HIERARQUIA DE MEMÓRIA BASEADA EM CACHE PARA OS MÓDULOS DE ESTIMAÇÃO DE MOVIMENTO E DE DISPARIDADE DA CODIFICAÇÃO DE VÍDEOS DE MÚLTIPLAS VISTAS

***SAMPAIO**, Felipe*
Grupo de Arquiteturas e Circuitos Integrados
Universidade Federal de Pelotas

***ZATT**, Bruno*
Universidade Federal do Rio Grande do Sul
Karlsruhe Institute of Technology, Alemanha

***AGOSTINI**, Luciano*
Grupo de Arquiteturas e Circuitos Integrados
Universidade Federal de Pelotas

1 INTRODUÇÃO

Nos últimos anos, houve grande desenvolvimento de aplicações que proporcionam sensações de realismo aos usuários, como a Televisão 3D (3DTV) e a Televisão com Múltiplos Pontos de Vista (FVT – *Free Viewpoint Television*) (MERKLE, 2007). Na 3DTV, o espectador se sente como se estivesse dentro da cena exibida com grande sensação de profundidade dos objetos. A FVT proporciona interatividade ao usuário de modo que ele possa escolher de qual ponto de vista ele deseja visualizar a cena.

A 3DTV e a FVT utilizam vídeos com mais de um ponto de observação como base. Estes vídeos, chamados de vídeos de múltiplas vistas (MVV – *Multiview Videos*), são obtidos a partir de duas ou mais câmeras dispostas em diferentes pontos de observação diante de uma dada cena 3D.

Com o desenvolvimento tecnológico atual, as grandes indústrias passaram a ter capacidade de projetar tais aplicações de alto realismo. Este fato abriu um campo de imensa pesquisa em formas de armazenamento e transmissão destes vídeos de múltiplas vistas. A idéia é a redução da representação destes vídeos, visto que vídeos sem compressão são praticamente intratáveis devido ao elevadíssimo volume de dados que devem ser tratados. Deste modo, a compressão de vídeo se torna essencial neste sistema de armazenamento e transmissão.

Em um cenário simples, cada uma das vistas do vídeo pode ser comprimida por um compressor de padrões conhecidos, como MPEG-2 ou H.264/AVC (RICHARDSON, 2003). Assim, todas as redundâncias existentes em cada uma das vistas serão tratadas de forma independente. No final, o vídeo comprimido, chamado de *bitstream*, será a concatenação das saídas de todas as compressões de cada vista independente.

Entretanto, este cenário despreza as redundâncias de informação existentes entre as vistas do vídeo. Dependendo do vídeo, as redundâncias entre vistas podem chegar a representar 30% do total de redundância que pode ser explorada por compressores de vídeo (MERKLE, 2007). Neste cenário, surge a codificação de vídeo que trata de maneira conjunta as múltiplas vistas do vídeo, chamada de compressão MVC (*Multiview Video Coding*) (MERKLE, 2007).

Dentre as ferramentas de codificação, a que provê maiores ganhos em taxa de compressão é a Estimação de Movimento (ME). A ME é responsável por

encontrar o sentido e o módulo do movimento do objeto dados dois quadros de uma mesma vista. Isto é realizado através de algoritmos de busca que, guiados por alguma métrica de similaridade, visam encontrar o melhor casamento do bloco alvo da compressão em quadros já processados.

A característica inserida pela inserção de múltiplas vistas é chamada de disparidade e está relacionada com o deslocamento dos objetos entre quadros de diferentes vistas. Para a busca do melhor casamento em regiões de quadros de outras vistas, o módulo de Estimação de Disparidade (DE) (MERKLE, 2007) é utilizado. A DE é, em suma, uma extrapolação da ME que considera características específicas da compressão MVC.

Existem alguns desafios que aparecem com o desenvolvimento de codificadores e decodificadores MVC. Dentre eles, um dos principais é a questão de acesso à memória externa. Para tanto, é necessário o projeto de uma hierarquia de memória que (1) interaja com a memória principal do codificador e que (2) entregue dados a uma taxa suficiente, de acordo com o processamento desejado.

Este trabalho realiza uma investigação em várias configurações de *cache*, de modo a avaliar os impactos no projeto de uma Hierarquia de Memória com tais configurações no projeto de uma ME/DE visando sua utilização na codificação MVC.

Uma organização de memória baseada em *cache* tem como função armazenar os dados acessados da memória principal localmente, de modo que, quando acessados novamente, tais dados já estejam disponíveis nesta memória interna. Esta característica é referida como localidade temporal. Além disso, a *cache* também pode explorar a localidade espacial, onde além dos dados que se necessita no momento, a *cache* pode armazenar dados adicionais que muito provavelmente possam ser acessados em um futuro próximo. A idéia é reduzir a largura de banda de comunicação entre a aplicação e a memória principal, tendo em vista que um acesso à memória externa é ordens de grandeza mais lento do que um acesso à *cache* local (PATTERSON, 2005).

As seções que se seguem estão organizadas da seguinte forma: a Seção 2 apresenta a metodologia utilizada e uma descrição dos experimentos realizados, a Seção 3 discute os resultados atingidos e, por fim, a Seção 4 conclui e planeja os trabalhos futuros.

2 METODOLOGIA

Este trabalho tem como objetivo avaliar diferentes parâmetros do projeto de uma Hierarquia de Memória baseada em *cache* para o processamento dos módulos de ME/DE para a codificação MVC.

O primeiro passo foi definir uma maneira de mapear um conjunto de amostras em uma posição da memória *cache*. Após, o desafio foi definir as variáveis a serem exploradas. Tais variáveis, bem como o esquema de mapeamento utilizado, estão representados na Figura 1.

Neste esquema, um quadro de referência é divido em blocos de *cache* de dimensões *bloco_w* e *bloco_h*. De acordo com um mapeamento, como o mostrado na Figura 1, define-se o número de posições da memória. No exemplo, o mapeamento mostrado é de 3x3 blocos, gerando uma memória *cache* de 9 posições, onde os mapeamentos estão também mostrados na Figura 1. Outra variável faz referência à associatividade da *cache*, ou seja, quantos blocos serão referenciados a cada posição da memória. Este grupo de blocos acessados, dada uma determinada posição, é referido por conjunto.

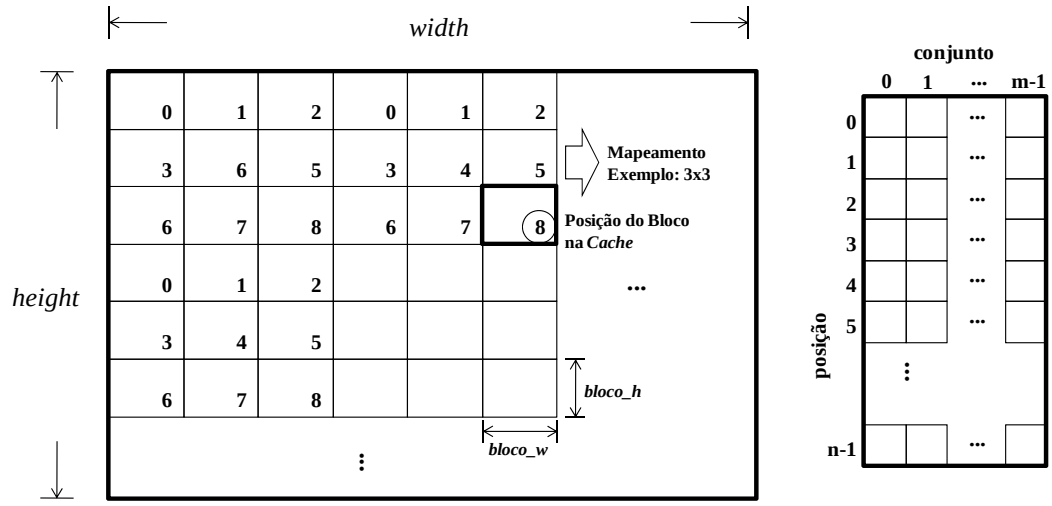

Figura 1 – Esquema de *cache* proposto neste trabalho.

Tendo como base todas as variáveis, desenvolveu-se um emulador na linguagem Python que realiza uma execução de buscas típicas da ME/DE e gera resultados de desempenho. Os resultados das execuções consideram um dos algoritmos de busca padrão do MVC, o *TZ Search*, e um limite máximo de pesquisa de ±96 amostras em cada dimensão a partir do centro de busca (JVT, 2006). Este algoritmo de busca adota um padrão de diamante, o qual expande seu alcance a cada iteração.

3 RESULTADOS E DISCUSSÕES

As Tabelas 1 e 2 trazem os resultados de largura de banda e tamanho da *cache* para vários cenários. A largura de banda considera o montante de dados que devem ser buscados da memória principal a cada vez que os dados requisitados não se encontram na *cache*, considerando o processamento de 30 quadros por segundo e vídeos de resolução 640x480 pixels (resolução padrão utilizada nos testes para a codificação MVC (JVT, 2006)). Além disso, como o foco está na codificação MVC, considerou-se a busca em dois quadros de referência, um sendo o vizinho temporal acessado pela ME e outro sendo buscado pela DE para o quadro vizinho de outra vista. Ambas as Tabelas 1 e 2 confrontam diferentes tamanho de blocos de *cache* (2x2, 3x3 e assim por diante), contra diferentes esquemas de mapeamento (20x20, 32x24 e 40x40) e contra diferentes associatividades (conjuntos de 1, 2 ou 4 posições).

Tabela 1 – Análise de Largura de Banda (Mbytes/segundo)

	20x20			**32x24**			**40x40**		
	1	**2**	**4**	**1**	**2**	**4**	**1**	**2**	**4**
2x2	23.836	16.523	10.474	21.655	12.640	5.424	26.430	10.510	1.904
3x3	16.230	9.779	3.491	11.601	3.404	1.370	10.112	1.601	1.429
4x4	10.838	3.401	1.354	5.795	1.395	1.312	2.147	1.428	351
5x5	7.031	1.525	1.304	3.042	1.313	721	1.428	1.359	**142**
6x6	4.091	1.312	1.303	1.664	1.312	258	1.423	1.047	140
7x7	1.976	1.303	965	1.318	1.171	**143**	1.422	**583**	140
8x8	1.328	1.303	172	1.307	806	**140**	**1.**421	140	140
9x9	1.303	1.274	142	1.305	472	140	**1.260**	140	140
10x10	1.303	1.260	140	1.303	**140**	140	**1.097**	140	140

Tabela 2 – Análise do Tamanho da *Cache* (Kbytes)

	20x20			32x24			40x40		
	1	**2**	**4**	**1**	**2**	**4**	**1**	**2**	**4**
2x2	1,56	3,13	6,25	3,00	6,00	12,00	6,25	12,50	25,00
3x3	3,52	7,03	14,06	6,75	13,50	27,00	14,06	28,13	56,25
4x4	6,25	12,50	25,00	12,00	24,00	48,00	25,00	50,00	100,00
5x5	9,77	19,53	39,06	18,75	37,50	75,00	39,06	78,13	**156,25**
6x6	14,06	28,13	56,25	27,00	54,00	108,00	56,25	112,50	225,00
7x7	19,14	38,28	76,56	36,75	73,50	**147,00**	76,56	**153,13**	306,25
8x8	25,00	50,00	100,00	48,00	96,00	**192,00**	100,00	200,00	400,00
9x9	31,64	63,28	126,56	60,75	121,50	243,00	**126,56**	253,13	506,25
10x10	39,06	78,13	156,25	75,00	**150,00**	300,00	**156,25**	312,50	625,00

Como base para a análise de desempenho, os resultados atingidos no trabalho (CHEN, 2006) foram considerados. Nele, os melhores resultados em termos de tamanho da memória interna estão entre 123 e 198 Kbytes. Considerando este intervalo, as configurações que se encaixam nesta restrição estão destacadas nas Tabelas 1 e 2. Dentre eles, a menor largura de banda requerida é 140 Mbytes por segundo, valor este compatível com as taxas apresentadas em (CHEN, 2006).

4 CONCLUSÕES

Este trabalho realizou uma exploração de esquemas de Hierarquia de Memórias baseadas em *cache* visando o projeto de um módulo de Estimação de Movimento e de Disparidade para a codificação de vídeo de múltiplas vistas. Uma série de variáveis foram consideradas e exploradas nas simulações. A análise foi realizada em taxas de acertos e erros na busca pelo dado na *cache* e de aspectos arquiteturais, como largura de banda e tamanho da *cache*. Os melhores resultados requerem uma largura de banda em torno de 140 Mbytes por segundo e cerca de 150 Kbytes de memória. Tais valores atingem os requisitos para este tipo de sistema de acordo com dados da literatura.

5 REFERÊNCIAS

CHEN, Ching-Yeh, et al. Level C+ Data Reuse Scheme for Motion Estimation with Corresponding Coding Orders. **IEEE Transactions on Circuits and Systems for Video Technology.** v. 16, n. 4, p. 553-558, 2006.

JVT Team. **Common Test Conditions for Multiview Video Coding.** Doc. JVT-T207, 2006.

MERKLE, Philipp, et al. Efficient Prediction Structures for Multiview Video Coding. **IEEE Transaction on Circuits and Systems for Video Technology**. v. 17, n. 11, p. 1461-1473, 2007.

PATTERSON, David A.; HENESSY, John L. **Organização e Projeto de Computadores: a interface hardware/software.** ed. 3. São Paulo: Editora Campus/Elsevier, 2005.

RICHARDSON, Ian. **H.264 and MPEG-4 Video Compression: Video Coding for the Next-generation Multimedia.** Chichester: Wiley, 2003.