Proposta de Solução de HW/SW para o Módulo de Transformadas de um Codificador H.264/SVC

Ronaldo Husemann, Eduardo Eick, Mariano Majolo, Daniel Santos, Victor Guimarães
Av. Osvaldo Aranha, 103 – Porto Alegre – RS – Brazil
+55 5133086127
rhusemann@inf.ufrgs.br

Valter Roesler, José Valdeni de Lima, Altamiro Amadeu Susin
Av. Bento Gonçalves, 9500 – Bloco IV – S.233
Porto Alegre – RS – Brazil +55 5133086127
{roesler, valdeni}@inf.ufrgs.br

RESUMO

Este artigo propõe uma solução híbrida baseada na cooperação de hardware/software visando implementações eficientes para codificadores escaláveis padrão H.264/SVC. Soluções escaláveis geram fluxos multicamadas complexos, resultando em maiores demandas computacionais e atrasos latentes quando comparados com alternativas não escaláveis, visto que requerem mais recursos para tratar a dependência de dados entre camadas. Na solução proposta, uma placa de hardware FPGA é proposta para executar as demandas computacionais intensas, enquanto o software é responsável pela gerência das interfaces externas e controle do sistema. Particularmente, o artigo sugere módulos de hardware projetados especialmente para aumentar o desempenho de algoritmos críticos, como as transformadas DCT e Hadamard.

ABSTRACT

This paper proposes a hybrid solution of hardware/software cooperation aiming efficient scalable H.264/SVC encoder implementation. Scalable solutions generate complex multi-layer streams, resulting in additional computational demands and latency delays when compared to single-layer alternatives, due to increased amount of resources needed to handle inherent inter-layer data dependency. In the proposal, a hardware FPGA board executes the high-demanding computational algorithms, while the software is responsible for external interface management and overall system control. Besides that, the paper suggests optimized hardware modules specially designed to improve the performance of critical algorithms, like DCT and Hadamard.

Categories and Subject Descriptors

I.4.2 [**Compress (Coding)**]: Data compaction and compression.

General Terms

Algorithms, Measurement, Performance, Experimentation.

Palavras-chave

Codificação de vídeo escalável, H.264 SVC, Co-design HW/SW.

1. INTRODUÇÃO

Visando adaptabilidade para aplicações de multimídia heterogêneas (IPTV e dispositivos móveis) foi proposta em 2007 a solução emergente H264/SVC (*Scalable Video Coding*) [1].

Esta solução de codificador escalável prevê a geração de um fluxo de vídeo composto por uma camada base (responsável pelos requisitos mínimos de apresentação) e distintas camadas de enriquecimento (cada uma delas aprimorando de forma complementar as características do vídeo da camada anterior). Um codificador SVC suporta distintos tipos de escalabilidade, como SNR, espacial e temporal, que podem ser usados ou descartados dependendo dos objetivos específicos da aplicação alvo [2].

Atualmente o padrão SVC pode ser considerado como o mais eficiente *codec* de vídeo escalável [3]. O fluxo multicamadas do codificador SVC provê o recurso de decodificação seletiva, que se mostra muito adequada para aplicações que tem bandas de recepção variável ou sujeitas a falhas de comunicação. Ou seja, o decodificador de vídeo mesmo que tenha perdido (ou descartado algumas camadas) ainda poderá remontar e exibir um vídeo ao usuário, tornando sistema adaptável ao meio [1].

Infelizmente a implementação prática de uma solução escalável multicamadas demanda por uma estrutura complementar própria, que acaba impactando no aumento da complexidade e atrasos do sistema. Esta estrutura complementar do SVC foi proposta para explorar a grande interdependência de dados entre consecutivas camadas (uma solução escalável precisa considerar informações de camadas anteriores para compor uma nova) [4].

O diagrama de blocos simplificado de um codificador SVC de duas camadas é apresentado na Figura 1. Pode-se observar que a complexidade e atrasos globais de qualquer solução H.264/SVC aumenta proporcionalmente com o número de camadas, o que pode tornar impraticável uma implementação de tempo real, adotando-se abordagens baseadas unicamente em software [2]. Metodologias de Hardware/software co-design, baseadas em plataforma ou IP (IBD) [5], podem ser usadas como alternativas para auxiliar projetos em software quando aplicações de alto desempenho são requeridas. Considerando este contexto, o presente artigo propõe uma arquitetura inovadora de co-design que visa aumentar o desempenho prático de um codificador SVC pela exploração de paralelismo em nível de hardware [6].

Basicamente propõe-se o uso de uma placa FPGA dedicada, que opera cooperativamente com uma solução em software. A

English Title: **Proposal of a HW/SW Solution for the Transform Module of a H.264/SVC encoder.**

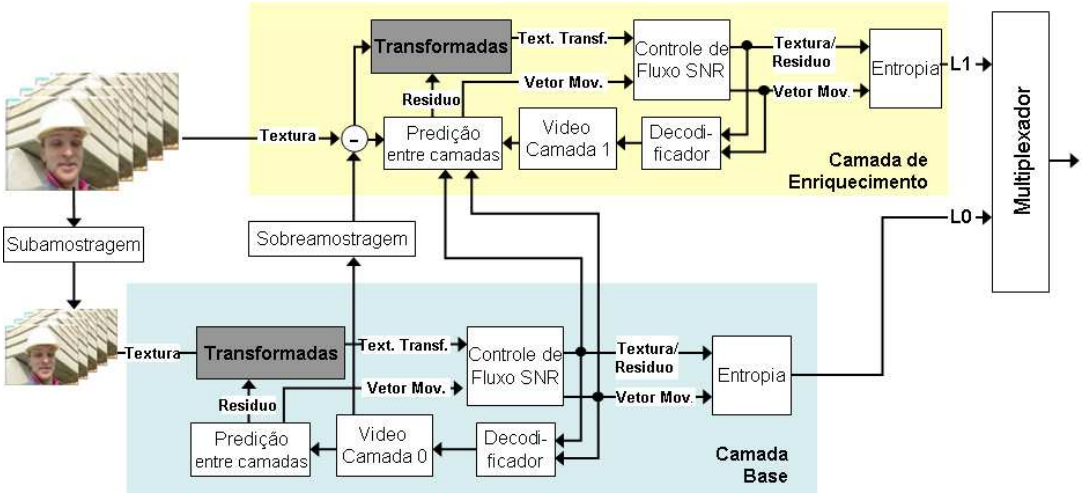

Figura 1: Diagrama de blocos de um codificador de duas camadas.

arquitetura proposta de hardware cooperativo foi desenvolvida para suportar o módulo de transformadas diretas de um codificador SVC (representado como blocos cinzas na Figura 1): que contém transformadas discretas de co-seno e Hadamard [7].

Todos os módulos propostos foram desenvolvidos em VHDL, adotando estruturas em *pipeline* otimizadas para operação paralela. Assim a proposta além de garantir alto desempenho também busca reduzir o custo total de recursos computacionais e atrasos relacionados com acessos à memória.

De forma geral, o artigo é organizado da seguinte maneira: a seção 2 apresenta a proposta da arquitetura, a seção 3 descreve os algoritmos desenvolvidos; a seção 4 mostra os resultados práticos obtidos e a seção 5 apresenta as conclusões dos autores.

2. REQUISITOS DO SISTEMA

O primeiro passo no projeto de qualquer solução baseada em co-design de hardware/software consiste em avaliar as características específicas da aplicação de software bem como os recursos de hardware disponíveis [8].

2.1 Características do software H.264/SVC

Uma aplicação baseada em software por si só depende de inúmeros fatores, que incluem a habilidade do programador e das capacidades da CPU. Entretanto a área de codificação de vídeo traz novos desafios principalmente quando o recurso de tempo real for demandado, devido ao grande número de dados processados e distintos algoritmos iterativos. Particularmente um codificador escalável no padrão SVC (Figura 1) é composto pelos algoritmos descritos a seguir.

O módulo "Transformadas" engloba os algoritmos de Transformada Discreta de Co-senos (DCT) e Hadamard (HAD), que transportam dados de pixel para o plano das freqüências, auxiliando na identificação e remoção de redundância espacial.

O módulo "Controle de Fluxo SNR" é responsável por gerenciar a taxa de bits ocupada individualmente por cada camada. Basicamente este módulo ajusta de forma adaptativa a qualidade do vídeo na saída. Pode gerar fluxos com diferentes qualidades, implementando a escalabilidade do tipo SNR [9].

O módulo "Decodificador" realiza a reconstrução do vídeo usando apenas informações codificadas. Este procedimento é necessário para se obter no codificador as mesmas imagens que estarão presentes no decodificador para correção de erros.

As imagens reconstruídas ("Video Camada*") serão usadas como quadros de referência para cálculo de resíduos ou refinamento de vetores em procedimentos de estimação de movimento intra ou inter-layer, realizado pelo módulo de "Predição entre camadas" (*Inter-Layer Prediction*) [4].

"Entropia" é responsável por montar as informações escaláveis específicas de cada camada em fluxos independentes. Finalmente o "Multiplexador" agrupa todas as camadas recebidas em um fluxo único com cabeçalhos de identificações próprios do padrão SVC.

Quando a escalabilidade espacial é habilitada no codificador SVC dois módulos adicionais são usados: "Subamostragem", um módulo que reduz a resolução da imagem e "Sobreamostragem" que restaura a resolução de imagens de camadas superiores.

Modernas abordagens de software utilizam-se de programação *multi-core* para explorar paralelismo no SVC, porém ainda são limitadas à capacidade das plataformas (número de núcleos e taxa de memória). A fim de mensurar demandas computacionais reais alguns experimentos foram realizados usando-se a ferramenta GPROF no JSVM (código de referência H.264/SVC disponibilizado pela entidade JVT) [10].

Foi usada uma aplicação alvo típica de IPTV em uma solução SVC de três camadas espaciais com configuração diádica (razão igual a dois entre camadas consecutivas). Os dados obtidos apontam que o módulo "Transformadas" é o mais crítico após a "Predição Inter-layer", ocupando em média 12% do tempo total de execução.

2.2 Características do software H.264/SVC

Dispositivos programáveis do tipo *Field Programmable Gate Array* (FPGA) permitem grande flexibilidade e múltiplos níveis de paralelismo, o que torna esta tecnologia ideal para implementação de algoritmos de alto desempenho.

Entretanto devido à grande quantidade de pixels de dados que devem ser processados em uma solução de codificação de vídeo, esta tecnologia requer o uso de memórias externas. Trabalhos relacionados indicam que alguns dos mais conhecidos gargalos de um sistema de codificação de vídeo embarcado são devidos à taxa de dados da memória e o mecanismo de comunicação entre a memória e a unidade computacional [11].

Considerando isto, a arquitetura de co-design proposta deve necessariamente levar em conta restrições de memória, que são diretamente afetados pela quantidade e taxa de dados transferidos [12]. Baseados nestas demandas dois requisitos específicos de hardware são definidos: (i) largura de memória da placa e (ii) taxa de bit da interface de comunicação.

Para esta especificação, levou-se em conta os valores práticos dos requisitos relacionados com uma aplicação IPTV alvo codificada com três camadas espaciais escaláveis. A aplicação alvo escolhida precisa implementar três instâncias de codificação de vídeo em configuração diádica (4CIF, CIF e QCIF). Considerando que os dados de entrada adotam o formato 4:2:0, onde cada sub-bloco de 2x2 amostras de componente luma corresponde a uma amostra de componente de croma azul (Cb) e uma de vermelho (Cr), explicando a multiplicação por 1,5 vista a seguir. Assim as quantidades de dados geradas são as seguintes:

Camada 1 (4CIF): 640x576 = 368.640 pixels.

368640*1,5 = 552.960 amostras.

Camada 2 (CIF): 320x288 = 92.160 pixels.

92160*1,5= 138.240 amostras.

Camada 3 (QCIF): 160x144 = 23.040 pixels.

23040*1,5 = 34.560 amostras.

Analisando-se estes valores, o total para todas as camadas é 725.760 amostras. De forma, geral os valores obtidos representam os requisitos espaciais para um quadro apenas. Entretanto, codificadores de vídeo precisam considerar quadros passados para identificar redundâncias temporais (predição de movimento). A memória total exigida depende assim do número de quadros de referência. O padrão SVC em especial adota um mecanismo de predição de movimento aprimorado chamado de estrutura B hierárquica, que visa simplificar a escalabilidade do tipo temporal, diferindo do esquema tradicional H.264 IBBP. Considerando uma configuração típica com GOP=8, o padrão SVC demanda até sete quadros de referência. Assim a placa de hardware passa a ter um requisito de memória de 5 Mbytes (7 * 725.760 ≈ 5M).

Já para a definição do canal de comunicação considerou-se uma taxa de exibição de vídeo de 30 quadros por segundo e resolução de 8 bits por amostra. Assim, a quantidade máxima de dados transferidos para a placa a cada segundo é dada por:

725.760*30 = 21.772.800 amostras/s, ou

21.772.800*8 = 174.182.400 bit/s

Alguns algoritmos de codificação que precisam rodar em placas de hardware (como transformadas) expandem o número de bits gerados. Considerando o caso extremo (resolução de 16 bits [13]) a quantidade máxima de dados recebidos por segundo será:

725.760*30 = 21.772.800 amostras/s, ou

21.772.800*16= 348.364.800 bits/s

A conclusão é de que o canal de comunicação precisa suportar ao menos 523 Mbit/s (transferidos mais recebidos). A Tabela 1 resume as principais soluções técnicas disponíveis [13]:

Tabela 1: Análise dos principais canais de comunicação

	Freq. (MHz)	Largura (# de bits)	Taxa (Mbit/s)	Suporta aplicação alvo?
USB v1.1	12	1	12	Não
USB v2	480	1	480	Não
PCI v1	33	32	1066	**Sim**
PCI v2	66	64	4266	**Sim**
PCI Express	2500	1	2000 [1]	**Sim**

2.3 Solução proposta

O presente trabalho sugere o uso de uma placa dedicada, com a tecnologia de FPGA que roda de forma cooperativa com uma solução de software em um computador convencional (Figura 2).

Com base nos cálculos da última seção foi especificado que a plataforma de hardware da proposta deve ser uma placa FPGA com memória local DDR2 de 64MB e interface PCI v2 (66 MHz com barramento de 64 bits).

Na solução proposta, o software na CPU principal (computador) irá executar uma versão adaptada do código de referência. Na placa com FPGA, módulos de hardware otimizados são instalados a fim de acelerar o desempenho SVC (Figura 2).

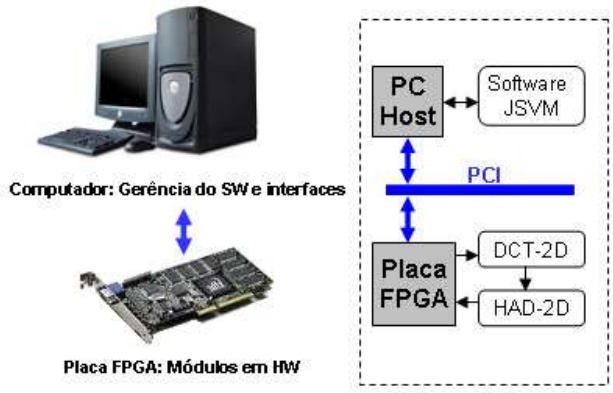

Figura 2: Solução proposta.

Assim, cada vez que um procedimento de transformada de dados (DCT e Hadamard) precisa ser realizado, o software no computador principal somente deve enviar os dados originais para

[1] PCI Express adota uma topologia CRC tipo 10b/8b com dois bits redundantes para cada byte.

a placa através do barramento PCI. A FPGA identifica esses dados e ativa os módulos de hardware correspondentes para realizar a operação requisitada. Após finalizar, os dados processados são armazenados em uma memória compartilhada para serem lidos pelo software no computador principal.

3. MÓDULOS DE HARDWARE EM FPGA

A fim de auxiliar o entendimento da proposta, as próximas subseções descrevem os módulos em hardware desenvolvidos especificamente para hardware que serão responsáveis por realizar as operações de transformadas (DCT e Hadamard).

3.1 Módulo DCT

O algoritmo de transformadas DCT, adotado pelo codificador H.264 SVC, deve ser aplicado sobre blocos de 4x4 pixels oriundos da entrada de vídeo (imagem capturada) quando se tratar de quadros tipo I ou sobre blocos de 4x4 pixels resultantes do cálculo de resíduos do mecanismo de estimativa de movimento. Este algoritmo foi especialmente desenvolvido visando facilitar sua implementação em hardware. Na prática, representa uma aproximação ortogonal da DCT original, uma vez que passa a utilizar apenas aritmética inteira sem erros de arredondamento (o algoritmo original exige notação em ponto flutuante) [14].

Genericamente a realização de uma DCT de duas dimensões (necessário ao se lidar com imagens) envolve duas operações principais: transformada direta (horizontal) e transformada transposta (vertical) [6]. Assim produz-se um mecanismo de cálculo de transformada de dois estágios, conforme identificado:

Estágio 1: $Y_1 = A.X$ (1)

Estágio 2: $Y = A.Y_1^T$ (2)

Onde:

X representa o bloco de entrada de 4x4 pixels;
A representa a matriz de transformada direta;
Y_1 representa o resultado do primeiro estágio;
Y_1^T representa a matriz Y1 transposta.

A matriz de transformada A da norma H.264 é definida como:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Figura 3: Matriz de transformada H.264 DCT-2D.

Pode-se observar que os valores das matrizes variam entre -2 e 2 em representação inteira. Esta característica permite a implementação deste algoritmo utilizando apenas operações simples como soma, subtração e deslocamento de um bit para a esquerda (multiplicação por dois) [15].

O cálculo da matriz Y_1^T é derivado do primeiro estágio do processo (equação 1). De forma geral, o algoritmo então se resume em realizar um estágio de multiplicação de matrizes, transpor os resultados obtidos e voltar a utilizar o mesmo estágio de multiplicação. Com isso pode-se facilmente implementar uma transformação DCT-2D a partir de dois estágios de transformação DCT-1D e um de transposição, gerando-se assim um algoritmo em hardware de três estágios [16].

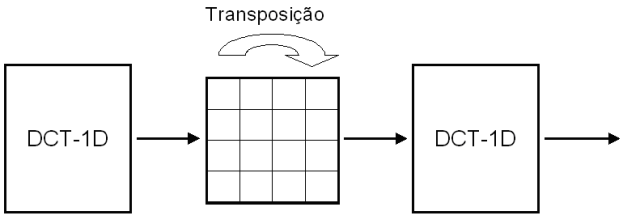

Figura 4: Abordagem de 3 estágios para realizar DCT-2D.

A abordagem adotada para a solução proposta segue o princípio apresentado nas equações (1) e (2), onde a transformação bidimensional é obtida por duas transformações unidimensionais mais simples, contribuindo para reduzir a complexidade global.

Deve-se ter em mente que a entrada do módulo será alinhada em linhas enquanto que a saída será alinhada em colunas (devido à etapa de transposição).

A fim de simplificar ainda mais a proposta a 1ª DCT incorpora internamente a funcionalidade de transposição, ou seja, recebe os dados, orientados em linhas, e entrega os mesmos, depois de transformados, orientados em colunas. Esta abordagem tende a reduzir a complexidade da implementação, pois aproveita os mesmos contadores e máquina de estados de controle para as duas operações (transformação e transposição). Desta forma o módulo de DCT-2D, conforme proposto, é composto por dois módulos, DCT-1D H (horizontal), que realiza operações de transformação e transposição e DCH-1D V (vertical), que faz apenas a operação de transformação. Um esboço desta arquitetura pode ser observado na figura abaixo.

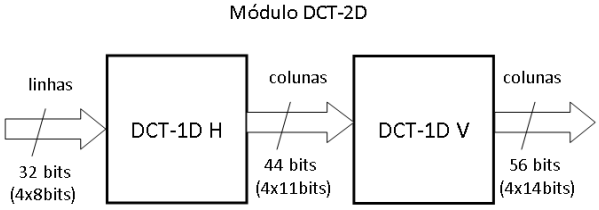

Figura 5: Abordagem proposta para DCT-2D

É importante observar que as larguras dos barramentos de entrada e saída são diferentes. Isso se deve pelo fato de que ao se realizar operações de soma e deslocamento de bits para a esquerda a resolução dos valores tende a aumentar. Em teoria o número máximo de bits acrescentados seria três para cada operação de transformada, resultando assim em saídas de 14 bits por pixel.

Cada módulo de transformada utiliza-se de uma estrutura em *pipeline* para implementar suas funcionalidades. Para executar as operações de transformação se utilizam 2 estágios, já para a transposição se utilizam de 4 estágios. Assim a DCT-1D H adota uma estrutura de *pipeline* de 6 estágios (2+4) enquanto que a DCT-1D V utiliza apenas 2 estágios de *pipeline*.

Com a abordagem proposta pode-se processar nesta estrutura até 4 amostras por ciclo de relógio, gerando saídas na mesma taxa, após transcorrer um período de latência total de 8 *clocks*.

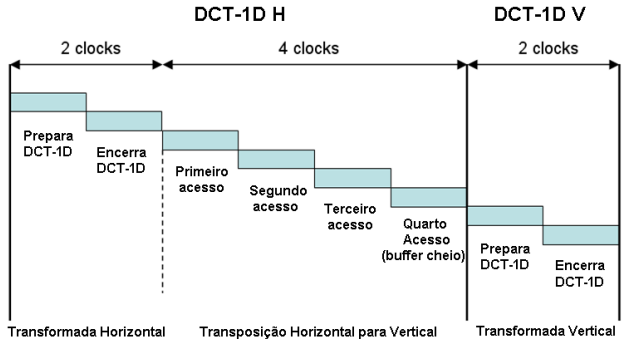

Figura 6. Descrição dos estágios internos da proposta.

Outro aspecto importante observado durante o desenvolvimento dos módulos em hardware foi a largura de pixels do canal de comunicação previsto para a placa de desenvolvimento. [13].

No caso de barramentos de 32 bits cada acesso representa um vetor de quatro amostras consecutiva de uma mesma linha (4 x 8 bits = 32 bits). Porém quando utilizando-se barramentos de 64 bits (caso da interface especificada) cada acesso contém 8 pixels consecutivos.

Existe uma forte dependência de dados entre linhas consecutivas de pixels, quando se realiza o cálculo de uma transformada bidimensional. A fim de aumentar o desempenho da solução em hardware foi desenvolvida uma arquitetura paralela capaz de processar dois blocos de pixels simultaneamente. Por estarem operando com blocos de pixels distintos elimina-se a dependência de dados anteriormente citada, garantindo-se o dobro do desempenho de uma solução de 32 bits.

Nesta proposta, após concluir o processamento da DCT-2D gera-se analogamente um vetor de 112 bits correspondente a oito valores de 14 bits processados na mesma representação utilizada na entrada, relacionada com dois blocos paralelos. Ou seja, das oito amostras geradas na saída do módulo, os quatro valores mais significativos (4x14=56 bits) representam uma coluna de valores processados do bloco superior e quatro restantes (4x14=56 bits) representam os valores processados do bloco inferior. O detalhamento do funcionamento interno do módulo de transformada completo está apresentado na Figura 7. Basicamente seus principais sinais implementados são:

- DCT_lin_val: linha de valores de entrada, simultaneamente referenciando-se a dois blocos de dados. Cada amostra é representada por 8 bits;
- DCT_lin_num: identificador do número da linha que está sendo informado na porta anterior (DCT_lin_val);
- DCT_col_val: linha de valores gerados pelo processamento do módulo. Cada valor de saída é representado por 14 bits.
- DCT_col_num: identificador de qual linha de saída está sendo gerada na saída DCT_col_val.

3.2 Módulo Hadamard

O modulo Hadamard opera sobre os dados resultantes da DCT, ou mais especificamente, sobre os coeficientes resultantes DC pelo módulo DCT-2D. Este mecanismo exige, portanto, a montagem dos blocos de entrada de Hadamard de acordo com a norma H.264 assim como é descrito na Figura 8. Os blocos em cinza representam os blocos que serão processados pelo módulo Hadamard, enquanto que os brancos serão descartados. No lado esquerdo se tem o mecanismo para pixel luma e na direita o mecanismo para croma [17].

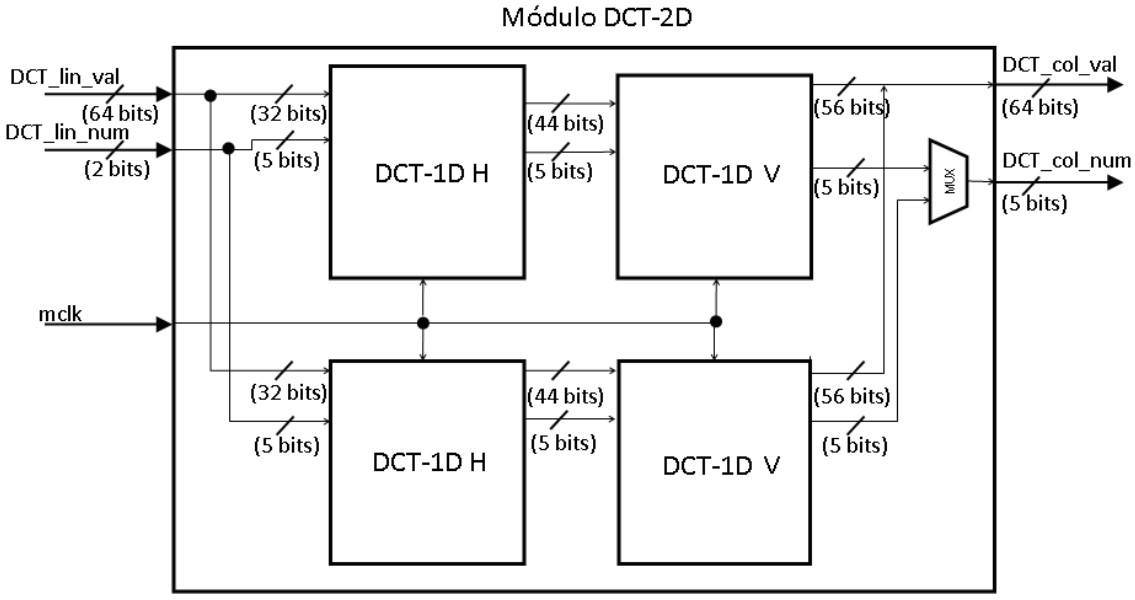

Figura 7: Arquitetura completa da implementação DCT-2D

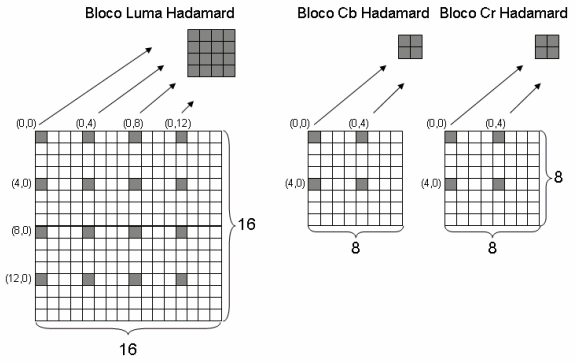

Figura 8. Esquema de montagem dos blocos Hadamard.

A fim de implementar esse mecanismo de montagem de blocos de entrada, o módulo Hadamard proposto precisou incorporar um buffer de memória interna de macroblocos de 256 amostras (chamado MB Buffer), suportando até 16 blocos de 4x4 pixels (pior caso). Essa memória trabalha como uma DP-RAM (*DualPort RAM*), permitindo escritas e leituras simultâneas, o que reduz a latência do módulo. Pelos processamentos de luma e croma ocorrem em momentos distintos, a mesma memória pode ser utilizada para ambos. Na prática, a transformada Hadamard é similar a outras transformadas (ex. DCT). Assim ela pode também ser implementada adotando-se uma abordagem de dois estágios:

Estágio 1: $Z_1 = H.Y$ (3)

Estágio 2: $Z = H.Z_1^T/2$ (4)

Onde:

H representa a matriz de transformada direta;

Y representa o bloco de entrada de 4x4 pixels;

Z_1 representa o resultado do primeiro estágio;

Z_1^T representa a matriz Z_1 transposta.

Observando a Figura 8 pode-se notar que o módulo de Hadamard precisa suportar dois tipos de blocos de entrada: blocos 4x4 (luma) e blocos 2x2 (croma). As matrizes H para ambas as situações são definidas como ilustrado na Figura 9 [18]:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

(a) (b)

Figura 9: Matrizes H para (a) luma e (b) croma.

Como já comentado, a implementação proposta para uma Hadamard bidimensional completa (HAD-2D) tem uma arquitetura bastante similar à utilizada pelo módulo DCT, nesse caso representada pelos módulos HAD-1D H e HAD-1D V. Particularmente cada amostra de entrada tem resolução de 14 bits, mas, após processadas, serão expandidas para uma resolução de 16 bits. Após serem realizados os cálculos completos da transformada Hadamard, um módulo final (chamado Output Composer) é responsável por montar os vetores de saída, a partir de dados oriundos ora do buffer original (MB Buffer) ora do buffer de dados processados (HAD Buffer). Considerando isso, os caminhos internos tiveram que ser ajustados para permitir uma saída de 128 bits (Figura 10).

De forma geral os principais sinais implementados para essa solução são:

- HAD_lin_val: linha de valores de entrada, simultaneamente referenciando-se a dois blocos de dados. Cada amostra de entrada é representada por 14 bits;
- HAD_lin_num: identificador do número da linha que está sendo informado na porta anterior (DCT_lin_val);
- HAD_col_val: linha de valores gerados pelo processamento do módulo. Cada valor de saída é representado por 16 bits.
- HAD_col_num: identificador de qual linha de saída está sendo gerada na saída HAD_col_val.

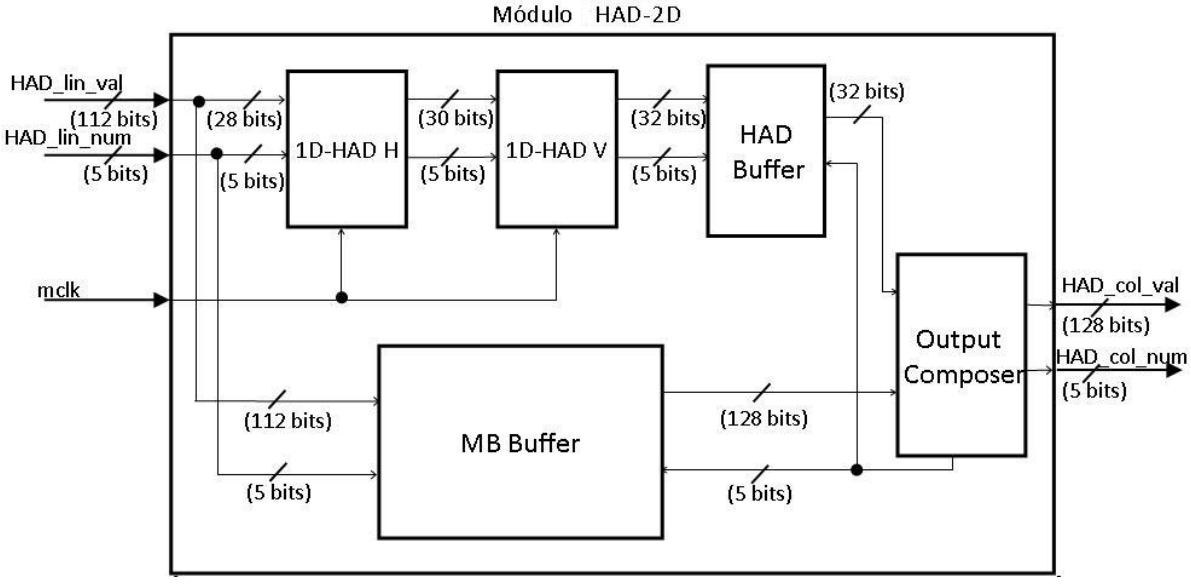

Figura 10: Arquitetura da implementação HAD-2D

3.3 Interface PCI

Conforme citado na seção 2.3, a interface entre o computador onde roda a versão em software do codificador SVC e a placa de hardware (transformadas) deve usar um canal de comunicação PCI v2. Assim foi necessário o desenvolvimento de um módulo em hardware de interface, capaz de interagir com o barramento PCI e trocar dados com os demais módulos desenvolvidos.

Este módulo foi implementado, a partir de um IP PCI LogiCORE fornecido pela empresa Xilinx. Este módulo PCI centraliza a configuração da placa junto ao sistema operacional no computador e manipula os diversos sinais do barramento PCI de forma independente, sem afetar o desempenho dos demais módulos. Os sinais externos são:

- PCI_lin_out: informa a posição na imagem de onde os dados recebidos são oriundos;
- PCI_val_out: contém as informações de pixel recebidos;
- PCI_lin_in: : informa a posição na imagem onde os dados fornecidos devem ser escritos;
- PCI_val_in: contém as informações de pixel fornecidos.

O barramento de endereços PCI_lin_out é usado assim para identificar o número das linhas de cada vetor (DCT_num_lin). Da mesma forma, o barramento de dados PCI_lin_out é ligado diretamente ao módulo de transformada (DCT_lin_val). No caminho inverso ligam-se os dados oriundos do módulo Hadamard (HAD_col_val e HAD_col_num respectivamente)

A fim de facilitar a integração com os módulos de transformada já desenvolvidos, foram utilizados duas FIFOs dupla-porta. São unidades do tipo FIFO (*First Input First Output*), de forma a absorver as diferenças de velocidade entre os módulos de transformada (mais rápidos) e a interface PCI (mais lenta). A seguir se apresenta o diagrama de blocos simplificado.

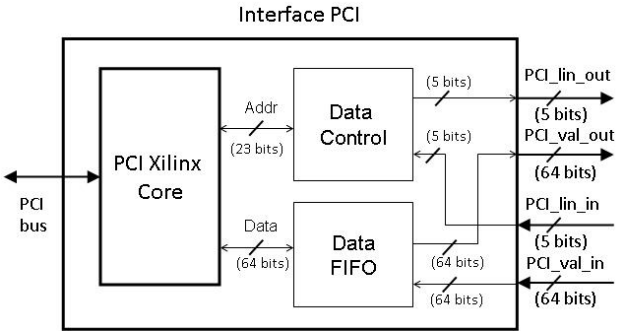

Figura 11: Arquitetura da interface PCI

4. RESULTADOS EXPERIMENTAIS

4.1 Módulos em Hardware

Os módulos anteriormente descritos foram programados em VHDL e sintetizados para o dispositivo FPGA V2P50FF896 da família Virtex II PRO da empresa Xilinx.

A Tabela 2 apresenta os resultados obtidos após a síntese de nossa arquitetura de transformadas proposta: (i) DCT-2D e (ii) HAD-2D.

Tabela 2: Resultados da arquitetura proposta para hardware

	Elementos Lógicos (LUTs)	**Período (ns)**	**Mamostras/s**
Transformada DCT			
DCT-1D H	214	4,027	936,3
DCT-1D V	100	3,069	1303
DCT-2D	618	4,27	1897
Transformada Hadamard			
HAD-1D H	714	9,72	412
HAD-1D V	234	4,321	925
HAD-2D	947	9,7	412
Interface PCI			
PCI_bus	109	3,925	925
Solução completa: DCT + Hadamard + PCI			
Módulo completo sem interface PCI	1565	4,27	1897
Módulo completo com interface PCI	1674	4,27	1897

Pode-se notar que a solução proposta atinge uma taxa de processamento de transforma realmente alta (quase 1900 Mamostras/s). A Tabela 3 compara os resultados obtidos com outros trabalhos relevantes de alto desempenho.

Tabela 3: Comparação de resultados com outros trabalhos

	Elementos Lógicos	**Mamostras/s**
Agostini [19]	3140	126.1
Dorneles [20]	1919	2390
Nossa proposta	**1565**	**1897**

4.2 Módulos em Software

Para validar a proposta foram analisados os módulos de transformada quando operando em uma solução baseada em software, representava pelo código de referência JSVM.

O desempenho individual do módulo de transformada (para cada bloco 4x4) não é afetado pela sequência de vídeo utilizada nem pelo número de camadas. Utilizou-se assim a sequência Foreman com apenas duas camadas escaláveis, ambas com mesma definição CIF (352x288 pixels). De forma geral o algoritmo de transformadas é independente de características do vídeo e por isso não foram experimentadas outras seqüências de vídeo.

Resultados experimentais para 100 amostras determinaram que o número de *Ticks* do processador para realizar a operação de transformadas é de 189 ciclos em média

4.3 Comparações entre Hardware e Software

Conforme identificado por dados experimentais o tempo para realizar as operações de transformadas H.264 em software é 192 para um bloco de 4x4 *clocks*, ou 384 ciclos para dois blocos, o que representa um número bem superior ao registrado para implementações em hardware rápidas. Mais particularmente para o caso de nossa proposta em hardware, após vencida a latência dos *pipelines* internos, consegue-se entregar dois blocos de 4x4 pixels a cada 4 *clocks*. Isto representa um desempenho quase 100 vezes superior ao da versão em software.

Entretanto, para uma devida comparação prática deve-se levar em conta as freqüências de operação envolvidas em cada entidade e inclusive no canal de comunicação entre elas.

Considerando-se a freqüência de trabalho do computador em 2,2GHz, pode-se assim determinar que o tempo de execução é:

Tempo de execução SW= 384 x (1/ 2,2x10^9) = 174,5ns

Para os módulos em hardware a freqüência é de 125 MHz.

Tempo de execução HW= 4 x (1/125x10^6) = 32ns

Considerando que a interface PCI v2 consegue escrever dois blocos 4x4 em quatro ciclos, o tempo total para escrita e leitura é:

Tempo da interface = 8 x (1/66x10^6) = 121ns

Ou seja, nesta avaliação prática a interface de comunicação constitui um gargalo. Mesmo assim registra-se um ganho de 31%.

5. CONCLUSÕES

O presente artigo apresenta uma proposta de solução de hardware/software co-design, que adota uma placa de FPGA para acelerar o desempenho de uma aplicação de codificação H.264 SVC. Nesta placa foram instalados dois módulos otimizados de transformada bidimensional (DCT-2D e HAD-2D).

Durante o desenvolvimento da solução, práticos aspectos como organização da memória e vazão de dados foram considerados visando definir uma solução realizável praticamente.

A comparação de nossos resultados com trabalhos relacionados (Tabela 3) indica que a solução proposta é bastante eficiente. A solução de Dorneles [20] que consegue processar mais amostras que nossa proposta, exige um barramento de 128 bits, além de consumir um número superior de elementos lógicos.

Percebe-se que o desempenho destas soluções rápidas são limitadas pelos canais de comunicação (como PCI e PCI Express) que não conseguem prover e consumir tão grande número de amostras por segundo. Mesmo considerando, porém as restrições de taxa de dados nos canais de comunicação, o presente trabalho representa uma abordagem relevante para aumentar o desempenho no processamento em um codificador H.264 SVC, quando comparado com soluções unicamente baseadas em software.

6. REFERÊNCIAS

[1] Wien M. et. al Real-Time System for Adaptive Video Streaming Based on SVC. In Proc. IEEE Trans.Circuits and Systems for Video Technology, v. 17, n. 9, Sept. 2007.

[2] Rieckl J. Scalable Video for Peer-to-Peer Streaming. Master Thesis University of Wien, 2008. 53 p.

[3] Huang H-S, Peng W-H and Chiang T. Advances in the Scalable Amendment of H.264/AVC. Proc. Adv. Visual Cont. Analysis Adapt. Mult. Comm., jan. 2007. pp 68-76.

[4] Husemann R., Roesler V. and Susin A. "Introduction of a Zonal Search Strategy for SVC Inter-Layer Prediction Module". In: XVII VLSI-SOC. Florianopolis, Brazil.2009.

[5] Fröhlich A. A. and -Preikschat W. S. Scenario adapters: Efficiently adapting components. In Proceedings of the 4th World Multiconf. Syst., Cybernet. and Informat., July 2002.

[6] Hannig F. et. al Co-Design of Massively Parallel Embedded Processor Architectures. ReCoSoC, 2005, pp:27-34.

[7] Husemann R. et al. "Análise Prática e Refinamento de um Codificador de Vídeo Escalável padrão SVC". In XV WEBMEDIA, Fortaleza, CE. 2009.

[8] Qiu Y.,. Badawy W. M, Turney R. D.: A Prototyping Co-design Platform with A Simplified Architecture for Video Codec Implementation. ISCAS 2007: 1220-1224

[9] Wien M., Schwarz H. and Oelbaum T. Performance Analysis of SVC. Proc. of IEEE Trans. on Circuits and Systems for Video Technology, Vol. 17, n. 9, Sept. 2007. pp 1194-1203.

[10] JSVM 9.15, Joint Video Team (JVT) of ISO/IEC and MPEG & ITU-T VCEG, N8963, San Jose, US, April 2007.

[11] Yang Z.; Gao W.; Liu Y. "Performance-Complexity Analysis of High Resolution Video Encoder and its Memory Organization for DSP Implementation" In: Multim. Systems, Architectures and Hardware, 2006. pp. 1261-1264.

[12] Colenbrander R. R. et. al. Co-design and Implementation of the H.264/AVC Motion Estimation Algorithm Using Co-simulation Proc. 11th EUROMICRO 2008. pp. 210-215

[13] Richardson I.E.G. Richardson, H.264 and MPEG-4 Video Compression, UK: Wiley & Sons, 2003.

[14] K. Chen, J. Guo, and J. Wang, "An Efficient Direct 2-D Transform IEEE International Symposium on Circuits and Systems, ISCAS 2005, pp. 4517-4520, 2005.

[15] H. Lin et al. "Combined 2-D Transform and Quantization Architectures for H.264 Video Coders", IEEE International Symposium on Circuits and Systems, pp. 1802- 1805, 2005.

[16] Prasoon A. K.; Rajan K.; "4×4 2-D DCT for H.264/AVC". International Conference on Advances in Computing, Communication and Control. 2009. pp. 573-577.

[17] R. Kordasiewicz, and Shirani, S. "Hardware Implementation of the Optimized Transform and Quantization Blocks of H.264", CCECE, pp. 943-946, Cheng 2004.

[18] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-Complexity Transform and Quantization in H.264/AVC", IEEE Transactions on Circuits and Systems for Video Technology, v. 13, n. 7, pp. 598-603, 2003.

[19] Agostini, L. et al. High Throughput Architecture for H.264/AVC Forward Transforms Block. In: 16th ACM Great Lake Symposium on VLSI – GLSVLSI. Philadelphia, 2006.

[20] Dornelles R. et al, Arquitetura de um Módulo T Dedicado à Predição Intra do Padrão de Compressão de Vídeo H.264/AVC para Uso no SBTVD. Revistas Eletrônicas. PUC-RS, V. 32, N 62, Porto Alegre, 2008. p 75-82.