Resumo

MINHONI, D. C. R.(2006). Detecção de movimento de objetos em tempo real utilizando dispositivos de lógica programável complexa. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

Um sistema que realiza a detecção de movimento procura, numa sequência de imagens, sinais que confirmem a existência de movimentação no ambiente monitorado. Uma vez realizada a detecção do movimento, pode-se realizar o rastreamento (“*tracking*”) do objeto na cena em questão.

A detecção e o rastreamento de objetos, em tempo real, são técnicas que estão despertando grande interesse por parte de pesquisadores e empresas pois, estas técnicas, podem ser utilizadas em diversas áreas que se estendem desde a engenharia e computação até áreas como a geologia e medicina. Sendo assim, seguindo-se a idéia básica de detecção e rastreamento, encontram-se diversas aplicações para estas técnicas como: sistemas de vigilância, análise de movimentos humanos, sistemas de detecção e rastreamento de pedestres ou veículos, dentre outras.

Neste trabalho é mostrado um sistema que foi desenvolvido para armazenamento de imagens em tons de cinza de uma sequência de vídeo e um posterior processamento dessas imagens para detecção de características que indiquem movimento. O processamento se resume em integrar o sinal de vídeo, que está armazenado nas memórias, nas direções

horizontal e vertical gerando os histogramas de intensidade horizontal e vertical. Comparando os histogramas de quadros diferentes da sequência de vídeo será possível detectar a presença de movimento e a região da imagem onde este ocorreu. Devido à necessidade de um processamento rápido das imagens e no interesse de produzir um sistema dedicado com hardware reduzido, utilizou-se de dispositivos de lógica programável complexa (CPLDs).

PALAVRAS CHAVE: detecção de movimento, rastreamento, integração, CPLDs.

Abstract

MINHONI, D. C. R. (2006). Real time detection of moving objects using programmable logic devices. MSc dissertation – São Carlos Engineering School, University of São Paulo.

A system that performs movement detection in a sequence of images looks for signs that confirm the occurrence of the movement in the controlled environment. Once the movement of the object is detected it is possible to perform the tracking of the object.

Real time object detection and tracking techniques are of great interests to researchers and industries because these techniques can be used in several areas going from engineering and computing to geology and medicine. There is a wide field of applications of detection and tracking techniques, such as: surveillance systems, human movement analysis, pedestrians or vehicle detection.

This work presents an implementation able to store a gray level image from a video sequence and from these images detect in real time a object movement in the scene. The detection will be performed integrating an image from the video sequence in the horizontal and vertical directions in order to obtain the intensities histograms in these directions. Comparing the histograms with those of a different frame of the video sequence it will be possible to detect the presence of movement and locate where in the image the movement

occurs. Due to real time digital image processing requirements and in order to produce a reduce dedicated hardware, Complex Programmable Logic Devices (CPLDs) were used.

KEY WORDS: movement detection, tracking, integration, Complex Programmable Logic Devices (CPLDs).

Índice

Resumo.....	i
Abstract.....	iii
Índice.....	v
Lista de Figuras.....	vii
Lista de Tabelas	x
Lista de Abreviaturas e Siglas	xi
1 Introdução	1
1.1 Motivação.....	1
1.2 Objetivo	10
1.3 Organização.....	11
2 Dispositivos Lógicos Programáveis (PLDs)	12
2.1 A Evolução dos Dispositivos Lógicos Programáveis.....	17
2.1.1 Arquiteturas Básicas	17
2.1.2 PROM (Programmable Read-Only Memory)	19
2.1.3 PLA (<i>Programmable Logic Array</i>)	21
2.1.4 PAL (<i>Programmable Array Logic</i>)	22
2.2 Elementos de Programação	24
2.2.1 Fusível (<i>Fuse</i>)	25
2.2.2 Antifusível (Antifuse).....	26
2.2.3 Células SRAM (<i>SRAM Cells</i>).....	27
2.2.4 EPROM e EEPROM	28
2.3 CPLDs (Complex PLDs).....	34
2.3.1 CPLDs da Altera.....	36
2.3.2 A Tecnologia FPGA	38
2.3.3 Arquiteturas das FPGAs	41
2.3.4 Arquitetura Cyclone	43
2.3.5 Aplicações de FPGAs	47
3 Captação, Digitalização e Exibição de Imagens Monocromáticas.....	48
3.1 Considerações Iniciais.....	48
3.1.1 O Sinal de Vídeo Composto	48
3.2 Circuito de Captação, Digitalização, e Exibição de Imagens Monocromáticas... 56	
3.2.1 Captação, Digitalização e Exibição de Imagens Monocromáticas.....	57
3.2.1.1 Separador de Sincronismo	58
3.2.1.2 Conversor A/D.....	59
3.2.1.3 Buffer.....	60
3.2.1.4 Conversor D/A.....	61
3.2.1.5 Saída de Vídeo.....	62

4	Armazenamento de Imagens Monocromáticas	63
4.1	Controle para Armazenamento de Imagens Monocromáticas	63
4.1.1	Memórias	64
4.1.2	Unidade de Controle	65
4.1.2.1	Unidade de Exclusão de Pulsos de Sincronismo	69
4.1.2.2	Unidade de Controle Interna.....	70
4.1.2.3	Unidade de Controle das Memórias	71
4.2	Considerações Finais.....	73
5	Integração do Sinal de Vídeo	75
5.1	Considerações Iniciais.....	75
5.1.1	Integração do Sinal de Vídeo na Direção Horizontal	78
5.1.2	Integração do Sinal de Vídeo na Direção Vertical	81
6	Testes e Resultados	88
6.1	Considerações Iniciais.....	88
6.1.1	Materiais	89
6.1.2	Simulação da unidade de controle.....	90
6.1.3	Simulação e Teste do somador de <i>pixels</i> horizontais	91
6.1.4	Simulação e teste do somador de <i>pixels</i> verticais.....	100
6.2	Considerações Finais.....	106
7	Conclusões	108
7.1	Considerações Iniciais.....	108
7.2	Análise dos Resultados.....	110
7.3	Propostas para Trabalhos Futuros	112
Apêndice A.....		117
Apêndice B.....		118
Apêndice C.....		119
Apêndice D.....		120
Apêndice E.....		121
Apêndice F		123
Apêndice G		125
Apêndice H		127

Lista de Figuras

Figura 2-1 Metodologias de projeto	13
Figura 2-2 Dispositivos de lógica programável (“PLDs”) comerciais	16
Figura 2-3 Representação PLDs	18
Figura 2-4 Circuito simplificado de uma PROM	20
Figura 2-5 Circuito simplificado de um PLA	22
Figura 2-6 Circuito simplificado de um PAL	23
Figura 2-7 Foto antifusível. (a) Antes da programação. (b) Após programação. (c) Vista de cima, um antifusível é aproximadamente do mesmo tamanho que um contato.	27
Figura 2-8 Tecnologia de programação SRAM	28
Figura 2-9 (a) Seção de corte e (b) Símbolo do circuito do transistor de porta flutuante usado como célula	29
Figura 2-10 Ilustração do deslocamento na característica $i_D - v_{GS}$ de um transistor de porta flutuante como resultado da programação.....	30
Figura 2-11 O transistor de porta flutuante durante a programação.....	32
Figura 2-12 Estrutura de um CPLD.....	35
Figura 2-13 Arquitetura de um LAB	37
Figura 2-14 Macro célula MAX7000	38
Figura 2-15 Arquiteturas de FPGAs	42
Figura 2-16 Diagrama em blocos da arquitetura Cyclone EP1C12.....	45
Figura 2-17 Estrutura de um LAB da Cyclone.....	46
Figura 3-1 Processo de varredura entrelaçada. De A a B tem-se o traço horizontal. De B a C tem-se o retraço horizontal. De D a E e de F a A tem-se o retraço vertical, enquanto que o traço vertical corresponde à ação de deslocamento do feixe de baixo para cima.	50
Figura 3-2 Formato de uma linha de sinal de vídeo composto.....	54
Figura 3-3 Formato do sinal padrão de vídeo composto	56
Figura 3-4 Diagrama em blocos do circuito de captação, digitalização e exibição de imagens monocromáticas.	57
Figura 3-5 Sinais extraídos do separador de sincronismo. Em (a), tem-se o sinal de entrada (vídeo composto padrão). Em (b), tem-se o sinal de sincronismo composto (cs). Em (c), tem-se o sinal de sincronismo vertical (vs). Em (d), tem-se o sinal de tipo de campo (par/ímpar) que será baixo quando o campo for par e alto quando o campo for ímpar. E em (e), tem-se o sinal de pódio posterior (bp).	59
Figura 3-6 Diagrama de tempos do conversor A/D TLC5540. O sinal é amostrado na borda de descida do clock e os dados ficam disponíveis a partir da quarta borda de subida do clock	60
Figura 3-7 Diagrama de tempo do conversor D/A TLC5602.....	61
Figura 4-1 Módulo de armazenamento e controle de imagens monocromáticas	63
Figura 4-2 Diagrama de tempo da memória. (a) ciclo de leitura, (b) ciclo de escrita	65
Figura 4-3 Unidade de controle	67
Figura 4-4 Diagrama esquemático da unidade de controle	68
Figura 4-5 Bloco lógico da unidade de exclusão de pulsos equalizadores e serrilhados	69

Figura 4-6 Diagrama esquemático da unidade de exclusão de pulsos equalizadores e serrilhados.....	70
Figura 4-7 Bloco lógico da unidade de controle interna	71
Figura 4-8 Bloco lógico da unidade de controle das memórias	72
Figura 4-9 Diagrama esquemático da unidade de controle das memórias	73
Figura 4-10 Diagrama em blocos dos dois módulos propostos.....	74
Figura 5-1 Quadros sucessivos de imagem com o deslocamento de um objeto. À direita de cada quadro aparece o valor médio de cada linha	76
Figura 5-2 Quadros sucessivos de imagem com o deslocamento de um objeto. Abaixo de cada quadros aparece o valor médio do sinal de linha	77
Figura 5-3 Diagrama em bloco da etapa de integração do sinal de vídeo	77
Figura 5-4 Bloco lógico do somador de <i>pixels</i> horizontais	78
Figura 5-5 Diagrama esquemático do somador de <i>pixels</i> horizontais	79
Figura 5-6 Matriz 4x4, com os valores dos bits em binário	82
Figura 5-7 Leitura seqüencial dos bits da matriz 4x4.....	82
Figura 5-8 Matriz 4x4 transposta	83
Figura 5-9 Leitura seqüência dos bits da matriz 4x4 transposta	83
Figura 5-10 Contadores para a transposição dos endereços das memórias.....	84
Figura 5-11 Diagrama esquemático da unidade de controle das memórias com a etapa para realizar a transposição dos endereços das memórias.....	85
Figura 5-12 Bloco lógico do somador de <i>pixels</i> verticais	86
Figura 5-13 Diagrama esquemático do somador de <i>pixels</i> verticais	87
Figura 6-1 Hardware implementado para obtenção dos dados.....	89
Figura 6-2 Simulação da Unidade de Controle	90
Figura 6-3 Simulação do somador de <i>pixels</i> horizontais com dezessete bits de resolução ..	92
Figura 6-4 Simulação do somador de <i>pixels</i> horizontais. Os oito bits mais significativos estão representados no canal S	93
Figura 6-5 Simulação do somador de <i>pixels</i> horizontais com os oito bits mais significativos	93
Figura 6-6 Clock do circuito (CK1), dados obtidos da memória (A0) e dados obtidos na saída do "ff_horiz1" (A1)	94
Figura 6-7 Imagem filmada para obtenção do histograma	95
Figura 6-8 Dados da memória (A0), dados do "ff_horiz1" e histograma da imagem.....	95
Figura 6-9 Imagem com o objeto deslocado na cena	96
Figura 6-10 Dados da memória (A0), dados do "ff_horiz1" (A1) e histograma da imagem com o objeto deslocado na cena	96
Figura 6-11 Clock do circuito (CK1), dados das memórias (A0), dados na saída do "ff_horiz1" (A1) e oito bits mais significativos do resultado da saída do acumulador horizontal (A2)	97
Figura 6-12 Dados amostrados em forma de ondas e lista	98
Figura 6-13 Dados amostrados. Canal A2: saída do acumulador horizontal. Canal A3: saída do registrador, com o valor da soma truncada no final de uma linha ativa de vídeo ...	99
Figura 6-14 Simulação dos endereços transpostos	100
Figura 6-15 Simulação dos endereços transpostos	101
Figura 6-16 Simulação do somador de <i>pixels</i> na direção vertical com dezessete bits de resolução.....	101

Figura 6-17 Simulação do somador de <i>pixels</i> verticais mostrando os oito bits mais significativos na saída.....	102
Figura 6-18 Dados lidos diretamente das memórias (A0) e "carry out" (CK1)	103
Figura 6-19 Leitura de dados das memórias (A0) e leitura de dados do "ff_vert1" (A1) ..	104
Figura 6-20 Dados obtido da uma imagem de um objeto escuro em um fundo claro.....	105
Figura 6-21 Dados obtidos de um objeto escuro deslocado na cena	106

Lista de Tabelas

Tabela 2-1 Tecnologias de programação.....	33
Tabela 2-2 Comparação de FPGAs com relação à capacidade, arquitetura, célula básica e métodos de programação.	42
Tabela 2-3 Características das arquiteturas da família Cyclone.....	44
Tabela 2-4 Recursos dos dispositivos da família Cyclone	45

Lista de Abreviaturas e Siglas

AHDL – Altera Hardware Description Language

ASIC – Application Specific Integrated Circuits

CAD – Computer Aided Design

CCD – Charge-Coupled Device

CLB – Complex Logic Block

CMOS – Complementary Metal Oxide Semiconductor

CPLD – Complex Programmable Logic Device

DDR – Double Data Rating

DSL – Digital Subscriber Line

DSP – Digital Signal Processing

ECL – Emitter Coupled Logic

EECMOS – Electrically Erasable CMOS

EEPROM – Electrically Erasable PROM

EPLD – Erasable Programmable Logic Device

EPROM – Erasable Programmable Read Only Memory

FPGA – Field Programmable Gate Array

GAL – Generic Array Logic

HCPLD – High Capacity Programmable Logic Device

HDL – Hardware Description Language

IC – Integrated Circuit

IOB – Input Output Block

IOE – Input Output Element

ISP – In System Programmability

JTAG – Joint Test Action Group

LAB – Logic Array Block

LE – Logic Element

MOS – Metal Oxide Semiconductor

MOSFET – Metal Oxide Semiconductor Field Effect Transistor

MPGA – Mask Programmable Gate Array

MSPS – Mega Samples Per Second

NTSC – National Television Systems Committee

OTP – One Time Programmable

PAL – Programmable Array Logic

PCI – Personal Computer Interface

PIA – Programmable Interconnection Array

PLA – Programmable Logic Array

PLD – Programmable Logic Device

PLE – Programmable Logic Element

PLL – Phase-Locked Loop

PROM – Programmable Read Only Memory

RAM – Random Access Memory

ROM – Read Only Memory

SMD – Surface Mounted Device

SRAM – Static Random Access Memory

TTL – Transistor-Transistor Logic

UVC MOS – Ultraviolet Complementary Metal Oxide Semiconductor

VHDL – Very high speed integrated circuit Hardware Description Language

VHF – Very High Frequency

VLSI – Very Large Scale Integration

Capítulo 1

1 Introdução

1.1 Motivação

A capacidade de detectar movimento constitui parte fundamental do sistema de visão de animais. O desenvolvimento de sistemas, capazes de fazer com que máquinas possuam capacidade semelhante, vem recebendo maior atenção a cada dia (Krüger, S.; Calway, A.; 1996).

Dentre as várias linhas de pesquisa nesta área, podem-se citar aquelas que utilizam seqüências de imagens para detectar e rastrear um ou vários objetos na cena. Existem várias maneiras de se realizar esse rastreamento (“*tracking*”). Algumas possuem informações prévias sobre os objetos a serem rastreados e, a partir dessas informações, constroem-se modelos matemáticos para serem utilizados na detecção e rastreamento de objetos. Outras, não possuem informações prévias sobre o objeto de interesse e por isso baseiam-se exclusivamente em regiões da imagem ou da cena em questão. O que existe em comum entre essas abordagens é a restrição do processamento em tempo real, exigindo que se focalize atenção para o desempenho dos métodos implementados.

A detecção e o rastreamento de objetos são técnicas que estão sendo desenvolvidas e empregadas em diversas áreas, como na engenharia e computação, que por sua vez podem oferecer subsídios à medicina, geologia, dentre outras.

Seguindo o conceito de detecção e rastreamento de objetos, encontram-se diversas aplicações onde estas técnicas podem ser empregadas, tais como:

- 1- Sistemas de vigilância: Neste tipo de aplicação, o sistema utiliza os dados, obtidos por uma câmera digital, para procurar continuamente indícios de movimento na área apontada pela câmera.
- 2- Análise de movimentos humanos: Esta técnica além de utilizar conceitos de processamento de imagens, em particular morfologia matemática (Wren, C.; Azarbayejani, A.; Darrell, T.; Pentland, A.; 1997), utiliza a aplicação de rastreamento dos movimentos humanos, através da análise de sequência de imagens digitais.

Na literatura encontram-se outras aplicações dessas técnicas em diferentes situações e propósitos.

A percepção de movimento e o rastreamento de objetos móveis (1 e 2) através de um sistema formado por um grupo de robôs dotados de câmeras (denominados de Agentes Visuais Móveis) é um problema a ser resolvido utilizando-se a técnica de rastreamento de objetos móveis. Neste caso (Pio, J. L. S.; Campos, M. F. M.; 2002), o

problema é dividido em duas etapas: a primeira consiste na detecção do alvo e, a segunda, no rastreamento do mesmo que, por sua vez, é formada por sete estágios: aquisição da imagem, detecção da presença do objeto na cena, rastreamento do objeto na cena, filtragens, reconhecimento, ajustes e efetivação dos ajustes no sensor. Três algoritmos são propostos para a detecção e rastreamento de objetos. O primeiro consiste na captura de movimento com a subtração do fundo da cena, que é a forma mais simples de detecção de movimento porém, não indica a direção em que o movimento ocorreu. O segundo realiza a captura do movimento sobre a região, que tem como objetivo acelerar o processo de rastreamento. O processamento é realizado numa janela obtida dinamicamente na sequência de imagens. Este ainda realiza a adequação do primeiro método, calculando a diferença de imagens com dois quadros consecutivos sobre uma região. O último algoritmo realiza a extração do objeto ou seja, ele detecta o objeto e faz o rastreamento a partir de uma base móvel, e ainda tenta detectar o mesmo objeto em movimento em cada imagem.

De acordo com (Ghiasi, S.; Moon, H. J.; Sarrafazdeh, M.; 2004), o desenvolvimento de um sistema para rastreamento de objeto em tempo real reconfigurável e colaborativo foi proposto a fim de tornar o mesmo capaz de adaptar-se aos eventos externos. Levando em consideração a dificuldade de se prever esses eventos a solução encontrada seria a utilização de um hardware reconfigurável, uma vez que este fornece flexibilidade e adaptabilidade para o sistema proposto. O sistema é composto por câmeras, que permitem a aquisição e o processamento da imagem, por um computador que, atua como uma central e comanda o processamento, e por uma rede, que permite a comunicação dos dados. O sistema é implementado para detectar continuamente e

rastrear objetos em movimento que estejam dentro do campo de visão da câmera. Se o objeto sair do campo de visão a câmera deverá buscá-lo ou o controle deverá passar para outra câmera. Um FPGA é utilizado para selecionar os pontos característicos da cena (cantos abruptos, variação de intensidade na imagem). Essas características são passadas para um algoritmo de rastreamento. Como o número de pontos característicos muda na cena, um algoritmo foi desenvolvido para detectar essas mudanças e adaptá-las para compensar as mudanças no ambiente. Sendo assim, o número de pontos característicos assegura a exatidão do rastreamento. Isto pode ser melhorado através da reconfiguração do FPGA.

Uma técnica para rastreamento de objeto baseado na aproximação da Hipótese Múltipla Bayesiana de Rastreamento (MHT) utiliza dois algoritmos combinados para gerar o rastreador do objeto. O primeiro é empregado para a segmentação do contorno, baseado num mapa de borda. O segundo é usado no rastreamento temporal de um objeto selecionado de um quadro inicial. Um objeto é representado por pontos característicos, que são extraídos dele. Tais pontos são obtidos usando informações do mapa de bordas. Esses pontos são rastreados através da sequência de imagens. Para confirmar o rastreamento correto desses pontos, a posição de cada ponto na trajetória é comparada com o objeto segmentado em cada quadro (Tissainayagam, P.; Suter, D.; 2000).

Segundo (Brémond, F.; Thonnat, M.; 1998), um método para rastrear múltiplos objetos não rígidos em uma sequência de vídeo é desenvolvido com o intuito de se ter um sistema de interpretação para ser usado em inspeções automáticas de cenas internas

e externas por uma câmera. Dada a seqüência de uma cena, o sistema de interpretação analisa o comportamento dos objetos reais (humanos ou não). Esse sistema de interpretação é composto por três módulos. Um módulo de processamento de imagem, que detecta regiões de movimento, um de rastreamento, que associa essas regiões para rastrear os alvos que correspondem ao objeto real e, um módulo de reconhecimento do cenário, que identifica os alvos como objetos reais e reconhece o cenário relativo ao comportamento deles. O problema de rastreamento está na ordem de interpretação da cena pois, a perda do objeto rastreado impossibilita a análise do comportamento. Este método é baseado no rastreamento da aparência, ou seja, é rastreada a percepção do movimento do objeto na cena e não o objeto em real.

O rastreamento de veículos em tempo real, através do processamento de imagem de vídeo pode ser uma solução promissora para congestionamento de tráfego. Este método pode fornecer parâmetros do tráfego como o fluxo e velocidade, bem como mudanças de ruas e trajetórias dos veículos. Para que essa ferramenta seja eficiente, um sistema de processamento de imagem de vídeo deve possuir algumas exigências, tais como: segmentação automática de cada veículo, do fundo e dos outros veículos que são detectados; detecção de todos os tipos de veículos; suporte para variações de condições de iluminação e, por fim, operação em tempo real. O sistema proposto utiliza vídeo, obtido através de câmeras, e processa-os. Então, o dado é transmitido para localização central, como se fosse um Centro de Gerenciamento de Tráfego, para comparação e cálculo dos parâmetros. O processo ocorre em três estágios: segmentação da cena em veículos individuais e rastreamento de cada veículo dentro de uma zona para refinar e atualizar a posição e velocidade em coordenadas 3D; processamento dos dados da

trajetória para o cálculo dos parâmetros do tráfego local e, no Centro de Gerenciamento, os parâmetros de cada lugar são comparados e, as informações de sinais das câmeras vizinhas são processadas para calcular os parâmetros da seção (Coifman, B.; Beymer, D.; 1998).

Em (Kakadiaris, I.; Metaxas, D.; 2000), são apresentadas técnicas e formulações para estimação de movimento humano através de múltiplas câmeras. Este método é baseado na análise espaço temporal da silhueta da pessoa, não sendo necessário o uso de marcadores ou similares. O rastreamento de movimento humano em imagens 3D está dando suporte a inúmeras aplicações, tais como: análise de postura e modo de andar de atletas, animações do corpo humano, mãos e faces e, anotações automáticas de atividades humanas em base de dados de vídeo. Algumas informações são importantes para certas aplicações como, por exemplo, o centróide da silhueta e, informações detalhadas de forma e movimento. Em certos casos, mesmo os modelos mais práticos e animações não são capazes de reproduzir exatamente as formas e movimentos necessários, sendo então, preciso criar atores virtuais para o estudo do movimento. O rastreamento do movimento pode ser executado usando técnicas mecânicas eletromagnéticas e baseadas em imagens. Sistemas baseados em imagens podem ser tratados em três categorias: Técnicas que utilizam marcadores ativos, passivos e que não utilizam marcadores. O método preferível é o método de sensoriamento não intrusivo baseado em visão. Neste trabalho a oclusão é computada, baseada no movimento preditivo e o rastreamento é baseado nos contornos oclusos. Para superar problemas do uso de modelos aproximados para a estimação do movimento 3D, um método que estima a forma das partes do corpo foi proposto. O método é baseado na fusão da

observação (de várias câmeras) de um conjunto das performances do indivíduo de acordo com um protocolo designado para revelar a estrutura do corpo humano.

Aplicações em segurança e inspeção necessitam de monitoramento de áreas com visão de 360°. Tradicionalmente são usadas tanto câmeras rotacionadas ou um número de câmeras com pequena área de visão. No entanto, ambas apresentam desvantagens. Por exemplo, câmeras com rotação apresentam problemas com imagens não estáticas e com campo de cobertura não permanente. Soluções com múltiplas câmeras levam a problemas com fronteiras quando o objeto em movimento rastreado está se movendo do campo de visão de uma câmera para outro. Por outro lado, ambas as soluções são caras, pois são necessárias plataformas de rotação ou câmeras ópticas. Em (Földesy, P.; Szatmári, I.; Zarándy, Á.; 2002) um modo eficiente e barato para solucionar estes problemas é apresentado. Duas tecnologias foram combinadas. A primeira é o sistema de Lentes Panorâmicas Anelares (PAL), o qual projeta o campo de visão de 360° dentro de uma imagem em forma de disco. A segunda tecnologia é o Computador Visual Aladdin (computador de alta performance), o qual é equipado pelo chip ACE4K 64x64 CNNUN¹ (I. Szatmári, 2000). A combinação dessas duas tecnologias leva a um sistema de monitoramento por uma única câmera com ângulo de visão de 360° em tempo real.

¹ Processador para Rede Neural Universal (CNNUN – Cellular Neural Network Universal), com arranjo de células de 64 x 64.

Tem havido um aumento repentino no número de câmeras de inspeção nos últimos anos. Circuitos fechados de televisão têm crescido significativamente, sendo usados por companhias para proteção pessoal, tornando-se uma ferramenta usada por autoridades para a inspeção de áreas públicas. Entretanto, várias questões importantes devem ser consideradas antes de se confiar na inspeção por vídeo como uma ferramenta efetiva para prevenção de crimes. Em (Niu, W.; Jiao, L.; Han, D.; Wang, Y.; 2004) é dada uma visão sobre uma estrutura de um sistema de inspeção de vídeo com multi-câmeras que abrange esquemas para a fusão de dados espaços-temporais, representação de eventos e reconhecimento. Este sistema pode ainda rastrear múltiplos carros e detectar hábitos suspeitos de direção (dirigir em zig-zag). Esse sistema de inspeção de veículos por vídeo reconhece tipos de manobras pela análise das trajetórias. Várias câmeras são colocadas em um estacionamento para aumentar a cobertura do espaço. Cada câmera é calibrada e registrada para um quadro comum de referência. A trajetória do veículo é integrada do resultado do rastreamento de várias câmeras. Com a ajuda da informação do registro da câmera, a trajetória pode ser fundida no sistema de coordenadas. O algoritmo de reconhecimento, que utiliza as trajetórias como entrada, consiste de duas etapas: a primeira transfere os dados numéricos da trajetória em descrições semânticas, tais como curvas e paradas; e o segundo reconhece os padrões de movimentos. O sistema é focado na generalização dessa estrutura para o reconhecimento de ação humana. Enquanto o fluxo de processamento de detecção – representação – reconhecimento pode ser válido para o reconhecimento de atividades humanas e veiculares, o rastreamento detalhado, esquemas de representação e reconhecimento devem ser adaptados para silhuetas deformáveis e figuras humanas. A segmentação e o rastreamento de múltiplas pessoas em tempo real é um desafio. Por isso, nesse sistema

de inspeção, pode-se detectar automaticamente objetos em movimento e classificar o movimento dos mesmos em categorias semânticas como carros e/ou humanos. Se o objeto rastreado é uma pessoa, um único rótulo será referenciado a essa pessoa. Informações como centróide, altura e o valor médio da intensidade da região ocupada pela pessoa rastreada serão gravados como uma assinatura. Um filtro de Kalman é usado para rastrear cada pessoa. Como resultado, múltiplas pessoas podem entrar e sair da cena aleatoriamente. A irregularidade, como fusão de silhuetas, é manuseada e indivíduos podem ser rastreados corretamente depois que um grupo de pessoas se dividirem. Se a mesma pessoa entrar novamente no campo de visão, em um período de tempo consideravelmente curto, o sistema reconhece e assimila o mesmo rótulo para a pessoa.

Um dos maiores desafios na área de visão computacional é o desenvolvimento de sistemas capazes de realizar o processamento de imagens digitais em tempo real a um custo baixo. Para isso, faz-se o uso de computadores de uso geral, que permitem uma rápida implementação de sistemas e de vários tipos de algoritmos. Sabe-se, no entanto, que existem várias limitações para esses computadores e, uma maneira pra contornar esses problemas, seria o uso de uma arquitetura dedicada, projetada em hardware, para executar os algoritmos em tempo real. O uso de dispositivos programáveis CPLDs e FPGAs possibilita um rápido desenvolvimento, análise e implementação de projetos mais complexos a um custo mais baixo.

Esta primeira parte da dissertação faz uma revisão da literatura sobre detecção de movimento e rastreamento de objetos em tempo real.

1.2 Objetivo

O objetivo deste trabalho é desenvolver um sistema capaz de detectar o movimento de objetos em tempo real a partir do sinal de vídeo padrão, proveniente de uma câmera de vídeo monocromática CCD (*Charge-Coupled Device*). O sistema proposto será composto por um módulo de captação, digitalização e exibição do sinal de vídeo e, por um segundo módulo responsável pelo armazenamento das imagens digitalizadas e, também pela integração do sinal de vídeo. O sistema deverá ser capaz de detectar o movimento de objetos nas direções horizontal e vertical. Para a detecção de movimento pode-se utilizar a informação média de cada linha da imagem assim, processando a informação média de cada linha e comparando a informação de quadros sucessivos será possível detectar se ocorreram variações na direção. Para deslocamentos na direção horizontal, que correspondem a maior parte dos movimentos, a integração terá de ser realizada nas linhas da imagem correspondentes à posição vertical do objeto na cena já, para deslocamentos na direção vertical a integração será realizada nas linhas da imagem correspondentes à posição horizontal do objeto na cena. O sistema de detecção de movimento conta com um módulo de aquisição, armazenamento e exibição de imagens monocromáticas, que será detalhado mais adiante.

1.3 Organização

Essa dissertação é composta por sete capítulos:

Neste primeiro capítulo, é feita uma introdução ao assunto, apresentando algumas aplicações e suas importâncias, onde foram apresentados alguns assuntos relacionados aos temas, encontrados na literatura e, por último apresentou-se o objetivo desse trabalho. No capítulo 2, será apresentada uma visão geral sobre dispositivos de lógica programável complexa (CPLDs), incluindo sua arquitetura e funcionamento. No capítulo 3 será feita uma explicação detalhada sobre o sinal de vídeo composto e sobre a arquitetura desenvolvida para a captação, digitalização e exibição de imagens monocromáticas. A etapa de armazenamento da imagem digitalizada, utilizando CPLDs, será apresentada de maneira detalhada no capítulo 4.

No capítulo 5, será descrito o processo de integração dos *pixels*, tanto na direção horizontal quanto na direção vertical. Os testes e resultados do circuito implementado serão apresentados no capítulo 6. Por fim, no capítulo 7, serão apresentadas as conclusões e algumas propostas para trabalhos futuros.

Capítulo 2

2 Dispositivos Lógicos Programáveis (PLDs)

O projeto de sistemas digitais tem sofrido grandes mudanças nos últimos anos devido ao desenvolvimento dos dispositivos lógicos programáveis (PLDs). Conseqüentemente, os projetistas têm se deparado com uma grande quantidade de novos dispositivos e com seus respectivos softwares. A fim de amenizar a dificuldade na hora da escolha do dispositivo mais eficiente para o projeto é necessário que o projetista tenha um conhecimento sólido das bases de funcionamento tanto do componente quanto do software que será utilizado (Jenkins, J. H., 1994).

Segundo (Chan e Mourad; 1994), com os componentes dos circuitos evoluindo de transistores a circuitos integrados em larga escala [VLSI (Very Large Scale Integration)] e com as ferramentas CAD (Computer Aided Design) o ciclo de projeto de circuitos digitais tornou-se mais simples e mais rápido. Além disso, não é mais necessário montar diferentes componentes ou desenhar portas lógicas individuais, uma vez que as linguagens de descrição de hardware [Hardware Description Languages (HDL)] têm facilitado a descrição de projetos em um nível hierárquico mais complexo. Ferramentas de síntese lógica automática também estão disponíveis para mapear circuitos em diversas tecnologias. Devido às rápidas mudanças na tecnologia e técnicas de projeto de VLSI, o ciclo de vida

dos produtos modernos está se tornando cada vez menor em relação ao tempo necessário para o projeto e desenvolvimento do mesmo, tornando-se necessário uma prototipação mais rápida.

Circuitos integrados digitais (ICs) podem ser implementados em diferentes tecnologias, dependendo do seu tamanho e de sua função no sistema. A implementação de circuitos digitais pode ser agrupada em duas categorias: *totalmente customizados* e *semicustomizados*, como pode ser visto na figura que se segue.

Figura 2-1 Metodologias de projeto

De acordo com a figura 2-1, a última categoria (Lógica Programável), subdivide-se em PLDs e FPGAs. Estas duas subcategorias têm facilitado o projeto e a produção de circuitos integrados para aplicações específicas [ASICs (Application Specific Integrated Circuits)]. Ainda, de acordo com a figura anterior tem-se:

- **CIs totalmente customizados ou ASICs (*Application Specific Integrated Circuits*):** Estes CIs são criados usando máscaras para todas as camadas durante o processo de fabricação, uma vez que neste tipo de projeto cada função lógica é manualmente projetada e otimizada. Além disso, o projetista controla todos os estágios do “*layout*” do chip possibilitando assim máxima flexibilidade e alta performance. Conseqüentemente, o CI é mais compacto e oferece um maior desempenho com um baixo consumo de energia. Para aplicações que necessitem de grande volume de produção, estes CIs customizados fornecem uma alternativa de baixo custo.

- **MPGAs (*Mask-Programmable Gate Arrays*):** Neste tipo de implementação, a especificação das funções lógicas deve ser feita antes do processo de fabricação do CI. Esta especificação é feita através de um mapeamento de transistores pré-fabricados numa pastilha. Para o processo de fabricação são utilizadas máscaras genéricas de módulos pré-projetados, para a metalização das camadas. Essas máscaras criam um arranjo modular de blocos funcionais. Para a interconexão desses blocos funcionais o processo necessita de máscaras específicas. O projeto destes CIs é usualmente facilitado pelo uso de uma biblioteca de células, fazendo com que a pouca habilidade do projetista não seja um fator crítico, como no caso das metodologias totalmente customizadas. Os MPGAs oferecem um tempo de desenvolvimento mais curto e custos mais baixos do que os CIs customizados.

- **Standard Cells:** Assim como na implementação das MPGAs, este tipo de projeto é facilitado pelo uso de módulos pré-projetados. Os módulos, *standard cells*, são

geralmente salvos em uma base de dados. Os projetistas selecionam as células da base de dados para realizar seus projetos. Estas células são dispostas em colunas e interligadas. O roteamento é feito dentro de canais que podem ter tamanhos variáveis. Sendo assim, percebe-se que se trata de um processo totalmente automatizado aonde as etapas descritas de fabricação são realizadas com o auxílio de ferramentas CAD. Comparando com os CIs customizados, os circuitos implementados em *standard cells* são menos eficientes em tamanho e desempenho; porém, o custo de fabricação é mais baixo e seu tempo de projeto é menor.

- **PLDs (Programmable Logic Devices):** Dispositivos lógicos programáveis, são dispositivos utilizados para implementar funções lógicas onde o *chip* pode ser programado (pelo usuário) via software. Este software geralmente é específico para o dispositivo em questão e é fornecido pelo fabricante do mesmo. Com essa capacidade de configuração e reconfiguração esta tecnologia elimina o processo de fabricação ou seja, o projetista pode produzir CIs de aplicações específicas (ASICs) em um curto período de tempo e com um custo reduzido. Os PLDs podem ser encontrados no mercado com características de densidade, performance e custo equivalente ao de um dispositivo não programável (ASIC). Estes fatores, aliados com o tempo de produção e flexibilidade na programação ajudam a manter o crescimento desta tecnologia, de alta densidade, no desenvolvimento de sistemas digitais.

Como mostrado na figura 2-2, a seguir, os dispositivos de lógica programável se dividem em dois tipos, são eles:

1. **SPLDs (Simple PLDs):** Esses PLDs usam tecnologia CMOS e oferecem elementos de memória do tipo EPROM, EEPROM e FLASH e englobam os PALs, GALs e os obsoletos PLAs.
2. **HCPLDs (High Capacity PLDs):** Os dispositivos lógicos de alta capacidade também utilizam tecnologia CMOS com memória EPROM, EEPROM, FLASH, SRAM e antifusível. Nesta família encaixam-se os CPLDs (Complex PLDs) e os FPGAs (Field Programmable Gate Arrays).

Figura 2-2 Dispositivos de lógica programável (“PLDs”) comerciais

2.1 A Evolução dos Dispositivos Lógicos Programáveis

2.1.1 Arquiteturas Básicas

As arquiteturas da primeira geração de PLDs baseiam-se no conceito de soma de produtos, que representa uma função seja ela combinacional ou sequencial. Para essa manipulação lógica são usadas as leis de De Morgan e teoremas da álgebra booleana.

Basicamente, esses componentes são formados por dois arranjos de portas lógicas: uma matriz de portas lógicas AND (*array AND*), que é conectada aos sinais de entrada do CI, e por uma matriz de portas lógicas OR (*array OR*), que fornece os sinais de saída do circuito. Então, combinando-se as entradas no arranjo AND, os termos produtos são gerados, e estes, por sua vez, são enviados ao arranjo OR, sendo somados, resultando na soma de produtos, responsáveis pela implementação da lógica requerida.

Dependendo do tipo de PLD, uma ou ambas as matrizes de portas lógicas podem ser programáveis sendo assim, de acordo com a programação das matrizes lógicas AND e/ou OR, os PLDs podem ser classificados da seguinte forma:

- **PROM:** apresenta a matriz AND fixa e a matriz OR programável;
- **PAL:** apresenta a matriz AND programável e a matriz OR fixa;
- **PLA:** apresenta ambas as matrizes AND e OR programáveis.

Uma representação simbólica foi criada para um melhor entendimento dos PLDs. Esta representação segue o conceito de *linha de produto* e *linha de soma*, aonde tanto as entradas das portas lógicas AND e OR são representadas apenas por uma linha, como mostradas na figura seguinte.

Figura 2-3 Representação PLDs

Inicialmente, os PLDs permitiam a implementação de circuitos de baixa complexidade (em torno de 1.000 portas equivalentes) no entanto, hoje em dia, os PLDs podem alcançar densidades de até 250.000 portas (para os mais modernos), tornando possível projetar subsistemas inteiramente digitais com um único dispositivo. Sabe-se que, no início, os PLDs possuíam apenas um bloco lógico que realizava globalmente as operações, conectado aos pinos de entrada e saída do dispositivo. Porém, com o aperfeiçoamento das tecnologias de fabricação, os PLDs receberam uma estrutura interna de interconexões programáveis (como mostrado na figura 2-3), flexibilizando ainda mais a programação.

2.1.2 PROM (Programmable Read-Only Memory)

As formas de implementação de circuitos customizados (ASICs), personalizáveis após o encapsulamento, ou PLDs, surgiram em meados da década de 70, e o primeiro tipo de *chip* programável pelo usuário, que poderia implementar circuitos lógicos, foi a PROM (*Programmable Read-Only Memory*), desenvolvido pela Monolithic Memories Inc.. Estes componentes podem ser usados eficientemente tanto como memória de alta velocidade, para a armazenagem de dados, como para a implementação de circuitos lógicos, onde as linhas de endereçamento podem ser usadas como entradas do circuito lógico e as linhas de dados como saídas.

Quando uma PROM é utilizada como memória o arranjo AND, que decodifica o endereço da memória, é predefinido em fábrica e, somente o arranjo OR, que fornece o valor na saída, é programável. A figura 2-4, mostra uma PROM que funciona como um tipo de PLD.

Figura 2-4 Circuito simplificado de uma PROM

Segundo (Monolithic Memories Inc. 1984), quando um componente PROM for utilizado para substituição de lógica aleatória (lógica que não segue um padrão rígido de roteamento) este passa a ser chamado de PLE (“Programmable Logic Element”). Neste tipo de aplicação, cada porta lógica do arranjo AND corresponde a um termo produto, com variáveis fixas, previamente decodificadas neste arranjo. As saídas são as somas dos produtos, definidos pelo usuário, no arranjo programável OR. Sabe-se que as equações booleanas, que descrevem o funcionamento de um circuito lógico, possuem um número limitado de termos produto e, com a flexibilidade do arranjo programável OR nas PROMs esta arquitetura torna-se desnecessária e ineficiente para a implementação de circuitos lógicos, sendo raramente empregadas na prática.

2.1.3 PLA (*Programmable Logic Array*)

Os PLAs são os componentes que apresentam maior flexibilidade na implementação de soma de produtos, uma vez que ambas as matrizes de portas lógicas AND e OR são programáveis. Esses dispositivos foram os primeiros a serem criados especificamente para a implementação de circuitos lógicos. Foram introduzidos no mercado pela Philips, no início dos anos 70. A arquitetura do PLA contém um dispositivo de programação por função lógica a mais do que as arquiteturas PAL e PROM, correspondendo a um acréscimo em área de pastilha e uma diminuição da velocidade. Um dispositivo PLA é estruturado de tal forma que cada saída da matriz AND pode corresponder a qualquer termo produto das entradas. Da mesma forma, cada saída da matriz OR pode ser configurada para produzir a soma lógica de quaisquer saídas do plano AND. A figura 2-5 mostra o esquema de um dispositivo PLA. Os principais problemas apresentados por esses dispositivos eram o alto custo de fabricação e o fraco desempenho de velocidade, devido principalmente aos dois níveis de lógica reconfigurável, pois os planos lógicos eram difíceis de se fabricar e introduziam atrasos significantes de propagação dos sinais elétricos. Para superar essas deficiências foi desenvolvida a tecnologia PAL (*Programmable Array Logic*).

Figura 2-5 Circuito simplificado de um PLA

2.1.4 PAL (*Programmable Array Logic*)

Os PALs foram desenvolvidos pela Monolithic Memories Inc., em 1978. Estes dispositivos possuem uma matriz de portas lógicas AND, programável, seguido por uma matriz de portas lógicas OR fixas. O fato de estes dispositivos possuírem a matriz AND programável torna possível que estes tenham um maior número de entradas e, juntamente com a matriz OR, fixa, o dispositivo tende a ser menor (o que acarreta um custo reduzido) e mais rápido, porém com um número limitado de termos produto por saída (AMD – Advanced Micro Device, June 1993). Os PALs podem ser encontrados em diversas tecnologias, tais como TTL, CMOS e ECL (*Emitter Coupled Logic*). Os componentes CMOS permitem as mesmas funções dos TTLS e podem ser utilizados com a mesma

configuração de pinos, com um menor consumo de potência. Permitem ainda que sejam reprogramados, pois podem ser configurados com dispositivos EPROM ou EEPROM. Os PALs TTL são mais rápidos e estão disponíveis apenas em tecnologia PROM. Já, por sua vez, os componentes ECL, destinam-se a aplicações de alta velocidade e necessitam de considerações de projeto completamente diferentes dos dispositivos com tecnologia CMOS e TTL, não podendo ser substituídos por componentes desta tecnologia. Na figura 2-6, tem-se um exemplo de um dispositivo PAL.

Figura 2-6 Circuito simplificado de um PAL

Todos os PLDs, incluindo PLAs, PALs e similares são agrupados na categoria dos SPLDs (*Simple PLDs*), como mostrado anteriormente na figura 2-2. Esta categoria tem como principais características o baixo custo e o alto desempenho. No entanto, uma das dificuldades encontradas seria com relação ao aumento da capacidade da arquitetura destes

dispositivos, uma vez que a estrutura das matrizes lógicas programáveis aumenta muito rapidamente com o aumento do número de entradas. Uma solução seria integrar vários SPLDs em um único *chip*, e prover interconexões programáveis para a conexão dos diversos PLDs. Com o avanço da tecnologia foi possível esta solução, surgindo assim os CPLDs (*Complex CPLDs*), que serão descritos mais adiante.

2.2 Elementos de Programação

A programação dos dispositivos PLDs é feita através de “*chaves*” que permitem conectar, ou não, dois pontos das matrizes programáveis, definindo assim os caminhos de roteamento de sinais e/ou as funções dos blocos lógicos configuráveis. Estes elementos (*chaves*) são a base para a construção de componentes personalizáveis após o encapsulamento pois, através da sua programação, configura-se o PLD para que ele realize determinadas funções lógicas.

Para um bom entendimento do funcionamento e eficiência dos PLDs é importante tomar conhecimento de duas características que são de grande importância para a compreensão das vantagens de certas famílias de componentes e sua aplicabilidade; são elas:

- **Volatilidade:** refere-se à dependência do componente em manter sua configuração na ausência do sinal de alimentação;

- **Reprogramabilidade:** refere-se à possibilidade de reconfiguração das “chaves” do componente mesmo após ele ter sido configurado anteriormente.

Os PLDs, disponíveis comercialmente, apresentam como elementos de programação (*chaves*) fusíveis (PROM), antifusíveis, células SRAM, dispositivos EPROM ou EEPROM.

2.2.1 Fusível (*Fuse*)

O fusível [PROM (*Programmable Read-Only Memory*)] foi o primeiro tipo de comutador programável pelo usuário final. São formados por material condutivo de baixo ponto de fusão e mantêm conectados dois pontos, a menos que sejam queimados, provocando seu rompimento e isolando estes pontos. Esta tecnologia é denominada OTP (*One Time Programmable*) sendo portanto, programável uma única vez. Ocupam uma pequena área da pastilha, suficiente para colocar o material a ser queimado e garantir a isolamento dos pontos após a programação. Estes elementos são construídos com tecnologia bipolar, que por sua vez, não permite a alta integração de transistores, como ocorre com a tecnologia MOS.

Os fusíveis foram, por muito tempo, os responsáveis pela existência dos PLDs da primeira geração, como os PALs e ainda hoje são utilizados para a fabricação de SPLDs

bipolares. Porém, pelo fato de serem não reprogramáveis, e serem fabricados com tecnologia bipolar de menor densidade de transistores do que a tecnologia MOS, estes elementos não acompanham a evolução dos componentes programáveis.

2.2.2 Antifusível (Antifuse)

Este tipo de comutador programável, assim como os fusíveis são OTP, sendo programável apenas uma única vez, figura 2-7 (Greene, J. , 1993). O antifusível (*antifuse*), é um dispositivo de dois terminais, que quando programados oferecem uma baixa resistência, e quando não, apresentam uma alta impedância entre seus terminais, comportando-se como um circuito aberto. Para a programação deste elemento é necessário que uma tensão entre 11 e 20 volts seja aplicada ao *antifusível* que é então “queimado”, criando uma conexão de baixa impedância. Além disso, para a programação, é necessária a presença de transistores com larga área, a fim de suportar as correntes de programação (~5mA). Uma vantagem dos antifusíveis com relação aos fusíveis é que eles são fabricados com tecnologia MOS, justificando a necessidade de transistores em larga área, como dito anteriormente. É interessante ressaltar que este elemento de programação ocupa uma pequena área, a qual é um pouco maior do que dois segmentos transversais de fios metálicos. Esta vantagem é reduzida pela larga área necessária para os transistores de programação e pela inclusão de transistores de isolamento, que são necessários para a isolamento dos circuitos de programação, já que os mesmos trabalham com tensão de até 20V.

Figura 2-7 Foto antifusível. (a) Antes da programação. (b) Após programação. (c) Vista de cima, um antifusível é aproximadamente do mesmo tamanho que um contato.

2.2.3 Células SRAM (*SRAM Cells*)

Esta tecnologia de programação utiliza células SRAM (*Static Random Access Memory*) que controlam transistores de passagem ou multiplexadores que, por sua vez, realizam as conexões dos pontos para a configuração do PLD. Os transistores de passagem permitem conectar ou isolar dois pontos, de acordo com o dado armazenado na célula SRAM, enquanto que o uso de multiplexadores permite inúmeras combinações possíveis entre os pontos. Os multiplexadores são úteis para selecionar qual sinal será utilizado como variável de entrada, dentre os diversos que chegam a uma célula lógica configurável.

A célula SRAM consiste de dois inversores realimentados sendo assim, um dispositivo volátil, facilmente reconfigurável e que ocupa uma área equivalente a de seis transistores MOS, ou seja: cinco transistores de célula SRAM, mais a chave de conexão (transistor de passagem). Devido à sua volatilidade, as configurações das células devem ser restabelecidas sempre que o dispositivo for religado, necessitando-se portanto de um dispositivo de memória externo do tipo PROM, EPROM, EEPROM ou disco magnético. A

grande desvantagem desta tecnologia de programação é a extensa área ocupada no *chip*. No entanto, existem duas grandes vantagens: uma é a rápida reprogramabilidade e a outra é que requer somente tecnologia padrão de processos de circuitos integrados. A figura 2-8 mostra um exemplo da tecnologia de programação SRAM.

Figura 2-8 Tecnologia de programação SRAM

2.2.4 EPROM e EEPROM

Segundo (Sedra, A. S.; Smith, K. C.; 2005), as EPROMs (*Erasable Programmable Read-Only Memory*) utilizam variantes de célula de memória cuja seção transversal está mostrada na figura 2-9. A célula é basicamente um MOSFET canal *n* com duas portas feitas de silício policristalino. Uma das portas não é eletricamente conectada a nenhuma outra

parte do circuito; em vez disso, é deixada flutuante e é apropriadamente chamada **de porta flutuante**. A outra porta, chamada de **porta de seleção**, funciona do mesmo modo que a porta de um MOSFET tipo enriquecimento normal.

O transistor MOS da figura 2-9 (a) é conhecido como **transistor de porta flutuante** e é representado pelo símbolo de circuito mostrado na figura 2-9 (b). Nesse símbolo, a linha tracejada representa a porta flutuante. A célula de memória é conhecida como **célula de portas empilhadas**.

Figura 2-9 (a) Seção de corte e (b) Símbolo do circuito do transistor de porta flutuante usado como célula

Antes de a célula ser programada, não há cargas na porta flutuante e o dispositivo funciona como um MOSFET canal n enriquecimento normal. Portanto, ele exibe a característica $i_D - v_{GS}$, mostrada como curva (a) na figura 2-10.

Observe que, nesse caso, a tensão de limiar (V_t) é muito baixa. Esse estado do transistor é conhecido como **estado não programado**. Esse é um dos dois estados em que

o transistor de porta flutuante pode-se encontrar. Tomar-se á o estado não-programado para representar um “1” armazenado. Isto é, o transistor de porta flutuante cuja característica $i_D - v_{GS}$ está mostrada como a curva (a) na figura 2-10 é dito estar armazenando um “1”.

Para programar o transistor de porta flutuante, é aplicada uma tensão elevada (cerca de 16 a 20V) entre o dreno e a fonte. Ao mesmo tempo, é aplicada uma tensão elevada (cerca de 25V) em sua porta de seleção. A figura 2-11 mostra um MOSFET de porta flutuante durante a programação. Na ausência de cargas na porta flutuante, o dispositivo se encontra como um MOSFET de canal n tipo enriquecimento normal. Uma camada de inversão tipo n (canal) é criada na superfície da lâmina como resultado da tensão positiva elevada aplicada à porta de seleção. Devido à aplicação da tensão positiva elevada no dreno, o canal adquire uma forma afunilada.

Figura 2-10 Ilustração do deslocamento na característica $i_D - v_{GS}$ de um transistor de porta flutuante como resultado da programação

A tensão dreno-fonte acelera os elétrons através do canal. Quando esses elétrons alcançam o dreno no final do canal, eles adquirem energia cinética suficientemente alta e são chamados de *elétrons quentes*. A tensão positiva elevada na porta de seleção (maior do que a tensão de dreno) estabelece um campo elétrico no óxido isolante. Esse campo elétrico atrai os elétrons quentes e os acelera em direção à porta flutuante. Desse modo, a porta flutuante fica carregada e a carga nela acumulada fica aprisionada.

Felizmente, esse processo de carregamento da porta flutuante é autolimitante. A carga negativa que se acumula na porta flutuante reduz a intensidade do campo elétrico no óxido até o ponto em que, eventualmente, ele se torna incapaz de continuar acelerando os elétrons quentes.

A carga negativa aprisionada na porta flutuante fará com que os elétrons sejam repelidos da superfície do substrato. Isso significa que, para formar o canal, a tensão positiva que é aplicada à porta de seleção será maior do que aquela necessária quando a porta flutuante não está com carga. Em outras palavras, a tensão de limiar V_t do transistor programado será maior do que a do transistor não-programado. Na realidade, a programação faz com que a característica $i_D - v_{GS}$ se desloque para o ponto denominado (b) na figura 2-11.

Figura 2-11 O transistor de porta flutuante durante a programação

Nesse estado, conhecido como *estado programado*, a célula é dita estar armazenando um 0.

Uma vez programado, o dispositivo de porta flutuante retém esse deslocamento na característica $i - v$ (curva b), mesmo quando a fonte de alimentação é desligada. De fato, resultados experimentais indicam que o dispositivo pode permanecer no estado programado por um tempo muito longo.

A leitura do conteúdo da célula empilhada é fácil: uma tensão V_{GS} , entre os valores das tensões de limiar baixa e alta (figura 2-10), é aplicada à porta de seleção. Enquanto um dispositivo programado não conduz, um dispositivo não-programado conduzirá plenamente.

Para retornar o MOSFET de porta flutuante para seu estado não-programado, a carga armazenada na porta flutuante deve retornar ao substrato. Esse processo de

apagamento pode ser conseguido iluminando-se a célula com uma luz ultravioleta de comprimento de onda correto (253,7 nm) por um tempo determinado. A luz ultravioleta fornece energia suficiente aos elétrons aprisionados, permitindo que esses elétrons vençam a barreira de energia existente e passem através do óxido, voltando ao substrato. Para permitir esse processo de apagamento, a EPROM tem uma janela de quartzo em seu encapsulamento. Finalmente, deve ser observado que o dispositivo é extremamente durável e pode ser apagado e programado várias vezes.

Uma ROM programável mais versátil é a PROM apagável eletricamente (*electrically - erasable PROM - EEPROM*). Como o próprio nome sugere, uma EEPROM pode ser eletricamente apagada e programada sem a necessidade de iluminação ultravioleta. As EEPROMs utilizam uma variante do MOSFET com porta flutuante.

A tabela 2-1 resume as características mais importantes dos elementos de programação.

Tabela 2-1 Tecnologias de programação

LEMENTOS	VOLATIL	REPROGRAMÁVEL	TECNOLOGIA
Fusível	Não	Não	Bipolar
Anti-Fusível	Não	Não	CMOS+
SRAM	Sim	Sim	CMOS
EPROM	Não	Sim	UVC MOS
EEPROM	Não	Sim	EECMOS

2.3 CPLDs (Complex PLDs)

Os CPLDs consistem de múltiplos blocos lógicos, que se assemelham aos SPLDs (especialmente as PALs). Entretanto, os CPLDs são muito mais sofisticados, em relação à estrutura interna, dos que os SPLDs. Estes dispositivos foram introduzidos no mercado pela Altera Corp. e, inicialmente, esta família de *chips* foi chamada de EPLDs (*Erasable PLDs*), e então seguida de três séries adicionais de dispositivos, chamadas MAX5000, MAX7000 e MAX9000. O princípio de funcionamento dos primeiros CPLDs tem origem nos PALs, a única diferença é que enquanto no PAL existe apenas uma matriz AND programável, no CPLD existem vários sub-circuitos, com arquitetura e capacidade semelhante a um PAL, interconectados por uma matriz programável de interconexão global. A estes sub-circuitos dá-se o nome de *Logic Array Blocks* (LABs). Cada LAB contém de 4 a 16 macrocélulas (dependendo da sua arquitetura), blocos de I/O e um circuito de expansão de termos produtos. Cada macrocélula é formada por uma matriz AND conectada a uma matriz OR, para a implementação de soma de produtos, mais *flip-flops* que, em alguns casos, são responsáveis pela realimentação de sinais na matriz AND.

Os blocos de I/O funcionam independentes da macrocélula devido ao uso de uma célula tri-state entre o bloco de I/O e a macrocélula. O circuito de expansão de termos produto corresponde a uma matriz extra de portas lógicas AND, que pode ser usado para suprir a falta de termos produto na matriz da macrocélula ou para a implementação de *flip-flops*, para a armazenagem de dados. Como dito anteriormente, os LABs são interligados através de uma matriz de interconexão programável (PIA – *Programmable Interconnection*

Array), formados por dispositivos EPROM ou EEPROM e trilha de metal fixas. Esta matriz de interconexão é responsável pela alta capacidade de integração permitida pelos CPLDs.

Segundo (Pedrino C. E.; 2003), algumas características na arquitetura diferenciam os CPLDs tais como o número de termos produto por macrocélula, se estes podem ser emprestados para outras macrocélulas e se a matriz programável pode ser totalmente ou parcialmente conectada. Se o número de termos produto exceder a capacidade de uma macrocélula, termos adicionais de outras macrocélulas são emprestados, no entanto, isto pode aumentar o atraso interno de propagação.

Figura 2-12 Estrutura de um CPLD

2.3.1 CPLDs da Altera

Como dito anteriormente, a Altera desenvolveu três famílias de *chips* que se encaixam dentro da categoria dos CPLDs: MAX5000, MAX7000 e MAX9000. A arquitetura escolhida para o desenvolvimento deste trabalho foi a MAX7000, por ser uma das mais popularizadas e possuir características similares às outras arquiteturas de CPLDs.

A arquitetura geral da série MAX7000 da altera está ilustrada na figura 2-12. Ela inclui os seguintes elementos: um arranjo de blocos lógicos (LABs – *Logic Array Blocks*), macrocélulas, expensor de termos produtos, blocos de entrada e saída (I/Os) e de uma matriz de interconexão programável (PIA – *Programmable Interconnect Array*). O PIA é capaz de conectar qualquer entrada e saída de um LAB a qualquer outro LAB. Também, as entradas e saídas (I/Os) do *chip* são conectadas diretamente ao PIA e aos LABs.

Um LAB pode ser imaginado como sendo uma estrutura complexa similar a um SPLD, sendo assim, todo o *chip* pode ser considerado com sendo um arranjo de SPLDs. Os dispositivos da série MAX7000 estão disponíveis nas tecnologias EPROM e EEPROM. Até recentemente, os dispositivos com tecnologia EEPROM podiam ser programados apenas fora do circuito, em um programador especial. Em 1996, a Altera lançou a série 7000S, a qual permite a reprogramação no próprio circuito. A estrutura de um LAB pode ser vista na figura 2-13.

Figura 2-13 Arquitetura de um LAB

Cada LAB consiste de dois conjuntos de oito macro células (figura 2-13), onde cada macro célula compreende um conjunto de termos produto programáveis que alimentam uma porta OR e um flip-flop. Os *flip-flops* podem ser configurados como tipo D, JK, T SR ou podem ser transparentes. Como ilustrado na figura 2-14, o número de entradas para a porta OR, em uma macro célula, é variável. A porta OR pode ser alimentada por quaisquer ou por todos os termos produto dentro da macro célula, e ainda pode ter até 15 termos produto extras de outras macro células dentro do mesmo LAB. Esta flexibilidade de termos produto permite que essa série de dispositivos tenha uma maior eficiência em termos de área de chip, uma vez que as funções lógicas não necessitam de mais do que cinco termos produto, e a arquitetura suporta funções mais complexas quando necessário.

Além da Altera, vários outros fabricantes produzem dispositivos que podem ser categorizados como CPLDs. Por exemplo, AMD (*Mach Family*), Lattice (*série pLSI e ispLSI*), Xilinx (*XC7000*) e ICT (*PEEL array*) (Brown, S.; Rose, J; 1996).

Figura 2-14 Macro célula MAX7000

2.3.2 A Tecnologia FPGA

Os FPGAs (*Field Programmable Gate Array*) e, os CPLDs (*Complex Programmable Logic Device*) são circuitos integrados que podem ser configurados via software para implementar circuitos digitais, como processadores, controladores e decodificadores. Basicamente, um FPGA consiste em uma matriz de três tipos de elementos programáveis. O primeiro consiste de vários circuitos idênticos, compostos por alguns *flip-flops* e lógica combinacional extra, sendo conhecidos por CLBs (*Configuration Logic*

Blocks), e que formam um arranjo bi-dimensional. É com o auxílio dos CLBs que se projeta a lógica sequencial da máquina a ser implementada. O segundo consiste de circuitos de interfaceamento das saídas dos CLBs com o exterior do FPGA, chamados de IOBs (*Input Output Blocks*). Estes IOBs são constituídos por *buffers* bidirecionais com saída em alta impedância. Através de uma programação adequada de um IOB, configura-se um pino do FPGA para funcionar como entrada ou saída. O terceiro grupo é composto pelas interconexões (chaves) entre os CLBs e os IOBs. Estas chaves de interconexão são organizadas como canais de roteamento horizontal e vertical entre as linhas e colunas dos CLBs e IOBs. O processo de escolha das interconexões é chamado de roteamento.

Os FPGAs disponíveis comercialmente podem ser divididos em dois grupos, segundo as tecnologias de programação mais comumente utilizadas para se implementar os comutadores programáveis, são elas: FPGA com células **SRAM** e FPGA com **Antifuse**.

Quanto à arquitetura, os blocos lógicos dos FPGAs podem ser classificados em blocos de granulosidade grossa e blocos de granulosidade fina. A granulosidade pode ser definida de diversas formas com por exemplo: número de funções booleanas implementáveis por blocos lógicos, número equivalente de portas NAND ou XOR de duas entradas, número total de transistores, área total normalizada, número de entradas e saídas. Sendo assim, a granulosidade grossa é designada aos blocos lógicos maiores e mais complexos enquanto que a granulosidade fina aos blocos lógicos menores e mais simples.

Blocos de Granulosidade Grossa

Geralmente, estes blocos são formados por um ou mais *flip-flops* e duas ou mais *look-up tables* (LUTs). Uma *look-up table* é um tipo de bloco lógico que contém células de armazenamento que são utilizadas para implementar pequenas funções lógicas. Essas células de armazenamento podem ser vistas como uma memória SRAM sendo assim, a tabela verdade para uma função lógica de K-entradas é armazenada em uma SRAM $2^K \times 1$. As linhas de endereço da SRAM funcionam como entradas e, a saída fornece o valor da função lógica. A vantagem das *look-up tables* é sua grande funcionalidade. Uma LUT pode implementar qualquer função de K entradas, sendo assim, vão existir 2^n funções, onde $n=2^K$. A desvantagem é que um número maior do que 5 entradas acarretariam em um tamanho inaceitável da LUT. Enquanto o número de funções implementáveis cresce, estas funções adicionais são raramente utilizadas em projetos, e são muito difíceis de serem exploradas por ferramentas de síntese de lógica. Desta forma, uma LUT grande é sub utilizada.

A granulosidade pode acarretar efeitos na performance e na densidade dos blocos lógicos. Um estudo realizado mostrou que para otimizar a área do FPGA, o bloco lógico deve possuir cerca de 4 entradas ($K=4$) e, com relação à performance, uma LUT de 5 entradas apresenta um melhor desempenho.

Blocos de Granulosidade Fina

Os blocos lógicos de granulosidade fina são mais simples do que os de granulosidade grossa. Geralmente são formados por portas lógicas básicas ou multiplexadores de quatro entradas e um flip-flop. O melhor exemplo para um bloco lógico de granulosidade fina seria um bloco contendo alguns transistores interconectáveis ou portas lógicas básicas. A principal vantagem de se utilizar blocos de granulosidade fina é que os blocos são totalmente utilizados. A desvantagem desta arquitetura é que ela necessita de muitos segmentos de ligação e chaves programáveis para as interconexões e, nestes casos, será preciso mais células lógicas para se implementar uma função que seria implementada com menos células em uma arquitetura de granulosidade grossa. Isto também sobrecarrega o roteador, tornando-o lento e ocupando uma grande área no *chip*. Com isso, a tecnologia antifuse é a mais indicada para a fabricação deste tipo de FPGA, devido ao tamanho reduzido dos comutadores antifuse. Portanto, os FPGAs de granulosidade fina apresentam uma menor densidade do que os blocos lógicos de granulosidade grossa.

2.3.3 Arquiteturas das FPGAs

Existem várias indústrias, no ramo de semicondutores, que estão desenvolvendo dispositivos FPGAs. Atualmente há uma grande variedade desses dispositivos disponíveis comercialmente, como pode ser visto na tabela 2-2.

Tabela 2-2 Comparação de FPGAs com relação à capacidade, arquitetura, célula básica e métodos de programação.

Fabricante	Capacidade	Arquitetura	Célula Básica	Método de Programação
Actel	2.000 - 8.000	Gate Array	MUX	Antifuse
Concurrent	3.000 - 5.000	Matriz	XOR, AND	SRAM
Crosspoint	5.000	Gate Array	Transistores	Antifuse
Plessey	2.000 - 40.000	Mar-de-portas	NAND	SRAM
QuickLogic	1.200 - 1.800	Matriz	MUX	Antifuse
Xilinx	2.000 - 10.000	Matriz	RAM	SRAM

De acordo com a tabela acima, vê-se que os FPGAs possuem arquiteturas que permitem uma classificação à parte. Sendo assim, esses dispositivos podem ser classificados em: arranjo simétrico, arranjo em linhas, mar de portas lógicas e PLD hierárquico. A figura 2-15 exemplifica cada classe de arquitetura.

Figura 2-15 Arquiteturas de FPGAs

2.3.4 Arquitetura Cyclone

Uma das famílias de FPGAs de mais baixo custo e que possui uma alta densidade desenvolvida pela Altera é a família de FPGAs Cyclone. Além de possuir uma alta densidade essa família se destaca por possuir um custo mais baixo do que as outras famílias de altas densidades.(ALTERA, *Cyclone Device Handbook*, 2003) A seguir será feita uma breve descrição sobre a arquitetura desta família pois, esta foi utilizada no projeto proposto nesse trabalho.

Primeiramente será feita uma descrição funcional sobre esse dispositivo.

Esta família possui um arranjo simétrico bidimensional - linhas x colunas. Este tipo de arquitetura é baseada em células de armazenamento do tipo SRAM, fabricada com interconexões internas em cobre, o que aumenta muito a velocidade de operação. A arquitetura em questão é similar à da família Stratix, com algumas modificações para reduzir o custo do componente (maior objetivo). A arquitetura Cyclone é baseada em LABs e, cada LAB (*Logic Array Blocks*) contém 10 LEs (*Logic Elements*). As linhas e colunas são interconectadas com velocidade variadas, fornecendo interconexões eficientes entre LABs e LEs. As características das arquiteturas da família Cyclone podem ser observadas na tabela 2-3.

Tabela 2-3 Características das arquiteturas da família Cyclone

CARACTERÍSTICAS	EP1C3	EP1C4	EP1C6	EP1C12	EP1C20
LEs	2.910	4.000	5.980	12.060	20.060
RAM M4K (128x36 bits)	13	17	20	52	64
Total bits (RAM)	59.904	78.336	92.160	239.616	294.912
PLLs	1	2	2	2	2
Pinos I/O	104	301	185	249	301

Os blocos de memórias M4K RAM são agrupados nas colunas, entre alguns LABs. Os pinos de I/O suportam diferentes padrões como o padrão PCI (*Personal Computer Interface*) e o padrão para interface de memória externa DDR (*Double Data Rating*). Essas arquiteturas suportam como modo de configuração os modos Passive Serial, JTAG e Active Serial. A figura 2-16 apresenta o diagrama em blocos da arquitetura Cyclone EP1C12.

Figura 2-16 Diagrama em blocos da arquitetura Cyclone EP1C12

O número de M4K RAM , PLLs (*Phase-Locked Loops*), colunas e linhas (de LABs) variam entre os dispositivos Cyclone. O dispositivo EP1C6, utilizado no projeto, possui 20 blocos de RAM, 32 colunas de LABs e 20 linhas de LABs. A tabela 2-4 mostra os recursos dos dispositivos desta família.

Tabela 2-4 Recursos dos dispositivos da família Cyclone

Dispositivo	M4K RAM		PLLs	Colunas	Linhas
	Colunas	Blocos			
EP1C3	1	13	1	24	13
EP1C4	1	17	2	26	17
EP1C6	1	20	2	32	20
EP1C12	2	52	2	48	26
EP1C20	2	64	2	64	32

Cada LAB (*Logic Array Block*) consiste de 10 LEs (*Logic Element*), LE carry, interconexões locais, LUT e registradores. A estrutura interna de um LAB da Cyclone pode ser vista na figura abaixo.

Figura 2-17 Estrutura de um LAB da Cyclone

As interconexões locais fazem a interconexão dos LEs dentro de um LAB já, as interconexões diretas minimizam o uso de linhas e colunas, permitindo uma maior performance e flexibilidade. Cada LE está diretamente conectado a outros 30 LEs através de uma conexão direta.

Um LE é a menor unidade lógica da arquitetura Cyclone, cada LE contém uma LUT (*Look Up Table*) de quatro entradas, um registrador programável e um carry. Cada LE

possui os seguintes tipos de interconexão: local, linha, coluna, registrador programável e link direto.

Todo o sistema de desenvolvimento para os componentes desta família é gratuito, podendo ser “baixado” da Internet e utilizado sem nenhum custo. A ferramenta utilizada neste projeto é o Quartus II.versão 5.1.

A Cyclone possui o menor preço do mercado para as densidades disponíveis. Seu desempenho chega a ser 75% mais rápido do que produtos de baixo custo similares disponíveis no mercado.

2.3.5 Aplicações de FPGAs

Os FPGAs têm conquistado uma grande aceitação e um grande desenvolvimento nesses últimos anos devido à sua grande aplicação em diversas áreas. Dentre essas aplicações pode-se citar a integração de múltiplos SPLDs, controladores de dispositivos, prototipação de projetos, processamento de sinais (*DSP*), processamento de imagens, processamento de vídeo em tempo real, sistemas de comunicação, entre outros [Vassányi, (1997); Thomas et al. (1999); Chan & Mourad (1994); Brown & Rose (1996); Hauck (1998)].

Capítulo 3

3 Captação, Digitalização e Exibição de Imagens

Monocromáticas

3.1 Considerações Iniciais

Neste capítulo será apresentado o funcionamento de um sistema capaz de realizar a aquisição, digitalização e exibição de imagens monocromáticas. As imagens serão obtidas através de uma câmera monocromática CCD (*Charge-Coupled Device*). O sistema contará ainda com memórias, responsáveis pela etapa de armazenamento das informações digitalizadas do sinal de vídeo, e um CPLD, que será responsável pelo controle de todo o sistema.

3.1.1 O Sinal de Vídeo Composto

Inicialmente é importante o entendimento do sinal de vídeo composto, gerado pela câmera CCD, uma vez que o mesmo apresenta várias características que serão de extrema importância para o sistema a ser implementado.

A adição, ao sinal de vídeo, dos impulsos de extinção horizontal e vertical, dos impulsos de sincronismo vertical e dos dois trens de impulsos de equalização, produz o sinal de vídeo composto (Schure, A.; 1958). O sinal de vídeo utilizado neste trabalho segue as especificações do padrão convencional NTSC (*National Television Systems Committee*). Este padrão foi definido por uma associação entre um comitê, emissoras de TV e fabricantes de receptores, nos EUA, no início dos anos 50, originalmente especificando imagens em níveis de cinza, que deveriam ser transmitidas analogicamente de um ponto a outro. O espectro de altas frequências VHF havia sido dividido em 13 canais, em 1945, pelo US Federal Communications Commission, determinando com isso um tamanho máximo de banda para cada um. Sendo assim, tiveram de ser criadas especificações que fizessem com que a quantidade de informação transmitida coubesse no espaço destinado a cada canal.

Com isso, foi estabelecido que a frequência de troca de quadros (frequência de varredura vertical) na imagem seria de 60 campos por segundo (30 quadros por segundo), igual aos 60Hz (ciclos/seg.) utilizados na corrente elétrica nos EUA, a quantidade de linhas seria de 525 linhas por quadro, ou seja 262,5 linhas por campo, mostradas alternadamente a cada 1/60seg, pois a largura de banda disponível não era suficiente para transmitir uma imagem completa, com todas as linhas 60 vezes por segundo – conceito esse denominado de varredura entrelaçada (2:1). Tendo-se a frequência de varredura vertical (f_v), a frequência de varredura horizontal (f_h) pode ser obtida multiplicando-se o número de linhas por quadro pela frequência de quadros por segundo: $525 \times 30 = 15,75$ KHz. Ou seja, para varrer cada linha horizontal são necessários 63,5 μ s. A unidade de tempo necessária para

varrer uma linha horizontal, incluindo o retorno, é universalmente denominada por “H”. A figura 3-1 ilustra o processo de varredura entrelaçada onde as linhas sólidas correspondem ao campo ímpar e as tracejadas ao campo par.

Figura 3-1 Processo de varredura entrelaçada. De A a B tem-se o traço horizontal. De B a C tem-se o retraço horizontal. De D a E e de F a A tem-se o retraço vertical, enquanto que o traço vertical corresponde à ação de deslocamento do feixe de baixo para cima.

De acordo com a figura acima, o processo de varredura entrelaçada pode ser entendido da seguinte maneira: o processo de deflexão dirige o feixe eletrônico sobre a superfície a ser varrida da esquerda para a direita. Essa trajetória é chamada de *desenho de varredura* ao longo da superfície. Para que o deslocamento do feixe eletrônico ocorra deve-se aplicar à bobina defletora horizontal do cinescópio uma tensão dente de serra adequada. Ao mesmo tempo, o mesmo processo deve ocorrer à bobina de deflexão vertical, para que o feixe possa mover-se de cima para baixo.

Para a varredura do primeiro campo, o feixe eletrônico de varredura da câmera começa a mover-se ao longo da tela. Devido à disposição dos sistemas vertical e horizontal de deflexão e às tensões aplicadas, a posição inicial do feixe é apontada para a margem esquerda da faixa de informação de imagem. A primeira faixa corresponde à linha 1 da superfície a ser varrida. Graças à forma e à amplitude da tensão defletora horizontal, as trajetórias horizontais do feixe descrevem as faixas de números ímpares, omitindo as de números pares. Quando o feixe atinge o lado direito da tela, perfazendo uma linha, ele deve voltar ao lado esquerdo para recomeçar uma nova linha. Durante esse intervalo de tempo, o feixe não deve ser visível portanto, é aplicado à grade da válvula de imagem um pulso de apagamento horizontal, que é mais preto que o preto. Nesta primeira etapa, o feixe de elétrons cobre 247,5 linhas ativas e o retorno vertical desse campo cobre 15 linhas inativas que permanecerão apagadas no monitor.

Uma vez varridas as linhas ímpares da imagem, chega-se à extremidade final da superfície. A informação de imagem, tal como foi produzida durante a varredura das faixas ímpares é transmitida aos receptores ponto por ponto. Com isso, tem-se que a varredura completa das linhas ímpares da imagem é a varredura de um campo (campo ímpar).

Após completar o primeiro campo, o feixe de varredura volta à parte superior do alvo, no qual existe a imagem completa. Agora, porém, graças à posição de partida do feixe de varredura, as faixas pares é que serão varridas, para fornecerem a informação de vídeo, enquanto que as de número ímpares serão puladas. Igualmente ao que ocorreu com o primeiro campo, o feixe cobre um total de 262,5 linhas ($247,5 + 15$).

Ao terminar de varrer as linhas pares o segundo campo fica completo (campo par). Igualmente ao primeiro campo, a informação fornecida ponto a ponto é transmitida como um trem de sinais. Com o segundo campo transmitido, a figura visível na tela do monitor é a imagem completa, tal como se apresentava inicialmente no alvo da válvula da captação de imagem.

Na reconstituição da imagem as faixas pares do segundo campo estão intercaladas entre as ímpares, que formavam o primeiro campo. O espaço para a intercalação ficou reservado pelas correntes que atuaram no sistema de deflexão.

Observa-se que o processo de varredura não altera a localização das faixas de informação de imagem; apenas selecionam, na câmera, as faixas ímpares para o primeiro campo e as pares para o segundo. Sendo assim, fica claro que a imagem é enviada em duas partes.

É interessante notar que só se vê a figura completa quando aparecem os dois campos consecutivos intercalados. Quando esses dois campos são transmitidos diz-se que foi transmitido um quadro.

Para que esse tipo de varredura possa se realizar, é necessário que a frequência de varredura horizontal seja 262,5 vezes maior que a vertical, assegurando que a proporcionalidade entre as frequências seja um número fracionário de modo a garantir meia linha no final e no início de campos consecutivos.

Voltando ao sinal de vídeo, sabe-se que sua forma pura traz informações referentes apenas à luz e sombra de uma imagem e, que estas informações são insuficientes. Se um feixe eletrônico descreve uma linha em uma imagem, como mostrada na figura 3-1, as informações da imagem são recolhidas e, a amplitude desta está situada entre o nível de preto e o nível de branco. A tensão correspondente à determinada linha horizontal começa no ponto em que o feixe começa a varrer a imagem horizontalmente e termina aonde termina a varredura e começa o retorno horizontal de extinção. Considerando que a ação de varredura é contínua (linha por linha), tem-se que o pré-amplificador de vídeo, na câmera, estará recebendo um trem de tensões de vídeo com intervalos de extinção intercalados. Esse intervalo de extinção ocorre quando o feixe chega à extremidade de uma linha e precisa retornar ao começo da linha subsequente para recomençar a varredura. Porém, no final da linha a intensidade luminosa do ponto explorado pode ser qualquer, inclusive a mais clara. Para que o feixe recomece a varredura, é necessário acrescentar aos sinais de vídeo e extinção o impulso de sincronismo horizontal. Este impulso é muito breve, com uma duração de apenas $5\mu\text{s}$. Sua polaridade é a mesma do impulso de extinção horizontal, permitindo assim a separação do sinal de vídeo no receptor. O tempo de sua aparição é tal que o impulso coincide com a chegada do feixe no fim da varredura horizontal para a direita e força o sistema da geração de deflexão horizontal a iniciar a porção de retorno do ciclo de varredura horizontal. Em outras palavras, o impulso de sincronismo horizontal aparece no fim de cada varredura horizontal. Esse impulso ocorre na região abaixo do nível de preto, conhecida como mais preto que o preto. A figura 3-2 mostra o formato de uma linha de sinal de vídeo juntamente com os pulsos de extinção e de sincronismo horizontal. O nível de preto corresponde a uma tensão de 0 volt e o nível de branco tem amplitude de

0,7 volts. Os níveis de cinza situam-se entre esses dois extremos. O impulso de sincronismo possui uma amplitude de $-0,3$ volts e, este impulso é programado para aparecer ligeiramente após o início do impulso de extinção horizontal de $10\mu\text{s}$, que surge no fim de cada varredura horizontal. Com duração de apenas $5\mu\text{s}$, o impulso de sincronismo termina antes do impulso de extinção. Conseqüentemente, o impulso de sincronismo ocorre dentro do período de duração do impulso de extinção. Uma vez que a amplitude do impulso de sincronismo é maior que a do impulso de extinção, aquele primeiro estende-se além da base do impulso de extinção, posição em que ambos aparecem reunidos, tornando possível a retirada do impulso de sincronismo e do impulso de extinção, e usá-lo como um sinal de controle de tempo.

Figura 3-2 Formato de uma linha de sinal de vídeo composto

Quando o feixe termina a varredura da última linha da parte inferior da imagem o mesmo deve ser apagado e levado à parte superior da tela. Nesse instante são aplicados pulsos de extinção e de sincronismo vertical. O impulso de sincronismo vertical é aplicado ao gerador de deflexão vertical, onde provoca a repentina inversão da direção da corrente

de deflexão vertical. Com isso tem início o movimento do feixe de varredura para cima em determinado momento do ciclo de varredura completo.

O retorno vertical do feixe de varredura ocorre uma vez em cada campo: é quando o feixe, tendo varrido a última linha horizontal, está pronto para voltar à margem superior da tela que está sendo explorada, começando a varredura do campo seguinte. O impulso de sincronismo vertical aparece em cada intervalo de $1/60$ segundos, ou 60 vezes por segundo. O intervalo do impulso é de $190\mu s$, podendo-se dizer que é um impulso de longa duração, em relação ao impulso horizontal. Os impulsos de sincronismo vertical são precedidos e sucedidos por 6 impulsos de duração menor, denominados de impulsos equalizadores. Os impulsos equalizadores possuem a mesma polaridade do impulso de sincronismo vertical. São formados de modo que o intervalo entre dois impulsos consecutivos corresponda a um ciclo de varredura horizontal, fornecendo assim, o equivalente a três impulsos de sincronismo horizontal imediatamente antes de aparecer o retorno vertical e outros três impulsos de sincronismo horizontal imediatamente depois do impulso de sincronismo vertical. Assim, mantém em frequência o gerador de deflexão horizontal durante o período em que o gerador de deflexão vertical se apronta para a inversão da corrente de deflexão; dando início ao retorno vertical.

A necessidade de um intervalo de espera provém do fato de cada campo conter uma meia linha. O instante de partida do retorno vertical de cada um dos campos, de linhas ímpares e pares, difere de uma meia linha horizontal. A última dessas linhas no campo ímpar termina no meio da margem inferior da superfície sujeita a varredura, completando a

última linha horizontal do campo par, que termina no canto direito da margem inferior. Os impulsos equalizadores em cada campo encarregam-se da diferença de tempo e dão partida ao retorno vertical dos campos sucessivos nos momentos próprios, de modo a se realizar a correta intercalação das linhas de varredura horizontal.

O formato final do sinal de vídeo composto padrão é ilustrado na figura 3-3.

Figura 3-3 Formato do sinal padrão de vídeo composto

3.2 Circuito de Captação, Digitalização, e Exibição de Imagens

Monocromáticas

Nesta seção será descrito como foi desenvolvido o circuito de captação, digitalização, e exibição de imagens monocromáticas. É importante ressaltar que, primeiramente, o circuito foi montado e testado em um “*proto-board*”. Isto foi feito para que cada parte fosse testada separadamente, facilitando a depuração e ajustes do circuito.

3.2.1 Captação, Digitalização e Exibição de Imagens Monocromáticas

O diagrama em blocos do módulo de captação, digitalização e exibição de imagens monocromáticas está ilustrado na figura 3-4.

Figura 3-4 Diagrama em blocos do circuito de captação, digitalização e exibição de imagens monocromáticas.

De acordo com o diagrama em blocos apresentado acima, pode-se descrever a função que cada bloco executa. O *hardware* desenvolvido para esta etapa do projeto constitui-se de dois módulos; um módulo de aquisição, digitalização e exibição e um outro de armazenamento e controle, que será descrito mais adiante. No apêndice A tem-se o esquemático do circuito referente ao módulo de aquisição, digitalização e exibição de imagens monocromáticas.

O circuito do módulo de aquisição, digitalização e exibição, gerado pela ferramenta Orcad, pode ser visto no apêndice D.

A seguir, serão descritas as características, e suas respectivas funções, de cada componente que compõe o circuito apresentado na figura 3-4.

3.2.1.1 Separador de Sincronismo

O separador de sincronismo utilizado foi o EL4581C da Elantec, cuja função é extrair as informações de sincronismo vertical, horizontal, póstico posterior e de tipo de campo de um sinal padrão de vídeo composto. Estes sinais serão importantes porque através deles a unidade de controle (CPLD) poderá fazer a identificação do início de uma nova linha de vídeo ou de um novo campo, gerando os sinais de controle e os endereços para as memórias. A figura 3.5 mostra os sinais de saída do EL4581C, quando um sinal de vídeo composto padrão é aplicado na sua entrada.

Figura 3-5 Sinais extraídos do separador de sincronismo. Em (a), tem-se o sinal de entrada (vídeo composto padrão). Em (b), tem-se o sinal de sincronismo composto (cs). Em (c), tem-se o sinal de sincronismo vertical (vs). Em (d), tem-se o sinal de tipo de campo (par/ímpar) que será baixo quando o campo for par e alto quando o campo for ímpar. E em (e), tem-se o sinal de pórtico posterior (bp).

3.2.1.2 Conversor A/D

O conversor A/D utilizado nesse projeto é o TLC5540 de 8 bits de resolução com uma taxa de conversão de 40MSPS (Texas Instruments, 2004). A escolha desse conversor deu-se devido às características do sinal de vídeo. Sabe-se que a informação de uma linha de vídeo é de aproximadamente $53\mu\text{s}$ e, para se obter 512 amostras nesse tempo é necessário que o conversor seja capaz de realizar a conversão em aproximadamente 100ns, o que equivale a 10 milhões de amostras por segundo. Na figura 3-6 é apresentado o diagrama de tempo do conversor A/D TLC5540.

Figura 3-6 Diagrama de tempos do conversor A/D TLC5540. O sinal é amostrado na borda de descida do clock e os dados ficam disponíveis a partir da quarta borda de subida do clock

3.2.1.3 Buffer

O *buffer* é uma unidade intermediária entre dois ou mais componentes, neste caso o *buffer* está entre o conversor A/D e o conversor D/A e memórias. Internamente, o *buffer* contém uma área de memória que é utilizada para armazenamento temporário de informações. Normalmente um *buffer* oferece operações adicionais que permitem consultar se existem dados disponíveis para leitura, no *buffer* de entrada, ou para solicitar que os dados presentes no *buffer* de saída sejam enviados imediatamente para o dispositivo de saída.

O *buffer* utilizado nesse circuito foi o SN74LV244AT (Texas Instruments, 2004).

3.2.1.4 Conversor D/A

O conversor D/A que foi utilizado é o TLC5602C de 8 bits de resolução com uma taxa de operação de 30MSPS (Texas Instruments, 2002).

Dentre as condições de operação recomendadas pelo fabricante o conversor TLC5602C apresenta uma tensão de referência analógica (V_{ref}) de valor nominal igual a 4V. Para a obtenção dessa tensão de referência, fez-se uso de um regulador de tensão. Este regulador de tensão foi projetado para que sua saída apresentasse uma tensão de 4V enquanto sua entrada estivesse recebendo a tensão de alimentação padrão do sistema, 5V. O regulador de tensão utilizado foi o LM317LZ (National Semiconductor, 2000). O projeto do regulador junto ao conversor pode ser visto no apêndice A.

O diagrama de tempos do conversor D/A pode ser observado na figura seguinte.

Figura 3-7 Diagrama de tempo do conversor D/A TLC5602

3.2.1.5 Saída de Vídeo

Nesta etapa final o sinal de vídeo composto é reconstruído, utilizando os dados do conversor D/A e do separador de sincronismo.

A foto do circuito desenvolvido para esta etapa pode ser observada no apêndice D.

Capítulo 4

4 Armazenamento de Imagens Monocromáticas

4.1 Controle para Armazenamento de Imagens Monocromáticas

O armazenamento de imagens monocromáticas será realizado em duas memórias e, a Unidade de Controle I, será a responsável pelo “gerenciamento” dessa etapa. A seguir, será descrito o funcionamento, de forma detalhada, do módulo responsável pelo controle e armazenamento de imagens monocromáticas. A figura 4-1, ilustra esse módulo e seus componentes.

Figura 4-1 Módulo de armazenamento e controle de imagens monocromáticas

4.1.1 Memórias

As memórias fazem parte do módulo de armazenamento e controle do sistema. Como as memórias utilizadas são SMD (*Surface Mount Devices*), desenvolveu-se um soquete para que fosse possível a utilização das mesmas nos dois módulos de armazenamento e controle que foram desenvolvidos. Assim, não seria necessário que as memórias fossem soldadas às placas, o que restringiria a sua utilização.

Uma vez que o sinal de vídeo é entrelaçado, será necessário o uso de duas memórias para armazenar um quadro da imagem. Uma das memórias armazenará o campo par e a outra o campo ímpar.

A memória adotada foi a SRAM BS62LV1024SC-70. Esta memória é de alta performance e de baixo consumo; possui um tempo de acesso de aproximadamente 70ns e é organizada com 131072 palavras de 8 bits (128K x 8). A figura 3-7 mostra os diagramas de tempo para os ciclos de leitura e escrita (Brilliance Semiconductor, INC; 2001). O controle dos endereços das memórias será gerado pela unidade de controle, respeitando os tempos dos diagramas da figura 4-2 onde, as siglas OE, CE, WE, Dout e Din representam, respectivamente, os sinais “*Output Enable*”, “*Chip Enable*”, “*Write Enable*”, “*Data Out*” e “*Data In*”.

(a) Ciclo de leitura

(b) Ciclo de escrita

Figura 4-2 Diagrama de tempo da memória. (a) ciclo de leitura, (b) ciclo de escrita

4.1.2 Unidade de Controle

A unidade de controle será implementada utilizando o CPLD EPM7128SLC84-7 da família MAX 7000S da Altera. Para a programação do CPLD será utilizado o software Quartus II, também da Altera. De acordo com (PEDRINO, E. C., 2003), esta unidade será

composta por outras sub-unidades, tais como: unidade de exclusão de pulsos de sincronismo, unidade de controle interna e unidade de controle das memórias.

O software Quartus II (e MAX+plus II) apresenta uma hierarquia para representação de projetos. Sendo assim, a unidade de controle está representada o topo dessa hierarquia e, suas sub-unidades, estão representando os níveis mais baixos, que não são visíveis até que o projeto seja compilado ou simulado.

De acordo com o projeto desenvolvido, o componente padrão - *unidade de controle* – representando o topo da hierarquia, será o responsável pelo armazenamento dos quadros da imagem nas memórias SRAM. A figura 4-3 mostra o referido componente.

Figura 4-3 Unidade de controle

A seguir, na figura 4-4 tem-se o diagrama esquemático da “unidade de controle”.

Por este diagrama, é possível identificar todos os blocos lógicos que o constituem os quais, serão descritos com mais detalhes nas seções que se seguem.

Todas as subunidades foram implementadas utilizando a linguagem de programação desenvolvida pela Altera, AHDL (Altera HDL). A seguir, será apresentado cada bloco lógico que constitui a unidade de controle e, será feita uma breve descrição das funções desenvolvidas por cada bloco.

4.1.2.1 Unidade de Exclusão de Pulsos de Sincronismo

A unidade de exclusão de pulsos de sincronismo tem como objetivo a exclusão dos pulsos equalizadores e serrilhados fornecidos pelo sinal de sincronismo composto (*cs*). Esta unidade está representada pelo bloco lógico “*exclusao_pulsos_sinc*”, representada na figura 4-5. Este bloco lógico é composto por um contador e por uma máquina de estados. O contador é responsável pela contagem dos pulsos de sincronismo logo após o início de cada campo da imagem e, a máquina de estados, por sua vez, é responsável por enviar um sinal à unidade de controle interna (que será vista mais adiante), indicando que os pulsos equalizadores e serrilhados foram excluídos.

Figura 4-5 Bloco lógico da unidade de exclusão de pulsos equalizadores e serrilhados

O diagrama esquemático desta unidade está representado na figura 4-6 e, os códigos desenvolvidos em AHDL, podem ser vistos no apêndice H.

Figura 4-6 Diagrama esquemático da unidade de exclusão de pulsos equalizadores e serrilhados

4.1.2.2 Unidade de Controle Interna

A unidade de controle interna possui uma função responsável pela geração de um sinal de disparo que vai habilitar as memórias para as operações de leitura e escrita. É

importante ressaltar que as memórias ficarão habilitadas apenas nos períodos de linha ativa do sinal de vídeo.

O bloco lógico que representa esta unidade é o bloco “*unid_contr_int*”, mostrado na figura 4-7. Este bloco é composto por um contador de pulsos e por dois *flip-flops* tipo D.

O código, em AHDL, da unidade de controle interna pode ser visto no apêndice H.

Figura 4-7 Bloco lógico da unidade de controle interna

4.1.2.3 Unidade de Controle das Memórias

Por último tem-se a unidade de controle das memórias, que é responsável pela geração dos endereços para as mesmas e pelo controle de desabilitação no final de cada linha. Esta unidade é composta por dois contadores e por dois *flip-flops* tipo D.

Um dos contadores será responsável pela contagem dos 512 *pixels* de cada linha da imagem e, o outro contador, contará as 256 linhas de cada campo da imagem. É importante ressaltar que a cada final de contagem, tanto do contador de *pixels* (“*contador_pixels*”) quanto do contador de linhas (“*contador_linhas*”), um sinal de controle, denominado

“*fim_pixel*” e outro denominado “*fim_linha*” será enviado, “dizendo” que ambas as contagens atingiram seus valores máximos. Estes sinais serão importantes para o controle da etapa de integração das linhas e colunas do sinal de vídeo; etapa esta que será explicada adiante com mais detalhes.

Na figura 4-8 está representado o bloco lógico “*controle_memorias*” enquanto, que na figura 4-9 tem-se o diagrama esquemático da unidade de controle das memórias.

Os códigos dos contadores, escritos em AHDL, podem ser vistos no apêndice H.

Figura 4-8 Bloco lógico da unidade de controle das memórias

Figura 4-9 Diagrama esquemático da unidade de controle das memórias

4.2 Considerações Finais

Como dito anteriormente, o CPLD e as memórias foram projetados separadamente da unidade de digitalização do sinal de vídeo. Um dos motivos pelo qual se optou por esta alternativa foi pela possibilidade de se utilizar um dispositivo de controle mais denso do que o CPLD EPM7128SLC84-7, caso fosse necessário pois, dependendo da lógica o CPLD poderia ser insuficiente.

Para a confecção de um novo módulo de armazenamento e controle fez-se uso de um FPGA EP1C6Q240CB, pertencente à família Cyclone II da Altera. O diagrama

esquemático está representado no apêndice C e, módulo contendo o FPGA pode ser visto no apêndice F.

Além dos dois módulos desenvolvidos um terceiro foi projetado contendo dois CPLDs e duas memórias. O intuito ao projetar esse módulo foi o de utilizar um CPLD para controle de armazenamento de imagens e o outro para controle da etapa de integração do sinal de vídeo.

O diagrama esquemático e o circuito do módulo contendo os CPLDs, gerado pelo Orcad, podem ser vistos nos apêndices B e E respectivamente.

Com base no que foi proposto, um diagrama em blocos contendo os dois módulos é apresentado na figura 4-10. O circuito, contendo os dois módulos pode ser visto no apêndice G.

Figura 4-10 Diagrama em blocos dos dois módulos propostos

Capítulo 5

5 Integração do Sinal de Vídeo

Como dito, no capítulo anterior, o sistema de detecção de movimento será composto por um circuito responsável pela captação, digitalização e exibição de imagens monocromáticas, por um circuito que realizará o controle e armazenamento e por um outro responsável pela integração do sinal de vídeo.

Neste capítulo, será feita uma descrição detalhada da etapa de integração do sinal de vídeo tanto na direção horizontal como na vertical.

5.1 Considerações Iniciais

De acordo com a figura 3-5, os sinais fornecidos pelo separador de sincronismo, “*cs*” (sincronismo composto) e “*odd/even*” (campo par e campo ímpar) serão importantes pois, através destes, será possível distinguir qual campo da imagem está sendo adquirido (par ou ímpar) e também controlar o início e o final do período de integração de cada linha de vídeo (sincronismo composto).

Para a detecção de movimento em apenas uma direção pode-se utilizar a informação média de cada linha da imagem (imagem = 525 linhas). Se for considerado que a imagem de vídeo é composta por linhas horizontais e, processando a informação média de cada linha, ao invés da informação pontual, será possível detectar variações na direção perpendicular às linhas horizontais, após uma comparação das informações de quadros sucessivos (não necessariamente sequenciais), correspondendo a um deslocamento de objetos no sentido vertical da imagem. A figura 5-1 ilustra o caso.

Figura 5-1 Quadros sucessivos de imagem com o deslocamento de um objeto. À direita de cada quadro aparece o valor médio de cada linha

Para a detecção de movimento no sentido horizontal, que normalmente corresponde a maior parte dos movimentos que ocorrem na cena, a integração será realizada nas linhas da imagem correspondentes à posição vertical dos objetos da cena. A figura 5-2 ilustra o deslocamento descrito.

Figura 5-2 Quadros sucessivos de imagem com o deslocamento de um objeto. Abaixo de cada quadros aparece o valor médio do sinal de linha

O diagrama em blocos apresentado na figura 5-3 apresenta, de forma simplificada, a operação de integração que será realizada no sinal de vídeo obtido através da câmera CCD.

Figura 5-3 Diagrama em bloco da etapa de integração do sinal de vídeo

5.1.1 Integração do Sinal de Vídeo na Direção Horizontal

Como dito no tópico anterior, a integração será realizada nas direções horizontal e vertical. Primeiramente, será descrito o método desenvolvido para a integração (soma) dos *pixels* na direção horizontal, ou seja, será realizada uma soma de todos os *pixels* que uma linha ativa de vídeo contém (512 no total).

O programa que realizará esta operação foi desenvolvido no software Quartus II e o bloco lógico padrão recebeu o nome de “*somador_pixels_horizontais*” e pode ser visto na figura 5-4.

Figura 5-4 Bloco lógico do somador de *pixels* horizontais

O diagrama esquemático, referente ao bloco “*somador_pixels_horizontais*”, está representado na figura 5-5.

Figura 5-5 Diagrama esquemático do somador de *pixels* horizontais

De acordo com o diagrama esquemático, para que a soma dos *pixels* nas linhas horizontais fosse realizada fez-se uso de três registradores de oito bits, constituídos por *flip-flops* tipo D, um acumulador e um *latch*. O acumulador possui entrada para oito bits de dados e saída pra dezessete bits (figura 5-5). O intuito para a utilização de uma saída com dezessete bits no acumulador é para que durante a soma truncada os bits mais significativos sejam mantidos, caso que não aconteceria se a saída do acumulador fosse de apenas oito bits. Uma vez que os dados da saída do acumulador já estejam disponíveis, pode-se então, trabalhar com os oito bits mais significativos, os quais serão suficientes para a análise dos

resultados. Sendo assim, os demais componentes podem ser ajustados para operarem com oito apenas bits de resolução (os mais significativos).

Para o controle desse bloco, foi necessária a utilização de alguns sinais fornecidos pela unidade de controle (CPLD I). Estes sinais são: *clock*, representado pelo sinal “*saída [0]*”, fim da contagem de *pixels*, representado pelo sinal “*fim_pixel*”, leitura das memórias, representado pelo sinal “*contr_le*” e pelos dados, obtidos da leitura das memórias, representado pelo sinal “*dados [7..0]*”.

O primeiro *flip-flop*, denominado “*ff_horiz1*”, tem como função segurar os dados que chegam das memórias por um período de um ciclo de *clock* e, só então esses dados são liberados (borda de subida do *clock*) para o acumulador, denominado “*acum_horizontal*”. Uma vez estando com os dados na entrada, o acumulador começa a operar sincronizado com o *clock*. Inicialmente seu valor é zero e, a partir do primeiro ciclo de *clock*, a cada borda de subida do mesmo, sua saída apresenta o valor da soma do dado anterior com o atual, caracterizando uma soma truncada dos dados, que nesse caso são os valores dos *pixels*. Este acumulador possui dois sinais de controle: um deles permite que o mesmo seja habilitado para a operação (*clken* = *clock enable*) e, um outro sinal permite com que o conteúdo do acumulador seja “zerado” quando este sinal estiver em nível alto. Sendo assim, faz-se uso dos sinais “*contr_le*”, para que o componente em questão seja habilitado apenas no momento em que os sinais de controle das memórias habilitam a leitura de dados nas portas das mesmas e “*fim_pixel*”, que “limpará” o acumulador a cada final de contagem de *pixels* da linha ativa de vídeo. O segundo *flip-flop*, denominado “*ff_horiz2*”, realiza a mesma tarefa do primeiro, retendo os dados por um período de um ciclo de *clock*. Já, o

registrador denominado “*reg_horiz*”, recebe os dados do segundo *flip-flop* e libera apenas o último dado ou seja, o dado que é o resultado da soma truncada dos *pixels* da referida linha de vídeo. Devido às várias etapas dessa operação, valores indesejados podem aparecer atrapalhando os resultados, por isso foi necessária a utilização de vários flip-flops. O último *flip-flop*, denominado “*ff_horiz3*”, armazena somente o valor desejado e então pode enviá-lo a um dispositivo de armazenamento, como por exemplo as memórias SRAMs utilizados nesse mesmo projeto.

Algumas das simulações realizadas para o desenvolvimento dessa etapa, assim como os resultados obtidos utilizando um analisador lógico poderão ser vistas no capítulo referente aos testes e resultados.

5.1.2 Integração do Sinal de Vídeo na Direção Vertical

A integração do sinal de vídeo na direção vertical requer uma atenção especial na etapa de leitura dos dados a serem somados. Isso se deve ao fato de que quando a soma é feita na direção horizontal (tópico anterior) os dados podem ser lidos e somados diretamente conforme sua ordem de chegada. Para exemplificar melhor pode-se imaginar uma imagem, como sendo uma matriz 4x4, com seus valores de *pixels* (em binário), mostrados conforme a figura 5-6.

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Figura 5-6 Matriz 4x4, com os valores dos bits em binário

Para a etapa de integração dos *pixels* na direção horizontal, os dados que estão armazenados nas memórias são lidos e enviados ao primeiro *flip-flop* (vide figura 5-5) de maneira seqüencial ou seja, os dados são enviados conforme mostrado na figura 5-7.

Figura 5-7 Leitura seqüencial dos bits da matriz 4x4

Sendo assim, a leitura e conseqüentemente a integração (soma) pode ser feita de maneira seqüencial, linha por linha.

Para o segundo caso, a integração terá de ser feita nas colunas da imagem. Em outras palavras, os *pixels* a serem somados serão os primeiros *pixels* de cada linha a seguir serão somados os segundos *pixels* de cada linha e assim por diante, até completar o total de linhas do campo da imagem e *pixels* das linhas (256 linhas por campo e 512 *pixels* por linha). Para que essa operação fosse realizada fez-se o uso de um recurso que torna a matriz da imagem em uma matriz transposta ou seja, as linhas da imagem original são trocadas

pelas colunas e vice-versa. Para exemplificar de uma melhor maneira imagine a mesma matriz 4x4 utilizada anteriormente. A matriz transposta 4x4, da matriz mostrada na figura 5-6, é apresentada na figura 5-8.

0000	0100	1000	1100
0001	0101	1001	1101
0010	0110	1010	1110
0011	0111	1011	1111

Figura 5-8 Matriz 4x4 transposta

Desse modo, os dados podem ser lidos de modo seqüencial, como no exemplo anterior porém, os valores dos *pixels* serão referentes aos valores dos *pixels* das colunas da imagem. A figura 5-9 mostra como os dados chegam ao primeiro *flip-flop* e, logo depois ao acumulador.

Figura 5-9 Leitura seqüência dos bits da matriz 4x4 transposta

Para que fosse possível a leitura dos dados na forma transposta realizaram-se alguns testes para se observar como os valores seriam apresentados para a integração. Para tal fez-se uso de dois contadores: um contador de oito bits e um outro de nove bits. A maneira

como os contadores foram interligados pode ser observada na figura 5-10, aonde o contador denominado “*cont_vert*” é alimentado com os endereços mais significativos, gerados pelo bloco lógico “*controle_memorias*”. Quando a contagem do contador “*cont_vert*” atingir seu valor máximo a contagem estoura e um bit “*carry out*” é enviado ao segundo contador, “*cont_vert1*”, como sendo “*carry in*”, complementando assim a contagem do segundo contador. Com isso, a concatenação das saídas dos dois contadores fornece os novos endereços para que os dados possam ser lidos das memórias de forma transposta.

Figura 5-10 Contadores para a transposição dos endereços das memórias

Para aproveitar as mesmas linhas de endereços, utilizadas na etapa de integração das linhas horizontais, optou-se por alocar a etapa de transposição dos endereços no mesmo bloco lógico “*controle_memorias*”. Sendo assim a mesma saída do bloco lógico poderá ser usada para a leitura dos dados tanto na forma normal quanto na forma transposta. Para realizar essa tarefa utilizou-se um multiplexador de duas entradas cada uma com dezessete bits de resolução. Para que a seleção das entradas pudesse ser realizada nos tempos adequados, utilizou-se um contador de dezessete bits. Ao término da contagem desse

Uma vez que a saída do multiplexador endereça as memórias com os valores transpostos, os dados a serem integrados são lidos das memórias e então enviados ao CPLD responsável pela etapa de integração. No entanto, um novo bloco lógico foi desenvolvido de modo que este seja responsável apenas pela integração do sinal de vídeo na direção vertical. O bloco lógico referente a esta etapa pode ser visto na figura 5-12.

Figura 5-12 Bloco lógico do somador de *pixels* verticais

O princípio de operação do bloco lógico “*somador_pixels_verticais*” é idêntico ao da etapa de soma dos *pixels* horizontais. Este bloco lógico também apresenta três *flip-flops*, um acumulador e um registrador, todos com oito bits de resolução.

A figura 5-13 apresenta o diagrama esquemático do bloco lógico “*somador_pixels_verticais*”.

Figura 5-13 Diagrama esquemático do somador de *pixels* verticais

Capítulo 6

6 Testes e Resultados

6.1 Considerações Iniciais

Neste capítulo serão apresentados alguns resultados obtidos através do circuito implementado.

Devido às várias etapas do projeto serão apresentadas também as simulações de algumas delas. Todas as simulações foram feitas utilizando os softwares Max-Plus II 9.4 e, Quartus II 5.1, ambos da Altera. Essas simulações servirão como base para comparações com os resultados obtidos do circuito implementado. Para a obtenção dos dados do circuito implementado, utilizou-se o analisador lógico da empresa Tektronix, modelo TLA613. O arranjo, utilizado para a obtenção dos dados, está mostrado na figura 6-1.

Figura 6-1 Hardware implementado para obtenção dos dados

6.1.1 Materiais

Para a montagem de todo o hardware proposto, assim como o suporte para o funcionamento do mesmo, foram necessários os seguintes recursos:

- Uma câmera monocromática CCD;
- Kit da Altera (University Program UP2 Development Kit);
- Placa de prototipagem do tipo Proto Board;

- Osciloscópio;
- Fontes de alimentação;
- Analisador Lógico Tecktronix, modelo TLA 613;
- Microcomputador, para simulações e programação do CPLD;
- Ferramentas de desenvolvimento MAX + Plus II e Quartus II, da empresa Altera.

6.1.2 Simulação da unidade de controle

A simulação da unidade de controle, descrita no capítulo 4, pode ser observada a seguir, para um caso hipotético de um sinal de vídeo com cinco linhas.

Figura 6-2 Simulação da Unidade de Controle

Como pode ser visto na figura 6-2, a simulação da unidade de controle foi realizada no software MAX + Plus II 9.4. Sendo assim, observa-se que os canais de entrada “*clk*”, “*sinc_comp*” e “*sinc_vert*”, que representam os sinais de *clock*, sincronismo composto e sincronismo vertical, respectivamente, estão de acordo com os sinais que serão transmitidos ao CPLD, pelo separador de sincronismo. Com isso, alguns sinais de saída como por exemplo, o sinal “*start*”, que representa o início das linhas ativas de vídeo, pode ser visto de maneira clara na simulação. Além do mais, o sinal “*linha_ativa*”, que representa as cinco linhas de vídeo, tem seu início na borda de descida do sinal “*start*”. Outros sinais podem ser observados pela simulação, como é o caso dos sinais “*fim_pixel*”, que apresenta um pulso a cada contagem de 512, e os sinais de endereçamento das memórias representados pelos nomes “*out[16..0]*” e “*endereços_transpostos[16..0]*”.

6.1.3 Simulação e Teste do somador de *pixels* horizontais

As simulações dessa etapa foram realizadas no software Quartus II 5.1 e, os resultados podem ser observados nas figuras que se seguem.

Figura 6-3 Simulação do somador de *pixels* horizontais com dezessete bits de resolução

Observando a figura 6-3, nota-se que como a saída do “*acum_horiz*” é de dezessete bits, quando a contagem ultrapassa o valor de 256 (oito bits de dados), os bits mais significativos não são perdidos, o que não aconteceria se a saída do acumulador fosse de oito bits. Foram utilizados dezessete bits de resolução devido aos números de *pixels* por linha, $2^9 = 512$, e à quantidade de linhas da imagem, $2^8 = 256$. Ainda observando a figura 6-3, para um valor arbitrário da entrada “*fim_pixel*”, quando ocorre um pulso nesse sinal, indicando que a contagem de *pixels* de uma linha de vídeo atingiu seu valor máximo, o acumulador horizontal é zerado e, concomitantemente o bloco lógico “*reg_horiz*” libera apenas o último valor do acumulador horizontal. Para garantir que apenas esse dado seja capturado, sem a presença de “lixos” um último bloco lógico denominado “*ff_horiz3*” é utilizado liberando assim, apenas o valor de interesse.

Como dito anteriormente, o acumulador horizontal possui uma saída de dezessete bits no entanto, apenas os oito bits mais significativos são de interesse, pois com essa

resolução consegue-se a representação de qualquer imagem em tons de cinza . Sendo assim realizaram-se mais duas simulações. Uma apresenta os oito bits mais significativos apenas na saída do último bloco lógico (“*ff_horiz3*”), e a outra realiza as operações com os oito bits mais significativos a partir da saída do acumulador horizontal (“*acum_horiz*”). Ambas as simulações podem ser vistas nas figuras 6-4 e 6-5 respectivamente.

Figura 6-4 Simulação do somador de *pixels* horizontais. Os oito bits mais significativos estão representados no canal S

Figura 6-5 Simulação do somador de *pixels* horizontais com os oito bits mais significativos

Os testes realizados com o hardware apresentaram resultados que serão mostrados e descritos a seguir. Lembrando que, para a obtenção desses resultados, utilizou-se um analisador lógico de cento e dois canais da Tektronix, modelo TLA613.

A figura 6-6 apresenta o resultado obtido dos valores dos dados armazenados nas memórias (canal A0) e os mesmos dados capturados na saída do bloco lógico denominado “*ff_horiz1*” (canal A1). O intuito dessa etapa foi mostrar que os dados que serão enviados ao acumulador horizontal (“*acum_horiz*”) estão livres de ruído, o que poderia prejudicar na operação de soma. O canal CK1 está representando o *clock* do circuito.

Figura 6-6 Clock do circuito (CK1), dados obtidos da memória (A0) e dados obtidos na saída do “ff_horiz1” (A1)

A figura 6-8 está representando os dados obtidos diretamente da memória (canal A0) os dados obtidos da saída do “*ff_horiz1*” e também um histograma da imagem filmada que, no caso, era uma imagem com um fundo claro homogêneo e uma figura quadrada, como apresentada na figura 6-7.

Figura 6-7 Imagem filmada para obtenção do histograma

Figura 6-8 Dados da memória (A0), dados do "ff_horiz1" e histograma da imagem

As figuras 6-9 e 6-10 representam, respectivamente, a imagem filmada com o objeto deslocado na cena e os dados obtidos da memória, da saída do "ff_horiz1" e o histograma da imagem filmada.

Figura 6-9 Imagem com o objeto deslocado na cena

Figura 6-10 Dados da memória (A0), dados do "ff_horiz1" (A1) e histograma da imagem com o objeto deslocado na cena

A figura 6-11 apresenta, no canal CK1 o *clock* do circuito, no canal A0 os dados obtidos diretamente das memórias (durante o ciclo de leitura das mesmas), no canal A1

estão representados os dados obtidos na saída do “ff_horiz1” e, por último, no canal A2 estão representados os oito bits mais significativos resultantes da saída do acumulador horizontal (“s_acum [16..9]”, vide figura 5-5). Os dados obtidos da saída do acumulador serão enviados a outros blocos lógicos, que serão descritos mais adiante.

Figura 6-11 Clock do circuito (CK1), dados das memórias (A0), dados na saída do "ff_horiz1" (A1) e oito bits mais significativos do resultado da saída do acumulador horizontal (A2)

A figura 6-12 apresenta a mesma situação mostrada na figura 6-10 porém com os dados apresentados em forma de lista e em forma de ondas.

Figura 6-12 Dados amostrados em forma de ondas e lista

Uma vez que os valores da soma truncada foram obtidos, lembrando que apenas os oito bits mais significativos estão sendo levados em conta, esses valores são enviados a um registrador de oito bits (constituído por oito *flip-flops* tipo D) denominado “*ff_horiz2*” cuja função é armazenar esses dados por um período de *clock* e então enviá-los a um *latch* de oito bits (“*reg_horiz*”) que irá liberar os dados apenas quando o *gate* receber um pulso. Este pulso será responsável pela liberação do dado que irá para a última etapa, constituída pelo registrador composto de oito *flip-flops* tipo D, denominado “*ff_horiz3*”. O sinal responsável pela liberação dos dados do “*reg_horiz*” será o sinal enviado pela unidade de controle, mais especificamente o bloco lógico “*controle_memorias*” que possui um contador de nove bits, responsável pela contagem dos *pixels* de uma linha ativa de vídeo. Sendo assim, a cada final de contagem o contador libera um pulso (“*fim_pixel*”) e este, por

sua vez, aciona o *gate* do “*reg_horiz*” liberando assim, o dado correspondente à soma truncada no final de uma linha ativa de vídeo. A figura 6-13 representa os oito bits mais significantes na saída do acumulador horizontal (“*acum_horiz*”), representados pelo canal A2 e o dado resultante da saída do registrador (“*reg_horiz*”), no canal A3. Nota-se também que o sinal “*fim_pixel*” é responsável por “zerar” o acumulador horizontal a cada fim de contagem dos *pixels* de cada linha ativa de vídeo.

Figura 6-13 Dados amostrados. Canal A2: saída do acumulador horizontal. Canal A3: saída do registrador, com o valor da soma truncada no final de uma linha ativa de vídeo

Como pode ser observado na figura 6-13, o resultado de interesse é o dado “AC”, representado na forma hexadecimal. Este dado pode ser armazenado em uma memória, juntamente com os dados resultantes da soma dos *pixels* das demais linhas e então, enviados a um microprocessador para uma futura comparação dos dados.

6.1.4 Simulação e teste do somador de *pixels* verticais

Várias simulações foram realizadas para a etapa de soma dos *pixels* verticais.

A figura 6-14 mostra a simulação realizada para a geração dos endereços transpostos que serão utilizados para a leitura das memórias na etapa da integração dos *pixels* na direção vertical. O diagrama esquemático desta etapa está representado na figura 5-10 do Capítulo 5.

Figura 6-14 Simulação dos endereços transpostos

Pela simulação apresentada, observa-se que a saída “s_cont_vert” apresenta os valores correspondentes dos endereços dos *pixels* na direção vertical ou seja, o valor “0” corresponde ao endereço do primeiro *pixel* da primeira linha de vídeo e, por sua vez, o valor “512” corresponde ao endereço do primeiro *pixel* da segunda linha e assim por diante.

Quando a contagem atingir seu valor máximo, que nesse caso será de 256, o “cont_vert” (figura 5-10) produz um bit de “carry out” esse bit será enviado a um outro

contador, denominado “*cont_vert1*” (figura 5-10) como sendo um bit de “*carry in*”, complementando assim a contagem. Assim, ao término de cada ciclo de contagem, os valores dos endereços dos *pixels* vão sendo incrementados, como pode ser observado na figura 6-15.

Figura 6-15 Simulação dos endereços transpostos

Realizou-se também, a simulação da unidade responsável pela soma dos *pixels* na direção vertical. A figura 6-16 representa o comportamento desta etapa.

Figura 6-16 Simulação do somador de *pixels* na direção vertical com dezessete bits de resolução

Na simulação, representada na figura 6-16, as entradas “*endereços*”, “*dados*” e “*fim_pixel*” são aleatórias. Os valores obtidos nas saídas “*s_reg_vert*” e “*s*” representam o valor da soma truncada dos valores dos *pixels* na direção vertical da imagem.

Uma outra simulação foi realizada mostrando na saída “*s*” apenas o valor correspondente aos oito bits mais significativos dos dezessetes que compõem os dados na saída do acumulador, denominado “*acum_vert*”. Esta simulação pode ser vista na figura 6-17.

Figura 6-17 Simulação do somador de *pixels* verticais mostrando os oito bits mais significativos na saída.

Para os testes realizados no hardware foram adotados os mesmos procedimentos do somador de *pixels* horizontais.

É importante ressaltar que, observando-se a figura 5-11 (Capítulo 5), os endereços que serão ativados para a leitura dos dados das memórias serão os endereços transpostos,

gerados pelos contadores descritos anteriormente. Para ativar os contadores, que produzem os endereços transpostos, utilizou-se um contador de dezessete bits. Quando esse contador atinge seu valor máximo um bit “*carry out*” é enviado ao contador principal da etapa de transposição de endereços, com o objetivo de habilitar esse contador e, assim começar a gera, os endereços transpostos. Outra função do bit “*carry out*” é servir como *clock* de um *flip-flop* tipo D que irá manter o sinal de saída desse *flip-flop* em nível alto durante o tempo de contagem do contador de dezessete bits possibilitando assim, que a soma dos *pixels* na vertical seja realizada no tempo adequado.

Alguns testes realizados são representados pelas figuras que se seguem.

Figura 6-18 Dados lidos diretamente das memórias (A0) e "carry out" (CK1)

Na figura 6-18 o canal CK1 representa quando o contador de dezessete bits atinge seu valor máximo, gerando um pulso. A partir desse momento o multiplex é habilitado, através de um *flip-flop* tipo D e apenas os endereços transpostos são lidos nas memórias. No canal (A0) os valores a partir do pulso representado em CK1 são os valores transpostos lidos das memórias.

A próxima figura apresenta os dados obtidos das memórias (A0) e na saída do primeiro registrador, denominado “ff_vert1” (A1).

Figura 6-19 Leitura de dados das memórias (A0) e leitura de dados do "ff_vert1" (A1)

Tomando a imagem mostrada na figura 6-7 obtiveram-se os resultados apresentados na figura 6-19.

Figura 6-20 Dados obtido da uma imagem de um objeto escuro em um fundo claro

Na figura acima, o canal CK1 está representando o “carry out” do contador de dezessete bits, o canal A0 representa os dados na saída do “*ff_vert1*” e, o canal A2 está mostrando o dado resultante da soma dos *pixels* de uma coluna vertical. Na figura também estão representados os dados na forma de lista e um histograma da imagem.

A figura 6-21 mostra os dados na forma de ondas e lista e também um histograma da imagem filmada. Neste caso a imagem foi a mesma representada na figura 6-9.

Figura 6-21 Dados obtidos de um objeto escuro deslocado na cena

6.2 Considerações Finais

Durante os testes com o hardware, todos os níveis de tensão utilizados foram compatíveis com o nível TTL. Todas as informações técnicas dos dispositivos e aparelhos, em especial o analisador lógico TLA 613, podem ser encontradas em manuais e *datasheets* específicos de cada fabricante. As impedâncias de entrada e saída do sinal de vídeo composto são de 75 ohms.

A frequência do *clock* do circuito é de 10 MHz e, para a obtenção dessa frequência foi realizada uma divisão por 2 da frequência fornecida por um oscilador de cristal de 20 MHz. Para realizar essa operação de divisão utilizou-se o bit menos significativo de um contador de nove bits.

Para a programação dos CPLDs utilizou-se a interface JTAG (*Joint Test Action Group*), desenvolvida nos anos 80, segundo o padrão IEEE 1149.1-1990. Esta interface permite testar e programar um CPLD pela porta serial ou paralela. A conexão com o *chip* é feita por quatro pinos dedicados: TCK (*Test Clock*), TMS (*Test Mode Select Input*), TDI (*Test Data Input*) e TDO (*Test Data Output*).

Capítulo 7

7 Conclusões

7.1 Considerações Iniciais

O processamento de vídeo e a visão computacional são áreas que estão crescendo muito rapidamente nos últimos anos e, possuem aplicações em diversas áreas, tais como: visão de máquina, sistemas de vigilância, análise de movimentos humanos, sistemas de detecção e rastreamento de pedestres e veículos, dentre outras.

Muitas das aplicações de processamento de vídeo e visão computacional requerem processamento em tempo real. Uma das maneiras de se obter altas taxas de processamento nestas aplicações é pela utilização de arquiteturas e estruturas de hardware especializadas e dedicadas. Levando-se em consideração essa abordagem, foi desenvolvido e implementado neste trabalho um sistema dedicado, com o objetivo de detectar a existência de movimentos, em seqüências de vídeo, obtidas do sinal fornecido por uma câmera CCD (*Charge-Coupled Device*), monocromática. A implementação do sistema foi feita utilizando dispositivos de lógica programável complexa (CPLDs).

Todo o hardware desenvolvido foi projetado e confeccionado no Laboratório de Instrumentação Virtual Microprocessada, na Escola de Engenharia da USP de São Carlos. Para a confecção das placas de circuito impresso utilizou-se um equipamento (fresadora LPKF para circuitos impressos) que permite a confecção de placas de circuito impresso de dupla face.

Como dito anteriormente, no capítulo 4, utilizaram-se dois CPLDs para realizar o processamento do sinal de vídeo. Isto foi necessário pois somente a unidade de controle, responsável pelo armazenamento da imagem nas memórias, utilizou 66 pinos de I/O de um total de 68 (97%) e, 65 macrocélulas sendo, que esta família de CPLDs possui um total de 128 ou seja, 51% das macrocélulas estavam sendo utilizadas. Para a etapa de integração das linhas utilizou-se um segundo CPLD no entanto, os testes foram realizados separadamente para a integração das linhas horizontais e verticais.

Durante a integração das linhas na direção horizontal foi utilizado um total de 75 macrocélulas (59 % da capacidade) e 55 pinos de I/O (81%). Para a integração na direção vertical os mesmos números e porcentagens foram observados. É importante ressaltar que o número total de pinos utilizados , para cada direção de integração é alto devido ao fato da necessidade de se observar os dados na saída de cada bloco lógico. Sendo assim, uma vez que o sistema de integração esteja funcionando corretamente pode interligar as saídas e entradas dos blocos lógicos diminuindo assim o número total de pinos utilizados.

Uma alternativa encontrada foi a utilização de um dispositivo mais denso que suportaria toda a etapa de controle e integração. Para tal utilizou-se o FPGA

EP1C6Q240CB, pertencente à família Cyclone II da Altera. No entanto, devido a problemas de funcionamento, não foi possível realizar os testes a tempo.

7.2 Análise dos Resultados

Os resultados obtidos, tanto os realizados através da simulação, utilizando os softwares MAX + Plus II e Quartus II, quanto os obtido através do hardware implementado foram apresentados no capítulo anterior. De acordo com a simulação realizada para a unidade de controle (MAX + Plus II), que pode ser observada na figura 6-1 observou-se que, mesmo com as entradas aleatórias impostas, os resultados apresentaram-se coerentes e quando programadas na CPLD (interface JTAG) seu desempenho foi como o esperado, pois as imagens capturadas pela câmera CCD foram armazenadas nas memórias.

Com relação às etapas de integração do sinal de vídeo tanto na direção horizontal quanto na vertical, as simulações realizadas apresentaram resultados satisfatórios ao que se esperava tanto quando realizadas com os dezessete bits de resolução quanto quando realizadas apenas os oito bits mais significativos. Várias simulações foram realizadas para a etapa de endereçamento das memórias, para a leitura dos dados, durante a etapa de integração do sinal de vídeo na direção vertical. Os resultados dessas simulações estão de acordo com o que foi proposto.

Os resultados obtidos diretamente do hardware, utilizando-se o analisador lógico, se comportaram de maneira semelhante aos resultados das simulações, demonstrando que o propósito de se adquirir o valor da soma truncada no final de cada linha ativa de vídeo foi atingido. A maior dificuldade até então havia sido a aquisição desses valores quando a integração acontecia na direção vertical, pois como o tempo de atualização dos dados no analisador lógico é da ordem de alguns segundos, o processamento dos dados obtidos diretamente do hardware apresentava um período de atraso. Sendo assim, o tempo para começar a adquirir os valores da integração nessa direção apresentava um atraso de aproximadamente três segundos.

O sistema desenvolvido gera para cada imagem armazenada na memória os histogramas de intensidade vertical e horizontal através da integração dos *pixels* da imagem. Estes histogramas ficam disponíveis para serem enviados a um sistema de processamento simples, como um microcontrolador, que ficará encarregado de processá-los para fornecer dados da ocorrência de movimento nas seqüências de imagens adquiridas.

Como resultado do trabalho desenvolvido pode-se concluir que a metodologia proposta para detecção de movimento é viável, porém grande esforço ainda é necessário para produzir um sistema completo e dedicado para a detecção de movimentos em seqüências de imagens.

7.3 Propostas para Trabalhos Futuros

Com relação ao projeto desenvolvido nesse trabalho, algumas melhorias podem ser feitas. Primeiramente poder-se-ia utilizar um dispositivo de lógica programável de maior capacidade, tornando, dessa maneira, o hardware mais compacto.

A etapa seguinte, não realizada, é a utilização de um microcontrolador para realizar o processamento que determine a existência de movimento a partir dos histogramas de intensidade das imagens.

Uma outra proposta para trabalhos futuros seria a implementação de um sistema que, a partir da informação dos dados da integração do sinal de vídeo, este pudesse realizar o rastreamento (“*tracking*”) do objeto na cena. Porém sabe-se que essa não é uma tarefa muito fácil pois várias medidas teriam que ser levadas em consideração como por exemplo: o fundo da imagem, tamanho e textura do objeto, oclusão, parcial ou total, entre outras.

Referências Bibliográficas

ALTERA, Corp. (1998). Advantages of ISP-Based PLDs over Tradicional PLDs. Product Information Bulletin, n. 24.

ALTERA, Corp. (2002). *CPLDs x FPGAs, Comparing High-Capacity Programmable Logic*. Product Information Bulletin, n.18.

ALTERA, Corp. (2003). Cyclone Devices Handbook, 2003. Disponível em www.altera.com.

ANALOG DEVICES, (2005). *AD5424, 8bit, High Bandwidth, 20.4MSPS DIGITAL-TO-ANALOG CONVERTER (Datasheet)*.

ARANDA, J.; CLIMENT, J.; GRAU, A. (1998). *A FPGA Implementation of a Video Rate Multi-Target Tracking System*. IEEE Computer Society – Digital Library, (último acesso: 10 Agosto, 2005).

BOURIDANE, A. et al. (1999). *A high level FPGA-based abstract machine for image processing*. Journal of System Architecture, v.45, p. 809-824.

BRÉMOND, F.; THONNAT, M. (1998). *Tracking Multiple Nonrigid Objects in Video Sequences*. IEEE Transactions On Circuits And Systems For Video Tecnology, v.8, n.5, September.

BROWN, S.; ROSE, J. (1996). *FPGA and CPLD Architectures : A Tutorial*. IEEE Design & Test Of Computers, v.13, n.2, p. 42-57.

BUENAPOSADA, J. M.; BAUMELA, L. (2002). *Real-Time Tracking and Estimation of Plane Pose*. Proc. ICPR 2002, v. 2, p. 697-700, IEEE. Quebec, Canada, August.

BULLOCK, D.; ZELEK, J. (2002). *Real-Time Tracking for Visual Interface Applications in Cluttered and Occluding Situations*. <http://www.cipprs.org/vi2002/pdf/s2-3.pdf> (último acesso: 10 Agosto, 2005).

CHAN, P. K.; MOURAD, S. (1994). *Digital Design Using Field Programmable Gate Arrays*. Prentice Hall.

CLARKE, J. C.; ZISSERMAN, A. (1996). *Detection and Tracking of Independent Motion*. Image and Vision Computing, v. 14, p. 565-572.

COIFMAN, B.; BEYMER, D. (1998). *A Real-Time Computer Vision System for Vehicle Tracking*. <http://www.ccegs.ohio-state.edu/~coifman/documents/TR-Crw.pdf> (último acesso: 10 Agosto, 2005).

CONEXANT (1999). *20 MSPS Monolithic CMOS 8-bit Flash Video A/D Converter (Datasheet)*.

ELANTEC (1993). *EL4581C Sync Separator, 50% Slice, S-H, Filter (Datasheet)*.

ERDEM, Ç. E.; TEKALP, A. M.; SANKUR, B. (2003). *Video Object Tracking With Feedback of Performance Measures*. IEEE Transactions On Circuits And Systems For Video Technology, v. 13, n. 4, p.310-324, April.

FENG, Y.; HARVEY, A. L.; JENNINGS, A. (2003). *Background Modeling For Tracking Object Movement*. <http://www.aprs.org.au/wdic2003/CDROM/123.pdf> (último acesso: 10 Agosto, 2005).

FÖLDESY, P.; SZATMÁRI, I.; ZARÁNDY, Á. (2002). *Moving Object Tracking on Panoramic Images*. http://www.sztaki.hu/~szatmari/CNNA2002_PAL_camera.pdf (último acesso: 10 Agosto, 2005).

FORESTI, G. L. (1998). *A Real-Time System for Video Surveillance of Unattended Outdoor Environments*. IEEE Transactions On Circuits And Systems For Video Technology, v. 8, n. 6, p. 697-704, October.

GARDNER, W. F.; LAWTON, D. T. (1996). *Interactive Model-Based Vehicle Tracking*. IEEE Transactions On Pattern Analysis And Machine Intelligence, v.18, n.11, November.

GHIASI, S.; MOON, H. J.; SARRAFZADEH, M. (2004). *Collaborative and Reconfigurable Object Tracking*. The Journal Of Supercomputing, v.30, p. 213-238.

GREENE, J. et al. (1993). *Antifuse Field Programmable Gate Arrays*. Proceedings Of The IEEE, v.81, n.7, July.

GUŠTIN, V. (1998). *An FPGA extension to ALU functions*. Microprocessors and Microsystems, v.22, p. 501-508, December.

HAUCK, S. (1998). *The Roles of FPGA's in Reprogrammable Systems*. Proceedings Of The IEEE, v.86, n.4, p. 615-638, April.

I. SZATMÁRI (2000). *The implementation of a Nonlinear Wave Metric for Image Analysis and Classification on the 64x64 I/O CNN-UM Chip*. CNNA 2000, 6th IEEE International Workshop on Cellular Neural Networks and their Applications, May 23-25, 2000, University of Catania, Italy.

JENKINS, J. H. (1994). *Designing With FPGAs and CPLDs*. Prentice Hall.

KAKADIARIS, I.; METAXAS, D. (2000). *Model-Based Estimation of 3D Human Motion*. IEEE Transactions On Pattern Analysis And Machine Intelligence, v.22, n.12, December.

KRÜEGER, S.; CALWAY, A. (1996). *Multiresolution Motion Estimation Using an Affine Model*. CSTR 96-002, University of Bristol – Departament of Computer Science, February.

MARQUES FILHO, O.; VIEIRA NETO, H. (1999). *Processamento Digital de Imagens*. Rio de Janeiro, Brasport.

MINHONI, D. C. R.; RODA, V. . *Detecção de Movimento de Objetos em Tempo Real utilizando Dispositivos de Lógica Programável Complexa*. II Southern Conference on Programmable Logic – Mar Del Plata, Argentina (2005).

NINCE, U. S. (1991). *Sistemas de Televisão e Vídeo – Câmeras, Vídeo-Teipes, Televisão*. 2. ed. Livros Técnicos e Científicos Editora.

NIU, W.; JIAO, L.; HAN, D.; WANG, Y. F. (2004). *Real-Time Multi-person Tracking in Video Surveillance*. <http://www.cs.ucsb.edu/~yfwang/papers/PCM03.pdf> (último acesso: 10 Agosto, 2005).

OLDFIELD, J. V.; DORF, R. C. (1995). *Field Programmable Gate Array – Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems*. 1. ed. John Wiley & Sons, Inc – Wiley Interscience.

OLIVEIRA, I. N. S. (2003). *Metodologia para Detecção Rápida de Movimento em Sequências de Imagens*. Tese (Doutorado) – Universidade de São Paulo, São Carlos, Brasil.

PARK, J.; YAGI, N.; ENAMI, K.; AIZAWA, K.; HATORI, M. (1994). *Estimation of Camera Parameters from Image Sequence for Model-Based Video Coding*. IEEE Transactions On Circuits And Systems For Video Technology, v. 4, n. 3, p. 288-296, June.

PEDRINO, E. C. (2003). *Arquitetura Pipeline para Processamento Morfológico de Imagens Binárias em Tempo Real utilizando Dispositivos de Lógica Programável Complexa*. Dissertação (Mestrado) – Universidade de São Paulo, São Carlos, Brasil.

HEIKKILÄ, J.; SILVÉN, O. (2004). *A Real-Time System for Monitoring of Cyclists and Pedestrians*. Image and Vision Computing, v. 22, p. 563-570.

PIO, J. L. S.; CAMPOS, M. F. M. (2002). *Percepção do Movimento por Agentes Visuais Móveis*. Belo Horizonte, Brasil, Universidade Federal de Minas Gerais. Relatório Técnico. 30.

RICQUEBOURG, Y.; BOUTHEMY, P. (2000). *Real-Time Tracking of Moving Persons by Exploiting Spatio-Temporal Image Slices*. IEEE Transactions On Pattern Analysis And Machine Intelligence, v. 22, n. 8, p. 797-808, August.

ROMA, N. F. V. (2001). *Processadores Dedicados para Estimação de Movimento em Sequências de Vídeo*. Dissertação (Mestrado) – Universidade Técnica de Lisboa, Portugal.

ROSE, J. et al. (1993). *Architecture of Field-Programmable Gate Arrays*. Proceedings Of The IEEE, v.81, n.7, p. 1013-1029, July.

SCHOEPFLIN, T.; CHALANA, V.; HAYNOR, D. R.; KIM, Y. (2001). *Video Object Tracking With a Sequential Hierarchy of Template Deformations*. IEEE Transactions On Circuits And Systems For Video Technology, v. 11, n. 11, p. 1171-1182, November.

SCHURE, A. (1964). *Televisão Básica*. Rio de Janeiro, Livraria Freitas Bastos.

SEDRA, A. S.; SMITH, K. C. *Microeletrônica*, 4ª edição, Makron Books, 2005.

SENATORI, N. O. B.; SUKYS, F. (1987). *Introdução à Televisão e ao Sistema PAL-M*. Rio de Janeiro, Guanabara.

SOLDEK, J.; MANTIUK, R. (1999). *A reconfigurable processor based on FPGAs for pattern recognition, processing, analysis and synthesis of images*. Pattern Recognition Letters, v.20, p. 667-674.

TEKALP, A. M. (1995). *Digital Video Processing*. Prentice-Hall.

TEXAS INSTRUMENTS, (2004). *TLC5540, 8 bit, High-Speed, 40MSPS ANALOG-TO-DIGITAL CONVERTER (Datasheet)*.

TISSAINAYAGAM, P.; SUTER, D. *Object Tracking in Images Sequences using Point Features*. <http://www.aprs.org.au/wdic2003/CDROM/69.pdf> (último acesso: 10 Agosto, 2005).

THOMAS, F. et al. (1999). *A hardware/software codesign for improved data acquisition in a processor based embedded system*. Microprocessors and Microsystems, v.24, p. 129-134, September.

VASSÁNYI, I. (1997). *FPGAs and cellular algorithms : Two implementation examples*. Journal of System Architecture, v.43, p. 23-26.

WANG, D. (1998). *Unsupervised Video Segmentation Based on Watersheds an Temporal Tracking*. IEEE Transactions On Circuits And Systems For Video Technology, v. 8, n. 5, p. 539-546, September.

WANG, Y.; DOHERTY, J.; VAN DYCK, R. E. (2000). *Moving Object Tracking in Video*. IEEE Computer Society, Imagery Pattern Recognition Workshop, p. 95.

Apêndice A

Apêndice E

Circuito final do módulo de armazenamento e controle utilizando dois CPLDs

EPM7128SLC84-7

Apêndice F

Circuito do módulo de armazenamento e controle utilizando um FPGA
EP1C6Q240CB.

Apêndice G

Fotos do hardware implementado, proposto no capítulo 4

Apêndice H

Códigos das subunidades, da Unidade de Controle; escritos em AHDL

Período Inicial de Linhas Ativas

```
% Periodo Inicial das Linhas Ativas %
% Março 2006 %
% Danilo Carlos Rossetto Minhoni %
```

```
SUBDESIGN inicio_linhas_ativas
```

```
(
  clk, clr, enb: INPUT;
  out[7..0], maximo: OUTPUT;
)
```

```
VARIABLE
```

```
    count[7..0]: DFF;
```

```
BEGIN
```

```
    count[].clk = clk;
    count[].clrn = clr;
```

```
    IF enb THEN
        count[].d = count[].q+1;
    ELSE
        count[].d = count[].q;
    END IF;
```

```
    out[] = count[];
    maximo = (out[7]==H"32");
```

```
END;
```

Contador de Pulsos Serrilhados e Equalizadores

```
% Contador de Pulsos Serrilhados e Equalizadores %
% Março 2006 %
% Danilo Carlos Rossetto Minhoni %
```

```
SUBDESIGN contador_pulsos_serrilhados
```

```
(
  clk,clr,enb,o_e: INPUT;
  out[4..0], maximo: OUTPUT;
)
```

```
VARIABLE
```

```
  count[4..0]: DFF;
```

```
BEGIN
```

```
  count[].clk=clk;
  count[].clrn=clr;
```

```
  IF enb THEN
```

```
    count[].d=count[].q+1;
```

```
    IF (count[].q==H"10") & (o_e==1) THEN
```

```
      maximo=B"1";
```

```
    ELSIF
```

```
      (count[].q==H"f") & (o_e==0) THEN
```

```
        maximo=B"1";
```

```
    ELSE
```

```
      maximo=B"0";
```

```
    END IF;
```

```
  ELSE
```

```
    count[].d=count[].q;  
  
END IF;  
out[]=count[];  
  
END;
```

Unidade de Controle Interna

```

% Unidade de Controle Interna %
% Março 2006 %
% Danilo Carlos Rossetto Minhoni %

SUBDESIGN unid_contr_int

(
  start, clk, reset, sinc_comp, contr_le: INPUT;
  inlinhativa, sin_le: OUTPUT;
)

VARIABLE

  estado: MACHINE OF BITS (q1, q0) WITH STATES (
    i_ap_vert, % inicio do apagamento vertical %
    ap_vert,   % apagamento vertical %
    i_linha    % inicio da linha %
  );

BEGIN

  estado.clk=clk;
  estado.reset=reset;

  IF contr_le THEN

    sin_le=B"1";

  ELSE

    sin_le=B"0";

  END IF;

  TABLE

    estado, start, sinc_comp => estado, inlinhativa;

    i_ap_vert, 0, X => i_ap_vert, 0;
    i_ap_vert, 1, X => ap_vert, 0;
    ap_vert, 0, X => i_linha, 0;
    i_linha, 0, 0 => i_linha, 1;

```

```
        i_linha, 0, 1 => i_linha, 0;  
    END TABLE;  
END;
```


Controle de Exclusão de Pulsos Serrilhados e Equalizadores

```
% Controle de Exclusao dos Pulsos Serrilhados e Equalizadores %
```

```
% Março 2006 %
```

```
% Danilo Carlos Rossetto Minhoni %
```

```
SUBDESIGN controle_exclusao_pulsos
```

```
(
    maximo_in, clk, reset : INPUT;
    start : OUTPUT;
)
```

```
VARIABLE
```

```
    estado : MACHINE OF BITS (q1,q0)
```

```
        WITH STATES (
```

```
            inicio_exclusao, % inicio exclusao %
```

```
            exclusao,        % exclusao %
```

```
            fim_exclusao     % fim exclusao %
```

```
        );
```

```
BEGIN
```

```
    estado.clk=clk;
```

```
    estado.reset=reset;
```

```
    TABLE
```

```
        estado, maximo_in => estado, start;
```

```
    inicio_exclusao, 0 => exclusao, 0;
```

```
        exclusao, 0 => exclusao, 1;
```

```
        exclusao, 1 => fim_exclusao, 0;
```

```
    END TABLE;
```

```
END;
```

Contador de Linhas de Vídeo

```

% Contador de Linhas de Video %
% Março 2006 %
% Danilo Carlos Rossetto Minhoni %

SUBDESIGN contador_de_linhas

(
  clk, clr, enb, fim_pixel: INPUT;
  out[16..9], fim_linha: OUTPUT;
)

VARIABLE

  count[16..9]: DFF;

BEGIN

  count[].clk=clk;
  count[].clrn=clr;

  IF enb THEN

    count[].d=count[].q+1;

  ELSE

    count[].d=count[].q;

  END IF;

  out[]=count[];
  fim_linha=(out[]==H"ff")&(fim_pixel==1);

END;
```

Contador de *Pixels*

```

% Contador de Pixels %
% Março 2006 %
% Danilo Carlos Rossetto Minhoni %

SUBDESIGN contador_de_pixels

(
  clk, clr, enb: INPUT;
  out[8..0], fim_pixel: OUTPUT;
)

VARIABLE

  count[8..0]: DFF;

BEGIN

  count[].clk=clk;
  count[].clrn=clr;

  IF enb THEN

    count[].d=count[].q+1;

  ELSE

    count[].d=count[].q;

  END IF;

  out[]=count[];
  fim_pixel=(out[7]==H"1ff") % &(!clk) %;

END;
```