Universidade Federal de Pernambuco
Centro de Informática

Pré-projeto

Modelo RTL de módulo de operações aritméticas, lógicas e de conversão sob números em ponto flutuante sintetizável em FPGA

Lucas Fernando da Silva Cambuim

Trabalho de Graduação

Recife
20 de outubro de 2014

CAPÍTULO 1

Contexto

Devido aos avanços da tecnologia de fabricação, a capacidade do silício está dobrando a cada 18 meses como previsto pela lei de Moore [11]. Isto permite o desenvolvimento e implementação de sistemas mais complexos em um único chip. Contudo, com circuitos integrados (CIs) mais complexos, o seu custo e o seu tempo de desenvolvimento aumentam, uma vez que todas as suas funcionalidades, cada vez mais complexas, precisam ser validadas antes que o CI seja lançado no mercado. O tempo entre o desenvolvimento e o lançamento do primeiro protótipo no mercado é chamado de time-to-market e quanto menor for esse tempo maior é a chance de deter uma fatia maior do mercado para esse produto. Dessa forma, afim de reduzir o gap de produtividade, os engenheiros de projeto desenvolveram novas metodologias e técnicas de projeto para suportar a complexidade crescente inerente nesses grandes CIs.

Uma metodologia emergente é o projeto system-on-chip (SoC), onde blocos pré-projetados e verificados, frequentemente chamados de blocos de propriedade intelectual (IP), IP cores ou componentes virtuais, são obtidos a partir de fontes internas, ou de terceiros e combinados em um único chip [10]. Esses IP cores reusáveis podem incluir processadores embarcados, blocos de memória, blocos analógicos e componentes que manipulem funções de processamento específico da aplicação. IP core é a forma mais popular de projeto reusável na indústria de hoje, que pode ser dividido em três categorias: soft, firm e hard. Os IPs hard são manifestações físicas do projeto de IP. Esses são melhores para aplicações do tipo plug-in-play e são menos portáveis e flexíveis do que os outros dois tipos de cores. Como os IPs hard, os IPs firm também carregam informações do processo de layout mas são configuráveis para varias aplicações. O mais flexível dos três, os IPs soft existem ou como uma netlist, uma lista das portas lógicas e interconexões associadas compondo um CI, ou código em nível de transferência de registrador (RTL), tais como System-verilog [1], verilog e VHDL. A diferença entre essas categorias quanto a critérios como fator de risco e custo é mostrada em [6].

Os IP-cores podem ser utilizados como elementos do projeto de ASICs (Aplication Specific Integrated Circuit) ou de modelos de lógica reconfigurável FPGA (Field Programmable Gate Arrays). Os ASICs são customizados para uma aplicação especifica e o seu chip é produzido em empresas chamadas foundry. Uma vez que o chip é confeccionado é impossível modificar sua funcionalidade sem passar novamente por todo o processo de produção de um novo chip. Já os FPGAs são hardwares formados por blocos lógicos, componentes de armazenamento e chaveamento, que podem ser reconfigurados para implementar diferentes funções lógicas. A vantagem dos FPGAs em relação aos ASICs está no rápido time-to-market e baixo custo de engenharia não recorrente [12] já que para modificar sua funcionalidade basta apenas sintetizar o novo código através de ferramentas EDA tais como Quartus II da Altera e Synplify da Synopsys e gravar novamente na FPGA. Uma vez que os FPGAs alcançaram um nível de inte-

gração suficiente para incorporar todo o sistema ou vários sistemas dentro de um único chip de silício, a tendência são os projetos passarem a ser do tipo system-on-a-programmable (SOPC).

IP cores bastante utilizados em CIs são aqueles que operam números no formato de ponto flutuante (PF). Este formato representa os números reais no computador. Em virtude da capacidade limitada dos dispositivos computacionais essa representação possui um erro associado. Em outras palavras, PF é uma representação aproximada dos números reais. O padrão mundialmente utilizado para representação dos números em PF é o IEEE-754 [2]. Esse padrão define um conjunto de representações e condições excepcionais para manipular tais números.

A importância dada a essa representação está na capacidade de fornecer simultaneamente grande espaço de números e um alto grau de precisão quando comparado com a representação de números no formato de ponto fixo, uma outra forma de representar números reais. Aplicações na área de processamento de imagens como em [8], processamento de sinais como em [4] necessitam de números com alta precisão. Contudo a manipulação de números em PF é mais complexa do que a operação com números em ponto fixo. Como consequência, uma grande quantidade de processadores já incluem um hardware específico para tratar números em PF afim de facilitar qualquer projeto que envolva a manipulação de tais números. Como exemplo, o processador mips R3000 [5] inclui a possibilidade de adicionar um coprocessador R3010 [9] externamente para fornecer um conjunto de instruções em PF.

CAPÍTULO 2

Objetivos

Tendo em vista a importância dos números em PF, do desenvolvimento de módulos reusáveis e das vantagens de se usar FPGA no desenvolvimento do projeto, o presente trabalho tem como objetivo desenvolver no nível de abstração RTL de uma unidade que implementa um conjunto de operações com números em PF como as aritméticas, de comparação e de conversão seguindo o padrão definido pelo IEEE-754. Ao todo serão 25 operações no qual cada uma deverá manipular tanto números precisão simples como em precisão dupla. A linguagem de desenvolvimento que será utilizada para esse projeto é o System-verilog e o IP core será validado através da prototipação em FPGA.

CAPÍTULO 3

Metodologia

O IP core em questão será desenvolvido seguindo a metodologia de desenvolvimento denominado ip-PROCESS [7] até sua última etapa do processo: prototipação em FPGA.

Dessa forma, o trabalho iniciará com o levantamento bibliográfico mais detalhado sobre a representação de PF onde serão extraídos os requisitos funcionais e não funcionais de cada instrução. Após a definição dos requisitos, serão estudados os algoritmos e, se necessário, suas otimizações, que implementam cada instrução.

Após o levantamento bibliográfico, será iniciada a codificação de cada instrução usando a linguagem System-Verilog. As ferramentas que serão utilizadas para auxiliar o desenvolvimento da unidade serão o ModelSim, Quartus II e o Synplify. Tanto o ModelSim como o Quartus II serão utilizados para simular o projeto com um conjunto de vetores de teste. O Synplify será utilizado para otimizar o código para a plataforma FPGA DE2-35, ou seja, buscar maiores frequências de operação para a unidade.

Em paralelo, o ambiente de verificação funcional para simulação será desenvolvido afim de validar cada operação. Esse ambiente diferentemente do ambiente oferecido pelo ModelSim, gera estímulos randômicos afim de abranger todos os casos possíveis de cenário. A metodologia adotada é a OVM-tpi [3]. De forma breve, foi escolhida essa metodologia pois ela resolve problemas de sincronismo entre a unidade e o ambiente de validação que as outras metodologias anteriores não resolveram, reduzindo o tempo do desenvolvimento desse ambiente. Vale mencionar, que o autor desse trabalho não deverá participar do processo de desenvolvimento do ambiente de validação e sim, da etapa de validação da unidade. Em outras palavras, após o ambiente de validação tiver sido finalizado, o autor poderá utilizá-lo para validar a unidade. Isso envolve executar o ambiente de validação, inserindo-o a unidade desenvolvida nesse trabalho, também chamado de DUV, e verificando se os resultados da unidade são iguais aos resultados do modelo de referência definido no ambiente de validação. Se todos os valores forem cobertos sem erro, então pode-se dizer que a unidade foi validada por simulação.

E por fim, o código será validado através da prototipação na plataforma FPGA. Isso envolve construir um ambiente que permita validar o código na FPGA e/ou construir alguma aplicação que demonstre o uso da unidade desenvolvida.

CAPÍTULO 4

Cronograma

Atividades	Setembro	Outubro	Novembro	Dezembro	Janeiro
1. Levantamento dos requisitos funcionais e não funcionais	•	•			
2. Definição das interfaces dos módulos	•	•			
3. Estudo e implementação dos algoritmos que atendam aos requisitos elucidados para cada operação	•	•	•	•	
4. Validação através do ambiente de teste			•		
5. Validação através da prototipação em FPGA			•		
6. Validação de módulos através de aplicação prática como processamento de imagem				•	
7. Obtenção dos resultados através de simulação e prototipagem				•	
8. Escrita do trabalho			•	•	•
8. Apresentação do trabalho					•

CAPÍTULO 5

Assinaturas

Cientes do programa:

Edna Natividade da Silva Barros - Professora Orientadora

Lucas Fernando da Silva Cambuim - Aluno

Referências Bibliográficas

[1] Systemverilog 3.1a language reference manual. Technical report, Accellera Organization, 2004.

[2] Ieee standard for floating-point arithmetic - redline. *IEEE Std 754-2008 (Revision of IEEE Std 754-1985) - Redline*, pages 1–82, Aug 2008.

[3] R. Camara. Desenvolvimento de mecanismo de verificação baseado em systemverilog. Master's thesis, Centro de Informatica, Universidade Federal de Pernambuco, Recife, PE, 2010.

[4] Y. Cui, B. Chen, and S. Zhang. Design of floating-point operation based on fpga and it's application. In *Signal Processing, 2006 8th International Conference on*, volume 4, pages –, Nov 2006.

[5] K. Gerry and H. Joe. *Mips Risc Architeture*. Upper Saddle River (NJ), 2 edition, 1992.

[6] D.-K. Kim, K.-W. Kwon, J.-C. Choi, and C.-D. Lee. Reusable intellectual property cores in pc data protection asic design. In *ASICs, 1999. AP-ASIC '99. The First IEEE Asia Pacific Conference on*, pages 278–281, 1999.

[7] M. Lima, A. Aziz, D. Alves, P. Lira, V. Schwambach, and E. Barros. ipprocess: using a process to teach ip-core development. In *Microelectronic Systems Education, 2005. (MSE '05). Proceedings. 2005 IEEE International Conference on*, pages 27–28, June 2005.

[8] M. Macieira, L. Cambuim, L. Souza, L. Oliveira, M. Rios, and E. Barros. The design of an image converting and thresholding hardware accelerator. *Brazilian Symposium on Computing Systems Engineering*, November 2014.

[9] C. Rowen, M. Johnson, and P. Ries. The mips r3010 floating-point coprocessor. *Micro, IEEE*, 8(3):53–62, June 1988.

[10] R. Saleh, S. Wilton, S. Mirabbasi, A. Hu, M. Greenstreet, G. Lemieux, P. Pande, C. Grecu, and A. Ivanov. System-on-chip: Reuse and integration. *Proceedings of the IEEE*, 94(6):1050–1069, June 2006.

[11] R. Schaller. Moore's law: past, present and future. *Spectrum, IEEE*, 34(6):52–59, Jun 1997.

[12] F. Sun, X. Li, Q. Wang, and C. Tang. Fpga-based embedded system design. In *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pages 733–736, Nov 2008.