



## Streaming Virtual Machine and Two-Level Compilation Model for Streaming Architectures and Languages

Peter Mattson Richard Lethin Vassily Litvinov  
Reservoir Labs, Inc.

François Labonté Ian Buck  
Christos Kozyrakis Mark Horowitz  
Stanford University

## Credits

- **Streaming Virtual Machine specification**
  - Peter Mattson, William Thies, Lance Hammond, and Mike Vahey (primary authors)
  - University of Southern California Information Sciences Institute, University of Texas at Austin, Georgia Institute of Technology, IBM Austin Research Laboratory, Reservoir Labs (additional contributions)

- **Morphware Forum** [www.morphware.org](http://www.morphware.org)



- **DARPA Polymorphous Computing Architectures program** F03602-03-C-0033



This presentation is based in part on the paper *The Stream Virtual Machine* by François Labonté, Ian Buck, Peter Mattson, Christos Kozyrakis, and Mark Horowitz, presented at PACT 2004.

## Motivation: Hardware Inflection Point

- **Going out, because they get negative marginal performance/power:**
  - Higher clock rates and longer instruction pipelines
  - Higher degrees of instruction level parallelism and dynamic instruction issue
  - Increasing depth of cache hierarchy on chip
- **Coming in, because they expose the latent capability of next generation silicon and are power-efficient:**
  - Chip multiprocessing, tiled architectures (e.g., Monarch, Smart Memories, TRIPS, Raw...)
  - Wide SIMD/VLIW within the tile (e.g., Imagine, TRIPS, Merrimac)
  - Explicit software control of on-chip memory, with emphasis on DMA

## Streaming Architectures

- **stream input data into local memories**
- **stream output data to global or other local memories**
- **The potential of this new streaming architecture:**
  - Merrimac predicts 150 GFlops/Watt (single precision, L=90um) dynamic power consumption achieved performance.

## Motivation: Stream Programming Model

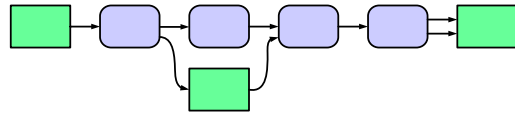
### Stream computation

- separated into
  - computational kernels
  - communication streams
- kernels compute in parallel on individual data elements

### Key application areas

- DSP
- image processing
- wireless
- games
- scientific

## Stream Programs Expose Parallelism and Locality



### Exposed locality:

- within a kernel (local data)
- stream locality (spatial)

### Exposed parallelism:

- data-level parallelism across stream elements
- task parallelism across kernels

Fits data intensive applications with regular communication patterns

## Languages for Stream Programming

- C
- HPF; Fortran
- ZPL
- MATLAB
- Brook
- StreamIt
- (your favorite data parallel programming language here)

## Sounds Like M\*N Problem?

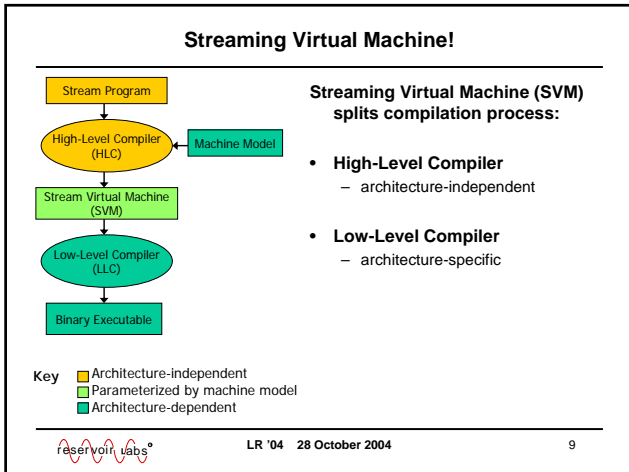
### Would like

- coverage of languages, architectures, legacy apps
  - but writing a new compiler for each (language, architecture) pair is impractical

### Other issues

- interoperability among languages
- debugging, profiling, . . .
- portability
  - would like to avoid re-grouping into kernels for each architecture
- deployment of new architectures

And the solution is . . .



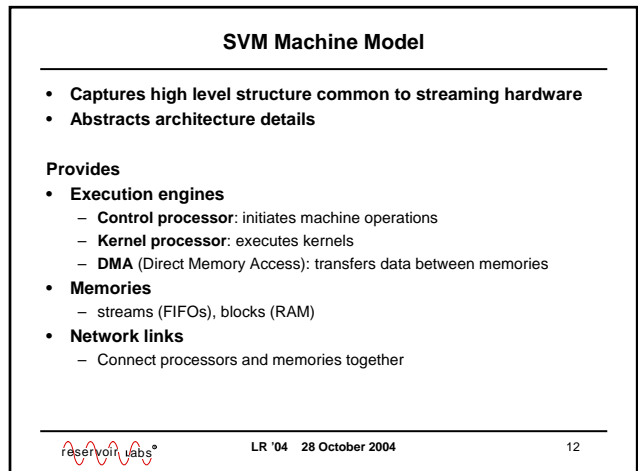
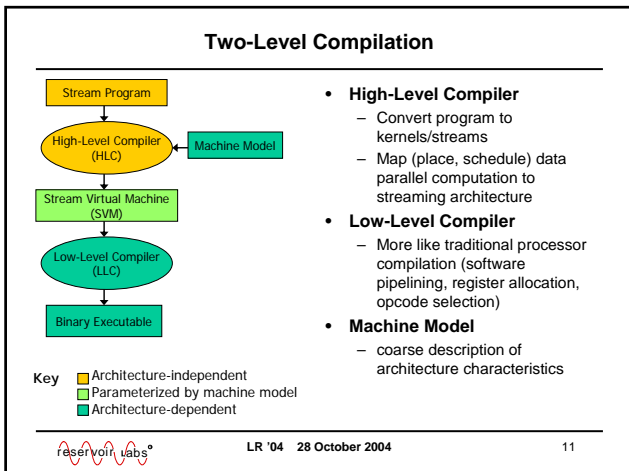
### Outline

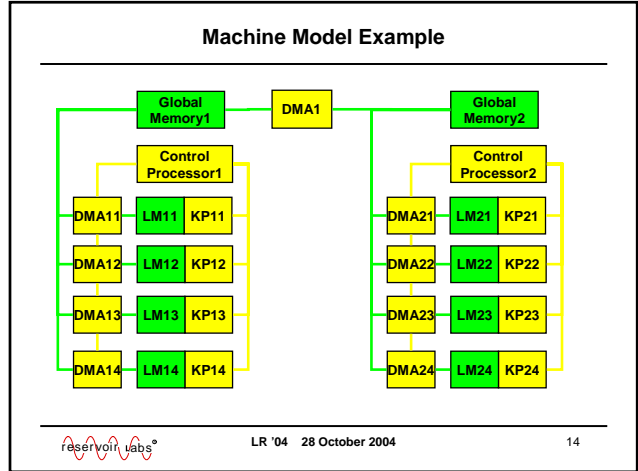
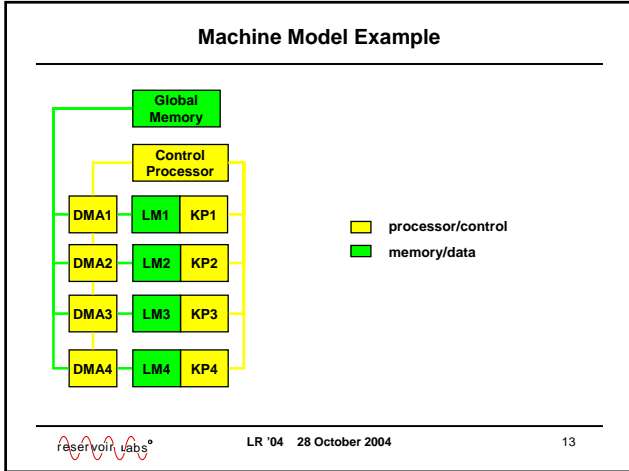
---

- ✓ Motivation
- **Streaming Virtual Machine**
  - Two-level compilation architecture
  - Machine model
  - SVM interface
- **Issues in two-level compilation**
- **Implementation status**
- **Conclusions**

---

reservoir labs<sup>®</sup>      LR '04 28 October 2004      10





### SVM: API in C

- bootstrap LLC development (e.g. gcc + library implementation)

Functions to:

- **allocate local memories**  
svm\_streamInit, svm\_blockInit
- **initialize kernels**  
svm\_kernellnit
- **initialize DMA processors**  
svm\_moveS2SInit, svm\_moveB2BInit,  
svm\_stridedGatherB2BInit, svm\_indexedScatterB2BInit
- **add dependencies**  
svm\_kernelAddDependence
- **run!**  
svm\_kernelRun

reservoir labs®      LR '04 28 October 2004      15

### Example of SVM code

```

// Load unfiltered data from global memory
svm_moveB2BInit(&load1, DMA1, inputB1, unfilterB1);
svm_kernelRun(&load1);
// Filter data in local memories after it has been loaded
svm_kernellnit(&filter1, Kernel1, unfilterB1, filterB1);
svm_kernelAddDependence(&filter1, &load1);
svm_kernelRun(&filter1);
// Compress data in local memories after it is filtered
svm_kernellnit(&compressor1, Kernel1, filterB1, compressB1);
svm_kernelAddDependence(&compressor1, &filter1);
svm_kernelRun(&compressor1);
// Store compressed data to global memory
svm_moveB2BInit(&store1, DMA1, compressB1, outputB1);
svm_kernelAddDependence(&store1, &compressor1);
svm_kernelRun(&store1);

```

The diagram shows a vertical flow of data. At the top, a box labeled 'Global Memory' has an arrow pointing down to a box labeled 'Filter'. Below the filter is a box labeled 'Kernel'. Below the kernel is a box labeled 'Compressor'. Below the compressor is a box labeled 'Store'. Arrows indicate the flow of data from Global Memory to Filter, then to Kernel, then to Compressor, and finally to Store. A dashed line separates the Filter, Kernel, and Compressor boxes from the Store box.

reservoir labs®      LR '04 28 October 2004      16

## Outline

---

- ✓ **Motivation**
- ✓ **Streaming Virtual Machine**
  - ✓ Two-level compilation architecture
  - ✓ Machine model
  - ✓ SVM interface
- **Issues in two-level compilation**
- **Implementation status**
- **Conclusions**

## Issues in Two-Level Compilation

---

- **Machine Model framework needs to be**
  - general across stream architectures
  - yet accurate so compiler can base decisions on it
- **SVM needs to be**
  - general enough to handle multiple architectures
  - yet allow efficient implementation on stream architectures
- **Need behavioral spec, not just type interfaces for SVM**
  - underspecified feature is likely to be interpreted differently by HLC vs. LLC
- **API-like nature of SVM may obscure high-level information for LLC**

## Implementation Status

---

- **SVM specification**
  - bulk of specification has been completed
- **R-Stream™ as HLC**
  - input: C with programmer annotations
  - output: SVM code
  - proof of concept, Release 2.0 upcoming
- **LLCs under development by architecture teams**
- **Blackbird™ as LLC**
  - based on MIPSpro from SGI
  - includes state-of-the-art software pipeliner
  - in conceptual stage

## Conclusions

---

- **SVM matches stream architectures, fits data-parallel computations**
- **2-level compilation strategy facilitates rapid tool development for new chips**
- **Reservoir has an HLC and a potential LLC in the works**
- **Architecture collaborators are working on their LLCs**