High Performance Embedded Computing Software Initiative (HPEC-SI)

Dr. Jeremy Kepner / Lincoln Laboratory

This work is sponsored by the Department of Defense under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.
Outline

- Introduction
  - Demonstration
  - Development
  - Applied Research
  - Future Challenges
- Goals
- Program Structure
- Summary
Overview - High Performance Embedded Computing (HPEC) Initiative

Common Imagery Processor (CIP)

- Shared memory server
- Embedded multi-processor

ASARS-2

Enhanced Tactical Radar Correlator (ETRAC)

**Challenge:** Transition advanced software technology and practices into major defense acquisition programs
Why Is DoD Concerned with Embedded Software?

- COTS acquisition practices have shifted the burden from “point design” hardware to “point design” software.
- Software costs for embedded systems could be reduced by one-third with improved programming models, methodologies, and standards.

Source: “HPEC Market Study” March 2001

Estimated DoD expenditures for embedded signal and image processing hardware and software ($B)
Issues with Current HPEC Development

Inadequacy of Software Practices & Standards

- High Performance Embedded Computing pervasive through DoD applications
  - Airborne Radar Insertion program
    85% software rewrite for each hardware platform
  - Missile common processor
    Processor board costs < $100k
    Software development costs > $100M
  - Torpedo upgrade
    Two software re-writes required after changes in hardware design

Today – Embedded Software Is:
- Not portable
- Not scalable
- Difficult to develop
- Expensive to maintain
Evolution of Software Support Towards “Write Once, Run Anywhere/Anyysize”

- Application software has traditionally been tied to the hardware
- Many acquisition programs are developing stove-piped middleware “standards”
- Open software standards can provide portability, performance, and productivity benefits
- Support “Write Once, Run Anywhere/Anyysize”
**Quantitative Goals & Impact**

**Program Goals**
- Develop and integrate software technologies for embedded parallel systems to address portability, productivity, and performance
- Engage acquisition community to promote technology insertion
- Deliver quantifiable benefits

**Portability**: reduction in lines-of-code to change port/scale to new system

**Productivity**: reduction in overall lines-of-code

**Performance**: computation and communication benchmarks
Organization

- Partnership with ODUSD(S&T), Government Labs, FFRDCs, Universities, Contractors, Vendors and DoD programs
- Over 100 participants from over 20 organizations
HPEC-SI Capability Phases

Phase 1
Applied Research: Unified Comp/Comm Lib
Development: Object-Oriented Standards
Demonstration: Existing Standards

Phase 2
Applied Research: Fault tolerance
Development: Unified Comp/Comm Lib
Demonstration: Object-Oriented Standards

Phase 3
Applied Research: Hybrid Architectures
Development: Fault tolerance
Demonstration: Unified Comp/Comm Lib

Functionality

Time

High-level code abstraction (AEGIS)
• Reduce code size 3x

Unified embedded computation/communication standard
• Demonstrate scalability

Demonstrate insertions into fielded systems (CIP)
• Demonstrate 3x portability

• First demo successfully completed
• Second Demo Selected
• VSIPL++ v0.8 spec completed
• VSIPL++ v0.2 code available
• Parallel VSIPL++ v0.1 spec completed
• High performance C++ demonstrated

MITRE
MIT Lincoln Laboratory
AFRL

www.hpec-si.org
Outline

- Introduction

**Demonstration**

- Development

- Applied Research

- Future Challenges

- Summary

- **Common Imagery Processor**
  - AEGIS BMD (planned)
Common Imagery Processor (CIP) is a cross-service component

Sample list of CIP modes
- U-2 (ASARS-2, SYERS)
- F/A-18 ATARS (EO/IR/APG-73)
- LO HAE UAV (EO, SAR)
- System Manager

* CIP picture courtesy of Northrop Grumman Corporation
Common Imagery Processor
- Demonstration Overview -

- Demonstrate standards-based platform-independent CIP processing (ASARS-2)
- Assess performance of current COTS portability standards (MPI, VSIPL)
- Validate SW development productivity of emerging Data Reorganization Interface
- MITRE and Northrop Grumman

Shared-Memory Servers

Common Imagery Processor

Embedded Multicomputers

Commodity Clusters Massively Parallel Processors

Single code base optimized for all high performance architectures provides future flexibility

Slide-12
www.hpec-si.org
Software Ports

Embedded Multicomputers
- CSPI - 500MHz PPC7410 (vendor loan)
- Mercury - 500MHz PPC7410 (vendor loan)
- Sky - 333MHz PPC7400 (vendor loan)
- Sky - 500MHz PPC7410 (vendor loan)

Mainstream Servers
- HP/COMPAQ ES40LP - 833-MHz Alpha ev6 (CIP hardware)
- HP/COMPAQ ES40 - 500-MHz Alpha ev6 (CIP hardware)
- SGI Origin 2000 - 250MHz R10k (CIP hardware)
- SGI Origin 3800 - 400MHz R12k (ARL MSRC)
- IBM 1.3GHz Power 4 (ARL MSRC)
- Generic LINUX Cluster
Portability: SLOC Comparison

- **Main Support Kernel**
  - Sequential
  - VSIPL
  - Shared Memory VSIPL
  - DRI VSIPL

- **Total**

- **SLOCs**
- **VSIPL SLOCs**

- ~1% Increase
- ~5% Increase
Application can now exploit many more processors, embedded processors (3x form factor advantage) and Linux clusters (3x cost advantage)
Form Factor Improvements

Current Configuration

- IOP: 6U VME chassis (9 slots potentially available)
- IFP: HP/COMPAQ ES40LP

Possible Configuration

- IOP could support 2 G4 IFPs
  - form factor reduction (x2)
- 6U VME can support 5 G4 IFPs
  - processing capability increase (x2.5)
HPEC-SI Goals

1st Demo Achievements

Portability: **zero** code changes required
Productivity: DRI code 6x smaller vs MPI (est*)
Performance: 2x reduced cost or form factor

Portability: reduction in lines-of-code to change port/scale to new system
Productivity: reduction in overall lines-of-code
Performance: computation and communication benchmarks

HPEC Software Initiative

**Achieved**
Goal 3x Portability

**Achieved**
Goal 3x Productivity

**Performance Goal 1.5x Achieved**
Outline

• Introduction
• Demonstration

• Development
  • Object Oriented (VSIPL++)
  • Parallel (||VSIPL++)

• Applied Research
• Future Challenges
• Summary
Emergence of Component Standards

HPEC Initiative - Builds on completed research and existing standards and libraries

Definitions
VSIPL = Vector, Signal, and Image Processing Library
||VSIPL++ = Parallel Object Oriented VSIPL
MPI = Message-passing interface
MPI/RT = MPI real-time
DRI = Data Re-arrangement Interface
CORBA = Common Object Request Broker Architecture
HP-CORBA = High Performance CORBA
**VSIPL++ Productivity Examples**

**BLAS zherk Routine**

- **BLAS** = Basic Linear Algebra Subprograms
- Hermitian matrix $M$: $\text{conj}(M) = M^t$
- $\text{zherk}$ performs a rank-$k$ update of Hermitian matrix $C$:
  \[
  C \leftarrow \alpha \ast A \ast \text{conj}(A)^t + \beta \ast C
  \]

**VSIPL code**

```plaintext
A = vsip_cmcreate_d(10,15,VSIP_ROW,MEM_NONE);
C = vsip_cmcreate_d(10,10,VSIP_ROW,MEM_NONE);
tmp = vsip_cmcreate_d(10,10,VSIP_ROW,MEM_NONE);
vsip_cmprodh_d(A,A,tmp); /* A*conj(A)^t */
vsip_rscmmul_d(alpha,tmp,tmp);/* \alpha*A*conj(A)^t */
vsip_rscmmul_d(beta,C,C); /* \beta*C */
vsip_cmadd_d(tmp,C,C); /* \alpha*A*conj(A)^t + \beta*C */
vsip_cblockdestroy(vsip_cmdestroy_d(tmp));
vsip_cblockdestroy(vsip_cmdestroy_d(C));
vsip_cblockdestroy(vsip_cmdestroy_d(A));
```

**VSIPL++ code (also parallel)**

```plaintext
Matrix<complex<double> > A(10,15);
Matrix<complex<double> > C(10,10);
C = alpha * prodh(A,A) + beta * C;
```

**Sonar Example**
- K-W Beamformer
- Converted C VSIPL to VSIPL++
- 2.5x less SLOCs
## Results

- Hand coded loop achieves good performance, but is problem specific and low level
- Optimized VSIPL performs well for simple expressions, worse for more complex expressions
- PETE style array operators perform almost as well as the hand-coded loop and are general, can be composed, and are high-level

### Software Technology

<table>
<thead>
<tr>
<th>AltiVec loop</th>
<th>VSIPL (vendor optimized)</th>
<th>PETE with AltiVec</th>
</tr>
</thead>
<tbody>
<tr>
<td>• C</td>
<td>• C</td>
<td>• C++</td>
</tr>
<tr>
<td>• For loop</td>
<td>• AltiVec aware VSIPRO Core Lite (<a href="http://www.mpi-softech.com">www.mpi-softech.com</a>)</td>
<td>• PETE operators</td>
</tr>
<tr>
<td>• Direct use of AltiVec extensions</td>
<td>• No multiply-add</td>
<td>• Indirect use of AltiVec extensions</td>
</tr>
<tr>
<td>• Assumes unit stride</td>
<td>• Cannot assume unit stride</td>
<td>• Assumes unit stride</td>
</tr>
<tr>
<td>• Assumes vector alignment</td>
<td>• Cannot assume vector alignment</td>
<td>• Assumes vector alignment</td>
</tr>
</tbody>
</table>
Parallel Pipeline Mapping

Signal Processing Algorithm

- Filter: $X_{OUT} = \text{FIR}(X_{IN})$
- Beamform: $X_{OUT} = w \cdot X_{IN}$
- Detect: $X_{OUT} = |X_{IN}| > c$

Mapping

Parallel Computer

- Data Parallel within stages
- Task/Pipeline Parallel across stages
Scalable Approach

```c
#include <Vector.h>
#include <AddPvl.h>

void addVectors(aMap, bMap, cMap) {
    Vector< Complex<Float> > a('a', aMap, LENGTH);
    Vector< Complex<Float> > b('b', bMap, LENGTH);
    Vector< Complex<Float> > c('c', cMap, LENGTH);

    b = 1;
    c = 2;
    a=b+c;
}
```

**Single Processor Mapping**

```
A = B + C
```

**Multi Processor Mapping**

```
A = B + C
```

**Lincoln Parallel Vector Library (PVL)**
- Single processor and multi-processor code are the same
- Maps can be changed without changing software
- High level code is compact
Outline

• Introduction
• Demonstration
• Development
• Applied Research
• Future Challenges
• Summary

• Fault Tolerance
• Parallel Specification
• Hybrid Architectures (see SBR)
Dynamic Mapping for Fault Tolerance

- Switching processors is accomplished by switching maps
- No change to algorithm required
- Developing requirements for ||VSIPL++
Parallel Specification

Clutter Calculation (Linux Cluster)

- Matlab is the main specification language for signal processing
- pMatlab allows parallel specifications using same mapping constructs being developed for ||VSIPL++

```
% Initialize
pMATLAB_Init; Ncpus=comm_vars.comm_size;

% Map X to first half and Y to second half.
mapX=map([1 Ncpus/2],{},[1:Ncpus/2])
mapY=map([Ncpus/2 1],{},[Ncpus/2+1:Ncpus]);

% Create arrays.
X = complex(rand(N,M,mapX),rand(N,M,mapX));
Y = complex(zeros(N,M,mapY));

% Initialize coefficients
coefs = ...
weights = ...

% Parallel filter + corner turn.
Y(:,:,1) = conv2(coefs,X);
% Parallel matrix multiply.
Y(:,:,1) = weights*Y;

% Finalize pMATLAB and exit.
pMATLAB_Finalize; exit;
```
Outline

• Introduction
• Demonstration
• Development
• Applied Research
• Future Challenges
• Summary
Optimal Mapping of Complex Algorithms

Application

Input
\( X_{IN} \)

Low Pass Filter
\( X_{IN} \)
FIR1
\( W_1 \)

FIR2
\( W_2 \)

\( X_{OUT} \)

Beamform
\( X_{IN} \)
\( mult \)
\( W_3 \)

\( X_{OUT} \)

Matched Filter
\( X_{IN} \)
FFT
\( W_4 \)

IFFT
\( X_{OUT} \)

Different Optimal Maps

Workstation

Intel Cluster

PowerPC Cluster

Embedded Board

Embedded Multi-computer

Hardware

• Need to automate process of mapping algorithm to hardware
HPEC-SI Future Challenges

End of 5 Year Plan

**Phase 5**
- Applied Research: Higher Languages (Java?)
- Development: Self-optimization

**Phase 4**
- Applied Research: PCA/Self-optimization
- Development: Hybrid Architectures
- Demonstration: Fault tolerance
  - Hybrid VSIPL
- Demonstrate Fault Tolerance
  - Increased reliability

**Phase 3**
- Applied Research: Hybrid Architectures
- Development: Fault tolerance
- Demonstration: Unified Comp/Comm Lib
  - Parallel VSIPL++
  - Unified Comp/Comm Standard
  - Demonstrate scalability

**Phase 2**
- Development: Hybrid Architectures
  - Fault Tolerance
  - FT VSIPL
- Demonstration: Fault tolerance
- Demonstrate Fault Tolerance
- Portability across architectures
  - RISC/FPGA Transparency

**Phase 1**
- Applied Research: Hybrid Architectures
- Development: Hybrid Architectures
- Demonstration: Fault tolerance

Time

Functionality
Summary

• HPEC-SI Program on track toward changing software practice in DoD HPEC Signal and Image Processing
  – Outside funding obtained for DoD program specific activities (on top of core HPEC-SI effort)
  – 1st Demo completed; 2nd selected
  – Worlds first parallel, object oriented standard
  – Applied research into task/pipeline parallelism; fault tolerance; parallel specification

• Keys to success
  – Program Office Support: 5 Year Time horizon better match to DoD program development
  – Quantitative goals for portability, productivity and performance
  – Engineering community support
Web Links

High Performance Embedded Computing Workshop
http://www.ll.mit.edu/HPEC

High Performance Embedded Computing Software Initiative
http://www.hpec-si.org/

Vector, Signal, and Image Processing Library
http://www.vsipl.org/

MPI Software Technologies, Inc.
http://www.mpi-softtech.com/

Data Reorganization Initiative
http://www.data-re.org/

CodeSourcery, LLC
http://www.codesourcercy.com/

MatlabMPI
http://www.ll.mit.edu/MatlabMPI