Implementations of Signal Processing Kernels using Stream Virtual Machine for Raw Processor

Jinwoo Suh, Stephen P. Crago, Dong-In Kang, and Janice O. McMahon

University of Southern California

Information Sciences Institute

September 20, 2005

Outline

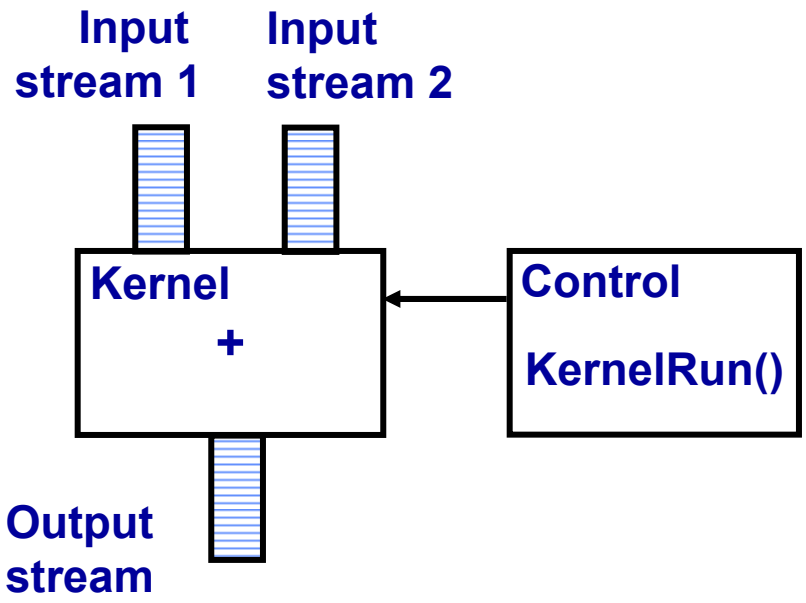
- Stream Virtual Machine (SVM) framework
  - What is the SVM? Why is it useful?
- Raw processor
- Signal processing kernels implementation results
  - Ground Moving Target Indicator (GMTI)
  - Matrix multiplication
- Conclusions

Stream Processing

- **Stream processing**
  - Processes input stream data and generates output stream data
    - Ex.: multimedia processing
  - Exploits the properties of the stream applications
    - Parallelism
    - Throughput-oriented
- **Stream Virtual Machine (SVM) framework**
  - Developed by Morphware Forum
    - Community supported stream processing framework
      - Sponsored by DARPA IPTO
      - Academia
      - Industry
  - Multiple languages
    - StreamIt (MIT)
    - Extended C (Reservoir Labs)
      - Identifies stream data
  - Multiple architectures
    - Raw (MIT)
    - Smart Memories (Stanford)
    - TRIPS (Univ. of Texas)
    - MONARCH (USC/ISI)

Stream Virtual Machine (SVM)

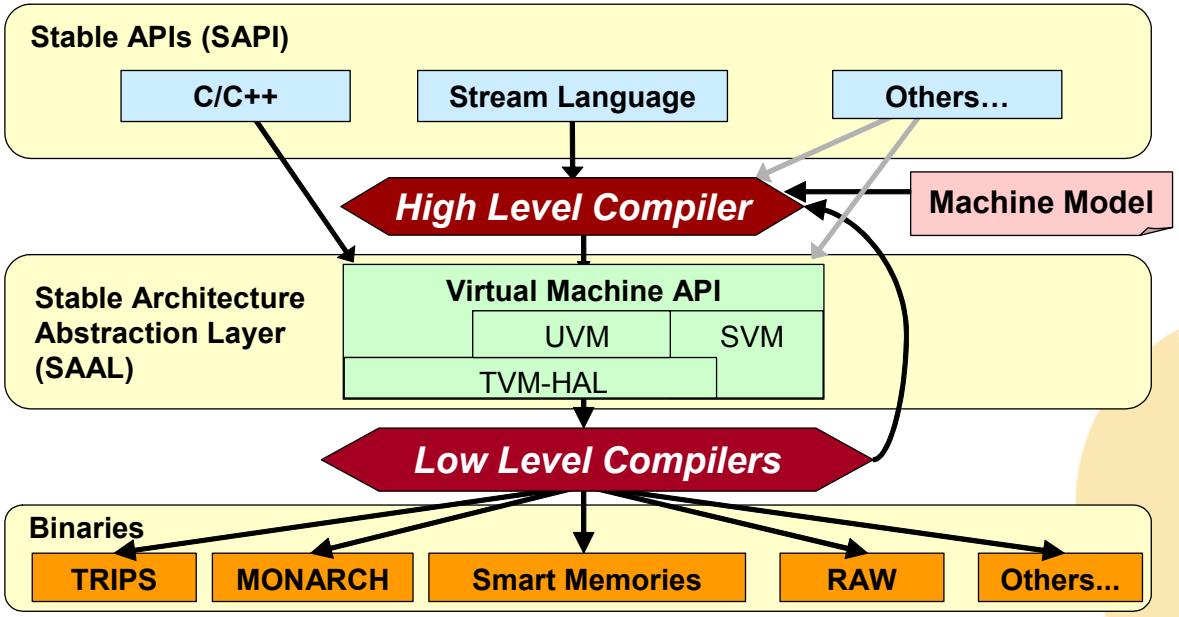
- **Streams**
  - Entities that contain an ordered collection of data elements of a given type
- **Kernels**
  - Entities that contain a locus of streaming code
  - Consume zero or more input streams and produce zero or more output streams
- **Controls**
  - Entities that embody a locus of control code
  - Initiate, monitor, and terminate the execution of kernels



* Definitions from SVM Specification 1.0.1

Two Level Approach of SVM

- **High Level Compiler (HLC)**
  - Parallelism detection, load balancing, coarse-grain scheduling of stream computations, and memory management for streaming data
- **Low Level Compiler (LLC)**
  - Software pipelining, detailed routing of data items, management of instruction memory, and interfacing between stream processors and control processors



* From SVM Specification 1.0.1

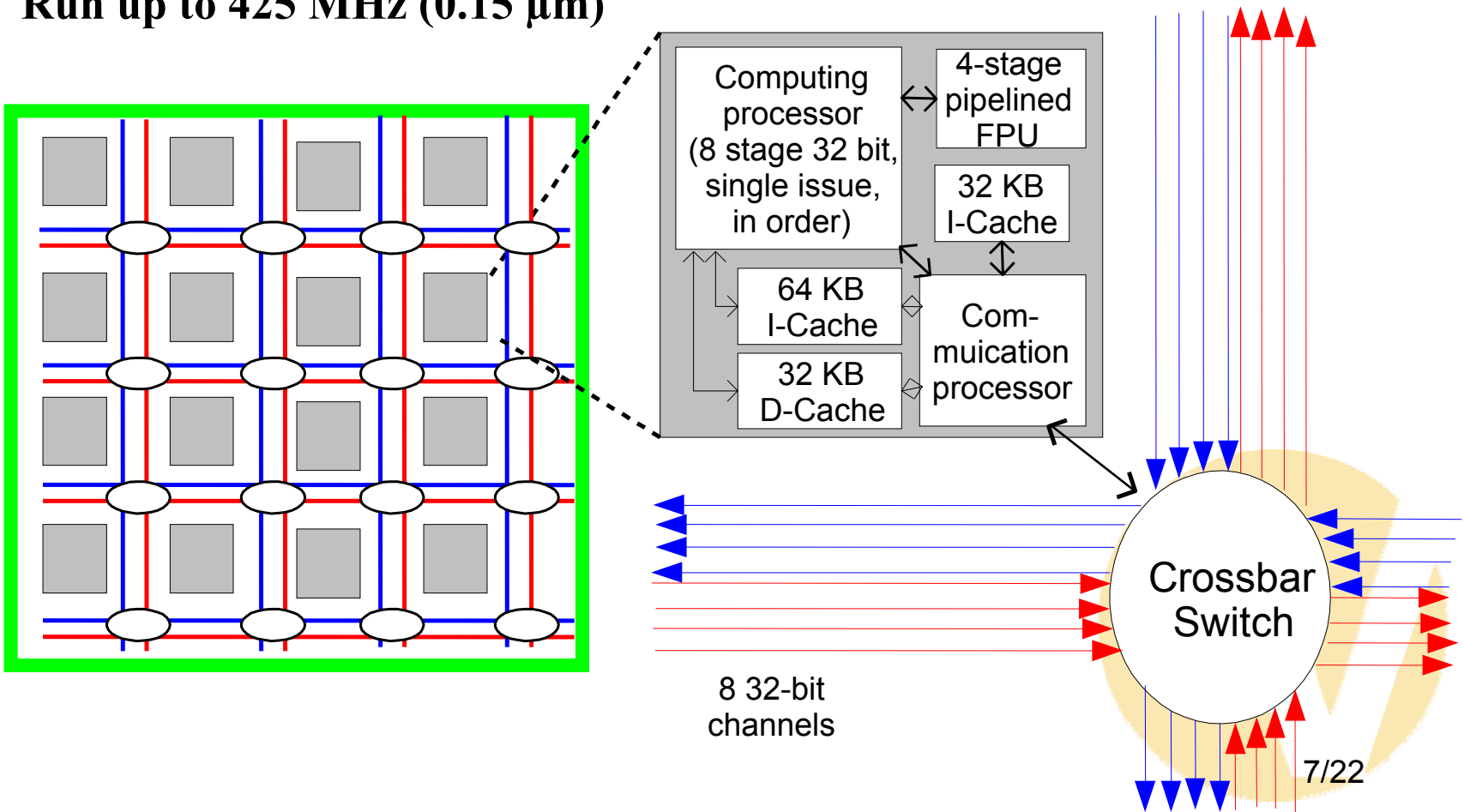
Advantages of SVM

- **Efficiency**
  - Compiler can generate efficient code by exposing communication and computation to compiler.
    - SVM API provides primitives for stream communication and computation.
    - Streams provide optimization hints.
      - Ex.: ordered data, memory space for data, etc.
- **Portability**
  - Support for multiple languages and architectures in a single framework
  - Portability across multiple architectures
- **Low development cost**
  - Adding new language
    - Only the HLC needs to be written.
  - Adding new architecture
    - Only the LLC needs to be written.
  - Programming applications
    - Ex. HLC provides parallelism.

* For more information, visit http://www.morphware.org

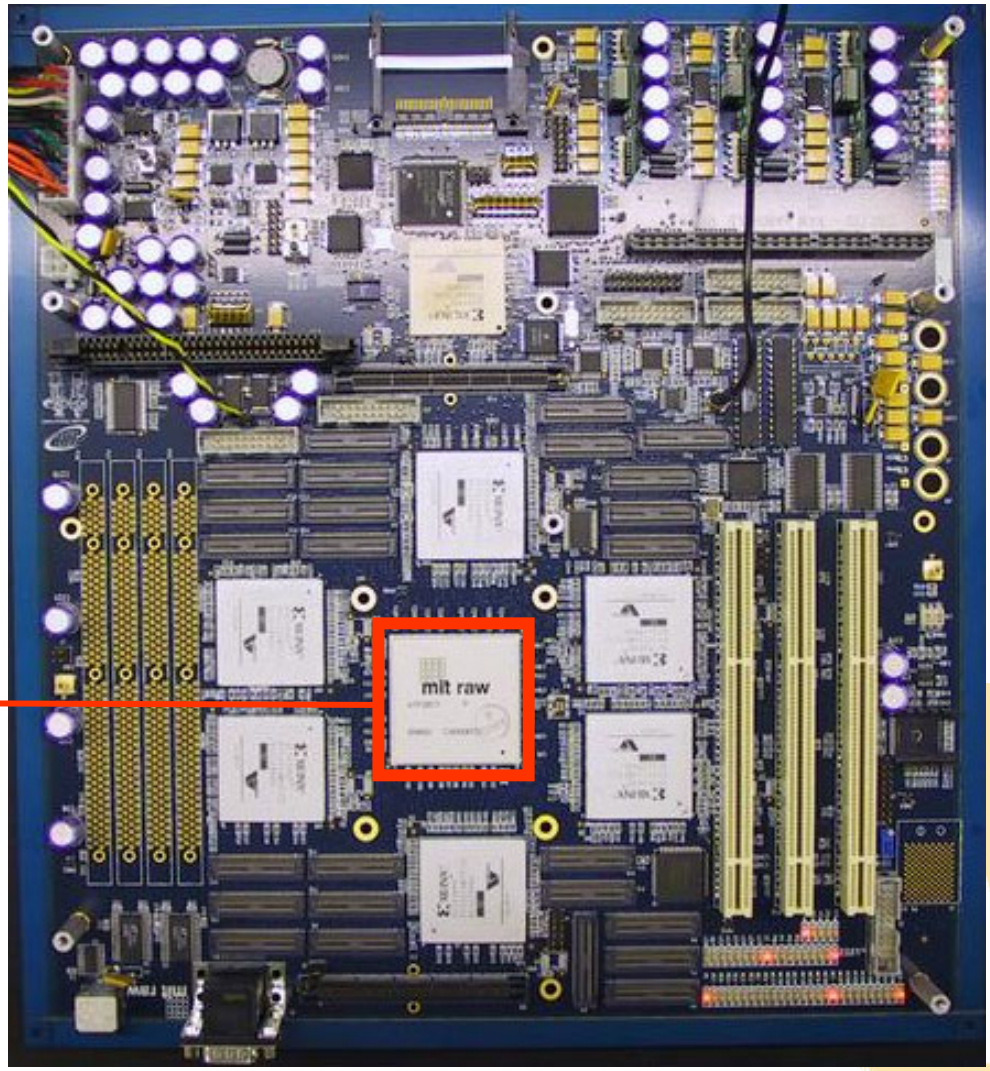
Raw Processor

- Developed by MIT
  - Small academic development team
- 16 tiles in a chip
- Run up to 425 MHz (0.15 µm)



Raw "Handheld" Board

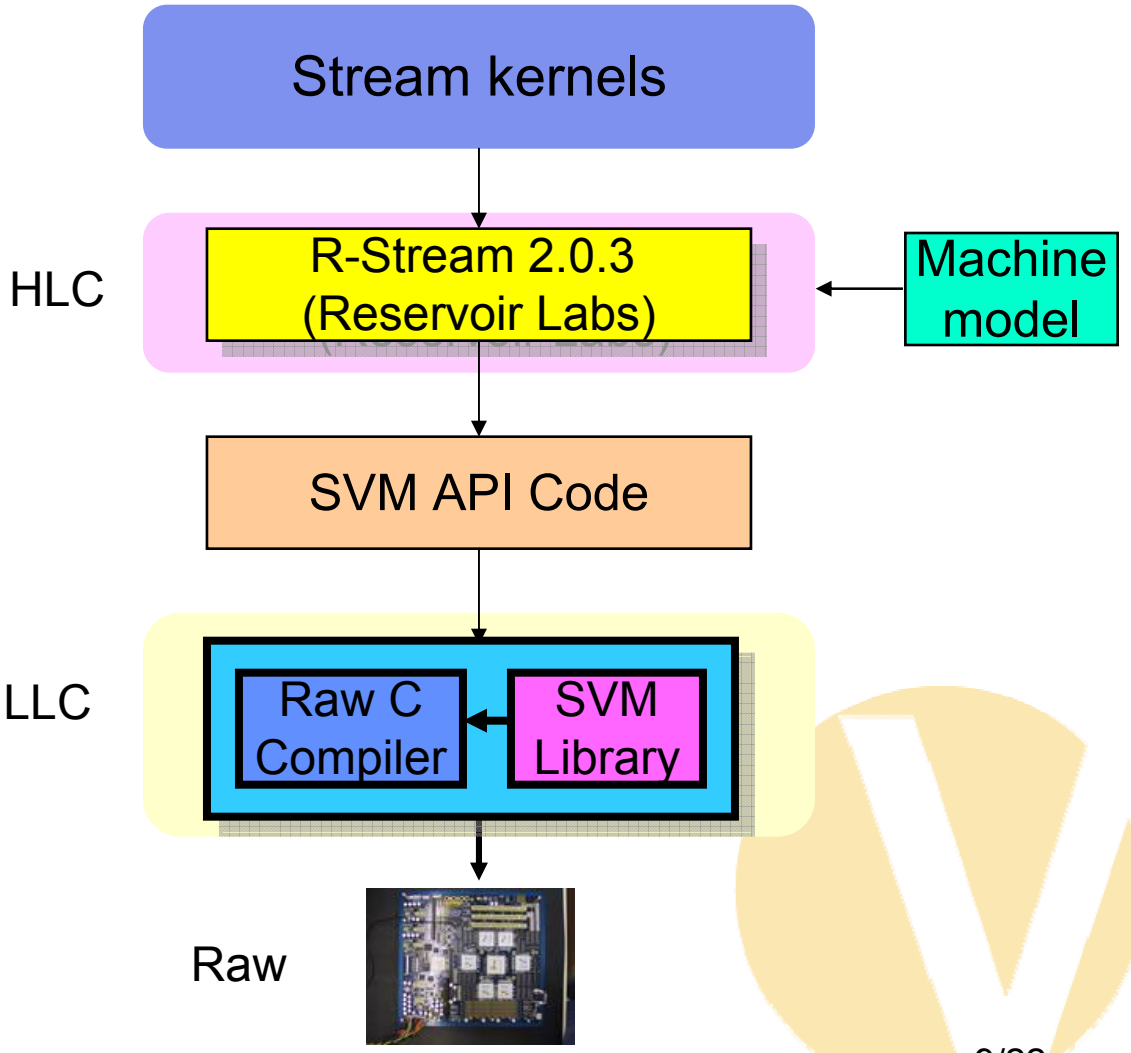
- Developed by USC/ISI in conjunction with MIT
- One chip on a board
- Board up to 300 MHz



Raw chip

HLC and LLC for Raw

- HLC
  - R-Stream -- developed by Reservoir Labs
- LLC
  - Raw C compiler by MIT
  - SVM library by USC/ISI



Signal Processing Kernels

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

Matrix multiplication (Streaming matrix Multiplication)

*** Results show current status of the tool chain in SVM framework**
*** Potential performance†**

HLC

R-Stream 2.0.3 (Reservoir Labs)

*** Results show potential performance†**

SVM API Code

LLC

Raw C Compiler

SVM Library

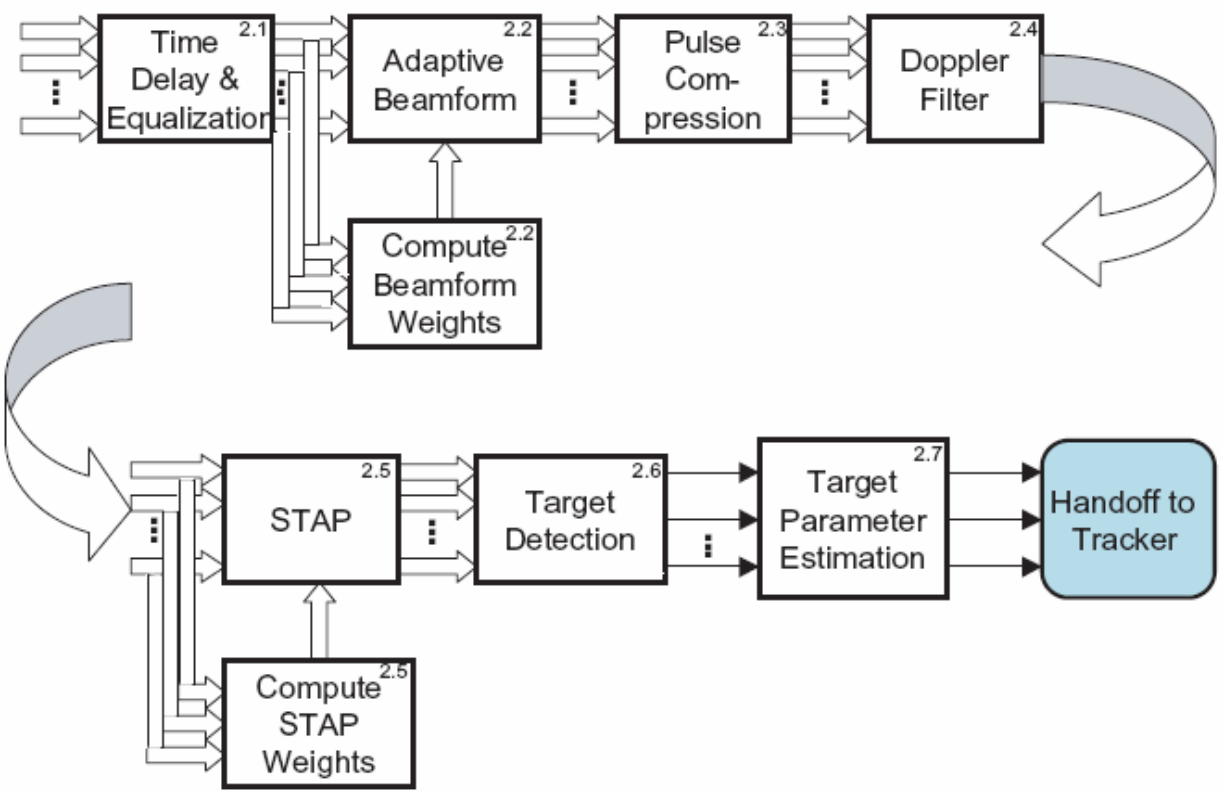
Hand-optimization

Raw

†Currently achieved using hand coding

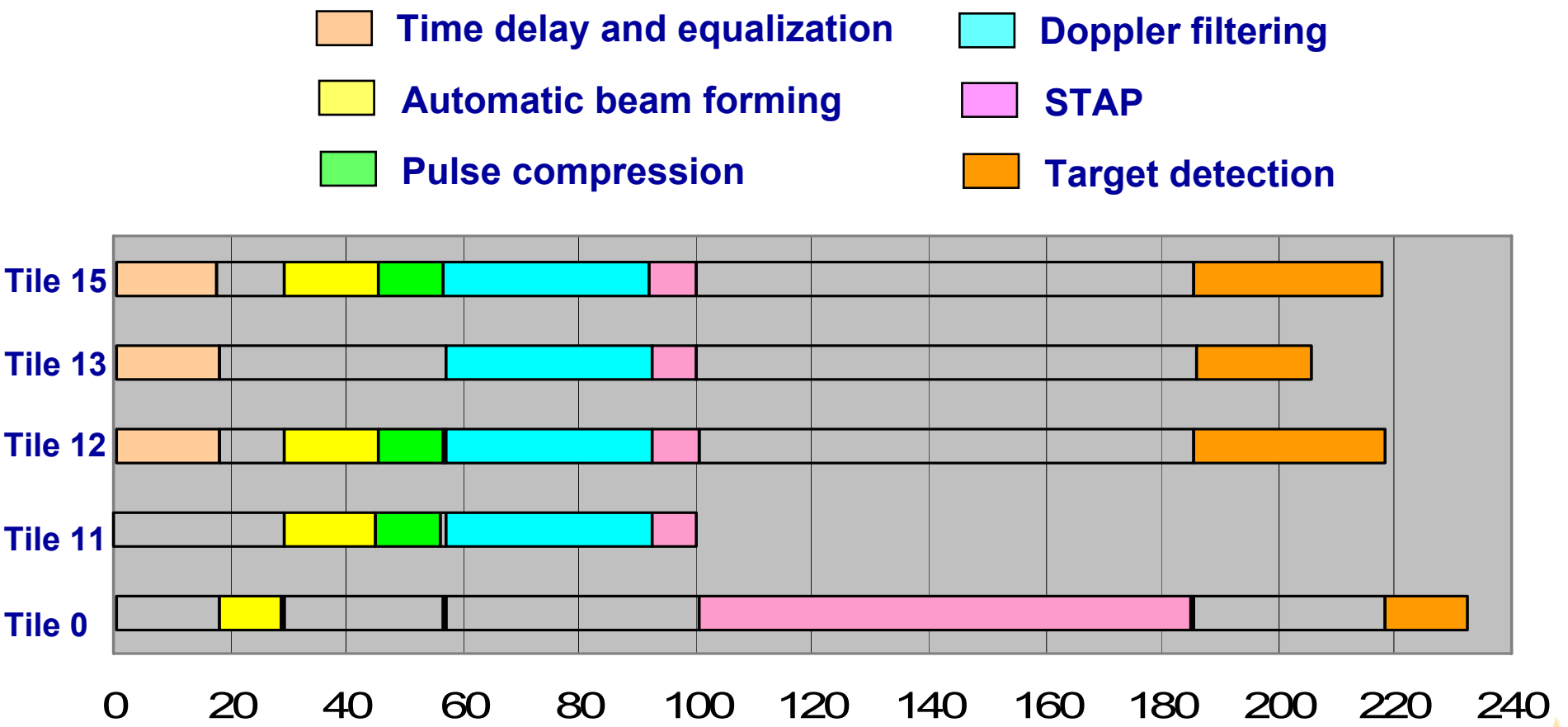
Ground Moving Target Indicator

- **Ground Moving Target Indicator (GMTI)**
  - **Detects targets from input radar signal.**
  - **Consists of 7 stages.**
    - **First 6 stages implemented.**



A.I. Reuther, "Preliminary Design Review: GMTI Narrowband for the Basic PCA Integrated Radar-Tracker Application," Project Report PCA-IRT-3, Lincoln Labs, 2004.

GMTI Execution Schedule

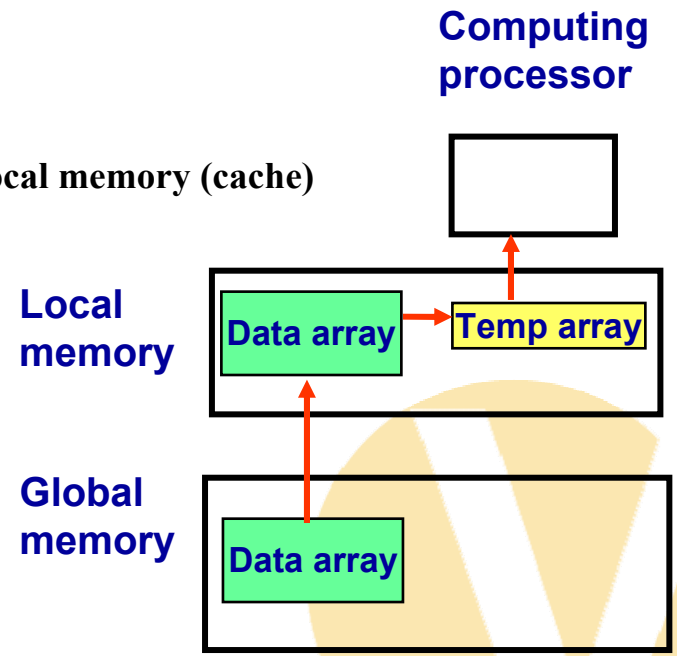


GMTI Execution Analysis

- **Parallelization**
  - **Currently, up to 4 tiles are used.**
    - **The latest results (not shown in this presentation) show up to 16 tile parallelization.**
  - **STAP looks like it is not parallelized.**
    - **Actually, STAP uses software pipeline**
    - **This will be clear if there are more than one data cube.**
- **Performance**
  - **We are working on improvement of performance.**
  - **Possible improvement methods in next slides**

Time Delay And Equalization (TDE)

- **Chosen as representative kernel for detailed analysis.**
- **TDE stage**
  - Convolution operation
  - Parameters
    - Input: 36 complex numbers
    - Filters: 12 complex numbers
- **Implemented**
  - Steps
    - Move data from global memory (main memory) to local memory (cache)
    - Move data to a temporary array
    - FFT
    - Multiplication
    - IFFT
    - Move data from temporary array to local memory
    - Move data from local memory to global memory
  - Algorithmic optimization
    - Radix-4 FFT and IFFT used
    - Bit-reverse eliminated



TDE Stage Optimizations

- **Elimination of duplicated code**
  - HLC generated code has code that does essentially the same thing more than once.
  - We manually eliminated duplicated code.
- **Direct copy**
  - Copy operations using SVM APIs are optimized using direct C code when possible.
- **No copy to local memory**
  - Copy operations are replaced with code that relays pointer.
- **Hand-assembly**
  - Use assembly code for core operations, such as FFT.

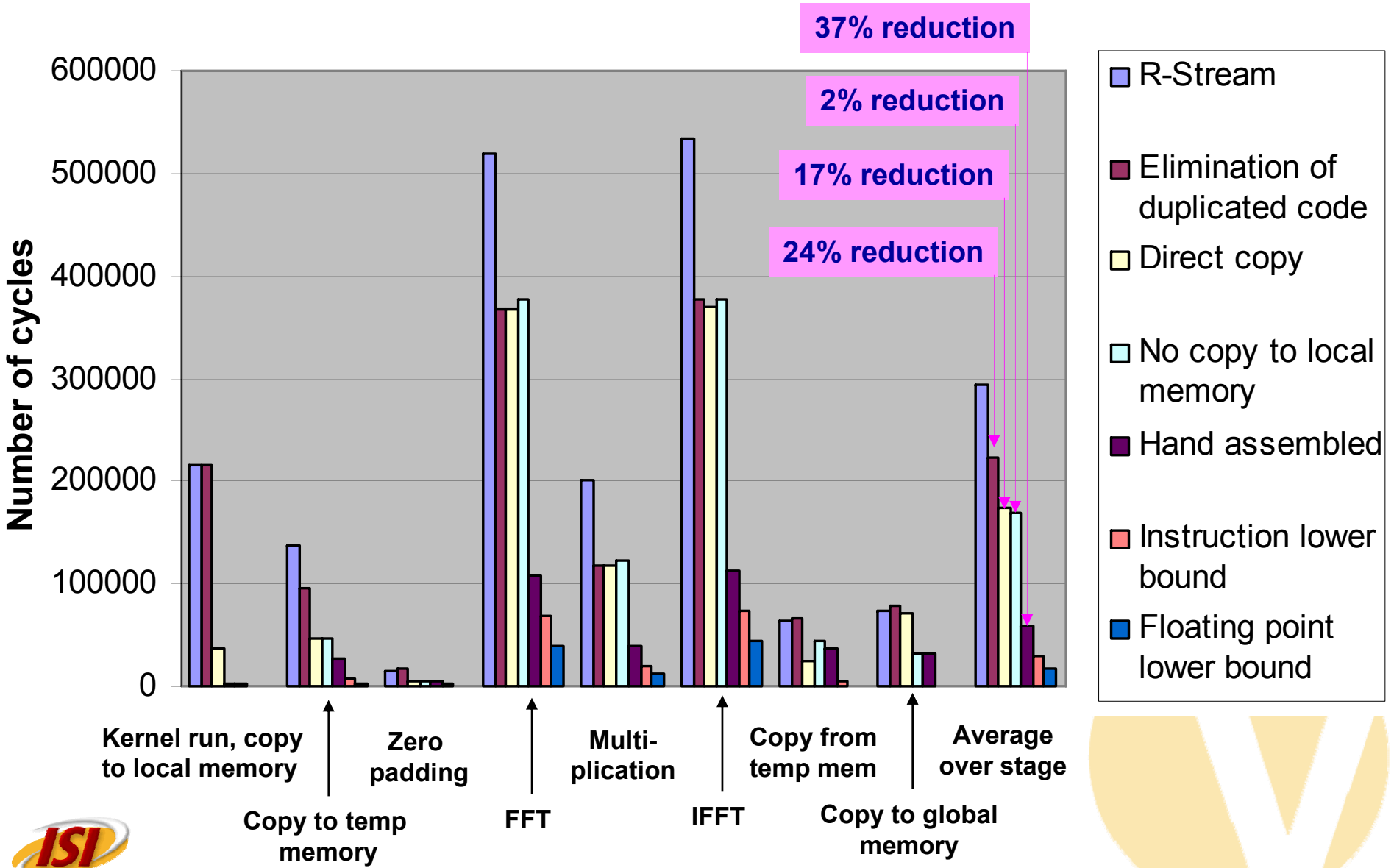
Lower Bound Definitions

- **Floating point lower bound**
  - Count only number of floating point operations
- **Instruction lower bound**
  - Count minimum instructions needed
- **Example**

```
For (i=0; i<10; i++)
   c[i] = a[i] + b;
```

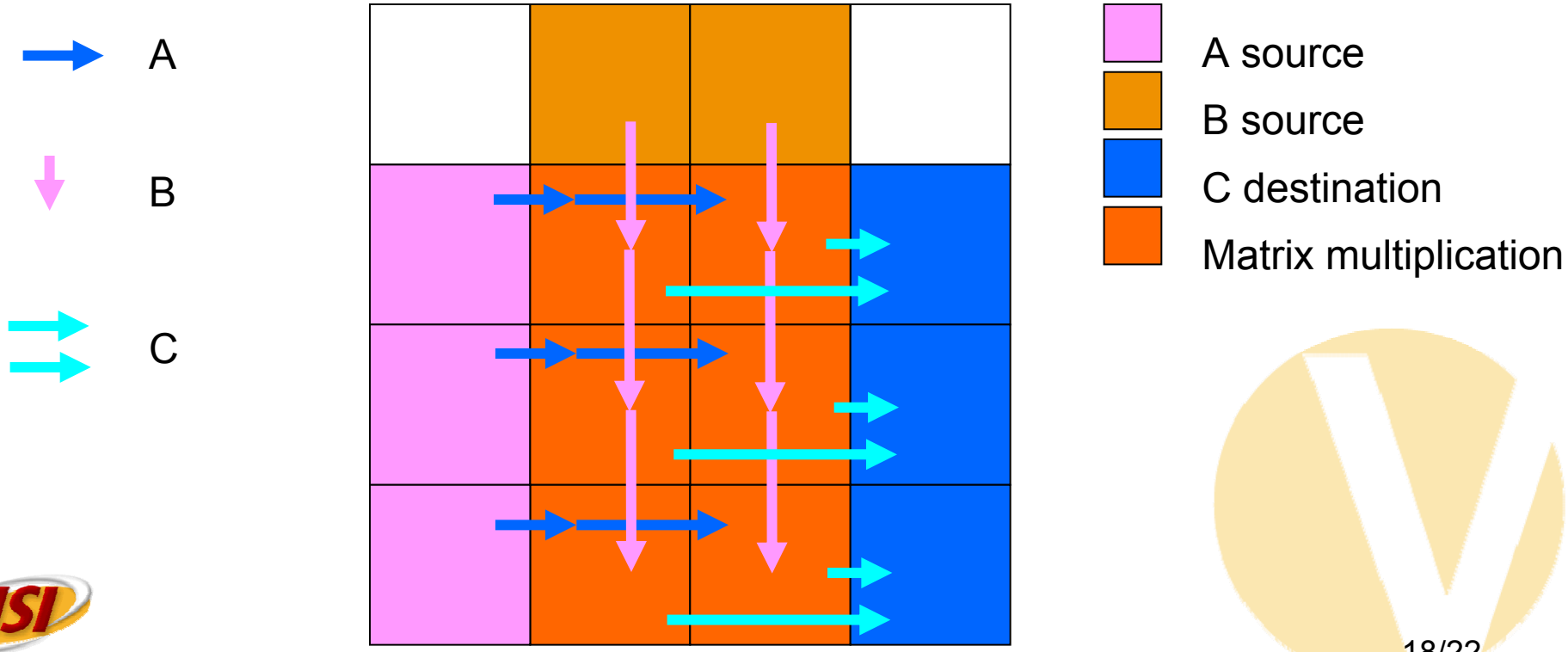
  - Floating point lower bound = 10 cycles
  - Instruction lower bound = 31 cycle
    - 10 load instructions for loading elements of *a*
    - 1 load instruction for scalar variable *b*
    - 10 floating point add operations for each computed element of *c*
    - 10 store instructions for elements of *c*
    - Not counted: loop variable, index calculation
      - These can be eliminated by optimizations.

TDE Stage Results



Matrix Multiplication

- C = AB
- Boundary tiles emulate network input/output by generating and consuming data

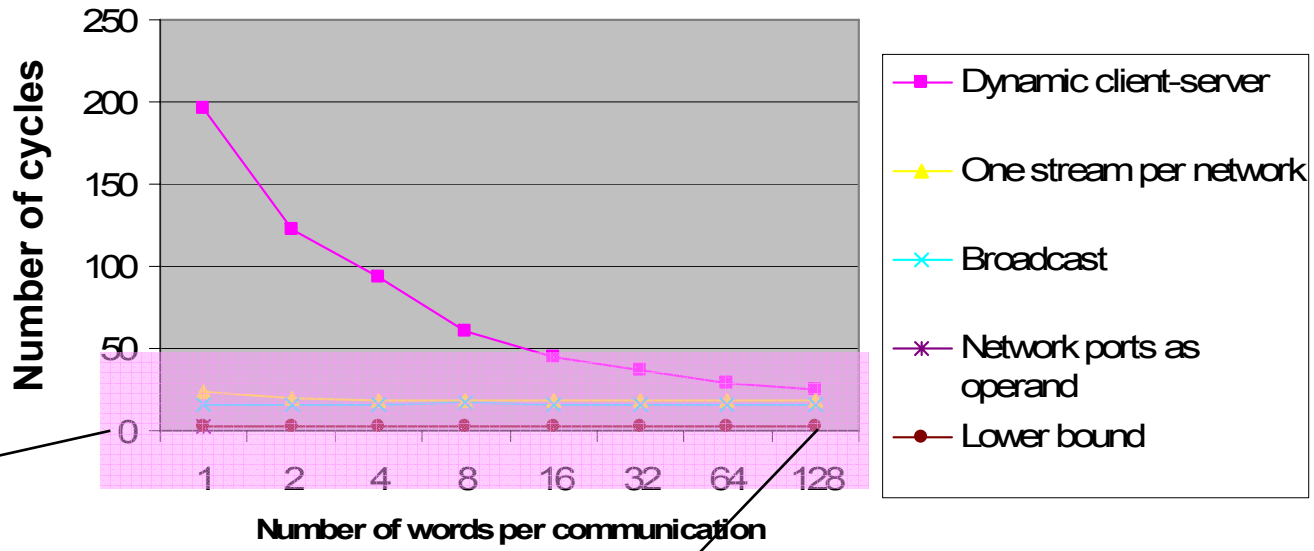


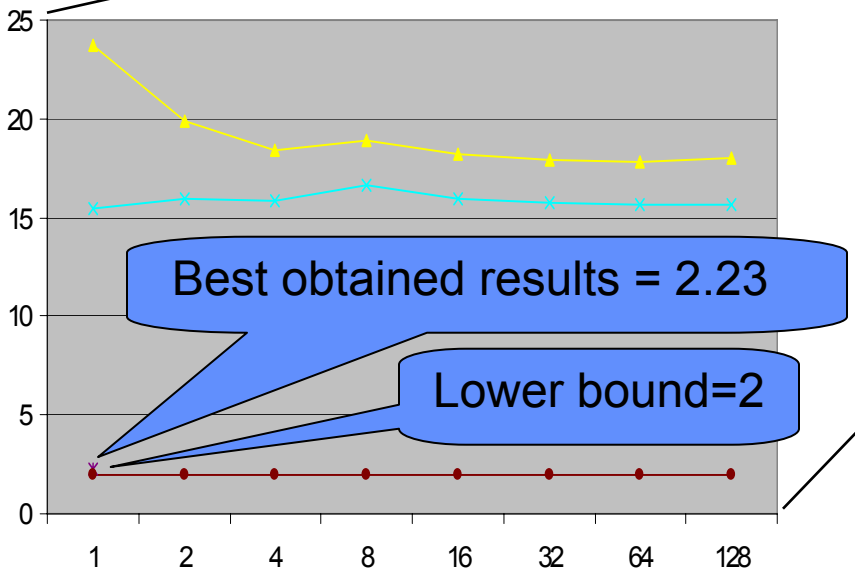
Matrix Multiplication Implementation

- Hand coded using the SVM API (not HLC-generated code)
- Cost analysis and optimizations
  - Full implementation
    - Full SVM stream communication through Raw network
  - One stream per network
    - Each stream is allocated to a Raw scalar operand 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





Conclusions

- **Evaluated tool chain in SVM framework on Raw**
  - **Implemented signal processing kernels**
    - **GMTI and matrix multiplication**
  - **SVM framework functionally works well on Raw.**
    - **With minor modifications of code from HLC**
  - **Performance**
    - **Currently, without optimization, there is a big difference between peak performance and obtained performance.**
    - **Both HLC and SVM library have room for improvement.**
      - **These are in early development stages and being improved continuously.**
    - **Optimizations boost performance close to the upper bound.**

The SVM's potential performance is promising.

Acknowledgements

- The authors gratefully acknowledge the MIT Raw team for the use of their compilers, simulators, Raw processor, and their generous help.
- The authors gratefully acknowledge the Reservoir Labs for the use of their compilers and their generous help.
- The authors also acknowledge MIT Lincoln Labs for providing the GMTI application.
- Effort sponsored by Defense Advanced Research Projects Agency (DARPA) through the Air Force Research Laboratory (AFRL), USAF.