlen(aux);i++)
      if ((aux[i] == '/') || (aux[i] == '-'))
      {
        teste = 1;
        j=0;
        while ((j<strlen(cadeiavelha)) & (teste))
        {
          if (aux[i+j+1] == cadeiavelha[j])
             teste = teste * 1;
          else
             teste = teste * 0;
          j++;
        }
        if ((teste) && (aux[i+j+1] == '/'))
        {
          limpa_array(aux1,100);
          if (aux[i] != '-')
          {
            sprintf(aux1,"/%s/",cadeianova);
            troca=1;
          }
          else
          {
            sprintf(aux1,"-%s/",cadeianova);
            troca=0;
          }
          fputs(aux1,out);
          i = i+strlen(cadeiavelha)+1;
        }
        else
        {
          for (k=0;k<=j;k++)
             fputc(aux[i+k],out);
          i = i + k -1;
        }
      }
      else
        fputc(aux[i],out);
    if (fim)
      fputc('\0',out);
    else
      fputc('\n',out);
```

/* se nao foi efetuada troca na linha, chama o programa novamente, pois a linha corresponde a uma estrutura hierarquica */

```
      if (!troca && (strlen(aux)>0))
      {
        l = strlen(aux);
        i = 0;
        while (aux[i++] != '/')
           ;
        inicio = i-1;
        i = 0;
        while (aux[l-i-1] != '/')
          i++;
        termino = l-i-1;
        j=0;
        for (i=inicio;i<termino;i++)
          novoproj[j++] = aux[i];
        novoproj[j] = '\0';
        printf("\n%s\n",novoproj);
        sprintf(aux,"/user/luiz/src/conv.bin '%s' '%s'\n",novoproj,cadeianova);
        system(aux);
      }
   }
   fclose(in);
   fclose(out);
```

/* apaga arquivo original e copia novo arquivo */

```
   sprintf(aux,"/bsd4.3/bin/rm '%s'-'%s'/sheet*.*.ref",projeto,cadeianova);
   system(aux);
   sprintf(aux,"/com/sh cpf temporario.%s.txt '%s'\n",projeto,arq);
   system(aux);
   sprintf(aux,"/bsd4.3/bin/rm temporario.%s.txt",projeto);
   system(aux);
   return (1);
}

int pega_nome(char nome[ ])
{
   char aux[100];
   char aux1[100];
   FILE *arquivo;
   limpa_array(aux1,100);
   arquivo = fopen("temporario.txt","rt");
```

```
    fscanf(arquivo,"%s",aux1);
    fclose(arquivo);
    sprintf(aux,"/bsd4.3/bin/rm temporario.txt");
    system(aux);
    strcpy(nome,aux1);
    return(1);
}

int checa_ref(char ref[ ])
{
    char aux[100];
    char aux1[100];
    FILE *arquivo;
    limpa_array(aux1,100);
    sprintf(aux,"/bsd4.3/bin/ls %s/sheet*.*.ref >temporario.txt",ref);
    system(aux);
    arquivo = fopen("temporario.txt","rt");
    fscanf(arquivo,"%s",aux1);
    fclose(arquivo);
    if (strlen(aux1) > 0)
       return (1);
    else
    {
       sprintf(aux,"/bsd4.3/bin/rm temporario.txt");
       system(aux);
       return (0);
    }
}

int checa_diretorio(char diretorio[])
{
    char aux[100];
    char aux1[100];
    FILE *arquivo;
    limpa_array(aux1,100);
    sprintf(aux,"/bsd4.3/bin/ls -d '%s' >temporario.txt",diretorio);
    system(aux);
    arquivo = fopen("temporario.txt","rt");
    fscanf(arquivo,"%s",aux1);
    fclose(arquivo);
    sprintf(aux,"/bsd4.3/bin/rm temporario.txt");
    system(aux);
    if (strlen(aux1) > 0)
       return (1);
    else
```

```
        return (0);
}

int apaga_diretorio(char diretorio[ ])
{
   char aux[100];
   sprintf(aux,"/bsd4.3/bin/rm -r '%s'",diretorio);
   system(aux);
   return(1);
}

int copia_diretorio(char diretorio1[ ], char diretorio2[ ])
{
   char aux[100];
   sprintf(aux,"/com/sh cpt '%s' '%s'",diretorio1,diretorio2);
   system(aux);
   return(1);
}

int limpa_array(char array[ ], int tamanho)
{
   int i;
   for (i=0;i<tamanho;i++)
      array[i] = '\0';
   return(1);
}
```

Bibliografia

[1] KILBY, j.s. Invention of the integrated circuit. **IEEE Transactions Electron Devices**, Vol. ED. 23. July, 1976. pp. 648-54.

[2] OHR, Stephen. IC, storage & display technologies have turned the world digital. **Computer Design**. January 1, 1991. pp. 52-61.

[3] McCLEAN, J. **ASIC Outlook 1990** - An Application Specific IC Report and Directory. Integrted Circuit Engineering Corp. Arizona, 1990.

[4] Manufacturing - The art of building Pentium Processors. Intel Technology Briefing. Literature packet #74. **Catálogo demonstrativo** divulgado na PC Magazine, vol. 13, no. 5. March 15, 1994.

[5] HURST, Stanley L. **Custom VLSI Microelectronics**. Prentice Hall International Ltd, Hemel Hempstead, 1992.

[6] MEINDL, J.D. Ultra-Large Scale Integration. **IEEE Transactions Electron Devices**, Vol. ED. 31. 1984. pp. 1555-61.

[7] MM5HC / 74HC High-speed CMOS Family. National Semiconductor Corporation. **Data book**, advanced information. Santa Clara, CA, 1981.

[8] High-speed CMOS PC74HC / HCT / HCU Logic Family. Philips Components. **Data handbook**. Book IC06. Netherlands, 1988.

[9] Motorola CMOS Application-Specific Digital-Analog Integrated Circuits. Motorola Inc. **Data book**. Austin, TX, 1990.

[10] -,- Functional flexibility keeps ASICs in demand. **Journal of Electronic Engineering** - JEE (Japan). January, 1993. Vol. 30, No. 313, Dempa Publications. pp. 66-69.

[11]REIS, A.I., GÜNTZEL, J.L. & RIBAS, R.P. Algumas formas de implementação de ASICs. **Anais** do VII Simpósio Brasileiro de Concepção de Circuitos Integrados, pp.15-34, Rio de Janeiro - RJ, setembro de 1992.

[12] GÜNTZEL, José Luís Almada. Geração de circuitos utilizando matrizes de células pré-difundidas. **Dissertação de mestrado**. CPGCC / Instituto de Informática - UFRGS. Porto Alegre, 1993.

[13] REIS, André Inácio. Geração de células transparentes. **Dissertação de mestrado**. CPGCC / Instituto de Informática - UFRGS. Porto Alegre, 1993.

[14] KMETOVICZ, Ron. "...Perspective on Time-to-Market". **Electronic Design**. December 5, 1991. pp 120.

[15] SIMÕES, S.A.; CHAVEZ, F.; RIBAS, R.P.; FINCO, S.; CUIN, M. & BEHRENS, F.H. Matriz Gate Array CMOS avançada, configurável por um único nível de metal. **Anais** do VII Congresso da SBMicro, pp. 281-291, São Paulo - SP, julho de 1992.

[16] LPCI / IM / CTI. Biblioteca de células do gate array GA2500. **Documento interno** 010/93. Instituto de Microeletrônica, Fundação Centro Tecnológico para Informática, 1993.

[17] TONG, Cliff S. A new max EPLD architecture which provides logic density, speed and flexibility. **Journal of Semicustom ICs**, Vol. 7, No. 1. England, 1989. pp 12-19.
[18] Altera Data Book. Altera Semiconductor Corporation. **Data book**. Santa Clara, CA, 1993.
[19] SMITH, M.J.S. ASIC Technologies. IEEE International ASIC Conference and Exhibit, 5. **Proceedings**. September 21-25, 1992. pp 97-105.
[20] The Programmable Gate Array Data Book. Xilinx Inc. **Data book**. San Jose, CA, 1992.
[21] -,- Fujitsu limited markets ASICs for wide variety of applications. **Journal of Electronic Engineering** - JEE (Japan), January 1993. Vol. 30, No. 313, Dempa Publications. pp. 54-76.
[22] EGAN, Barbara T. Designers search for the secret to ease ASIC migration. **Computer Design**, December 1991. pp 78-94.
[23] SMALL, Charles H. FPGA conversion. **EDNASIA**, August 1992. pp. 45-56.
[24] EGAN, B.T. AT&T provides path from FPGAs to arrays. **Computer Design**. January, 1992. pp 117.
[25] BAKER, Marc & COLI, Vince. The PAL22RA10 Story - The Customization of a Standard Product. **IEEE Micro**, 1986. Reprint in PAL Device Handbook, ADM / MMI, 1988. pp 2-683 a 2-698.
[26] Mask-programmed logic devices. Altera Corp., Santa Clara, CA. **Product information bulletin 14**. January 1992 ver. 1.
[27] EGAN, B.T. FPGA synthesis tool has architecture-specific optimizers. **Computer Design**. August, 1991. pp 122.
[28] ROSE, J.; GAMAL, A. & SANGIOVANNI-VICENTELLI, A. Architecture of Field-Programmable Gate Arrays. **Proceedinds** of the IEEE, Vol. 81, No. 7. July, 1993. pp 1013-1029.
[29] COLE, Bernard C. Programmable Logic Devices: The Second Generation. **Electronics**. May 12, 1988. pp 61-63.
[30] BROWN, S.D.; FRANCIS, R.J.; ROSE, J. & VRANESIC, Z.G. **Field-Programmable Gate Arrays**, Kluwer Academic Publishers, 1992.
[31] TRIMBERGER, S. A Reprogrammable Gate Array and Applications. **Proceedinds** of the IEEE, Vol. 81, No. 7. July, 1993. pp 1030-1041.
[32] MAES, H.E. et al. Trends in semiconductor memories. **Microelectronics Journal**, vol. 20, Nos. 1-2. England, 1989. pp 9-58.
[33] CHIANG, Steve et al. Experimental evidence for the morphology of the antifuse. **Electronic Engineering**. June, 1991. pp 45-48.
[34] NEALE, R. A reflow model for anti-fuse. **Electronic Engineering**, April, 1991. pp 31-40.
[35] KAZEROUNIAN, R. & EITAN, B. A Single Poly EPROM for Custom CMOS Logic Applications. In: Custom Integrated Circuits Conference, 1986, IEEE. **Proccedings**. pp 59-62.
[36] HUFF III, R. G. & GOLDBERG, A. EEPROM technology update. **Integrated Circuit Magazine**, Vol. 2, No. 5, 1984.
[37] PLE Handbook. Monolitic Memories Inc. **Handbook**. Santa Clara, CA, 1984.

[38] PAL Device Handbook. Advanced Micro Device - Monolitic Memories. **Handbook**. Sunnyvale, CA, 1988.
[39] Semi-Custom Programmable Logic Devices (PLD). Philips Components - Signetics Corp. **Data handbook**. Eindhoven, Netherlands, 1990.
[40] LEUNG, R. et al. A 7.5ns 350mW BiCMOS PAL - type Device. IEEE Custom Integrated Circuits Conference, 1989. **Proceedings**. pp 5.6.1-5.6.4.
[41] GUDGER, Keith H. PLD Enhancements for Implementing Sub Systems. **VLSI Systems Design**. October, 1986. pp 48-52.
[42] GOWN, Shiva P. et al. A 12ns, CMOS Programmable Logic Device for Combinatorial Applications. IEEE Custom Integrated Circuits Conference, 1989. **Proceedings**. pp 5.5.1-5.5.4.
[43] ALLEN, Michael J. A 6nsec CMOS EPLD with uW Standby Power. IEEE Custom Integrated Circuits Conference, 1988. **Proceedings**. pp 5.1.1-5.1.4.
[44] GREENE, J.; HAMDY, E. & BEAL, S. Antifuse Field Programmable Gate Arrays. **Proceedings** of the IEEE, Vol. 81, No. 7. July, 1993. pp 1042-1056.
[45] The Programmable Gate Array Data Book. Xilinx Inc. **Data book**. San Jose, CA, 1993.
[46] EXEL E^2PROMs Databook. EXEL Microelectronics. **Data book**. San Jose, CA, 1993.
[47] SMALL, Charles H. FPGA Design Methods. **EDN**. August 5, 1991. pp 114-122.
[48] WONG, S.C. et al. A 5000-gate CMOS EPLD with multiple logic and interconnect arrays. In: Custom Integrated Circuits Conference, 1989, IEEE. **Proccedings**. pp 5.8.1-5.8.4.
[49] BURSKY, Dave. FPGA advances cut delays add flexibility. **Electronic Design**. October 1, 1992. pp 35-43.
[50] GAMAL, A. et al. An Architecture for Electrically Configurable Gate Arrays. IEEE Custom Integrated Circuits Conference, 1988. **Proceedings**. pp 15.4.1-15.4.4.
[51] BIRKNER, J. et al. A Very-High-Speed Field-Programmable Gate Array Using Metal-to-Metal Antifuse Programmable Elements. **Microelectronics Journal**, No. 23, 1992. pp 561-568.
[52] EL-AYAT, Khaled A. et al. A CMOS Electrically Configurable Gate Array. **IEEE Journal of Solid-State Circuits**, Vol. 24, No. 3, June 1989. pp 752-756.
[53] GAMAL, Abbas E. et al. An Architecture for Electrically Configurable Gate Arrays. **IEEE Journal of Solid-State Circuits**, Vol 24, No. 2, April 1989. pp 394-398.
[54] CARTER, William S. et al. A User Programmable Reconfigurable Logic Array. In: Custom Integrated Circuits Conference, 1986, IEEE. **Proccedings**. pp 233-235.
[55] QUINNELL, Richard A. FPGA family offers speed, density, on-chip RAM, and wide-decode logic. EDN. December 6, 1990. pp 62-64.

[56] BURSKY, Dave. RAM-Based Logic Arrays up Density, Cut Delays. **Electronic Design**. October 1, 1992. pp 45-49.
[57] BURSKY, Dave. Reprogrammable FPGAs Offer Gate-Array Speed. **Electronic Design**. January 9, 1992. pp 143-146.
[58] HSIEH, Hung-Cheng et al. A 9000-Gate User-Programmable Gate Array. In: Custom Integrated Circuits Conference, 1988, IEEE. **Proceedings**. pp 15.3.1-15.3.7.
[59] JENKINS, J.H. Bridging the Gap Between PLDs and Gate Arrays. **EDS Magazine**. January, 1989.
[60] AGRAWAL, O. & BECK, J. PLDs As Semicustom Substitutes. **VLSI Design**. June, 1985. pp 30-46.
[61] Field Programmable Gate Array Data Manual. Texas Instruments Inc. **Data Book**. Dallas, Texas, 1993.
[62] EGAN, B.T. PLDs catch up to FPGAs in logic capacity and I/O. **Computer Design**. October 1, 1990. pp 44-47.
[63] 2-Micron Gate Array Databook. Austria Mikro Systeme International (AMS). **Data book**. Unterpremstätten, Austria, 1988.
[64] LPCI / IM / CTI. Estruturas e Circuitos de Teste do Projeto GA2500. **Documento interno** 011/93. Instituto de Microeletrônica, Fundação Centro Tecnológico para Informática, 1993.
[65] CHAVEZ, F.; FINCO, S.; ROHWEDDER, J.J.R. & PASQUINI, C. Integrated multichannel voltammetric detection system: design description. **Anais** do VIII Congresso da SBMicro, pp. XII.34-36, Campinas - SP, setembro de 1993.
[66] PREP Benchmarks For Programmable Logic Devices (White Paper). **Documento de divulgação**. Altera Corp., Santa Clara, CA. 1993.
[67] A Systems Perspective on Interpreting the PREP Benchmarks. **Documento de divulgação** da Xilinx, Inc.
[68] Documentação de Projeto do TB45. **Documento interno** do CPqD / TELEBRÁS. Campinas - SP, 1994.
[69] Documentação de Projeto do TB46. **Documento interno** do CPqD / TELEBRÁS. Campinas - SP, 1994.
[70] ANTOGNETTI, P. & MASSABRIO, G. **Semiconductor device modeling with SPICE**. McGraw Hill, 1988.
[71] JAIN, S. K. & AGRAWAL, V. D. Test generation for MOS circuit using D-algorithm. 20th Design Automation Conference, IEEE, 1983. **Proceedings**. pp. 64-70.
[72] CORTNER, J. M. **Digital test engineering.** John Wiley & Sons, 1987.
[73] ERCOLANI, S. & DE MICHELI, G. Technology Mapping for Electrically Programmable Gate Arrays. 28th ACM/IEEE Design Automation Conference. **Proceedings**. 1991. pp 234-239.
[74] ROSE, J. et al. The Effect of Logic Block Complexity on Area of Programmable Gate Arrays. IEEE Custom Integrated Circuits Conference, 1989. **Proceedings**. pp 5.3.1-5.3.5.

[75] ROSE, J. et al. Architecture of Field Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency. **IEEE Journal os Solid-State Circuits**, Vol. 25, No. 5. October, 1990. pp 1217-25.

[76] ROSE, J. & BROWN, S. Flexibility of Interconnection Structures for Field-Programmable Gate Arrays. **IEEE Journal of Solid-State Circuits**, Vol. 26, No. 3. March, 1991. pp 277-82.

[77] SINGH, S. et al. The Effect of Logic Block Architecture on FPGA Performance. **IEEE Journal of Solid-State Circuits**, Vol. 27, No. 3. March, 1992. pp 281-87.

[78] LEE, E.K.F. & GULAK, P.G. Field Programmable Analogue Array Based on Mosfet Transconductors. **Electronics Letters**. January 2, 1992. Vol. 28, No. 1. pp 28-29.

[79] NASCIMENTO, M.C.; MESQUITA, A. & FERNANDES, E.S. Proposta de uma arquitetura configurável por software. **Anais** do VI Simpósio Brasileiro de Concepção de Circuitos Integrados, pp.111-20, Jaguariúna - SP. Setembro, 1991.

[80] SIMÕES, E.V. et al. Projeto e Implementação de uma Célula Lógica Programável do Tipo Field Programmable Gate Array. **Anais** do VII Simpósio Brasileiro de Concepção de Circuitos Integrados, pp.107-120, Rio de Janeiro - RJ, setembro de 1992.

[81] WONG, S. et al. Novel Circuit Techniques for Zero-Power 25-ns CMOS Erasable Programmable Logic Devices (EPLD's). **IEEE Journal of Solid-State Circuits**. Vol. SC-21, No. 5. October, 1986. pp 775-84.

[82] PATHAK, J. et al. A 19-ns 250mW CMOS Erasable Programmable Logic Device. **IEEE Journal of Solid-State Circuits**. Vol. SC-21, No. 5. October, 1986. pp 775-84.

[83] McCLURE, M.D. PLD Breadboarding of Gate Array Designs. **VLSI Systems Designs**. February, 1987. pp 36-41.

[84] SMALL, Charles H. Family tree sorts out high-density PLDs. **EDN**. September 16, 1991. pp 75-80.