ANÁLISE DA OPERAÇÃO DE DIVISÃO EM PONTO FLUTUANTE EM HARDWARE PARA A CODIFICAÇÃO DE VIDEO HDR

ALEX MACHADO BORGES; BRUNO ZATT; MARCELO PORTO

Universidade Federal de Pelotas – Grupo de Arquiteturas e Circuitos Integrados
{amborges, zatt, porto}@inf.ufpel.edu.br

1. INTRODUÇÃO

A qualidade de vídeos vem crescendo ao longo dos anos e novas tecnologias de vídeo surgem, uma delas é a tecnologia *High Dynamic Range* (HDR) (LUTHRA et.al., 2014), capaz de levar representar uma imagem mais próxima ao que o olho humano é capaz de perceber (REINHARD, et.al., 2010). No entanto, a codificação de vídeos em HDR necessita de etapas extras a fim de possibilitar a codificação e visualização destes vídeos. Por ser uma tecnologia muito particular, não há codificadores de vídeos especializados em HDR, fazendo-se uso de codificadores tradicionais, especificamente do *High Efficiency Video Coding* (HEVC) (SULLIVAN et.al., 2013), que exige um procedimento de preparar o vídeo HDR para ser processável pelo HEVC, etapa essa chamada de pré-processamento de vídeos HDR. No trabalho desenvolvido por Borges (2016) apresentou-se um software referência de pré-processador de vídeos HDR, do qual se realizou otimizações e foi proposto um hardware para o bloco temporalmente mais custoso desse software. A justificativa do hardware se dá pelo custo de se entregar os dados pré-processados em um tempo aceitável, mesmo que não em tempo real. Sobre o bloco escolhido para desenvolvimento em hardware é o *Segmented_Spline*, responsável por modificar a representação as cores, de forma a valorizar tons escuros, mais perceptíveis ao olho humano (BORGES, 2016). Toda a arquitetura utiliza operações entre ponto-flutuantes de 32 bits, segundo o padrão IEEE 754 (IEEE, 2008).

Neste trabalho foi possível o desenvolvimento de uma arquitetura de hardware capacitada a operar sob uma frequência de execução de 16.93MHz, utilizando barreiras temporais que operam na mesma frequência da etapa gargalo deste hardware. No entanto, o objetivo deste trabalho foi obter uma arquitetura capacitada a operar em tempo real, e para obter isso, propôs-se um nível de paralelismo, que com cinco núcleos já se torna capacitada a trabalhar com vídeos em alta definição de 1280x720 pixels e 30 quadros por segundo (HD720p@30). Ao fim deste trabalho, enfatizou-se que a operação gargalo do hardware é a operação matemática de divisão, necessitando de uma revisão da biblioteca VHDL utilizada para dar suporte às operações com ponto-flutuante.

Este artigo vem pra realizar essa análise de bibliotecas a fim de encontrar uma biblioteca de ponto-flutuante compatível com a IEEE 754 para substituir a atual biblioteca utilizada no trabalho apresentado, a fim de obter melhorias no hardware já desenvolvido, de forma a possibilitar que a arquitetura esteja capacitada a operar em vídeos UHD (*Ultra High Definition*).

2. METODOLOGIA

As análises propostas foram feitas desenvolvendo uma arquitetura que envolve a divisão de dois números em ponto-flutuante de 32 bits segundo o padrão IEEE 754. Na Figura 1 está representado um diagrama em alto nível dessa arquitetura. Onde o bloco principal dessa arquitetura deve ser substituído conforme a operação de divisão oferecida por cada biblioteca utilizada.

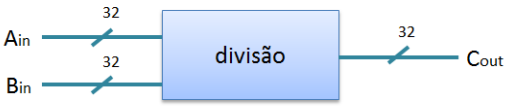

Figura 1. Diagrama em alto nível da arquitetura teste desenvolvida

Foram utilizadas duas bibliotecas de ponto-flutuante, a original utilizada no trabalho é a *IEEE_PROPOSED*[1] e a biblioteca utilizada como comparativo e em conformidade a IEEE 754 é a *fplib*[2], no entanto houve incompatibilidades com essa biblioteca devido a atualizações na biblioteca padrão *IEEE* presente no VHDL, necessitando de algumas correções no algoritmo original da biblioteca.

É importante ressaltar que a biblioteca *IEEE_PROPOSED* possui facilidade em seu uso, pois define dentro do VHDL uma abstração de tipo denominado float32, possibilitando a aplicação das operações básicas da matemática diretamente, tal qual se fosse uma linguagem padrão de programação. Já a biblioteca *fplib*, foi desenvolvida para dar suporte a diversos formatos de representação do ponto-flutuante, entre eles o IEEE 754. E por isso, não há abstração de tipos, como ocorre na biblioteca original utilizada, todos os números devem estar como *std_logic_vector*. E por não haver abstrações, todas as operações são feitas através do uso de blocos de hardware (utilizando-se *port-maps*), isso torna o desenvolvimento de um hardware complexo, como é o caso do pré-processador, extremamente complexo. Há em *fplib* um sistema de conversão entre representações de ponto-flutuantes, em especial do IEEE 754 para FP, ponto-flutuante interno, possuindo dois bits a mais que os utilizados pelo ponto-flutuante declarado pelo usuário, usados como sinais de erro. No entanto a conversão de IEEE 754 para FP de 32 bits é uma associação direta, dispensando a chamada extra desse bloco de conversão, e deixando dois bits livres para uso pela biblioteca *fplib*.

O código VHDL desenvolvido para a *IEEE_PROPOSED* pode ser visto na Figura 2, onde é possível observar o uso da abstração *float32* e a divisão sendo feita de forma intuitiva e direta. Já o código para o *fplib*, apresentado parcialmente na Figura 3, possui os sinais declarados como *STD_logic_vector(33 downto 0)*, com dois bits a mais como já explicado anteriormente. A operação de divisão é feita chamando o bloco de divisão, havendo necessidade de definir o tamanho do expoente (*wE*) e da mantissa (*wF*), internamente a biblioteca já inclui o bit do sinal e os dois bits de erro. Há também a possibilidade de ativar as barreiras temporais internas a operação, através do atributo interno *reg*.

3. RESULTADOS E DISCUSSÃO

Apresentadas as três possíveis arquiteturas para o cálculo de divisão entre ponto-flutuantes, foram então processados pela ferramenta de arquiteturas Altera Quartus II 10.2b, no qual foram expostos em duas FPGAs diferentes, *Ciclone V 5CEBA2F17A7* e *Stratix V 5SGSMD4E1H29C1*, cujos resultados estão apresentados na Tabela 1 e Tabela 2, respectivamente às placas FPGAs. E como é observável, há uma diferença notória entre a frequência e área entre as diferentes bibliotecas, apesar da *fplib* utilizando barreiras temporais (pipeline) ter 14 vezes mais registradores, todavia compensando na frequência atingida, que é superior em 20 vezes à biblioteca utilizada no trabalho original.

[1] A biblioteca *IEEE_PROPOSED* pode ser obtida em http://www.vhdl.org/fphdl/
[2] A biblioteca *fplib* pode ser obtida em http://www.ens-lyon.fr/LIP/Arenaire/Ware/FPLibrary/

```
library IEEE;
library IEEE_PROPOSED;
library work;
use IEEE.std_logic_1164.ALL;
use IEEE_PROPOSED.float_pkg.all;
entity divisoes is
   port(
      clk:  in STD_logic;
      Ain:  in STD_logic_vector(31 downto 0);
      Bin:  in STD_logic_vector(31 downto 0);
      Cout: out STD_logic_vector(31 downto 0)
   );
end divisoes;
architecture dvfp of divisoes is
   signal fa: float32;
   signal fb: float32;
   signal fc: float32;
   signal vO: STD_logic_vector(31 downto 0);
   begin
      fc <= fa / fb;
      dfvp: process(clk) begin
         if rising_edge(clk) then
            fa <= to_float(Ain);
            fb <= to_float(Bin);
            vO <= to_slv(fc);
         end if;
      end process dvfp;
   Cout <= vO;
end dvfp;
```

Figura 2. Código VHDL para cálculo de divisão utilizando a *IEEE_PROPOSED*

```
. . .
library fplib;
use fplib.pkg_fplib.all;
      . . .
architecture dvfp of divisoes is
   component Div_Clk is
      generic (
         fmt : format;
         wE : positive := 8;
         wF : positive := 23;
         reg : boolean := true
      );
      port (
         nA : in STD_logic_vector(wE+wF+2 downto 0);
         nB : in STD_logic_vector(wE+wF+2 downto 0);
         nR : out STD_logic_vector(wE+wF+2 downto 0);
         clk : in STD_logic
      );
   end component;
   . . .
   begin
      theDiv: Div_Clk
         generic map (
            fmt => FP, wE  => 8,
            wF  => 23, reg => true
         )
         port map (
            nA => fa, nB => fb,
            nR => fc, clk => clk
              );
      dvfp: process(clk) begin
         . . .
```

Figura 3. Código VHDL parcial para cálculo de divisão utilizando a *fplib*

Tabela 1. Síntese das arquiteturas usando a FPGA da família Ciclone V

	ALMs	Registradores	FMax (MHz)
IEEE_PROPOSED	1 265	96	7.75
fplib	785	96	12.28
fplib pipelined	843	1 410	139.33

Tabela 2. Síntese das arquiteturas usando a FPGA da família Stratix V

	ALMs	Registradores	FMax (MHz)
IEEE_PROPOSED	1 231	96	17.37
fplib	745	96	28.36
fplib pipelined	817	1 411	350.26

Ao se obter as frequências apresentadas nas tabelas, obtendo-se evidente melhoria na operação matemática de divisão entre números em ponto-flutuante, operação gargalo da arquitetura desenvolvida no trabalho anterior já apresentado, torna-se possível, teoricamente, construir uma nova arquitetura com rearranjo de barreiras temporais a fim de possibilitar que a arquitetura de Segmented_Spline opere em tempo real para vídeos de HD720@30 com um único núcleo, considerando a FPGA Ciclone V, ou mesmo para vídeos HD1080p@30 (1920x1080p) se considerado a FPGA Stratix V. Garantindo ainda uma margem de perda na frequência causados pelo acréscimo dos demais operadores presentes na arquitetura original.

4. CONCLUSÕES

Neste artigo apresentou-se o trabalho original onde foi proposto o desenvolvimento de uma arquitetura em hardware para o bloco temporalmente mais custoso do pré-processamento de vídeos HDR. O estudo de uma biblioteca em ponto-flutuante em conformidades a IEEE 754 foi realizado, estudando o seu impacto na operação matemática de divisão, gargalo do hardware anteriormente desenvolvido. A biblioteca escolhida para realização da comparação permite utilização nativa de operações básicas da matemática com uso de barreiras temporais, permitindo que se obtenham frequências de execuções 20 vezes superiores a um custo de 14 vezes o número de registradores, em relação à biblioteca original utilizada. A frequência obtida utilizando com essa nova biblioteca, quando aplicado barreiras temporais, é de 350MHz, capacitando realizar essa operação matemática de divisão para vídeos de alta resolução, com um único núcleo de processamento. O próximo passo da pesquisa é adequar toda arquitetura para trabalhar com a nova biblioteca, a fim de obter uma nova arquitetura mais adaptada para vídeos em resolução UHD.

5. REFERÊNCIAS BIBLIOGRÁFICAS

BORGES, A.M. Otimização e Projeto de Hardware Dedicado para o Pré-Processamento de Vídeo em High Dynamic Range (HDR). 2016. 76f. Monografia – Graduação em Ciência da Computação, Universidade Federal de Pelotas.

IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. **IEEE Std 754-2008**, [S.l.], p.1–70, Aug 2008. DOI:10.1109/IEEESTD.2008.4610935.

LUTHRA, A; FRANÇOIS, E.; HUSAK, W.. **Draft Requirements and Explorations for HDR/WGC Content Distribuition and Storage.** The 108th MPEG Meeting. Valencia - Espanha, 31 mar. 2014. Acessado em 19 jun. 2016. Online. Disponível em: http://mpeg.chiariglione.org/standards/exploration/high-dynamic-range-and-widecolour-gamut-content-distribution/n14510-draft.

REINHARD, E.; WARD, G.; PATTANAIK, S.; DEBEVEC, P.; HEIDRICH, W.; MYSKOWSKI, K. **High Dynamic Range Imaging, Second Edition**: Acquisition, Display, and Image-Based Lighting. segunda. ed. San Francisco: Morgan Kaufmann, 2010. ISBN: 978-0123749147.

SULLIVAN, G. J.; BOYCE, J. M.; YINGCHEN; OHM, J.-R.; SEGALL, C. A.; VETRO, A. Standardized Extensions of High Efficiency Video Coding (HEVC). **IEEE Journal of Selected Topics in Signal Processing**, [S.l.], v.7, n.6, p.1001–1017, 2013. Acessado em 19 jun. 2016. Online. Disponível em http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6630053.