Universidade Federal de Campina Grande - UFCG

Circuito integrado para extração de fundo não homogêneo de imagens dinâmicas em tempo real.

André Luiz Printes

Dissertação de Mestrado submetida ao Programa de Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande – Campus I, como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica

Área de concentração: Processamento da Informação

Elmar Uwe Kurt Melcher, Dr.
Raimundo Carlos Silvério Freire, Dr.
Orientadores

Campina Grande. Paraíba. Brasil

P954c 2002

Printes, André Luiz

Circuito integrado para extração de fundo não homogêneo de imagens dinâmicas em tempo real/ André Luiz Printes. - Campina Grande: UFCG, 2002.

103 p.: il.

Dissertação (mestrado) - UFCG/DEE
Inclui bibliografia

1. Circuitos integrados 2. ASIC 3. Processamento de Imagem - Segmentação I.Título

CDU: 621.3.049.77

CIRCUITO INTEGRADO PARA EXTRAÇÃO DE FUNDO NÃO HOMOGÊNEO DE IMAGENS DINÂMICAS EM TEMPO REAL VISANDO BAIXO CUSTO

ANDRÉ LUIZ PRINTES

Dissertação Aprovada em 02.12.2002

ELMAR UWE KURT MELCHER, Dr., UFCG
Orientador

RAIMUNDO CARLOS SILVÉRIO FREIRE, Dr., UFCG
Orientador

JOÃO MARQUES DE CARVALHO, Ph.D., UFCG
Componente da Banca

JOSÉ EWERTON POMBO DE FARIAS, Dr., UFCG
Componente da Banca

AMAURI OLIVEIRA, D.Sc., UFBA
Componente da Banca

CAMPINA GRANDE - PB
Dezembro – 2002

Dedicatória

Dedico este trabalho a minha esposa Theia e a meus filhos André e Thais que sempre me incentivaram e souberam compreender minha ausência durante o período de elaboração deste trabalho.

Agradecimentos

Gostaria de registrar meus agradecimentos a todos que longo destes quase 3 anos contribuíram de alguma forma para a conclusão deste trabalho. Em particular aos professores Raimundo Freire e Elmar Melcher que viabilizaram este projeto e possibilitaram realização de um sonho de quase uma década.

Abstract

This thesis describes the implementation of part of a system, aimed at making the extraction of objects from images with static and non-homogeneous backgrounds in real time, capable of tolerating slight global and local brightness variations.

This system proposes a re-configurable hardware implementation of a background subtraction algorithm, which was adapted in order to allow a low cost hardware implementation, and real time operation, to render it suitable for applications in the entertainment industry.

Simulation results are presented, which validate the algorithm functionality for such applications.

A GPIP-01 (General Purpose Image Processor) platform is also presented. This platform has been designed and implemented to achieve the minimum requirements of a prototyping embedded system for real time image processing applications, needed for development of the proposed system.

Resumo

Este trabalho descreve a implementação de parte de um sistema voltado à extração de objeto em imagens com fundo estático e não homogêneo em tempo real, capaz de suportar pequenas variações globais e locais de luminosidade.

O sistema descrito, propõe uma implementação em *hardware* reconfigurável de um algoritmo para extração de objetos, que foi adaptado objetivando permitir uma implementação em *hardware*, com baixo custo e operação em tempo real, visando as aplicações do mercado de entretenimento.

São apresentados resultados de simulações que validam a funcionalidade do algoritmo para as aplicações em questão.

Neste trabalho também é apresentada a plataforma GPIP-01 (General Purpose Image Processor), concebida e implementada com o objetivo de atender aos requisitos mínimos de um sistema embarcado de prototipagem para aplicações em processamento de imagens em tempo real, necessário para o desenvolvimento do sistema proposto.

Sumário

Índice de Figuras

Lista de Símbolos

Símbolo	Descrição
i	Representa uma posição (x,y) qualquer de um *pixel* em um quadro.
E_i	Composto por $\mu_R(i), \mu_G(i), \mu_B(i)$, representa o valor esperado para o *pixel i* na imagem de referência.
$\mu_{i(R)}, \mu_{i(G)}, \mu_{i(B)}$	Valor médio de cada uma das componentes R,G e B do *pixel i* na imagem de referência, calculado em um número N de quadros.
I_i	Composto por $I_R(i), I_G(i), I_B(i)$, representa o valor do *pixel i* na imagem corrente.
α_i	Distorção de Brilho.
$I_{i(R)}, I_{i(G)}, I_{i(B)}$	Cada uma das componentes R,G e B do *pixel i* na imagem corrente.
CD_i	Distorção de Cromaticidade
S_i	Composto por $\sigma_R(i), \sigma_G(i), \sigma_B(i)$, representa o desvio padrão dos valores de cada um dos *pixels* de uma imagem estática de fundo.
$\sigma_{i(R)}, \sigma_{i(G)}, \sigma_{i(B)}$	Desvio Padrão de cada uma das componentes R, G e B de um *pixel i* na imagem de referência, calculado em um número N de quadros.
a_i	Variação da Distorção de Brilho, fator de normalização.
b_i	Variação da Distorção de Cromaticidade, fator de normalização.
M_i	Máscara para classificação dos pixels.
$\hat{CD}_i$	Distorção de cromaticidade normalizada
$\hat{\alpha}_i$	Distorção de brilho normalizada
τ_{CD}	Limiar de decisão para $\hat{CD}_i$.
$\tau_{\alpha 1}$	Limiar superior para $\hat{\alpha}_i$.
$\tau_{\alpha 2}$	Limiar inferior para $\hat{\alpha}_i$.
$\tau_{\alpha lo}$	Limite inferior para o valor da distorção de brilho normalizada.
$P_{i(C)}$	Valor de um *pixel* i sendo C cada uma das componentes de cor R, G e B.

1 Introdução

Nas últimas décadas temos verificado uma grande evolução na área de processamento digital da informação, decorrente principalmente do avanço da capacidade de processamento e das técnicas computacionais, desenvolvidas neste período.

O processamento digital de imagens foi uma das áreas que experimentou maior desenvolvimento, com aplicações nos mais diversos campos, tais como: medicina, prospecção de petróleo, militar, automação industrial, entretenimento, dentre outros [1], Dessa forma, processamento digital de imagens tem despertado grande interesse de pesquisadores das mais diversas áreas do conhecimento.

Os conceitos básicos ligados a processamento de imagens são: a aquisição da imagem, o armazenamento, o processamento, a transmissão e a exibição da imagem resultante. Neste trabalho trataremos do processamento das imagens, em especial da segmentação, que "...é o processo que subdivide uma imagem em suas partes básicas constituintes" [2].

Uma das aplicações do processamento digital de imagens encontra-se no mercado de entretenimento, para o qual vem sendo desenvolvido um número cada vez maior de equipamentos utilizando esta tecnologia. Uma técnica já bastante conhecida e que vem sendo aperfeiçoada no decorrer dos últimos anos é a de extração de objetos de um fundo conhecido, em tempo real. Com esta finalidade são utilizadas diversas técnicas, desde o *chroma key*[1] que efetua a extração, em tempo real, de um objeto desconhecido em um fundo homogêneo e conhecido em ambiente controlado, até técnicas mais sofisticadas tais como a extração de objetos por visão estereoscópica [3] em ambientes desconhecidos, envolvendo alta capacidade de processamento em tempo real.

1. Técnica de extração de objeto cuja finalidade é eliminar o fundo de uma cena (sendo o fundo homogêneo azul o mais utilizado), com o propósito de isolar o objeto de interesse para ser inserido em outra imagem, resultando em uma terceira imagem que é a combinação das duas primeiras. Esta técnica é bastante aplicada e difundida em estúdios de produção de vídeos. O termo *chroma key*, na linguagem de origem, é largamente utilizado na área de processamento de imagens. Uma tentativa de tradução poderia desvirtuar e comprometer o entendimento do texto.

O objetivo proposto neste trabalho é dar um ponto de partida para o desenvolvimento de um sistema capaz de capturar uma imagem através de uma câmera de TV colorida convencional, extrair o objeto principal de um fundo qualquer heterogêneo, mas relativamente fixo e conhecido, para em seguida inserir este objeto, sincronizado, em uma segunda imagem de fundo diferente, proveniente de outra fonte de vídeo qualquer, em tempo real.

O foco deste trabalho refere-se à principal diferença entre o sistema descrito acima e o *chroma key* [4] já existente, que é a proposta de extrair, em tempo real, o objeto de um fundo qualquer, heterogêneo, mas relativamente fixo e conhecido. Este sistema deve ser robusto o suficiente para reconhecer pequenas variações de iluminação generalizadas ou localizadas (sombras), que são inerentes a ambientes não controlados. Nos últimos anos, trabalhos propondo algoritmos e técnicas direcionadas a este propósito vêm sendo publicados. Baseando-se nesses trabalhos e fazendo-se adaptações pode-se chegar a implementação da funcionalidade pretendida com esta dissertação. As principais adaptações a serem efetuadas decorrem do fato de que, para a implementação da funcionalidade em tempo real um aspecto a ser considerado é a velocidade de processamento, permitindo a execução das demais etapas em tempo adequado à exibição de uma imagem gerada a uma taxa de 30 quadros por segundo. Outro aspecto relevante é que, por tratar-se de uma aplicação voltada para entretenimento e mercado de consumo, o sistema também deve visar baixo custo. Dessa forma a implementação da funcionalidade aqui descrita se dará através do desenvolvimento de um *hardware* que terá como elemento principal um "Circuito integrado ASIC[2] para extração de fundo não homogêneo de imagens dinâmicas em tempo real", visando baixo custo. Este trabalho visa estabelecer um ponto de partida para a concretização deste objetivo.

O escopo deste trabalho limita-se ao processo de extração de objeto, bem como o desenvolvimento de uma plataforma de *hardware* para aplicações em processamento de imagens em tempo real, capaz de efetuar a aquisição, armazenamento, processamento e exibição da imagem processada.

[2] ASIC – Application Specific Integrated Circuit (Circuito Integrado de Aplicação Específica)

Neste trabalho é descrito o processo de desenvolvimento de um sistema a ser implementado em um circuito integrado para a execução da funcionalidade proposta acima, o texto é composto por oito capítulos, assim organizados:

No capítulo 2 é feita uma revisão bibliográfica onde são mostrados as principais técnicas e métodos atualmente utilizados para a extração de objetos de uma imagem enfatizando as principais vantagens e desvantagens de cada um em relação aos demais. Também serão descritos os principais problemas relativos à aplicação em questão.

No capítulo 3 é descrito detalhadamente o algoritmo escolhido para a solução do problema, suas características, vantagens e desvantagens concernentes à aplicação proposta, bem como os resultados obtidos por simulação.

No capítulo 4 é apresentada a implementação da plataforma de *hardware* chamada GPIP-01 (General Purpose Image Processor), concebida com o objetivo de atender aos requisitos mínimos de um sistema para aplicações em processamento de imagens em tempo real, necessários para o desenvolvimento da aplicação proposta neste trabalho.

No capítulo 5 é apresentada a descrição do processo a ser implementado para a realização do algoritmo proposto, são descritas as adaptações necessárias à aplicação em questão, diagramas de blocos e máquinas de estados envolvidos no processo de extração de objeto baseado na técnica adotada.

No capítulo 6 são descritos os resultados atingidos ao final deste trabalho, no capítulo 7 são apresentadas as conclusões e no capítulo 8 são feitas sugestões para trabalhos futuros.

2 Extração de Objeto

A extração de objetos de uma seqüência de vídeo é um problema fundamental e crucial para muitos sistemas de visão tais como vídeo de vigilância, monitoração de tráfego, detecção e rastreamento de pessoas em vídeo teleconferência, edição de vídeo e mais recentemente para a descrição de conteúdo no contexto de MPEG-7 [5], além de muitas outras aplicações [6].

Nos sistemas de segmentação em processamento de imagem, existem várias técnicas que se propõem a extrair objetos de uma cena, dentre as quais podemos citar: *Chroma key* [4], Extração de Objetos por Subtração de Fundo [6, 7], Detecção de Objeto em Imagem Codificada em MPEG [8, 9], além de outras técnicas cujo grau de complexidade é determinado pelo problema a ser resolvido.

O objetivo deste capítulo é descrever as técnicas mais usadas na extração de objetos em movimento em um fundo fixo, suas aplicações, bem como suas principais vantagens e desvantagens em relação às demais. A seguir será feita uma breve explanação das principais técnicas de extração de imagem, dando ênfase à subtração de imagem de fundo. As técnicas mais conhecidas são:

. *Chroma key*;
. Subtração de Imagem de Fundo;
. Detecção de Objeto em Imagem Codificada em MPEG;

2.1 Chroma Key

O *Chroma key* [4] é uma técnica bastante conhecida de processamento de imagens cujo objetivo é eliminar o fundo de uma cena, com a finalidade de isolar o objeto de

interesse, que posteriormente poderá ser inserido em outra imagem, resultando em uma terceira imagem que é a combinação das duas primeiras. Está técnica é bastante aplicada e difundida em estúdios de produção de vídeos, onde o fundo e as condições de iluminação (posicionamento e intensidade da luz, reflexo, brilho, etc.) podem ser totalmente controlados.

A técnica tem como finalidade, extrair um objeto ou imagem de primeiro plano de uma cena cujo fundo é uniforme e homogêneo, usualmente uma das cores primárias, vermelho, verde ou azul, sendo que o mais utilizado é o fundo azul. A segmentação do objeto é conseguida pela filtragem dos *pixels*[3] cujas características de cor e brilho coincidem com os parâmetros definidos para a imagem de fundo. O resultado deste processo é a extração do objeto de interesse cujos *pixels* serão inseridos em uma segunda imagem para a obtenção do efeito desejado de superposição.

A principal vantagem do *Chroma key* refere-se à sua facilidade de implementação, já que esta técnica se propõe a lidar com a extração de objetos em imagens com fundo homogêneo e em ambientes controlados. O fato de lidar com uma imagem de fundo homogêneo e conhecido, implica em uma menor necessidade de memória para armazenamento, já que esta técnica não utiliza uma imagem de referência.

Como desvantagens desta técnica podemos citar sua incapacidade de tratar imagens com fundo heterogêneo e sua falta de robustez em relação a variações de iluminação locais e globais, que são requisitos fundamentais para a aplicação proposta neste trabalho.

2.2 Subtração da Imagem de Fundo

A técnica de extração de objetos por subtração de fundo consiste em subtrair a imagem corrente de um modelo da uma imagem de referência. Do processo de subtração de fundo resultam somente objetos novos ou não estacionários. A técnica é muito usada em sistemas de visão como um passo do pré-processamento para detecção e rastreamento de

3. *Pixel* - Menor elemento constituinte da imagem. Termo largamente utilizado na área de processamento de imagens cuja tentativa de tradução poderia desvirtuar e comprometer o entendimento do texto.

objetos. Alguns trabalhos têm sido publicados propondo-se a implementar a segmentação de objeto por subtração de fundo [6, 7, 10, 11], considerando um fundo estático. Os maiores problemas enfrentados são as variações de brilho tanto local quanto global e a implementação em tempo real.

Outra característica a ser observada nos métodos de extração de objeto em fundo fixo, como será apresentado nas secções 2.2.1 e 2.2.2, é que as estratégias usadas pressupõem a câmera em uma posição fixa, para obtenção de melhores resultados na segmentação [12]. Portanto, neste trabalho não trataremos de algoritmos de segmentação de objetos, tais como o *Pfinder* (*Person Finder*) [10], voltado principalmente para localização e busca de pessoas em movimento, além de outras aplicações como interface homem-máquina e sistema de segurança automatizado, que pelo fato de admitir mudança na posição da câmera, exige grande poder de processamento. O *Pfinder* tem excelente performance para a aplicação a que se destina, no entanto apresenta várias deficiências no que refere à extração de objetos propriamente dita.

O esquema básico de um algoritmo para extração de objetos pela subtração de fundo, é apresentado a seguir:

. Construir um modelo ou imagem de referência;

. Determinar a seleção de valores de limiar apropriados para a operação de subtração, obtendo assim a melhor razão de detecção;

. Efetuar a "subtração" *pixel* a *pixel* da imagem corrente com a imagem de referência, para determinação do que é objeto e o que é imagem de fundo;

Um estudo comparativo realizado por Toyama et al. [12], no qual são identificados e avaliados os principais algoritmos utilizados para subtração de fundo, nos mostra uma análise da performance de cada técnica em cenas controladas, visando a geração proposital de determinados erros de detecção.

Em seguida, como pode ser visto na Figura 2-1 [12], são apresentados os resultados de testes de 10 algoritmos de subtração de objeto (alguns com atualização de background) aplicados a sete cenas diferentes, cada uma, propositadamente evidencia um problema típico para algoritmos de extração de objeto. Em cada linha são apresentados os

resultados de um algoritmo e em cada coluna é apresentada uma cena evidenciando o problema a ser avaliado. Na linha de cima é mostrada a imagem da seqüência na qual o processo foi interrompido para avaliação. Na segunda linha é apresentada a "imagem gabarito" do objeto a ser extraído, para comparação qualitativa.

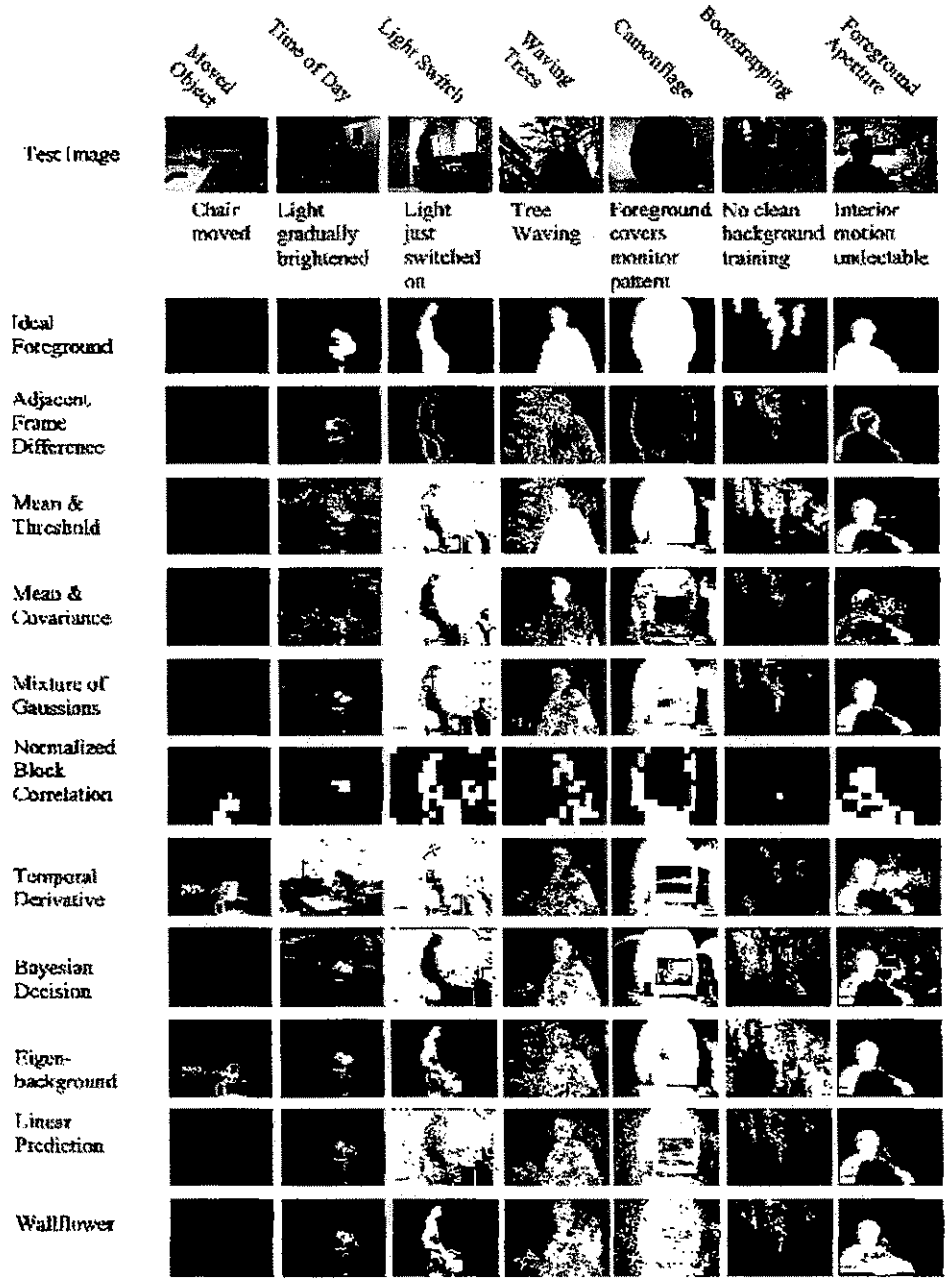


Figura 2-1 – Resultado de testes de 10 algoritmos de subtração de objeto (alguns com atualização de fundo) aplicado a sete cenas diferentes. Cada uma propositadamente evidencia um problema típico para algoritmos de extração de objeto.

Pela análise dos resultados, concluímos que ainda não existe um algoritmo perfeito para a extração de objetos de uma cena, no entanto se estabelecermos determinadas restrições para nossa aplicação, os resultados podem ser bastante satisfatórios. As principais restrições a serem estabelecidas são que a câmera deve ser fixa e o fundo deve ser estático. Porém, a técnica escolhida deve necessariamente poder lidar com variações de luminosidade. Dessa forma, a priori, podemos suprimir a necessidade de atualização ou "manutenção de fundo", o que reduz significativamente a demanda de esforço computacional e atende aos requisitos de nossa aplicação.

As técnicas de segmentação que se propõem a extrair objetos por subtração de fundo, heterogêneo e estático, capturados por câmera fixa, e em tempo real, podem ser classificados segundo os métodos utilizados no modelamento da imagem de referência, como descritos a seguir:

. Métodos estatísticos de subtração de fundo [6, 11, 15]

. Método estático geométrico da imagem de fundo [7]

2.2.1 Métodos Estatísticos de Subtração de Fundo

O processo consiste em construir uma representação da imagem de referência, por um processo estatístico, e efetuar a extração do objeto pelo processo de subtração de fundo. O primeiro passo nesta técnica é adquirir uma seqüência de quadros do vídeo que não contêm o objeto, denominada obtenção da imagem de referência. Tipicamente esta seqüência é relativamente longa, cerca de alguns segundos, para um melhor modelamento da imagem de referência, fornecendo uma boa estimativa de covariância de cor associado com cada *pixel* da imagem.

Uma vez finalizado o modelo da imagem de fundo, novas imagens são capturadas e o sistema passa a detectar desvios do modelo gerado. Os valores de *pixels* são comparados em relação ao fundo, já conhecido, usando medidas de similaridade, com o modelo estatístico das propriedades de textura da imagem de referência, considerando a estatística da geometria, refletância e iluminação da cena, classificando assim cada *pixel* como

pertencente ao objeto ou à imagem de fundo. Este processo, pelo fato de utilizar uma média estatística da variação dos *pixels*, apresenta resultados bastante satisfatórios, principalmente no que se refere à instabilidade gerada por ruído usual da câmera.

A principal vantagem das técnicas de extração pelos métodos estatísticos de subtração de fundo, é a capacidade de processar imagens com fundo heterogêneo.

Como desvantagens podemos apontar a dificuldade de lidar com grandes variações de iluminação e sombras, bem como a necessidade de memória para armazenamento do modelo da imagem de referência.

2.2.2 Método Estático Geométrico da Imagem de Fundo

É um método de segmentação de objetos que utiliza um processo geométrico para o modelamento da imagem de fundo [7]. A implementação desta técnica utiliza duas ou mais câmeras e um mapa de disparidade criado durante a inicialização do sistema. A segmentação é efetuada testando-se os valores de intensidade de cor de cada *pixel* correspondente (imagem chave versus imagem de referência) . Se os valores casarem, a diferença de fundo é validada e o *pixel* dentro da imagem chave é assumido como pertencente ao fundo, caso contrário, é assumido como parte do objeto. Por causa do fundamento desta comparação é que desvios na cena causados por variações de iluminação ou refletância, não afetam significativamente o resultado, como será visto a seguir.

Esta técnica utiliza duas ou mais câmeras para o processo de extração do objeto, uma câmera chave, responsável pela obtenção da imagem principal (da qual será extraído o objeto) e pelo menos uma outra câmera, denominada *câmera de referência,* posicionada na linha de base a uma distância de compromisso determinada matematicamente [13]. A técnica consiste em construir um mapa de disparidade de cada *pixel* para as diversas câmeras utilizadas, durante o processo de inicialização, possibilitando a comparação de cada *pixel* da imagem capturada pela câmera chave com cada *pixel* correspondente nas imagens capturadas pelas câmeras de referência. O mapa de disparidade pode ser construído estabelecendo-se pontos conhecidos na imagem de fundo sem a presença do

objeto, por exemplo utilizando-se marcas físicas ou a *laser* em alguns pontos na cena, sendo os demais pontos determinados por interpolação.

Uma vez montado o mapa de disparidade pode-se iniciar o processo de extração do objeto em tempo real, para isso é feita a avaliação dos *pixels* em cada quadro de entrada, através do processo de varredura, verificando a correspondência entre as imagens das câmeras. Como a comparação dos *pixels* da imagem chave é realizada em relação a uma imagem de referência capturada no mesmo instante (e não em relação a uma imagem de referência adquirida no momento da inicialização), variações de iluminação ou refletância (mesmo instantâneas) não afetam significativamente o resultado. O esquema básico para a implementação do algoritmo para verificação da disparidade de fundo está descrito a seguir para cada *pixel* dentro da imagem chave, considerando-se um par de câmeras.

. Analisar cada *pixel* da imagem chave (câmera chave) e fazer a correspondência com o *pixel* da imagem de referência (câmera de referência);

. Se o *pixel* de referencia tem a mesma cor e luminosidade do *pixel* chave, o *pixel* é assumido como parte do fundo;

. Se o *pixel* de referência tem cor e luminosidade diferente do *pixel* chave, então este é parte do objeto ou está em uma "região de oclusão" (região do fundo que não pode ser vista pela câmera de referência por causa da presença do objeto) e que por esta razão será considerado, erroneamente, como parte do objeto. Para a solução do problema de oclusão é necessária a utilização de mais de duas câmeras.

As principais vantagens das técnicas de extração de objetos de uma cena pelo modelamento estático geométrico da imagem de fundo, é a capacidade de processar imagens com fundo heterogêneo e sua robustez a variações de iluminação e sombras.

Como desvantagens podemos citar a necessidade de utilização de duas ou mais câmeras para o processamento da extração, que implica em custo elevado da implementação, bem como a necessidade de um processo de inicialização "complicado" para ser implementado por um usuário leigo.

2.3 Detecção de Objeto de Imagem Codificada em MPEG

Esta técnica é baseada em uma análise de movimento das imagens codificadas em MPEG (Motion Picture Expert Group) [8, 9]. Este método executa a extração de objetos em movimento de uma seqüência de imagens, previamente codificada em MPEG [14], pela manipulação direta dos dados relativos aos vetores de movimento (MV), sem a necessidade do processo de decodificação. Para extração do objeto, é utilizado um processo conhecido como casamento entre blocos, em que é utilizada a característica intrínseca da imagem para detecção de redundância, no MPEG chamado de Compensação de Movimento.

As principais vantagens desta técnica estão na simplicidade de implementação, e reduzida necessidade de memória de armazenamento. No entanto possui algumas desvantagens que são fundamentais para a aplicação em questão, por apresentar falhas graves de detecção em objetos estacionários e superfícies homogêneas além de necessitar de um codificador de MPEG2 que, no estágio tecnológico atual, ainda inviabiliza a solução em termos de custo.

2.4 Resumo do Capítulo

Como já mencionado no capítulo 1 desta dissertação, a proposta de implementação deste trabalho destina-se a aplicações para o mercado de consumo em sistemas de entretenimento, portanto devemos buscar soluções de baixo custo, capazes de lidar com fundos heterogêneos, e robustas em relação a pequenas variações de iluminação, sem a exigência de ambientes controlados. Outro aspecto importante a ser considerado é que devido à necessidade de baixo custo, o algoritmo deverá ser implementado em *hardware* tendo como objetivo final o projeto de um ASIC. Isto implica na escolha de um algoritmo que embora requeira alta velocidade, tenha baixa demanda de poder de processamento, fundamental para a redução da área de silício, que tem grande influência no custo final do circuito integrado.

Considerando as diversas técnicas analisadas, destacando-se as descritas neste capítulo, verifica-se que a técnica que mais se adequa aos requisitos da aplicação mencionada é a extração de objetos por subtração de fundo. O algoritmo escolhido para a implementação deste projeto foi proposto por Horprasert et al. [6], que adota uma abordagem estatística para a extração do objeto e baseia-se em um modelo computacional que decompõe a imagem nas componentes de brilho e cromaticidade.

As principais vantagens da técnica adotada são: a possibilidade de processamento de imagens com fundo heterogêneo, sua robustez em relação a variações locais e globais de iluminação, possibilidade de implementação em tempo real e demanda de processamento relativamente baixa, o que implica em menor área de silício.

No capítulo seguinte é feita uma descrição mais detalhada do algoritmo escolhido para a implementação deste trabalho, seus princípios fundamentais, suas principais vantagens e problemas a serem resolvidos.

3 A Solução Proposta

Neste capítulo é apresentado o algoritmo no qual é baseada esta dissertação. Trata-se de uma técnica para detecção de objetos em uma cena com fundo estático, robusta o suficiente para implementar a detecção e eliminação de sombras e variações globais de iluminação [6]. Este algoritmo, por sua vez, baseia-se em um modelo computacional que decompõe a imagem nas componentes de brilho e cromaticidade [15].

Como já mencionado anteriormente, um dos principais problemas das técnicas de subtração de fundo tradicionais é a dificuldade em lidar com variações de iluminação, tanto localizadas com globais. A técnica apresentada neste capítulo propõe-se a solucionar este problema com um método confiável e robusto.

3.1 Modelo Computacional

O modelo computacional utilizado baseia-se em uma característica da visão humana conhecida como constância de cor [16]. A visão humana percebe uma cor constante em um objeto, mesmo se submetido a variações de luminosidade no tempo e espaço. A cor percebida em um determinado ponto em uma cena depende de vários fatores tais como as propriedades físicas deste ponto na superfície do objeto. Importantes propriedades físicas de um objeto para a percepção da cor são as propriedades de reflectância espectral de sua superfície, as quais não variam com mudanças na iluminação, composição da cena ou geometria.

Em outras palavras, um objeto cuja superfície tem determinada cor, será percebido pela visão humana como a mesma cor, mesmo se submetido a variações de luminosidade. Esta constatação possibilita a detecção de sombras em um fundo conhecido, desde que se consiga detectar variações de cor associada a variações de luminosidade.

A seguir será mostrado um modelo computacional que se adequa a esta linha de raciocínio. Trata-se de um modelo tridimensional com as componentes de cor R,G e B no qual serão representados todos os pontos relativos a uma determinada cena, neste caso, com seus valores discretizados. Na Figura 3-1 é mostrada uma representação gráfica do modelo proposto com seus três eixos ortogonais R,G e B, e um ponto E_i qualquer.

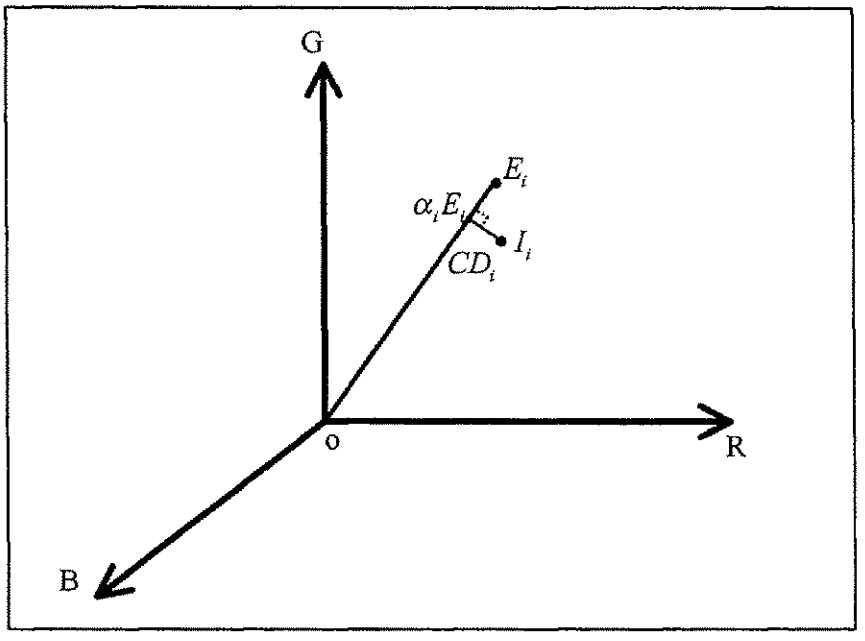


Figura 3-1- Representação gráfica do modelo proposto com seus três eixos ortogonais R,G e B, e um ponto E_i qualquer.

Este modelamento possibilita a representação de cada *pixel* como uma composição de dois termos separados, brilho e cromaticidade como ilustrado pela Figura 3-1. Considere um *pixel* i qualquer na imagem; suponha $E_i = \left[E_{i(R)}, E_{i(G)}, E_{i(B)}\right]$ o valor esperado para o *pixel* naquele ponto da imagem referência. A linha $\overline{OE_i}$ que passa pela origem e pelo ponto E_i é chamada de *linha de cromaticidade esperada.* Também é mostrado o ponto $I_i = \left[I_{i(R)}, I_{i(G)}, I_{i(B)}\right]$ que representa o mesmo *pixel* na imagem corrente, o qual desejamos subtrair ou extrair da imagem referência. Basicamente esta técnica consiste em medir a distorção de I_i em relação a E_i, pela decomposição em duas componentes, *distorção de brilho* e *distorção de cromaticidade.*

3.2 Distorção de Brilho

A distorção de brilho (α) é um valor escalar que representa a diferença de luminosidade entre o *pixel* corrente e o *pixel* equivalente na imagem referência. Seu valor é obtido buscando-se a menor distância entre o ponto I_i e a *linha de cromaticidade esperada*. Portanto, α_i é a projeção de I_i no vetor $\overline{OE_i}$ e pode ser maior ou menor que a unidade dependendo da variação de luminosidade, ou seja, se o *pixel* observado for mais claro ou mais escuro que o *pixel* da imagem referência. Caso o *pixel* observado tenha o mesmo nível de luminosidade que o seu equivalente na imagem referência, α_i terá valor unitário . O cálculo de α_i é realizado minimizando-se a expressão:

$$\phi(\alpha_i) = (I_i - \alpha_i E_i)^2 \qquad \textbf{(3-1)}$$

3.3 Distorção de Cor

A distorção de cor (CD) é definida como a distância ortogonal entre o ponto que representa o *pixel* observado e a *linha de cromaticidade esperada*, e representa a diferença de cor de um *pixel* i entre a imagem correntemente observada e a imagem de referência. A distorção de cor de um determinado *pixel* i é dada por:

$$CD_i = \|I_i - \alpha_i E_i\| \qquad \textbf{(3-2)}$$

3.4 Característica da Imagem Colorida

Para o perfeito entendimento do algoritmo para detecção de um objeto de um fundo estático baseado no modelo computacional descrito neste trabalho, torna-se necessário a compreensão de certas características físicas de imagem coloridas reais capturadas por câmeras CCD's típicas.

O sensor CCD transforma linearmente, um espaço espectral de cor dimensionalmente infinito em um espaço tridimensional, nas componentes RGB das cores. Existem algumas características da imagem gerada por um CCD, as quais precisam ser consideradas na implementação do algoritmo descrito.

3.4.1 Variação de Cor

De fato, ao analisarmos uma imagem estática capturada por um sensor CCD, raramente observamos o mesmo valor para um mesmo *pixel* em determinado período de tempo. Este fato se deve principalmente a ruídos da câmera e flutuação na iluminação causada por variações das fontes de luz. Portanto estas variações devem ser consideradas e mensuradas, este é um dos principais motivos da necessidade de uma abordagem estatística para a extração de um objeto. Dessa forma, não é suficiente a simples aquisição e análise de um único quadro da imagem de referência, torna-se necessário à aquisição e análise de uma amostra estatisticamente significativa de quadros da imagem de referência conhecida, permitindo assim a determinação de parâmetros de normalização relativos às variações do sistema como um todo.

3.4.2 Desbalanceamento de Cores

Câmeras CCD's tipicamente apresentam diferentes sensibilidades para cores diferentes, devido principalmente a diferenças na sensibilidade dos sensores de RGB. Verificam-se respostas diferentes para a mesma variação de intensidade de cor de R, G e B. Dessa forma torna-se necessário fazer o balanceamento ou normalização dos valores capturados de modo a equalizar as respostas. Neste algoritmo a ponderação é realizada tomando-se o desvio padrão (S_i) de cada um dos *pixels* de uma imagem estática de fundo, calculado em um número N de quadros. O desvio padrão (S_i) é dado por:

$$S_i = \left[\sigma_{i(R)}, \sigma_{i(G)}, \sigma_{i(B)}\right] \qquad \textbf{(3-3)}$$

Onde $\sigma_{i(R)}$, $\sigma_{i(G)}$ e $\sigma_{i(B)}$ são os desvios padrão de cada *pixel* (i) tomados em N quadros da imagem de fundo.

De acordo com [6], o comportamento do desvio padrão S_i ainda pode ser expresso em apenas três valores globais, um para cada componente R, G e B sem perda significativa no resultado ou seja $S_i = S = [\sigma_R, \sigma_G, \sigma_B]$, que pode ser expresso por:

$$S = [\sigma_{(C)}] = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{i(C)})^2} \qquad \textbf{(3-4)}$$

$P_{i(C)}$ representa o valor digitalizado de um determinado *pixel* i, sendo C cada uma das componentes de cor R, G e B. Somente para efeito de nomenclatura manteremos a representação deste parâmetro como S_i mas com apenas uma componente para cada cor $S_i = [\sigma_{(R)}, \sigma_{(G)}, \sigma_{(B)}]$.

3.4.3 Grampeamento

Os sensores de câmeras CCD's têm resposta com faixa dinâmica limitada, este fato restringe a gama de cores para o espaço contido dentro do cubo formado pelos eixos ortogonais R, G e B que representam as cores primárias. Em uma imagem colorida representada em 24 bits, teremos toda a gama de distribuição de cores dentro de um cubo de [0,0,0] a [255,255,255]. Cores fora deste espaço (negativas ou maiores que 255) não podem ser representadas baseadas neste modelamento. Como resultado, nesta representação, os valores dos *pixels* localizados fora do cubo são grampeados a valores máximos ou mínimos dentro da faixa possível de representação deste modelo (0 a 255). Isso gera uma distribuição de cor atípica dos *pixels* saturados, que por terem seus valores reais grampeados seja em 0 ou em 255, não têm a variação típica dos demais *pixels*, apresentando variações mínimas. Uma vez que o desvio padrão e a variação de cor são dados fundamentais na análise da imagem proposta nesta técnica, este fato também deve ser considerado na implementação do algoritmo.

3.5 Subtração da Imagem de Fundo

O princípio básico para a subtração do *background* (imagem de fundo) é subtrair a imagem corrente da imagem de referência previamente adquirida, processada e armazenada. Tipicamente os passos básicos do algoritmo são:

. ***Modelamento da Imagem de Fundo***, que é a construção da imagem de referência que representa a imagem de fundo.

. ***Seleção do Nível de Limiar***, que é a seleção do patamar de decisão apropriado para a operação de segmentação visando a obtenção da menor taxa de erro detecção.

. ***Operação de Subtração ou Classificação de Pixel***, que classifica o tipo de um determinado *pixel*. Um *pixel* pode ser parte do fundo ou da imagem do objeto a ser separada.

3.5.1 Modelamento da Imagem de Fundo

Durante o processo de modelamento da imagem de fundo, uma imagem de fundo de referência e alguns parâmetros submetidos a uma normalização, são processados a partir de um determinado número de quadros de uma imagem de fundo estática. A imagem de fundo é modelada estatisticamente com base nas características individuais de cada *pixel*.

O modelo do *pixel* é formado por um conjunto de 4 parâmetros $< E_i, S_i, a_i, b_i >$, sendo E_i o valor esperado, S_i o *desvio padrão* relativo ao valor do *pixel* definido em **(3-4)**, a_i a variação de distorção de brilho e b_i a variação da distorção de cromaticidade do *pixel* i. E_i, a_i e b_i serão definidos a seguir.

O valor esperado do *pixel i* é dado por:

$$E_i = \left[\mu_{i(R)}, \mu_{i(G)}, \mu_{i(B)} \right] \qquad \textbf{(3-5)}$$

Na qual, $\mu_{i(R)}$, $\mu_{i(G)}$ e $\mu_{i(B)}$ são as médias aritmética dos valores de R, G e B do *pixel i* tomados em N quadros. O conjunto de todos os E_i *pixels* formam um quadro de referência, no qual cada *pixel i* representa a média dos valores dos N *pixels* correspondentes a esta posição calculados em N quadros.

Uma vez definidos os valores de S_i e E_i, e tendo sido mencionada a ponderação necessária para a correção do balanceamento de cor aplicando o *desvio padrão* (S_i), podemos partir para uma nova definição para a distorção de brilho (α) e distorção de cromaticidade (CD), assim, baseado em (3-1) temos:

$$F(x_i) = \left(\frac{I_{i(R)} - x_i\mu_{i(R)}}{\sigma_{(R)}}\right)^2 + \left(\frac{I_{i(G)} - x_i\mu_{i(G)}}{\sigma_{(G)}}\right)^2 + \left(\frac{I_{i(B)} - x_i\mu_{i(B)}}{\sigma_{(B)}}\right)^2 \qquad \textbf{(3-6)}$$

Na qual, x_i representa um fator de escala para Ei; Assim:

$$\alpha_i = x_i \left| \frac{\partial}{\partial t} F(x_i) \right. = 0 \qquad \textbf{(3-7)}$$

Sendo α_i o valor de x_i para a menor distância entre I_i e a linha de cromaticidade.

Logo a distorção de brilho será :

$$\alpha_i = \frac{\left(\frac{I_{i(R)}\mu_{i(R)}}{\sigma_{(R)}^2} + \frac{I_{i(G)}\mu_{i(G)}}{\sigma_{(G)}^2} + \frac{I_{i(B)}\mu_{i(B)}}{\sigma_{(B)}^2}\right)}{\left(\left(\frac{\mu_{i(R)}}{\sigma_{(R)}}\right)^2 + \left(\frac{\mu_{i(G)}}{\sigma_{(G)}}\right)^2 + \left(\frac{\mu_{i(B)}}{\sigma_{(B)}}\right)^2\right)} \qquad \textbf{(3-8)}$$

e ainda, baseado em (3-2), a distorção de cromaticidade será dada por:

$$CD_i = \sqrt{\left(\frac{I_{i(R)} - \alpha_i\mu_{i(R)}}{\sigma_{(R)}}\right)^2 + \left(\frac{I_{i(G)} - \alpha_i\mu_{i(G)}}{\sigma_{(G)}}\right)^2 + \left(\frac{I_{i(B)} - \alpha_i\mu_{i(B)}}{\sigma_{(B)}}\right)^2} \qquad \textbf{(3-9)}$$

A seguir, consideraremos as distorções de brilho e cromaticidade do fundo ao longo do tempo e espaço. Sabemos que diferentes *pixel* produzem diferentes distribuições de (α) e (CD). Estas variações são consideradas, e estão presentes no modelamento da imagem de referência como (a_i) e (b_i) que fazem parte do modelo do fundo, e são usados como fatores de normalização. Para efetuarmos a análise dos dados precisamos comparar as variações relativas entre os componentes de brilho e cromaticidade. Portanto faremos os cálculos do valor RMS da distância entre αE_i e E_i correspondente a (a_i) e do valor RMS da distância entre αE_i e I_i correspondente a (b_i). Dessa forma teremos ambos os valores comparáveis como descrito a seguir.

(a_i) representa o valor RMS da variação da distorção de brilho do *pixel i* , e é dada por:

$$a_i = RMS(\alpha_i - 1) = \sqrt{\frac{\sum_{k=0}^{N-1}(\alpha_{i(k)} - 1)^2}{N}} \qquad \textbf{(3-10)}$$

(b_i) representa o valor RMS da variação da distorção de cromaticidade do *pixel i* , e é dada por:

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{k=0}^{N-1}(CD_{i(k)})^2}{N}} \qquad \textbf{(3-11)}$$

3.5.2 Operação de Subtração ou Classificação de *Pixel*

Nesta etapa, é avaliada a diferença entre a imagem de referencia e a imagem corrente. A diferença é decomposta nos componentes de brilho e cromaticidade.

Aplicando o limiar adequado à distorção de brilho (α) e distorção de cromaticidade de cada *pixel i* (CD), obtém-se a máscara M_i que indica o tipo do *pixel*. Este método classifica um dado *pixel* em uma das quatro categorias seguintes:

. ***Fundo Original*** (B) - se o *pixel* tem brilho e cromaticidade similar ao seu equivalente na imagem de referência.

. ***Sombra ou Fundo sombreado*** (S) - se o *pixel* tem cromaticidade similar, mas brilho menor que seu equivalente na imagem de referência. Esta definição baseia-se no conceito de que uma sombra é uma região semi-transparente da imagem, que portanto mantém as principais características da superfície da imagem tais como: padrão, textura ou valor da cor [17]. A região de sombra pode ser causada pelo próprio objeto, por uma variação na posição das fonte de luz, pela redução global da iluminação etc.

. ***Fundo Iluminado*** (H) - se o *pixel* tem cromaticidade similar, mas brilho mais elevado que seu equivalente na imagem de referência (background). Baseia-se no mesmo princípio descrito no item anterior e pode ser causado por uma reflexão de luz do próprio objeto, por uma variação na posição das fontes de luz, pela elevação global da iluminação etc.

. ***Objeto*** (F) - se o *pixel* tem cromaticidade diferente do valor esperado relativo ao seu equivalente na imagem de referência.

Como já mencionado anteriormente diferentes *pixel* produzem diferentes distribuições de (α_i) e (CD_i), portanto, de forma a utilizar um mesmo limiar para todos os *pixels*, precisamos normalizar (α_i) e (CD_i). Para isso usamos os fatores de normalização (a_i) e (b_i).

Então façamos a distorção de brilho normalizada:

$$\hat{\alpha}_i = \frac{\alpha_i - 1}{a_i} \qquad \textbf{(3-12)}$$

e ainda a distorção de cromaticidade normalizada;

$$\hat{CD}_i = \frac{CD_i}{b_i} \qquad \textbf{(3-13)}$$

Com base nestas definições, um *pixel* seria classificado em uma das quatro categorias B, S, H ou F pelo seguinte processo de decisão:

$$M_i = \begin{cases} F : \hat{CD}_i > \tau_{CD}, \text{ se não} \\ B : \hat{\alpha}_i < \tau_{\alpha 1} \text{ E } \hat{\alpha} i > \tau_{\alpha 2}, \text{ se não} \\ S : \hat{\alpha}_i < 0, \text{ se não} \\ H : \text{ outros casos} \end{cases} \quad \textbf{(3-14)}$$

Na qual, τ_{CD}, $\tau_{\alpha 1}$ e $\tau_{\alpha 2}$ são limiares usados para determinar a similaridade de cromaticidade e brilho entre a imagem de background e a imagem corrente. Na próxima seção discutiremos o método para seleção dos limiares adequados.

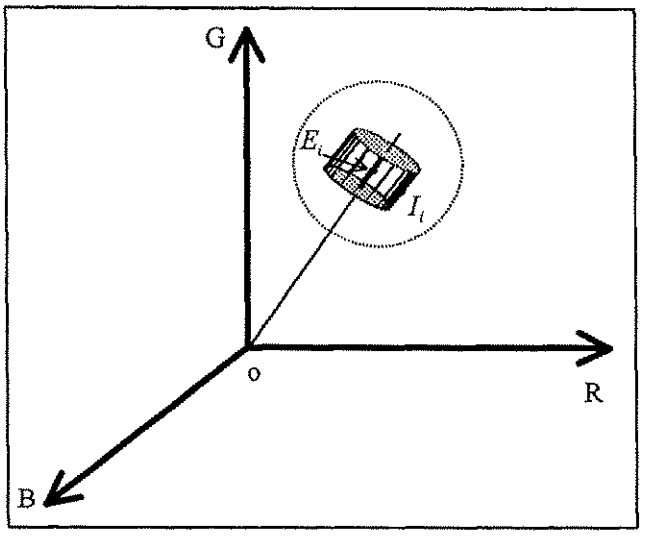
Contudo existem casos em que um *pixel* de um objeto na imagem corrente contém baixos valores de RGB. Este *pixel* escuro será sempre classificado, erroneamente, como uma sombra. Devido ao ponto localizar-se próximo à origem no espaço cúbico de RGB e ao fato de que todas as linhas de cromaticidade no espaço de RGB têm um ponto comum na origem, portanto este ponto de cor seria considerado muito próximo ou similar a qualquer linha de cromaticidade. Para evitar este problema foi introduzido um limite inferior para a distorção de brilho normalizada ($\tau_{\alpha lo}$). Então o processo de decisão dado por (3-15) passa a ser descrito como segue:

$$M_i = \begin{cases} F : \hat{CD}_i > \tau_{CD} \text{ OU } \hat{\alpha}_i < \tau_{\alpha lo}, \text{ se não} \\ B : \hat{\alpha}_i < \tau_{\alpha 1} \text{ E } \hat{\alpha} i > \tau_{\alpha 2}, \text{ se não} \\ S : \hat{\alpha}_i < 0, \text{ se não} \\ H : \text{ outros casos} \end{cases} \quad \textbf{(3-15)}$$

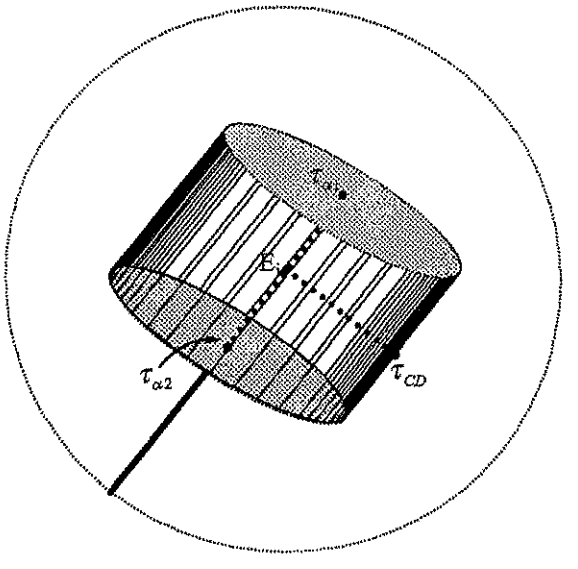
3.5.3 Seleção Automática dos Limiares

Para o caso de uma distribuição Gaussiana da distorção, o nível desejado de detecção, τ seria obtido por $K\sigma$, sendo K a constante determinada por τ, e σ o desvio padrão da distribuição. No entanto, $\hat{\alpha}_i$ e $\hat{CD}_i$ não apresentam distribuição Gaussiana [6]. Portanto, para a obtenção do nível de detecção devem ser avaliados os histogramas da distorção de brilho normalizada $\hat{\alpha}_i$ e distorção de cromaticidade normalizada $\hat{CD}_i$, baseados em uma seqüência de cenas capturadas durante o processo de aquisição da imagem de referência. O número total de amostras para os histogramas, para uma imagem X por Y *pixels* com N quadros, será de $N \cdot X \cdot Y$.

Uma vez montados os histogramas, os limiares de detecção podem ser facilmente determinados. Um limiar τ_{CD} para a distorção de cromaticidade $\hat{CD}_i$, que define o nível de decisão para a similaridade de cor de um *pixel*, e dois limiares ($\tau_{\alpha 1}$ e $\tau_{\alpha 2}$) para a faixa da distorção de brilho $\hat{\alpha}_i$. $\tau_{\alpha 2}$ representa o limiar inferior de decisão para um *pixel* com brilho menor e $\tau_{\alpha 1}$ representa o limiar superior de decisão para um *pixel* com brilho maior que o *pixel* correspondente na imagem de referência. Os três limiares $\tau_{\alpha 1}$, $\tau_{\alpha 2}$ e τ_{CD} delimitam um sólido de forma cilíndrica em torno do ponto E_i, cujo eixo é a linha de cromaticidade esperada, como mostrado na Figura 3-2. Um determinado *pixel* i, contido no interior do "cilindro de referência" será considerado como sendo pertencente à imagem de fundo.



(a)



(b)

Figura 3-2 - (a) – Espaço das componentes de cor RGB com o Cilindro de Referência; (b) - Cilindro de Referência delimitado pelos limiares distorção de cromaticidade e brilho. Todos os pixels contidos no interior do cilindro serão classificados com fundo.

3.5.4 Redução de Erros de Detecção

A introdução de um limite inferior para a distorção de brilho normalizada (τ_{alo}) como descrito na seção anterior, minimiza a possibilidade de detecção errônea de regiões ou pontos escuros do objeto, como sombra. No entanto ainda existe outra situação que pode gerar erros de detecção, trata-se de pontos ou regiões que possuem baixa variação de distorção de cromaticidade, ou seja, baixo b_i. Quando estes baixos valores de b_i são usados para a normalização (3-13), produzem altos valores de $\hat{CD}_i$, que provavelmente excederão os valores dos limiares gerando falsa detecção de *pixel* de objeto na cena.

Para evitar este problema propõe-se a adoção de um valor mínimo para b_i chamado de $b_i(\min)$. O processo para a obtenção deste valor consiste em estabelecer um valor arbitrário para $b_i(\min)$, substituir todos os valores de b_i menores que $b_i(\min)$. Feito isto podemos efetuar o processo de extração de objeto ainda na imagem de referência. Um *pixel* na imagem de referência detectado como pertencente a um objeto, constitui por definição um erro de detecção. A seguir comparamos a taxa de erro nos *pixels* com b_i maior que $b_i(\min)$ com a taxa de erro nos *pixels* assinalados com o valor $b_i(\min)$. O ponto em que obtivermos valores semelhantes das duas taxas, será o ponto ótimo para $b_i(\min)$.

3.6 Resultados Obtidos em Simulação

O algoritmo proposto por [6], descrito neste capítulo e adotado para a solução do problema, foi implementado no MATLAB para validação da idéia levando em conta as restrições relativas a uma futura implementação em *hardware*. Os resultados obtidos com imagens aleatórias foram satisfatórios e atendem às expectativas.

A seguir são apresentados dois resultados obtidos através de simulações da extração de objeto de seqüências de imagens com 64 quadros. Ambas as imagens foram submetidas ao mesmo processo de extração de objeto descrito neste capítulo e implementado no

MATLAB, do qual são extraídas figuras apresentadas em seguida que representam algumas etapas do processo, bem como o resultado final.

A primeira imagem representa um caso com baixa complexidade, ou seja uma imagem de fácil tratamento pelo algoritmo proposto. A segunda imagem representa um caso mais complexo, no qual o fundo e o objeto propositadamente buscaram reforçar os principais problemas descritos na literatura, relativos ao algoritmo proposto. A Figura 3-3 de (a) a (f) bem como a Figura 3-6 de (a) a (f) mostram os seguintes resultados.

a) - a média dos 64 quadros adquiridos do fundo (parâmetro E_i), um dos parâmetros resultantes do processo de aquisição da imagem de fundo de referência;

b) - imagem contendo o objeto a ser extraído, introduzido após o processo de aquisição do fundo;

c) - mapa de bits dos *pixels* cuja distorção de cromaticidade é superior ao nível de limiar, estes *pixels* supostamente mudaram de cor e portanto fazem parte do objeto;

d) - mapa de bits dos *pixel* cuja distorção de luminância é maior que o limiar superior;

e) - mapa de bits dos *pixel* cuja distorção de luminância é menor que o limiar inferior;

f) - mapa de bits do objeto, descriminando todos os *pixels* da imagem capturada que "supostamente" fazem parte do objeto;

- Imagem de baixa complexidade

A Figura 3-3 apresenta uma imagem de baixa complexidade para extração de objeto pelo algoritmo implementado. As principais características desta imagem relevantes ao algoritmo implementado, são:

- Fundo liso e monocromático;
- Não há objetos escuros ou pretos no fundo;
- O objeto Introduzido será bastante diferente da cor de fundo;
- Praticamente não há sombreamento no fundo;

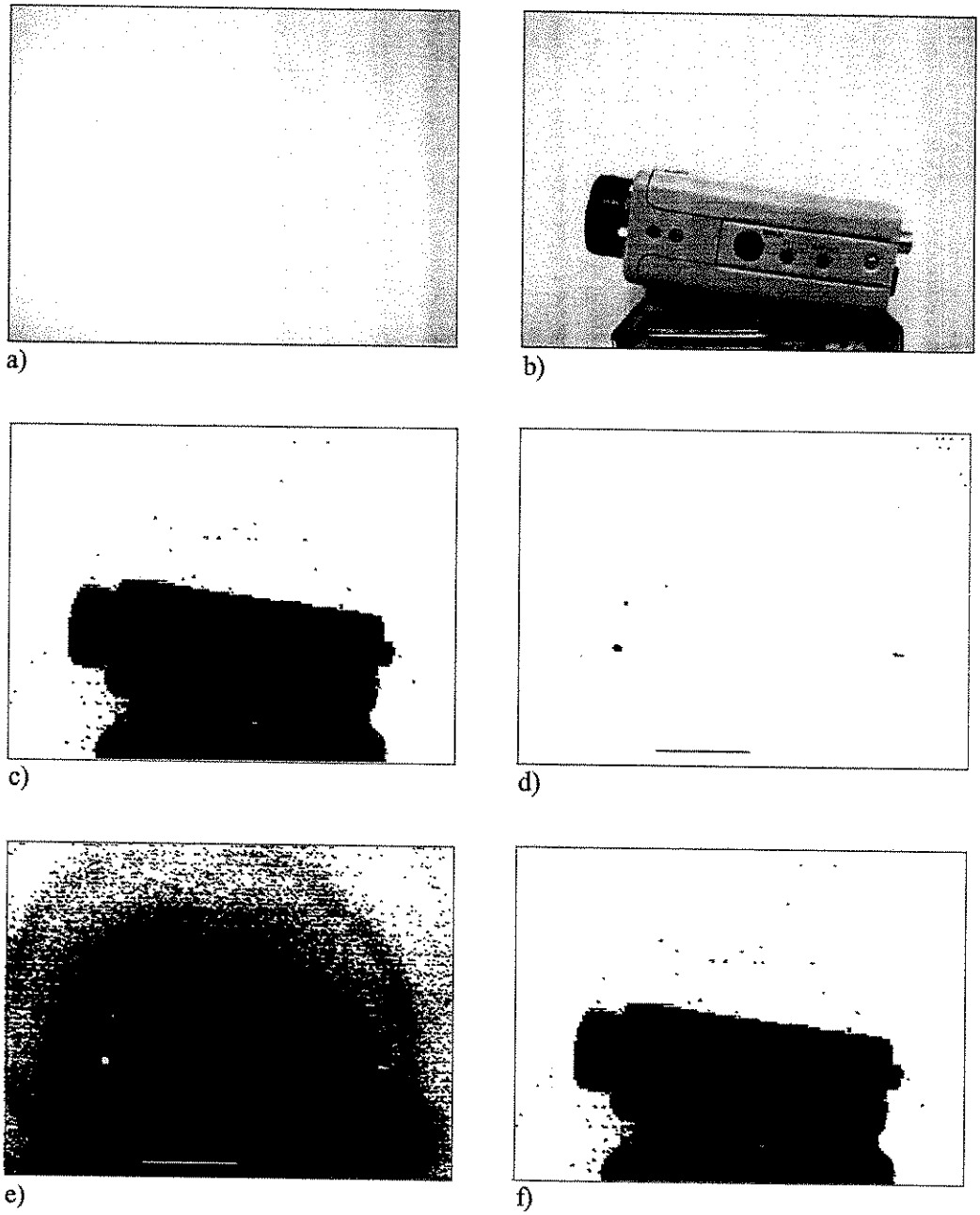


Figura 3-3 – Resultados de simulação em uma imagem de baixa complexidade

Na Figura 3-4 é mostrada a imagem resultante do processo de extração, baseada no mapa de bits do objeto.

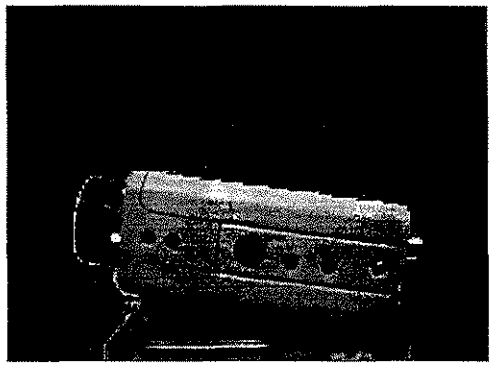


Figura 3-4 – Imagem resultante da extração de objeto de um fundo de baixa complexidade

- **Imagem com fundo complexo**

A seguir são apresentados resultados de simulações que buscaram reforçar os principais problemas descritos na literatura relativos a este algoritmo. Note a escolha de características da imagem de fundo e objeto visando enfatizar os casos mais complicados em relação à implementação proposta. Tais características são descritas a seguir e apresentadas pela Figura 3-5.

a - Fundo texturizado ao invés de liso e monocromático;

b - Introdução proposital de objetos de fundo, escuros ou pretos;

c - Introdução de objeto de fundo, com cor e textura "semelhante" ao objeto;

d - Introdução de objeto de fundo, com variação de sombra;

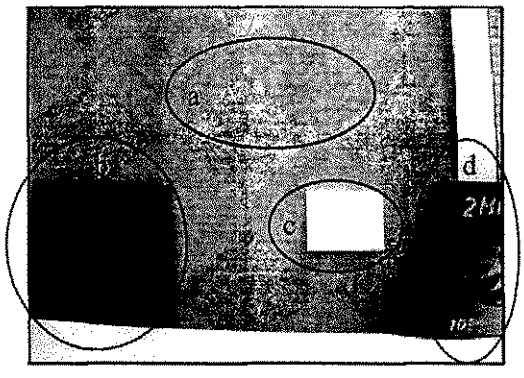


Figura 3-5 – Imagem de fundo enfatizando alguns problemas para a extração do objeto

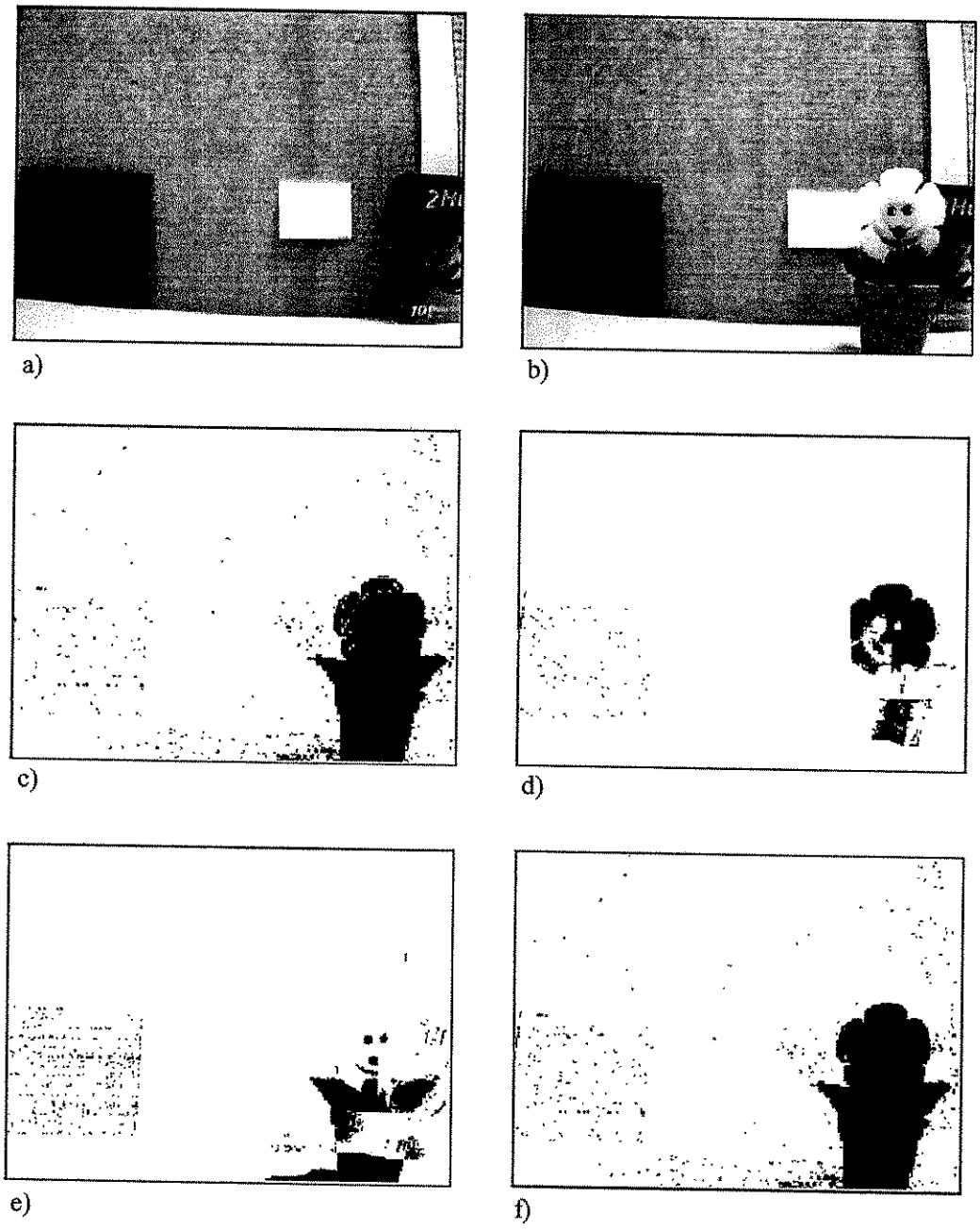


Figura 3-6 – Resultados de simulação em uma imagem com fundo complexo

Na Figura 3-7 é mostrada a imagem resultante do processo de extração, baseada no mapa de bits do objeto;

Figura 3-7 – Imagem resultante da extração de objeto de um fundo complexo

Analisando os resultados das duas figuras verificamos que mesmo no caso de uma imagem bastante simples ainda podem ser encontrados erros de detecção. No entanto estes erros são previstos estatisticamente quando da determinação da taxa de erro esperada e podem ser minimizados adotando-se as técnicas descritas no tópico 3.5.4 e ainda pela utilização de filtros espaciais. No caso da seqüência de imagens "preparada" para enfatizar as deficiências do processo, podemos verificar que, conforme esperado, o algoritmo implementado apresenta um número bem maior de falhas de detecção, que podem ser vistos na Figura 3-7 como pontos do fundo detectados como objeto e pontos do objeto suprimidos. No entanto, resultado final é satisfatório para a aplicação proposta, direcionada ao mercado de entretenimento.

3.7 Resumo do Método Apresentado

Verificamos que o método apresentado neste capítulo, proposto por Horprasert et al. [6], tem como principal característica, a abordagem estatística para a solução do problema de extração de objetos de imagens dinâmicas em um fundo estático não homogêneo. Esta técnica baseia-se em um modelo computacional que decompõe a imagem nas suas componentes de brilho e cromaticidade.

As principais vantagens da técnica adotada são: a possibilidade de processamento de imagens com fundo heterogêneo, sua robustez em relação a pequenas variações locais e globais de iluminação, possibilidade de implementação em tempo real e demanda de processamento relativamente baixa, o que implica em menor área de silício.

Para implementação do algoritmo devemos ter em mente a nossa aplicação, que exige execução em tempo real e voltada para aplicações em produtos de consumo, que implica em uma solução de baixo custo. No entanto a implementação física requer o desenvolvimento de uma plataforma de *hardware* aplicada a processamento digital de imagens capaz de conter todo o sistema e prover as facilidades de interfaceamento humano, indispensáveis para a concretização deste trabalho.

No capítulo seguinte é apresentada a plataforma GPIP-01 (General Purpose Image Processor) desenvolvida para comportar a implementação referente a este trabalho, mas que pode ser utilizada em aplicações gerais voltadas a processamento de imagens em tempo real.

4 Implementação da Plataforma de *Hardware*

O sistema descrito neste capítulo, chamado GPIP-01 (General Purpose Image Processor), foi concebido com o objetivo de atender aos requisitos mínimos de um sistema embarcado de prototipagem para aplicações em processamento de imagens em tempo real, necessário para o desenvolvimento da aplicação proposta neste trabalho.

Esta plataforma é composta por 4 estágios implementados em *hardware* fixo e as respectivas interfaces de controle desenvolvidas em *hardware* reconfigurável, enumerados abaixo e descritos a seguir:

. Estágio de Aquisição de Imagens

. Estágio de Armazenamento

. Estágio de Processamento

. Estágio de Apresentação

. Interfaces de Controle Reconfiguráveis

4.1 Descrição do *Hardware* Fixo

O *hardware* fixo foi desenvolvido a partir de circuitos integrados específicos existentes no mercado cujas especificações atendessem aos requisitos mínimos necessários para o funcionamento do sistema.

4.1.1 Estágio de Aquisição de Imagens

O estágio de aquisição corresponde a uma placa de aquisição de imagens que recebe sinais de vídeo composto no padrão M [19] e efetua a decodificação para RGB e posterior digitalização do sinal. Na configuração adotada o sinal de entrada é convertido para o formato RGB 24 bits com um *clock* de *pixel* de 13.5 MHz.

A placa de aquisição foi desenvolvida com base no circuito integrado SAA7111 fabricado pela Philips Semiconductors. A documentação técnica contendo esquema elétrico, leiaute, especificações detalhadas, configuração I^2C bem como a descrição de funcionamento, encontram-se anexos a este documento [anexo 01].

4.1.2 Estágio de Armazenamento

O estágio de armazenamento corresponde a uma placa contendo um banco de memória de 264Kx24bits com *clock* máximo de 66 MHz, composta por 4 chips de memória UT6164C32 que são bancos de ram estática (SRAM) de 64Kx32 fabricado pela UTRON. A documentação técnica contendo esquema elétrico, leiaute, especificações, cartas de tempo para cada modo de operação, bem como descrição de funcionamento, encontram-se anexos a este documento [anexo 02].

4.1.3 Estágio de Processamento

O estágio de processamento em tempo real corresponde a uma placa com um dispositivo de *hardware* reconfigurável capaz de suportar um sistema de médio porte, e foi projetada para uma FPGA da família APEX (EP20K200EQC240-2X) fabricada pela Altera Corporation, com as seguintes características básicas: 200K portas equivalentes, 240 pinos, PLL interno e ESB (Embedded System Block) que possibilita a implementação de cache interno. A placa possui um sistema mínimo de entrada/saída composto por chaves e led's e pode ser configurada via interface JTAG [18] ou via interface serial do PC, diretamente na FPGA ou nos dispositivos de configuração (EPC2LC20) do mesmo fabricante.

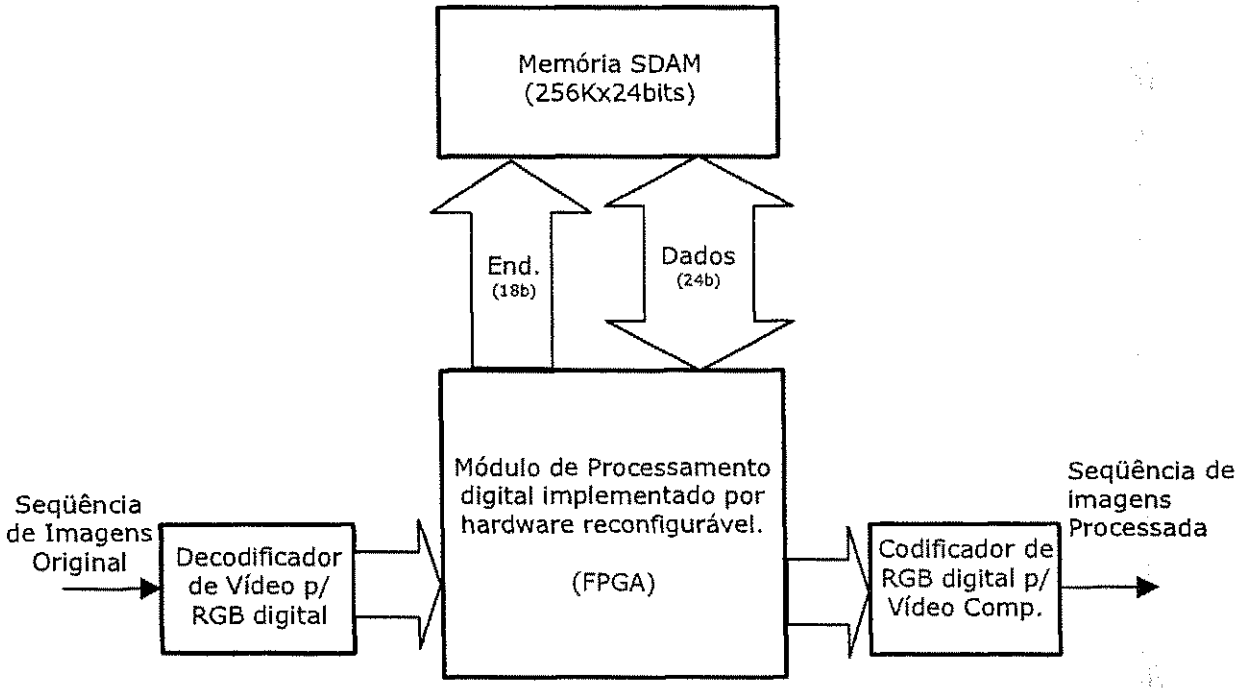
Todo o sistema de controle e interfaceamento com o *hardware* externo foi desenvolvido em Verilog e faz parte do conjunto de ferramentas da plataforma. A documentação técnica contendo esquema elétrico, leiaute, especificações, bem como descrição de funcionamento, encontram-se anexos a este documento [anexo 03].

4.1.4 Estágio de Apresentação

O estágio de apresentação corresponde a uma placa codificadora de vídeo que converte o sinal de entrada no formato RGB 24 bits para um sinal de vídeo analógico em RGB na saída, que possibilita a apresentação da imagem resultante do processamento.

A placa codificadora de vídeo foi projetada tendo como cerne o chip THS8134B fabricado pela Texas Instruments, um triplo conversor D/A de 8 bits para aplicações de vídeo. A documentação técnica contendo esquema elétrico, leiaute, especificações detalhadas, bem como a descrição de funcionamento, encontram-se anexos a este documento [anexo 04].

4.1.5 Diagrama em Blocos da Plataforma Implementada



4.2 Características Gerais

Uma vez descrita a plataforma implementada para o desenvolvimento do sistema proposto, passemos para a descrição das características específicas deste sistema de extração de objeto em tempo real.

4.2.1 Definição da Resolução da Imagem

Um dos aspectos de maior relevância para o desenvolvimento deste sistema é a definição da resolução de imagem a ser adotada. Embora a especificação da placa decodificadora de vídeo defina uma resolução de 720x480, a determinação da resolução do sistema deve considerar a capacidade máxima de processamento dos diversos componentes do sistema.

Uma resolução de 720 *pixels* por linha implica em um tempo de 75 ns para o processamento independente de cada *pixel*, conforme mostrado a seguir:

Tempo da linha horizontal visível - $T_V \cong 54\ \mu s$

Número de *pixels* visíveis em uma linha (resolução) - $R = 720$ pixels

Tempo de processamento para cada *pixel* - $T_p = \frac{T_V}{R} = \frac{54\ \mu s}{720} \Rightarrow T_p = 75\ ns$

A análise das máquinas de estados elaboradas neste trabalho referentes aos processos a serem realizados em tempo real para a extração de objeto, nos levam a uma estimativa aproximada de 35 ciclos de máquina para o processamento de cada *pixel*. Aplicando esta estimativa a uma resolução de 720 *pixel* por linha (75 *ns* para cada *pixel*) resulta em um ciclo de máquina de cerca de $\frac{75\ ns}{35} \cong 2{,}14\ ns$, ou seja o *clock* do sistema seria de $467\ MHz$. Este valor não é praticável para a plataforma na qual será implementado o sistema experimental. Devemos encontrar um valor de compromisso entre a viabilidade do sistema em termos de velocidade de processamento e uma resolução aceitável para apresentação da imagem processada.

Tendo em mente este compromisso adotamos a freqüência de *clock* de 50 MHz (um ciclo de máquina de 20 ns), que é um valor aceitável para a plataforma GPIP-01. Baseados neste valor e na estimativa de 35 ciclos para o processamento de cada *pixel*, devemos ter um tempo mínimo de $(35 \times 20n) = 700\ ns$ para o processamento de um *pixel*. Este tempo nos permitiria processar $\frac{54\ \mu s}{700\ ns} \cong 77\ pixels$ por linha, que é uma resolução inaceitável. No entanto por esta análise estamos sacrificando a resolução de *pixels* (resolução horizontal) sem a contrapartida na resolução de linhas (resolução vertical). Um compromisso entre resolução horizontal e vertical pode ser obtido se processarmos 240 *pixels* por linha em 160 linhas por quadro. Dessa forma, processando 240 *pixels* de uma linha no tempo correspondente a 3 linhas, teremos aproximadamente 794 ns para o processamento de cada *pixel*, conforme mostrado a seguir:

Tempo de total de uma linha - $T_h \cong 63{,}6\ \mu s$

Tempo de 3 linhas - $T_{3L} \cong 190{,}68\ \mu s$

Tempo para *pixel* - $T_P \cong \frac{T_{3L}}{N_p} = \frac{190{,}68\ \mu s}{240} \Rightarrow T_P \cong 794{,}5\ ns$

A adoção da resolução de 240x160 soluciona o problema de velocidade de processamento na plataforma adotada, mas implica em uma mudança na forma de armazenamento dos *pixels* de entrada. A impossibilidade de processamento de cada *pixel* à medida que é capturado, gera a necessidade da implementação de um buffer para o armazenamento de uma linha inteira, que deve ser implementado no sistema.

4.2.2 Especificações Gerais do Sistema

Para o propósito deste trabalho podemos estabelecer que a aquisição deverá atender, de forma satisfatória, às especificações definidas pelo padrão M de transmissão de TV [19], adotado no Brasil. Considerando os valores "efetivos" de resolução horizontal e vertical, o caráter experimental deste trabalho visando demonstrar a viabilidade de implementação do algoritmo em tempo real, e as limitações de velocidade do sistema,

adotamos uma resolução de imagem de 240x160, com uma taxa de 30 quadros por segundo (60 campos entrelaçados) e resolução de 8 bits para cada componente de cor R,G e B, que se adequa ao modelo matemático do algoritmo proposto, e facilita a implementação com o *hardware* do decodificador de vídeo disponível.

A seguir, algumas especificações mais relevantes do sistema:

. Resolução da imagem do objeto: 240x160 *pixels*
. *Clock* do sistema: 50 MHz
. *Clock* da memória: 50 MHz (20 ns)
. Tempo de acesso para leitura (read): 4 ciclos (4 x 20 ns = 80 ns)
. Tempo de acesso para leitura em burst: 2 ciclos (2 x 20 ns = 40 ns)
. Tempo de acesso para escrita (write): 3 ciclos (3 x 20 ns = 60 ns)
. Tempo de acesso para escrita em burst: 2 ciclos (2 x 20 ns = 40 ns)
. Tamanho da memória externa (SRAM): 262.144 (256 KB x 24 bits)

4.2.3 Decodificador de Vídeo

. Freqüência de amostragem: 13.5 MHz
. Resolução máxima de *Pixel*: 720x480
. Número de linhas por campo: 240
. Formato do Sinal: RGB (24 bits)

4.2.4 Codificador de Vídeo

. Freqüência de *Pixel*: 4.5 MHz
. Resolução de *Pixel*: 240x160
. Formato do Sinal: RGB (24 bits)

4.3 Interfaceamento Externo

O desenvolvimento de um sistema de propósito geral para processamento de imagens requer uma série de periféricos para interfaceamento externo os quais foram implementados em Verilog e estão embarcados na plataforma GPIP-01 conforme mostrado pela Figura 4-1 . Os periféricos são enumerados abaixo e descritos no decorrer deste tópico.

. Interface com o decodificador de vídeo;

. Interface com o codificador de vídeo;

. Interface de acesso à memória externa;

. Interface I^2C;

. Controlador de cache;

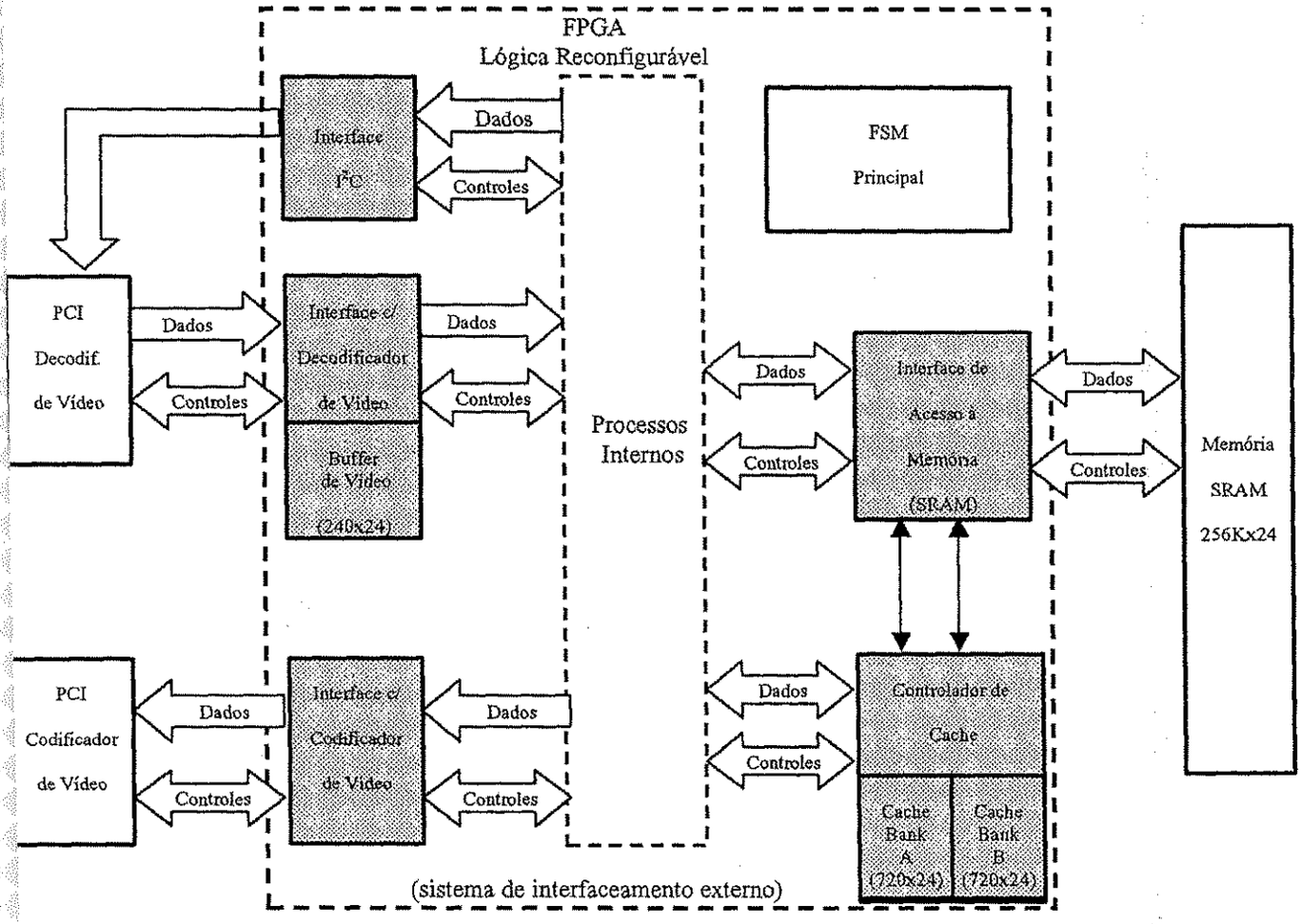


Figura 4-1 – Diagrama geral da plataforma GPIP-01, destacando os módulos reconfiguráveis de interfaceamento com o *hardware* externo.

4.3.1 Interface com o Decodificador de Vídeo (IDV)

A interface com o decodificador de vídeo, é o bloco responsável pela aquisição da imagem digitalizada pelo decodificador de vídeo, selecionando e armazenando todos os *pixels* válidos de uma linha válida em um buffer do tipo FIFO, até a chegada de uma nova linha válida que se sobrepõe à anterior no buffer. A cada solicitação de leitura do buffer (read_*pixel*), o módulo IDV disponibiliza um novo *pixel* na ordem da seqüência de entrada. Enquanto efetua uma escrita no buffer, que leva apenas um ciclo de máquina (20 ns) dentro dos 212,22 ns disponíveis para o processamento de cada *pixel*, o módulo IDV sinaliza indisponibilidade (buff_busy). Portanto, nos intervalos em que não esteja sendo executada a leitura dos dados da placa decodificadora de vídeo, poderiam ser solicitadas e executadas várias operações de leitura de *pixels* do buffer de entrada.

Um aspecto importante a ser mencionado é que a operação interna de escrita dos *pixels* de entrada no buffer é prioritária em relação à solicitação de leitura. A interface IDV é composta por dois blocos básicos: Controlador do Decodificador de Vídeo e Buffer de Vídeo cujas funções são descritas separadamente a seguir.

4.3.1.1 Controlador do Decodificador de Vídeo

O Controlador do Decodificador de Vídeo efetua a aquisição dos *pixel* no formato RGB provenientes da placa decodificadora de vídeo. A seleção e captura de *pixel* válidos em linhas válidas deve ser processada como descrito a seguir.

. Aquisição de *Pixels*

Devido à diferença entre a taxa de *pixel* entregue pelo decodificador (720 *pixels*/linha) e a taxa de *pixel* especificada para o sistema (240 *pixels*/linha), a interface IDV deve capturar somente um *pixel* a cada três recebidos. A partir do primeiro *pixel* de cada linha, iniciando um processo cíclico desprezando os dois *pixels* subseqüentes e capturando o próximo, ou seja, captura um *pixel* e despreza os dois seguintes, conforme mostrado pela Figura 4-2a. Tempo entre aquisições de *Pixels*: $3 \times \left({}^{1}/_{13.5\ MHz} \right) = 222.22\ ns$.

. **Aquisição de Linhas**

Para a obtenção da resolução vertical de 160 linhas, a interface de entrada deve capturar somente uma linha a cada três entregues pelo decodificador em cada um dos campos.

No campo ímpar a interface deve detectar e capturar a primeira linha (linha 1), e iniciar um processo cíclico desprezando as duas linhas subseqüentes (linhas 3 e 5) e capturando a próxima (linha 7) e assim por diante, ou seja captura uma linha e despreza as duas seguintes conforme mostrado pela Figura 4-2.

No campo par a interface deve identificar a primeira linha (linha 2) que deve ser desprezada, capturar a segunda linha (linha 4) e iniciar um processo cíclico desprezando as duas linhas subseqüentes (linhas 6 e 8) e capturando a próxima (linha 10) e assim por diante, ou seja captura uma linha e despreza as duas seguintes.

. Tempo entre aquisições de linhas: $3 \times 63.56\mu s = 190,67\mu s$

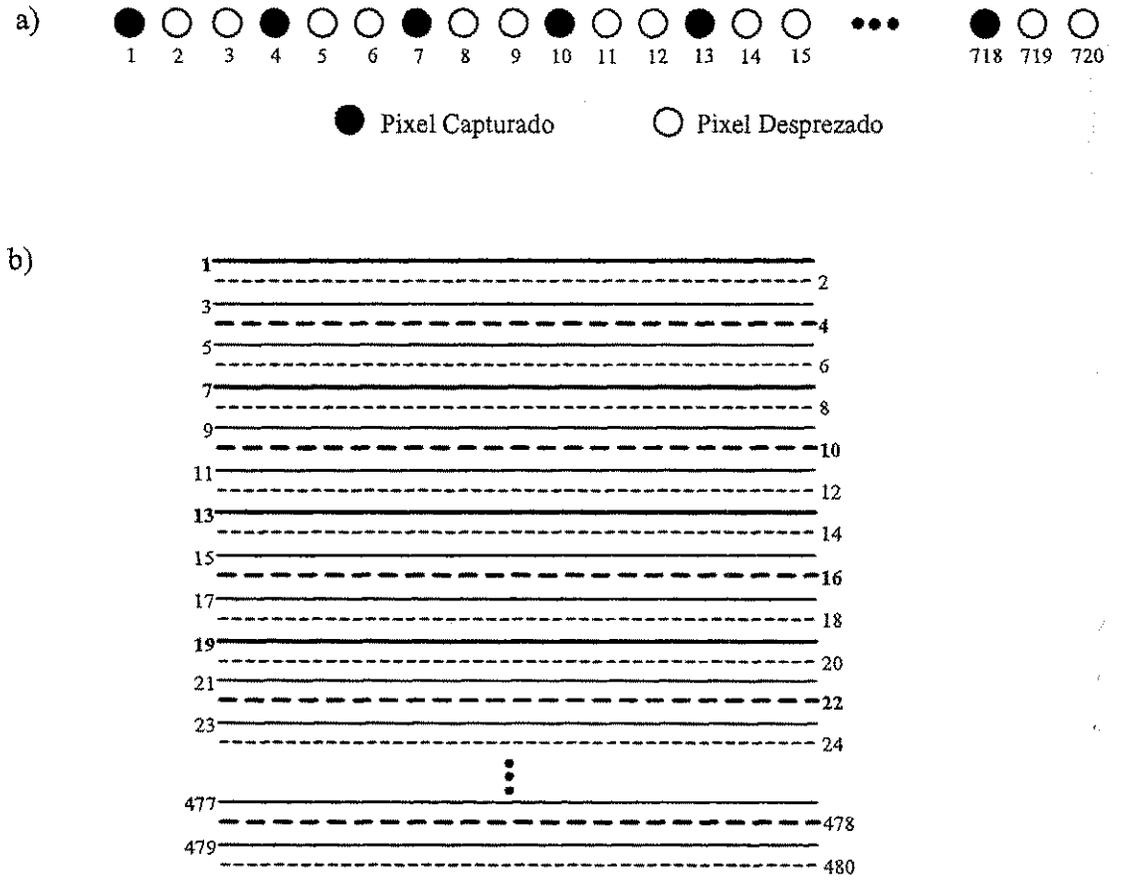


Figura 4-2 – (a) Processo para seleção de *pixels* válidos; (b) Processo para seleção de linhas válidas

4.3.1.2 Buffer de Vídeo

Durante o processo de captura, a interface de entrada IDV deve alimentar um buffer do tipo FIFO com capacidade para armazenamento de uma linha completa, ou seja, 240x8bitsx3*pixels* (240 posições de 24 bits).

A escrita no buffer ocorre em apenas um ciclo de máquina (20 ns) ficando o restante do tempo entre capturas de *pixels* (202,22 ns) disponível para leitura por outros dispositivos internos componentes do sistema.

4.3.1.3 Diagrama em Blocos

O diagrama em blocos simplificado da interface IDV, implementada em *hardware* reconfigurável, bem como suas conexões com a placa do decodificador de vídeo e conexões internas é apresentado pela Figura 4-3 .

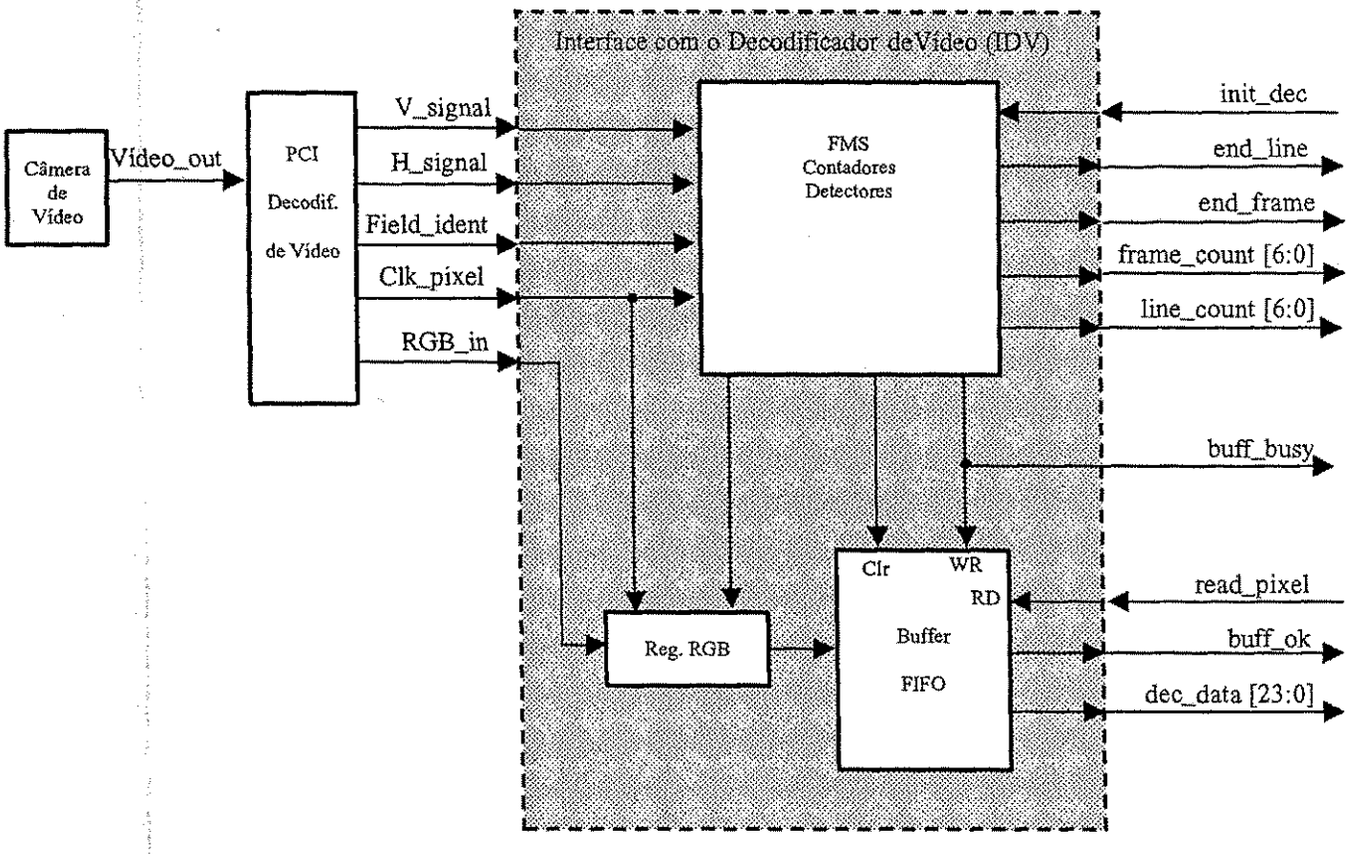


Figura 4-3 – Diagrama em blocos simplificado da interface com o decodificador de vídeo (IDV)

4.3.2 Interface de Acesso à Memória (IAM)

A interface de acesso à memória (IAM) realiza todas as operações de leitura e escrita de cada palavra de 24 bits diretamente na SRAM.

Durante a operação de escrita, a interface IAM recebe o endereço (18 bits) e o dado de 24 bits a ser escrito na memória, efetua a decodificação para a seleção do banco executa a operação e sinaliza a conclusão. Este é o único módulo que tem acesso direto ao barramento externo da memória SRAM.

. Tempo de acesso para escrita (write single): 3 ciclos (3 x 20ns = 60 ns)

. Tempo de acesso para escrita seqüencial (write burst): 2 ciclos (2 x 20 ns = 40 ns)

Durante a operação de leitura, a interface IAM recebe o endereço (18 bits), efetua a decodificação para a seleção do banco, executa a operação, disponibiliza o dado (24 bits) na saída e sinaliza a conclusão.

. Tempo de acesso para leitura (read single): 4 ciclos (4 x 20 ns = 80 ns)

. Tempo de acesso para leitura seqüencial (read burst): 2 ciclos (2 x 20 ns = 40 ns)

O procedimento para acesso à memória deve ser implementado segundo as cartas de tempo disponíveis no [anexo 02].

4.3.2.1 Diagrama em Blocos

O diagrama em blocos simplificado da interface IAM, implementada em *hardware* reconfigurável, bem como suas conexões com a placa de memória SRAM e conexões internas é apresentado pela Figura 4-4 .

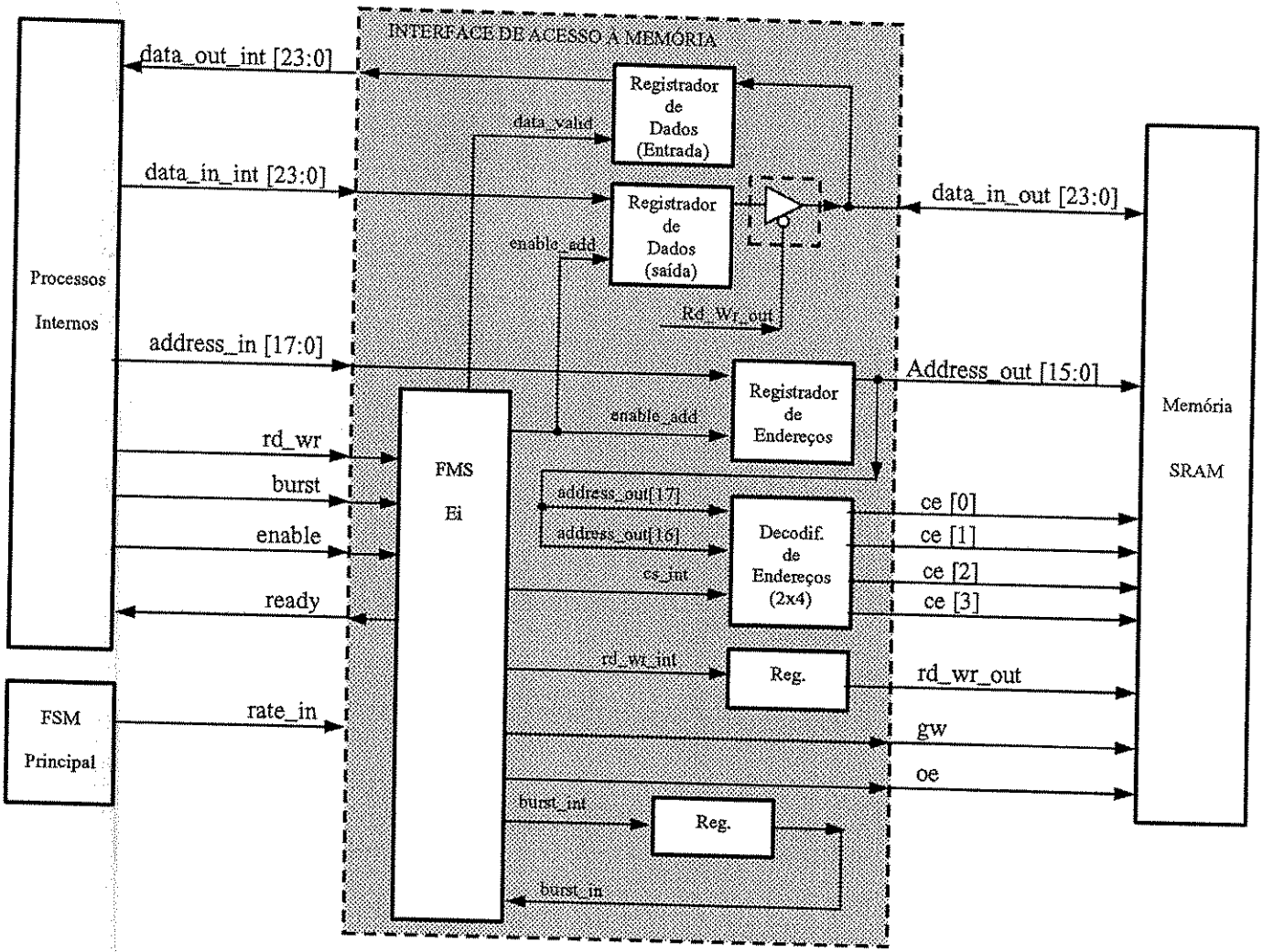


Figura 4-4 – Diagrama em blocos da interface de aceso à memória

4.3.3 Interface I^2C

A interface I^2C, neste sistema tem o objetivo de efetuar a configuração inicial dos periféricos externos tais como o decodificador de vídeo e codificador de vídeo, durante o processo de inicialização. O padrão I^2C foi desenvolvido com o intuito de interfacear dispositivos em um mesmo sistema [20].

4.3.3.1 Diagrama em Blocos

O diagrama em blocos da interface I^2C, implementada em *hardware* reconfigurável, bem como suas conexões internas e externas é apresentado pela Figura 4-5.

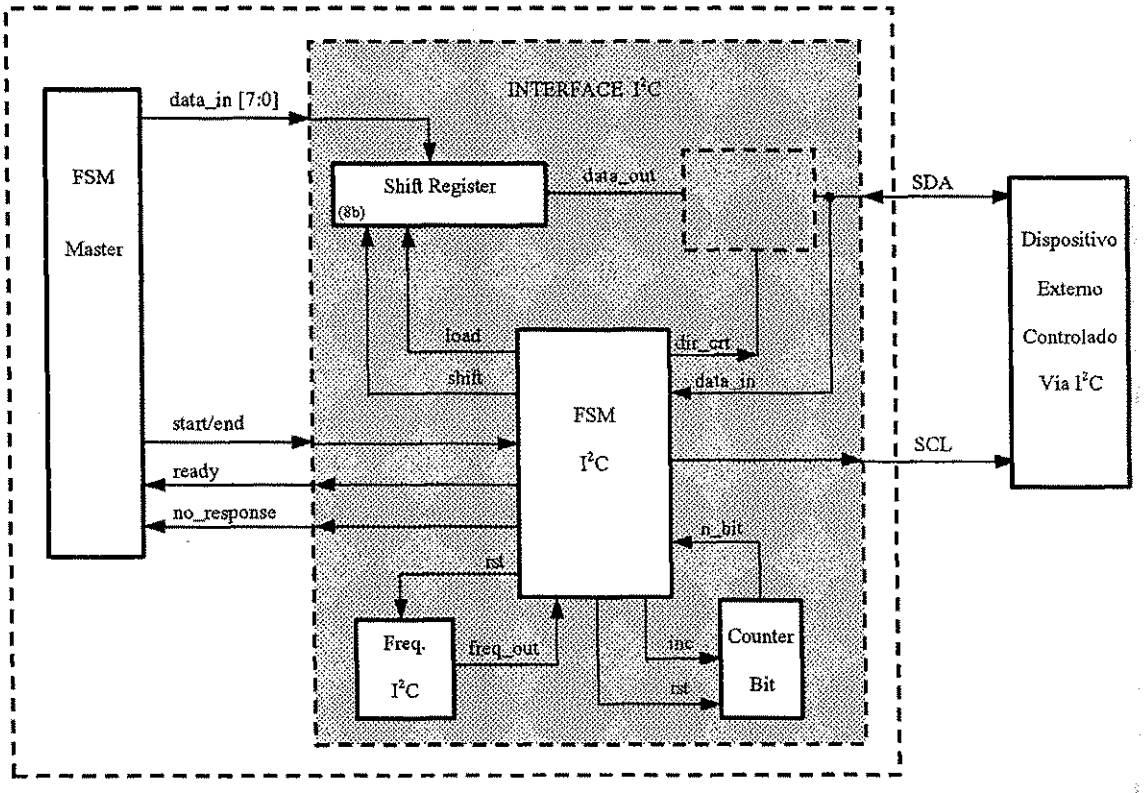


Figura 4-5 - Diagrama em blocos da interface I^2C

4.3.4 Controlador de Cache (CC)

O bloco controlador de cache é responsável pelo controle de carga ou descarga de áreas de memória especificadas pelo sistema, bem como o armazenamento e controle de acesso aos bancos de memória internos utilizados como cache.

O bloco de cache é dividido em:

. Módulo de controle do Cache

. Bancos de cache A e B

4.3.4.1 Módulo de Controle do Cache

Este bloco controla o acesso ao cache e o fluxo de dados durante as operações de carga e descarga do cache na SRAM, através da interface de acesso à memória. Recebe uma configuração inicial através do barramento de controle (ctr_cache) contendo o endereço inicial de carga do cache, o tamanho do bloco de carga, o endereço inicial de descarga do cache e o tamanho do bloco de descarga. Ao receber um comando de carga (cache_load) ou descarga do cache (cache_unload) o controlador inicia o processo a partir do endereço inicial até atingir o tamanho do bloco especificado. Enquanto estiver efetuando uma operação de carga ou descarga do banco inativo, o controlador sinaliza (cache_busy) para o sistema, o que não impede a realização de qualquer operação de escrita (write_cache) ou leitura (read_cache) do banco ativo, que tem endereço e barramento de dados independente do banco inativo.

4.3.4.2 Bancos de Cache

Os bancos de cache armazenam os dados de uma parte da memória SRAM externa com o objetivo de acelerar o acesso. Os dois bancos de cache (A e B) têm a mesma função, estando sempre um deles ativo, enquanto o outro estará sendo carregado ou descarregado para a memória externa. O tamanho dos bancos é de (3x240) 720 posições de 24 bits cada um, suficiente para armazenar 03 posições de memória para cada *pixel* de uma linha. Este processo será realizado da seguinte maneira.

- O banco A estará ativo (conectado ao barramento interno) e previamente carregado com um bloco de dados da SRAM (linha n) e estará sendo acessado pelo barramento interno de leitura/escrita.

- O banco B estará inativo (conectado ao seu barramento de controle) efetuando uma operação de descarga (atualização) da memória, para uma área definida pelo bloco de controle, através do bloco de interface de acesso à memória, com os valores adquiridos no processamento da linha anterior (linha n-1). Uma vez concluído o procedimento de descarga do cache, o banco B pode iniciar uma operação de carga dos dados de uma área da SRAM a serem processados na linha seguinte (linha n+1).

- Finalizado o processo de carga dos dados para o banco B, e concluído o processamento interno com o banco A, é efetuada a inversão dos barramentos dos bancos A e B, ficando o banco A inativo e conectado ao barrramento de controle, enquanto o banco B (previamente carregado) estará ativo e conectado ao barrramento interno, pronto para o acesso interno dos dados. Na Figura 4-6 é apresentada uma descrição esquemática do processo.

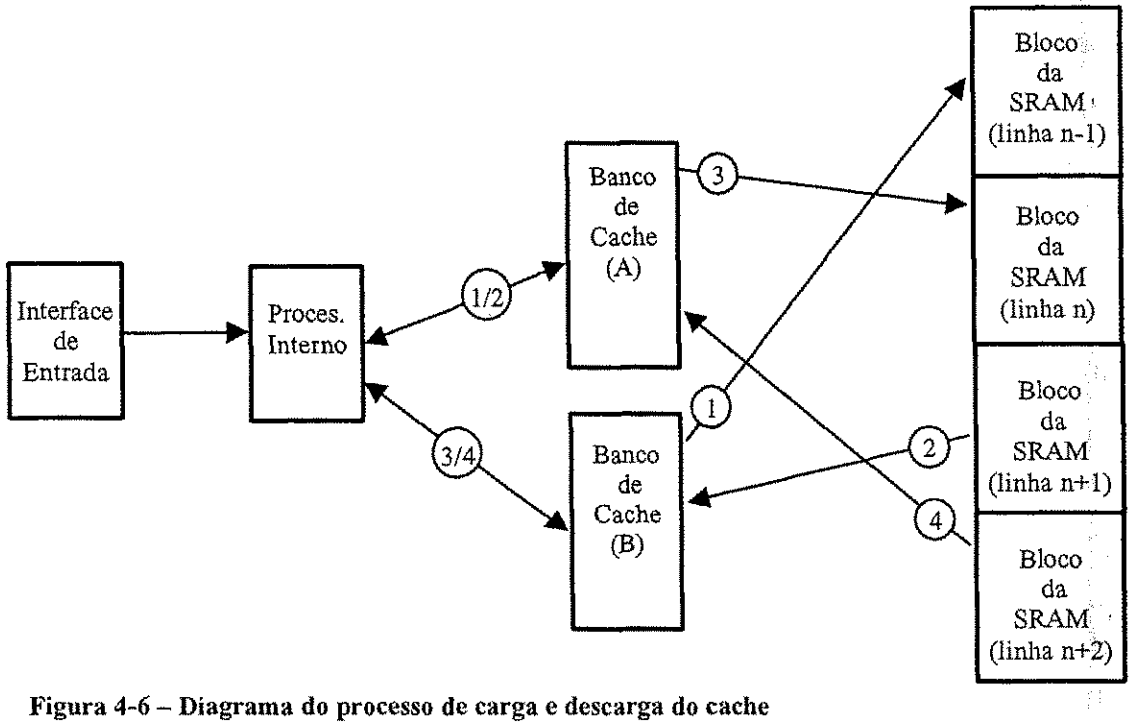


Figura 4-6 – Diagrama do processo de carga e descarga do cache

4.3.4.3 Diagrama em Blocos

O diagrama em blocos simplificado do controlador de cache, implementado em *hardware* reconfigurável, bem como suas conexões com a interface IAM e as conexões internas é apresentado pela Figura 4-7.

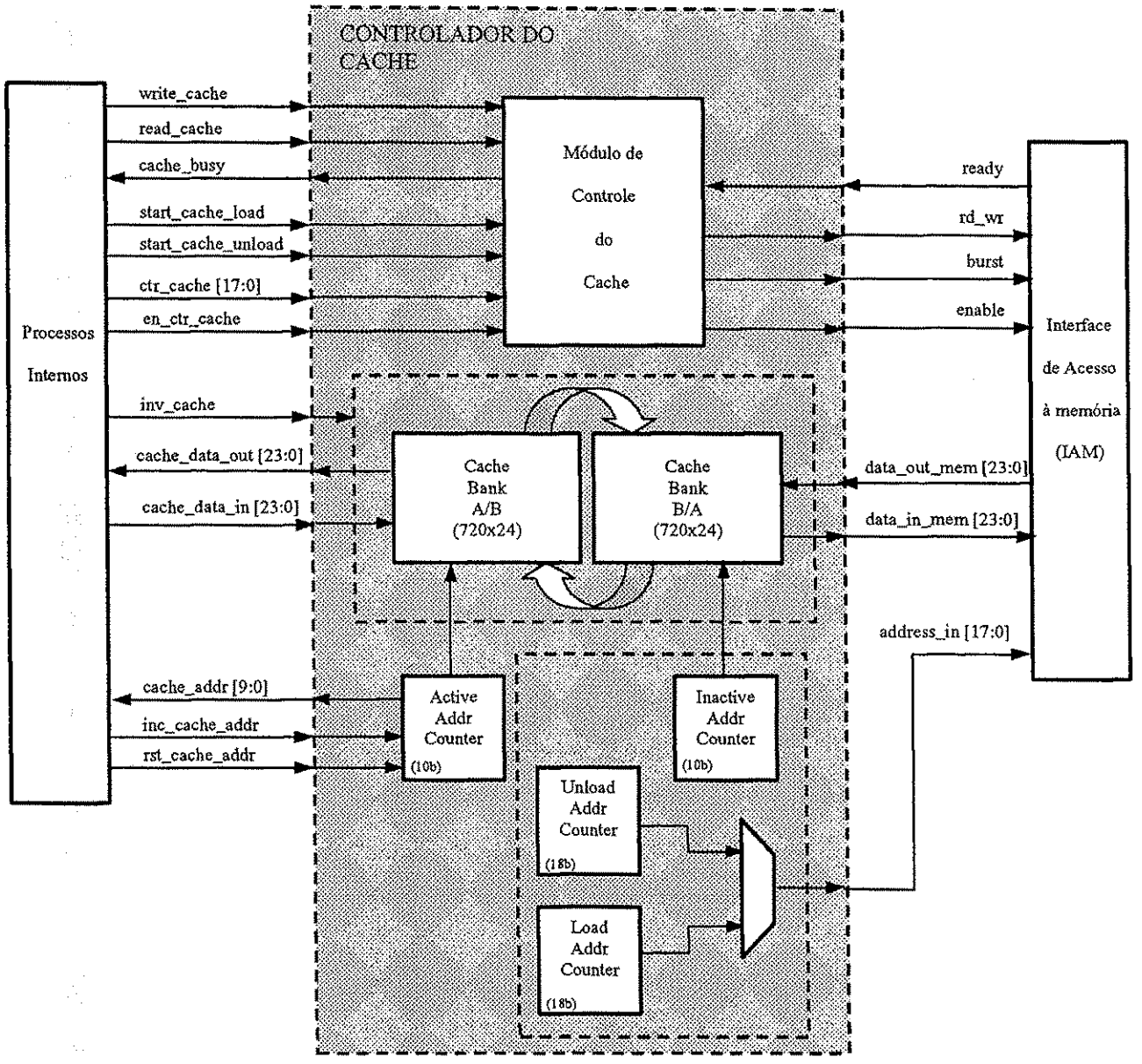


Figura 4-7 - Diagrama simplificado do controlador de cache

4.3.5 Interface com o Codificador de Vídeo (ICV)

A interface com o codificador de vídeo, é o bloco responsável pela apresentação ou monitoramento da imagem processada. O sinal digital previamente processado deve ser adequadamente entregue à placa codificadora de vídeo. A interface ICV seleciona linha e *pixel* a ser apresentado a cada instante, mantendo a compatibilidade entre as diferentes taxas de bits e linhas amostradas para o processamento, e a especificação do padrão M.

A interface com o codificador de vídeo é composta por 3 módulos básicos. O controlador do codificador de vídeo, o controlador do buffer de saída e o buffer de saída, que serão descritos no decorrer deste tópico.

Conforme descrito no item 4.2.2 deste capítulo adotamos 240 *pixels* por 160 linhas como padrão para a amostragem das imagens a serem processadas. No entanto o sinal esperado pelo codificador de vídeo é de 720 *pixels* por 480 linhas, gerando a necessidade de efetuarmos uma adequação na seqüência para apresentação da imagem no codificador de vídeo.

. Apresentação de *Pixels*

Como durante o processo de aquisição da imagem assumimos uma resolução de 240 *pixels*/linha, para a apresentação da imagem a uma taxa de 720 *pixel*/linha podemos simplesmente efetuar o processo inverso ao que adotamos na aquisição da seqüência de imagens, e repetir o mesmo *pixel* durante 3 ciclos seguidos de escrita no codificador de vídeo, o que significa, efetuar a escrita no codificador e uma taxa 3 vezes menor, ou seja entregamos os 240 *pixels* de uma linha a uma freqüência de 4.5 MHz, conforme mostrado pela Figura 4-8 .

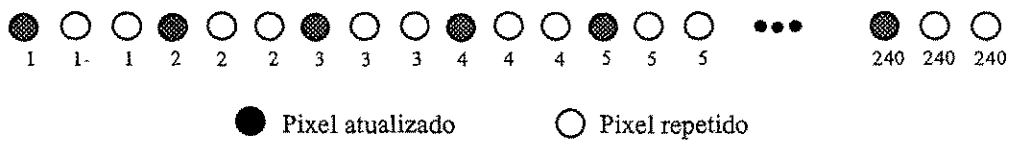


Figura 4-8 – Esquema para apresentação dos pixels processados

. **Apresentação das linhas**

O processo para apresentação da seqüência correta de linhas é ligeiramente mais complicado que a apresentação de *pixels*. Para evitar erros na seqüência de apresentação das linhas não podemos simplesmente repetir a mesma linha por 3 vezes consecutivas. Como a seqüência de linhas é entrelaçada, o processo deve ser executado de maneira a garantir o perfeito entrelaçamento das linhas, conforme mostrado pela Figura 4-9.

Ao apresentar o campo impar a interface ICV deve alimentar o codificador com a seguinte seqüência de linhas: 1,1,4,7,7,10,13,13,16,19,19,22... e assim sucessivamente. Ao apresentar o campo par a seqüência deve ser: 4,4,7,10,10,13.16,16,19,22,22,25 ... e assim sucessivamente. No entanto, não podemos esquecer de que durante o processo de aquisição deliberadamente desprezamos duas a cada três linhas. Assim, a seqüência de linhas efetivamente processadas e armazenadas na memória para apresentação (linhas válidas) passa a ser: 1,1,2,3,3,4,5,5,6,7,7,8...239,239,240 no campo impar e , 1,2,2,3,4,4,5,6,6,7,8,8...239,240,240 no campo par. Portanto a interface deve ter a capacidade de armazenamento de pelo menos duas linhas completas, uma do campo par e outra do campo impar para serem apresentadas na seqüência definida no parágrafo a cima e que pode ser visualizada conforme as setas laterais na Figura 4-9.

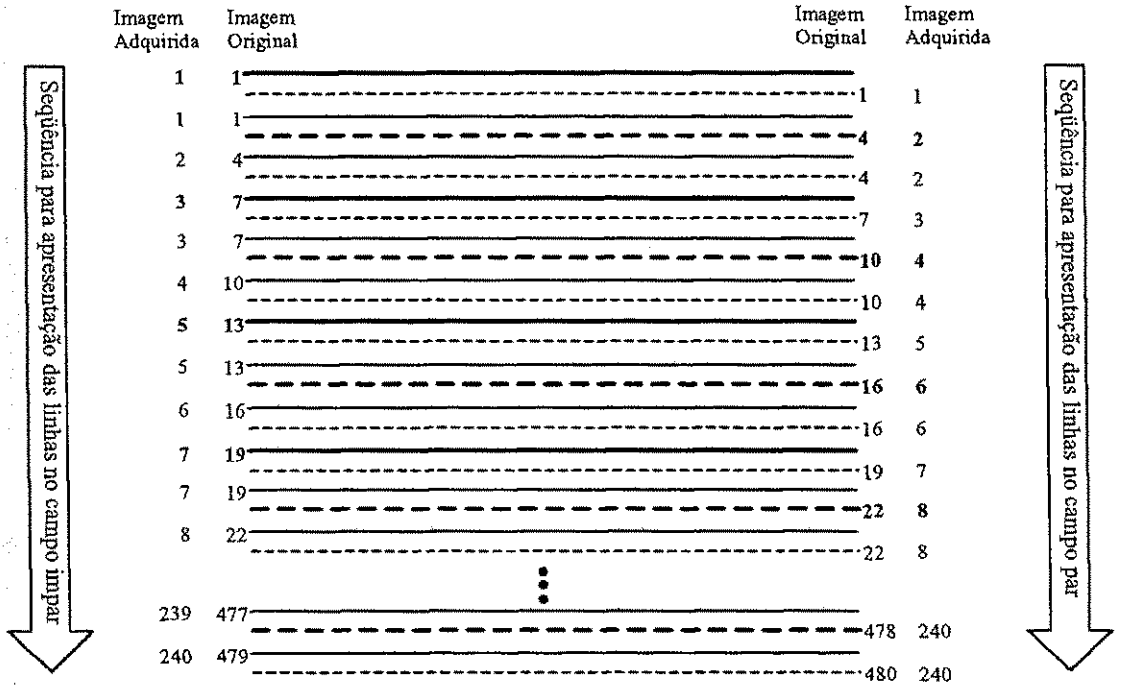


Figura 4-9 – Seqüência para apresentação das linhas processadas

4.3.5.1 Controlador do Codificador de Vídeo

O controlador do codificador de vídeo fornece os dados dos *pixels* de saída (resultantes do processamento), diretamente para a placa codificadora no formato RGB 24 bits, juntamente com os sinais de sincronismo horizontal e vertical.

4.3.5.2 Controlador do Buffer de Saída

O controlador do buffer realiza o acesso compartilhado do buffer de saída, entre a descarga no controlador do codificador de vídeo e a carga de linhas processadas armazenadas da memória SRAM. Funciona de forma análoga ao módulo de controle do cache, descrito anteriormente no tópico 4.3.4.1.

4.3.5.3 Buffer de Saída

O buffer de saída é um cache interno composto por 2 bancos (A e B) cada um com capacidade para uma linha processada, 240 posições de 24 bits, estando um deles conectado ao controlador do codificador de vídeo alimentando a saída, enquanto o outro está conectado à interface de acesso à memória sendo atualizado com a próxima linha a ser apresentada. A atualização dos dados da próxima linha ocorre nos intervalos "ociosos" de acesso à memória, nos quais o controlador de cache (tópico 4.3.4) já finalizou o processo de carga dos parâmetros para o processamento da imagem de entrada. O funcionamento do buffer de saída é um análogo simplificado dos bancos de cache descritos no tópico 4.3.4.2, uma vez que o banco ativo será sempre sendo lido enquanto o inativo estará sendo escrito.

Como já mostrado quando descrito o processo de apresentação das linhas processadas, para a apresentação de um campo (par ou impar) são necessárias informações referentes a linhas de 2 campos da imagem original. Dessa forma, enquanto a leitura do banco ativo, que contem informação da "linha n" de um campo, é repetida duas vezes pelo o controlador do codificador de vídeo, o banco inativo é carregado com a "linha n+1" do outro campo.

4.3.5.4 Diagrama em Blocos

O diagrama em blocos simplificado da interface ICV, implementada em *hardware* reconfigurável, bem como suas conexões com a placa do codificador de vídeo são apresentado pela Figura 4-10.

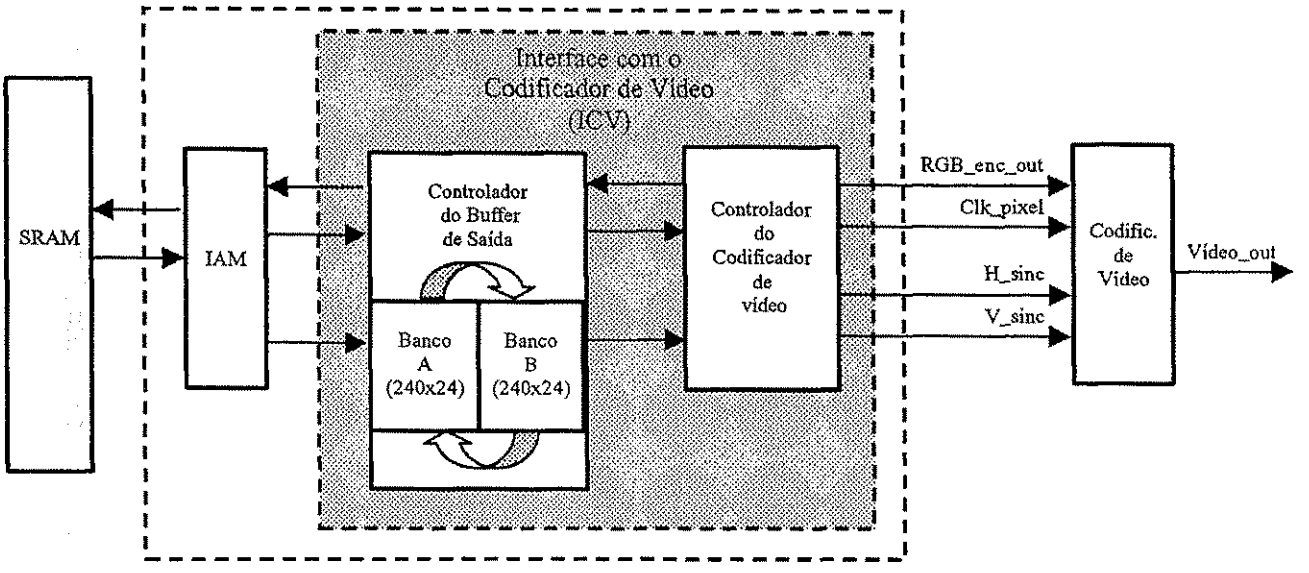


Figura 4-10 – Diagrama em blocos simplificado da interface com o codificador de vídeo

4.4 Resumo do Capítulo

Neste capítulo foi descrito o desenvolvimento da plataforma GPIP-01 para aplicações em processamento de imagens, com capacidade para acomodar o sistema para extração de objeto proposto neste trabalho. Como um complemento à implementação do *hardware* fixo, foi necessária a elaboração do sistema de interfaceamento que possibilita o controle e acesso aos dispositivos externos à FPGA, além de módulos auxiliares tais como o cache de memória interna com o propósito de aumentar a velocidade de acesso à memória SRAM externa, viabilizando a implementação do sistema em tempo real. Este sistema foi implementado em verilog, testado e encontra-se embarcado na plataforma, disponível para utilização. Finalizada a apresentação da plataforma GPIP-01, pode-se partir para a implementação física do algoritmo de segmentação e extração do objeto proposto. No capítulo seguinte são apresentadas as adaptações necessárias, bem como a metodologia proposta para solução do problema.

5 Implementação Parcial do Algoritmo Proposto

Com base nas informações descritas nos capítulos anteriores, podemos partir para a implementação do algoritmo proposto. Neste capítulo são descritas as adaptações necessárias à aplicação em questão, os processos envolvidos na aquisição da imagem de referência e na extração de objeto baseado na técnica adotada, bem como a metodologia a ser seguida no desenvolvimento do projeto.

A implementação da técnica escolhida pode ser dividida em 4 etapas ou processos distintos, apresentados a seguir e descritos no decorrer deste capítulo.

. Processo de aquisição da seqüência de imagens;
. Processo de geração da imagem de fundo de referência;
. Processo de seleção dos limiares;
. Processo de extração do objeto;

Segundo a literatura [6], para uma implementação em tempo real seria necessário a utilização de dois computadores tipo Pentium operando a 400 MHz trabalhando em processamento paralelo, que é uma alternativa interessante para validação da idéia, mas muito onerosa para nossos propósitos. Como estamos objetivando uma implementação de baixo custo, para aplicação em produtos de consumo esta solução se mostra inviável. Soluções utilizando DSP's (Digital Signal Processor) para aplicações em vídeo são possíveis, mas também muito caras, quando se considera a produção de altos volumes. Para atender aos requisitos de velocidade de processamento e custo final, temos a possibilidade da implementação de um ASIC, no qual teremos o algoritmo implementado em *hardware*.

5.1 Adaptações Necessárias

Em uma avaliação mais detalhada do algoritmo no qual está baseado este trabalho, considerando imagens com resolução de 320x240 *pixels* [21], com cada *pixel* representado por suas três componentes RGB de 8 bits, e o processamento estatístico utilizando um universo de 100 quadros de imagem de referência para a geração dos parâmetros [6], verifica-se uma grande demanda de processamento e memória para o armazenamento do conjunto de parâmetros $<E_t, S_t, a_t, b_t>$ relativos a cada *pixel*, sendo que os parâmetros E_t e S_t são representados por suas três componentes em RGB. Dessa forma, nota-se a necessidade de efetuarmos adaptações buscando otimizar os recursos de memória disponíveis no sistema, através da adequação do algoritmo às particularidades da aplicação em questão, principalmente no que se refere a operações utilizando ponto flutuante, amplamente utilizada nos cálculos propostos no algoritmo original e que para nossa aplicação em *hardware* devem ser evitados ou mesmo eliminados completamente através de artifícios e arranjos matemáticos que nos permitam manter a precisão necessária, sem o comprometimento da quantidade de recursos lógicos utilizados.

5.2 Aquisição da Seqüência de Imagens

Nesta etapa ocorre a aquisição dos dados a serem processados, que embora esteja fora do escopo deste trabalho, precisa ser bem definida e deve atender a determinados requisitos já que todos os processos subseqüentes dependem da formatação dos dados adquiridos.

Os elementos utilizados para aquisição da imagem devem ainda atender ao requisito de baixo custo, portanto deve-se lançar mão de uma câmera digital, disponível no mercado, ou de uma câmera analógica convencional aplicada a um módulo conversor de vídeo para RGB.

Neste trabalho adotamos a plataforma de *hardware* GPIP-01 desenvolvida com o intuito de prover os meios físicos necessários para esta implementação, dotada de capacidade de aquisição, processamento, armazenamento e apresentação da imagem processada.

Estando definida a plataforma de desenvolvimento e as interfaces de controle e acesso ao *hardware* externo, podemos iniciar a descrição dos blocos internos responsáveis pelo processamento do sinal, neste caso efetuando a operação de extração do objeto de um fundo heterogêneo e conhecido. Na técnica adotada neste trabalho, o processo de extração de objeto é composto por 3 etapas bem distintas que são explicitadas a seguir e podem ser apresentadas no formato de um fluxograma como mostrado pela Figura 5-1 e que deve ser implementado pelo sistema.

As 3 etapas envolvidas no processo de extração de objeto são:

. Modelamento da imagem de referência (geração dos parâmetros)

- . Parâmetro E_i - Imagem média
- . Parâmetro S_i - Desvio médio
- . Parâmetro A_i - Variação da distorção de brilho
- . Parâmetro B_i - Variação da distorção de cromaticidade

. Determinação automática dos níveis de limiar

- . Geração dos Histogramas
- . Seleção dos limiares

. Extração do objeto

- . Classificação dos *pixels* da imagem
- . Exibição do Objeto

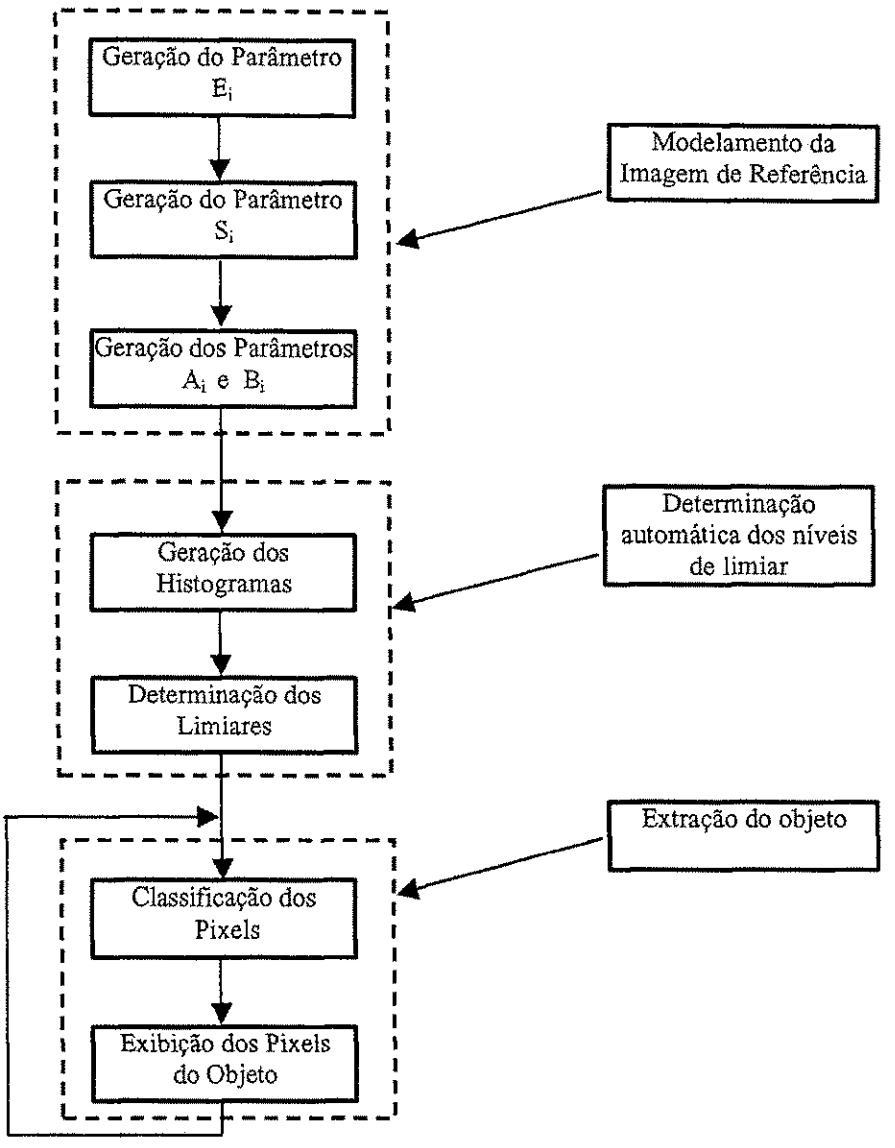


Figura 5-1 – Fluxograma do processo completo de extração de objeto.

Na Figura 5-2 é apresentada uma visão geral do sistema completo em blocos, mostrando o conjunto de processos a serem implementadas em *hardware* reconfigurável, necessários para a execução do processamento das imagens em tempo real, juntamente com o restante do sistema de *hardware* fixo e interfaceamento. Cada um dos blocos referentes ao processamento interno a ser implementado, será descrito em detalhes no decorrer deste capítulo conforme a técnica apresentada no capítulo 3 deste trabalho. Os demais já foram mostrados e descritos anteriormente.

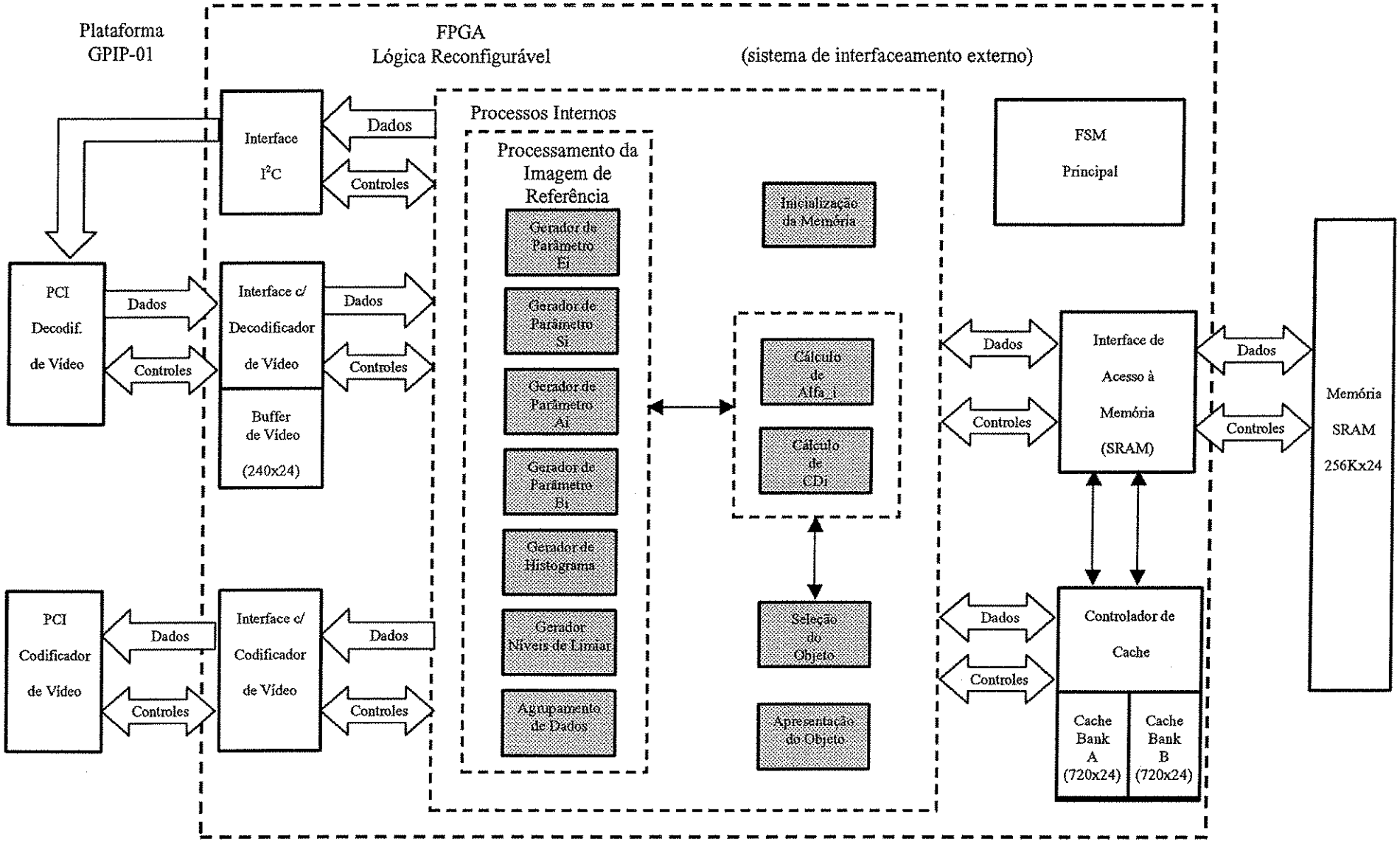


Figura 5-2 – Diagrama geral do sistema embarcado na plataforma GPIP-01

5.3 Geração da Imagem de Fundo de Referência

A geração da imagem de fundo de referência consiste no processo de geração dos parâmetros para armazenamento da imagem que servirá de referência para a extração do objeto. O processo de "aprendizado" da imagem de fundo, nesta técnica, pode ser executado durante a inicialização do sistema e portanto, não precisa necessariamente ocorrer em tempo real, embora seja desejável que sua execução não tenha uma demanda de tempo superior a 10 segundos, para o caso da necessidade de efetuarmos correções ou reaquisições da imagem de referência utilizando períodos de ausência de objeto em frente á câmera. Vale salientar que a imagem de referência deve ser necessariamente estática.

O algoritmo adotado tem uma abordagem estatística e, portanto precisa de um espaço amostral significativo. Considerando o tempo de execução, demanda de memória e relevância do universo de amostras, bem como a conveniência para as operações binárias, adotamos o número de 64 quadros para o processo de geração da imagem de fundo de referência. Uma vez definido o espaço amostral, passemos para a fase de geração dos parâmetros para a construção da imagem de referência $<E_i, S_i, a_i, b_i>$.

5.3.1 Geração do Parâmetro E_i

O parâmetro E_i é formado pelas suas três componentes no espaço RGB, $E_i = \left[\mu_{i(R)}, \mu_{i(G)}, \mu_{i(B)}\right]$ e representa a média dos valores de N = 64 amostras medidas para cada um dos *pixels* (x,y), como mostrado em (5-1) a seguir.

$$E_i = \left[\mu_{i(C)}\right] = \frac{1}{N}\sum_{i=0}^{N-1} P_{i(C)} \qquad \textbf{(5-1)}$$

Na qual, $P_{i(C)}$ representa o valor digitalizado de um determinado *pixel* i sendo C cada uma das componentes de cor R, G e B.

5.3.1.1 Descrição do Processo

Para melhor compreensão do processo para a geração do parâmetro E_i, a seguir é apresentada a seqüência de operações a serem executadas pelo *hardware*.

- Inicializar os acumuladores
- Carregar os acumuladores correspondentes à primeira linha no cache
- Capturar cada *pixel* corrente da linha corrente para o buffer de entrada
- Ler um *pixel* do buffer de entrada (RGB)
- Ler os acumuladores de R, G e B do *pixel* correspondente no cache
- Efetuar a soma dos acumuladores com os valores de R, G e B do *pixel* corrente
- Armazenar os novos valores dos acumuladores no cache
- Repetir este processo para todos os 240 *pixels* da linha
- Descarregar o cache de linha na memória
- Repetir o processo para todas as linhas ímpares e pares
- Se fim de quadro, incrementar o contador de quadros
- Repetir o processo durante 64 quadros
- Finalizado o processo de acumulação
- Carregar os acumuladores correspondentes à primeira linha no cache
- Efetuar a divisão de cada acumulador por 64
- Armazenar os valores no cache, no formato |R|G|B| (24bits)
- Repetir o processo para toda a linha
- Descarregar o cache de linha na memória
- Repetir o processo para todas as linhas ímpares e pares
- Fim do processo.

5.3.1.2 Definições Numéricas

Considerando a soma das 64 amostras de 8 bits, cada um dos acumuladores para R, G e B de cada *pixel* deve ter pelo menos (255 x 64 = 16320) 14 bits, e ocupa uma posição de memória (24 bits), portanto para cada *pixel* (3 acumuladores) teremos 03 posições de memória. Para armazenar todos os acumuladores utilizamos (240x160x3) 115.200

posições de memória para armazenamento temporário durante o processo de geração do parâmetro. Para o armazenamento definitivo do parâmetro são utilizadas apenas 38.400 posições, já que o parâmetro E_i é um número inteiro (não requer casas decimais) e pode ser representado com um número binário de 8 bits, apenas uma posição é necessária para o armazenamento dos valores de R,G e B de cada *pixel* como pode ser verificado pela Figura 5-3 .

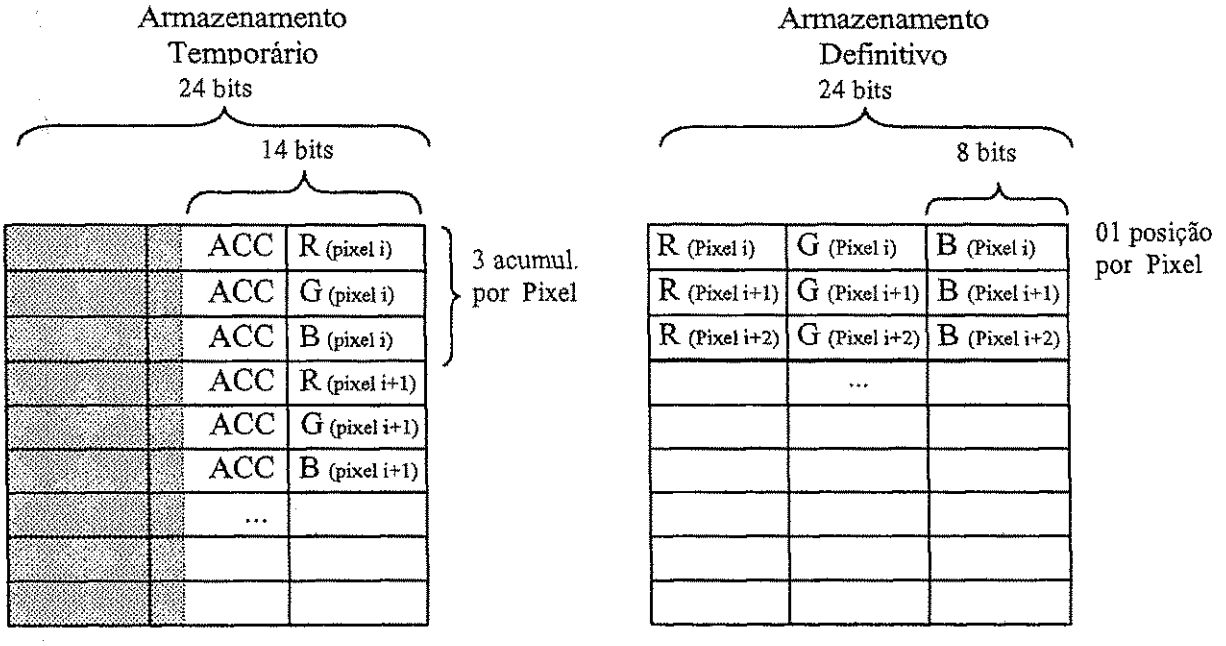


Figura 5-3 – (a) Armazenamento temporário do parâmetro Ei ; (b) Armazenamento definitivo

5.3.1.3 Diagrama em Blocos

O diagrama em blocos do *hardware* responsável pela implementação do processo de geração do parâmetro E_i, bem como suas conexões com os módulos periféricos é apresentado pela Figura 5-4 .

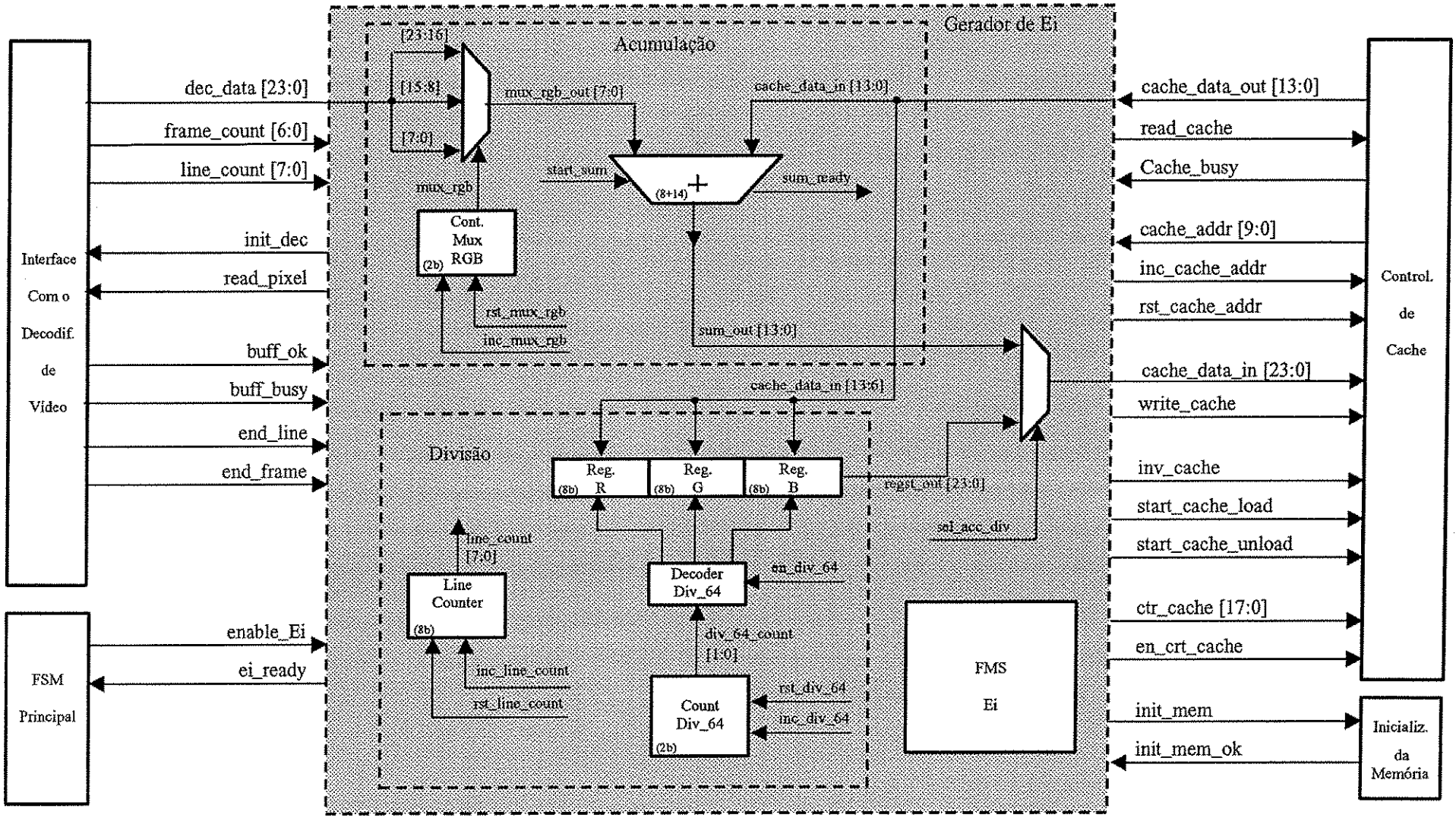
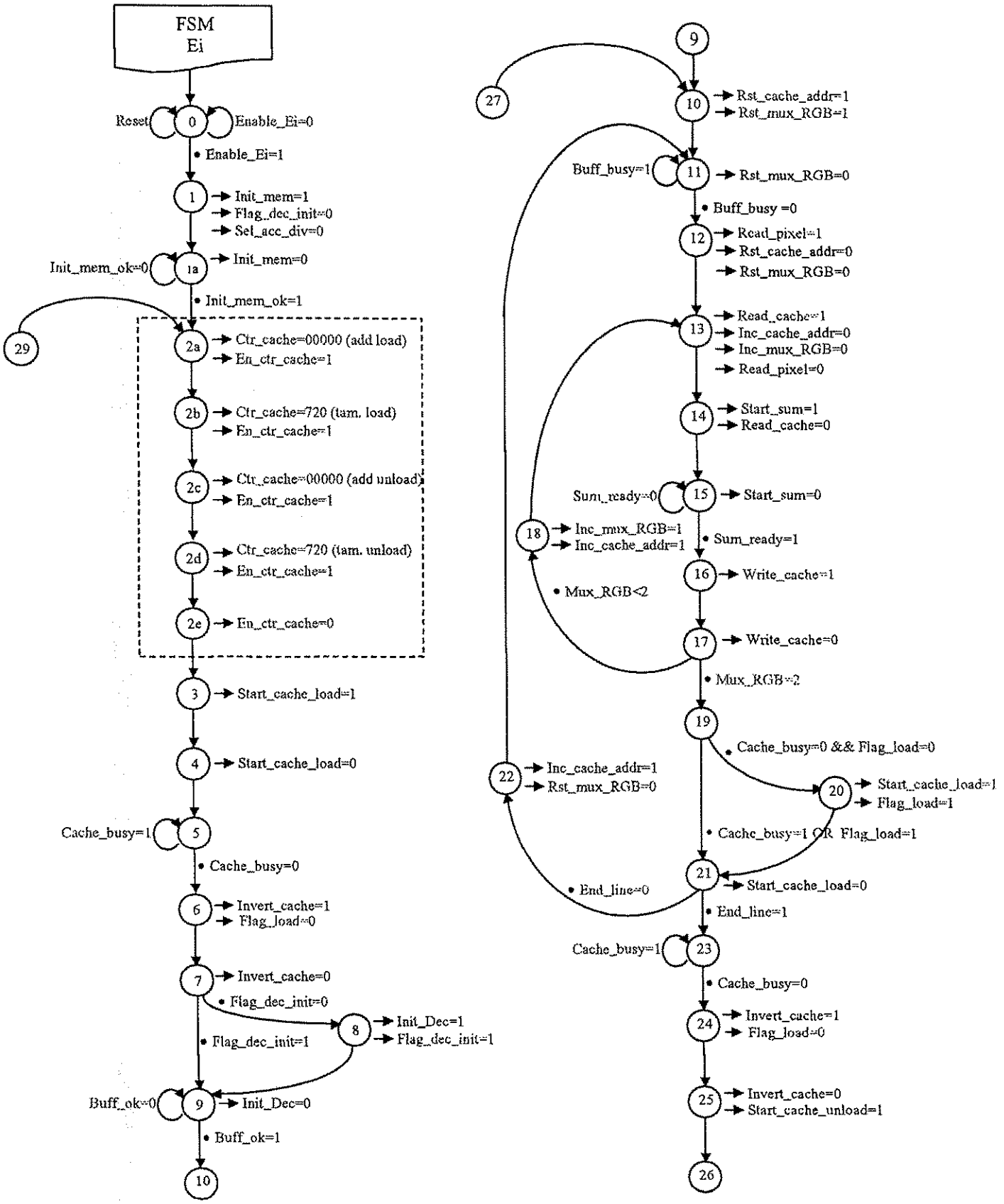
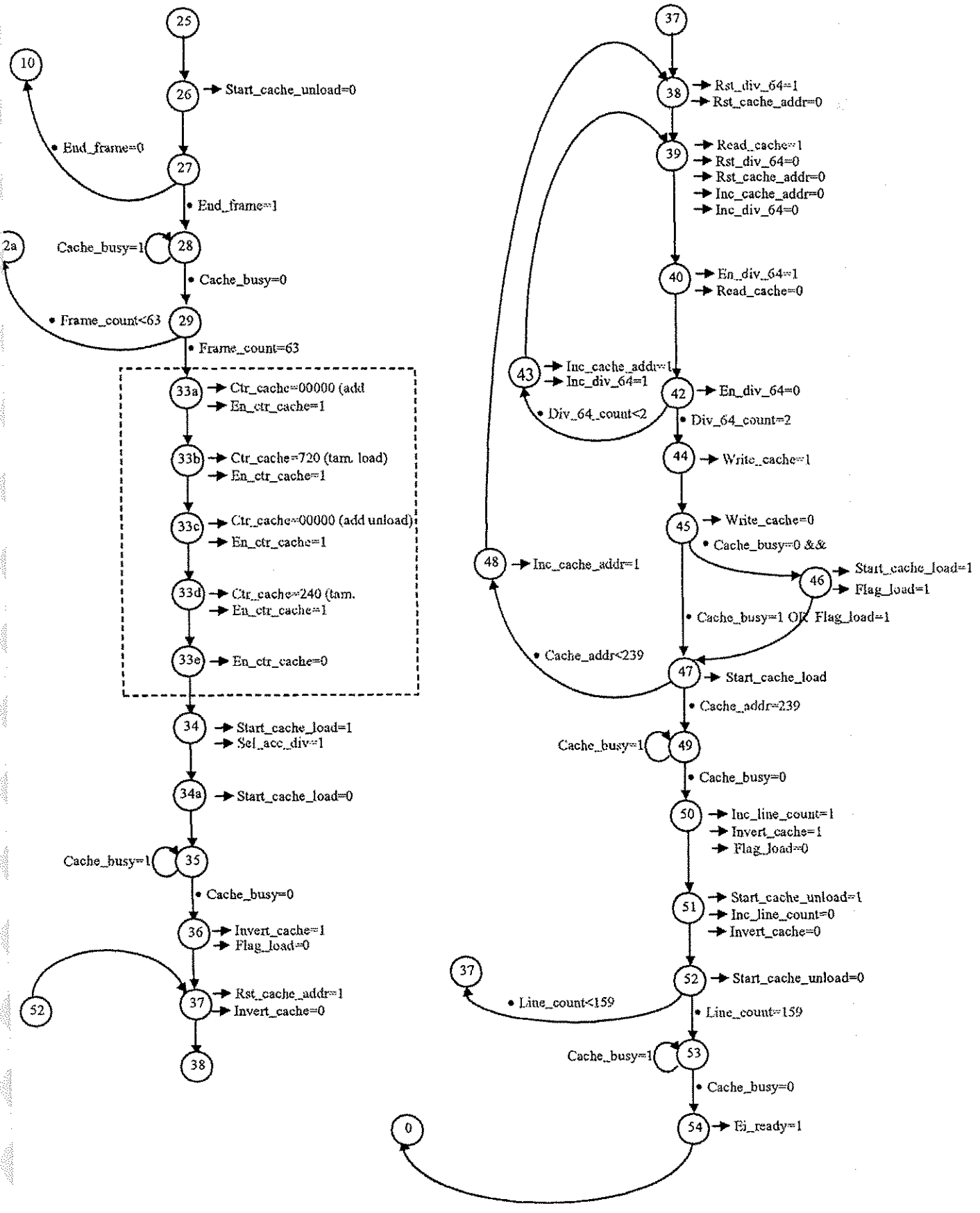


Figura 5-4 - Diagrama em blocos do Módulo Gerador do parâmetro E_i

5.3.1.4 Máquina de Estados




25
10
26
Start_cache_unload=0
End_frame=0
27
End_frame=1
2a
Cache_busy=1
28
Cache_busy=0
Frame_count<63
29
Frame_count=63
33a
Ctr_cache=00000 (add
En_ctr_cache=1
33b
Ctr_cache=720 (tam. load)
En_ctr_cache=1
33c
Ctr_cache=00000 (add unload)
En_ctr_cache=1
33d
Ctr_cache=240 (tam.
En_ctr_cache=1
33e
En_ctr_cache=0
34
Start_cache_load=1
Sel_acc_div=1
34a
Start_cache_load=0
Cache_busy=1
35
Cache_busy=0
36
Invert_cache=1
Flag_load=0
52
37
Rst_cache_addr=1
Invert_cache=0
38
37
38
Rst_div_64=1
Rst_cache_addr=0
39
Read_cache=1
Rst_div_64=0
Rst_cache_addr=0
Inc_cache_addr=0
Inc_div_64=0
40
En_div_64=1
Read_cache=0
43
Inc_cache_addr=1
Inc_div_64=1
42
En_div_64=0
Div_64_count<2
Div_64_count=2
44
Write_cache=1
45
Write_cache=0
Cache_busy=0 &&
48
Inc_cache_addr=1
46
Start_cache_load=1
Flag_load=1
Cache_busy=1 OR Flag_load=1
Cache_addr<239
47
Start_cache_load
Cache_addr=239
Cache_busy=1
49
Cache_busy=0
50
Inc_line_count=1
Invert_cache=1
Flag_load=0
51
Start_cache_unload=1
Inc_line_count=0
Invert_cache=0
37
52
Start_cache_unload=0
Line_count<159
Line_count=159
Cache_busy=1
53
Cache_busy=0
0
54
Ei_ready=1

5.3.2 Geração do Parâmetro S_i

O parâmetro S_i é formado pelas suas três componentes no espaço RGB, $S_i = \left[\sigma_{(R)}, \sigma_{(G)}, \sigma_{(B)}\right]$ e representa uma "estimativa" da variação [1] das sensibilidades dos sensores da câmera para R, G e B expresso em (3-4), reescrita a seguir.

$$S_i = \left[\sigma_{(C)}\right] = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{i(C)})^2}$$

Na qual, $P_{i(C)}$ representa o valor digitalizado de um determinado *pixel* i sendo C cada uma das componentes de cor R, G e B. Note que para o cálculo do parâmetro S, deve-se calcular previamente o parâmetro E_i.

Neste trabalho para a geração do parâmetro S_i calculamos o desvio em R, G e B de cada um dos *pixels* em 27 quadros. O número de 27 quadros foi escolhido por atender aos requisitos estatísticos de um amplo espaço amostral e facilitar os cálculos permitindo uma divisão binária com um erro bastante reduzido (1,1%) para duas casas decimais. A partir deste ponto, todos os cálculos efetuados exigem a precisão de 2 casas decimais, de forma que devemos lançar mão de alguns artifícios matemáticos que nos permitam efetuar as operações em ponto fixo, mantendo esta equivalência de precisão com o mínimo de requisito de *hardware*.

Para o caso do parâmetro S_i faremos:

$$\sigma^2_{(C)} = \frac{1}{M}\sum_{i=0}^{M-1}(P_{i(C)} - \mu_{i(C)})^2 \quad \text{p/ 27 quadros} \quad \Rightarrow M = 240\text{x}160\text{x}27 = 1.036.800$$

$$\sigma^2_{(C)} = \frac{1}{1.036.800}\sum_{i=0}^{M-1}(P_{i(C)} - \mu_{i(C)})^2 \; ; \quad \text{ou ainda} \quad \sigma^2_{(C)} = \frac{1}{4.050 \text{ x } 2^8}\sum_{i=0}^{M-1}(P_{i(C)} - \mu_{i(C)})^2$$

Fazendo $\mathrm{Dp}_{(C)} = 2^8 \times \sigma^2_{(C)} = \frac{1}{4.050}\sum_{i=0}^{M-1}(P_{i(C)} - \mu_{i(C)})^2$;

Dessa forma $\boxed{\sigma^2_{(C)} = \frac{\mathrm{Dp}_{(C)}}{2^8}}$; **(5-2)**

e ainda: $Dp_{(C)} = \frac{1}{4.050}\sum_{i=0}^{M-1}(P_{i(C)} - \mu_{i(C)})^2 \therefore$

$$\boxed{Dp_{(C)} \equiv \frac{1}{2^{12}}\sum_{i=0}^{M-1}(P_{i(C)} - \mu_{i(C)})^2} \qquad \text{(5-3)}$$

Portanto para mantermos a precisão numérica, evitando operações em ponto flutuante e por conveniência nas operações binárias devemos armazenar $Dp_{(C)} = 2^8 \times \sigma^2{}_{(C)}$ para utilização nas operações subseqüentes.

5.3.2.1 Descrição do Processo

O processo para a geração do parâmetro S_i consiste na seguinte seqüência:

- Rebitar os 03 acumuladores internos
- Carregar os parâmetros E_i correspondentes à primeira linha no cache
- Capturar cada *pixel* corrente da linha corrente para o buffer de entrada
- Ler um *pixel* do buffer de entrada (RG B)
- Ler os valores do parâmetro E_i em R, G e B do *pixel* correspondente no cache
- Verificar o maior ($E_{i(C)}$ ou $P_{i(C)}$)
- Efetuar a operação de subtração do maior pelo menor
- Elevar o resultado ao quadrado
- Efetuar a soma dos acumuladores com os valores de R, G e B do *pixel* corrente
- Armazenar os novos valores nos acumuladores internos
- Repetir este processo para todos os 240 *pixels* da linha
- Não é necessário descarregar o cache de linha na memória
- Repetir o processo para todas as linhas ímpares e pares
- Se fim de quadro, incrementa contador de quadros
- Repetir o processo para 27 quadros
- Finalizado o processo de acumulação
- Efetuar a divisão de cada acumulador por 2^{12} (obtendo $Dp_{(C)} = 2^8 \times \sigma^2{}_{(C)}$)
- Armazenar os valores nos 3 acumuladores internos de S_i
- Fim do processo.

5.3.2.2 Definições Numéricas

Como adotamos 02 casas decimais para os valores de S_i, e este parâmetro limitado a valores inferiores a 3 verificamos que teremos a necessidade de utilização de $(240 \times 160 \times 27 \times 3^2 = 9.331.200)$ 24bits para o armazenamento temporário de cada um dos 3 acumuladores para cada uma das componentes de cor. Com o objetivo de evitar divisões por zero deve-se estabelecer um limite mínimo para os valores de S_i [6]. O limite mínimo estabelecido será de 0,5 que corresponde a 128 em binário com duas casas decimais.

Para o armazenamento definitivo de $Dp_{(C)}$ devemos considerar $Dp_{(C)} = \frac{9.331.200}{4050} = 2.304$ (max) ou seja os 03 registros para armazenamento definitivo de $Dp_{(C)}$ ($Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$) devem ser dimensionados para 12 bits. Esta abordagem evita a extração da raiz quadrada de $\sigma^2_{(C)}$ bem como a perda de precisão no valor final, nos cálculos seguintes passaremos a utilizar Dp_C no lugar de $\sigma^2_{(C)}$, no entanto sabemos $\sigma^2_{(C)} = \frac{Dp_{(C)}}{2^8}$, e que em todos os casos a necessidade de substituição ocorre nos denominadores de frações, ao efetuarmos a substituição, devemos multiplicar o numerador pelo fator 2^8 (256) que ainda nos permitirá um acréscimo na precisão do número resultante.

5.3.2.3 Diagrama em Blocos

O diagrama em blocos do *hardware* necessário à execução do processamento de S_i é apresentado pela Figura 5-5. Notemos que todas as operações são realizadas em ponto fixo e com números inteiros sem sinal, por este motivo foi introduzido um bloco comparador antes do subtrator, dessa forma o número resultante na saída do subtrator será sempre positivo. As multiplicações e divisões binárias são realizadas pelo artifício do deslocamento nos barramentos de dados entre os blocos.

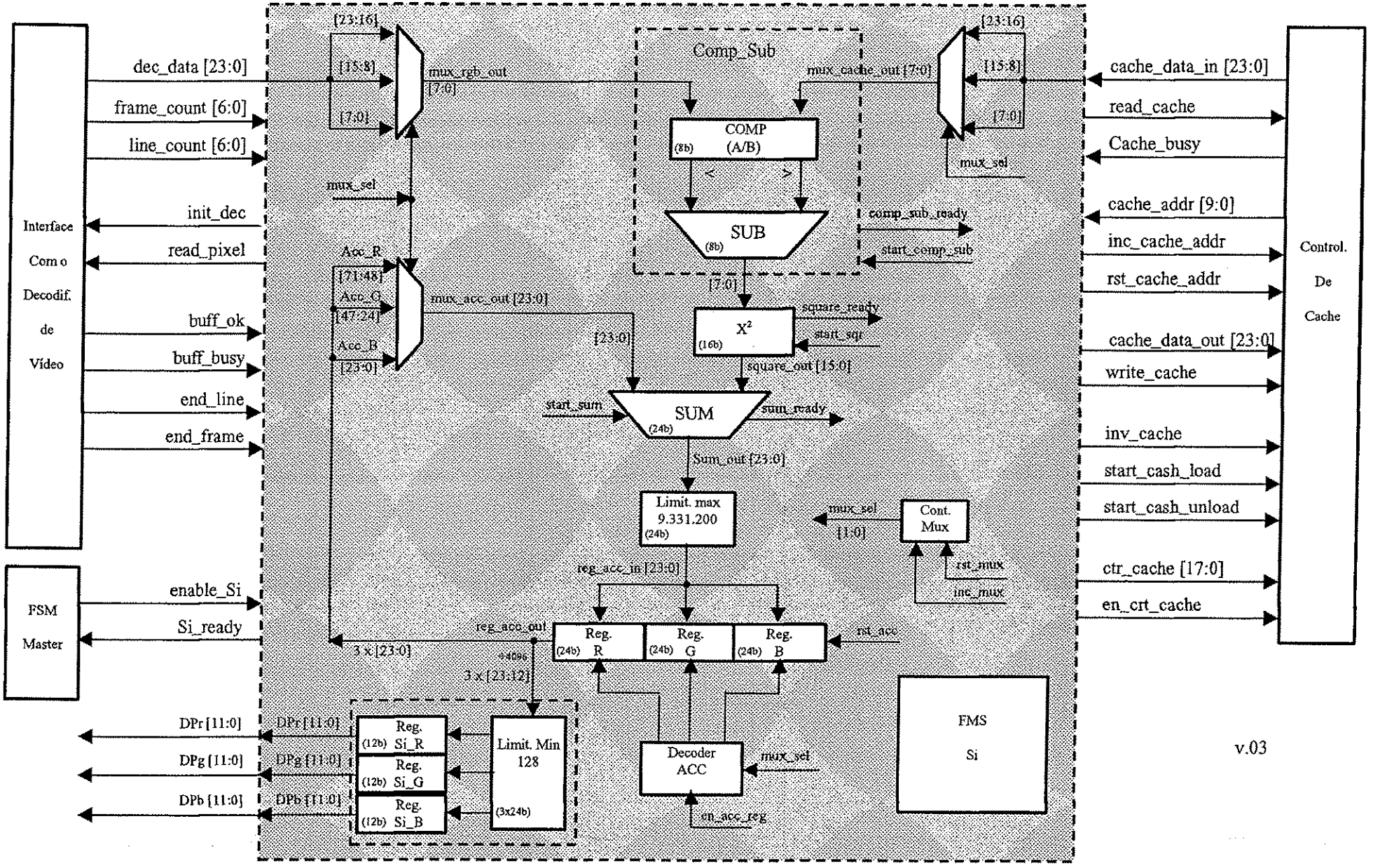
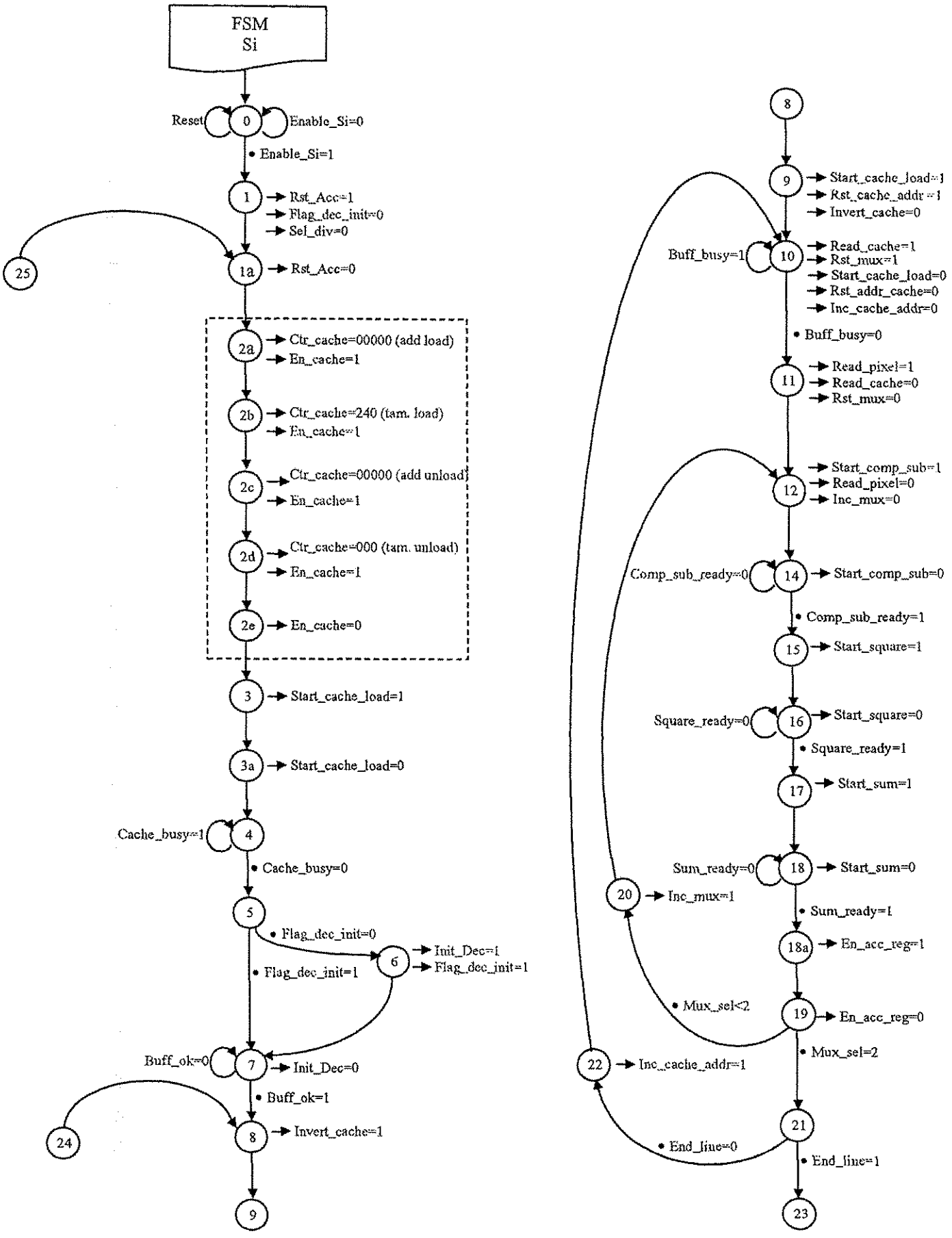
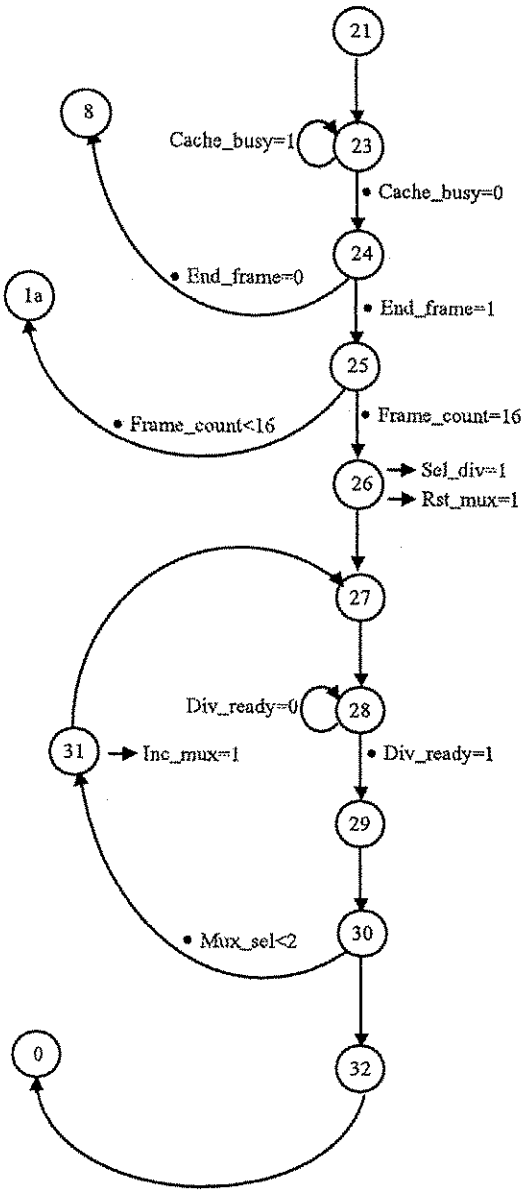


Figura 5-5 - Diagrama em blocos do processo de geração do parâmetro S_i

5.3.2.4 Máquina de Estados





Uma vez descrito processo para geração dos E_i e S_i, devemos partir para a descrição da geração do parâmetros α_i e CD_i, necessários para a geração dos demais parâmetros.

5.3.3 Geração do Parâmetro α_i

O parâmetro α_i que corresponde à distorção de brilho de cada ponto em relação à imagem de referência e é expresso por (3-8), reescrita a seguir de modo a facilitar o entendimento do texto:

$$\alpha_i = \frac{\left(\frac{I_{i(R)}\mu_{i(R)}}{\sigma_{(R)}^{\ 2}} + \frac{I_{i(G)}\mu_{i(G)}}{\sigma_{(G)}^{\ 2}} + \frac{I_{i(B)}\mu_{i(B)}}{\sigma_{(B)}^{\ 2}} \right)}{\left(\left(\frac{\mu_{i(R)}}{\sigma_{(R)}} \right)^2 + \left(\frac{\mu_{i(G)}}{\sigma_{(G)}} \right)^2 + \left(\frac{\mu_{i(B)}}{\sigma_{(B)}} \right)^2 \right)}$$

Para a geração deste parâmetro devemos capturar um *pixel* corrente e efetuar a operação conforme descrito a seguir.

Com o objetivo de facilitar o cálculo deste parâmetro tomemos o termo $\frac{I_{i(C)}\mu_{i(C)}}{\sigma_{(C)}^{\ 2}}$; na qual, C representa cada uma das componentes R,G ou B. Como já vimos anteriormente, $\sigma^2_{(C)} = \frac{\mathrm{Dp}_{(C)}}{2^8}$ portanto temos: $\frac{2^8 \times I_{i(C)} \cdot \mu_{i(C)}}{Dp_{(C)}}$, que deve ainda ser multiplicado por um fator de 2^7 para a obtenção da precisão requerida de 02 casas decimais, então façamos cada parcela do numerador $Nun_{i(C)} = \frac{2^7 \times 2^8 \times I_{i(C)} \cdot \mu_{i(C)}}{Dp_{(C)}}$, que considerando o valor mínimo de $\mathrm{Dp}_{(C)}$ (128), resulta em um número máximo com 23 bits. Utilizando o mesmo artifício obtemos as parcelas do denominador $Den_{i(C)} = \frac{2^7 \times 2^8 \times \mu_{i(C)} \cdot \mu_{i(C)}}{Dp_{(C)}}$ também com 23 bits. Somadas as 3 parcelas do numerador e as 3 parcelas do denominador o sistema multiplica o numerador por 2^7, necessário para a precisão de 2 casas decimais e efetua a divisão que resulta em um número com 10 bits, limitado pelo sistema entre 8,00 e 0,10 que são limites apenas operacionais e não afetam o resultado da avaliação uma vez que os valores de α_i efetivamente analisados sempre estarão dentro de limites menores que os estabelecidos acima.

A operação realizada será: $\alpha_i = \frac{(Num_R(i) + Num_G(i) + Num_B(i)) \times 2^7}{(Den_R(i) + Den_G(i) + Den_B(i))}$ **(5-4)**

Com 33 bits no numerador e 26 bits no denominador, o estabelecendo dos limites para $0,5 \leq \alpha_i \leq 8,00$, leva a um resultado com um número máximo de 10 bits.

5.3.3.1 Diagrama em Blocos

O detalhamento das operações descritas acima é apresentado no diagrama de blocos do processo para geração do parâmetro αi mostrado pela Figura 5-6.

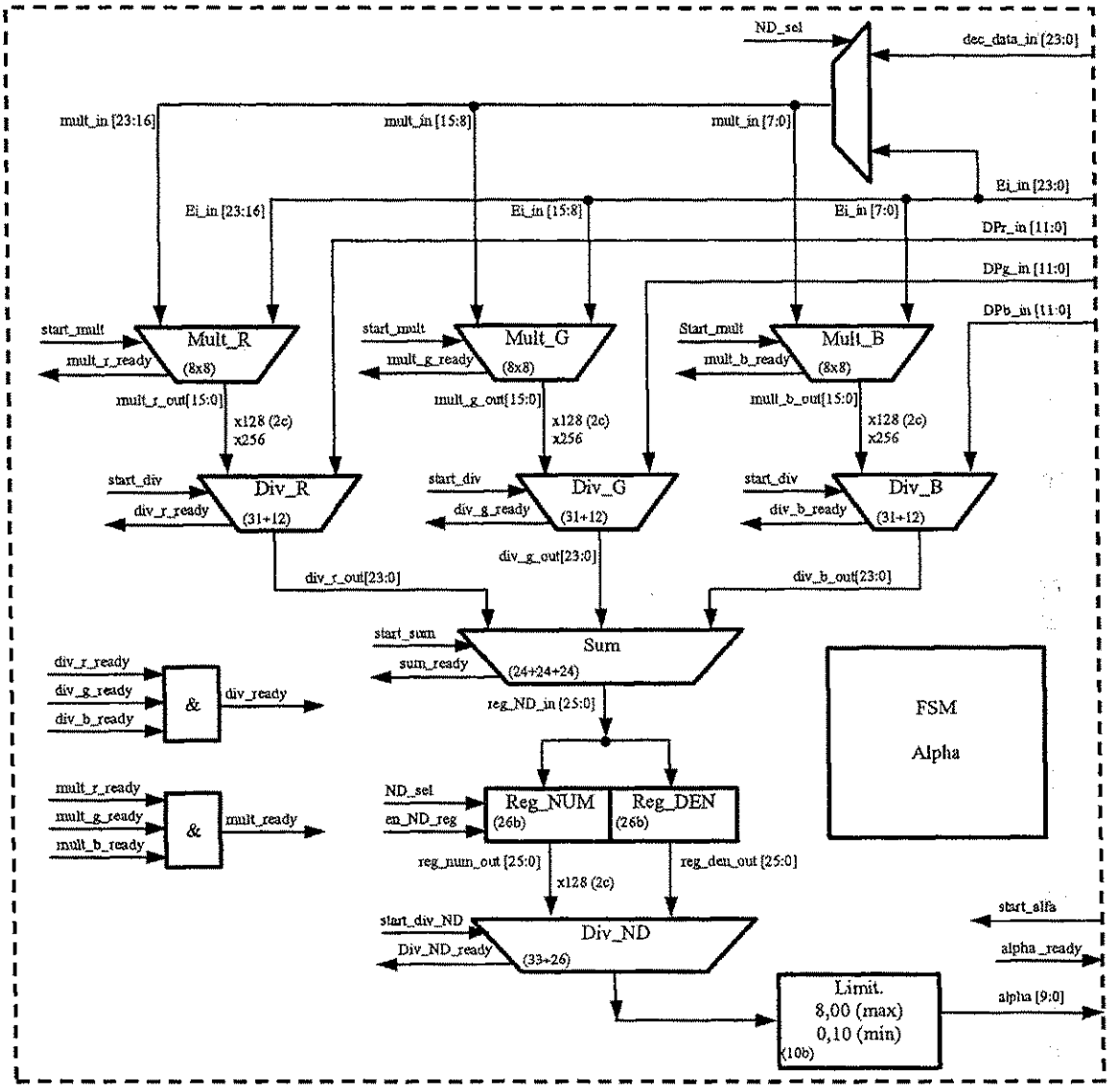
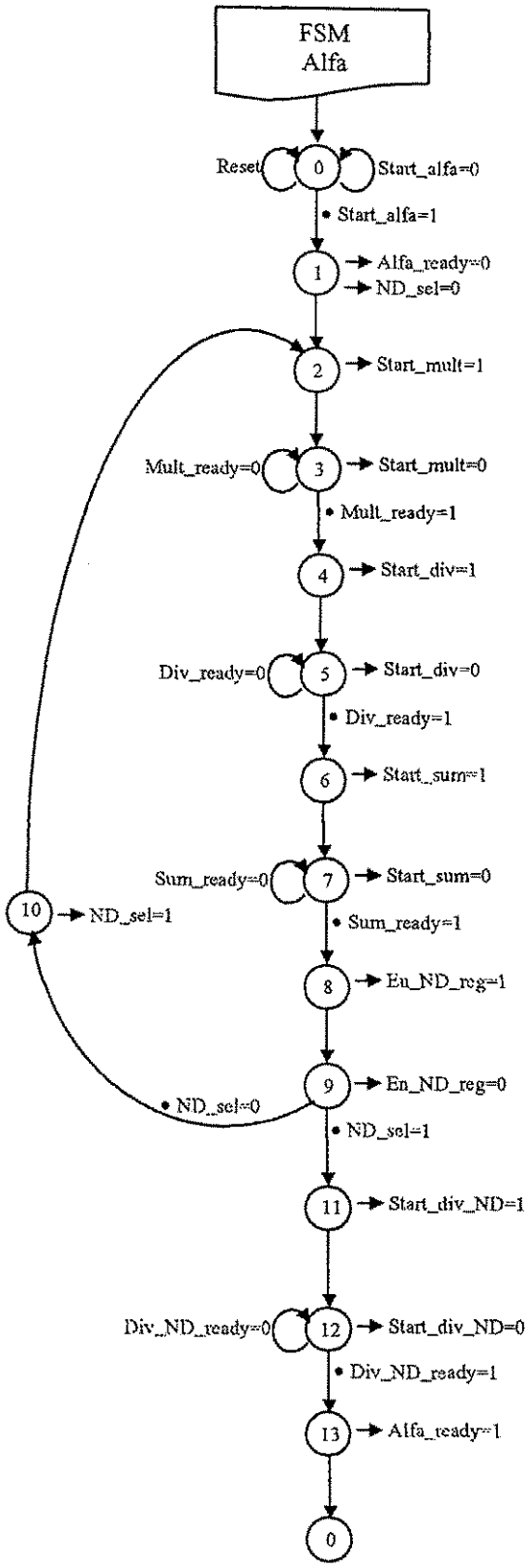


Figura 5-6 – Diagrama em blocos do processo de geração do parâmetro αi

5.3.3.2 Máquina de Estados



5.3.4 Geração do Parâmetro CD_i

O parâmetro CD_i que corresponde à distorção de cromaticidade de cada ponto em relação à imagem de referência é expresso por (3-9), reescrita a seguir de modo a facilitar o entendimento do texto:

$$CD_i = \sqrt{\left(\frac{I_{i(R)} - \alpha_i \mu_{i(R)}}{\sigma_{(R)}}\right)^2 + \left(\frac{I_{i(G)} - \alpha_i \mu_{i(G)}}{\sigma_{(G)}}\right)^2 + \left(\frac{I_{i(B)} - \alpha_i \mu_{i(B)}}{\sigma_{(B)}}\right)^2}$$

O processo de geração do parâmetro CD_i consiste em capturar o *pixel* corrente e efetuar a operação conforme descrito a seguir.

Pela análise da expressão podemos ver a necessidade da extração da raiz quadrada da soma de 3 parcelas, que representam os desvios de cromaticidade em R, G e B.

Façamos $CD_{i(C)} = \left(\frac{I_{i(C)} - \alpha_i \mu_{i(C)}}{\sigma_{(C)}}\right)^2$ ou ainda $CD_{i(C)} = \frac{\left(I_{i(C)} - \alpha_i \mu_{i(C)}\right)^2}{\sigma^2_{(C)}}$;

Assim teremos: $CD_i = \sqrt{CD_{i(R)} + CD_{i(G)} + CD_{i(B)}}$ **(5-5)**

A exemplo do cálculo de α_i apresentado anteriormente, devemos efetuar arranjos matemáticos na equação de forma a manter a precisão de duas casas decimais nos cálculos. A equação pede a execução de $\alpha_i . \mu_{i(C)}$, sendo α_i um número de 8 bits (já com precisão de 2 casas decimais) e $\mu_{i(C)}$ um número de 8 bits (sem casas decimais), a operação será realizada por um multiplicador de 10x8 bits, tendo como resultado um número de 18 bits com a precisão de 2 casas decimais.

Para a operação de $I_{i(C)} - \alpha_i \mu_{i(C)}$, devemos efetuar uma comparação antes da subtração de forma a manter o resultado como um número inteiro e positivo. O número $I_{i(C)}$ deve entrar nos cálculos multiplicado por 2^7 (multiplicação efetuada por deslocamento de barramento). A operação será realizada por um comparador de 18 bits, seguido de um subtrator de 18 bits com duas casas decimais, cujo resultado deve ser

limitado em 64,00, já que não há necessidade de avaliarmos *pixels* com distâncias superiores a este valor, eliminando assim a necessidade de *hardware* adicional. O resultado da limitação é um número de 13 bits que deve ser elevado ao quadrado, obtendo $\left(I_{i(C)} - \alpha_i \mu_{i(C)}\right)^2$ que terá como resultado um número de 26 bits, no entanto com 4 casas decimais, posteriormente o *hardware* efetua a eliminação das 2 casas excedentes.

A operação realizada para o cálculo de $\frac{\left(I_{i(C)} - \alpha_i \mu_{i(C)}\right)^2}{\sigma^2{}_{(C)}}$, exige um pequeno arranjo. Lembrando que $\sigma^2{}_{(C)} = \frac{\mathrm{Dp}_{(C)}}{2^8}$, teremos: $\frac{2^8 \times \left(I_{i(C)} - \alpha_i \mu_{i(C)}\right)^2}{Dp_{(C)}}$; no entanto como temos no numerador um número com 4 casas decimais, de acordo com o formato adotado neste trabalho devemos efetuar a divisão do resultado por 2^7, resultando em $\frac{2 \times \left(I_{i(C)} - \alpha_i \mu_{i(C)}\right)^2}{Dp_{(C)}}$, que é a operação efetivamente realizada pelo *hardware*, notemos esta arranjo permite a menor perda de precisão na operação de divisão mantendo-se o mesmo número de bits. A operação é realizada por um divisor que tem como numerador um número de 27 bits com 2 casas decimais e como denominador $\mathrm{Dp}_{(C)}$, que é um número de 12 bits inteiro (sem casas decimais) com valor mínimo limitado em 128. Portanto a saída do divisor apresenta um número de no máximo 20 bits com duas casas decimais correspondente a uma das 3 parcelas de $CD_{i(C)}$. Vale ressaltar que o cálculo das 3 parcelas $CD_{i(R)}$, $CD_{i(G)}$ e $CD_{i(B)}$, embora pudesse ser processado pela mesma máquina em 3 tempos, deve ser efetuado em paralelo por tratar-se de processamento executado em tempo real. A operação $CD_{i(R)} + CD_{i(G)} + CD_{i(B)}$, é efetuada por um somador de 3 entradas de 20 bits resultando em uma saída de 22 bits com 2 casas decimais.

A execução de $\sqrt{CD_{i(R)} + CD_{i(G)} + CD_{i(B)}}$ com a manutenção de 2 casas decimais requer que a extração da raiz seja realizada após a correção das casas decimais. A operação efetivamente realizada é $\sqrt{\left(CD_{i(R)} + CD_{i(G)} + CD_{i(B)}\right) \times 2^7}$ que resulta em CD_i, um número de 14 bits com 2 casas decimais.

5.3.4.1 Diagrama em Blocos

O detalhamento das operações para a obtenção de CD_i descritas acima, é apresentado no diagrama de blocos mostrado pela Figura 5-7.

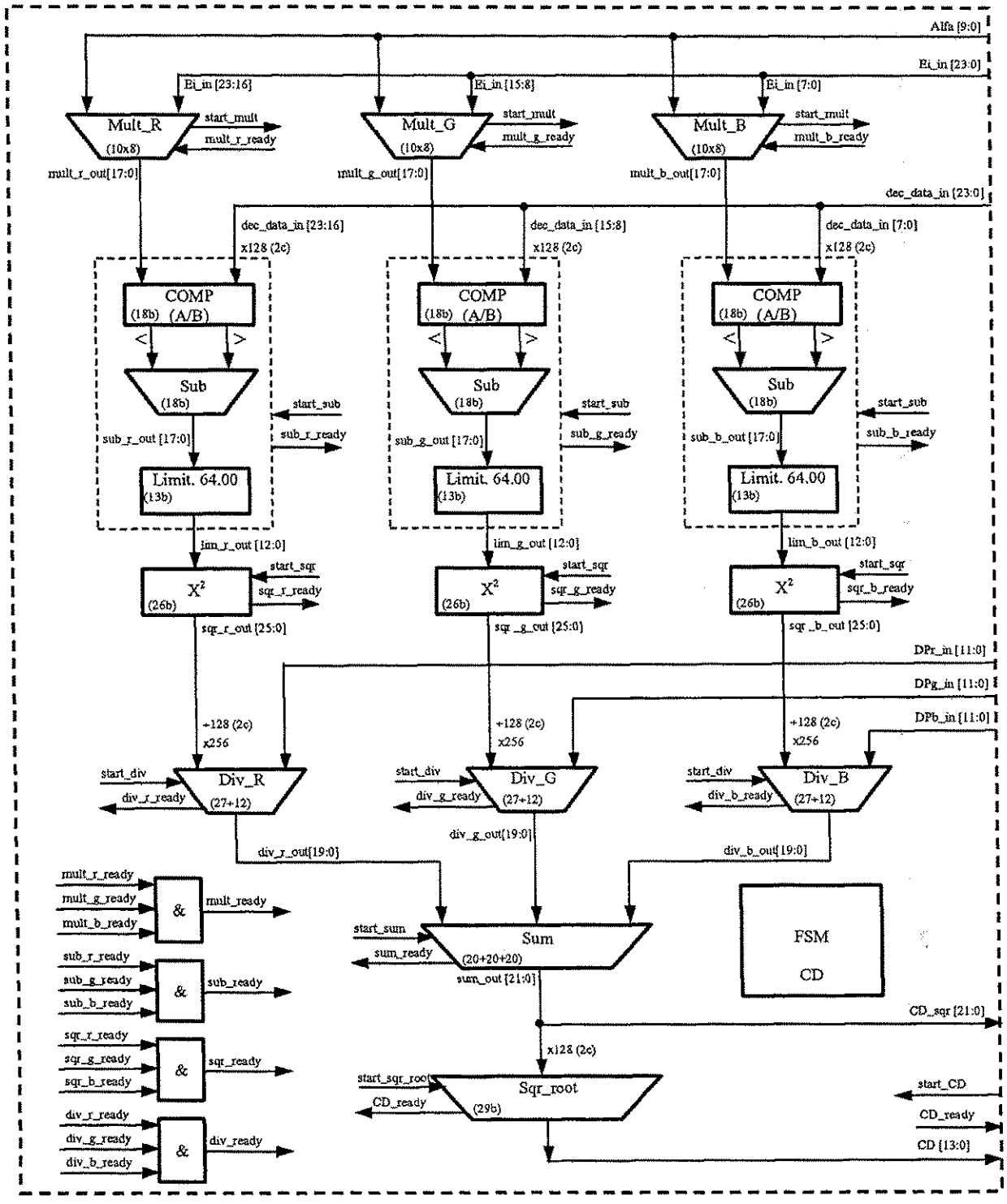
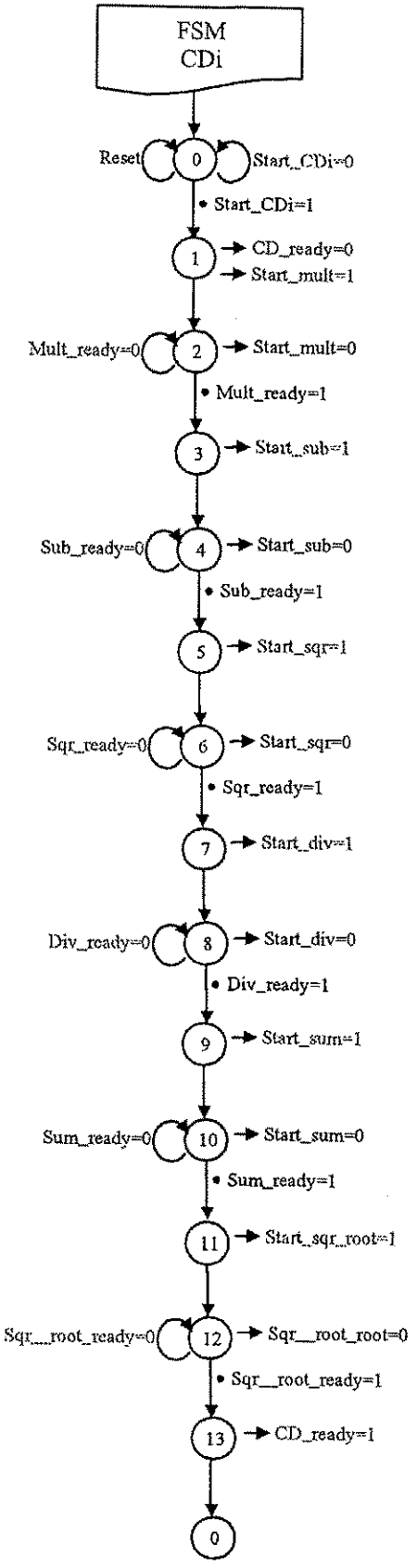


Figura 5-7 - Diagrama em blocos do processo de geração do parâmetro CD_i

5.3.4.2 Máquina de Estados



5.3.5 Geração dos Parâmetros A_i e B_i

O parâmetro a_i (ou A_i) representa a média da distribuição de α_i nos valores de 64 amostras medidas para cada um dos *pixels* (x,y). Em (3-10) pode-se verificar que o cálculo de A_i depende de α_i que é computado por (3-8), que por sua vez depende dos parâmetros E_i, S_i e do valor corrente I_i do *pixel* durante o processo de geração da imagem de referência. O parâmetro b_i (ou B_i) representa a média da distribuição de CD_i nos valores de 64 amostras medidas para cada um dos *pixels* (x,y). Em (3-11) pode-se verificar que o cálculo de B_i depende de CD_i que é computado por (3-9), que por sua vez depende dos parâmetros E_i, S_i e do valor corrente I_i do *pixel* durante o processo de geração da imagem de referência. Para facilitar a compreensão do texto as equações de A_i e B_i são reescritas a seguir.

$$a_i = RMS(\alpha_i - 1) = \sqrt{\frac{\sum_{k=0}^{N-1}(\alpha_{i(k)} - 1)^2}{N}}; \qquad b_i = RMS(CD_i) = \sqrt{\frac{\sum_{k=0}^{N-1}(CD_{i(k)})^2}{N}}$$

Antes de iniciarmos a descrição do processo de geração dos parâmetros A_i e B_i, devemos citar a necessidade de um pequeno arranjo no posicionamento dos parâmetros na memória de modo a reduzir o tempo de acesso às informações durante o processo final de seleção de *pixels*. Tal arranjo deve ser implementado nesta etapa, antes da geração dos parâmetros A_i e B_i. Os parâmetros devem ser agrupados na memória de modo a permitir sua leitura seqüencial (leitura em burst) que tem um tempo de acesso reduzido em relação à leitura aleatória (2 ciclos contra 4 ciclos).

A disposição final dos parâmetros na memória deve ser conforme mostrado pela Figura 5-8, na qual podemos ver a inserção de dois espaços entre os parâmetros E_i para a introdução de A_i e B_i. Dessa forma cada *pixel* terá seus parâmetros correspondentes, dispostos de forma seqüencial ma memória.

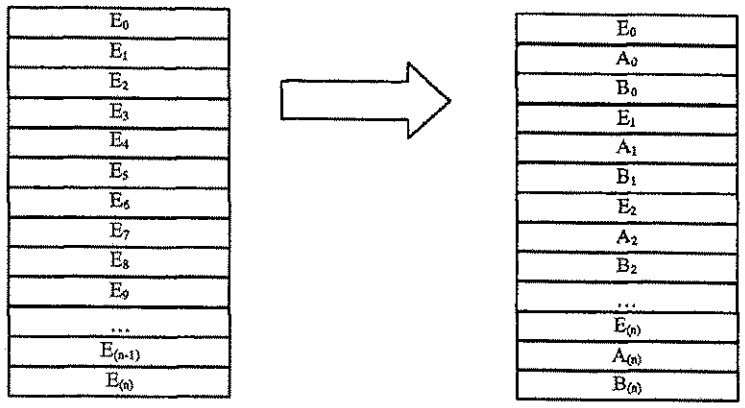


Figura 5-8 – Disposição dos parâmetros E_i, A_i e B_i na memória.

Considerando implementada a nova disposição dos dados na memória, e lembrando que os parâmetros E_i e S_i, nesta etapa já encontram-se calculados e armazenados, passemos para o processo de geração de A_i e B_i.

Genericamente o cálculo de A_i, consiste na análise da variação de brilho de cada *pixel* em um universo de 64 quadros. Cada *pixel* corrente adquirido no buffer do decodificador de vídeo, juntamente com o parâmetro E_i (calculado anteriormente) adquirido do cache, aplicados ao bloco gerador de α_i é processado pelo bloco gerador de A_i e armazenado na memória, conforme disposição apresentada pela Figura 5-8.

Da mesma forma o cálculo de B_i, consiste na análise da variação de brilho de cada *pixel* em um universo de 64 quadros. No entanto, para o cálculo de B_i deveríamos acumular os 64 valores de $(CD_i)^2$ (lembrar que CD_i é um número de 14 bits). Devido à dificuldade técnica da operação e acumulação de um número maior que 24 bits na memória o cálculo de B_i deve sofrer uma simplificação. Assim para a realização do cálculo de B_i sugerimos uma simplificação descrita pela equação apresentada a seguir:

$$b_i = RMS(CD_i) = \frac{\sum_{k=0}^{N-1} CD_{i(k)}}{N} \quad \text{(5-6)}$$

Esta aproximação evita a geração e acumulação de CD_i^2 e, considerando que todos os valores de CD_i são positivos e que B_i é apenas um fator de normalização que é aplicado igualmente a todos os *pixels*, tanto no processo de geração do histograma para a determinação dos nível de limiar na imagem de referência como durante o processo de seleção dos *pixels* do objeto, a variação introduzida pela aproximação sugerida não irá afetar o resultado final da extração do objeto. Voltemos então ao processo de geração deste parâmetro.

Para a geração de B_i utilizamos o valor do *pixel* corrente adquirido no buffer do decodificador de vídeo e o parâmetro E_i adquirido do cache, que são aplicados ao bloco gerador de CD_i, processados pelo bloco gerador de B_i e armazenado na memória, conforme disposição apresentada pela Figura 5-8.

Vale ressaltar que, para o processamento de A_i e B_i são necessários os mesmos arranjos matemáticos utilizados nos processamentos descritos anteriormente conforme pode ser verificado pelo diagrama de blocos e máquina de estados apresentadas a seguir.

5.3.5.1 Diagrama em Blocos

O detalhamento das operações para a obtenção de A_i e B_i descritas acima, bem como as interconexões necessárias com os blocos internos do *hardware* reconfigurável são apresentados no diagrama de blocos mostrado pela Figura 5-9.

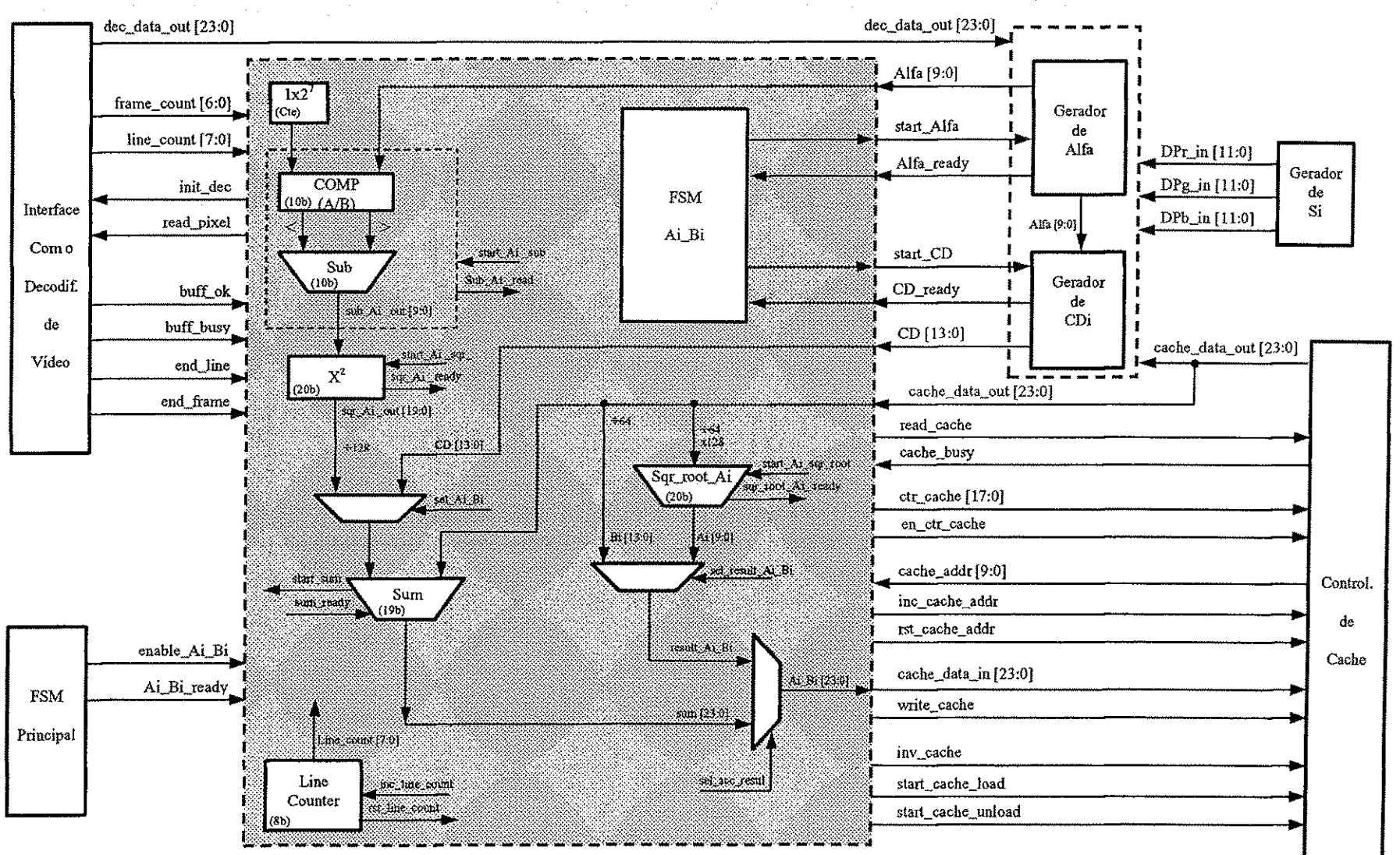
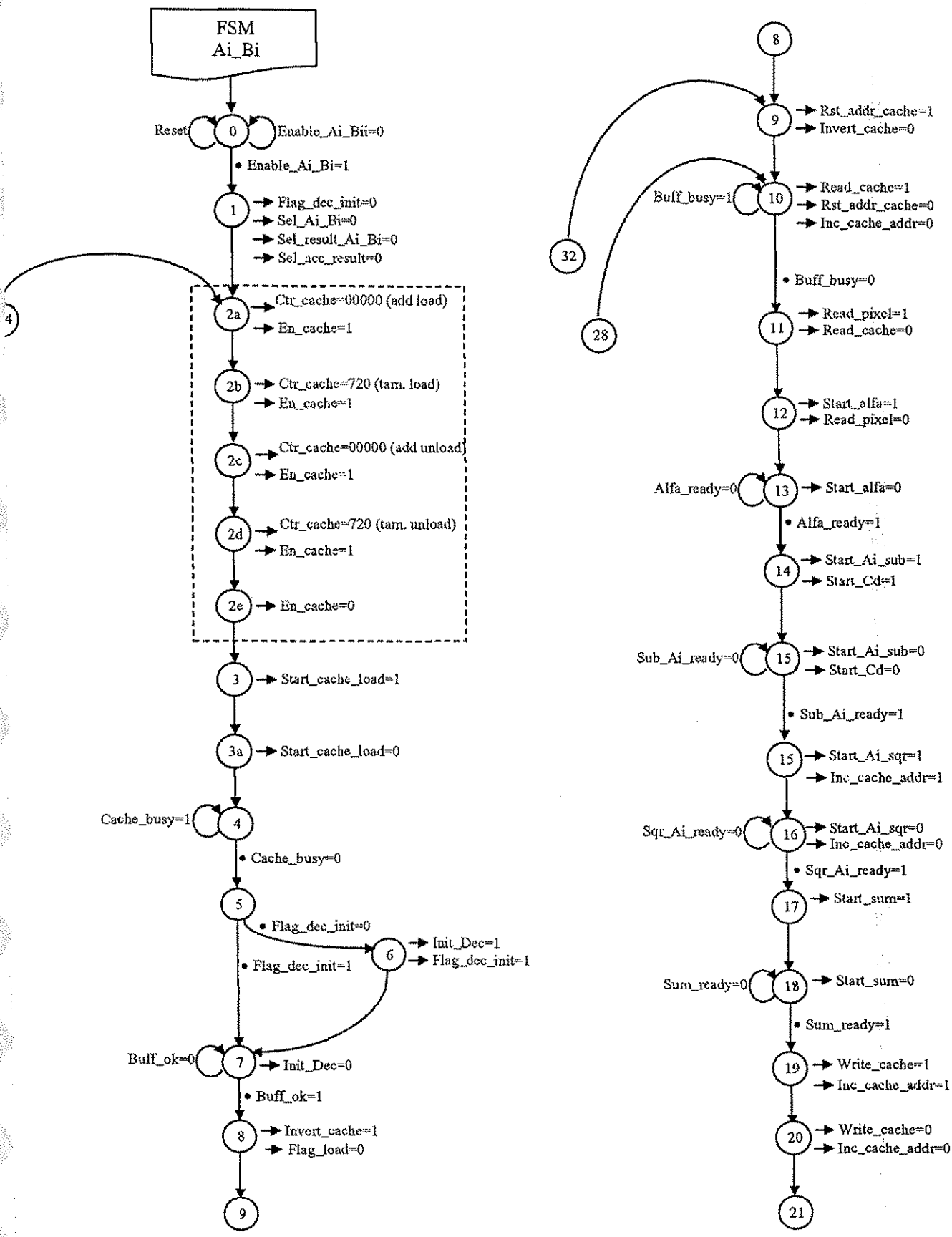
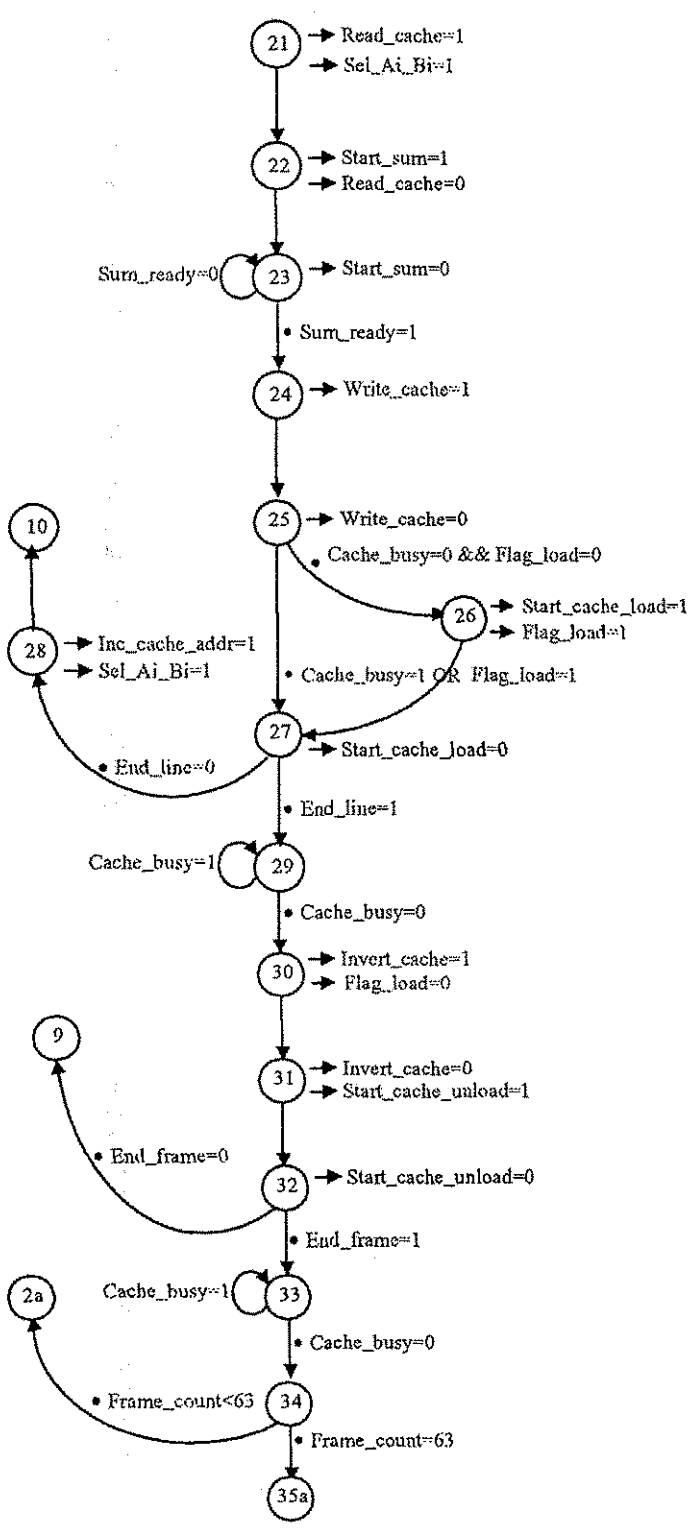
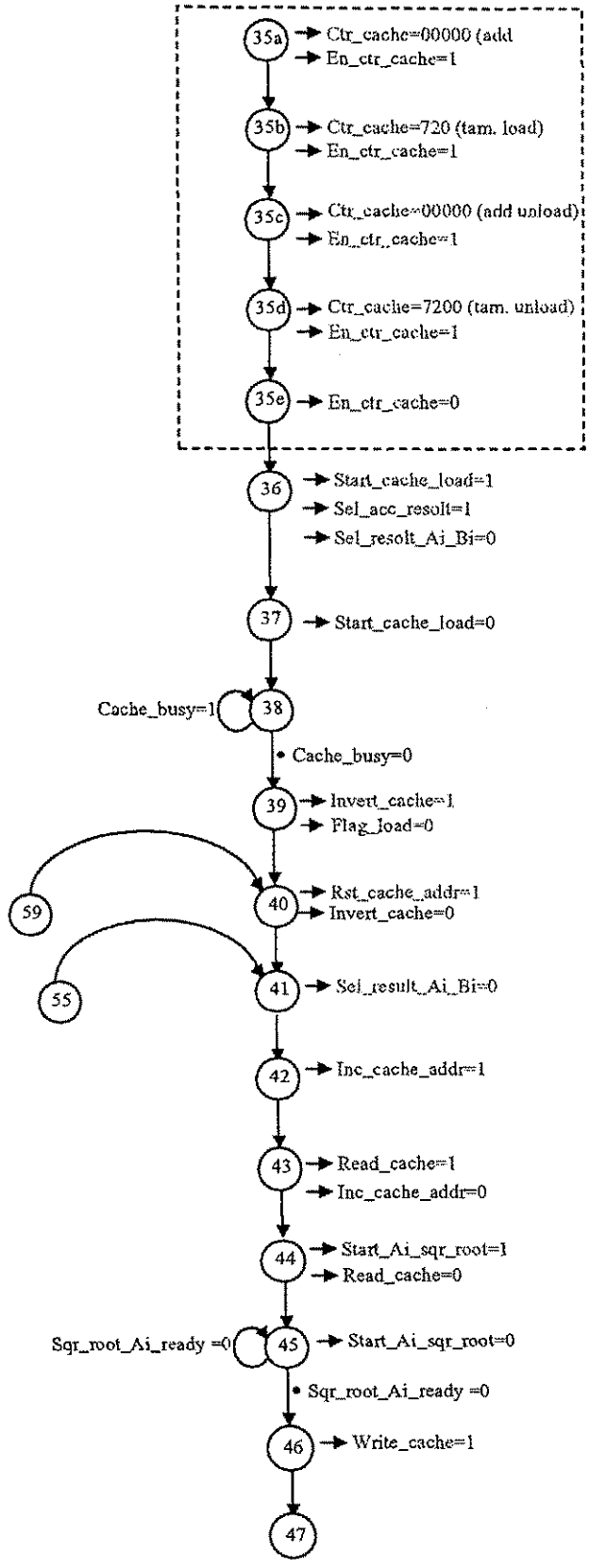


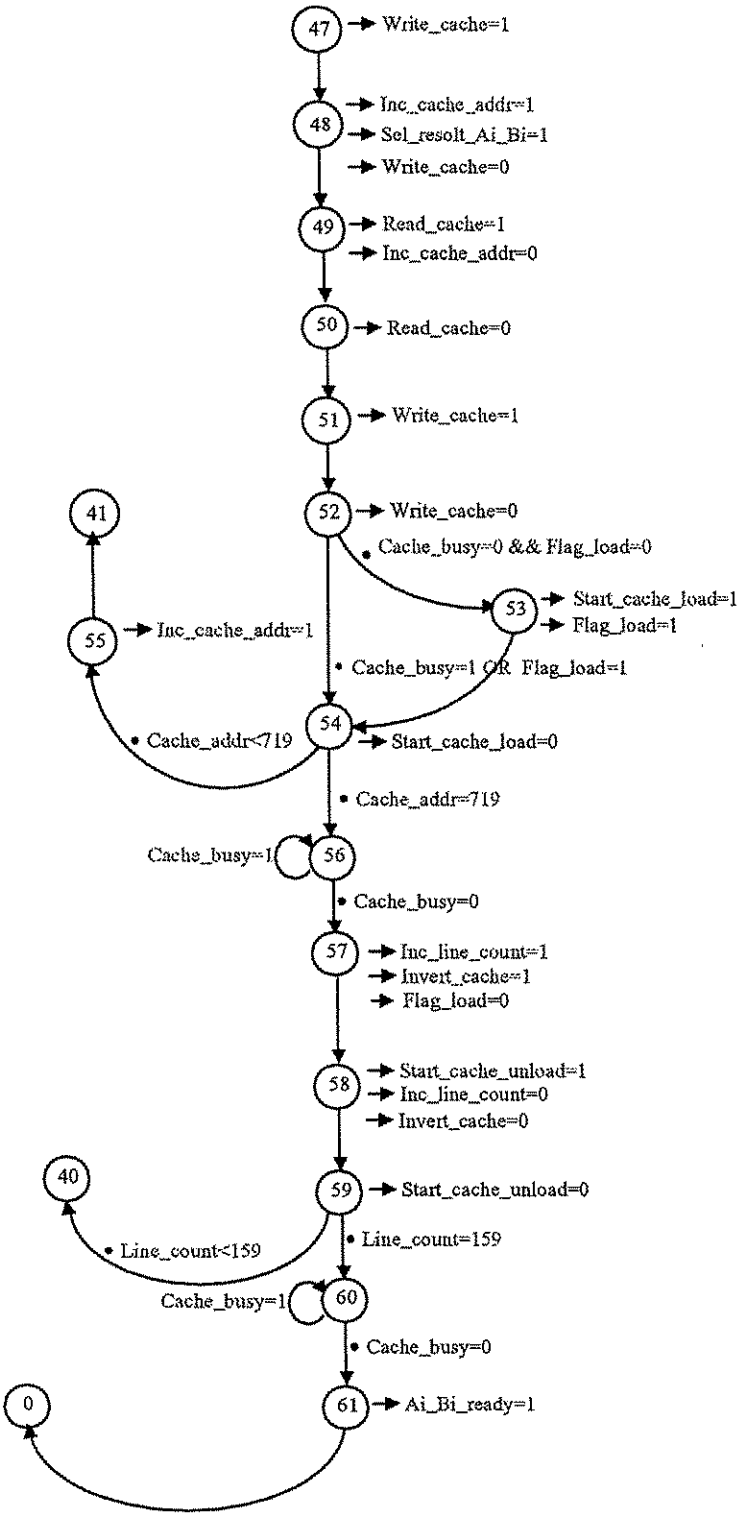
Figura 5-9 - Diagrama em blocos do Módulo Gerador de Ai e Bi

5.3.5.2 Máquina de Estados




21
Read_cache=1
Sel_Ai_Bi=1
22
Start_sum=1
Read_cache=0
Sum_ready=0
23
Start_sum=0
Sum_ready=1
24
Write_cache=1
10
25
Write_cache=0
Cache_busy=0 && Flag_load=0
26
Start_cache_load=1
Flag_load=1
28
Inc_cache_addr=1
Sel_Ai_Bi=1
Cache_busy=1 OR Flag_load=1
27
Start_cache_load=0
End_line=0
End_line=1
Cache_busy=1
29
Cache_busy=0
30
Invert_cache=1
Flag_load=0
9
31
Invert_cache=0
Start_cache_unload=1
End_frame=0
32
Start_cache_unload=0
End_frame=1
2a
Cache_busy=1
33
Cache_busy=0
Frame_count<63
34
Frame_count=63
35a


35a
Ctr_cache=00000 (add
En_ctr_cache=1
35b
Ctr_cache=720 (tam. load)
En_ctr_cache=1
35c
Ctr_cache=00000 (add unload)
En_ctr_cache=1
35d
Ctr_cache=7200 (tam. unload)
En_ctr_cache=1
35e
En_ctr_cache=0
36
Start_cache_load=1
Sel_acc_resolt=1
Sel_resolt_Ai_Bi=0
37
Start_cache_load=0
Cache_busy=1
38
Cache_busy=0
39
Invert_cache=1
Flag_load=0
59
40
Rst_cache_addr=1
Invert_cache=0
55
41
Sel_result_Ai_Bi=0
42
Inc_cache_addr=1
43
Read_cache=1
Inc_cache_addr=0
44
Start_Ai_sqr_root=1
Read_cache=0
Sqr_root_Ai_ready =0
45
Start_Ai_sqr_root=0
Sqr_root_Ai_ready =0
46
Write_cache=1
47


47
Write_cache=1
48
Inc_cache_addr=1
Sel_resolt_Ai_Bi=1
Write_cache=0
49
Read_cache=1
Inc_cache_addr=0
50
Read_cache=0
51
Write_cache=1
52
Write_cache=0
Cache_busy=0 && Flag_load=0
53
Start_cache_load=1
Flag_load=1
41
55
Inc_cache_addr=1
Cache_busy=1 OR Flag_load=1
54
Cache_addr<719
Start_cache_load=0
Cache_addr=719
Cache_busy=1
56
Cache_busy=0
57
Inc_line_count=1
Invert_cache=1
Flag_load=0
58
Start_cache_unload=1
Inc_line_count=0
Invert_cache=0
40
59
Start_cache_unload=0
Line_count<159
Line_count=159
Cache_busy=1
60
Cache_busy=0
0
61
Ai_Bi_ready=1

5.4 Definição dos Limiares

Uma vez finalizado o processo de geração dos parâmetros da imagem de referência, o passo seguinte é a seleção dos limiares. Esta operação também pode ser executada durante o processo de inicialização e consiste na escolha automática dos níveis adequados de modo a conseguirmos a melhor performance durante o processo de detecção do objeto.

5.4.1 Geração dos Histogramas

Como já mencionado anteriormente, as distorções de brilho e cromaticidade normalizadas, $\hat{\alpha}_i$ e $\hat{CD}_i$ respectivamente, não possuem uma distribuição gaussiana, portanto para determinação dos limiares devemos construir seus histogramas. Dessa forma efetuando uma nova aquisição de 32 quadros da imagem de referência calculamos $\hat{\alpha}_i$ e $\hat{CD}_i$ por (3-12) e (3-13) (reescritas a seguir) e computamos a freqüência de cada valor para a obtenção do nível correspondente à taxa de erro predeterminada.

$$\hat{\alpha}_i = \frac{\alpha_i - 1}{a_i}; \quad \text{e} \qquad \hat{CD}_i = \frac{CD_i}{b_i}$$

No entanto como estamos interessados apenas na extração do objeto e não na determinação das variações de iluminação, devemos nos ater à determinação do nível de limiar de $\hat{CD}_i$. Se considerarmos em nossa análise apenas os *pixels* com alguma probabilidade de pertencerem ao objeto, podemos estabelecer um valor máximo para $\hat{CD}_i = 16{,}00$, que com 2 casas decimais corresponde a um valor binário máximo de 2048 (um número de 11 bits), logo $\hat{CD}_i$ pode assumir valores entre 0 e 2047. Para a geração do histograma, o que temos a fazer é contar quantas vezes cada um destes 2048 valores ocorre em um conjunto de amostras.

5.4.1.1 Descrição do Processo

. Para a montagem do histograma vamos proceder como mostrado pela Figura 5-10

. Adotamos uma posição inicial de memória que tenha os últimos 11 bits de endereço todos zerados, esta posição corresponde ao contador de ocorrência do valor 0;

. Cada uma destas posições corresponde a um contador de ocorrências, relativo ao valor dos últimos 11 bits de seu endereço.

. Para registrar uma determinada ocorrência, concatenamos os 7 bits mais significativos do endereço inicial do histograma com o próprio valor da ocorrência (de 0 a 2047), endereçando (por indexação) e incrementando o contador desta ocorrência.

. Ao final deste processo temos 2048 posições de memória correspondentes a cada um dos valores possíveis para $C\hat{D}_i$, cada uma delas com o registro do número de ocorrências nos 32 quadros analisados.

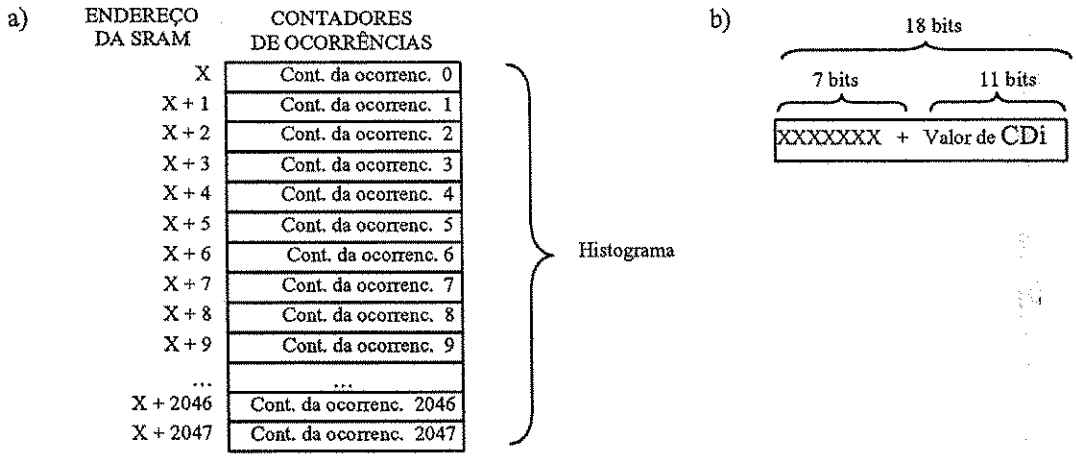


Figura 5-10 – (a) - Disposição do histograma de $C\hat{D}_i$ na memória; (b) – Composição do endereço;

5.4.2 Definição do Limiar de $C\hat{D}_i$ (τ_{CD})

Finalizada a geração do histograma de $C\hat{D}_i$, devemos agora determinar o nível de limiar para $C\hat{D}_i$, chamado de τ_{CD}. Para isso devemos estabelecer o valor admissível para erros de detecção, em nosso caso vamos adotar 0,1% , que em um universo de

240x160x32=1.228.800 amostras corresponde a 1228 ocorrências. Portanto o que temos a fazer é ler e acumular todas as ocorrências no histograma, decrementando o endereço desde a menor probabilidade (o valor 2047), até que tenhamos acumulado um valor superior a 1228. Os últimos 11 bits do endereço em que ocorrer o estouro (>1228) será o valor de τ_{CD}.

5.5 Extração do Objeto

Uma vez finalizado processo de obtenção dos parâmetros da imagem de referência, podemos iniciar a seqüência de extração de objeto que deve necessariamente ser executada em tempo real. Neste processo, para cada *pixel* devemos calcular os valores de α_i e CD_i conforme (3-8) e (3-9) respectivamente. Na seqüência calculamos $\hat{\alpha}_i$ e $\hat{CD}_i$ aplicando (3-12) e (3-13) respectivamente. Para finalizar o processo, aplicando (3-15) fazemos a seleção dos *pixel* que fazem parte do objeto. A demanda de memória durante esta fase é relativamente pequena se comparada com o processo de obtenção dos parâmetros da imagem de referência. Devemos ter na memória um quadro completo, cerca de 38.400 posições.

Neste capítulo foram descritos detalhadamente os processos a serem seguidos para a implementação do algoritmo escolhido, em *hardware* reconfigurável utilizando a plataforma GPIP-01. Nesta fase do trabalho, apenas o parâmetro E_i foi implementado em Verilog, ficando a implementação dos demais parâmetros e processos para a fase seguinte, complementar a este trabalho, a ser desenvolvida posteriormente.

6 Resultados Atingidos

Neste capítulo, são apresentados os resultados efetivamente atingidos ao final deste trabalho, cujo objetivo foi estabelecer um ponto de partida para o desenvolvimento de um sistema implementado em *hardware* com o propósito de efetuar a extração de objetos, em tempo real, de imagens com fundo estático e não homogêneo, capaz de suportar pequenas variações de iluminação tanto global como local.

O cerne deste sistema, por tratar-se de um circuito completamente digital, poderá culminar com a geração de um circuito integrado ASIC, destinado a aplicações no mercado de entretenimento, vigilância, dentre outras.

A implementação completa do sistema envolve 5 etapas:

1) Planejamento do Sistema;
2) Implementação e simulações do algoritmo em software;
3) Projeto e desenvolvimento da plataforma de *hardware*;
4) Implementação do sistema em *hardware* reconfigurável;
5) Geração do circuito integrado (ASIC).

As etapas 1, 2, 3 e a descrição do processo para a etapa 4 foram implementadas neste trabalho. Os resultados e conclusões de cada uma das etapas são descritos a seguir.

6.1 Planejamento do Sistema

O planejamento do sistema refere-se à concepção, pesquisa da literatura específica e escolha da técnica adotada para a implementação do projeto. Esta etapa foi completamente concluída neste trabalho, conforme descrito pelos capítulos 1 e 2 e os tópicos 3.1 a 3.5 do capítulo 3, de onde podemos concluir a aplicabilidade da técnica

proposta por Horprasert et al. [5], que descreve uma abordagem estatística para a extração de objetos de um fundo heterogêneo passível de implementação em tempo real desde que efetuadas as adequações necessárias a uma implementação em *hardware*.

6.2 Implementação e Simulações do Algoritmo em Software

Conforme descrito no tópico 3.6 do capítulo 3, foi elaborado um programa para a execução do algoritmo adotado, implementado no MATLAB (http://www.mathworks.com/) mas levando em consideração a limitação das casas decimais (2 casas), que é uma das maiores restrições impostas para nossa implementação em *hardware*. Em seguida foram realizadas simulações com seqüências de imagens aleatórias e "preparadas" para enfatizar as deficiências do processo.

Pelos resultados obtidos em simulação, mostrados no tópico 3.6, podemos concluir que a implementação do algoritmo é viável, embora seja aconselhável a implementação de filtros espaciais para a redução de erros de detecção que aparecem na forma de pontos aleatórios distribuídos pelo quadro.

6.3 Projeto e Desenvolvimento da Plataforma de *Hardware*

A plataforma desenvolvida para acomodar o sistema (GPIP-01), descrita no capítulo 4, mostrou-se satisfatória em termos de funcionalidade, ficando a questão relativa á sua capacidade de suportar a quantidade de portas equivalentes necessária para a implementação do sistema completo a ser respondida após a complementação da etapa seguinte. Os periféricos desenvolvidos em *hardware* reconfigurável para o interfaceamento com o *hardware* fixo e dotando a plataforma das ferramentas necessárias para a manipulação de dados na memória, também atendem aos requisitos do sistema.

A seguir é apresentada a Tabela 1 que mostra a taxa de utilização do dispositivo reconfigurável (FPGA) e a freqüência máxima conseguida para o relógio, após realizado o roteamento de cada um dos blocos implementados até o presente momento.

	Nome do Módulo	Limite de *Clock*	Taxa de Utilização de Elementos Lógicos
1	Interface c/ o Codificador de Vídeo	94,55 MHz	1%
2	Interface c/ o Decodificador de Vídeo	60,32 MHz	3%
3	Controlador de Cache	45,36 MHz	7%
4	Interface de Acesso à Memória	126,0 MHz	<1%
5	Gerador do Parâmetro Ei	57,58 MHz	2%

Tabela 1

A Tabela 1 mostra que os requisitos de velocidade estabelecidos para o sistema foram parcialmente atingidos, com exceção do item 3 da tabela, correspondente à implementação do controlador de cache que atingiu uma freqüência de relógio inferior a 50 MHz, requerido para o sistema. Portanto neste ponto verificamos a necessidade da otimização do código a fim de corrigir este problema. Uma vez que o *hardware* reconfigurável implementado faz parte da plataforma mínima, necessária para a implementação do sistema completo, podemos buscar também a otimização das máquinas de estados, pelo compartilhamento ou reaproveitamento de *hardware* entre blocos diferentes, visando a redução da taxa de utilização da FPGA, que permitirá maior liberdade para a implementação do sistema completo.

Outra alternativa que pode vir a ser adotada para a maximização dos recursos de *hardware* é a reprogramação da FPGA entre etapas do processamento, desde que preservados os dados da SRAM.

6.4 Implementação do Sistema em *Hardware* Reconfigurável

A etapa de implementação do sistema em *hardware* reconfigurável foi somente iniciada, foram definidos e descritos os processos, elaborados os diagramas em blocos e máquinas de estado, bem como as operações matemáticas necessárias para a realização dos cálculos dos parâmetros em ponto fixo com 2 casas decimais, conforme descrito no capítulo 5.

8 Trabalhos Futuros

Como já mencionado, o objetivo deste trabalho foi apenas estabelecer um ponto de partida para o desenvolvimento de um circuito integrado ASIC. Por este motivo a lista de sugestões para trabalhos futuros é bastante extensa, e será dividida em 2 tipos de atividades: trabalhos referentes ao aperfeiçoamento do que já foi realizado nesta fase e trabalhos complementares a serem realizados.

8.1 Aperfeiçoamento do Trabalho Realizado

O presente trabalho apresentou alguns pontos que merecem atenção e devem ser aperfeiçoados, portanto sugerimos como trabalhos a serem desenvolvidos:

. Busca de métodos para melhorar o desempenho da técnica proposta no que se refere a pontos escuros e pontos saturados da imagem;

. Implementação de filtros espaciais para a redução de erros de detecção aleatórios;

. Melhoria na implementação física do leiaute placa de memória permitindo aumento da velocidade de acesso, que pode chegar a 66 MHz contra somente 50 MHz conseguidos com a implementação atual.

8.2 Trabalhos complementares

Os trabalhos complementares são relativos às etapas restantes para a conclusão do projeto completo e portanto ainda não realizadas. Sugerimos que as etapas seguintes sejam realizadas em duas fases.

Fase I I - Implementação completa do sistema em *hardware* reconfigurável

. Otimização dos processos propostos através do compartilhamento de *hardware*;

. Pesquisa de algoritmos otimizados para as operações matemáticas propostas;

. Implementação de cada um dos diagramas em blocos e máquinas de estado;

. Simulação das descrições de *hardware* utilizando as imagens de referência e os resultados obtidos nas simulações com o MATLAB, para validação dos resultados obtidos;

. Síntese do sistema completo na plataforma GPIP-01;

. Validação final do sistema;

Fase III – Geração do Circuito Integrado (ASIC)

. Síntese física do leiaute;

. Simulação pós-síntese;

. Implementação dos processos envolvidos na testabilidade do chip;

. Processo de envio para fabricação;

. Modificação do projeto da plataforma para substituição do *hardware* reconfigurável pelo ASIC;

. Verificação da funcionalidade do ASIC.

9 Referências Bibliográficas

[1] Steven W. Smith, " *The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Publishing* ", USA, 1997, ISBN 0-9660176-3-3.

[2] Gonzalez, Rafel C., Woods, Richard C., " *Digital Image Processing* ", Addison-Wesley 1992.

[3] A. Utsumi, H. mori, J. Ohya, and M. Yachida " *Multiple-Human Tracking using Multiple Cameras* ", Proceedings of the Third International Conference on Automatic Face and Gesture Recognition (FG'98), April 1998.

[4] L. Tonietto. " *Análise de algoritmos para Chroma-key.* " Disponível em: http://www.inf.unisinos.br/~marcelow/ensino/tc/ckey/ckey.html. Acesso em: 15 de fevereiro de 2002.

[5] José M. Martínez (UPM-GTI, ES). " *Coding of Moving Pictures and Audio – MPEG-7 Overview* ". Disponível em: http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htmMPEG_7. Acesso em 10 de dezembro de 2002.

[6] T. Horprasert, D. Harwood, L.S. Davis. " *A statistical approach for Real Time Robust background subtraction.* " in IEEE ICCV'99 FRAME-RATE WORKSHOP. 1999.

[7] Y. Ivanov, A. Robick, J. Liu. " *Fast lighting independent background subtraction.*" International Journal of Computer Vision, 37(2), pp.199-209, June 2000.

[8] H. Zen, T. Hasegama, S. Osawa " *Moving object detection from MPEG coded picture.* " In IEEE International confference on Image Processing, ICIP'99, Kobe, October 1999.

[9] A. Yoneyama, Y Nakajima, H. Yanagihara, M. Sugano " *Moving object detection and Identification from MPEG Coded Data.* " International Conference on Image Processing, vol.2, pp. 934-938, Oct. 1999.

[10] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland " *Pfinder: Real-time tracking of the human body.* " IEEE Trans. Pattern Analysis and Machine Intelligence, 19(7), pp.780-785, July 1997.

[11] A. Elgammal, D. Harwood, L. Davis " *Non-parametric model for background subtraction.* " in Proc. 6th European Conference on Computer Vision,(Dublin,Ireland), 2000.