

Faculdade de Engenharia da Universidade do Porto



FEUP

Cálculo de mapas de profundidade para imagens estéreo em tempo real usando FPGAs

Marc Pinto Antunes

Dissertação de Projecto realizada no âmbito do

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Major Telecomunicações

Orientador: Prof. João Canas Ferreira

Julho de 2008

Resumo

O trabalho descrito neste documento tem como principal objectivo implementar a determinação de mapas de disparidade a partir de imagens estéreo, em tempo real, com base em dois sensores de imagem CMOS.

A implementação utiliza uma placa de desenvolvimento com uma FPGA Xilinx XS3 1500, ligada a um par de sensores de imagem CMOS.

O estudo de métodos da correspondência entre imagens resultou na escolha de algoritmos de correspondência por área em detrimento da correspondência por contornos.

São apresentadas formas de reduzir o tempo de processamento, diminuindo o número de correspondências por aproveitamento das características das imagens estereoscópicas, eliminando zonas da imagem que nunca podem ser homólogas.

A dissertação inclui duas versões de cálculo de disparidade baseado na métrica SAD.

A primeira versão apresenta um grande nível de falsas disparidades para concentrações de objectos de tamanho reduzido. Para solucionar este problema foi desenvolvida uma segunda versão. O resultado foi à custa de um tempo de processamento cerca de 8 vezes superior. Os recursos gastos por ambas as versões são muito semelhantes. No total são utilizados 21% dos recursos da FPGA.

Todos os módulos desenvolvidos para este projecto são descritos e as respectivas bancadas de teste são apresentadas. Também foram realizados testes de integração sobre a totalidade do projecto. Foi determinada a influência da variação dos 3 parâmetros controláveis, isto é, largura, altura e gama de procura, sobre o número de ciclos de relógios necessários.

Foi dada especial atenção à validação do cálculo de disparidades, tendo sido utilizado um conjunto de imagens sintética. Os valores obtidos durante a simulação são comparados com os esperados, calculados com *Matlab*, confirmando assim o bom funcionamento geral do sistema. A imagem gerada pela simulação é convertida para um ficheiro no formato PGM, facilitando assim a visualização do resultado da simulação.

A dissertação inclui ainda propostas de melhoria do trabalho.

Abstract

The present work's main purpose is to describe the hardware implementation of methods for the determination of disparity maps from stereo images, in real time, using two CMOS image sensors.

The implementation uses an evaluation platform with a Xilinx FPGA XS3 1500, connected to a pair of CMOS image sensors.

The study of matching methods between images resulted in the choice of an area based algorithm over one based on edge detection.

In an effort to reduce processing time of the characteristics of the image pair were exploiting, in order to discard image areas that could never have homologue regions.

The work includes two versions of disparity calculation based on the SAD metric. The first version shows a great number of false disparities for images with high concentration of small objects. To solve this problem a second version was developed. The improvement of the solution was achieved at the cost of an eightfold increase of processing time. The resources used by both solutions are similar: for the complete solution, only 21% of the FPGA resources are used.

All the modules developed for the project are described and the corresponding testbenches are shown. The complete solution was also subjected to integration tests. The influence of three controllable parameters (image height, width and search range) on the number of required clock cycles was also investigated.

Special attention was given to the validation of the disparity calculation by means of a set of synthetic images. The values obtained from the simulation runs were compared to the expected values, obtained using *Matlab*, thereby confirming the anticipated overall behavior of the system. The images generated by a simulation run are stored in the PGM file format, allowing the direct visualization of the simulation results.

Proposals for improvement are also included in the thesis.

Índice

Resumo	i
Abstract	iii
Índice	v
Lista de figuras	vii
Lista de tabelas	ix
Lista de abreviaturas	x
Capítulo 1.....	1
Introdução	1
1.1 - Caracterização do problema.....	1
1.2 - Objectivos pretendidos	2
1.3 - Estrutura do documento.....	2
Capítulo 2.....	5
Estado da Arte	5
2.1 - Visão estéreo	5
2.1.1 - Fontes de erros	5
2.2 - Calibração.....	6
2.2.1 - Geometria epipolar	7
2.3 - Correspondência	8
2.4 - Reconstrução	9
2.5 - Material disponibilizado	10
2.5.1 - Escolha da tecnologia.....	10
2.5.2 - Placa de desenvolvimento	11
2.5.3 - Particularidades da FPGA	12
2.5.4 - Ficheiros <i>toplevel</i> , <i>ucf</i> e descrição <i>hardware</i>	13
2.5.5 - Câmaras	14
2.5.6 - Placa VGA adicional	14
2.5.7 - Software de desenvolvimento	15
2.6 - Conclusões	15
Capítulo 3.....	17
Descrição da Solução	17
3.1 - Correspondência por áreas.....	17
3.2 - Correspondência por contornos	19
Capítulo 4.....	23
Desenvolvimento	23

4.1 - Introdução à implementação prática	23
4.1.1 - Barramentos necessários	23
4.1.2 - Instalação da placa VGA.....	24
4.1.3 - Teste dos barramentos disponibilizados.....	24
4.1.4 - Atraso das saídas VGA	26
4.1.5 - Escolha da memória.....	27
4.1.6 - Escolha do tamanho das janelas de observação	34
4.1.7 - Redução do número de ciclos de relógio	36
4.2 - Descrição do algoritmo	38
4.3 - Implementação dos diferentes módulos	40
4.3.1 - Módulo de geração das janelas de observação	41
4.3.2 - Módulo de cálculo da métrica SAD	44
4.3.3 - Módulo de atraso vídeo	45
4.3.4 - Módulo de interface com a SDRAM	46
4.3.5 - Módulo de captação vídeo	52
4.3.6 - Módulo de correspondência.....	58
4.3.7 - Interligação dos vários módulos.....	67
4.4 - Síntese e implementação do circuito	70
4.5 - Testes práticos	70
4.6 - Determinação do tempo de processamento	75
Capítulo 5.....	79
Análise de resultados	79
5.1 - Resultados obtidos.....	79
5.1.1 - Teste prático da primeira versão	82
5.1.2 - Implementação e teste da segunda versão	83
Capítulo 6.....	87
Conclusões e trabalhos futuros	87
Referências	89

Lista de figuras

Figura 2.1 - Efeitos da perspectiva.	5
Figura 2.2 - Reflexão especular	6
Figura 2.3 - Suporte das câmaras	7
Figura 2.4 - Geometria epipolar (2).	7
Figura 2.5 - Disparidade.	8
Figura 2.6 - Placa de desenvolvimento disponibilizada	11
Figura 2.7 - Módulos elementares de uma FPGA.	12
Figura 2.8 - Bloco lógico configurável.	12
Figura 2.9 - Quasar Electronics - M3188A.....	14
Figura 3.1 - Correspondência de janelas sobre a janela de referência.....	18
Figura 3.2 - Janela de correspondência 3x3.....	20
Figura 3.3 - Imagem original e resultado da detecção de contornos (11).	21
Figura 4.1 - Integração do módulo delayVideo.	26
Figura 4.2 - Recursos internos de uma CLB (12).	29
Figura 4.3 - Diagrama lógico dos blocos da memória (13).	30
Figura 4.4 - Ciclo de leitura (com controlo WE desactivo) (13).....	31
Figura 4.5 - Ciclo de escrita com controlo WE (13).	32
Figura 4.6 - Organização da memória SDRAM.	34
Figura 4.7 - Janela de observação.	35
Figura 4.8 - Escolha da imagem de referência.	36
Figura 4.9 - Par de imagens com banda de observação e janelas de observação.	39
Figura 4.10 - Janela com memória auxiliar 8X8pixels.	42
Figura 4.11 - Janela com memória auxiliar 12X8pixels.....	42
Figura 4.12 - Bloco lógico de uma janela de observação.	42
Figura 4.13 - Resultado simulação TB_MEM12X8bit.	43
Figura 4.14 - Resultado simulação TB_MEM12X8Pixel.	43
Figura 4.15 - Bloco lógico do módulo de cálculo SAD.....	44
Figura 4.16 - Resultado simulação TB_Calcul.	45
Figura 4.17 - Bloco lógico do módulo delayVideo.	46
Figura 4.18 - Bloco lógico da interfaceRAM.	47
Figura 4.19 - Fluxograma interfaceRAM.....	49
Figura 4.20 - Resultado simulação TB_interfaceRAM - parte1.	51
Figura 4.21 - Resultado simulação TB_interfaceRAM - parte2.	52
Figura 4.22 - Bloco lógico da interfaceVideo.....	53

Figura 4.23 - Fluxograma do módulo de interfaceVideo.	55
Figura 4.24 - Resultado simulação TB_interfaceVideo.	57
Figura 4.25 - Bloco lógico do módulo gestor.	58
Figura 4.26 - Fluxograma módulo gestor - parte 1.	60
Figura 4.27 - Fluxograma módulo gestor - parte 2.	61
Figura 4.28 - Resultado da simulação TB_Gestor - parte1.	64
Figura 4.29 - Resultado da simulação TB_Gestor - parte2.	65
Figura 4.30 - Resultado da simulação TB_Gestor - parte3.	66
Figura 4.31 - Resultado da simulação TB_Gestor - parte4.	66
Figura 4.32 - Resultado da simulação TB_Gestor - parte5.	67
Figura 4.33 - Interligação dos diferentes módulos.	68
Figura 4.34 - Diagrama de blocos lógicos toplevel.	69
Figura 4.35 - Par de imagens de teste (15).	71
Figura 4.36 - Resultados da simulação com o uso da memória virtual.	74
Figura 4.37 - Influência de H sobre o número de ciclos de relógio necessários.	75
Figura 4.38 - Influência de W sobre o número de ciclos de relógio necessários.	76
Figura 4.39 - Influência de maxDisp sobre o número de ciclos de relógio necessários.	77
Figura 5.1 - Par de imagem teste - a risca.	79
Figura 5.2 - Resultado da disparidade "risca".	80
Figura 5.3 - Décima correspondência.	81
Figura 5.4 - Décima primeira correspondência.	82
Figura 5.5 - Décima oitava correspondência.	82
Figura 5.6 - Disparidade do pentágono (escala 0-50).	83
Figura 5.7 - Janela 8x8.	83
Figura 5.8 - Segunda versão do cálculo da disparidade.	84
Figura 5.9 - Resultado das disparidades - versão 2.	84

Lista de tabelas

Tabela 3.1 - Métricas de alguns algoritmos de correspondência por janelas.....	18
Tabela 3.2 - Cálculo da primeira derivada.	20
Tabela 3.3 - Cálculo da segunda derivada.	20
Tabela 4.1 - Descrição dos barramentos mais relevantes.....	23
Tabela 4.2 - Características dos recursos disponíveis de várias Spartan 3 (12).	28
Tabela 4.3 - Descrição dos barramentos e pinos de controlo da SDRAM.	31
Tabela 4.4 - Descrição dos barramentos interfaceRAM.	48
Tabela 4.5 - Descrição dos barramentos interfaceVideo.....	54
Tabela 4.6 - Descrição dos barramentos do módulo Gestor.	59
Tabela 4.7 - Volume de recursos necessários para implementação na FPGA.	70
Tabela 4.8 - Influência do número de linhas (H).	75
Tabela 4.9 - Influência do número de colunas (W).....	76
Tabela 4.10 - Influência do nível máximo de disparidade (maxDisp).	77
Tabela 4.11- Tempo de processamento para imagens de 640x480.....	78

Lista de abreviaturas

HDL	<i>Hardware Description Language</i>
ASIC	<i>Application Specific Integrated Circuit</i>
RISC	<i>Reduced Instruction Set Computer</i>
FPGA	<i>Field Programmable Gate Array</i>
PCB	<i>Printed Circuit Board</i>
CLB	<i>Configurable Logic Bloc</i>
LUT	<i>Look-up Table</i>
PGM	<i>Portable Gray Map</i>

Capítulo 1

Introdução

Neste capítulo pretende-se dar um enquadramento ao tema deste documento, apresentar os objectivos pretendidos com o desenvolvimento deste trabalho assim como dar uma descrição geral da estrutura do documento.

1.1 - Caracterização do problema

Tal como no mundo animal, a visão estéreo é necessária para se conseguir obter informação sobre as distâncias a que os objectos se encontram do espectador. O cérebro encarrega-se com base na experiência dos níveis de discrepância das imagens de cada olho, de transmitir-nos uma noção do nível de proximidade do objecto focado.

A captação de imagens com uma só objectiva apenas permite trabalhar sobre os contornos dos objectos mas não sobre os seus volumes ou distâncias à objectiva.

A introdução de vídeo com imagem estéreo permite da mesma forma tirar partido da diferença das imagens obtidas por cada câmara de forma a calcular uma terceira dimensão: a profundidade dos objectos na imagem, isto é, a distância deles às câmaras. Este tipo de sistema tem diversas aplicações, como por exemplo vídeo vigilância, robótica, entre outras possíveis.

Em muitas dessas aplicações, a existência de movimento na filmagem por si só não é suficiente para determinar se o movimento detectado é relevante para registo ou não. Para se conseguir um melhor nível de filtragem nesse tipo de aplicações, ter essa informação de distância do objecto à imagem é fulcral.

De referir também que, com esse nível de filtragem de *frames* de interesse, consegue baixar-se drasticamente o volume de dados armazenados em disco para posterior análise. Dependendo da aplicação pode mesmo o armazenamento resumir-se a guardar apenas a distância a que ocorre o movimento e eventualmente também a hora da ocorrência. Por exemplo para contagem do número de entradas num recinto junto a uma zona passante onde se pode quer retirar dados estatísticos ao longo do tempo.

Existem vários algoritmos com características diferentes, mas nenhum é eficaz para todos os tipos de cenas. Previu-se portanto, implementar vários protótipos onde os resultados obtidos experimentalmente seriam importantes para alteração do próximo protótipo. Foi portanto um processo com uma evolução iterativa. Para tal, a utilização de um suporte de desenvolvimento que se possa rapidamente reprogramar é essencial para o desenvolvimento deste trabalho, assim sendo, uma placa de desenvolvimento com núcleo FPGA é o dispositivo que mais se adequa a este tipo de trabalho.

Neste documento é feita uma abordagem sobre o estado da arte onde se explicam alguns conhecimentos e noções essenciais para a compreensão do trabalho. São dadas a conhecer técnicas e algoritmos possíveis, e a razão da preferência de uns face a outros. De seguida é exposto o desenrolar do trabalho propriamente dito, com as dificuldades enfrentadas e as implementações feitas. Por fim é apresentado o resultado obtido pelo algoritmo desenvolvido. Posteriormente é acrescentada uma introdução a um possível algoritmo para implementação futura como alternativa.

1.2 - Objectivos pretendidos

Este trabalho consiste na concretização de uma ideia de projecto já iniciada por outros alunos como trabalho de fim de curso, onde cada aluno, assim como vários professores, deram o seu contributo no desenvolvimento de vários módulos no sentido da captação vídeo, sincronismo entre as câmaras e colocação dos valores na saída VGA.

Este trabalho consiste, numa perspectiva final, projectar e desenvolver um algoritmo de correspondência de forma a gerar um mapa de disparidade entre as duas imagens.

Com este trabalho pretende-se transmitir informação sobre esta área de trabalho e dar a conhecer formas de solucionar este problema. Pretende-se também fomentar a curiosidade humana em reflectir sobre esta área de trabalho em grande desenvolvimento em aplicações do dia-a-dia.

Com a implementação prática, pretende-se amadurecer o conhecimento sobre a descrição hardware e otimizar o aproveitamento da tecnologia FPGA reprogramável, contornando as limitações dos recursos disponibilizados pelo modelo fornecido.

1.3 - Estrutura do documento

Esse documento inicia-se por uma introdução ao tema e exposição da razão da sua existência, onde se explica o problema a solucionar.

No segundo capítulo é apresentado o estado da arte na área de estereoscopia e sobre a tecnologia FPGA reprogramável. São abordados conceitos sobre visão estéreo tais como:

- erros característicos desse tipo de filmagem,
- calibração,
- geometria epipolar,
- pontos e linhas homólogos,
- correspondência,
- reconstrução de um cenário 3D,
- e uma abordagem do material disponibilizado.

No terceiro capítulo são expostas duas perspectivas sobre métodos de cálculo de correspondências. Um deles opera por correspondência directa de áreas; o outro aplica, antes da correspondência directa, uma filtragem para determinar os contornos da imagem. No final deste capítulo é explicada a razão da escolha do método de correspondência directa em detrimento do outro.

Com isto é iniciado o quarto capítulo com uma perspectiva mais aprofundada do desenvolvimento prático do sistema proposto para implementação. Sendo um capítulo descritivo da implementação prática, é iniciado pelo enquadramento prévio dos dispositivos e ferramentas fornecidas para a aplicação do algoritmo, pois antes de se desenvolver o sistema, é necessário saber quais são os recursos e interfaces disponíveis para utilização. O capítulo continua com a exposição da possível implementação da correspondência directa por áreas, onde as sucessivas etapas do algoritmo são apresentadas, explicando a razão de cada uma delas. Desde a escolha do tamanho das janelas, passando pelo gestão do acesso à memória até a determinação das diversas condições de paragem. Ao longo das descrições tenta dar-se uma ideia da progressiva evolução de cada passo e obstáculos encontrados durante a implementação de cada módulo.

O quinto capítulo é dedicado a apresentação dos resultados práticos, onde é feita uma interpretação dos resultados no sentido de serem corrigidas para uma versão seguinte. Perante estes resultados é elaborada uma melhoria da primeira versão.

Por fim, no sexto e último capítulo, são tiradas conclusões sobre o método. São feitas diversas análises ao nível de melhorias apresentadas, sobretudo centradas sobre o requisito da resposta considerada em tempo real. Por fim, são apresentadas propostas para diminuir o número de ciclos de relógio melhorando o aproveitamento dos módulos.

Capítulo 2

Estado da Arte

2.1 - Visão estéreo

Filmar nas mesmas condições uma cena em duas perspectivas diferentes gera, obviamente, duas imagens diferentes. Tomando por exemplo um objecto no cenário, esse vai aparecer nas imagens em sítios e formas diferentes. Mas se esse objecto estiver mais longe das objectivas, esse nível de discrepância diminui. É este tipo de comportamento que é explorado na visão estéreo, isto é, os diferentes níveis de disparidade das posições relativas dos objectos filmados nas duas imagens. Este valor indica a profundidade a que se encontra o objecto na cena.

2.1.1 - Fontes de erros

Trabalhar com visão estéreo traz consigo vários problemas característicos da troca de perspectiva numa mesma cena. Certos objectos não vão surgir em ambas as imagens ou então apenas em parte. O mesmo objecto vai surgir nas imagens com dimensões e formas diferentes entre elas, fenómeno chamado **distorção**. Outro caso frequente em cenas com vários objectos em planos de profundidade diferentes, a **oclusão**, designa que um objecto ou parte dele, está a ser obstruído por um outro que se encontra mais próximo da objectiva. A Figura 2.1 tenta ilustrar estes casos descritos.

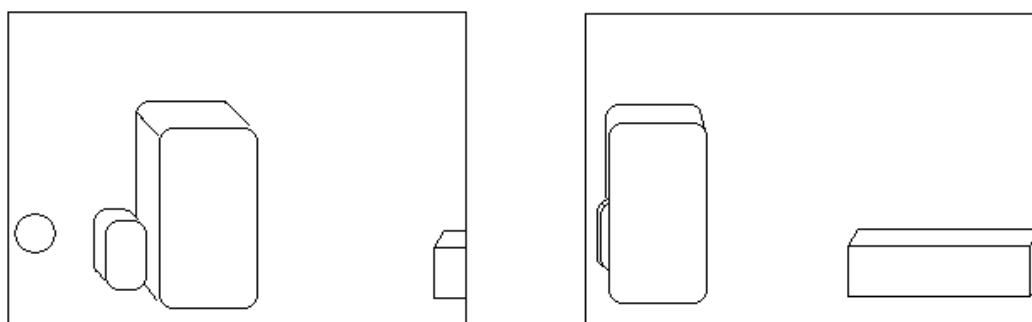


Figura 2.1 - Efeitos da perspectiva.

Devido a diferente perspectiva, a esfera presente na imagem esquerda, já não surge na imagem direita. Vejamos agora o par de caixas verticais. Pode-se observar que a caixa mais pequena se encontra parcialmente oculta pela grande, pois essa última é de maior dimensão e encontra-se mais próxima das câmaras. Logo o objecto estando à esquerda, não vai ser tão visível pela câmara da direita, vai ser ocultado pela caixa maior. Pelo contrário, a caixa horizontal é mais visível na imagem direita do que na esquerda por se encontrar à direita do cenário. Numa análise computacional, esse objecto pode parecer ter formas diferentes, quando na realidade uma parte está fora da imagem.

Nem só o que realmente está na cena provoca efeitos indesejáveis, pois em objectos lisos ou brilhantes, a diferente perspectiva leva a que surjam no par de imagens diferenças visuais. Tome-se como exemplo um objecto espelhado. É sabido do dia-a-dia que consoante a perspectiva em que nos encontramos, a imagem que nos é reflectida é diferente. Estando as câmaras em diferentes perspectivas a imagem aparentada no mesmo objecto, devido a **reflexão especular** (1), vai ser diferente entre as imagens. Tal acontecimento vai induzir em erro na correspondência, pois o sistema vai ver, naquela imagem reflectida, uma cena real e por consequente gerar correspondências que na realidade não têm relação com a real posição do objecto reflector. A Figura 2.2 mostra o caso prático de um objecto com alta reflexão especular, um espelho.



Figura 2.2 - Reflexão especular

2.2 - Calibração

Estando a recolher imagens de duas fontes diferentes, para que se tornem comparáveis necessário saber em que condições foram tiradas. No caso particular de ter duas objectivas sobre o mesmo cenário e pretendemos melhorar o algoritmo reduzindo a zona de correspondência correcta possível apenas a uma zona da imagem ou também poder determinar uma distância real a partir de um valor de disparidade obtido, é necessário conhecer certos parâmetros sobre o par de objectivas.

Uma objectiva possui um foco fictício que apenas se pode determinar de uma forma prática. Este parâmetro, entre outros, é necessário para a conversão do valor da disparidade por uma distância real do objecto às câmaras. A colocação das objectivas com as linhas de varrimento colineares permite limitar a zona de pesquisa. Uma melhor explicação será dada na secção seguinte sobre geometria epipolar.

A colocação da orientação das objectivas permite também reduzir certos problemas falados na secção anterior. Ao coloca-las ligeiramente viradas uma para a outra, isto é, de forma convergente, aumenta-se a zona de sobreposição das imagens, reduzindo também a oclusão. O que aumenta o número de correspondências correctas, pois existem uma maior semelhança entre as imagens.

Pode-se definir que o processo de **calibração** afirmando que consiste na colocação e determinação dos parâmetros de filmagem das objectivas.

No módulo fornecido, as duas câmaras estão fixas num suporte rígido permitindo a conservação desses parâmetros ao longo de todo o trabalho como é mostrado na Figura 2.3



Figura 2.3 - Suporte das câmaras

Durante a fixação, teve-se o cuidado de não deixar as objectivas apontadas em direcções paralelas, mas sim ligeiramente convergentes, (orientação convergente dos focos), aumentando a área comum entre as duas imagens e minimizando também assim a oclusão.

2.2.1 - Geometria epipolar

Sempre com o objectivo de minimizar a complexidade do problema, optou-se por fazer uma análise mais criteriosa das posições das câmaras.

A Figura 2.4 assim como o seu texto respectivo, pretendem dar a conhecer a noção de geometria epipolar (2).

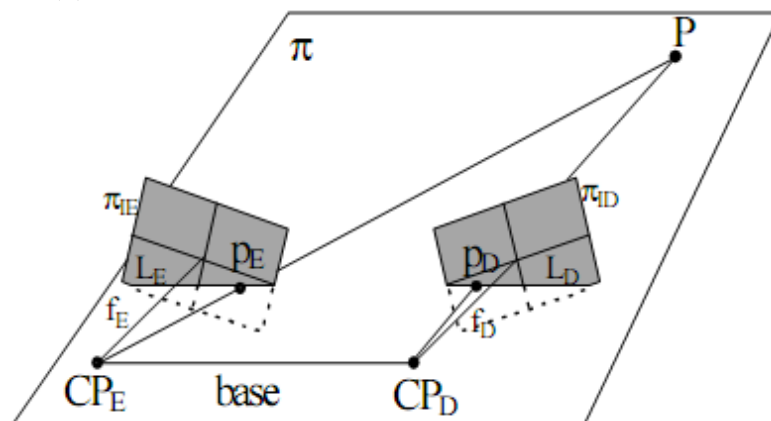


Figura 2.4 - Geometria epipolar (2).

O plano epipolar (π) é formado por três pontos: O ponto físico filmado (P) e os centros perspectivos de ambas as câmaras (CPE e CPD). Na ilustração pode-se observar, nos planos de imagem (π_{IE} e π_{ID}), os **pontos homólogos** (PE e PD) que são os pontos projectados em cada imagem, correspondentes ao mesmo ponto físico. As intersecções dos planos π_{IE} e π_{ID} com o plano epipolar geram então as **linhas epipolares** (LE e LD). Desta forma qualquer ponto de uma das imagens, se tem homólogo (na outra imagem), este encontra-se sobre a linha epipolar à sua.

Como base nesse resultado, para efeitos de melhoramento de pesquisa, concluiu-se que convém que os planos de imagem (π_{IE} e π_{ID}) sejam paralelos e que as suas linhas horizontais de mesmo índice sejam linhas epipolares.

Tal geometria reduz o leque possível de pontos candidatos a pontos homólogos a uma só linha da imagem (linha epipolar colinear com a sua linha horizontal da imagem). Isto porque, se o pixel obtido na linha índice Y da imagem esquerda existir na imagem direita, esse estará também sobre a linha índice Y. Nesse caso passamos a reduzir a nossa pesquisa a um varrimento unidimensional (apenas uma linha da imagem) e vez de toda a imagem. Reduzindo assim o peso computacional, consequentemente o tempo de processamento necessário.

2.3 - Correspondência

Os pixels homólogos, como já foi dito, são os pixels captados em diferentes perspectivas mas referentes ao mesmo ponto físico. Estes usualmente encontram-se em posições diferentes nas frames ou então pode acontecer que o pixel não tenha homólogo. Ao processo de determinação dos pixels homólogos dá-se o nome de **correspondência**.

A diferença de posição entre os pixels homólogos chama-se **disparidade**. A Figura 2.5 tenta ilustrar a disparidade ente dois pontos homólogos situados em X1 e X2.

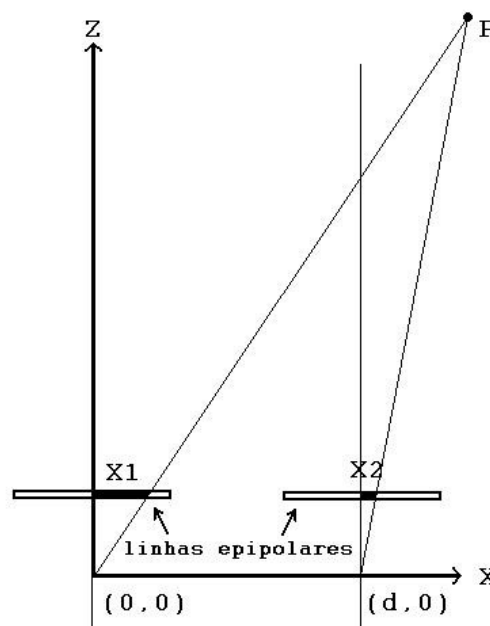


Figura 2.5 - Disparidade.

Notar que se fez referência as coordenadas apenas segundo X pois como foi referido no tópico sobre a calibração, as linhas epipolares das nossas câmaras são colineares com as linhas horizontais das imagens.

O conhecimento do valor de disparidade é essencial para se poder determinar a distância dos vários pontos às objectivas.

Essa quantidade de dados é organizada sobre a forma de **mapas de disparidade**. São vectores de informação que podem ser representados sobre a forma de uma “imagem” resultante de todas as disparidades entre as duas imagens.

Um mapa de disparidade pode ser **denso** ou **esparso** consoante se obtenha directamente, ou não, respectivamente, um valor de disparidade para todos os pixels da imagem de referência directamente do algoritmo de correspondência. No caso dos mapas esparsos, os pixels que não tenham resultado directo, o seu valor pode ser obtido, por exemplo, por cópia directa do valor do ponto calculado mais próximo, interpolação linear entre dois ou mais pontos vizinhos entre outros métodos possíveis de aproximação por pontos vizinhos.

A determinação do valor da disparidade, correspondência, torna-se difícil devido a vários fenómenos tais como:

- Oclusão.
- Reflexão especular - reflexo de uma imagem sobre um objecto liso pertencente à cena.
- Perspectiva dos objectos. (Diferem na forma e no tamanho como são vistos pelas câmaras).

2.4 - Reconstrução

Uma vez obtida a disparidade entre os pontos homólogos, pode ser de interesse prático converter esses dados em distâncias físicas. A esse processo dá-se o nome de reconstrução por se estar a “reconstruir” a cena numa perspectiva real (3D), isto é introduzir na imagem a informação da distância de cada ponto às câmaras.

Para se determinar essa distância (Z) com base no valor da disparidade (d) é necessário conhecerem-se vários parâmetros físicos tais como:

- Base (b) - é o valor da distância entre os centros perspectivos de ambas as câmaras (CPE e CPD).
- Distância dos centros perspectivos (f)

O resultado é dado pela fórmula seguinte, considerando os parâmetros b e f como fixos.

$$Z(d) = \frac{b * f}{d}$$

Convém ressaltar também que a disparidade é sempre positiva ou nula. No caso particular de ser nula esse valor será indefinido mas sabe-se que se encontra a uma distância

tal que já não se consegue distinguir uma diferença de posição, pois aquele ponto encontra-se na mesma posição para ambas as câmaras.

2.5 - Material disponibilizado

2.5.1 - Escolha da tecnologia

Implementar-se os protótipos em tecnologia ASIC (*Application Specific Integrated Circuit*) (3) teria custos elevados, pois seria necessário gerar para cada protótipo um novo *layout*, assim como as suas máscaras respectivas para se poder levar para fundição. Esses processos são dispendiosos (na ordem dos milhares de euros por cada protótipo) e demorados (várias semanas para obtenção do chip final). O que coloca inquestionavelmente de parte essa possibilidade, mesmo se esse tipo de tecnologia permita que se obtenha um elevado desempenho e baixo consumo, por ser um circuito dedicado.

Numa vertente oposta, a tentativa de ser implementado em RISC (*Reduced Instruction Set Computer*) (4) isto é, apenas por meio de software emulado por um computador, é desmesurado devido ao elevado nível de computação pedida pelos algoritmos em “tempo real”. Esse nível de complexidade será demonstrado mais a frente.

Pretende-se neste projecto usar uma tecnologia onde a realização de protótipos seja de baixo custo e reduzido tempo de espera mas também que se possa esperar um bom desempenho. Para tal é necessário um circuito com uma configurabilidade física muito próxima do exclusivamente dedicado.

Com base nessas condições a escolha de uma plataforma FPGA (*Field Programmable Gate Array*) parece a alternativa ideal, pois permite, com baixo custo (apenas o da aquisição), testar os vários protótipos em desenvolvimento. Este tipo de tecnologia permite uma reconfiguração total do circuito que se pretende implementar em questões de milissegundos, um número ilimitado de vezes e possibilitando uma lógica fortemente dedicada. Notar o uso do termo “fortemente dedicado”, pois embora visto de fora se pode obter o mais variado tipo de comportamento, internamente a lógica é composta por células básicas chamadas **CLB** (*Configurable Logic Block*) que são configuradas e interligadas de forma a se obter o comportamento desejado. Tal será explicado mais adiante.

Interessa para já focar que assim sendo é necessário existir uma lógica interna para encaminhar os sinais pelos módulos lógicos elementares que compõem o real hardware disponível da FPGA. Esse encaminhamento traz um atraso combinacional, daí uma FPGA não atingir uma frequência de funcionamento muito elevada quando comparada com uma tecnologia semelhante mas puramente dedicada. Mesmo assim essa alternativa não deixa de ser de longe, uma opção com melhor desempenho que uma solução RISC e menor custo e tempo de espera (*turn-around time*) que em ASIC.

2.5.2 - Placa de desenvolvimento

Para a realização desse projecto foi fornecido uma placa de desenvolvimento, **Xilinx Spartan 3 Development Kit** (5). A razão da escolha dessa placa reside no facto de possuir interfaces e recursos internos considerados suficientes para o desenvolvimento dos módulos necessários. Mas também se junta o facto de o trabalho de aquisição vídeo por parte das duas câmaras já estar implementado nessa placa. Tendo assim as interligações dos dispositivos com a FPGA já configuradas e comprovada a correcta propagação dos sinais para esse hardware.

Dos componentes deste kit destacam-se, como mais importante para este trabalho, os seguintes:

- Núcleo reprogramável FPGA Spartan 3 XC3S1500-FG676 (1.5M de gates);
- Frequência de funcionamento até 100MHz;
- 2MB SRAM;
- Conectores AvBus de 140 pinos I/O;

Uma placa de desenvolvimento com FPGA (6) é uma placa PCB (Printed CircuitBoard) onde se tem como núcleo da lógica principal, um chip do tipo FPGA reprogramável.

Esse tipo de placa possui uma grande variedade de módulos, afim de se poder interagir com o chip de uma forma cómoda e fácil. Os periféricos encontrados são muito diversificados, tais como pinos de acesso genérico, portas de dados (USB, Ethernet, RS232, PS2, VGA, etc.), ecrãs *oled*, *displays*, ou mesmo simples leds para dar *feedback* sobre um estado ou registo. Tal diversidade de periféricos nesse tipo de placas é devida à grande reconfigurabilidade lógica da FPGA que permite as mais variadas aplicações. Este integrado é formado por um conjunto de células lógicas que podem ser interligadas e configuradas, de uma forma muito livre, a fim de se obter o comportamento desejado, e isto, um número ilimitado de vezes. Essa tecnologia será explicada em maior detalhe de seguida.



Figura 2.6 - Placa de desenvolvimento disponibilizada

2.5.3 - Particularidades da FPGA

A FPGA é formada por um conjunto de portas lógicas configuráveis pelo utilizador através de um computador por meio de um software para tal desenvolvido, que permite assim ser programada um número de vezes sem conta.

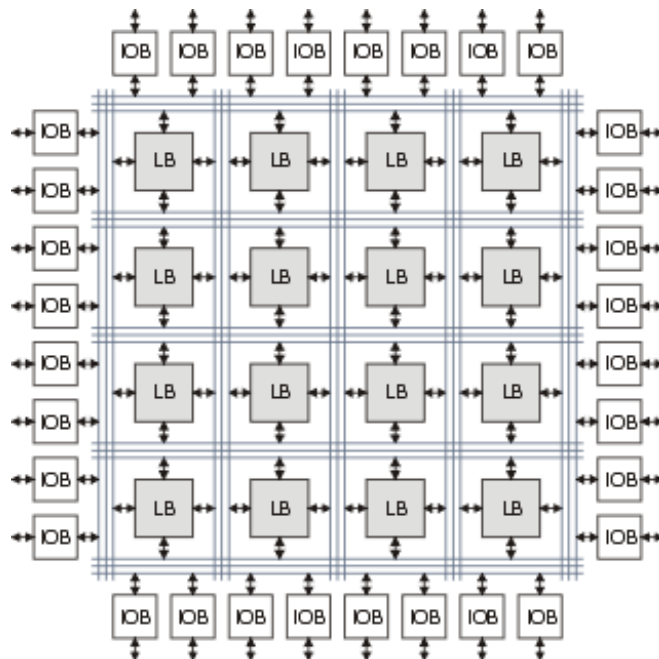


Figura 2.7 - Módulos elementares de uma FPGA.

Assim sendo, permite testar ideias de protótipos com facilidade de leveza, pois caso o protótipo falhe (o que acontece muitas vezes) podemos implementar rapidamente um novo sem qualquer custo adicional a nível da sua implementação na FPGA.

Interessa referir que uma FPGA é constituída na base pelos seguintes blocos:

- CLB - Configurable Logic Bloc : constituídos por flip-flops e portas lógicas combinacionais. É composta em duas *slices*, SLICEM e SLICEL, cada uma delas possui 2 LUT (*Look Up Table*). Essa unidade será descrita em maior detalhe no capítulo 3.

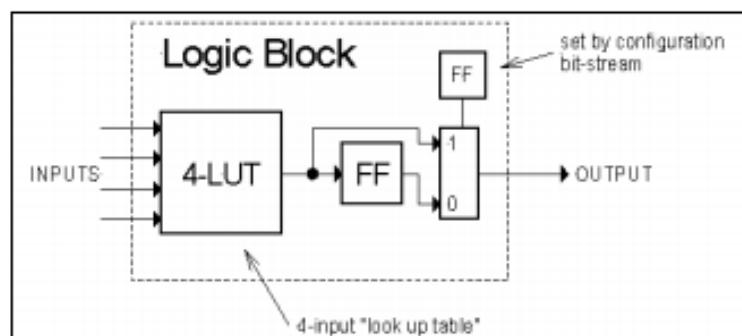


Figura 2.8 - Bloco lógico configurável.

- IOB - In/Output Block: andares de buffers que são responsáveis pela interface entre a lógica interna e as portas externas bidireccionais.
- Switch Matrix: elementos configurados pelo software de forma a configurar interligações entre LBs e outros dispositivos da FPGA de forma a se poder implementar o código fonte sintetizado.
- Circuitos de *Hardware* Dedicados: são blocos de lógica pré-fabricada com funções específicas (RAM, Processador, multiplicadores, etc.). Existem devido a frequente necessidade desse tipo de módulos nos sistemas comuns. Permitindo um melhor desempenho (totalmente dedicado para essa função) e a libertação dos restantes recursos para fins mais diversos. No modelo utilizado interessa focar como recursos deste tipo, as *blockRAM*, que são unidades de memória já implementadas para este fim. Esse assunto é visto em maior detalhe durante o desenvolvimento do trabalho prático, durante o estudo da escolha do tipo de memória a utilizar.

2.5.4 - Ficheiros *toplevel*, *ucf* e descrição *hardware*.

A descrição do comportamento desejado, chamada descrição hardware (HDL), pode ser feita numa vertente **comportamental** ou **estrutural**. No primeiro caso, descreve-se por meio de uma linguagem, o comportamento pretendido. No segundo caso, descreve-se as estruturas, células lógicas que se pretende sintetizar. Ambas têm por objectivo facilitar a sua elaboração e compreensão, pois são atribuídos nomes aos barramentos e registos implementados que permitem ao compilador (pessoa) visualizar com facilidade o código já desenvolvido. Durante o trabalho, o comportamento da FPGA é descrito por meio de uma descrição *hardware* em linguagem comportamental **Verilog**.

A descrição pode ser estruturada em diferentes módulos de forma a organizar ou instanciar várias vezes o mesmo módulo. Todos eles encontram-se dentro do **ficheiro *toplevel***. Esse ficheiro tem a particularidade de, tal como o nome o indica em inglês, conter o módulo “mestre”, onde a partir dele se vão instanciar os restantes módulos e ser a interface da descrição com o **ficheiro *ucf***.

Os pinos da FPGA estão fisicamente, logo inalteravelmente, ligados aos barramentos da placa de desenvolvimento. Cada um deles tem um nome específico, permitindo ao software, pelo modelo da FPGA em questão, saber onde esse pino se encontra. O ficheiro *ucf*, é o elo de ligação entre os barramentos IO da descrição *hardware* do *toplevel* com o *pinout* da FPGA, pois é nele que se associa o nome dos registos do *toplevel* aos pinos físico da FPGA.

A título de exemplo. Em particular, a linha 54 do ficheiro *ucf* indica a associação do pino físico “U4” da FPGA ao registo contido no *toplevel* como barramento para o exterior, definido com o nome *led4*.

```
54 - NET "led1"    LOC = "U4"    ;
```

2.5.5 - Câmaras

Para obter uma imagem estéreo foram disponibilizadas duas câmaras independentes de iguais características (mesmo modelo). O que permite desde já minimizar discrepâncias na aquisição das imagens, tais como:

- Diferenças de sensibilidade à intensidade dos pixels;
- Níveis de focagem diferentes;
- Início de varrimento em instantes diferentes.

O modelo utilizado é o M3188A da Quasar Electronics, ilustrado na Figura 2.9.



Figura 2.9 - Quasar Electronics - M3188A

Esse modelo fornece uma saída digital com débito constante de 8bit, isto para cada uma das câmaras.

Consta no relatório final sobre o trabalho já desenvolvido sobre a aquisição de vídeo, que para efeito de sincronização foi usado um relógio externo às câmaras, proveniente da FPGA, em vez do relógio local de cada uma delas, isto para anular os desfasamentos dos relógios e consequentemente as posições dos pixels actuais no barramento de saída. Para se poder manter as disposições das objectivas com geometria epipolar, foi fornecido juntamente com as câmaras, uma PCB (placa de circuito impresso) onde elas se encontram montadas.

A resolução de cada objectiva é de 640 X 480 pixels, a cadência ronda as 30 imagens por segundo. Valores que permitem uma qualidade de imagem e um efeito vídeo com níveis satisfatórios para uso geral.

2.5.6 - Placa VGA adicional

A interface VGA disponibilizada juntamente com a placa de desenvolvimento possui incompatibilidades de sincronismo com o monitor. Para ultrapassar esse problema foi adicionada uma placa VGA externa que se liga via um barramento de 50 pinos. Essa ligação foi possível utilizando um dos barramentos de uso geral disponíveis na placa de desenvolvimento.

A sua montagem será descrita em maior pormenor durante o desenvolvimento do trabalho.

2.5.7 - Software de desenvolvimento

Para o desenrolar de todo o processo de desenvolvimento sobre a FPGA é utilizando o programa fornecido pela Xilinx, o ISE WebPACK 8.2i (7). Este contém todos os diferentes *softwares* necessários.

Durante o desenrolar do projecto, são desenvolvidos os diferentes módulos no programa e seu respectivo *testbench* HDL na linguagem Verilog. O *testbench* é uma descrição hardware que invoca o módulo que se pretende testar. Para tal, aplica-se um conjunto de estímulos que simulam o comportamento real do sistema sobre o módulo em teste e observa-se os resultados obtidos nas saídas. Durante a **simulação**, é possível observar o estado dos registos internos assim como o das interligações, permitindo uma completa supervisão do sistema para a confirmação do seu bom funcionamento.

O passo seguinte consiste na **síntese** da descrição HDL para um **ficheiro configurador**. É esse ficheiro que configura a lógica interna da FPGA para que essa se comporte tal como pretendido no código HDL. O código configurador envia a programação JTAG (8), por meio de um cabo ligado à placa de desenvolvimento auxiliado de um software integrado no ISE, *IMPACT* (9).

2.6 - Conclusões

Neste capítulo foram introduzidos conceitos importantes para a compreensão do desenvolvimento do trabalho. Foram abordados assuntos na área do processamento de imagem e visão estéreo, tais como noções sobre os problemas introduzidos pela visão estéreo, calibração das câmaras segundo uma geometria epipolar, princípio do processo de correspondência, e conversão dos resultados para métricas reais.

Foi dada uma panorâmica do material disponibilizado para se poder desde já saber como se deve lidar com ele e quais são as suas limitações, mas também ter uma ideia dos resultados que se podem esperar.

Capítulo 3

Descrição da Solução

No levantamento do estado da arte sobre esse tema foram encontrados diferentes tipos de abordagens. Concluiu-se que não existe um método que sobressaia dos outros pela sua eficácia universal. Se por um lado, uns métodos são melhores para imagens mais homogêneas, por outro lado, outros são mais adequados para imagens com grande concentração de objectos.

O conceito base do processo de correspondência resume-se em retirar um conjunto de informações relativas a uma zona de uma das imagens e tentar encontrar essa mesma característica na outra imagem, isto é, encontrar a zona homóloga. Podemos tomar como padrão de pesquisa a intensidade ou cor dos *pixels* ou então a forma de um contorno de um determinado objecto entre outras possibilidades ou combinações dessas. O nosso estudo concentra-se em imagens a preto e branco deixando de lado o campo da análise das cores.

Entre vários métodos pensados, dois tipos destacaram-se pela simplicidade do conceito em se que baseiam, pois não utilizam algoritmos demasiados pesados, requisito essencial para se manter uma resposta considerada “em tempo real” como também devido a uma certa limitação de recursos da própria FPGA.

3.1 - Correspondência por áreas

Este método baseia-se na intensidade dos pixels, no caso das imagens a preto e branco. Para a determinação das disparidades são focadas partes de uma das imagens e comparadas várias áreas candidatas a homólogas na outra, tentando encontrar a mais semelhante.

Para tal toma-se uma das duas imagens como imagem de referência. Convém salvar, para já, que a escolha da **imagem de referência** não é aleatória. A sua escolha será justificada durante a descrição do algoritmo implementado. A atribuição do nome “imagem de referência” indica que esta imagem é tomada como padrão para comparação com a outra. É sobre esta imagem que se vão gerar as janelas de referência.

Durante o processo, cada janela de referência é sucessivamente comparada com janelas candidatas à janela homóloga da outra imagem. De referir que o leque de janelas candidatas está restrito a mesma **banda de observação**, isto é, situam-se sobre linhas de mesmo índice que as linhas da janela de referência. Isto é possível graças a particular calibração das

câmaras, geometria epipolar, como já foi falado. A Figura 3.1 auxilia a compreensão do processo.

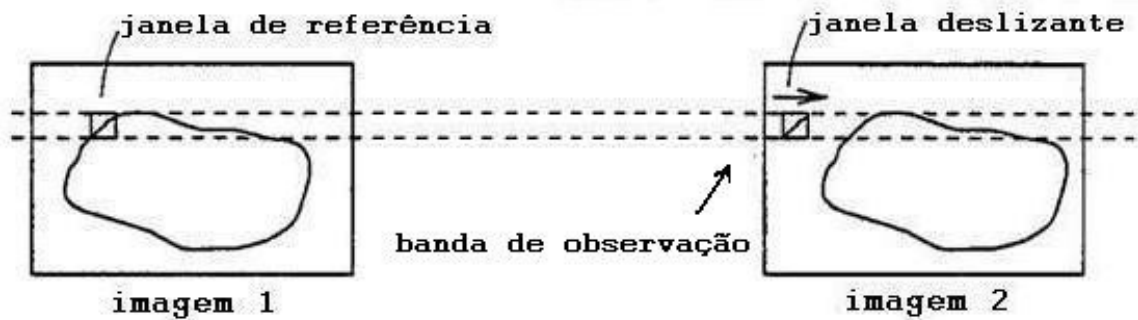


Figura 3.1 - Correspondência de janelas sobre a janela de referência.

Pode se observar na Figura 3.1, uma cópia da janela de referência a deslizar sobre a imagem 2, com o objectivo de demonstrar a evolução do percurso de correspondência para uma janela de referência.

Convém esclarecer que a totalidade da imagem padrão é varrida por sucessivas janelas de referência, onde cada uma dessas janelas origina o varrimento da respectiva banda de observação na outra imagem, como atrás descrito.

Pode desde já prever que este processo consome um número considerável de ciclos de relógio visto o elevado número de comparações em jogo. Pode-se referir que durante a concepção da implementação do algoritmo foram analisados casos que permitam reduzir significativamente esse número de comparações mantendo a integridade dos resultados.

Para cada comparação entre duas janelas, é obtido um valor de correspondência que indica o grau de afinidade entre elas. Para uma dada janela de referência, findo o varrimento do leque de janelas candidatas, é guardado como resultado final, o valor da **distância** que separa esse par de janelas, isto é, a disparidade entre as duas janelas que se mostram, pelo algoritmo, serem **janelas homólogas**.

No fim do varrimento de toda a imagem padrão, este processo gera um mapa de disparidade denso, pois a correspondência contempla todos os pixels existentes nessa imagem.

Existem várias métricas utilizadas para a determinação do nível de correspondência. Todas elas tentam determinar a melhor janela correspondente, homóloga. As métricas SAD, SSD e NCC usam a intensidade dos pixels da janela para medir a similaridade. A forma como eles combinam esses valores está resumida na Tabela 3.1.

Tabela 3.1 - Métricas de alguns algoritmos de correspondência por janelas.

Soma das Diferenças Absolutas	SAD	$\sum_{(u,v) \in W} I_1(u, v) - I_2(x + u, y + v) $
Soma da diferença dos quadrados	SSD	$\sum_{(u,v) \in W} (I_1(u, v) - I_2(x + u, y + v))^2$
Correlação Cruzada Normalizada	NCC	$\frac{\sum_{(u,v) \in W} I_1(u, v) \cdot I_2(x + u, y + v)}{\sqrt{\sum_{(u,v) \in W} I_1^2(u, v) \cdot \sum_{(u,v) \in W} I_2^2(x + u, y + v)}}$
Census	HAM	$\sum_{(u,v) \in W} Hamming(I'_1(u, v), I'_2(x + u, y + v))$

A métrica SAD foi a escolhida para indicar o nível de correspondência entre as duas imagens, por motivos de simplicidade do cálculo e, conseqüentemente, o baixo nível de ocupação de recursos face às restantes. Se, no entanto, na aplicação prática essa métrica não se mostrar válida para um determinado tipo de cenários ou para demasiados cenários, essa será substituída. No entanto, tal como a fórmula o indica, o valor da correspondência é obtido pela soma das diferenças dos valores dos *pixels* em posições idênticas no par de imagens. Essa métrica baseia-se no facto de no caso de os *pixels* coincidentes serem homólogos, a suas contribuições para o resultado será nula. No caso de coincidir com um pixel que na realidade é vizinho do homólogo, geralmente o seu valor não deve diferir muito, resultando um baixo acréscimo ao valor final. Para correspondências de janelas muito diferentes esse valor será elevado, indicando que as janelas são muito pouco parecidas.

3.2 - Correspondência por contornos

Este método determina pontos homólogos nas imagens com base nos contornos dos objectos neles contidos. O processo começa por tratar cada uma das imagens separadamente de forma a determinar os contornos existentes na imagem. O resultado é guardado sob a forma de dois níveis de intensidade, preto ou branco, para indicar se o *pixel* nessa posição é ou não um contorno do objecto. Esse resultado reduz a quantidade de dados envolvida para ser processada pelos andares seguintes mantendo, no entanto, um nível de informação suficiente para posterior análise.

De seguida pode-se optar por aplicar o método de correspondência por áreas ou fazer uma correspondência directa sobre a totalidade da linha. Essa decisão será analisada com maior detalhe de seguida.

Um **contorno** pode ser descrito como sendo uma variação abrupta de intensidade dos valores dos *pixels* devido a mudança de objecto. De notar desde já que essa análise é possível com imagens onde a diversidade de cor entre objectos na imagem seja notória para se obter a necessária transição de cores. Notar também que, embora geralmente na literatura se trate o problema de contornos de uma forma ideal, na verdade estes apresentam na realidade vários problemas tais como:

- Desfoque da imagem devido própria diferença de planos de distância no cenário.
- Atenuação de transições devido às sombras causadas pelos próprios objectos ou objectos adjacentes.
- Transições indefinidas devido a objectos esféricos ou fortemente arredondados.
- Reflexões já atrás descritas, tais como reflexão especular ou difusa.

Todos estes problemas diminuem a eficiência do método, podendo em certos caso o contorno não ser detectado.

Se observarmos uma janela 3x3 numa imagem, de um modo geral simplista, o contorno vertical será quando se verificar uma variação significativa dos valores dos *pixels* segundo o eixo horizontal e vice-versa para um contorno horizontal.

Sejam os *pixels* de índice N, S, E, W os *pixels* vizinhos do pixel central (C) tal como se ilustram na Figura 3.2.

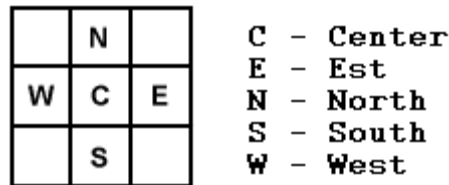


Figura 3.2 - Janela de correspondência 3x3.

Existem vários métodos de detecção de contornos mas grande parte deles podem ser incluídas em dois subgrupos, “pesquisa” (*search-based*) ou cruzamento-zero (*zero-crossing based*) (10).

- O método baseado em “pesquisa” tira a informação da primeira derivada do pixel em análise aplicando a equação exposta na tabela seguinte sobre o cálculo das primeiras derivadas.

Tabela 3.2 - Cálculo da primeira derivada.

1ª Derivada
$Dx = E - W$
$Dy = N - S$

De seguida procura os máximos e mínimos locais na linha da imagem onde se encontram os contornos dos objectos, marcando-os no resultado como valor de contorno.

- O método de **cruzamento-zero** procura os zeros da segunda derivada, aplicando a equação dada na tabela seguinte sobre cálculo das segundas derivadas, para encontrar o ponto de contorno.

Tabela 3.3 - Cálculo da segunda derivada.

2ª Derivada
$D2x = E - 2C + W$
$D2y = N - 2C + S$

Tal como no método anterior, esses pontos são guardados para serem utilizados para a aplicação desejada.

O resultado esperado será uma imagem com um aspecto semelhante ao da Figura 3.3, onde à esquerda temos a imagem original e à direita o resultado obtido com o algoritmo de determinação dos contornos.

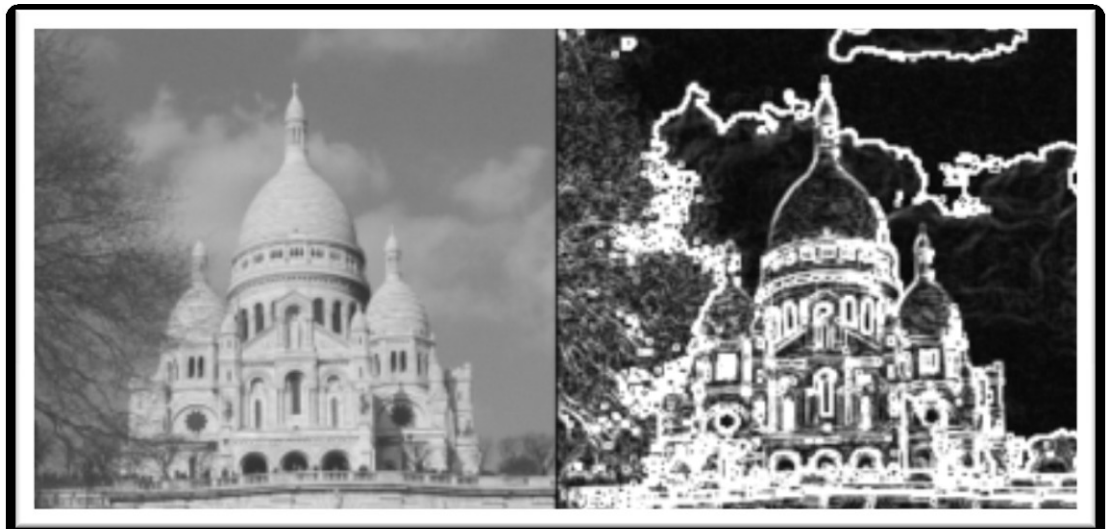


Figura 3.3 - Imagem original e resultado da detecção de contornos (11).

Concluindo, a detecção de contornos gera uma imagem com informação sobre os contornos dos objectos na imagem. No nosso caso, por motivos de resposta em tempo real, este processo ocorre em paralelo para ambas as imagens. Sobre este resultado, aplica-se a correspondência por áreas para se determinar as disparidades entre **contornos homólogos** dos objectos. Por este motivo este método pode ser visto como sendo um filtro que se pode aplicar sobre o par de imagens antes de ser processado por correspondências por áreas.

De notar que, aplicar primeiramente este filtro, em vez de correr directamente o processo de correspondência por áreas, deve permitir trazer vantagens ao nível de sucesso das correspondências, pois os contornos mostram-se mais sobressaídos. Interessa sobretudo lembrar que a sua determinação provindo da informação da vizinhança do pixel em causa e não apenas dele mesmo, leva a que um pequeno nível de ruído de delimitação do contorno na imagem seja minimizado.

Numa fase inicial, decidiu-se implementar o algoritmo de correspondência baseado em áreas. A escolha desse algoritmo, em detrimento do algoritmo de correspondência por contornos, justifica-se pelo facto de ele poder ser utilizado directamente sobre a imagem mas com algumas modificações pode ser aplicado no método de determinação de contornos.

Capítulo 4

Desenvolvimento

4.1 - Introdução à implementação prática

Como ponto de partida para o desenvolvimento do trabalho foram fornecidas as seguintes ferramentas:

- Placa de desenvolvimento Xilinx Spartan 3 Development Kit;
- Cabos de programação;
- Computador com o software ISE da Xilinx e acesso à Internet.

Juntamente com o material, foi facultado o projecto final do aluno João Alves (12) que se ocupou deste projecto anteriormente. Nesse projecto encontravam-se, entre outros, o ficheiro *ucf* e os ficheiros com a descrição Verilog dos diferentes módulos já desenvolvidos.

4.1.1 - Barramentos necessários

Do conjunto de módulos já desenvolvidos, somente algumas saídas e entradas requereram especial atenção, nomeadamente as que dizem respeito à disponibilização síncrona dos *pixels* de entrada e controlo da saída VGA..

Tabela 4.1 - Descrição dos barramentos mais relevantes.

Pino/barramento	Descrição
<i>clock</i>	Sinal de relógio mestre a 100MHz
<i>reset</i>	Sinal de “reset” inicial do próprio sistema para efeitos de sincronização
<i>Pin1 e Pin2</i>	Saída do valor dos pixels de cada câmara
<i>pclken</i>	Sinal que indica o momento onde os valores dos <i>pixels</i> são válidos para leitura
<i>X e Y</i>	Coordenadas (X, Y) dos <i>pixels</i> actuais fornecidos pelas câmaras. ¹
Barramento VGA	Sinais do barramento de controlo de saída para a placa VGA (inclui o barramento com o valor do <i>pixel</i> a colocar no monitor).

¹ Uma vez sincronizadas as câmaras, as posições de ambos os *pixels* das câmaras são idênticas. Teste e demonstração poderão ser encontrados na secção 4.1.3.2.

A pesquisa desses barramentos foi centrada no ficheiro *toplevel*, chamado *stereocam*, onde se instanciam os diferentes módulos, onde se podem consultar os seus barramentos e suas interligações

Numa análise criteriosa foram determinados os barramentos que iriam fornecer os dados e sinais de controlo das entradas e das saídas necessárias tais como os apresentados na Tabela 4.1.

4.1.2 - Instalação da placa VGA

Como foi descrito anteriormente, a placa VGA disponibilizada tinha sérios problemas de sincronismo com determinados monitores. Posto isto, optou-se por se substituir a placa por um modelo compatível. Como era previsível, sendo um modelo diferente, esta placa possui um *pinout* diferente.

De referir, antes de mais que nos barramentos de uso geral disponibilizados pela FPGA, certos pinos são exclusivamente para alimentação e ligação à massa. Ora esse pormenor obrigou a que se tivesse em atenção que esses pinos fossem coincidentes com os pinos de alimentação da nova placa.

O passo seguinte foi a descoberta do *pinout* de ambas as placas VGA para se poder efectuar a devida troca, por comparação do *pinout*, a nível do ficheiro *ucf* da FPGA.

Uma vez reconfigurado o barramento de saída, foi necessário adaptar a ligação física regulando o nível a que fica a placa, de forma a garantir uma utilização segura, mas também colocar um conector compatível que permita interligar os dois barramentos.

Uma forma de testar a ligação e o funcionamento correcto dos barramentos de saída VGA consiste em enviar para o ecrã uma imagem toda branca, bastando para isto, colocar os valores das três cores (R, G, B) com o valor máximo. Os resultados provenientes deste teste foram os esperados.

4.1.3 - Teste dos barramentos disponibilizados

Para se poder utilizar os barramentos de entrada de uma forma confiante é necessário verificar que o seu funcionamento corresponde ao esperado. Para tal foram desenvolvidos uma série de testes que se passam a demonstrar.

4.1.3.1 - Leitura de ambas as câmaras

Um segundo passo consistiu na implementação da leitura directa de ambas as câmaras para o monitor. Com este teste verifica-se o correcto funcionamento dos pinos *clock*, *reset* e *pclken* assim como dos barramentos *Pin1*, *Pin2* e *Pout*. Tendo a saída apenas um barramento, *Pout*, optou-se por se adicionar um botão de uso geral para se poder seleccionar qual das duas imagens se pretende visualizar. Para tal utilizou-se um dos botões, de uso geral, existentes na placa de desenvolvimento, neste caso foi tomado o botão *sw3*.

Notar que para tal mais uma vez é necessário alterar o ficheiro *ucf*, para associar o botão *sw3* ao registo *switch*, adicionado no ficheiro *toplevel* como sendo uma saída.

Aproveita-se este caso para dar um exemplo simples sobre a relação entre ficheiro ucf e o toplevel. Segundo a *datasheet* da placa de desenvolvimento, o nome do pino físico na FPGA que liga ao botão sw3 é o fio H15, como se pode ver no exemplo seguinte.

```
NET "switch" LOC = "H15" ;
```

Esta linha associa o pino H15, botão sw3, ao registo chamado *switch* no *toplevel*. A descrição Verilog implementada para o teste que foi embutida no módulo *stereocam* foi a seguinte:

```
. . .
if(pclken)
begin
  if(sw3)
    Pout <= Pin1;
  else
    Pout <= Pin2;
  end
. . .
```

O teste da colocação das imagens provenientes das câmaras e sua comutação com base no estado de pressionamento do botão sw3 apresentou os resultados esperados.

4.1.3.2 - Coordenadas X e Y dos *pixels* actuais

Para se garantir que os índices dos *pixels* eram iguais em ambas as câmaras, implementou-se uma condição que coloca no ecrã um pixel branco no caso de as posições fornecidas pelos módulos de sincronização das câmaras serem iguais ou preto no caso contrário.

Para o teste, a descrição foi alterada para:

```
. . .
if(pclken)
begin
  if(X1 ==X2 & Y1 == Y2)
    Pout =8'FF;
  else
    Pout = 8'h00;
  end
. . .
```

O resultado foi uma imagem totalmente branca como era suposto.

Este resultado permitiu que a partir desse momento se possa usar apenas um referencial de coordenadas para ambos os *pixels* provenientes das câmaras. Aleatoriamente foram escolhidas as provenientes do módulo responsável pela câmara esquerda.

4.1.4 - Atraso das saídas VGA

De notar que o sinal *pclken* fica activo durante um ciclo de relógio para a chegada de cada par de *pixels*. Este instante é o adequado para se colocar o valor de *Pout* na entrada da placa VGA. Nesse ciclo de relógio a saída ainda vai conter o valor *Pout* anterior, devido ao atraso do próprio registo. Este atraso acarreta um outro problema a nível das coordenadas do pixel actual, pois neste instante estes valores estão adiantados em relação ao *Pout*. Ter acesso a valores de coordenadas do pixel actual sempre correctos na saída foi considerado um parâmetro necessário para efeitos de ligação de futuros módulos em série com o deste trabalho. É portanto fundamental garantir o perfeito sincronismo entre *Pout* e suas coordenadas.

O método implementado para solucionar esse problema consistiu em se fazer passar a totalidade do barramento de controlo da placa VGA assim como as coordenadas do *pixel* actual por um andar de registos. A Figura 4.1 representa a implementação do módulo de atraso, *delayVideo*, criado para separar do conteúdo do módulo geral.

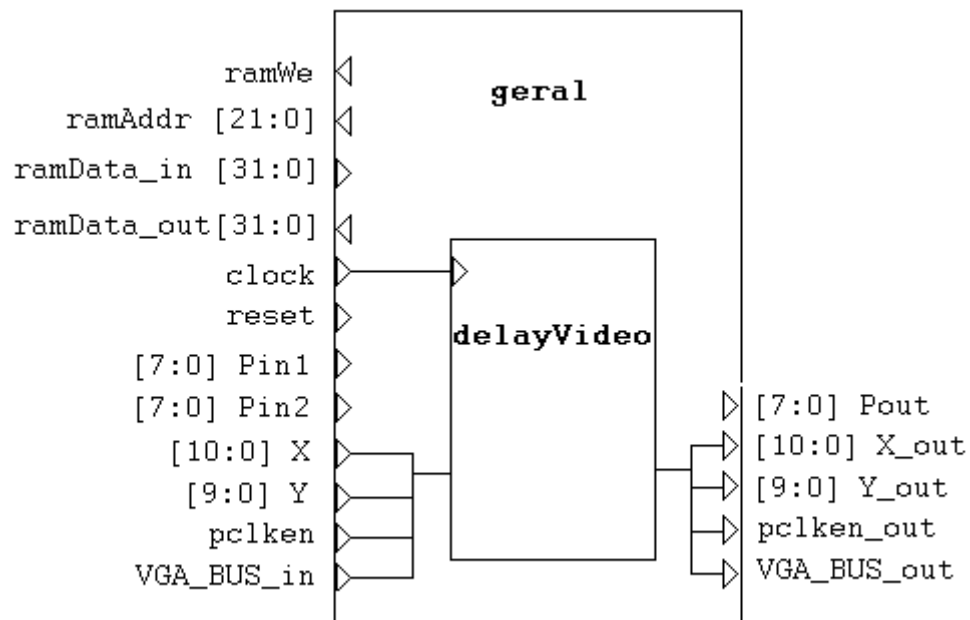


Figura 4.1 - Integração do módulo *delayVideo*.

Esse módulo vai introduzir um atraso de um ciclo de relógio, o que vai fazer com que o valor de *Pout* se mantenha síncrono com o barramento na saída e com as coordenadas da posição do pixel actual *Pout*.

4.1.5 - Escolha da memória

Durante o levantamento do estado da arte tornou-se óbvio que qualquer processamento da correspondência não pode ser feito directamente sobre os valores do par de *pixels* que chega das câmaras, pois consiste em comparar um ou um conjunto de *pixels* com vários ou vários conjuntos de *pixels* da outra imagem, tal como foi visto durante a exposição dos diversos métodos. É necessário ter-se em memória, um conjunto de *pixels* já passados, pertencentes ao mesmo par de imagens, como uma forma de poder voltar a aceder a esses *pixels*.

Com base nos dispositivos disponibilizados pela placa de desenvolvimento temos várias alternativas possíveis:

- Gerar uma memória na própria FPGA.
 - Block RAM
 - RAM distribuída (Com base no uso dos CLBs)
- Utilizar a SDRAM disponível na placa de desenvolvimento.

A escolha entre uma das hipóteses depende de dois aspectos: tempos de acessos e espaço disponível. Os acessos à memória, na própria FPGA, são mais simples e rápidos do que o da SDRAM. No entanto a SDRAM disponibilizada permite um armazenamento até 2MB em quanto que gerar uma memória com base no uso de CLBs ocupa recursos de uso genérico da FPGA que podem ser necessários para a implementação do algoritmo concretamente dito.

Existe também a possibilidade de se usar recursos dedicados para esse fim na própria FPGA. Nomeadamente o uso das *Block RAM* que são blocos existentes na FPGA exclusivamente para usar como memória auxiliar. Segue-se uma análise mais criteriosa, por pontos, do espaço necessário para o armazenamento dos diferentes dados.

Toma-se como exemplo a resolução das imagens de 640X480 como referência para a determinação do espaço necessário em memória por ser a resolução oferecida pelos sensores.

Cada *pixel* é formado por um byte, cada linha 640 *pixels*, cada imagem 480 linhas, cada correspondência é feita com duas imagens, o que totaliza 600KB. A estes dados juntam-se o resultado obtido pela correspondência. Interessa para já saber sobre a geração do resultado que cada conjunto de 8 linhas na imagem vai gerar uma imagem de tamanho sensivelmente idêntico as imagens de origem. É necessário assim reservar mais 300KB de espaço para a colocação desse resultado.

São necessários 900KB em memória para armazenar o par de imagens e o resultado.

Espaço requerido para armazenar um par de imagens

- 1 linha = 640 pixels;
- 1 imagem = 480 linhas;
- 1 processamento = 2 imagens;
- Espaço necessário = $640 * 480 * 2 = 600\text{KB}$.

Espaço requerido para armazenar os resultados da disparidade:

- Análise para uma imagem de dimensões (W x H).

- Número de linhas = $\lfloor H/8 \rfloor * 8$
- Número de Colunas = $\lfloor (W - 7)/4 \rfloor * 4$

Estas equações explicam-se com base no modo de cálculo. O número de linhas é devido ao varrimento da imagem ser feito de por conjuntos de 8 linhas. Logo o último conjunto completo de 8 linhas será o último varrimento efectuado sobre a imagem. Sendo o deslizamento das janelas feito coluna a coluna, o último cálculo será efectuado aquando a janela chegar a última coluna. Tendo a janela um tamanho de 8 pixels (colunas) a última atribuição acontece para a oitava coluna a contar do fim, coluna número W-7. No entanto, o resultado das disparidades só é enviado se se totalizar um conjunto de 4 resultados, o que se traduz na parte inteira de (W-7)/4. Por fim cada envio contém 4 resultados, o que significa que cada um representa 4 resultados independentes.

- Caso particular de uma imagem de 640 x 480.

- Número de linhas = 480;
- Número de colunas = 633;
- Espaço necessário = $633 * 480 = 296.72\text{KB}$.

Total necessário

- $600\text{K} + 296.72\text{K} = 896.72\text{KB} = 0.86\text{ MB}$.

Para poder decidir convenientemente sobre o tipo de memória a escolher é necessário saber o que é disponibilizado tanto ao nível da FPGA como da placa de desenvolvimento.

A Tabela 4.2 fornece a informação sobre os recursos disponibilizados por vários modelos de Spartan 3, onde entre outras se encontra o modelo utilizado neste trabalho, XC3S1500.

Tabela 4.2 - Características dos recursos disponíveis de várias Spartan 3 (12).

Device	System Gates	Equivalent Logic Cells ¹	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50 ²	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200 ²	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400 ²	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000 ²	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344

Desta tabela interessa focar o número máximo de RAM distribuída que se pode implementar, 208Kbits, e a capacidade de memória Block RAM existente, 576Kbits, isto é, 72KB.

- CLB / RAM distribuída.

A Figura 4.2 ilustra a estrutura interna de uma CLB da nossa FPGA, mostrando os seus diversos recursos.

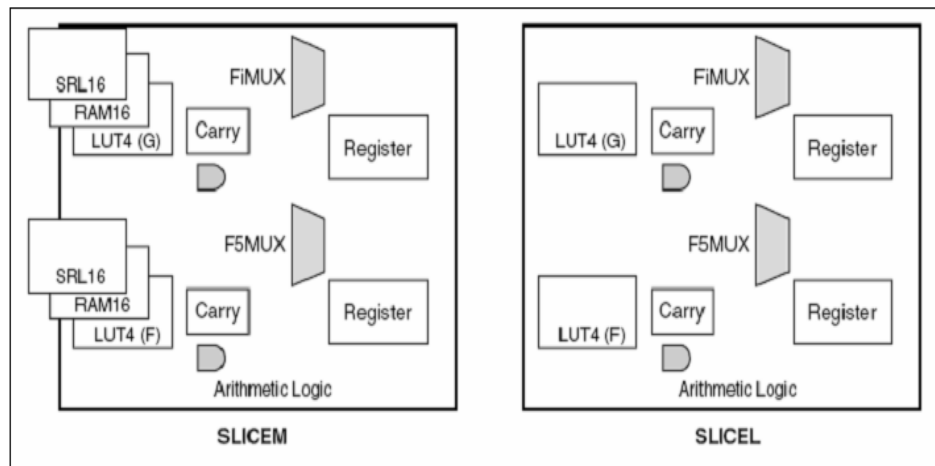


Figura 4.2 - Recursos internos de uma CLB (12).

Como se pode observar cada CLB possui duas *slices*, *SLICEM* e *SLICEL*, onde cada uma delas contém duas LUT de quatro entradas, dois elementos de memória (registos) e elementos de lógica. Cada CLB dispõe então de um total de 4 registos de 1bit cada.

Visto isso pode prever-se que se vá ocupar na FPGA, só a nível de armazenamento, um total de cerca de 1 836 500 CLBs ($896.72\text{KB} \times 8 / 4$). Valor largamente superior aos 3328 disponíveis, inviabilizando por completo esta hipótese.

- *Block RAM*

É dito que se pode conter até um total de 576Kbit o que equivale a 72KB. Valor este muito inferior ao 896.72KB necessários para guardar os dados. Poder-se-ia pensar em armazenar apenas os valores dos resultados, pois baixaria o número de acesso à memória libertando-a para outras possíveis aplicações, mas tal conteúdo exige 300KB de espaço, o que é largamente superior aos escassos 72KB disponíveis.

- *SDRAM*

A placa de desenvolvimento possui uma memória SDRAM (13) que permite um armazenamento até 2MB. O que representa um espaço superior aos 0.86MB necessários para armazenar a totalidade dos dados em questão.

Com base nesse estudo ficou posto de parte a hipótese de se armazenar os dados na própria FPGA em detrimento da escolha da memória SDRAM. É necessário agora determinar como aceder à esta memória. Nesse sentido, foi levantado o modelo presente na placa de desenvolvimento e estudada a sua *datasheet* (13) de forma a se tornarem conhecidos os seus tempos e modos de acesso.

4.1.5.1 - Características e ciclos de acesso à SDRAM

Para um melhor enquadramento segue-se Figura 4.3 que representa o esquema lógico da memória assim como a Tabela 4.3 com a descrição dos barramentos.

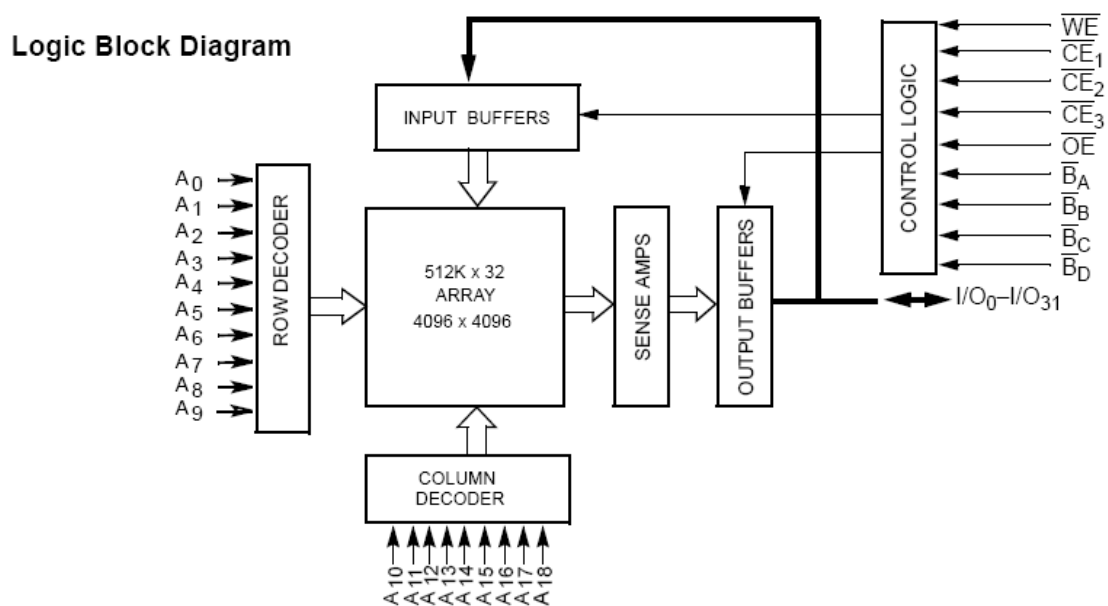


Figura 4.3 - Diagrama lógico dos blocos da memória (13).

Tabela 4.3 - Descrição dos barramentos e pinos de controlo da SDRAM.

Nome	Descrição dos pinos	Observações
A [18:0]	Barramento de endereço	Endereçar a escrita ou leitura dos dados
I/O [31:0]	Barramento de dados bidireccional	Entrada ou saída de dados
BA, BB, BC, BD	Seleccionador da parcela do I/O	Permite seleccionar parte do barramento I/O
CE1, CE2, CE3	Chip enable	Para activar/desactivar o bloco de memória
OE	Output enable	O buffer tri-state coloca o barramento de saída em alta impedância quando a nível alto. Permite a partilha do mesmo barramento de dados por vários periféricos.
WE	Write enable	Activa a escrita e liberta a linha I/O para se poder partilhar o mesmo barramento por vários periféricos.

A consulta da *datasheet* mostra que existem vários tipos de ciclos para a escrita e leitura de dados na SDRAM. Após análise, optou-se pelos métodos onde se passa de um ciclo de escrita ou leitura apenas pelo controlo do pino **WE**, respeitando obviamente os respectivos tempos de acesso. Métodos que envolvem também outros pinos de controlo existem para permitir uma partilha do mesmo barramento de endereço e dados por várias memórias ou periféricos. No caso dessa placa de desenvolvimento estes barramentos podem ser partilhados com a memória Flash. Não sendo do nosso interesse, esses tipos de acesso não serão tidos em atenção.

Notar que, com estes dois ciclos escolhidos, os bits de controlo OE, CS, B estão sempre activos (ao nível lógico baixo). Apenas o bit WE, muda de valor, indicando que se trata de uma escrita ou leitura. Quando activo (nível lógico alto) permite a escrita na memória. Caso contrário, dá-se o processo de leitura, a memória coloca no barramento de dados o valor existente na posição indicada no barramento de endereço. A Figura 4.4 ilustra o ciclo de leitura.

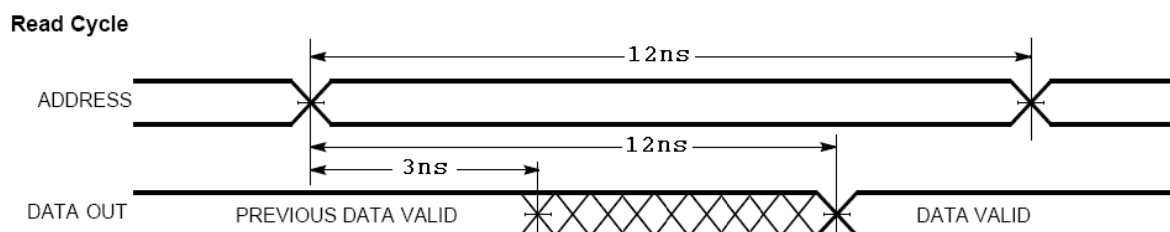


Figura 4.4 - Ciclo de leitura (com controlo WE desactivo) (13).

Pela figura é visível que o tempo mínimo necessário é de pelo menos 12 ns para se concluir o processo correctamente um processo de leitura. A cadência da FPGA é de 100 MHz,

o que equivale a um período de 10ns. Assim, torna-se necessário garantir um tempo de espera de pelo menos dois ciclos de relógio (20ns) para se obter o valor na saída da memória. O tempo mínimo aqui referenciando como sendo 12ns, é o intervalo de tempo a partir do qual os valores de endereço e controlo chegam à entrada da memória e a saída é apresentada pela memória. Um terceiro ciclo de relógio adicional, tem de ser contabilizado para a propagação destes valores desde a saída do registo até à entrada da memória e da saída da memória até à entrada do registo encarregue da recepção dos dados. Espera-se que a soma destes dois tempos de propagação seja inferior a um ciclo de relógio. Posto isto, conclui-se que um ciclo de leitura consuma três ciclos de relógio.

Apresenta-se agora um ciclo de escrita controlado pelo pino WE. A Figura 4.5 ilustra o pior caso possível, isto é, a alternância de um ciclo de leitura com um ciclo de escrita que obriga a espera da colocação do barramento de dados em alta impedância. O ciclo de escrita projectado baseia-se neste caso, garantindo assim o seu correcto funcionamento em qualquer caso.

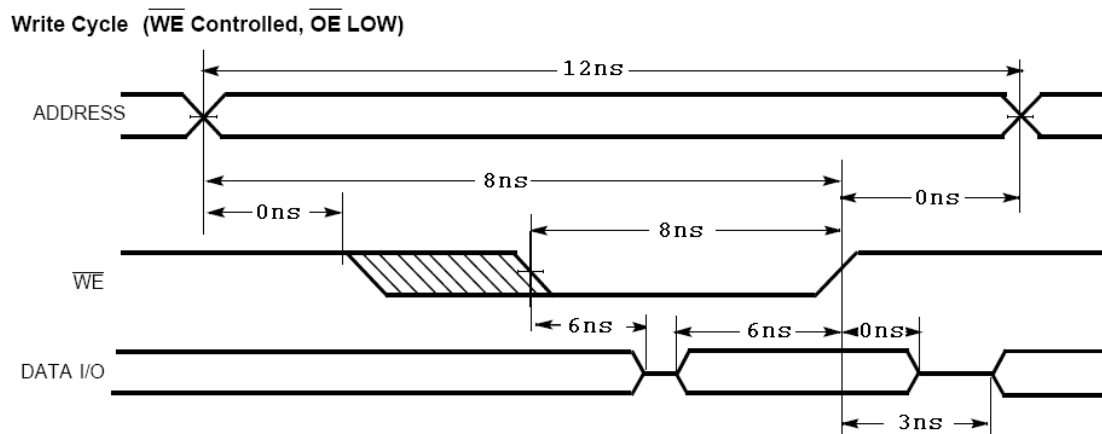


Figura 4.5 - Ciclo de escrita com controlo WE (13).

Inicialmente, tal como no ciclo de leitura, é necessário um ciclo de relógio para colocar os valores de endereço e controlo (WE activo - nível lógico baixo) na entrada da memória. No caso do ciclo de escrita, é preciso garantir que o barramento I/O deixe de conter os valores à saída da memória para permitir que a linha passe para o nível de alta impedância, isto é, não ocorrer contenção. Com o pino WE activo, a memória demora no máximo 6ns a colocar o barramento I/O em alta impedância, como se pode ver pela Figura 4.5. Para tal um segundo período de espera é fornecido. Este ciclo sendo de 10ns, garante uma margem de tempo desde a estabilização dos comandos na entrada da memória até a colocação do estado de alta impedância. Depois desse intervalo de tempo, é enviado para o barramento I/O, os valores para escrita na memória. Com o intuito de respeitar o tempo de *setup* (6ns), vai ser gasto mais um terceiro ciclo de relógio. Para assegurar o tempo de *hold*, é necessário desactivar o controlo WE pelo menos 3ns antes de se alterarem os valores de endereço ou dados. Como se pode comprovar através do fluxograma da Figura 4.19, existe sempre pelo menos um ciclo de relógio entre o fim de um ciclo e o início de outro, seja ele de escrita ou leitura. Assim sendo, o tempo de *hold* é garantido durante esse ciclo de relógio onde o valor do endereço é mantido constante. De notar que o tempo de propagação dos dados pelos barramentos não

obriga a colocação de ciclos de espera, pois o tempo que o endereço e os comandos demoram a chegar a memória é idêntico a propagação dos dados.

Concluindo, uma escrita na memória requer ao processo, um total de três ciclos de relógio.

4.1.5.2 - Integração da SDRAM no projecto

A próxima etapa consiste agora em conectar os barramentos da SDRAM ao *toplevel* do nosso projecto. Para tal foram associados os pinos da FPGA, aos registos implementados no módulo da descrição *hardware*, através do ficheiro *ucf*, esse vai fazer corresponder no ficheiro *toplevel* aos barramentos da memória os seguintes portos:

```
output  sram_OE, sram_WE, sram_CS, FLASH_CS;
output  [ 3:0]      sram_EN;
output  [21:0]      MEM_ADDR;    //barramento de endereço
inout   [31:0]      MEM_D;        // Barramento I/O
```

Com base nos tempos de acesso da memória, foi elaborada uma máquina de estados simples que apenas colocasse um valor num endereço fixo e logo de seguida fosse ler esse valor. Caso esse valor fosse igual, um pixel branco, seria enviado para todo o ecrã. Caso contrário, seria um pixel preto. Tendo assim uma forma visível e prática de saber se a memória está a ser acedida devidamente. O resultado foi negativo, tal não aconteceu.

Perante o resultado, optou-se por aumentar os tempos de acesso para garantir que o problema não fosse por se utilizar tempos muito reduzidos para a devida propagação nos barramentos. Mesmo quadruplicando esses tempos o problema manteve-se.

Durante várias semanas foi pesquisado exaustivamente a fonte do problema. Pensou-se em razões do possível erro, como por exemplo, verificar se o ficheiro *ucf* estaria errado. Mas como previsto ele encontra-se conforme o fornecido pela Xilinx num projecto teste já elaborado por eles. Os nomes dos pinos foram verificados no diagrama de interligações da placa de desenvolvimento fornecida.

Após reflexão resta-nos a hipótese de a placa de desenvolvimento estar mal configurada para a memória SDRAM. Assim sendo foi necessário rever a sua documentação para ver qual poderia ser a causa do problema. Resta também a hipótese de a memória SDRAM estar danificada por qualquer motivo. Até à data não foi solucionado o problema, inviabilizando o decorrer prático do trabalho sobre a FPGA.

De forma a não se consumir mais semanas de trabalho com esse percalço, decidiu-se então prosseguir com o desenvolvimento do algoritmo de reconhecimento e optar por testar a totalidade do trabalho em ambiente de simulação. Como será explicado mais adiante a memória foi substituída por uma virtual no ambiente de simulação, permitindo assim o tratamento de imagens estereoscópicas reais.

4.1.5.3 - Armazenamento das imagens e resultados

Como foi visto anteriormente existe a necessidade de se armazenar os dados provenientes das câmaras para serem posteriormente tratados. Pretendendo-se recolher os resultados de uma forma ordenada e sistemática, o resultado gerado também vai ser armazenado na memória. Optou-se, assim, por uma organização dos dados por paginação, isto é, existe uma alocação da memória onde os dados são colocados consoante a sua origem.

A título de exemplo, os *pixels* provenientes da imagem esquerda são armazenados na página desde o endereço 0000h até ao 12C00h inclusive. Esse espaço é exclusivamente reservado para os *pixels* provenientes dessa câmara. O mesmo processo acontece com a imagem da direita e com o conjunto de resultados gerados, tal como demonstra a Figura 4.6.

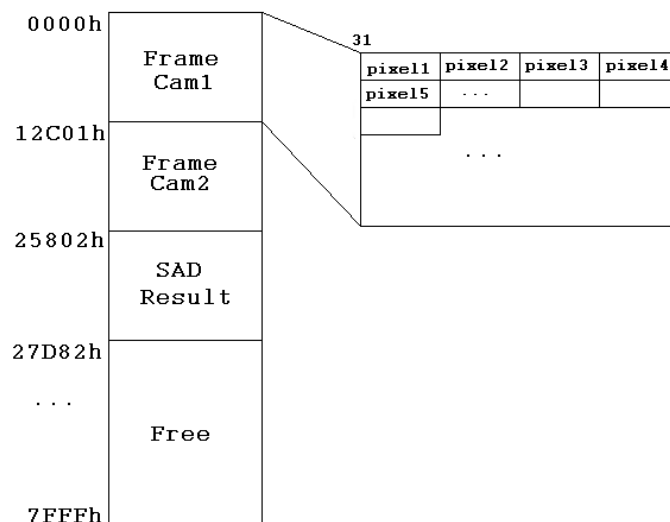


Figura 4.6 - Organização da memória SDRAM.

4.1.6 - Escolha do tamanho das janelas de observação

Surge agora a questão sobre o tamanho apropriado das janelas. Este problema divide-se em duas partes. A primeira sobre o tamanho das janelas de observação, a segunda, sobre o tamanho das janelas de correspondência. Entende-se por janelas de observação o conjunto de janelas de correspondências tratadas em paralelo, isto é durante o varrimento de uma banda de observação. Embora o conjunto de janelas de correspondência sejam varridas ao mesmo tempo os valores são geridos independentemente uns dos outros. Este processo é explicado em maior detalhe na secção 4.3.6. Uma janela de correspondência é uma linha completa da janela de observação, tal como mostra a Figura 4.7.

janela de correspondência 1
janela de correspondência 2
janela de correspondência 3
janela de correspondência 4
janela de correspondência 5
janela de correspondência 6
janela de correspondência 7
janela de correspondência 8

Figura 4.7 - Janela de observação.

O tamanho da janela da observação prende-se com a escolha do número de janelas de correspondência a incluir durante um varrimento. É tentador dizer que quantas mais linhas melhor, pois menor será o número de varrimentos são feitos sobre o par de imagens. No entanto tal acarreta dois problemas. Um deles, o peso da lógica envolvida no módulo de cálculo e registos auxiliares. Um outro problema reside na totalidade do varrimento da imagem, pois apenas se geram bandas de observação que ainda caibam totalmente dentro da imagem. Seja a variável N o número de linhas da janela de observação. Sendo este trabalho aplicável para qualquer tamanho de imagem, é muito provável que fique por processar varias linhas, no pior caso as últimas $(N-1)$ linhas da imagem. Assim sendo estipulou-se desenvolver janelas de com 8 linhas por se considerar um valor razoável.

Resta agora definir o tamanho da janela de correspondência. O aumento do comprimento da janela implica uma redução da resolução sobre os objectos perto e junto das objectivas.

Considerando o exemplo de dois objectos relativamente perto um do outro, num plano próximo das câmaras, é de esperar que eles tenham posições relativas muito diferentes entre as imagens. Se a janela de observação for muito grande, a sensibilidade à distinção dos dois objectos vai ser baixa. O nível de discrepância entre as imagens vai ser atenuado pelos *pixels* da zona circundante, tendo esta, uma elevada contribuição percentual na janela e sendo semelhantes nas duas imagens, existirá um grande conjunto de janelas com resultados muito semelhantes entre elas.

Utilizando janelas de tamanho reduzido esse problema será atenuado, pois os pixels semelhantes, do mesmo objecto, têm agora um peso relativo na janela mais elevado. Logo, os níveis de disparidade com as janelas candidatas serão mais acentuados, permitindo assim uma melhor correspondência da totalidade da imagem, isto é, uma maior resolução. Por outro lado, um par de janelas menor propicia a que a se encontre mais frequentemente a mesma sequência de pixels durante o varrimento da linha de observação, o que leva a um aumento de correspondências falsas. Se as janelas forem demasiado pequenas esse acontecimento pode levar a que não se obtenha resultados correctos.

Concluindo, numa primeira fase, definiu-se uma área de janelas de observação com um tamanho de 8 por 8 *pixels* sendo cada uma das 8 linhas uma janela de correspondência independente das restantes 7. Sendo a dimensão das imagens devolvidas pelas câmaras de 640x480, a escolha da dimensão das janelas de observação permite que se faça um

varrimento total da imagem sem deixar nenhuma linha por analisar, isto é, o número de linhas da imagem é múltiplo da largura da janela. Este parâmetro leva a que a imagem fica dividida em 60 bandas de observação, varrendo a totalidade das linhas do par de imagens ($60 \times 8 = 480$ linhas).

4.1.7 - Redução do número de ciclos de relógio

Como já foi referido o parâmetro mais importante no desenvolvimento do método reside no número de ciclos de relógios necessários para totalizar a correspondência do par de imagens. Esse número foi reduzido por 8 optando-se por tratar logo 8 linhas de cada vez, usando janelas de observação de tamanho 8×8 pixels. De notar que sendo linhas epipolares estamos a proceder a uma processamento paralelo, o que faz com que uma janela de referência apenas possa ter a sua janela homóloga na banda de observação do mesmo índice de linha, não se tendo assim de varrer várias bandas na tentativa de encontrar a janela homóloga graças à colocação das câmaras em orientações epipolares. No entanto, fazendo uma análise das imagens e olhando para as suas características, observa-se que uma forma possível de reduzir o número de ciclos de relógio pode ser feita estreitando o leque de janelas candidatas à janela homóloga de uma dada janela de referência. Nesse sentido temos duas características próprias desse tipo de par de imagens, em perspectivas lado a lado.

- 1-Escolha da imagem de referência

Durante a apresentação dos métodos de correspondência foi dito que a escolha da imagem de referência é importante, pois depende do algoritmo de varrimento utilizado. Veja-se a Figura 4.8, onde se representa uma hipotética cena filmada por duas câmaras.

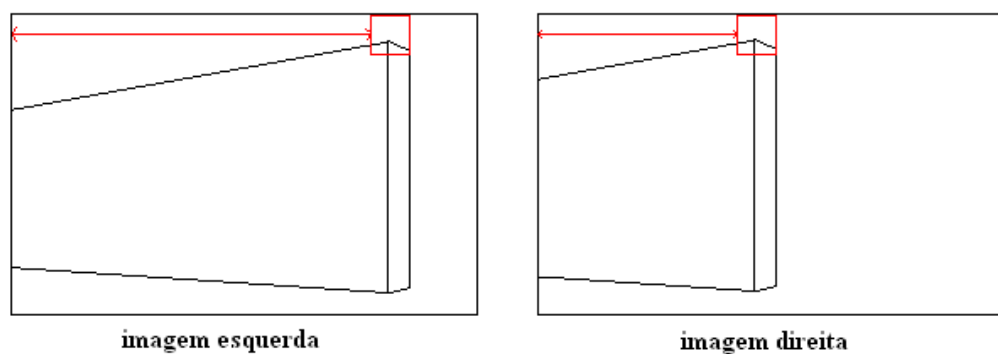


Figura 4.8 - Escolha da imagem de referência.

As imagens estão respectivamente à esquerda e à direita tal como na realidade se filmariam. Como se pode ver existe uma deslocação média da cena entre as imagens. A imagem direita parece ser uma deslocação da imagem esquerda para a esquerda.

Fixando uma janela padrão na imagem esquerda, facilmente se conclui que a janela homóloga, se existe, encontra-se posicionada mais à esquerda ainda. Pensando

ao contrário, colocando uma janela de referência na imagem da direita, se a janela homóloga existe, essa encontra-se mais a direita, isto é. sempre desde a posição igual ou maior. Consoante o objecto em questão esteja muito longe ou perto respectivamente. Decidindo, por exemplo, efectuar um varrimento no sentido crescente das coordenadas dos *pixels*, tomando a janela da direita como imagem de referência, se existir janela homóloga, essa encontra-se sempre na mesma posição ou mais à frente. Esse resultado permite eliminar um grande número de correspondências que iriam ser desnecessárias. Com este resultado, define-se que pesquisa se inicia na imagem da esquerda sempre a partir da mesma posição que a própria janela de referência, na imagem direita.

- 2-Gama de correspondência possível

Com base no resultado anterior, reduziu-se o numero de correspondências, no entanto, para janelas no inicio da banda de observação essa redução não é significativa. Sempre com o intuito de tirar o máximo partido das propriedades das imagens, observou-se durante a fase de testes dos barramentos fornecidos, numa perspectiva qualitativa, objectos acerca de um metro das câmaras possuem um nível de disparidade já baixo, isto é, o mesmo objecto não se encontrava em posições muito diferentes entre as duas imagens. Outro factor importante, reside no facto de que objectos muito perto das câmaras surgem desfocados, pois as objectivas estão reguladas para cenários distantes. Assim sendo definiu-se que um objecto nunca aparece na outra imagem com um nível de disparidade com mais de trinta por cento da linha da imagem, o que levou a definir-se que o valor máximo possa caber em 8 bits, isto é, 255 *pixels*.

Usando a possibilidade de se definir imagens de menor tamanho pelos parâmetros W (*width* - dimensão horizontal) e H (*height* - dimensão vertical), o valor absoluto de deslocação máxima pode ser alterado no parâmetro *maxDisp*, permitindo assim que o varrimento máximo seja alterado consoante se espere que o módulo se insira num meio com cenários distantes ou perto, pois num cenário distante a disparidade será baixa. Baixando esse valor é reduzido o número de correspondências efectuadas para uma janela de referência, o que leva a que o sistema devolva os resultados a uma taxa superior.

A tomada destas medidas práticas permite, para cada janela padrão, reduzir o leque de janelas candidatas a janela homóloga sem no entanto sujeitar à perda de eficiência da correspondência do algoritmo.

Convém salientar que o processo de correspondência, requerendo um elevado número de ciclos de relógio, não possibilita que sejam tratadas todas as imagens provenientes das câmaras. As imagens que surgem durante o processamento de um par de imagens, não são guardadas na SDRAM, são direccionadas para a saída VGA de forma a dar ao utilizador uma filmagem contínua da cena mas também possibilitar que sejam utilizadas por outros módulos que possam estar à jusante, em aplicações futuras.

4.2 - Descrição do algoritmo

O estudo prévio dos recursos disponíveis nas secções anteriores, permitiu angariar um conhecimento concreto sobre parâmetros práticos para o desenvolvimento do algoritmo, tais como:

- Aceder aos *pixels* provenientes das câmaras;
- Onde e como armazenar os dados das imagens e os resultados;
- Enviar dados (imagens) para o monitor.

Tendo definidas as condições de que se dispõe para se desenvolver este trabalho, pode-se prosseguir para a fase de implementação prática do algoritmo.

Antes de passar à descrição dos módulos implementados, é importante dar a conhecer a forma como se pretende gerir todo o processo. Este foi subdividido em três fases que na realidade correspondem apenas a dois tempos distintos.

Uma primeira fase consiste na recolha de um par de imagens para a memória SDRAM. De seguida é feito o processamento propriamente dito e armazenamento do resultado na memória. Por fim esse resultado é colocado na saída sincronicamente com o barramento VGA e as coordenadas dos *pixels* actuais. Para um melhor enquadramento com o texto explicativo, aconselha-se a consulta da Figura 4.33, de onde apenas interessa retirar a panorâmica das interligações dos módulos.

A primeira e terceira fase decorrem ao mesmo tempo, daí ser mencionados como possuindo apenas dois tempos. Nesse intervalo de tempo, o módulo responsável pela interface vídeo gere o seu tempo entre colocar os valores dos *pixels* provenientes das câmaras em duas zonas distintas da memória e a ler os resultados das disparidades contidos numa terceira zona da memória. Durante a segunda fase ocorre a leitura das duas zonas de dados onde se encontram as imagens e a escrita dos resultados que se vão sucessivamente obtendo.

Devido a elevada complexidade do problema, esse trabalho foi mitigado em vários módulos, o que permitiu, também, durante a implementação poder-se testar eficientemente passo a passo o bom funcionamento dos diversos módulos. O termo divisão faz referência a gestão dos diferentes passos de processamento. Dividiu-se no módulo de acesso a memória SDRAM, módulo de captação e envio vídeo e módulo de processamento das disparidades.

Estes módulos assim como os seus *testbenches* (bancadas de testes) serão descritos mais a frente.

Observando agora o fluxo do processo numa perspectiva alto nível, a memória é preenchida com um par de imagens e só depois é activado um testemunho indicando a completa recepção do par de imagens. Durante esse mesmo intervalo de tempo é colocado na saída o resultado obtido da correspondência anterior presente na memória. Esse resultado é enviado sincronicamente com as coordenadas dos *pixels* actuais permitindo indicar a que posição corresponde na imagem.

Convém desde já frisar que cada endereço de memória contém 32 bits, o que permite aceder a 4 *pixels* de cada vez. Esta característica traz vantagens ao nível do número de acessos à memória. Mas também aumenta a complexidade do sistema, pois obriga a que se implemente um conjunto de buffers de armazenamento, visto que os resultados chegam a

cadência de 2 *pixels* de cada vez. Sendo esses provenientes de câmaras diferentes e estando definido que a memória é organizada por imagens, é necessário acumular os 8 *pixels*, 4 de cada câmara, antes de os enviar para a memória. Notar desde já que o mesmo acontece para a colocação dos resultados como será lembrado mais adiante. Chegando ao *pixel* de coordenadas $(X, Y) = (639, 479)$, é enviado o último par de dados para a memória. Logo de seguida é “libertado” o testemunho para habilitar o segundo tempo do sistema, processamento do par de imagens e inibir a recolha de dados das câmaras para a memória. Portanto os *pixels* cheguem das câmaras, não são retidos na memória mas continuam visíveis na saída.

O propósito do segundo compasso do sistema consiste, como já foi dito, na determinação das disparidades entre os objectos na imagem. Para isto toma-se como imagem de referência, a imagem direita, Cam1, (a razão da sua escolha foi falada na secção 4.1.7). Sobre ela inicia-se um varrimento completo e seguido de janelas de correspondência padrão, onde estão, as 8 janelas de referência, também chamadas janelas padrão. O processamento do par de imagens pode ser visto como sendo um macro-processo decomposto em vários sub-processos, isto é, a imagem é tratada não no seu todo mas por bandas de observação. Onde uma **banda de observação** corresponde a 8 linhas completas da imagem. O conjunto de 60 bandas de observação forma a imagem completa ($480/8$). Cada janela de correspondência padrão vai originar um varrimento de um conjunto de janelas de correspondência candidatas à janela homóloga, sobre a banda de observação. Para um melhor entendimento da explicação do processo e dos termos usados, segue uma imagem exemplificativa e legendada, a Figura 4.9.

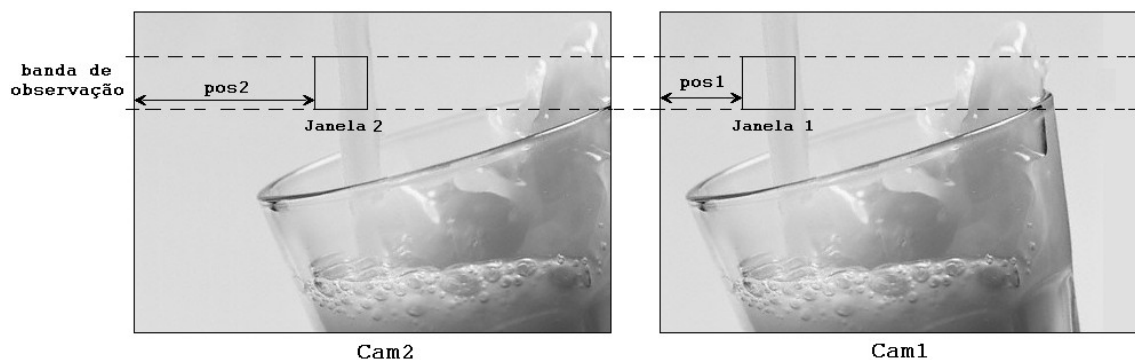


Figura 4.9 - Par de imagens com banda de observação e janelas de observação.

Em cada ciclo é fixado uma janela de observação padrão, janela1, na imagem direita, Cam1. Denomina-se “ciclo” ao processo de varrimento da janela de observação na imagem Cam2, janela2. No fim deste ciclo são obtidos os valores das 8 disparidades resultantes das 8 correspondências efectuadas ao longo do varrimento. Na imagem Cam2 é iniciado um varrimento da janela2 a partir da posição idêntica à da janela 1 até um máximo de *maxDisp* deslocações ou então até chegar ao fim da banda da imagem. O significado de *maxDisp* deslocações foi explicada na secção 4.1.7. O varrimento da linha é feito incrementando sucessivamente a posição da janela de uma unidade. A cada nova janela2 são obtidos 8 novos valores de correspondências, um para cada linha. Cada um destes valores é comparado com o melhor valor obtido até agora, para aquela janela1. Se esse valor actual for melhor, este, assim como o valor de disparidade entre as duas janelas, são actualizados. Caso contrário nada é guardado, pois indica que o par actual não corresponde tanto como um anterior já passado. Notar que para cada novo ciclo, estes registos são carregados com o pior valor

possível. Este valor será estudado na secção 4.3.2. Interessa reter para esta análise a ideia de carregar inicialmente o registo com o pior valor possível para garantir que as correspondências seguintes sejam sempre melhores que a inicial, obtendo-se deste modo, no fim do ciclo a disparidade para a melhor correspondência (a que apresentou menor valor).

No final de cada ciclo são aproveitados os 8 valores de disparidade, onde cada um deles representa a diferença de posição entre o par de janelas nas imagens que se mostrou como sendo o par mais homólogo, mais semelhante. Estes resultados só são realmente guardados na memória ao fim de 4 ciclos, logo 4 resultados, pois cada endereço de memória contém 32bit e sendo cada resultado de 0 à maxDisp (com $\text{maxDisp} \leq 255$), este ocupa 8 bits, logo, é possível armazenar os 4 resultados num só endereço, o que se traduz num só pedido de escrita na memória, por cada 4 *pixels* da imagem. O resultado final das disparidades será colocado na saída, durante a recolha de um novo par de imagens pelo módulo de interface Vídeo, como já foi referido.

Completado o ciclo, inicia-se outro, deslocando a janela1 de uma posição e repetindo o mesmo processo anterior. Os ciclos repetem-se até a janela de referência, janela1, chegar ao fim da banda de observação. Atingido esse ponto, é colocado o endereço no início da próxima banda de observação, a fim de se começar a pedir a memória o conteúdo de uma nova janela1 para recomençar um ciclo de varrimento idêntico ao da banda de observação anterior. Este novo endereço corresponde a um salto de 8 linhas completas na imagem.

Este macro-processo repete-se até varrer a totalidade da imagem. Chegado este ponto é reactivado o módulo de recepção vídeo, para que seja renovado o par de imagens a analisar e ao mesmo tempo sejam colocados os valores das disparidades obtidas, na saída, sincronicamente com os *pixels* provenientes das câmaras.

Os resultados colocados na SDRAM podem ser directamente visualizados num monitor. Os *pixels* com nível disparidade elevada, isto é, *pixels* homólogos distantes serão representados por níveis mais claros e vice-versa. Portanto a imagem resultante oferece pela escala dos tons cinza, do preto ao branco, a informação se aquele ponto está mais ou menos distante das câmaras. Um *pixel* escuro traduz um baixo nível de disparidade portanto encontra-se a uma grande distância da objectiva e vice-versa.

4.3 - Implementação dos diferentes módulos

Esse processo foi pensado numa perspectiva *top-down*, começando por definir os vários módulos e atribuindo-lhes funções. Durante o desenvolvimento dos módulos, foram também criados as bancadas de teste respectivas.

Esta secção começa por expor o funcionamento interno do módulo e utiliza os resultados obtidos na simulação como forma de validação do correcto funcionamento deste último. Considerou-se que a exposição completa da descrição das bancadas de teste não traria uma melhoria para a compreensão geral do módulo, como tal apenas serão focados os resultados obtidos. As bancadas de teste foram estruturadas em tarefas, de forma a ter uma construção mais eficiente e melhorada a sua compreensão. A totalidade da descrição *hardware* dos módulos e das respectivas bancadas de teste encontra-se na página Web do trabalho (14).

Para uma melhor compreensão dos módulos envolvidos e sua integração no sistema é aconselhado a consulta da Figura 4.33 onde se expõe o resultado final da integração e interligação dos diferentes módulos que são expostos nesta secção.

Certos módulos sofreram ligeiras alterações com a realização da parte prática do trabalho, numa perspectiva de melhorar o resultado final. Todas as alterações são referidas e explicadas, não deixando nenhuma alteração pode referir. Considerou-se mais importante dar ao leitor uma descrição dos módulos iniciais e das melhorias trazidas em detrimento da exposição do módulo final, transmitindo as dificuldades encontradas assim como justificar as escolhas tomadas.

Como ponto de partida, implementaram-se as duas janelas de observação, de tamanho 8 por 8 pixels. Estas duas janelas auxiliares são geradas na própria FPGA, permitindo que se tivesse acesso a um par de saídas com o valor dos 64 pixels em cada uma delas, isto de maneira a se poder colocar à jusante um módulo de cálculo, **módulo SAD**, puramente combinacional. Optou-se por desenvolver a métrica do cálculo do nível de disparidade num módulo independente, isto é, numa arquitectura modular, permite que se troque facilmente caso se considere a métrica SAD, desenvolvida neste trabalho, inadequada, podendo assim substituir por outro módulo ou de uma forma alternativa desenvolver um multiplexador que permita fazer coexistir várias métricas e comutar para uma delas no sentido de se obter o melhor resultado consoante o tipo de cena presente.

Passa-se a descrever o processo de implementação dos vários módulos juntamente com uma descrição de certos passos julgados mais relevantes para a compreensão global do módulo respectivo.

4.3.1 - Módulo de geração das janelas de observação

São utilizadas duas memórias auxiliares idênticas como janelas de observação dos *pixels* em análise. Estas janelas, como já foi referenciado, têm tamanho 8 x 8 *pixels*. Entende-se por janela o conjunto de *pixels* que estão directamente ligados ao módulo SAD, descrito mais a frente, portanto o conjunto de *pixels* “visíveis” para o sistema. A razão dessa afirmação será explicada durante esta secção. Interessa para já reter que nem todos os *pixels* existentes na memória auxiliar estão visíveis na saída. Sendo o par de memórias idêntico, passa-se a descrever apenas uma delas. Aliás, na prática, apenas é gerado um módulo. Este é instanciado duas vezes, gerando duas memórias perfeitamente idênticas.

Numa primeira versão, pensou-se em conceber uma memória com o tamanho idêntico ao da janela, isto é, 8 por 8 *pixels*, mas sempre que se deslocasse o conteúdo da janela seria necessário ir à memória para se obter novos valores agora em falta em cada uma das 8 linhas, o que leva a um desperdício de dados, tal como demonstra a Figura 4.10.

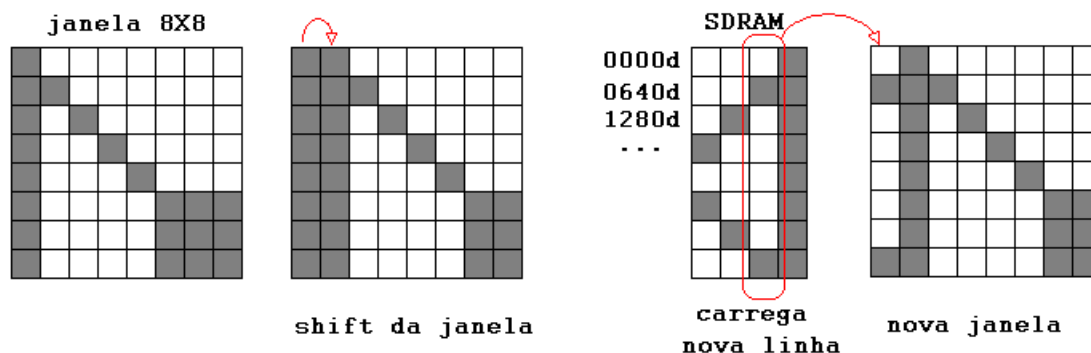


Figura 4.10 - Janela com memória auxiliar 8x8pixels.

A memória devolve, a cada pedido de leitura, um conjunto de 4 *pixels* consecutivos mas apenas se aproveita um deles a cada vez, o que obriga a pedir a memória quatro vezes o mesmo conteúdo. Notar que o processo de leitura da memória requer 5 ciclos de relógios como já foi determinado na secção 4.1.5.1, portanto é conveniente aproveitar todos os dados provenientes de um pedido de leitura. Uma forma de tornar o sistema mais eficiente consiste em aproveitar estes valores para reutilizar nas deslocações seguintes.

Assim sendo, as janelas foram criadas sobre novas memórias, agora de 12 *pixels* por linha. Embora a janela de observação continue a ser de apenas de 8x8, temos agora a possibilidade de armazenar 4 *pixels* na linha da memória, tal como demonstra a figura 4.10. Esta zona da memória tem a função de memória tampão, mais conhecido por *buffer*.

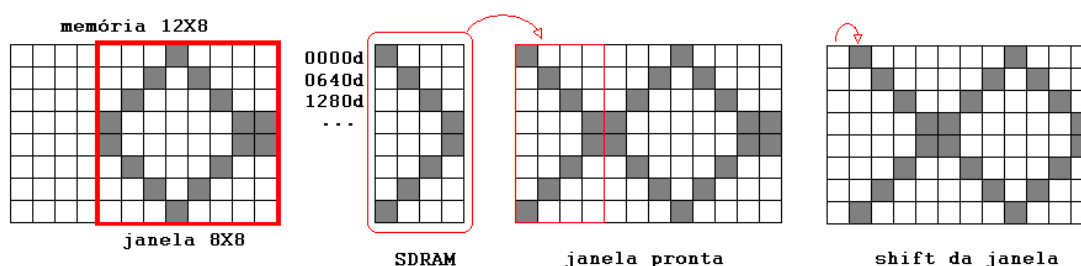


Figura 4.11 - Janela com memória auxiliar 12x8pixels.

Assim sendo, sempre que seja necessário aceder à memória, descarregam-se os 4 *pixels* para o *buffer* da memória auxiliar. O seu aproveitamento permite que durante as próximas 4 deslocações não seja necessário aceder novamente à memória e reduzindo o número de ciclos de relógio gastos neste processo e por consequência pelo processo todo. Durante as próxima 3 novas janelas, a deslocação faz-se apenas num só ciclo de relógio. A janela de observação é representada pelo bloco lógico mostrado na Figura 4.12.

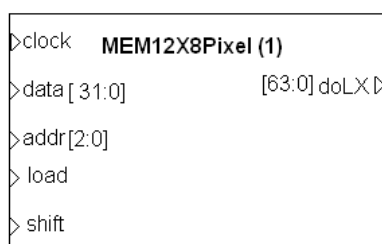


Figura 4.12 - Bloco lógico de uma janela de observação.

Trata-se de uma memória que trabalha sincronicamente com o relógio mestre, *clock*. O barramento *addr* indica em qual das 8 linhas da memória se colocam os valores presentes no barramento *data*. Esse valor é passado para a memória apenas quando o pino *load* está activo, isto é, ao nível lógico alto. O pino *shift* desloca os valores contidos nas 8 linhas, de uma posição, gerando assim uma nova janela. Notar que para uma correcta utilização da memória, essa função é repetida 4 vezes entre cada *load*. Esse controlo é assegurado pelo módulo *Gestor* descrito mais a frente.

A sua saída, *doLX*, contém o valor dos 64 *pixels*, formando a janela em observação.

Na realidade esta memória de 12x8 *pixels*, MEM12X8pixel, é gerada a partir de 8 memórias de 12 por 8 bits, MEM12X8bit, permitindo assim mitigar o grau de complexidade em diferentes etapas.

De forma a certificar do bom funcionamento módulo foi elaborado um *testbench* onde se tenta implementar todas as funcionalidades da memória para uma utilização normal. Mais uma vez foi inicialmente desenvolvido um *testbench* para a memória MEM12X8bit. A Figura 4.13 mostra o comportamento obtido pelo módulo MEM12X8bit perante a simulação.

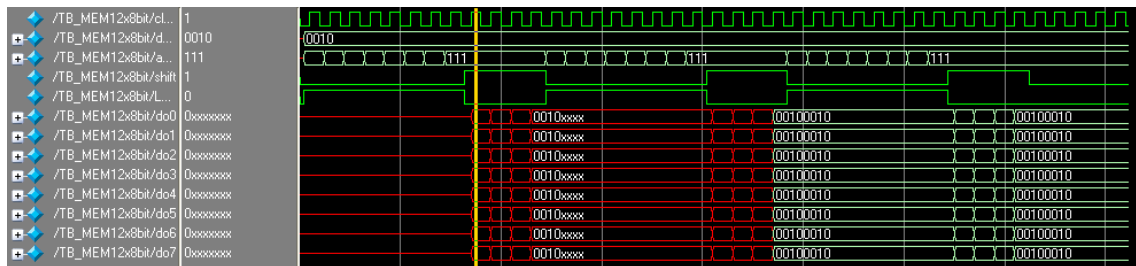


Figura 4.13 - Resultado simulação TB_MEM12X8bit.

Essa simulação consiste em carregar as 12 posições da memória. Como se pode ver o teste decorreu como esperado. A barra amarela realça o primeiro *shift* da memória.

Um segundo teste, muito semelhante, realizou-se com a MEM12X8Pixel, tal com é apresentado na Figura 4.14.

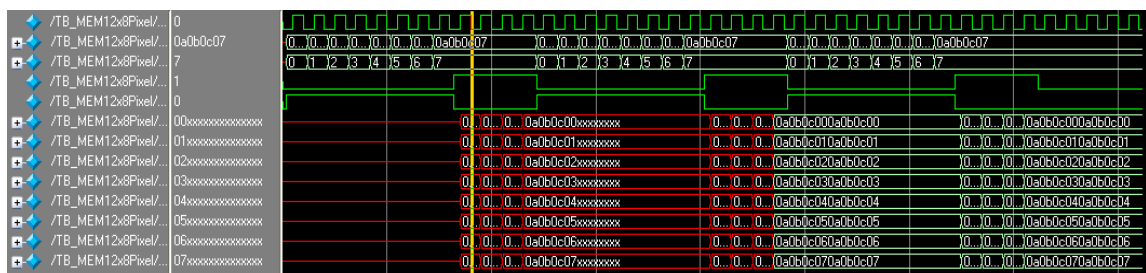


Figura 4.14 - Resultado simulação TB_MEM12X8Pixel.

Os valores de entrada são agora 0a0b0c0Xh, onde X corresponde ao número da linha. Testando assim também o correcto funcionamento da colocação dos valores na respectiva linha. Notar que como era de espera, a aplicação de um comando *shift* é enviado para as 8 “submemórias”, gerando na saída um resultado com 8 dos 64 bits definidos. Após estes testes pode se afirmar que é fiável integrar este módulo no trabalho.

4.3.2 - Módulo de cálculo da métrica SAD

Este módulo é responsável por determinar o nível de correspondência entre cada linha das duas janelas actuais. Esse valor é determinado pela métrica SAD já demonstrada na Tabela 3.1. Versões *pipeline* chegaram a ser pensadas numa tentativa de a baixar a quantidade de recursos utilizados, mas optou-se por um bloco totalmente combinacional numa perspectiva, mais importante, de baixar o número de ciclos necessários durante o processo de correspondência. Factor que se julga ser mais importante, perante um requisito de processamento em tempo real.

O módulo desenvolvido tem como entradas *doLin1* e *doLin2*, onde se ligam os pixels de saída das duas janelas de observação e como saídas um conjunto de 8 barramentos, *result_R0* até *result_R7*, onde cada que devolve o nível de correspondência entre as duas linhas epipolares do par de janelas, de uma forma combinacional. As saídas são de 11 bits cada, para possibilitar a representação do pior caso, isto é, 2040 (8*255), que ocorre para um par de janelas de correspondência extremas, isto é uma janela toda preta perante uma janela toda branca. Este exemplo é apresentado de seguida na simulação do módulo. A Figura 4.15 apresenta o bloco lógico deste módulo.

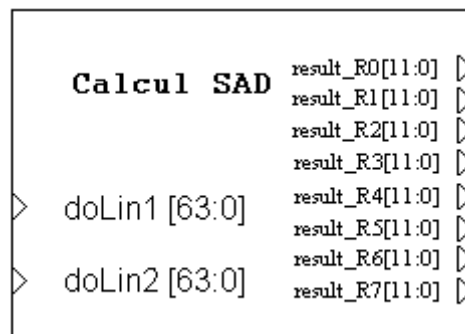


Figura 4.15 - Bloco lógico do módulo de cálculo SAD.

Para estimular o módulo, durante a simulação, foram utilizadas as duas memórias auxiliares como entradas do módulo *Calcul*. O que permitiu testar a correcta interligação dos mesmos com este módulo.

Os estímulos aplicados neste módulo resumiram-se em preencher completamente a parte visível das memórias com os valores extremos das janelas afim de se poder facilmente confirmar o valor da saída apresentada.

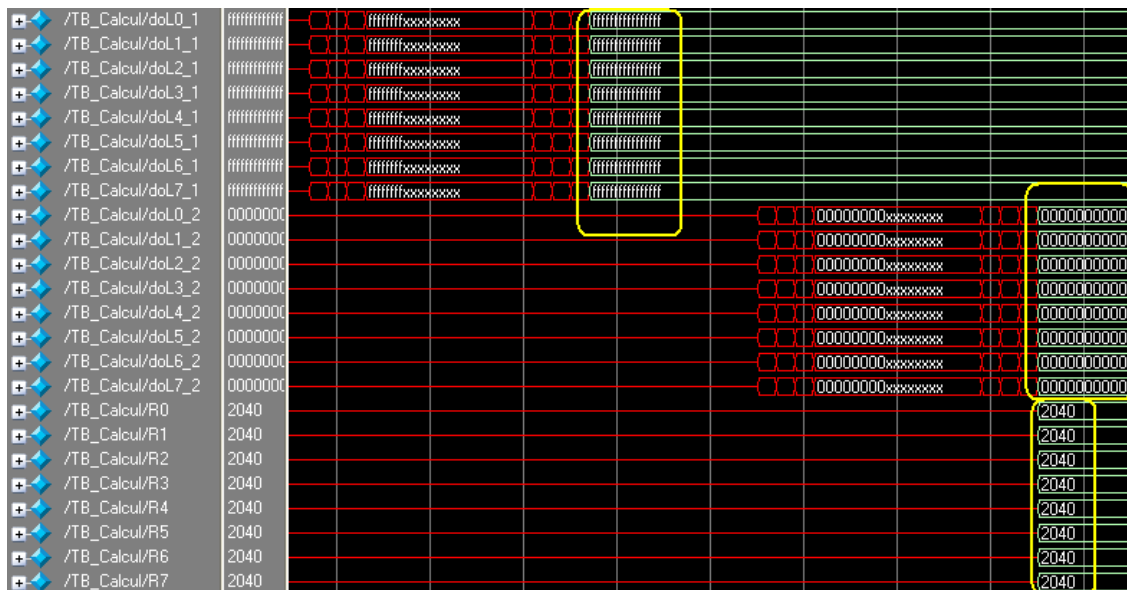


Figura 4.16 - Resultado simulação TB_Calcul.

Como se pode ver pela Figura 4.16 o pior caso possível, isto é, uma janela branca perante uma preta, o módulo devolveu os valores 2040 como previsto.

Notar o caso de o módulo trabalhar em módulo absoluto sobre o par de *pixels*, isto é, ele subtrai sempre o pixel de menor valor ao maior dos dois, obrigando ao valor obtido a ser sempre positivo, pois caso contrário um valor negativo iria fornecer uma contribuição subtractiva ao valor final. O que traria um efeito como sendo ainda melhor que o caso de um par de pixels idênticos, que origina um valor nulo.

4.3.3 - Módulo de atraso vídeo

Este módulo tem como função gerar um atraso de um ciclo de relógio nas seguintes saídas:

- Barramento de controlo da placa VGA,
- Pino de *pclken*,
- Barramento de saída do pixels das câmaras,
- Barramento das coordenadas do pixel actual.

O atraso gerado, permite a manter o sincronismo entre estes barramentos e a saída Pout que contém os resultados das correspondências, tal como foi explicado na secção 4.1.4. A Figura 4.17 apresenta o bloco lógico deste módulo.

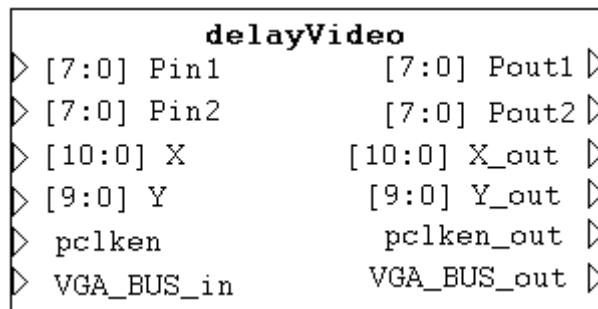


Figura 4.17 - Bloco lógico do módulo delayVideo.

De referir que este módulo funciona de uma forma totalmente paralela e independente dos restantes módulos. Permitindo que se tenha sempre disponível na saída os valores dos *pixels* de ambas as câmaras para aplicações que se possa colocar a jusante.

Devido à simplicidade do algoritmo para a grande quantidade de barramentos envolvidos, optou-se por não gerar nenhuma bancada de teste, visto o módulo resumir-se a um registo de deslocamento de um bit de todas as entradas para a respectiva saída.

Foram pensadas diversas arquitecturas para a concepção dos módulos de recolha das imagens, cálculo das disparidades e colocação dos resultados na saída.

Seguindo o conceito de dividir para conquistar, dividiu-se o sistema em três módulos: interfaceRAM, interfaceVideo e Gestor.

4.3.4 - Módulo de interface com a SDRAM

Este módulo atende os pedidos de acesso provenientes dos módulos interfaceVideo e Gestor, que serão apresentados de seguida. Sendo o único módulo que tem acesso à memória, garante-se que não existe conflitos de acesso pelos diversos módulos que possam pretender aceder a SDRAM ao mesmo tempo.

O pedido de escrita ou de leitura proveniente de um dos dois módulos (interfaceVideo ou Gestor) é feito pelo seu respectivo pino de entrada. Por outro lado, um pino de saída indica quando esse pedido está a ser atendido, voltando ao seu estado inactivo ao fim do processo pedido. Sendo os dois casos possíveis: escrita ou leitura, leva a activação das saídas *writing* ou *loading* respectivamente, durante o desenrolar do pedido. Esse protocolo permite que se tenham vários pedidos pendentes em simultâneo sem gerar conflitos, pois apenas um deles é atendido. O módulo entra numa máquina de estados própria de cada pedido que só após a sua conclusão volta ao estado geral onde avalia a existência ou não de mais pedidos.

Uma outra razão de não ter embutido este módulo nos outros dois, advém do caso de não se ter conseguido aceder à memória. Desenvolvendo um módulo separadamente, facilmente se pode alterar os tempos e modos de acesso, sem ter de alterar os restantes módulos. De igual forma, o número de ciclos necessários para uma escrita ou leitura pode ser diferente do esperado, isto é, pode ser que na prática sejam necessários mais ciclos de espera para um acesso. Como o módulo que fez o pedido funciona com o princípio do *token*, aguarda pelo fim do sinal *loading* ou *writing*, consoante se trate de um pedido de leitura ou escrita respectivamente, indicando a conclusão do processo.

O bloco lógico do módulo é apresentado na Figura 4.18 ao qual se segue a Tabela 4.4 com uma breve descrição dos seus barramentos.

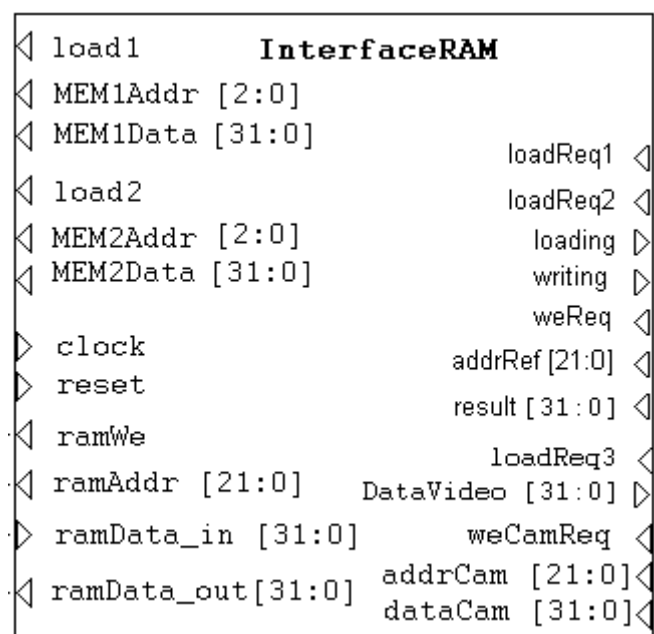


Figura 4.18 - Bloco lógico da interfaceRAM.

Tabela 4.4 - Descrição dos barramentos interfaceRAM.

Pino	Descrição
<i>clock</i>	Relógio mestre do sistema
<i>reset</i>	Reinicializa o sistema
<i>MEM1Addr</i> <i>MEM2Addr</i>	Saída que indica em qual das 8 linhas da memória auxiliar X se deve copiar o que está em MEMXData.
<i>MEM1Data</i> <i>MEM1Data</i>	Saída que contém os valores a colocar na memória auxiliar X.
<i>load1</i> <i>load2</i>	Habilita a escrita do conteúdo de MEMXData na linha MEMXAddr da memória auxiliar X.
<i>loadReq1</i> <i>loadReq2</i>	Pedido de colocação na memória auxiliar X, de 4 novos pixels nas suas 8 linhas, contido na memória no endereço da primeira linha em <i>AddrRef</i> .
<i>loadReq3</i>	Pedido de colocação em <i>DataVideo</i> dos resultados contido na memória no endereço <i>AddrCam</i> .
<i>loading</i>	Pedido de leitura em processamento.
<i>weReq</i>	Pedido de escrita do valor de 4 resultados, no endereço <i>addrRef</i> , contidos em <i>result</i> .
<i>weCamReq</i>	Pedido de escrita de 4 novos pixel, no endereço <i>addrCam</i> , contidos em <i>dataCam</i> .
<i>writing</i>	Pedido de escrita em processamento.
<i>addrCam</i>	Endereço do pedido proveniente do módulo interfaceVideo
<i>dataCam</i>	Valor a colocar na memória, proveniente do módulo interfaceVideo
<i>addrRef</i>	Endereço do pedido proveniente do módulo Gestor
<i>Result</i>	Contém 4 resultados de disparidade a guardar no endereço <i>addrRef</i>
<i>DataVideo</i>	Valor do resultado das disparidade, guardado no endereço <i>addrCam</i> , a enviar para o módulo interfaceVideo.
<i>ramAddr</i>	Endereço do pedido à SDRAM
<i>ramData_in</i>	Dados provenientes de pedidos de leituras
<i>ramData_out</i>	Dados enviados para a SDRAM
<i>ramWe</i>	Pino de controlo entre escrita ou leitura no endereço <i>ramAddr</i> da SDRAM

A Figura 4.19 mostra o fluxograma do módulo de forma a dar um apoio gráfico à descrição do funcionamento interno do módulo exposta a seguir.

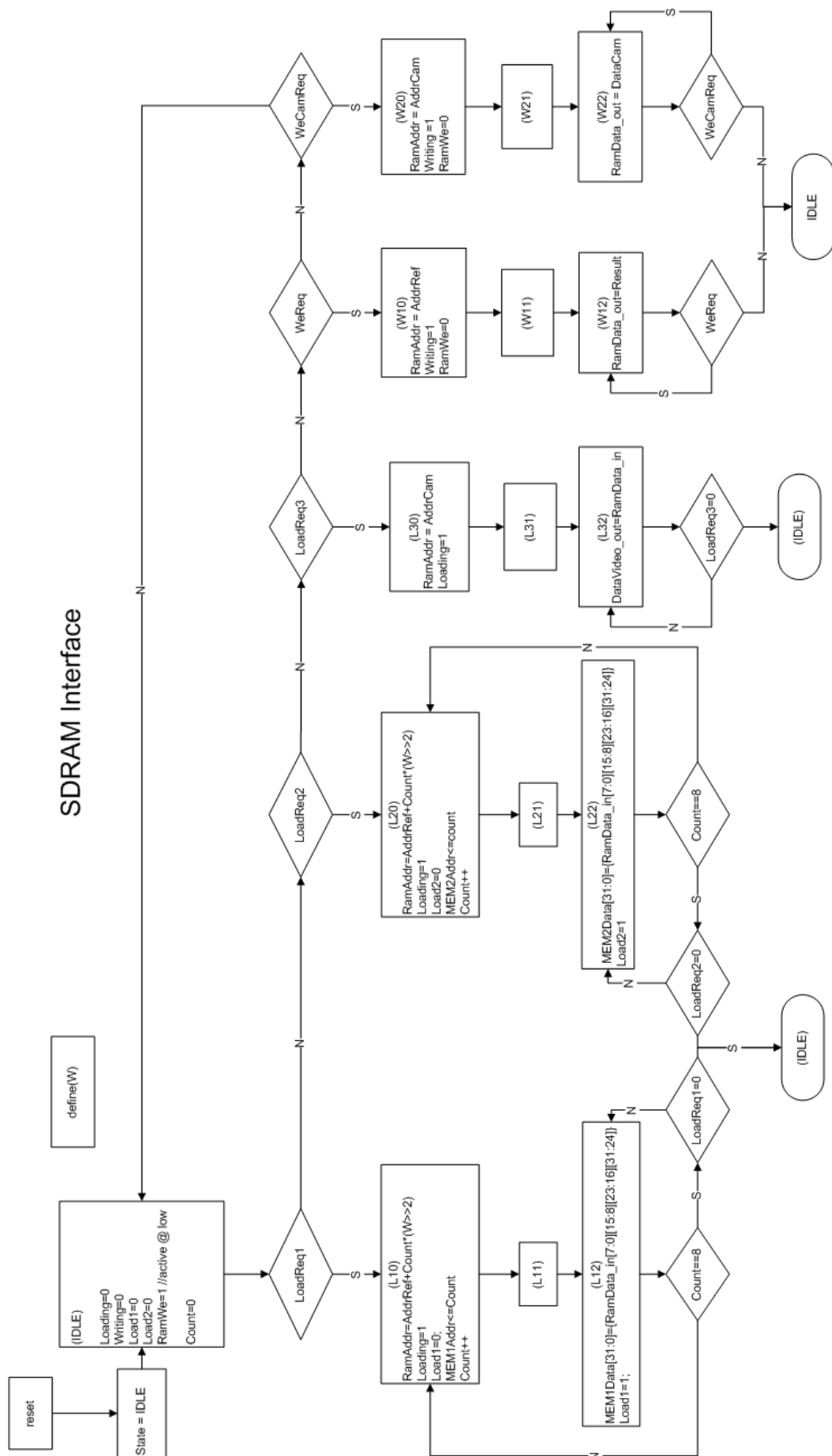


Figura 4.19 - Fluxograma interfaceRAM.

Este módulo consiste numa máquina de estados, onde o estado comum (IDLE) aguarda um pedido por parte de um dos dois módulos ao qual este está ligado. Caso um pedido seja atendido, a máquina de estados entra numa sequência própria do pedido, deixando outros possíveis pedidos em espera até o processo ser finalizado.

O pedido *loadReq1* e *loadReq2* são equivalentes, apenas diferem quanto a memória auxiliar para a qual são enviados os dados, sendo eles enviados para a memória 1 e 2 respectivamente. Por esse motivo apenas se passa a explicar um deles.

Quando é recebido um pedido *loadReq1*, por exemplo, o registo *loading* é activo, colocado ao nível lógico alto, indicando assim ao módulo Gestor que o pedido está a ser atendido. Ao mesmo tempo é iniciado o processo de carregamento das 8 linhas da memória auxiliar com 4 novos pixels em cada linha. Para isto, é pedido à memória SDRAM o conteúdo existente no endereço *RamAddr*.

Esse registo contém o endereço onde se encontra a primeira linha a ir buscar, enviado pelo módulo Gestor, via o registo *AddrRef*, ao qual se soma o registo *Count* multiplicado por a menor inteiro da divisão da largura *W* da imagem por 4 ($W \gg 2$). Este valor é o número de posições na SDRAM entre cada linha na imagem. O valor contido em *Count* indica a linha da memória para a qual se vão enviar estes dados. Este registo é incrementado a cada vez que se colocam 4 novos pixel na memória auxiliar.

De notar o estado L11 que representa um ciclo de relógio de espera necessário para a memória encaminhar até ao registo *RamData* correctamente os valores contidos na memória.

No ciclo seguinte os valores são enviado para a memória auxiliar e activo o contador *load1* para habilitar a escrita na memória. Sendo agora incrementado o valor de *Count*, esse indica qual a próxima linha para onde se vai escrever na memória auxiliar. Quando este registo atingir o valor 8, indicando que se preencheram as 8 linhas da memória. O registo *loading* volta ao estado desactivo. A máquina de estados regressa ao estado IDLE e é reposto o valor do registo *Count* a zero.

Um pedido *loadReq3* indica que o módulo *interfaceVideo* pretende ler da memória o valor das disparidades contidas no endereço *AddrCam*. Esse processo é análogo ao anterior, diferindo apenas no sentido em que este pedido gera somente um único pedido de leitura à memória e envia esse resultado para o registo de saída *DataVideo*. O uso do registo *loading* activo, nas mesmas condições que o caso anterior.

Em ambos os casos a máquina de estado apenas volta para o estado IDLE se o registo do pedido for desactivado, de forma a se poder distinguir dois pedidos diferentes, pois neste caso o sistema é obrigado a desactivar o pedido para a saída *loading*.

Com base no mesmo princípio que o uso do registo *loading*, o registo *writing* é activo no acto de atendimento dos dois pedidos seguintes, *weReq* e *weCamReq*. Da mesma forma, a máquina de estados apenas regressa para o estado IDLE se o pino que originou o pedido foi desactivo.

Um pedido *weReq* indica que o módulo Gestor tem 4 resultados de disparidade para colocar no endereço indicado pela entrada *addrRef*. O registo *ramWe* está directamente ligado ao pino seleccionador de escrita ou leitura da memória SDRAM. Quando posto ao nível lógico baixo, este prepara a escrita do valor contido em *RamData_out*, barramento de saída de dados, no endereço *RamAddr*, barramento de endereço da memória.

Os dados contidos na entrada *Result* são enviados para *RamData_out*, possibilitando o envio destes valores para a SDRAM. Frisar que estes valores estão seguros na saída pelo menos

durante dois ciclos de relógio. Se no entanto numa implementação prática se verificar que estes dois ciclos não são suficientes, a solução a aplicar é análoga a já referida sobre propagação de dados, isto é, intercalação de ciclos de espera no final da cadeia antes de entrar no estado IDLE.

Por fim o pedido *weCamReq* é em tudo semelhantes ao pedido *weReq*, diferindo nas entradas das quais ele adquire os dados e endereço. Este pedido indica que o valor a escrever na memória SDRAM, no endereço contido em *AddrCam*, está na entrada *DataCam*, entrada proveniente do módulo *interfaceVideo*, contendo o valor de 4 pixels provenientes de umas das câmaras.

Todos os registos de controlo, *load1*, *load2*, *loading*, *writing* e *ramWe* são desactivados no estado IDLE, cada um ao seu nível correspondente.

Para uma melhor compreensão deste módulo aconselha-se a consulta o fluxograma supra fornecido.

De forma a certificar do bom funcionamento módulo foi elaborado de seguida uma bancada de teste onde se tenta implementar todas as funcionalidades do módulo *interfaceRAM*. Visto a grande quantidade de processos que envolve a simulação, optou-se por decompor a sua apresentação em várias sequências:

- Parte1 - Simulação dos pedidos provenientes do módulo Gestor, onde é apresentado o atendimento dos pedidos de leitura *loadReq1* e *loadReq2* e de escrita *WerReq*.

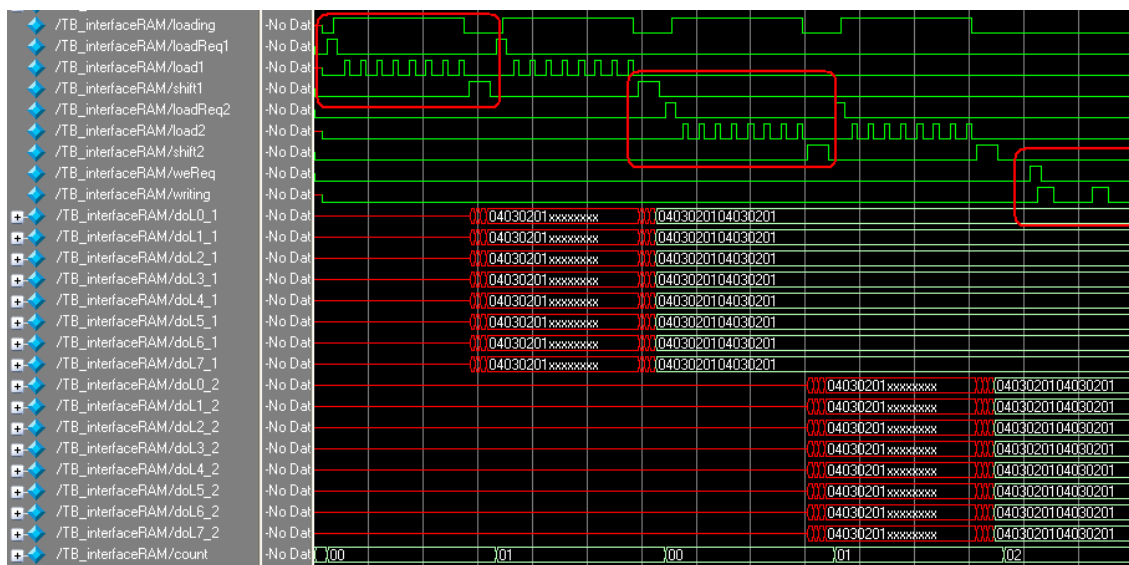


Figura 4.20 - Resultado simulação TB_interfaceRAM - parte1.

- Parte 2 - Simulação dos pedidos provenientes do módulo *interfaceVideo*, onde é apresentado o atendimento dos pedidos de leitura *loadReq3* e *weCamReq*.

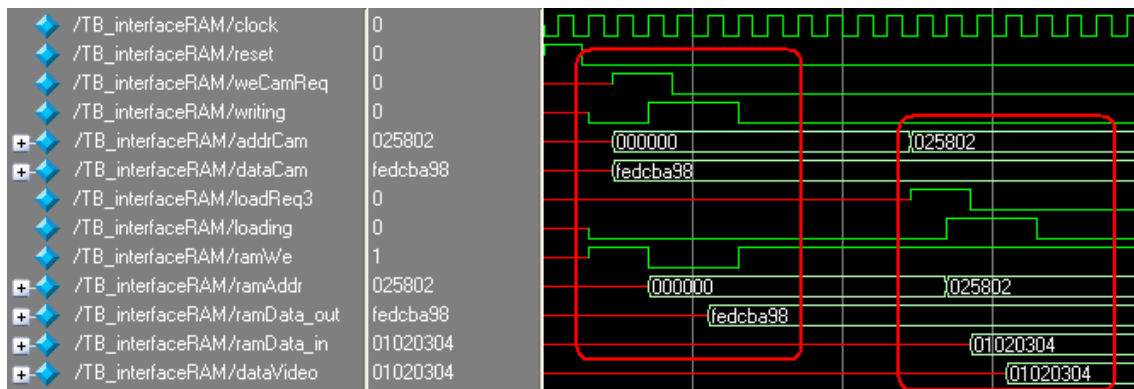


Figura 4.21 - Resultado simulação TB_interfaceRAM - parte2.

Como se pode observar através da Figura 4.20 e Figura 4.21, a simulação apresenta os resultados nas respectivas saídas. Os valores 32'h012c01, em *addrRef*, para o pedido de leitura *loadReq2*, e 32'h025802, em *ramAddr*, para o pedido de leitura *loadReq3* são endereços iniciais de zonas de memória, da segunda imagem e dos resultados respectivamente. Estes valores podem ser consultados na secção armazenamento das imagens e resultados.

O módulo *interfaceRAM* gere os pedidos entre os módulos requerentes, *interfaceVideo* e Gestor, e a memória SDRAM tal como pretendido. A simulação demonstra o sucesso da implementação do módulo.

Os dois módulos que se seguem são complementares quanto aos pedidos de acesso à memória. Tal como já foi indicado, o módulo *interfaceVideo*, pede para escrever na memória os valores provenientes das câmaras, assim como para ler para a saída, os resultados fornecidos pela memória. Já o módulo responsável pelo processamento, chamado Gestor, usa os dados dos *pixels* lá colocados e escreve nela os resultados obtidos pelo processo de correspondência. Como não faz sentido estar-se a processar a imagem e ao mesmo tempo estar a alterar o valor dos *pixels*, temos de garantir que os módulos não estão activos simultaneamente. Para isto usou-se o princípio da passagem do testemunho (*token*) entre os dois módulos. Garante-se assim também, que não existem ciclos de relógio sem processamento útil, pois o testemunho é logo libertado no fim do processo, o que activa de seguida o outro módulo e assim sucessivamente.

4.3.5 - Módulo de captação vídeo

Este módulo é responsável por recolher a informação proveniente de ambas as câmaras e estabelecer os pedidos de colocação destes valores na memória via o módulo *interfaceRAM*, mas também, como já foi referido, pela mesma via, pedir os resultados obtidos na correspondência do par de imagens anterior e coloca-los sincronamente com as saídas do módulo *delayVideo*.

O módulo coloca em *Pout* (ver Figura 4.33) o *pixel* com a mesma coordenada do que a do par de *pixels* actuais proveniente das câmaras, garantido assim o sincronismo entre estes valores e as imagens provenientes das câmaras, o que é vantajoso caso se queira usar aplicações onde essa característica seja necessária.

Recordar que, este módulo só entra em funcionamento se lhe for dado a permissão pela presença do testemunho. Para o caso deste surgir durante o varrimento de uma imagem, o módulo aguarda pelo início o próximo varrimento para salvar a totalidade de duas imagens, Cam1 e Cam2, na memória.

No fim desse processo é libertado o sinal testemunho para o módulo de correspondência, inibindo qualquer actividade neste módulo.

O seu bloco lógico é apresentado na Figura 4.22, ao qual se segue a Tabela 4.5 com uma breve descrição dos seus barramentos.

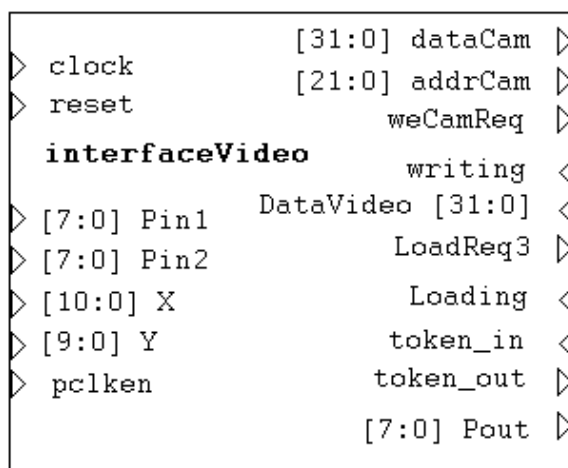


Figura 4.22 - Bloco lógico da interfaceVideo.

Tabela 4.5 - Descrição dos barramentos interfaceVideo.

Pino	Descrição
<i>clock</i>	Relógio mestre do sistema
<i>reset</i>	Reinicializa o sistema
<i>Pin1</i> <i>Pin2</i>	Barramento com o valor dos pixels actuais provenientes das câmaras 1 e 2 respectivamente.
<i>X</i> <i>Y</i>	Coordenadas dos pixels actuais
<i>pclken</i>	Indica a validade dos dados <i>Pin1</i> , <i>Pin2</i> , <i>X</i> e <i>Y</i> durante o ciclo de relógio a que ocorre.
<i>weCamReq</i>	Pedido de escrita na memória dos valores presentes na saída <i>dataCam</i> .
<i>writing</i>	Pedido de escrita em processamento.
<i>dataCam</i>	Saída contendo 4 pixels para serem guardados.
<i>addrCam</i>	Endereço com a posição onde deve ser guardado o conteúdo de <i>dataCam</i> .
<i>loadReq3</i>	Pedido de leitura dos resultados contidos na memória com o endereço <i>addrCam</i> .
<i>loading</i>	Pedido de leitura em processamento.
<i>DataVideo</i>	Registo com os resultados pedidos pelo comando <i>loadReq3</i> .
<i>Pout</i>	Saída com o valor da disparidade obtida durante o processo de correspondência do par de imagens anteriores para o pixel com as coordenadas (<i>X</i> , <i>Y</i>).
<i>token_in</i>	Entrada do token proveniente do módulo Gestor.
<i>Token_out</i>	Saída ao nível activo, quando o módulo terminou a sua função. Habilita o módulo Gestor.

É apresentado na Figura 4.23 o fluxograma do módulo de forma a dar um apoio gráfico a descrição do funcionamento interno do módulo exposta a seguir.

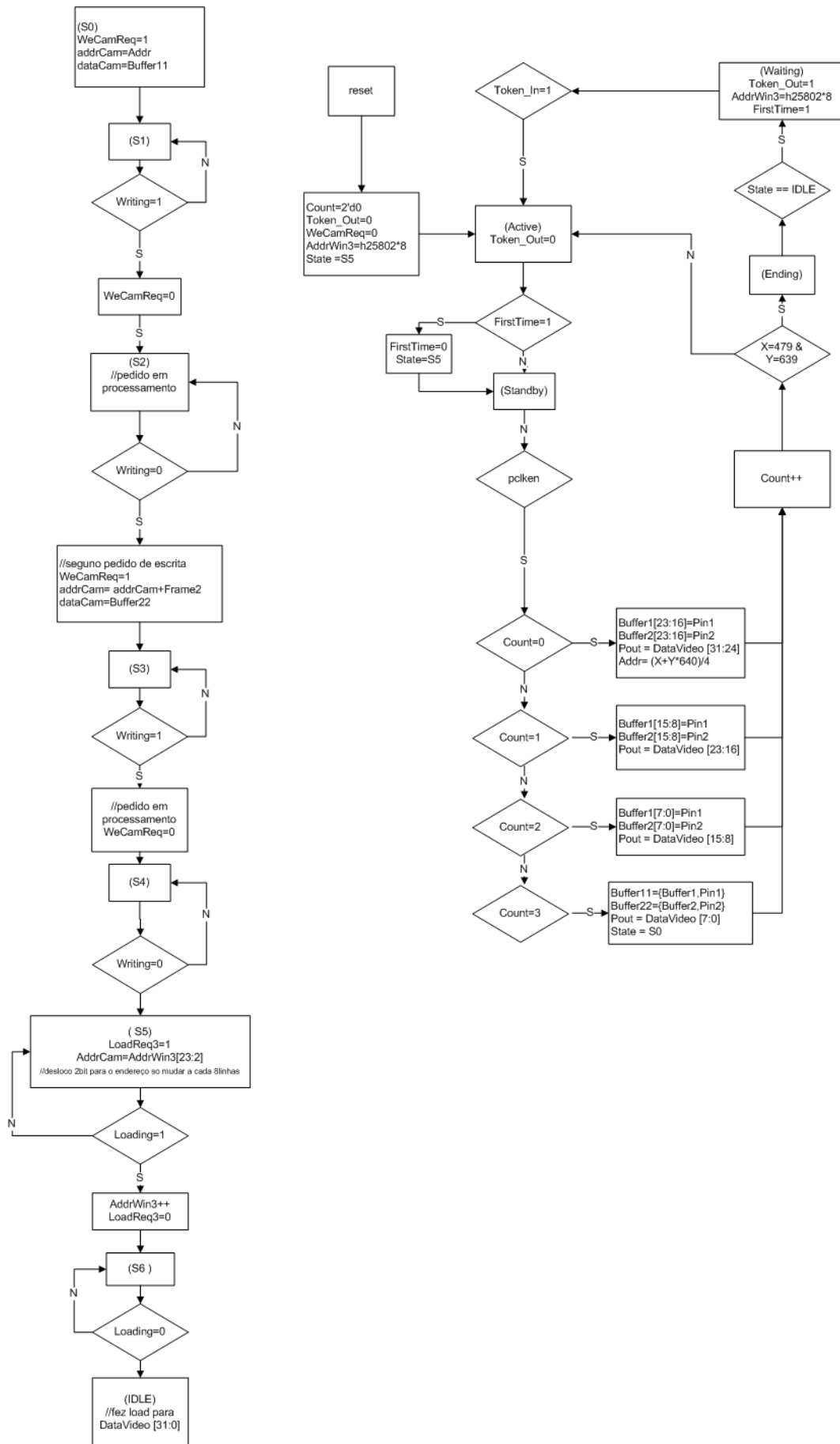


Figura 4.23 - Fluxograma do módulo de interfaceVideo.

Para uma melhor compreensão é atribuído o termo controlador ao fluxo que se encontra à direita e o termo sequência ao fluxo da esquerda. O controlador pode reiniciar a sequência para um estado anterior, enquanto a sequência apenas pode passar para o estado seguinte. A sequência gere o acesso a SDRAM, enquanto que o controlador encarrega-se de colocar sincronicamente os valores na entrada da placa VGA, recolher os valores para serem enviados para a SDRAM e habilitar o estado inicial da sequência para esta proceder ao envio dos dados e recolha dos resultados para colocar em *Pout*.

Apresenta-se de seguida uma análise sobre o desenrolar dos sucessivos processos deste módulo. Primeiramente, concentrando-se nos estados da sequência, desta forma podemos definir as suas diferentes macro-etapas, o que permite apresentar uma panorâmica de mais alto nível acerca dos pedidos efectuados pela máquina de estados à cadeia.

Chegando ao último estado, a sequência permanece sem condição possível para transitar para outro estado. Apenas o controlador pode alterar o estado da sequência para um estado a montante. Observe-se o decorrer destes estados.

Numa primeira fase é desencadeado o pedido de escrita dos 4 *pixels* obtidos pela câmara1, o mesmo se passa para a câmara2. Sendo o processo de envio idêntico para ambas as escritas, apenas se vai descrever o processo para uma delas. Para tal são colocados na saída os registos *addrCam*, que contém o valor do endereço de memória onde se pretende guardar os 4 pixels, *dataCam*, que contém os 4 pixels a guardar e é activa a saída *camWeReq*, que indica o pedido de escrita. É aguardado que a entrada ligada ao registo *writing* seja detectada indicando que o pedido está a ser atendido. Chegado esse momento pode-se desactivar a *flag* do pedido de escrita, *weCamReq*. De seguida aguarda-se o fim do processo, anunciado com o regresso de *writing* ao nível lógico nulo. Findo o processo, o sistema está disponível para atender outro pedido. Tal como foi dito executam-se dois pedidos de escrita, um para cada imagem.

Numa terceira fase, é feito um pedido de leitura dos valores de disparidades obtidos pela correspondência do par de imagens anteriores. Para isso é actualizado na saída o registo *AddrCam*, com endereço onde se encontram estes 4 valores, e activo o pedido de leitura, *loadReq3*. Analogamente para o processo de escrita, aguarda-se o início do processamento do pedido, pela transição da entrada *loading*. Chegado esse caso, desactiva-se o registo *loadReq3* e aguarda-se a conclusão do processo de leitura, marcado pelo regresso da entrada *loading* ao estado nulo. A partir desse momento pode-se trabalhar com o valor na entrada *DataVideo*. Não há necessidade de o copiar para um registo interno, pois é garantido que esta entrada só é alterada com um outro pedido idêntico que só ocorre depois de se ter recebido 4 novos pixels das câmaras, portanto também já foram enviados estes resultado para a saída.

Tendo findo a análise sobre a sequência de acesso à SDAM, é feita uma análise sobre a máquina de estados.

Como se pode ver, durante o reset, o *token_out* é logo definido como desactivo, inibindo assim o módulo Gestor. Não faz sentido iniciar pelo processamento de um par de imagens, se ainda não foi colocado na memória nenhum dado. Esta tarefa corresponde a função do módulo *interfaceVideo*. Assim sendo, numa situação inicial, este módulo é o primeiro dos dois a ser activado.

Pode se observar que o controlador é colocado no estado *Active*, indicando que está habilitado a receber os dados provenientes das câmaras e colocar os valores das disparidades em *Pout*. A cadeia de acesso à SDRAM é colocada no estado de recolha de resultados existentes na memória, isto mesmo não existindo lá qualquer valor, pois ainda não foi sequer

recolhido um par de imagens. Mas este aparente erro apenas se vai sentir durante a determinação da primeira imagem de disparidades. Esse processo poderia ser evitado, por exemplo, colocando um registo como uma *flag* indicando que ainda se trata do primeiro par de imagens, mas, como foi dito, essa resposta transitória não traz qualquer influência visível para o utilizador, pois apenas ocorre quando se inicia o sistema.

Feito este pedido, o controlador entra no estado *standby*, que consiste na espera de um par de *pixels* provenientes das duas câmaras. Esse acontecimento é indicado pela transição ascendente da entrada *pclken* durante um ciclo de relógio. Os valores dos pixels são armazenados num registo auxiliar chamado *Buffer1* e *Buffer2* respectivamente. Esse registo apenas tem espaço para os 3 primeiros *pixels*, quando surge um quarto, o pixel assim como o conteúdo de *Buffer1* / *Buffer2* são copiados para o registo *Buffer11* / *Buffer22* respectivamente. Possibilitando assim que se receba mais pixels enquanto se processa o envio destes dois conjuntos de *pixels* para a memória. Ao mesmo tempo a sequência foi colocada no seu primeiro estado, preparando-a assim para enviar, para a memória, os dados contidos em nestes “*buffers*”, via o módulo interfaceRAM, como já foi referido.

Falta referir que sempre que se recebe um novo par de *pixels*, é colocado na saída *Pout* o valor da disparidade do pixel com a mesma posição que o par de pixels actuais, contido em *DataVideo*. Este processo repete-se até ao fim da imagem, até (X, Y) = (479, 639), isto é, até se escrever na memória a totalidade do par de imagens e ler a totalidade dos resultados de disparidade para a saída *Pout*. Neste caso o *token_out* é passado para o nível lógico alto, activando agora o módulo Gestor e deixando este módulo a espera da activação do *token_in* para poder sair do estado de espera.

Seguindo o mesmo processo de teste que foi aplicado nos módulos anteriores, foi elaborada de uma bancada de teste onde se implementou todas as funcionalidades do módulo para uma utilização normal, isto é, duas escritas, seguidas de uma leitura e a colocação em *Pout* dos respectivos valores. Uma forma visual do resultado está na simulação seguinte.

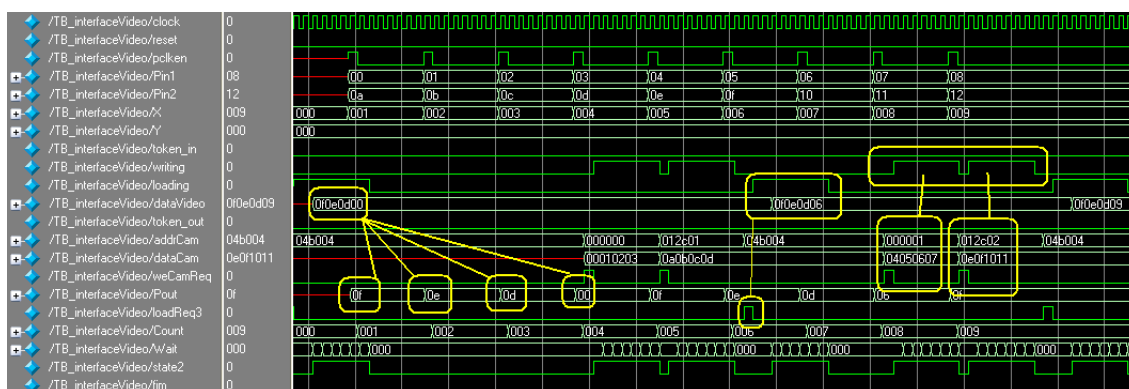


Figura 4.24 - Resultado simulação TB_interfaceVideo.

O comportamento do módulo é o esperado. Olhando para este módulo, apercebemo-nos que este módulo é o que requer o maior fluxo de dados, duas escritas e uma leitura, durante o tempo que chegada 4 novos pixels de cada câmara. A taxa de chegada de um *pclken*, novo par de *pixels* proveniente das câmaras, é de 7 ciclos de relógio. Permite um total de 28 ciclos de relógio (4*7). Contando com 3 ciclos por cada leitura ou escrita, mais 2 ciclos para a passagem dos valores de controlo entre cada novo acesso, o que totaliza 15 ciclos de relógio (1+3+2+3+2+3+1), valor inferior aos 28 ciclos disponíveis.

4.3.6 - Módulo de correspondência

Tal como foi explicado durante o módulo anterior, depois de guardadas na memória as duas imagens e os resultados serem passados para a saída Pout, o testemunho é transmitido para o módulo de correspondência.

Esse módulo é responsável pelo decorrer do algoritmo de correspondência propriamente dito, pois é ele que acede aos valores dos *pixels* guardados na memória e com estes abastecer as memórias auxiliares de forma a possibilitar as correspondências. Também tem por tarefa enviar os pedidos de escrita dos resultados da disparidade na memória ao módulo *interfaceRAM*. Tal será explicado em maior detalhe nesta secção.

O seu bloco lógico é apresentado na Figura 4.25 a qual se segue a Tabela 4.6 com uma breve descrição dos seus barramentos.

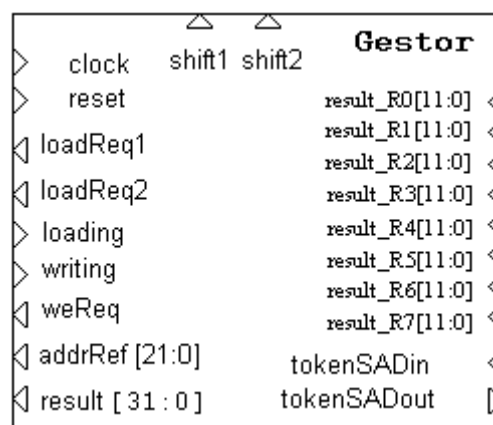


Figura 4.25 - Bloco lógico do módulo gestor.

Tabela 4.6 - Descrição dos barramentos do módulo Gestor.

Pino	Descrição
<i>clock</i>	Relógio mestre do sistema
<i>reset</i>	Reinicializa o sistema
<i>loadReq1</i> <i>loadReq2</i>	Pedido de colocação na memória auxiliar X, de 4 novos pixels nas suas 8 linhas, contido na memória no endereço da primeira linha em <i>AddrRef</i> .
<i>loading</i>	Pedido de leitura em processamento.
<i>weReq</i>	Pedido de escrita do valor de 4 resultados, no endereço <i>addrRef</i> , contidos em <i>result</i> .
<i>writing</i>	Pedido de escrita em processamento.
<i>addrRef</i>	Endereço da posição de memória dos pedidos.
<i>result</i>	Contém 4 resultados de disparidade a guardar no endereço <i>addrRef</i> .
<i>Result_RX</i>	Valor devolvido pela saída do módulo de cálculo referente a linha X.
<i>Shift1</i> <i>Shift2</i>	Activa a deslocação, de uma coluna por ciclo de relógio, do conteúdo da janela de observação 1 e 2 respectivamente.
<i>tokenSADin</i>	Entrada do token proveniente do módulo interfaceVideo
<i>tokenSADout</i>	Saída do token para o módulo interfaceVideo.

O fluxograma do módulo Gestor está representado nas Figura 4.26 e Figura 4.27, de forma a dar um apoio gráfico a descrição do funcionamento interno do módulo exposta logo de seguida. Notar que este fluxograma embora esteja dividido por duas folhas, a segunda parte está simbolicamente representada, dentro da primeira, por um único bloco “ciclo varrimento linha2”. Esta divisão teve lugar unicamente por motivos de espaço, isto é, para se representar mais claramente todo o fluxo do módulo.

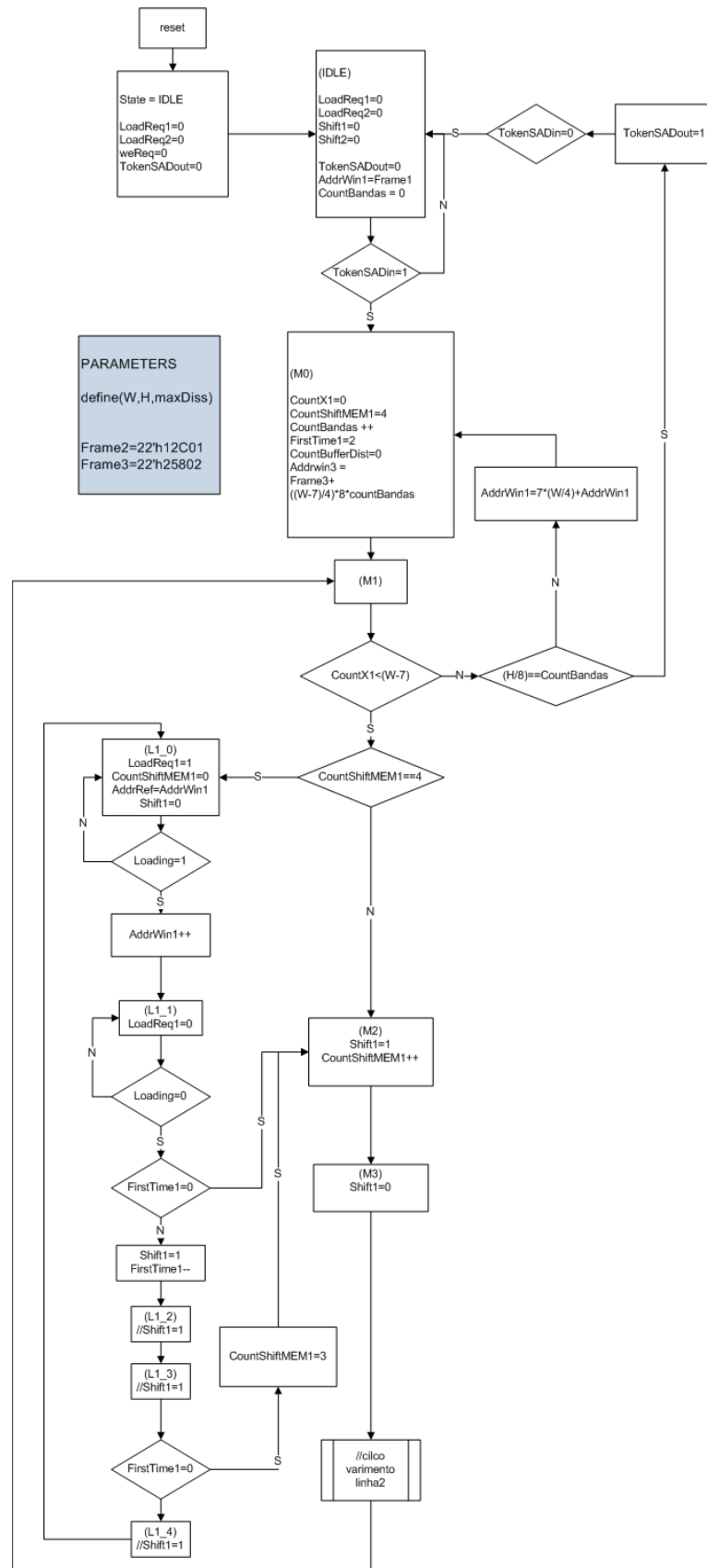


Figura 4.26 - Fluxograma módulo gestor - parte 1.

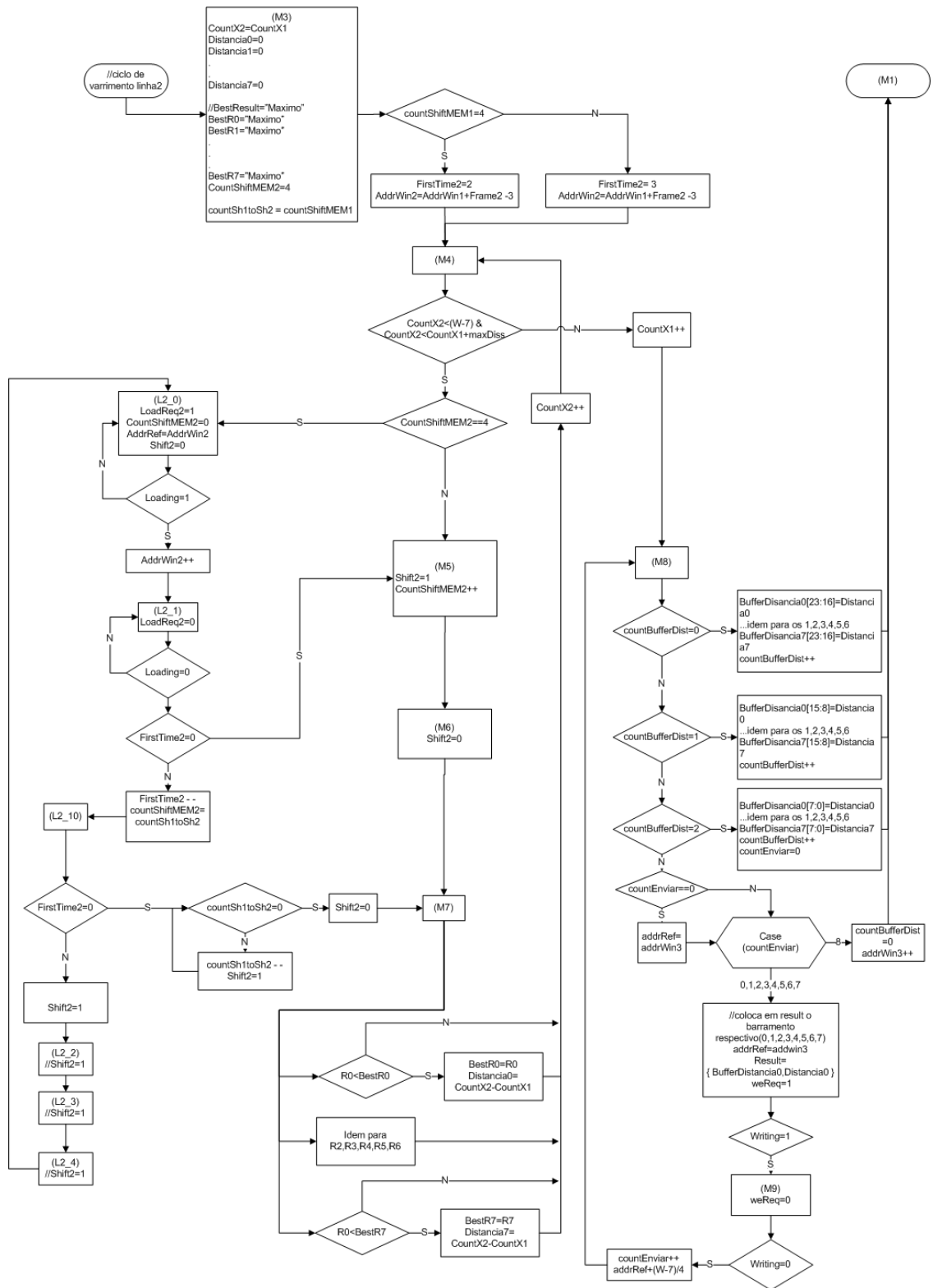


Figura 4.27 - Fluxograma módulo gestor - parte 2.

Antes de mais convém referir que o fluxograma é estruturado por sucessivos varrimentos de linhas da imagem1 para se percorrer a imagem toda. Dentro deste varrimento, para uma janela de referência, se processa o varrimento na banda de observação homóloga da imagem2, isto é é iniciada uma janela de correspondência a partir da mesma posição da janela1 até um máximo de *MaxDisp* posições (com *MaxDisp* <256). Para cada janela obtém-se 8 valores de correspondência nas saídas respectivas de *result_RX*, onde X indica o índice da linha em questão. Ao fim do varrimento da linha2, são guardados os melhores resultados obtidos para cada uma das 8 linhas e passado para a próxima janela de referência. No fim do processamento da quarta janela padrão, os 32 (4*8) resultados acumulados são enviados, pelo auxílio do módulo interfaceRAM, para a memória.

Tendo agora uma panorâmica do desenrolar deste módulo, pode-se agora passar para uma descrição mais pormenorizada. A ocorrência de um *reset* leva a máquina de estados deste módulo para o estado de espera do testemunho proveniente do módulo interfaceVideo. Durante este estado de espera são reinicializados registos internos e definidos como desactivas as saídas de controlo. É importante referir que o registo *CountX1* contém a posição da janela de referência na linha da imagem1, permitindo assim saber-se quando chegou ao fim da imagem. O registo *CountShiftMEM1* indica quantas deslocações foram feitas na memória auxiliar 1 desde o último *load*, permitindo assim saber se ainda existem pixel em espera na memória ou se é necessário pedir mais quatro à memória.

Sempre que pretenda gerar uma nova janela, que não seja uma simples deslocação de uma posição, a máquina de estados tem de preencher a janela com 8 novas colunas de pixels. Para isso não basta aplicar um único pedido de *loadReq1* ao módulo interfaceRAM, mas sim duas vezes. No fim de cada um deles é necessário proceder-se a 4 deslocações, para retirar estes valores da zona de descarga. Esse procedimento inicial é possível graças ao registo *FirstTime1*, que não é mais do que uma *flag* para indicar que no início de uma nova linha é necessário carregar a memória com os 8 novos *pixels*. Este registo é reinicializado no fim do varrimento de cada imagem. O mesmo ocorre com a janela 2. Como se pode ver na parte 2 do fluxograma, os registos envolvidos são em tudo semelhantes, apenas se altera para a janela2. De referir a inicialização de *CountX2* por cópia de *CountX1*, garantindo assim a correcta indicação da posição da janela2. Tendo a janela1 pronta, o mesmo processo de carga inicial dos 8 pixels ocorre para a janela2. Tendo os valores prontos nas janelas, os 8 resultados da métrica SAD são propagados para as entradas *Result_RX*. Assim sendo, para cada uma das 8 linhas, é determinado o valor da correspondência comparando-o com o melhor valor obtido para aquela janela1, contido no registo *BestRX*. Notar que este registo é reinicializado a cada nova janela de referência como valor máximo, 12'hFFF, garantido que qualquer correspondência que surja a seguir seja menor que o valor inicial. Quando o valor de *Result_RX* é inferior significa que obteve-se uma melhor correspondência, este é copiado para *BestRX*. Ao mesmo tempo é guardado no registo *Distancia*, o valor da diferença de posição entre a janela candidata actual e a janela de referência, indicado pelos registos *CountX1* e *CountX2*, respectivamente, como já foi referido. Caso contrário nada é actualizado, passando-se logo para a próxima etapa.

É necessário agora pedir uma nova janela2, para isso é pedido um novo *loadReq2*, pois não existe nenhum valor no *buffer* da memória auxiliar. Sem esquecer, depois disso, o *shift* necessário para deslocar o conteúdo da janela de uma posição para fora do *buffer*. Durante as 3 próximas janelas, o processo de geração de uma nova janela será feito com um simples

pedido de um *shift* de duração de um ciclo de relógio, pois o *buffer* da memória contém ainda mais 3 *pixels*. De notar, no estado M3, a linha seguinte:

```
countSh1toSh2 <= countShiftMEM1
```

Essa linha permite que para pedidos de uma primeira janela2, se possa proceder a correcta geração da janela. É útil para resolver o caso de janelas de referência que sejam resultado de um, dois ou três *shifts*, isto é, que não sejam coincidente com o endereço *CountX1*, mas sim intermédias. Este registo transmite para a geração inicial da janela2, o número de *shift* a executar depois de carregar o *buffer* com o último *load*. Se esta informação não fosse indicada, como *CountX1* só é incrementado de 4 em 4 janelas, a janela2 inicial seria a mesma para a 4 janelas1 possíveis. Para remediar isto, indica-se também de quanto já foi deslocada a janela1. Garantindo assim que a janela2 inicial começa na mesma posição que a janela1.

O processo de deslocação das janelas2 repete-se até se percorrer na linha *maxDisp* posições ou então obedecendo a condição de paragem de fim de linha. Neste ponto são copiados os valores dos registos *distanciaX* para o registo *BufferDistanciaX*. Cada um destes registos auxiliares armazenam os valores de 4 disparidades e só depois os envia para a memória, via o módulo *interfaceRAM*, activando para tal o pedido *weReq*. De notar que o módulo envia, para o módulo *interfaceRAM*, o endereço, *addrRef*, carregando com valor contido em *addrWin3*. Este último é incrementado de $(W-7)/4$ a cada pedido de escrita *weReq*, garantindo assim o correcto salto de endereços na memória a fim de colocar na posição equivalente a linha mesmo por baixo da anteriormente enviada. No fim de cada ciclo, *addrWin3* é incrementado garantido a deslocação para os 4 *pixels* ao lado da imagem resultado.

Chegado ao fim da linha1, isto é, *CountX1*, o endereço de início de linha (banda de observação) é incrementado de $7*(W/4)$ posições. O processo repete-se assim sucessivamente até se varrem as $H/8$ bandas de observação. No fim deste último caso, o módulo liberta o testemunho, *tokenSADout*, de forma a activar o módulo *interfaceVideo*. A sucessiva activação mútua permite ao processo repetir-se indefinidamente, gerando assim os valores de disparidade na saída *Pout* como é pretendido.

Dito isso resta agora demonstrar o módulo implementado em funcionamento por meio de uma bancada de teste adequada. Notar que este módulo requer um número elevadíssimo de ciclos de relógio pois gera uma grande quantidade de instruções e processos. Fazer uma cópia da totalidade da simulação seria inviável. Convém então tentar focar os pontos mais importantes da simulação com base nas seguintes figuras.

1. Inicialização, pedido de janelas iniciais, determinação do nível de correspondência e geração das sucessivas janelas2.

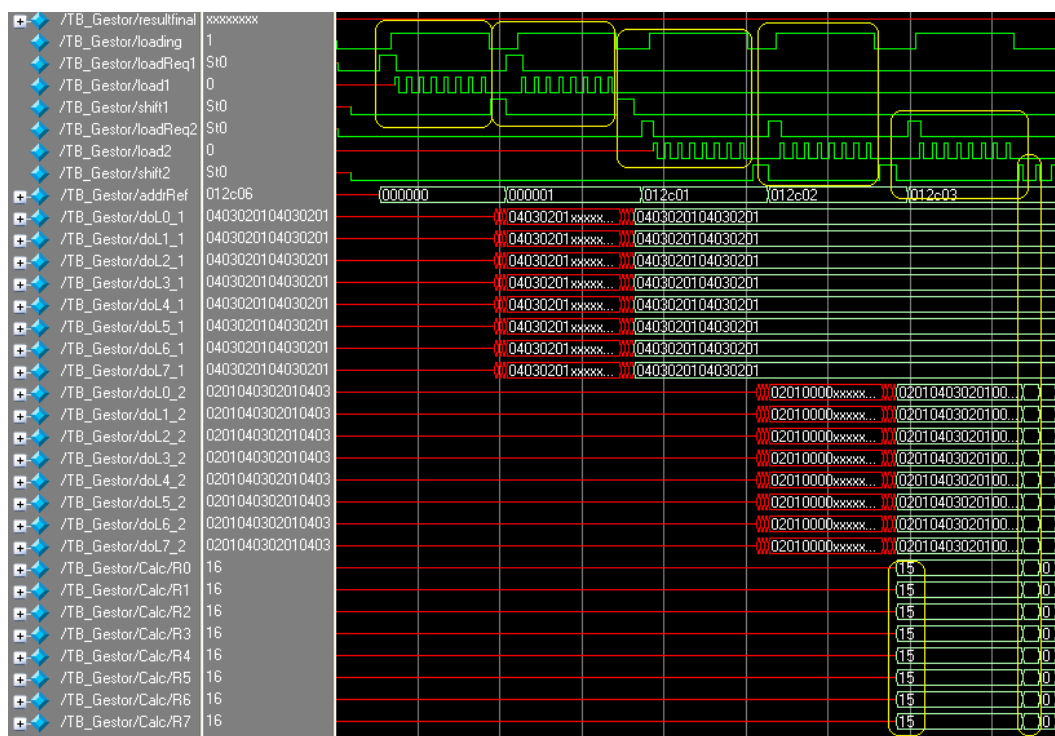


Figura 4.28 - Resultado da simulação TB_Gestor - parte1.

Na Figura 4.28 pode observar-se os dois pedidos de *loadReq1* onde entre cada um deles se fazem 4 *shifts* para esvaziar a zona de memória de depósito, a fim de se poder colocar mais 4 novos *pixels* em cada uma das 8 linhas. Notar que o módulo apenas envia um *loadReq1* para o módulo interfaceRAM. Este último encarrega-se de preencher a memória com os 4 novos pixels nas 8 linhas da memória. O mesmo acontece com a janela2. Notar o correcto incremento do endereço e mudança deste com a troca de janela pedida em *addrRef*. Preenchidas as zonas visíveis da memória o módulo cálculo transmite os resultados dos 8 níveis de correspondência para as entradas do módulo Gestor. Feitos os passos de comparação observa-se que é pedido uma nova janela2. Não tendo esta, conteúdo no seu *buffer* é feito um pedido *loadReq2*. Pode agora ser feito um *shift* do seu conteúdo para actualizar a janela, tal como previsto. É de novo realizada a comparação. Notar agora que para a geração das três próximas janelas, é desencadeado directamente com um *shift*. É visível que o número de ciclos de relógio é muito menor.

Esse processo repete-se até o contador interno *CountX2* ser incrementado até *maxDisp* vezes ou chegar ao fim da linha da imagem. A próxima simulação verifica se tal acontece.

2. Mudança de janela de referência.

Para ser possível verificar esse controlo foi necessário monitorizar estes sinais internos ao módulo. Foi colocado maxDisp igual a 255. Notar também que a imagem utilizada tem um valor W de 640, assim sendo o varrimento tem por condição final maxDisp.

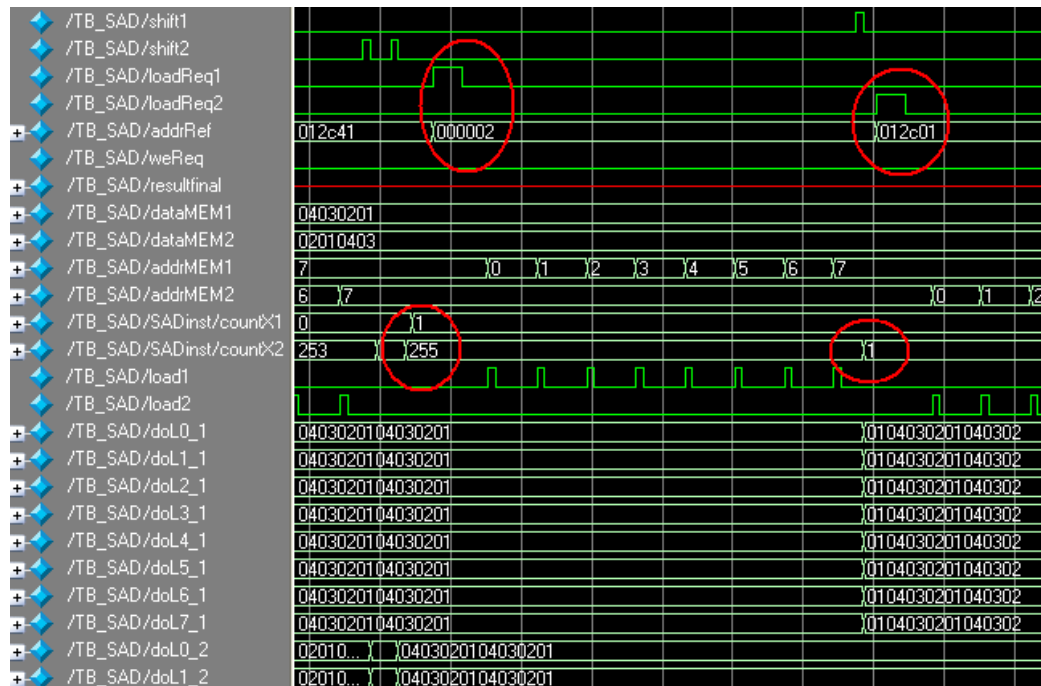


Figura 4.29 - Resultado da simulação TB_Gestor - parte2.

Tal como era esperado, a figura 4.28 mostra o valor para *CountX2* máximo de 255, pois *CountX2* é inicializado com *CountX1* igual a 1 ao início da linha. *CountX2* é um indicador de posição na imagem. Com base na diferença destes dois registos sabe-se quantas janelas já foram propostas para aquela janela1. De notar também o incremento do *CountX1*, pois foi pedida uma nova janela1 pois o *buffer* da memória auxiliar 1 não tinha dados. Notar a actualização de *CountX2* com o valor de *CountX1* como suposto, a fim de se poder saber em que posição da janela se encontra a janela2.

3. Pedido de escrita dos resultados, *weReq*.

Ao fim da atribuição do nível de correspondência de 4 janelas padrão deve ser activo o pedido de envio dos 8 resultados para a memória.

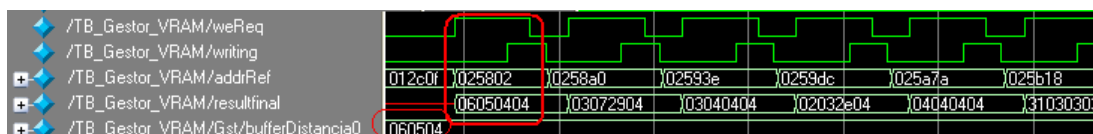


Figura 4.30 - Resultado da simulação TB_Gestor - parte3.

A Figura 4.30 ilustra o registo de contagem do número de resultados guardados, *countBufferDist0*, com um valor máximo de 3. Quando chega um novo resultado esse é enviado juntamente com os outros 3 para a saída *result*.

De notar a troca do valor na saída para o endereço onde se pretende guardar este resultado. Apenas foi focado o caso da linha de índice 0, pois como já foi verificado, as restantes 7 linhas comportam-se como suposto.

4. Condição de fim de banda de observação e início de uma nova.

O processo de geração de janelas e determinação dos resultados de correspondência estando confirmados, falta agora verificar a mudança de banda de observação, isto é, verificar se quando se chega ao fim de uma linha da imagem1, o sistema reinicia bem o mesmo ciclo para as 8 linhas seguintes da imagem, banda de observação seguinte. Este teste foi realizado para uma imagem de 640*480.

A Figura 4.31 contém o intervalo de tempo que mostra claramente a passagem da primeira banda de observação para a segunda.



Figura 4.31 - Resultado da simulação TB_Gestor - parte4.

Esse valor é guardado no registo *countBandas*, onde se vê a transição para do valor. O registo interno do endereço da janela1, *addrWin1*, foi actualizado com o valor 1280 (640 pixels / 4 pixel por endereço * 8 linhas da imagem). Tal ocorreu realmente para a última janela 1 possível, pois *CountX1* igual a 633. Se somarmos as restantes 7 colunas da janela, obtemos bem o valor 640.

De notar o pedido de *loadReq1*, para preencher a nova janela1 a partir do endereço -1200.

5. Condição de fim de imagem

Resta verificar a condição de fim de imagem a fim de garantir que o fim da imagem é bem detectado e que o testemunho é libertado, permitindo assim a reactivação do módulo *interfaceVideo*. Este teste foi realizado para uma imagem de 640*480.

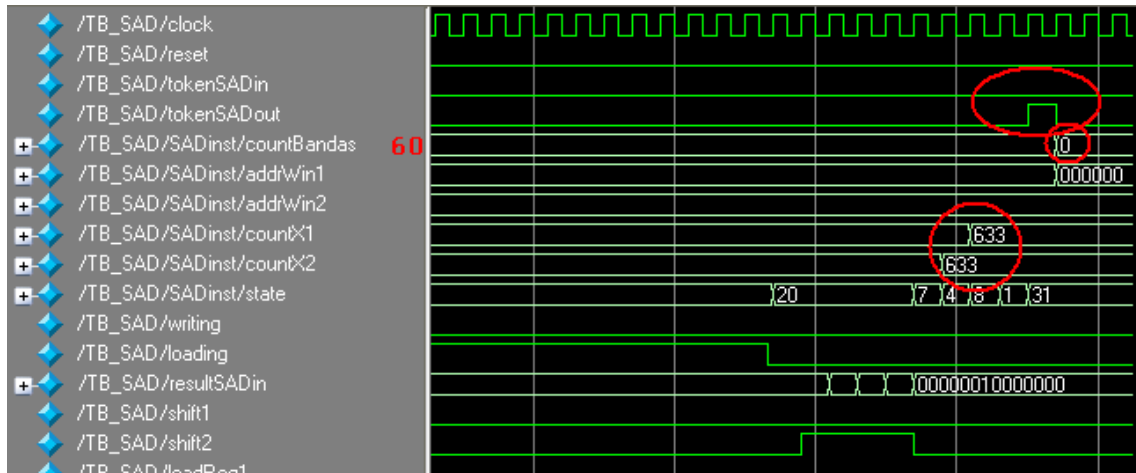


Figura 4.32 - Resultado da simulação TB_Gestor - parte5.

A Figura 4.32 não permite que seja visto o valor 60 do contado de Bandas por motivos de largura de imagem. Mas pode ver-se a passagem do *tokenSADout* para activo assim com a reinicialização dos registos *countBandas* e *addrWin1* indicando o “reset” geral do módulo. Notar que o regresso do *tokenSADout* ao nível inactivo, não indica a início de um novo ciclo. Esse só começa com a transição ascendente da entrada *tokenSADin*, proveniente do módulo *interfaceVideo* no fim do seu processo.

Como já foi referenciado mais acima na secção 4.3.1, perante os resultados obtidos foram feitas alterações ao algoritmo. Estas alterações obrigaram a modificações neste módulo assim como no módulo de cálculo da métrica, no entanto não se achou necessário voltar a expor todo o funcionamento do módulo pois a generalidade do módulo permanece inalterada. As alterações feitas serão todas estipuladas no capítulo 5.

4.3.7 - Interligação dos vários módulos

Este conjunto de módulos está integrado no módulo principal, chamado Geral, a fim de se poder integrar com facilidade no restante trabalho já desenvolvido. A Figura 4.33 mostra o resultado visível dos blocos lógicos interligados formando o módulo Geral.

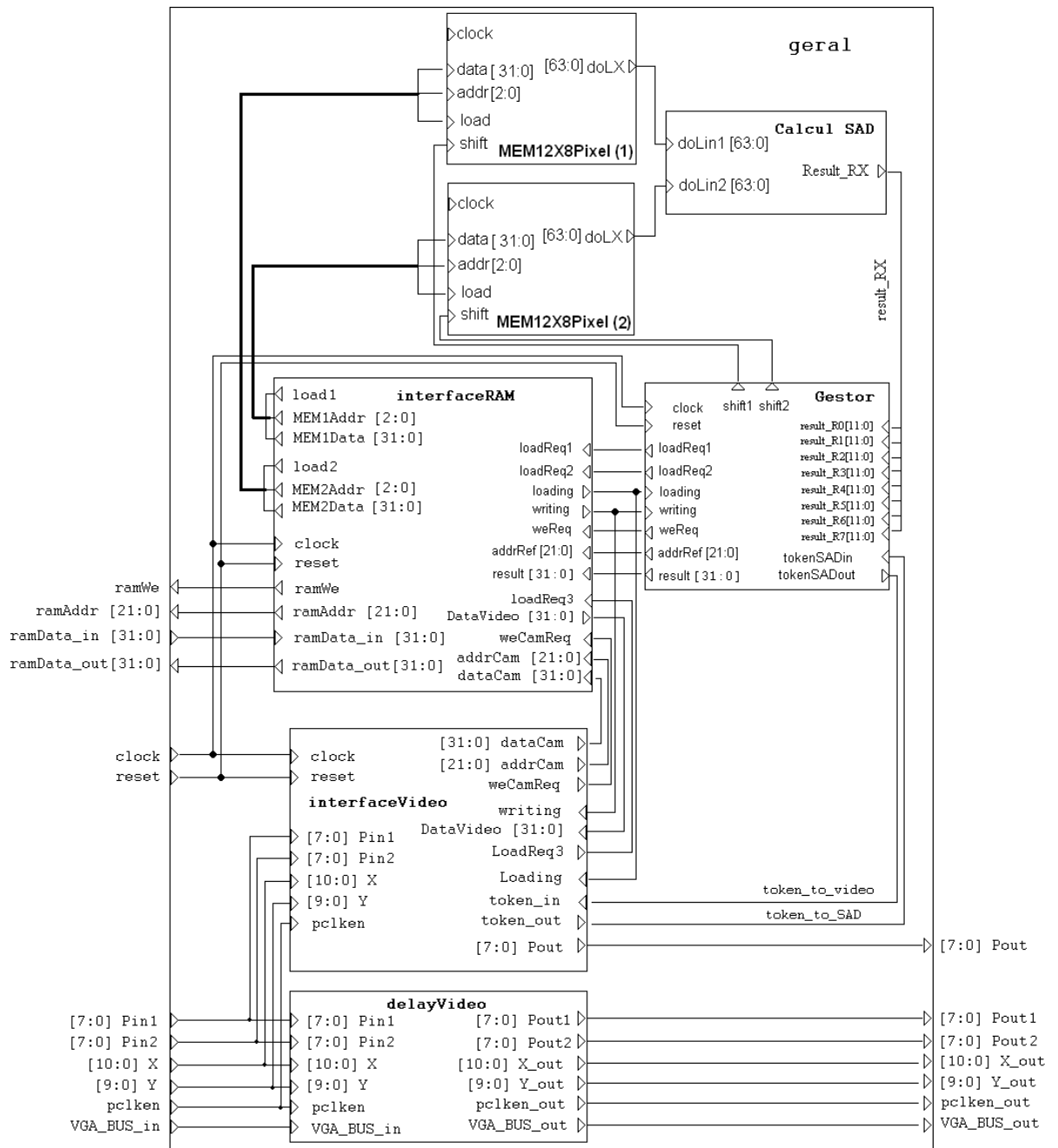


Figura 4.33 - Interligação dos diferentes módulos.

O módulo Geral por sua vez integra-se no ficheiro stereocam, onde se interliga com os restantes módulos fornecidos como ponto de partida para o trabalho, excepto a ligação à memória SDRAM. A Figura 4.34 pretende dar uma panorâmica da presença do módulo desenvolvido e as suas interligações para o exterior.

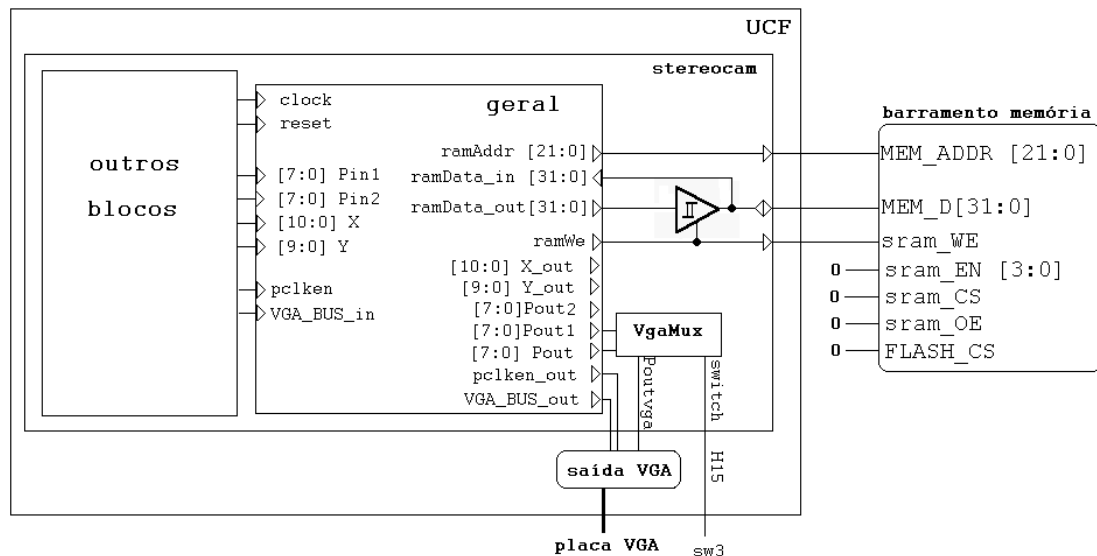


Figura 4.34 - Diagrama de blocos lógicos toplevel.

Falta referir a implementação do *buffer-tristate* para permitir a utilização bidireccional do barramento da memória SDRAM. O seleccionador é o mesmo pino de controlo que activa a escrita na memória, *ramWE*. Quando ao nível lógico baixo, é habilitado o modo de escrita na memória. Portanto este dispositivo coloca na linha o valor contido em *ramData_out*. Caso contrário, com *ramWe* activo, não deixa passar os sinais para a linha. Possibilita assim a recepção dos valores provenientes da memória para a entrada *ramData_in*.

Graças ao módulo *vagaMux* pode ser visto na saída, a imagem de referência, directamente proveniente da câmara Cam1, ou então a saída dos níveis de disparidade calculados, pressionando o botão *sw3* existente na placa de desenvolvimento. Este módulo é um simples multiplexador de 8 bits, daí não se ter apresentado nenhuma secção sobre este módulo. A cor do *pixel* representa a distância do próprio ponto às câmaras, por níveis de cinzento consoante ele se encontre mais longe ou mais perto, ficando mais ou menos escuro respectivamente.

4.4 – Síntese e implementação do circuito

Durante a implementação e teste dos diferentes módulos no ISE, determinaram-se os recursos dispendidos na implementação em FPGA. A Tabela 4.7 apresenta um resumo destes diferentes parâmetros.

Tabela 4.7 - Volume de recursos necessários para implementação na FPGA.

Módulo	Slice Flip-flop	%	LUT de 4 entradas	%	Total LUT
interfaceRAM	73	1%	203	1%	212
interfaceVideo	220	1%	244	1%	293
Gestor	189	1%	382	1%	447
delayVideo	0	0%	0	0%	0%
Geral	2104	7%	4417	16%	4540
vgaMux	0	0%	8	1%	8
stereocam	2699	10%	5622	21%	5810

Olhando para as percentagens, podemos concluir, como foi dito durante o decorrer deste documento, que o volume de recursos ocupados não é factor crítico, pois o volume de lógica ocupada ronda um total de 21%, isto integrando a totalidade do trabalho contido no módulo stereocam. Justifica-se assim a preocupação com o número de ciclos de relógio em detrimento dos recursos ocupados.

4.5 – Testes práticos

Sem acesso à memória SDRAM não foi possível armazenar as imagens, o que inviabiliza a implementação prática do trabalho e assim a visualização de resultados reais. No entanto foi implementada uma solução alternativa que consiste no uso de uma bancada de teste onde se integrou uma memória alternativa à SDRAM. Uma melhor explicação sobre o desenrolar do teste, mencionando os passos importantes para a obtenção dos resultados, é dada de seguida.

Para se poder testar o sistema é necessário ter-se um par de imagens estereoscópicas. Numa fase introdutória, com o objectivo de testar o correcto funcionamento da totalidade do módulo, foi utilizado um par de imagens obtidas no site Web (15), apresentadas aqui na Figura 4.35.



Figura 4.35 - Par de imagens de teste (15).

Como se pode ver existe discrepância entre elas. Estas imagens têm formatação *bitmap*, o que impossibilita que se trabalhe directamente sobre elas, pois esta formatação não organiza os dados sobre valores binários de 0 a 255 por cada pixel tal como acontece com o par de câmaras. Assim sendo é necessário converter este par de imagens para um formato onde a intensidade de cada pixel fosse representada por um valor binário de 0 a 255. Para tal utilizou-se o *software* Matlab. Este dispõe de uma função que permite ler o ficheiro (imagem) e enviar para um ficheiro o resultado da leitura em hexadecimal, tal como é pretendido.

```
imagem=imread('pentagon_left.bmp', 'bmp')
fid = fopen('pentagon_left.txt', 'w')
fprintf(fid,'%x\r\n',imagem)
fclose(fid)
```

Estas quatro linhas permitem gerar a codificação hexadecimal da imagem “pentagon_left.bmp”, imagem mais a esquerda na Figura 4.35. Um processo idêntico é feito para a imagem direita, “pentagon_right”. É pressuposto que o leitor tenha conhecimento sobre este tipo de instruções e das suas funções.

O passo seguinte consiste em criar uma zona de memória na bancada de teste e invocar a função *\$readmemh* que lê o conteúdo do ficheiro e coloca por ordem crescente de endereços os valores nele contidos, preenchendo assim a memória com o conteúdo do ficheiro “pentagon_left.txt”. O mesmo se processa para a imagem direita, no entanto o resultado é descarregado para a zona da memória reservada para a segunda imagem, isto é, a partir do endereço 12C01h até 25801h. O enxerto da descrição Verilog com estas duas funções é apresentado de seguida.

```
//----preencher as memórias.
$readmemh ( "pentagon_left.txt" , RamMEM, 0,32'h12C00 );
$readmemh ( "pentagon_right.txt", RamMEM, 32'h12C01, 32'h25801 );
```

A partir deste momento a memória contém as duas imagens. As imagens sendo de tamanho 256x256 *pixels* não preenchem a totalidade do espaço para elas reservado. No início de cada simulação é necessário a introdução das dimensões da imagem para permitir tanto ao módulo de teste como ao módulo Gestor saberem até onde se encontram dados na memória para serem processados e onde se encontra cada linha da imagem.

O módulo de teste tem de atender os pedidos de acesso à memória tanto para leitura como para escrita dos resultados. O pedido de leitura desencadeia um conjunto de 8 leituras, carregando para a janela de observação respectiva as 8 linhas consecutivas da imagem. A descrição Verilog não é aqui colocada mas pode ser consultada no endereço Web da página do trabalho caso se queira saber com maior pormenor a forma utilizada para resolver este problema de gestão de atendimento dos pedidos e seus endereçamentos.

De salientar que a cada pedido de escrita, é colocado o resultado na memória e apenas no fim é passado para um ficheiro de forma a ser possível a consulta do resultado das disparidades obtidas. Para tal é invocada a função *\$fdisplay*.

Antes de prosseguir com a descrição dos passos implementados, é necessário fazer uma introdução a formatação PGM (*portable gray map*).

Os pixels da imagem são armazenados sobre a forma de valores binários e dispostos na forma de uma matriz bidimensional com o tamanho da imagem. Para uma melhor informação aconselha-se a consulta da referência (16). Com isto pretende-se apresentar a via encontrada para a visualização dos resultados de uma forma rápida e confortável. De notar a necessidade de colocar, no início do teste, um cabeçalho onde se indica a dimensão da imagem e a escala dos níveis de cinza. Uma forma eficiente de se gerar automaticamente o valor do tamanho da matriz consiste em obter estes valores como sendo o resultado de uma equação genérica com parâmetros W e H do par de imagens em estudo. Essa linha é apresentada de seguida

```
$fdisplay(mcd,"%d%d\n%d",((W-7)>>2)*4,(H>>3)*8,((W-7)>>2)*4);
```

De referir que *mcd* é o descritor do ficheiro onde se processa a escrita. Os três parâmetros são respectivamente o comprimento da imagem, largura e escala de cinza. A sua determinação está descrita na secção 4.1.5. Tal como foi dito, o ficheiro é preenchido com os valores das disparidades obtidas no fim do processo. A linha seguinte mostra a forma com é feita a transferência dos dados.

```
for(pre=0;pre<(((W-7)>>2)*((H>>3)*8));pre=pre+1)//clico de escrita
begin
  ramwire = RamMEM[32'h25802+pre];
  $fdisplay(mcd,"%d %d %d %d",
    ramwire[31:24],ramwire[23:16],ramwire[15:8],ramwire[7:0]);
end
$fclose(mcd); //fechar descritor
```

Onde *pre* é um contador de 32 bit, que vare a totalidade da zona dos resultados, *ramwire* é um barramento de 32 pinos da saída da memória. Este barramento, como se pode ver na quarta linha, existe para simplificar a descrição da separação em blocos de 8 bits o

barramento de 32 da saída da memória afim de se poder guardar o valor no ficheiro os quatros resultados na forma decimal a fim de respeitar a codificação PGM.

Antes de passar para a análise dos resultados é convenientes assegurar-se que os valores estão a ser enviados para as janelas na ordem correcta e que os resultados expostos pelo módulo de cálculo são bem passados para o módulo Gestor pelas entradas R0 a R7. Para este teste é crucial a utilização de uma figura onde o valor dos pixels fosse o mais variado possível.

Com recurso ao *software* Matlab, foram gerados os dois ficheiros de texto, tal como já foi apresentado, para permitir carregar a memória virtual da bancada de teste. Para uma comparação fiável e cómoda, foram introduzidos os comandos de forma a serem apresentadas as 8 primeiras colunas e linhas de cada imagem. O Matlab apresentou o seguinte par de matrizes:

left =	right =
148 147 144 142 150 146 149 151	146 142 141 144 141 142 137 133
145 144 139 142 147 144 146 147	134 143 143 139 136 137 133 137
145 145 142 145 145 149 149 148	138 145 150 147 146 142 137 141
159 154 154 146 146 138 143 142	145 154 161 161 159 154 153 145
163 153 154 146 145 135 129 141	141 151 161 164 160 151 151 144
158 153 145 146 142 138 140 137	132 139 152 159 156 153 143 137
161 155 153 145 137 137 138 114	131 140 149 152 154 151 150 140
158 151 148 147 138 129 136 123	138 147 155 158 155 145 146 141

Esta base é decimal, no entanto a visualização do barramento na simulação pode ser feita byte a byte apresentando o conteúdo em formato hexadecimal. Assim sendo foram convertidas as apresentações das matrizes *left* e *right* pelo meio destes comandos:

```
fprintf('%x %x %x %x %x %x %x %x\n',right') e
fprintf('%x %x %x %x %x %x %x %x\n',left')
```

Os resultados obtidos foram:

(esquerda)	(direita)
94 93 90 8e 96 92 95 97	92 8e 8d 90 8d 8e 89 85
91 90 8b 8e 93 90 92 93	86 8f 8f 8b 88 89 85 89
91 91 8e 91 91 95 95 94	8a 91 96 93 92 8e 89 8d
9f 9a 9a 92 92 8a 8f 8e	91 9a a1 a1 9f 9a 99 91
a3 99 9a 92 91 87 81 8d	8d 97 a1 a4 a0 97 97 90
9e 99 91 92 8e 8a 8c 89	84 8b 98 9f 9c 99 8f 89
a1 9b 99 91 89 89 8a 72	83 8c 95 98 9a 97 96 8c
9e 97 94 93 8a 81 88 7b	8a 93 9b 9e 9b 91 92 8d

A sequência de dados enviados para as janelas pelo módulo, assim como o resultado dos cálculos dos níveis de correspondência durante a simulação estão apresentados na Figura 4.36.

/TB_Gestor_VRAM/doL0_1	979592968e909394	8e909394xxxx	979592968e909394				
/TB_Gestor_VRAM/doL1_1	939290938e8b9091	8e9b9091xxxx	939290938e8b9091				
/TB_Gestor_VRAM/doL2_1	94959591918e9191	918e9191xxxx	94959591918e9191				
/TB_Gestor_VRAM/doL3_1	8e8f8a92929a9a9f	929a9a9fxxxx	8e8f8a92929a9a9f				
/TB_Gestor_VRAM/doL4_1	8d818791929a99a3	929a99a3xxxx	8d818791929a99a3				
/TB_Gestor_VRAM/doL5_1	898c8a8e9291999e	9291999exxxxx	898c8a8e9291999e				
/TB_Gestor_VRAM/doL6_1	728a898991999ba1	91999ba1xxxx	728a898991999ba1				
/TB_Gestor_VRAM/doL7_1	7b88818a9394979e	9394979exxxxx	7b88818a9394979e				
/TB_Gestor_VRAM/doL0_2	7f7b858f94928e8a			908d8e92xxxxxx	85898e8c9f08d8e92		
/TB_Gestor_VRAM/doL1_2	6c7184878b8c898b			8b8f8f88xxxxxx	898f89888b8f88		
/TB_Gestor_VRAM/doL2_2	65697e7d7c8b8b8a			9396918axxxxxx	8d898e929396918a		
/TB_Gestor_VRAM/doL3_2	65697078848a8587			a1a19a91xxxxxx	81999a91a1a19a91		
/TB_Gestor_VRAM/doL4_2	64605f07a817d87			a4a1978dxxxxxx	909797a0a4a1978d		
/TB_Gestor_VRAM/doL5_2	66615f5f71858683			9f988b84xxxxxx	898f998c9f988b84		
/TB_Gestor_VRAM/doL6_2	6e635d6662748581			98958c83xxxxxx	8c96979a98958c83		
/TB_Gestor_VRAM/doL7_2	7470635e697d827d			9e9b938axxxxxx	8d92919b9e9b938a		
/TB_Gestor_VRAM/R0	93				55		
/TB_Gestor_VRAM/R1	113				60		
/TB_Gestor_VRAM/R2	171				44		
/TB_Gestor_VRAM/R3	206				78		
/TB_Gestor_VRAM/R4	268				105		
/TB_Gestor_VRAM/R5	259				82		
/TB_Gestor_VRAM/R6	260				125		
/TB_Gestor_VRAM/R7	224				103		

Figura 4.36 - Resultados da simulação com o uso da memória virtual.

Como se pode observar, estes valores coincidem perfeitamente com os elementos das matrizes. Notar que a ordem aparece invertida, pois o fluxo das janelas de observação é invertido, isto é, o primeiro pixel a chegar a memória vai sendo deslocado sucessivamente para a direita, até se encontrar no menor índice.

Os valores dos níveis de correspondência obtidos têm de ser confirmados, realizando os cálculos sobre os valores presentes no par de janelas. No sentido simplificar o processo de cálculo deste valor, foi utilizado novamente o Matlab para o cálculo da matriz

Aplicou-se sobre as matrizes *left* e *right* para o cálculo do resultado da primeira linha, a seguinte equação. De notar que estão definidas como inteiros sem sinal, logo um resultado negativo é substituído pelo valor nulo, daí se fazer a soma das duas diferenças possíveis.

```
>> R0=sum((left(1,:)-right(1,:))+(right(1,:)-left(1,:)))
```

R0 =

55

Este passo foi feito para as restantes 7 linhas. Os valores coincidem com o apresentado na simulação pondo-se assim de parte a hipótese dos cálculos estarem a ser feitos de forma errada.

4.6 - Determinação do tempo de processamento

Sendo o tempo um requisito essencial para este trabalho, é importante estimar-se a taxa de geração dos mapas de disparidade.

Para tal incorporaram-se na banca de teste os ciclos de relógio que cada processo consome na realidade, contabilizando deste modo o número total de ciclos de relógio dispendidos para o processamento de um par de imagens. Esta simulação contabiliza também o número de pedidos feitos a memória durante o processamento da imagem, no entanto a análise destes valores fica fora do nosso âmbito. Foi determinada a influência de cada um dos 3 parâmetros sobre o número de ciclos necessários.

Como é de esperar o número de pedido de escrita de resultados (weReq) e de leitura de janelas1 (loadReq) apenas depende do tamanho do par de imagens, pois o endereço destes pedidos apenas ocorre uma só vez, ao contrário das janelas2 que são lidas mais uma vez, isto sempre que se gera uma nova janela1 grande parte da vezes são relidas maxDisp janelas2, o que se traduz num aumento do número d. No entanto o interesse deste estudo centra-se na análise do número dos ciclos de relógio dispendidos para o processamento de um par de imagens.

- Influência da variação altura (H) da imagem.

Tabela 4.8 - Influência do número de linhas (H).

Influência de H					
W	256				
H	256	200	128	50	25
MaxDisp	25				
Clk	3932840	3072533	1966424	737414	368711
Load1Req	16384	12800	8192	3072	1536
Load2Req	536064	418800	268032	100512	50256
weReq	7936	6200	3968	1488	744

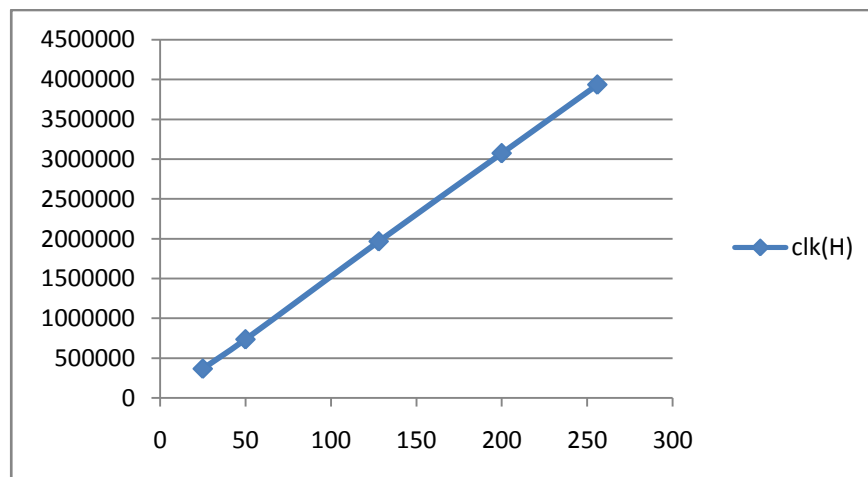


Figura 4.37 - Influência de H sobre o número de ciclos de relógio necessários.

Como se pode comprovar pela consulta do gráfico da Figura 4.37, que é a representação gráfica dos valores da Tabela 4.8, o número de ciclos de relógio varia linearmente com o valor H, tal se explica pelo facto que o varrimento sendo feito na horizontal, se uma imagem tiver metade da altura, esse vai gerar metade das bandas de observação. No entanto esta afirmação não está de todo correcta, fica-o se for dito que o número de ciclos de relógio depende linearmente com o número de bandas de observação. Consultando a secção 4.1.5, foi determinada a fórmula da relação entre o número de bandas de observação B em função do número de linhas da imagem H:

$$B = \lfloor H/8 \rfloor * 8$$

Assim sendo com base uma imagem com um valor W e maxDisp iguais pode-se calcular com exactidão o número de ciclos que vão ser necessário para uma nova imagem com um número diferente de linhas.

- Influência da variação da largura (W) da imagem.

Tabela 4.9 - Influência do número de colunas (W).

Influência de W				
W	640	350	250	150
H	480			
MaxDisp	60			
Clk	3745986 8	1963254 8	1345854 8	7284548
Load1Req	76800	42240	30240	18240
Load2Req	5093760	2672160	229020	124020
weReq	37920	20400	14400	8400

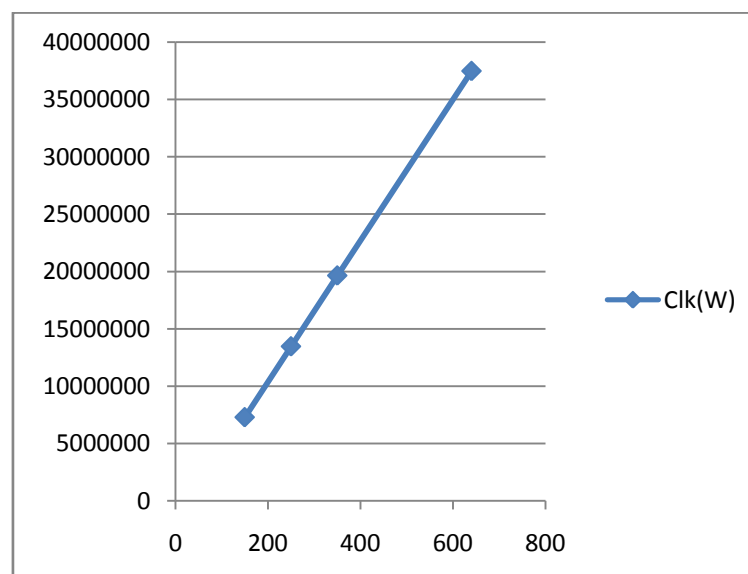


Figura 4.38 - Influência de W sobre o número de ciclos de relógio necessários.

Pela consulta do gráfico da Figura 4.38 referente a Tabela 4.9 pode observar-se que um aumento deste parâmetro traduz-se num aumento linear do número de ciclos de relógio necessários, pois o aumento de uma unidade de largura W traduz-se no aumento de maxDisp correspondências.

- Influência da variação do nível de pesquisa da máxima disparidade (maxDisp) na imagem.

Tabela 4.10 - Influência do nível máximo de disparidade (maxDisp).

Influência de maxDisp					
W	256				
H	256				
MaxDisp	255	180	100	50	25
Clk	15711656	14607208	10513768	6422056	3932840
Load1Req	16384	16384	16384	16384	16384
Load2Req	2055360	1979392	1426432	872960	536064
weReq	7936	7936	7936	7936	7936

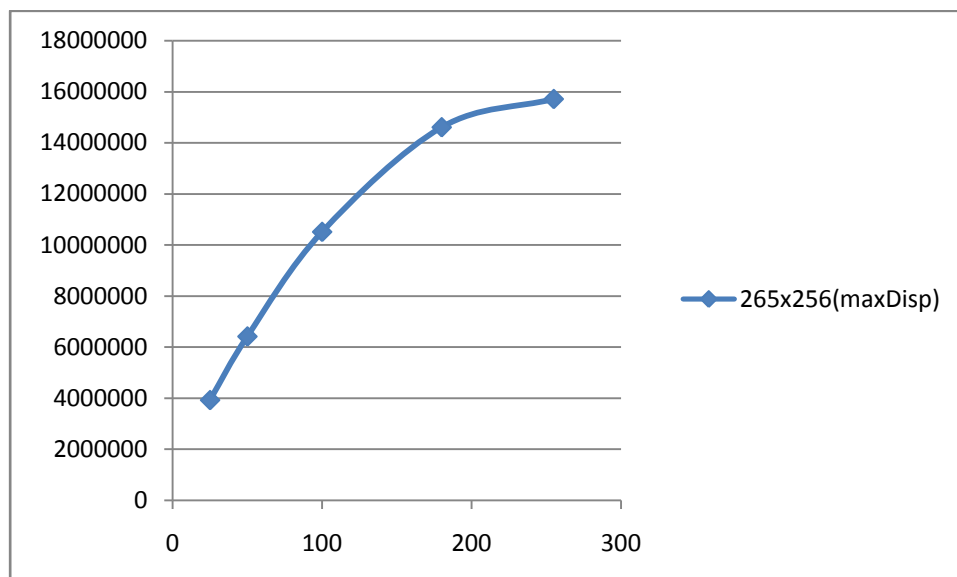


Figura 4.39 - Influência de maxDisp sobre o número de ciclos de relógio necessários.

O gráfico da Figura 4.39 representa os valores obtidos na Tabela 4.10 onde se pode observar o efeito da variação de maxDisp no tempo de processamento. Sendo maxDisp responsável pelo nível de correspondências atribuídas a uma janela padrão a sua variação vai reflectir-se numa influência quadrática, pois o aumento de uma unidade de maxDisp implica um aumento de cerca de $W \cdot H$ correspondências.

Desta análise pode se concluir que é importante conseguir arranjar um compromisso entre a resolução da imagem e o tempo de processamento necessário. A escolha do valor da máxima disparidade esperada (maxDisp) tem um peso computacional exponencial sobre o processo, assim sendo é crucial saber-se para que tipo de cenário se pretende utilizar o sistema, pois um cenário distante terá um nível de disparidade muito baixo, já um cenário mais junto das câmaras oferece disparidades mais elevadas. Mas por outro lado se o cenário for perto não é necessário gerar imagens com tão alta definição, já nos cenários distantes esta característica é crucial para se poder distinguir os objectos na imagem.

Tabela 4.11- Tempo de processamento para imagens de 640x480.

Influência de maxDisp para imagem (640x480)				
W	640			
H	480			
MaxDisp	200	100	50	25
Clk	99708068	57015068	32351108	19187828
Load1Req	76800	76800	76800	76800
Load2Req	13527360	7743360	4401600	2617920
weReq	37920	37920	37920	37920

A Tabela 4.11 pretende dar uma panorâmica dos tempos necessários para o processamento um par de imagens com a máxima resolução que as câmaras fornecem, 640x480. Pode-se verificar que esta análise foi feita para vários valores de disparidade esperada, maxDisp.

Configurado o valor de disparidade máxima num valor alto, 200, conclui-se que o processo demora 99708068 ciclos de relógio, funcionando este a cadência de 100MHz, totaliza 0.997seg, isto é 1segundo. Esta informação diz-nos que a taxa de actualização, para este tipo de imagem e configuração, é de 1 segundo. Este valor pode parecer demasiado baixo, mas também convém ressaltar que estas condições não fazem sentido na medida em que se está a utilizar a máxima resolução (para cenários distantes), com um nível de disparidade esperado elevado (para cenários próximos).

Capítulo 5

Análise de resultados

Como já foi referido a implementação prática consistiu no processamento do par de imagens estereoscópicas apresentadas na Figura 4.35. Este capítulo inicia-se pela apresentação do resultado obtido com uma primeira versão, explicada em detalhe durante o quarto capítulo. São analisados os problemas encontrados e apresentadas diversas melhorias da versão. De forma a dar ao leitor uma melhor noção das mudanças, são focadas as partes que são alteradas nas duas versões.

5.1 - Resultados obtidos

Feita a verificação da correcta colocação dos valores e cálculo dos níveis de correspondência, foi iniciada uma nova simulação contendo agora um novo par de imagens. Estas imagens possuem um baixo nível de complexidade. Foram criadas com o propósito de gerarem um resultado de simples observação, permitindo observar-se nitidamente o resultado, interpretar o comportamento do algoritmo desenvolvido, tirar conclusões sobre a sua eficiência e sugerir alterações ao módulo para tentar melhorar a resposta final.



Figura 5.1 - Par de imagem teste - a risca.

Como caso inicial foi desenhada uma imagem de 48x48 *pixels* contendo apenas uma risca vertical sobre um fundo homogéneo. A imagem correspondente direita, consiste na deslocação dessa risca para a direita, tal como ilustra a Figura 5.1.

Mais uma vez estas imagens são criadas com o formato *bitmap*. É necessário recorrer ao uso do Matlab para gerar os ficheiros com a codificação hexadecimal das imagens. Em caso de dúvidas aconselha-se a consulta da secção 4.5, onde é explicada a geração destes ficheiros assim como a passagem do conteúdo para a memória virtual *RamMEM* da bancada de teste. Importante referir que a imagem resultante já não é quadrada, isto é, 48x48, mas sim de 40x48. A razão destes valores pode ser revista na secção 4.1.5. Sendo a imagem muito estreita, apenas 40 pixels de comprimento, o valor máximo possível de uma disparidade nunca pode ultrapassar 40. Numa escala de 256 níveis, o caso de maior disparidade possível, 33 (40-7) continua a ser um valor pouco diferenciável do valor nulo (cor preta). Como já foi falado sobre na apresentação do formato PGM, as imagens contêm um cabeçalho onde uma linha é reservada a escala dos valores em jogo. De forma a melhorar a visualização, o valor máximo 255 passou a ser definido automaticamente como sendo a dimensão da imagem resultante, neste caso concreto 40. Repete-se aqui a linha de código apresentada na secção 4.5, onde se ilustra a utilização deste parâmetro, terceiro e último da linha.

```
$fdisplay (mcd,"%d%d\n%d",((W-7)>>2)*4),(H>>3)*8,((W-7)>>2)*4);
```

Com esta alteração de gama consegue-se obter uma imagem com níveis de disparidades visíveis, tal como o mostra a Figura 5.2, onde se apresenta o resultado obtido.

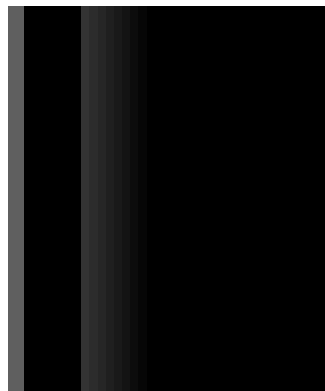


Figura 5.2 - Resultado da disparidade "risca".

Antes de mais convém dizer que a análise deste resultado resume-se ao estudo de uma das linhas, pois são todas iguais entre elas, tal como sucede no par de imagens que deram origem a este resultado.

Sem entrar em grande detalhe consegue-se desde logo observar que ocorreu correspondência, pois surge na imagem uma tira cinza e mais à frente um “*dégradé*”. Para poder prestar uma análise mais legível e cómoda, foi extraído a parte inicial dos valores dos pixels da imagem do resultado.

```
15 15 0 0 0 0 0 0 0 8 7 6 5 4 3 2 1 0 0 0
```

Apenas foram apresentados os 20 primeiros *pixels* da linha pois os restantes 20 contêm todos a informação de *pixel* nulo. Recorda-se que os valores destes *pixels* traduzem, a distância a que foi encontrada a janela homóloga, mais correctamente, a janela mais semelhante à janela de referência.

O primeiro par de correspondências encontradas ocorreu 15 pixels mais a frente. Para confirmação é apresentado de seguida as primeiras linhas de ambas as imagens.

Sequência dos 20 primeiros pixels da primeira linha de *risca_left*:

```
192 128 192 192 192 192 192 192 192 192
192 192 192 192 192 192 192 192 192 192
```

Sequência dos 20 primeiros pixels da primeira linha de *risca_right*:

```
192 192 192 192 192 192 192 192 192 192
192 192 192 192 192 192 128 192 192 192
```

Como se pode ver o fundo da imagem é composto por pixels de intensidade 192 e a risca pelo valor 128. A primeira atribuição será feita com a janela de referência:

```
192 128 192 192 192 192 192 192
```

Começando o varrimento da janela2 na imagem *risca_right* a partir da mesma posição que começa a janela1 observa-se que a correspondência é máxima ao fim de 15 deslocações tal como surgiu no resultado.

A segunda correspondência tem por imagem de referência, janela1:

```
128 192 192 192 192 192 192 192
```

Fazendo a mesma análise que para o caso anterior, conclui-se que a melhor correspondência ocorre também ao fim de 15 deslocações.

Sabendo que o resto da linha é todo preenchido com o valor de fundo de imagem, 192, poder-se-ia pensar que as correspondências seguintes seriam todas de valor nulo. No entanto olhando para o resultado gerado tal não acontece. A décima correspondência devolve o valor 8 e as 7 correspondências seguintes também devolvem um resultado não nulo. Este resultado acontece devido ao posicionamento inicial da janela2 na imagem direita.

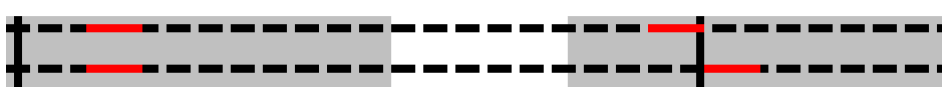


Figura 5.3 - Décima correspondência.

A Figura 5.3 exemplifica o caso da décima correspondência. Na primeira linha estão representadas as janelas iniciais, a vermelho. A janela esquerda apenas contém pixel de fundo de imagem (valor 192), o mesmo já não acontece com a janela2 que contém entre os pixels de fundo de imagem, um pixel pertencente a risca (valor 128). No entanto ao fim de 8 deslocções da janela2, este pixel fica de fora, passando a janela 2 contém também apenas pixels de valor 192. Nesta posição acontece o melhor nível de correspondência até agora. Este caso tendo ocorrido para a oitava deslocção, o valor 8 é guardado nesta posição. O que explica o tom cinza apresentado na imagem.

Uma análise idêntica acontece para a décima primeira correspondência, isto é o valor 7 atribuído como correspondência. Ambas as janelas começam agora uma posição mais a frente na imagem. A janela1 continua a possuir apenas pixels pertencentes ao fundo da imagem. Já a janela 2, também recomeça com um pixel pertencente a risca, mas este pixel encontra-se agora deslocado de uma posição tal como o ilustra a Figura 5.4.



Figura 5.4 - Décima primeira correspondência.

Assim sendo vão ser necessários apenas 7 deslocações para obter o nível de correspondência nula. O mesmo acontece para as 6 correspondências seguintes. Desde já se entende a razão do gradiente decrescente ao longo destas 8 atribuições. Notar que este gradiente possui tamanho finito 8. A Figura 5.5 ilustra o caso da décima oitava correspondência onde se pode ver que o valor da correspondência vai ser nulo, pois ambas as janelas apenas contêm pixels de fundo de imagem.



Figura 5.5 - Décima oitava correspondência.

Com o auxílio da imagem pode-se concluir que o tamanho do gradiente é igual a largura das janelas de observação. Assim se explica a razão da existência daquele “efeito de sombra” presente no resultado obtido. Com este exemplo comprova-se o bom funcionamento do algoritmo. Pode-se concluir que este método não apresenta bons resultados para imagens com zonas homogêneas demasiado grandes.

5.1.1 - Teste prático da primeira versão

Assim sendo passa-se a testar o método com o par de imagens reais estereoscópicas apresentadas na Figura 4.35, pois estas duas imagens apresentam um fundo com conteúdos diversificados, permitindo evitar o “efeito de sombra”.

Foi alterada a bancada de testes de forma a colocar na memória virtual as duas imagens, tal como aconteceu na secção 4.5. Em caso de dúvidas sobre o procedimento a executar aconselha-se a consulta desta secção. Analisando o par de imagens, observa-se que as discrepâncias entre os pontos não são elevadas. Assim sendo, numa perspectiva de reduzir o tempo de simulação, decidiu-se optar por um varrimento máximo de 50 posições por janela, isto colocando o registo maxDisp igual a 50.

O resultado obtido está apresentado na Figura 5.6.



Figura 5.6 - Disparidade do pentágono (escala 0-50).

O resultado permite definir o plano onde se encontra o pentágono pela população de *pixels* mais claros, indicando maior nível de disparidade, o que se traduz como estando num plano mais próximo do que o solo que se encontra geralmente em tons mais escuros. No entanto observa-se que existe muitos casos de correspondências brancas, o que indica que não foi encontrada correspondência perto.

Uma causa possível deste elevado nível de atribuições erradas pode se deve ao facto de as imagens não serem mesmo epipolares. As imagens aparentam ser epipolares, no entanto basta o desalinhamento vertical de uma linha para que esta versão do módulo Gestor não correlacione as duas janelas, pois a faixa de observação é de apenas uma linha.

Em meios onde a variedade de formas é elevada mas toda muito semelhante, uma outra janela de observação pode ser considerada homóloga por ser a que devolve um resultado mais semelhante mesmo não estando na zona perto da verdadeira janela homóloga, o que explica a elevada variedade de cores de pixels em zonas pertencente ao mesmo plano de distância.

5.1.2 - Implementação e teste da segunda versão

Uma primeira tentativa de redução deste índice consiste na integração do resultado sobre as 8 linhas, isto é, o cálculo da disparidade de um ponto passa a ser o resultado da análise dos 65 pixels circundantes, tal como ilustra a Figura 5.7, onde o pixel colorido é o pixel para o qual se devolve o resultado.

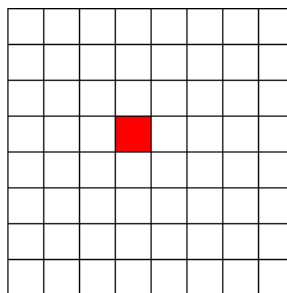


Figura 5.7 - Janela 8x8.

Para implementar este sistema é necessário alterar o módulo Gestor, no entanto sendo as alterações de baixo nível não faz sentido estar a repetir a totalidade do módulo, assim sendo apenas são mencionadas as alterações feitas ao módulo existente.

Uma primeira abordagem consiste na soma das 8 linhas num só resultado. Tendo um só resultado também apenas existe um registo para comparar com o melhor resultado e um só registo para guardar o valor da respectiva distância e um único buffer de resultados. Por fim, a cada quarteto de janelas padrão apenas se processa um pedido de escrita em vez dos oito anteriores. No que toca ao varrimento das imagens, o salto para uma nova banda de observação é apenas de uma linha em vez das oito da primeira versão. Pois a análise dos 64 *pixels* apenas devolve o resultado para um só pixel e não para um pixel em cada uma das oito linhas. Notar que está versão vai multiplicar sensivelmente por 8 o número de ciclos de relógio gastos para o processamento de um par de imagens, pois obriga a geração de (H-7) varrimentos da imagem, sendo H a dimensão vertical da imagem. O resto do módulo é mantido constante. A Figura 5.8 ilustra o varrimento desta segunda versão.



Figura 5.8 - Segunda versão do cálculo da disparidade.

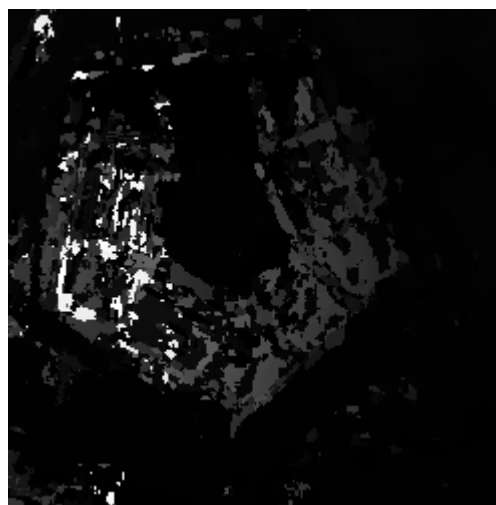


Figura 5.9 - Resultado das disparidades - versão 2.

O resultado obtido está representado na Figura 5.9. É notório a melhoria obtida a nível de ruído da imagem. Os planos estão bem mais definidos e os valores estão claramente mais estáveis. Não surgem pixels isolados brancos, sinónimo de ruído. Em contra partida foi severamente filtrada muita informação sobre a imagem, pois sobressai da imagem sobretudo o plano do pentágono e algumas zonas circundantes, embora estas últimas não apresentem formas muito distintas.

Notar na imagem resultado, a zona superior do pentágono indica que as disparidades encontradas são menores. Este fenómeno explica-se por não se ter na realidade imagens epipolares. Sendo a natureza do algoritmo de analisar a variação da imagem sobre o eixo horizontal se ocorre uma variação da imagem sobre o eixo vertical faz o subir o pentágono, o que faz com que a porção de tecto diminuída devido a disparidade horizontal seja aumentada. Assim sendo a janela que mais se assemelha ocorre nas correspondências iniciais, pois as janelas seguintes possuem uma porção do tecto do pentágono cada vez maior, o que leva ao abaixamento do nível de disparidade.

Capítulo 6

Conclusões e trabalhos futuros

Finda a apresentação do trabalho, resta tirar conclusões sobre o trabalho desenvolvido.

Os resultados obtidos mostram o correcto funcionamento do algoritmo implementado, no entanto, como se pode ver, a segunda versão produz um melhor resultado final, pois cancela as numerosas falsas correspondências encontradas ao nível do solo da imagem.

O desenvolvimento dos módulos e das suas implementações práticas resultou em implementações funcionais que, devido à simplicidade do método aplicado, não são os mais nítidos. Por falta de tempo durante a realização deste trabalho não foi possível implementar mais alternativas a fim de serem testadas e validadas a sua eficiência prática. Este trabalho possui duas vertentes, que são a eficiência das correspondências sobre os pares de imagem e o nível de resposta em tempo real, isto é, o tempo de processamento. As melhorias propostas são então divididas nestes dois grupos.

O trabalho desenvolvido concentrou-se sobretudo na apresentação de resultados deixando para mais tarde o aprofundamento o problema do tempo de processamento. Observa-se que o algoritmo desenvolvido acaba por ler várias vezes da SDRAM os mesmos valores da imagem esquerda (imagem que não é a de referência). Isto acontece sempre que se passa para uma nova janela padrão, em que parte das janelas candidatas é comum. Aceder a memória acarreta ciclos de espera por parte do módulo de processamento. Uma forma de minimizar esta espera passaria pela utilização das *blockRAM* para armazenar a banda de observação. Existiria o carregamento para a *blockRAM* da parte em falta, mas os restantes pixels já estariam presentes, permitindo assim que os valores estivessem disponíveis para o módulo à cadência do relógio principal. Uma análise qualitativa pode ser feita pensando que para um nível de correspondência possível de maxDisp pixels é necessário carregar de novo por cada linha maxDisp pixels . Esta nova versão, no pior dos casos, necessitaria de aceder a memória uma vez apenas de 4 em 4 janelas de referência sucessivas.

Uma segunda perspectiva de melhoramento do tempo de resposta pode passar pela abordagem seguinte. O sistema desenvolvido funciona de uma forma muito autónoma entre blocos, isto é, um bloco não depende do tempo que o outro demora a processar o pedido, pois aguarda que este lhe indique que tenha terminado a sua tarefa. Para tal, utiliza-se neste tipo de arquitectura o conceito de passagem do testemunho. É um sinal que dá a informação

aos módulos exteriores sobre o estado do processo. No caso deste trabalho, com esse tipo de funcionamento, o módulo em espera não se encontra a fornecer trabalho útil, o que representa um desperdício de recursos. Uma forma de melhorar o sistema consistiria em ter 4 zonas de memória para o armazenamento de imagens, divididas em pares. Não esquecer uma quinta zona para a colocação dos resultados. Assim sendo, o módulo limitante, módulo Gestor, seria melhor aproveitado, pois ele estaria a processar um par de imagens, enquanto o módulo interfaceVideo colocasse na segunda zona um novo par de imagens. Sendo o módulo Gestor muito mais demorado, esse nunca estaria inactivo. O tempo poupado seria sensivelmente o tempo do varrimento de uma imagem, pois é o intervalo de tempo em que o módulo Gestor se encontra inactivo. De referir que a gestão de acesso à memória SDRAM iria ser mais complexa, pois com esse tipo arquitectura passaríamos a escrever e ler ao dobro da cadência em certos momentos, isto é, quando ambos os módulos se encontram activos. No entanto esta alternativa é de certeza viável se for aplicada a versão anterior, onde se tira partido das *blockRAMs*, que nesta versão o acesso a SDRAM é fortemente aliviado. Com esta versão pode ser útil a criação de buffers de armazenamento para garantir um acesso por rajadas, optimizando o tempo de acesso.

Passando agora para uma perspectiva de melhoramento dos resultados; como foi afirmado neste trabalho, assumir que as imagens provêm da filmagem epipolar não é um pressuposto trivial. Calibrar as objectivas para esta configuração está sujeita a uma afinação muito delicada e pressupõe que as objectivas estejam fortemente fixas e imunes a deslizamento sobre o suporte. Uma solução mais dinâmica e robusta passaria por um botão de pressão onde as câmaras capturariam um par de imagem e se aplicasse um algoritmo exaustivo de correspondência sobre toda a imagem, que poderia demorar até vários segundos, e devolveria o valor do desfasamento vertical, *offset* entre linhas. Com base nesse valor o sistema passaria a fazer correspondência sobre as reais linhas epipolares e já não sobre a linha com a mesma coordenada vertical *Y*. O método a utilizar fica em aberto para trabalhos futuros.

Por uma questão de tempo, a detecção de contornos não foi mais abordada neste trabalho. No entanto tendo sido dada uma introdução ao algoritmo em questão, conta-se desenvolver uma metodologia para a sua implementação. Este método pode mesmo substituir a implementação da detecção do desfasamento vertical, pois como foi indicado durante a apresentação, no capítulo 3, a detecção do contorno poder ser feita no nosso caso apenas para contornos horizontais ou então também verticais, facilitando assim o processamento dos dados para a correspondência. É deste modo viável o cálculo do desfasamento vertical para determinação do *offset* vertical entre as imagens já mencionado.

Conclui-se este trabalho esperando que com este documento se tenha conseguido confrontar o leitor com a realidade das dificuldades inerentes à geração de mapas de disparidade em tempo real. Deixa-se uma porta aberta para a continuação da investigação nesta área, no sentido de encontrar métodos mais cada vez mais completos, robustos e rápidos, isto é, mais eficientes.

Referências

1. Reflexão especular. *Wikipedia*. [Online] 15 de Março de 2008. [Citação: 20 de Maio de 2008.] http://en.wikipedia.org/wiki/Specular_reflection.
2. F. M. Nogueira, C. L. Tozzi. *Geração automática de mapas de disparidade em visão estéreo*. s.l. : UNICAMP-FEEC-DAC, 1998.
3. ASIC. *Wikipedia*. [Online] 15 de Março de 2008. [Citação: 20 de Maio de 2008.] http://en.wikipedia.org/wiki/Application-specific_integrated_circuit.
4. RISC. *Wikipedia*. [Online] 11 de Março de 2008. [Citação: 20 de Maio de 2008.] <http://pt.wikipedia.org/wiki/Risc>.
5. xilinx. *Xilinx Spartan 3 Development Kit - User's Guide*. [pdf] s.l. : Xilinx, 2004.
6. FPGA. *Wikipedia*. [Online] 11 de Março de 2008. [Citação: 20 de Maio de 2008.] <http://pt.wikipedia.org/wiki/Fpga>.
7. ISE. *Xilinx*. [Online] 2008. [Citação: 01 de Junho de 2008.] http://www.xilinx.com/ise/logic_design_prod/webpack.htm.
8. JTAG. *Wikipedia*. [Online] 04 de Maio de 2008. [Citação: 01 de Junho de 2008.] <http://en.wikipedia.org/wiki/JTAG>.
9. Impact. *Xilinx*. [Online] 2008. [Citação: 01 de Junho de 2008.] http://www.xilinx.com/products/design_tools/logic_design/design_entry/impact.htm.
10. Edge detection. *Wikipédia*. [Online] 13 de Maio de 2008. [Citação: 20 de Maio de 2008.] http://en.wikipedia.org/wiki/Edge_detection#Differential_edge_detection.
11. Detection de contours. *Wikipedia*. [Online] 20 de Junho de 2008. [Citação: 20 de Maio de 2008.] http://fr.wikipedia.org/wiki/D%C3%A9tection_de_contours.
12. Alves, João P.C. *Construção de uma estrutura para processamento de imagem estéreo*. s.l. : FEUP - DEEC, 2006.
13. CYPRESS Semiconductor Corporation. 512K x 32 Static RAM CY7C1062AV33. 2003.
14. Antunes, Marc. Cálculo de mapas de profundidade para imagens estéreo em tempo real usando FPGAs. *Google Pages*. [Online] 20 de Junho de 2008. <http://mpantunes.googlepages.com/dissertacao>.
15. Experiments in stereo vision. *wikidot*. [Online] <http://disparity.wikidot.com/>.
16. Poskanzer, Jeffrey A. PGM. *Netpbm format*. [Online] Wikipedia, 27 de Junho de 2008. http://en.wikipedia.org/wiki/Portable_Gray_Map.
17. Reid B. Poter, Neil W. Bergmann. *A Generic Implementation Framework For FPGA Based*. Queensland University of Technology : s.n., 1997.
18. Verilog. *Wikipedia*. [Online] 26 de Maio de 2008. [Citação: 01 de Junho de 2008.] <http://en.wikipedia.org/wiki/Verilog>.
19. Xilinx. *Xilinx*. [Online] 2008. [Citação: 01 de Junho de 2008.] <http://www.xilinx.com/>.
20. PGM Format Specifications. *Netpbm*. [Online] Jef Poskanzer, 1991. <http://netpbm.sourceforge.net/doc/pgm.html>.