Abstract

This thesis describes the design of a code generator for a configurable programmable platform implementing an ultra-wide *Single Instruction Multiple Data* (SIMD) architecture with explicit datapaths. The distinguishing characteristic of this architecture is a wide array of *Processing Elements* (PEs) that exploit parallelism by processing many operations concurrently. Therefore, a high throughput can be achieved at a low clock frequency and thus low voltage, thereby with a high energy efficiency.

A LLVM-based compiler that targets this architecture exists, but lacks support for configurable explicit datapaths. This compiler is implemented completely within LLVM because it provides auto-vectorization that does support a configurable array size

This work has a focus on extending the compiler with explicit bypassing capabilities. Explicit bypassing consists of two compiler optimizations, e.g. operand forwarding and dead result elimination. With operand forwarding, a value or result of an operation is forwarded from a pipeline stage to the *Instruction Decode* stage, thereby, bypassing the *Register File* (RF). When all consumers of a variable obtain it using forwarding, it is never read from the RF. Therefore, that variable does not need to be stored in a register file, since it is not read from it anyway. Dead result elimination (which is possible with explicit bypassing) consists of avoiding such redundant store accesses. With implicit bypassing the hardware performs bypasses automatically, while it is the compilers responsibility to find and allocate bypasses with explicit bypassing.

Compiling with explicit datapaths has been an active topic of research and several architectures face similar challenges. The *Transport Triggered Architecture* (TTA) effectively faces the same challenge, except that compilation for TTAs is even more challenging because the datapaths is exposed in the instruction set. What TTAs have in common with code generation for SIMD with explicit datapaths is that explicit reads and writes are explicitly stated in the instruction set.

Several approaches to support explicit datapaths for the target SIMD architecture within LLVM have been ranked on implementation effort and expected results. Subsequently, one approach has been implemented and the efficiency of the generated code was accessed based on simulation results.

With explicit bypassing, around 40% of the accesses can be avoided. Since the RF consumers around 35% of the total energy consumption and around 40% of the communication with the RF can be avoided, an estimated improvement of around 15% follows.

Keywords: Compilers, SIMD, LLVM, Embedded Systems, Power Efficiency, Explicit Bypassing