Aceleração Computacional usando FPGAs

Tiago Matos Santos[1], Deusdete Miranda Matos Junior[2] Vinicius Petrucci[3]

[1] Escola Politécnica – UFBA

[2]SENAI CIMATEC – SENAI Bahia

[3]Departamento de Ciência da Computação – UFBA

tiago.matos@zoho.com, dmmj.eng@gmail.com,petrucci@dcc.ufba.br

Resumo. *Sistemas computacionais de alto desempenho podem integrar recursos diversos: CPUs multi-cores, GPUs e dispositivos lógicos programáveis como os FPGAs. Tanto GPUs quantos FPGAs são utilizados como aceleradores para aumentar o paralelismo e a escalabilidade na execução dos programas. Diversas aplicações fazem uso de cálculos matemáticos especializados, e que consumem grande parte do tempo de processamento das CPUs de propósito geral, isso justifica que suas implementações façam uso de uma entidade de processamento adaptada em paralelo. O objetivo deste trabalho é projetar e avaliar um protótipo de acelerador usando FPGA, buscando melhorar o desempenho de aplicações com alta demanda computacional.*

1. Introdução

O uso de serviços computacionais tem apresentado uma expansão contínua, e acompanha a disponibilização de aplicações com funcionalidades inovadoras. Alguns exemplos são: reconhecimento de voz, tradução de textos, caracterização de imagens, entre outras, que em comum possuem alto custo computacional. As diversas aplicações implementam algoritmos específicos com características diferentes no uso dos recursos computacionais. Assim, existem variações na infraestrutura que acaba sendo exigida. Algumas aplicações podem ser mais intensivas no uso de cálculos numéricos com ponto flutuante, ou podem exigir mais operações de acesso a memória, outras requerem apenas cálculos com valores inteiros ou lógicos, e assim em diante [Boutaba et al. 2012].

Os processadores usualmente empregados nos computadores são projetados para serem generalistas em relação às aplicações, nem sempre apresentando afinidade com os recursos exigidos pelos algoritmos. Dispositivos de *hardware* específicos para determinadas aplicações configuram os chamados co-processadores ou aceleradores. Tal abordagem tem sido largamente utilizada por meio das placas gráficas (GPU - *Graphic Processor Unit*), que proveem ao sistema sua capacidade de processamento, inclusive, não limitadas apenas às aplicações gráficas.

Existe uma categoria de circuitos integrados que tem a característica de ser reconfigurável, isto é, podem ter seu circuito digital completamente reprogramado por várias vezes depois de fabricados, este é o caso dos FPGAs. Estes dispositivos podem ser utilizados para implementar aceleradores totalmente customizados às necessidades das aplicações, e nos poupar das etapas de fabricação, encapsulamento, e teste dos circuitos integrados. Assim, podemos seguir da etapa de projeto digital diretamente para um chip utilizável, tendo também grande flexibilidade para implementar quaisquer modificações futuras [Hauck and DeHon 2010].

2. Aceleração de Aplicações usando *Hardware* Reconfigurável

Computadores de propósito geral executam as instruções de um programa armazenado em memória numa unidade central de processamento (CPU), que é projetada para fornecer um determinado conjunto de instruções mais básicas tais como, movimentação de dados e operações lógicas/aritméticas ao programa [Patterson and Hennessy 2011]. Alguns cálculos em certas aplicações consomem vários ciclos de CPU, e em muitos casos, são executados repetidamente. Especificar um circuito dedicado somente para essas funções abre a possibilidade de otimizar o desempenho de um sistema, podendo inclusive oferecer melhor eficiência energética [Fowers et al. 2012]. Assim, com mais desempenho e menor consumo, o uso de aceleradores em *hardware* vem se mostrando promissor para processos computacionais de alto desempenho [Putnam et al. 2014].

2.1. Vantagens do uso de FPGAs

Circuitos integrados possuem um extenso e complexo processo de desenvolvimento, que envolve diferentes níveis de abstração. Tal processo inclui como etapas a definição das especificações, da arquitetura, o projeto lógico, diversos testes, preparação de layouts do circuito físico, fabricação dos dies, apenas para citar algumas, e cada qual apresentando incrementos os custos do desenvolvimento[Wolf 2008]. Existem circuitos integrados que são reconfiguráveis, isto é, podem depois de fabricados assumir funcionalidades diferentes de acordo com o que neles se desejar programar ou configurar. Uma das classes mais proeminentes destes dispositivos são os FPGAs (*Field-Programmable Gate Arrays*).

A utilização de chips FPGA é vista como um meio termo entre a abordagem baseada microprocessadores e o desenvolvimento de ASICs, no que diz respeito a complexidade e performance. Apesar disso, existem semelhanças entre FPGAs e microprocessadores no que diz respeito a flexibilidade, mas no sentido de permitir que modificações no design sejam aplicadas de forma ágil [Hauck and DeHon 2010].

3. Projeto de um Acelerador em FPGA

Neste projeto, estudamos a aceleração de uma aplicação que executa a operação de convolução entre sequencias numéricas. Buscamos uma forma de oferecer ao programa principal uma interface com o co-processador sintetizado no chip FPGA, de forma que os cálculos sobre os dados de entrada sejam todos efetuados no circuito digital customizado implementado.

3.1. Placa de Desenvolvimento

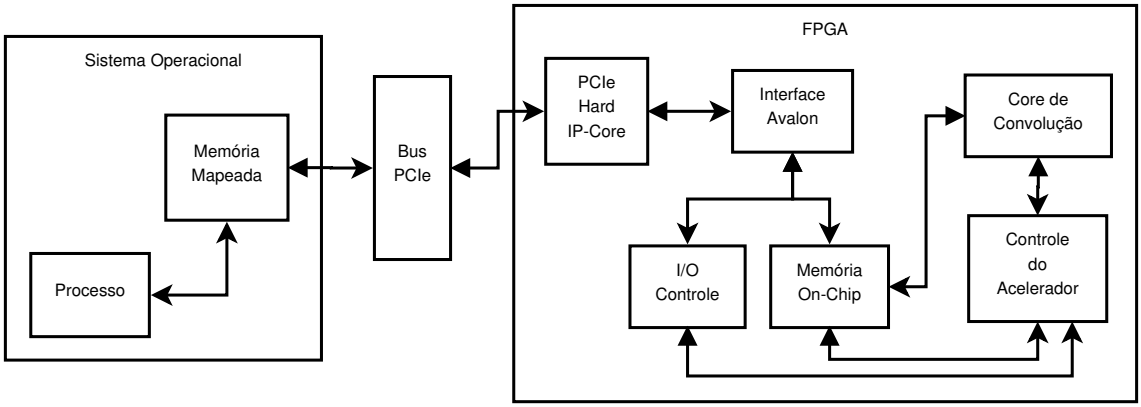
Para implementação projeto, houve a necessidade de um sistema computacional com algum tipo de suporte a interface com um FPGA. Devido à disponibilidade, foi escolhida a placa DE2i-150 fabricada pela Terasic, que contém uma CPU Intel® Atom™ N2600 e um FPGA Altera® Cyclone® IV GX, que são interligados por meio de barramento PCIe [de2]. Esta placa é basicamente um computador do tipo PC, com um processador que suporta tantos as arquiteturas x86 quanto x64. O sistema operacional a ser executado foi o Yocto Linux [1], devido à maior facilidade de integração com a placa utilizada.

[1]Site oficial: `www.yoctoproject.org`.

3.2. Arquitetura do Sistema

Em resumo, a arquitetura consiste de um processo em execução no sistema operacional, e de um circuito digital que executa alguma operação. Deve existir comunicação entre eles à medida que o processo envia dados de entrada, e recebe os dados de saída resultantes dos cálculos no acelerador. Na figura 1 vemos como isso foi estruturado, destacando os principais elementos utilizados.

Figura 1. Diagrama de blocos em alto nível do sistema.



3.3. Comunicação entre Aplicação e FPGA

Para esta parte, contamos com o suporte já existente no núcleo sistema Linux ao barramento PCIe, e com a disponibilidade da função `mmap`. Ela permite que seja feito um mapeamento entre o espaço de endereçamento de memória virtual de um processo e algum arquivo ou dispositivo do computador Tal dispositivo pode inclusive ser a memória principal: `/dev/mem`. Neste caso, sabendo que a alguns dispositivos de hardware são reservados determinados endereços de memória, faz-se possível um meio de comunicação direta entre o processo e um dispositivo como um FPGA conectado ao barramento PCIe [Corbet et al. 2005].

Uma biblioteca com rotinas para manipulação do dispositivo mapeado na memória foi desenvolvida. Fornecendo assim uma API simples para uso por alguma aplicação na transferência dos dados e no controle do acelerador. É possível com ela: criar ou desfazer o mapeamento, enviar dados de entrada, ler dados de saída, e controlar de forma básica o acelerador.

4. Aspectos de Implementação

O circuito digital de nosso acelerador pode ser funcionalmente dividido em: comunicação com o barramento PCIe, barramentos internos, controlador geral do acelerador, núcleo que implementa a convolução e memória interna. Controladores de barramento e demais componentes necessários em um hardware como o que está sendo desenvolvido são circuitos complexos. É usual no projeto digital VLSI que sejam utilizados componentes já prontos para essas funcionalidades, os IP-Cores. O *Quartus Prime Lite Edition* (na versão 16.0) foi o software de projeto para dispositivos lógicos programáveis utilizado para desenvolvimento no FPGA Cyclone® IV GX presente em nossa placa. Por meio dele, tivemos acesso e pudemos integrar alguns componentes IP necessários:

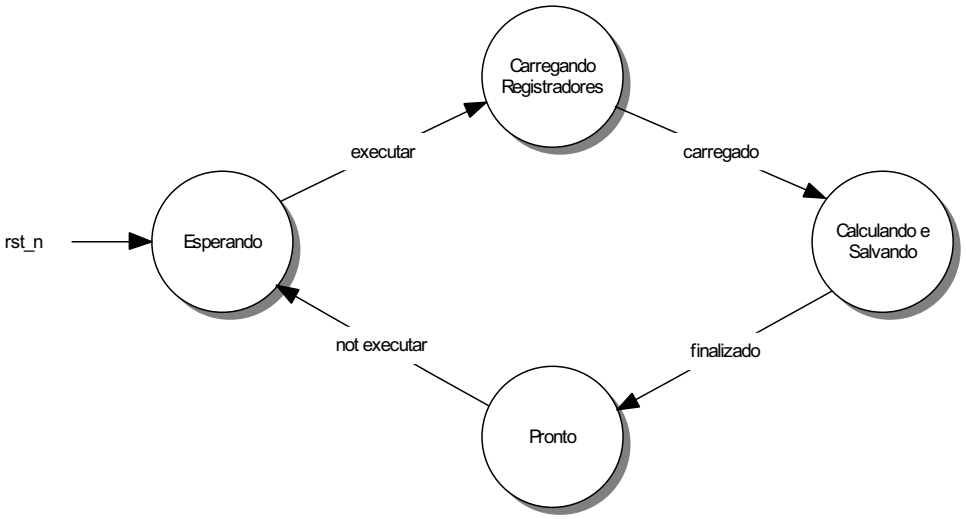
- PCI Express Hard IP: para o controlador do barramento PICe.
- Avalon Memory-Mapped Interface: as interfaces Avalon fornecem meios para interconexão de componentes na FPGA de uma forma simplificada. Neste caso, temos conexões do tipo mestre-escravo, com leituras e escritas baseadas em endereço.
- On-chip memory: a memória interna da FGPA que armazena os dados de entrada e saída dos cálculos. Sendo esta mapeada para o espaço do processo.
- Parallel input/output (PIO) core: utilizado exclusivamente para comunicação dos sinais de controle entre o acelerador e a aplicação.

Desta forma, concentramos nossos esforços no desenvolvimento das partes exclusivamente relacionadas ao aceleramento da operação, são elas o controlador do acelerador e IP-Core da operação acelerada.

4.1. Controlador

O controlador é implementado como uma máquina de estados finitos que tem como entradas: um *reset*, o sinal de comando para início ou término da operação (originado da aplicação), sinais internos para indicar carregamento de registradores com dados ou o término da operação sobre eles. Como saídas: sinais para comunicação com a memória interna (realizando operações de leitura e escrita), sinal de ativação do módulo de cálculo e um sinal que indica a conclusão da operação (destinado ao processo).

Figura 2. Diagrama de estados simplificado do controlador.



Ele espera por um sinal para execução dos cálculos, após isso, carrega dados de entrada para registradores internos, em seguida, efetua os cálculos da saída e simultaneamente salva estes valores para a memória. Ao final, vai para um estado onde envia um sinal de conclusão para o programa que o invocou, este por sua vez responde com o negativo do sinal para execução, liberando o acelerador.

4.2. Core da Operação

Neste trabalho, usaremos como exemplo a operação de convolução entre duas sequências discretas $x[n]$ e $h[n]$, que serão representadas por meio de *arrays* de números inteiros. Isto

resulta em uma nova sequência $y[n]$ conforme a seguinte equação: $y[n] = \sum_{k=0}^{N} x[n-k] * h[k]$. Esta é uma computação que deve ser efetuada para cada elemento da sequencia de entrada, gerando uma respectiva saída. Sendo assim, são necessárias $N+1$ multiplicações e N somas por elemento calculado na sequência de saída.

Aproveitando-se das recursos do hardware reconfigurável, uma oportunidade de melhoria para este caso seria expandir o circuito dos somadores/acumuladores fazendo com que os cálculos sejam efetuados em uma quantidade menor de pulsos de *clock*.

5. Conclusão

Neste trabalho analisamos de forma prática a abordagem da utilização de FPGAs para aceleração de processamento em aplicações. Isso envolveu o projeto de uma arquitetura que contou tanto com componentes de software quanto de hardware, desenvolvidos com sua integração em mente. Os objetivos alcançados foram a implementação de um meio para comunicação entre processos num sistema operacional e hardware em FPGA, além de um modelo de IP-Core customizado para a aplicação em convolução. Nosso projeto ainda encontra-se em andamento, temos como objetivos em aberto a realização de análises da performance do acelerador e melhorias na robustez de toda da arquitetura apresentada.

Referências

Terasic - de main boards - de2i-150 fpga development kit. Disponível em: `http://de2i-150.terasic.com`. Acesso em: 03 de abril de 2017.

Boutaba, R., Cheng, L., and Zhang, Q. (2012). On cloud computational models and the heterogeneity challenge. *Journal of Internet Services and Applications*, 3(1):77–86.

Corbet, J., Rubini, A., and Kroah-Hartman, G. (2005). *Linux Device Drivers*. Nutshell handbook. O'Reilly Media.

Fowers, J., Brown, G., Cooke, P., and Stitt, G. (2012). A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '12, pages 47–56, New York, NY, USA. ACM.

Hauck, S. and DeHon, A. (2010). *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*. Systems on Silicon. Elsevier Science.

Patterson, D. and Hennessy, J. (2011). *Computer Organization and Design: The Hardware/Software Interface*. The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Science.

Putnam, A., Caulfield, A. M., Chung, E. S., Chiou, D., Constantinides, K., Demme, J., Esmaeilzadeh, H., Fowers, J., Gopal, G. P., Gray, J., Haselman, M., Hauck, S., Heil, S., Hormati, A., Kim, J.-Y., Lanka, S., Larus, J., Peterson, E., Pope, S., Smith, A., Thong, J., Xiao, P. Y., and Burger, D. (2014). A reconfigurable fabric for accelerating large-scale datacenter services. *SIGARCH Comput. Archit. News*, 42(3):13–24.

Wolf, W. (2008). *Modern VLSI Design: IP-Based Design*. Prentice Hall Modern Semiconductor Design Series. PRENTICE HALL.