A Hardware-software integrated solution for improved Single-Instruction Multi-Thread processor efficiency

Michael Steffen

Ph. D. Final Defense

Friday April 6 at 10am in 3043 ECpE Addition

ABSTRACT

This thesis proposes using an integrated hardware-software solution for improving Single-Instruction Multiple-Thread branching efficiency. Unlike current SIMT hardware branching architectures, this hardware-software solution allows programmers the ability to fine tune branching behavior for their application or allow the compiler to implement a generic software solution. To support a wide range of SIMT applications with different control flow properties, three branching methods are implemented in hardware with configurable software instructions. The three branching methods are the contemporary Post-Dominator Re-convergence that is currently implemented in SIMT processors, a proposed Hyper-threaded SIMT processor cores for maintaining statically allocated thread warps and a proposed Dynamic Micro-Kernels that modified thread warps during run-time execution. Each of the implemented branching methods have their strengths and weaknesses and result in different performance improvements depending on the application. SIMT hyper-threading turns a single SIMT processor core into multiple virtual processors. These virtual processors run divergent control flow paths in parallel from threads in the same warp. Controlling how the virtual processor cores are created is done using a per-warp stack that is managed through software instructions. Dynamic Micro-Kernels creates new threads at run-time to execute divergent control flow paths instead of using branching instructions. A spawn instruction is used to create threads at run-time and once created are placed into new warps with similar threads follow the same control flow path.

This thesis's integrated hardware-software branching architectures are evaluated using different realistic benchmarks with varying control flow divergence. Synthetic benchmarks are also used for evaluation and are designed to test specific branching conditions and isolate common branching behaviors. Each of the hardware implemented branching solutions are tested in isolation using different software algorithms. Algorithms are designed for general purpose use or to target specific types of branching conditions. Results shows improved performance for divergent applications and using different software algorithms will affect performance.