ARQUITETURA PARA ESTIMAÇÃO DE MOVIMENTO DO PADRÃO H.264/AVC COM ALGORITMO *DIAMOND SEARCH* E PRECISÃO DE *QUARTER-PIXEL*

CORRÊA, Marcel[1]; SANCHEZ, Gustavo[1]; PORTO, Marcelo[2]; AGOSTINI, Luciano[3]

[1]Universidade Federal de Pelotas; Curso de Ciência da Computação.
e-mail: mmcorrea@inf.ufpel.edu.br; gfsanchez@inf.ufpel.edu.br
[2]Universidade Federal de Pelotas; Centro de Artes.
e-mail: porto@inf.ufpel.edu.br
[3]Universidade Federal de Pelotas; Centro de Desenvolvimento Tecnológico.
e-mail: agostini@inf.ufpel.edu.br

1 INTRODUÇÃO

A estimação de movimento (EM) é a etapa de maior complexidade computacional entre todos os passos da compressão de vídeos digitais, mas também é a etapa que traz os maiores ganhos de compressão. A EM procura em quadros já processados a maior similaridade com o bloco que está sendo codificado. Quando a maior similaridade é encontrada, é gerado um vetor para indicar esta posição (RICHARDSON, 2003), que é chamado de vetor de movimento. Assim, a EM busca reduzir as redundâncias de informações presentes em quadros vizinhos de um vídeo.

A EM usa um algoritmo de busca que define como a similaridade será encontrada dentro dos quadros já processados, além de usar alguma métrica de similaridade para comparar os blocos. A métrica de similaridade mais usada na área é a soma das diferenças absolutas (*Sum of Absolute Differences* – SAD) (RICHARDSON, 2003) e esta métrica será usada neste trabalho. Já em relação aos algoritmos de busca, existem muitas opções na literatura. O algoritmo de Busca Completa (*Full Search* – FS) gera sempre os melhores resultados, por comparar todos os blocos possíveis (RICHARDSON, 2003). Como ponto negativo está o excessivo número de cálculos realizados. Para minimizar este problema, heurísticas de busca foram propostas, gerando diferentes algoritmos chamados de algoritmos rápidos. Estes algoritmos possuem uma complexidade muito inferior ao FS, mas com algumas perdas de qualidade. Neste trabalho, um destes algoritmos será usado e este algoritmo é chamado de Busca em Diamante (*Diamond Search* – DS) (THAM, 1998), que usa uma heurística iterativa que aplica um padrão de diamante sob o bloco central e mais oito blocos ao seu redor. O DS é um algoritmo com resultados de qualidade muito bons e complexidade muito menor que o FS.

Em geral a EM é realizada para fazer uma busca de pixels inteiros, no entanto, o padrão H.264/AVC (JVT, 2003) permite o uso de uma busca fracionária com precisão de até um quarto de pixel (*quarter-pixel*). Isto aumenta ainda mais a complexidade computacional do processo de compressão, por outro lado, permite um grande aumento na taxa de compressão obtida.

Para gerar os pixels fracionários, é necessário interpolar a área ao redor dos pixels inteiros, gerando assim, novos pixels de precisão meio pixel (*half-pixel*). Então, ocorre uma nova etapa de interpolação usando os pixels inteiros e os *half-pixels* e gerando amostras com precisão de *quarter-pixel* (RICHARDSON, 2003).

2 METODOLOGIA (MATERIAL E MÉTODOS)

A arquitetura de hardware desenvolvida neste trabalho foi dividida em três módulos individuais mostrados na Figura 1. O primeiro módulo é responsável por fazer uma busca usando pixels inteiros e implementa o algoritmo DS e hardware. O segundo módulo é responsável por fazer a interpolação e a busca com a precisão *half-pixel.* O terceiro módulo realiza a interpolação e a busca com precisão de *quarter-pixel*. No final do processo, um vetor de movimento com precisão de *quarter-pixel* é gerado na saída da arquitetura.

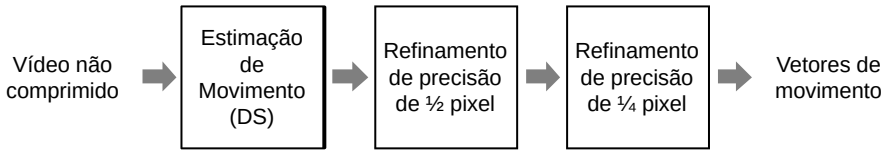

Figura 1 – Módulos desenvolvidos para EM fracionária.

Após a busca ser realizada, é gerado um vetor apontando para a posição de precisão de pixel inteiro. A arquitetura também gera a área ao redor da melhor posição encontrada para que se realize a interpolação e a busca sobre posições fracionárias apenas ao redor da melhor posição inteira escolhida com o DS.

O objetivo da interpolação de *half-pixel* e de *quarter-pixel* é permitir que o vetor gerado pela EM possa apontar para posições fracionárias e isso reflete em um aumento da precisão da EM. A precisão de um quarto de pixel é especialmente importante para vídeos de alta resolução e alta movimentação, onde o movimento entre dois quadros não pode ser mapeado com a qualidade desejada apenas usando as posições inteiras.

A interpolação de *half-pixels* utiliza a área escolhida pela busca sobre pixels inteiros para gerar os *half-pixels*. Existem três tipos de *half-pixel*, que estão apresentados na Figura 2 (a): os horizontais (x), que são calculados com base nos pixels inteiros alinhados horizontalmente; os verticais (h), que são calculados com base nos pixels inteiros alinhados verticalmente, e os diagonais (j), que são calculados com base nos *half-pixels* horizontais ou verticais previamente calculados.

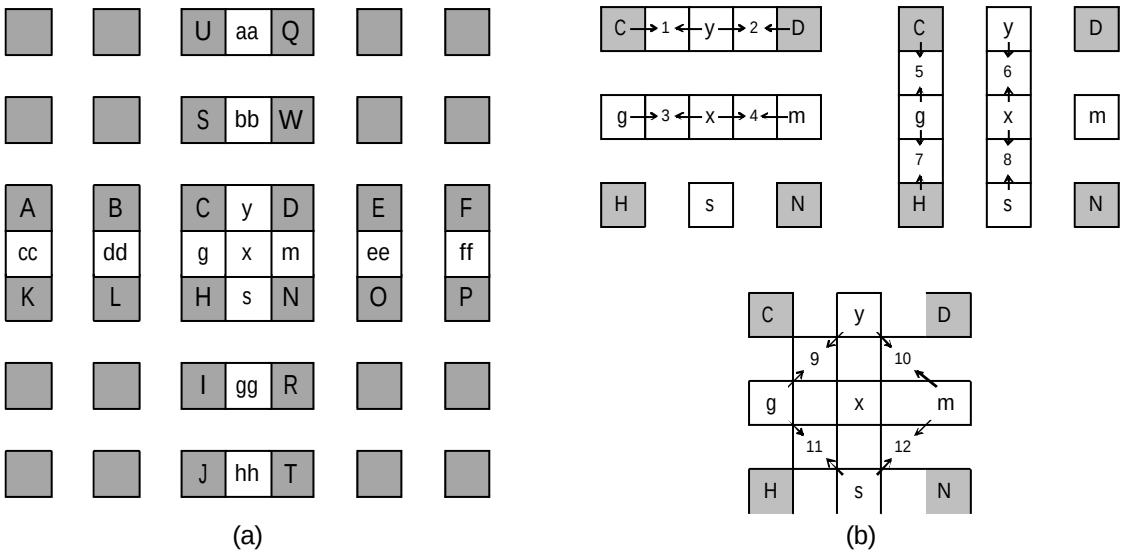

Figura 2 – (a) Posições de *half-pixel* (em branco) entre posições inteiras (em cinza) e (b) posições de *quarter-pixel* (numeradas) entre posições inteiras e de *half-pixel*.

O processo do cálculo dos *half-pixels* é composto por duas etapas, a etapa do cálculo e a etapa de arredondamento. O cálculo realizado na primeira etapa em todos os tipos de *half-pixel* está apresentado em (1). A equação apresentada em (2) mostra o cálculo de arredondamento aplicado aos *half-pixels* horizontais e verticais, enquanto que a equação apresentada em (3) mostra o cálculo de arredondamento aplicado aos *half-pixels* diagonais.

$$y_1 = A - 5B + 20C + 20D - 5E + F \quad (1)$$

$$y = Clip_{0\text{-}255}[(y_1 + 16) >> 5] \quad (2)$$

$$y = Clip_{0\text{-}255}[(y_1 + 512) >> 10] \quad (3)$$

Após a interpolação dos *half-pixels* ocorre a etapa de refinamento que irá procurar, entre os *half-pixels* gerados, se existe um melhor casamento do que o encontrado entre os pixels inteiros. O vetor de melhor casamento é passado para o bloco de interpolação de *quarter-pixel*, juntamente com uma área de busca para a realização da próxima etapa de interpolação.

O processo de interpolação de *quarter-pixel* utiliza pixels inteiros e *half-pixels* para realizar o cálculo das amostras de *quarter-pixel*. Existem três tipos de *quarter-pixel*, a Figura 2 (b) mostra as amostras que são utilizadas no cálculo e cada um dos tipos sendo: (a) horizontal, (b) vertical e (c) diagonal.

O cálculo de todos os tipos de *quarter-pixel* é realizado através equação (4). Em seguida, é realizada a busca nos *quarter-pixels* calculados, comparando os seus resultados com o melhor vetor gerado pela interpolação de *half-pixel*. Como resultado é gerado o melhor vetor dentre as três buscas, que é o resultado final da EM fracionária.

$$w = (A + B + 1) >> 1 \quad (4)$$

A arquitetura foi desenvolvida usando a técnica de *pipeline*, para ampliar seu desempenho. Foram usados dois estágios de *pipeline* na arquitetura completa. Além disso, o uso de recursos de hardware foi minimizado através de um processo de desenvolvimento arquitetural detalhado. A arquitetura desenvolvida para o DS está apresentada na Figura 3 (a), enquanto que a arquitetura desenvolvida para o refinamento de *half-pixel* e *quarter-pixel* está apresentada na Figura 3 (b).

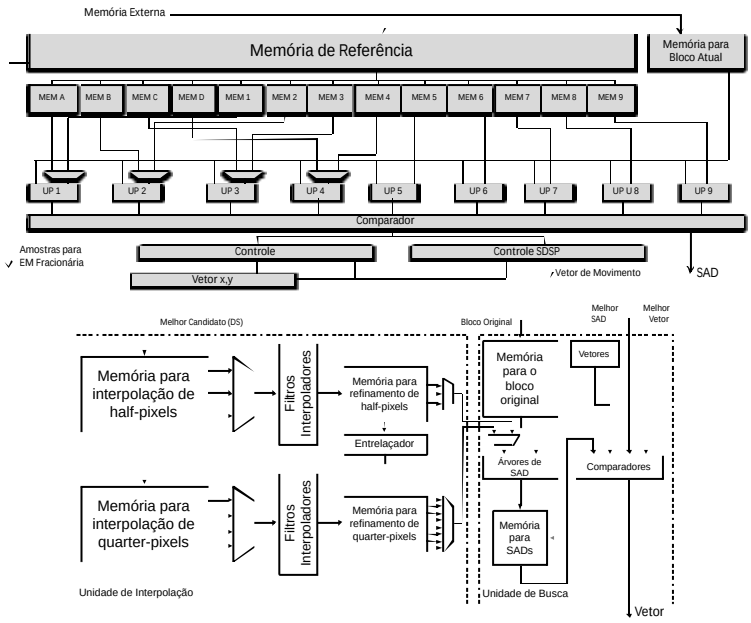

Figura 3 – (a) Arquitetura para o DS e (b) Arquitetura para o refinamento de *half-pixel* e *quarter-pixel*.

3 RESULTADOS E DISCUSSÃO

Três arquiteturas de hardware foram desenvolvidas neste trabalho: uma para o algoritmo DS em posições inteiras, uma para o refinamento de *half-pixel* e outra para o refinamento de *quarter-pixel*. As arquiteturas foram descritas em hardware com a linguagem VHDL e validadas individualmente. Em seguida, estas três arquiteturas foram integradas para gerar a EM fracionária completa seguindo o padrão H.264/AVC de compressão de vídeo.

Quando sintetizada para um dispositivo FPGA Stratix III ep3s150f780c2 da Altera (ALTERA, 2011) a arquitetura atingiu uma frequência de operação máxima de 261 MHz. Além disso, a arquitetura completa consumiu 6.697 ALUTs (18%), 10.192 registradores (27%) e 3.200 bits de memória (<1%) do dispositivo alvo.

O consumo de recursos do FPGA alvo não foi muito elevado, já que a EM é o módulo mais complexo em um codificador de vídeo. Este baixo consumo de recursos é função do uso de um algoritmo rápido (DS) na base da EM e da estratégia de desenvolvimento das arquiteturas com o maior grau possível de otimização de recursos.

A frequência de operação atingida permite que a arquitetura processe, no pior caso, 43 quadros por segundo em uma resolução Full HD (1920x1080 pixels) ou 96 quadros por segundo em uma resolução HD (1280x720 pixels). Como 30 quadros por segundo são suficientes para dar a impressão de movimento perfeito, a arquitetura é capaz de processar com folga vídeos nestas resoluções em tempo real.

4 CONCLUSÃO

Este trabalho apresentou uma arquitetura em hardware para a EM fracionária do padrão H.264/AVC, que é o estado da arte em compressão de vídeo atualmente no mercado. Esta arquitetura implementa o algoritmo de busca rápido DS para as posições inteiras e implementa o refinamento com precisão de *half-pixel* e de *quarter-pixel*.

Esta arquitetura apresenta resultados bastante satisfatórios em relação ao baixo consumo de recursos de hardware e ao alto desempenho atingido. Com ela, é possível atender a necessidade da codificação de vídeo em alta resolução em tempo real. Como o projeto foi desenvolvido em hardware, esta solução está apta para ser inserida em diversos sistemas embarcados, incluindo dispositivos móveis como celulares e câmeras de vídeos.

5 REFERÊNCIAS

RICHARDSON, I. **H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia**. Chichester: John Wiley and Sons, 2003.

JOINT VIDEO TEAM OF ITU-T AND ISO/IEC JTC 1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 or ISO/IEC 14496-10 AVC), 2003.

Tham, J; Ranganath, S; Ranganath, M; Kassim, A; **"A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation"**, **IEEE Transactions on Circuits and Systems for Video Technology**, 1998.

ALTERA: FPGA CPLD and ASIC from Altera. Disponível em: <http://www.altera.com> Acesso em: 17 ago. 2011.