UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Este exemplar corresponde à redação final da tese defendida por Henrique Sérgio Gutierrez da Costa e aprovada pela Comissão Julgadora em 28/02/92.

Orientador

ARQUITETURA DEDICADA À DETECÇÃO DE BORDAS EM IMAGENS MONOCROMÁTICAS

Por : HENRIQUE SÉRGIO GUTIERREZ DA COSTA

Orientador: PROFESSOR DOUTOR CLÉSIO LUIS TOZZI

Tese de Mestrado apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas.

FEVEREIRO DE 1992.

Agradecimentos:

Aos profissionais do Centro Tecnológico para Informática pelo apoio técnico e pela amizade.

Aos profissionais da Itaucom pelo apoio técnico, pelas ferramentas e pelo incentivo a este trabalho.

À minha família e minha noiva, cujo apoio e paciência tornaram esta jornada possível.

Ao Clésio, pelo apoio no desenvolvimento deste trabalho como orientador e como amigo.

Resumo

Uma arquitetura dedicada a detecção de bordas implementada como um circuito integrado é apresentada. São analisadas várias alternativas para a implementação de seus componentes básicos (somadores, multiplicadores, memórias, etc) considerando a velocidade de operação, modularidade e estilo de projeto.

São apresentados os projetos de cada módulo componente do sistema e sua implementação em VLSI, bem como resultados de simulações do modelo.

É efetuada uma comparação entre a arquitetura implementada e outras alternativas apresentadas na literatura.

Abstract

The implementation of an edge detection architecture as an integrated circuit is presented. Alternatives for the implementation of the basic modules (adders, multipliers, memories, etc) are analised considering the operation speed, the modularity and the design style.

The design of each module, its VLSI implementation and the simulation results are presented.

The implemented architecture is compared to other alternatives shown in the literature.

ÍNDICE

CAPÍTULO I

INTRODUÇÃO

I.1 - Importância do Processo de Detecção de Bordas

Este trabalho vem a contribuir principalmente com um estudo de adequação de uma arquitetura para detecção de bordas descrita em alto nível a uma implementação como circuito integrado analisando todos os fatores relacionados a este tipo de implementação, tais como a configuração lógica (incluindo o uso de arquiteturas inovadoras), acesso aos barramentos, análise de tempos, dimensionamento de portas lógicas, desenho físico de circuitos integrados, geração de vetores de teste e análise de resultados.

A operação de detecção de bordas é parte essencial do processo de reconhecimento de padrões, sendo que este permite a extração de características de uma imagem-alvo que podem ser úteis a várias aplicações, tais como o processamento de documentos, automação industrial, medicina, biologia, sensoriamento remoto, etc.

Este processo compõe-se basicamente dos estágios mostrados na figura I.1.

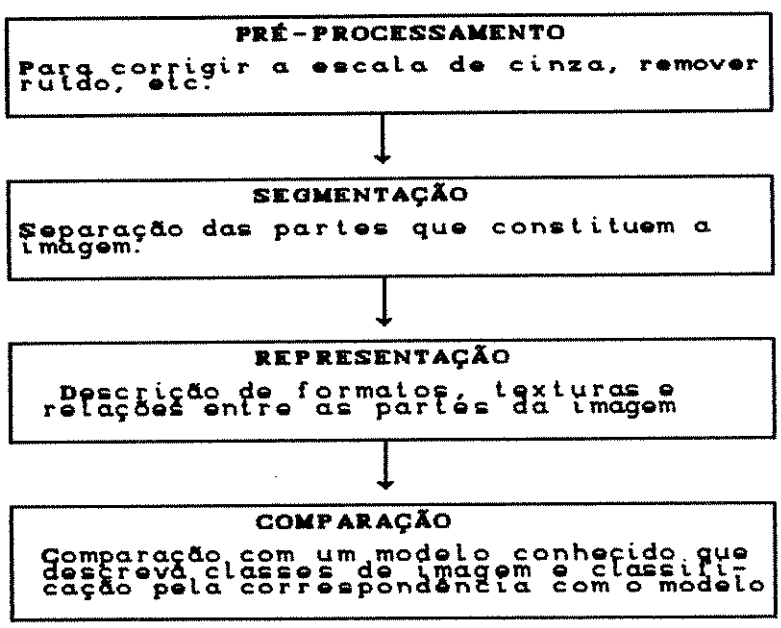


Figura I.1 - O Processo de Reconhecimento de Padrões

Dentro do processo de segmentação de imagem destaca-se a operação de detecção de bordas, a qual permite a definição de fronteiras entre as diferentes regiões que compõem a imagem. Posteriormente, os pontos então reconhecidos como bordas serão submetidos a algoritmos que permitirão a representação das partes componentes da imagem, possibilitando a comparação com modelo conhecido.

I.2 - O ALGORITMO A SER UTILIZADO, O OPERADOR DE SOBEL

Entre os métodos mais utilizados para a detecção de bordas destaca-se o uso de médias locais ponderadas, que juntamente com a comparação com um valor-limite permite a detecção de bordas em várias direções. Estas médias locais atuam sobre o ponto de imagem e à sua vizinhança (Figura I.2). Ao conjunto de pesos usados no cálculo destas médias dá-se o nome de "máscara" ou "kernel" (Figura I.3).

Assim, dada uma máscara com coeficientes w1, w2, w3, w4, w5, w6, w7, w8 e w9, a operação executada será:

$$T[f(x,y)] = w1.f(x-1,y-1) + w2.f(x-1,y) + w3.f(x-1,y+1) + w4.f(x,y-1) + w5.f(x,y+1) + w6.f(x,y+1) + w7.f(x+1,y-1) + w8.f(x+1,y) + w7.f(x+1,y+1)$$

O algoritmo a ser implementado neste trabalho (operador de Sobel) pertence a uma classe de métodos de transformação de imagens denominados "métodos do domínio espacial", ou seja, métodos que atuam diretamente sobre os "pixels" de imagem [9] e não no domínio da frequência. Este algorítmo é normalmente expresso sob a forma de máscaras ou "kernels" (Figura I.3).

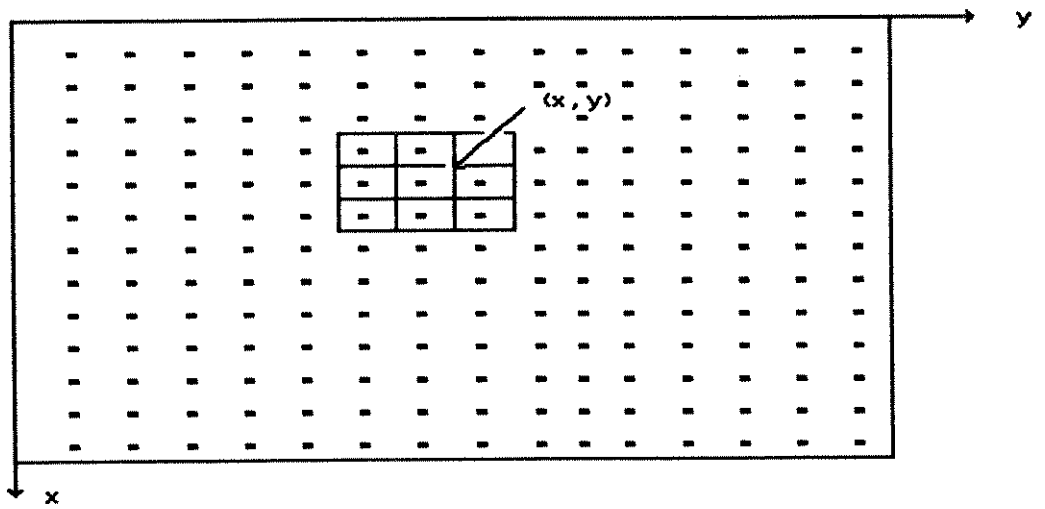


Figura I.2 - Cálculo de média ponderada sobre um ponto e os pontos vizinhos a ele

v1	v2	v3
v4	v5	v6
v7	v8	v9

Figura I.3 - Algoritmo expresso sob a forma de máscara.

A combinação dos operadores de Sobel permite a detecção de bordas segundo qualquer direção (Figura I.4)

-1	-1	-1
2	2	2
-1	-1	-1

-1	-1	2
-1	2	-1
2	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

2	-1	-1
-1	2	-1
-1	-1	2

Figura I.4 - Operadores de Sobel para a detecção de bordas.

Na proposta apresentada por [4], a ser descrita no capítulo III, propõe-se uma arquitetura para a detecção de bordas que utiliza os operadores de Sobel segundo as direções horizontal e vertical (Figura I.5). Os resultados obtidos pela aplicação dos dois operadores são somados, sendo o resultado desta operação comparado a um valor-limite (threshold) o qual determina finalmente se o ponto pertence ou não a uma borda.

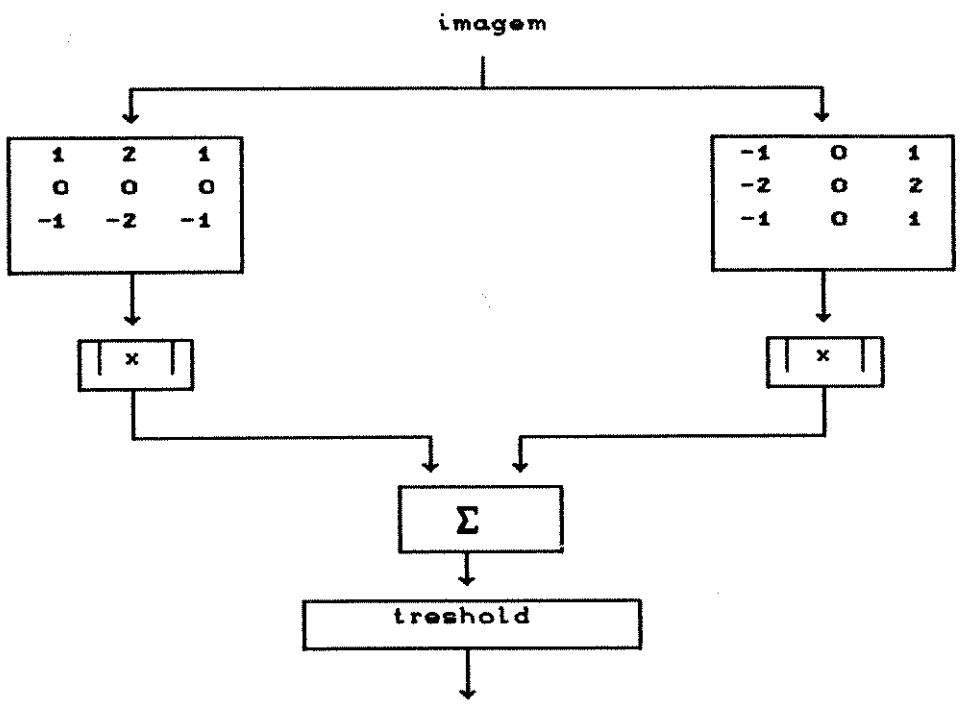


Figura I.5 - O algoritmo utilizado

I.3 - O QUE SE PROPÕE NESTE TRABALHO

O presente projeto consiste na implementação de uma solução de "hardware" como um circuito integrado , o mesmo foi executado em várias etapas cujos resultados obtidos foram estruturados em capítulos que serão discutidos no desenvolver do trabalho e cuja síntese é apresentada a seguir.

No capítulo II discutem-se várias arquiteturas de outros autores, cujas informações auxiliaram a síntese do projeto e permitiram obter-se os parâmetros necessários para avaliação comparativa de características, tais como desempenho, área de silício, dissipação térmica, frequência de operação, funcionamento, flexibilidade, etc.

No capítulo III a proposta original da arquitetura a ser implementada é descrita e estudada.

No capítulo IV são discutidas as melhores alternativas para a implementação física do projeto. Aspectos como a opção tecnológica (matrizes de células, células-padrão, "layout" dedicado ao projeto, ou "layout" parcialmente dedicado ao projeto), dimensionamento e desenho das portas lógicas, de "drivers" de saída e de células de proteção de entrada são estudados, sendo então selecionadas aquelas alternativas que apresentavam maior adequação aos projetos a serem desenvolvidos.

No capítulo V são definidas as interfaces entre os blocos que compõem a arquitetura e também a comunicação com os barramentos externos. Define-se o funcionamento lógico desejado e as propostas que pudessem satisfazer a este.

Ainda neste capítulo, as propostas para cada bloco são estudadas determinando-se como poderiam ser adaptadas à forma de circuitos integrados, se satisfariam aos requisitos de desempenho, quais opções demandariam menor lógica, se os tempos de operação cumpririam as exigências de interface com outros blocos, se as soluções de lógica encontradas poderiam ser utilizadas em outras partes do circuito decidindo-se finalmente pela implementação lógica a ser usada em cada bloco.

Ao final deste capítulo, é discutido o projeto de dois circuitos integrados-protótipo para a validação da arquitetura que foi realizada com o auxílio de ferramentas de simulação lógica. Tendo como base os resultados obtidos o projeto lógico sofreu algumas alterações.

No capítulo VI são definidas as dimensões físicas de todo o circuito e a adaptação de alguns blocos da arquitetura lógica à área de silício disponível, de modo a conseguir implementar alguns blocos importantes do circuito.

No capítulo VII os resultados das simulações são apresentados, analisados e comparados com os resultados esperados. Neste capítulo também são apresentadas as conclusões sobre todo o trabalho desenvolvido.

Nos apêndices I e II são mostrados o "layout" dos circuitos integrados difundidos e as características das portas lógicas da biblioteca de células-padrão.

CAPÍTULO II

REVISÃO DAS ARQUITETURAS IMPLEMENTADAS COMO CIRCUITOS INTEGRADOS PARA A DETECÇÃO DE BORDAS

II.1 - INTRODUÇÃO

Neste capítulo são analisadas arquiteturas propostas na literatura com o objetivo de obter subsídios ao trabalho em sua implementação lógica, física e funcional.

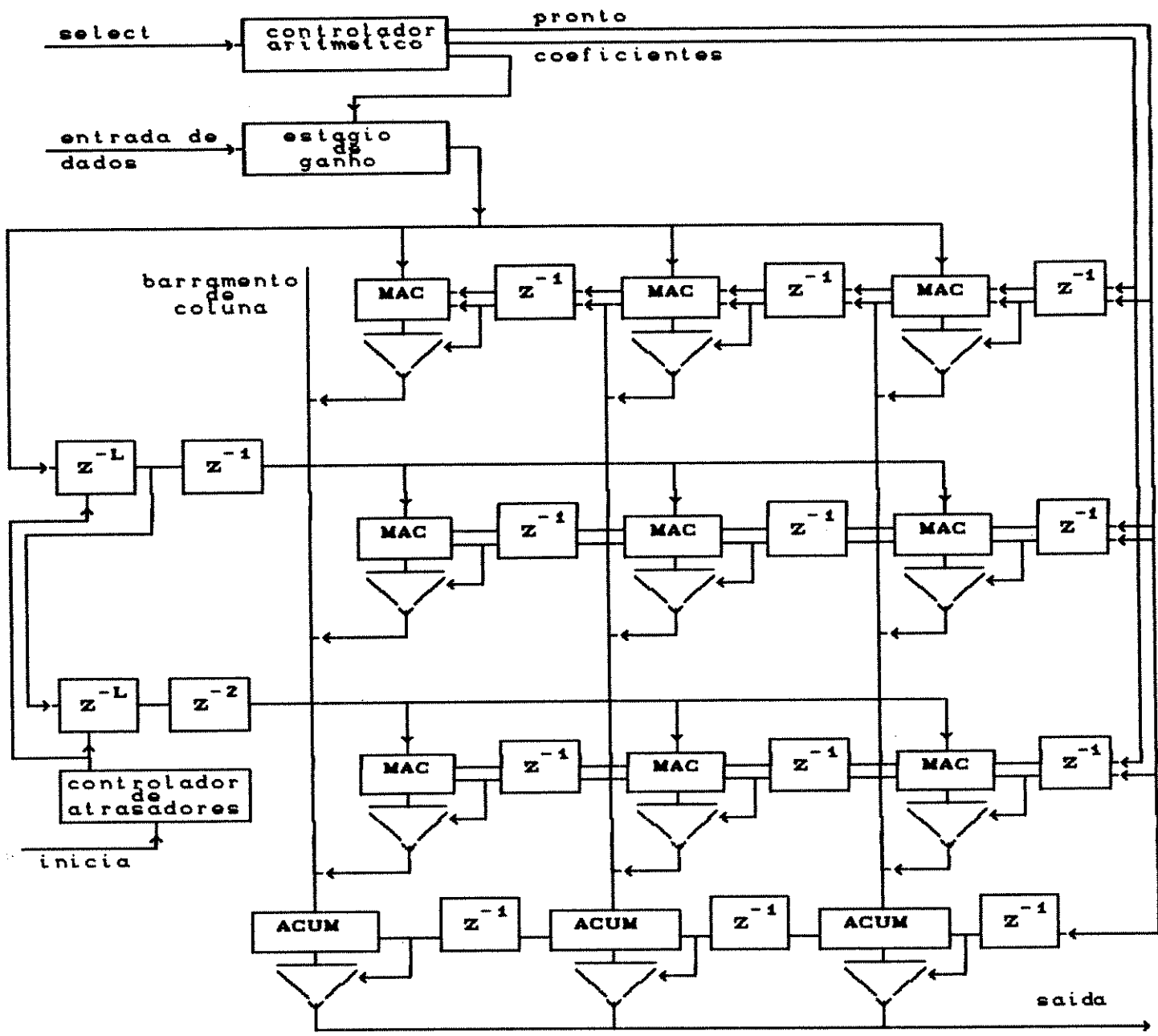
Inicialmente são descritas várias arquiteturas para a detecção de bordas, nos ítens:

II.2 - Arquitetura de Ruetz [18] para Convolução.
II.3 - Arquitetura de Arambepola [3] para Convolução Bidimensional.
II.4 - Arquitetura de Kanopoulos [14] para Detecção de Bordas.
II.5 - Arquitetura de Aono [2] para Convolução Bidimensional

Ao final do capítulo é apresentada uma tabela comparativa de características das arquiteturas descritas , fornecendo subsídios à comparação posterior com arquitetura a ser implementada.

II.2 - Arquitetura de Ruetz para Convolução

A arquitetura de Ruetz (Figura II.1) executa convolução em imagens de 512 X 512 "pixels" usando uma máscara de convolução 3X3.



z^{-1} - atrasador de caractere
z^{-L} - atrasador de linha
MAC - multiplicador-acumulador
ACUM - acumulador

Figura II.1 - Arquitetura de Ruetz para Detecção de Bordas

A arquitetura consiste basicamente de um estágio de ganho, dois atrasadores de linha (que são memórias FIFO com número de posições igual ao número de "pixels" numa linha de imagem), nove multiplicadores-acumuladores dispostos segundo uma matriz 3 X 3, três acumuladores, controladores para os atrasadores de linha e unidades aritméticas.

Funcionamento

Inicialmente o dado de entrada é normalizado pelo estágio de ganho, evitando assim "overflow" nas unidades aritméticas. O dado de saída do estágio de ganho é levado aos multiplicadores-acumuladores da primeira linha da matriz de multiplicadores-acumuladores (ver fig.II.1) e aos atrasadores de linha [Z^{-L}] ,de maneira que,quando os "pixels" da terceira linha de imagem estiverem à saída do estágio de ganho, os "pixels" da segunda e primeira linhas de imagem também estarão fazendo parte dos cálculos que estarão sendo executados respectivamente na segunda e na terceira linhas da matriz de multiplicadores-acumuladores (MAC). Isto é necessário visto que o cálculo da convolução sobre um "pixel" envolve os "pixels" vizinhos da linha anterior e posterior a linha que o contém.

Assim, os "pixels" das sucessivas linhas irão sendo serialmente carregados em linhas consecutivas da matriz de multiplicadores-acumuladores, de maneira que resultados parciais da convolução 3X3 sejam calculados a cada ciclo, tendo-se ao fim de três ciclos o resultado final da convolução para um ponto.

Os coeficientes do "kernel" de convolução são inicialmente carregados nos multiplicadores-acumuladores. Durante o cálculo, estes coeficientes irão sendo deslocados entre os multiplicadores--acumuladores, de modo que tenha-se a cada ciclo de relógio apenas um sub-resultado por coluna de MAC's. Este sub-resultado equivale ao cálculo da convolução 3x3 para uma coluna de coeficientes do "kernel" de convolução, o qual será somado a outros sub-resultados que são sucessivamente calculados até que se tenha o cálculo completo da convolução 3x3.

O circuito opera a 10 Mhz sobre imagens de 512 X 512 "pixels" e foi implementado em uma única pastilha com tecnologia NMOS 4 μ, ocupando uma área de silício de 42 mm^2. O circuito trabalha com dados de 8 bits e o resultado do cálculo tem 10 bits.

Nada foi citado sobre os métodos usados para arredondamento ou truncamento de dados.

Este circuito foi concebido com arquitetura dedicada à execução de algoritmos de convolução adaptáveis a um formato de "kernel" 3X3 , assim, não foram previstas características que o dotassem de maior flexibilidade, tais como a possibilidade de cascateamento para uso de "kernels" de convolução diferentes de 3X3, o uso de coeficientes não múltiplos de 2, etc. As vantagens que se obtém com as limitações anteriormente citadas são a possibilidade de implementação das funções lógicas com circuitos mais simples (que ocupam menor área de silício) e o aumento de desempenho proporcionado por estes circuitos. Como um exemplo poderia-se citar a implementação de multiplicadores com deslocamento (shift) de bits do dado, aproveitando-se do fato de que os coeficientes usados são múltiplos de 2.

A única característica mencionada que torna o circuito mais flexível é a possibilidade de se carregar um microcódigo que comandará a execução de determinado algoritmo.

II.3 - Arquitetura de Arambepola para Convolução Bidimensional

A arquitetura proposta em [3] é um circuito para o cálculo de convolução unidimensional (kernel de 1X10) ou bidimensional (kernel de 2X5 ou 3X3) sobre dados de 10 bits usando coeficientes de 8 bits produzindo um resultado de 26 bits.

A arquitetura é basicamente composta de um estágio de convolução e um estágio de cascateamento.

O estágio de convolução mostrado na figura II.1 é composto de uma matriz de multiplicadores-acumuladores e uma lógica de atraso para configurar o circuito como um convolutor 3x3, 2x5 ou 1x10.

O circuito tem 3 "ports" de entrada XA, XB e XC. Caso se queira executar convolução 3x3 os dados de entrada de três linhas consecutivas são aplicados aos três "ports", sendo os dados de XA e XB atrasados de 3 e 5 ciclos de relógio respectivamente para compensar os atrasos introduzidos pelos registradores "d" dispostos entre os multiplicadores-acumuladores para armazenar os sub-resultados dos cálculos, os quais irão se somar aos sub-resultados calculados nos ciclos de relógio posteriores.

A leitura de dados se faz linha a linha de imagem usando-se o artifício de atrasar os dados de 2 linhas, como pode ser visto na figura II.3.

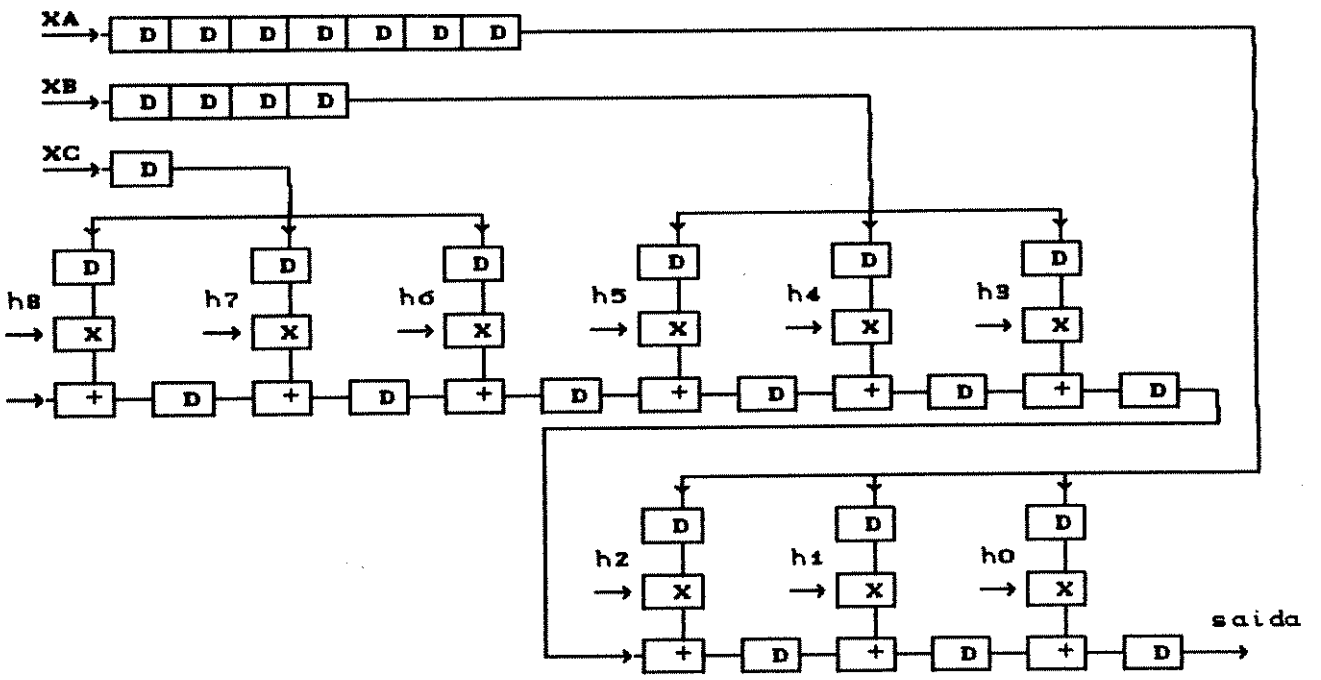


Figura II.2 - Arquitetura de Arambepola para Convolução Bidimensional

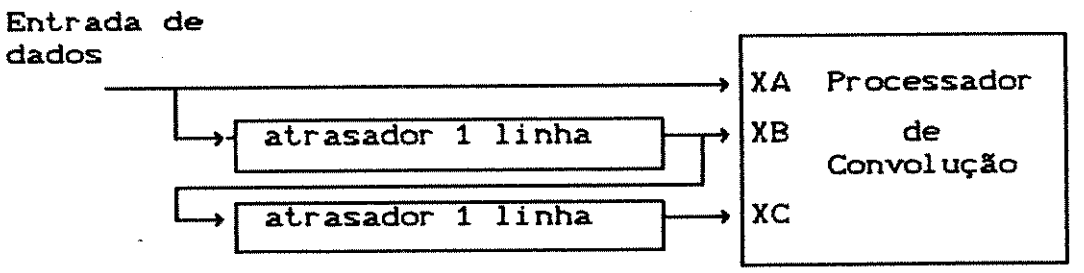


Figura II.3 - Atrasadores de linha

Funcionamento

Os dados de 3 linhas sucessivas são levados aos "ports" XA, XB e XC do circuito, sendo cada "pixel" da linha multiplicado por 3 coeficientes do "kernel" 3x3 de convolução. O resultado desta multiplicação é armazenado no atrasador "d" (que é um registrador) e, quando ocorre o próximo ciclo de relógio, somado ao próximo resultado da multiplicação de um novo "pixel" por coeficientes do "kernel" de convolução, o processo se repete até que as somas sucessivas resultem na convolução 3x3 de um "pixel".

A ordem de entrada de "pixels" e dos deslocamentos e somas de sub-resultados garante o cálculo da convolução 3x3 na ordem correta ocorrendo portanto, o cálculo da convolução 3x3 sobre os sucessivos "pixels" a cada ciclo de relógio.

Lógica de Cascateamento

Consiste de um circuito para somar o resultado da convolução calculado no próprio módulo de convolução aos resultados de outras módulos numa cascata (figura II.4). Assim o circuito pode ser facilmente configurado para usar "kernels" de convolução diferentes de 3x3, ou 2x5, ou 1x10.

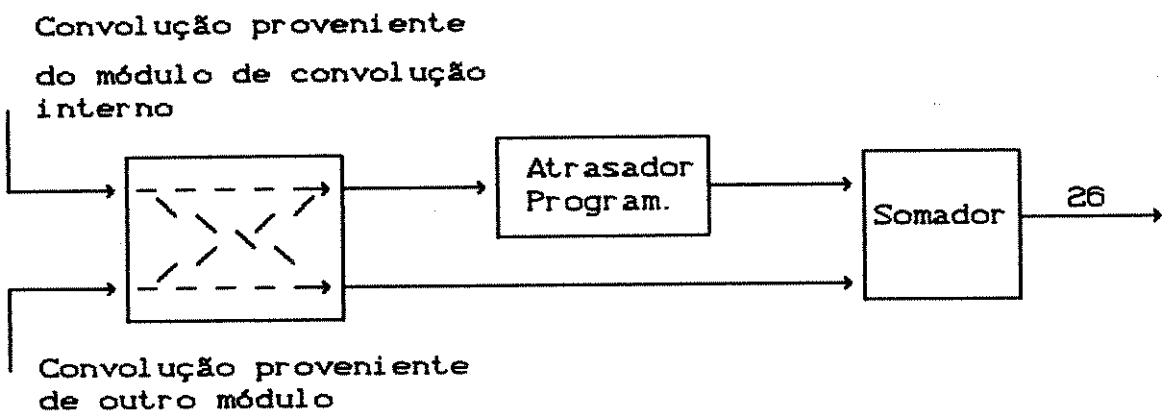


Figura II.4 - Módulo de cascateamento

O circuito descrito foi implementado em tecnologia CMOS-SOS, 3 micra e opera a 20 MHz e é capaz de realizar convolução tanto 3X3 quanto 2X5.

Nada foi citado sobre a área de silício nem sobre a dissipação.

II.4 - Arquitetura de Kanopoulos para Convolução Bidimensional

O processador proposto em [14] consiste de 4 estágios que calculam convolução com o operador de Sobel nas direções, vertical, horizontal e nas diagonais à esquerda (135°) e à direita (45°) (como pode ser visto na figura II.3). Uma vez efetuados os cálculos combinam-se os resultados de maneira a se obter a magnitude e a direção das bordas.

Valor Calculado pelo Processador de Magnitude e Direção

Magnitude da Borda :

$$MAG = máx \langle | Eh |, | Ev |, | E45^{\circ} |, | E135^{\circ} | \rangle + \langle K.(E_{\perp}) \rangle$$

Onde Eh, Ev, $E45^{\circ}$, $E135^{\circ}$ e o gradiente segundo as direções horizontal, vertical, 45 graus e 135 graus.

$E_{\perp}$ = magnitude na direção perpendicular à da máxima magnitude

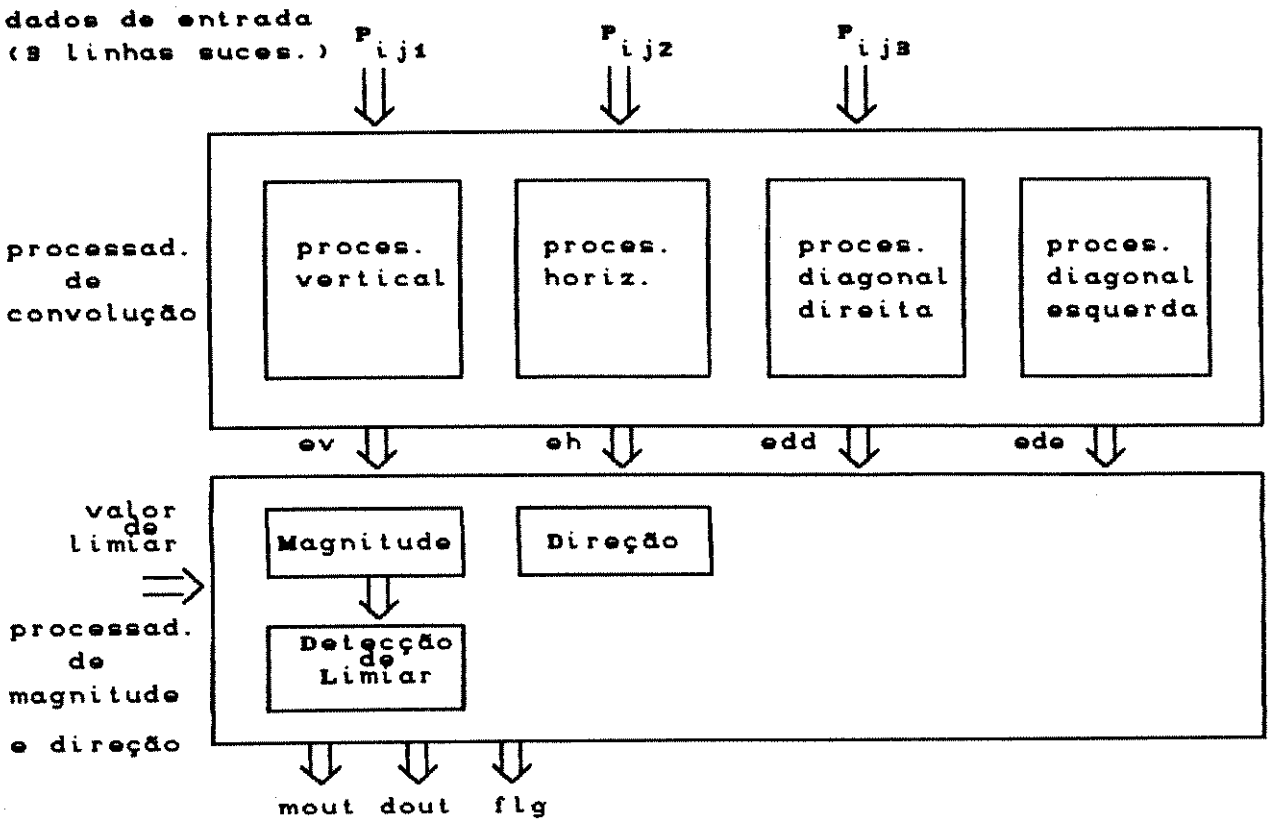
K = é um coeficiente dependente da variação

Direção:

$$\theta = ARCTAN (Ev/Eh)$$

O circuito que calcula os valores do operador de Sobel pode ser visto na figura II.5.

O cálculo da convolução nas direções 45° e 135° acrescenta, segundo o autor, precisão aos valores de magnitude e direção obtidos.



ev - resultado da convolução utilizando "kernel" de detecção de bordas na direção vertical
eh - resultado da convolução utilizando "kernel" de detecção de bordas na direção horizontal
ede - resultado da convolução utilizando "kernel" de detecção de bordas segundo a diagonal a esquerda
edd - resultado da convolução utilizando "kernel" de detecção de bordas segundo a diagonal a direita
mout - dado de magnitude de saída
dout - dado de direção de saída
flg - indicação de que os valores obtidos nas convoluções são menores que o valor-limite

Figura II.5 - Arquitetura de Kanopoulos para Detecção de Bordas

Os estágios do convolutor (figura II.6 a II.8) têm uma estrutura "pipeline" adaptada ao cálculo do operador de Sobel, na medida em que ignoram-se produtos por zero, somam-se dados que irão ser multiplicados por coeficientes iguais e complementam-se dados que iriam ser multiplicados por "-1". Esta adaptação foi feita como opção por um circuito mais rápido e de menor custo.

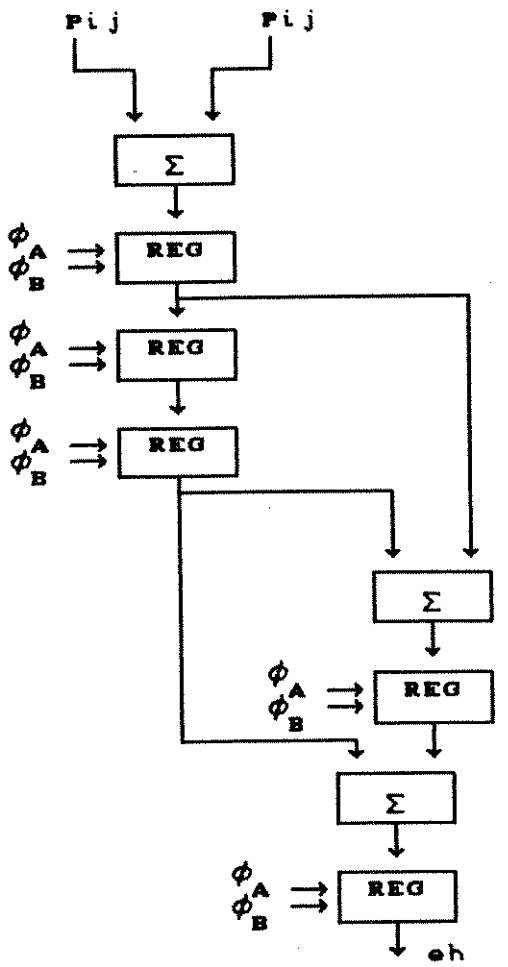


Figura II.6 - Detector de Bordas Segundo a Direção Horizontal

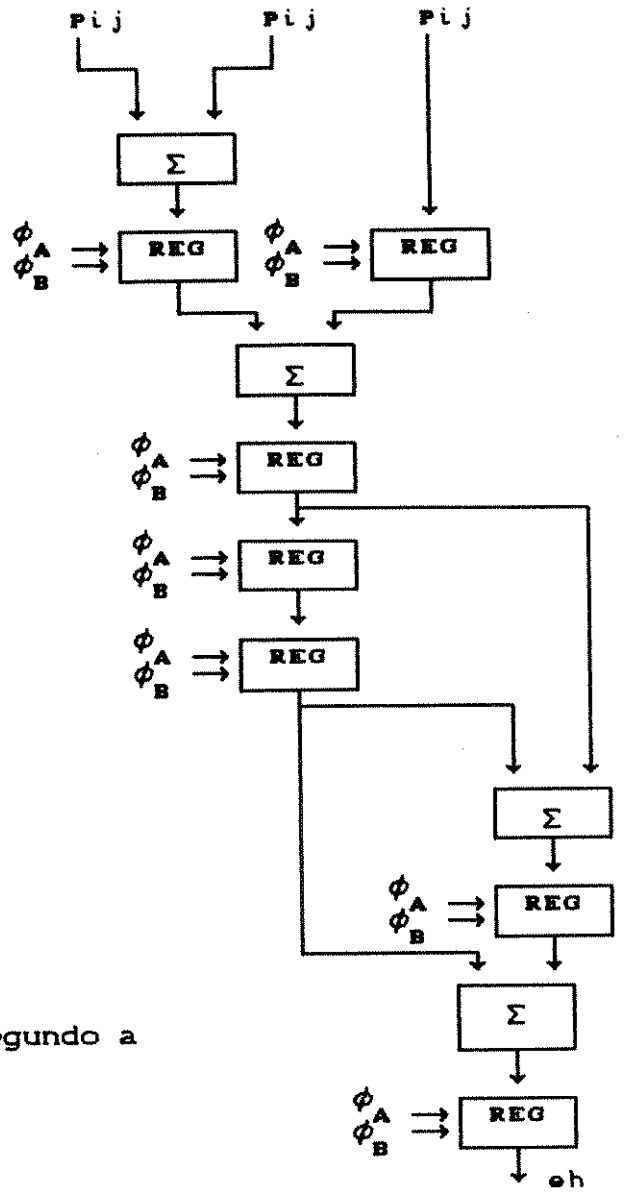


Figura II.7 - Detector de Bordas Segundo a Direção Vertical

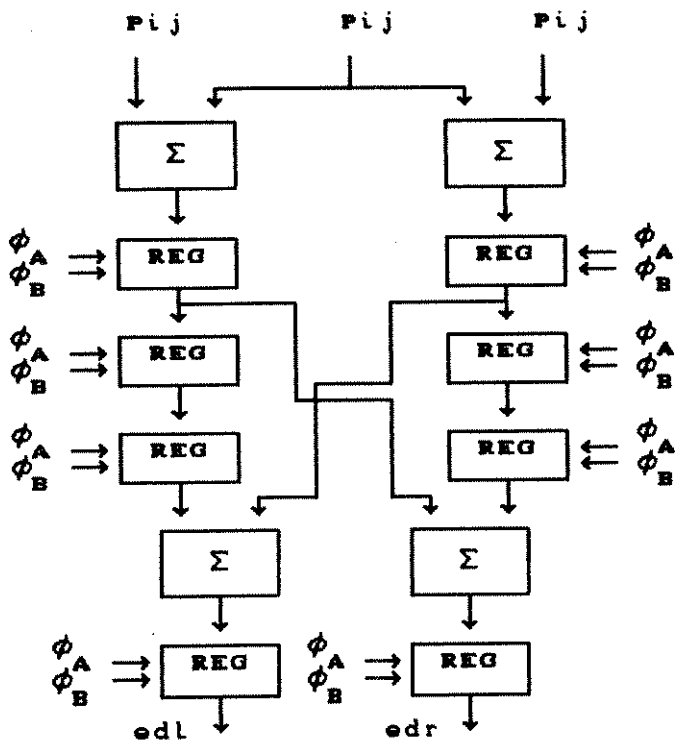


Figura II.8 - Detector de Bordas segundo as diagonais a direita e esquerda

Memórias FIFO externas são usadas de maneira que os dados de três linhas sucessivas estejam sempre presentes nas entradas do processador.

O circuito é síncrono, sendo controlado por duas fases de relógio não coincidentes (ϕa e ϕb). A latência do circuito é de 10 ciclos de relógio, após os quais se tem um resultado (composto de magnitude e direção) a cada ciclo.

Funcionamento

São lidos dados de 3 linhas sucessivas e levados aos 4 processadores do operador de "Sobel" segundo as direções horizontal, vertical, 45 e 135 graus. Uma vez calculados os gradientes de variação segundo as 4 direções os resultados são levados ao processador de magnitude e direção, o qual compara os valores, obtendo o valor máximo. Este resultado será usado para a determinação da direção e cálculo da magnitude do gradiente, a qual será levada à saída somente se o seu valor for maior ou igual ao valor-limite, caso contrário o valor na saída será zero.

O circuito descrito foi implementado em tecnologia CMOS, 2 micra, de dupla camada de metal e funciona a 10 Mhz (2 fases de clock). Nada foi citado na bibliografia sobre as dimensões da pastilha ou sua dissipação.

Como o circuito foi concebido totalmente dedicado ao operador de Sobel, não há possibilidade de uso de "kernels" de convolução diferentes de 3x3.

II.5 - Arquitetura de Aono para Convolução Bidimensional

O circuito proposto por [2] é uma arquitetura composta basicamente de um registrador de imagem local, um bloco aritmético e uma unidade de controle microprogramável como pode ser visto na figura II.8. O registrador de imagem local (Figura II.10) consiste de um conjunto de registradores dispostos em 3 linhas e colunas e um bloco com 3 memórias FIFO. Este bloco tem 3 "ports" de entrada o que permite o carregamento de dados de 3 linhas.

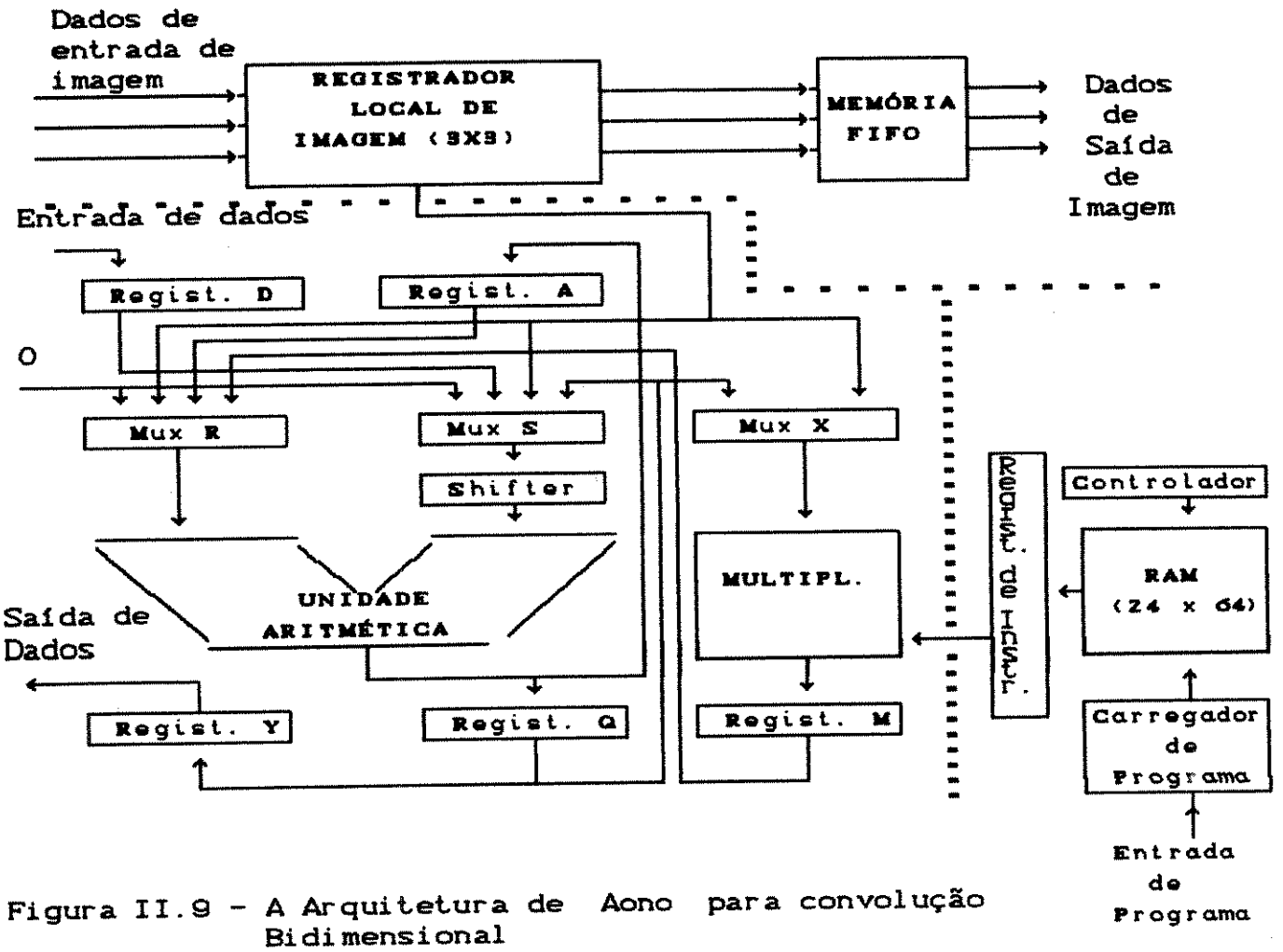


Figura II.9 - A Arquitetura de Aono para convolução Bidimensional

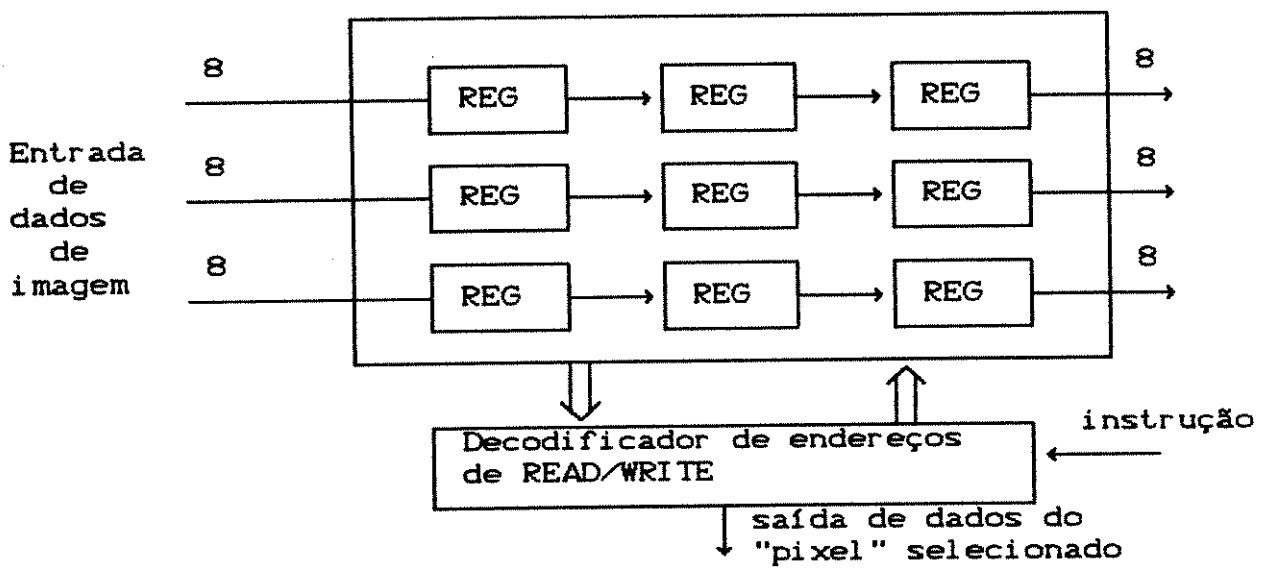


Figura II.10 - Registrador de Imagem Local

O bloco aritmético é composto de uma unidade aritmética de 16 bits, uma matriz multiplicadora, um registrador de deslocamento, multiplexadores e registradores temporários.

A unidade de controle executa um microprograma previamente carregado em memória RAM interna ao circuito. Os comandos são do tipo operação ou salto.

O multiplicador usado no bloco aritmético é uma matriz 8x7 usando soma com CSA (somador com não propagação de bit de transporte). Para a execução da operação de divisão há um registrador de deslocamento controlado por microprograma.

A unidade aritmética foi projetada sem circuitos de antecipação de transporte e pode executar operações de adição e subtração com dados de 16 bits.

Funcionamento

Pixels de imagem de 3 linhas sucessivas são introduzidos no registrador de imagem local, a partir daí os dados são selecionados um a um pelo microprograma e transferidos ao bloco aritmético para serem processados.

A capacidade de processamento do circuito pode ser aumentada através do uso dos recursos de paralelismo e "pipelining" disponíveis.

Paralelismo : usando dois processadores para executar o algoritmo sobre partes da mesma imagem.

Pipelining : expansão do "kernel" de convolução pelo cascateamento de vários processadores.

Comentários

A capacidade de executar um microprograma aumenta a flexibilidade do circuito, já que esse pode passar a executar outros algoritmos.

II.6 - TABELA COMPARATIVA DAS ARQUITETURAS DESCRITAS NESTE CAPÍTULO

A tabela I.1 abaixo resume as características mais relevantes dos circuitos que permitirão uma análise comparativa com a arquitetura a ser adotada neste trabalho. Esta análise será realizada no capítulo VII , onde são também apresentadas as conclusões.

	Arq. II.1	Arq II.2	Arq II.3	Arq II.4
Freq.Rel.	10 MHz	20 MHz	10 MHz (1)	30 MHz
Freq.Img.	15 Quad/s	60 Quad/s	-------	------
Tecnolog.	NMOS - 4u	CMOS-SOS-3u	CMOS - 2u	ECL
Área de Si	35 mm2	------	66.27 mm2	36 mm2
Diss.Pot.	275 mW	------	430 mW	1,6 W
Dimensões do Kernel	3x3	3x3 2x5 1x10	3x3	3x3 a 16x16
Bits coef. dado E/S	7 8 / 10	8 10 / 26	não citado 8 / 12	16 16 / 16
Possib. de cascateam.	não ha	há	não há	há
Possib. de executar outros algoritmos	somente convolução 3x3	somente convolução 3x3	somente operador de Sobel	quaisquer algoritmos implement. com o set de instr.

(1) - Duas fases de relógio

Tabela I.1 - Características dos circuitos

CAPÍTULO III

A Arquitetura A Ser Implementada Neste Trabalho

III.1 - Introdução

Neste capítulo será descrita a arquitetura que serviu de base para a implementação do circuito [4] realizada neste trabalho. Inicialmente será apresentado um diagrama em blocos com a idéia básica da arquitetura a qual servirá para descrever as principais partes do circuito e o seu funcionamento.

A implementação com circuitos comerciais será comentada ao final deste capítulo.

III.2 - A Arquitetura Básica

A arquitetura proposta em [4] (Figura III.1) é um circuito "pipeline" composto de:

- Um registrador de deslocamento de entrada
- Nove multiplicadores 8x4 (cada um dos blocos multiplicadores da figura representa na verdade três multiplicadores 8X4)
- Três somadores
- Duas memórias FIFO (memórias para as quais o primeiro dado de entrada é o primeiro a sair).
- Um registrador de deslocamento de saída
- Registradores internos aos blocos, os quais "separam" os estágios do circuito "pipeline"

Nesta arquitetura , os tempos de atraso de cada estágio do "pipeline" são mantidos abaixo de meio período de relógio, quando os dados são carregados pelos registradores que separam os estágios do "pipeline".

Segundo o trabalho uma frequência de relógio de 10MHz seria suficiente para a operação do circuito, assim o máximo tempo de atraso por estágio deve ser de:

$$T_{máx} = 50\ ns$$

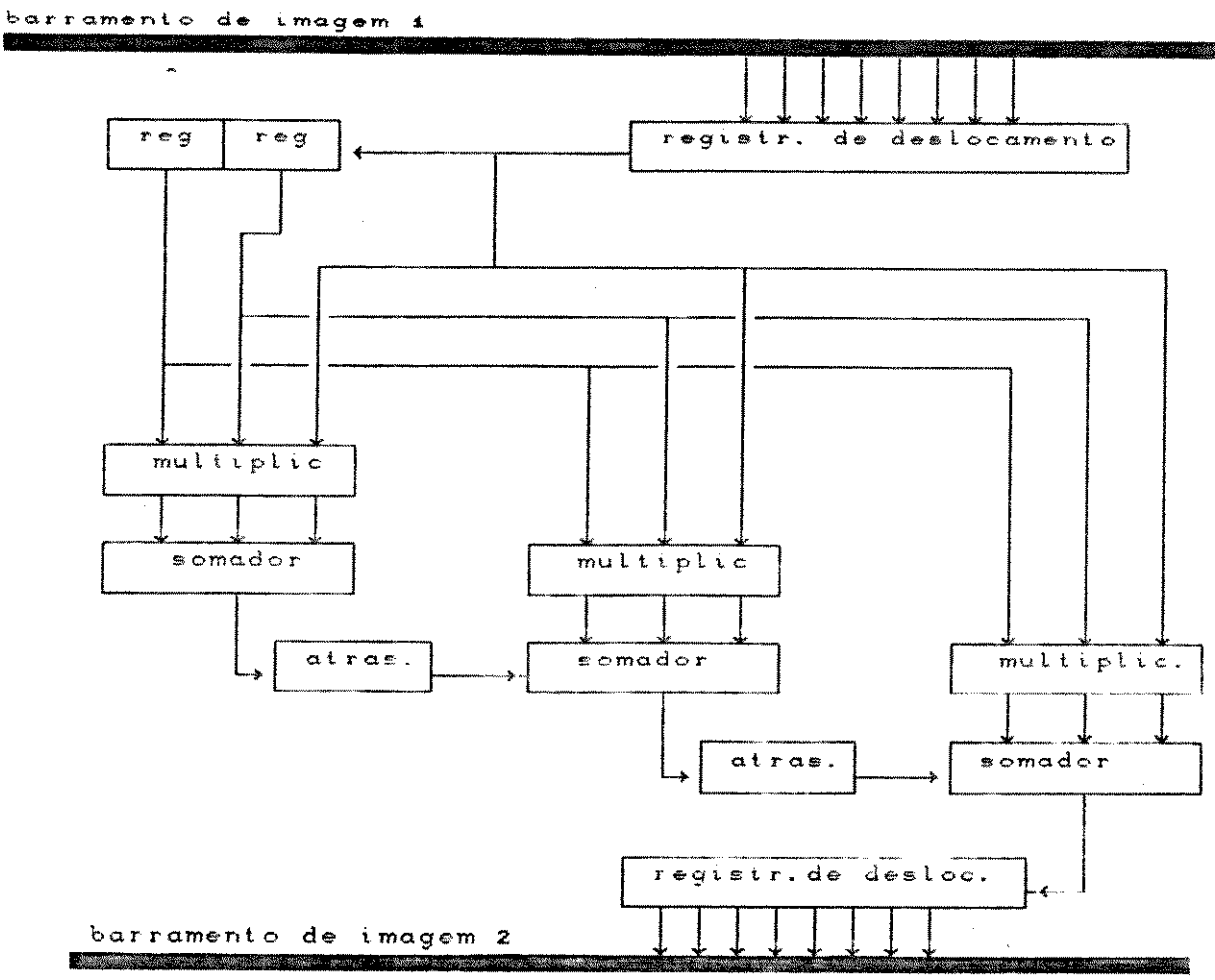


Figura III.1 - Arquitetura original proposta

O funcionamento do circuito pode ser resumidamente explicado :

O registrador de deslocamento recebe dados do barramento em blocos de 8 bytes e os transfere aos circuitos multiplicadores através de registradores. Assim, o circuito de convolução propriamente dito trabalha com um barramento de 8 bits, o que é necessário para que cada dado seja multiplicado pelos 9 coeficientes do "kernel" de convolução o que efetivamente ocorre à medida em que este dado vai sendo deslocado nos registradores de deslocamento de entrada.

Os circuitos multiplicadores executam o produto de três coeficientes do "kernel" de convolução por três "pixels" adjacentes. Como o circuito tem três multiplicadores, todos os 9 coeficientes do "kernel" de convolução serão multiplicados pelos respectivos "pixels" transferidos 3 a 3 a cada ciclo de clock.

Estes produtos são somados 3 a 3 pelos somadores e depois armazenados em memórias FIFO de 512 posições, sincronizando o fluxo de dados de maneira que, após o tempo de latência da máquina (no qual ainda não se tem um resultado na saída, o que equivaleria ao tempo para "preencher" com dados válidos todos os estágios do circuito) tenhamos a cada ciclo de relógio o resultado da convolução sobre um "pixel" de imagem.

Assim, explorando o paralelismo temporal e a triplicação dos somadores e multiplicadores se obteve uma arquitetura capaz de executar processamnto em tempo real.

A leitura de dados, bem como a execução de operações são comandadas pelo sinal de relógio , que controla a transferência de dados entre um estágio e outro.

III.3 - A IMPLEMENTAÇÃO FÍSICA DO CIRCUITO

A implementação física deste circuito, foi realizada com componentes comerciais, o que forçou a introdução de simplificações no mesmo (tais como a substituição de operações de multiplicação por operações de deslocamento de bits, possibilitada pelo fato de que os coeficientes dos "kernel"s para o operador de Sobel são do tipo 2^n que resultaram em circuitos mais rápidos, porém restringindo possibilidades de aplicação do módulo de convolução.

Assim, partindo-se desta arquitetura decidiu-se pela adaptação da arquitetura inicial a uma implementação como um circuito integrado, eliminando as limitações citadas anteriormente e adiconando as vantagens de melhor desempenho, baixa dissipação e menor custo.

CAPÍTULO IV

OPÇÕES TECNOLÓGICAS

IV.1 - INTRODUÇÃO

Neste projeto, haviam duas opções de tecnologia de silício , CMOS e IIL.

Decidiu-se pela tecnologia CMOS por esta permitir maior densidade de integração e menor dissipação de potência.

Uma vez definida a tecnologia e a configuração lógica em alto nível é importante que se opte por um estilo de projeto, ou seja, uma "opção tecnológica de implementação física", já que a alternativa influenciará na edição do layout e no melhor aproveitamento da área de silício disponível.

Existem atualmente várias possibilidades de estilo de projeto de circuitos integrados, entre estas dispõe-se de configurações que resultam na diminuição do tempo de projeto em detrimento ao aproveitamento do silício (e muitas vezes do desempenho do circuito), como é o caso de implementação por matrizes de células (gate array) ou células padrão (standard cells). Existem ainda tecnologias que implicam em um alto grau de aproveitamento do silício e melhor desempenho se comparados com os "gate arrays", implicando em um maior tempo de projeto, como é o caso de implementação por macrocélulas, desenvolvidas para a aplicação (macros customizadas) ou de circuitos totalmente voltados a aplicação (full custom).

As várias alternativas são discutidas no ítem IV.2, no ítem IV.3 as alternativas são comparadas e no ítem IV.4 são relatadas as alternativas escolhidas para a implementação deste trabalho.

IV.2 - ESTILOS DE PROJETO

As opções de implementação mais comuns podem ser segundo [6] resumidas por :

- .IV.2.1 - Gate Array
- .IV.2.2 - Master Image
- .IV.2.3 - Standard Cells
- .IV.2.4 - Macros Customizadas
- .IV.2.5 - Circuitos Full Custom

Cresce o grau de Customização

A classificação nas opções anteriormente citadas leva em conta o que se chama "customização" de um circuito, ou seja, o quanto cada parte é feita como um projeto único em vez de se definir macroblocos possíveis de ser alocados em várias partes do circuito.

IV.2.1 - GATE ARRAY

Nesta opção de implementação VLSI parte-se de um conjunto de transistores idênticos dispostos segundo uma matriz (ver fig. IV.1). Este conjunto é comumente chamado de "background" de um "gate array". Estes transistores serão posteriormente interconectados com a utilização de ferramentas adequadas (CAD), num processo que comumente se denomina "roteamento" para formar os circuitos e as funções lógicas desejadas.

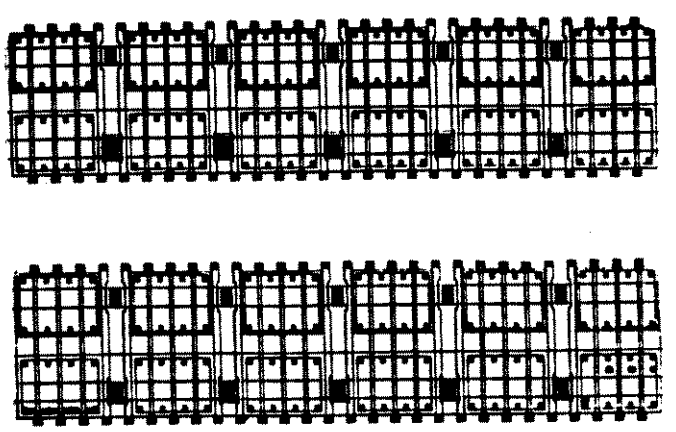


Figura IV.1 - Implementação de "layout" segundo o estilo "gate array"

É importante ressaltar que o mesmo "background" pode ser usado por vários projetos, desta maneira podem ser fabricados vários "wafers" com o mesmo "background" e estes podem ser armazenados para posteriormente receberem a camada de metalização adequada para a formação dos circuitos desejados. Isto, além de ocasionar uma diminuição bastante significativa no tempo de fabricação implica em uma redução de custos de fabricação de máscaras já que só um nível de máscara difere de projeto para projeto.

Neste tipo de implementação a "customização" se dá pela criação das funções lógicas, porém a implementação física dos circuitos (alocação das portas lógicas e sua interconexão) é feita totalmente pelo programa de roteamento das células. Este programa de roteamento define a configuração dos níveis de metal (dois níveis de metal é o caso mais comum), que servem para conectar os transistores-padrão para formar as portas lógicas e as interconexões entre elas, a forma final da camada de metal é diferente de um projeto para outro. Como em "gate array" usam-se transistores de tamanho e especificações padrão não há customização de circuitos ou de transistores individualmente.

IV.2.2 - Implementação por "Master Image"

Esta forma de implementação é composta de conjuntos de transistores de tamanhos diferentes [6] para se obter diferentes capacidades de corrente. Da mesma forma que os "gate arrays", um mesmo "background" pode ser utilizado em vários projetos, porém para se obter células com diferentes capacidades de corrente deverá haver um conjunto de interconexões preferencial para a formação dos circuitos lógicos.

A antecipação das interconexões que deverão ser realizadas implica num posicionamento preferencial de dispositivos resultando numa implementação otimizada de algumas funções em particular do que aquela que seria obtida usando "gate arrays".

Não há exemplos de aplicação comercial citados na literatura.

IV.2.3 - Implementação por "Standard Cells"

Neste tipo de implementação os outros níveis de máscara, além de metal e contato, que eram os únicos que podiam ser definidos pelo projetista em "gate array" devem ser "customizados", assim todos os transistores de uma célula ("gate" lógico) são implementados de maneira a maximizar o aproveitamento de área de silício. Esta forma de implementação implica em que, uma mesma porta lógica terá a mesma implementação física em qualquer parte do circuito onde seja necessária e possivelmente em outros circuitos onde esta possa ser aproveitada.

Para a implementação física se fornece à um programa de posicionamento e interconexão de células a descrição lógica do circuito em formato apropriado, sendo todo o processo feito automaticamente. Para isto o layout das células deve permitir a justaposição das mesmas formando linhas de portas lógicas adjacentes dispostas paralelamente entre as quais é feita a conexão entre as portas lógicas para formar o circuito final.

IV.2.4 - Macros Customizadas

Nesta forma de implementação são criadas funções lógicas mais complexas , como por exemplo ALU's, memórias, etc, sendo implementadas como um bloco físico.

Estes macroblocos são implementados fisicamente e testados, assim, desenvolve-se uma biblioteca de macroblocos de formatos regulares que podem ser utilizados em vários projetos.

Neste estilo de projeto as interfaces de sinais entre os blocos e mesmo a disposição dos blocos na pastilha devem ser bem estabelecidas a princípio.

Deste tipo de implementação resultam layouts bastante regulares, com densidade e desempenho geralmente superiores a dos circuitos obtidos pelas alternativas de projeto apresentadas anteriormente. Um exemplo de "layout" implementado neste estilo é mostrado na figura IV.2.

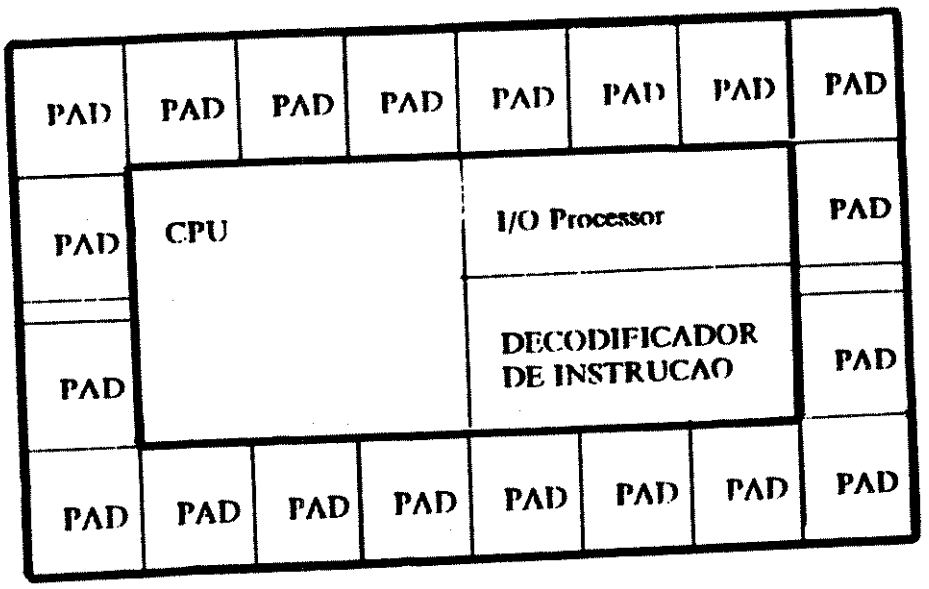


Figura IV.2 - Layout Implementado com Macros Customizadas

IV.2.5 - Circuitos Totalmente Dedicados (Full Custom)

Neste tipo de implementação os circuitos são projetados de maneira única visando o máximo aproveitamento da área de silício e aumento do desempenho dos circuitos.

Cada porta lógica é dimensionada levando em conta a velocidade, o "fanout", a dissipação, a carga representada pelas interconexões, etc.

Inicialmente é feito o "floor planning" de toda a pastilha levando em conta a função lógica de cada parte e seu acesso as células de interface com o mundo exterior. Os blocos podem ter formas irregulares, ao contrário da implementação por "macros" customizadas.

As capacitâncias obtidas são tipicamente menores que as obtidas pelos métodos anteriores, assim o desempenho dos circuitos é normalmente superior ao obtido por métodos com um menor grau de customização. Os "layouts" desenvolvidos neste estilo são os que permitem o melhor aproveitamento da área de silício disponível, porém, devido às etapas de dimensionamento das células e geração de "layout" o tempo que projetos feitos neste estilo consomem é muito superior ao de todas as alternativas superiores.

Normalmente este estilo só é utilizado em projetos que necessitem ter baixo custo por pastilha, ou que a quantidade de circuitos integrados a ser vendida seja muito grande.

Um exemplo de projeto "full custom" pode ser visto na figura IV.3.

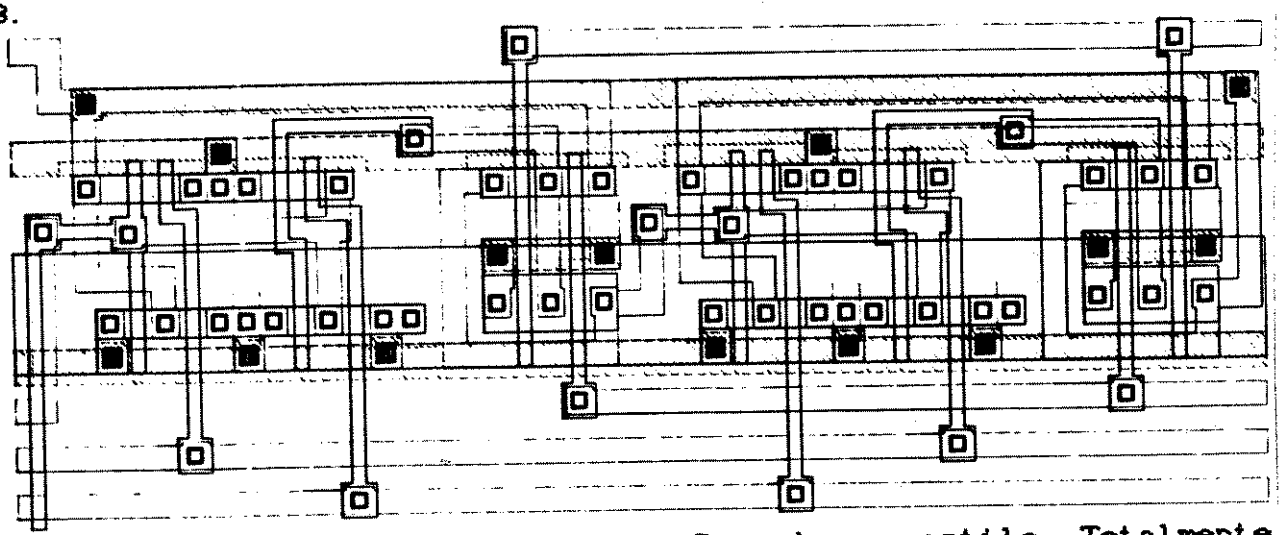


Figura IV.3 - Layout Desenvolvido Segundo o estilo Totalmente Dedicado

IV.3 - Comparação Entre as Alternativas de Estilos de Projeto Vistas Anteriormente

O aumento do nível de "customização" ou personalização produz melhor aproveitamento de silício, porém levam a tempos de projeto e fabricação maiores.

Na opção "gate-array" a personalização dos circuitos só se dá pelo nível de metal (que difere para dois circuitos com implementação lógica diferente apesar de o "background" ser o mesmo) que estabelece conexões entre os transistores para a formação dos "gates" lógicos.

A opção por "gate array" implica em aumento da rapidez no processo de fabricação, pois menos etapas devem ser realizadas, além do fato de que as algumas das máscaras do processo de fabricação poderão ser reutilizadas em outros circuitos lógicos diferentes.

Caso se utilize um processo de fabricação por "eletron beam" não tem sentido falar em reutilização de máscaras já que estas não são utilizadas neste processo, onde a sensibilização do resiste é feita diretamente no silício pelo feixe de elétrons.

Em contraposição aos "gate arrays", circuitos mais "customizados" produzem layouts com níveis de máscaras completamente diferentes resultando em longos tempos de projeto e fabricação e menor possibilidade de reutilização de níveis de máscara em outros projetos. Uma alternativas intermediária seria, por exemplo, o uso de "standard cells" que oferecem diferentes compromissos entre tempos de projeto e de fabricação e aproveitamento de silício e desempenho.

Atualmente vários estilos de projetos vem sendo usados num mesmo projeto, assim, um projeto pode ter blocos puramente lógicos implementados como "gate array" ou "standard cells" e outros blocos, como por exemplo blocos de memória, implementados como macros customizadas e ainda blocos analógicos implementados como circuitos "full custom" visando obter um compromisso aceitável entre tempo de projeto e aproveitamento de silício.

Um fator importante a ressaltar é que projetos com maior grau de "customização" tendem a ter um custo maior pois consome-se mais tempo com simulações elétricas e com a elaboração de layout de cada bloco além da necessidade de pessoal e programas de edição gráfica.

A escolha de uma ou outra alternativa deve ser analisada para cada caso.

IV.4 - As Opções Utilizadas Neste Projeto

Neste trabalho decidiu-se utilizar basicamente o estilo "standard cell" com algumas macros customizadas para a implementação do protótipo, como uma alternativa intermediária que permitisse rapidez de projeto e aproveitamento do silício superior ao dos "gate arrays" para a implementação do protótipo.

Capítulo V

A Arquitetura a Ser Implementada Neste Trabalho

V.1 - INTRODUÇÃO

A análise da arquitetura apresentada no capítulo III leva a estudos que resultarão em um esquema lógico implementável como um circuito integrado, aproveitando as vantagens decorrentes deste tipo de implementação.

Algumas das decisões aqui tomadas estão baseadas em características lógico/elétricas e tem como objetivo se utilizar estruturas mais adequadas a uma implementação como circuito integrado, outras serão baseadas nos resultados de simulações e outras impostas pela limitação da área de silício disponível.

V.2 - ASPECTOS GERAIS

V.2.1 - VELOCIDADE

Este sistema foi previsto para operar no pior caso com uma taxa de amostragem de até 60 quadros por segundo (taxa de regeneração de 60 Hz) com varredura não entrelaçada de modo que a imagem possa ser processada à medida em que vai sendo digitalizada sem necessidade de armazenamento intermediário.

Uma imagem de 262144 pixels "512 x 512" deve ser processada em 16,67 ms (1/60 s), com um tempo de retraço horizontal que é de cerca de 10% do período total da linha e tempo de retraço vertical normalmente menor que 5% do tempo de um ciclo completo [10].
Logo teremos:

$$\text{Período de uma linha} = \frac{(1/60)*(1 - 0,05)}{512} * 90\% = 27,83\ \mu s$$

$$\text{Período de um "pixel"} = \frac{27,83 * 10^{-6}}{512} = 54,35\ ns$$

Resultando numa frequência para "pixel" de 18,4 MHz.

Assim a taxa em que os "pixels" deverão ser fornecidos ao processador deve ser de 18,4 Mbytes/s.

V.2.2 - CARGA

Decidiu-se projetar um circuito que deverá interfacear somente com lógica CMOS, que necessitam de baixa corrente de entrada. Esta decisão traz como vantagem adicional a possibilidade do uso de "buffers" de saída pequenos em comparação aos que se teria que ter para interfacear com outras famílias de circuitos, por exemplo TTL, ECL, etc. A carga típica para as famílias CMOS comerciais é de 60 portas CMOS.

V.2.3 - TEMPERATURA DE OPERAÇÃO

A temperatura de operação deve ser a faixa comercial, de 0 a 70°C.

V.2.4 - ALIMENTAÇÃO

O circuito deve operar com tensões de alimentação de 0 e 5V por serem valores padrões, estarem disponíveis no computador hospedeiro e estarem abaixo da máxima tensão de alimentação suportada pela tecnologia.

V.2.5 - MARGENS DE RUÍDO

Decidiu-se adotar as margens de ruído igual à da maioria das famílias CMOS disponíveis no mercado para tensão de alimentação de +5V [15] :

Valores: Para nível lógico "0" : 0,0 a 1,0 V
Para nível lógico "1" : 4,0 a 5,0 V.

V.2.6 - TESTABILIDADE

Como a área de silício para a implementação do protótipo é pequena, decidiu-se não realizar um projeto voltado à testabilidade, pois isso acarretaria gasto a mais de área de silício.

V.3 - Implementação Lógica/Elétrica

A primeira decisão tomada do ponto de vista de implementação lógica foi a de dimensionar o número de bits do coeficiente do "kernel" de convolução.

Para a aplicação proposta, em que os valores dos coeficientes variam de -2 a +2, bastariam 2 bits para a representação do módulo do coeficiente mais um bit de sinal. Observa-se porém que o circuito proposto poderia executar outras operações que não fossem apenas a detecção de bordas e utilizar valores para os coeficientes maiores que o limite citado, razão pela qual se optou por utilizar coeficientes de 4 bits.

O dado é de 8 bits, valor que permite a representação em 256 níveis de cinza.

Sabendo-se o número de "bits" do coeficiente e o dado da arquitetura proposta no capítulo III ,reapresenta-se na figura V.1 a mesma arquitetura com algumas modificações, ou seja com os números de bits em cada barramento de interface entre os blocos e também com alterações que refletem opções de implementação lógica, a saber :

- O uso de complementadores à saída dos multiplicadores transformando os números da forma "sinal-magnitude" (1 bit de sinal + 13 de dado) para a forma "complemento de 2" (13 bits).
- A substituição dos registradores de deslocamento de entrada por um circuito de multiplexação dos dados de entrada.
- O uso de complementadores de saída para converterem o resultado final da forma "complemento de 2" para a forma "sinal-magnitude".

Esta figura servirá de base para a definição lógica de cada bloco.

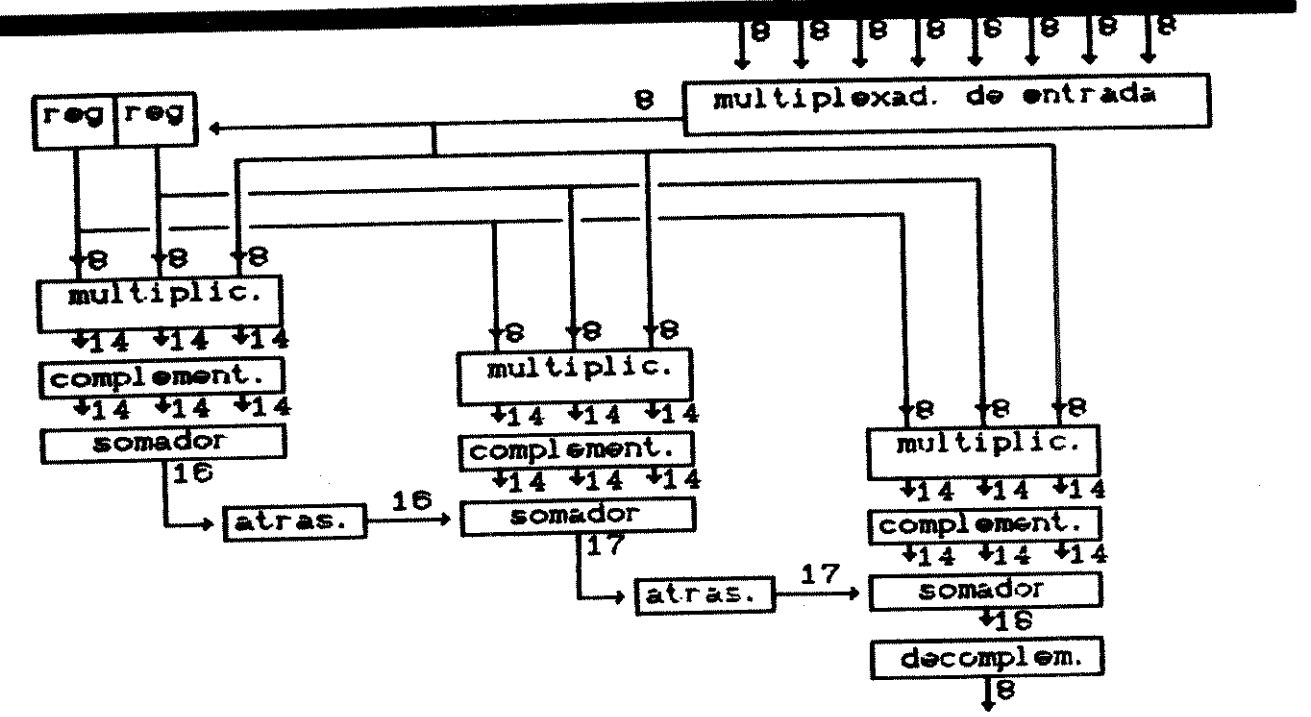


Figura V.1 - Arquitetura a ser implementada

V.3.1 - Análise de Propagação de Erros no Circuito

Quanto a quantização do nível de cinza pela câmera, pode ocorrer erro de até 1/2 nível de cinza, já que um ponto com nível de cinza compreendido entre dois dos valores que a representação permite (num total de 256) será aproximado para um destes extremos.

Após a amostragem, quantização e armazenamento serão executadas várias operações as quais foram explicadas no capítulo III e são listadas a seguir acompanhadas dos respectivos erros introduzidos para o pior caso:

1 - Produto por um coeficiente de 4 bits.

Desde que o maior valor para um coeficiente seja 15 , qualquer erro na quantização poderá ser multiplicado por este valor.

2 - Soma de 3 resultados de produtos (primeira soma).

Supondo que no pior caso o erro de quantização tenha ocorrido e tenha sido multiplicado por 15 para os 3 produtos, este será multiplicado por 3.

3 - Soma dos resultados da primeira soma (segunda soma).

Como são somados 3 fatores o erro será multiplicado por 3 no pior caso.

4 - Erro de Normalização de 18 para 8 bits

Equivale à divisão por 2^{10}

Assim o erro final será no máximo :

$$Erro\ Final = 1/2 * 15 * 3 * 3 * 1/2^{10} = 0,066$$

V.3.2 - INTERFACE COM O SISTEMA

Após o "power on reset" ocorre um ciclo de carregamento dos coeficientes do "kernel" de convolução que serão fornecidos pelo "host", o qual sinaliza ao circuito o envio de coeficientes através do sinal COEF. Uma vez carregados os 9 coeficientes, a entrada de dados é habilitada através do sinal DADO enviado pelo "host"e, a cada ciclo de relógio, um dado é levado ao processador de convolução.

Após o tempo de latência da máquina (ou seja, o tempo em que todos os estágios estão preenchidos com dados válidos), tem-se um resultado a cada ciclo de relógio até que o sinal DADO vá a zero ou que se indique carga de novo coeficiente, ou que o circuito seja "resetado".

A porta de entrada de dados da arquitetura apresentada no capítulo III (ver figura III.1) é de 8 bits; como o barramento de dados é de 64 bits (8 bytes) é necessário algum tipo de multiplexação de dados.

A saída de dados do circuito completo será feita em 14 bits, de modo que não se tenha que fazer nenhum arredondamento ou truncamento de bits intermediários durante o cálculo.

O resultado final a ser levado ao controlador de vídeo ou armazenado em memória será normalizado para 8 bits no estágio final. Já que tanto a imagem de entrada quanto a de saída são codificadas em 8 bits, esta normalização ocorrerá pelo truncamento dos bits menos significativos.

V.3.3 - Diagrama em Blocos

Os blocos básicos identificados na arquitetura descrita no capítulo III serão estudados nesta ordem:

V.3.3.1 - Lógica de Controle
V.3.3.2 - Circuito de Multiplexação dos Dados de Entrada
V.3.3.3 - Somadores
V.3.3.4 - Multiplicadores
V.3.3.5 - Memórias
V.3.3.6 - Complementadores
V.3.3.7 - Decomplementadores

V.3.3.1 - Lógica de Controle

Foi criada uma lógica de controle para habilitar o carregamento de coeficientes e entrada de dados no circuito processador; esta lógica pode ser vista no circuito da figura V.2 .

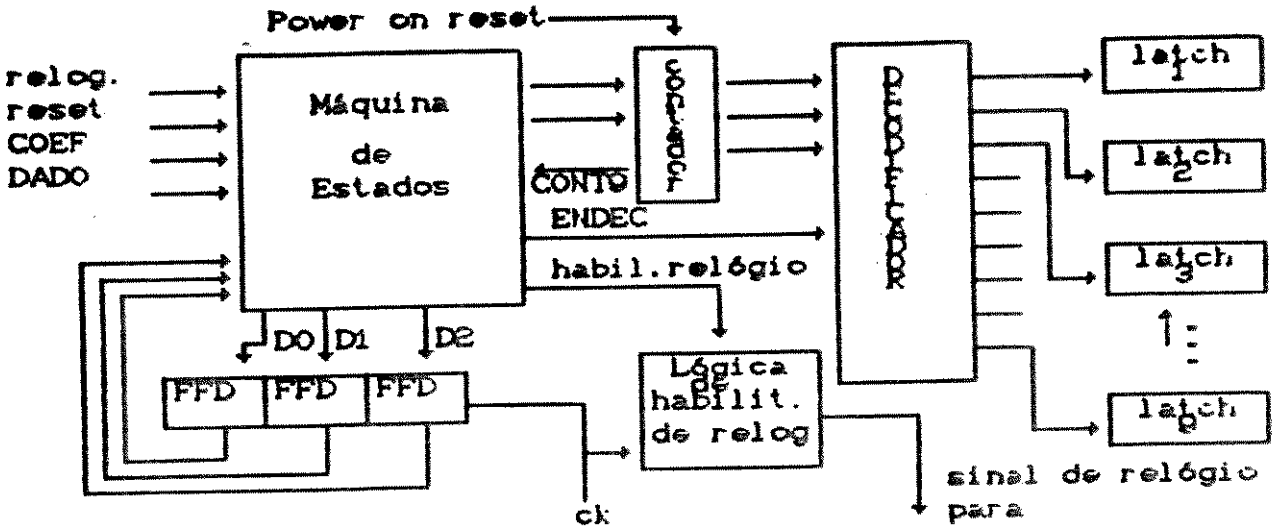


Figura V.2- Lógica de Controle

A lógica de controle é composta de uma máquina de estados, um contador, um decodificador, uma lógica de habilitação de relógio para os dados e quatro registradores para armazenamento do estado atual da máquina de estados.

Após o "power on reset" (onde todos os contadores e registradores são inicializados com zero lógico) a máquina de estados permanece num ciclo de espera do sinal COEF, que indica que o "host" quer enviar um coeficiente para o circuito. A máquina de estados comanda então o incremento do contador, sendo o número à saída do contador levado ao decodificador, acionando o relógio e fazendo com que o primeiro registrador de coeficiente (de um total de nove, associados a 9 multiplicadores respectivamente) leia o coeficiente presente no barramento de coeficientes. A máquina de estados permanece então aguardando o sinal COEF ir a zero lógico, indicando que acabou a transferência do primeiro coeficiente. Ocorrerão mais oito leituras de coeficientes, quando se terá o "kernel" de convolução completo armazenado no circuito.

Uma vez lidos todos os coeficientes, o que é indicado pelo sinal CONTO gerado pelo contador, a máquina de estados aguarda o sinal DADO ir a "1" lógico, indicando ao circuito que haverá envio de dados a ele, quando então habilitará o sinal de relógio para todos os registradores do processador de convolução permitindo assim o processamento.

O processamento continua até que o sinal DADO vá a zero lógico, quando o circuito passa a aguardar novo coeficiente ou novo dado.

V.3.2 - Circuito de Multiplexação de Dados de Entrada

Como se pode observar na figura V.1 os "pixels" a ser processados são provenientes de um barramento de 64 bits mas o circuito de convolução propriamente dito trabalha com dados de 8 bits (cada um representando um "pixel" de imagem), assim é necessário um circuito de multiplexação para realizar a interface entre o barramento e o circuito.

Duas alternativas principais foram imaginadas para alinhamento dos dados de entrada de 64 para 8 bits, a primeira consistindo de um arranjo de 8 registradores de deslocamento com carregamento paralelo de 8 bits e saída de 1 bit (figura V.3) com o mesmo sinal de relógio e a segunda composta de um registrador de 64 bits e um multiplexador de 64 bits de entrada e um de saída (Figura V.4).

Ambas as alternativas serão analisadas a seguir:

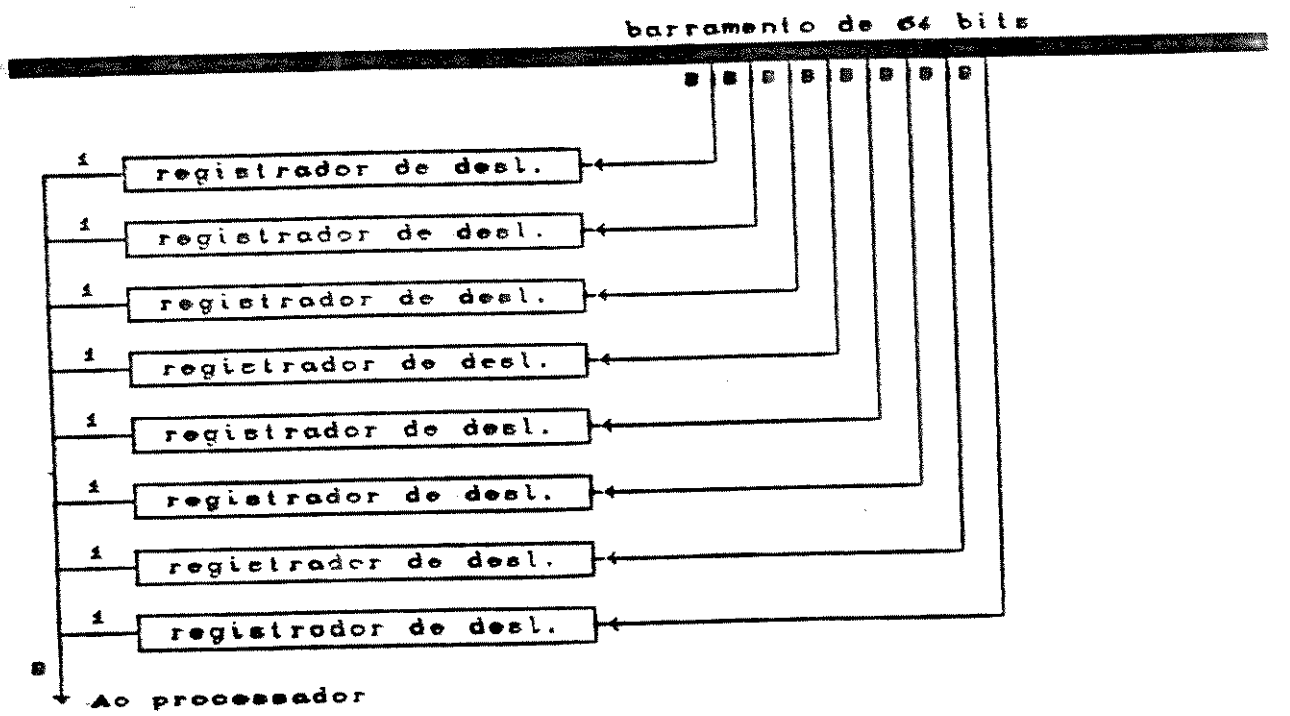


Figura V.3 - Circuito de Alinhamento com Arranjo de Registradores de Deslocamento.

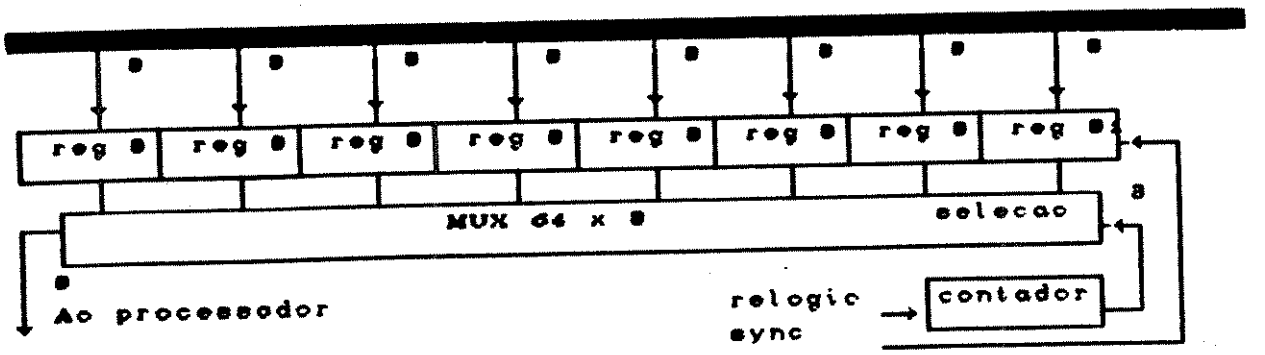


Fig V.4 - Circuito de Alinhamento com registrador de 64 bits e multiplexador

V.3.3.2.1 - Circuito de Multiplexação de Dados com Arranjo de Registradores de Deslocamento

O circuito, que serve para introduzir dados de 8 bits a partir de um barramento de 64 bits mantendo a taxa de transferência necessária para garantir processamento em tempo real, consiste basicamente de 8 registradores de deslocamento de 8 bits trabalhando à frequência de clock, ou seja 18,4 MHz.

Cada registrador de deslocamento (composto de flip-flops tipo D) permite carregamento paralelo de 8 bits através de multiplexadores 2:1, como pode ser visto na figura V.5.

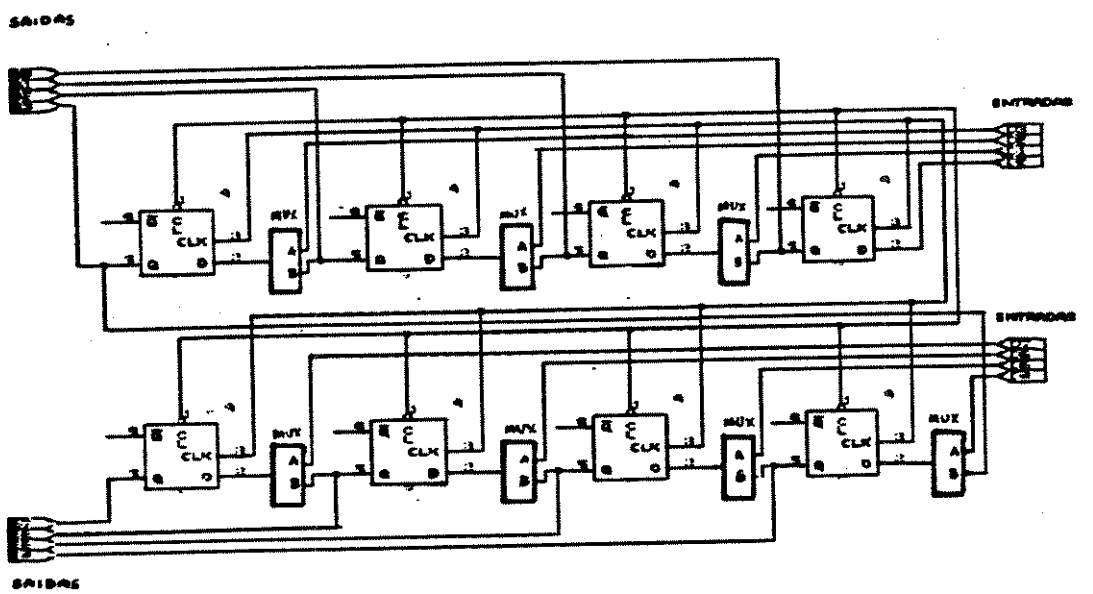


Figura V.5 - Registrador de deslocamento utilizado no circuito de multiplexação de dados com arranjo de registradores de deslocamento

FUNCIONAMENTO

Assim que o dado de 64 bits (8 "pixels") estiver presente no barramento, o sinal CARREGA(0)/DESLOCA(1) vai a zero lógico, habilitando a carga de um novo dado de 64 bits; esta se dá armazenando os bits de mesma ordem de todos os "pixels" nos mesmos registradores de deslocamento.

Assim, supondo que se tenha 8 "pixels" : A, B, C, D, E, F, G, H tais que:

$A = A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$ $\qquad$ $E = E_7\ E_6\ E_5\ E_4\ E_3\ E_2\ E_1\ E_0$

$B = B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1\ B_0$ $\qquad$ $F = F_7\ F_6\ F_5\ F_4\ F_3\ F_2\ F_1\ F_0$

$C = C_7\ C_6\ C_5\ C_4\ C_3\ C_2\ C_1\ C_0$ $\qquad$ $G = G_7\ G_6\ G_5\ G_4\ G_3\ G_2\ G_1\ G_0$

$D = D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$ $\qquad$ $H = H_7\ H_6\ H_5\ H_4\ H_3\ H_2\ H_1\ H_0$

No primeiro registrador de deslocamento serão armazenados:
$A_7\ B_7\ C_7\ D_7\ E_7\ F_7\ G_7\ H_7$

No segundo registrador de deslocamento serão armazenados:
$A_6\ B_6\ C_6\ D_6\ E_6\ F_6\ G_6\ H_6$

Uma vez carregados todos os registradores de deslocamento, um "pixel" estará à saída do circuito multiplexador a cada ciclo de relógio.

Após a transferência de 8 "pixels" ao processador de convolução é realizada nova leitura do barramento de 64 bits.

Deve haver sincronismo entre a entrada do último "pixel" no processador de convolução e a leitura de um novo dado; este pode ser obtido através do sinal DADO da máquina de estados do circuito controlador. Se o sinal DADO for levado a zero lógico a máquina desabilita o sinal de relógio aos registradores do processador de convolução interrompendo seu funcionamento até que o sinal DADO vá novamente para 1 lógico. No tempo em que DADO = 0 pode-se carregar novos 64 bits (8 pixels) nos registradores de deslocamento.

Como a frequência de operação é de 18,4 MHz consegue-se deslocar os "pixels" no registrador de deslocamento em 434,8ns, assim os dados devem estar presentes no barramento de 64 bits a cada 434,8 ns, o que resulta numa taxa de transferência de 18,4 Mbytes/s.

V.3.3.2.2 - Circuito de Alinhamento com Multiplexador e Registrador de 64 Bits

Este circuito é constituído por 8 registradores de 8 bits e um multiplexador que seleciona um "pixel" de cada vez comandado por um contador de três bits.

O registrador de 8 bits pode ser visto na figura V.6.

O circuito multiplexador pode ser visto na figura V.7.

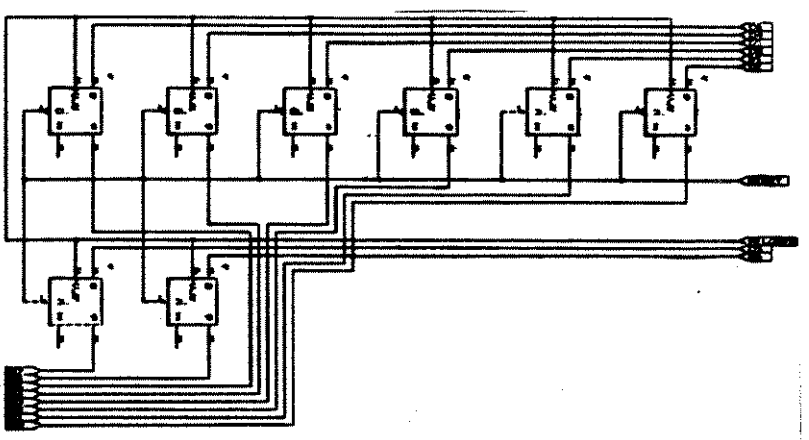


Figura V.6 - Registrador de 8 bits

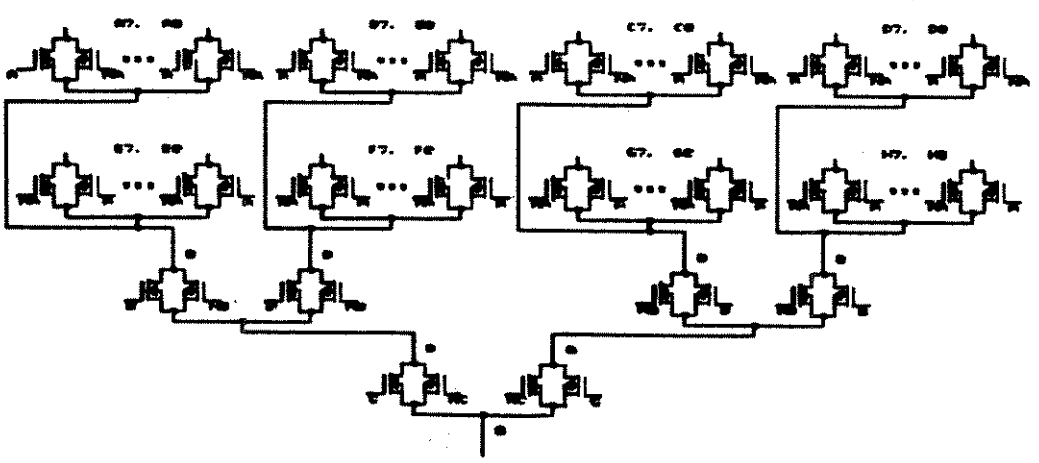


Figura V.7 - Multiplexador usado no circuito de multiplexação composto de um registrador de 64 bits e um multiplexador

Do ponto de vista de uma implementação como circuito integrado são importantes o número de portas lógicas e as dimensões do layout final.

O contador é síncrono, comandado pelo mesmo sinal de relógio do processador (Figura V.8).

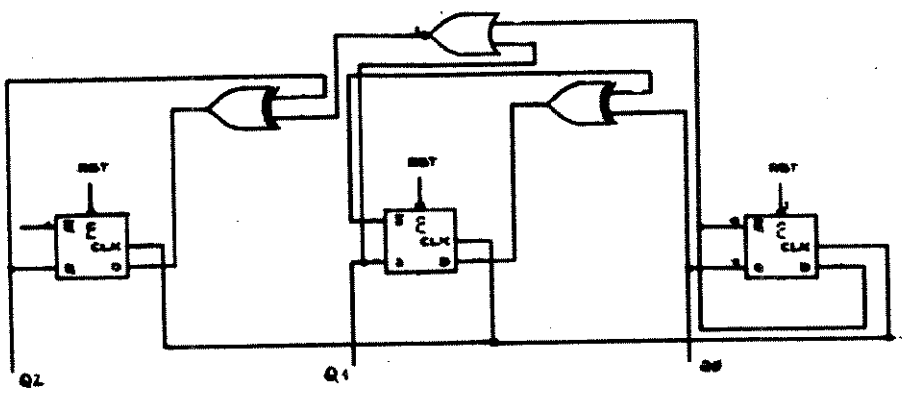


Figura V.8 - Contador Síncrono de 3 bits

V.3.3.2.3 - Lógica Envolvida nos Circuitos:

Circuito com Arranjo de registradores de Deslocamento

Cada registrador de deslocamento pode ser facilmente implementado como uma cascata de flip-flops tipo D.

Uma configuração de flip-flop tipo D com RESET é muito comum para circuitos integrados, a qual foi usada neste projeto e é mostrada na figura V.9 .

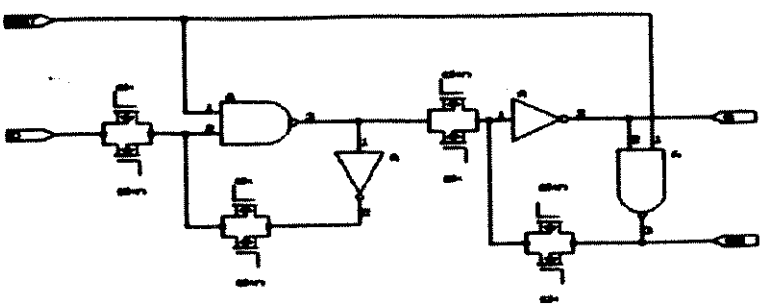


Figura V.9 - Flip-flop tipo D com RESET

Assim, como no circuito em questão tem-se 64 flip-flops D e 64 multiplexadores 2:1, teremos para a lógica:

Um flip-flop D é composto de :

Portas de transmissão	4
NAND de 2 entradas	2
Inversores	2

Número de portas lógicas total:

384 portas de transmissão	384
128 portas NAND de 2 entradas	128
128 inversores	128

CIRCUITO DE ALINHAMENTO COM REGISTRADOR DE 64 BITS E MULTIPLEXADOR

Nesta alternativa tem-se três blocos básicos:

Registrador

Multiplexador

Contador

BLOCO REGISTRADOR

Lógica do Registrador:

Flip - flops tipo D	64

Total de portas lógicas:

Portas de transmissão	256
NAND de 2 entradas	128
Inversores	128

Total de lógica envolvida:

Portas de transmissão	112
NAND de 3 entradas	8
Inversores	114

BLOCO CONTADOR

Decidiu-se utilizar um contador síncrono (figura V.8), para evitar com isso a passagem por estados intermediários durante a contagem ocasionados por atrasos na propagação dos sinais, como poderia ocorrer se um contador assíncrono fosse utilizado.

Total de lógica do contador :

Portas de transmissão	12
NAND de 2 entradas	6
Inversores	6
XOR	2
NOR	1

TOTAL

Para o circuito inteiro teríamos:

Portas de transmissão	380
NAND de 2 entradas	134
Inversores	248
NAND de 3 entradas	8
XOR	2
NOR	1

Comparando-se o número de portas lógicas nas duas alternativas de circuito decidiu-se adotar a primeira implementação (arranjo de registradores de deslocamento).

Um fato importante a ser levado em conta é que, seja qual for a implementação escolhida, o número de entradas é bastante grande já que se tem 64 bits de dados. Este número de entradas acarreta no "layout" físico do circuito integrado, um número bastante grande de células de entrada que devem estar dispostas na periferia da pastilha e também um grande número de pinos no encapsulamento.

Tecnologias que permitam tamanho de células de entrada menores e encapsulamentos com grande número de pinos, como é o caso dos encapsulamentos disponíveis para montagem em superfície, possibilitam a implementação de blocos com grande número de sinais de entrada e saída.

V.3.3.3 - Implementação de Somadores

Sabendo-se que o sistema proposto deve operar em frequências da ordem de 18,4 MHz necessárias para operação em tempo real, como demonstrado no ítem V.2.1, foram verificadas as alternativas lógicas disponíveis para somadores levando-se em conta que, como se deseja fazer uma estrutura "pipeline" a arquitetura deve ser adequada a este tipo de implementação, ou seja, deve ter características como:

- Possibilidade de particionamento da soma em sub-etapas independentes
- Sub-etapas executáveis em tempos semelhantes
- Fluxo linear de dados

Os somadores utilizados no circuito são aqueles mostrados na figura III.1 e repetidos em detalhe na figura V.10 abaixo.

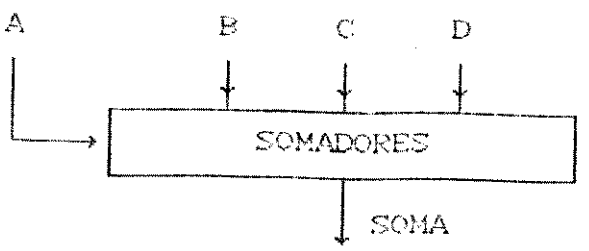


Figura V.10 - Somador utilizado na arquitetura a ser Implementado

Das alternativas pesquisadas destacam-se duas:

- Adição Múltipla com Bits Particionados
- Somadores do Tipo CSA (somador com propagação de transporte) e CPA (somador com propagação de transporte)

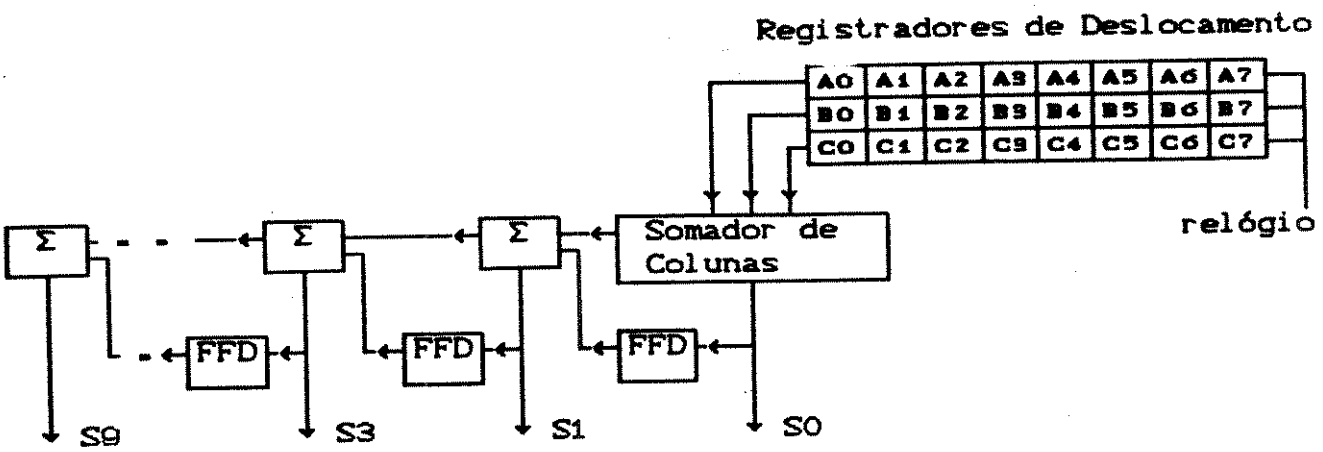
V.3.3.3.1 - Adição Múltipla com Bits Particionados

A soma de vários operandos com bits particionados consiste em dados *K* números de *N* bits realizar a soma dos bits de mesma ordem armazenando os bits de transporte resultantes até o próximo ciclo de relógio, quando será efetuada a soma dos bits de maior ordem. Desta forma tem-se uma soma a cada *N* ciclos de relógio.

Assim supondo-se que se quisesse realizar a soma de três números de 8 bits poderia ser usado o circuito da figura V.11 [12].

Este circuito é composto de:

- Registradores de deslocamento de 8 bits
- Somador de Colunas
- Meios somadores



Σ - Meio somador

FFD - Flip-flop tipo D

Figura V.11 - Somador do Tipo Bit Particionado para Três Números de 8 bits

Os registradores de deslocamento de entrada alinham os bits de mesma ordem dos três números a serem somados para a entrada no somador de coluna. A cada ciclo de relógio estes são adicionados, resultando dois "bits" de saída que serão levados a outra parte do circuito composta de flip-flops tipo D e meios somadores. As sucessivas parcelas serão geradas a cada ciclo de relógio, sendo que após 8 ciclos obtém-se o resultado final da soma na saída.

A estrutura apresentada tem como principais vantagens o fato de possuir implementação como um "pipeline" e a grande velocidade de operação possível de se obter devido aos baixos tempos de propagação em cada estágio. Isto pode ser constatado observando que o atraso no pior caso seria apenas o atraso de 9 meios-somadores, o que é equivalente (Figura V.12) ao atraso de 8 portas NAND de duas entradas mais o de 8 inversores e os de uma porta XOR o que, usando os dados obtidos por simulação do apêndice II, resulta em:

$$t_{atraso} = 8 * t_{NAND2} + 8 * t_{INV} + t_{XOR} = 8 * 1,3 + 8 * 0,025 + 0,355$$
$$= 18,15ns$$

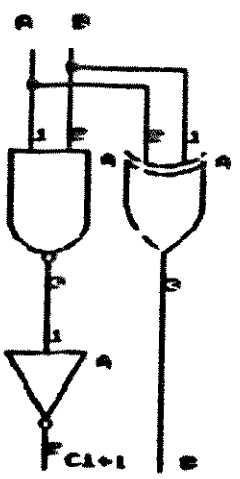


Figura V.12 - Meio Somador

A principal desvantagem do ponto de vista da implementação que se deseja realizar é o fato de que a entrada de dados não é feita de forma serial e sim paralela o que requereria um circuito que trabalhasse a uma velocidade 8 vezes superior a da entrada de dados para manter a mesma taxa na saída. Para isso se necessitaria usar fases diferentes de relógio e um circuito de controle de armazenamento dos dados de entrada no registrador de deslocamento.

V.3.3.3.2 -SOMADORES DO TIPO CSA (SOMADOR SEM PROPAGAÇÃO DE TRANSPORTE) E CPA (SOMADOR COM PROPAGAÇÃO DE TRANSPORTE)

Os somadores do tipo CSA/CPA baseiam-se no fato de que quando se somam vários operandos é possível armazenar temporariamente os bits de transporte até que todas somas parciais tenham sido realizadas e depois completar a propagação desses bits obtendo a soma final.

Na célula básica do somador tipo CSA, que é um somador comum, são somados 3 bits como pode ser visto na figura V.13, dois destes são levados às entradas A e B e outro na entrada C que seria naturalmente destinada ao "bit" de transporte de entrada. À saída deste bloco obtém-se dois valores, o bit de soma e o de transporte. Devem ser usado um número de células idêntico ao número de bits que se queira somar.

Como a célula básica do somador tem 3 entradas , pode-se somar com a estrutura da figura V.14 no máximo 3 números. A soma de mais de três números simultaneamente pode ser obtida utilizando uma das configurações mostradas na figura V.15.

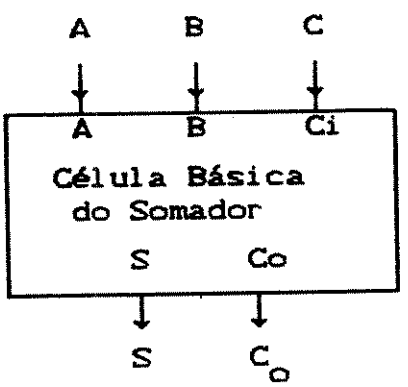
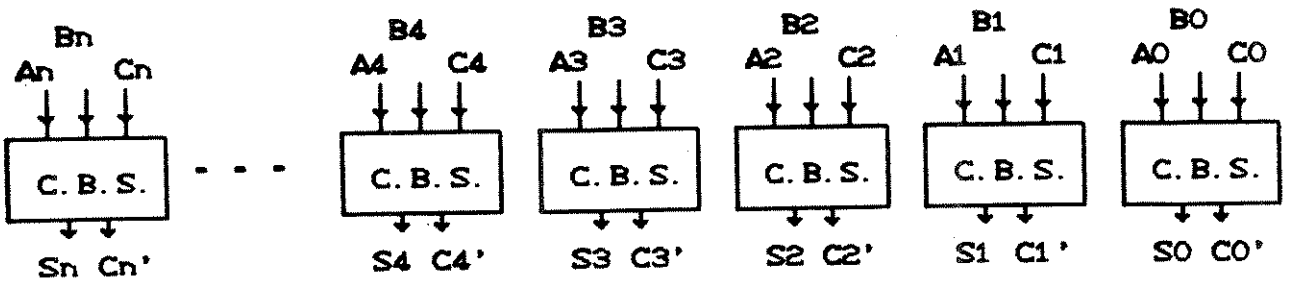


Figura V.13 - Célula básica do somador usado na CSA

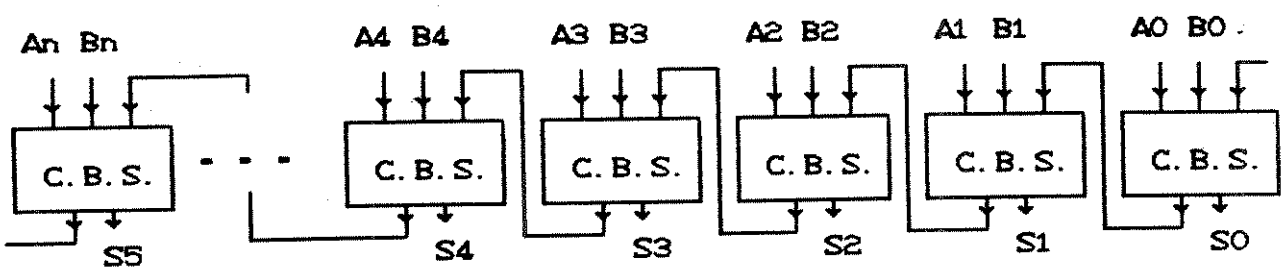
Obtidos os resultados deve-se propagar o "bit" de transporte , o que se faz utilizando um somador do tipo CPA (somador com propagação de transporte) como mostrado na figura V.16.



C.B.S. = Célula Básica do Somador

Figura V.14 - Estrutura de uma CSA para a soma de 3 números

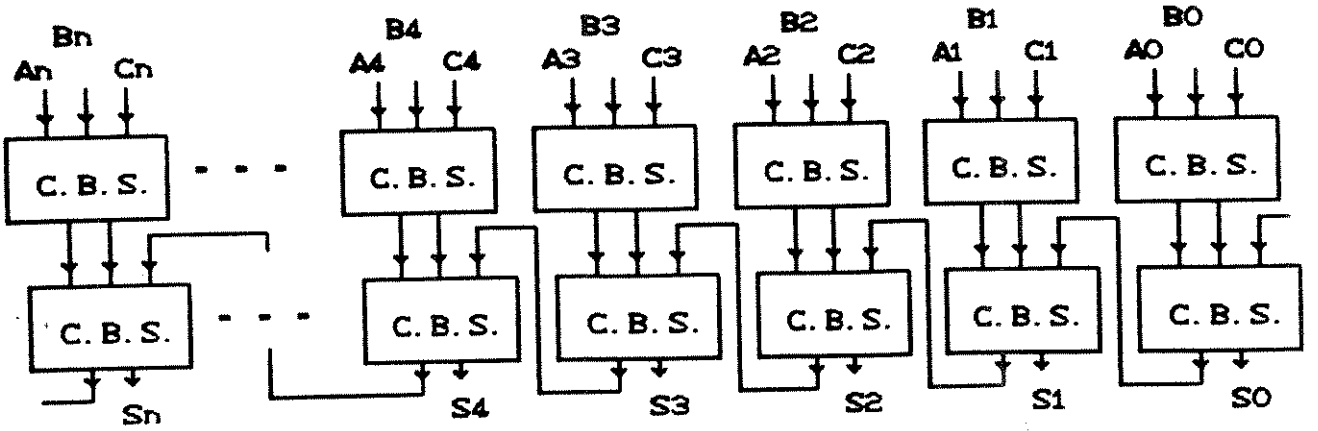
Figura V.15 - Configurações de CSA para a soma de mais de 3 números



C.B.S. - Célula Básica do Somador

Figura V.16 - Circuito CPA (soma com propagação de transporte)

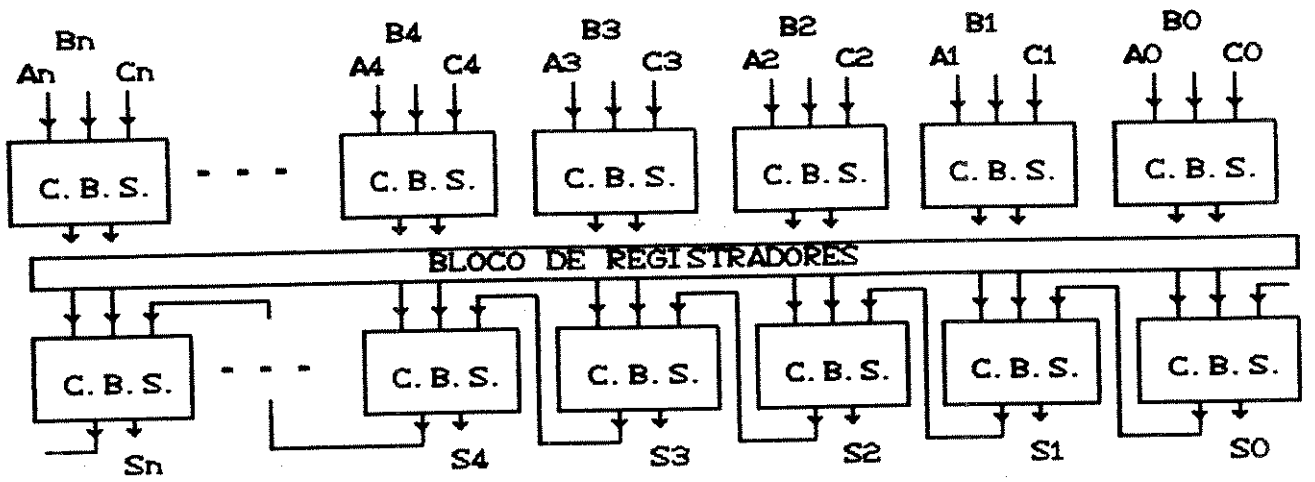
Combinando um CSA e CPA obtém-se um somador multi-operandos completo, como mostrado na figura V.17.



C. B. S. - Célula Básica do Somador

Figura V.17 - Somador combinando CSA e CPA

Circuitos deste tipo são facilmente configuráveis como um "pipeline" intercalando-se blocos de registradores entre os módulos, como mostrado na figura V.18.



C. B. S. - Célula Básica do Somador

Figura V.18 - Somador do tipo CSA/CPA configurado como um "pipeline"

No "pipeline" da figura V.16 a frequência de relógio fica condicionada ao tempo de propagação da CPA, já que esta tem um tempo de propagação maior que o da CSA. Estes tempos de propagação podem ser calculados utilizando os valores apresentados na tabela V.1.

Porta	Notação	Atraso (ns)
Inversor	t_{inv}	0,92 ns
Porta de transmissão	t_{gate}	0,26 ns
Porta complexa	t_{gcompx}	3,5 ns

Tabela V.1 - Tempos de propagação nas portas lógicas que constituem a CSA e a CPA (provenientes da tabela apresentado no apendice II)

Tempo de propagação da CSA = Tempo de propagação em apenas 1 somador = $t_{inv} + t_{gate} + t_{gcompx}$ = 0,92 + 0,26 + 3,5 = 4,68 ns

Tempo de propagação da CPA com N somadores = tempo de propagação em N somadores = N * 4,68 ns.

Se o tempo de propagação da CPA for muito grande, inviabilizando a operação em tempo real, pode-se dividir a CPA em partes, sendo cada uma delas um estágio adicional do "pipeline", como mostrado na figura V.19. Assim, mantendo 4 somadores por estágio o atraso máximo será de 18,72 ns o que permitiria operação em 26 MHz.

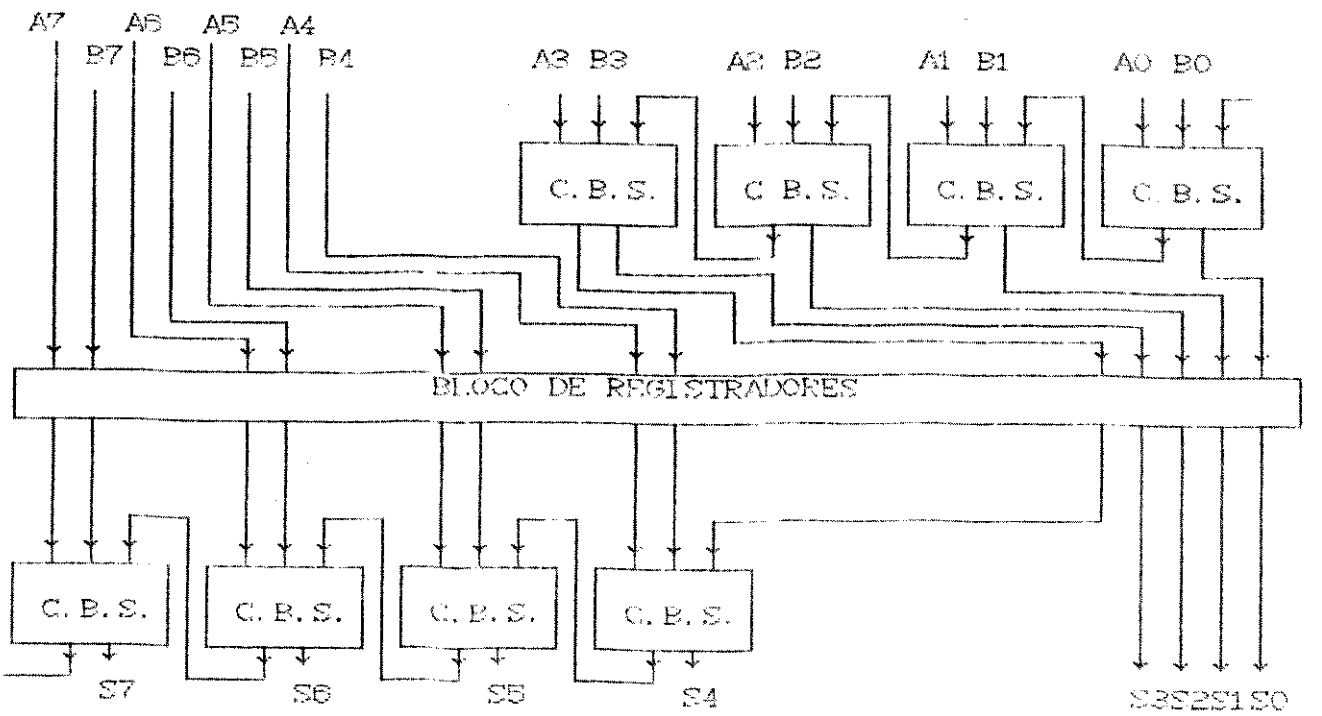


Figura V.19 - CPA dividida em vários estágios

Esta implementação tem como vantagens a velocidade de operação, que depende do atraso da CPA, a possibilidade de uso do mesmo sinal de relógio dos outros blocos do circuito, a entrada paralela de dados e o baixo número de estágios de "pipeline" se comparados à arquitetura de bits particionados. Por estes motivos decidiu-se pelo uso destes somadores para o circuito em questão.

Na arquitetura do capítulo III observam-se 3 tipos de somador:

- Um somador de 3 operandos de 14 + 14 + 14 bits
- Um somador de 4 operandos de 14 + 14 + 14 + 16 bits
- Um somador de 4 operandos de 14 + 14 + 14 + 19 bits

O circuitos somadores finais podem ser vistos nas figuras V.20, V.21 e V.22.

No último somador são "truncados" os 10 bits menos significativos, já que o resultado deve ser em 8 bits, o que equivale a dividir o número por 2^{10} e ignorar a mantissa do número assim obtido.

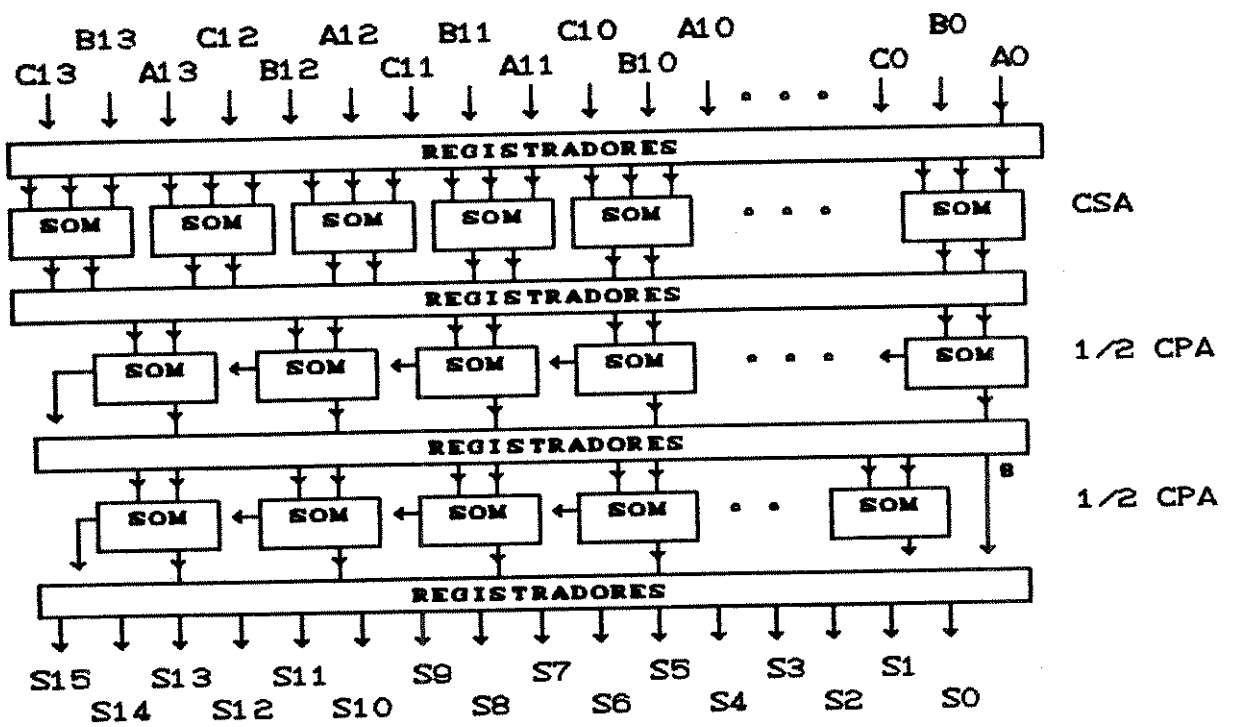


Figura V.20 - Somadores de 3 operandos de 14 + 14 + 14 bits

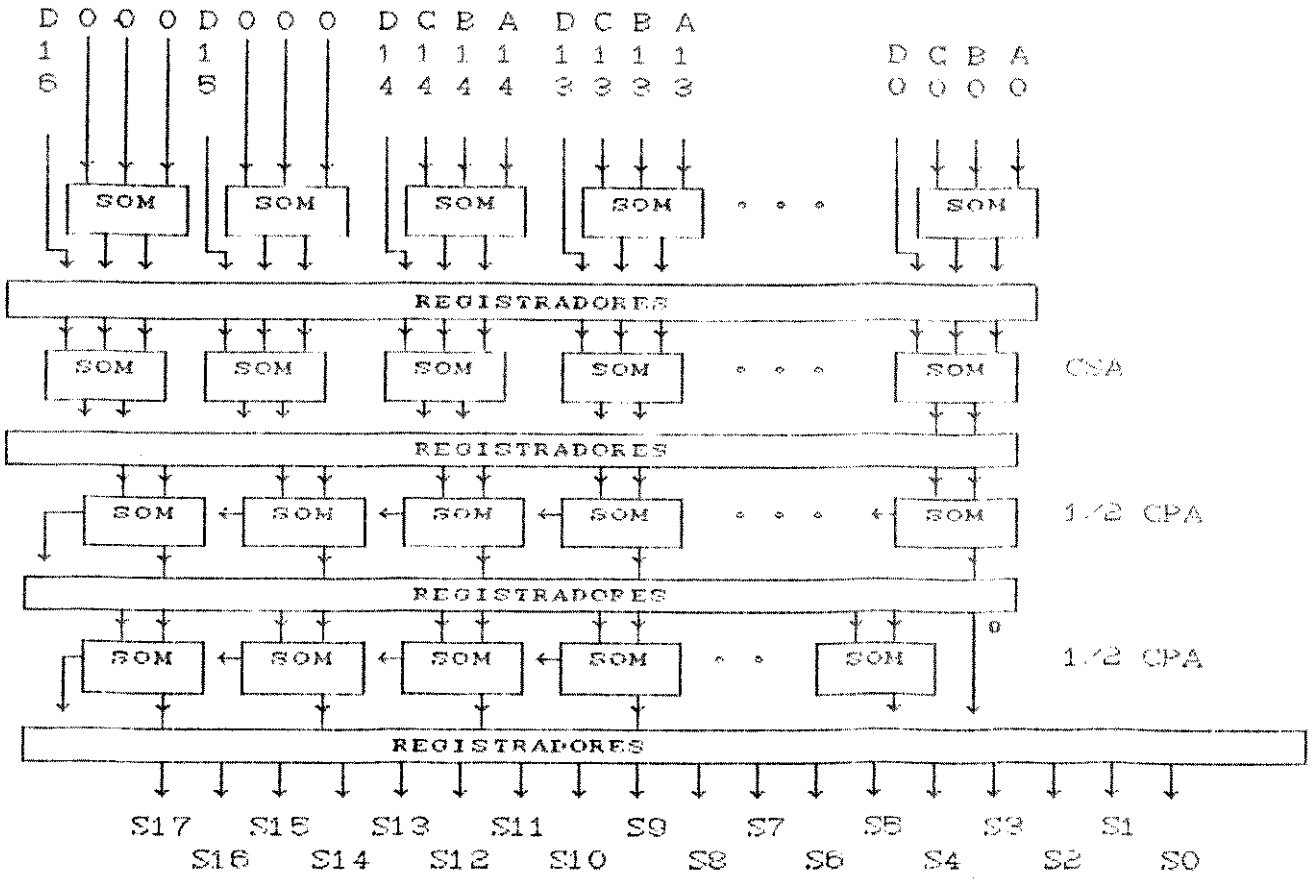


Figura V.22 - Somadores de 4 operandos de 14 + 14 + 14 + 17 bits

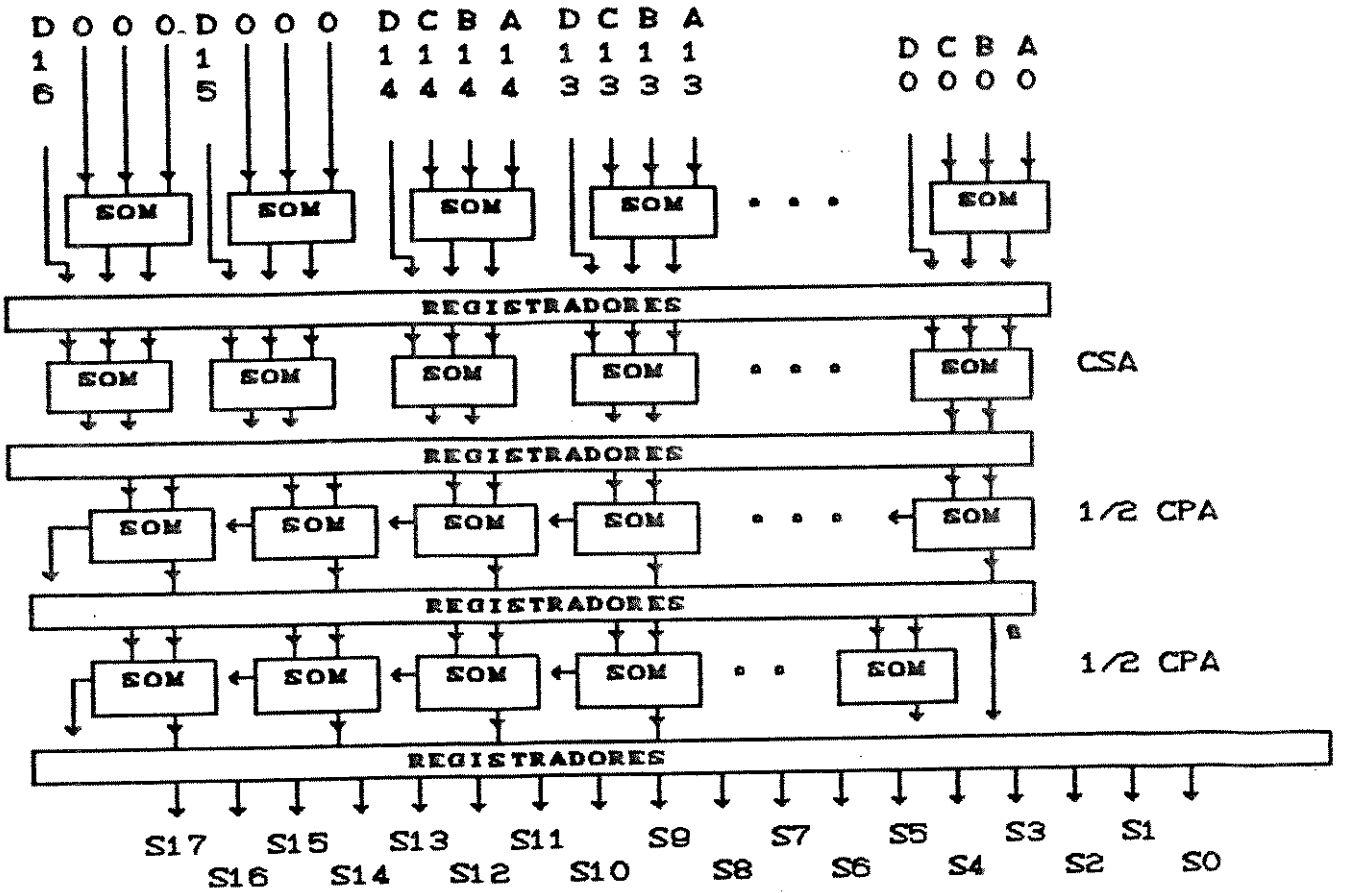


Figura V.22 - Somadores de 4 operandos de 14 + 14 + 14 + 17 bits

V.3.3.4 - Implementação de Multiplicadores

Das arquiteturas pesquisadas que satisfazem aos requisitos anteriores, foram selecionadas :

- Matrizes multiplicadoras
- Arquitetura de CSA e CPA

V.3.3.4.1 - Matrizes Multiplicadoras

Multiplicadores do tipo matriz (Figura V.19) têm como principal limitação os atrasos que os sinais sofrem percorrendo toda a estrutura. Assim, por exemplo, dados 2 números A= A7 A6 A5 A4 A3 A2 A1 A0 e B= B7 B6 B5 B4 B3 B2 B1 B0, supondo que o atraso por célula do multiplicador seja de "*tap*" para a geração do bit de transporte, teremos o bit do produto atrasado de 11**tap*, já que os sinais percorrem 11 blocos no caminho crítico (mais longo).

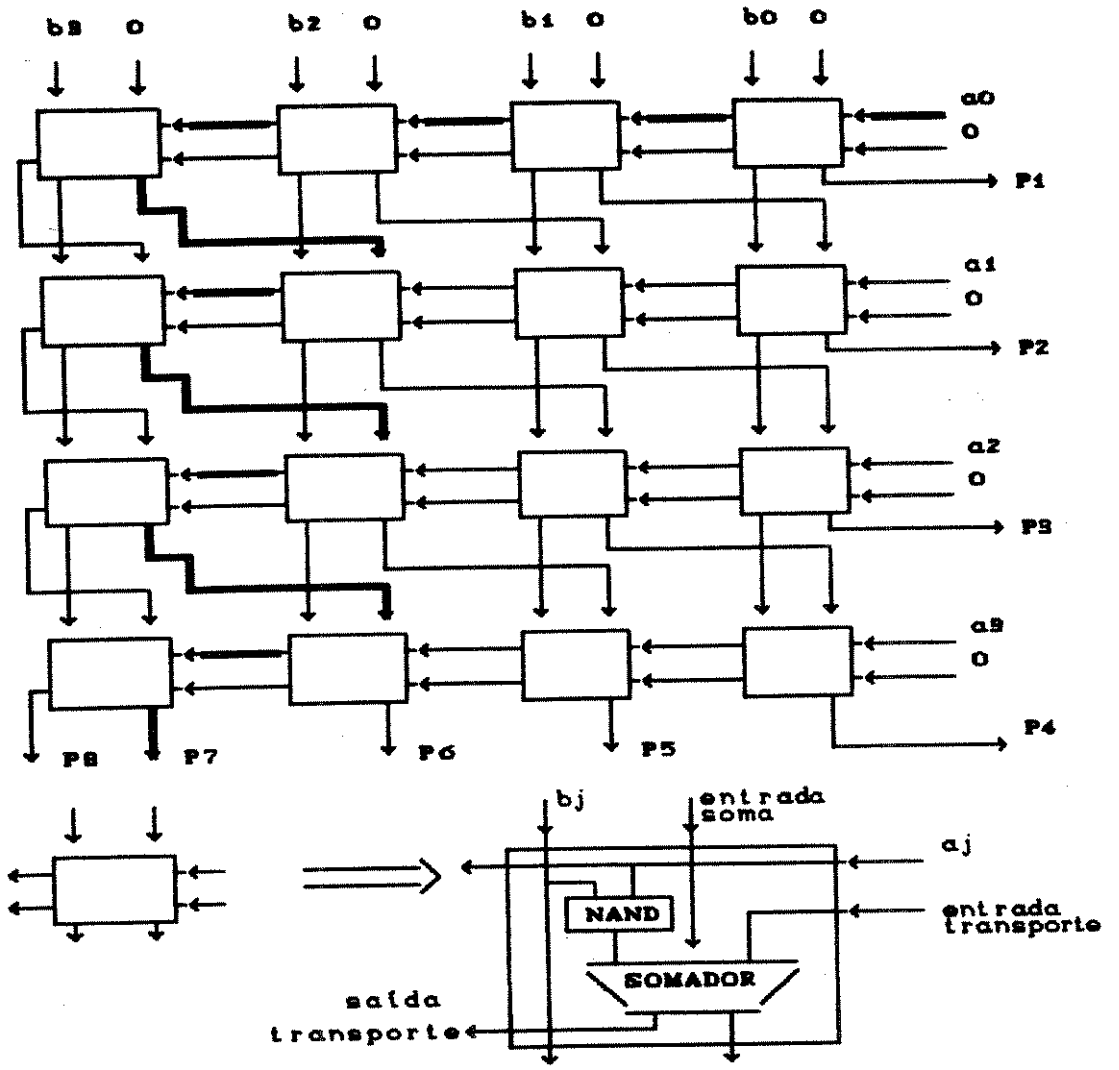
A célula básica do multiplicador em questão é constituída de uma porta NAND e de um somador, como pode ser visto na figura V.23, usando células da biblioteca "standard cell" usada no projeto pode-se implementar a célula como mostrado na figura V.24.

Somando os atrasos das portas (apêndice II) para um "fanout" igual a 1 e considerando as portas lógicas que fazem parte do caminho de maior atraso do circuito, como indicado na fig V.23 tem-se o valor de "*tap*" . Estes atrasos podem ser calculados com os dados apresentados na tabela V.2.

$tap = t_{nand2} + t_{inv} + t_{gate} + t_{gcompx} = 6,11$ ns

Porta	Notação	Atraso (ns)
NAND de duas entradas	t_{nand2}	1,43 ns
Inversor	t_{inv}	0,92 ns
Porta de transmissão	t_{gate}	0,26 ns
Porta complexa	t_{gcompx}	3,5 ns

Tabela V.2 - Atrasos das portas que compõem a matriz multiplicadora



——— - caminho crítico

Figura V.23 - Matriz Multiplicadora

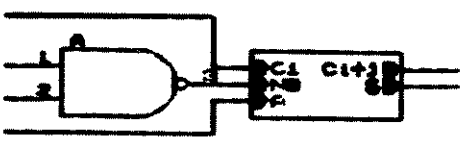


Figura V.24 - Implementação da célula básica do Multiplicador tipo Matriz

Em 11 estágios teremos um atraso total de 67,21ns o que inviabiliza a operação em tempo real de 18,4 MHz que tem um período de 54,35 ns.

Configurando a matriz multiplicadora como um circuito "pipeline" como mostrado na figura V.25 podem-se obter atrasos máximos da ordem de $4*t_{ap}$ = 24.44 ns o que é aceitável para processamento em tempo real.

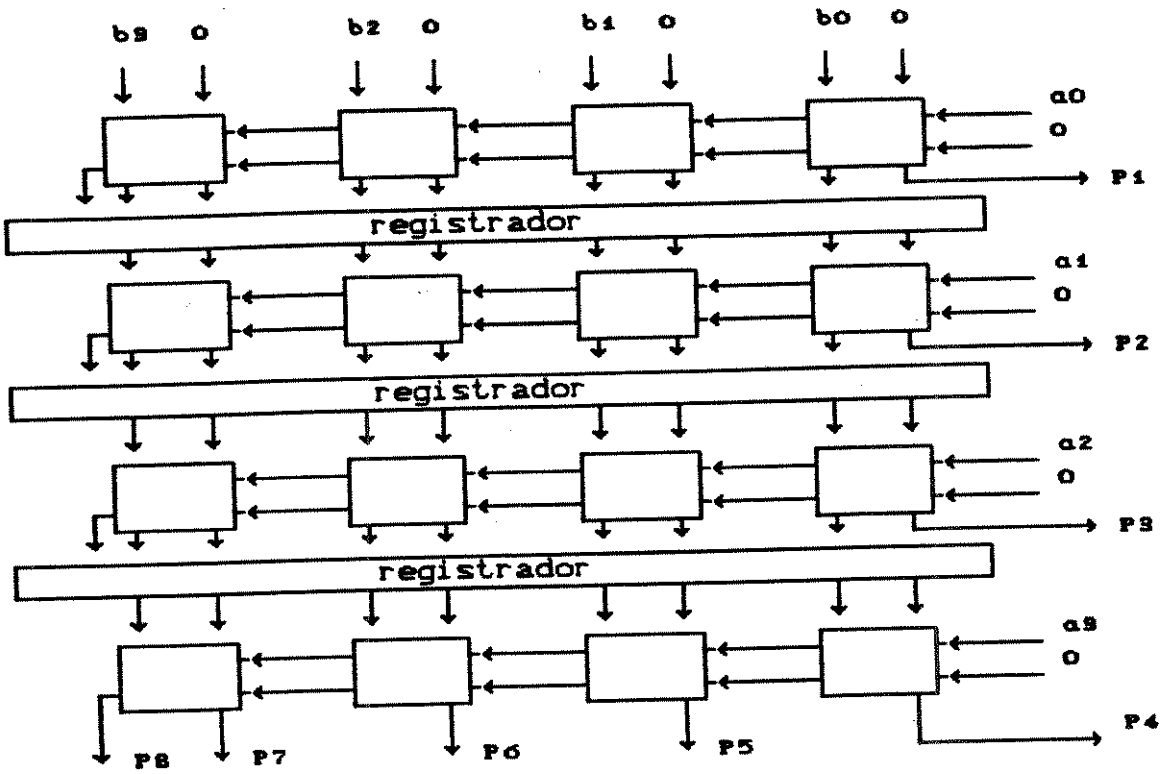


Figura V.25 - Matriz Multiplicadora Configurada como um Circuito "Pipeline"

V.3.3.4.2 - ARQUITETURA DE CSA E CPA PARA MULTIPLICADORES

Uma outra alternativa é realizar a multiplicação como soma de parcelas resultantes da operação "AND" de cada bit do multiplicador pelo multiplicando dos dois números A e B, onde:

$A = A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$ $\qquad$ $B = B_3\ B_2\ B_1\ B_0$

É importante observar que algumas das parcelas obtidas pela operação "AND" deverão ser deslocadas à esquerda o que, efetivamente ocorre na multiplicação.

Assim, as parcelas obtidas serão representadas como:

$A_7.B_3\ A_6.B_3\ A_5.B_3\ A_4.B_3\ A_3.B_3\ A_2.B_3\ A_1.B_3\ A_0.B_3\ 0\ 0\ 0$ ← 11 bits

$A_7.B_2\ A_6.B_2\ A_5.B_2\ A_4.B_2\ A_3.B_2\ A_2.B_2\ A_1.B_2\ A_0.B_2\ 0\ 0$ ← 10 bits

$A_7.B_1\ A_6.B_1\ A_5.B_1\ A_4.B_1\ A_3.B_1\ A_2.B_1\ A_1.B_1\ A_0.B_1\ 0$ ← 9 bits

$A_7.B_0\ A_6.B_0\ A_5.B_0\ A_4.B_0\ A_3.B_0\ A_2.B_0\ A_1.B_0\ A_0.B_0$ ← 8 bits

Onde os "zeros" adicionais já refletem o deslocamento das parcelas.

Calcula-se A X B fazendo a seguinte soma:

$$
\begin{array}{rcccccccccccc}
 & A_7.B_3 & A_6.B_3 & A_5.B_3 & A_4.B_3 & A_3.B_3 & A_2.B_3 & A_1.B_3 & A_0.B_3 & 0 & 0 & 0 \\
+ & A_7.B_2 & A_6.B_2 & A_5.B_2 & A_4.B_2 & A_3.B_2 & A_2.B_2 & A_1.B_2 & A_0.B_2 & 0 & 0 & \\
+ & A_7.B_1 & A_6.B_1 & A_5.B_1 & A_4.B_1 & A_3.B_1 & A_2.B_1 & A_1.B_1 & A_0.B_1 & 0 & & \\
+ & A_7.B_0 & A_6.B_0 & A_5.B_0 & A_4.B_0 & A_3.B_0 & A_2.B_0 & A_1.B_0 & A_0.B_0 & & & \\
\hline
 & & & & & & & & A X B & & &
\end{array}
$$

Desta maneira a arquitetura do multiplicador é realizada com vários somadores de vários operandos.

Estruturas do tipo CSA e CPA explicadas no ítem V.3.3.2 associadas a registradores intermediários (figura V.25) para a ← obtenção de paralelismo temporal através de uma estrutura "pipeline", se mostraram bastante adequadas à realização da multiplicação em questão por apresentarem um bom compromisso entre velocidade de operação e tamanho de lógica a ser implementada. Por este motivo, selecionou-se este circuito para a implementação.

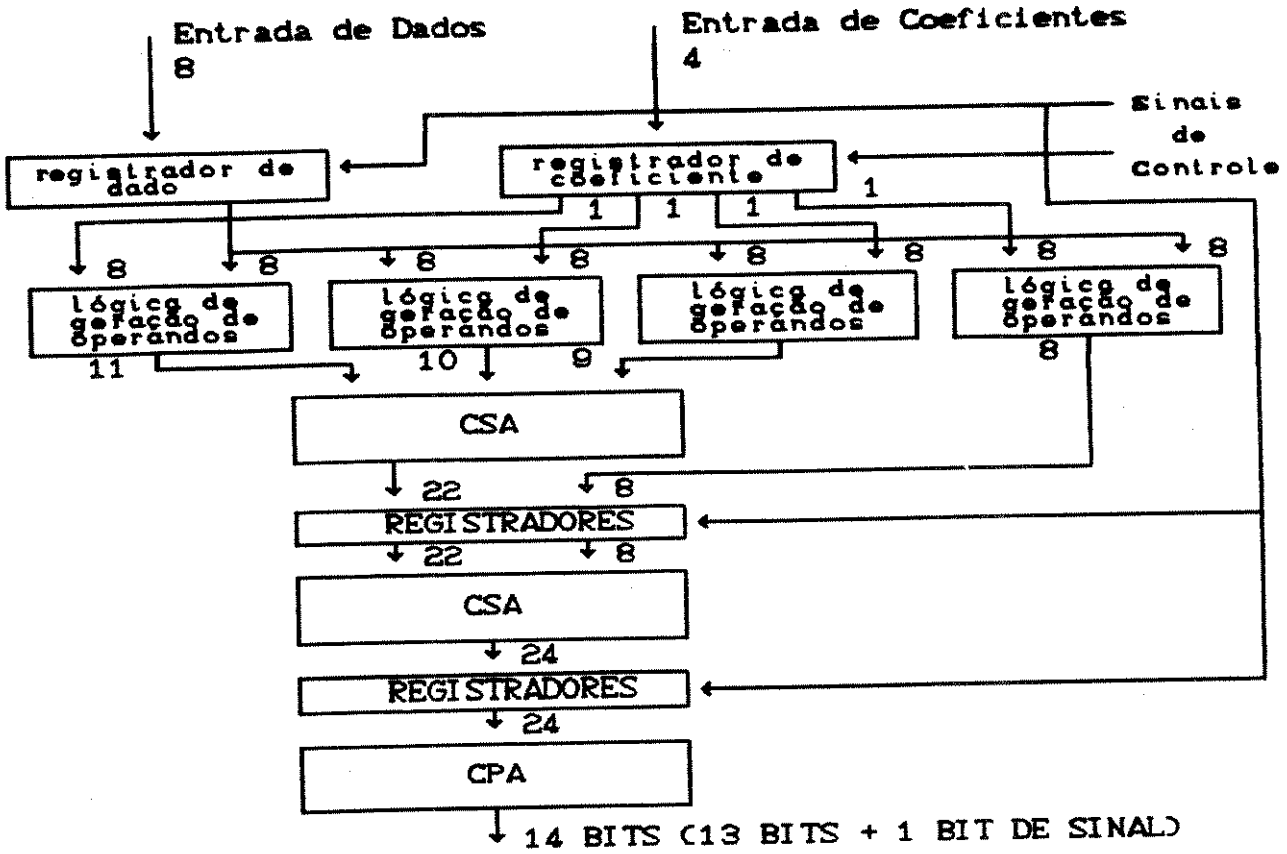


Figura V.26 - Multiplicador implementado segundo uma estrutura de CSA e CPA

Esta estrutura é composta dos seguintes blocos

- Lógica de geração de operandos
- Primeiro "CSA" (somador sem propagação de transporte)
- Segundo "CSA"
- CPA (somador com propagação de transporte)

A geração de parcelas é feita pelo circuito gerador de parcelas (Figura V.27). Neste circuito se optou por calcular a NAND entre os bits do multiplicador e os dos multiplicandos, já que pode-se implementar um somador com entradas negadas sem aumentar o tamanho de sua lógica, como pode ser visto na figura (V.28)

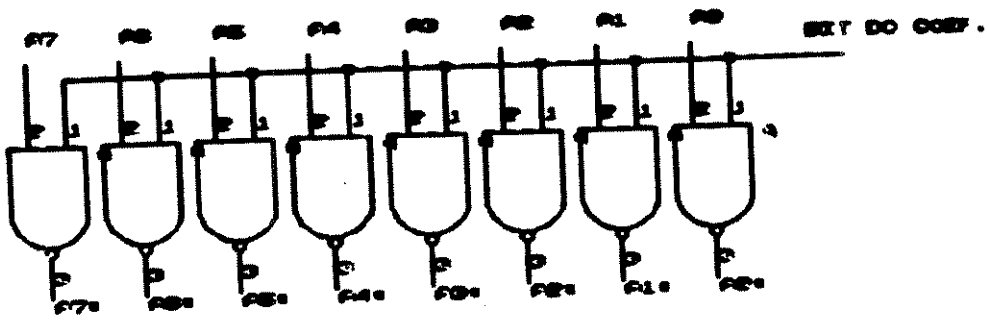


Figura V.27 - Circuito de Geração de Parcelas

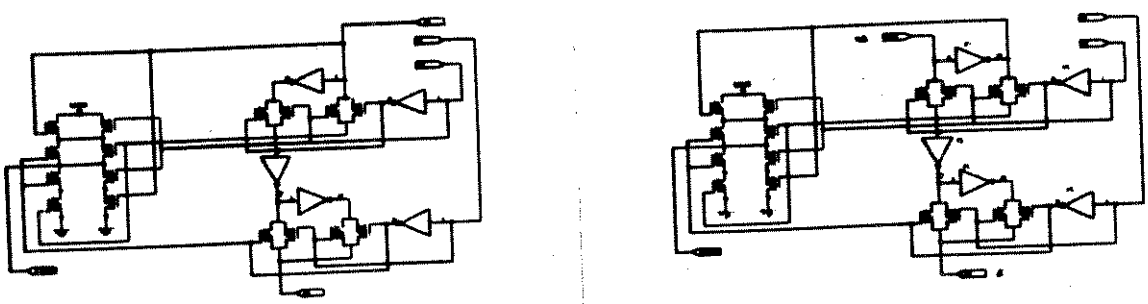


Figura V.28 - Somador Normal e Modificado (entradas negadas)

Para efetuar a soma das parcelas usaram-se duas CSA's uma para a soma de 3 das parcelas e outro para a soma da parcela restante com o resultado obtido na primeira CSA.

Para efetuar a soma final de parcelas se utilizou um CPA.

Como no primeiro estágio (primeira CSA e lógica de geração de parcelas) não há propagação do bit de transporte, o atraso pode ser calculado como o atraso da lógica de operandos (NAND) mais o atraso do somador:

t CSA = t NAND + t SOM = 11,2 ns

No segundo estágio (segunda CSA) o atraso é dado por t SOM = 7,2 ns

No terceiro estágio (CPA), como há propagação do bit de transporte, o atraso máximo é dado por N * t SOM, onde N é o número de somadores cascateados. Se este atraso for excessivo pode-se "seccionar" a CPA em vários estágios do "pipeline".

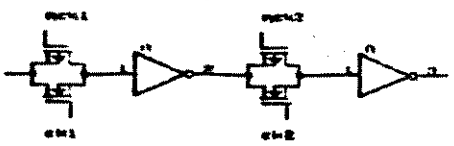
Decidiu-se usar a estrutura de CSA e CPA por esta apresentar um menor número de portas lógicas em comparação com as matrizes multiplicadoras e um número menor de estágios de "pipeline".

O circuito final é o próprio circuito da figura V.26, onde já consta o número de bits à saída de cada estágio.

V.3.3.5 - Implementação de Memórias

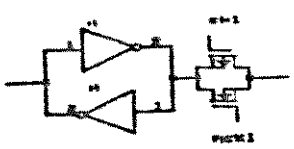
Decidiu-se utilizar memórias estáticas implementadas com flip-flops D modificados para a obtenção de um "layout" menor , por estes não necessitarem de nenhum circuito adicional de controle ou de "refresh".

Várias estruturas foram propostas na literatura para a implementação das memórias FIFO, desde uma estrutura de memória estática, até estruturas de memória dinâmica [7] [17]. Os circuitos lógicos/elétricos de algumas das propostas podem ser vistos na figura V.29.

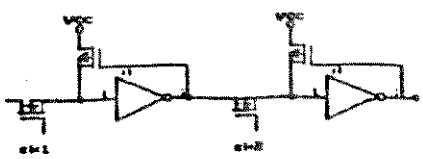


V.29.A

V.29.B



V.29.C



V.29.D

Figura V.29 - Propostas para a implementação da memória FIFO

Para a estimativa de área de memória decidiu-se usar a célula da figura V29.B por esta ter o menor número de transistores.

V.3.3.6 - COMPLEMENTADOR PARA COMPLEMENTO DE 2

Como o dado à saída do multiplicador pode ser positivo ou negativo decidiu-se efetuar todas as somas posteriores à multiplicação em complemento de 2.

Assim um número da forma sinal-magnitude de 14 bits (13 bits mais um de sinal)à saída do multiplicador é convertido em um número de 13 bits na forma de complemento de 2.

O circuito de complementação usado pode ser observado na figura V.30.

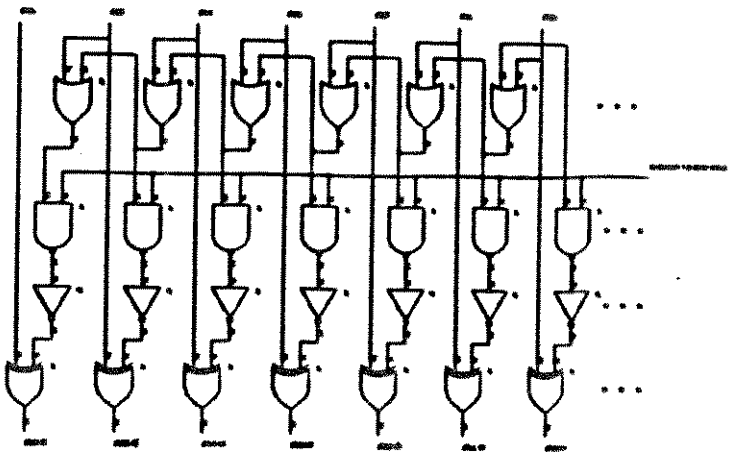
Figura V.30 - Complementador para Complemento de 2

V.3.3.7 - DECOMPLEMENTADOR PARA COMPLEMENTO DE 2

Como os valores somados estão todos na forma de complemento de 2 o resultado final será desconvertido da forma de complemento de 2 para a forma de sinal-magnitude já que nessa última pode ser levado ao controlador de vídeo.

Após a desconversão final é realizada a "truncamento" de dados que converte o resultado para 8 bits.

O circuito de decomplementação pode ser visto na figura V.31.



V.31 - Circuito de Decomplementação de Dados de Saída

V.3.4 - Contabilização Final

O circuito completo de convolução é composto pelos seguintes blocos nas seguintes quantidades :

BLOCO	QUANTIDADE
Lógica de Controle	1
Circuito de multiplexação de dados de entrada	1
Somador de 14 + 14 + 14	1
Somador de 14 + 14 + 14 + 16	1
Somador de 14 + 14 + 14 + 17	1
Multiplicador	9
Memória FIFO de 16 X 512	1
Memória FIFO de 17 X 512	1

Células de entrada e saída:

BLOCO	QUANTIDADE
Célula de Entrada	68
Célula de Saída	18
PADs de Alimentação	2

O número de PADs de alimentação foi calculado levando em conta a dissipação de potência do circuito.

CONTABILIZAÇÃO FINAL DA LÓGICA DO CIRCUITO

Neste sub-ítem serão apresentadas as somas das portas lógicas que compõe cada um dos blocos do circuito de convolução (tabela V.5), e as somas finais para o circuito todo (tabela V.5). O número total de pinos do circuito é apresentado na tabela V.6.

A codificação usada para cada porta lógica é a apresentada na tabela V.3.

PORTA LÓGICA	CODIFICAÇÃO
Inversor	INV
NAND de 2 entradas	NAND2
NAND de 3 entradas	NAND3
NOR de 2 entradas	NOR2
NOR de 3 entradas	NOR3
Porta de Transmissão	TG
Ou-exclusivo	XOR
Flip-flop tipo D	FFD
Célula de memória	CMEM
Somador	SOM

Tabela V.3 - Codificação utilizadas para as portas lógicas que compõem o circuito de convolução

BLOCO	PORTA LÓGICA	QUANTIDADE
Lógica de Controle	FFD NAND2 NOR2 NAND3 NOR3 INV XOR	43 17 12 8 6 6 3
Circuito de Multiplexação de Dados de Entrada	TG INV	128 64
Somador de 14 + 14 + 14	FFD SOM INV	70 29 4
Somador de 14 + 14 + 14 + 16	FFD SOM INV	80 33 4
Somador de 14 + 14 + 14 + 17	FFD SOM INV	85 35 4
Multiplicador	FFD SOM INV	121 17 4
Memória FIFO de 16 X 512	CMEM	8192
Memória FIFO de 17 X 512	CMEM	8704

Tabela V.4 - Número de portas lógicas por bloco

TOTAL DE PORTAS LÓGICAS NO CIRCUITO

PORTA	QUANTIDADE
INV	86
NAND2	17
NAND3	8
NOR2	12
NOR3	6
TG	128
XOR	3
FFD	399
CMEM	16896
SOM	114

Tabela V.5 - Número total de portas lógicas do circuito

TOTAL DE PINOS

PINOS	QUANTIDADE
Alimentação	2
Pinos de Entrada	64
Pinos de Saída	18
Pino de Relógio	1
Pino de RESET	1
TOTAL	86

Tabela V.6 - Número de pinos total do circuito

V.3.5 - A Implementação do Protótipo

Como havia uma limitação na área de silício disponível para a implementação dos circuito decidiu-se implementar somente duas estruturas lógicas para com elas se validar a lógica montando um circuito-teste numa placa de circuito impresso.

As estruturas escolhidas para se implementar são um multiplicador 8 x 4 e um somador 4 X 4 X 4 cascateável com as mesmas arquiteturas vistas anteriormente.

Capítulo VI

Estimativas de Área de Silício e Implementação do Protótipo

VI.1 - INTRODUCÃO

Tendo-se uma estrutura lógica original,os blocos que a compõem sabendo-se a tecnologia que se deseja utilizar e o estilo de projeto,define-se o tamanho da pastilha de silício na qual o circuito vai ser difundido.

Neste capítulo serão apresentados métodos para se estimar o tamanho da pastilha de silício de um circuito numa determinada tecnologia, utilizando uma biblioteca de células (apêndice II) criada para o processo de fabricação através de simulação elétrica (usando o programa SPICE) e um editor gráfico (KIC) para desenvolvimento dos "layouts".

Serão também apresentadas as partições de circuitos decididas face à limitação da área de silício disponível.

VI.2 - DEFINIÇÕES INICIAIS

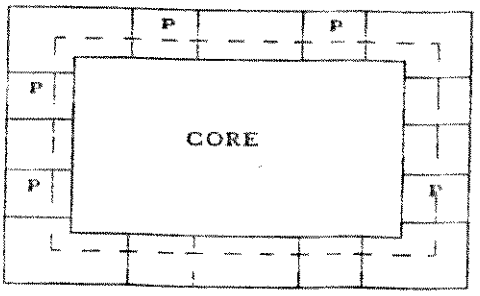
A expressão "PAD" designará o conjunto formado por um "PAD" e uma célula de proteção de entrada ou um "PAD" e um "buffer" de saída.

Quando do dimensionamento de um circuito integrado podem ocorrer duas situações:

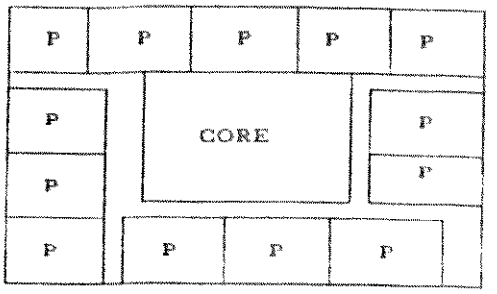
- Circuito limitado pelo "core"
- Circuito limitado pelos "PAD"s

Circuitos limitados pelo "core" são aqueles em que o perímetro do "core" é maior que o perímetro formado pelas bordas internas dos "PAD"s (ver figura VI.1).

Circuitos limitados pelos "PAD"s são aqueles em que o perímetro do "core" é menor que o perímetro formado pelas bordas internas dos "PAD"s (ver figura VI.1).



Circuito limitado pelo "core"



Circuito limitado pelos "PAD"s

Figura VI.1 - Circuitos limitados pelo "core" e pelos "PADs"

Circuitos limitados pelo "core" podem ter sua área melhor aproveitada se houver a possibilidade de se colocar "gates" lógicos nos espaços disponíveis entre os "PAD"s.

Circuitos limitados pelos "PAD"s podem ter sua área melhor aproveitada se a razão de aspecto das células de entrada e saída for modificada para aproveitar melhor a área de silício que possa sobrar do "core".

Dois métodos principais foram desenvolvidos para estimar a área do "die" a ser ocupada pelo circuito a ser implementado [8] :

- O método da "densidade média"
- O método da adição das áreas reais

VI.3 - O Método da Densidade Média

Este método consiste em, tendo-se posse de dados da "foundry" e o diagrama esquemático do circuito a ser implementado, e aplicando estes dados a uma equação se obtém a área estimada do "die"

Equação:

X = comprimento do lado do "die" (supondo-se "die" quadrado)

= altura do "core" + 2 x (altura dos "PAD"s)

$$= \sqrt{Ng \times Ag \times I} + 2 \times (Hp + B + S)$$

Onde:

Ng = número de "gates" equivalentes do circuito lógico (equivalentes a uma NOR de 2 entradas)

Observação: "gates equivalentes" é o número que se obtém somando o número de transistores da lógica e dividindo-se pelo número de transistores da NOR de 2 entradas (4 transistores)

Ag = densidade média das células do "core" (micra^2/gate)

I = fator de interconexão (valor empírico)

Hp = altura das células de PAD usadas + 1 mil (milésimo de polegada) de espaço para interconexão

B = largura do "bus" de VDD que corre à periferia do "die"

S = distância a ser deixada para "scribe line"

OBS: "scribe line" é a linha em que as pastilhas serão cortadas do "Wafer".

Detalhes do Método das densidades médias:

Ag = densidade média das células no "core" (micra^2/gate)

= G x Agg + F x Agf

Onde:

G = % dos "gates" mais simples (NAND, NOR, INVERSOR, XOR) da lógica.

Agg = densidade média para células simples (micra^2/gate)

F = % da lógica que é composta somente de flip-flops, registradores, contadores, etc.

Agf = densidade média para flip-flops (micra^2/gate)

VI.4 - O MÉTODO DA ADIÇÃO DAS ÁREAS REAIS

Este método consiste de, tendo-se o diagrama esquemático do circuito e os dados da "foundry" calcular:

1 - A área total ocupada pelas células da lógica (= Σ Ng x Ag)
2 - Estimar cuidadosamente I (fator de interconexão) e calcular a área do "core" = Ac = I x (Σ Ng x Ag)
3 - Calcular o perímetro da periferia do "core" = Psc = 4 x $\sqrt{Ac}$
4 - Calcular o perímetro formado pelas bordas dos "PAD"s = Pp
5 - Determinar se o circuito integrado é limitado pelo "core", limitado pelo "PAD" ou Intermediário.

INTERMEDIÁRIO se : P x (10 mils) < Psc < P x (10 mils) + 64 mils
Onde p= número de "PAD"s. Neste caso é aconselhável revisar a escolha feita para altura das células de "PAD", aumentando as alturas destas num dos lados do circuito, para que se tenha Pp = Psc. Teremos então um circuito limitado pelo "core".

LIMITADO PELO "CORE" se Psc > Pp, usando células de "PAD" curtas e "die" quadrado. Neste caso calcula-se a área pelo método das densidades médias.

LIMITADO PELOS "PAD"S se Pp < Psc. Aconselha-se aqui o uso de células de "PAD" mais altas. Caso típico em que se necessita de um número muito grande de "buffers" de saída.

6 - Verificar se encapsulamento comporta o "die", com a razão de aspecto inicial e modificar esta razão caso não. Calcular a razão de aspecto final (A.R.)

7 - Área do "die" = X1 x X2.

onde:

$X2 = X1 \times A.R.$ - Comprimento do C.I.

$$X1 = \frac{(Pp/2) + Hp1 + Hp2 + Hp3 + Hp4 + 4 \times B + 4 \times S}{1 + A.R.}$$

Na tabela VI.1 são apresentadas as notações usadas nas equações para as portas lógicas.

PORTA LÓGICA	ÁREA OCUPADA
Nand de duas entradas (NA2)	A_{NA2}
Nand de três entradas (NA3)	A_{NA3}
Nor de duas entradas (NO2)	A_{NO2}
Nor de três entradas (NO3)	A_{NO3}
Inversor (INV)	A_{INV}
Porta Ou-Exclusivo (XOR)	A_{XOR}
Porta de transmissão (TG)	A_{TG}

Tabela VI.1 - Notações usadas para as portas lógicas

E com as seguintes células de interface com as respectivas áreas:

- Célula de proteção de entrada (PROTIN)
- "Buffer" de saída (BUFOUT)

Supondo que na lógica a ser implementada se tivesse os seguintes "gates" e células de interface nas seguintes quantidades:

Gates: a.NA2, b.NA3, c.NO2, d.NO3, e.INV, f.INVCLK, g.XOR, h.TG .
Células de interface: i.PROTIN, j.BUFOUT

A área ocupada somente pelos "gates" seria:

$A_{gates} = a.A_{NA2} + b.A_{NA3} + c.A_{NO2} + d.A_{NO3} + e.A_{INV} + f.A_{INVCLK} + g.A_{XOR} + h.A_{TG}$

A área estimada para a lógica seria:

$A_{lógica} = k . A_{gates}$

Onde k = 1 + (% gasta pelo roteamento)

Adicionando-se a área ocupada pelas células de interface com o mundo exterior teríamos a área estimada para o circuito integrado:

$A_{estimada} = A_{core} + i . A_{PROTIN} + j . A_{BUFOUT}$

VI.5 - A Biblioteca "Standard Cell" Criada para a Implementação Física do Circuito Integrado

A biblioteca criada através de simulação elétrica das portas pelo uso do programa SPICE se compõe das seguintes células com as respectivas áreas:

Gate	Área (μm^2)	Símbolo
Nand de 2 entr.	35,5 X 35,0	NAND2
Nand de 3 entr.	43,5 X 35,0	NAND3
Nor de 2 entr.	34,5 X 35,0	NOR2
Nor de 3 entr.	42,5 X 35,0	NOR3
Inversor	31,5 X 35,0	INV
Buffer de clock	35,5 X 35,0	INVCLK
Ou-Exclusivo	71,5 X 35,0	XOR
Porta de Transm.	37,5 X 35,0	TG
Flip-Flop D	80.5 X 35,0	FFD
Somador	128 X 104,5	SOM

Tem-se ainda as áreas das células de interface com o mundo exterior e as áreas dos "pads" de alimentação do circuito:

Célula de Interface	Área (μm)^2
Cél.de Proteção de Entr.	212 X 249
Buffers de Saída	119 X 200
Pads de VDD e VSS	109 X 109

VI.6 - Estimativas de Área de Silício

Serão calculadas estimativas de área de pastilha de silício para o circuito todo baseadas na contabilização apresentada no capítulo anterior , estimando-se a área de cada bloco e somando-se a área devida às células de entrada e saída.

Ao final do capítulo se determinará quais blocos serão implementados para a construção do protótipo.

VI.7 - Determinação do Tamanho do Circuito Integrado Que Comporta todo o Processador

Nesta estimativa será usado o método da densidade média.

Usando-se a porta NOR de duas entradas (dois transistores P e dois N) se determina o número de "gates" equivalentes.

PORTA	Trans P	Trans N	Gates Eqv.	Quant	Total Gates Eqv.
INV	1	1	0,5	86	43
NAND2	2	2	1	17	17
NAND3	3	3	1,5	8	12
NOR2	2	2	1	12	12
NOR3	3	3	1,5	6	9
TG	1	1	0,5	128	64
XOR	3	3	1,5	3	4,5
CMEM	1	2	0,75	16896	12672
			SUBTOTAL	17156	12833,50
FFD	10	10	2,5	399	997,5
SOM	13	13	3,25	114	370,5
			SUBTOTAL	513	1368,00
			TOTAL	17669	14201,00

Tabela VI.2 - Contagem de portas equivalentes

Na tabela VI.2 são apresentados o número de transistores P e N por porta lógica, o número de portas equivalentes e o número total de portas equivalentes por porta em todo o circuito.

Usando as fórmulas indicadas no início do capítulo, temos:

$$X = \sqrt{Ng * Ag * I} + 2 * (Hp + B + S)$$

$$Ag = G * Agg + F * Agf$$

Total de portas lógicas: 17669

G = 17156/17669 = 97,10 %

F = 513/17669 = 2,90 %

Agg = Densidade média para células simples

Uma média ponderada pelo número de portas equivalentes de cada tipo deverá levar a resultados mais precisos:

Agg = [17 * (35,5 * 35)/1 + 8 * (33,5 * 35) /1,5 + 12 * (34,5 * 35)/1 + 6 * (42,5 * 35)/1,5 + 86 * (31,5 * 35)/0,5 + 3 * (71,5 * 35)/1,5 + 128 * (37,5 * 35)/0,5] + 16896 * (28 * 24)/0,75] / (17 + 8 + 12 + 6 + 86 + 3 + 128 + 16896) = 916.42 μ^2/porta

Agf = [399 * (80,5 * 35)/2,5 + 114 * (128 * 104,5)/3,25] / (399 + 114) = 1791.15 μ^2/porta

Assim, utilizando-se um I=2, já que, segundo [1] a área ocupada pelas interconexões é frequentemente igual à área ocupada pelos blocos lógicos.

Ag = 0,9710 * 916,42 + 0.0290 * 1791,15

Ag = 941,78

$$X = \sqrt{14201 * 941,78 * 2} + 2 * (212 + 0 + 120)$$

X = 5885,89 μm

Assim, para implementar o circuito todo seria preciso um chip de 5.8 por 5.8 mm^2.

Como se dispõe de uma área de silício de 2000 X 2000 μ^2decidiu-se implementar, para o protótipo, um ou mais blocos que ocuparem uma área com um valor próximo da disponível.

Inicialmente decidiu-se calcular somente as áreas do "core" de todos os blocos para só então se decidir pela implementação de um bloco, os resultados são apresentados na tabela VI.3.

BLOCO	DIMENSÕES DO CORE (μm)	ADICIONANDO SCRIBE + CEL
Multiplicador	1230,51	1945,31
Som 14+14+14	1111,84	1826,64
Som 15+14	844,34	1559,14
Som 16+14	1002,50	1717,30
Circ. Alinh.	726,07	1440,87
FIFO 15x512	3212,77	3927,57
FIFO 16x512	3318,14	4032,94

Tabela VI.3 - Áreas de "core" dos blocos do circuito

Outro ponto a ser analisado é que por imposição das regras de projeto as células de entrada e de saída, bem como os "PADs" de alimentação deverão estar dispostos na periferia da pastilha , assim, determinou-se o número de pinos que cada bloco teria se fosse implementado em separado e o perímetro de pastilha necessário para o circuito integrado. Estes valores são listados nas tabelas VI.4 e VI.5.

BLOCO	CEL. E/S	NÚMERO	LARGURA	PERÍMETRO NECES.
Multiplicador	Entradas	10	249	
	Saídas	14	200	
	Aliment.	2	110	5510
Som 14+14+14	Entradas	44	249	
	Saídas	15	200	
	Aliment.	2	110	14176

Tabela VI.4 - Áreas e perímetros total consumida pelas células de entrada, saída e "pads" de alimentação

BLOCO	CÉL/S	NÚMERO	LARGURA	PERÍMETRO NECES.
Som 15+14	Entradas	31	249	
	Saídas	16	200	
	Aliment.	2	110	11139
Som 16+14	Entradas	30	249	
	Saídas	17	200	
	Aliment.	2	110	11080
Circ. Alinh.	Entradas	68	249	
	Saídas	8	200	
	Aliment.	2	110	19254
FIFO de 15x512	Entradas	16	249	
	Saídas	15	200	
	Aliment.	2	110	7204
FIFO de 16x512	Entradas	17	249	
	Saídas	16	200	
	Aliment.	2	110	7653

Tabela VI.5 - Áreas e perímetros total consumida pelas células de entrada, saída e "pads" de alimentação

Sabendo-se que a área disponível é de 2000 μ^2 x 2000 μ^2, tem-se um perímetro de 8000 μm .

Assim, tanto pelo critério da área de "core" quanto pelo das células de entrada e saída, os blocos possíveis de se implementar são o multiplicador e as memórias FIFO.

Os somadores, porém, são possíveis de se implementar para a construção do protótipo em blocos cascateáveis com desempenho similar a blocos únicos.

Após refeitos os cálculos chegou-se à conclusão que é possível implementar um somador "pipelined" cascateável para três números de 4 + 4 + 4 bits com o desempenho desejado.

O número de portas equivalentes para o somador 4 + 4 + 4 é apresentado na tabela VI.6.

PORTAS	QUANT	PORTAS EQ.	PORTAS EQ. TOTAIS
INV	37	0,5	18,5
XOR	36	1,5	54
NAND2	46	1,0	46
NAND3	3	1,5	4,5
NOR2	19	1,0	19
FFD	28	2,5	70
	169		212

Tabela VI.6 - Número de portas equivalentes para o somador 4 + 4 + 4

Efetuando os cálculos chega-se a :

Ag = 2323,69

X = 1450,65 μm (pastilha de 1450,65 X 1450,65)

Número de pinos (ver diagrama esquemático no apêndice I) :

	PINOS	LARGURA	LARGURA TOTAL
ENTRADA	24	249	5976
SAÍDA	12	200	2400
ALIMENTAÇÃO	2	110	220
			8596

Tabela VI.7 - Número de pinos do somador 4 + 4 + 4

O número de pinos permite que, modificando o "layout" da célula de entrada de modo a torná-la mais estreita e mais comprida, utilizando a área de silício não ocupada pelo "core", se tenha o perímetro total mostrado na tabela VI.8.

	PINOS	LARGURA	LARGURA TOTAL
ENTRADA	24	150	3600
SAÍDA	12	200	2400
ALIMENTAÇÃO	2	110	220
			6220

Tabela VI.8 - Perímetro das células de entrada, saída e alimentação totais do somador 4 + 4 + 4

Assim, decidiu-se pela implementação para o protótipo dos circuitos multiplicador e somador 4 + 4 + 4 como circuitos integrados.

As áreas de "core" para o multiplicador e para o somador são:

	REAL	ESTIMADO
MULTIPLICADOR	1726672,35 (= 1314,03^2)	1514154,86 (=1230,51^2)
SOMADOR	1279638,85 (= 1131,21^2)	985234,91 (=992,59^2)

Erro Percentual entre área real e a estimada:

	ÁREA	LADO DO QUADRADO DE ÁREA EQUIV.
MULTIPLICADOR	12,30	6,35
SOMADOR	23,01	13,96

As discrepâncias entre os valores estimados e obtidos se devem principalmente às diferenças no índice I usado no cálculo da área de silício estimada , este índice indica a área gasta com as interconexões entre as portas lógicas.

Uma forma de se obter um valor mais preciso para este índice seria executar vários roteamentos em vários circuitos usando o mesmo programa de roteamento ou utilizando algoritmos para a previsão de espaço de roteamento, como o usado em [11].

CAPÍTULO VII

CONCLUSÕES E RESULTADOS

VII.1 - Trabalho Realizado

Neste trabalho foram analisadas várias estruturas para a detecção de bordas.

A partir da arquitetura proposta por [4] desenvolveu-se um estudo para a viabilização de um implementação como circuito integrado.

Foram discutidos assuntos relevantes do ponto de vista da implementação como circuito integrado, tais como a seleção da tecnologia, seleção do estilo de projeto, opções de circuitos lógicos mais adequados a uma implementação como circuito integrado, particionamento de circuitos, edição do "layout", etc.

Descreveram-se as opções adotadas para cada bloco do circuito e foi dada a justificativa para cada escolha.

Foi estimado o tamanho da pastilha de silício necessário para a implementação do processador e dos blocos difundidos para o protótipo, no final a comparação entre os resultados estimados e obtidos resultou num erro máximo inferior a 14% para lado do quadrado de área equivalente, este erro se deveu principalmente à imprecisão no valor do coeficiente "I" que estima a porcentagem do "core" ocupada pelas interconexões entre as portas lógicas. Sugerem-se métodos para a melhor determinação de um valor para I.

Foi criada uma biblioteca de células padrão (standard cell) a partir dos parâmetros do processo de fabricação e de simulações elétricas como o programa SPICE, esta biblioteca pode ser vista no apêndice II.

Foram gerados os diagramas esquemáticos dos circuitos, os quais serviram de base para a simulação com a qual se validou a lógica e posteriormente para o programa de roteamento automático (programa que faz a interligação entre as portas lógicas).

Foi analisado o erro introduzido pela implementação nos resultados finais.

Assim, se determinou a viabilidade da implementação como circuito integrado da arquitetura proposta sendo apresentadas as dimensões e desempenho esperado.

VII.2 - Contribuição do Trabalho

A principal contribuição que este trabalho vem a trazer provém da seleção e adaptação dos circuitos lógicos para uma implementação como circuito integrado, buscando assim uma configuração de circuito que possa aproveitar as características de uma implementação deste tipo para resolver as limitações da arquitetura originariamente proposta. Assim, serão descritas a seguir as opções realizadas.

Outra contribuição que se poderia citar são os próprios circuitos aqui desenvolvidos os quais podem ser futuramente usados outras implementações de outros algorítmos.

VII.2.1 - Interface com o Barramento de 64 Bits

Na proposta descrita no capítulo III mostrava-se que o dado vinha ao circuito através de um barramento de 64 bits. Este barramento permite, segundo comprovado na referência bibliográfica citada, aumentar em muito a taxa de transferência de dados entre a memória e o processador. Entretanto a arquitetura do circuito processador de bordas permite que somente 1 "pixel" seja levado ao processador a cada ciclo de relógio. Assim, foi buscada uma solução que permitisse compatibilizar a taxa de transferência de dados do barramento com a do circuito, aproveitando assim as vantagens do barramento de 64 bits.

A solução encontrada foi o circuito multiplexador de entrada controlado pela lógica de controle (ambas descritas no capítulo V). O circuito multiplexador de entrada utiliza a vantagem de se poder escolher um encapsulamento de vários pinos para implementar dentro do circuito a interface entre os barramentos através de uma lógica simples e facilmente implementável como circuito integrado, o que não ocorre no caso de uma implementação com componentes comerciais já que se teria que interligar vários circuitos integrados, ocupando espaço de placa de circuito impresso, dissipação de potência adicional, etc.

A lógica de controle controla o fluxo de dados e coeficientes no processador, sua implementação como uma máquina de estados é bastante conveniente para uma configuração como circuito integrado pois sendo este dedicado à função a que se destina, pode ser realizado com uma lógica reduzida.

Se o desenho físico deste circuito for feito com uma matriz programável como é o caso de uma PLA ou ROM, tem-se ainda a vantagem de poder reconfigurar a máquina de estados em futuras versões do circuito alterando muito pouco o desenho físico.

VII.2.2 - Arquitetura do Somador e do Multiplicador

Os circuitos CSA (somador sem propagação de transporte) e CPA (somador com propagação de transporte) configurados segundo um "pipeline" são bastante adequados à implementação como circuito integrado já que o elevado número de registradores entre os estágios que este circuito demanda não representa uma limitação significativa para um circuito integrado, o que não ocorre com uma implementação com circuitos comerciais já que se necessitaria de vários circuitos integrados. Além disso as dificuldades de distribuição do sinal de relógio e o número adicional de pinos acarretariam problemas de "fanout", de interferência entre linhas e de espaço de placa de circuito impresso aumentando os custos do circuito o que pode ser muito mais eficientemente controlado num circuito integrado.

VII.2.3 - Memórias

O uso de memórias FIFO embutidas dentro do próprio circuito permite a diminuição do tempo de acesso ao dado o que implica em maior velocidade de operação para o circuito.

VII.3 - Resultados

Finalmente se chegou a um circuito com capacidade de operação em tempo real (60 quadros por segundo), para a execução de convolução 3x3 sobre uma imagem de 512 x 512 "pixels" de imagem.

A validação do circuito e suas características de operação foram obtidas através de simulações lógicas mostradas ao final deste capítulo.

Foram feitos os desenhos físicos de dois circuitos integrados-protótipo, os quais foram enviados para fabricação.Estes circuitos integrados são o início de uma família de circuitos integrados para processamento "pipeline" de imagem que, pela generalidade de suas funções (soma e multiplicação) podem ser usados em várias outras aplicações. Os mesmos vem sendo testados junto ao CTI (Centro Tecnológico para Informática).

VII.4 - Características do Circuito e Comparação com as Arquiteturas Estudadas

Frequência de Relógio	18,4 MHz
Frequência de Imagem	60 Quadros/s
Tecnologia	CMOS 2μ
Área de Silício	5,7 X 5,7 mm^2
Kernel	3 X 3
Número de Bits dos coefic.	7 bits
Número de bits do dado ent/sai	8/8
Possibilidade de Cascateamento	Não há
Possibilidade de executar outros algoritmos	Somente convolução 3X3

A arquitetura deste trabalho tem desempenho superior as outras listadas no capítulo II e a menor área de silício de todas. Assim como a maioria das arquiteturas vistas executa somente algoritmos de convolução do tipo 3x3.

A possibilidade de aproveitar a extensão do barramento (64 bits), com tecnologias mais avançadas, e portanto com menor atraso de propagação nas portas lógicas permite vislumbrar a obtenção de taxas de transferência de dados bastante elevadas.

O controle por uma máquina de estados possibilitou a implementação de características adicionais ao circuito, tais como a possibilidade de interromper o processamento e mudar os coeficientes do "kernel" de convolução.

VII.5 - Indicação de Tendências Futuras

Finalmente, como uma indicação de tendências futuras, seria válido citar que o uso de tecnologias CMOS mais avançadas ou outras tecnologias, tais como ECL permitirão diminuir o tamanho dos circuitos e melhorar seu desempenho. O uso destas tecnologias pode inclusive favorecer o uso do outras configurações de circuitos lógicos.

É importante citar que, devido à falta de ferramentas de "CAD" realizou-se simulações unicamente elétricas e a nível de portas lógicas. O ideal seria se ter realizado inicialmente simulações em alto nível utilizando linguagens do tipo VHDL ou HDL para se validar a lógica e só então se partir para um mapeamento do circuito em portas lógicas de uma determinada tecnologia, o que permitiria o uso dos circuitos aqui desenvolvidos em qualquer tecnologia.

VII.6 - Simulações

Idealmente seria realizada inicialmente a simulação da arquitetura proposta em uma linguagem de alto nível (HDL ou VHDL) com a qual a mesma seria validada, após o qual se realizaria um "mapeamento" da arquitetura numa dada tecnologia realizando-se a simulação a nível de portas lógicas. Como não se dispunha de tais ferramentas foi realizada uma simulação a nível de portas lógicas utilizando-se a biblioteca "standard cell" projetada.

Foram realizadas simulações de cada bloco do circuito para a validação da configuração adotada, os resultados destas serão apresentados na seguinte ordem:

VII.6.1 - Simulação da Lógica de Controle
VII.6.2 - Simulação do Circuito de Multiplexação de Entrada
VII.6.3 - Simulação dos Somadores
VII.6.4 - Simulação do Multiplicador
VII.6.5 - Simulação do Complementador
VII.6.6 - Simulação do Decomplementador

VII.6.1 - Simulação da Lógica de Controle

A simulação realizada testa a máquina de estados em todos os seus estados e o contador síncrono sendo, os sinais de entrada, saída e controle apresentados a seguir.

SINAIS DE ENTRADA

COEF - sinal que indica que o computador "host" está enviando um coeficiente do "kernel" de convolução a ser lido pelo circuito

DADO - sinal que indica que o computador "host" está enviando um dado a ser processado pelo circuito de convolução

SINAIS DE CONTROLE

CK, CKN - sinais de relógio

RST - sinal de inicialização de todos os "flip-flops" do circuito com "0" lógico.

SAÍDAS DO CIRCUITO

E2, E1, E0 - sinais que identificam o estado atual da máquina de estados

C3, C2, C1, C0 - saídas do contador de 4 bits que indica em qual dos 9 circuitos multiplicadores o coeficiente será armazenado

S9, S8, S7, S6, S5, S4, S3, S2, S1 - saídas do decodificador que seleciona o multiplicador no qual o coeficiente será armazenado

CONT9 - saída do contador de 4 bits que indica que o nono coeficiente foi armazenado no respectivo circuito multiplicador

ENCK, ENCKN - sinais que comandam o incremento do contador de 4 bits

ENRST - sinal que habilita a inicialização com "0" lógico do contador de 4 bits

ENDEC - sinal que habilita seleção do multiplicador no qual o coeficiente será armazenado pelo circuito decodificador

ENDADO - sinal que habilita a leitura de dados pelo circuito de convolução habilitando o sinal de relógio a todos os seus registradores

CKDADO - sinal de relógio que comanda o circuito de convolução permitindo sua operação e leitura de dados

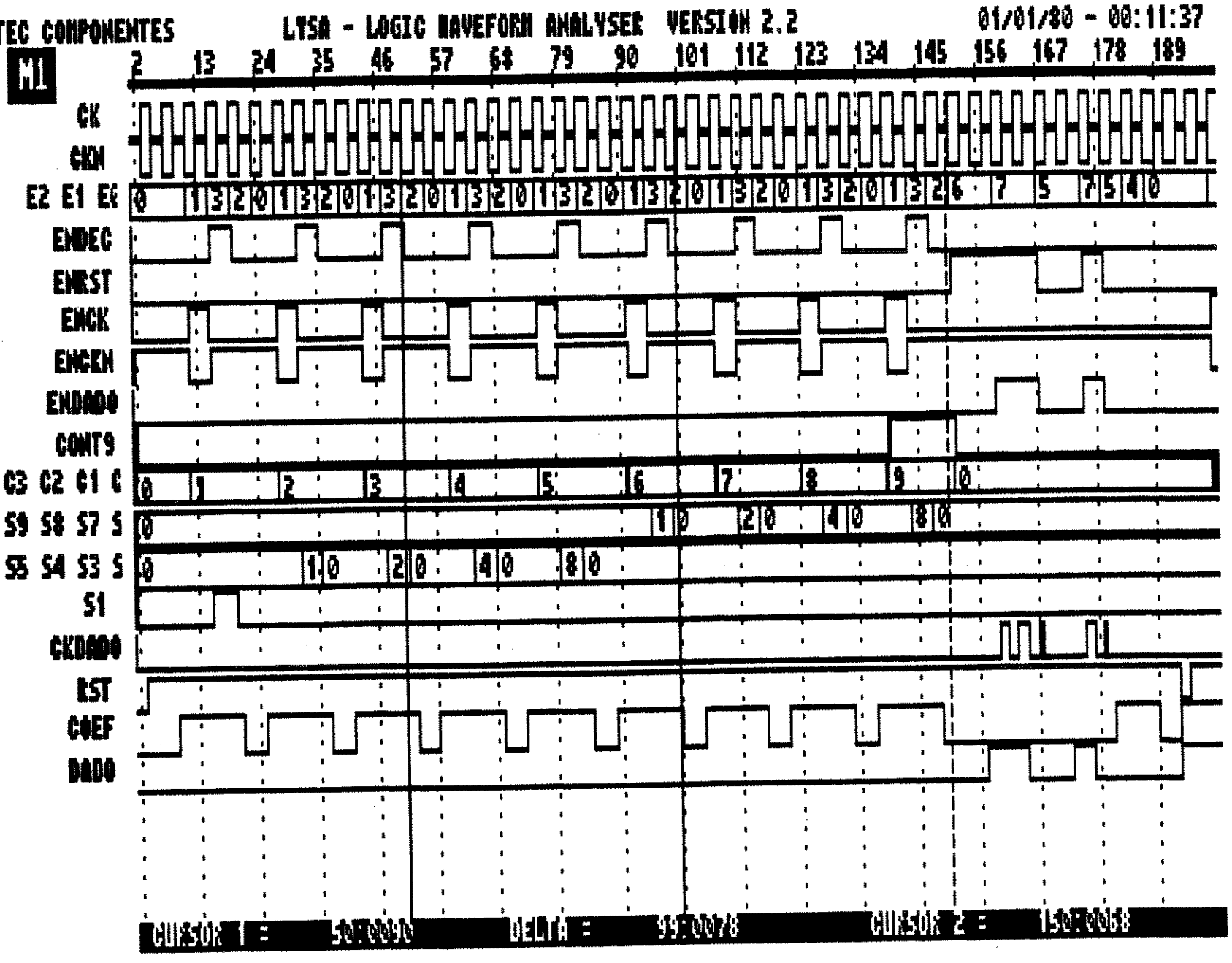
DESCRIÇÃO DA SIMULAÇÃO REALIZADA

Inicialmente todos os "flip-flops" do circuito são inicializados com zeros (RST = 0), assim a máquina de estados também é levada ao estado inicial (E2 E1 E0 = 000) o circuito permanece neste estado até que o sinal COEF assume o valor "1" lógico indicando que o "host" está pronto para enviar um coeficiente do "kernel" de convolução para os multiplicadores.

Sendo COEF=1, a máquina de estados passa ao estado 001 habilitando o incremento do contador de 4 bits através do sinal ENCK, após isso a máquina passa ao estado 011 quando o coeficiente é escrito no primeiro multiplicador (ENDEC=1).

Do estado 011 a máquina vai ao estado 010 onde permanece até que o sinal COEF vá para "0" lógico indicando que o coeficiente não mais está disponível. Uma vez que COEF=0 a máquina vai para o estado inicial "000" onde permanece até que COEF vá a "1" lógico novamente. Este procedimento se repete até que o nono coeficiente seja lido, o que é indicado pelo sinal CONT9, quando então a máquina de estados vai ao estado 110 onde permanece aguardando que o sinal DADO vá a "1" lógico indicando que o "host" está enviando um dado ao circuito; a máquina passa ao estado 111 então onde o sinal de relógio é habilitado aos "flip-flops" do circuito de convolução permitindo o processamento dos dados lidos sucessivamente a cada ciclo de relógio.

A máquina permanece no estado 111 até que o sinal DADO vá a zero lógico indicando que por algum motivo o "host" interromperá o envio de dados, se isto ocorrer a máquina vai ao estado 101 onde permanece aguardando um novo envio de dados pelo "host" (indo então para o estado 111) ou uma nova entrada de coeficientes (indo então ao estado 100 e depois ao estado inicial).


TEC COMPONENTES
LTSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:11:37
M1
2 13 24 35 46 57 68 79 90 101 112 123 134 145 156 167 178 189
CK
CKN
E2 E1 E0
ENDEC
ENRST
ENCK
ENCKN
ENDADO
CONT9
C3 C2 C1 C0
S9 S8 S7 S6
S5 S4 S3 S2
S1
CKDADO
RST
COEF
DADO
CURSOR 1 =
DELTA =
CURSOR 2 =

VII.6.2 - Simulação do Circuito de Multiplexação de Entrada

Foi simulado o circuito de multiplexação de entrada verificando-se assim o funcionamento do circuito na leitura de dados do barramento de 64 bits e a escrita no barramento de 8 bits.

Os sinais de entrada, controle e saída do circuito são apresentados a seguir.

SINAIS DE ENTRADA

H7, H6, H5, H4, H3, H2 , H1 , H0 - sinais de entrada do barramento de 64 bits

G7, G6, G5, G4, G3, G2 , G1 , G0 - sinais de entrada do barramento de 64 bits

F7, F6, F5, F4, F3, F2 , F1 , F0 - sinais de entrada do barramento de 64 bits

E7, E6, E5, E4, E3, E2 , E1 , E0 - sinais de entrada do barramento de 64 bits

D7, D6, D5, D4, D3, D2 , D1 , D0 - sinais de entrada do barramento de 64 bits

C7, C6, C5, C4, C3, C2 , C1 , C0 - sinais de entrada do barramento de 64 bits

B7, B6, B5, B4, B3, B2 , B1 , B0 - sinais de entrada do barramento de 64 bits

A7, A6, A5, A4, A3, A2 , A1 , A0 - sinais de entrada do barramento de 64 bits

CAR - sinal que indica que um novo dado de 64 bits está disponível no barramento

SINAIS DE CONTROLE

CK, CKN - sinais de relógio

RST - sinal de inicialização de todos os "flip-flops" do circuito com "0" lógico.

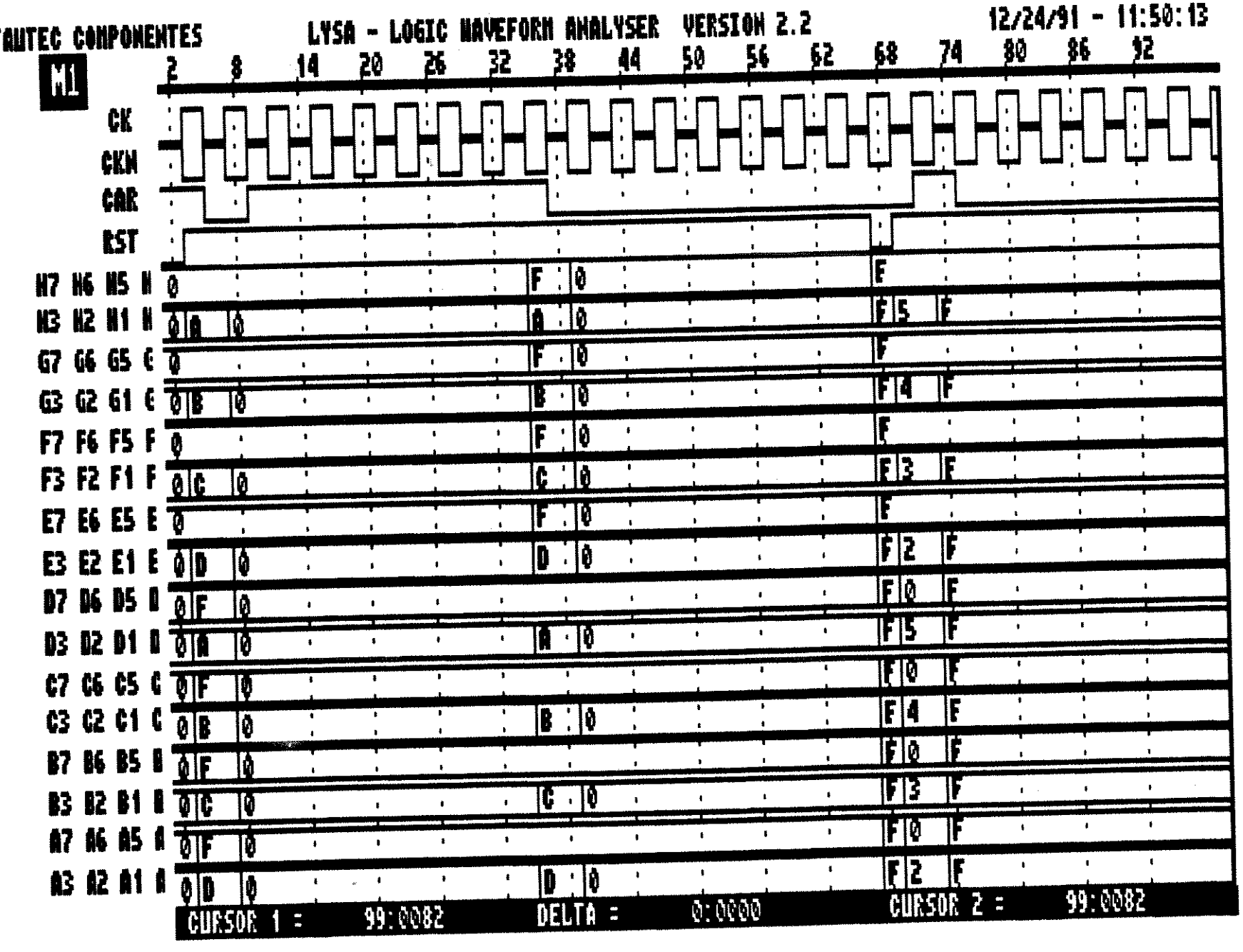
SAÍDAS DO CIRCUITO

OUT7, OUT6, OUT5, OUT4, OUT3, OUT2, OUT1, OUT0 - saída do circuito (8 bits).

DESCRIÇÃO DA SIMULAÇÃO

Inicialmente todos os "flip-flops" do circuito são inicializados com "0" lógico. Após isto, o sinal CAR vai a "0" lógico indicando que um dado de 64 bits está disponível no barramento.

No próximo ciclo de relógio o dado de 64 bits é carregado nos registradores de deslocamento. Após o carregamento se terá na saída um dado de 8 bits a cada ciclo de relógio.


AUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
12/24/91 - 11:50:13
M1
2 8 14 20 26 32 38 44 50 56 62 68 74 80 86 92
CK
CKN
CAR
RST
H7 H6 H5 H
H3 H2 H1 H
G7 G6 G5 G
G3 G2 G1 G
F7 F6 F5 F
F3 F2 F1 F
E7 E6 E5 E
E3 E2 E1 E
D7 D6 D5 D
D3 D2 D1 D
C7 C6 C5 C
C3 C2 C1 C
B7 B6 B5 B
B3 B2 B1 B
A7 A6 A5 A
A3 A2 A1 A
CURSOR 1 = 99:0082
DELTA = 0:0000
CURSOR 2 = 99:0082

TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
12/24/91 - 11:50:13
OUT7 OUT6 OUT5
OUT3 OUT2 OUT1
= 1 HOSKINS
= 2 HOSKINS

VII.6.3 - SIMULAÇÃO DOS SOMADORES

Foram realizadas as simulações de três somadores usados no circuito:

VII.6.3.1 - Somador de 14 + 14 + 14

VII.6.3.2 - Somador de 14 + 14 + 16

VII.6.3.3 - Somador de 14 + 14 + 17

Os sinais de entrada, controle e saída são listados a seguir para cada somador.

VII.6.3.1 - SOMADOR DE 14 + 14 + 14

SINAIS DE ENTRADA

A13 - A0 - dado de entrada
B13 - B0 - dado de entrada
C13 - C0 - dado de entrada

SINAIS DE CONTROLE

CK, CKN - sinais de relógio
RST - sinal de inicialização de todos os "flip-flops" do circuito com "0" lógico.

SAÍDAS DO CIRCUITO

S11 - S0 - saídas da CSA (bits de soma)
C11 - S0 - saídas da CSA (bits de transporte)
S015 - S00 - resultado da soma

VII.6.3.2 - SOMADOR DE 14 + 14 + 16

SINAIS DE ENTRADA

A13 - A0 - dado de entrada
B13 - B0 - dado de entrada
C13 - C0 - dado de entrada
D15 - D0 - dado de entrada

SINAIS DE CONTROLE

CK, CKN - sinais de relógio

RST - sinal de inicialização de todos os "flip-flops" do circuito com "0" lógico.

SAÍDAS DO CIRCUITO

SA17 - S00 - resultado da soma

VII.6.3.3 - SOMADOR DE 14 + 14 + 17

SINAIS DE ENTRADA

A13 - A0 - dado de entrada

B13 - B0 - dado de entrada

C13 - C0 - dado de entrada

C16 - C0 - dado de entrada

SINAIS DE CONTROLE

CK, CKN - sinais de relógio

RST - sinal de inicialização de todos os "flip-flops" do circuito com "0" lógico.

SAÍDAS DO CIRCUITO

S11 - S0 - saídas da CSA (bits de soma)

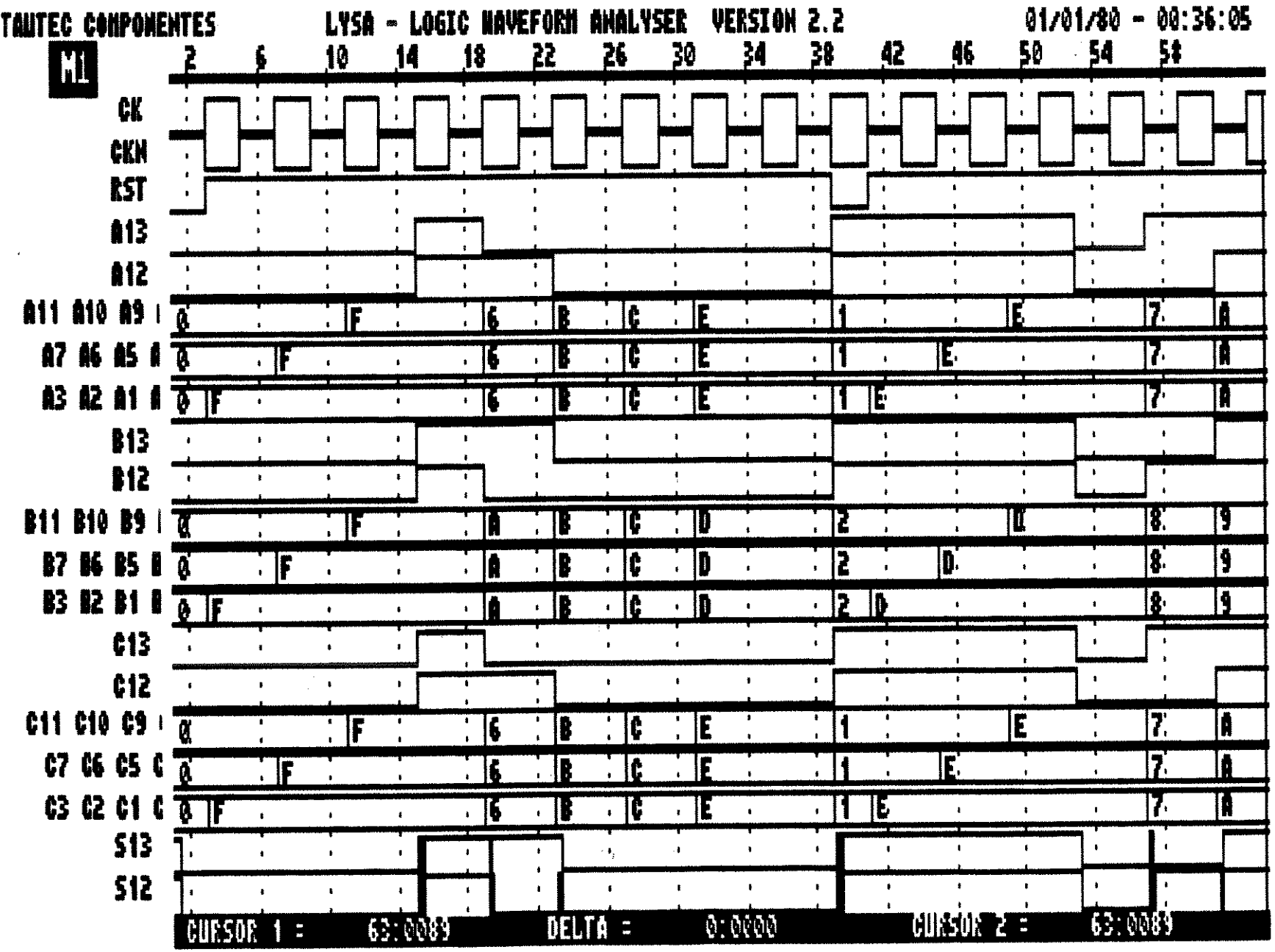
C11 - S0 - saídas da CSA (bits de transporte)

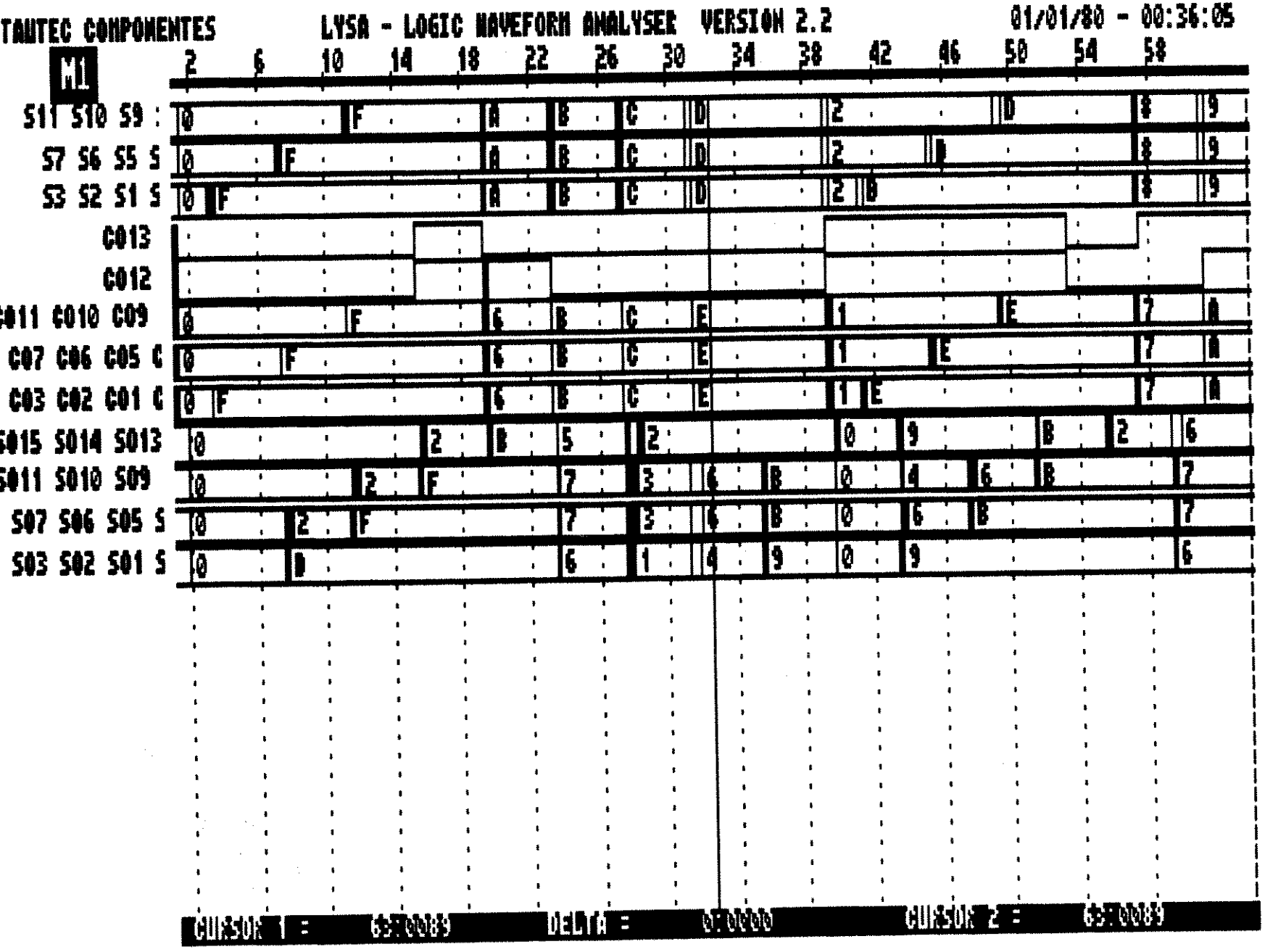
S015 - S00 - resultado da soma

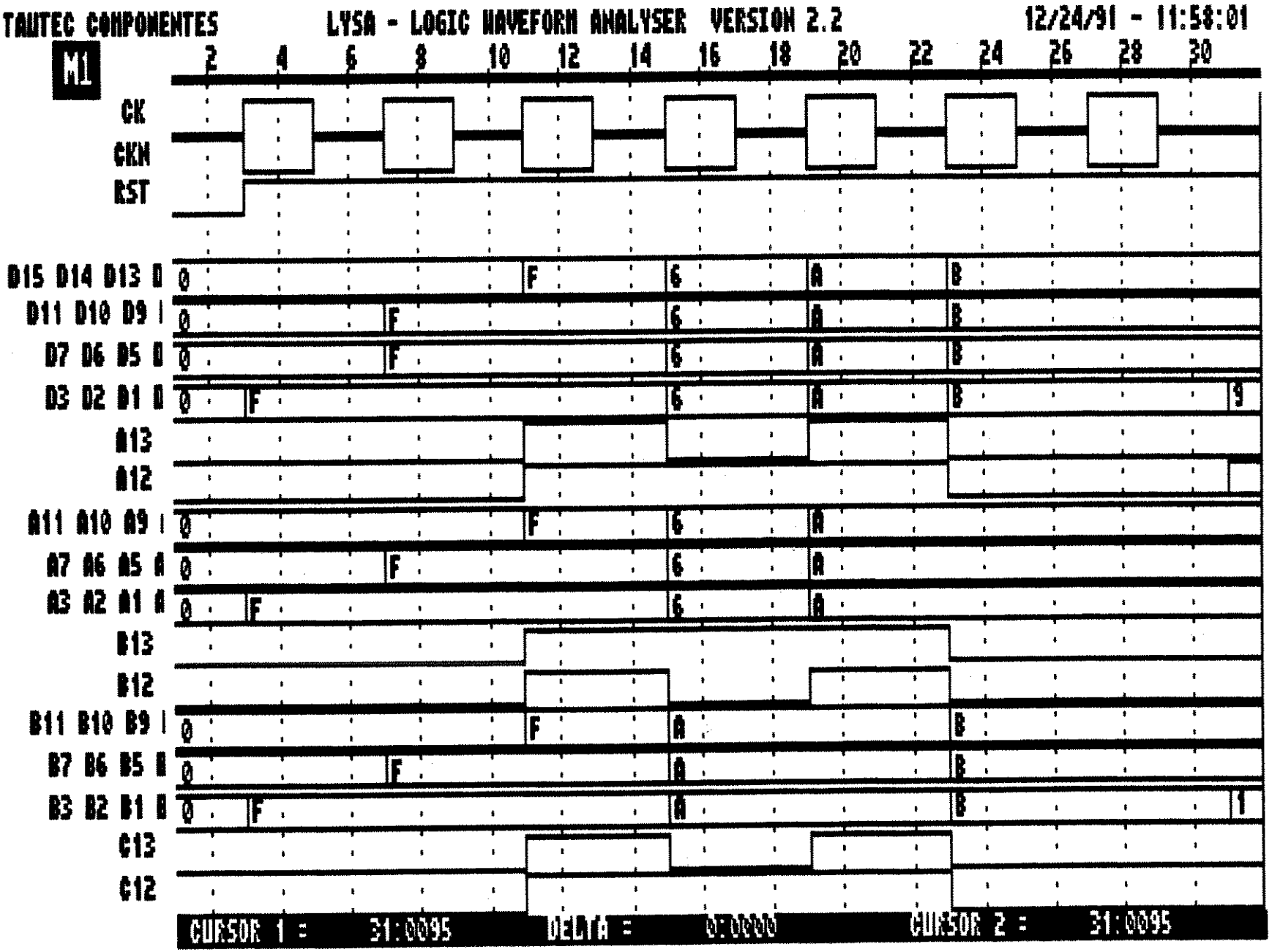
DESCRIÇÃO DA SIMULAÇÃO

Inicialmente todos os "flip-flops" do circuito são inicializados com "0" lógico através do sinal RST (RST=0), após isso, a cada ciclo de relógio 3 dados (A, B e C) são introduzidos no circuito para serem somados.

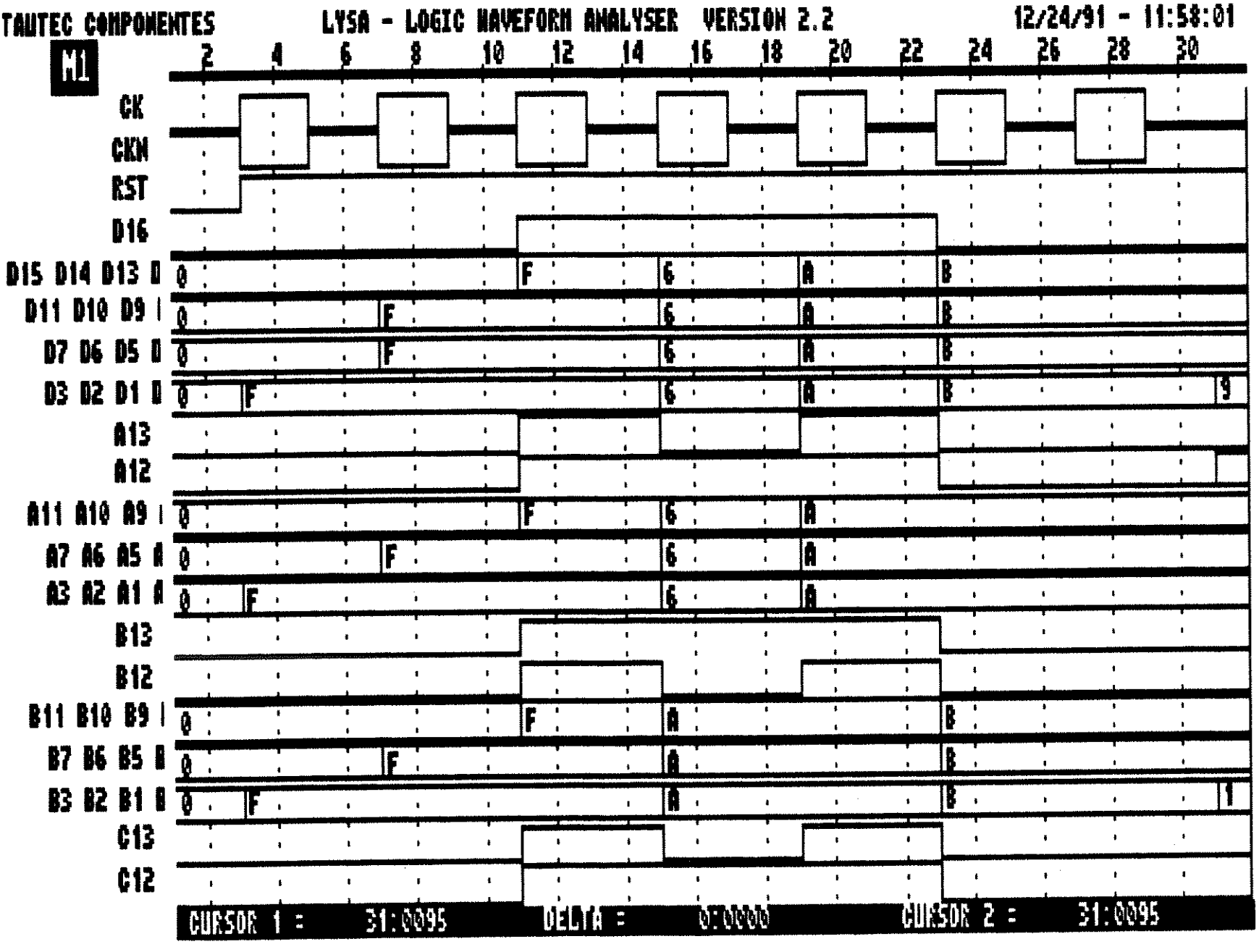
Após o tempo de latência da máquina obtém-se à saída um resultado a cada ciclo de relógio (S0ii).

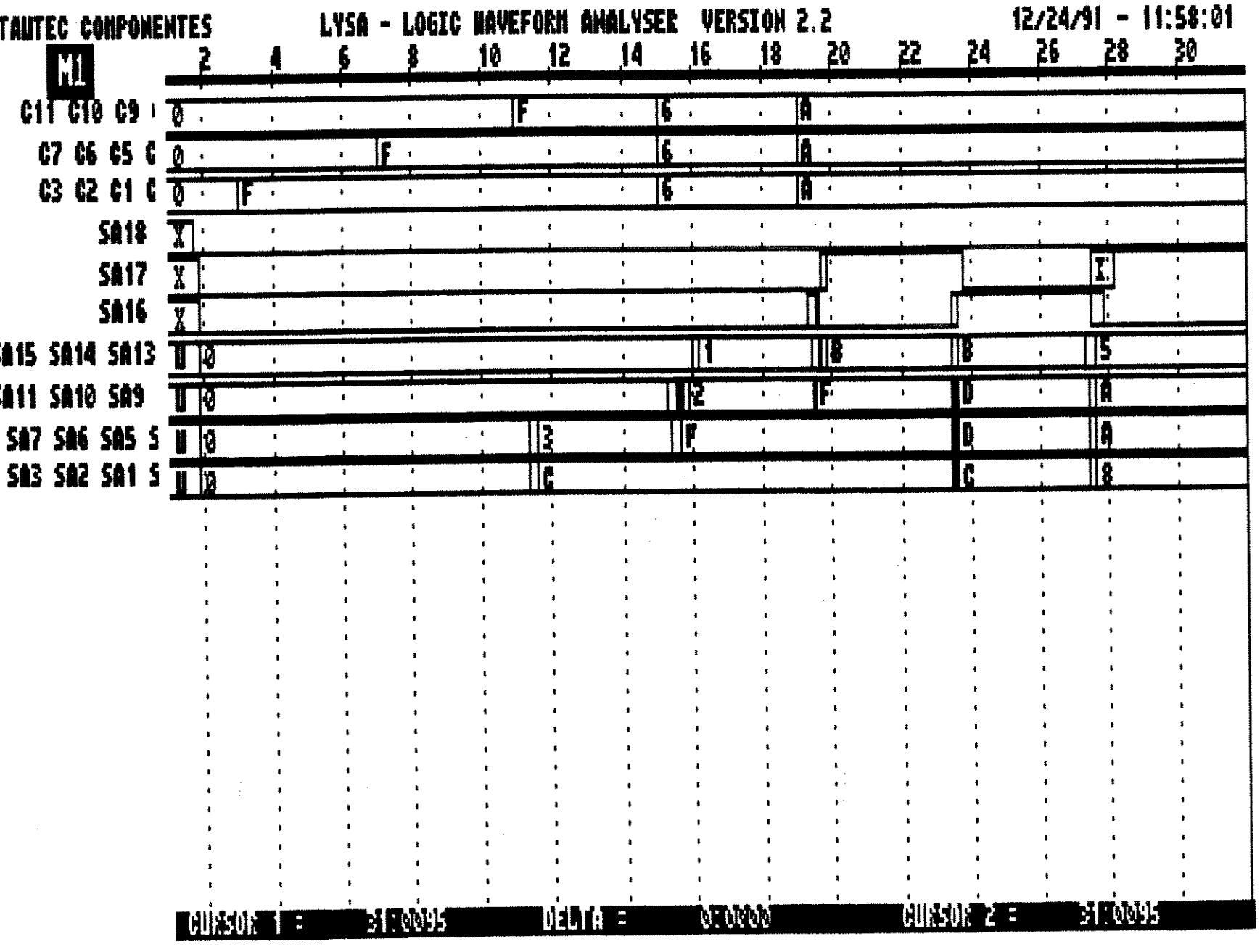

TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:36:05
M1
2 6 10 14 18 22 26 30 34 38 42 46 50 54 58
CK
CKN
RST
A13
A12
A11 A10 A9
A7 A6 A5
A3 A2 A1
B13
B12
B11 B10 B9
B7 B6 B5
B3 B2 B1
C13
C12
C11 C10 C9
C7 C6 C5
C3 C2 C1
S13
S12
CURSOR 1 = 63:0089
DELTA = 0:0000
CURSOR 2 = 63:0089


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:36:05
M1
2 6 10 14 18 22 26 30 34 38 42 46 50 54 58
S11 S10 S9 :
S7 S6 S5 S
S3 S2 S1 S
C013
C012
C011 C010 C09
C07 C06 C05 C
C03 C02 C01 C
S015 S014 S013
S011 S010 S09
S07 S06 S05 S
S03 S02 S01 S
CURSOR 1 =
DELTA =
CURSOR 2 =


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
12/24/91 - 11:58:01
M1
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
CK
CKN
RST
D15 D14 D13 D
D11 D10 D9 D
D7 D6 D5 D
D3 D2 D1 D
A13
A12
A11 A10 A9 A
A7 A6 A5 A
A3 A2 A1 A
B13
B12
B11 B10 B9 B
B7 B6 B5 B
B3 B2 B1 B
C13
C12
CURSOR 1 = 31:0095
DELTA = 0:0000
CURSOR 2 = 31:0095

TAITEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
12/24/91 - 11:58:01
CLK
S16
S17


TAUTEC COMPOSANTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
12/24/91 - 11:58:01
M1
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
CK
CKN
RST
D16
D15 D14 D13 D
D11 D10 D9 D
D7 D6 D5 D
D3 D2 D1 D
A13
A12
A11 A10 A9 A
A7 A6 A5 A
A3 A2 A1 A
B13
B12
B11 B10 B9 B
B7 B6 B5 B
B3 B2 B1 B
C13
C12
CURSOR 1 = 31:0095
DELTA = 0:0000
CURSOR 2 = 31:0095


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
12/24/91 - 11:58:01
M1
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
C11 C10 C9
C7 C6 C5
C3 C2 C1
SA18
SA17
SA16
SA15 SA14 SA13
SA11 SA10 SA9
SA7 SA6 SA5
SA3 SA2 SA1
CURSOR 1 =
DELTA =
CURSOR 2 =

VII.6.4 - SIMULAÇÃO DO MULTIPLICADOR

Foi realizada uma simulação para verificar a multiplicação do dado de 8 bits pelo coeficiente de 4 bits.

SINAIS DE ENTRADA

IN7, IN6, IN5, IN4, IN3, IN2, IN1, IN0 - dado de entrada
COEF3, COEF2, COEF1, COEF0 - coeficiente

SINAIS DE CONTROLE

CK, CKN - sinais de relógio
RST - sinal de inicialização de todos os "flip-flops" do circuito com "0" lógico.

SAÍDAS DO CIRCUITO

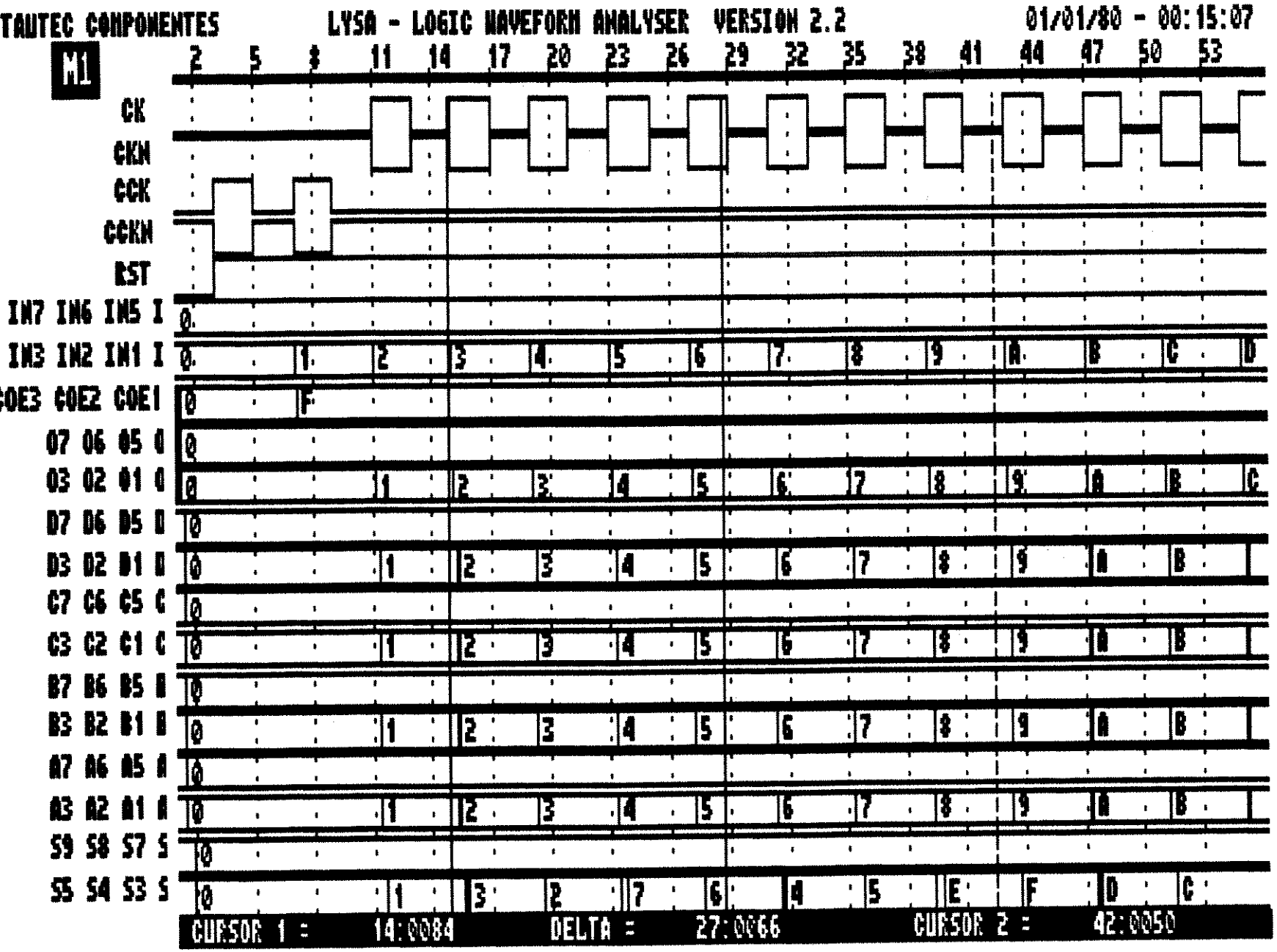
D7 - D0 - parcelas geradas internamente representando o produto de cada bit do coeficiente pelo dado
C7 - C0 - parcelas geradas internamente representando o produto de cada bit do coeficiente pelo dado
B7 - B0 - parcelas geradas internamente representando o produto de cada bit do coeficiente pelo dado
A7 - A0 - parcelas geradas internamente representando o produto de cada bit do coeficiente pelo dado
S9 - S0 - saídas de soma da primeira CSA
CS9 - CS0 - saídas de transporte da primeira CSA
R11-R0 - saídas de soma da segunda CSA
CP11 - CP0 - saídas de transporte da segunda CSA
SP12 - SP0 - saídas da CPA (resultado da multiplicação)

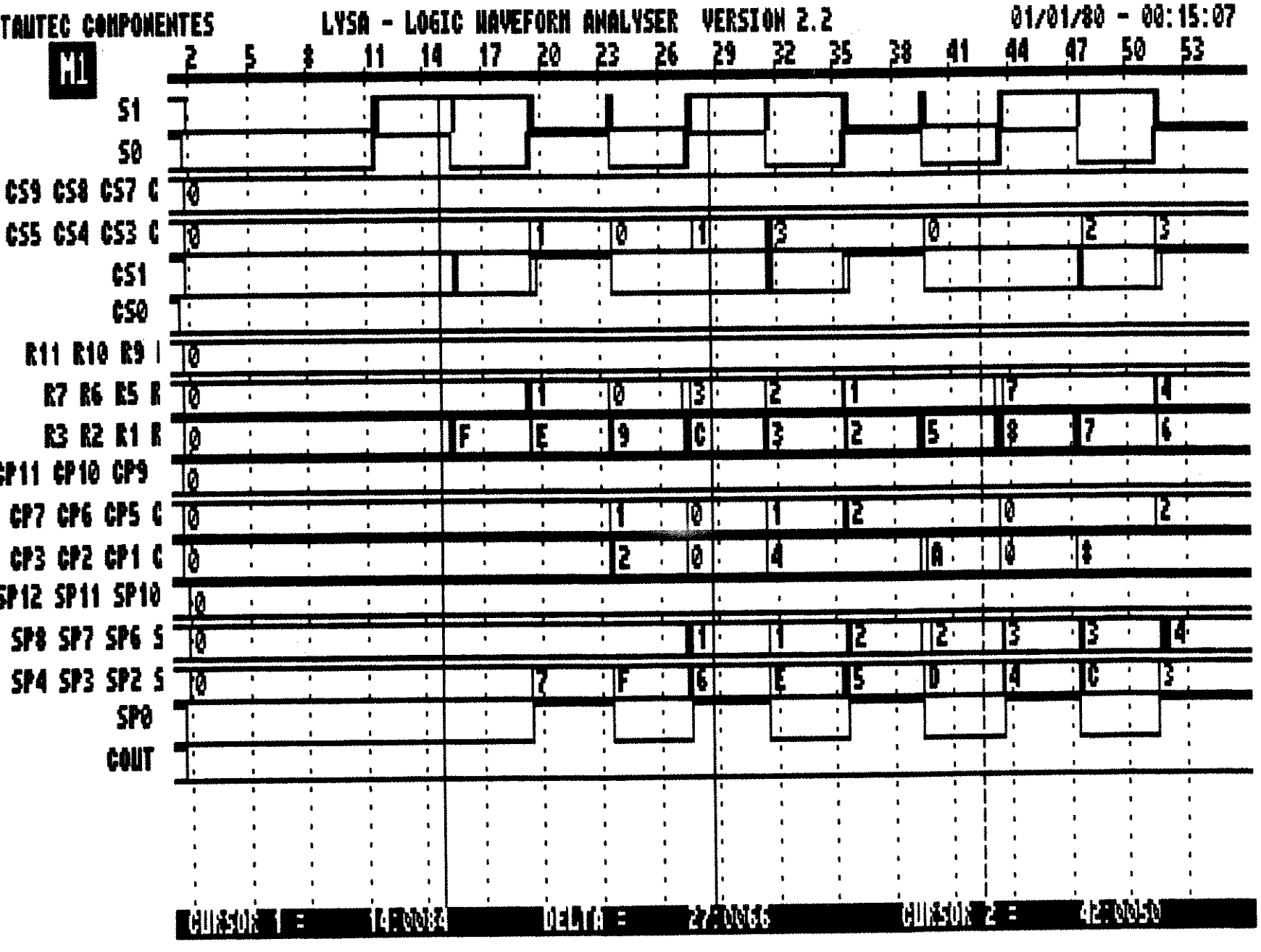
DESCRIÇÃO DA SIMULAÇÃO

Inicialmente todos os "flip-flops" do circuito são inicializados com "0" lógico através do sinal RST (RST=0), após isto é carregado o coeficiente no registrador de coeficiente, operação que é comandada pelos sinais CCK e CCKN enviados pela máquina de controle.

Uma vez carregado o coeficiente, será lido um dado a cada ciclo de relógio calculando-se o produto deste pelo coeficiente.

Após o tempo de latência do multiplicador observa-se um resultado a cada ciclo de relógio.


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:15:07
M1
2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53
CK
CKN
CCK
CCKN
RST
IN7 IN6 IN5 I
IN3 IN2 IN1 I
COE3 COE2 COE1
O7 O6 O5 O
O3 O2 O1 O
D7 D6 D5 D
D3 D2 D1 D
C7 C6 C5 C
C3 C2 C1 C
B7 B6 B5 B
B3 B2 B1 B
A7 A6 A5 A
A3 A2 A1 A
S9 S8 S7 S
S5 S4 S3 S
CURSOR 1 = 14:0084
DELTA = 27:0066
CURSOR 2 = 42:0050


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:15:07
M1
2 5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53
S1
S0
CS9 CS8 CS7 C
CS5 CS4 CS3 C
CS1
CS0
R11 R10 R9 I
R7 R6 R5 R
R3 R2 R1 R
CP11 CP10 CP9
CP7 CP6 CP5 C
CP3 CP2 CP1 C
SP12 SP11 SP10
SP8 SP7 SP6 S
SP4 SP3 SP2 S
SP0
COUT
CURSOR 1 =
DELTA =
CURSOR 2 =

VII.6.5 - Simulação do Complementador

Foi realizada a simulação do circuito complementador para demonstrar o funcionamento do circuito. Os sinais de entrada, controle e saída são listados a seguir.

SINAIS DE ENTRADA

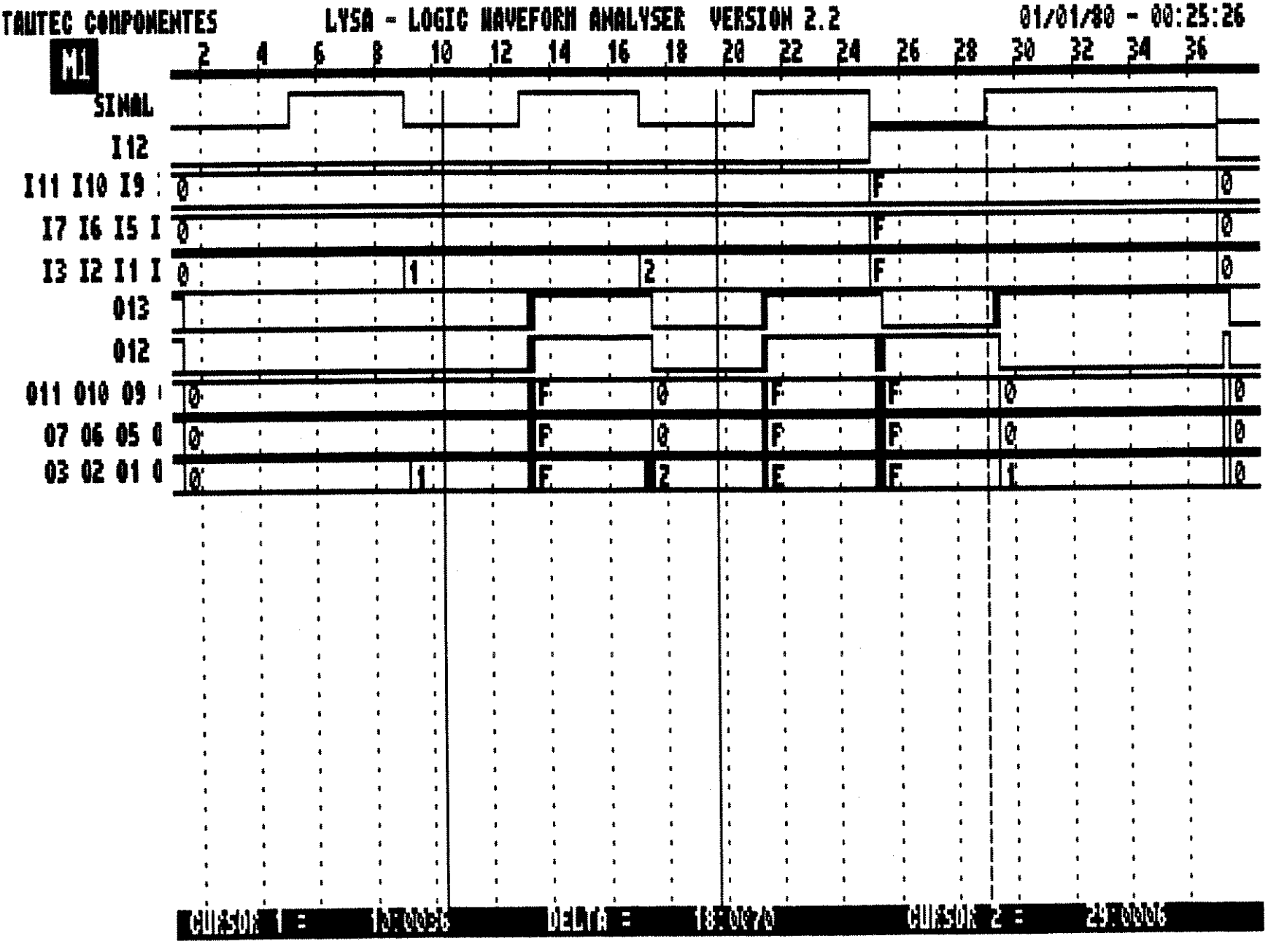
SINAL - sinal do dado (que é na forma sinal-magnitude)
I12 - I0 - dado de entrada

SAÍDAS DO CIRCUITO

O15 - O0 - dado de saída na forma de complemento de 2

DESCRIÇÃO DA SIMULAÇÃO

O dado é levado ao circuito sob a forma sinal-magnitude e é convertido para a forma de complemento de 2.


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:25:26
M1
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
SIGNAL
I12
I11 I10 I9
I7 I6 I5 I
I3 I2 I1 I
O13
O12
O11 O10 O9
O7 O6 O5 O
O3 O2 O1 O
CURSOR 1 =
DELTA =
CURSOR 2 =

VII.6.6 - Simulação do Decomplementador

Foi realizada a simulação do circuito decomplementador para demonstrar o funcionamento do circuito. Os sinais de entrada, controle e saída são listados a seguir.

SINAIS DE ENTRADA

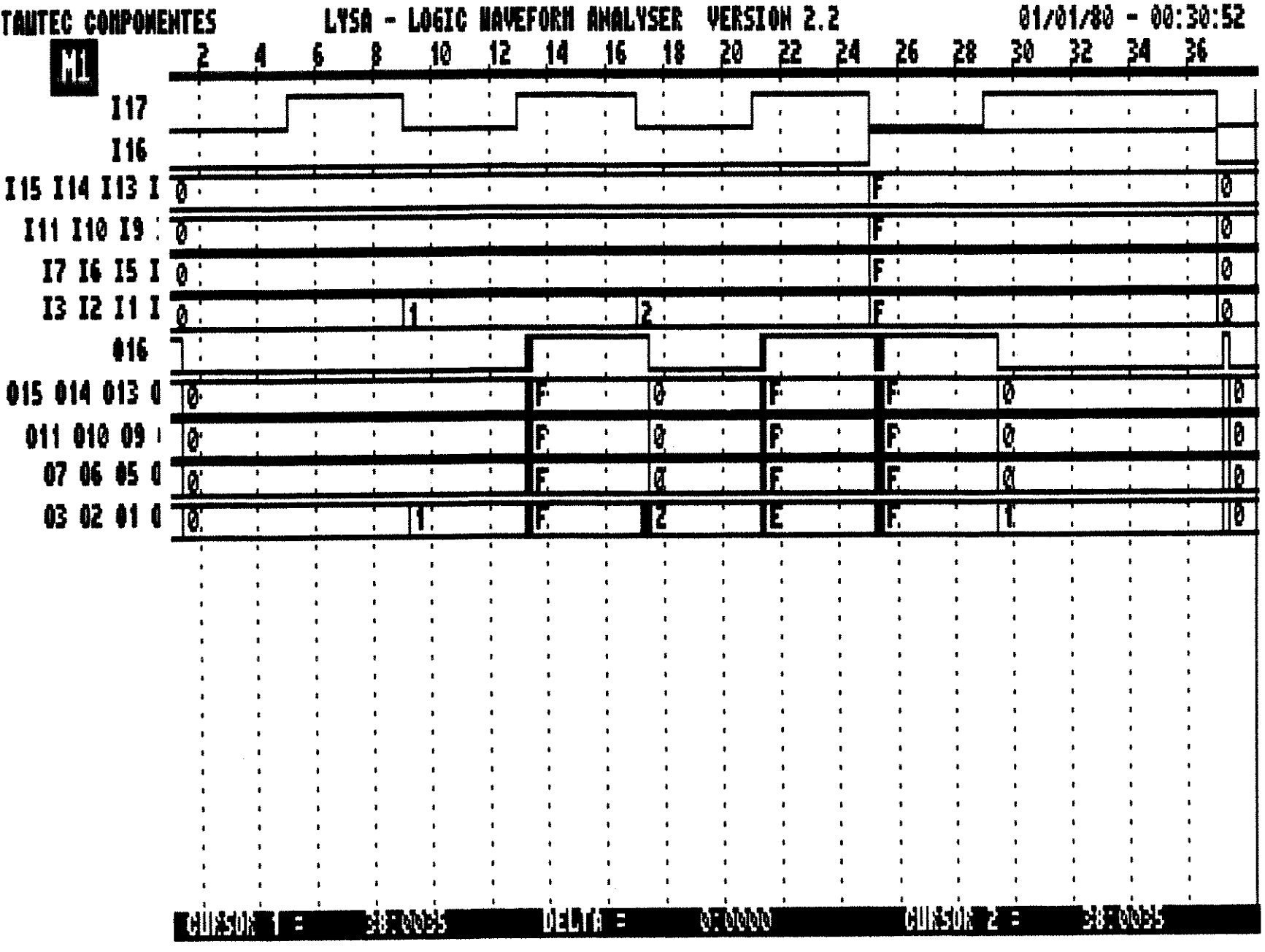
I17 - I0 - dado de entrada na forma de complemento de 2

SAÍDAS DO CIRCUITO

O16 - O0 - dado de saída

DESCRIÇÃO DA SIMULAÇÃO

O dado é levado ao circuito sob a forma de complemento de 2 e é convertido para a forma de sinal-magnitude.


TAUTEC COMPONENTES
LYSA - LOGIC WAVEFORM ANALYSER VERSION 2.2
01/01/80 - 00:30:52
M1
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
I17
I16
I15 I14 I13 I
I11 I10 I9 :
I7 I6 I5 I
I3 I2 I1 I
016
015 014 013 0
011 010 09 -
07 06 05 0
03 02 01 0
CURSOR 1 = 38.0085
DELTA = 0.0000
CURSOR 2 = 38.0085

Referências Bibliográficas

1 ANCEAU, F. & Reis, R.. Design Strategy Prentice/Hall International. VLSI Architecture, Prentice Hall International. England. 1983. p. 128-148.
Inc., 1983

2 AONO, Kunitoshi et alii. Implementation of a Bipolar Real-Time Image Signal Processor - RISP-II. IEEE Journal of Solid State Circuits. USA. The Institute of Electrical and Electronics Engineers Inc. *sc-22*(3):403-408. 1987.

3 ARAMBEPOLA, Bernard et alii. Cascadable One/Two-Dimensional Digital Convolver", IEEE Journal of Solid-State Circuits. USA. The Institute of Electrical and Electronics Engineers Inc. *23*(2):351-357. 1988.

4 CASTANHO, José Eduardo Cogo. Uma Estação de Trabalho para Visão Computacional. Campinas, Brasil. Universidade Estadual de Campinas. 1990. 140p.

5 CHAPMAN, Ken. Finding Circles with Sobel Algorithms. ESD: The Electronic System Magazine. USA. Ago. 1987.

6 ERDELYI, Charles K. et alii. Design Methodologies. North-Holland. 1986.

7 GLASSER, Lance A. & Daniel W. Dobberpuhl. The Design and Analysis of VLSI Circuits. USA. Addison-Wesley Publishing Company. 1986. 473p.

8 GOLD AMI, Inc. Semicustom Design Seminar. USA. 1987. p. 89-96

9 GONZALES, Rafael C. & Paul Wintz. Digital Image Processing. USA. Addison-Wesley Publishing Company. 1987. 503p.

10 GROB, Bernard. Televisão Básica Princípios e Reparação. Basic Television Principles and Servicing. Trad. Ivan José Albuquerque. Brasil. Editora Guanabara. 1987. 520p.

11 HELLER, W.R.et alii. Prediction of Wiring Space Requirements for VLSI. Hu, T.C. & Hu, Ernest S. VLSI Circuit Layout: theory and Design. New York, USA. 1985. p.62-75.

12 HWANG, Kai. Computer Arithmetic - Principles Architecture and Design. John Wiley and Sons. 1979. 423p.

13 HWANG, Kai & Briggs, Fayé A. Computer Architecture and Paralell Processing. McGraw-Hill Book Company. 1985. 846p.

14 KANOPOULOS, Nick et alii. Design of an Image Edge Detection Filter Using the Sobel Operator. IEEE Journal of Solid-State Circuits. USA. The Institute of Electrical and Electronic Engineers Inc. 23(2):358-367. 1988.

15 MAMMANA, Carlos Ignácio Zamitti et alii. Manual do Aluno. Brasil. Escola Brasileiro-Argentina de Informática. 1988. 174p.

16 NOIJE, W.A.M e Romão F.L. More Flexible Sea-of-Gates Structure Sociedade Brasileira de Microeletrônica. Anais do V Congresso da Sociedade Brasileira de Microeletrônica. Brasil. 1990 p.372-373

17 RUETZ, Peter A. & Robert W. Brodersen. Architectures and Design Techniques for Real-Time Image-Processing IC's. IEEE Journal of Solid-State Circuits. USA. Institute of Electrical and Electronics Engineers Inc. sc-22(2). 1987. p.233-250.

18 SECHEN, Carl. The TimberWolf3.2 Standard Cell placement and Global Routing Program - User's Guide. 1986.

19 TOZZI, Clésio Luís et alli. An Image Processing System Based on Algorithmically Dedicated Functional Units SPIE - the International Society for Optical Engineering. Analls of the 1990 Symposium on Laser Science and Optics Applications. Boston, USA. 1990.

APÊNDICE I

Layout dos Circuitos Integrados Protótipo Difundidos

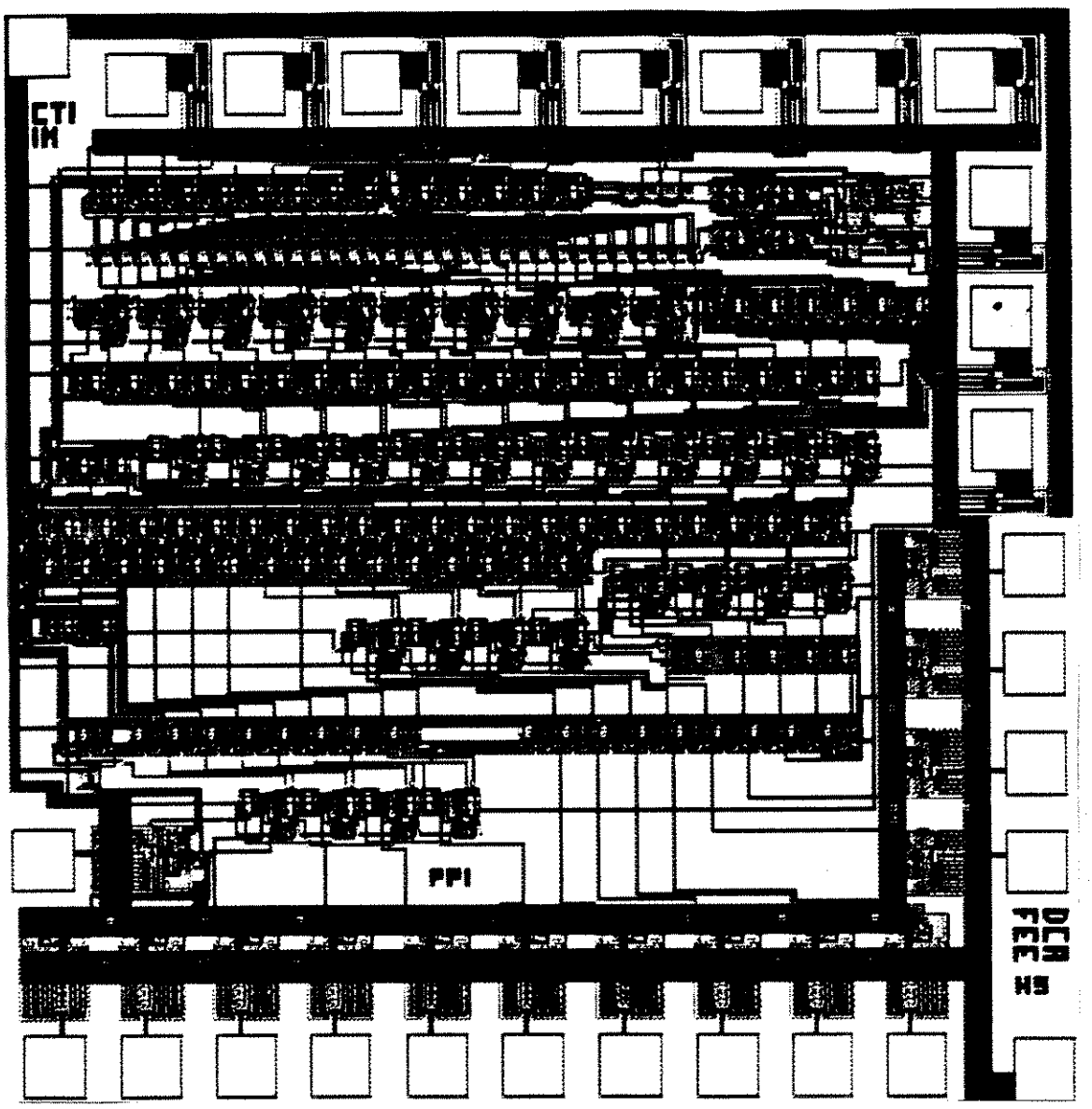


Figura 1 - Desenho Físico do Circuito Multiplicador 8X4

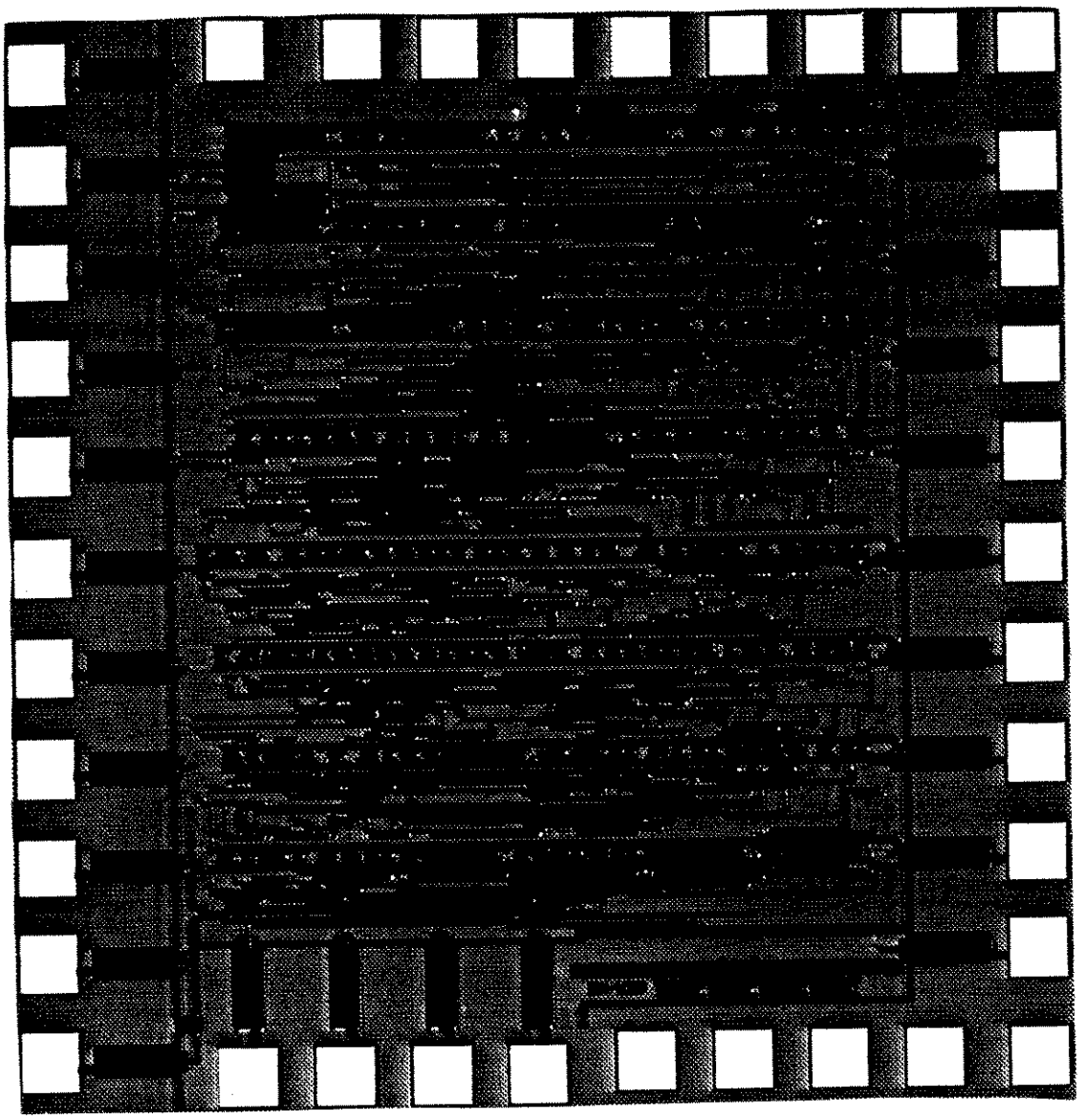


Figura 2 - Desenho Físico do Circuito Somador 4 + 4 + 4

APÊNDICE II

CARACTERÍSTICAS DA PORTAS LÓGICAS DA BIBLIOTECA DE CÉLULAS- PADRÃO

Os atrasos no simulador a ser utilizado são calculados da seguinte forma:

$$t_{atraso} = (t_{PHL} + t_{PLH})/2$$

$$t_{PHL} = K_d * C + t_{dxd}$$

$$t_{PLH} = K_s * C + t_{dxs}$$

Onde:

t_{atraso} = o tempo de atraso da porta calculado pelo simulador

t_{PHL} = atraso de descida do sinal de saída com relação às entradas

t_{PLH} = atraso de subida do sinal de saída com relação às entradas

K_d = coeficiente que dá o acréscimo de atraso na descida de sinal causado pela carga capacitiva para a qual a porta lógica fornece sinal em ns/fF

K_s = coeficiente que dá o acréscimo de atraso na subida de sinal causado pela carga capacitiva para a qual a porta lógica fornece sinal em ns/fF

C = carga capacitiva para a qual a porta fornece sinal em fF

t_{dxd} = atraso intrínseco de descida do sinal de saída com relação às entradas em ns

t_{dxs} = atraso intrínseco de subida do sinal de saída com relação às entradas em ns

TABELA DE PARÂMETROS DAS PORTAS LÓGICAS

PORTA	K_d	K_s	t_{dxd}	t_{dxs}	C
INV	0,0058	0,0057	0,7885	0,7736	24,44
NAND2	0,0060	0,0108	1,0359	1,4078	18,80
NAND3	0,0050	0,0107	1,2078	1,6267	13,16
NOR2	0,0051	0,0046	1,6854	1,1350	22,56
NOR3	0,0054	0,0082	1,5662	1,7435	16,92
TG	0,0038	0,0040	0,1531	0,1673	12,22
XOR	0,0038	0,0040	0,2000	0,2000	36,66

OBS: C é a capacitância de entrada de cada porta