Transformation of Sequential Software into Parallel FPGA Hardware: A Case Study using the SPEC CPU 2006 Benchmark

Dr. Raymond R. Hoare II
Concurrent EDA, LLC, rayhoare@ConcurrentEDA.com

Abstract

FPGAs have proven themselves be faster than sequential processors for a variety of applications. This is especially true for applications in which the parallelism is extracted and an FPGA hardware implementation is created by an expert.

What happens when the application has been written in a sequential language, like C, and compiled into a binary? Can a high-performance FPGA hardware implementation be created from the sequential binary? Can parallelism be extracted and utilized?

In this paper we examine the SPEC CPU2006 benchmarks and show that a high-speed FPGA hardware implementation is possible and that it can be automated using Concurrent EDA's tools.

This paper presents a study performed by Concurrent EDA using its *Embedded Adrenaline* design automation tool that dramatically improves the performance of software through the use of FPGA accelerators. The core concept is to identify the bottlenecks in the code and accelerate them by creating an application-specific FPGA co-processor. The objective is to provide as much performance improvement as possible within a given FPGA's resource constraints.

The novelty of our approach is that we exploit both static and dynamic knowledge of the software. That is to say that we examine the software binary at the assembly level and during execution. Recall that a desktop processor can execute over a billion instructions per second. There is a great deal of knowledge that is gained from dynamically analyzing the software. This paper presents a summary of our findings from executing the CPU 2006 benchmarks.

SPEC CPU 2006 Benchmarks

The following is a list is a partial list of benchmarks in the SPEC CPU 2006 benchmark suite. For the final submission, this abstract will provide an overview of our methodology and a summary of the results that we obtained.

- 401.bzip2 C Compression Julian Seward's bzip2 version 1.0.3, modified to do most work in memory, rather than doing I/O.
- 429.mcf C Combinatorial Optimization Vehicle scheduling. Uses a network simplex algorithm (which is also used in commercial products) to schedule public transport.
- 445.gobmk C Artificial Intelligence: Go Plays the game of Go, a simply described but deeply complex game.
- 456.hmmer C Search Gene Sequence Protein sequence analysis using profile hidden Markov models (profile HMMs)
- 462.libquantum C Physics / Quantum Computing Simulates a quantum computer, running Shor's polynomial-time factorization algorithm.
- 464.h264ref C Video Compression A reference implementation of H.264/AVC, encodes a videostream using 2 parameter sets. The H.264/AVC standard is expected to replace MPEG2
- 482.sphinx3 C Speech recognition A widely-known speech recognition system from Carnegie Mellon University

THISIN ITS FINAL FORM WILL CONTAIN THE FOLLOWING SECTIONS

Methodology

A description of our methodology.

Computational Complexity

A description of the computational complexity of the benchmark code that collective consumes 80% of the execution time.

Extracted Parallelism

A description of the parallelism extracted using static and dynamic analysis.

Performance Improvement

A description of the performance improvement that is achievable using FPGAs.

Conclusions