Missile Seeker Common Computer
Signal Processing Architecture for
Rapid Technology Upgrade∗

Daniel V. Rabinkin, Edward M. Rutledge, and
Paul Monticciolo

Embedded Digital Systems Group
September 26, 2002

MIT Lincoln Laboratory, 244 Wood Street
Lexington, MA 02420

∗ This work is sponsored by the United States Navy Standard Missile Program PMS-422. Work performed by MIT Lincoln Laboratory is covered under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force or the United States Navy.
Outline

- Introduction
- Signal Processor Architecture
  - Hardware
  - Software
- System Implementation and Demonstration
- Summary
STANDARD Missile/Project Hercules/THAAD Signal Processor Upgrade Program

- Risk mitigation effort for next generation missile processor and software architectures
- Implementing seeker algorithms in real-time using parallel processing techniques
- Developed demonstration test beds

SM-3 Seeker and Project Hercules Discrimination Algorithms

Real-Time IR Scene Generation
Seeker Experimental System (SES)

SM-3 IR Sensor

Project Hercules Approved Data

Testbeds
Network of Workstations
Quad G4 Embedded System

SM-3 Seeker
G4 Power PC/PCI Processor Architecture
Processor Independent Signal Processing Software Architecture

Application Level
Library Level (PVL, VSIPL, MPI)
Kernel Level

Principal Processor Components
Outline

• Introduction

• Signal Processor Architecture
  – Hardware
  – Software

• System Implementation and Demonstration

• Summary
Signal Processing Architecture Example: Basis for Benchmarking

Focal Plane Array

- Integrator
- A/D
- Background Estimation
- Adaptive Nonuniformity Compensation
- Aimpoint Selection*
- Target Selection (Discrimination)
- Data Fusion
- Feature Extraction
- Tracking
- IMU
- Front-End Video Processing
- Back-end Target Processing
- Multi-Frame Processing
- CFAR Detection
- Dither Processing
- Bulk Filter*

Guidance

To thrusters/ fins

Target update from ship’s radar

SM-3 Requirements

OPCOUNT: 300 MFLOPS Current
          ~7 GFLOPS Advanced
Memory: 10 Mbytes Current
        200 Mbytes Advanced

Form Factor, Power, Weight Constraints

Near-Term Processing requirements met by COTS quad G4 board
# Processor Technology Overview 3Q ’02

<table>
<thead>
<tr>
<th>Processor</th>
<th>Clock (MHz)</th>
<th>MFLOPS (Peak)</th>
<th>Avg. Power (Watts)</th>
<th>Cache Memory</th>
<th>Ext. Bus Mbytes/S @MHz</th>
<th>Prog. Effort</th>
<th>Typical Operating Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1 (onboard)</td>
<td>L2 (onboard)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Itanium2 (GPP)</td>
<td>1000</td>
<td>~8,000</td>
<td>100</td>
<td>32KB</td>
<td>256KB</td>
<td>6400@400</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>MPC7455 (GPP)</td>
<td>1000</td>
<td>7,000</td>
<td>21.3</td>
<td>32KB Ins 32KB dat</td>
<td>256KB (onboard)</td>
<td>2MB (offboard)</td>
<td>1064@133</td>
</tr>
<tr>
<td>MPC7451 (GPP)</td>
<td>667</td>
<td>5,336</td>
<td>14.5</td>
<td>32KB Ins 32KB dat</td>
<td>256KB (onboard)</td>
<td>2MB (offboard)</td>
<td>1064@133</td>
</tr>
<tr>
<td>MPC7410 (GPP)</td>
<td>500</td>
<td>4,000</td>
<td>5.3</td>
<td>32KB Ins 32KB dat</td>
<td>2MB (offboard)</td>
<td>---</td>
<td>1064@133</td>
</tr>
<tr>
<td>TMS320C6713 (DSP)</td>
<td>225</td>
<td>1,350</td>
<td>1.2</td>
<td>4KB Ins 4KB dat</td>
<td>64MB (onboard)</td>
<td>192KB (offboard)</td>
<td>900@255</td>
</tr>
<tr>
<td>Virtex II-Pro (FPGA)</td>
<td>300</td>
<td>~30,000 (MOPS)</td>
<td>~1</td>
<td>1.25MB (onboard)</td>
<td></td>
<td></td>
<td>High</td>
</tr>
</tbody>
</table>

- Development cost tied to OS, library, and tool support
- Architecture track record must be considered for upgrade cost projections
- Memory and communication bandwidth usually set limit on GPP and DSP performance
  - Large cache and good cache hierarchy needed to achieve opcount potential
- PowerPC architecture is best compromise for high performance/easy development
  - Many PowerPC COTS multiprocessor signal processing platforms available today
DY4 COTS-Based G4-Based Processor Architecture & Board

Principal Attributes

- Application scalable
- Uses industry-standard programmable processor and bus
- Excellent performance-to-power ratio
- Commercially available development tools
- COTS board manufacturers will form-factor board to specification
Focal plane data is parallelized among 3 processors to overcome front-end bottleneck:
Should consider new readout for next generation seekers to fully exploit parallel processing
Outline

• Introduction
• Signal Processor Architecture
  – Hardware
  – Software
• System Implementation and Demonstration
• Summary
Software Support for Lifecycle Maintainability

- Moore’s Law means commercial processor hardware will change several times within the system lifetime
- Application software has traditionally been tied to the hardware
  - Significant recoding required to migrate to new hardware
- Many acquisition programs are developing stove-piped middleware “standards”
- Open software standards provides portability, performance, and productivity benefits
Extending the Standards-Based Approach to Parallel Processing: PVL

Single Processor

Multilayered Software Architecture

C/Ada

Existing Libraries
Vector, Signal, Image Processing (VSIPL)
Message Passing Interface (MPI)

Application

Library

Kernel

Multi-Processor

Parallel Vector Library (PVL)

C++

PVL

Attributes

Object Oriented applications for portability and modular development

Extends layered approach to parallel processing environment

PVL Benefits

• Portability
  • Network-of-Workstations
  • Multiprocessors

• High performance

• Increased productivity

• Simplified processor mapping, computation, and communication

• Teaming with Government HPEC-SI effort to define new library standard

• Working with industry to develop AEGIS SPY and Missile Processor PVL Versions
Scalable Code Development

```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;
}
```

- Code is the same
- Only map changes to for new computation and comm. distribution

Scalable code requires very limited changes as number of processor changes
Important in development phases: don’t work on entire processor system
Rapid Prototyping Development Model

Workstation → Workstation → Network of Workstations → DY4 Quad G4 Board

- Implement algorithm in Matlab and verify
- Implement in PVL and verify
- Verify parallel behavior
- Optimize

Better development tools → Better performance

Development model verifies one aspect of the system at a time
Outline

• Introduction
• Signal Processor Architecture
  – Hardware
  – Software
• System Implementation and Demonstration
• Summary
# Productivity: PVL Code Size

<table>
<thead>
<tr>
<th></th>
<th>Matlab LOC</th>
<th>PVL/C++ LOC</th>
<th>Matlab-to-C Compiler LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADNUC</td>
<td>36</td>
<td>143</td>
<td>538</td>
</tr>
<tr>
<td>Dither Sub.</td>
<td>28</td>
<td>160</td>
<td>437</td>
</tr>
<tr>
<td>Integration</td>
<td>11</td>
<td>63</td>
<td>178</td>
</tr>
<tr>
<td>CFAR</td>
<td>2</td>
<td>60</td>
<td>90</td>
</tr>
<tr>
<td>Bulk Filter</td>
<td>5</td>
<td>152</td>
<td>211</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>82</strong></td>
<td><strong>578</strong></td>
<td><strong>1454</strong></td>
</tr>
</tbody>
</table>

- PVL/C++ Code includes
  - Signal processing code
  - Task and data parallel code
  - Embedded control code
- Matlab and Converted C code includes only
  - Signal processing code

Estimate LOC for an embedded, parallel C program

\[
= 1500 \times \text{Expansion factor}^{1}
\]

= 1500 \times 2 \approx 3000

\(^{1}\text{Expansion Factor = 2-3 based on previous projects}\)

C++/PVL saves substantial development effort
Porting PVL to the Quad G4 Board

Porting Challenges:
- Network of Workstations to embedded hardware
- Unix/Linux to VxWorks
- Workstation tools to embedded tools
- MPI library to DMA library

Network of Workstations

Quad G4 Embedded System

Vendor libraries

PVL

VSIPL

MPI

GCC

Total View

Purify

... Unux/Linux

Tools

OS

Hardware

Application

Vendor libraries

PVL

VSIPL

MPI

GCC

Total View

Purify

... Unix/Linux

Tools

VxWorks

Scope Pak

Cross Wind

Cross GCC

...
Performance

Required Throughput: 60 frames/sec   Required Latency: ~1/60 sec = 16.7 msec

1/9/02: 3460 msec
ADNUC 2120 msec  Dither Processing 507 msec  Multi-Frame Processing 48 msec  CFAR Detection 652 msec
Index Vector Optimization
2/11/02: 1020 msec
ADNUC 448 msec  Dither Processing 507 msec  Multi-Frame Processing 48 msec  CFAR Detection 22 msec
Compiler Optimizer
2/12/02: 136 msec
ADNUC 43.1 msec  Dither Processing 85.2 msec  Multi-Frame Processing 4.5 msec  CFAR Detection 2.6 msec
Optimized VSIP
4/24/02: 21.0 msec
ADNUC 10.8 msec  Dither Processing 6.2 msec  Multi-Frame Processing 1.8 msec  CFAR Detection 1.0 msec
Data Vectorization
9/13/02: 14.7 msec
ADNUC 6.91 msec  Dither Processing 3.95 msec  Multi-Frame Processing 1.03 msec  CFAR Detection 0.92 msec  Subpixel Processing 0.26 msec

Get it right, then get it fast: Currently at ~0.9x real-time
Current Status of Signal Processing Architecture: SM3

Focal Plane Array

Integrator  A/D  Guidance

To
thrusters

Background Estimation  Adaptive Nonuniformity Compensation  Dither Processing  CFAR Detection

Multi-Frame Processing  Bulk Filter*

From other sensors

Target Selection (Discrimination)  Feature Extraction  Tracking  Subpixel Processing

MATLAB  PVL – Network of Workstations

SM-3 Signal Processing Model
Signal Processing Chain Demonstration

IR Focal Plane(s)

A/D → Adaptive Non-Uniformity Compensation → Dither Processing

Multi-Frame Processing → CFAR Detection

Sub-Pixel Processing → Tracking

Target

No Target

Normalized Pixel Intensity

0

0.2

0.4

0.6

0.8

1

Target

No Target

Normalized Pixel Intensity

0

0.2

0.4

0.6

0.8

1
Outline

• Introduction
• Signal Processor Architecture
  – Hardware
  – Software
• System Implementation and Demonstration
• Summary
Summary and Future Work

• Next generation missile seeker processor development methodology under development
• COTS-based Development hardware platforms has been selected & assembled
• Layered, standards-based, software approach for portable and easily upgradeable application code has been developed
• Real-time seeker processing demonstrations now operational
• Demonstrating advanced Project Hercules algorithm performance on RT platform
• Future work:
  – Demonstrate operation with sensor testbed
  – Continue development and porting of back-end and advanced algorithms to RT platform