VDR – Uma Ferramenta de Programação Visual para um Processador de Imagens

Bolsista: Marcos R. Boschetti
marcosrb@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bloco IV
Agronomia - POA - RS
Caixa Postal: 15064 CEP 91501-970

Abstract

This paper presents a visual aid programming tool for the synthesis of a dynamically reconfigurable architecture. This tool is a graphical interface for the definition, optimization and pre-synthesis of image processing algorithms that are implemented on Dynamically Reconfigurable Image Processor (DRIP). The output is customized VHDL model to be synthesized onto FPGA components.

Resumo

Este trabalho apresenta uma ferramenta de programação visual que auxilia na síntese de uma arquitetura dinamicamente reconfigurável. Essa ferramenta constitui uma interface gráfica para a definição, otimização e pré-síntese de algoritmos de processamento de imagem que devem ser implementados sobre um processador de imagens dinamicamente reconfigurável (DRIP). O resultado final são modelos VHDL customizados prontos para serem sintetizados em componentes FPGA.

Palavras-Chave: Arquiteturas Reconfiguráveis, Processamento Digital de Imagens, Ferramenta de Síntese

1. Introdução

As arquiteturas reconfiguráveis constituem um novo paradigma no projeto e construção de sistemas digitais, oferecendo grande flexibilidade e, ao mesmo tempo, possibilidade de alto desempenho.

Os GPP`s (General Pourpose Processors) têm por característica a capacidade de executar uma vasta gama de tarefas computacionais, possuindo, portanto, um alto grau de generalidade. Contudo, para cumprir esse papel, existem alguns custos associados:

- De hardware: Em geral os GPP`s são maiores (possuem mais unidades funcionais) que o necessário, havendo desperdício de área/recursos.
- Energia: Muita potência desperdiçada com elementos que poderão não ser usados durante grande parte do processamento.

Além desses aspectos, o fato de ser um processador de propósito geral implica uma menor eficiência para a realização das tarefas. Quando se faz necessário um desempenho mais elevado para uma determinada aplicação a alternativa é a utilização de processadores com arquiteturas de propósito especial, específicos, que recebem a nomenclatura de ASP`s (Application Specific Processors). Os ASP`s caracterizam-se por serem adequados a um tipo de aplicação ou otimizados para um dado conjunto de requisitos de projeto. Em um ASP são incluídas somente as unidades funcionais e instruções necessárias à aplicação-alvo e as mesmas são altamente otimizadas para a classe específica de problemas a que se prestam.

O maior inconveniente na utilização de processadores de propósito específico reside na falta de flexibilidade. Não é possível realizar a inclusão de novas funções que estendam o conjunto original, como também não se pode atualizar a estrutura do dispositivo com o surgimento de uma nova técnica. A execução de melhorias em um sistema de propósito especial implica a substituição do mesmo.

Com o advento de dispositivos de lógica programável com densidade de centenas de milhares de portas por chip, como os componentes FPGA [BRO92] mais modernos, surgiu a possibilidade da construção de arquiteturas reconfiguráveis que se propõem a unir um desempenho considerável com uma maior flexibilidade. Sendo possível reconfigurar as unidades funcionais para a execução de diferentes tarefas, passa a existir um maior equilíbrio entre eficiência e generalidade.

O projeto de uma arquitetura reconfigurável distingue-se do projeto de outros sistemas digitais, pois há a necessidade de contar com ferramentas especialmente projetadas que considerem a arquitetura-alvo e o problema a ser tratado. O sistema de síntese deve, portanto, ser especificado e construído de acordo com os aspectos particulares da aplicação, de forma a proporcionar facilidades para o usuário/desenvolvedor e garantir a validação do projeto.

Este trabalho apresenta e descreve os esforços realizados pelo bolsista de iniciação científica durante o desenvolvimento de uma ferramenta chamada VDR (Visual interface for Dynamic Reconfiguration) que auxilia na síntese de um processador de imagens reconfigurável. Serão mostrados os vários aspectos referentes à arquitetura-alvo e aplicação, as características e funcionalidades da ferramenta, o fluxo de projeto no qual a ferramenta está inserida e, finalizando, conclusão e trabalhos futuros.

2. Aplicação

Existe uma classe restrita de aplicações nas quais verificam-se as características básicas que sugerem sua implementação sob uma arquitetura reconfigurável, essas características são:

- Regularidade: devem realizar um reduzido conjunto de operações básicas repetidamente.
- Alta concorrência: deve haver a possibilidade de executar simultaneamente um grande número de operações.
- Pequena granularidade de dados: os operandos são formados por um pequeno número de bits.

Como exemplo de aplicações que se enquadram nessas especificações, podem ser citadas: ciframento/deciframento, emparelhamento de strings, aritmética especializada e processamento digital de imagens.

O processamento digital de imagens (PDI) pode beneficiar-se com a abordagem reconfigurável, uma vez que operações simples, execução massivas de blocos básicos do programa e operandos de pequeno tamanho, são características que definem os algoritmos de PDI.

3. A arquitetura-alvo

A arquitetura-alvo da ferramenta é um processador de imagens chamado DRIP (Dynamically Reconfigurable Image Processor). O DRIP constitui a versão reconfigurável do processador de vizinhança NP9 [ADA97]. Um processador de vizinhança processa uma imagem de entrada gerando uma nova imagem, na qual cada pixel de saída é função de seu correspondente na imagem de entrada e seus vizinhos mais próximos.

O DRIP simula um processador matricial [FOU87] , sendo composto por uma matriz bidimensional de células interconectadas também chamadas processadores elementares (PE`s).

3.1 O Processador Elementar

O processador elementar do DRIP/NP9 é funcionalmente simples e pode executar apenas duas operações básicas: soma (ADD), representando a classe dos algoritmos lineares e máximo (MAX), representando a classe dos algoritmos não lineares.

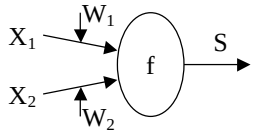

Figura 1: Modelo do processador elementar

Cada PE recebe dois pixels de entrada, a cada entrada existe um peso inteiro (-1, 0 ,1) associado. Sob essas condições é possível implementar 8 funções distintas, desconsiderando funções equivalentes ou simétricas em relação às entradas, por exemplo, MAX(X1 * 0, X2 * 1) é equivalente a MAX(X1 * 1, X2 * 0) com inversão na ordem das entradas. O conjunto das funções forma uma biblioteca de funções básicas, utilizada na implementação dos algoritmos.

Original Operation	Function
add(0,0), max(0,0)	Zero
add(0,X2),add(X1,0)	X
add(0,-X2),add(X1,0)	-X
add(X1,X2), add(X1,X2)	Xa + Xb
add(X1,-X2), add(-X1,X2)	Xa – Xb
max(0,X2), max(X1,0)	If positive(X) then X else 0
max(0,-X2), max(-X1,0)	If negative(X) then X else 0
other possibilities of max	Max(Xa, Xb)

Tabela 1: Funções customizadas do PE

4. A Ferramenta VDR

A VDR é uma ferramenta de auxilio à síntese do DRIP. A interação com o usuário ocorre através de uma interface na qual é possível definir algoritmos de processamento digital de imagens (uma janela VDR é mostrada na figura 2). A ferramenta constitui um *front end* do sistema de reconfiguração baseado no processador DRIP. Espera-se, num primeiro momento, melhorar a produtividade na especificação de algoritmos de processamento de imagens. A interface compreende uma representação em alto nível da arquitetura NP9/DRIP e para seu desenvolvimento utilizou-se a linguagem TCL/TK.

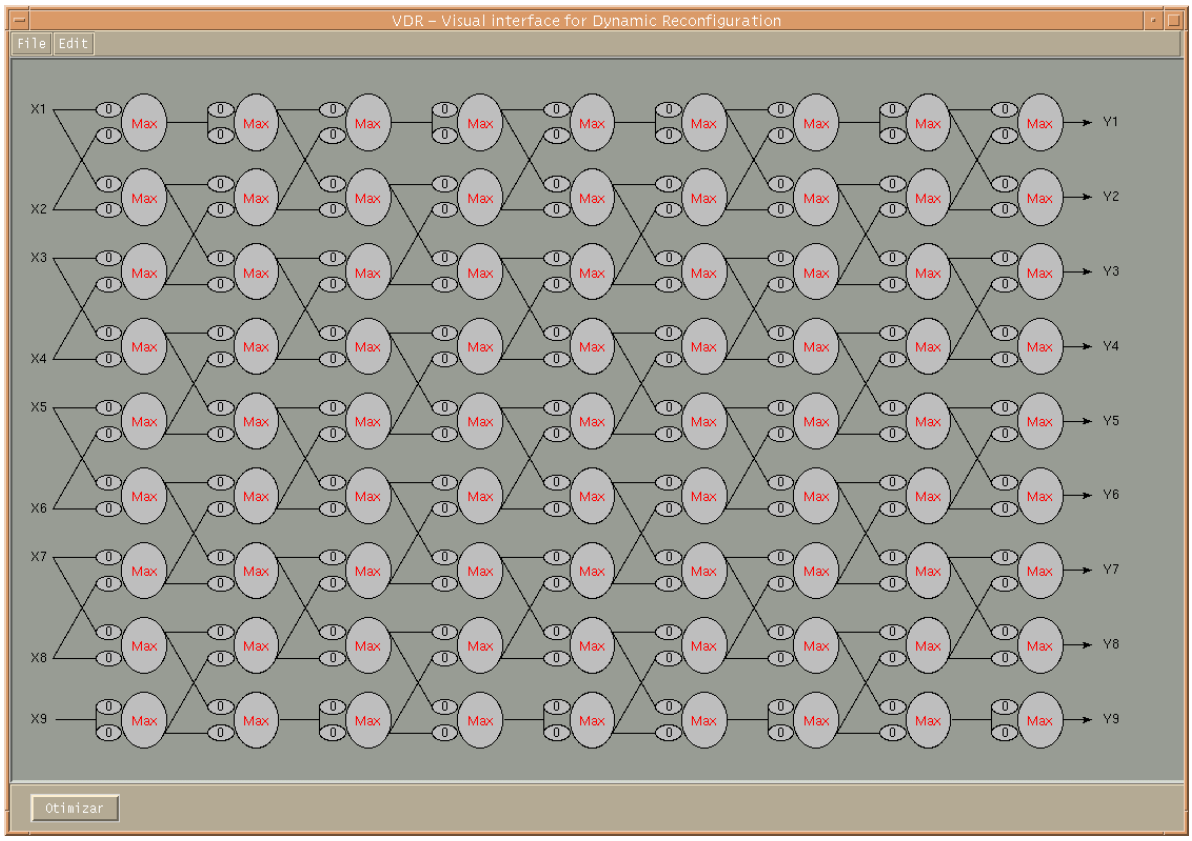

Figura 2: Interface da ferramenta representando os 81 PE`s

4.1 A Linguagem TCL/TK

A linguagem TCL/TK [OUS 94] nasceu na Universidade da California em Berkeley com o intuito de facilitar a construção de ferramentas para projeto de circuitos integrados. Trata-se de uma linguagem de propósito geral projetada para ser, ao mesmo tempo, de fácil utilização e poderosa. TCL e TK são, na verdade, entidades distintas. TK é utilizada para criar componentes que podem ser parte de um programa ou mesmo uma aplicação inteira. TCL contém as instruções e características de uma linguagem de programação completa, sua maior utilização é no controle de componentes TK. Dessa forma, TCL e TK proporcionam um sistema de programação completo para o desenvolvimento e utilização de interfaces gráficas.

O estilo de programação TCL/TK e suas características se adaptaram perfeitamente as necessidades existentes para a implementação da interface VDR. A linguagem é portável e flexível, disponível sem custos, e há uma grande quantidade de informações técnicas disponíveis.

4.2 A Interface VDR

Uma ferramenta de auxílio à síntese de uma arquitetura reconfigurável deve refletir o mais diretamente possível as características da arquitetura-alvo. Dessa maneira, a interface VDR apresenta uma representação gráfica dos 81 PE`s, dispostos numa matriz 9x9 que compõem o pipeline do processador DRIP (como mostrado na figura 1).

Toda a funcionalidade de um PE é completamente configurável, através de cliques é possível alterar os pesos de entrada e a função executada pelo PE.

O código foi escrito sobre uma plataforma Unix X-Window, mas é portável para qualquer sistema suportado por TCL/TK.

4.3 Fluxo de Projeto

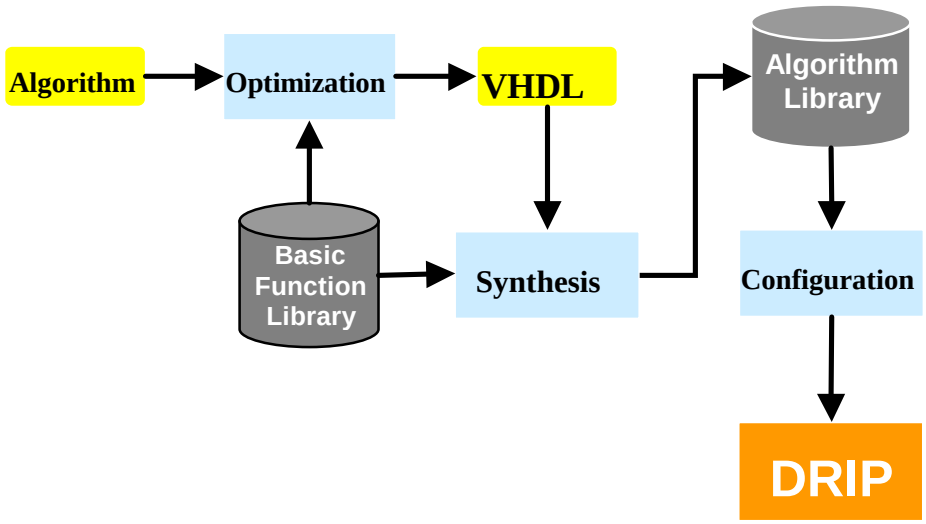

Figura 3: Esquemático do fluxo de projeto

A configuração do DRIP inicia-se dentro da VDR. A ferramenta contém três estágios do fluxo de projeto. Em um primeiro passo, um algoritmo de processamento digital de imagens é visualmente definido sobre a estrutura do DRIP, utilizando-se a interface gráfica. A seguir, a VDR mapeia o algoritmo para um grafo orientado acíclico e o otimiza suprimindo funções redundantes e desnecessárias, gerando um novo grafo de fluxo de dados, que representa o algoritmo.

A otimização remove PE`s que, por exemplo, apenas propagam o valor zero, fazendo a eliminação que economiza recursos do FPGA em um futuro mapeamento e faz surgir a possibilidade de redimensionar outros elementos como os registradores intermediários do pipeline.

Após a otimização, cujas rotinas encontram-se em uma biblioteca de funções desenvolvidas na linguagem C, a ferramenta oferece realimentação visual ao usuário/desenvolvedor apresentando a rede de PE`s otimizada (figura 4). Neste exemplo em particular o resultado mostrado não representa a otimização de nenhum algoritmo de PDI específico, o objetivo é mostrar as transformações que VDR pode provocar em um determinado grafo de fluxo de dados.

O terceiro e último passo é a geração de modelos VHDL a partir do grafo de fluxo de dados, utilizando como componentes básicos as células customizadas existente na biblioteca de funções básicas do DRIP.

Está em desenvolvimento também uma biblioteca de tratamento de grafos que completa o processo de otimização. A biblioteca contém objetos C++ que implementam soluções ao problema do escalonamento em arquiteturas reconfiguráveis, alguns algoritmos de escalonamento como ASAP (As Soon As Possible) e ALAP (As Late As Possible) já foram construídos e estão presentes no conjunto de funções.

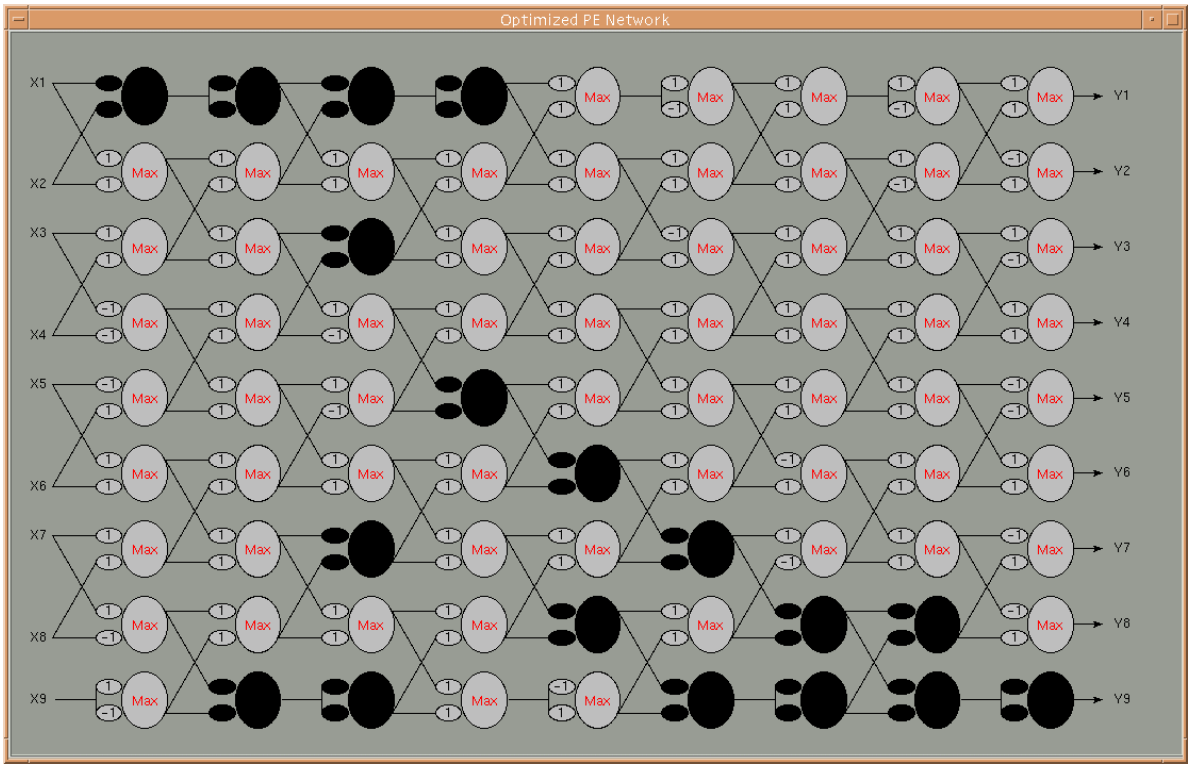

Figura 4: rede otimizada destacando os PE`s eliminados

5. Conclusão e Trabalhos Futuros

Este trabalho apresentou o contexto e os principais aspectos de uma ferramenta gráfica chamada VDR, que funciona como um *front end* para o sistema de configuração do DRIP. O foco está na aceleração do processo de síntese através da otimização e pré-síntese de algoritmos de processamento de imagem. Os esforços de projeto da ferramenta VDR relacionam-se com o desenvolvimento de uma ferramenta de programação que reflita a estrutura do DRIP e considere as necessidades de síntese de um sistema reconfigurável. A interface e o sistema de otimização estão concluídos. Os próximos passos vão no sentido de permitir a criação automática dos modelos VHDL e o desenvolvimento de um ambiente que possibilite a execução/simulação de algoritmos internamente à ferramenta VDR. Espera-se realizar, com a continuidade do trabalho, maiores contribuições no campo de pesquisa das arquiteturas reconfiguráveis.

6. Bibliografia

[ADA97] ADÁRIO, A. M. S.; CÔRTES, M. L.; LEITE, N. J. "A FPGA Implementation of a Neighborhood Processor for Digital Image Applications" In: 10 Brazilian Symposium on Integrated Circuit Design, Ago 1997. ***Proceedings...***, 1997 p. 125-134.

[ADA99] ADÁRIO, A. M. S, ROEHE, E. L., BAMPI, S. "Dynamically Reconfigurable Architecture for Image Processor Applications". In: 36 Design Automation Conference, New Orleans, June 1999. ***Proceedings***..., 1999, *pg 623-628*.

[FOU87] FOUNTAIN, T.J. ***Processor Arrays***: Architecture and Applications. London: Academic Press, 1987.

[OUS94] OUSTERHOUT, J. K. ***Tcl and Tk Toolkit***. Addison Wesley, 1994.

[BRO92] BROWN, Stephen. D.; FRANCIS, Robert J.; ROSE, Jonathan; VRANESIC, Zvonko G. Field-Programmable Gate Arrays. Kluwer Academic Publishers, 1992