Evaluation of Stream Virtual Machine on Raw Processor

Jinwoo Suh, Stephen P. Crago, Janice O. McMahon, Dong-In Kang
University of Southern California
Information Sciences Institute

Richard Lethin
Reservoir Labs

March 26, 2007
Overview

- Stream Virtual Machine
  - High Level Compiler and Low Level Compiler

- Raw Processor

- Signal Processing Applications and Implementation Results
  - Matrix Multiplication
  - FIR bank
  - Ground Moving Target Indicator

- Conclusion
Stream Virtual Machine

- Stream processing processes input stream data and generates output stream data
  - Exploits the properties of the stream applications such as parallelism and throughput-oriented
- A uniform approach for stream processing for multiple input languages and multiple processor architectures
  - Developed by Morphware forum (morphware.org)
- Centered around Stable Architecture Abstraction Layer
- Part of the layer is Stream Virtual Machine (SVM)
- Consists of three major components
  - High Level Compiler
  - Low Level Compiler
  - Machine model
Advantages of SVM Framework

- **Efficiency**
  - Compilers can generate efficient code by exposing communication and computation to compiler.

- **Portability**
  - Support for multiple languages and architectures in a single framework

- **Low development cost**
  - Adding new language
    - Only the high level compiler needs to be written.
  - Adding new architecture
    - Only the low level compiler needs to be written.
  - Programming applications
    - Ex. High level compiler provides parallelization
Raw Handheld

- Raw processor was developed by MIT
- Raw handheld board was developed by MIT and ISI-East
- A Raw chip contains 16 tiles (cores) with 2D mesh networks
- Each tile is MIPS-based RISC processor with floating point unit
- Network port is mapped to a register that saves communication time
High Level Compiler

- R-Stream being developed by Reservoir Labs (reservoir.com)
  - Compile C code to SVM APIs

- Easy to program
  - Input code is normal C code
  - No explicit parallelization is needed

- Portability
  - The same code works on several architectures.

- Generally good parallelization capability
  - Able to parallelize up to all tiles for some cases.

- Good performance for some codes
  - TDE stage in GMTI performance is about 1/3 of hand-assembled code.
Low Level Compiler was developed as a form of library and C compiler
- C compiler for Raw developed by MIT
- Library for SVM developed by ISI-East

- Easy and quick solution
- Provides a reasonably good performance
- Very useful in quick assessment of SVM framework
Benchmark Implementations on Raw

Ground Moving Target Indicator (GMTI)
(Compact radar signal processing application, by Reservoir Labs)

Matrix multiplication and FIR bank

R-Stream 2.1 (Reservoir Labs)

SVM API Code

Raw C Compiler

SVM Library

Hand-optimization

HLC

LLC

* Results show current status of the whole tool chain in SVM framework

†Currently achieved using hand coding

* Results show potential performance†
Matrix Multiplication Implementation

- Hand coded using the SVM API (not HLC-generated code)

Cost analysis and optimizations

- Full implementation
  - Full SVM stream communication through a dynamic network

- One stream per network
  - Each stream is allocated to a network.

- Broadcast
  - With broadcasting by switch processor
  - Communication is off-loaded from compute processor.

- Network ports as operands
  - Raw can use network ports as operands
  - Reduces cycles since load/store operations eliminated
Matrix Multiplication Results

- **Number of cycles per multiplication-addition pair**
- **Lower bound = 2**
  - Multiplication
  - Addition

![Graph showing number of cycles vs. number of words per communication]

- Best obtained results = 2.23
- Lower bound = 2
FIR Banks

- Multiple FIR filters specified by Lincoln Lab
- Implemented by using radix-4 FFT, multiplication, and radix-4 IFFT

Optimizations using hand-assembly in core operations

- Minimize pipeline bubbles
  - Manual instruction scheduling
- Prevent register spilling
  - Prone to this problem since radix-4 FFT requires more registers
  - Minimizing register requirement
  - Code expansion
- Minimize address calculation
  - Using offset
  - Duplicated and rearranged twiddle factors
- Minimize data copy operation
  - Reverse the order of processing: back to front
FIR Bank Results

- **Definitions**
  - LB (UB): lower (upper) bound based on the number of floating point operations
  - ILB (IUB): lower (upper) bound based on the number of floating point operations and load/store instructions
  - Hand Optimization: hand-assembly work results
  - Compiler Optimization: only compiler optimization was done

- **One FFT-multiplication-IFFT**
  - For 64 sample data

![Bar chart showing throughput for different optimization techniques.](chart.png)
- Detects targets from radar signal
- Consists of 7 stages
- Used both high level compiler and low level compiler

High parallelization in many stages

On other stages, lower parallelization
- due to R-Stream parallelization policy, software task pipeline use, and hard-to-parallelize code
- Reservoir is working on a new parallelization policy in new R-Stream version

<table>
<thead>
<tr>
<th>Tile</th>
<th>SM/SP 0</th>
<th>SM/SP 1</th>
<th>SM/SP 2</th>
<th>SM/SP 3</th>
<th>SM/SP 4</th>
<th>SM/SP 5</th>
<th>SM/SP 6</th>
<th>SM/SP 7</th>
<th>SM/SP 8</th>
<th>SM/SP 9</th>
<th>SM/SP 10</th>
<th>SM/SP 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*SM: secondary master
SP: stream processor
Bars represent kernel executions or primary master executions
Conclusion

- Assessed SVM on Raw processor by implementing benchmarks
  - GMTI: shows full path from high level compiler to hardware execution
    - Some stages show good performance
    - Other stages show room for improvement
  - Matrix multiplication and FIR bank: show high fraction of peak performance with optimizations
    - Current performance is reasonably good
    - Identified optimizations to be included in compilers

- Two level approach of the stream virtual machine has a potential for performance, portability, and low development cost