

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ESTRUTURAS SISTÓLICAS PARA A REALIZAÇÃO DA
OPERAÇÃO PRODUTO INTERNO COM ALTA TAXA DE
ENTRADA E SAÍDA DE DADOS

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA
PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

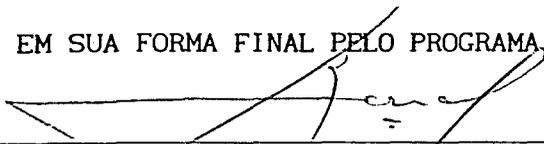
JOCELI MAYER 

FLORIANÓPOLIS, OUTUBRO DE 1991

ESTRUTURAS SISTÓLICAS PARA A REALIZAÇÃO DA
 OPERAÇÃO PRODUTO INTERNO COM ALTA TAXA DE
 ENTRADA E SAÍDA DE DADOS

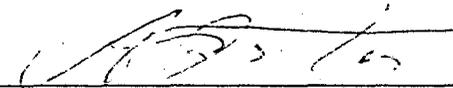
JOCELI MAYER

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO
 DE MESTRE EM ENGENHARIA, ESPECIALIDADE ENGENHARIA ELÉTRICA, E
 APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO


 Prof. José Carlos Moreira Bermudez, Ph. D. ¹/₂
 Orientador

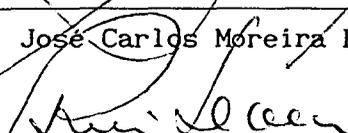

 Prof. Rui Seara, Dr. Ing.

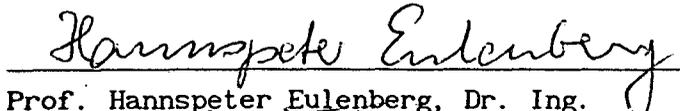
Co-orientador


 Prof. João Pedro Assumpção Bastos, Dr. D'Etat
 Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

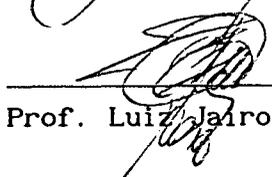
BANCA EXAMINADORA


 Prof. José Carlos Moreira Bermudez, Ph. D.


 Prof. Rui Seara, Dr. Ing.


 Prof. Hannspeter Eulenberg, Dr. Ing.


 Prof. Joni da Silva Fraga, Dr.


 Prof. Luiz Jairo Branco Machado, Dr. Ing.

À minha namorada, Marlise

AGRADECIMENTOS

Ao Prof. José Carlos Moreira Bermudez, pela orientação.

Ao Prof. Rui Seara, pela co-orientação.

Ao Prof. Hannspeter Eulenberg, pela orientação no ZEL/KFA (Alemanha).

Ao CNPQ e à CAPES pelo apoio financeiro.

À FINEP pelo suporte financeiro e material.

Aos colegas do Laboratório de Instrumentação Eletrônica (LINSE).

Aos colegas do Zentrallabor Für Elektronik (ZEL).

Ao LINSE (Laboratório de Instrumentação Eletrônica) do Departamento de Engenharia Elétrica da UFSC, onde este trabalho foi desenvolvido.

Aqueles que, diretamente ou indiretamente, colaboraram para a realização do presente trabalho.

RESUMO

A operação produto interno (somatório de produtos) aparece freqüentemente em diversos algoritmos de processamento digital de sinais, tais como filtragem, multiplicação de matrizes e vetores, correlação, entre outros. A eficácia na realização desta operação influencia sobremaneira o desempenho de velocidade dos algoritmos citados e, para tanto, diversas arquiteturas paralelas têm sido propostas na literatura visando a aumentar a eficácia desta operação. Uma alternativa que tem se mostrado muito interessante para aplicações que requeiram alto desempenho, tais como processamento em tempo real e processamento massivo de dados, é a utilização de estruturas sistólicas. Estas estruturas são especialmente adequadas à implementação em VLSI, devido às suas características de modularidade, regularidade, conexões locais e vários níveis de paralelismo e "pipelining".

Neste trabalho serão propostas estruturas sistólicas constituídas de um arranjo de células processadoras operando com palavras de n/k bits para a realização da operação produto interno apresentando alto desempenho de velocidade. Os resultados das simulações, comparações e implementações, permitem afirmar que as estruturas propostas constituem uma alternativa eficaz para a realização de diversas aplicações que utilizam-se da operação produto interno. Além disso, estas estruturas têm desempenho de velocidade e área requeridas na implementação proporcional ao fator k (nível de partição das palavras de entrada) utilizado, fornecendo ao projetista uma flexibilidade na escolha da estrutura que melhor satisfaça os requisitos da aplicação. Apesar da área requerida aumentar proporcionalmente ao fator k , a velocidade obtida nestas estruturas propostas compensa favoravelmente o acréscimo de área necessário para a implementação em VLSI.

A B S T R A C T

The inner-product operation is frequently used in many Digital Signal Processing, for example filtering, matrix multiplication and vector multiplication, correlation and others. The velocity performance of these algorithms depends strongly on the performance of this operation. Therefore, many architectures have been proposed in the literature to enhance the performance of the inner-product operation. One of these architectures, which has been extensively used in high speed applications, is the systolic structures. These structures are very appealing to VLSI implementation of real-time and massive processing systems, specially because of its characteristics of modularity, regularity, local connections and high level of parallelism and pipelining.

This work will propose new systolic structures based on cells processing words of n/k -bits width to realize the inner-product operation efficiently, namely, with high performance and low design costs. The results of the simulations, comparisons and implementations will assure that the proposed structures are an excellent alternative to realize many algorithms based on the inner-product operation. Moreover, these structures have performances of velocity and required area dependable on k value, namely, partition level of the input words. This characteristic gives a flexibility to system designer allowing to choice one of the structures proposed which satisfy the trade off between area and performance of the involved application.

S U M Á R I O

CAPÍTULO 1 - INTRODUÇÃO.....	1
 CAPÍTULO 2 - DESENVOLVIMENTO DE ESTRUTURAS SISTÓLICAS	
2.1 - Introdução.....	11
2.2 - Operação Produto Interno.....	11
2.3 - Algoritmo de Multiplicação.....	15
2.4 - Estruturas Sistólicas.....	18
2.5 - Redução do Número de Conexões.....	25
2.6 - Estruturas Operando com Números Bipolares.....	32
2.7 - Otimização do Algoritmo para Números Bipolares.....	34
2.8 - Conclusões.....	39
 CAPÍTULO 3 - AVALIAÇÃO DE DESEMPENHO DAS ESTRUTURAS	
3.1 - Introdução.....	40
3.2 - Determinação da Área e do Tempo de Propagação de Dados.....	40
3.3 - Parâmetros das Células das Estruturas Otimizadas.....	43
3.4 - Parâmetros das Células das Estruturas não Otimizadas.....	46

3.5 - Taxa de Entrada e Saída de Dados e Área Total das Estruturas....	47
3.6 - Comparações e Valores Numéricos de Desempenho.....	49
3.7 - Conclusões.....	55
 CAPÍTULO 4 - VALIDAÇÃO DAS ESTRUTURAS	
4.1 - Introdução.....	56
4.2 - Simulações via Computador Digital.....	56
4.3 - Implementação via Arranjo Sistólico de Processadores.....	62
4.4 - Implementação via Multiplicadores e Somadores Comerciais.....	63
4.5 - Implementação via Arranjo Programável de Portas Lógicas (PGA)...	65
4.6 - Conclusões.....	70
 CAPÍTULO 5 - CONCLUSÕES.....	
	72
 APÊNDICE A - PROGRAMA DE SIMULAÇÃO DA ESTRUTURA.....	
	74
 APÊNDICE B - DIAGRAMAS ESQUEMÁTICOS DAS CÉLULAS IMPLEMENTADAS EM PGA..	
	83
 APÊNDICE C - DIAGRAMA ESQUEMÁTICO DA PLACA DE TESTES.....	
	94
 REFERÊNCIAS.....	
	96

C A P Í T U L O 1

I N T R O D U Ç Ã O

O Processamento Digital de Sinais teve origem no século XVII e desde então tem servido a inúmeras aplicações nos campos da ciência e tecnologia, tais como Sismologia, Radar, Sonar, Comunicação de Dados e Voz, Processamento de Imagens e Acústica, entre outras.

No Processamento Digital de Sinais (PDS), os sinais são representados por seqüências de números ou símbolos e processados de maneira a transformar estas seqüências em outras ou estimar os parâmetros que as caracterizam. Exemplos de transformação e estimação de parâmetros de seqüências são a Transformação de Fourier e as determinações da média e da variância de uma determinada seqüência.

Antes do aparecimento das técnicas de integração em larga escala que permitiram a realização dos primeiros circuitos integrados com alta densidade e alta velocidade, os sistemas que requeriam alta taxa de processamento somente eram realizáveis através de implementação analógica, devido, sobretudo às limitações de velocidade dos sistemas digitais. Atualmente, com a evolução das tecnologias de integração, inúmeras aplicações de alto desempenho estão sendo implementadas por sistemas digitais que apresentam, muitas vezes, vantagens sobre as implementações analógicas, como programabilidade e repetibilidade, insensibilidade a variações de temperatura, à troca de componentes e a variações na tensão de alimentação, entre outras.

São inúmeras as possibilidades de realização de processamento por sistemas digitais. Dentre os algoritmos mais utilizados em implementações, destacam-se os de filtragem, convolução, correlação, integração, diferenciação, interpolação, dizimação e multiplicação de matrizes e vetores. A crescente necessidade de sistemas mais velozes e com melhor desempenho, originada pelo aumento, cada vez maior, do número de informações a ser processado e da dinâmica dos acontecimentos na sociedade moderna, incentivou sobremaneira modificações nos algoritmos de PDS visando-se a maior eficácia. Um exemplo típico deste aprimoramento foi o surgimento da Transformação de Fourier Rápida (FFT), reduzindo sensivelmente o número de multiplicações e acumulações necessário no cálculo da Transformação de Fourier Discreta (DFT) e, permitindo assim, implementações mais eficazes de uma grande quantidade de aplicações de processamento baseadas nesta transformação [1].

Esta necessidade de maior velocidade surge, principalmente, de aplicações que requerem processamento em tempo real ou processamento massivo de dados. Nas aplicações de processamento em tempo real, as amostras do sinal de saída do sistema são processadas na mesma taxa em que chegam os dados de entrada. Portanto, os sistemas de processamento em tempo real requerem que os circuitos de processamento operem na taxa de entrada de dados, que é, em muitas aplicações, bastante elevada para a tecnologia disponível.

No caso de aplicações de processamento massivo, a grande quantidade de informações a serem processadas pode resultar em um tempo excessivo de processamento. Portanto, tais aplicações são grandemente beneficiadas pela utilização de circuitos de processamento de alta velocidade.

Com a evolução da Tecnologia de Integração Digital, a densidade e a

velocidade atingíveis em um circuito integrado (CI) têm aumentado significativamente, resultando em redução de área e no aumento de desempenho por CI. Além disso, os custos por unidade foram reduzidos, permitindo implementações econômicas de sistemas de PDS de alto desempenho.

Embora o estado da arte de integração de circuitos tenha evoluído constantemente, os custos decorrentes dos avanços da tecnologia crescem exponencialmente para cada aumento linear da densidade de circuitos nos CIs. Além disso, estimativas recentes mostram que a integração de circuitos com dimensões internas menores do que $0,5 \mu\text{m}$ demandaria um esforço tecnológico extremo [2]. Como a velocidade de operação dos circuitos digitais convencionais depende diretamente da redução das dimensões mínimas da tecnologia utilizada, observa-se que o aumento de velocidade nos CIs apresenta uma curva assintótica tendendo a uma velocidade limite, devido aos limites tecnológicos, custos e mesmo limites teóricos no aumento de densidade. Uma opção para aumentar-se o desempenho dos sistemas PDS consiste em distribuir o processamento a diversas estruturas operando paralelamente. A partir desta filosofia surgiram as arquiteturas paralelas, que de certa forma viabilizaram a utilização de aplicações de PDS onde até então não era possível seu emprego devido às limitações de velocidade.

Com a disponibilidade de CIs a baixos custos e com o acesso aos modernos sistemas de projeto auxiliado por computador (CAD) para VLSI, surgem estruturas de processamento distribuído, as quais possibilitam um aumento na complexidade e na velocidade de processamento devido ao paralelismo utilizado sem, entretanto, incrementar os custos e os esforços tecnológicos inerentes a um crescimento da densidade de circuitos por CI. Existem vários tipos de

arquiteturas paralelas, os quais podem ser agrupados segundo suas características comuns. Michael J. Flynn [3] apresenta uma classificação conforme o número de instruções e dados que a estrutura processa simultaneamente. Nesta classificação tem-se 4 tipos básicos de arquitetura:

i) Arquitetura SISD ("Single Instruction Single Data"), processa um dado utilizando uma instrução e, portanto, destina-se a processamentos sequenciais. Um exemplo é a clássica arquitetura de Von Neumann.

ii) Arquitetura SIMD ("Single Instruction Multiple Data"), processa vários dados utilizando uma instrução, ou seja, o processamento total é realizado através da utilização de várias unidades processadoras conectadas entre si, cada qual executando um mesmo algoritmo. É nesta classe de arquiteturas, a qual é apropriada para processamento especializado de alto desempenho, que estão incluídos os chamados arranjos sistólicos.

iii) Arquitetura MIMD ("Multiple Instruction Multiple Data"), processa vários dados utilizando múltiplas instruções. Largamente utilizada em sistemas de multiprocessamento, normalmente utiliza-se de vários processadores interagindo entre si gerenciados por um controlador geral ou por um sistema de "handshaking". Esta arquitetura é especialmente apropriada para processamento de propósito geral.

iv) Arquitetura MISD ("Multiple Instruction Single Data"), processa um dado utilizando múltiplas instruções. Esta arquitetura não apresenta aplicações práticas até o momento.

Esta classificação, apesar de bastante difundida, tem-se mostrado incapaz

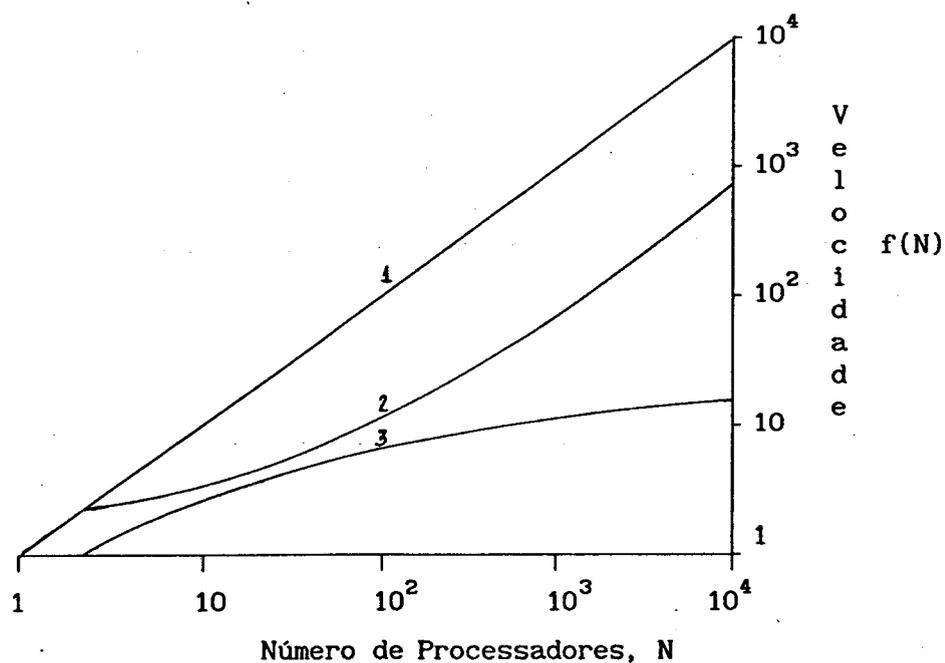
de diferenciar algumas estruturas recentemente desenvolvidas e que apresentam desempenhos bastante distintos. Assim, diferentes classificações têm sido propostas na literatura [3], as quais baseiam-se em outros critérios de diferenciação. Neste trabalho será adotada uma classificação que divide as arquiteturas paralelas em dois grandes grupos, arquiteturas de propósito geral e arquiteturas de propósito específico.

As arquiteturas de propósito geral destinam-se a satisfazer um grupo numeroso de aplicações e devem alocar seus recursos de processamento conforme o algoritmo a ser realizado. Este tipo de arquitetura requer uma estrutura sofisticada de controle e gerenciamento das unidades constituintes, de forma a obter-se um desempenho adequado para diversas aplicações. Conseqüentemente, devido à sua generalidade, tais arquiteturas permitem apenas se alcançar um desempenho sub-ótimo para aplicações específicas [4].

As arquiteturas de propósito específico são estruturas dedicadas, de forma a otimizar determinado parâmetro de interesse como, por exemplo, a velocidade, em uma dada aplicação. No desenvolvimento destas arquiteturas deve-se ter um pleno conhecimento do algoritmo de processamento paralelo a ser utilizado, assim como, das características específicas da implementação pretendida, de forma a obter-se o máximo desempenho em termos do(s) parâmetro(s) de interesse [11].

Um parâmetro consistente de medida da eficácia de arquiteturas paralelas é dado pela relação entre a velocidade de processamento e o número de processadores utilizados na arquitetura [2]. No caso de multiprocessamento, onde N processadores são conectados a um único barramento, à medida que N cresce, ocorre o "engarramento" do barramento, limitando assim a velocidade

do sistema. Esta limitação é prevista pela Conjetura de Minski, onde a curva de velocidade em função de N é do tipo $\log_2 N$. Os projetistas dos recentes supercomputadores têm, através de aperfeiçoamentos estruturais, melhorado esta relação para $N/\log_2 N$, que é conhecida como Lei de Amdahl. No caso das arquiteturas sistólicas propostas por H.T.Kung [19] esta relação chega a N , o que representa uma utilização ótima dos N processadores em termos de velocidade de processamento. As curvas de velocidade em função do número N de processadores utilizados, para as arquiteturas de processamento paralelo citadas acima, estão representadas na Fig. 1.1.



Curva 1 - Arranjos sistólicos (100 % de eficiência) onde $v = f(N) = N$.

Curva 2 - Supercomputadores (Lei de Amdahl) onde $v = f(N) = N/\log N$.

Curva 3 - Multiprocessamento (Conjetura de Minsky) onde $v = f(N) = \log_2 N$.

Fig. 1.1 - Eficácia das Arquiteturas Paralelas

Além disso, as estruturas sistólicas e outras estruturas como, por exemplo, as estruturas "wave front array" [4], são baseadas em unidades conectadas localmente, ou seja, conectadas apenas com as unidades vizinhas. Esta característica é especialmente atrativa para processamento em alta velocidade porque a utilização de conexões globais resulta em acréscimos de área e em atrasos na propagação do sinal, levando a uma redução de velocidade no processamento e podendo até mesmo inviabilizar o "layout" da estrutura em virtude da maior complexidade de conexões.

Devido às características mencionadas, estas arquiteturas têm despertado um grande interesse por parte dos projetistas de sistemas de processamento digital de sinais em aplicações que necessitam alto desempenho e em aplicações computacionalmente intensivas, tais como, sistemas de processamento massivo e de processamento em tempo real.

As estruturas sistólicas são bastante semelhantes aos "wave front arrays" nos aspectos de conexões locais, regularidade e modularidade da estrutura, propriedades que facilitam o desenvolvimento do "layout" e reduzem a complexidade da estrutura. Outra característica comum é o alto grau de paralelismo e "pipelining" realizável por estas estruturas, o que resulta em altas taxas de entrada e de saída de dados, refletindo diretamente na velocidade de processamento. A diferença básica consiste na sincronização das unidades. No caso das estruturas sistólicas todas as unidades são sincronizadas pelo mesmo sinal de clock e os dados são processados ritmicamente, difundindo-se de forma que um fluxo regular é mantido por toda a

estrutura. O termo "sistólico" provém da fisiologia, representando a contração rítmica do coração, o qual pulsa sangue por todo o corpo. Observando-se uma estrutura sistólica nota-se esta similaridade entre o fluxo rítmico dos dados e o fluxo rítmico de sangue no organismo. Esta característica implica na necessidade de conexões globais no "layout" para realizar a rede de clock. Devido à existência da sincronização global nas estruturas sistólicas, estas apresentam um fenômeno denominado escorregamento de clock ("clock skew"). Este fenômeno é decorrente dos atrasos de propagação do sinal de clock devido às resistências e capacitâncias inerentes às conexões da rede de clock, e das residuais diferenças de tempo de processamento entre as unidades sincronizadas da estrutura. Estes fatores implicam em redução da taxa de clock e conseqüentemente em redução na taxa de entrada e saída de dados, de forma que seja garantido o sincronismo de todas as unidades desta estrutura.

Nos "wave front arrays", a sincronização é local e o fluxo de dados é coordenado por circuitos de "handshaking" existentes em cada unidade da estrutura, os quais aumentam a área da estrutura e tendem a reduzir a velocidade de processamento. Em estruturas que se utilizam de unidades processadoras básicas muito mais complexas do que o circuito de "handshaking", a influência destes circuitos na área total e velocidade da estrutura pode ser desprezível. Se, por outro lado, unidades processadoras de baixa complexidade forem utilizadas, os circuitos de "handshaking" aumentam sua relevância na área total e no tempo de propagação em cada unidade, de forma a reduzir significativamente a eficiência da estrutura. Além disso, sistemas síncronos que apresentem características dependentes da frequência de clock como, por exemplo, filtros digitais, não são adequados à implementação por sistemas

assíncronos do tipo "wave front array".

Neste trabalho são desenvolvidas novas estruturas para a realização da operação produto interno (produto escalar), a qual é a operação básica necessária à implementação de muitos importantes algoritmos em PDS. Neste desenvolvimento são utilizadas estruturas sistólicas, as quais apresentam, conforme discutido anteriormente, características muito atraentes à implementação em VLSI. Para o desenvolvimento das estruturas em questão utilizam-se arranjos sistólicos de células operando com palavras de n/k bits ("word level systolic array"), alternativa até o momento pouco explorada na realização da operação produto interno. Embora inúmeras estruturas tenham sido propostas na literatura para este fim, dentre elas, a nosso ver, não figuram estruturas sistólicas com características de flexibilidade e desempenho semelhantes às conseguidas pelas estruturas propostas neste trabalho. A comprovação dos resultados é obtida através da análise de desempenho das novas estruturas e da comparação das mesmas com diversas outras estruturas apresentadas na literatura. Todas as estruturas foram simuladas em computador digital e as células que compõem as arquiteturas propostas foram implementadas utilizando-se circuitos integrados disponíveis comercialmente, de forma a validar os resultados de simulação.

No Capítulo 2, são apresentadas as formas de implementação da operação produto interno mais difundidas na literatura, assim como as vantagens e desvantagens destas em relação à implementação utilizando-se estruturas sistólicas realizadas por células operando com palavras de n/k bits. Posteriormente, é apresentado o estudo de um algoritmo regular para a operação de multiplicação de duas palavras de n bits. Para a implementação deste

algoritmo são então propostas novas estruturas sistólicas que, além de realizar a operação de multiplicação realizam a operação produto interno. As características e limitações desta família de estruturas são discutidas, conduzindo ao novo algoritmo otimizado e à respectiva realização por estruturas sistólicas.

No Capítulo 3, são apresentadas análises de desempenho em termos de área de integração e de velocidade das novas estruturas desenvolvidas. Comparações com estruturas existentes na literatura, confirmam a eficácia e a flexibilidade das estruturas propostas.

No Capítulo 4, são apresentados os resultados da implementação de uma estrutura representativa da nova classe proposta. As células constituintes desta estrutura foram implementadas utilizando-se um arranjo sistólico de DSPs ("Digital Signal Processors"). Outras implementações foram realizadas através de CIs comerciais e PGAs ("Programmable Gate Array"). Os resultados obtidos destas implementações são descritos neste capítulo e permitem validar as estruturas desenvolvidas.

No Capítulo de conclusões, são traçadas considerações a respeito das características das estruturas propostas e de seu desempenho, além de avaliações de ordem geral do trabalho desenvolvido.

Complementando o trabalho, os apêndices A, B e C apresentam respectivamente a listagem do programa utilizado para a simulação da estrutura do Capítulo 4, os diagramas esquemáticos da implementação das células em circuitos PGAs e o diagrama esquemático da placa de testes utilizada para verificar e avaliar o funcionamento da estrutura implementada em PGAs.

C A P Í T U L O 2

ESTRUTURAS SISTÓLICAS PARA REALIZAR A OPERAÇÃO PRODUTO INTERNO

2.1 - Introdução

Neste capítulo será apresentada a operação produto interno e suas características. Serão discutidas as vantagens e desvantagens de relevantes trabalhos propostos na literatura, os quais visam a implementar esta operação.

O algoritmo da operação de multiplicação será apresentado com um enfoque direcionado ao desenvolvimento das estruturas sistólicas que serão propostas neste trabalho. Posteriormente, serão apresentadas as estruturas sistólicas para a realização da operação produto interno e o respectivo fluxo de processamento de dados destas estruturas.

Na seqüência, será desenvolvida uma nova família de estruturas que apresentam redução do número de conexões e reduzido número de células. Neste mesmo capítulo serão apresentadas estruturas que realizam a operação produto interno com dados numéricos bipolares e, posteriormente, será apresentada uma simplificação no algoritmo de multiplicação que permite uma redução considerável na quantidade de células da estrutura.

2.2 - Operação Produto Interno

Uma grande parte dos algoritmos de Processamento Digital de Sinais (PDS) utiliza-se de operações do tipo somatório de produtos de palavras

representadas em n bits. Este tipo de operação, denominada de operação produto interno (PI) ou produto escalar, é encontrada em diversas aplicações envolvendo, por exemplo, correlação, convolução, operações matriciais, filtragens FIR e IIR [2].

Uma grande percentagem do tempo de processamento destes algoritmos é despendida na execução da operação produto interno. Este fato tem motivado inúmeros pesquisadores a buscar estruturas mais eficazes para a sua realização, de maneira a aumentar o desempenho dos algoritmos [5].

A estrutura básica de implementação do produto interno, constitui-se de um multiplicador e de um somador realimentado, configurando a estrutura conhecida por MAC (Multiplicador Acumulador), representada na Fig. 2.1.

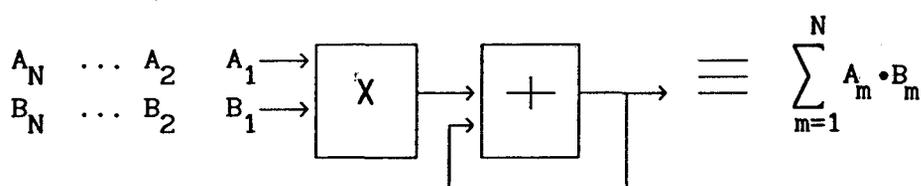


Fig. 2.1 - Multiplicador acumulador básico.

Esta estrutura, com as unidades multiplicadora e acumuladora operando em "pipelining", tem sido utilizada na arquitetura de vários DSPs (Processador Digital de Sinais) de propósito geral [6], realizando inúmeros algoritmos com bom desempenho. Contudo, a taxa de entrada e saída de dados desta estrutura é limitada pelo tempo de propagação da unidade mais lenta da estrutura, ou seja, o multiplicador de $(n \cdot n)$ bits [7].

Várias estruturas têm sido propostas na literatura visando à redução do tempo de propagação na operação de multiplicação e, conseqüentemente, aumentando a velocidade de processamento das estruturas MAC que utilizam

"pipelining". Estas estruturas podem ser classificadas em dois grandes grupos: estruturas sistólicas e não sistólicas. A grande maioria das estruturas não sistólicas que apresenta alto desempenho, baseia-se na técnica de particionar as palavras de entrada A e B, em 2 conjuntos de k sub-palavras de n/k bits cada uma, gerar os k^2 produtos parciais entre as sub-palavras e, posteriormente, somá-los de forma a obter o resultado da operação de multiplicação $A \cdot B$ [16]. Em [8], os produtos parciais de sub-palavras de um bit ($n/k = 1$) são gerados simultaneamente por k^2 multiplicadores de um bit (porta "E") e posteriormente adicionados por um arranjo ("árvore") de somadores completos ("full adders"). Em [9], produtos parciais de sub-palavras de n/k bits são gerados simultaneamente por um conjunto de k^2 multiplicadores de n/k bits e, posteriormente, adicionados por uma estrutura eficiente de soma. Nestes trabalhos são apresentadas alternativas atrativas para a realização da operação de multiplicação, apesar de não usufruírem das vantagens das estruturas sistólicas.

Estruturas sistólicas constituídas de um arranjo de células operando a nível de bit têm sido propostas na literatura, visando a aumentar ainda mais a eficiência na realização da operação de multiplicação. Estas estruturas apresentam uma alta taxa de entrada e de saída de dados, $Taxa_{E/S}$, proporcionada pelos vários níveis de paralelismo e de "pipelining" implementados [10]. Estas estruturas têm sido intensivamente utilizadas em diversas aplicações tais como em filtros FIR e convolvedores, apresentando bom desempenho de velocidade [10, 15]. Apesar da alta $Taxa_{E/S}$ conseguida nestas estruturas, vários níveis de "pipelining" normalmente resultam em aumento do escorregamento de clock ("clock skew"), da área e do tempo de latência [10], parâmetros que afetam diretamente a eficácia da estrutura. Portanto, existe um compromisso, entre estes parâmetros e os níveis de "pipelining" implementado,

que deve ser cuidadosamente levado em consideração no desenvolvimento de novas estruturas [12]. Outro problema inerente às estruturas com excessivos níveis de "pipelining", é o pico de dissipação de potência na transição do clock. Nas estruturas sistólicas todas as células são sincronizadas pelo mesmo clock, e, como a granularidade (nível de complexidade) das células em sistemas com muitos níveis de "pipelining" é muito alta, isto significa que a grande maioria das portas da estrutura comutam simultaneamente na frequência do clock, levando a uma dissipação indesejável de potência [13].

As estruturas sistólicas e não sistólicas citadas acima destinam-se à realização da operação de multiplicação. Para a sua utilização em operações produto interno, é necessário incluir um acumulador junto ao multiplicador. No entanto, à medida que a taxa de entrada e de saída do multiplicador aumenta devido às otimizações no projeto, o tempo de propagação da estrutura acumuladora torna-se o fator limitante de velocidade na estrutura. Portanto, para aumentar o desempenho da operação PI é necessário o desenvolvimento de estruturas que realizem tanto a operação de multiplicação quanto a de acumulação com alta velocidade.

Uma alternativa importante e pouco explorada, para realização da operação produto interno, é a utilização de estruturas sistólicas constituídas por células operando a nível de palavra. Alguns trabalhos apresentados na literatura [7] utilizam-se de N células processadoras, onde cada célula contém um multiplicador de $(n \cdot n)$ bits e uma unidade somadora, para a realização do produto interno de N termos. Em cada célula é armazenado um elemento de um dos vetores a serem multiplicados. Os elementos do segundo vetor são então difundidos pelas N células. Na saída da N -ésima célula tem-se o resultado da operação. Estas estruturas apresentam uma Taxa_{E/S} proporcional à Taxa_{E/S} da

célula processadora, independentemente do número de termos N a serem processados. Apesar da alta Taxa E/S obtida, estas estruturas requerem N células processadoras, ou seja, N multiplicadores de $(n \cdot n)$ bits e N unidades somadoras, levando a uma excessiva complexidade para valores grandes de N . Por exemplo, em aplicações de filtragem digital FIR, um número N de coeficientes maior do que 100 é normalmente requerido para uma filtragem com alta seletividade, resultando em uma excessiva área e, até mesmo, podendo inviabilizar a implementação através deste tipo de estrutura.

Neste trabalho são propostas estruturas sistólicas utilizando células que operam com palavras de n/k bits. As novas estruturas apresentam uma grande flexibilidade no compromisso entre área e velocidade de processamento. Além disso, estas estruturas são especialmente atrativas para implementações em VLSI devido às suas características de modularidade, regularidade e conexões locais entre células.

Inicialmente, o algoritmo de multiplicação será descrito em um enfoque adequado ao desenvolvimento das estruturas sistólicas a serem propostas neste trabalho.

2.3 - Algoritmo de Multiplicação

Considere duas palavras representadas em n bits, denominadas A e B , as quais são particionadas em k sub-palavras de n/k bits, $W_i(A)$ e $W_i(B)$ para $i = 1, \dots, k$ e para um valor n/k inteiro:

$$A = W_1(A) W_2(A) \dots W_k(A). \quad (2.1)$$

$$B = W_1(B) W_2(B) \dots W_k(B). \quad (2.2)$$

O produto S destas duas palavras pode ser representado em função das k sub-palavras de A e B :

$$S = A \cdot B = [W_1(A) \ W_2(A) \ \dots \ W_k(A)] \cdot [W_1(B) \ W_2(B) \ \dots \ W_k(B)] \quad (2.3)$$

Considerando o produto parcial P_{ij} , entre duas sub-palavras de A e B , definido como:

$$P_{ij} = \begin{cases} W_i(A) \cdot W_j(B) , & \text{para } 0 < i, j \leq k. \\ 0 , & \text{para outros valores de } i, j. \end{cases}$$

Como as sub-palavras $W_i(A)$ e $W_j(B)$ estão representadas em n/k bits, o produto parcial P_{ij} deve ser representado em $2n/k$ bits. Para facilitar as explicações, a seguir, são introduzidas aqui as seguintes notações:

L_{ij} representa os n/k bits menos significativos de P_{ij} .

H_{ij} representa os n/k bits mais significativos de P_{ij} .

$$S = A \cdot B = W_1(S) \ W_2(S) \ \dots \ W_{2k}(S). \quad (2.4)$$

O algoritmo de multiplicação das palavras A e B pode ser representado por somas das sub-palavras L_{ij} e H_{ij} deslocadas entre si por passos de n/k bits.

Para exemplificar, considere os algoritmos de multiplicação para $k = 2$ e $k = 4$, dados como:

Para $k = 2$:

$$S = 2^0 \cdot L_{22} + 2^{n/k} \cdot (H_{22} + L_{12} + L_{21}) + 2^{2n/k} \cdot (H_{12} + H_{21} + L_{11}) + 2^{3n/k} \cdot H_{11} \quad (2.5)$$

Para $k = 4$:

$$\begin{aligned}
 S = & 2^0 \cdot L_{44} + 2^{n/k} \cdot (H_{44} + L_{34} + L_{43}) + 2^{2n/k} \cdot (H_{34} + H_{43} + L_{24} + L_{33} + L_{42}) + \\
 & + 2^{3n/k} \cdot (H_{24} + H_{33} + H_{42} + L_{14} + L_{23} + L_{32} + L_{41}) + \\
 & + 2^{4n/k} \cdot (H_{14} + H_{23} + H_{32} + H_{41} + L_{13} + L_{22} + L_{31}) + \\
 & + 2^{5n/k} \cdot (H_{13} + H_{22} + H_{31} + L_{12} + L_{21}) + 2^{6n/k} \cdot (H_{12} + H_{21} + L_{11}) + 2^{7n/k} \cdot H_{11} \quad (2.6)
 \end{aligned}$$

Este resultado pode ser desenvolvido para outros valores de k e , por indução matemática, é obtida uma expressão geral para o algoritmo de multiplicação em função das partes mais significativas e menos significativas dos produtos parciais P_{ij} :

$$S = \sum_{i=0}^{2k-1} 2^{in/k} \cdot \left[\sum_{r=0}^i L_{k-i+r, k-r} + \sum_{r=0}^{i-1} H_{k-i+r+1, k-r} \right] \quad (2.7)$$

Neste algoritmo de multiplicação, uma vez calculadas as partes mais e menos significativas dos produtos parciais P_{ij} , estas são deslocadas para a esquerda em passos de n/k bits conforme o grau de significância de cada uma e, posteriormente, são somadas resultando no produto das duas palavras A e B.

Para um melhor entendimento deste algoritmo observe a Fig. 2.2 onde a operação de multiplicação $A \cdot B$, para $k = 2$, é realizada através da multiplicação das sub-palavras $W_i(A)$ e $W_j(B)$ resultando nos sub-produtos H_{ij} e L_{ij} . Estes sub-produtos são dispostos em colunas conforme o grau de significância destes, onde cada coluna (i) está associada a um deslocamento de $(i) \cdot n/k$ bits. Em cada coluna (i) são somadas as sub-palavras de n/k bits

dispostas nesta coluna com as sub-palavras propagadas pela coluna (i-1) a direita, sendo os n/k bits menos significativos desta soma mantidos nesta coluna (i) e os n/k bits mais significativos são propagados para a coluna (i+1) a esquerda. Após a finalização deste processo de acumulação e propagação de sub-palavras de n/k bits, tem-se o resultado do produto $A \cdot B$ na composição das sub-palavras R_i , ou seja, $A \cdot B = R_1 R_2 R_3 R_4$. Este processo de multiplicação, acumulação e propagação de sub-palavras de n/k bits, ilustrado na Fig. 2.2, permite que se proponham estruturas que utilizam-se de unidades processadoras (multiplicadores e acumuladores) que operam com sub-palavras de n/k bits e conexões de n/k bits entre estas unidades para a realização da operação de multiplicação. Baseando-se neste algoritmo regular, na seção seguinte serão desenvolvidas estruturas sistólicas para a realização da operação produto interno.

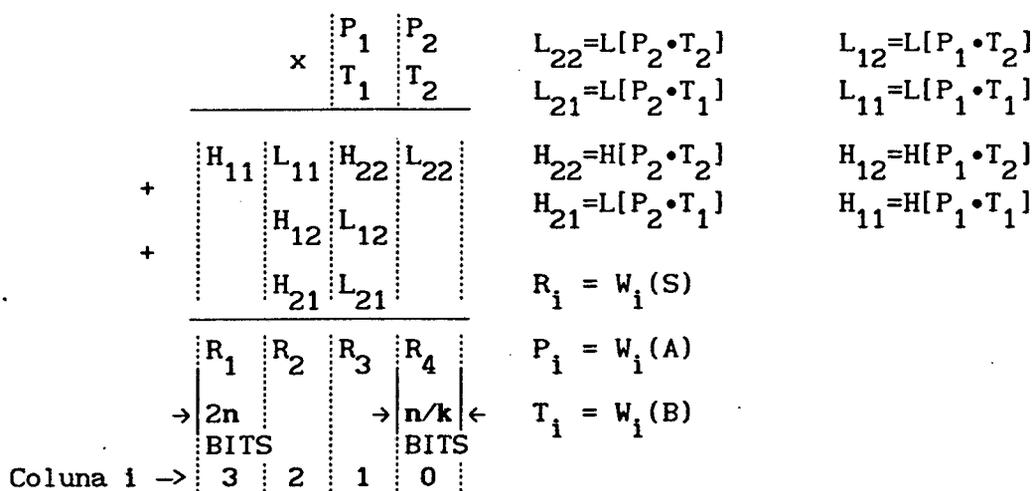


Fig. 2.2 - Exemplificação da operação de multiplicação para $k = 2$.

2.4 - Estruturas Sistólicas

Para implementar a operação de multiplicação em uma estrutura sistólica, deve-se, a princípio, realizar tanto os produtos parciais, como a soma destes, através de um arranjo de unidades processadoras operando paralelamente e conectadas localmente, apresentando um fluxo regular e sincronizado de dados. Neste trabalho, optou-se pela utilização de unidades multiplicadoras e somadoras operando com palavras de n/k bits. Estas unidades são então conectadas localmente e isoladas por registradores de armazenamento de $2n/k$ bits controlados por um único pulso de clock T_{ck} . Um diagrama destas unidades é mostrado na Fig. 2.3.

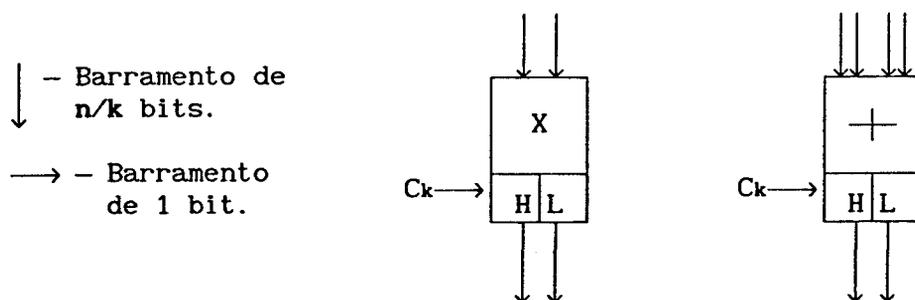


Fig. 2.3 - Diagrama das unidades multiplicadora e somadora.

Os multiplicadores são utilizados para realizar os produtos parciais P_{ij} (H_{ij} L_{ij}) e os somadores para realizar a acumulação destes produtos. As unidades são conectadas de forma a satisfazer a equação (2.7), gerando a estrutura mostrada na Fig. 2.4. Esta estrutura implementa, para $k = 4$, a operação produto interno de N termos, $\sum_{m=1}^N A_m \cdot B_m$, através de um arranjo de unidades multiplicadoras e somadoras agrupadas em dois tipos diferentes de células, apresentadas na Fig. 2.5. A célula tipo I consiste de um multiplicador de n/k bits, para a realização do produto parcial P_{ij} , e de uma

estrutura somadora com 4 entradas de n/k bits. Esta estrutura somadora realiza a acumulação dos produtos parciais e de palavras de n/k bits provenientes de células adjacentes. Cada unidade somadora e multiplicadora apresenta um registrador na saída sincronizado por um pulso de clock T_{ck} . Os registradores temporários R_{ij} desta célula, sincronizados pelo pulso de clock T_{ck} , difundem as palavras $W_i(A_m)$ e $W_j(B_m)$ para as outras células do tipo I. A célula tipo II consiste de uma unidade somadora de n/k bits com 3 entradas.

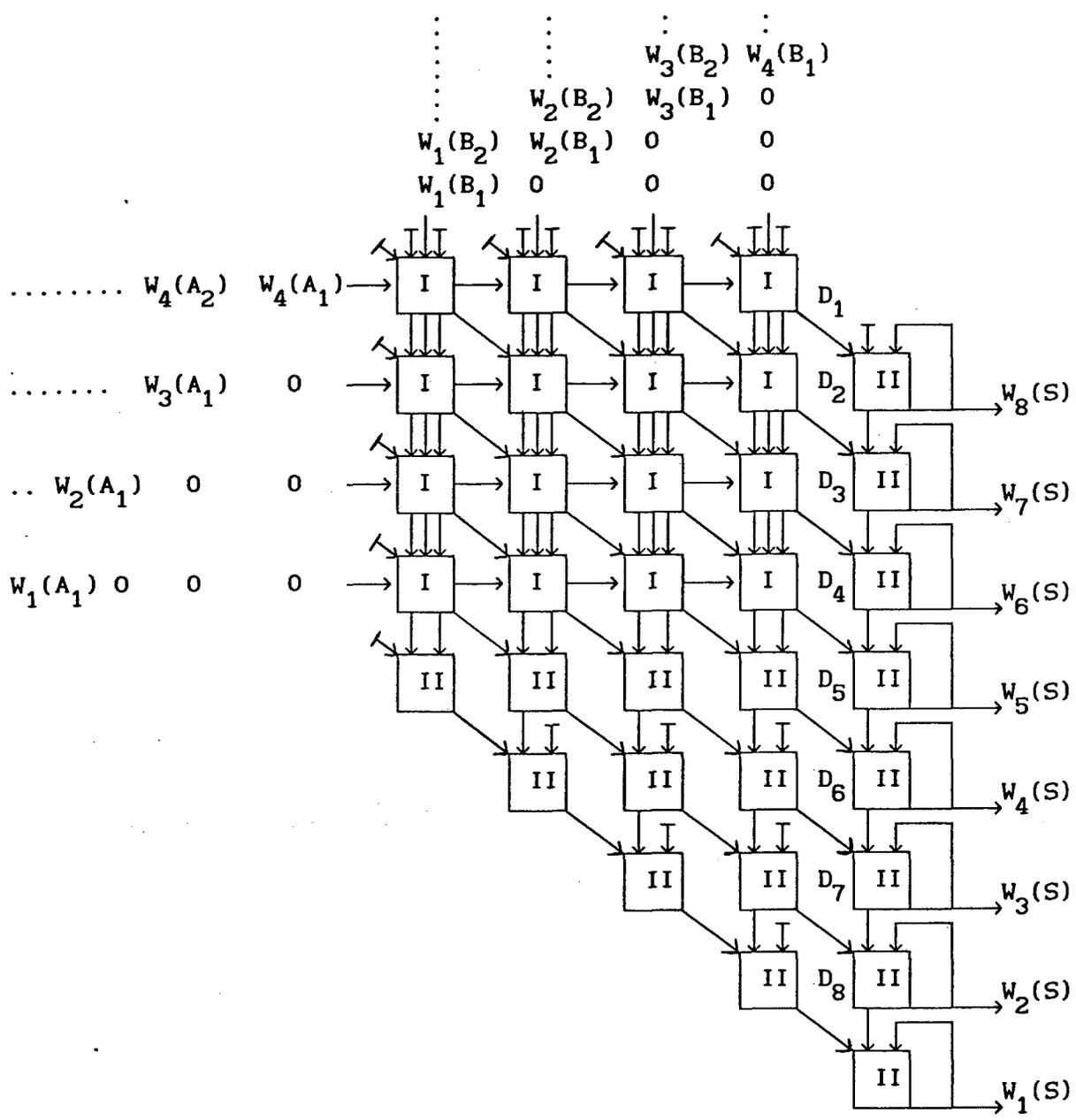
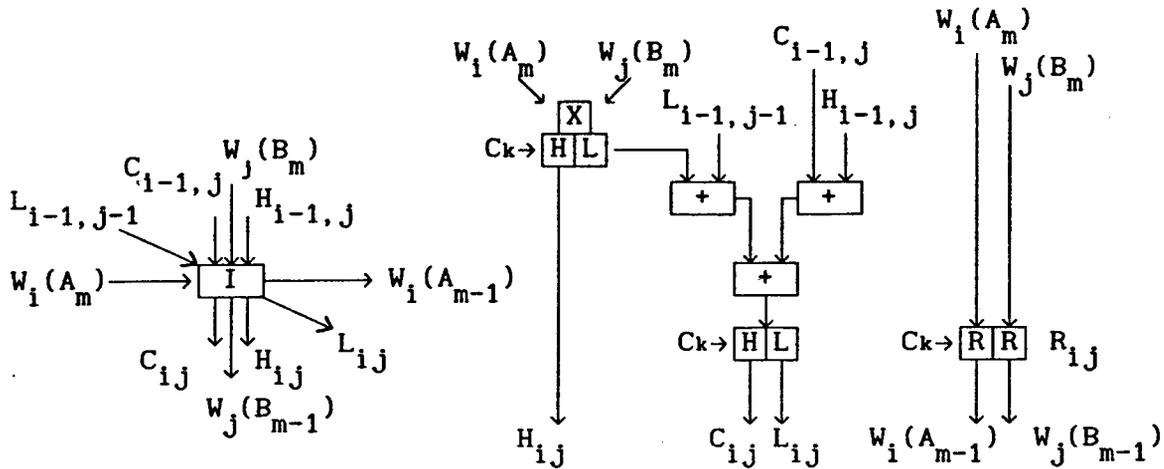


Fig. 2.4 - Processador sistólico da operação PI para $k = 4$.

Célula tipo I (posição (i,j) no arranjo sistólico):



Célula tipo II (posição (i,j) no arranjo sistólico):

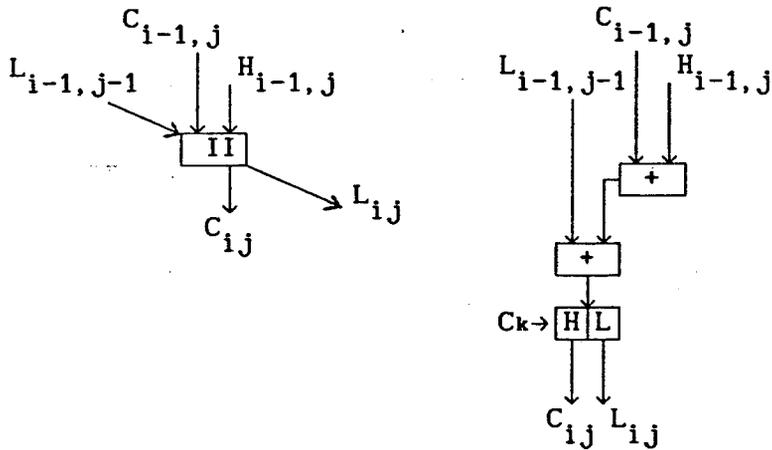


Fig. 2.5 - Conteúdo das células tipo I e tipo II.

O fluxo e o processamento de dados na estrutura da Fig. 2.4 são explicados a seguir:

1) As m -ésimas sub-palavras de n/k bits de A_m e B_m , $W_i(A_m)$ e $W_j(B_m)$, entram na estrutura através das células da primeira linha e da primeira coluna, sendo mantidas durante um período de clock T_{ck} nos registradores temporários, R_{ij} . As sub-palavras $W_i(A_m)$ são difundidas horizontalmente para a

direita a cada pulso de clock. As sub-palavras $W_j(B_m)$, por sua vez, são difundidas verticalmente para baixo a cada pulso de clock. A cada pulso de clock novas sub-palavras de A_m e B_m entram na estrutura e os registradores temporários R_{ij} são atualizados. As sub-palavras colocadas na entrada dos multiplicadores, através dos registros R_{ij} , no tempo $t_0 = m \cdot T_{ck}$, são processadas durante um intervalo de tempo T_{ck} , produzindo os produtos parciais $P_{ij} = (H_{ij} L_{ij})$, os quais são armazenados nos registradores de $2n/k$ bits na saída dos multiplicadores, no tempo $t_1 = (m+1) \cdot T_{ck}$. Estes resultados são mantidos nestes registradores durante um intervalo T_{ck} , do tempo $t_1 = (m+1) \cdot T_{ck}$ até o tempo $t_2 = (m+2) \cdot T_{ck}$, estando disponíveis, durante este intervalo de tempo, para as unidades somadoras conectadas aos mesmos.

ii) A cada pulso de clock são acumuladas 4 palavras de n/k bits nos somadores das células tipo I e acumuladas 3 palavras de n/k bits nos somadores das células tipo II. Para que não ocorra "overflow" na saída dos somadores é necessário que no registrador de $2n/k$ de saída do acumulador, o qual pode acomodar um valor máximo igual a $(2^{2n/k} - 1)$, possam ser acumuladas 4 palavras de valor $(2^{n/k} - 1)$. Para que isto ocorra, a desigualdade abaixo deve ser verificada:

$$4 \cdot (2^{n/k} - 1) \leq (2^{2n/k} - 1) \quad (2.8)$$

Esta desigualdade é verificada somente para $n/k \geq 2$. Para $n/k = 1$, que seria o caso das estruturas a nível de bit, esta desigualdade não é satisfeita. Posteriormente, neste capítulo, será apresentada uma estrutura alternativa que não apresenta esta limitação.

iii) As conexões destas unidades somadoras e multiplicadoras são locais e

regulares. Cada unidade de uma diagonal D_i na estrutura da Fig. 2.4 recebe palavras de n/k bits no tempo $t_0 = m \cdot T_{ck}$. Após um intervalo de tempo T_{ck} ($t_1 = (m+1) \cdot T_{ck}$) os resultados do processamento de cada unidade são armazenados nos seus registradores, mantendo-se disponíveis até o início do próximo período de clock. Neste instante ($t_2 = (m+2) \cdot T_{ck}$), novos dados serão armazenados nestes registradores, provenientes do processamento dos dados mantidos na entrada da unidade de $t_1 = (m+1) \cdot T_{ck}$ até $t_2 = (m+2) \cdot T_{ck}$. Assim, a cada pulso de clock, novos dados são armazenados nos registradores de saída das unidades.

Os $2n/k$ bits de cada registrador, na saída das unidades, são particionados em parte alta e baixa com n/k bits cada. As partes altas são os n/k bits mais significativos dos resultados armazenados nos registradores e são difundidos para as unidades somadoras da diagonal seguinte D_{i+1} . As partes baixas são os n/k bits menos significativos dos resultados armazenados nos registradores e são difundidos para as unidades somadoras da mesma diagonal D_i . Estas diagonais e as transferências de dados entre células estão ilustradas nas figuras 2.4 e 2.5.

iv) No final das diagonais D_i são colocadas unidades somadoras, com as partes baixas dos resultados realimentadas às suas entradas. As partes altas são difundidas para a célula tipo II da diagonal seguinte D_{i+1} . Em cada célula tipo II do final da diagonal D_i são acumulados os resultados parciais de $W_i(S_m)$. Assim, após um certo número N_L de períodos de clock, completa-se o tempo latência da estrutura, $T_L = N_L \cdot T_{ck}$, e os registradores de armazenamento das partes menos significativas conterão o resultado correto de $W_i(S_m)$. Desta forma, tendo-se as sub-palavras $W_i(S_m)$, obtém-se o produto $A_m \cdot B_m$. No entanto, a operação produto interno requer a realização do somatório de produtos de palavras A_m e B_m . Para isto, colocam-se na entrada da estrutura novas palavras

A_m e B_m , a cada ciclo de clock, de forma que após $N + N_L$ períodos de clock, tem-se o resultado do somatório dos N produtos armazenado nos registradores associados as palavras $W_i(S_{N+N_L})$.

Para que as sub-palavras de A_m e B_m cheguem sincronizadamente em cada célula, produzindo o produto parcial correto, é necessário incluir atrasos relativos entre as sub-palavras $W_i(A_m)$ e $W_j(B_m)$, os quais são representados por zeros na Fig. 2.4. Estes atrasos não alteram a taxa de entrada/saída de dados.

Na Fig. 2.4 é apresentada uma estrutura sistólica para $k = 4$. Devido à regularidade desta estrutura, outras estruturas similares podem ser facilmente obtidas para quaisquer outros valores de k (para n/k inteiro e diferente de 1), caracterizando assim uma família de estruturas sistólicas. As estruturas pertencentes a esta família apresentam k^2 células do tipo I dispostas quadrangularmente, um arranjo triangular de $k^2/2 + k/2$ células do tipo II e uma coluna de $2k + 1$ células do tipo II dispostas na última coluna da estrutura. Assim, utilizando-se esta lei de formação, pode-se obter qualquer estrutura da família e determinar sua respectiva complexidade e desempenho, respeitando a restrição de que a relação n/k deve ser inteira e diferente de 1.

Este arranjo das unidades segue rigorosamente o algoritmo de multiplicação, realizando o produto interno pela acumulação dos produtos, $A_m \cdot B_m$, nas unidades colocadas no final das diagonais.

2.5 - Redução do Número de Conexões

Na implementação destas estruturas é desejável uma redução do número de

conexões entre as células, reduzindo assim a complexidade do "layout" e a área, tanto para implementações em VLSI quanto através de circuitos discretos. As células tipo I difundem duas linhas de dados de n/k bits, C_{ij} e H_{ij} , para as células tipo I situadas abaixo. No entanto, através de alterações nas células do tipo I, consegue-se eliminar a conexão C_{ij} , resultando na eliminação do arranjo triangular de células tipo II, anteriormente necessário para a acumulação das palavras C_{ij} propagadas pelas células anteriores. Na estrutura da Fig. 2.4, as palavras de n/k bits, H_{ij} e C_{ij} , das partes altas dos resultados da célula tipo I são transmitidas e somadas nas células seguintes. Obviamente esta soma pode ser realizada dentro da célula tipo I, o que reduzirá o número de conexões entre células. Este procedimento libera também uma das entradas das células tipo II, permitindo uma redução substancial do número de células no arranjo, conforme explicado a seguir.

Na Fig. 2.6 tem-se o diagrama da nova célula tipo I, com uma entrada e uma saída a menos. Para que não ocorra "overflow" no registrador acumulador na saída do somador de três palavras de n/k bits, a desigualdade abaixo deve ser verificada:

$$(2^{n/k}-1) \cdot (2^{n/k}-1) + 2 \cdot (2^{n/k}-1) \leq (2^{2n/k}-1) \quad (2.9)$$

Operando-se esta desigualdade tem-se:

$$2^{2n/k} - 2 \cdot 2^{n/k} + 1 + 2 \cdot 2^{n/k} - 2 \leq (2^{2n/k}-1) \quad (2.10)$$

$$(2^{2n/k}-1) \leq (2^{2n/k}-1) \quad (2.11)$$

de células tipo II.

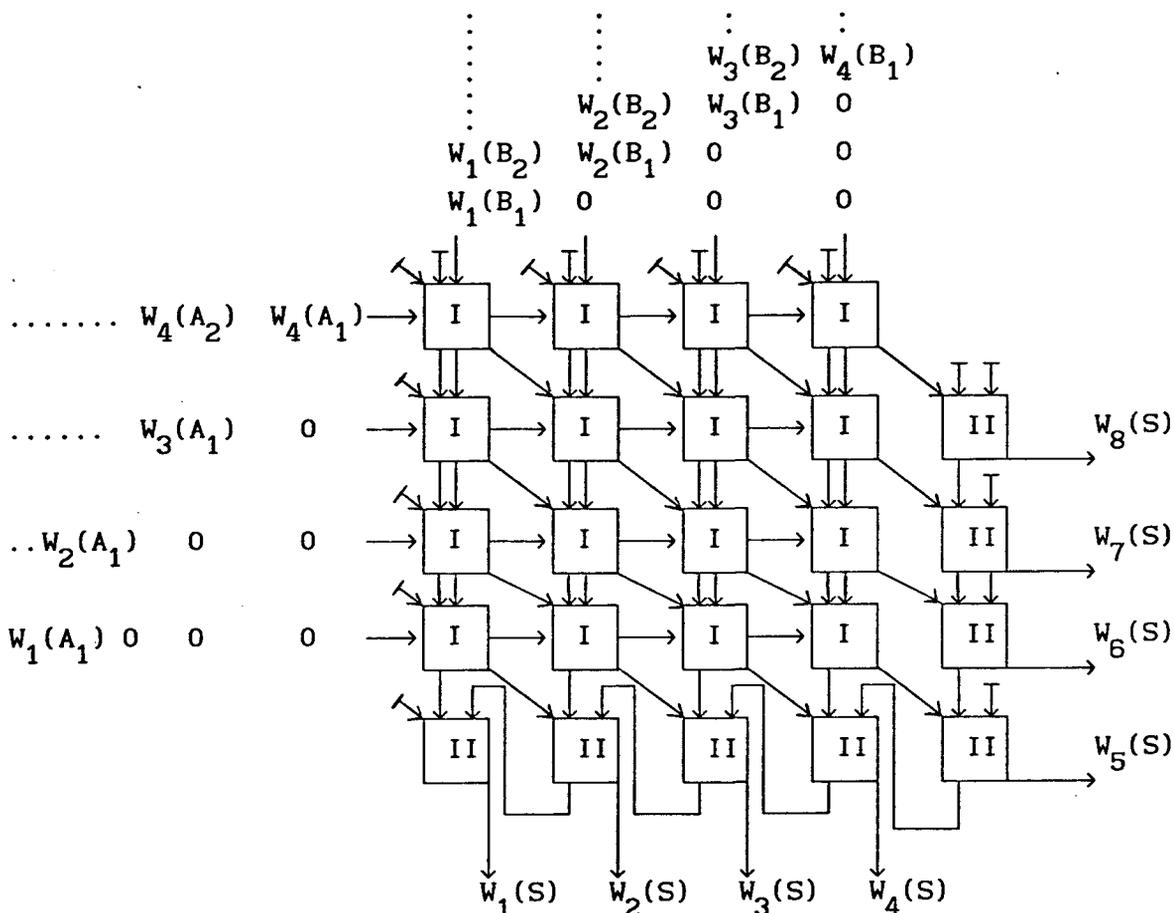
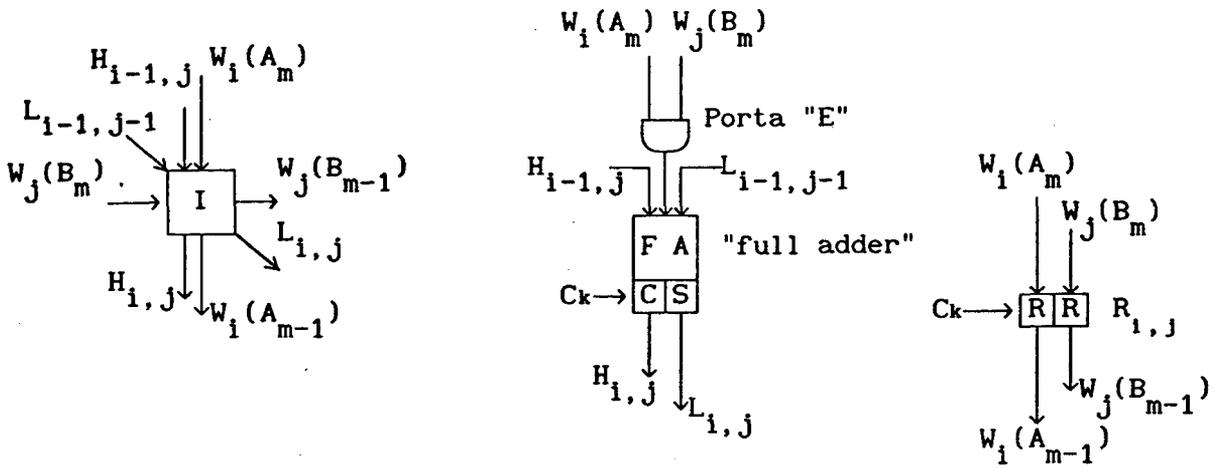
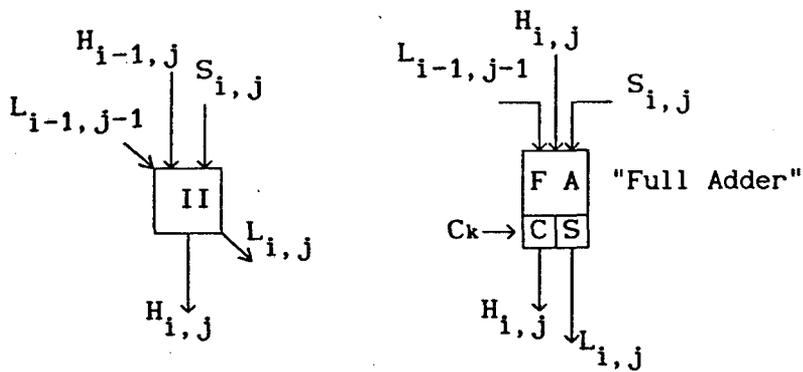


Fig. 2.7 - Processador sistólico otimizado com $k = 4$.

Esta estrutura sistólica pode ser facilmente estendida para outros valores de k , utilizando-se a seguinte lei de formação para compor novas estruturas: k^2 células tipo I dispostas em um arranjo quadrangular, k células tipo II colocadas na linha logo abaixo destas e uma coluna de k células colocadas ao lado do arranjo quadrangular. Desta forma pode-se obter qualquer estrutura para esta família, desde que a relação n/k seja inteira e diferente

de 1.

Ambas as famílias apresentadas anteriormente são limitadas para $n/k \geq 2$. Pode-se também implementar uma estrutura para $n/k = 1$ utilizando-se a mesma filosofia empregada no desenvolvimento das estruturas anteriores. Esta estrutura utiliza células tipo I com número de conexões reduzido apresentada na Fig. 2.6, mas necessita de um arranjo triangular de células tipo II apresentada na Fig. 2.5. As células utilizadas operam a nível de bit, sendo constituídas de portas "E" para realizar a multiplicação e de "full adders" para realizar a acumulação. Os diagramas destas células estão mostradas na Fig. 2.8 e a estrutura sistólica a nível de bit, utilizando-se a filosofia aqui desenvolvida, é ilustrada pela Fig. 2.9, para $n = k = 2$. Da mesma forma que as duas famílias de estruturas anteriores são estendidas para outros valores de k , pode-se obter outras estruturas a nível de bit, como aquela apresentada para $k = 2$, seguindo-se a lei de formação: k^2 células tipo I, um arranjo triangular de células do tipo II e uma coluna de células do tipo II.

Célula tipo I:Célula tipo II:Fig. 2.8 - Conteúdos das células da estrutura sistólica com $n = k$.

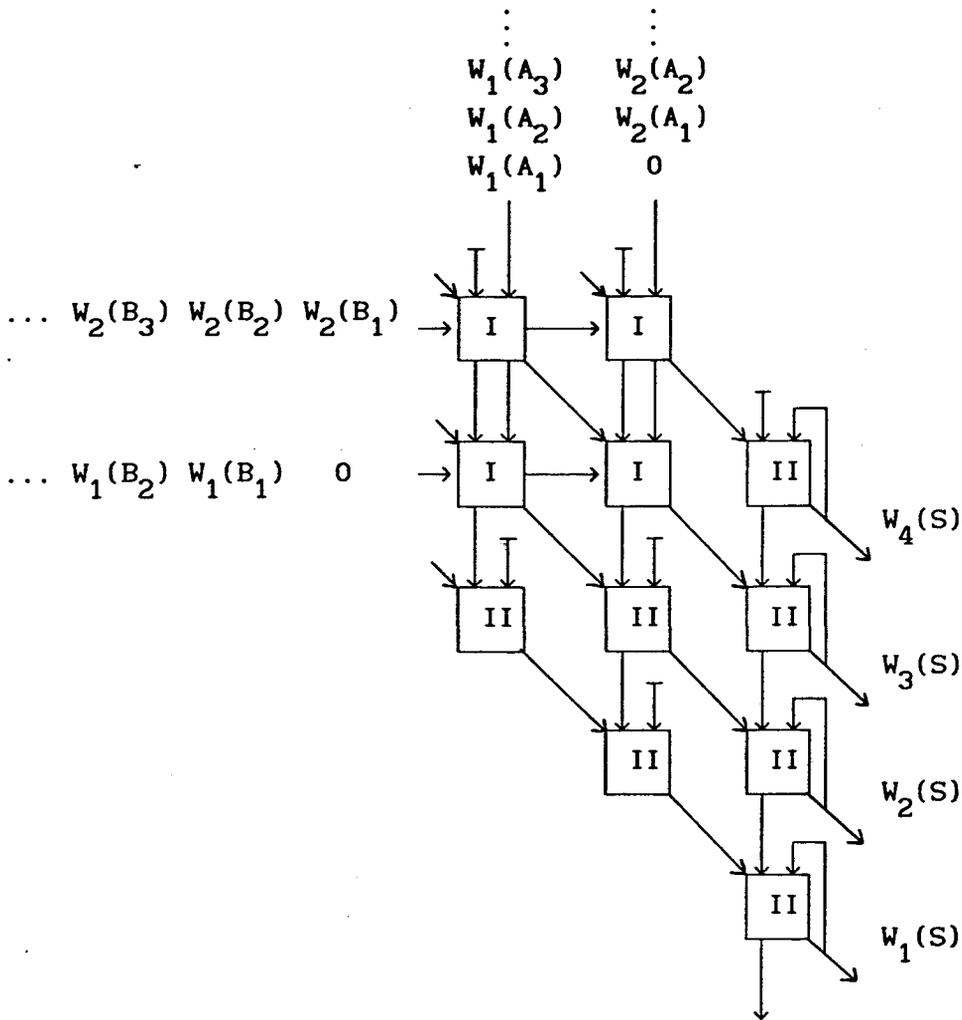


Fig. 2.9 - Estrutura sistólica ($n = k$).

As estruturas apresentadas até aqui operam somente com números unipolares. Em PDS é muito importante a possibilidade de trabalhar com seqüências de números que podem assumir valores positivos ou negativos. Estas seqüências são denominadas bipolares. Na próxima seção serão desenvolvidas estruturas sistólicas que operam com seqüências bipolares realizando a operação produto interno.

2.6 - Estruturas Operando com Números Bipolares

Existem duas formas mais freqüentemente utilizadas para a representação de números binários negativos: representação em sinal e magnitude e representação em complemento de 2. Na maioria das aplicações, utiliza-se a representação em complemento de 2, pois a mesma requer circuitos mais simples para realizar acumulações de números binários bipolares [14]. Assim, neste trabalho será considerado que os dados A_m e B_m estão representados em complemento de 2.

Considere-se dois números, A_m e B_m , de n bits cada um, representados em complemento de 2. Estes números são particionados em k sub-palavras de n/k bits:

$$A_m = W_1(A_m) W_2(A_m) \dots W_k(A_m).$$

$$B_m = W_1(B_m) W_2(B_m) \dots W_k(B_m).$$

Os bits de sinais destes números estão posicionados no bit mais significativo, sendo iguais a 1 para números negativos e iguais a zero para números positivos. Realizando-se o produto destas duas palavras, representadas em complemento de 2, através de um multiplicador de n bits, resulta em uma palavra de $2n$ bits, enquanto que o bit de sinal do produto destas, posiciona-se na n -ésima posição, adicionando-se aos bits associados à magnitude do produto. Desta forma, a correta magnitude é alterada pela inclusão do bit de sinal na n -ésima posição. Devido a esta característica do produto de números representados em complementos de 2, é necessário que se estenda o comprimento das palavras A_m e B_m para, no mínimo, $2n$ bits. Estas palavras A_m ou B_m são estendidas através da inclusão de, no mínimo, n bits

iguais aos bits de sinal das palavras, de forma que ~~o bit de sinal do produto~~ se estabeleça na posição $2n$ ou mais a esquerda, não alterando o resultado do produto. As novas palavras estendidas, A_{me} e B_{me} , serão compostas das palavras originais, A_m e B_m , e de palavras adicionais denominadas $W_s(A_{me})$ e $W_s(B_{me})$. Estas palavras adicionais consistem de $n + n/k$ bits iguais ao bit de sinal da palavra original. Estas palavras são geradas pela própria estrutura, não requerendo portanto que as palavras de entrada sejam previamente estendidas para $2n + n/k$ bits. Nestas palavras adicionais são considerados n/k bits de guarda para a acumulação de novos produtos. Desta forma as palavras estendidas são definidas como:

$$A_{me} = W_s(A_{me}) \quad A_m = W_{s1}(A_{me}) W_{s2}(A_{me}) \dots W_{s(k+1)}(A_{me}) W_1(A_m) \dots W_k(A_m)$$

$$B_{me} = W_s(B_{me}) \quad B_m = W_{s1}(B_{me}) W_{s2}(B_{me}) \dots W_{s(k+1)}(B_{me}) W_1(B_m) \dots W_k(B_m)$$

Uma estrutura que realiza o produto interno das palavras bipolares A_m e B_m está apresentada na Fig. 2.10. Esta estrutura aplica-se para $k = 2$ e pode ser desenvolvida para outros valores de k . Contudo, esta implementação apresenta um número muito grande de células tipo I, quando comparada com as estruturas anteriores para números unipolares, resultando-a ineficiente. Este acréscimo do número de células tipo I é devido à realização dos produtos envolvendo as sub-palavras de $W_s(A_{me})$ e $W_s(B_{me})$. Na próxima seção será apresentado um algoritmo otimizado que transforma as operações de multiplicação, relacionadas a estas palavras, em operações de acumulações, reduzindo-se substancialmente o número de células da estrutura.

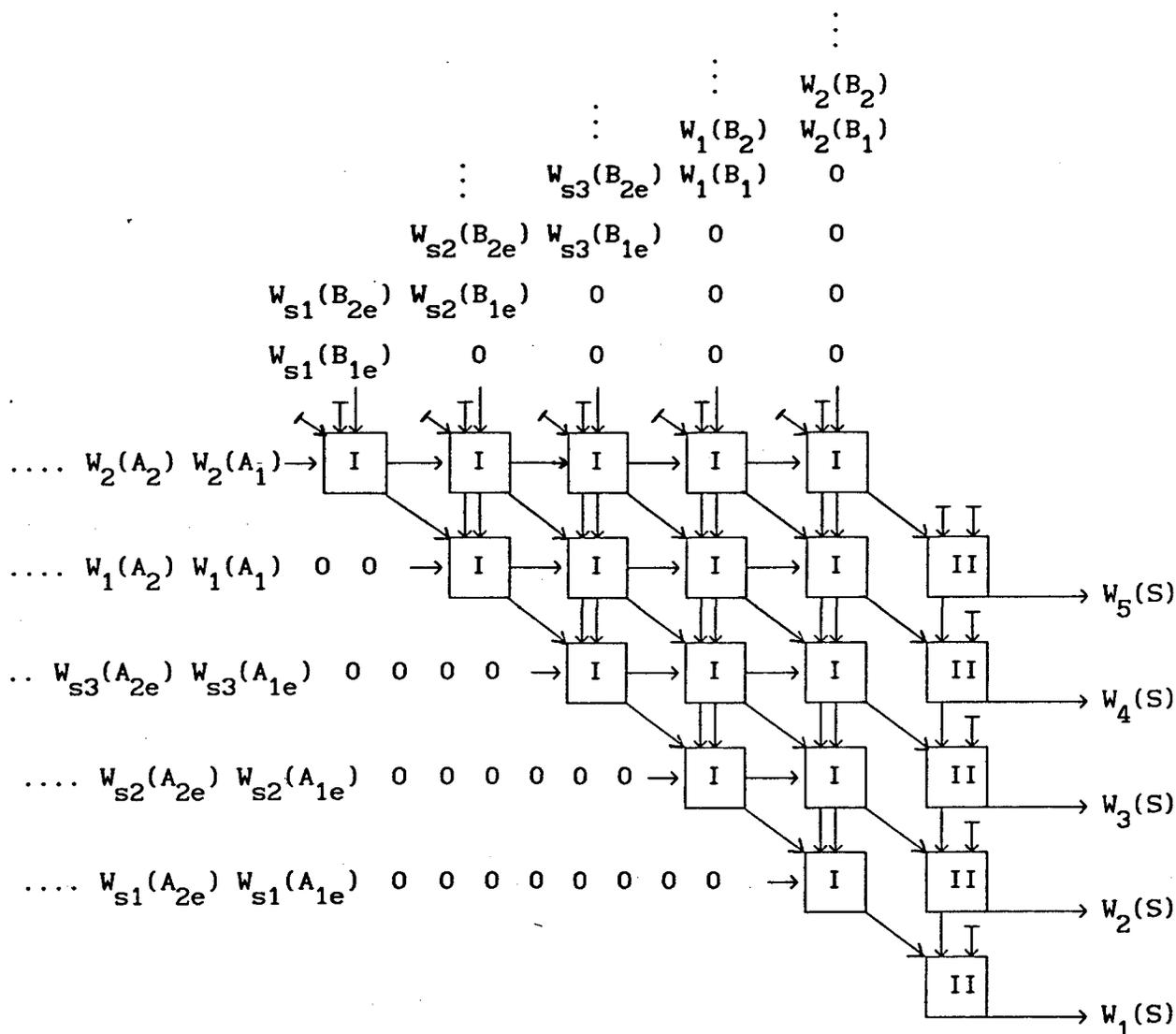


Fig. 2.10 - Processador sistólico PI para números bipolares ($k = 2$).

2.7 - Otimização do Algoritmo para Números Bipolares

Seja o produto das palavras estendidas A_{me} e B_{me} representado abaixo por

S_{me} :

$$S_{me} = \begin{bmatrix} W_s(A_{me}) & A_m \end{bmatrix} \cdot \begin{bmatrix} W_s(B_{me}) & B_m \end{bmatrix} = \quad (2.12)$$

Expandindo-se esta operação em produtos parciais, obtém-se:

$$= A_m \cdot B_m + A_m \cdot W_S(B_{me}) \cdot 2^n + B_m \cdot W_S(A_{me}) \cdot 2^n + W_S(A_{me}) \cdot W_S(B_{me}) \cdot 2^{2n} \quad (2.13)$$

Analisando a expressão acima, e lembrando que $W_S(X)$ constitui-se de $n + n/k$ bits iguais a 1 (X negativo) ou iguais a 0 (X positivo), observa-se que:

$$W_S(A_{me}) = 0, \text{ se } A_m \geq 0 \text{ e } W_S(A_{me}) = 2^{(k+1)n/k} - 1, \text{ se } A_m < 0 \quad (2.14)$$

$$W_S(B_{me}) = 0, \text{ se } B_m \geq 0 \text{ e } W_S(B_{me}) = 2^{(k+1)n/k} - 1, \text{ se } B_m < 0 \quad (2.15)$$

Baseando-se nestas considerações, o produto S_{me} pode ser escrito como:

$$S_{me} = A_m \cdot B_m + C_1 + C_2 + C_3 \quad (2.16)$$

Onde:

$$C_1 = A_m \cdot 2^n \cdot (2^{(k+1)n/k} - 1), \text{ se } B_m < 0 \text{ e } C_1 = 0, \text{ se } B_m \geq 0 \quad (2.17)$$

$$C_2 = B_m \cdot 2^n \cdot (2^{(k+1)n/k} - 1), \text{ se } A_m < 0 \text{ e } C_2 = 0, \text{ se } A_m \geq 0 \quad (2.18)$$

$$C_3 = (2^{(k+1)n/k} - 1)^2 \cdot 2^{2n}, \text{ se } A_m < 0 \text{ e } B_m < 0 \text{ caso contrário } C_3 = 0 \quad (2.19)$$

Assim, notando que qualquer bit posicionado à esquerda do bit de sinal do produto S_{me} , posicionado na $(2n + n/k)$ -ésima posição, pode ser desprezado, pois não é relevante para resultado, os termos C_1 , C_2 e C_3 podem ser simplificados da seguinte forma:

$$C_1 = -A_m \cdot 2^n \text{ se } B_m < 0 \text{ e } C_1 = 0 \text{ se } B_m \geq 0; \quad (2.20)$$

$$C_2 = -B_m \cdot 2^n \text{ se } A_m < 0 \quad \text{e} \quad C_2 = 0 \text{ se } A_m \geq 0; \quad (2.21)$$

$$C_3 = 2^{2n} \text{ se } A_m < 0 \text{ e } B_m < 0 \quad \text{e} \quad C_3 = 0 \text{ caso contrário.} \quad (2.22)$$

Assim, os produtos relacionados às palavras $W_s(X)$ são realizados por simples acumulações das palavras originais A_m e B_m , lembrando que o produto da palavra P por um número 2^d pode ser realizado por um simples deslocamento da palavra P de d bits para a esquerda.

Este algoritmo pode ser realizado por uma estrutura do tipo unipolar apresentada na Fig. 2.7, juntamente com um pequeno circuito adicional para realizar os complementos de 2 das palavras A_m e B_m necessários para o cálculo de C_1 , C_2 e C_3 . É necessário incluir uma lógica de controle associada a este circuito para indicar quando A_m e/ou B_m devem ser complementados, realizando C_1 , C_2 e C_3 . Esta lógica de controle consiste simplesmente de uma verificação dos bits de sinal das palavras A_m e B_m . Na Fig. 2.11 está representada uma nova estrutura para $k = 2$ que realiza a operação produto interno para números bipolares. A célula tipo III consiste de $2n/k$ portas "OU-Exclusivo", $2n/k$ portas "E", um acumulador de 3 entradas de n/k bits e um registrador de $2n/k$ bits, conforme ilustrado na Fig. 2.12. Para realizar os termos C_1 , C_2 e C_3 , além das células tipo III, são necessários um circuito meio somador ("half adder") conectado à célula tipo III e uma porta "E", a qual realiza o termo C_3 . Como nos outros casos, é possível obter-se facilmente as estruturas para outros valores de k . Esta família de estruturas tem a mesma lei de formação do que a família apresentada para números unipolares, com a inclusão de $(k + 1)$ células do tipo III, um "half adder" e uma porta "E" conectadas da forma ilustrada na Fig. 2.11.

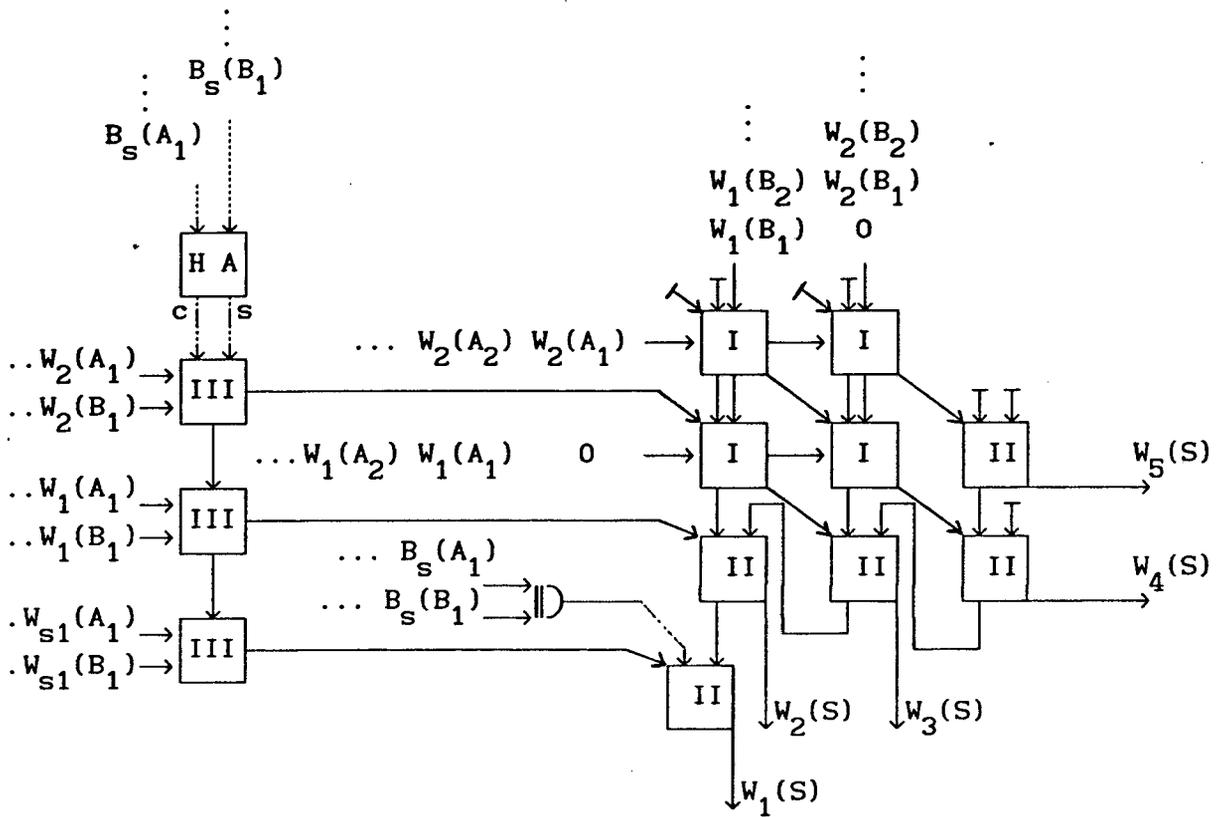


Fig. 2.11 - Estrutura otimizada para números bipolares ($k = 2$)

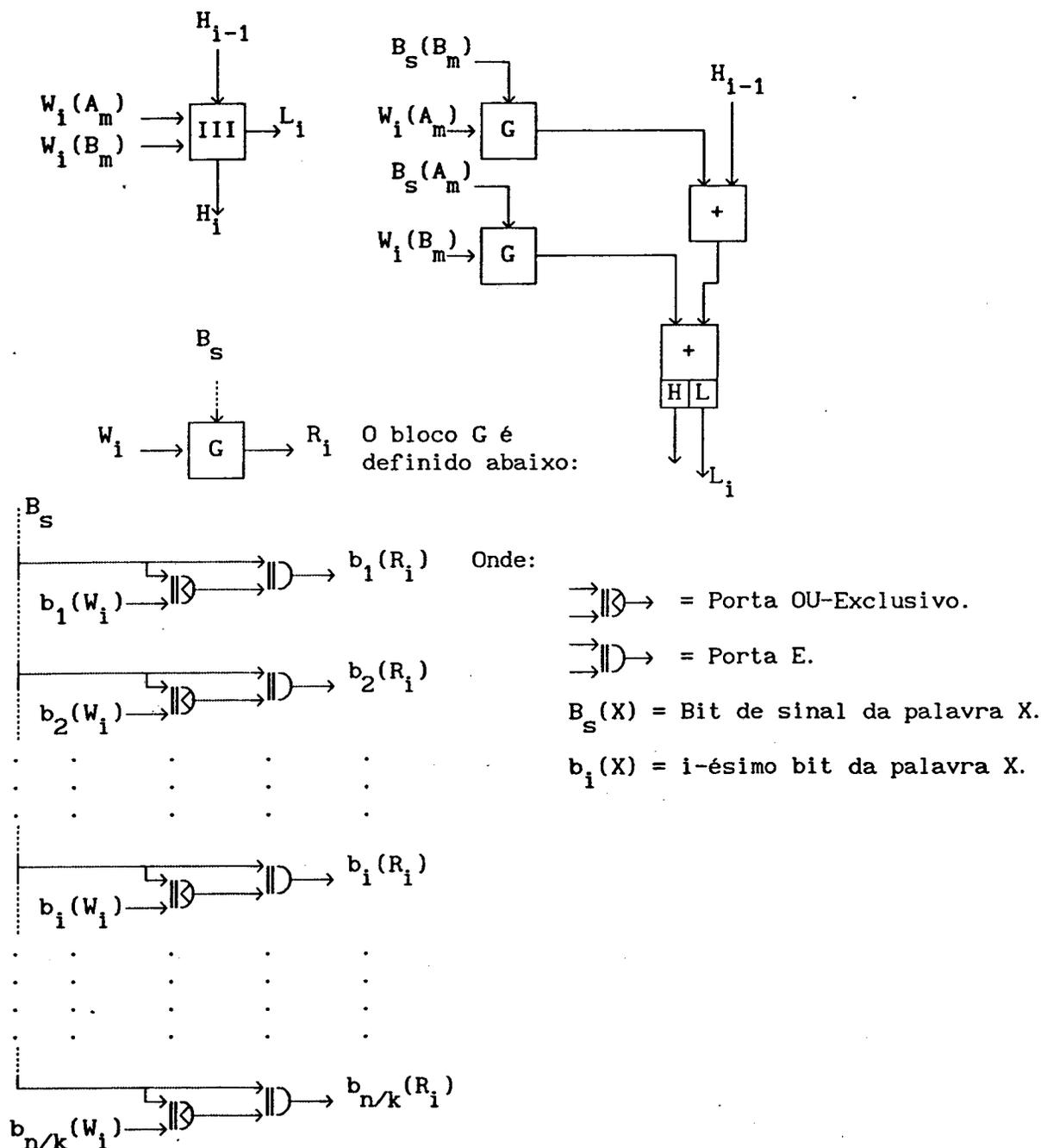


Fig. 2.12 - Diagrama da célula tipo III

A célula tipo III pode ser simplificada se uma das seqüências, A_m ou B_m , for unipolar. Além disso, estruturas semelhantes às apresentadas acima, podem ser projetadas para seqüências com comprimentos de palavra diferentes, ou

seja, A_m representada por n bits e B_m representada por r bits. Esta diferença de representações resulta em uma estrutura composta por um arranjo retangular $n \cdot r$ de células tipo I, juntamente com as células tipo II colocadas na linha e na coluna da extremidade inferior e a direita do arranjo. Todas as otimizações se aplicam para este tipo de estrutura, desde que $r = n + q \cdot n/k$ ou $n = r + q \cdot n/k$, com q um valor inteiro. Esta estrutura foi simulada em computador digital e o seu desenvolvimento está apresentado no Capítulo 4, para $r = 12$ e $n = 16$. As células constituintes desta estrutura também foram implementadas utilizando-se circuitos integrados comerciais, com o intuito de validar as estruturas propostas neste trabalho.

2.8 - Conclusões

Neste capítulo foram apresentadas as estruturas para a realização da operação produto interno, sem maiores considerações em relação ao desempenho ou às áreas envolvidas na implementação destas estruturas. Foram também estabelecidas as leis de formação das famílias de estruturas desenvolvidas e operações com números unipolares e bipolares foram consideradas. No próximo capítulo, análises de desempenho destas estruturas e comparações com outras estruturas propostas na literatura serão apresentadas, dando condições ao projetista de optar pela estrutura que melhor se adapta ao compromisso de custo e de desempenho na implementação de uma aplicação desejada.

CAPÍTULO 3

Avaliação de Desempenho das Estruturas

3.1 - Introdução

Neste capítulo serão determinadas as principais características das estruturas propostas no capítulo anterior, assim como, comparações entre estas estruturas e outras apresentadas na literatura. Estas comparações proporcionam uma visão clara das principais características das famílias de estruturas desenvolvidas neste trabalho. Serão apresentados gráficos e tabelas explicitando o compromisso existente entre a área requerida para a integração e a velocidade de processamento obtida na utilização das novas estruturas propostas.

3.2 - Determinação da Área e do Tempo de Propagação de Dados

Os parâmetros de área e de tempo de propagação serão determinados e normalizados em unidades de área, A_E , e de tempo de propagação, T_E , de uma porta "E" [7]. Desta forma, por exemplo, o tempo de propagação de um inversor pode ser aproximado para $0,5 \cdot T_E$. A utilização dos parâmetros da porta "E" como unidade de comparação apresenta a vantagem de reduzir a dependência dos resultados com a tecnologia de implementação das portas utilizadas. Isto porque existem variações na área e no número de transistores utilizado na implementação de uma determinada porta, dependendo do "layout", do projeto e da tecnologia utilizada.

Para a determinação das áreas requeridas e dos tempos de propagação das estruturas, é necessário obter inicialmente os parâmetros das unidades processadoras constituintes destas estruturas, ou seja, os multiplicadores e os somadores de n/k bits, e os registradores de $2n/k$ bits.

Os somadores utilizados nas estruturas serão do tipo apresentado em [14], o qual utiliza n somadores completos ("full adders") para realizar a soma de duas palavras de n bits, como ilustrado na Fig. 3.1. Os multiplicadores utilizados nas estruturas, serão do tipo multiplicador sistólico apresentado em [17]. Este multiplicador, para $n = 4$, está ilustrado na Fig. 3.2, onde o bloco I utiliza de um somador completo e uma porta "E" para realizar o produto parcial, $a_i \cdot b_j$, a nível de bit.

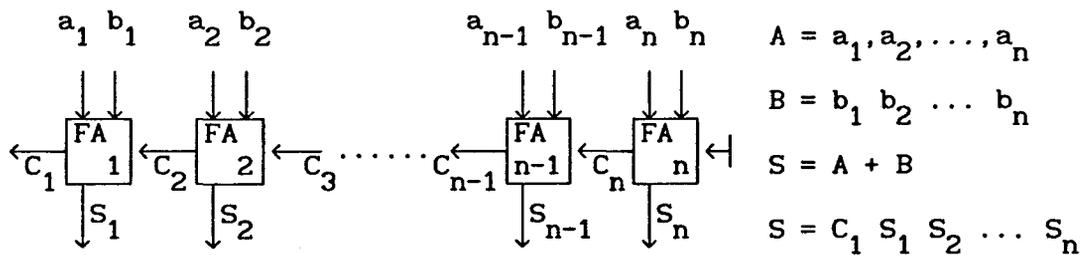


Fig. 3.1 - Diagrama de um somador de duas palavras de n bits

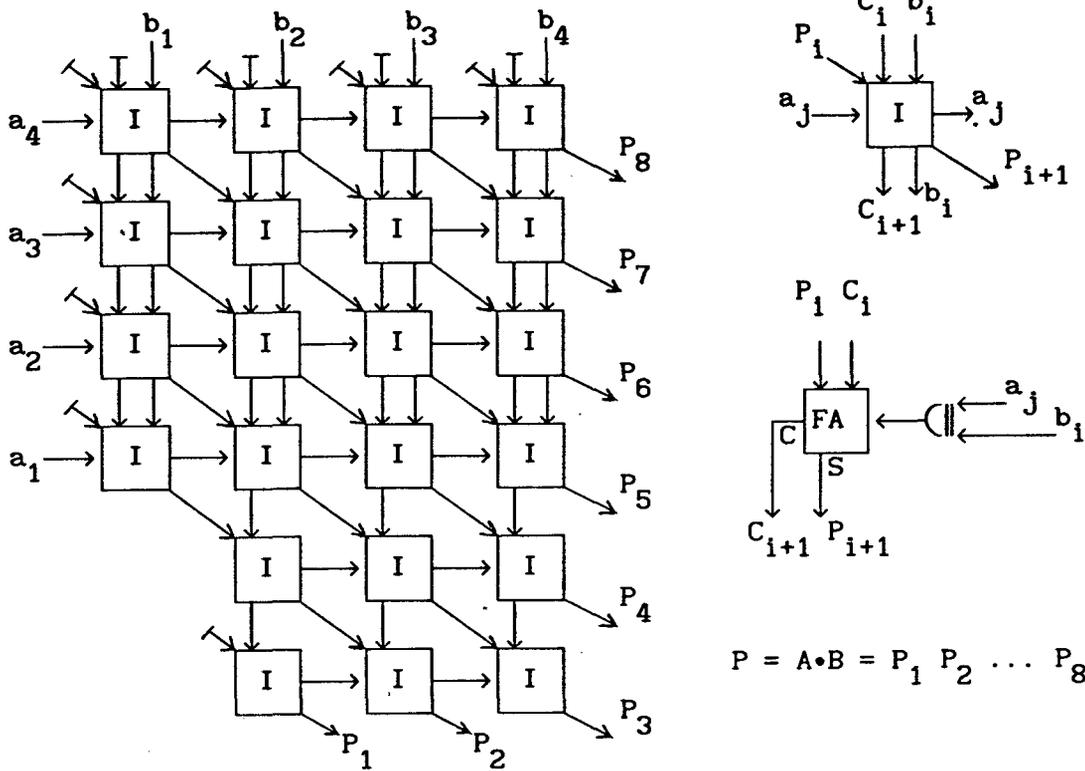


Fig. 3.2 - Multiplicador sistólico para $n = 4$.

Os tempos de propagação e as área requeridas pelas unidades processadoras constituintes das novas estruturas são determinados abaixo:

i) Multiplicador de duas palavras de n bits:

$$T_{\text{mul}(n)} = (2n+1) \cdot T_{\text{FA}} + T_E, \text{ ver referência [17].}$$

$$T_{\text{FA}} \cong 3 \cdot T_E, \text{ ver referência [7].}$$

$$T_{\text{mul}(n)} \cong (6n + 4) \cdot T_E \tag{3.1}$$

$$A_{\text{mul}(n)} = (3n^2 - n) \cdot (A_{\text{bloco I}}) / 2$$

$$A_{\text{bloco I}} = A_{\text{FA}} + A_{\text{E}}$$

$$A_{\text{FA}} \cong 4 \cdot A_{\text{E}}, \text{ ver refer\u00eancia [7].}$$

$$A_{\text{mul}(n)} \cong (7,5 \cdot n^2 - 2,5 \cdot n^2) \cdot A_{\text{E}} \quad (3.2)$$

ii) Somador de duas palavras de n bits:

$$T_{\text{som}(n)} = n \cdot T_{\text{FA}} \cong 3n \cdot T_{\text{E}} \quad (3.3)$$

$$A_{\text{som}(n)} = n \cdot A_{\text{FA}} \cong 4n \cdot A_{\text{E}} \quad (3.4)$$

iii) Registrador de uma palavra de n bits:

$$T_{\text{reg}(n)} \cong T_{\text{E}}, \text{ ver refer\u00eancia [7].} \quad (3.5)$$

$$A_{\text{reg}(n)} \cong n \cdot A_{\text{E}}, \text{ ver refer\u00eancia [7].} \quad (3.6)$$

Baseando-se nas \u00e1reas e nos tempos de propaga\u00e7\u00e3o das unidades processadoras, ser\u00e3o calculados os par\u00e2metros das c\u00e9lulas tipos I, II e III existentes nas estruturas propostas. Os c\u00e1lculos ser\u00e3o parametrizados para n e k , mantendo-se a rela\u00e7\u00e3o n/k inteira e diferente de 1.

3.3 - Par\u00e2metros das C\u00e9lulas das Estruturas Otimizadas

A c\u00e9lula tipo I da fam\u00edlia das estruturas otimizadas (Fig. 2.7 e Fig. 2.11), consiste de um multiplicador de n/k bits, 2 registradores de $2n/k$ bits e 3 somadores de n/k bits. O tempo de propaga\u00e7\u00e3o do multiplicador ser\u00e1 adicionado ao tempo de acomoda\u00e7\u00e3o do registrador colocado em sua sa\u00edda,

resultando no tempo de propagação da unidade multiplicadora. Assim,

$$\begin{aligned}
 T_{\text{unimul}} &= T_{\text{mul}(n/k)} + T_{\text{reg}(2n/k)} \\
 T_{\text{mul}(n/k)} &\cong (6 \cdot n/k + 4) \cdot T_E \\
 T_{\text{reg}(2n/k)} &\cong T_E \\
 T_{\text{unimul}} &\cong (6 \cdot n/k + 5) \cdot T_E
 \end{aligned} \tag{3.7}$$

O tempo de propagação da unidade somadora de 4 entradas de n/k bits é dado pelo atraso dos somadores adicionado ao tempo de acomodação do registrador de $2n/k$ bits:

$$\begin{aligned}
 T_{\text{unisol}} &= 3 \cdot T_{\text{som}(n/k)} + T_{\text{reg}(2n/k)} \\
 T_{\text{som}(n/k)} &\cong 3 \cdot (n/k) \cdot T_E \\
 T_{\text{unisol}} &\cong (9n/k + 1) \cdot T_E
 \end{aligned} \tag{3.8}$$

A área requerida pela célula tipo I da estrutura otimizada é apresentada a seguir:

$$\begin{aligned}
 A_{\text{cell I}} &= A_{\text{mul}(n/k)} + 2 \cdot A_{\text{reg}(2n/k)} + 3 \cdot A_{\text{som}(n/k)} \\
 A_{\text{cell I}} &\cong \left[(7,5 \cdot (n/k)^2 - 2,5 \cdot n/k) + 2 \cdot (2 \cdot n/k) + 3 \cdot (4 \cdot n/k) \right] \cdot A_E \\
 A_{\text{cell I}} &\cong \left[7,5 \cdot (n/k)^2 + 13,5 \cdot n/k \right] \cdot A_E
 \end{aligned} \tag{3.9}$$

A célula tipo II das estruturas otimizadas consiste de somadores de n/k bits e um registrador de $2n/k$ bits. O tempo de propagação desta célula é:

$$T_{celIII} = T_{som(n/k)} + T_{som(n/k+1)} + T_{reg(2n/k)}$$

$$T_{celIII} \cong \left[3 \cdot n/k + 3 \cdot n/k + 3 + 1 \right] \cdot A_E$$

$$T_{celIII} \cong \left[6 \cdot n/k + 4 \right] \cdot A_E \quad (3.10)$$

A área requerida pela célula tipo II da estrutura otimizada é dada por:

$$A_{celIII} = 2 \cdot A_{som(n/k)} + A_{som(n/k+1)} + A_{reg(2n/k)}$$

$$A_{celIII} \cong \left[14 \cdot n/k + 4 \right] \cdot A_E \quad (3.11)$$

As células tipo III das estruturas otimizadas consistem de $2n/k$ portas OU-exclusivo, $2n/k$ portas "E", um acumulador de 3 entradas de n/k bits e um registrador de n/k bits. Considerando-se os parâmetros (tempo de propagação e área) da porta OU-exclusivo aproximados por : $T_{EXOU} \cong T_E$ e $A_{EXOU} \cong A_E$, obtém-se para o tempo de propagação desta célula :

$$T_{celIII} \cong T_E + T_E + 2 \cdot T_{som(n/k)} + T_{reg(2n/k)}$$

$$T_{celIII} \cong \left[6 \cdot n/k + 3 \right] \cdot T_E \quad (3.12)$$

e, para a área:

$$A_{celIII} = 2n/k \cdot \left[A_{EXOU} + A_E \right] + A_{som(n/k)} + A_{som(n/k+1)} + A_{reg(2n/k)}$$

$$A_{cellIII} \cong \left[14n/k + 1 \right] \cdot A_E \quad (3.13)$$

3.4 - Parâmetros das Células das Estruturas não Otimizadas

No caso das células tipo I das estruturas não otimizadas, pode-se calcular os tempos de propagação das unidades somadoras e multiplicadoras aplicando-se o mesmo procedimento utilizado na determinação dos parâmetros das estruturas otimizadas. Assim:

$$T_{unimul} \cong (6 \cdot n/k + 5) \cdot T_E \quad (3.14)$$

$$T_{unisol} = T_{som(n/k)} + T_{som(n/k+1)} + T_{reg(2n/k)} \cong (6n/k + 4) \cdot T_E \quad (3.15)$$

A área requerida pela célula tipo I das estruturas não otimizadas, é dada por:

$$A_{cellI} = A_{mul(n/k)} + 2 \cdot A_{reg(2n/k)} + 2 \cdot A_{som(n/k)} + A_{som(n/k+1)}$$

$$A_{cellI} \cong \left[7,5 \cdot (n/k)^2 + 13,5 \cdot n/k + 4 \right] \cdot A_E \quad (3.16)$$

De forma semelhante ao que foi efetuado na seção anterior, pode-se obter o tempo de propagação da célula tipo II das estruturas não otimizadas:

$$T_{cellII} = 2 \cdot T_{som(n/k)} + T_{reg(2n/k)}$$

$$T_{cellII} \cong \left[6n/k + 1 \right] \cdot A_E \quad (3.17)$$

A área requerida por esta célula tipo II é dada por:

$$A_{celII} = 2 \cdot A_{som(n/k)} + A_{reg(2n/k)}$$

$$A_{celII} \cong \left[14 \cdot n/k \right] \cdot A_E \quad (3.18)$$

Com os parâmetros das células constituintes das estruturas otimizadas e não otimizadas devidamente determinados, pode-se obter as áreas totais e as taxas de entrada e saída para estas estruturas.

3.5 - Taxa de Entrada / Saída de Dados e Área Total das Estruturas

O parâmetro $Taxa_{E/S}$ das estruturas pode ser facilmente calculado lembrando-se que as unidades somadoras e multiplicadoras operam em "pipelining". Assim, a $Taxa_{E/S}$ da estrutura será limitada pela unidade com maior tempo de propagação. A área total da estrutura será obtida através do somatório das áreas das diversas unidades que a compõem. Estes parâmetros são calculados para cada família de estruturas apresentada.

i) Estrutura não otimizada para números unipolares:

$$Taxa_{E/S} = 1/T_{unimul} \cong \left[(8n/k + 5) \cdot T_E \right]^{-1} \quad (3.19)$$

$$Área Total = k^2 \cdot A_{celI} + \left[k^2/2 + k/2 \right] \cdot A_{celII} + \left[2k + 1 \right] \cdot A_{celIII}$$

$$\cong \left[k^2 \cdot \left[7,5 \cdot (n/k)^2 + 13,5 \cdot n/k + 4 \right] + \left[k^2/2 + 5 \cdot k/2 + 1 \right] \cdot \left[14 \cdot n/k \right] \right] \cdot A_E$$

$$= \left[7,5 \cdot n^2 + 20,5 \cdot n \cdot k + 4 \cdot k^2 + 35 \cdot n + 14 \cdot n/k \right] \cdot A_E \quad (3.20)$$

ii) Estrutura não otimizada para números bipolares:

$$\text{Taxa}_{E/S} = 1/T_{\text{unimul}} \cong \left[(8n/k + 5) \cdot T_E \right]^{-1} \quad (3.21)$$

$$\text{Área Total} = \left[\left[(2k+1)^2/2 + (2k+1)/2 \right] \cdot A_{\text{celI}} + (2k+1) \cdot A_{\text{celII}} \right]$$

$$\cong \left[30 \cdot n^2 + 54 \cdot n \cdot k + 16 \cdot k^2 + 37,5 \cdot n^2/k + 95,5 \cdot n + 20 \cdot k + \right. \\ \left. + 11,25 \cdot n^2/k^2 + 27,5 \cdot n/k + 6 \right] \cdot A_E \quad (3.22)$$

iii) Estrutura otimizada para números unipolares:

$$\text{Taxa}_{E/S} = 1/T_{\text{unisol}} = \left[(9 \cdot n/k + 1) \cdot T_E \right]^{-1} \quad (3.23)$$

$$\text{Área Total} = \left[k^2 \cdot A_{\text{celI}} + 2k \cdot A_{\text{celII}} \right] \cong$$

$$\cong \left[7,5 \cdot n^2 + 13,5 \cdot nk + 28 \cdot n + 8 \cdot k \right] \cdot A_E \quad (3.24)$$

iv) Estrutura otimizada para números bipolares:

$$\text{Taxa}_{E/s} = 1/T_{\text{unisolm}} = \left[(9 \cdot n/k + 1) \cdot T_E \right]^{-1} \quad (3.25)$$

$$\begin{aligned} \text{Área Total} &= \left[k^2 \cdot A_{\text{celI}} + (2k+1) \cdot A_{\text{celII}} + (k+1) \cdot A_{\text{celIII}} \right] \cong \\ &\cong \left[7,5 \cdot n^2 + 13,5 \cdot nk + 42 \cdot n + 9 \cdot k + 28 \cdot n/k + 5 \right] \cdot A_E \end{aligned}$$

É importante notar que os parâmetros acima foram obtidos utilizando-se os parâmetros de unidades multiplicadoras, somadoras e registradoras específicas. A utilização de unidades com outros parâmetros implicará na alteração dos resultados obtidos acima. Por exemplo, utilizando-se unidades com tempos de propagação menor, resultará em taxas de entrada e saída maiores nas estruturas. Este raciocínio também é válido na determinação das áreas requeridas pelas estruturas, pois unidades com menores áreas resultarão em estruturas mais densas.

3.6 - Comparações e Valores Numéricos de Desempenho

Com base nos parâmetros calculados, as novas estruturas propostas serão comparadas com algumas estruturas apresentadas na literatura. Conforme mencionado anteriormente, para as comparações quantitativas serão utilizadas as unidades de área e de tempo de propagação de uma porta "E", A_E e T_E , de forma a padronizar as comparações. As estruturas apresentadas na literatura utilizam-se de diferentes estratégias para a realização da operação produto interno, assim suas características mais relevantes serão descritas brevemente. Na Tabela 3.1 estão mostrados os parâmetros destas estruturas, cujas características são descritas como segue:

i) A estrutura multiplicadora-acumuladora (MAC) básica:

Esta estrutura descrita no Capítulo 2, utiliza-se de uma unidade multiplicadora e de uma unidade somadora operando em "pipelining". Desta forma, a taxa de entrada e saída de dados é inversamente proporcional ao tempo de propagação da unidade mais lenta da estrutura. Apesar de não ser classificada como estrutura sistólica, apresenta maior velocidade do que as estruturas constituídas de unidades multiplicadoras e somadoras operando sequencialmente. Como na grande maioria das implementações, os multiplicadores de n bits são mais lentos do que os acumuladores de n bits, a Taxa_{E/S} dos MAC's é normalmente limitada pelo tempo de propagação do multiplicador. Desta forma serão comparados duas estruturas MAC's que utilizam diferentes multiplicadores resultando em diferentes fatores de desempenho de velocidade e de área:

- Multiplicador sistólico proposto por E.L. Braun [2,17]. Este multiplicador, o qual foi utilizado nas estruturas desenvolvidas no Capítulo 2, destina-se à multiplicação de números unipolares, resultando em um MAC que opera somente com seqüências unipolares.

- Multiplicador sistólico Baugh-Wooley apresentado em [2]. Este multiplicador opera com números representados em complemento de 2, realizando a multiplicação de números bipolares. Este multiplicador permite a realização de um MAC que opera com seqüências bipolares.

ii) Processador da operação produto interno :

Neste processador, apresentado em [7], as operações de acumulação e de multiplicação são realizadas por um único conjunto de circuitos, não existindo

um circuito especializado para realizar a multiplicação e outro para realizar a acumulação. Esta estratégia apresenta melhor desempenho em termos de velocidade e de área requerida em relação à estratégia utilizada nos MAC's apresentados anteriormente.

Na Tabela 3.1 são apresentados os principais parâmetros das estruturas anteriormente mencionadas e das desenvolvidas no Capítulo 2. Na Tabela 3.2 são apresentados os valores dos parâmetros das estruturas para $n=64$. Destas tabelas pode-se inferir a respeito do desempenho de velocidade, flexibilidade de projeto, complexidade das estruturas, etc. Verifica-se, por exemplo, que as estruturas propostas neste trabalho apresentam uma maior Taxa_{E/S} do que as outras estruturas. Além disso, a Taxa_{E/S} das novas estruturas sistólicas é proporcional ao nível de partição k das palavras. Quanto maior a partição das palavras maior será a Taxa_{E/S} da estrutura. Naturalmente, deve-se pagar um preço por este aumento de velocidade nas estruturas e, conseqüentemente, existe um acréscimo de área requerida pelas estruturas, o qual também é proporcional ao nível de partição k . Os valores de área calculados na Tabela 3.2, indicam que as estruturas deste trabalho requerem uma maior área do que as estruturas anteriores. Este acréscimo em área é entretanto compensado pela alta Taxa_{E/S} conseguida pelas novas estruturas, como pode-se verificar através de uma figura de mérito, $\delta(k) = \text{Taxa}_{E/S}(k)/\text{Área}(k)$, que leva em consideração ambos os fatores. Quanto maior o valor desta figura de mérito, melhor a relação velocidade versus custo de projeto. Observa-se através dos resultados apresentados que aumentando-se o valor de k tem-se um aumento maior de Taxa_{E/S} do que da área requerida pela estrutura. Esta característica está ilustrada na Fig. 3.3, a qual apresenta as curvas de área e Taxa_{E/S} em relação ao valor de n/k , utilizando-se os parâmetros da estrutura proposta por Braun [7]

normalizada como referência.

Tabela 3.1 - Área e Taxa_{E/S} das estruturas

A_E : Área de uma porta "E".

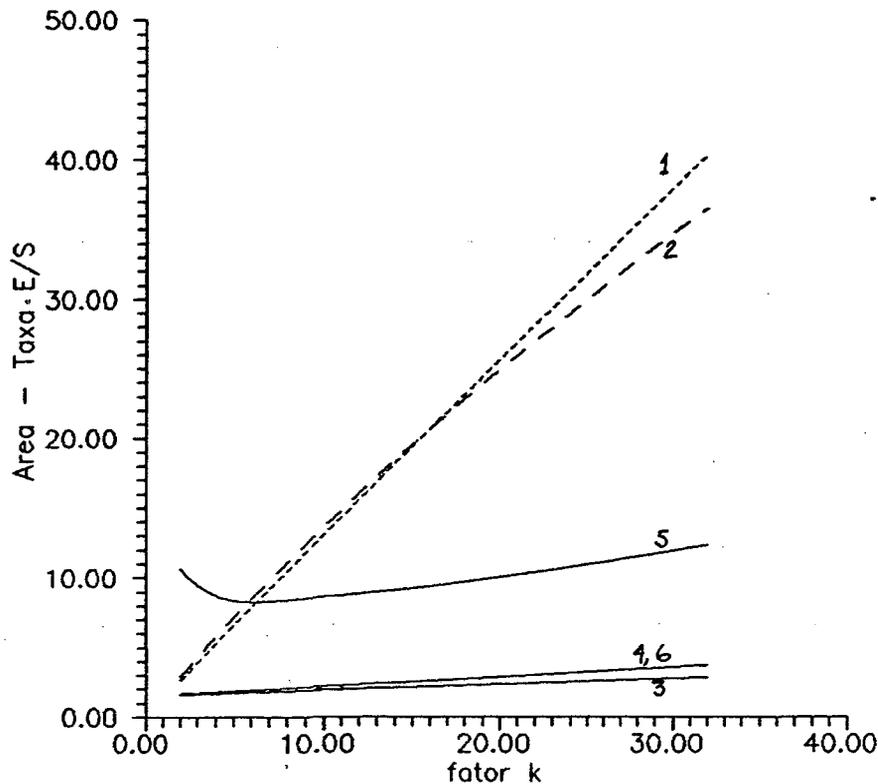
T_E : Tempo de propagação de uma porta "E".

Taxa_{E/S} : Taxa de entrada e saída de dados.

Estrutura	Área / A_E	$(\text{Taxa}_{E/S} \times T_E)^{-1}$
Braun's (Unipolar)[7]	$5n^2 + 10n$	$12n - 2$
De Mori's (Unipolar)[7]	$5n^2 + 14n$	$12n + 1$
Baugh-Wooley' (Bipolar) [7]	$5n^2 + 11n + 12$	$12n + 4,5$
Ahmad & ... (Unipolar)[7]	$5n^2 + 8n$	$6n + 3$
Ahmad & ... (Bipolar) [7]	$5n^2 + 7n + 1$	$6n + 3$
Proposta (Unipolar) Normal	$7,5 \cdot n^2 + 20,5 \cdot nk + 4 \cdot k^2 + 35 \cdot n + 14 \cdot n/k$	$8n/k + 5$
Proposta (Bipolar) Normal	$30 \cdot n^2 + 54 \cdot nk + 16 \cdot k^2 + 37,5 \cdot n^2/k + 95,5 \cdot n + 20 \cdot k + 11,25 \cdot n^2/k^2 + 27,5 \cdot n/k + 6$	$8n/k + 5$
Proposta (Unipolar) Otimizada	$7,5 \cdot n^2 + 13,5 \cdot nk + 28 \cdot n + 8 \cdot k$	$9 \cdot n/k + 1$
Proposta (Bipolar) Otimizada	$7,5 \cdot n^2 + 13,5 \cdot nk + 42 \cdot n + 9 \cdot k + 28 \cdot n/k + 5$	$9 \cdot n/k + 1$

Tabela 3.2 - Exemplos numéricos para $n = 64$ bits.

ESTRUTURAS	Área / A_E	$\delta(k) \times 10^{-9}$	$(\text{Taxa}_{E/S} \times T_E)^{-1}$	Velocidade em relação à Braun[7]
Braun (Unipolar) [7]	21120	61,81	766	1
De Mori (Unipolar) [7]	21370	60,85	769	0,996
Baugh-Wooley (Bipolar) [7]	21196	61,07	772.5	0,992
Ahmad & Poornalah [7]	20992 (Unipolar) 20929 (bipolar)	123,09 123,46	387 (Unipolar) 387 (Bipolar)	1,979 1,979
Estrutura proposta para $k = 2$	36048 (Unip, Norm) 225214 (Bipo, Norm) 34256 (Unip, Otim) 36055 (Bipo, Otim)	106,29 17,01 101,01 95,97	261 (Unip, Norm) 261 (Bipo, Norm) 289 (Unip, Otim) 289 (Bipo, Otim)	2,935 2,935 2,650 2,650
Estrutura proposta para $k = 4$	38496 (Unip, Norm) 184878 (Bipo, Norm) 36000 (Unip, Otim) 37353 (Bipo, Otim)	195,31 40,66 191,57 184,63	133 (Unip, Norm) 133 (Bipo, Norm) 145 (Unip, Otim) 145 (Bipo, Otim)	5,759 5,759 5,283 5,283
Estrutura proposta para $k = 8$	43824 (Unip, Norm) 177970 (Bipo, Norm) 39488 (Unip, Otim) 40621 (Bipo, Otim)	330,70 81,43 346,91 337,23	69 (Unip, Norm) 69 (Bipo, Norm) 73 (Unip, Otim) 73 (Bipo, Otim)	11,10 11,10 10,49 10,49
Estrutura proposta para $k = 16$	55032 (Unip, Norm) 198600 (Bipo, Norm) 46464 (Unip, Otim) 47493 (Bipo, Otim)	491,11 136,09 581,68 569,07	37 (Unip, Norm) 37 (Bipo, Norm) 37 (Unip, Otim) 37 (Bipo, Otim)	20,70 20,70 20,70 20,70
Estrutura proposta para $k = 32$	79068 (Unip, Norm) 261514 (Bipo, Norm) 60416 (Unip, Otim) 61405 (Bipo, Otim)	602,25 182,08 871,15 857,12	21 (Unip, Norm) 21 (Bipo, Norm) 19 (Unip, Otim) 19 (Bipo, Otim)	36,48 36,48 40,32 40,32



- Curva 1 - Taxa_{E/S} das estruturas unipolares.
 Curva 2 - Taxa_{E/S} das estruturas bipolares.
 Curva 3 - Área das estruturas unipolares não otimizadas.
 Curva 4 - Área das estruturas unipolares otimizadas.
 Curva 5 - Área das estruturas bipolares não otimizadas.
 Curva 6 - Área das estruturas bipolares otimizadas.

Fig. 3.3 - Taxa_{E/S} e área das estruturas normalizadas em relação à Braun [7]

Um outro fator importante relativo às novas estruturas, é a flexibilidade que o projetista possui na utilização das estruturas das famílias propostas pois, existe um variado conjunto de estruturas que apresentam diferentes figuras de mérito $\delta(k)$, fornecendo, desta forma, ao projetista a possibilidade de escolha daquela que se adapte melhor aos compromissos entre área e Taxa_{E/S} necessários à aplicação desejada. Uma outra característica interessante das novas estruturas é a possibilidade da utilização de CAD's para circuitos

integrados (por exemplo, O CAD SOLO 2000), pois, as estruturas utilizam-se de unidades processadoras disponíveis nas bibliotecas destes "softwares". Além disso, pode-se utilizar facilmente os circuitos integrados para aplicações específicas (ASIC) no projeto de estruturas complexas. As estruturas sistólicas são construídas a partir de poucos tipos de células básicas, as quais são conectadas localmente, características que auxiliam enormemente o projeto baseado em ASIC. O principal atrativo de ASIC's é o custo relativamente baixo, pois o projeto de um CI dedicado completo em VLSI ("custom") torna-se inviável para um pequeno número de unidades produzidas, uma vez que os custos fixos são muito altos, requerendo-se uma quantidade da ordem de milhares de unidades para amortizar o preço unitário de um "chip".

3.7 - Conclusões

Neste capítulo foram apresentadas figuras de comparação para avaliações de desempenho das novas estruturas em termos de área e de velocidade. As comparações entre as estruturas propostas na literatura e as estruturas desenvolvidas neste trabalho, mostraram a eficiência e a flexibilidade destas últimas. As análises de desempenho permitem afirmar que as novas estruturas representam uma interessante alternativa para a implementação da operação produto interno. No capítulo seguinte serão apresentados alguns resultados obtidos através de simulações em computador digital de uma das estruturas propostas, assim como, os resultados de implementações das células constituintes desta estrutura, utilizando-se circuitos integrados discretos comerciais e arranjos programáveis de portas lógicas.

CAPÍTULO 4

VALIDAÇÃO DAS ESTRUTURAS

4.1 - Introdução

Neste capítulo serão apresentados os resultados de simulações em computador digital de uma estrutura proposta para a realização da operação produto interno. Além disso, serão apresentadas implementações das células constituintes desta estrutura utilizando-se componentes discretos. Estas simulações e implementações visam a validar o projeto lógico das unidades processadoras e do funcionamento destas unidades operando em "pipelining". Inicialmente, serão apresentados os resultados das simulações de estruturas sistólicas realizando a operação produto interno das seqüências A_m e B_m representadas em palavras de 16 e 12 bits. Posteriormente, será analisada a implementação destas estruturas usando um sistema de processadores digitais de sinais (DSP). A implementação de uma célula tipo I, a qual utiliza multiplicadores e somadores comerciais, será apresentada, validando o projeto das unidades multiplicadora e somadora quando operando em "pipelining". Finalmente, todas as unidades constituintes da estrutura escolhida são implementadas em PGAs ("Programmable Gate Arrays") e testadas de forma a validar estas unidades.

4.2 - Simulações Via Computador Digital

Uma das grandes vantagens da utilização de sistemas digitais é a possibilidade de simulação precisa destes sistemas utilizando-se computadores

digitais. No desenvolvimento de novos sistemas digitais é, portanto, usual e aconselhável que se façam simulações via computador, detectando-se possíveis erros de projeto e permitindo assim que se façam eventuais ajustes antes da implementação do sistema. Este procedimento leva a uma redução dos custos e do tempo despendidos nas diversas etapas de um desenvolvimento, uma vez que possibilita um estudo prévio sobre a viabilidade das estruturas projetadas.

Para o estudo de viabilidade foi escolhida a estrutura apresentada na Fig. 4.1. Nesta estrutura as palavras A_m são consideradas bipolares e as palavras B_m unipolares. A utilização de representações distintas traduz o que ocorre em várias aplicações práticas de processamento de sinais, onde os sinais analógicos são convertidos em sinais digitais através de uma unidade CAD (conversor de sinais analógicos em digitais), as quais, geralmente apresentam uma saída digital unipolar (B_m). Estes dados são então processados por um sistema cujos coeficientes (A_m) podem assumir valores positivos ou negativos. Neste trabalho pretende-se aplicar esta estrutura, da forma mencionada anteriormente, em filtragem digital. Para os coeficientes bipolares (A_m) será utilizada a representação em complemento de 2. No caso dos dados B_m , estes podem ser bipolares antes da conversão A/D, porém são convertidos para unipolares adicionando-se, à entrada do conversor, um nível de tensão DC de valor previamente estipulado. Posteriormente, o valor de tensão DC correspondente é subtraído da saída do conversor D/A não alterando os resultados do processamento. Portanto, o processamento, do ponto de vista da entrada do conversor A/D e da saída do conversor D/A, realiza-se com dados e coeficientes bipolares. No entanto, do ponto de vista do sistema digital, o processamento realiza-se com dados unipolares e coeficientes bipolares.

A seqüência de dados será representada por 12 bits (os conversores de 12

bits oferecem atualmente o melhor compromisso entre custo e velocidade de conversão). A estrutura utilizará 16 bits na representação dos coeficientes bipolares, A_m , e 12 bits para representação dos dados unipolares, B_m . Esta estrutura será formada por um arranjo retangular 3 por 4 de unidades com $n/k = 4$. O algoritmo de otimização, apresentado no Capítulo 2, para seqüências representadas com um mesmo número de bits, é desenvolvido a seguir para seqüências A e B representadas com números de bits distintos.

Considere as palavras estendidas A_{me} e B_{me} representadas como segue:

$$A_{me} = W_{s1}(A_m) \dots W_{s5}(A_m) W_1(A_m) \dots W_4(A_m)$$

$$B_{me} = W_{s1}(B_m) \dots W_{s6}(B_m) W_1(B_m) \dots W_3(B_m)$$

Note que as palavras estendidas apresentam o mesmo comprimento, ou seja, $9 \cdot n/k$ bits, com $n/k = 4$ bits. O produto destas palavras apresentará o bit de sinal posicionado no 36° bit da palavra S_{me} do resultado.

Procedendo-se ao desmembramento destas palavras e, posteriormente, às devidas simplificações, tem-se:

$$S_{me} = A_{me} \cdot B_{me} = A_m \cdot B_m + A_m \cdot W_s(B_m) \cdot 2^{3n/k} + B_m \cdot W_s(A_m) \cdot 2^{4n/k} + W_s(B_m) \cdot W_s(A_m) \cdot 2^{7n/k}$$

Como a seqüência B_m é unipolar, a palavra $W_s(B_m)$ apresenta todos os bits iguais a zero. Desta forma, o resultado S_{me} é dado por:

$$S_{me} = A_m \cdot B_m, \text{ Para } A_m \text{ for positivo,}$$

caso contrário (A_m negativo), $W_s(A_m) = (2^{5n/k} - 1)$ e, desprezando-se os

bits posicionados acima da 36^{a} posição ($9n/k$) da resposta S_{me} , tem-se:

$$S_{me} = A_m \cdot B_m - B_m \cdot 2^{4n/k} = A_m \cdot B_m + (B_m)^2 \cdot 2^{4n/k}, \text{ para } A_m \text{ negativo.}$$

A estrutura da Fig. 4.1 realiza o algoritmo descrito acima, utilizando-se células de estruturas otimizadas. A célula tipo III utilizada nesta estrutura consiste de portas OU-Exclusivo e portas "E", como ilustrado na Fig. 4.2. Esta célula é uma simplificação da célula tipo III utilizada nas estruturas otimizadas das figuras. 2.7 e 2.11, apresentadas no Capítulo 2.

Sistemas de processamento constituídos por unidades processadoras operando em "pipelining", tais como as estruturas apresentadas, podem ser simulados por uma máquina SISD (ver Capítulo 1) (por exemplo, computadores tipo IBM-PC) através do processamento seqüencial por unidade, iniciando o processamento pela unidade posicionada mais próxima da saída do sistema e prosseguindo até a unidade processadora mais próxima da entrada de dados do sistema. Cada unidade é processada seqüencialmente, armazenando os resultados em registradores temporários. A cada iteração, que corresponde a um ciclo de clock na estrutura, todas as unidades são simuladas seqüencialmente pela máquina SISD. Nestas simulações foram geradas seqüências A_m e B_m representadas em 16 e 12 bits, respectivamente, e os resultados da simulação foram comparados com o resultado teórico esperado. Os resultados obtidos pela simulação da estrutura acima, validam o algoritmo e a viabilidade da estrutura para a realização da operação produto interno. A listagem do programa de simulação, escrito em linguagem de programação Pascal, encontra-se no Apêndice A.

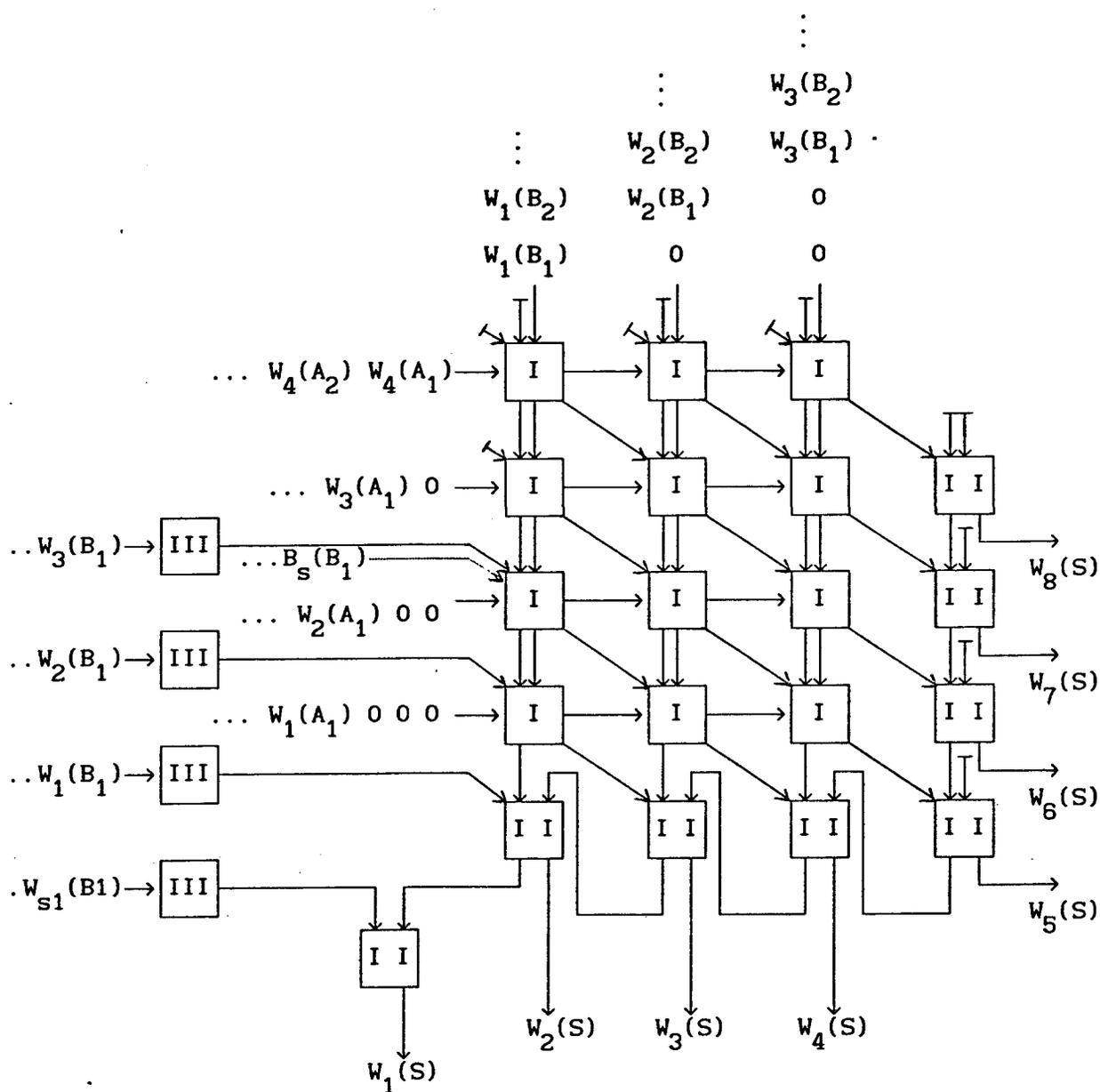
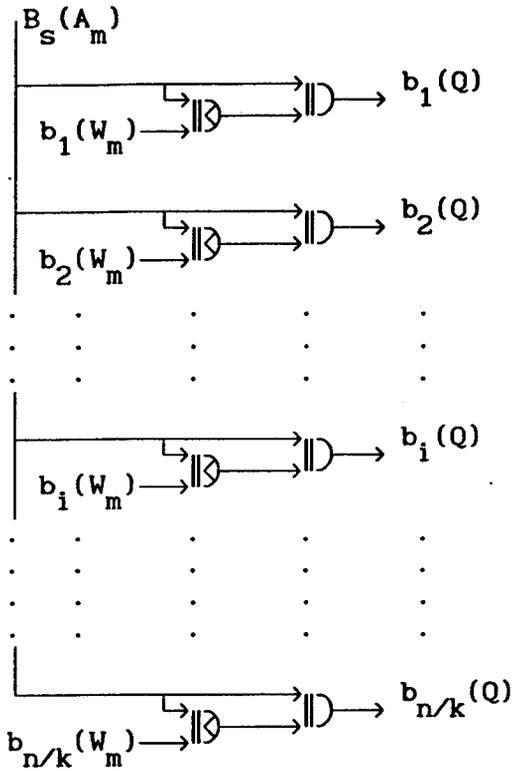


Fig. 4.1 - Estrutura sistólica com $n/k = 4$ bits.

Célula Tipo III :



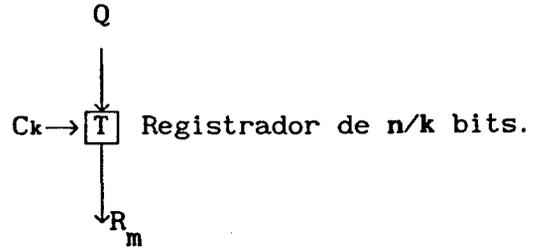
Onde:

$\rightarrow \oplus \rightarrow$ = Porta OU-Exclusivo.

$\rightarrow \parallel \rightarrow$ = Porta "E".

$B_s(A_m)$ = Bit de sinal da palavra A_m .

$b_i(W_m)$ = i-ésimo bit da palavra W_m .



$Q = \bar{W}_m$ se $A_m < 0$.

$Q = 0$ se $A_m \geq 0$.

\bar{W}_m = Complemento de 1 de W_m .

Fig. 4.2

Célula tipo III.

4.3 - Implementação via Arranjo Sistólico de Processadores

Neste tópico apresenta-se os resultados de um estudo de viabilidade da implementação da estrutura proposta via arranjo de PDS (Processadores Digitais de Sinais) MB8764 fabricados pela empresa Fujitsu. O sistema de desenvolvimento para tais arranjos foi desenvolvido no Laboratório Central de Eletrônica (ZEL) do Centro de Pesquisas de Jülich (KFA), na Alemanha. O arranjo consiste de processadores conectados entre si, operando em paralelo e utilizando o mesmo conjunto de instruções em cada processador. As conexões entre estes processadores são realizadas por portas bidirecionais, as quais podem ser configuradas pelo usuário do sistema. Os programas são desenvolvidos e compilados em um computador do tipo IBM-PC/XT/AT e podem ser executados diretamente nos processadores através de um emulador desenvolvido para este fim, o qual permite emular até 8 processadores operando simultaneamente. De acordo com a aplicação, o usuário configura tanto a estrutura quanto o programa a ser executado pelos processadores. Assim, o sistema presta-se à implementação eficaz de diversas aplicações de PDS (uni e multi dimensionais).

Entretanto, para a implementação das estruturas propostas neste trabalho, a utilização de um arranjo de DSPs comerciais não representa a solução mais adequada. Primeiramente, as estruturas propostas representam um arranjo de muitos processadores com funções extremamente simples. Esta filosofia é claramente incompatível com a utilização de DSPs, cujas arquiteturas apresentam um grau de flexibilidade e, conseqüentemente, de complexidade desnecessário para esta aplicação. Em segundo lugar, as novas estruturas requerem um número de interconexões entre células maior do que aquele

normalmente oferecido pelos DSPs comerciais.

Desta forma, verificada a inadequação dos arranjos utilizando-se DSPs para a implementação das estruturas propostas neste trabalho, abandonou-se esta estratégia de projeto.

4.4 - Implementação via Multiplicadores e Somadores Comerciais

Visando à implementação com células mais elementares, foram utilizados um multiplicador e somadores comerciais para a realização da célula tipo I da estrutura escolhida. O objetivo desta implementação consiste na avaliação da operação em "pipelining" das unidades somadoras e multiplicadora, além da necessidade de comprovação de que a Taxa_{E/S} desta célula é limitada pela unidade mais lenta operando em "pipelining". Uma vez validada a operação em "pipelining", pode-se estender o resultado para toda a estrutura, pois esta é composta por unidades operando em "pipelining", de forma que a Taxa_{E/S} da estrutura completa será também limitada pela unidade mais lenta.

A célula implementada é apresentada na Fig. 4.3, a qual opera com palavras de 4 bits, uma vez que a estrutura escolhida utiliza células operando com palavras de $n/k = 4$ bits. A unidade multiplicadora desta célula foi implementada utilizando-se um multiplicador ADSP1016 da Analog Devices, o qual apresenta um registrador de saída ("latch"), controlado por sinal externo. Este multiplicador apresenta um tempo de propagação máximo de 130 ns ("delay time"). Além disso, realiza a operação de multiplicação de palavras de 16 bits armazenando o resultado em um registrador de 32 bits. Contudo, esta unidade foi utilizada para a multiplicação de palavras de 4 bits resultando em uma palavra de 8 bits. Esta configuração é obtida através do desativamento das entradas não utilizadas do multiplicador.

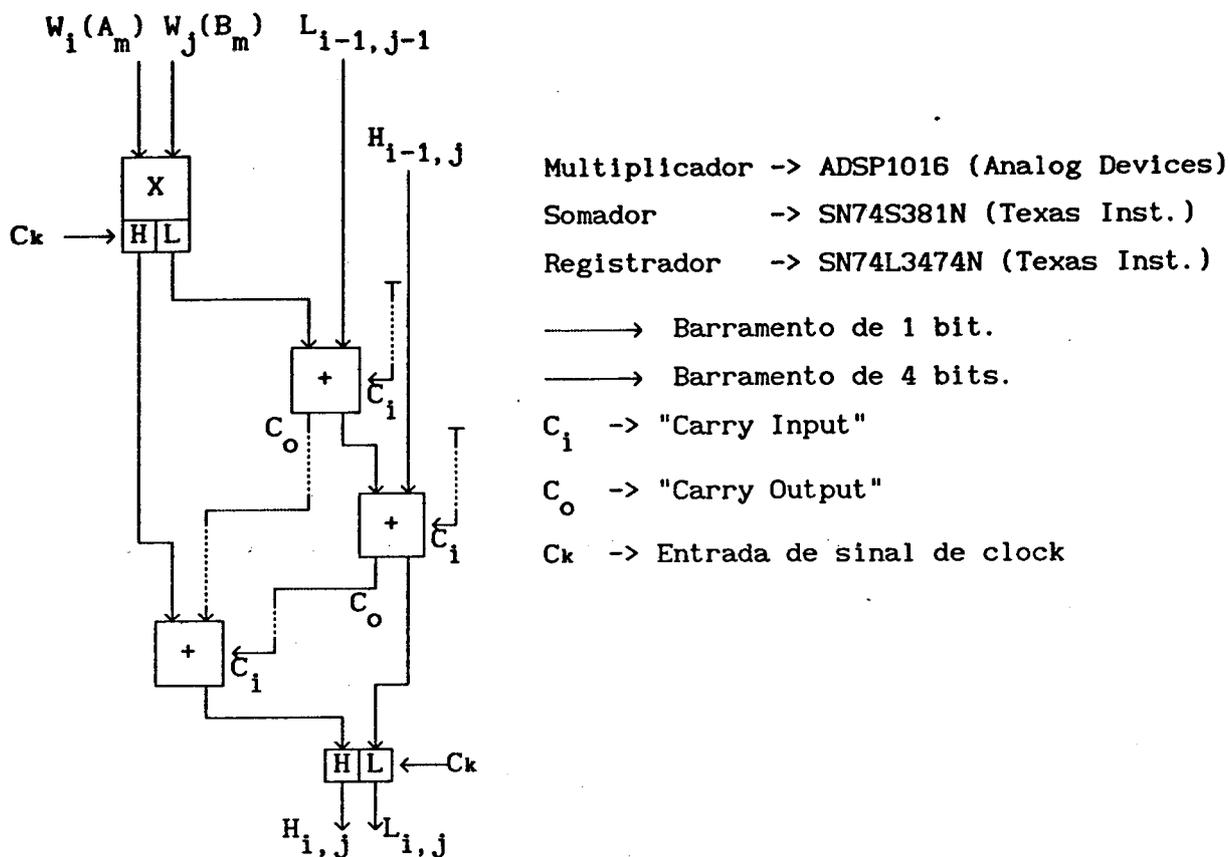


Fig. 4.3 - Implementação da célula tipo I utilizando CIs comerciais.

A unidade somadora foi implementada com ULAs (Unidade Aritmética e Lógica) SN74S381N fabricada pela Texas Instruments e com registradores temporários ("latch") SN74L374N do mesmo fabricante. Estas ULAs podem realizar a acumulação de duas palavras de 4 bits, além de outras operações lógicas e aritméticas. Esta unidade apresenta um tempo máximo de propagação de 26ns para a operação de acumulação de duas palavras de 4 bits.

As unidades são conectadas como mostra a Fig. 4.3, implementando a célula tipo I, através da utilização de um multiplicador de palavras de 4 bits, 3 ULAs e um registrador temporário de 8 bits.

Esta implementação foi testada utilizando-se um analisador lógico de

sinais, variando-se a taxa de entrada de dados com um gerador de clock de frequência variável, circuitos integrados SN74LS624N, e chaves do tipo "dip-switch". A máxima frequência de operação admissível pela estrutura foi de $1/125 \cdot 10^9$ Hz. Esta frequência aproxima-se da máxima frequência de operação do multiplicador utilizado ($F_{\text{máx}} = 1/130 \cdot 10^9$ Hz). Com base nestes resultados pode-se afirmar que a taxa de entrada e saída desta célula é limitada pelo tempo de propagação da unidade mais lenta, ou seja da unidade multiplicadora.

A implementação da estrutura completa com células processadoras construídas a partir dos componentes discretos mencionados acima apresentaria, entretanto, problemas de ordem prática. Esta implementação necessitaria uma quantidade de CIs muito grande (4 CIs por célula tipo I), resultando em um demasiado consumo de potência, além de uma excessiva área e grande número de conexões a serem alocadas em uma placa de circuito impresso.

4.5 - Implementação via Arranjo Programável de Portas Lógicas (PGA)

Como a estrutura proposta requer uma alta densidade de circuitos, tornam-se inadequadas as implementações por circuitos integrados comerciais de propósito geral. Uma alternativa atraente seria a implementação através de ASICs, ou seja, por meio de circuitos integrados desenvolvidos para aplicações específicas. As opções de ASICs atualmente oferecidas pelos fabricantes são as seguintes [18]:

1) Células padrão : Esta alternativa requer um conjunto padrão de máscaras para a maioria das camadas usadas na fabricação dos CIs. No entanto, devido à necessidade de fabricação total para cada aplicação específica, impõe custos extras e atrasos no desenvolvimento do projeto. Esta alternativa é

especialmente adequada para aplicações de alta complexidade (VLSI) e para grande número de unidades produzidas.

ii) Dispositivos lógicos programáveis ("Programmable Logic Devices"): São especialmente adequados para aplicações de baixa complexidade (SSI, MSI). Estes dispositivos são constituídos de uma estrutura planar de portas "E" e OU-Exclusivo, configurável pelo usuário. Apresentam limitações com respeito à reduzida quantidade de "flip-flops" e ao número de entradas e saídas disponíveis. Além disso, apresentam pouca flexibilidade na configuração da arquitetura e nas interconexões das portas lógicas, de forma que a utilização de uma determinada função implica na limitação de outras funções similares existentes no mesmo circuito.

iii) Arranjos de portas lógicas ("Gate Arrays"): Estes circuitos são configurados na etapa final do processo de fabricação, através da definição das máscaras de interconexão das portas lógicas pelo usuário. Estes circuitos destinam-se às aplicações de média a alta complexidade (LSI, VLSI), apresentam custos fixos por unidade e são especialmente adequados às aplicações que demandam um grande volume de produção.

iv) Arranjos programáveis de portas lógicas ("Programmable Gate Array): Estes arranjos podem ser configurados pelo usuário através da programação de uma memória PROM ("Programmable Read Only Memory"). Diferentemente dos "Gate Arrays", os quais são configurados na etapa final de fabricação do CI, estes PGAs não apresentam custos fixos.

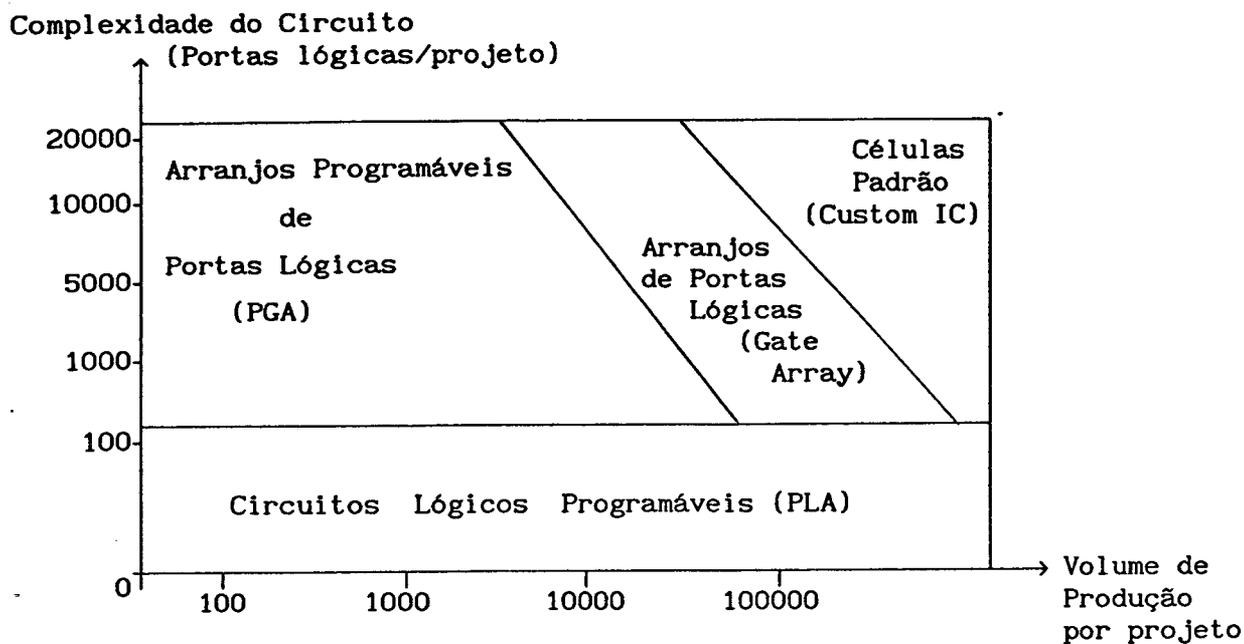


Fig. 4.4 - Quadro de alternativas ASICS em função da complexidade e volume de produção por projeto

A Fig. 4.4 apresenta um panorama da aplicabilidade de cada uma das alternativas de implementação em função do volume de produção e da complexidade do circuito. Nota-se que cada alternativa destina-se a uma faixa de aplicações conforme a complexidade requerida e números de unidades a serem produzidas.

Em função das características específicas do tipo de arquitetura que se pretende realizar (alta complexidade e reduzido volume), optou-se por implementações utilizando circuitos PGAs XC2064-100 fabricado pela empresa XILINX, os quais apresentam as seguintes características:

- a) Alta densidade de circuitos:

1200 portas lógicas por CI, 64 blocos lógicos que implementam qualquer função booleana com 4 entradas e 1 saída ou duas funções booleanas de 2 entradas e 2 saídas. Estas saídas podem ser conectadas a um "flip-flop" ou "latch" existente em cada bloco, permitindo realizar circuitos combinacionais e sequenciais. Este circuito apresenta blocos de entrada/saída configuráveis como três estados ("tri-state"), registrador temporário ("latch"), ou diretamente conectado ao pino de conexão externa ("pad").

b) Flexibilidade de interconexões entre blocos lógicos e de entrada saída:

Este circuito permite conexões entre quaisquer blocos através de chaves programáveis e linhas de conexão internas. Para auxiliar a programação do circuito é utilizado um programa de CAD que gera automaticamente o arquivo de dados a ser carregado na memória PROM, a qual armazenará a configuração do circuito. O carregamento da configuração nos circuitos PGA é totalmente realizada pelos circuitos internos do PGA, através da geração interna de endereços e sinais de controle durante a etapa de inicialização, a qual despende apenas alguns milisegundos, após conectar-se o pino de alimentação do circuito à fonte de tensão contínua de 5 volts.

As células da estrutura escolhida foram agregadas em 4 grupos diferentes e implementadas utilizando-se os circuitos PGAs. A Fig. 4.5 apresenta esta distribuição em grupos das células da estrutura. No Apêndice B encontram-se o diagrama esquemático da implementação de cada grupo de células e os diagramas fornecidos pelo programa de CAD onde são mostradas as interconexões entre os blocos e a configuração de cada bloco dentro do PGA.

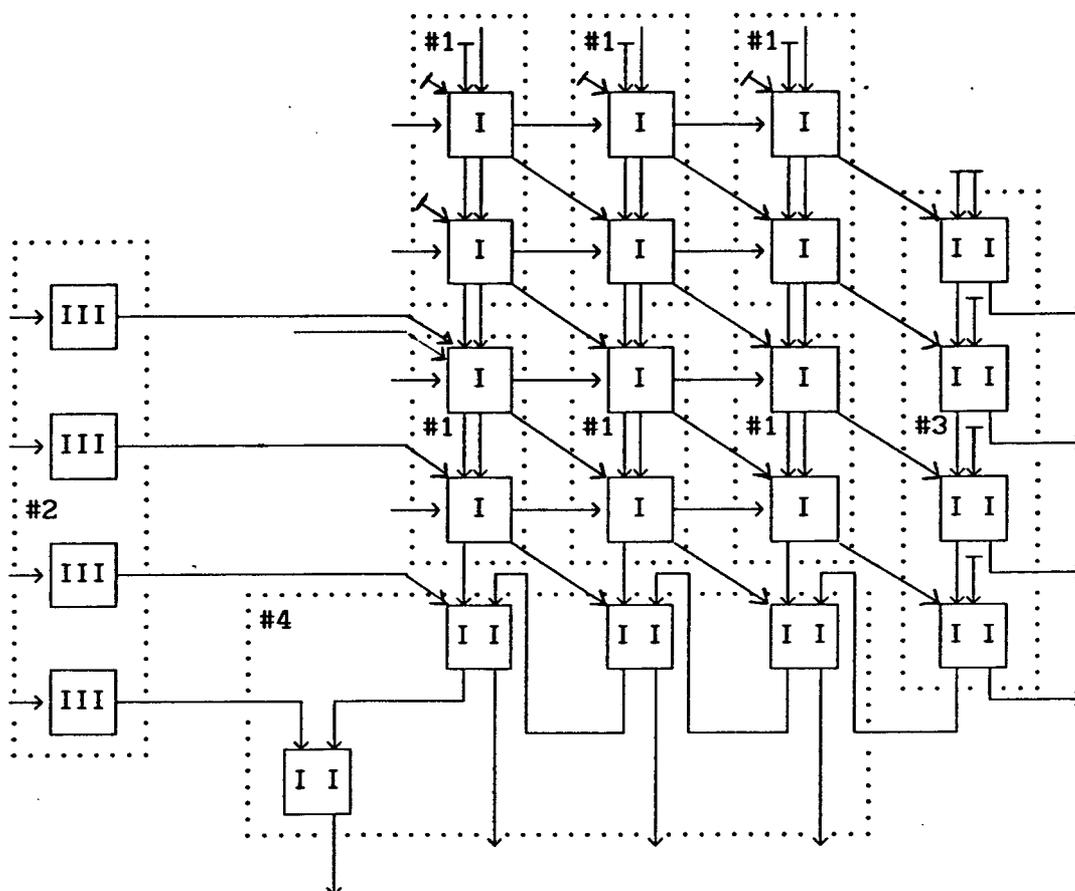


Fig. 4.5 - Distribuição das células processadoras em circuitos PGA.

- 6 unidades PGA implementando agrupamento #1 (2 células tipo I).
- 1 unidade PGA implementando agrupamento #2 (4 células tipo III).
- 1 unidade PGA implementando agrupamento #3 (4 células tipo II).
- 1 unidade PGA implementando agrupamento #4 (4 células tipo II).

Cada grupo de célula implementado por PGA (circuito XC 2064-100 MHz) foi testado isoladamente em bancada através da utilização de um circuito apresentado no Apêndice C. Para cada grupo de células implementado foi utilizada uma memória EPROM, a qual contém a configuração da célula.

O programa de CAD fornece os tempos de propagação máximos demandados pelos circuitos PGA. O tempo de propagação máximo de uma unidade operando em

"pipelining" nesta implementação resultou em 105 ns. Esta unidade foi a multiplicadora, a qual consiste de um multiplicador de 4•4 bits com registrador temporário ("latch") na saída. Desta forma, implementando-se toda a estrutura através das unidades realizadas por circuitos PGA, resultou em uma Taxa_{E/s} aproximadamente igual a 10MHz. Este desempenho é considerado baixo, pois, utilizando-se multiplicadores comerciais de 16•16 bits, tais como os apresentados anteriormente, consegue-se uma Taxa_{E/s} de aproximadamente 7,7MHz. Baseando-se nestas informações pode-se concluir que as implementações utilizando estes circuitos, apesar de realizarem inúmeras aplicações com bom desempenho, apresentam limitações de velocidade devido às suas generalidade e flexibilidade.

4.6 - Conclusões

Neste capítulo foram apresentadas simulações e implementações de uma das estruturas propostas neste trabalho, a qual encontra aplicabilidade em sistemas práticos. Os resultados obtidos permitem concluir, basicamente, que as estruturas propostas realizam adequadamente a operação produto interno, apresentando uma Taxa_{E/s} (velocidade) limitada pela unidade mais lenta da estrutura.

Apesar da possibilidade de implementação das estruturas através de circuitos integrados comerciais (Multiplicadores e ULAs) ou através de circuitos PGAs, vários fatores levam a concluir que estas formas de implementação não são as mais adequadas para este caso. A alternativa que exploraria ao máximo a potencialidade destas estruturas seria a implementação em VLSI. As características das estruturas sistólicas propostas (modularidade, regularidade e conexões locais) satisfazem perfeitamente os requisitos das

implementações em VLSI, resultando em alta eficácia na realização da operação produto interno.

Uma desvantagem desta alternativa é o alto custo e o tempo de fabricação requerido atualmente. Apesar destes fatores, existe atualmente um projeto de integração de circuitos denominado PMU (Projeto Multi-Usuário), o qual permite a inclusão de vários projetos em uma mesma "bolacha" ("wafer") no processo de fabricação. Esta estratégia reduz o custo total do processo por usuário, sendo atualmente disponível para as Universidades Brasileiras que pretendem implementar CIs.

CAPÍTULO 5

CONCLUSÕES

Neste trabalho foram desenvolvidas estruturas sistólicas destinadas à realização da operação produto interno (PI), a qual está inserida em inúmeros algoritmos de processamento digital de sinais. As arquiteturas sistólicas apresentam características interessantes, tais como: modularidade, regularidade e conexões locais, as quais são especialmente adequadas às implementações em VLSI. Desta forma, foram desenvolvidas estruturas que exploram a potencialidade das arquiteturas sistólicas e apresentam alto desempenho na realização da operação PI. As estruturas propostas foram classificadas de acordo com suas características arquitetônicas e com a capacidade de operarem com números bipolares ou unipolares. As estruturas apresentam os parâmetros de desempenho, Taxa_{E/S} e área, proporcionais ao nível k de partição das palavras. Esta característica fornece uma grande flexibilidade ao projetista, pois existe a possibilidade de escolha, dentre as várias estruturas, de uma que apresente uma relação de área (custo) e Taxa_{E/S} (velocidade) mais adequada à aplicação em questão. Neste trabalho foram apresentadas expressões matemáticas, tabelas e gráficos que auxiliam o projetista na escolha da estrutura mais adequada.

Os resultados obtidos das análises de desempenho e das comparações com outras estruturas permitem concluir que, apesar de um preço a ser pago em área nas implementações das estruturas propostas, o desempenho conseguido, proporcional ao nível de partição k , e as características estruturais das arquiteturas sistólicas, compensam sobremaneira a utilização destas estruturas em implementações que requeiram alto desempenho, como processamento em tempo

real e processamento massivo de dados. Estas estruturas podem ser eficazmente utilizadas em aplicações que envolvam filtragem digital (FIR e IIR), multiplicação de matrizes, correlação de seqüências, e em várias outras aplicações que utilizam a operação produto interno.

Apesar do empenho e aprofundamento realizado no estudo e desenvolvimento das estruturas aqui propostas, este trabalho apresenta múltiplas facetas que podem ser ainda trabalhadas, trazendo, dessa forma, novas contribuições científicas. Algumas dessas possíveis continuações são citadas a seguir:

a) Desenvolvimento do "layout" de algumas das estruturas apresentadas visando-se à implementação em VLSI. Esta continuação é perfeitamente viável, pois atualmente encontram-se disponíveis no LINSE/UFSC tanto os recursos humanos e materiais (programas CAD, estações de trabalho) necessários, como a possibilidade de utilização do PMU para a implementação de circuitos com alta complexidade e densidade. Esta continuação resultaria na integração de um "chip" que realizaria a operação produto interno com altíssimo desempenho, o qual poderia ser utilizado em uma das possíveis aplicações citadas neste trabalho.

b) Estudo para o aprofundamento da filosofia empregada na implementação do algoritmo da operação produto interno, visando-se à implementação de estruturas mais complexas como, por exemplo, de filtragem FIR. Estas estruturas seriam baseadas em arranjos de células operando sobre palavras de n/k bits. Esta continuação resultaria em estruturas dedicadas ao algoritmo escolhido, aumentando ainda mais o desempenho do processamento.

A P Ê N D I C E A

PROGRAMA DE SIMULAÇÃO DA ESTRUTURA

Neste Apêndice será listado um programa de simulação da estrutura apresentada no Capítulo 4, o qual pretende validar a funcionalidade da estrutura. Este programa foi desenvolvido em linguagem Pascal (Versão 6.0), e basicamente realiza as seguintes funções:

i) Gera os valores para as seqüências A_m e B_m a partir de um valor fornecido pelo usuário.

ii) Simula o processamento realizado pela estrutura através da simulação das células desta estrutura.

iii) Verifica se os resultados obtidos pela simulação das células e da estrutura estão de acordo com o resultado teórico esperado, caso contrário, emite uma mensagem de erro para o usuário do programa.

Listagem do Programa

A listagem do programa em linguagem Pascal, a seguir, foi retirada diretamente do ambiente de programação e depuração Pascal 6.0 após validada a estrutura através desta simulação.

Program Simula_Estrutura;

(* Simulacao da estrutura escolhida.

Sequencia B representada em 12 bits e sequencia A representada em 16 bits.

Sequencia B unipolar e sequencia A bipolar.

n/k = 4 bits.

*)

(* Definicao das constantes : *)

Const

```

Masc12 = $FFF;           (* Mascara de 12 bits iguais a 1 *)
Masc16 = $FFFF;        (* " " 16 " " " *)
Masc28 = $FFFFFF;     (* " " 28 " " " *)
Masc32 = $FFFFFFFF;    (* " " 32 " " " *)
Tam     = 200;         (* Comprimento dos vetores utilizados *)

```

(* Definicao dos vetores utilizados: *)

Type

```

Arint   = Array [1..Tam] of longint; (* Vetor de Tam posicoes *)
Str     = String[255];

```

(* Definicao das variaveis *)

Var

```

B1,B2,B3,A1,A2,A3,A4,A,B : Arint; (* Vetores das palavras particionadas
                                     Ai=Wi(A) e Bi=Wi(B) *)
Boc,B1c,B2c,B3c,Bc      : Arint; (* Vetores das palavras complementadas
                                     em 2, Bc = Complem. de 2 de B *)
Out                      : Array [1..20,1..4] of Longint;
                          (* Valores na saida das celulas *)
Fact, Check,Tk          : Longint; (* Variaveis auxiliares *)

```

(* Definicao das rotinas utilizadas *)

Procedure Show_Bit(X:Longint);

(* Mostra na tela o numero X em representacao binaria *)

Var

Masc,I : LongInt;

Begin

Masc := \$80000000; (* = 1000....000 (32 bits) *)

For I := 1 to 32 do

Begin

If (Masc And X) = Masc Then

Write('1')

Else

Write('0');

```

        If I < 32 Then Masc := Masc shr 1; (*Deslocamento a direita 1 bit*)
    End; (* For *)
    Writeln;
End; (* Show_Bit *)
(*-----*)

```

```

Procedure Ver_Out(Numcell:Longint);
(* Mostra os conteudos dos registradores da celula[Numcell] *)
Var
    I : Longint;
Begin
    Writeln('Celula numero = ',Numcell);
    For I := 1 to 4 do
        Writeln(Out[Numcell,I]);
    End; (* Ver_Out *)
(*-----*)

```

```

Function Num_Bit(X:Longint):Integer;
(* Calcula o comprimento da palavra X, desprezando os bits de valor zero
   posicionados a esquerda, por exemplo, X = 00010110 tera Num-bit = 5 *)
Var
    I,Cont,Masc : Longint;
Begin
    Masc := 1;
    Cont := 0;
    For I := 1 to 32 do
        Begin
            If (X And Masc) = Masc Then Cont := I;
            If (I < 32 ) Then Masc := Masc shl 1;
        End;
    Num_Bit := Cont;
End; (* Num_Bit *)
(*-----*)

```

```

Procedure C(Num_cell,I1,I2,I3,I4,I5 : Longint);
(* Simula o processamento de cada celula
   Distribuicao das celulas na estrutura e o respectivo numero das celulas:

```

```

X X X      1  2  3
X X X +    4  5  6 13
X X X +    7  8  9 14
X X X +   10 11 12 15
+ + + + + 20 19 18 17 16

```

X => Celula tipo I (1 - 12)

+ => Celula tipo II(13 -20)

Obs : as celulas tipo III, as quais sao responsaveis pela realizacao da operacao complemento 2 da palavra B sera realizada previamente pela rotina "Ajuste".

*)

Var

Auxs, Auxm, Lm, Hm : LongInt;

Begin

If Num_cell > 12 Then (* Significa que a celula eh do tipo II *)

Begin

Auxs := I1 + I2 + I3 + I4 + I5;

Out[Num_cell,2] := (Auxs And \$F); (* Bits L *)

Out[Num_cell,1] := (Auxs And \$F0) Shr 4; (* Bits H *)

Out[Num_cell,3] := 0; (* Nao utilizado *)

Out[Num_cell,4] := 0; (* Nao utilizado *)

(* Verificando resultado : *)

If Auxs <> (Out[Num_Cell,2]+(Out[Num_Cell,1] Shl 4)) Then

Writeln(' Erro na celula : ', Num_Cell);

End (* If *)

Else (* Se A celula eh do tipo I (Num_Cell <= 12) *)

Begin

Out[Num_cell,3] := I1; (* Difusao para a proxima celula *)

Out[Num_cell,4] := I2; (* Difusao para a proxima celula *)

Auxm := I1*I2; (* Operacao de multiplicacao *)

Lm := Auxm And \$F; (* Parte L da multiplicacao *)

Hm := (Auxm And \$F0) Shr 4; (* Parte H da multiplicacao *)

Auxs := Lm + I3 + I4 + I5;

Out[Num_Cell,2] := Auxs and \$F; (* Parte L da resposta *)

Out[Num_Cell,1] := Hm + ((Auxs And \$F0) Shr 4); (*Parte H da resposta*)

(* Verificando Resultado : *)

```

    If (AuxM+I3+I4+I5) <> (Out[Num_Cell,2]+ (Out[Num_Cell,1] Shl 4)) Then
      Writeln('Erro na celula : ',Num_Cell);
    End; (* Else *)
End; (* C *)
(* ----- *)

Procedure Ajuste;
(* Ajusta os vetores de entrada:
Particionando as palavras de entrada Am e Bm,
incluindo os atrasos necessarios entre as subpalavras Wi e
Complementando em 1 (Quando Am for negativo) a palavra Bm
*)
Var
  i, Buc : LongInt;
Begin
  For i := 1 to (Tam-3) Do (* Complemento em 1 *)
    Begin
      If A[i] < 0 Then
        Begin
          Buc := Not(B[i]); (* Complemento de 1 *)
          Boc[i] := $F;
          B3c[i] := Buc And $F;
          B2c[i] := (Buc And $F0) shr 4;
          Bc[i] := 1;      (* Bs(Am) *)
          B1c[i] := (Buc and $F00) Shr 8;
        End; (* If *)
        (* Particionamento e inclusao de atrasos *)
        B1[i] := (B[i] And $F00) Shr 8;
        B2[i+1] := (B[i] And $F0) Shr 4;
        B3[i+2] := (B[i] And $F);
        A4[i] := A[i] And $F;
        A3[i+1] := (A[i] And $F0) Shr 4;
        A2[i+2] := (A[i] And $F00) Shr 8;
        A1[i+3] := (A[i] And $F000) Shr 12;
      End (* For *)
    End; (* Ajusta *)
  (* ----- *)

```

Procedure Clear;

(* Zera todos os conteudos dos vetores utilizados = RESET *)

Var

I, J, K : Longint;

Begin

Check :=0;

For i := 1 to Tam do

Begin

A[i]:=0; B[i]:=0;

Boc[i]:=0; B1c[i]:=0; B2c[i]:=0; B3c[i]:=0;

A1[i]:=0; A2[i]:=0; A3[i]:=0; A4[i]:=0;

B1[i]:=0; B2[i]:=0; B3[i]:=0; Bc[i]:=0;

End; (* For *)

For J:=1 to 20 do

For K:=1 to 4 do

Out[J,K]:=0;

End; (* Clear *)

(* -----*)

Procedure Geracao;

(* Inicializa os vetores de entrada A e B*)

Var

I, Prod : LongInt;

Sum : Real;

Begin

Sum := 0.0;

Prod := 0;

For I:=1 to Trunc(Tam/4) do

Begin

B[i]:=i*fact;

A[i]:=-i*fact;

Prod:= A[i]*B[i];

Sum := Sum + prod;

if (A[i]>Masc12) or (B[i]>Masc16) Or (Sum>Masc32) Or (Prod>Masc28) Then

Begin

A[i]:=0; B[i]:=0;

End; (* If *)

End; (* For *)

End; (* Geracao *)

(* -----*)

Procedure Process(T:Longint);

(* Processamento de todas as celulas simulando um arranjo sistolico *)

Begin

C(20, boc[t], 0, out[20, 2], out[19, 1], 0);

C(19, b1c[t], out[10, 1], out[19, 2], out[18, 1], 0);

C(18, out[10, 2], out[11, 1], out[18, 2], out[17, 1], 0);

C(17, out[11, 2], out[12, 1], out[17, 2], out[16, 1], 0);

C(16, out[12, 2], out[15, 1], out[16, 2], 0, 0);

C(15, out[9, 2], out[14, 1], out[15, 2], 0, 0);

C(14, out[6, 2], out[13, 1], out[14, 2], 0, 0);

C(13, out[3, 2], 0, out[13, 2], 0, 0);

C(12, out[9, 3], out[11, 4], out[8, 2], out[9, 1], 0);

C(11, out[8, 3], out[10, 4], out[7, 2], out[8, 1], 0);

C(9, out[6, 3], out[8, 4], out[5, 2], out[6, 1], 0);

C(10, out[7, 3], A1[t], B2c[t], Out[7, 1], 0);

C(8, out[5, 3], out[7, 4], out[4, 2], out[5, 1], 0);

C(6, Out[3, 3], out[5, 4], out[2, 2], out[3, 1], 0);

C(7, out[4, 3], A2[t], B3c[t], out[4, 1], Bc[t]);

C(5, out[2, 3], out[4, 4], out[1, 2], out[2, 1], 0);

C(3, B3[t], out[2, 4], 0, 0, 0);

C(4, Out[1, 3], A3[t], 0, out[1, 1], 0);

C(2, B2[t], out[1, 4], 0, 0, 0);

C(1, B1[t], A4[t], 0, 0, 0);

End; (* Process *)

(* -----*)

Procedure Checkpro(T:Longint);

(* Calcula o resultado da equacao de otimizacao *)

Begin

Check:=Check + A4[t]*B3[T+2]+((A4[t]*B2[t+1]+A3[t+1]*B3[t+2]) Shl 4);

Check:=Check + ((A4[t]*B1[T]+B2[t+1]*A3[t+1]+A2[t+2]*B3[t+2]) Shl 8);

Check:=Check + ((A3[t+1]*B1[T]+A2[t+2]*B2[t+1]+A1[t+3]*B3[t+2]) Shl 12);

Check:=Check + ((A2[t+2]*B1[T]+A1[t+3]*B2[t+1]) Shl 16);

Check:=Check + ((A1[t+3]*B1[T]) Shl 20);

If A[t] < 0 Then (* Valor de Am eh negativo *)

```

Begin
  Check:=Check + ((1+B3c[t]) Shl 16);
  Check:=Check + (B2c[t] Shl 20) + (B1c[t] shl 24) + (Boc[t] Shl 28);
  End; (* If *)
End; (* CheckPro *)
(* -----*)

```

```

Procedure Resultado;

```

```

(* Calcula os resultados da equacao, da simulacao e o resultado teorico *)

```

```

Var

```

```

  Sum,res,i : Longint;

```

```

Begin

```

```

  (* Resultado da simulacao *)

```

```

  Res:=out[13,2]+(out[14,2] shl 4)+(out[15,2] shl 8)+(out[16,2] shl 12);

```

```

  Res:=Res+(out[17,2] shl 16)+(out[18,2] shl 20)+(out[19,2] shl 24);

```

```

  Res:=Res+(out[20,2] shl 28);

```

```

  Writeln(' Valor simulado da estrutura =',Res);

```

```

  Writeln(' Comprimento da palavra resultado ',Num_bit(Res));

```

```

  Show_Bit(Res);

```

```

  (*Resultado teorico*)

```

```

  Sum:=0;

```

```

  For I:=1 to Tam do

```

```

    Sum:= Sum+A[i]*B[i];

```

```

  Writeln(' Resultado teorico = ',Sum);

```

```

  Writeln('Comprimento do resultado ',Num_bit(Sum));

```

```

  Show_bit(Sum);

```

```

  (* Resultado da equacao de otimizacao *)

```

```

  Writeln(' Resultado da equacao ', Check);

```

```

  Writeln('Comprimento do resultado da equacao ',Num_bit(Check));

```

```

  Show_bit(Check);

```

```

  End; (* Resultado *)

```

```

  (* -----*)

```

```

Procedure CheckOut;

```

```

(* Confere a ocorrencia de "overflow" nos registros de saida das celulas *)

```

```

Var i,j : Integer;

```

```

Begin
  For I:=1 to 20 do
    For J:=1 to 4 do
      if ((NUm_Bit(out[i,j])) > 4) Then Writeln(' Estouro na celula ',i);
    End; (* CheckOut *)
  (*#####*)

  (* Comeco do programa principal *)
  Begin

    Clear; (* Zera os vetores de entrada e registradores *)
    Write(' Forneca o fator de geracao das entradas '); Readln(Fact);
    Geracao; (* Geracao dos vetores de entrada *)
    Ajuste; (* Realiza celula tipo III,
             Particionamento das palavras e inclusao de atrasos *)
    For TK:=1 to Trunc(Tam/2) do
      Begin
        Process(TK); (* processamento paralelo de todas as celulas *)
        CheckPro(Tk); (* Confere resultado de equacao *)
        CheckOut; (* Confere ocorrencia de estouro nas celulas *)
      End;
    Resultado; (* Verifica todos os resultados *)
  End.

```

A P Ê N D I C E B

DIAGRAMA ESQUEMÁTICO DAS CÉLULAS IMPLEMENTADAS EM PGA

Neste apêndice serão apresentados diagramas esquemáticos dos agrupamentos de células (Fig. 4.5) implementados em circuitos PGA.

Agrupamento #1

Nos circuitos PGAs que implementam este agrupamento, foram configurados dois multiplicadores sistólicos de palavras de 4 bits apresentado na Fig. 3.2 e somadores de palavras de 4 bits apresentado na Fig. 3.1. Estes somadores e multiplicadores são conectados de forma a realizar duas células tipo I. O diagrama esquemático da configuração dos blocos do PGA é apresentado na Fig. A.2, onde os blocos de configuração lógica (BCL) apresentam entradas A, B, C, D, K e saídas X e Y, e os blocos de entrada e saída (BES) apresentam entradas O, T, K e saída I, estando este bloco diretamente conetado ao "pad", como mostrado Fig. A.1:

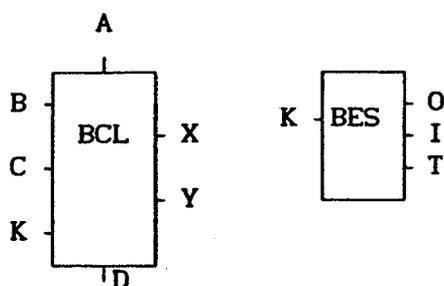


Fig. A.1 - Representação das entradas e saídas dos blocos CL e ES.

A configuração lógica de cada bloco da Fig. A.2 é apresentada abaixo:

a) Blocos x0, x1, x2, x3, z0, z1, z2, z3, w0, w1, w2, w3, y0, y1, y2 e y3 são configurados com entradas sincronizadas pelo pulso de clock conectado na entrada K, esta sincronização é realizada por um "latch" existente nos blocos BES. Esta configuração é representada por:

$$I = \text{PAD} (K)$$

Onde I representa bloco configurado como entrada, PAD representa que a entrada esta conectada ao pino do circuito integrado ("pad") e (K) significa entrada sincronizada pelo pulso de clock conectado ao pino K.

b) Blocos x00, x11, x22, x33, z00, z11, z22, z33, w00, w11, w22, w33, y00, y11, y22, y33, hb0, hb1, hb2, hb3, lb3, lb2, lb1, lb0, la3, la2, la1 e la0, são configurados como saídas sem sincronismo:

$$O = \text{PAD}$$

c) Blocos clk, ie0, ie1, ie2, ie3, ic3, ic2, ic1, ic0, nn, ib3, ib2, ib1 e ib0 são configurados como entradas sem sincronismo:

$$I = \text{PAD}$$

d) Blocos F1 e F1_B são configurados como portas E:

$$X = A.C \quad (K) \quad (\text{sincronizada})$$

$$Y = B.C \quad (\text{Sem sincronismo})$$

e) Blocos D2 e D2_B são configurados com a lógica abaixo:

$$X = \bar{B}.A.C + B.(A.C) \quad (K)$$

$$Y = B.A.C$$

f) Blocos A1, A12, A3, A32, A4, A1_B, A12_B, A3_B, A32_B e A4_B são configurados como portas E:

$$X = A.C$$

$$Y = B.C$$

g) Blocos B2, B22, B2_B e B22_B :

$$X = \bar{B}.A.C + B.(\overline{A.C})$$

$$Y = B.A.C$$

h) Blocos C2, C3, C4, C42, S1, S2, S3, S4, C2_B, C3_B, C4_B, C42_B, S2_B, S3_B e S4_B :

$$X = \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C$$

$$Y = C.A + B.(\bar{A}.C + A.\bar{C})$$

i) Blocos E3, E4, E52, E53, E3_B, E4_B, E52_B, E53_B, S6, S7, S8, S9, S6_B, S7_B, S8_B, S9_B :

$$X = \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C \quad (K)$$

$$Y = C.A + B.(\bar{A}.C + A.\bar{C})$$

j) Blocos E5, E5_B :

$$X = \bar{A}.B + A.\bar{B} \quad (K)$$

$$Y = A.B$$

k) Blocos A5, A5_B:

$$X = C \quad (K)$$

$$y = A.B$$

l) Blocos S5, S10, S11, S5_B, S10_B, S11_B :

$$X = A.\bar{C} + \bar{A}.C \quad (K)$$

$$Y = A.C$$

m) Blocos S12, S12_B :

$$X = A.\bar{C} + \bar{A}.C \quad (K)$$

n) Bloco S1_B :

$$X = \bar{A}.B + \bar{B}.A$$

$$Y = A.B$$

Agrupamento #2

No circuito PGA que implementa este agrupamento foram configuradas células do tipo III, mostradas na Fig. 4.2, realizando-se portas OU-Exclusivo e Portas "E". O diagrama esquemático apresentado na Fig. A.3 ilustra a realização deste agrupamento através da configuração de blocos do circuito PGA. A configuração lógica de cada bloco é apresentada a seguir:

a) Bloco inn:

$$I = PAD \quad (K)$$

b) Bloco onn:

$$O = PAD$$

c) Bloco clk:

$$I = PAD$$

d) Blocos a30, a31, a32, a33, a20, a21, a22, a23, a10, a11, a12 e a13 :

$$I = \text{PAD} \quad (K)$$

e) Blocos s30, s31, s32, s33, s20, s21, s22, s23, s10, s11, s12 e s13 :

$$O = \text{PAD}$$

f) Blocos x1, x2, x3, x4, x1_b, x2_b, x3_b, x4_b, x1_c, x2_c, x3_c e x4_c :

$$X = \bar{A}.B \quad (K)$$

Agrupamento #3

No circuito PGA que implementa este agrupamento são realizadas 4 células tipo II através da utilização de somadores completos "full adders" e "latches". O diagrama esquemático da realização deste agrupamento através de circuitos PGAs é apresentado na Fig. A.4. A configuração lógica de cada bloco utilizado para a realização deste agrupamento de células é apresentado a seguir:

a) Blocos clk, aa0, aa1, aa2, aa3, bb0, bb1, bb2, bb3, cc0, cc1, cc2, cc3, dd0, dd1, dd2 e dd3 :

$$I = \text{PAD}$$

b) Blocos blc, sa0, sa1, sa2, sa3, sb0, sb1, sb2, sb3, sc0, sc1, sc2, sc3, sd0, sd1, sd2 e sd3 :

$$O = \text{PAD}$$

c) Bloco a1:

$$X = A.\bar{C} + \bar{A}.C$$

$$Y = A.C$$

d) Blocos a2, a3, a4, b1, b2, b3, b4, c1, c2, c3, c4, d1, d2, d3 e d4 :

$$X = \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C \quad (K)$$

$$Y = A.B + C.(\bar{A}.B + A.\bar{B})$$

e) Blocos a5, b5 c5 e d5 :

$$X = C \quad (K)$$

Agrupamento #4

No circuito PGA que implementa este agrupamento são realizados 4 células tipo II, e o diagrama esquemático da realização deste agrupamento é apresentado na Fig. A.5. A configuração lógica implementada em cada bloco utilizado na realização deste agrupamento é apresentada a seguir:

a) Blocos ab0,aa0, ab1, aa1, ab2, aa2, ab3, aa3, bb0, ba0, bb1, ba1, bb2, ba2, bb3, ba3, cb0, ca0, cb1, ca1, cb2, ca2, cb3, ca3, clk e lcc :

$$I = PAD$$

b) Blocos ob3, ob2, ob1, ob0, oc0, oc1, oc2, oc3, od0, od1, od2, od3, od4, od5, oa0, oa1, oa2 e oa3 :

$$O = PAD$$

c) Blocos a1, a2, a3, a4, b1, b2, b3, b4, c1, c2, c3 e c4 :

$$X = \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C$$

$$Y = A.B + C.(\bar{A}.B + A.\bar{B})$$

d) Blocos a5, ao5, b5, bo5, c5 e co5 :

$$X = A \quad (K)$$

e) Bloco ao1 :

$$X = A.\bar{C} + \bar{A}.C \quad (K)$$

$$Y = A.C$$

f) Blocos ao2, ao3, ao4, bo1, bo2, bo3, bo4, co1, co2, co3, co4, do1, do2, do3, do4 e do5 :

$$X = \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + A.B.C \quad (K)$$

$$Y = A.B + C.(\bar{A}.B + A.\bar{B})$$

g) Blocos d1, d2, d3 e d4 :

$$X = A.\bar{C} + \bar{A}.C$$

$$Y = A.C$$

h) Bloco do6 :

$$X = A + B \quad (K)$$

Print Design: JOCCELL2.LCA (2064PC68-100), XACT 3.00, Mon Apr 29 13:15:56 1991

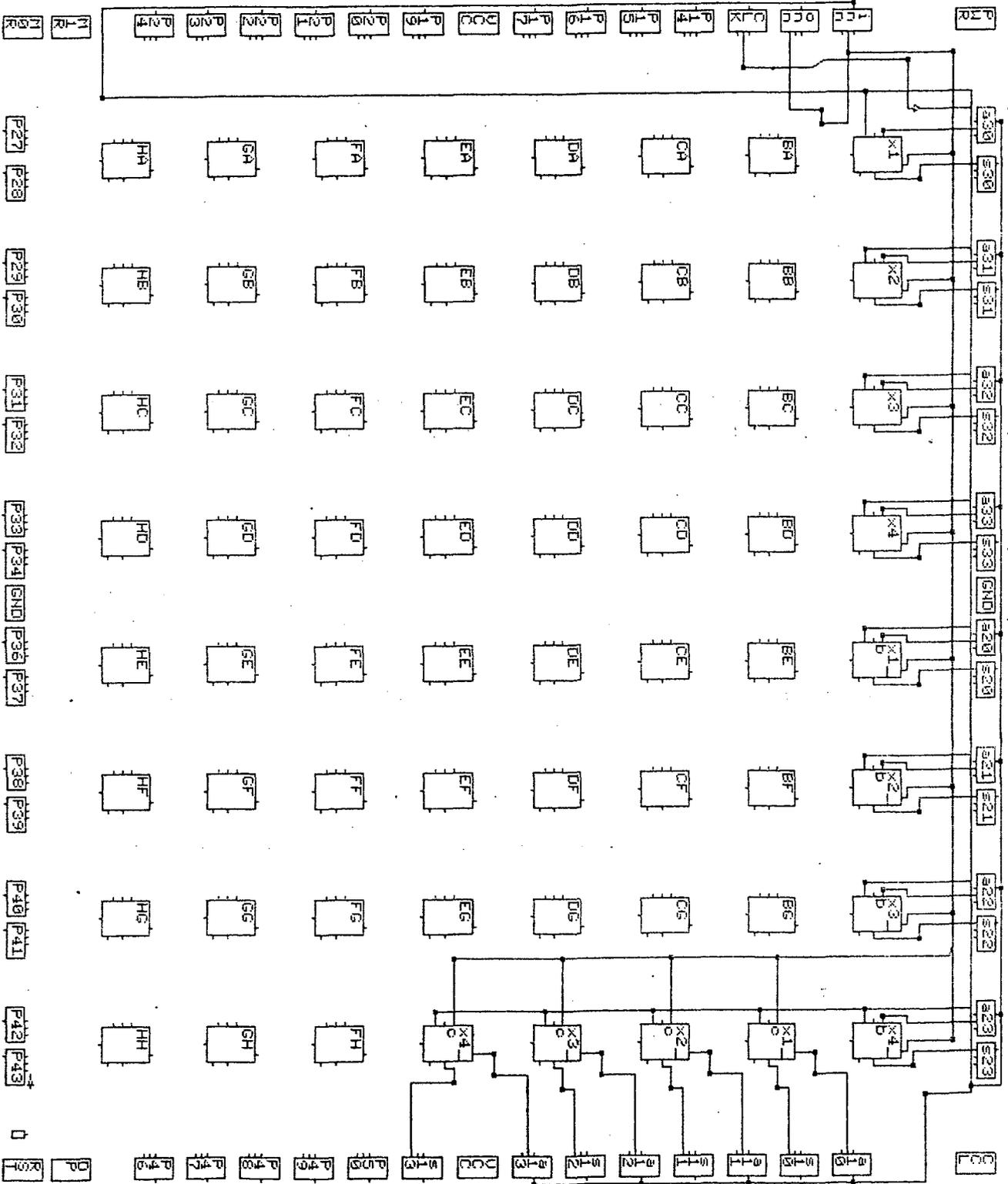


Fig. A-3 - Agrupamento 2.

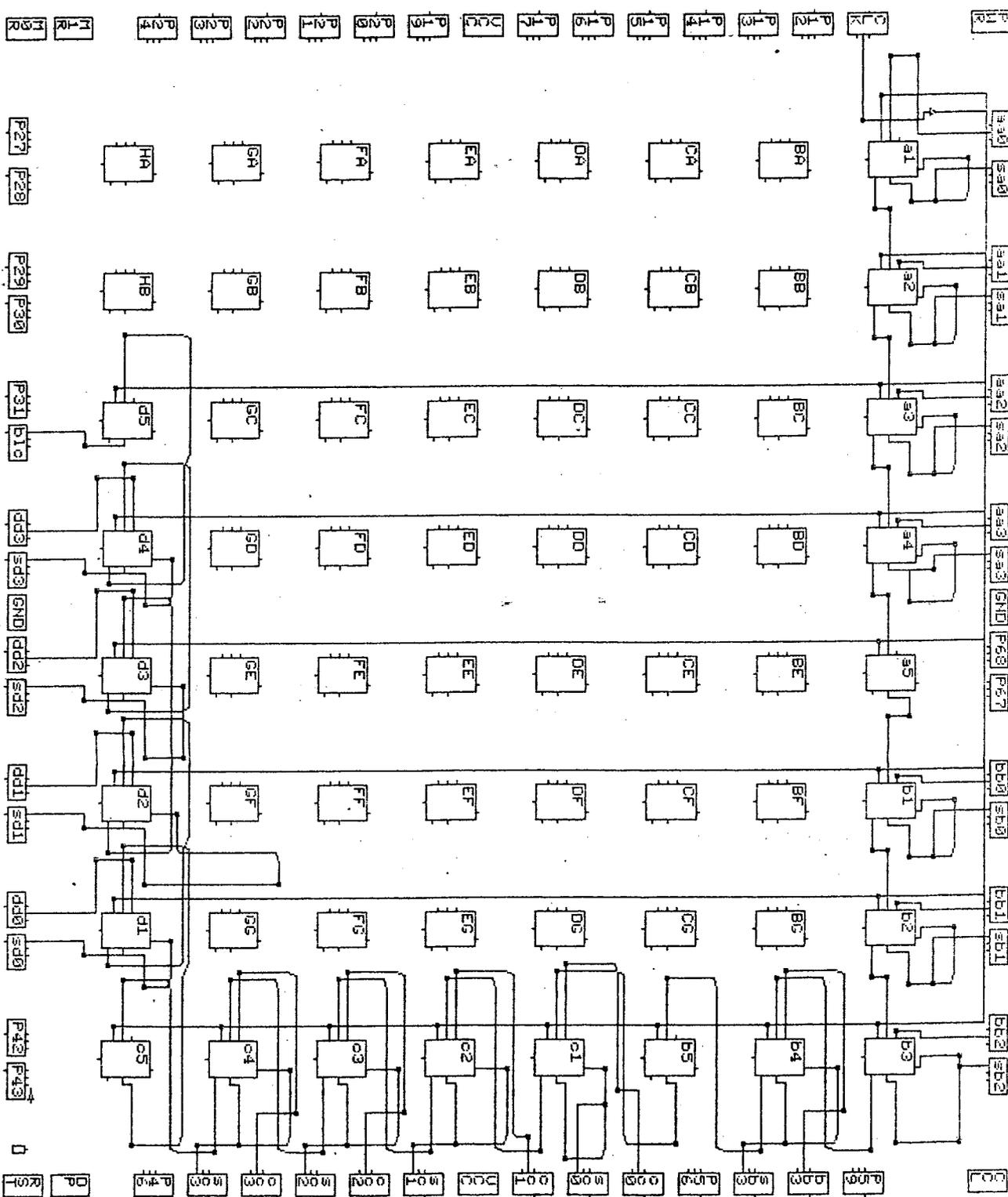


Fig. A-4 - Agrupamento 3.

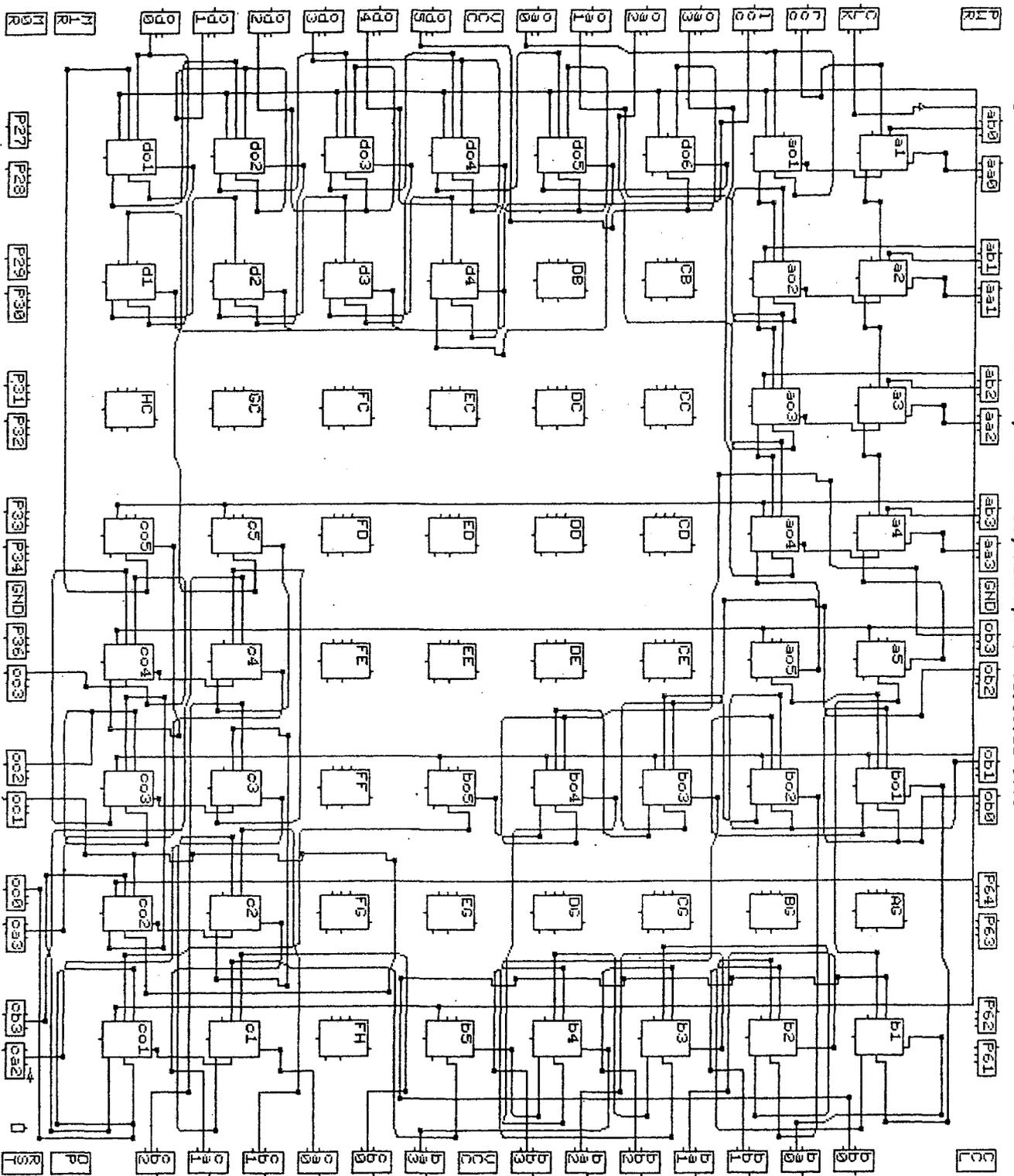


Fig. A-5 - Agrupamento 4.

APÊNDICE C

DIAGRAMA ESQUEMÁTICO DA PLACA DE TESTES

O diagrama esquemático apresentado a seguir presta-se a realização de uma placa de testes para os agrupamentos de células apresentados na Fig. 4.5. Para cada agrupamento será necessário realizar uma determinada bateria de testes e cada agrupamento tem sua específica configuração de pinos de entrada e saída. Portanto, para testar a funcionalidade de cada agrupamento de células implementado em um PGA é necessário que se carregue o programa de configuração em uma memória EPROM do tipo 2764, e conecte-se os pinos de entrada e saída a "LEDs" e "dip-switches" conforme a necessidade da configuração a ser testada. Esta placa foi implementada em "wire-wrap" e os testes necessários foram realizados, confirmando a funcionalidade de todos os grupos de células implementados em circuitos PGA.

Nestes testes foram verificados se os agrupamentos de células operavam corretamente. Esta verificação foi realizada através da modificação das entradas do PGA, o qual estava configurado para o agrupamento das células em teste, através das chaves "dip-switches" conectadas às entradas. As saídas do PGA, em verificação, são conectadas a "LEDs" para a verificação da resposta em função das entradas. Desta forma, todos os agrupamentos foram testados utilizando-se esta placa de testes.

REFERÊNCIAS

- [1] A. V. Oppenheim, R. W. Shafer, "Digital Signal Processing", Prentice-Hall, Inc., 1975.
- [2] S. Y. Kung, H. J. Whitehouse, T. Kailath, Editors, "VLSI and Modern Signal Processing", Prentice-Hall, Inc., 1985.
- [3] B. Ryan, "Multiprocessors Surf's Up", Byte Magazine, pp. 199-206, June 1991.
- [4] S. Y. Kung, "On Supercomputing with Systolic/Wave Front Array Processors", Proc. IEEE, Vol. 72, No.7, pp. 867-883, July 1984.
- [5] R. Roy, M. A. Bayoumi, "An Efficient Two's Complement Systolic Multiplier for Real-Time Digital Signal Processing", IEEE Trans. on CAS, Vol. 36, No. 11, pp. 1488-1493, November 1989.
- [6] A. Aliphas, J. A. Feldman, "The Versality of Digital Signal Processing Chips", IEEE Spectrum, pp. 40-45, June 1987.
- [7] M. O. Ahmad, D. V. Poornalah, "Design of an Efficient VLSI Inner-Product Processor for Real-Time DSP Applications", IEEE Trans. on CAS, Vol. 36, No. 2, pp. 324-329, February 1989.
- [8] C. S. Wallace, "A suggestion for a Fast Multiplier", IEEE Trans. on Electronic Computers, Vo. EC-3, pp. 14-17, February 1964.

- [9] C. L. Wey, T. Y. Chang, "Design and Analysis of VLSI-Based Parallel Multipliers", IEE Proc., Vol. 137, Pt. E, No.4, pp. 328-336, July 1990.
- [10] M. Hatamiam, G. L. Cash, "Parallel Bit-Level Pipelined VLSI Designs for High-Speed Signal Processing", Proc. of IEEE, Vol. 75, No. 9, pp. 1192-1202, September 1987.
- [11] S. Y. Kung, "VLSI Array Processors", IEEE ASSP Magazine, pp. 4-22, July 1985.
- [12] E. G. Friedman, J. H. Mulligan Jr., "Clock Frequency and Latency in Synchronous Digital Systems", IEEE Trans. Signal Processing, Vol. 39, No. 4, pp. 930-934, April 1991.
- [13] E. E. Swartzlander Jr., Editor, "Systolic Signal Processing Systems", Marcel Dekker Inc., 1987.
- [14] A. Malvino, "Microcomputadores e Microprocessadores", McGraw-Hill, 1985.
- [15] P. R. Cappelo, K. Steiglitz, "A Note on 'Free Accumulation' in VLSI Filter Architectures", IEEE Trans. CAS, Vol. 32, No. 3, pp. 291-296, March 1985.
- [16] G. K. Ma, F. J. Taylor, "Multiplier Policies for Digital Signal Processing", IEE ASSP Magazine, pp. 6-18, January 1990.
- [17] C. Mead, L. Conway, "Introduction to VLSI Systems", Addison-Wesley Publishing Inc. 1980.
- [18] Xilinx Inc., "The Programmable Gate Array Data Book", 1989.
- [19] H. T. Kung, "Why Systolic Architectures", IEEE Computer, Vol. 15, No. 1, January 1982.