OTIMIZAÇÕES NA FUNÇÃO DE CUSTO INTERNA DO FILTRO SAO DO PADRÃO DE COMPRESSÃO DE VÍDEO HEVC

FABIANE REDIESS; RUHAN CONCEIÇÃO; BRUNO ZATT; MARCELO PORTO; LUCIANO AGOSTINI

Universidade Federal de Pelotas – PPGC/Mestrado em Ciência da Computação
Grupo de Arquiteturas e Circuitos Integrados (GACI)
{fkrediess, radconceicao, zatt, porto, agostini}@inf.ufpel.edu.br

1. INTRODUÇÃO

A compressão de vídeo tem sido uma ferramenta imprescindível para o sucesso dos vídeos digitais, a ponto de tornar-se inviabilizada a utilização de tais vídeos sem ela, devido à grande quantidade de dados que precisariam ser armazenados e/ou transmitidos. O padrão H.264/AVC, atualmente utilizado em *Blu-rays*, dobrou as taxas de compressão dos padrões anteriores, mas também tornou-se mais complexo computacionalmente (RICHARDSON, 2003).

Mesmo com estes ganhos do padrão H.264/AVC, as pesquisas na área de codificação de vídeo não pararam e em 2013, foi lançado o mais novo padrão de compressão de vídeo, o *High Efficiency Video Coding* (HEVC) (ITU, 2013) cujo objetivo era dobrar as taxas de compressão alcançadas pelo H.264/AVC, mas mantendo ou até mesmo reduzindo a complexidade computacional.

A fim de alcançar estes objetivos, o HEVC propôs uma série de inovações, entre elas, o filtro chamado Deslocamento Adaptativo de Amostra, ou *Sample Adaptive Offset* (SAO). Este filtro objetiva reduzir distorção entre a imagem original e a imagem reconstruída atacando artefatos visuais, como efeitos de *ringing*, considerando as propriedades locais da imagem (SULLIVAN, 2012).

A aplicação do SAO consiste na divisão da imagem em regiões e a cada região pode ser aplicado um deslocamento de banda, o *Band Offset* (BO), ou um deslocamento de borda, o *Edge Offset* (EO). Este deslocamento corresponde à soma de um valor, o *offset*, ao valor da amostra reconstruída. Para cada um destes dois tipos de SAO, o BO e o EO, as amostras são ainda classificadas em categorias de acordo com a intensidade da amostra no caso do BO e com base em comparação com amostras vizinhas no caso do EO (SULLIVAN, 2012).

Nos dois tipos de SAO, o *offset* inicial é calculado como uma média das diferenças entre as amostras originais e reconstruídas daquela determinada categoria. Em seguida, este *offset* é iterado até que chegue a zero e o *offset* com melhor resultado nesta sequência será escolhido como o melhor *offset* daquele tipo e categoria de SAO (FU, 2012). Esta decisão é baseada em uma função de custo que considera heurísticas para estimar custo-benefício da aplicação deste filtro.

O software de referência do padrão HEVC utiliza, para esta função de custo, multiplicadores completos e dados em ponto flutuante. Ambos são críticos para implementações em hardware, uma vez que, demandam maior área em hardware e reduzem o desempenho. Cabe destacar que, neste caso, estes dois pontos críticos são ainda combinados, pois os multiplicadores completos são aplicados a dados em ponto flutuante o que, torna o projeto de hardware para esta solução ainda mais complexo.

Este trabalho propõe otimizações na função de custo usada internamente no filtro SAO do HEVC, bem como, traz as avaliações do impacto destas otimizações no software de referência. O foco destas otimizações encontra-se nos dois pontos

críticos para o desenvolvimento de hardware, o uso de multiplicadores completos e de dados em ponto flutuante.

2. METODOLOGIA

Este trabalho propõe otimizações no filtro de SAO do HEVC que focam na manipulação das equações matemáticas utilizadas no filtro buscando equações que possam ser implementadas, em hardware, de maneira a atingir um alto desempenho, utilizando um reduzido número de recursos do dispositivo alvo.

A fim de avaliar os impactos destas otimizações foram realizadas modificações do software de referência do HEVC, o HM 10 (KIM, 2013). O software com estas alterações foi executado para os vídeos das Condições Comuns de Teste estabelecidas.

3. RESULTADOS E DISCUSSÃO

A Figura 1 apresenta um pseudo-algoritmo para o SAO. Após a etapa de classificação e geração do *offset* inicial para cada tipo e categoria de SAO, o filtro satura (*clip*) este *offset* entre -7 e 7. Em seguida, este *offset* será incrementado ou decrementado até chegar a zero. O custo de todas as iterações será comparado e o *offset* com menor custo será escolhido para aquele tipo e categoria.

```
Para cada tipo de SAO
        Para cada categoria do tipo
                Se existem amostras neste tipo/categoria
                        iniOffset = accumDif / Count
                        Clip (ini_offset, -7, 7)
                        melhorCusto = λ
                        Enquanto iniOffset != 0
                                custo_temp = calcCusto (iniOffset)
                                Se melhorCusto < custoTemp
                                        melhorCusto = custoTemp
                                Se iniOffset > 0
                                        iniOffset --
                                Senão
                                        iniOffset ++
```

Figura 1. Pseudo-algoritmo para a função de custo do SAO.

A função de custo usada internamente no SAO aplica heurísticas para estimar a distorção e o número de bits necessários para representar o tipo e a categoria escolhidos. A heurística envolve dois passos. O primeiro consiste em estimar a distorção, conforme a equação em (1), onde ‘n’ corresponde ao número de ocorrências na categoria, ‘a’ à diferença acumulada e ‘b’ à diferença media entre as amostras originais e reconstruídas, ou seja, o *offset* inicial.

$$d = (nb^2) – (2ab) \qquad (1)$$

O segundo passo consiste na estimativa do número de bits necessários para representar a informação do tipo e categoria de SAO alvos. Esta estimativa considera o valor do *offset* e o tipo de SAO. Este passo pode ser representado pelas equações constantes em (2), onde ‘c’ é o custo resultante, ‘d’ é a distorção calculada em (1), λ é o multiplicador de Lagrange e ‘r’ é o número estimado de bits.

$$c = d + r.\lambda, \quad \text{se SAO\_type = EO}$$
$$c = d + (r+1).\lambda, \quad \text{se SAO\_type = BO} \qquad (2)$$

O algoritmo do SAO no software de referência do HEVC considera dados em ponto flutuante e utiliza multiplicadores completos sobre estes dados. Estes são dois pontos críticos para o projeto de hardware, uma vez que, demandam grandes áreas em chip e reduzem o desempenho.

Objetivando atacar estes dois pontos críticos, estes trabalho propões dois tipos de otimizações na função interna de custo do SAO do HEVC:

1. Usar de dados inteiros ou com precisão de ponto fixo variando o número de bits fracionários entre 1 e 8 bits no lugar de dados em ponto flutuante.
2. Utilizar desenrolamento de laço (*loop unrolling*), ou seja, processar todas as 7 possíveis iterações em apenas um passo.

Esta última otimização permitiu a eliminação de todos os multiplicadores completos utilizados nas equações (1) e (2), substituindo-os por multiplicações por constantes, que podem ser implementados de forma mais eficiente em hardware através de somas e deslocamentos. Esta otimização elimina também o divisor completo utilizado para calcular o *offset* inicial, pois como serão calculados para todos os *offsets*, não há necessidade de saber qual seria o inicial.

Como serão processadas todas as iterações ao mesmo tempo, é possível efetuar a substituição na equação (1) de cada possível *offset*, ou seja, entre 7 e 1 ou entre -7 e -1 e transformar as multiplicações por constantes em somas e deslocamentos. Por exemplo, para o *offset* igual a 5, a equação equivalente seria a presente em (3).

$$d_5 = 25n - 10a \qquad (3)$$

Os experimentos para a primeira otimização foram feitos para os 16 primeiros quadros de 11 vídeos das Condições Comuns de Teste estabelecidas pelo padrão. Os resultados médios em BD-rate da solução proposta em comparação com versão regular do HM constam na Tabela 1.

Tabela 1. Experimentos com dados inteiros e ponto fixo.

	Y BD-rate			
	Intra Only	***Random Access***	***Low Delay***	**Média**
Inteiros	-0,005	0,103	-0,014	0,028
1 bit	0,000	0,000	-0,006	-0,002
2 bits	0,000	-0,020	-0,008	-0,009
3 bits	0,000	-0,005	-0,008	-0,004
4 bits	0,000	-0,005	-0,007	-0,004
8 bits	0,000	0,000	-0,006	-0,002

Estes resultados demonstram que o uso de dados inteiros em substituição aos em ponto flutuante, resulta em um aumento de 0,028 no BD-rate, o que significa que para manter a mesma qualidade, esta proposta resultaria em um aumento de apenas 0,028% em *bitrate*. Analisando o caso com ponto fixo e precisão de 3 bits, a grande maioria das sequências de vídeo apresenta uma diferença menor que 0,01. Entretanto, o resultado médio geral representou um ganho de 0,004, demonstrando que o ponto flutuante não é necessário.

A segunda proposta de otimização consiste em processar todos os 7 possíveis *offsets* ao invés de começar a partir do *offset* inicial. Esta otimização também foi avaliada utilizando o software de referência utilizando vídeos das Condições Comuns de Teste. A Tabela 2 apresenta estes resultados comparando, a exemplo da primeira linha da tabela, a execução regular do HM com a execução do HM com desenrolamento de laço. De maneira análoga, a segunda linha compara a versão que usa inteiros, mas calculando o *offset* inicial, com a versão agora comparando todos os possíveis *offsets*.

Tabela 2. Experimentos para o desenrolamento de laço.

	Y BD-rate			
	Intra Only	***Random Access***	***Low Delay***	**Média**
HM	0,0000	0,0404	4,1215	1,3873
Inteiros	-0,0421	-0,0578	0,0542	-0,0153
1 bit	0,0000	0,0000	0,0000	0,0000
2 bits	0,0000	0,0000	0,0000	0,0000
3 bits	0,0000	0,0000	0,0000	0,0000

Para as duas primeiras linhas da tabela, HM e inteiros, alguns vídeos apresentam alguma variação. Entretanto, para os valores seguintes, a comparação entre a versão apenas com a utilização de dados em ponto fixo e a versão que propõe o desenrolamento de laço, os comportamentos são idênticos.

Com estes resultados, a decisão para uma implementação em hardware proposta neste trabalho considera a otimização com desenrolamento de laço com dados em ponto fixo com precisão de 3 bits, pois a diferença média é baixa, apresentando até um pequeno ganho, entretanto, a partir deste ponto, a maioria das sequências de vídeo e suas diferentes configurações apresentam diferenças menores que 0,01 comparando com o HM normal.

4. CONCLUSÕES

Este trabalho apresentou otimizações na função interna de custo do filtro SAO do padrão de compressão de vídeo HEVC. Esta decisão interna é responsável por decidir para cada tipo e categoria de SAO, qual *offset* será utilizado. As otimizações focam em dois pontos críticos para o desenvolvimento de hardware, o uso dados em ponto flutuante e de multiplicadores completos.

Com as otimizações propostas neste trabalho foi possível eliminar os multiplicadores/divisores completos necessários. Além disso, foi avaliada a utilização de dados inteiros e em ponto fixo.

Com base nos resultados, propõe-se para uma implementação em hardware uma precisão de 3 bits de ponto fixo, pois, nestes resultados verificou-e que na maioria das sequências de vídeo a diferença é menor que 0,01 e em média houve um pequeno ganho de 0,004.

5. REFERÊNCIAS BIBLIOGRÁFICAS

International Telecommunication Union (ITU). **ITU-T Recommendation H.265: High Efficiency Video Coding, Audiovisual and Multimedia Systems**, abril 2013. Online. Disponível em: http://www.itu.int/rec/T-REC-H.265-201304-I.

FU, C.-M. et al. Sample Adaptive Offset in the HEVC Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1755-1764, 2012.

KIM, I. et al. High Efficiency Video Coding (HEVC) Test Model 10 (HM10) Encoder Description. Joint Collaborative Team on Video Coding (JCT-VC) JCTVC-L1002. 12th Meeting. Genebra, 2013.

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

SULLIVAN, G. et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1649-1668, 2012.