Evento	Salão UFRGS 2018: FEIRA DE INOVAÇÃO TECNOLÓGICA DA UFRGS - FINOVA
Ano	2018
Local	Campus do Vale - UFRGS
Título	Implementação de Arquitetura Dedicada à Execução Paralela de Programas Funcionais Puros
Autor	RICARDO DE ARAÚJO COELHO
Orientador	GABRIEL LUCA NAZAR

UFRGS
SEDETEC

RESUMO

[máximo duas páginas]

TÍTULO DO PROJETO: Implementação de Arquitetura Dedicada à Execução Paralela de Programas Funcionais Puros

Aluno: Ricardo de Araujo Coelho
Orientador: Gabriel Luca Nazar

RESUMO DAS ATIVIDADES

1. Introdução:

   Com o aumento da frequência e dissipação de potência atingindo seus limites físicos, a indústria passou a desenvolver processadores com múltiplos núcleos como estratégia para ganhar velocidade de processamento. Entretanto, programar esses processadores não é uma tarefa fácil, porque desenvolvedores necessitam criar e sincronizar threads manualmente.

   Os algoritmos escritos com o paradigma de linguagens imperativas descrevem um procedimento sequencialmente, alterando o estado da memória (uso de variáveis). Essa mudança de estado gera efeitos colaterais que necessitam tratamento especial quando se procura rodar o algoritmo em paralelo, tornando a programação paralela difícil.

   Já as linguagens puramente funcionais não apresentam esse efeito colateral de modificação de estado. Elas têm a propriedade de uma função sempre retornar o mesmo valor se for aplicada a um mesmo argumento.

   Os algoritmos escritos com esse tipo de linguagem, ao invés de possuírem variáveis, fazem uso intenso de funções recursivas e manipulação de listas, tendo valores de retorno combinados com o retorno de outras funções independentes. Sendo assim, uma linguagem puramente funcional tem pontos de possibilidade de paralelismo trivialmente identificáveis.

   Apesar disso, as linguagens funcionais são mais complexas de serem executadas em um processador tradicional (que desde sempre foi projetado e otimizado pensando-se em execução de código imperativo), necessitando normalmente uso de pilhas extensas e cópias de valores que acabam aumentando em demasia o tempo de processamento. Assim, seu apelo é diminuído pela maior velocidade de processamento dos códigos escritos com linguagens imperativas.

   O acelerador ACQuA(Active Call QUeue Architecture) é proposto tendo o objetivo de explorar o paralelismo inerente em linguagens puramente funcionais automaticamente paralelizando aplicações de função independentes. Com o suporte direto da arquitetura, um desenvolvedor pode escrever programas em linguagem puramente funcional e ter o benefício do paralelismo em aplicações de funções de forma transparente, sem necessidade de recorrer a abstrações da linguagem para explicitamente criar, sincronizar e descrever a comunicação para múltiplas threads.

2. Atividades realizadas:

O meu trabalho nesse período de bolsa foi desenvolver uma versão completa, funcional e eficiente da arquitetura em VHDL para que seja sintetizado um hardware. Para isso foi necessário:

1. criar um compilador para se automatizar a geração do código compatível com o acelerador: O compilador tem a função de mapear as instruções criadas pela representação intermediária ($ACQuA_{IR}$) em código executável por um processador convencional embutido dentro do núcleo de processamento da arquitetura ACQuA;
2. realizar manutenção e extensão de funcionalidade na descrição do hardware para, de fato, conseguir executar o código gerado pelo compilador: Foi implementada alocação e liberação de contextos de memória, funções de alta ordem, entre outros).

3. Objetivos atingidos:

Foi atingido o objetivo de se construir um caminho completo entre a criação de um programa em linguagem funcional e sua execução na arquitetura ACQuA.

Utilizando-se o simulador de FPGA modelsim foi possível obter algumas medidas de tempo de execução para diferentes quantidades de núcleos configuradas para o acelerador.

4. Resultados obtidos:

Realizando-se medidas de execução com o modelsim para diferentes quantidades de núcleos, obteve-se um speedup aparentemente linear para execução de um programa específico que calcula o n-ésimo número da sequência de fibonacci. Sendo 1 o speedup da execução com um único núcleo, utilizando-se 16 núcleos de processamento o speedup ficou em torno de 18.

5. Conclusão:

Pelo experimento feito, algoritmos que tenham maior possibilidade de processamento paralelo tendem a ter um bom ganho de performance com a utilização do acelerador ACQuA.

Algumas funções adicionais ainda devem ser implementadas para se fazer mais testes com diferentes algoritmos.

É prevista a otimização de código compilado e implementação de funcionalidades como *map* (já definidas conceitualmente para a arquitetura), que devem gerar uma execução do programa mais eficiente.