UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

AVALIAÇÃO DE DESEMPENHO DE PARTES DE CONTROLE DE CIRCUITOS INTEGRADOS

por

Pedro Inácio Hübscher

Dissertação submetida como requisito parcial
para obtenção do grau de Mestre
em Ciência da Computação

Prof. Altamiro Amadeu Suzim
Orientador

Porto Alegre, maio de 1992.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Hübscher, Pedro Inácio

Avaliação de Desempenho de Partes de Controle de Circuitos Integrados / Pedro Inácio Hübscher. - Porto Alegre: CPGCC da UFRGS, 1992.

110p.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1992. Orientador: Suzim, Altamiro Amadeu.

Dissertação: avaliação de desempenho, partes de controle, síntese de leiaute, circuitos integrados, microeletrônica.

AGRADECIMENTOS

Ao curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Rio Grande do sul.

Ao CNPQ e CAPES pelo auxílio financeiro recebido sob forma de bolsa de estudos.

Ao meu orientador Altamiro Amadeu Suzim, pela paciência e espírito pensador, e pelas experiências transmitidas durante a realização deste trabalho.

Aos professores do grupo de microeletrônica, pelo conhecimento transmitido em aula e através das discussões. Em especial aos professores Sérgio Bampi, Tiarajú Wagner e Ricardo Reis, pelas idéias sugeridas.

Aos colegas do CPGCC e do grupo de microeletrônica, César Marcon, Gilberto Marchioro, Luigi Carro e Fernando Moraes.

SUMÁRIO

LISTA DE SÍMBOLOS E ABREVIATURAS

AHPL	- A Hardware Programming Language
CCITT	- Comitê Consultivo Internacional de Telegrafia e Telefonia
CIF	- Caltech Intermediate Format
CJ	- Capacitância de Junção
CMOS	- Complementary Metal Oxide Semiconductor
Cox	- Capacitância de óxido fino por unidade de área
GME-UFRGS	- Grupo de Microeletrônica da Universidade Federal do Rio Grande do Sul
Gnd	- Ground
HDC	- Hardware Description in C
L	- Comprimento do canal do transistor
MEF	- Máquina de Estados Finitos
μ	- Mobilidade dos portadores em um transistor MOS
PAC	- Projeto Assistido por Computador
PLA	- Programmable Logic Array
ROM	- Read Only memory
SPICE	- Simulation Program with Integrated Circuits Emphasis
TRAGO	- TRANCA Automatic Generator
Vdd	- Tensão de alimentação (5 V)
Vgs	- Tensão gate-source
VLSI	- Very Large Scale Integration
Vt	- Tensão de limiar de condução ("Threshold")
W	- Largura do canal do transistor

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho objetiva o estudo da avaliação de desempenho de partes de controle de circuitos integrados, em relação ao consumo de área em silício e atraso de propagação de sinais.

Para a implementação são adotados dois diferentes estilos de leiaute (PLA e *gate matrix*). Para ambos os casos foi utilizado um conjunto único de regras de projeto.

A análise dos circuitos visando implementação com PLA é feita com base em estimativas de área e atraso deste, sendo definidas as suas células básicas. Para *gate matrix*, é feita a síntese de leiaute com um gerador automático de leiaute para circuitos em lógica aleatória e o atraso é estimado por modelo simplificado.

A avaliação elétrica para calcular o atraso dos sinais é baseada em modelos simplificados de *timing*, previamente estudados, que levam em conta elementos parasitas das redes de transistores.

São analisadas partes de controle de sistemas reais e máquinas de estados finitos hipotéticas. O trabalho visa propor a melhor estratégia de implementação, através da previsão do desempenho dos circuitos, em função do tamanho e complexidade (em número de portas e sinais de interface) do circuito.

PALAVRAS-CHAVE
Desempenho, partes de controle, atraso, síntese de leiaute, circuitos integrados, VLSI.

ABSTRACT

The subject of this work is the performance analysis of control parts of integrated circuits, as a function of silicon area and signals propagation delay.

Two different layout styles are used for implementation (PLA and gate matrix). Both of them use the same design rules.

The analysis of the circuits implemented with PLA is based on area and delay estimation, with the basic cells already defined. For gate matrix, the layout synthesis is made with an automatic layout generator for random logic circuits and the delay is estimated by simplified models.

The electrical evaluation to compute the delay signal is based on simplified timing models, previously studied, taking into account parasitic elements of the transistor networks.

Control parts of real systems and finite state machines are analysed. This work aims to select the best implementation strategy, based on performance estimation, as a function of the size and complexity (gates and interface signals) of the circuit.

KEY WORDS
Performance, control parts, delay, layout synthesis, integrated circuits, VLSI.

1. INTRODUÇÃO

A atual complexidade dos circuitos integrados exige ferramentas de PAC (Projeto Assistido por Computador) para a implementação de sistemas digitais em VLSI. Partindo da definição de compilador de silício, um sistema de PAC ideal, não disponível atualmente, geraria as máscaras para fabricação, a partir de uma descrição do sistema a ser implementado em uma linguagem de alto nível. Assim, o projeto de um circuito integrado passa por uma sequência de transformações, em diferentes níveis de abstração, até a sua obtenção em silício: nível funcional, nível lógico, nível físico.

O projeto funcional é geralmente auxiliado por linguagens de descrição de hardware, como VHDL [STE 90], AHPL [HIL 73] e HDC [SUZ 89], que permitem obter uma especificação simulável a nível funcional. A simulação serve para certificar se a especificação do sistema está correta.

A representação funcional é então transformada em uma descrição lógica, que consiste em um *net list* de portas lógicas, incluindo elementos de memória. Nesta fase entram os programas de minimização lógica. Feita a minimização lógica, passa-se para o nível físico. A minimização lógica busca a redução do número de dispositivos a serem interconectados e, portanto, redução em área.

O projeto físico envolve a alocação e interconexão dos dispositivos. A representação final é o leiaute das máscaras usadas para fabricar o circuito. Este depende do processo de fabricação e das regras de projeto, e é descrito em uma linguagem geométrica, como CIF (Caltech Intermediate Format).

Vários estilos de leiaute para sistemas VLSI são

encontrados (*full-custom*, *gate array*, *standard-cell*, ROM, PLA). A opção por um ou outro vai responder a restrições de desempenho (área, velocidade) e vai depender da aplicação específica do circuito que está sendo projetado, bem como de ferramentas computacionais de auxílio ao projeto (programas) disponíveis.

A implementação pelo uso de ROMs, embora forneça um projeto bastante regular, facilitando assim a automação, apresenta desvantagens quanto à área de silício e tempo de propagação dos sinais (da entrada do decodificador até a saída da ROM).

O uso de *standard-cell* mostra-se vantajoso na maioria das vezes, em termos de área e velocidade, embora os algoritmos para síntese automática sejam mais complexos.

PLAs (Programmable Logic Arrays) apresentam vantagens sobre ROMS, pelo fato de as últimas serem muito lentas e ocuparem mais área, pois necessitam de um decodificador na entrada.

O objetivo da dissertação é empregar estilos variados (PLA, portas lógicas) de leiaute para implementação de sistemas digitais, podendo-se fazer uma estimativa de desempenho dos últimos, em termos de área consumida e atraso de propagação de sinais. Ao final do trabalho são obtidos dados que visam contribuir na escolha da melhor alternativa de implementação, a partir do número de portas e sinais de interface de um circuito. Os sistemas digitais avaliados são sistemas sequenciais síncronos constituindo-se em partes de controle que enviam sinais para partes operativas. Uma parte de controle pode ser representada por uma MEF (Máquina de Estados Finitos), e é constituída principalmente de uma lógica combinacional (PLA, ROM, portas lógicas) e elementos de memória (flip-flops, registradores).

1.1 Apresentação

O capítulo 2 faz uma rápida classificação dos sistemas digitais, situando máquinas de estados finitos e unidades de controle, e as principais formas de implementação em circuitos integrados.

Os principais estilos de projeto em VLSI para implementar sistemas digitais, são revisados no capítulo 3.

No capítulo 4, são mencionados simuladores de atraso para circuitos integrados, e pesquisados os principais modelos simplificados de obtenção de atraso em redes de transistores CMOS. Tais modelos se aplicam para obtenção de uma estimativa rápida do desempenho elétrico, para circuitos implementados em PLAs e com portas lógicas.

O capítulo 5 apresenta estimativa de desempenho para um PLA, e um gerador para circuitos em lógica aleatória, usado para comparação.

No capítulo 6, é feita a avaliação de desempenho, para a parte de controle de um sistema digital para equalização adaptativa usando PLA, que é comparado com um leiaute em lógica aleatória sintetizado com o gerador descrito no capítulo 5. A estimativa de atraso de sinais é obtida com os modelos descritos no capítulo 4.

Os capítulos 7 e 8 apresentam a mesma análise, para um pequeno circuito de controle de um multiplicador, um somador serial e uma máquina de estados finitos hipotética.

2. SISTEMAS DIGITAIS E UNIDADE DE CONTROLE

2.1 Sistema Digital

Um sistema digital pode ser definido como uma função, que para um determinado vetor discreto X (x_1, x_2, ..., x_n), fornece um vetor Z, composto pelos sinais discretos (binários) z_1, z_2, ..., z_m, tal que Z = f(X) (Figura 2.1). Pode haver inúmeros sistemas digitais que realizam a mesma função, e que são portanto equivalentes. Isto provém de diversos fatores, como topologia, estratégia de implementação e tecnologia.

A implementação do sistema digital pode ser realizada mediante a utilização de duas dimensões: temporal e espacial. O uso de uma ou outra, dependerá da função Z = f(X), e de ítens dispendiosos como área, velocidade e consumo. A partir disso, os sistemas (circuitos) digitais podem ser divididos em dois grupos: combinacionais e sequenciais (Figura 2.2).

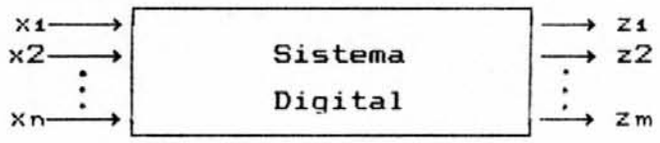


Figura 2.1 - Sistema digital.

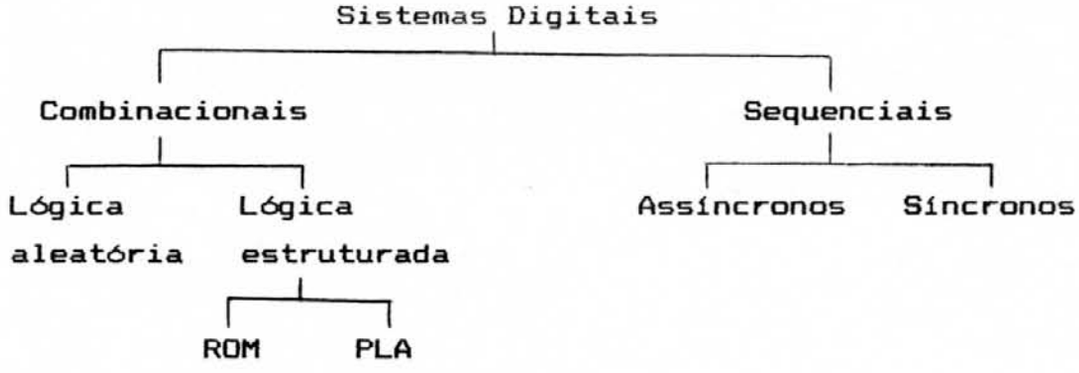


Figura 2.2 - Divisão dos sistemas digitais.

Nos sistemas combinacionais, a saída é uma função exclusivamente das entradas presentes. Portanto, não há memória que armazena informação, e não existe um laço de realimentação entre as saídas e as entradas. As saídas são defasadas das entradas devido somente aos atrasos relacionados aos dispositivos físicos do circuito.

Nos sistemas sequenciais (Figura 2.3), além da lógica combinacional, há uma capacidade de armazenamento (representada pelo vetor S), permitindo criar caminhos de realimentação. Assim, as saídas não são função apenas das entradas presentes, mas também dependem do valor das entradas passadas. Os caminhos de realimentação irão determinar o número de estados do sistema sequencial, e são estes que memorizam a informação para sequência das operações envolvidas. Assim, a partir das informações (X,S_i) de um dado instante (ciclo de relógio), produz-se um conjunto de saídas (Z) e um vetor S_{i+1} (próximo estado).

De forma geral, $(Z_1,S_1) = f(X_1,S_0)$, $(Z_2,S_2) = f(X_2,S_1)$ e $(Z_n,S_n) = f(X_n,S_{n-1})$, onde X_i são as entradas no instante i, Z_i as saídas no mesmo instante, S_i é o estado atual, S_{i-1} o estado anterior e $f(X_i,S_{i-1})$ as operações a serem executadas.

Figura 2.3 - Sistema sequencial.

Circuitos sequenciais onde as saídas são funções apenas do estado, são chamados circuitos Moore. Nestes, as mudanças

nas entradas devem primeiro influenciar no estado, para haver uma variação na saída. Quando as saídas são funções tanto do estado quanto das entradas, se trata de um circuito Mealy.

Sistemas sequenciais implementados sob a forma de circuitos Mealy, apresentam em geral menor número de estados e a vantagem de uma implementação mais econômica. Implementações com circuitos Moore, por outro lado, permitem acompanhar mais facilmente a operação do sistema através do sequenciamento dos estados, e tem a vantagem de as entradas não perturbarem as saídas antes da ocorrência do ciclo de relógio.

2.2 Máquinas Sequenciais

Os sistemas digitais sequenciais formam as chamadas máquinas sequenciais, ou máquinas de estados finitos [HIL 81]. Um sistema sequencial genérico, de acordo com o modelo de Mealy, está mostrado na figura 2.4. Neste caso, o sistema está sendo acionado apenas por uma fase.

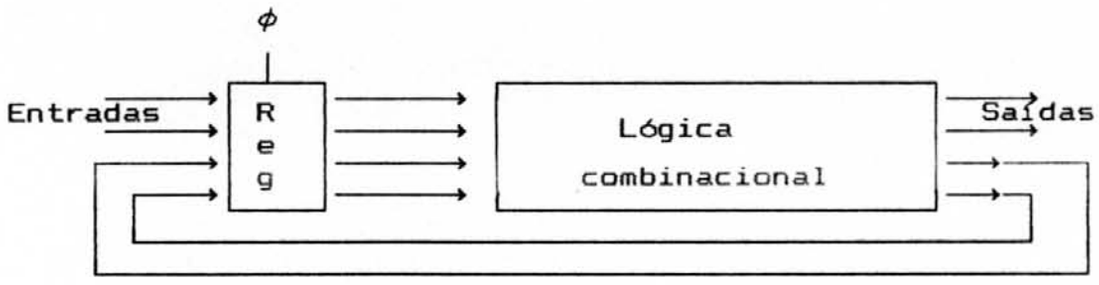


Figura 2.4 - Sistema sequencial com uma fase.

O sinal de relógio de uma fase, aparece na figura 2.5. O sinal apresenta as seguintes características:

Ta - tempo em que a fase ϕ está em 1;

Tb - tempo em que a fase ϕ está em 0;

Ta + Tb - período do sinal ou fase ϕ.

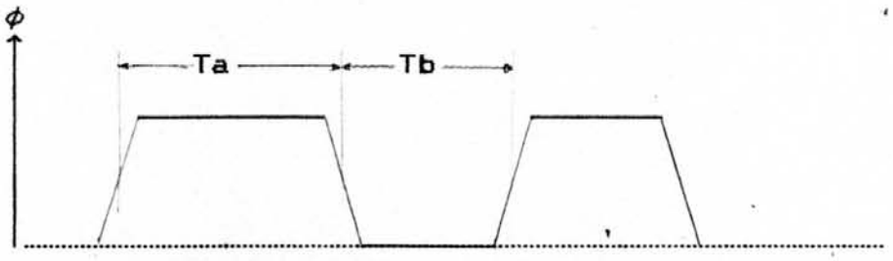


Figura 2.5 - Uma fase de relógio.

Os valores da frequência de operação, 1/(Ta + Tb), e dos tempos Ta e Tb, não podem ser escolhidos livremente. Para um funcionamento correto, a forma de onda do relógio, deve ser tal que:

- Ta < Trápido
- Ta + Tb > Tlento

onde Trápido é o menor atraso através da lógica combinacional (mais o atraso no registrador, se usado flip-flop), e Tlento, é o tempo que o sinal mais lento leva para se propagar através da mesma lógica.

Se Ta for maior que Trápido, os sinais se propagam de volta à lógica, resultando em erro, se o registrador for implementado, por exemplo, com flip-flops sensíveis ao nível, ou transistores de passagem. Se Tlento for maior que Ta + Tb (tempo de ciclo), o valor calculado para o estado seguinte chega muito tarde para ser armazenado.

O problema da propagação dos sinais mais rápidos, pode ser resolvido pelo uso de flip-flops sensíveis à borda, ou pelo uso de um segundo elemento de armazenamento (Figura 2.6), que bloqueie a realimentação dos sinais mais rápidos, e pelo uso de duas fases, Φ1 e Φ2 (Figura 2.7), acionando os registradores R1 e R2.

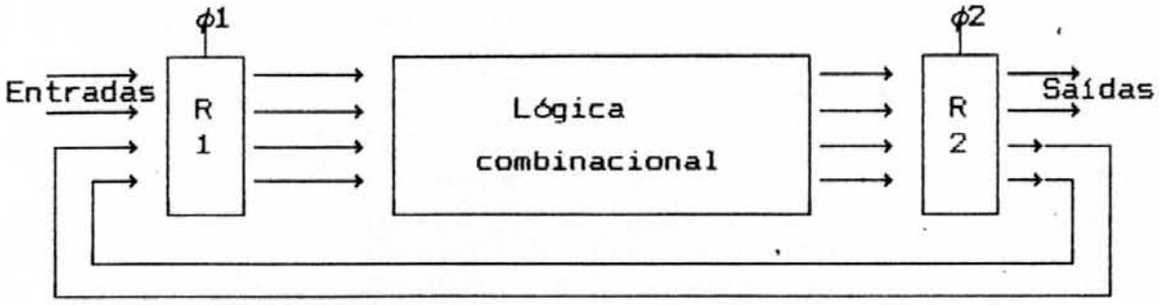


Figura 2.6 - Sistema sequencial com duas fases.

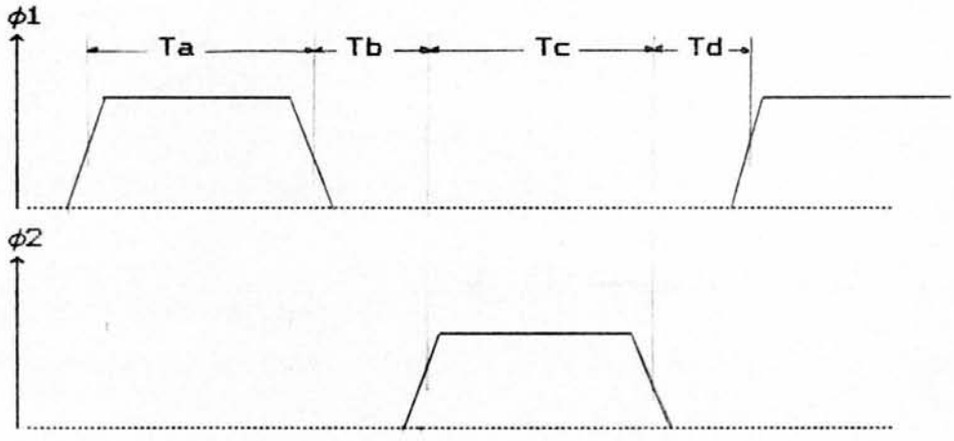


Figura 2.7 - Fases $\phi 1$ e $\phi 2$.

O valor do estado presente, é amostrado durante Ta, em R1. Após um intervalo de garantia, Tb, o valor do estado seguinte é armazenado no registrador R2, na subida de Tc.

Assim, um sinal muito rápido não provocará erro, pois quando os sinais do estado seguinte são liberados na subida de Tc, estes não estarão mais sendo copiados para o registrador R1. Então, a única restrição de tempo será:
$T_{lógica1} < T_a + T_b$ e $T_{lógica2} < T_c + T_d$
sendo Tlógica1 o maior atraso na lógica combinacional e R1, e Tlógica2 o atraso em R2. Na prática, usa-se $T_a = T_c$ e $T_b = T_d$.

O importante é que estas estratégias criam uma barreira

temporal segura de forma a oferecer um suporte para o projeto de máquinas sequenciais. Deparam-se assim o aspecto temporal (atraso dos sinais) e o aspecto lógico (evolução dos estados da máquina).

2.3 Unidade de Controle

Uma unidade de controle pode ser representada por uma máquina de estados finitos. Sendo o sistema sequencial que implementa a máquina de estados finitos uma combinação de circuitos combinacionais e memória, a parte (ou unidade) de controle genérica pode ser representada por circuitos combinacionais e memória.

A parte de controle (bloco de controle) é uma máquina sequencial que gera os sinais para cadenciar as operações, fornecendo, nos instantes adequados de tempo, os comandos que ativam a parte operativa (Figura 2.8). Dependendo dos sinais enviados pela parte operativa em resposta aos comandos recebidos, e de outras entradas externas, o controle sequencial vai para um ou outro estado, para iniciar outras operações.

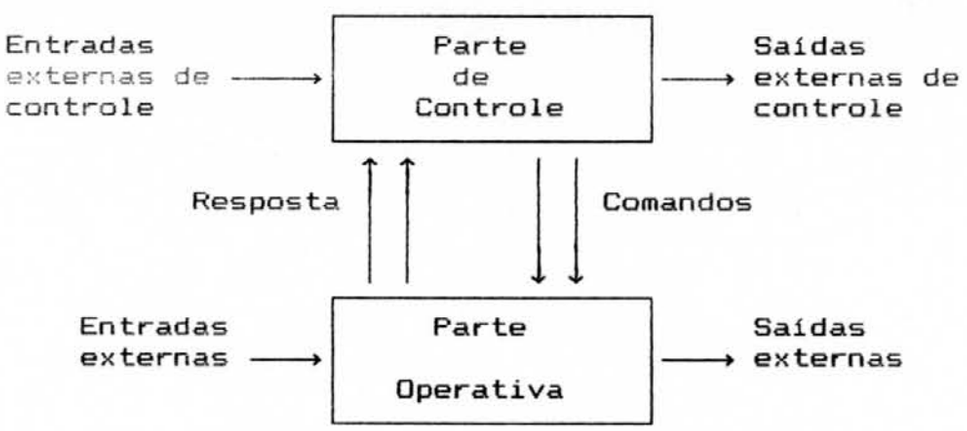


Figura 2.8 - Parte Controle e Parte Operativa.

Na unidade operativa são executadas funções, como adição, subtração, deslocamentos e funções booleanas. Para o caso de microprocessadores, é constituída de estruturas com registradores, unidades lógico-aritméticas e unidades de deslocamento.

O projeto funcional da unidade de controle requer inicialmente uma definição das micro-operações e o seu sequenciamento, e a especificação dos sinais de controle que ativarão a parte operativa.

No projeto lógico, é mapeada a descrição funcional em termos de variáveis lógicas. O projeto lógico inclui otimizações nas variáveis e minimização de estados, visando melhor desempenho elétrico e redução de área de silício.

O projeto, funcional e lógico da unidade de controle, segue o da parte operativa, e depende da opção escolhida para a última. Otimizações na parte operativa podem influir na parte de controle, e idealmente o projeto deve seguir um processo iterativo. Uma unidade operativa mais simplificada pode implicar em uma unidade de controle mais complexa e vice-versa.

O projeto físico depende do estilo de leiaute adotado. A implementação pode ser com portas lógicas (conhecido como lógica anárquica ou aleatória) ou com estruturas regulares como ROMs e PLAs (lógica estruturada). A escolha entre uma forma ou outra leva em conta restrições de desempenho impostas pelo projeto. O uso de formas regulares como ROMs é herdado dos conceitos de microprogramação, onde os vetores de controle estão armazenados na memória, e a unidade de controle é dita microprogramada. No nível de VLSI, o uso de portas lógicas permite, geralmente, uma operação mais rápida, e em relação a este fator, a escolha por esta opção torna-se preferível, desde que existam ferramentas de síntese

automática, principalmente para unidades de controle maiores.

Dentro da lógica estruturada, a opção entre ROM com microprograma e PLA é relativa, mas de uma forma geral a ROM de microprograma é usada em sistemas controladores grandes e complexos (os microprocessadores Motorola 68030 e Intel 80386 usam este tipo de controle), e PLAs são usados em máquinas menores e de menor complexidade.

3. ESTILOS DE PROJETO

Este capítulo revisa os principais estilos de projeto, que são normalmente usados para sintetizar sistemas digitais em VLSI. Inicialmente é feita uma comparação entre lógica randômica e lógica estruturada.

São discutidas diversas estruturas, como ROM, PLA, *gate array*, *Weinberger array* e *gate matrix*, relacionando-se as principais vantagens e desvantagens.

3.1 Lógica Estruturada e Lógica Randômica

O termo lógica randômica (aleatória, anárquica) descreve um estilo particular de projeto, e é usado para caracterizar circuitos que não possuem regularidade na colocação em silício dos dispositivos e interconexões. Lógica estruturada, por outro lado, é o termo usado para caracterizar formas que apresentam regularidade no leiaute e nas interconexões.

Muitos circuitos integrados no passado foram projetados manualmente com o uso de lógica aleatória. Os antigos microprocessadores, como o Intel 8080 e Motorola 6800, continham grandes seções de lógica randômica. No Motorola 6809 (Figura 3.1), 50% da área é em lógica aleatória [GEI 90]. Projetos deste tipo geralmente apresentam vantagens em relação à área de silício, principalmente se o leiaute final for compactado, e permitem a obtenção de circuitos com melhor desempenho elétrico. No entanto, apresentam dificuldades para teste e para modificações posteriores no leiaute.

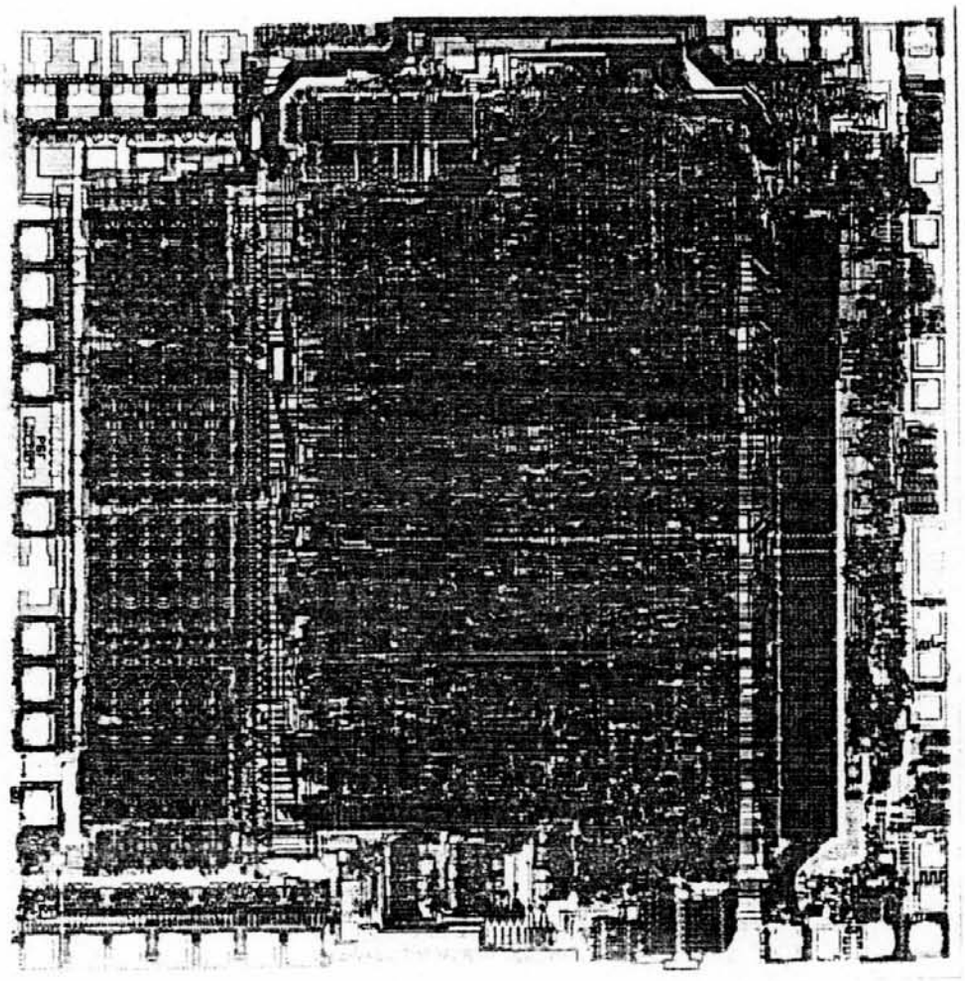


Figura 3.1 - Motorola 6809.
Extraído de [GEI 90]

Mais recentemente, circuitos foram projetados usando estruturas regulares, como por exemplo, os microprocessadores Intel 80386 (Figura 3.2) e Motorola 68030, que utilizam uma ROM que contém um microprograma para o sequenciamento das instruções. Nestes "chips", a área projetada em lógica aleatória é drasticamente reduzida.

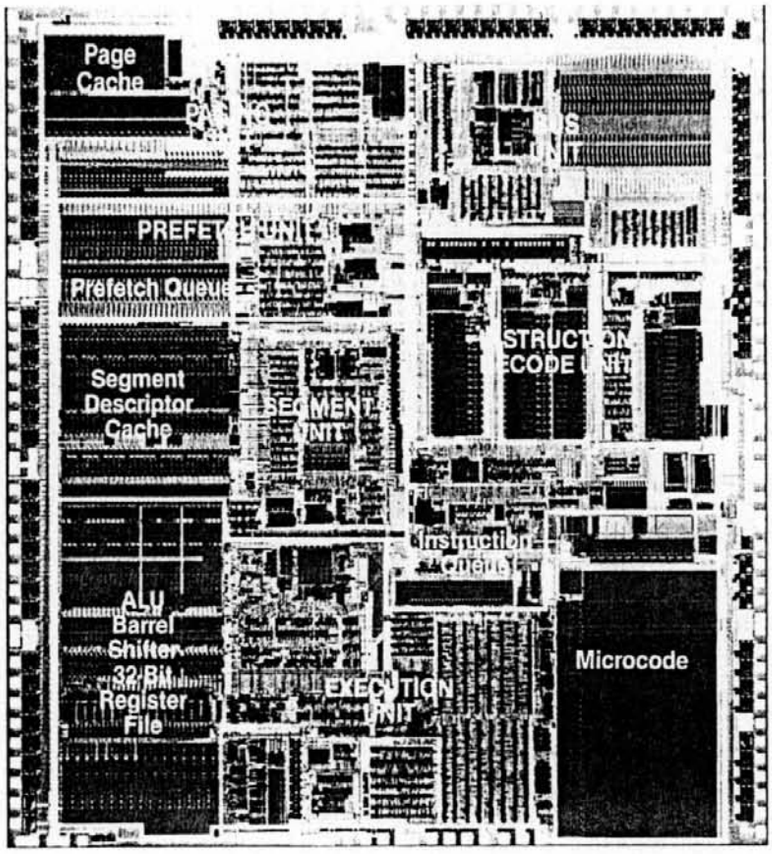


Figura 3.2 - Intel 80386.
Extraído de [GEI 90]

Estruturas típicas em lógica estruturada são ROMs, RAMs e PLAs. Estruturas usadas por geradores de módulos [MOR 90] (programas especializados em gerar leiaute) em lógica aleatória, são *gate array*, *standard cell*, *Weinberger array* e *gate matrix*. Certos autores [GEI 90] definem estas últimas estruturas como lógica estruturada, por apresentarem certa regularidade no leiaute. Pelo fato de facilitarem a implementação de circuitos projetados em lógica aleatória, será aqui usado o termo lógica aleatória.

3.2 ROM

Uma importante aplicação da ROM é a síntese de partes de controle, conhecido como controle microprogramado ou microprogramação.

As ROMs (Figura 3.3) compõem-se basicamente de um decodificador e uma matriz que relaciona cada sinal de entrada com um conjunto de sinais de saída. Para um conjunto de sinais de n entradas e m saídas, o tamanho da ROM será $2^n x m$.

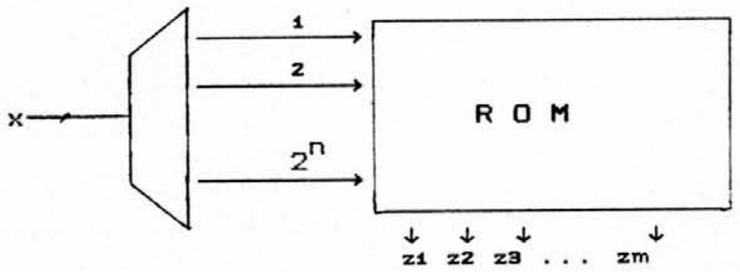


Figura 3.3 - ROM com decodificador.

O número de níveis no decodificador determina o número de portas lógicas pelas quais os sinais x deverão passar até atingir a ROM, onde sofrerão mais um retardo até formarem os sinais de saída z1, z2, ..., zm.

Para uma melhor otimização da ROM, é possível a colocação de multiplexadores nas saídas. Assim, os sinais em x são divididos em dois grupos, um para os decodificadores e outro para os multiplexadores. O ponto ideal corresponde ao menor perímetro para a ROM.

A ROM pode também ser otimizada, considerando que existam colunas iguais que geram a mesma função. No entanto, tal solução nem sempre é viável, pois pode aumentar o custo do roteamento.

Principais vantagens do estilo de projeto em ROM:

1. Flexível - permite alteração com facilidade.
2. Bloco regular - devido à regularidade, oferece grandes facilidades para implementação com geradores de módulos.
3. Não necessita de ferramentas de minimização booleana para ser utilizada.

As principais desvantagens são:

1. Atraso elevado - necessitam de um decodificador de endereços na entrada, que somado à própria ROM, aumenta bastante o atraso de propagação dos sinais da entrada para a saída.
2. Grande ocupação de área - para n sinais na entrada do decodificador, devem ser geradas todas as combinações possíveis destes sinais, resultando em uma ROM de 2^n entradas.
3. Não permitem a implementação de equações em lógica multinível.

3.3 PLA

O PLA (Programmable Logic Array) é uma estrutura regular, semelhante à ROM, é composta basicamente de dois planos (E e OU), e que permite a implementação de equações booleanas em dois níveis.

Uma boa estimativa do tamanho do PLA pode ser obtido da seguinte forma. Para n entradas, p termos-produto e m saídas, a área é dada por:

Área (matriz E) = $2.n.p.$Aand, sendo Aand a área de uma célula AND.

Área (matriz OU)= $m.p.$Aor, sendo Aor a área de uma célula OR.

O termo *n* aparece dobrado, pois cada entrada produz duas linhas no plano E (sinal e complemento). Para consideração de atraso, deve-se levar em conta que da entrada para a saída, o sinal passa por cinco níveis, incluindo os buffers e inversores de entrada, plano E, plano OU, interconexão entre os dois planos e inversores de saída.

A implementação de equações minimizadas diminui a área do PLA, e o minimizador lógico deverá atuar objetivando redução do número de entradas *n*, saídas *m* e termos-produto *p*.

Principais vantagens do projeto com PLA:

1. Flexíveis - permitem alteracões com facilidade, mas não são tão flexíveis quanto as ROMs.
2. São blocos também regulares e podem ser facilmente sintetizados por geradores de módulos.
3. Permitem implementação de equações a dois níveis e otimizadas.
4. Podem ser aplicadas otimizações topológicas.
5. Velocidade - são mais rápidos que as ROMs, pois não necessitam de decodificador na entrada.

Principais desvantagens:

1. Baixa densidade - PLAs podem ser muito esparsos, principalmente quando utilizados como codificadores e decodificadores.
2. Conexão - entradas e saídas muito próximas, dificultando conexões com outros blocos.
3. Embora permitam a implementação de equações minimizadas, necessitam de ferramentas como otimizadores de termos, para PALs comerciais, e otimizadores topológicos, para PLAs

implementados diretamente em VLSI.

Algumas otimizações topológicas que podem ser realizadas são:

- Otimização por dobramento (folded): é feita uma permutação das entradas e saídas, formando-se dois PLAs entrelaçados, que são interconectados formando um único PLA, com área bastante reduzida.

- Otimização triangular: os termos-produto são reordenados, de forma que as partes não conectadas se situem nas regiões exteriores do PLA, podendo então ser eliminadas, obtendo-se um PLA com formato triangular.

- Entradas e saídas laterais: algumas entradas e saídas são colocadas lateralmente, ao invés de em cima e em baixo. Diminui um pouco a área e facilita as conexões.

3.4 Gate Array

Fornecem um método simples para implementar sistemas digitais em VLSI.

Gate array compreende uma grande faixa de circuitos pré-difundidos, e todos são compostos de matrizes de células (transistores, portas lógicas), faltando apenas completar as conexões para as camadas de roteamento.

Como vantagens, permitem rápida prototipação e adaptam-se bem à lógica multinível.

Principais desvantagens:

1. Utilização ineficiente de área do silício, já que algumas camadas estão pré-definidas.

2. Dificuldade para implementar memórias de alta capacidade.
3. Baixo desempenho dos circuitos.
4. Dificuldades para uso completo dos elementos de circuitos disponíveis.

Algumas estratégias utilizadas com o objetivo de melhorar o desempenho do *gate array* são:

- Camadas de roteamento independentes das áreas ativas.
- Uso de células com área ativa maior e menor número de canais de roteamento.
- Uso de tecnologias de desempenho mais alto (TTL, ECL).

3.5 Weinberger Array

O estilo *Weinberger array* surgiu em 1967 e pode sintetizar o leiaute de circuitos em lógica multinível. Normalmente, a estrutura compõe-se unicamente de portas NMOS, implementando uma lógica NOR-NOR, embora possam existir dispositivos "pull-up" PMOS, com o *gate* preso à massa.

A forma básica é mostrada na figura 3.4. Cada porta consiste de duas linhas verticais de metal, uma conectando o transistor "pull-up", e a outra é a linha de terra (na prática, portas vizinhas compartilham a mesma linha de terra). As entradas são feitas através de linhas horizontais de polisilício.

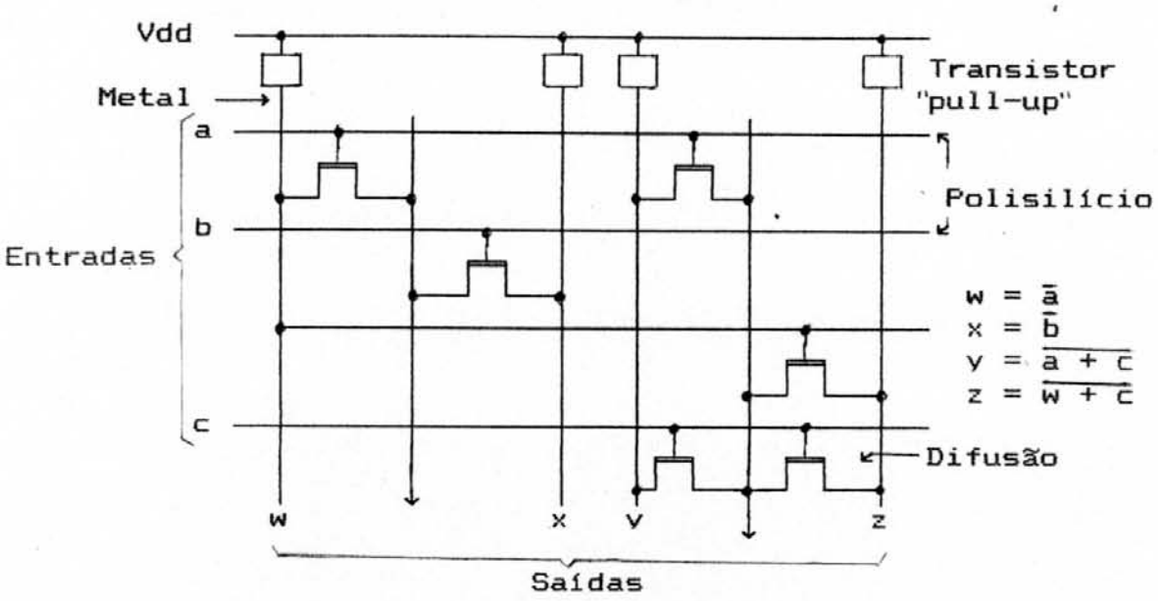


Figura 3.4 - Weinberger array.

A estrutura pode ser facilmente expandida, pela adição de entradas horizontais na base e colunas verticais à direita. Um aspecto negativo, é que a matriz resultante é muito esparsa. No entanto, por causa da sua forma simples, é facilmente sintetizada por programas que transformam as equações lógicas em uma descrição de máscaras. Um dos primeiros compiladores de silício [SIS 82], usou o *array* de *Weinberger*.

3.6 Gate Matrix

Semelhante ao *Weinberger array*, *gate matrix* (Figura 3.5) é composto de uma matriz de intersecção de linhas e colunas. Foi introduzido inicialmente por [LOP 80] e usado no desenvolvimento de microprocessadores de 32 bits [KAN 82].

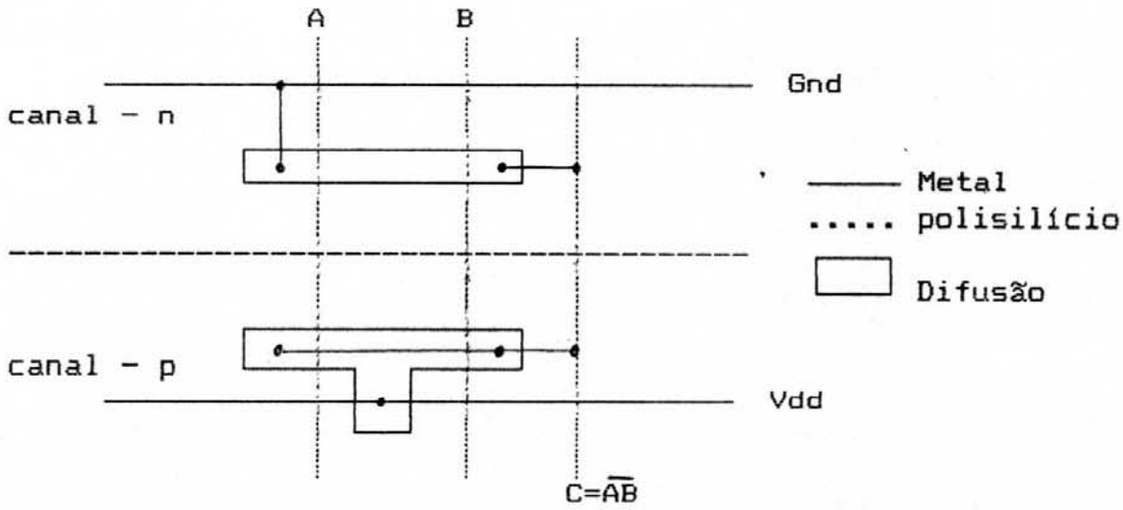


Figura 3.5 - Gate matrix.

Neste estilo, apropriado para implementação em CMOS, todos os transistores são colocados horizontalmente, conectando-se os transistores *p* e *n* que possuam o mesmo sinal na entrada, por meio de camada de polisilício. Camadas de metal 1 são dispostas horizontalmente, conectando drenos e fontes de mesmo potencial. Também podem ser feitas conexões verticais de difusão para conectar drenos e fontes equipotenciais.

A regularidade do leiaute facilita a automação. O sistema descrito em [MOR 90] gera o leiaute em *gate matrix*, otimizando o número de trilhas por banda (espaço entre as linhas de alimentação, Vdd e Gnd), sendo o número de colunas verticais fixo, determinado pelo número de *gates*. Em circuitos gerados com várias bandas, o roteamento vertical entre bandas e sinais de interface (definidos pelo usuário), é feito com metal 2, servindo as camadas de metal 1 para o roteamento interno.

4. ATRASO EM CMOS E MODELOS DE TIMING

Todos os circuitos integrados possuem uma frequência máxima de operação. Em uma máquina de estados finitos, a frequência de operação está limitada pelo mais longo caminho dentro de sua parte combinacional. Para circuitos compostos de vários blocos, a máxima frequência pode estar limitada pelas linhas que devem carregar informação entre eles. O projetista está encarregado de encontrar os caminhos do circuito que provocam o maior atraso, para assim tentar minimizá-los. Estes caminhos se denominam caminhos críticos.

Encontrar o caminho crítico para um circuito não é tarefa fácil. A técnica mais comum é utilizar simuladores elétricos, para analisar percursos que se suspeita de serem os mais longos. O problema está no fato de que um caminho não analisado pode ser justamente o crítico, além de simulações elétricas serem adequadas apenas para pequenos circuitos. Mais recentemente foram criados programas (analisadores, simuladores de *timing*), que fornecem o atraso, sem necessidade de simular o comportamento real dos circuitos, e se baseiam em modelos simplificados.

Este capítulo apresenta alguns simuladores de *timing* e dá maior ênfase em modelos para obter o atraso de propagação de sinais em redes de transistores. Os modelos se aplicam para estruturas como PLAs e *gate matrix*, aqui usadas na implementação de partes de controle (e máquinas sequenciais) que serão descritas nos capítulos posteriores.

4.1 Simuladores de Timing x Simuladores Elétricos

Definidas a arquitetura e estrutura lógica do sistema a ser implementado, o projeto físico do circuito integrado

correspondente necessita obviamente de informações sobre frequência máxima de operação e atraso de propagação de sinais, para que a arquitetura escolhida realmente possa ser implementada para operar na frequência desejada.

O uso de simuladores de *timing* permite um rápido pré-dimensionamento de transistores e fornece uma boa estimativa do desempenho elétrico do circuito, sem a necessidade de uma simulação elétrica detalhada. Isto evidencia a vantagem, já que simuladores elétricos necessitam de um exagerado tempo de CPU, para grande número de transistores, conforme pode ser visto na tabela 4.1, extraída de [ROB 90], que compara resultados obtidos com o simulador elétrico ELDO e com o simulador de *timing* PATH RUNNER.

Tabela 4.1 - ELDO x PATH RUNNER

Número de transistores	Tempo de processamento	
	ELDO	PATH RUNNER
56	42s	0,52s
112	1min372	0,77s
224	3min44s	1,45s
448	10min59s	2,53s
896	18min36s	6,05s
1344	31min39s	8,90s
1792	44min22s	12,10s
3136	1h13min	29,50s
4480	2h37min	47,30s

Além disso, simuladores elétricos requerem a especificação dos sinais de entrada para controlar a simulação, e portanto, somente caminhos do circuito para este particular conjunto de entradas são analisados. No segundo

grupo, todos os possíveis caminhos de sinal da entrada para a saída são mostrados.

Simuladores elétricos envolvem a solução de um sistema de equações diferenciais não-lineares, que modelam as características dinâmicas do circuito para um conjunto de tensões de entrada e para determinadas condições iniciais. A técnica usada em simuladores de *timing*, consiste em se considerar o transistor como uma chave perfeita, e o atraso de propagação de sinais é obtido pelo somatório de constantes RC, normalmente obtidas previamente de uma extração do leiaute.

4.2 Simuladores de Timing

Esta seção apresenta alguns simuladores para obtenção aproximada de atrasos. As etapas básicas em simuladores (analisadores) de *timing* podem ser resumidas como:

- Extração da topologia da rede a partir do leiaute.
- Conversão para o circuito elétrico equivalente.
- Cálculo dos atrasos, a partir do modelo RC escolhido.
- Análise e obtenção dos caminhos mais longos.

a) CRYSTAL

CRYSTAL [OUS 85] é um simulador de *timing* que foi inicialmente desenvolvido para avaliar estruturas RISC, desenvolvidas na Universidade da Califórnia, Berkeley. A análise é baseada em uma descrição do circuito que é extraída diretamente de uma especificação geométrica, que inclui dimensionamento dos transistores, capacitâncias e resistências.

O circuito é dividido em estágios, e os atrasos são obtidos estágio à estágio. A análise pode ser feita por meio

de três modelos diferentes. O terceiro modelo, mais preciso, aplica conceitos de Rubinstein-Penfield [RUB 83], e considera o efeito da forma de onda na entrada (*gate* do transistor). Permite a análise de caminhos críticos com estimativas de erro de 10% em relação ao SPICE.

b) TV

TV [JOU 83] assemelha-se muito à CRYSTAL, no entanto usa modelos mais simplificados. Fornece resultados com erros de até 30%, e tem um tempo de execução menor.

c) AUTO-DELAY

AUTO-DELAY [PUT 82] necessita como entrada para fazer a análise, uma descrição lógica do circuito, parâmetros de tecnologia, e uma lista de caminhos de sinal especificados pelo usuário. Utiliza um algoritmo de redução de redes RC, para simplificar a obtenção do resultado.

4.3 Modelos para Obtenção de Atraso

O estudo de modelos para verificação de atraso em circuitos integrados não é recente, conforme pode ser constatado pela existência dos simuladores descritos anteriormente.

Os modelos aqui descritos são todos lineares, e efeitos de forma da onda na entrada e o comportamento não-linear dos dispositivos semicondutores não são considerados.

Se for usado um modelo linear, o transistor pode ser considerado como uma chave perfeita, aberta para $Vgs < Vt$, e com uma resistência constante de canal, dada por (4.1), para $Vgs \geq Vt$.

$$R = (2.Vdd.L)/(K`.W(Vgs - Vt)^2) \quad (4.1)$$

sendo $K` = \mu.Cox$ ($\mu A/V^2$)

Para os parâmetros de processo típicos, da tecnologia de 2 μm (ECDM20 - ES2) e fazendo Vgs = 5V, têm-se:

$Rp = 41,52.Lp/Wp$ (kΩ) e $Rn = 14,23.Ln/Wn$ (kΩ)

Para Vtn = Vtp, Rp = Rn se $(Lp/Wp.\mu p) = (Ln/Wn.\mu n)$

4.3.1 Modelo de Rubinstein-Penfield

O modelo de Rubinstein-Penfield [RUB 83], aplica-se principalmente para redes RC em árvore, que ocorrem normalmente em circuitos projetados em lógica aleatória (portas lógicas). O transistor é considerado como um elemento linear, sendo substituído simplesmente por uma resistência constante, que pode ser obtida por (4.1).

Um estudo mais detalhado, para levar em conta o efeito da não-linearidade do transistor, é apresentado em [HOR 83]. Além disso, é considerado o efeito da forma de onda do sinal na entrada das portas, já que a resistência depende da tensão de porta.

Um exemplo simples de uma rede em árvore, para um estágio, pode ser observado na figura 4.1, onde o inversor aparece alimentando três *gates*. O modelo RC equivalente é mostrado na figura 4.2.

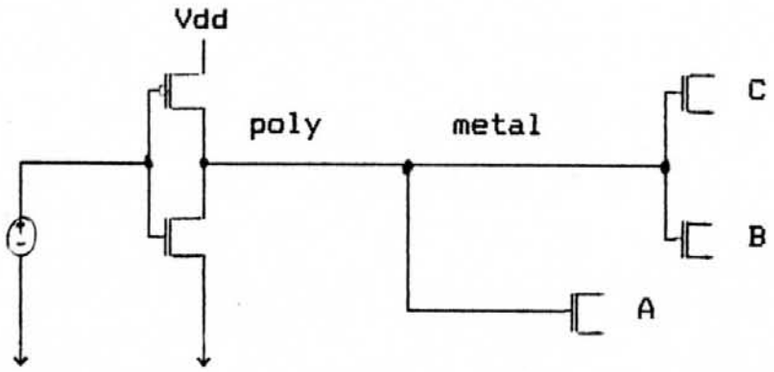


Figura 4.1 - Rede em Árvore.

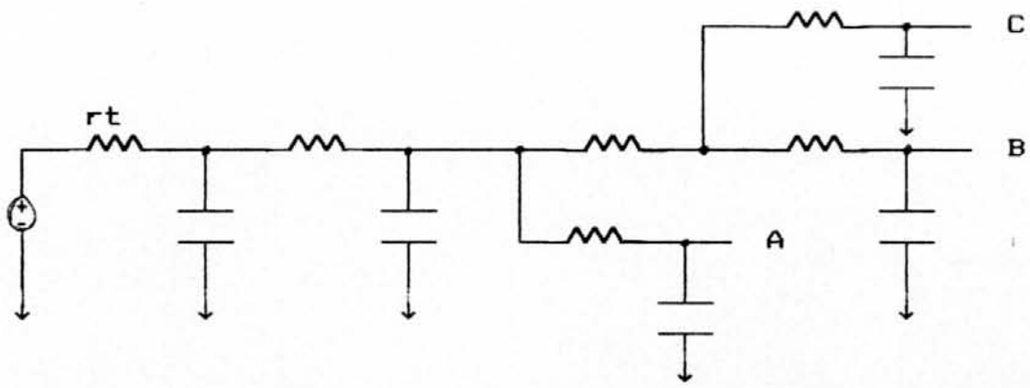


Figura 4.2 - Modelo RC equivalente.

Rubinstein e Penfield definem três constantes de tempo para a rede:

$$Tde = \sum R_{ke}C_k \qquad (4.2)$$

$$Tp = \sum R_{kk}C_k \qquad (4.3)$$

$$Tre = (\sum R^2_{ke}C_k)/R_{ee} \qquad (4.4)$$

R_{ke} é a resistência do caminho único entre a fonte e o nodo (A, B ou C), que é comum com a entrada e o nó *K*. R_{kk} é a resistência entre a entrada e o nó *k* e R_{ee} entre a entrada e a saída considerada (A, B ou C).

Obtidas as constantes, são definidos dois limites (superior e inferior), para a tensão no nodo *e*, dados por (4.5) e (4.6). A solução real se situa nos valores

intermediários entre os dois limites. Deve-se observar que os valores estão em uma escala em p.u., tomando-se valor base Vdd.

Limite superior:

$$Ve(t) \leq 1 - (Tde/Tp)\exp[-t/Tre], \quad \text{grande } t$$
$$Ve(t) \leq 1 - (Tde-t)/Tp, \quad \text{pequeno } t \qquad (4.5)$$

Limite inferior:

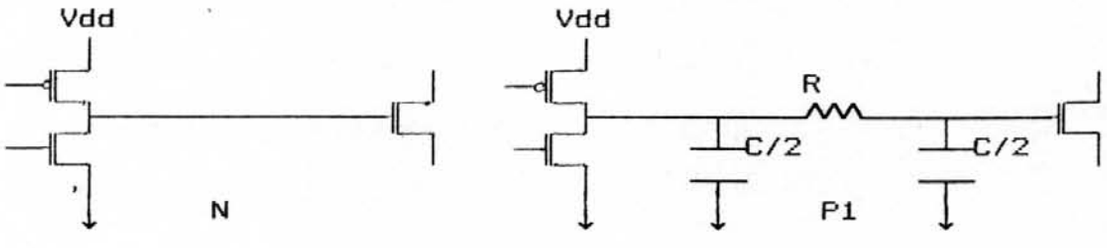
$$Ve(t) \geq 1 - Tde/(t+Tre), \quad \text{para } Tde-Tre \leq t \leq Tp-Tre$$
$$Ve(t) \geq 1 - (Tde/Tp)\exp[(Tp-Tre)/Tp]\exp[-t/Tp], \quad \text{para } Tp-Tre \leq t \qquad (4.6)$$

O modelo leva à obtenção de resultados com precisão de 20% ou mais.

4.3.2 Modelo de Sakuray

O modelo de Sakuray [SAK 83] tem melhor aplicação para obtenção do atraso em longas linhas de interconexão, principalmente de polisilício, como as que ocorrem normalmente em estruturas como PLAs e ROMs.

Para simulações elétricas (SPICE, ELDO), linhas de interconexão são substituídas por circuitos RC equivalentes (Figura 4.3). Quando é admitido um erro de apenas 3%, a tabela 4.2, extraída de [SAK 83], deve ser usada.



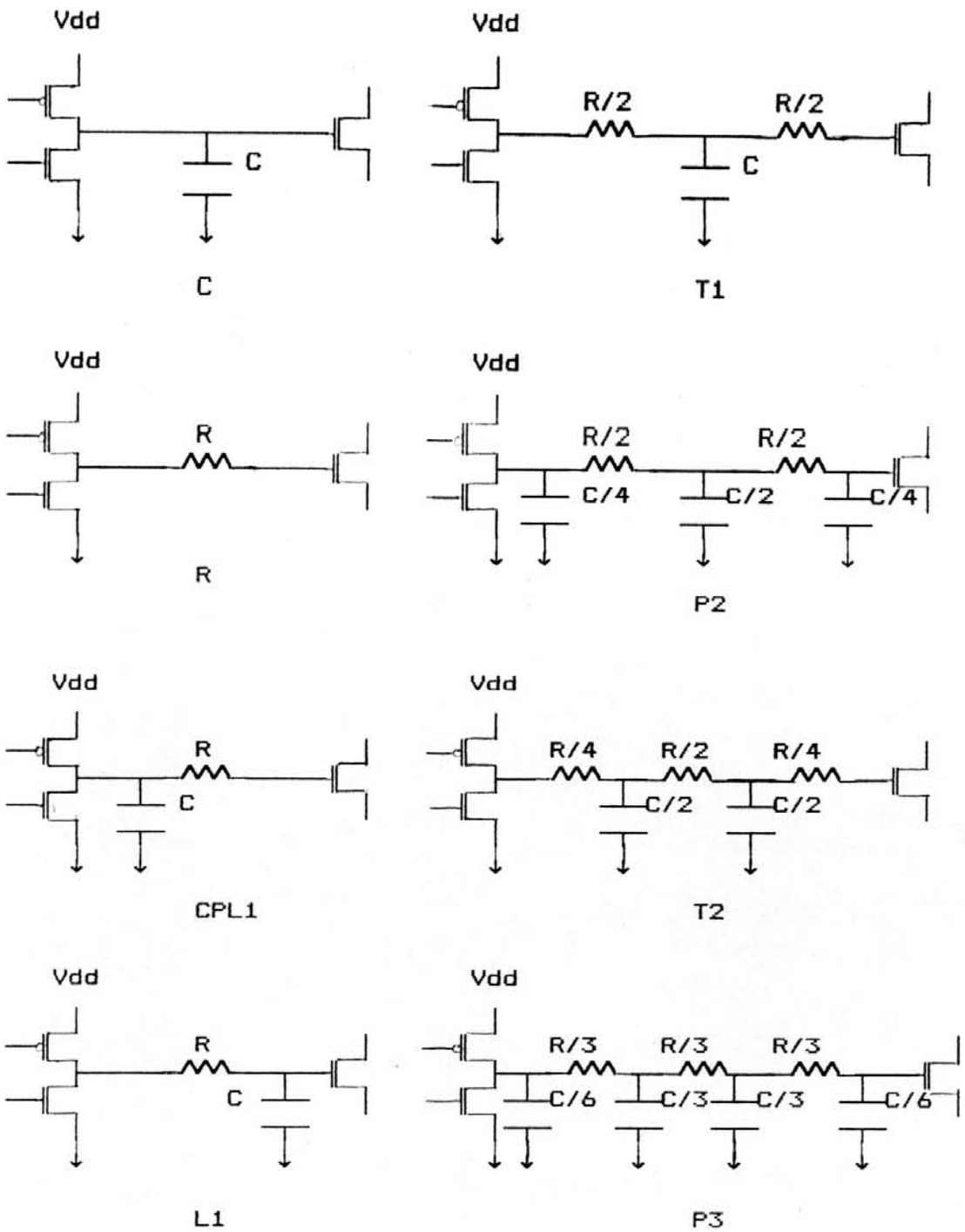


Figura 4.3 - Modelos para linhas RC distribuídas.

Tabela 4.2 - Circuito RC recomendado, erro de 3%.

CT\RT	0	.01	.1	.2	.5	1	2	5	10	20	50	100
0	P3	P3	P2	P2	P1	P1	P1	P1	P1	C	C	C
.01	P3	P3	P2	P2	P1	P1	P1	P1	P1	C	C	C
.1	T2	T2	P2	P2	P1	P1	P1	P1	P1	C	C	C
.2	T2	T2	P2	P2	P1	P1	P1	P1	P1	C	C	C
.5	T1	T1	T1	T1	P1	P1	P1	P1	P1	C	C	C
1	T1	T1	T1	T1	P1	P1	P1	P1	P1	C	C	C
2	T1	T1	T1	T1	P1	P1	P1	P1	L1	L1	C	C
5	P1	P1	P1	P1	P1	P1	P1	L1	L1	L1	C	C
10	P1	P1	P1	P1	P1	P1	L1	L1	L1	L1	C	C
20	R	R	R	R	R	R	L1	L1	L1	L1	C	C
50	R	R	R	R	R	R	R	R	R	R	C	N
100	R	R	R	R	R	R	R	R	R	R	N	N

r - resistência por unidade de comprimento.
c - capacitância por unidade de comprimento.
R - resistência total da linha distribuída.
C - capacitância total da linha distribuída.
rt - resistência do transistor que "drive" a linha.
ct - capacitância de carga no fim da linha.
RT = rt/R, CT = ct/C.

Sakuray fornece uma aproximação analítica para a resposta no ponto S (Figura 4.4), para uma excitação em degrau ou em rampa com pouca inclinação, na entrada do inversor.

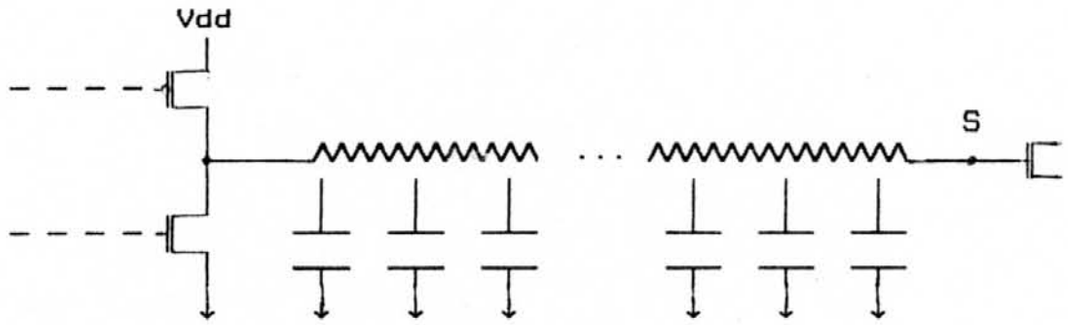


Figura 4.4 - Modelo de Sakuray.

O tempo para o ponto S passar de 0 para 90% de Vdd, é dado por (4.7).

$$t\ (90\%) = 1{,}02RC + 2{,}21(rtct + rtC + Rct) \qquad (4.7)$$

A equação (4.7) fornece erros menores que 3% para a faixa de valores em que rt << R. Para valores em que rt >> R, rt assume uma parcela maior, o que implica em aumento de erro, já que (4.7) considera o transistor como elemento linear.

O mesmo é válido se for considerado o efeito da forma de onda da entrada, pois para rt << R, a constante RC da linha predomina.

A tabela 4.3 compara valores obtidos a partir de simulações elétricas feitas com SPICE, usando-se o modelo P3, com a fórmula de Sakuray, para atingir 90% de Vdd no ponto S (Figura 4.5), para uma linha de polisilício característica, de 1000μm x 2μm, alimentada por um inversor. A entrada do inversor é uma rampa com tempo de subida de 2 ns. Utilizou-se tecnologia 2 μm (ECDM20 - ES2). No caso, ct = 0.

Tabela 4.3 - SPICE x Sakuray

W/L	rt (kOhm)	SPICE (ns)	Sakuray (ns)
3	13,84	2,70	3,01
4	10,58	2,25	2,49
5	8,30	1,94	2,13
6	6,92	1,76	1,91
9	4,61	1,44	1,54
12	3,46	1,29	1,36
15	2,77	1,19	1,25
18	2,31	1,12	1,18
21	1,98	1,07	1,12

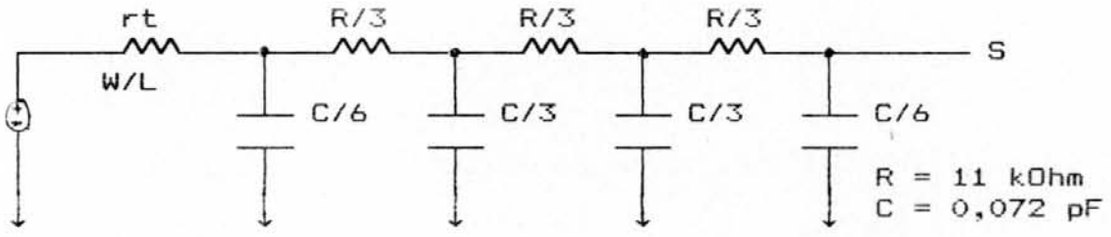


Figura 4.5 - Modelo equivalente da linha de polisilício.

4.3.3 Atrasos em Portas CMOS

Se forem desconsiderados os atrasos em linhas de interconexão (principalmente em linhas de metal), os modelos anteriores se simplificam bastante, e o atraso total na propagação dos sinais por várias portas, pode ser estimado pelo somatório do atraso de cada porta individualmente.

O fato de se poder desconsiderar o atraso nas linhas de interconexão é válido se estas possuírem comprimentos não superiores a certos limites. Comparando-se duas linhas (uma de metal e outra de polisilício), e aplicando-se o modelo Li [SAK 83], que fornece o pior caso, obteve-se:

- Metal r = 0,1 Ohm/□ c = 0,0186 fF/μm^2
  Dimensões 1000 μm x 4 μm atraso = 0,0043 ns

- Polisilício r = 22 Ohm/□ c = 0,036 fF/μm^2
  Dimensões 1000 μm x 4 μm atraso = 1,82 ns

O polisilício apresenta um atraso muito superior ao metal. Os circuitos que serão analisados posteriormente, possuem linhas de metal e polisilício de comprimento muito inferior ao exemplo acima, e para as estimativas de atraso não serão considerados os elementos de interconexão.

Assim, uma estimativa aproximada dos tempos de subida (tr), descida (tf) e atraso em portas CMOS pode ser obtida considerando apenas as resistências de canal dos transistores, em determinado ponto de operação, e as capacitâncias de gate (Cox) e de junção (CJ). Para a porta OR2 da figura 4.6, com uma transição de A e B de Vdd para 0 V, o tempo de descida (tf) do sinal na saída será simplesmente o somatório de dois termos RC, ao invés de se ter uma rede RC em árvore, o que implicaria no uso do modelo de Rubinstein-Penfield.

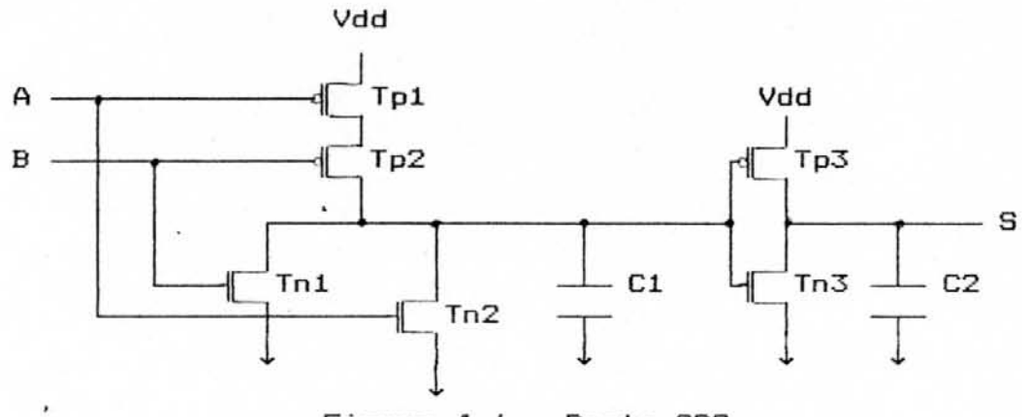


Figura 4.6 - Porta OR2.

Para uma transição de Vdd para 10% de Vdd na saída S:

$$tf = 2{,}3.[(Rp1 + Rp2).C1 + (Rn3).(C2)] \qquad (4.8)$$

onde: Rp1, Rp2 = resistências de canal dos transistores *p*.
Rn3 = resistência de canal do transistor *n* do inversor.
C1 = capacitância de dreno de Tp2 + Tn1 + Tn2, somada às capacitâncias de *gate* de Tp3 e Tn3.
C2 = capacitância de carga (C1), do estágio seguinte, somada às capacitâncias de dreno de Tp3 e Tn3.

A tabela abaixo compara o uso da expressão (4.8) com simulação elétrica, em função da capacitância de carga (C1) na saída.

tf calculado (ns)	tf SPICE (ns)	C1 (pF)
4,04	3,86	0,05
5,65	4,71	0,1
11,09	9,24	0,3
19,53	15,02	0,5
24,75	17,43	0,7

Definidos o tempo de subida, tr (0 à 90% de Vdd) e o tempo de descida, tf (Vdd à 10% de Vdd), para fins de projeto, o atraso médio (tm) de uma porta CMOS é calculado em função de tf e tr. Segundo [WES 85], o atraso médio de uma porta é dado por (4.9).

$$\text{atraso médio (tm)} = (tr + tf)/4 \qquad (4.9).$$

A tabela 4.4 mostra valores pra tf, tr e tm, para várias portas lógicas CMOS, em função do *fan-out*, para a tecnologia de 2 μm (ES2). Em todos os casos Ln = Lp = 2 μm, e Wn = 3 μm e Wp = 6 μm. Um *fan-out* igual à 1, significa que a saída da porta está ligada apenas a um transistor *p* e um transistor *n*, de mesma dimensão que os da porta. Para um *fan-out* igual à 2, a porta está ligada a dois transistores *p* e dois transistores *n*, e assim sucessivamente.

Tabela 4.4 - Atrasos em portas CMOS

	AND4	AND3	AND2
FAN-OUT	tr = 6,24 ns	tr = 4,23 ns	tr = 2,67 ns
1	tf = 2,57 ns	tf = 2,19 ns	tf = 1,90 ns
	tm = 2,20 ns	tm = 1,60 ns	tm = 1,14 ns
FAN-OUT	tr = 6,73 ns	tr = 4,72 ns	tr = 3,16 ns
2	tf = 2,91 ns	tf = 2,53 ns	tf = 2,24 ns
	tm = 2,41 ns	tm = 1,81 ns	tm = 1,35 ns
FAN-OUT	tr = 7,23 ns	tr = 5,22 ns	tr = 3,66 ns
3	tf = 2,61 ns	tf = 2,87 ns	tf = 2,58 ns
	tm = 2,46 ns	tm = 2,02 ns	tm = 1,56 ns
FAN-OUT	tr = 7,72 ns	tr = 5,71 ns	tr = 4,15 ns
4	tf = 3,25 ns	tf = 3,21 ns	tf = 2,92 ns
	tm = 2,74 ns	tm = 2,28 ns	tm = 1,77 ns
FAN-OUT	tr = 8,21 ns	tr = 6,20 ns	tr = 4,64 ns
5	tf = 3,59 ns	tf = 3,55 ns	tf = 3,26 ns
	tm = 2,95 ns	tm = 2,44 ns	tm = 1,98 ns

	NAND4	NAND3	NAND2
FAN-OUT	tr = 1,93 ns	tr = 1,59 ns	tr = 1,26 ns
1	tf = 5,32 ns	tf = 3,31 ns	tf = 1,74 ns
	tm = 1,81 ns	tm = 1,23 ns	tm = 0,75 ns
FAN-OUT	tr = 2,42 ns	tr = 2,08 ns	tr = 1,75 ns
2	tf = 6,68 ns	tf = 4,32 ns	tf = 2,42 ns
	tm = 2,28 ns	tm = 1,60 ns	tm = 1,04 ns
FAN-OUT	tr = 2,91 ns	tr = 2,58 ns	tr = 2,25 ns
3	tf = 8,04 ns	tf = 5,34 ns	tf = 3,10 ns
	tm = 2,74 ns	tm = 1,98 ns	tm = 1,34 ns
FAN-OUT	tr = 3,40 ns	tr = 3,07 ns	tr = 2,74 ns
4	tf = 9,40 ns	tf = 6,36 ns	tf = 3,78 ns
	tm = 3,20 ns	tm = 2,36 ns	tm = 1,63 ns
FAN-OUT	tr = 3,89 ns	tr = 3,56 ns	tr = 3,23 ns
5	tf = 10,76ns	tf = 7,37 ns	tf = 4,46 ns
	tm = 3,66 ns	tm = 2,73 ns	tm = 1,92 ns

Um outro exemplo procura mostrar a validade do modelo simplificado para obtenção dos tempos de subida, descida e atraso em portas CMOS.

O circuito escolhido é um estágio de um somador com *carry* de n estágios, conforme figuras abaixo.

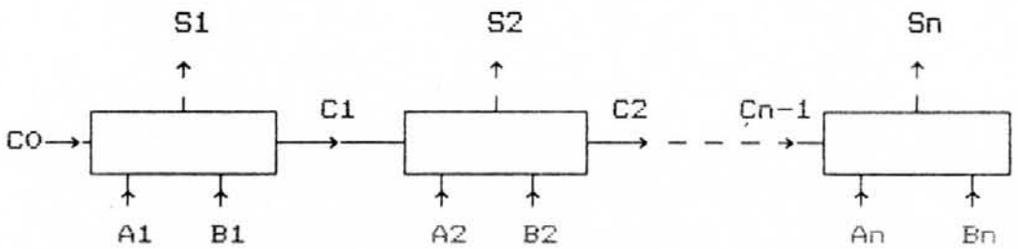


Figura 4.7 - Somador com *carry*.

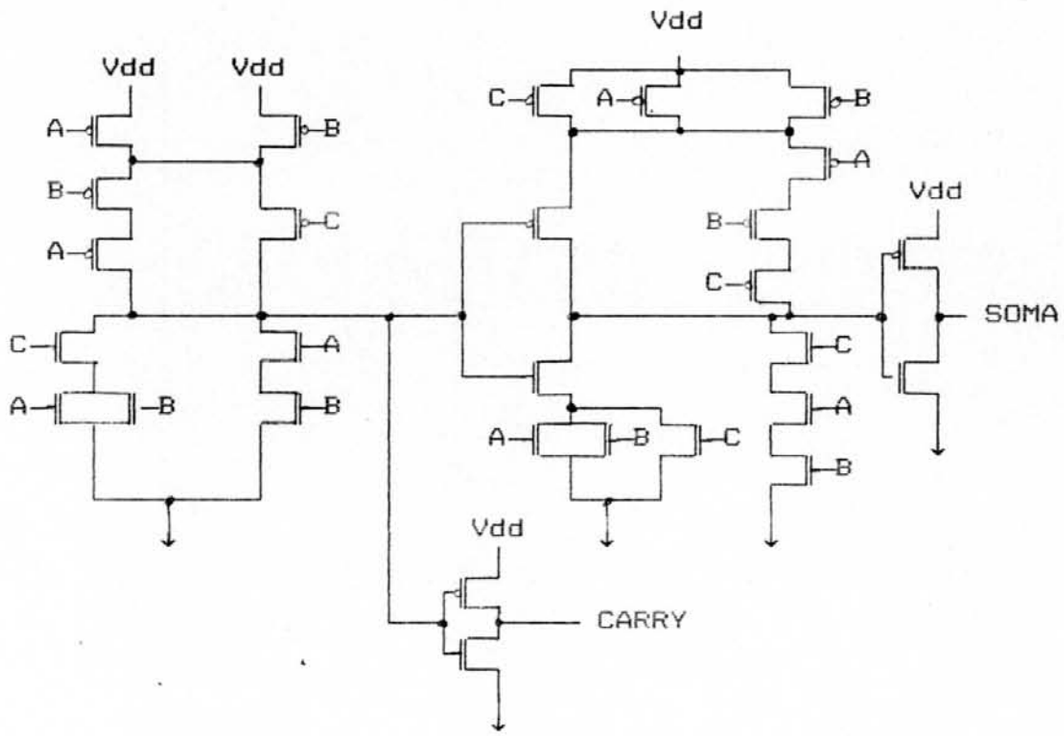


Figura 4.8 - Diagrama elétrico do somador com *carry*, para um estágio.

Para C = 0 V e A = 5 V, durante a transição da entrada B (rampa com descida de 2 ns) para Gnd, o atraso no sinal de *carry*, em relação ao sinal em B, obtido por simulação (Figura 4.9) é de 2,72 ns. Mantidas as mesmas condições, o atraso médio obtido por (4.9) forneceu 3,17 ns.

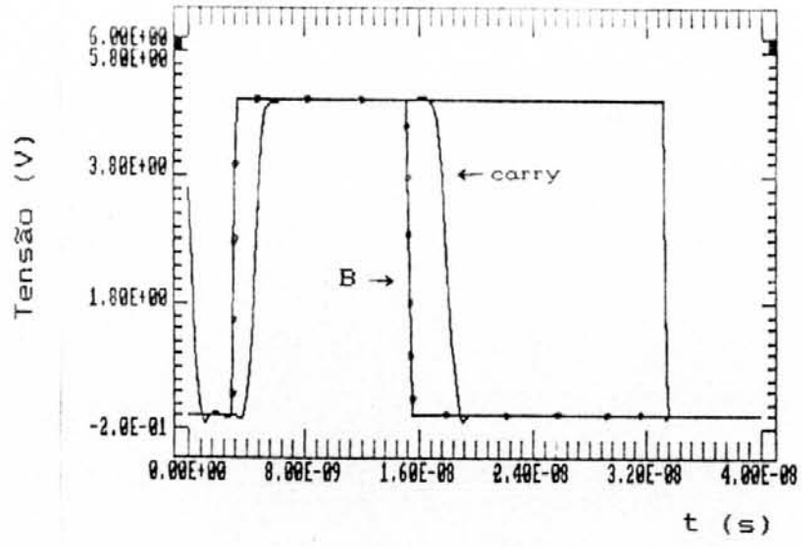


Figura 4.9 - Simulação SPICE.

5. ESTIMATIVAS DE DESEMPENHO

Este capítulo visa fornecer estimativas de desempenho para um PLA, e apresenta também o sistema de geração automática para circuitos em lógica aleatória, desenvolvido no GME - UFRGS. As duas alternativas serão comparadas.

No projeto de circuitos integrados é importante, se possível, obter informações sobre desempenho (área, frequência de operação), antes do desenho das máscaras que comporão o circuito. Esta tarefa torna-se mais fácil se forem empregadas estruturas regulares para o projeto. Se for conhecida a arquitetura do circuito, poderá assim ser definida a melhor alternativa para implementá-lo, ao invés de adotar-se vários estilos de leiaute (PLA, ROM, *gate matrix*, *gate array*, *standard cell*) e fazer-se posterior simulação elétrica e comparação de áreas obtidas entre os diversos estilos. A arquitetura interna de uma parte de controle, por exemplo, pode ser projetada com portas lógicas ou lógica estruturada. Desconsiderando ferramentas à disposição do usuário, ou seja, considerando somente o desempenho do circuito, o critério de seleção entre uma forma e outra depende do número de entradas, saídas, da lógica de seleção dos sinais de controle e da frequência e área desejadas no projeto.

Informações relativas ao desempenho podem ser facilmente incluídas em geradores de módulos regulares (ROM), onde a regularidade do leiaute facilita a automação e a pré-estimativa de desempenho do circuito que será implementado.

5.1 Estimativas de Desempenho de PLAs

Sendo o PLA uma estrutura regular, simplifica-se a tarefa de estimar área e velocidade, já que estas são funções do número de entradas, saídas e termos-produto, correspondentes às equações em lógica a dois níveis (and-or). Isto não ocorre para estruturas que visam implementar portas lógicas e circuitos em lógica multinível, pois a área vai depender do particionamento, posicionamento e roteamento dos elementos (células) do circuito.

5.1.1 Alternativas de Projeto

Nesta seção serão apresentadas algumas alternativas para PLAs encontradas na literatura.

a) PLA ESTÁTICO PSEUDO NMOS.

É uma das implementações mais simples e utiliza portas NOR NMOS nos dois planos. Para o transistor "pull-up" pode ser utilizado um transistor *p* com o *gate* ligado à massa, que desempenha a função de transistor de carga (figura 5.1). As principais vantagens são a simplicidade e o tamanho reduzido. Como desvantagens, há a dissipação estática de potência e a baixa velocidade, já que os transistores tipo *n* das células dos planos E e OU tem que se opor ao transistores *p*, para que as linhas sejam descarregadas.

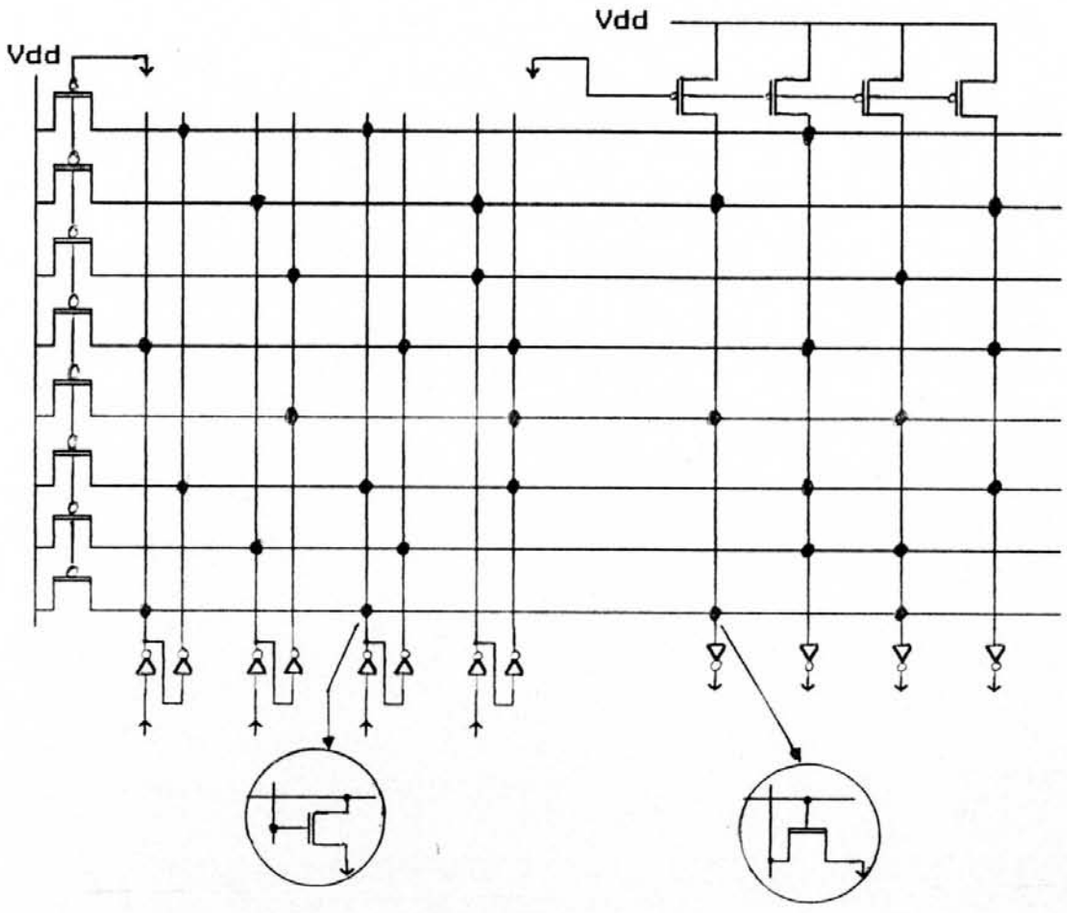


Figura 5.1 - PLA estático pseudo NMOS.

b) PLA DINÂMICO DE DUAS FASES.

Nesta arquitetura, o circuito do PLA trabalha em duas fases: uma de pré-carga e outra de execução dos planos. Tal técnica evita a dissipação estática de potência [WES 85], impedindo que os transistores "pull-up" dos dois planos estejam em contato com o nó de terra durante as fases de pré-carga. Ocorre aumento de área, com a colocação de "buffers" entre os planos.

c) DINÂMICO DE QUATRO FASES.

'Esta arquitetura não é aqui analisada em detalhe. Apresenta como grande desvantagem a necessidade de gerar as fases internamente com perda de área [WES 85], ou

externamente, deixando o problema para o usuário do PLA.

5.1.2 Detalhamento de Projeto

Esta seção apresenta o detalhamento do PLA estático pseudo NMOS, que é usado para estimar o desempenho dos sistemas digitais descritos nos capítulos posteriores. Será considerada a tecnologia de 2,0 μm (ECDM20 - ES2), com uma camada de metal para o PLA.

Os blocos básicos para o PLA, com as dimensões para os "buffers" de entrada e saída e dispositivos "pull-up", é mostrado na figura 5.2. Maiores detalhes sobre as células básicas são dados no anexo 1.

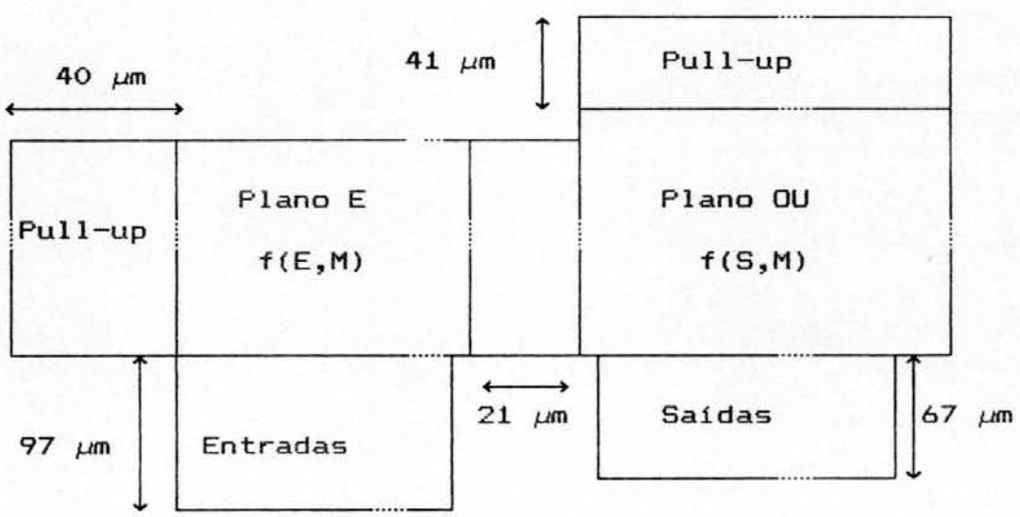


Figura 5.2 - Esquema do PLA.

A relação entre os transistores *p* e *n* foi estabelecida como sendo (Lp/Wp)/(Ln/Wn) ≃ 3. Uma estimativa do atraso é obtida sabendo-se que da entrada para a saída os sinais passam por cinco níveis: "buffers" de entrada, plano E, interconexão entre os dois planos, plano OU, e inversor na saída. Foi considerada a presença de transistores em todos os pontos nos dois planos. As linhas de polisilício e metal que

cruzam os dois planos, são modeladas conforme apresentado no capítulo 4, segundo o modelo de [SAK 83].

Em função do número de entradas (E), saídas (S) e termos-produto (M), tem-se:

$$\text{Área} = X.Y \qquad (6.1)$$

$$X = 61 + 17.E + 11.S \ (\mu m) \qquad Y = 138 + 12.M \ (\mu m)$$

$$\text{atraso (ns)} = 6{,}4 + 0{,}270.M + 0{,}137.E + 0{,}725.S \qquad (6.2)$$

A figura 5.3 mostra a variação de área e atraso em função de M.

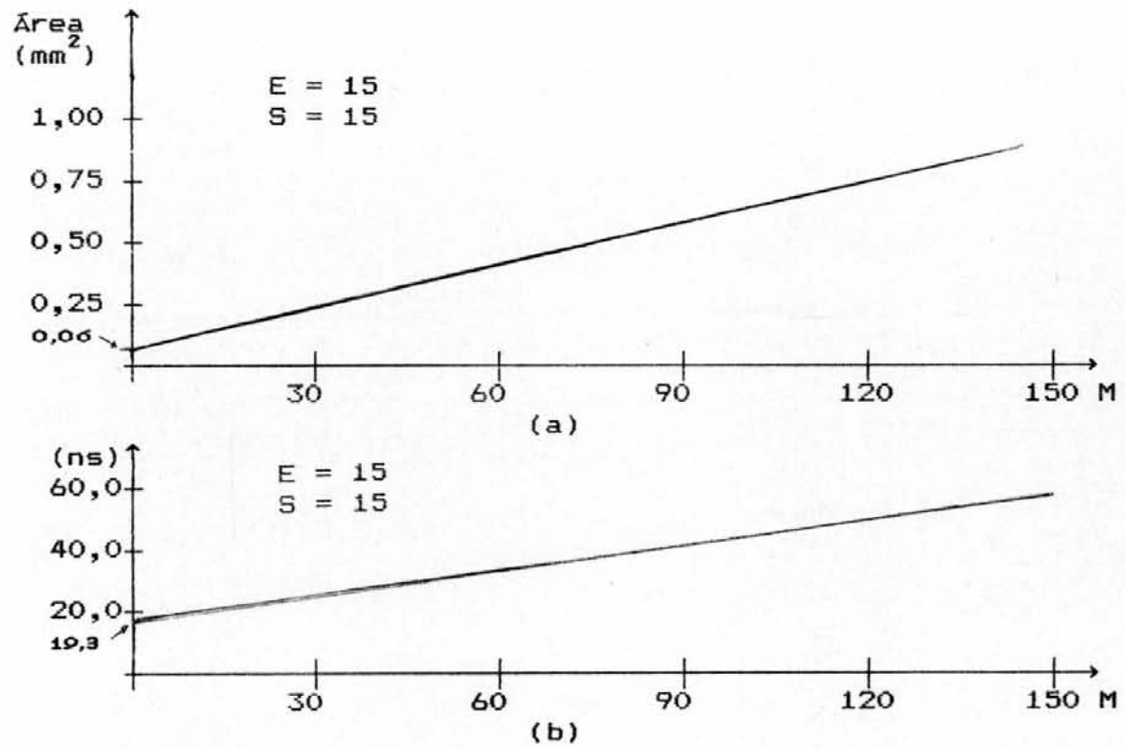


Figura 5.3 - PLA x M. (a) Área x M; (b) Atraso x M.

Para validar a expressão (6.2) fez-se a simulação elétrica do PLA (figura 5.4), para M = 3, S = 3 e E = 4, e que forneceu 7,18 ns. A expressão (6.2) fornece 9,33 ns, resultando em um erro de 27%. Com o aumento do PLA ocorre diminuição de erro, pois a constante (6,4 ns) é obtida considerando os transistores *p* "pull-up" como elementos lineares.

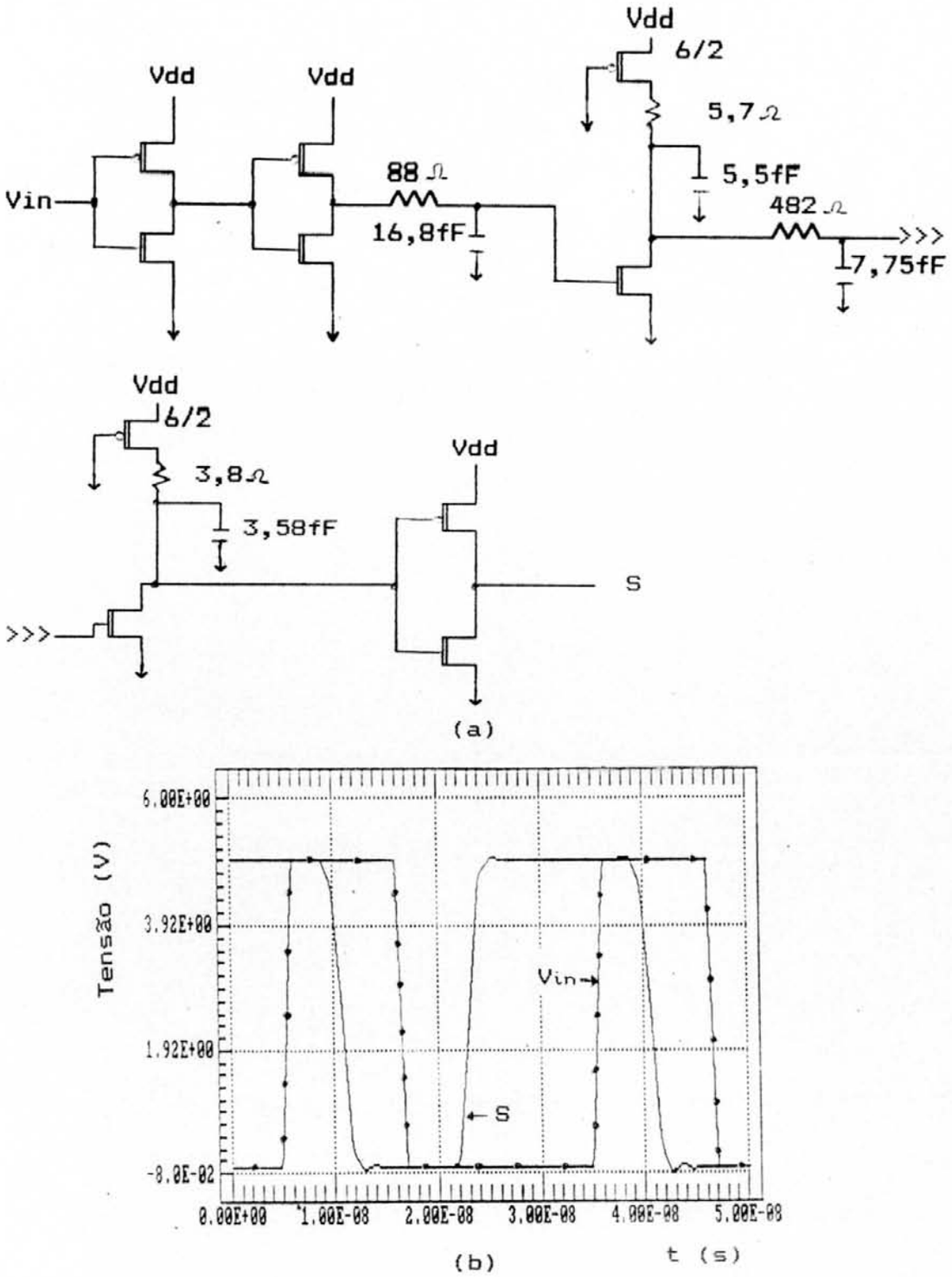


Figura 5.4 - Simulação do PLA. (a) Circuito equivalente; (b) Simulação SPICE.

5.2 Circuitos em Lógica Aleatória

Para circuitos em lógica aleatória, torna-se difícil obter uma precisão inicial para a área ocupada pelo leiaute. Estimativas podem ser feitas com base em experiências oriundas de projetos anteriores. Dimensões mínimas podem ser estimadas se for empregada uma biblioteca de células geométricas, com as dimensões fixas para dada tecnologia. No entanto, e principalmente no caso de geração automática, a área final vai depender do posicionamento, particionamento e roteamento das células que farão parte do circuito.

5.2.1 Sistema para Geração de Lógica Aleatória

O sistema para geração de lógica aleatória, TRAGO, desenvolvido por MORAES [MOR 90]], utiliza a estratégia *gate matrix* na síntese de leiaute, e os circuitos são gerados automaticamente a partir de uma descrição no nível lógico, com subcircuitos que permitem uma descrição no nível de transistores. Ao invés das tradicionais bibliotecas geométricas, utiliza bibliotecas estruturais, onde as células são descritas por um *net list*, sendo a geração feita a partir deste.

As diferenças principais entre bibliotecas estruturais e geométricas, evidenciando as vantagens das primeiras, são:

a) Quanto ao número de células.

As bibliotecas geométricas apresentam número fixo de células, e poucas opções em portas compostas para realizar funções mais complexas.

Nas bibliotecas estruturais basta inserir novas células, pela criação de um novo *net list*.

b) Quanto ao dimensionamento elétrico.

As bibliotecas geométricas são validadas eletricamente para suprir determinada carga. Um dimensionamento elétrico diferente, implicará na escolha de células diferentes.

Nas bibliotecas estruturais, a parametrização elétrica é obtida pela escolha dos valores W/L dos transistores, definidos na descrição das células básicas do circuito de entrada.

c) Quanto às regras de desenho.

As biblioteca geométricas são dependentes das regras de desenho. Uma alteração das regras, implica que todas as células terão que ser refeitas e revalidadas.

Nas bibliotecas estruturais, a independência às regras é obtida pelo leiaute simbólico. Assim, a síntese é dividida em duas fases: topológica, que parte do *net list* de entrada e gera a descrição simbólica do circuito, e a geométrica, que expande a descrição simbólica de acordo com as regras de desenho.

No estado atual, o sistema TRAGO não inclui um avaliador elétrico dos circuitos gerados. Assim, feita a geração do circuito, este deve ser simulado, e conforme o caso, há necessidade de alteração dos valores de W e L dos transistores no *net list* de entrada, para então fazer-se nova síntese. Para trabalhos futuros, poderia ser sugerida a inclusão dos modelos para atraso em portas CMOS do capítulo 4, podendo-se assim fazer um rápido pré-dimensionamento elétrico.

A figura 5.5 dá uma visão global do sistema TRAGO. expondo os principais módulos. Maiores informações sobre o detalhamento de cada módulo, podem ser obtidas em [MOR 90]. A síntese de leiaute é feita a partir de células básicas, definidas na descrição do circuito. O leiaute gerado apresenta bandas (espaço entre as linhas de Vdd e Gnd) com

altura variada, definida pelo número de trilhas necessárias para o roteamento. É possível definir o número de bandas desejado, ou simplesmente escolher a opção que calcula automaticamente o número de bandas, sendo que o algoritmo trabalha visando diminuição do número de trilhas.

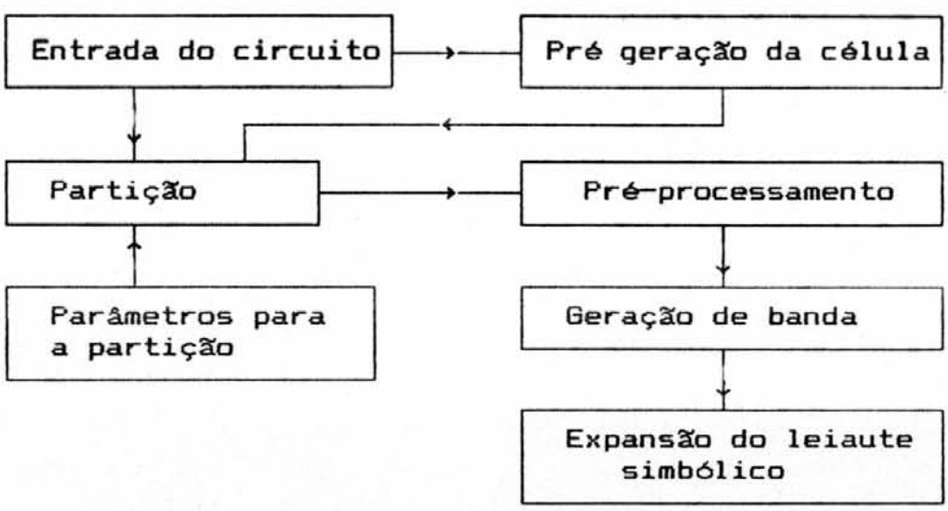


Figura 5.5 - Sistema TRAGO.

A figura 5.6 apresenta um circuito em lógica aleatória, e o leiaute sintetizado é mostrado na figura 5.7. Na figura 5.8 é mostrada a descrição para a entrada do gerador.

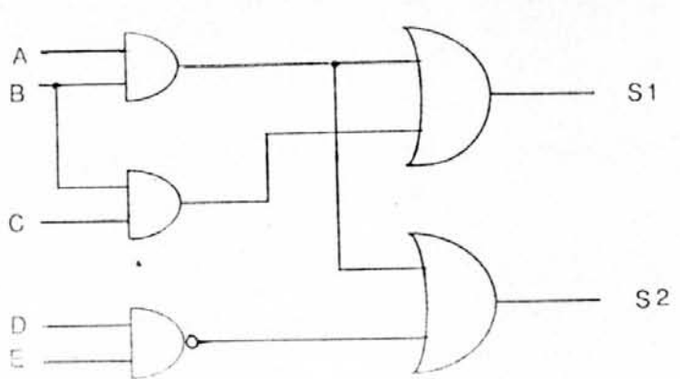


Figura 5.6 - Circuito exemplo.

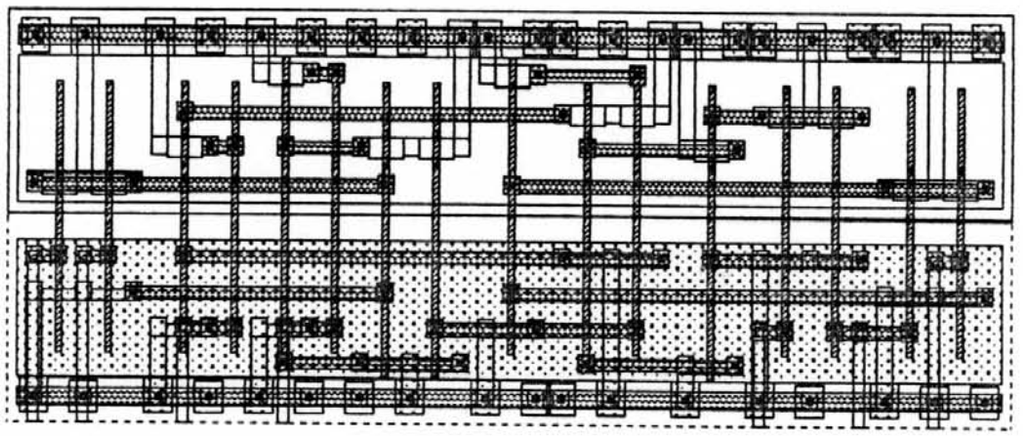


Figura 5.7 - Leiaute gate matrix.

```
.SUBCKT AND2 I1 I2 OUT vcc
MP1     2 I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2     2 I2  vcc  vcc PMOS L=2.0U W=6.0U
MN3     2 I1    3  0   NMOS L=2.0U W=3.0U
MN4     3 I2    0  0   NMOS L=2.0U W=3.0U
MN5  OUT  2     0  0   NMOS L=2.0U W=3.0U
MP6  OUT  2   vcc  vcc PMOS L=2.0U W=6.0U
.ENDS AND2

.SUBCKT NAND2 I1 I2 OUT vcc
MP1  OUT I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2  OUT I2  vcc  vcc PMOS L=2.0U W=6.0U
MN3  OUT I1    2  0   NMOS L=2.0U W=3.0U
MN4    2 I2    0  0   NMOS L=2.0U W=3.0U
.ENDS NAND2

.SUBCKT OR2 I1 I2 OUT vcc
MP1  2 I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2  S I2  2    vcc PMOS L=2.0U W=6.0U
MN3  S I1  0    0   NMOS L=2.0U W=3.0U
MN4  S I2  0    0   NMOS L=2.0U W=3.0U
MP5  OUT  S  vcc vcc PMOS L=2.0U W=6.0U
MN6  OUT  S  0   0   NMOS L=2.0U W=3.0U
.ENDS OR2

* CHAMADA DOS SUBCIRCUITOS *

X1 A  B  N1  vcc and2
X2 B  C  N2  vcc and2
X3 D  E  N3  vcc nand2
X4 N1 N2 S1  vcc or2
X5 N1 N3 S2  vcc or2

* sinais de interface *

*interface: A
*interface: B
*interface: C
*interface: D
*interface: E
*interface: S1
*interface: S2
.end
```

Figura 5.8 - Descrição estrutural hierárquica.

6. AVALIAÇÃO DO DESEMPENHO DA PARTE DE CONTROLE DE UM CIRCUITO PARA EQUALIZAÇÃO ADAPTATIVA

Este capítulo apresenta a avaliação do desempenho da parte de controle de um sistema digital para equalização adaptiva. Parte-se do projeto funcional do sistema, que foi desenvolvido no trabalho de [SAN 90].

6.1 O Sistema Digital

O sistema digital que incorpora a parte de controle é um filtro adaptativo com coeficientes complexos com 64 derivações, espaçadas de meio baud. O circuito equalizador atende as recomendações da norma V.29 do CCITT para modems de 9600 bits/s, em linhas telefônicas públicas a quatro fios.

O sistema foi especificado em [SAN 90], pressupondo uma interface com um processador DSP TMS320 [TEX 88], que realiza as funções de transmissão e recepção.

A função da equalização adaptativa (Figura 6.1) é a modificação do sinal recebido pelo receptor, mediante filtragem a coeficientes variáveis, dinâmica e automaticamente alterados para se adaptarem às características do canal de transmissão. Visa a minimização da interferência entre símbolos recebidos (yn).

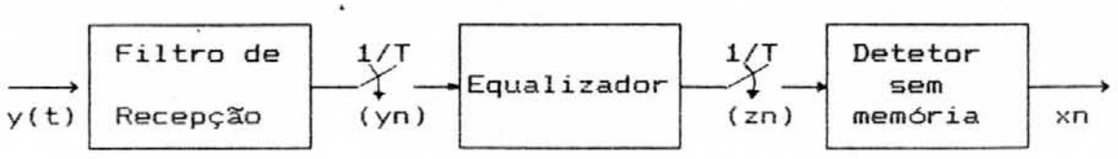


Figura 6.1 - Receptor (linear) com equalizador.

O equalizador é introduzido entre o filtro de recepção e

o detetor sem memória. A entrada (yn) e a saída (zn) do equalizador, são amostradas à taxa de símbolos 1/T. O detetor sem memória recompõe a sequência transmitida, y(t), a partir da sequência equalizada (zn).

Em canais de transmissão ideais, imunes a ruído (gaussiano, branco), a sequência de símbolos (yn) formaria um conjunto de dados estatísticos suficientes para obter a estimativa (xn) da sequência de símbolos recebida. Para um canal real, com distorção linear, a sequência de símbolos amostrados (yn), não mais representa suficientemente o conjunto de dados transmitidos, pois ocorre interferência entre símbolos. A simples inclusão do detetor sem memória, na figura 6.1, após a amostragem não é mais suficiente, já que o símbolo corrente é distorcido pela combinação linear dos símbolos passados. O equalizador, assim, manipula os símbolos de forma a recompor a sequência transmitida.

6.2 A Unidade Operativa

A parte operativa do sistema digital consiste basicamente em um arranjo de processadores idênticos.

O processador elementar constitui-se de três blocos básicos (Figura 6.2). O bloco **C** (combinacional) contém arranjos de portas lógicas e um recodificador de Booth. O bloco **PC** (pré-carga) contém o operador a ser implementado usando a técnica de pré-carga: o somador por propagação de *carry*. O bloco **M** (mèmória), contém todos os registradores do processador elementar, com exceção do registrador de deslocamento. Os blocos **C** e **PC** formam a unidade aritmética (**UAR**).

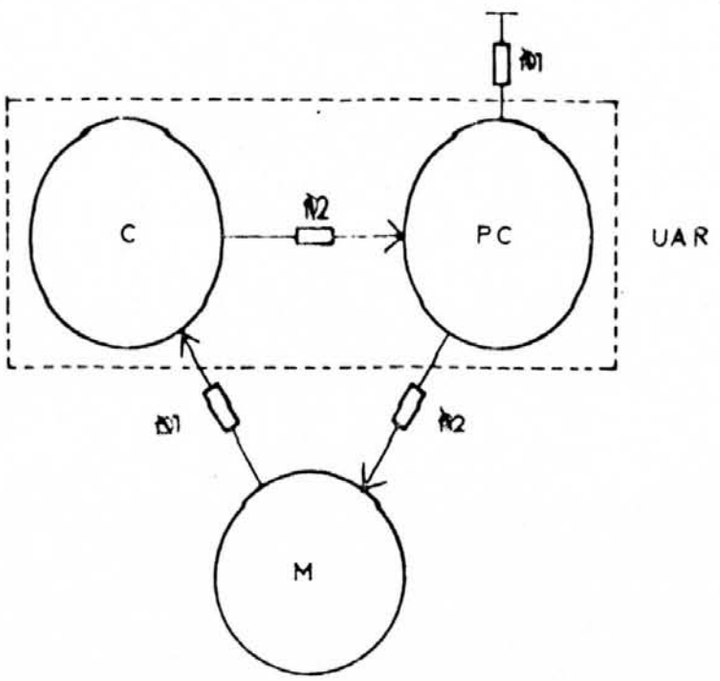


Figura 6.2 - Esquema para o processador elementar.

Dados são lidos durante a fase Φ1 e aplicados ao bloco combinacional **C**, cuja saída está estável na fase Φ2, coincidindo com a fase de avaliação do bloco **PC**, que é pré-carregado na fase Φ1. A saída deste deve estar estável na fase Φ2, para então ser escrita em um dos registradores do bloco **M**. Um barramento, não representado na figura, permite fluxo de dados de **M** para a **UAR**, durante a fase Φ1, e da **UAR** para **M**, durante Φ2.

A comunicação entre os processadores elementares, envolve três tipos de transferência de dados: deslocamento de valores amostrados, distribuição do valor de erro e soma dos produtos acumulados em cada processador elementar. Maiores informações sobre o processador elementar e a interconexão dos vários processadores, podem ser obtidas em [SAN 90].

As operações e transferência de dados entre processadores elementares, são controladas pelos sinais

vindos da parte de controle, definida no próximo ítem.

6.3 A Parte de Controle

A parte de controle (Figura 6.3) consiste basicamente em uma máquina de estados, formada por uma lógica combinacional (PLA, portas lógicas) e elementos de memória. Possui ainda um conjunto de contadores que auxiliam o sequenciamento e estabelecem a dinâmica de endereçamento da memória e que possuem as saídas conectadas às entradas da máquina de estados. Convém ressaltar já aqui, que as implementações (leiaute) que se seguem no decorrer deste capítulo, se restringem unicamente à máquina de estados. Os contadores e o bloco de lógica de endereçamento não estão incluídos.

Entre os contadores, **cs** é responsável pelo sequenciamento das somas e deslocamentos sucessivos que compõem uma multiplicação, **cm** é um contador de módulo 4, e que evolui quando uma multiplicação é completada. O contador **cp** tem módulo 16, evolui no final de cada multiplicação complexa e fornece os 4 bits mais significativos para a formação de um endereço na área de dados, e o bit menos significativo é fornecido por **cm**.

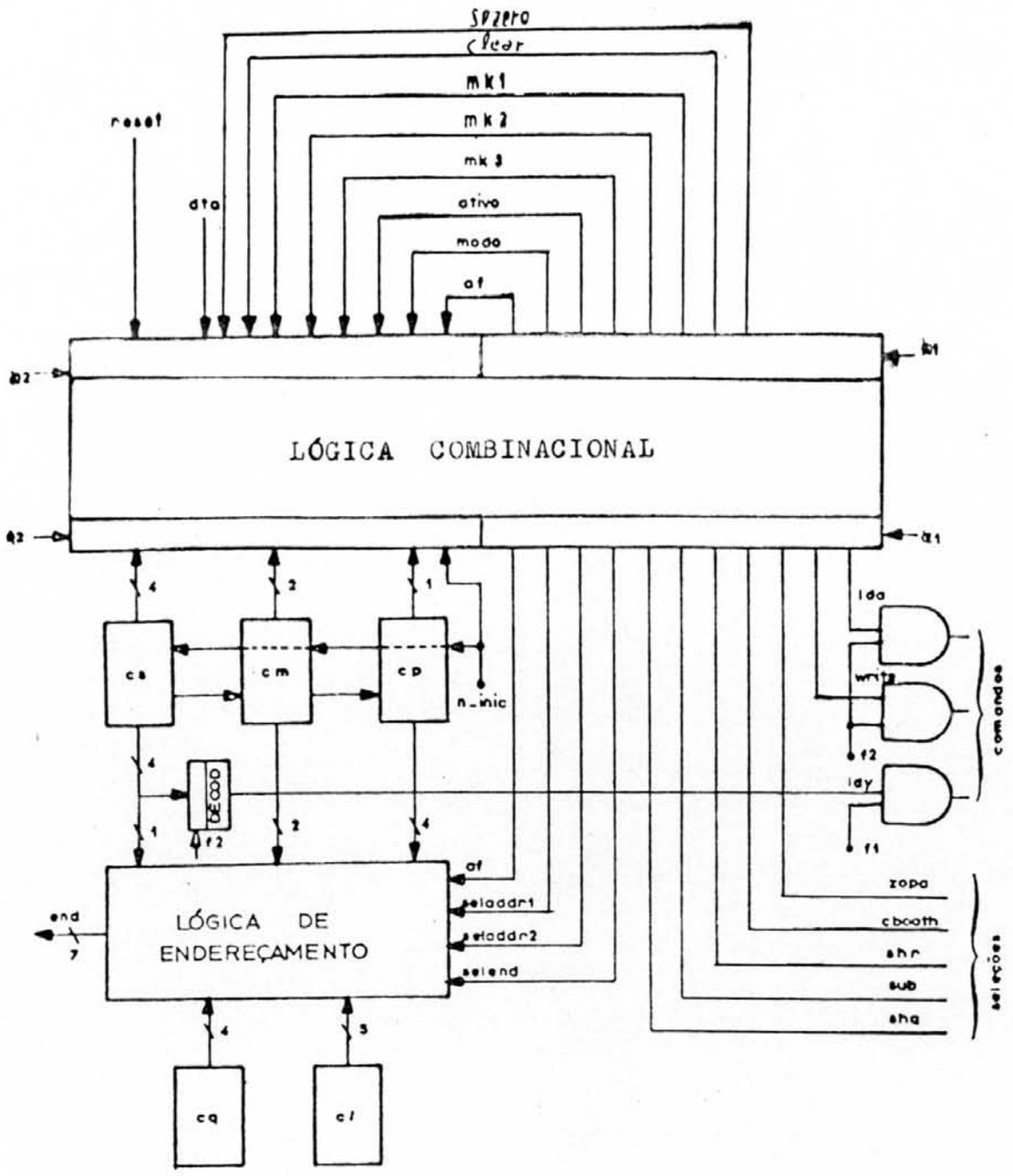


Figura 6.3 - Blocos da Parte de Controle.

O diagrama de estados da figura 6.4, extraído de [SAN 90], se refere ao sequenciamento das operações (entre processadores elementares), e os estados correspondem aos valores de mk3, mk2 e mk1.

O significado de cada estado é:

000 - inicialização

001 - "latches" de entrada → PE1

010 - PE1 → PE2

011 - PE2 → PE3

100 - PE3 → PE4

101 - PE4 → "latches" de saída

110 - nenhuma operação relevante

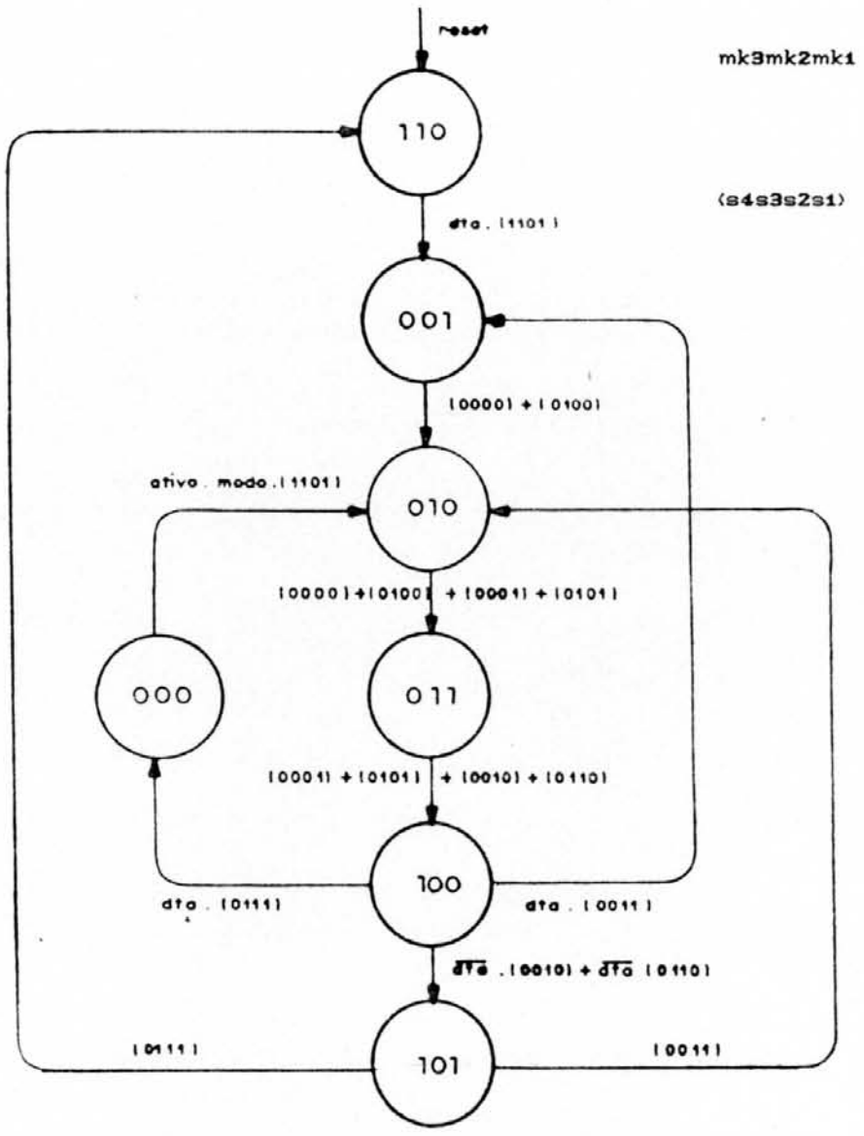


Figura 6.4 - Diagrama de estados.

Na inicialização do sistema, o estado (110) é posicionado, impedindo a escrita de dados em qualquer PE. Quando um valor de erro está disponível no "latch" de entrada, o sistema evolui para o estado (001), e ocorrem todas as escritas relevantes, a partir dos "latches" de entrada, na memória do primeiro PE.

Partindo-se do estado (110), a sequência (001) → (010) → (011) → (100) se refere à distribuição da parte real do valor de erro aos 4 PEs. O retorno à (001) refere-se à distribuição da respectiva parte imaginária, que terminada, deixa o sistema no estado (000). A transição (000) → (010) determina o início da soma de produtos acumulados.

A sequência (010) → (011) → (100) → (101) representa a soma das partes reais acumuladas. O retorno (101) → (010) representa o início da soma das partes imaginárias, que no final da evolução, deixa o sistema no estado (110), até os "latches" de entrada estarem com os conteúdos novamente disponíveis, podendo assim reiniciar o processo.

As entradas da máquina de estados que implementa o controlador dividem-se em três grupos: sinais externos (reset e dta, por exemplo), saída dos contadores e sinais de realimentação (estados e bits de sequenciamento). As saídas da máquina são sinais que selecionam funções da unidade aritmética (zopa, zopb, cbooth, shr, sub), selecionam endereços de operando (seldr1, seldr2, selend), estabelecem conexões (shq), ativam a escrita em elementos de memória (write, lda, ldy) e sinais de realimentação (mk3, mk2, mk1, ativo, modo, af, clear, spzero).

Os sinais de entrada e saída são:

a) Sinais de entrada.

- Entradas externas:

  reset, dta, n_inic

- Entradas que definem os estados:

  mk3in, mk2in, mk1in

- Entrada dos bits de sequenciamento:

  ativoin, modoin, afin, clearin, spzeroin

- Entrada a partir dos contadores:

  cpout, m2, m1, s4, s3, s2 s1

b) Sinais de saída:

- Bits dos estados:

  mk3, mk2, mk1

- Sequenciamento dos procedimentos básicos:

  ativo, modo, af, clear, spzero

- Sinais de controle:

  zopa, zopb, seldr1, seldr2, selend, cbooth, shr,
  sub, ldy, lda, write, shq

6.4 Implementação da Parte de Controle

Na implementação em um circuito integrado da parte de controle, foi feita a avaliação do desempenho considerando dois aspectos: área e atraso de propagação de sinais. O circuito é conhecido no nível lógico (equações booleanas, anexo 2), conforme especificado no trabalho de [SAN 90]. A partir daí, visou-se obter estimativas de desempenho, pelo uso de estilos diferentes de leiaute (PLA e *gate matrix*).

O esquema básico da máquina de estados, com disposição dos sinais de entrada e saída, que pertence à parte de controle, é mostrado na figura 6.5.

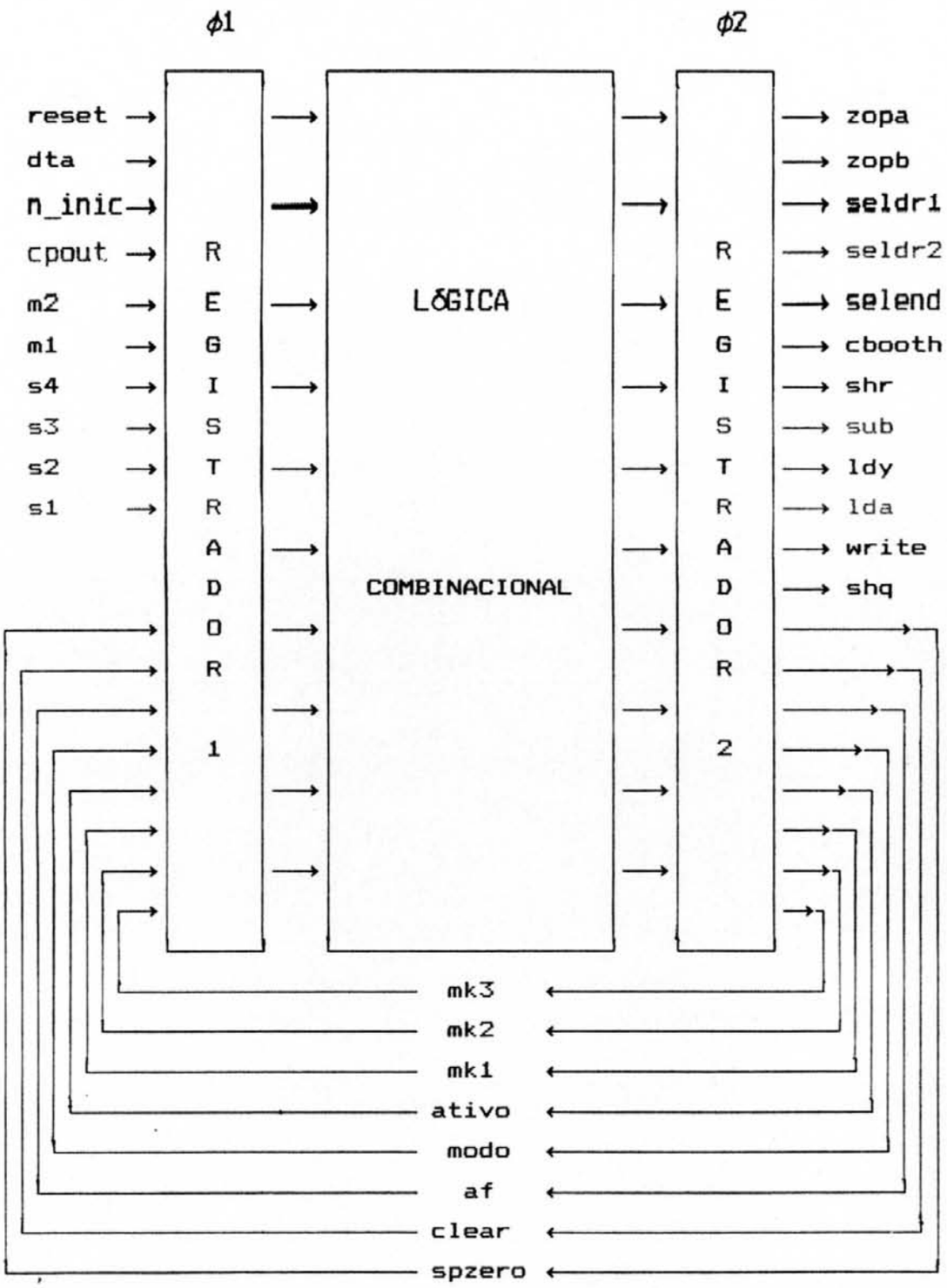


Figura 6.5 - Máquina de estados da Parte de Controle.

6.4.1 Implementação com PLA

A parte combinacional referente às equações booleanas em lógica multinível e listadas no anexo 2, pode ser sintetizada com lógica aleatória ou PLA.

Para o caso de PLA, as equações são reordenadas de forma a se ter uma lógica a dois níveis em termos de soma de produtos. Assim, o PLA obtido possui 18 entradas, 20 saídas e 116 termos-produto e sendo uma estrutura regular, é possível estimar o tamanho e atraso (capítulo 5), em função destes parâmetros. Para fins de comparação, é usado um PLA do sistema SOLO 2000 [SOL 88].

A tabela 6.1 compara resultados obtidos para o PLA estático, pseudo NMOS, e tecnologia de 2 μm, usando uma camada de metal.

Tabela 6.1 - PLA Estimado x PLA SOLO 2000

	Estimado	SOLO 2000
Área (mm^2)	0,901	0,955
Atraso (ns)	54,55	44,68

6.4.2 Implementação com Lógica Aleatória

Na implementação com lógica aleatória a área vai depender da forma final do leiaute gerado e é função do particionamento, posicionamento das células básicas (portas lógicas) e número de bandas. A geração automática do leiaute foi obtida com a ferramenta descrita no capítulo 5.

A avaliação elétrica é feita somando-se os atrasos individuais de cada porta, obtidos em função dos seus tempos de subida (tr) e descida (tf), para calcular o atraso total

das entradas para as saídas. O atraso nas linhas de polisilício que interligam os *gates* é desprezado, já que as diversas bandas apresentam altura reduzida. O maior atraso encontrado na parte combinacional, correspondente à figura 6.6a, é de 11,64 ns. A simulação elétrica da figura 6.6b valida o resultado, fornecendo 10,25 ns.

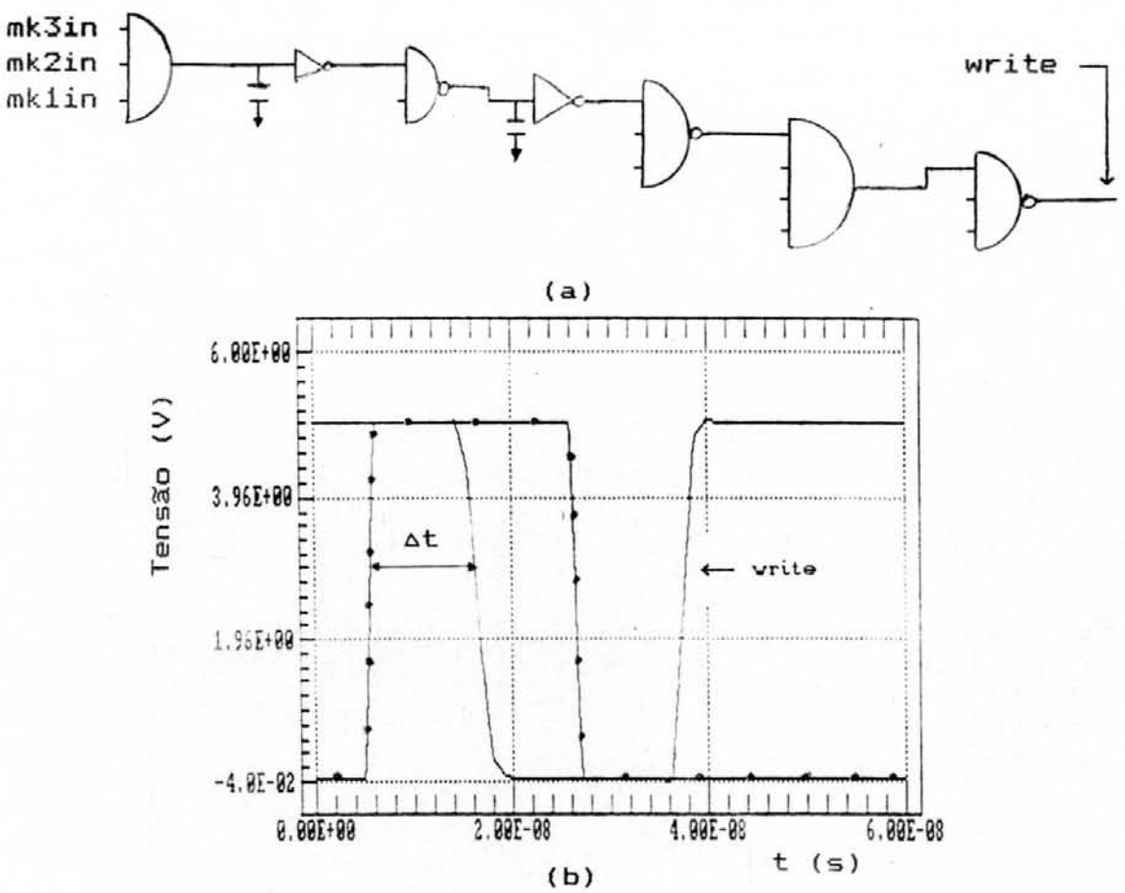


Figura 6.6 - Caminho crítico. (a) Portas lógicas; (b) Simulação.

Na síntese do leiaute, ao invés de usar uma lógica em termos de soma de produtos, optou-se por uma lógica nand-nand, pois apresenta menor consumo de área e menor atraso e utilizou-se portas com no máximo 4 entradas.

A tabela 6.2 mostra valores para as áreas obtidas no leiaute, para versões do circuito com diferente número de

bandas (espaço entre as linhas de alimentação, Vdd e Gnd), para a tecnologia de 2 micra (ECDM20-ES2) com dois níveis de metal. A figura 6.7 apresenta o leiaute, correspondente à versão com 6 bandas. O número total de transistores é 650. A síntese a nível de transistores foi feita em uma estação de trabalho SUN, devido à limitação de memória em máquinas IBM PC e compatíveis.

Tabela 6.2 - Área da Unidade de Controle em lógica aleatória

Número de bandas	Área (mm^2)	Transistores por mm^2
5	2,214	293,6
6	2,014	322,7
7	1,875	346,7
8	1,668	389,7
9	1,651	393,7
10	1,657	392,3
11	1,680	386,9

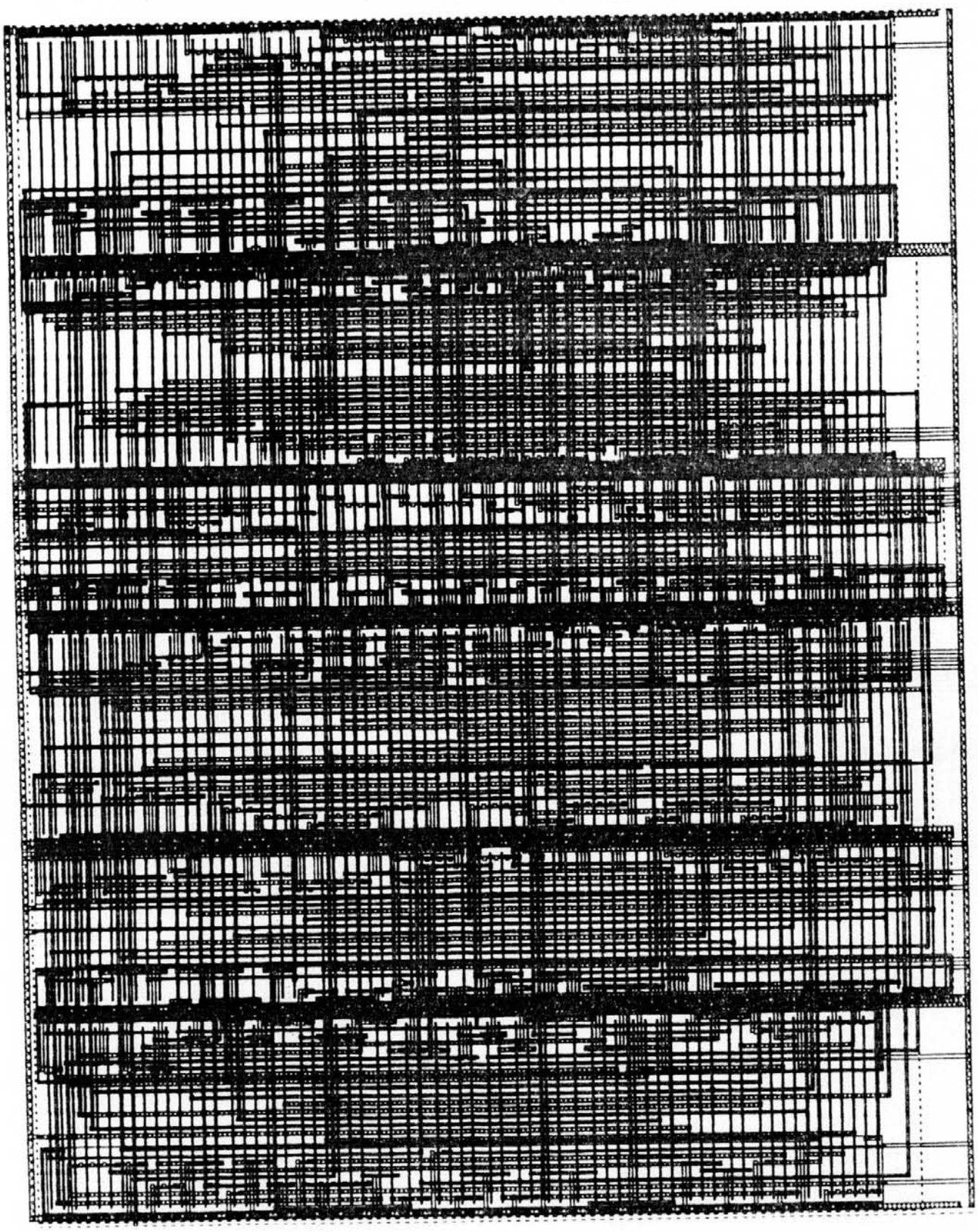


Figura 6.7 - Leiaute gate matrix da unidade de controle do equalizador.

6.4.3 Conclusão

A análise dos resultados anteriores mostra que a utilização de PLA fornece área menor e atraso maior. A redução em área foi em torno de 45 % em relação à versão em lógica aleatória com 9 bandas.

Os resultados das duas sub-seções anteriores, referem-se apenas à parte combinacional da parte de controle. A colocação dos elementos de memória (registradores) nas entradas e saídas, acionados pelas fases $\phi 1$ e $\phi 2$, pode ser feita de duas maneiras:

- Uso de transistores de passagem;
- Uso de flip-flops.

A opção por transistores de passagem, embora resulte em menor área, deverá ser feita manualmente, pois não se encontra disponível até o momento no GME-UFRGS, um sistema de geração automática para transistores de passagem. Se for escolhida a outra alternativa, a opção mais viável é fazer a síntese em *gate matrix*, usando-se flip-flops para implementar os elementos de memória (registradores R1 e R2, da figura 6.5). Para o caso, utilizou-se um flip-flp D, sensível à borda, conforme o diagrama lógico da figura 6.8.

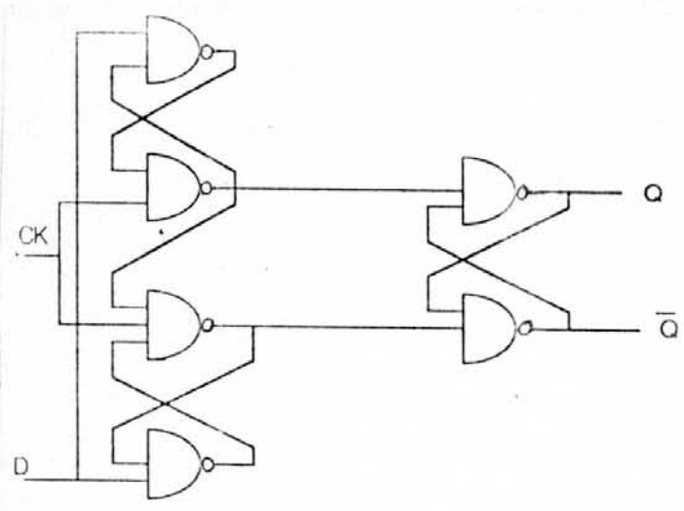


Figura 6.8 - Flip-flop D sensível à borda.

Se for considerado o atraso do flip-flop, a escolha da frequência para as fases $\phi 1$ e $\phi 2$, é feita considerando-se o atraso deste e o atraso na parte combinacional. Assim, são definidos os tempos abaixo:
Tlógica = 10,25 ns (atraso na lógica combinacional),
Tffd = 2,94 ns (atraso do flip-flop, thold + tsetup).

O período da fase $\phi 1$ deverá ser maior que Tlógica + Tffd. Se a fase $\phi 2$ for igual à fase $\phi 1$, para simplificar, a frequência da máquina será 1/(2(Tlógica + Tffd)). Ou, usando a nomenclatura do capítulo 2, Ta + Tb ≥ Tlógica + Tffd, e Tc = Td e Td = Tb, sendo f = 1/(Ta+Tb+Tc+Td).

A tabela 6.3 apresenta os valores para a máquina de estados completa, incluindo parte combinacional e elementos de memória, totalizando 1638 transistores. A descrição completa para a síntese, aparece no anexo 3. O leiaute final é semelhante ao da figura 6.7.

Tabela 6.3 - Área da Parte de Controle completa em lógica aleatória.

Número de bandas	Área (mm^2)	Transistores por mm^2
5	5,225	313,49
6	4,741	345,50
7	4,138	395,84
8	4,170	392,80
9	4,343	377,16
10	4,438	369,08
11	4,557	359,45

7. UM CONTROLADOR PARA UM CIRCUITO MULTIPLICADOR

Este capítulo apresenta um circuito e o controle bastante simples, para a multiplicação de dois números binários representados em sinal-magnitude. O produto obtido da multiplicação dos números de k bits cada, resulta em um número com $2k$ bits.

O processo da multiplicação por meio de somas e deslocamentos [SCO 85], para implementação em um sistema digital, pode ser ilustrado por um exemplo numérico.

Multiplicando: ———— 1010
Multiplicador: ———— 0101
1º bit do multiplicador = 1
cópia do multiplicando e deslocamento para a
direita, formando o 1º produto parcial ———— 01010
2º bit do multiplicador = 0
deslocamento para a direita do resultado
anterior, formando o 2º produto parcial ———— 001010
3º bit do multiplicador = 1
cópia do multiplicando ———— 1010
soma com o resultando anterior ———— 110010
deslocamento para a direita, formando o
3ºproduto parcial ———— 0110010
4º bit do multiplicador = 0
deslocamento para a direita do resultado
anterior, formando o produto final ———— 00110010

7.1 O Sistema Digital

A configuração básica do sistema que realiza a multiplicação, com inclusão dos sinais de controle, é apresentada na figura 7.1.

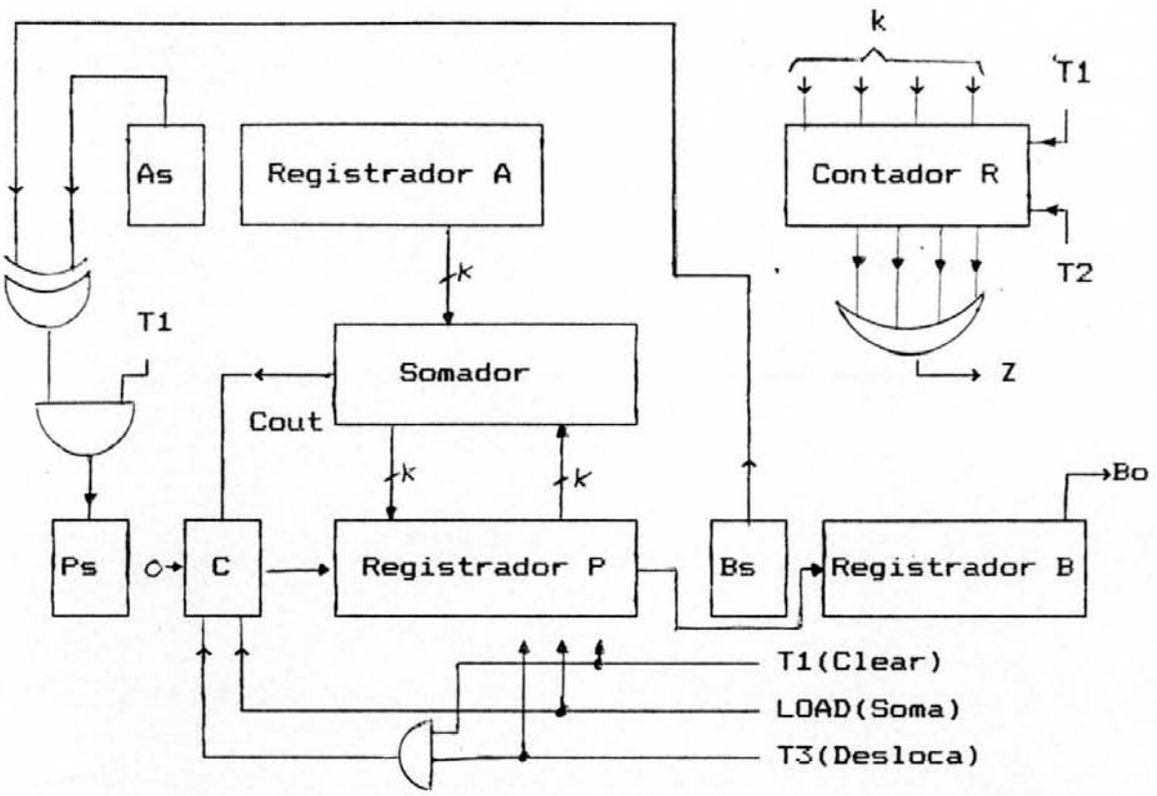


Figura 7.1 - Sistema para multiplicação binária.

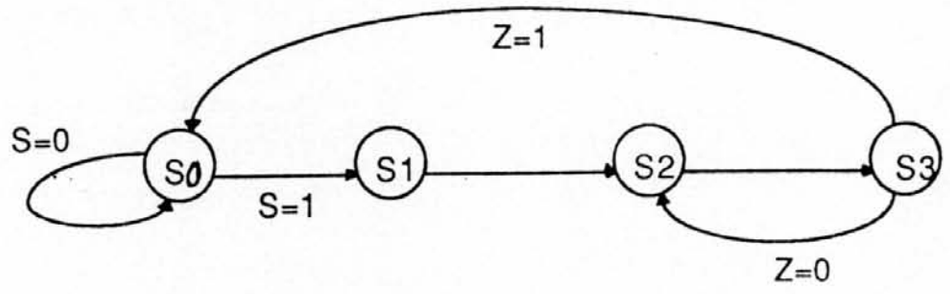
O multiplicando é armazenado no registrador A, o multiplicador é armazenado no registrador B, e o resultado parcial do produto é transferido para o registrador P. Os sinais do multiplicando, multiplicador e do resultado, estão respectivamente em As, Bs e Ps. O flip-flop C armazena o "vai um" resultante da adição dos registradores P e A. Os números apresentam n bits, onde o MSB é o sinal, e os $k = n - 1$ bits restantes, a magnitude. O contador R é inicialmente carregado com um número igual ao número de bits do multiplicador, e é decrementado após cada produto parcial. Quando seu conteúdo tornar-se zero, o produto final é armazenado nos registradores P e B, e o processo termina.

Quando o controle inicia a operação (S=1), a soma de A e P forma um produto parcial que é transferido para P. O "vai um" resultante da adição é transferido para C. O produto

parcial em P e o multiplicador em B, são deslocados um bit para a direita. O LSB de P, assume o lugar do MSB de B, e o "vai um" de C é deslocado para o MSB de P, e recebe 0. Após a operação de deslocamento, um bit do produto parcial está transferido para B, e o LSB de B, Bo, forma um sinal para a parte de controle. O outro sinal, Z', é resultante da verificação do conteúdo do contador.

7.2 A Parte de Controle

O diagrama de estados correspondente ao controle da multiplicação, com as operações envolvidas em cada estado, é mostrado na figura 7.2.



(a)

S0: Estado inicial
S1: P ← 0, C ← 0, R ← K, Ps ← As⊕Bs
S2: R ← R-1, P ← P + A, C ← Cout
S3: PB ← CPB, C ← 0

(b)

Figura 7.2 - Controle do multiplicador (a) Diagrama de estados; (b) Comandos para transferência entre registradores.

As operações envolvidas em cada estado são:

- Estado S0.

Permanece em S0, até S(start) tornar-se 1.

- Estado S1.

Quando S=1, o sistema vai para S1, e inicializa P, C e R, pelo comando T1. O sinal de As e Bs é comparado e transferido para Ps.

- Estado S2.

Após S1, no estado S2 o registrador R é decrementado e os conteúdos de P e A são somados, sendo o resultado armazenado em P e o "vai um " em C, Se Bo=1. Se Bo=0, P permanece inalterado. Os comandos associados são:

T2: R ← R − 1

LOAD = BoT2: P ← P + A, C ← Cout

- Estado S3.

O registrador composto de C, P e B, CPB, é deslocado para a direita, sendo que C recebe 0. O conteúdo de R é verificado. Se R=0, Z=1 e indica o final do processo. Se R≠0, Z=0 e é efetuado novo produto parcial a partir de S2.

Os sinais de entrada para o controle são os sinais Bo e Z, vindos da parte operativa, e a entrada externa S. Os comandos de saída são T0, T1, T2, T3 e LOAD.

Estabelecida a sequência de comandos gerados pela parte de controle, é necessário projetar o sistema sequencial correspondente, pelas técnicas clássicas conhecidas de projeto de sistemas digitais.

Assim, partindo-se do diagrama de estados e de uma tabela de estados, chega-se ao diagrama lógico da figura 7.3, que é um circuito Mealy e mostra o controle para a multiplicação binária, usando-se flip-flop D, sensível à borda.

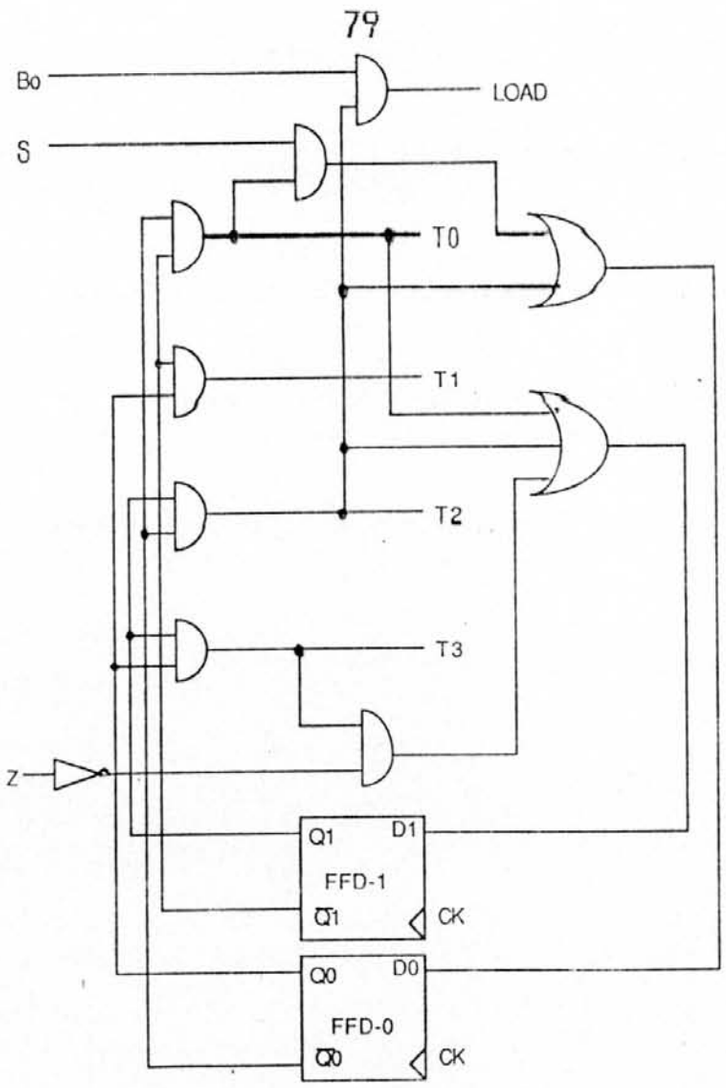


Figura 7.3 - Diagrama lógico da parte de controle do multiplicador.

7.3 Implementação da Parte de Controle

Definido o projeto lógico da unidade de controle, é preciso escolher a melhor alternativa para implementação em um circuito integrado, conforme mostra-se a seguir, seguindo os mesmos passos do capítulo anterior.

7.3.1 Com PLA

Conhecidos o número de entradas, saídas e monômios, torna-se possível estimar o desempenho. O PLA resultante

(portas lógicas da figura 7.3), apresenta 5 entradas (Q1, Q0, Bo, S e Z), 7 saídas (T0, T1, T2, T3, L, D1 e D0), e 7 monômios. O esquema elétrico aparece na figura 7.4. Na tabela 7.1 são dadas as áreas para o PLA (ECDM20 - ES2).

Tabela 7.1 - PLA da multiplicação

	Estimado	SOLO 2000
atraso (ns)	14,05	18,89
área (mm^2)	0,0495	0,0607

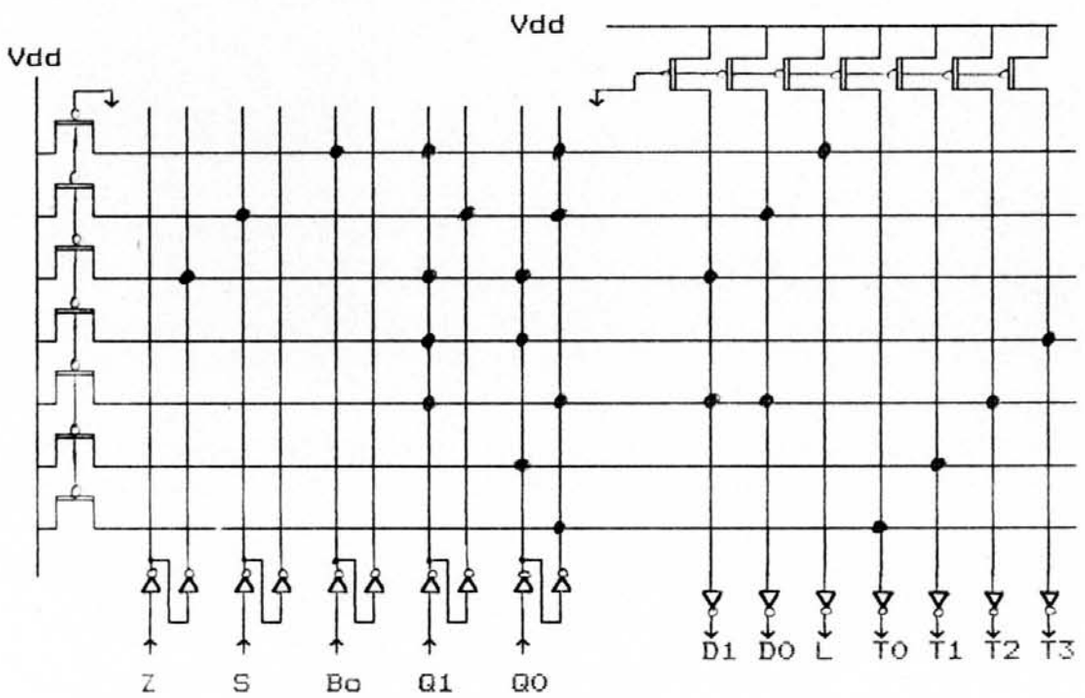


Figura 7.4 - Esquema elétrico do PLA de controle.

7.3.2 Com Lógica Aleatória

A tabela 7.2 apresenta os resultados para a implementação da parte de controle, usando-se agora portas

lógicas, sintetizadas no estilo *gate matrix*. Observa-se que em relação ao PLA estimado, houve um acréscimo de 31 % em área na parte combinacional, referente à versão com 3 bandas. O atraso no caminho mais longo da parte combinacional, obtido por modelo simplificado, é de 3,8 ns. A descrição para a síntese aparece no anexo 4 e o leiaute é mostrado na figura 7.5.

Tabela 7.2 - Áreas para a Parte de Controle do multiplicador

Bandas	Área (mm^2) Parte combinacional	Parte combinacional + flip-flops
1	0,0827	0,1377
2	0,0758	0,1754
3	0,0723	0,1901
4	0,0764	0,1447

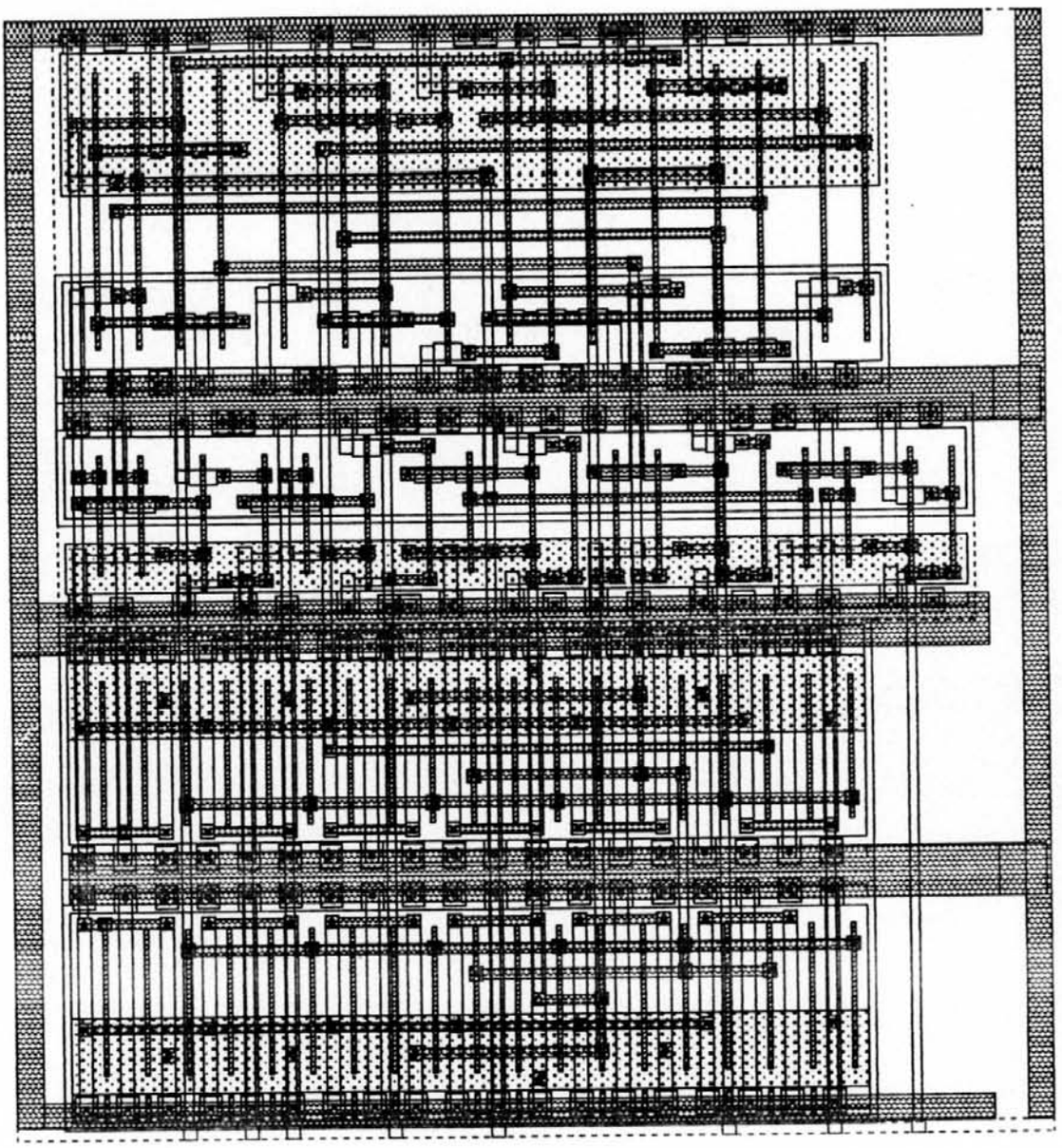


Figura 7.5 - Leiaute gate matrix da parte de controle da multiplicação.

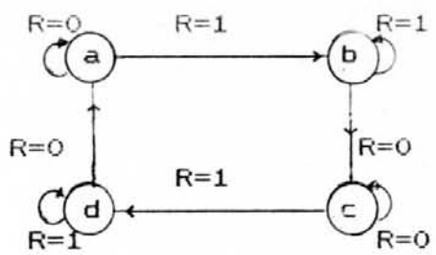
8. OUTROS EXEMPLOS

Este capítulo apresenta sumariamente os resultados da implementação de uma máquina de estados finitos hipotética e um somador serial. As comparações são feitas, também, usando-se PLAs e portas lógicas.

8.1 Exemplo 1

Este exemplo procura apenas apresentar as relações entre PLA e *gate matrix* para um circuito menor. Na prática, a máquina de estados, que é um circuito Moore, poderia estar representando uma unidade de controle simples.

A figura 8.1 apresenta a máquina de estados finitos, possuindo 4 estados, 1 entrada externa (R) e as saídas (S, A, e B).



(a)

$A = \bar{Q}1Q2$; $B = Q1\bar{Q}2$

$S = \bar{Q}1Q2 + Q1\bar{Q}2$

$D1 = \bar{R}Q2 + RQ1$

$D2 = \bar{R}Q2 + R\bar{Q}1$

(b)

Entrada R	Estados EP	Estados PE	SAÍDAS S	SAÍDAS A	SAÍDAS B
0	00	00	0	0	0
1	00	01	0	0	0
0	01	11	1	0	1
1	01	01	1	0	1
0	10	00	1	1	0
1	10	10	1	1	0
0	11	11	0	0	0
1	11	10	0	0	0

(c)

Figura 8.1 - Máquina sequencial (a) Diagrama de estados; (b) Equações; (c) Tabela de transição.

Na figura 8.2 aparece o esquema elétrico do PLA (estático). Para a realização da máquina sequencial, as saídas D1 e D2 deverão ser realimentadas de volta às entradas Q1 e Q2.

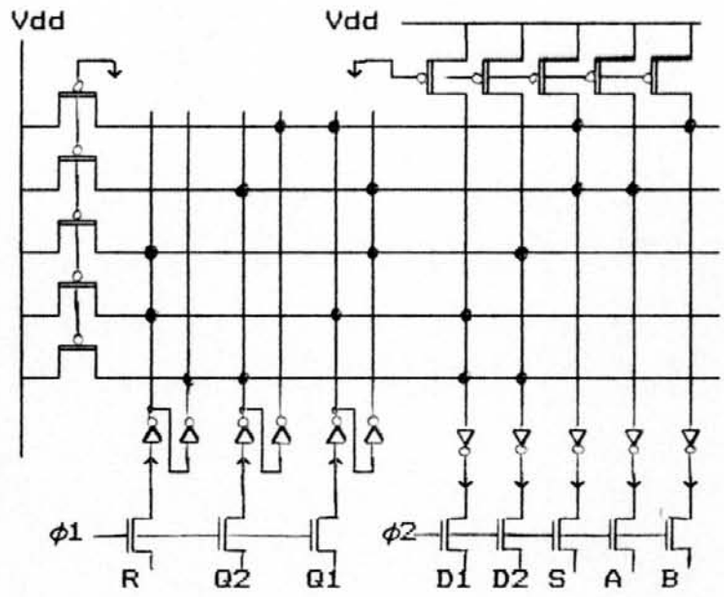


Figura 8.2 - PLA da máquina sequencial.

Na implementação em *gate matrix* (portas lógicas), o circuito (equações 8.1b) é novamente descrito à nível de transistores (formato SPICE), em uma lógica nand-nand, e segue-se a síntese automática. A tabela 8.1 apresenta os resultados, não incluindo barreira temporal (transistor de passagem). No PLA estão incluídos os *buffers* de entrada e saída. A não colocação dos *buffers* de entrada, em vista das dimensões reduzidas do PLA, não impede o seu funcionamento. Apenas leva a um desempenho elétrico pior.

Tabela 8.1 - Máquina de 4 estados

PLA	Estimado	SOLO 2000
atraso (ns)	12,47	18,21
área (mm^2)	0,0331	0,0433
Área (mm^2)	1 banda	0,0492
Gate	2 bandas	0,0638
Matrix	3 bandas	0,0556

8.2 Exemplo 2

Como segundo exemplo, é apresentado um somador serial, para duas entradas binárias sem sinal $X = x_n x_{n-1} \ldots x_0$ e $Y = y_n y_{n-1} \ldots y_0$ que produzem a saída $Z = z_n z_{n-1} \ldots z_0$. O bit de "vai um" resultante da adição é armazenado para o a soma no próximo ciclo. E assim, C_i, $z_i = x_i + y_i + C_{i-1}$.

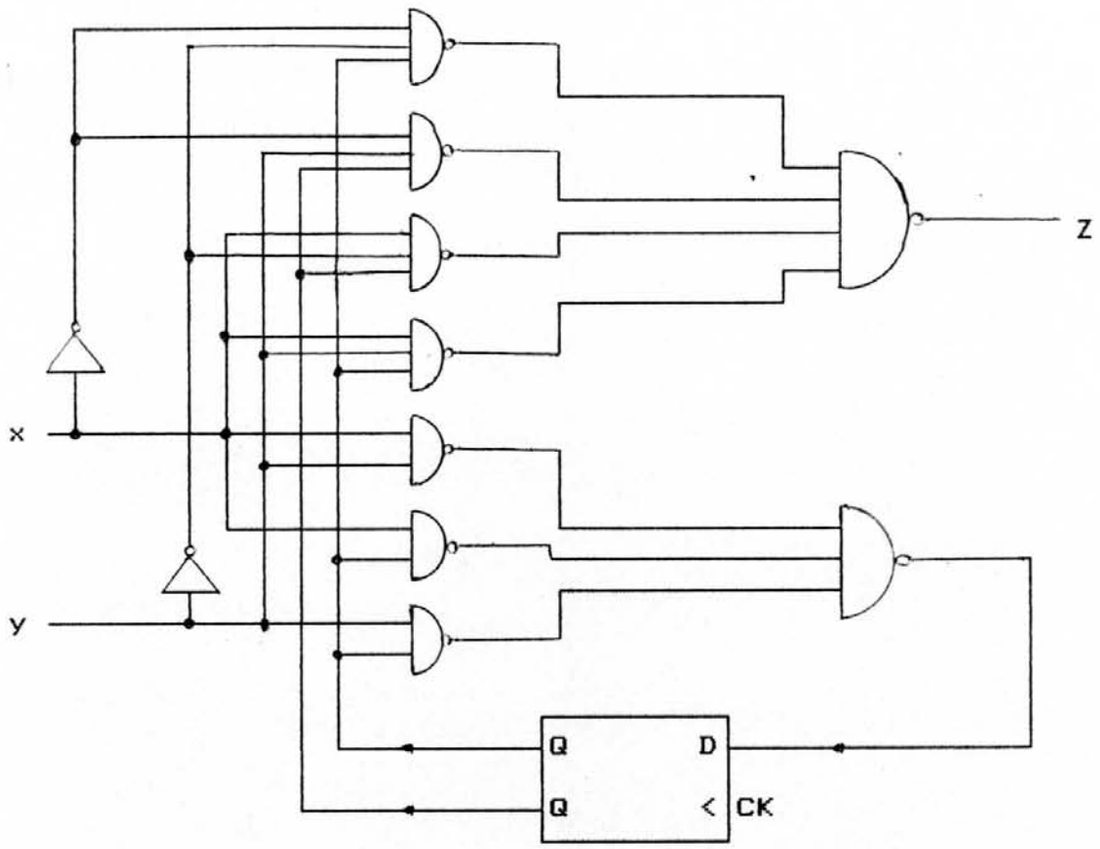
Para armazenar a informação do "vai um", o somador possui dois estados S1 e S0 (1 flip-flop), com a seguinte interpretação:

S0: "Vai um" $C_{i-1} = 0$, corresponde à Q = 0.
S1: "Vai um" $C_{i-1} = 1$, corresponde à Q = 1.

A figura 8.3 apresenta a tabela de estados e o diagrama lógico para o somador serial, usando uma lógica nand-nand.

Estado	Entrada xy 00	01	10	11
S0	S0,0	S0,1	S0,1	S0,0
S1	S0,1	S1,0	S1,0	S1,1

(a)



(b)

Figura 8.3 - Somador serial. (a)Tabela de estados; (b)Diagrama lógico.

A frequência na realização da soma vai depender do atraso na parte combinacional (2,5 ns com portas nand) e na memória (2,94 ns). Na tabela 8.2 são resumidos os resultados. O PLA refere-se à parte combinacional (nand-nand transformada em soma de produtos).

Tabela 8.2 - Somador serial

PLA	Estimado	SOLO 2000
atraso (ns)	10,15	14,73
área (mm^2)	0,0312	0,0403
Área (mm^2)	1 banda	0,0654
Gate	2 bandas	0,0615
Matrix	3 bandas	0,0524

CONCLUSÃO

O trabalho desenvolvido apresenta duas fases distintas:

- Pesquisa e apresentação de modelos simplificados de *timing* para circuitos integrados (redes RC).

- Avaliação de circuitos (partes de controle) implementados com diferentes estilos de leiaute.

A primeira etapa procura resumir modelos básicos para determinação de atrasos de propagação de sinais em circuitos integrados, aplicáveis para estruturas regulares (ROM, PLA) e portas lógicas. Apesar da simplicidade dos modelos (desconsideração da não-linearidade do transistor), obteve-se precisão razoável em relação a simulações elétricas.

Em trabalhos futuros, os modelos poderão ser empregados na construção de simuladores de *timing*, e no desenvolvimento de algoritmos para determinação de caminhos críticos, possibilitando um rápido dimensionamento de transistores, facilitando posterior simulação elétrica.

A segunda etapa procura fornecer relações de área e velocidade para alguns circuitos, visando a implementação com PLA e comparando com a síntese em *gate matrix*.

Considerando os exemplos analisados:

- Quanto ao desempenho elétrico:
  A implementação com portas lógicas forneceu em todos os casos uma frequência de operação maior que o PLA.

- Quanto ao consumo de área:
  A área estimada visando a utilização de PLA foi menor em relação à síntese em lógica aleatória utilizando *gate*

matrix. Este resultado refere-se ao gerador de lógica aleatória desenvolvido no GME-UFRGS, que na versão atual não faz compactação de leiaute.

Dos circuitos analisados, quando implementados com PLA, o equalizador apresentou a maior redução de área (tabela).

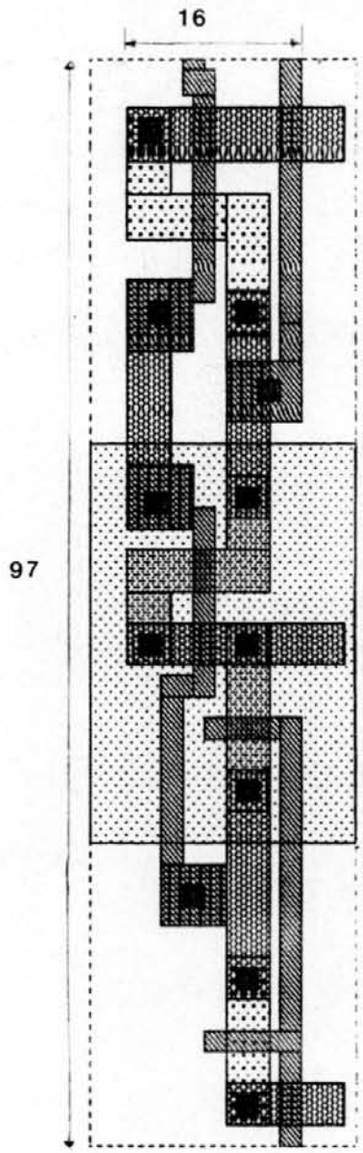
	Redução de área (%)
Equalizador	- 45
Multiplicador	- 31
Máquina de 4 estados	- 32
Somador serial	- 40

A partir dos exemplos analisados, apesar de reduzido número, como conclusão sugere-se a utilização de PLA para circuitos bastante grandes e pequeno número de sinais de saída e que possam operar a frequências baixas, sendo a lógica aleatória reservada para circuitos menores e rápidos.

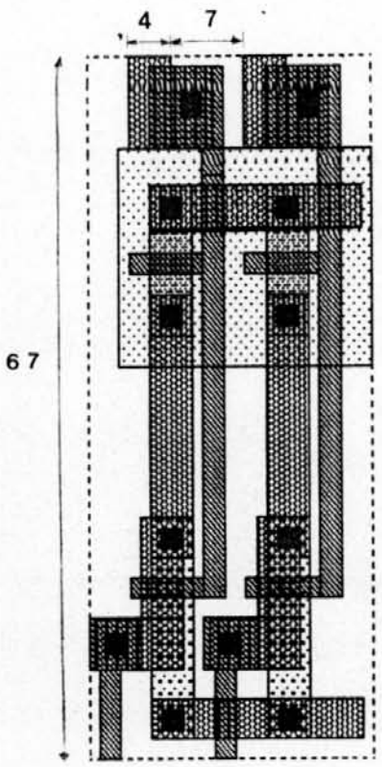
Além dos resultados particulares acima apresentados, cabe salientar ainda outras vantagens geralmente presentes na utilização de PLAs e estruturas regulares, para o projeto de unidades de controle e CIs:

- menor tempo de concepção;
- maior facilidade de correção de erros;
- maior facilidade para realização de testes;
- propicia melhor evolução do circuito;
- maior facilidade de automação do processo de concepção;
- melhor estimativa para *floorplanning*.

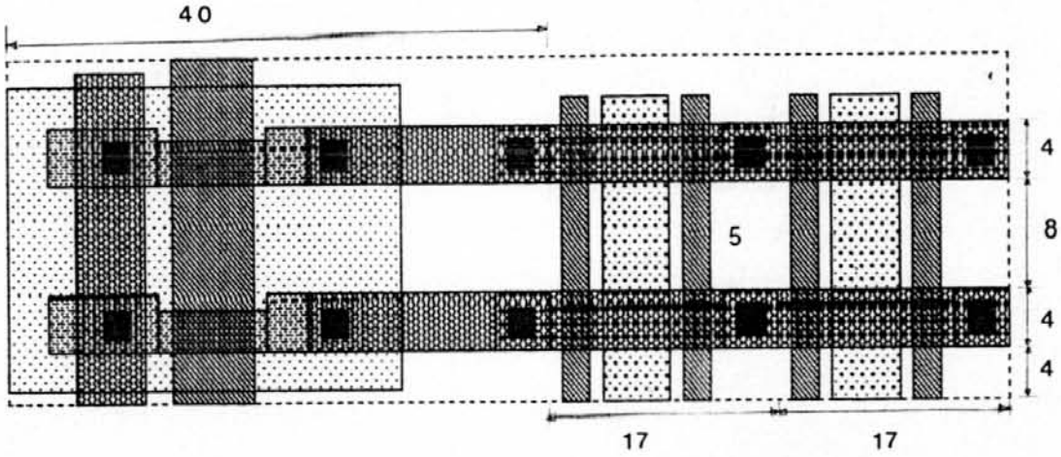
ANEXO 1 - Células básicas do PLA



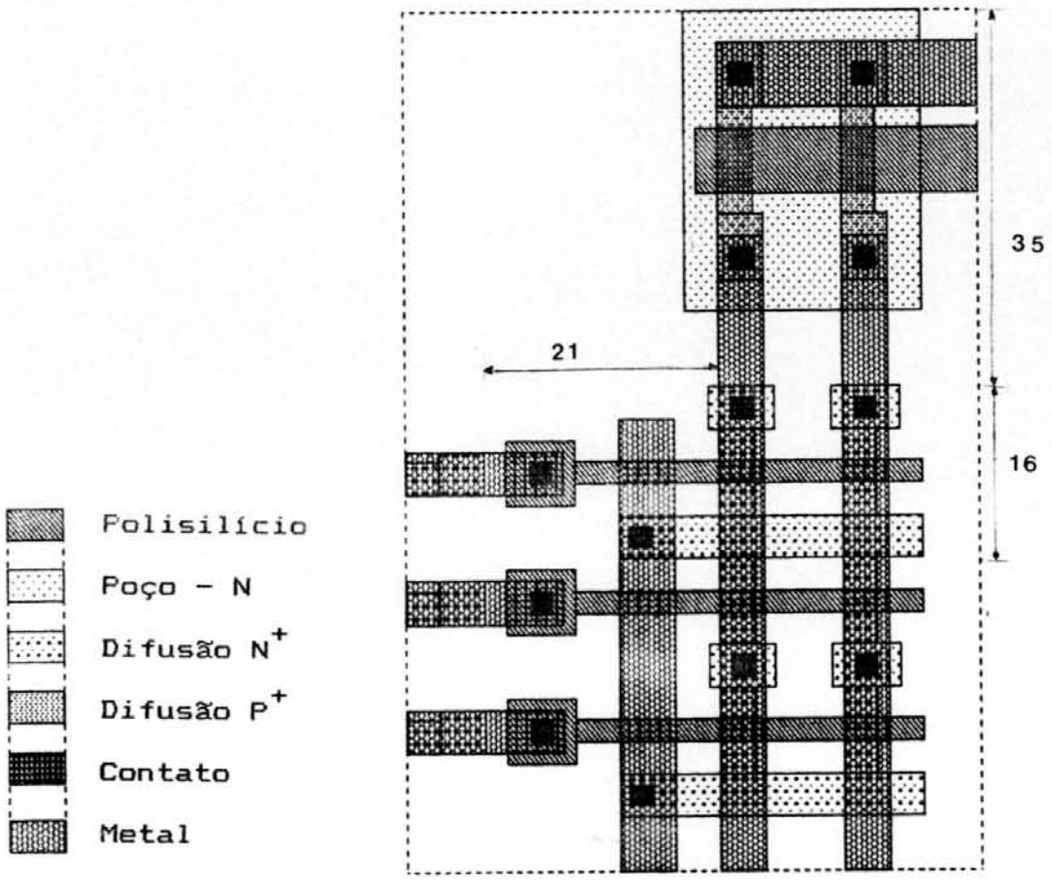
Buffer e inversor de entrada



Inversor de saída



Plano E



Plano OU

ANEXO 2 - Equações da Parte de Controle do equalizador .

```
mk1 = ((mk3in AND mk2in AND NOTmk1in AND dta AND  s4  AND  s3
      AND NOTs2 AND  s1)  OR     (NOTmk3in  AND  mk2in  AND
      NOTmk1in)  OR  (mk3in AND NOTmk2in  AND  NOTmk1in  AND
      dta AND NOTs4 AND NOTs3 AND s2 AND s1)  OR (mk3in  AND
      NOTmk2in AND NOTmk1in AND NOT dta ))AND reset;

mk2 = ((mk3in AND mk2in AND NOTmk1in AND NOT(dta AND  s4  AND
      s3 AND NOTs2 ANd s1))  OR  (NOTmk3in ANd NOTmk2in  AND
      mk1in) OR (NOTmk3in AND mk2in AND  NOTmk1in) OR (mk3in
      AND NOTmk2in ANd  mk1in) OR (NOTclearin  ANd  NOTmk3in
      AND NOTmk2in AND NOTmk1in AND NOTatin AND  modoin  AND
      s4 AND s3 AND NOTs2 AND s1) OR (clearin AND cpout  AND
      m2 AND m1 AND s4 AND s3  AND  NOTs2  AND  NOTs1))  AND
      reset;

mk3 = ((mk3in AND mk2in AND NOTmk1in AND NOT(dta AND  s4  AND
      s3  AND  NOTs2  AND  s1)) OR (NOTmk3in  AND  mk2in  AND
      mk1in) OR (mk3in AND NOTmk2in AND NOTmk1in AND NOTdta)
      OR (mk3in AND NOTmk2in AND mk1in AND NOTs4 AND  s3  AND
      s2 AND s1) OR (clearin AND cpout AND m2 ANd m1  AND  s4
      AND s3 AND NOTs2 AND NOTs1)) AND reset;

ativo = ((atin AND NOT(cpout AND m2 AND m1 AND s4 AND s3  AND
      NOTs2 AND NOTs1)) OR (NOT atin AND NOTmodoin  AND  afin
      AND NOTs4 AND s3 AND s2 AND s1) OR (mk3in AND  NOTmk2in
      AND NOTmk1in AND NOT atin AND modoin AND dta AND  NOTs4
      AND s3 AND s2 AND s1)) AND reset;

modo = (modoin  AND  NOT(mk3in  AND  NOTmk2in  AND   NOTmk1in
      AND  NOTatin  AND  dta   AND   NOTs4   ANDs3   AND   s2
      AND s1)) OR (NOTmodoin AND NOTatin AND afin  AND  NOTs4
      AND s3 AND s2 AND s1) OR NOTreset;

af  = ((afin AND (NOTs3 OR NOTs2 OR  NOTs1)) OR (NOTatin  AND
```

```
      NOTmodoin AND dta AND s4 AND s3 AND NOTs2  AND  NOTs1))
      AND)) AND reset;

clear = NOTreset OR clearin AND NOT(cpout AND m2 AND  m1  AND
      s4 AND s3 AND NOTs2 AND NOTs1));

spzero = (NOT atin AND NOTmodoin AND afin AND  NOTs4  AND  s3
      AND  s2 AND s1  OR  spin AND   NOT(atin AND modoin  AND
      NOTafin  AND  m2 AND NOTm1 AND s4 AND s3 AND NOTs2  AND
      s1)) AND reset;

zopa = ((mk3in OR mk2in OR  mk1in) AND dta) OR (NOTmk3in  AND
      NOTmk2in  AND NOTmk1in AND NOTatin AND  modoin  AND  s4
      AND  s3 AND NOTs2 AND s1) OR (mk3in  AND  NOTmk2in  AND
      mk1in  AND NOTs4 AND NOTs3 AND s2  AND  s1) OR afin  OR
      (NOTmk3in  AND NOTmk2in AND NOTmk1in AND NOTafin AND s4
      AND s3 AND NOTs2 AND s1) OR NOTreset OR clin;

zopb = spin AND NOTm2 AND NOTm1 AND s4 AND NOTs3 AND  s2  AND
      s1 OR spin AND m2 AND NOTm1 AND s4 AND NOTs3 AND s2 AND
      s1 OR NOTreset OR clin;

cbooth = (NOTmk3in AND NOTmk2in AND NOTmk1in AND NOTafin  AND
      atin AND (NOTs4 OR s3 OR NOTs2 OR NOTs1) AND (NOTs4  OR
      NOTs3 OR s2 OR s1)) OR (spin AND NOTm2 AND NOTm1 AND s4
      AND NOTs3 AND s2 AND s1) OR (spin AND m2 AND  NOTm1 AND
      s4 AND NOTs3 AND s2 AND s1) OR NOTreset OR clin;

shr = NOTmk3in AND NOTmk2in AND NOTmk1in AND NOTafin AND atin
      AND (NOTs4 OR s3 OR NOTs2 OR s1)  AND  (NOTs4 OR s3 OR
      NOTs2 OR NOTs1) AND (NOTs4 OR NOTs3 OR s2 OR s1);

sub = NOTmk3in AND NOTmk2in AND NOTmk1in AND NOTafin AND atin
      AND s4 AND NOTs3 AND s2 AND s1 AND ((modoin  AND  NOTm2
      AND m1) OR (NOT modoin AND (NOTm2 OR NOTm1)));
```

```
lda = (NOT dta AND (mk3in OR mk2in OR mk1in) OR (NOTmk3in AND
      NOT mk2in AND NOTmk1in AND (NOTs4 OR s3 OR NOTs2 OR NOT
      s1) AND (NOTs4 OR NOTs3 OR s2 OR s1));

shq = ((afin AND (NOTs3 OR NOTs2 OR NOTs1)) OR (NOTatin AND
      NOTmodoin AND dta AND s4 AND s3 AND NOTs2 AND NOT s1)
      OR (NOTatin AND modoin AND s4 AND s3 AND NOTs2 AND s1)
      OR mk3in OR mk2in OR mk1in) AND NOT(NOTreset OR clin);

sedr1 = (atin AND ((NOTmodoin AND s4 AND NOTs3 AND s2 AND s1)
      OR modoin)) OR (atin AND modoin AND cpout AND m2 AND m1
      AND s4 AND s3 AND NOTs2 AND s1) OR (NOTatin AND modoin
      AND NOTdta) OR (spin AND NOTm2 AND NOTm1 AND s4 AND
      NOTs3 AND s2 AND s1) OR (spin AND m2 AND NOTm1 AND s4
      AND NOTs3 AND s2 AND s1) OR NOTreset OR clin;

sedr2 = ((atin AND modoin AND s4 AND NOTs3 AND s2 AND s1) OR
      (atin AND modoin AND cpout AND m2 AND m1 AND s4 AND s3
      AND NOTs2 AND s1) OR (NOTatin AND modoin AND NOTdta) OR
      (spin AND NOTm2 AND NOTm1 AND s4 AND NOTs3 AND s2 AND
      s1) OR (spin AND m2 AND NOTm1 AND s4 AND NOTs3 AND s2
      AND s1)) AND NOT(NOTreset OR clin);

selend = atin AND s4 AND s3 AND NOTs2 AND NOTs1 AND
      NOT(NOTreset OR clin):

write = ((mk3in OR mk2in OR mk1in) AND dta AND NOT(NOTs4 AND
      s3 AND s2 AND s1) AND NOT(mk3in ANd mk2in ANd
      NOTmk1in)) OR (mk3in AND mk2in ANd NOTmk1in AND dta AND
      s4 AND s3 AND NOTs2 AND s1) OR (NOTmk3in AND NOTmk2in
      AND NOTmk1in AND ninic AND ((atin AND s4 AND NOTs3 AND
      s2 AND s1) OR (afin AND NOTs1))) OR (spin AND NOTm2 AND
      NOTm1 AND s4 AND NOTs3 AND s2 AND s1) OR (spin AND m2
      AND NOTm1 AND s4 AND NOTs3 AND s2 AND s1) OR NOTreset
      OR clin;
ldy = s4 AND s3 AND NOTs2 AND NOTs1;
```

ANEXO 3 - Descrição SPICE do equalizador

```
* CIRCUITO máquina de estados do equalizador
* CELULA: INVERSOR
.SUBCKT INV IN OUT vcc
MP1  OUT  IN vcc vcc PMOS L=2.0U W=6.0U
MN2  OUT  IN 0   0   NMOS L=2.0U W=3.0U
.ENDS INV
*
* CELULA: AND 2 ENTRADAS
.SUBCKT AND2 I1 I2 OUT vcc
MP1    2 I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2    2 I2  vcc  vcc PMOS L=2.0U W=6.0U
MN4    2 I1    3  0   NMOS L=2.0U W=3.0U
MN5    3 I2    0  0   NMOS L=2.0U W=3.0U
MN6  OUT  2    0  0   NMOS L=2.0U W=3.0U
MP3  OUT  2  vcc  vcc PMOS L=2.0U W=6.0U
.ENDS AND2
*
* CELULA: AND 3 ENTRADAS
.SUBCKT AND3 I1 I2 I3 OUT vcc
MP1  2    I1 vcc vcc PMOS L=2.0U W=6.0U
MP2  2    I2 vcc vcc PMOS L=2.0U W=6.0U
MP3  2    I3 vcc vcc PMOS L=2.0U W=6.0U
MN4  2    I1 3  0    NMOS L=2.0U W=3.0U
MN5  3    I2 4  0    NMOS L=2.0U W=3.0U
MN6  4    I3 0  0    NMOS L=2 0U W=3.0U
MP7  OUT  2  vcc vcc PMOS L=2.0U W=6.0U
MN8  OUT  2  0   0   NMOS l=2.0U W=3.0U
.ENDS AND3
*
* CELULA: AND 4 ENTRADAS
.SUBCKT AND4 I1 I2 I3 I4 OUT vcc
MP1  2    I1 vcc vcc PMOS L=2.0U W=6.0U
MP2  2    I2 vcc vcc PMOS L=2.0U W=6.0U
MP3  2    I3 vcc vcc PMOS L=2.0U W=6.0U
```

```
MP4  2    I4 vcc vcc PMOS L=2.0U W=6.0U
MN5  2    I1 3   0   NMOS L=2.0U W=3.0U
MN6  3    I2 4   0   NMOS L=2.0U W=3.0U
MN7  4    I3 5   0   NMOS L=2.0U W=3.0U
MN8  5    I4 0   0   NMOS L=2.0U W=3.0U
MP9  OUT  2  vcc vcc PMOS L=2.0U W=6.0U
MN10 OUT  2  0   0   NMOS L=2.0U W=3.0U
.ENDS AND4
*
* CELULA: NAND 2 ENTRADAS
.SUBCKT NAND2 I1 I2 OUT vcc
MP1  OUT I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2  OUT I2  vcc  vcc PMOS L=2.0U W=6.0U
MN3  OUT I1    3  0   NMOS L=2.0U W=3.0U
MN4    3 I2    0  0   NMOS L=2.0U W=3.0U
.ENDS NAND2
*
* CELULA: NAND 3 ENTRADAS
.SUBCKT NAND3 I1 I2 I3 OUT vcc
MP1  OUT  I1 vcc vcc PMOS L=2.0U W=6.0U
MP2  OUT  I2 vcc vcc PMOS L=2.0U W=6.0U
MP3  OUT  I3 vcc vcc PMOS L=2.0U W=6.0U
MN4  OUT  I1 3   0   NMOS L=2.0U W=3.0U
MN5  3    I2 4   0   NMOS L=2.0U W=3.0U
MN6  4    I3 0   0   NMOS L=2 0U W=3.0U
.ENDS NAND3
*
* CELULA: NAND 4 ENTRADAS
.SUBCKT NAND4 I1 I2 I3 I4 OUT vcc
MP1  OUT  I1 vcc vcc PMOS L=2.0U W=6.0U
MP2  OUT  I2 vcc vcc PMOS L=2.0U W=6.0U
MP3  OUT  I3 vcc vcc PMOS L=2.0U W=6.0U
MP4  OUT  I4 vcc vcc PMOS L=2.0U W=6.0U
MN5  OUT  I1 3   0   NMOS L=2.0U W=3.0U
MN6  3    I2 4   0   NMOS L=2.0U W=3.0U
MN7  4    I3 5   0   NMOS L=2.0U W=3.0U
```

```
MN8  5    I4 0  0    NMOS L=2.0U W=3.0U
.ENDS NAND4

* CELULA: FLIP FLOP D SENSIVEL A BORDA
.SUBCKT FFD D CK Q ~Q VCC
MN1  S1  S4  2      0  NMOS  L=2.0U W=3.0U
MN2   2  S2  GND    0  NMOS  L=2.0U W=3.0U
MN3  S2  S1  3      0  NMOS  L=2.0U W=3.0U
MN4   3  CK  GND    0  NMOS  L=2.0U W=3.0U
MN5  S3  S2  4      0  NMOS  L=2.0U W=3.0U
MN6   4  CK  5      0  NMOS  L=2.0U W=3.0U
MN7   5  S4  GND    0  NMOS  L=2.0U W=3.0U
MN8  S4  S3  6      0  NMOS  L=2.0U W=3.0U
MN9   6   D  GND    0  NMOS  L=2.0U W=3.0U
MN10  Q  S2  7      0  NMOS  L=2.0U W=3.0U
MN11  7  ~Q  GND    0  NMOS  L=2.0U W=3.0U
MN12 ~Q  S3  8      0  NMOS  L=2.0U W=3.0U
MN13  8   Q  GND    0  NMOS  L=2.0U W=3.0U
MP14 S1  S4  VCC  VCC  PMOS  L=2.0U W=6.0U
MP15 S1  S2  VCC  VCC  PMOS  L=2.0U W=6.0U
MP16 S2  S1  VCC  VCC  PMOS  L=2.0U W=6.0U
MP17 S2  CK  VCC  VCC  PMOS  L=2.0U W=6.0U
MP18 S3  S2  VCC  VCC  PMOS  L=2.0U W=6.0U
MP19 S3  CK  VCC  VCC  PMOS  L=2.0U W=6.0U
MP20 S3  S4  VCC  VCC  PMOS  L=2.0U W=6.0U
MP21 S4  S3  VCC  VCC  PMOS  L=2.0U W=6.0U
MP22 S4   D  VCC  VCC  PMOS  L=2.0U W=6.0U
MP23  Q  S2  VCC  VCC  PMOS  L=2.0U W=6.0U
MP24  Q  ~Q  VCC  VCC  PMOS  L=2.0U W=6.0U
MP25 ~Q  S3  VCC  VCC  PMOS  L=2.0U W=6.0U
MP26 ~Q   Q  VCC  VCC  PMOS  L=2.0U W=6.0U
.ENDS FFD
******* CHAMADA DOS SUBCIRCUITOS *****
X1 mk3in mk2in mk1inb n1      vcc and3
X2 s4 s3 s2b s1 n2            vcc and4
X3 mk3inb mk2in mk1inb  n3    vcc nand3
```

```
X4 mk3in mk2inb mk1inb  n4     vcc and3
X5 s4b s3b s2 s1 n5            vcc and4
X6 dta n1 n2 n6                vcc nand3
X7 n4 dta n5 n7                vcc nand3
X8 dtab n4       n8            vcc nand2
X9 n6 n3 n7 n8 n9              vcc nand4
X10 reset n9      MK1          vcc and2
X11 dta n2  a0                vcc nand2
X12 a0  n1  a1                vcc nand2
X13 mk3inb mk2inb mk1in a2    vcc nand3
X14 mk3in mk2inb mk1in a3     vcc nand3
X15 atinb moin a4             vcc  and2
X16 mk3inb mk2inb mk1inb a5   vcc and3
X17 a4 clinb n2 a5     a6     vcc nand4
X18 cpout m2 m1 a7            vcc and3
X19 s4 s3 s2b s1b   a8        vcc and4
X20 a7  clin  a8   a9        vcc  nand3
X21 a1 a2 a3 a6 a11           vcc and4
X22 a11 a9 n3 a10             vcc nand3
X23 a10 reset  MK2            vcc and2
X24 mk3inb mk2in mk1in b1     vcc nand3
X25 s4b s3 s2 s1 b2           vcc and4
X26 a3 b2 b3                  vcc nand2
X27 a1 b1 n8 b3 b4            vcc and4
X28 b4 a9 b5                  vcc nand2
X29 b5 reset  MK3             vcc and2
X30 a8 a7 c1                  vcc nand2
X31 atin c1   c2              vcc nand2
X32 b2 atinb moinb afin c3    vcc nand4
X33 b2 n4 dta atinb c5        vcc  and4
X34 c5 moin c4                vcc  nand2
X35 c2 c3 c4 c6               vcc  nand3
X36 c6 reset  ATIVO           vcc and2
X37 c5 d1                     vcc  inv
X38 d1 moin  d2               vcc  nand2
X39 reset d2 c3  MODO         vcc  nand3
```

```
X40 a8 atinb moinb dta e1  vcc nand4
X41 s3 s2 s1 e2            vcc nand3
X42 e2 afin e3             vcc nand2
X43 e1 e3 e4               vcc nand2
X44 reset e4 AF            vcc and2
X45 c1 clin t1             vcc nand2
x46 t1 reset CLEAR         vcc nand2
X47 n2 m2 m1b  afinb f1    vcc and4
X48 f1 atin moin  f2       vcc nand3
X49 spin  f2  f3           vcc nand2
X50 c3 f3   f4             vcc nand2
X51 reset f4 SPZERO        vcc and2
X52 a5 g1             vcc inv
X53 g1 dta  g2        vcc nand2
X54 a4 n2 a5 g3       vcc nand3
x55 a3 g7             vcc inv
X56 g7 n5 g4          vcc nand2
X57 n2 a5 afinb g5    vcc nand3
X58 g2 g3 g4 g5 g6    vcc and4
X59 g6 afinb reset clinb  ZOPA  vcc nand4
X60 s4 s3b s2 s1 h1      vcc and4
X61 h1 spin m1b h2       vcc nand3
X62 reset clinb h2 ZOPB vcc nand3
X63 a8 i1                vcc inv
X64 h1 i2                vcc inv
X65 i1 i2 a5 atin i3     vcc and4
X66 i3 afinb i4          vcc nand2
X67 h2 reset clinb i4  CBOOTH vcc nand4
X68 i4 j1             vcc inv
X69 s4 s3b s2 s1b j2 vcc nand4
X70 j1 j2 SHR         vcc and2
X71 h1 a5 atin afinb k1 vcc and4
X72 moin m2b m1 k2       vcc nand3
X73 m2 m1 k3             vcc nand2
X74 moinb k3  k4         vcc nand2
X75 k2 k4  k5            vcc nand2
```

```
X76 k1 k5  SUB          vcc and2
X77 i1 i2 a5 l1   vcc nand3
X78 a5 l2         vcc inv
X79 l2 dtab  l3   vcc nand2
X80 l1 l3    LDA  vcc nand2
X81 e2 afin mm1     vcc nand2
X82 a8 atinb moinb dta mm2 vcc nand4
X83 n2 a4 mm3       vcc nand2
X84 a5 mm4          vcc inv
X85 mm1 mm2 mm3 mm4 mm5 vcc nand4
X86 reset clinb mm5 SHQ vcc and3
X87 h1 moinb o1    vcc nand2
X88 o1 moinb o2    vcc nand2
X89 atin o2  o3    vcc nand2
X90 a7 n2 atin moin o4    vcc nand4
X91 atinb moin dtab o5    vcc nand3
X92 o3 o4 o5 h2 o6        vcc and4
X93 reset clinb o6  SEDR1 vcc nand3
X94 h1 atin moin p1       vcc nand3
X95 p1 o4 o5 h2  p2       vcc nand4
X96 reset clinb  p2 SEDR2 vcc and3
X97  g2 q1           vcc inv
X98  b2 q2           vcc inv
X99  n1 q3           vcc inv
X100 q1 q2 q3 q7     vcc nand3
X101 h1 atin q4      vcc nand2
X102 afin s1b q5     vcc nand2
X103 q4 q5 q6        vcc nand2
X104 a5 q6 nin q8    vcc nand3
X105 q7 n6 h2 q8 q9  vcc and4
X106 reset clinb q9  WRITE  vcc nand3
X107 reset clinb atin a8 SELEND vcc and4
X108 s4 s3 s2b s1b LDY  vcc and4
X109 mk3    ck2 mk3ot    mk3otb    vcc ffd
X110 mk2    ck2 mk2ot    mk2otb    vcc ffd
X111 mk1    ck2 mk1ot    mk1otb    vcc ffd
```

```
X112 ativo  ck2 ativoot  ativootb  vcc ffd
X113 modo   ck2 modoot   modootb   vcc ffd
X114 af     ck2 afot     afotb     vcc ffd
X115 clear  ck2 clearot  clearotb  vcc ffd
X116 spzero ck2 spzeroot spzerootb vcc ffd
X117 zopa   ck2 zopaot   zopaotb   vcc ffd
X118 zopb   ck2 zopbot   zopbotb   vcc ffd
X119 sedr1  ck2 sedr1ot  sedr1otb  vcc ffd
X120 sedr2  ck2 sedr2ot  sedr2otb  vcc ffd
X121 selend ck2 selendot selendotb vcc ffd
X122 cbooth ck2 cboothot cboothotb vcc ffd
X123 shr    ck2 shrot    shrotb    vcc ffd
X124 sub    ck2 subot    subotb    vcc ffd
X125 ldy    ck2 ldyot    ldyotb    vcc ffd
X126 lda    ck2 ldaot    ldaotb    vcc ffd
X127 write  ck2 writeot  writeotb  vcc ffd
X128 shq    ck2 shqot    shqotb    vcc ffd
X129 reseten  ck1 reset resetb     vcc ffd
X130 dtaen    ck1 dta   dtab       vcc ffd
X131 ninen    ck1 nin   ninb       vcc ffd
X132 mk3ot    ck1 mk3in mk3inb     vcc ffd
X133 mk2ot    ck1 mk2in mk2inb     vcc ffd
X134 mk1ot    ck1 mk1in mk1inb     vcc ffd
X135 ativoot  ck1 atin  atinb      vcc ffd
X136 modoot   ck1 moin  moinb      vcc ffd
X137 afot     ck1 afin  afinb      vcc ffd
X138 clearot  ck1 clin  clinb      vcc ffd
X139 spzeroot ck1 spin  spinb      vcc ffd
X140 cpouten  ck1 cpout cpoutb     vcc ffd
X141 m2en     ck1 m2    m2b        vcc ffd
X142 m1en     ck1 m1    m1b        vcc ffd
X143 s4en     ck1 s4    s4b        vcc ffd
X144 s3en     ck1 s3    s3b        vcc ffd
X145 s2en     ck1 s2    s2b        vcc ffd
X146 s1en     ck1 s1    s1b        vcc ffd
```

```
******* sinais de interface **********
*interface: reseten    * orientacao=0 *
*interface: dtaen      * orientacao=0 *
*interface: ninen      * orinetacao=0 *
*interface: cpouten    * orientacao=0 *
*interface: m2en       * orientacao=0 *
*interface: m1en       * orientacao=0 *
*interface: s4en       * orientacao=0 *
*interface: s3en       * orientacao=0 *
*interface: s2en       * orientacao=0 *
*interface: s1en       * orientacao=0 *
*interface: zopaot     * orientacao=L *
*interface: zopbot     * orientacao=L *
*interface: sedr1ot    * orientacao=L *
*interface: sedr2ot    * orientacao=L *
*interface: selendot   * orientacao=L *
*interface: cboothot   * orientacao=L *
*interface: shrot      * orientacao=L *
*interface: subot      * orientacao=L *
*interface: ldyot      * orientacao=L *
*interface: ldaot      * orientacao=L *
*interface: writeot    * orientacao=L *
*interface: shqot      * orientacao=L *
*interface: ck1        * orientacao=S *
*interface: ck2        * orientacao=S *
.end
```

ANEXO 4 - Descrição SPICE da unidade de controle do multiplicador

```
* CELULA: INVERSOR
.SUBCKT INV IN OUT vcc
MP1  OUT  IN vcc vcc PMOS L=2.0U W=6.0U
MN2  OUT  IN 0   0   NMOS L=2.0U W=3.0U
.ENDS INV
*
* CELULA: AND 2 ENTRADAS
.SUBCKT AND2 I1 I2 OUT vcc
MP1    2 I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2    2 I2  vcc  vcc PMOS L=2.0U W=6.0U
MN3    2 I1    3  0   NMOS L=2.0U W=3.0U
MN4    3 I2    0  0   NMOS L=2.0U W=3.0U
MN5  OUT  2    0  0   NMOS L=2.0U W=3.0U
MP6  OUT  2  vcc  vcc PMOS L=2.0U W=6.0U
.ENDS AND2
*
* CELULA: OR 2 ENTRADAS
.SUBCKT OR2 I1 I2 OUT vcc
MP1  2 I1  vcc  vcc PMOS L=2.0U W=6.0U
MP2  S I2  2    vcc PMOS L=2.0U W=6.0U
MN3  S I1  0    0   NMOS L=2.0U W=3.0U
MN4  S I2  0    0   NMOS L=2.0U W=3.0U
MP5  OUT  S  vcc vcc PMOS L=2.0U W=6.0U
MN6  OUT  S  0   0   NMOS L=2.0U W=3.0U
.ENDS OR2
**
* CELULA: OR 3 ENTRADAS
.SUBCKT OR3 I1 I2 I3 I4 OUT vcc
MP1  2  I1 vcc vcc PMOS L=2.0U W=6.0U
MP2  3  I2 2   vcc PMOS L=2.0U W=6.0U
MP3  S  I3 3   vcc PMOS L=2.0U W=6.0U
MN4  S  I1 0   0   NMOS L=2.0U W=3.0U
MN5  S  I2 0   0   NMOS L=2.0U W=3.0U
```

```
MN6  S  I3 0   0   NMOS L=2.0U W=3.0U
MP7  OUT S vcc vcc PMOS L=2.0U W=6.0U
MN8  OUT S 0   0   NMOS L=2.0U W=3.0U
.ENDS OR3
*
* CELULA: FLIP FLOP D SENSIVEL A BORDA
.SUBCKT FFD D CK Q ~Q VCC
MN1  S1  S4  2     0  NMOS  L=2.0U W=3.0U
MN2   2  S2  GND   0  NMOS  L=2.0U W=3.0U
MN3  S2  S1  3     0  NMOS  L=2.0U W=3.0U
MN4   3  CK  GND   0  NMOS  L=2.0U W=3.0U
MN5  S3  S2  4     0  NMOS  L=2.0U W=3.0U
MN6   4  CK  5     0  NMOS  L=2.0U W=3.0U
MN7   5  S4  GND   0  NMOS  L=2.0U W=3.0U
MN8  S4  S3  6     0  NMOS  L=2.0U W=3.0U
MN9   6   D  GND   0  NMOS  L=2.0U W=3.0U
MN10  Q  S2  7     0  NMOS  L=2.0U W=3.0U
MN11  7  ~Q  GND   0  NMOS  L=2.0U W=3.0U
MN12 ~Q  S3  8     0  NMOS  L=2.0U W=3.0U
MN13  8   Q  GND   0  NMOS  L=2.0U W=3.0U
MP14 S1  S4  VCC  VCC  PMOS  L=2.0U W=6.0U
MP15 S1  S2  VCC  VCC  PMOS  L=2.0U W=6.0U
MP16 S2  S1  VCC  VCC  PMOS  L=2.0U W=6.0U
MP17 S2  CK  VCC  VCC  PMOS  L=2.0U W=6.0U
MP18 S3  S2  VCC  VCC  PMOS  L=2.0U W=6.0U
MP19 S3  CK  VCC  VCC  PMOS  L=2.0U W=6.0U
MP20 S3  S4  VCC  VCC  PMOS  L=2.0U W=6.0U
MP21 S4  S3  VCC  VCC  PMOS  L=2.0U W=6.0U
MP22 S4   D  VCC  VCC  PMOS  L=2.0U W=6.0U
MP23  Q  S2  VCC  VCC  PMOS  L=2.0U W=6.0U
MP24  Q  ~Q  VCC  VCC  PMOS  L=2.0U W=6.0U
MP25 ~Q  S3  VCC  VCC  PMOS  L=2.0U W=6.0U
MP26 ~Q   Q  VCC  VCC  PMOS  L=2.0U W=6.0U
.ENDS FFD
****** CHAMADA DOS SUBCIRCUITOS ********
X1 Q1B  Q0B  T0      vcc and2
```

```
X2 Q1B  Q0   T1       vcc and2
X3 Q1   Q0B  T2       vcc and2
X4 Q1   Q0   T3       vcc and2
X5 Z    ZB            vcc inv
X6 ZB   T3   N1       vcc and2
X7 N1   T1   T2  D1   vcc or3
X8 S    T0   N2       vcc and2
X9 T2   N2   D0       vcc or2
X10 LSB T2   LOAD     vcc and2
X11 D1  CK   Q1 Q1B   vcc ffd
X12 D0  CK   Q0 Q0B   vcc ffd
******* sinais de interface ********
*interface: S
*interface: Z
*interface: LSB
*interface: LOAD
*interface: T0
*interface: T1
*interface: T2
*interface: T3
.end
```

BIBLIOGRAFIA

[BRA 84] BRAYTON, R. K.; HACHTEL, G. D.; McMULLEN, C. T.; SANGIOVANNI-VINCENTELLI, A. L. Logic Minimization Algorithms for VLSI Synthesis. Boston: Kluwer, 1984. 193 p. (The Kluwer international series in engineering and computer science; VLSI, computer architecture, and digital signal processing).

[CHE 90] CHEN, J.Y. CMOS Devices and Technology for VLSI. Englewood Cliffs: Prentice Hall, 1990. 348 p.

[DAV 83] DAVIO, M, DESCHANPS, J.P. & THAYSE. Digital Systems with Algorithm Implementation. New York: John Wiley, 1983. 505 p.

[FLE 80] FLETCHER, W. Sequential Machine Fundamentals. In: An ENGINEERING Approach to Digital Design. Englewood Cliffs: Prentice-Hall, 1980. p.275-334.

[GEI 90] GEIGER, R. L. et al. VLSI Design Techniques for Analog and Digital Circuits. New York: McGraw-Hill, 1990. 969 p.

[GLA 85] GLASSER, L. A. & DOBBERPUHL, D. W. Circuit Techniques. In: The DESIGN and Analisys of VLSI Circuits. Reading: Addison-Wesley, 1985. p.253-330.

[HIL 73] HILL, F. & PETERSON, G. Digital Systems: Hardware Organization and Design. New York: John Wiley, 1973. 481 p.

[HIL 74] HILL, F. & PETERSON, G. Introduction to Switching Theory and Logical Design. New York: John Wiley, 1974. 596 p.

[HOR 83] HOROWITZ, M. A. Timing Models for MOS Circuits. Stanford: Stanford University, Integrated Circuits Laboratory, 1983. 120 p. (Internal Report n. SEL83-003)

[HWA 87] HWANG, D. K. & FUCHS, W.K. An Efficient Approach to Gate Matrix Layout. IEEE Transactions on CAD, New York, V.6, n.5, p.802-809, Sept. 1987.

[JOU 83] JOUPPI, N. P. Timing Analisys for nMOS VLSI. In: DESIGN AUTOMATION CONFERENCE, 20, June 27-29, 1983, Miami Beach. Proceedings... New York: ACM/IEEE, 1983. p.411-418.

[KAM 82] KAMBE, T. et al. A Placement Algorithm for Polycell LSI and its Evaluation. IN: DESIGN AUTOMATION CONFERENCE, 19, June 29-July 2, 1982, Las Vegas. Proceedings... New York: ACM/IEEE, 1982. p. 655-662.

[LOP 80] LOPEZ, A. & LAW, H.S. A Dense Gate Matrix Layout Method for MOS VLSI. IEEE Transactions on Electron Devices, New York, V.27, n.8, p. 1671-1675, Aug. 1980.

[MAN 79] MANO, M. M. Digital Logic and Computer Design. Englewood Cliffs: Prentice-Hall, 1979. 612 p.

[MAN 88] MANO, M.M. Computer Engineering: Hardware design. Englewood Cliffs: Prentice-Hall, 1988. 434 p.

[MEA 80] MEAD, C. & CONWAY, L. Introduction to VLSI Systems. Reading: Addison Wesley, 1980. 396 p.

[MOR 89] MORAES, F.G. & REIS, R. EXTRALO - Um Extrator Lógico. In: SEMINÁRIO INTERNO DE MICROELETRÔNICA,

5, 17-18 Novembro, 1989, Tramandaí. Anais... Porto Alegre: GME/UFGRS, 1989. p. 99-102.

[MOR 90] MORAES, F.G. & REIS, R. Ferramenta para Síntese Automática de Módulos em Lógica Aleatória. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 5., 11-13 Julho, 1990, Campinas. Anais... Campinas: SBMICRO, 1990. p. 12-21.

[OBR 82] OBRESKA, M. Etude Comparative de Differents Methodes de Conception des Parties de Controle des Microprocesseurs. Grenoble: Institut National Polytechnique de Grenoble, 1982. 582 p.

[OUS 85] OUSTERHOUT, J.K. A Switch-Level Timing Verifier for Digital MOS VLSI. IEEE Transactions on CAD, New York, V.4, n. 3, p. 336-349, July 1985.

[PUT 82] PUTATUNDA, R. AUTO-DELAY: A Program for Automatic Calculation of Delay in LSI/VLSI Circuits. In: DESIGN AUTOMATION CONFERENCE. 19., June 29 - July 1982, Las Vegas. Proceedings... New York: ACM/IEEE 1982. p.616-620.

[ROB 90] ROBERT, M. et. al. PATH RUNNER: An Accurate and Fast Timing Analyser. In: THE EUROPEAN DESIGN AUTOMATION CONFERENCE, March 12-15, 1990, Glasgow, Scotland. Proceedings... Washington: IEEE, 1990. p. 529-533.

[RUB 83] RUBINSTEIN, J.; PENFIELD, P.; HOROWITZ, M. Signal Delay in RC Tree Networks. IEEE Transactions on CAD, New York, V.2, n.3, p. 202-211, July 1983.

[SAK 83] SAKURAY, T. Approximation of Wiring Delay in MOSFET LSI. IEEE Journal of Solid State Circuits, New

York, V.18, n.4, p. 418-426, Aug. 1983.

[SAL 90] SALEH, R. A. & NEWTON, A. R. Mixed Mode Simulation. Boston: Kluwer Academic, 1990. 214 p.

[SAN 90] SANTOS, L.C.V. dos. Um Sistèma Digital para Implementação de um Algoritmo de Equalização Adaptativa. Porto Alegre: CPGCC/UFRGS, 1990. 248 p. (Dissertação de Mestrado)

[SCO 85] SCOTT, N. R. Computer Number Systems and Arithmetic. Englewood Cliffs: Prentice-Hall, 1985. 254 p.

[SIS 82] SISKIND, J. M. et al. Generation Custom High Performance VLSI Designs from Succinct Algorithmic Descriptions. In: CONFERENCE ON ADVANCED RESEARCH IN VLSI, Jan. 25-27, 1982, Cambridge. 1982. Proceedings...Dedham, MA: Artech House, 1982. p. 28-39.

[SOL 88] SOLO 2000 - Design Reference Manual, June, 1988.

[SPI 86] SPICE Version 2G User's Guide, 1986.

[STE 89] STEMMER, M. & REIS, R. EXTRIBO: Um Extrator Hierárquico de Circuitos. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 4., Abril 12-14, 1989, Rio de Janeiro. Anais... Rio de Janeiro: SBC, 1989. p. 1-9.

[STE 90] STEVE, C. Introduction to HDL-Based Design Using VHDL. Mountain View:Synopsys, 1990. 176 p.

[SUM 87] SUMMER SCHOOL ON VLSI CAD TOOLS AND APPLICATIONS, July 21 - Aug. 1, 1986, Beatemberg, SU. VLSI CAD Tools and Applications. Boston: Kluwer academic,

1987. 552 p.

[SUZ 88] SUZIM, A. A. A CAD Frame for VLSI Design. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 3., 13-15 Abril, 1988, Gramado. Anais... Porto Alegre: SBCCI, 1988. p. 129-145.

[SUZ 89] SUZIM, A.A. & SANTOS, L.C.V. dos. HDC - uma Técnica de Descrição de Hardware para Simulação Funcional. In: SEMINÁRIO INTERNO DE MICROELETRÔNICA, 5., 17-18 Novembro, 1989, Tramandaí. Anais... Porto Alegre: CPGCC/UFRGS, 1989. p.81-84.

[SZE 83] SZE, S. M., Ed., VLSI Technology. Ney York: McGraw-Hill, 1983. 654 p.

[TEX 88] TEXAS INSTRUMENTS. First Generation' TMS320 User's Guide. 1988.

[WAG 87] WAGNER, F. R. & DAL SASSO-FREITAS, C. M. NILO - Uma Linguagem para a Descrição de Hardware no Nível de Portas Lógicas. Porto Alegre: CPGCC/UFRGS, 1987. 18 p. (RP 66)

[WES 85] WESTE, N. H. & ESHRAGUIAN, K. Circuit Characterization and Performance Estimation. In: PRINCIPLES of CMOS VLSI Design. Reading: Addison-Wesley, 1985. p. 119-158.

Informática
UFRGS

"Avaliação de desempenho de partes de controle de circuitos integrados".

Dissertação apresentada aos Srs.:

Prof. Dr. Altamiro Amadeu Suzim

Prof. Francisco Amaral Wendt (UFSM)

Prof. Dr. Sergio Bampi

Prof. Tiaraju Vasconcellos Wagner

Vista e permitida a impressão.
Porto Alegre, 20/05/92.

Prof. Dr. Altamiro Amadeu Suzim
Orientador.

Prof. Dr. Ricardo A. da L. Reis,
Coordenador do Curso de Pós-Graduação
em Ciência da Computação.