Using VSIPL as an Embedded DoD Application Programming Interface (API) on DARPA Polymorphous Computing Architectures

High Performance Embedded Computing Workshop 2006
MIT Lincoln Laboratory
19 September 2006

Joe Cook / Kristin Daly / Rick Pancoast - Lockheed Martin MS2
Steve Crago / Matt French / Karandeep Singh / Jinwoo Suh - USC-ISI
How Will DoD System Integrators Develop / Program / Integrate & Test PCA Architectures?

- Software Design and Development Costs Far Exceed the Hardware Development Costs for a Large DoD Program

- When New Chips and Architectures Become Available to the Community, How Do We Design Them into DoD Systems?

- DoD System Integrators Cannot Afford to Develop the Specialized Talent Required to Program Unique Chips and Architectures at the Lowest Levels (e.g. Microcode Applications and Fine Grain Data Movement)

- One Solution to the Dilemma Pursued by Lockheed Martin MS2 is to use Industry Standard APIs (e.g. MPI, VSIPL, HPEC-SI / VSIPL++, CORBA, Data Re-Org, etc.) to Implement the Application
  - The Portability Provided Significantly Reduces Software Development Cost and Enhances Re-Targeting and Reuse

- We Have Implemented One of Our Standard Processing Benchmarks, Radar Pulse Compression, on the PCA RAW Chip Architecture and the PCA TRIPS Simulator, Using a C VSIPL API Wrapper
AMP Phase 2 VSIPL / RAW Demo Plan

- Develop a MATLAB Program to Perform Frequency Domain Pulse Compression using an FFT - Complex Multiply by a Reference Function - Inverse FFT Algorithm; Provide Input Data Set, Frequency Domain Reference Waveform, and Output Data Set for Comparison [Complete - Has Been Run @ USC-ISI]

- Develop a C / VSIPL Generic Implementation (Using Randy Judd’s C - VSIPL Reference Library) of the MATLAB Pulse Compression Algorithm [Complete - Has Been Run @ USC-ISI]

- Develop a C Program that will Execute on the Raw Processor, and Execute a Streaming VSIPL (Wrapper) Algorithm (e.g Pulse Compression) on the RAW Processor on the Handheld Board [Complete - Has Been Run @ USC-ISI]

- Provide a Host Demonstration GUI in Java using Ptolemy Ptplot [Complete - Has Been Run @ USC-ISI]

- Morph the Environment to Switch to a Threaded Algorithm Process (e.g. Integrated Radar Search & Track Tracking Algorithm) on RAW; [Complete - Has Been Run @ USC-ISI]

- Compare the Output from RAW (Validate) with the Output From the MATLAB Simulation and Demonstrate for USC-ISI, DARPA and the Navy / MDA; Develop a DoD Transition Plan
Approach to Demonstrating VSIPL Pulse Compression With PCA Raw Architecture:

- Compile the C / VSIPL Code on the Host Processor
- Input Data (via File I/O) is Plotted on the Host Processor, then Downloaded to the PCA Raw Processor Chip
- Run the C / VSIPL Streaming Code Directly on Raw, Substituting Raw Assembly or Microcoded Functions (Primitives) that are Optimized for the Raw Architecture
- Output PC Data back to the Host Processor, store in a File and Plot the PC Output
- Morph the Raw Architecture for the Threaded VSIPL Track Processing on Raw
- Output Track Data back to the Host Processor, store in a File and Plot the Track Output
- The Above Cycle Repeats to Represent Real-Time Repetitive Radar Pulse Compression Operations
PCA Pulse Compression
VSIPL Demonstration

VSIPL (Vector, Signal, and Image Processing Library) standard application interface (API) used to program high-performance embedded systems.

VSIPL-based interface between Application Layer and Raw processor.

15x improvement over C implementation on Pentium III implemented in similar technology.
Morphing in PCA Pulse Compression Demonstration

- **Morphing types**
  - Static, compile-time
    - Each tile has different input and output connections determined at compile time
  - Dynamic, run-time
    - Communication topology changes between different stages of the computation

- **Morphing support**
  - Programmed static operand network implements topology morphing
  - Morphs are initiated by VSIPL host calls

- **Morphing cost**
  - Network switch processor can change topologies in a single processor cycle when topologies are pre-compiled

- **Benefits**
  - Communication latency reduced by over two orders of magnitude compared to software message passing (e.g. MPI latency is \(~1.2\) microseconds, Raw inter-tile latency is \(<10\) ns)
  - Estimated application performance improvement: 2x (over multi-core chip using MPI to communicate)
MATLAB VSIPL / RAW Data Sets

Pulse Compression Input (MatLab)
- 1 KHz PRF (1ms PRI)
- 20 MHz sampling rate
- 870 samples
- Echo (Envelope Shown)
  - 10 µs pulse
  - Linear FM chirp up
  - 200 samples
- Pulse Shifted to Simulate Object Range Movement

Pulse Compression Frequency Domain Reference (MatLab)
- Frequency Domain Reference
- 10 µs
- Linear FM chirp up
- 1024 complex samples
- Hamming weighting
- Bit-reversed to match optimized implementation possible

Pulse Compression Output (MatLab)
- 671 samples out of Pulse Compression
- Peak Indicates Detection / Range
- Range Shifts In Accordance With Input Pulse Shift
AMP VSIPL / RAW Demo Display

Input Data | Output Data
--- | ---
Stream | Stream | Stream | Threaded

Tracker Data

Display

Application
- C Code (Pulse Compression & Tracker)
- VSIPL API
- Optimized Raw Assembly (FFT, CMPY, IFFT)
- Raw Hardware

Distribution Statement “A” - Approved for Public Release, Distribution Unlimited
void pulseCompress(in, ref, out)

{...

Previously Developed
Generic Pulse Compression Code ...

void pulseCompress( vsip_cvview_f * in, vsip_cvview_f* ref, vsip_cvview_f* out)

{ vsip_length size = vsip_cvgetlength_f(in);

//FFT OBJECT SETUP REQUIRED BY VSIPL
vsip_fft_f *forwardFft = vsip_ccfftop_create_f(size, 1.0, VSIP_FFT_FWD,1,VSIP_ALG_SPACE);
vsip_fft_f *inverseFft = vsip_ccfftop_create_f(size, 1.0/size, VSIP_FFT_INV, 1, VSIP_ALG_SPACE);

//TEMPORARY VIEWS TO HOLD INTERMEDIATE OUTPUTS
vsip_cvview_f *tmpView1=vsip_cvcreate_f(size,VSIP_MEM_NONE);
vsip_cvview_f *tmpView2=vsip_cvcreate_f(size,VSIP_MEM_NONE);

//FORWARD FFT
vsip_ccfftop_f(forwardFft,in,tmpView1);

//COMPLEX MULTIPY BY REFERENCE WAVEFORM
vsip_cvmul_f(tmpView1,ref,tmpView2);

//INVERSE FFT
vsip_ccfftop_f(inverseFft,tmpView2,out);

//CLEAN-UP
vsip_cvalldestroy_f(tmpView1);
vsip_cvalldestroy_f(tmpView2);
vsip_fft_destroy_f(forwardFft);
vsip_fft_destroy_f(inverseFft) }

Previously Developed
Generic Pulse Compression Code
void pulseCompress(in, ref, out)
{... Generic Pulse Compression Code ... }

VSIPL API Interface

Optimized VSIPL for Raw

Optimized VSIPL for Platform B

Raw Hardware

Traditional Platform

Common VSIPL interface allows Application Designer to develop “Write Once, Use Anywhere” Code

Reduced Portability Costs, and Increasing Platform Options
I/O Bandwidth Issues

- Increased computational resources make managing bandwidth more critical
  - I/O has not scaled as fast as computational power
  - Architects have always known this, but the software has not caught up

![Diagram showing memory, network I/O, traditional processor, and Scaled Tiled Architecture]
Interprocessor Communication

- Low latency connections between compute tiles expose new software issues
- Library-based message passing paradigm insufficient

Traditional Multiprocessor

<10 cycles latency between tiles
Important Software Issues

- Inter-procedural optimizations critical
  - Data cannot be sent to off-chip memory modules between all computations
- Compiler must understand data movement issues
  - Language cannot obscure data flow

- PCA just beginning to address these issues
  - R-stream, StreaMIT, SVM
- HPEC-SI just starting to address multiprocessors
  - Focused on standardization and tech transfer, not research
  - Just starting to think about tiled architectures
PCA Raw / HPEC-SI Demo Summary

- Frequency Domain Pulse Compression Demonstration is Complete Using C / VSIPL API, Implemented on RAW
  - A Very Simple VSIPL Wrapper Has Been Demonstrated - For DoD Success, a More Comprehensive (VSIPL Core Light?) Native Library Would be Needed for Each Targeted Architecture

- Successfully Morphed the Environment to Switch to a Threaded Algorithm Process (Tracker) on RAW Using a Demonstration GUI

- Validated the Output from RAW and Demonstrated for USC-ISI; Demonstrations Planned for DARPA and the Navy / MDA

- Lockheed Martin has Initiated Briefings and Demonstrations for Key Insertion Programs (such as Aegis Ballistic Missile Defense Signal Processor) and Will Develop a DoD Spiral Transition Concept

- Desire to Run Similar Demonstration Benchmarks (Using a High Level API, such as VSIPL or VSIPL++) on Other PCA Architectures, such as TRIPS, MONARCH and SMART MEMORIES
  - Initially Using Simulator Tools
  - Ultimately on the Actual Hardware

Distribution Statement “A” - Approved for Public Release, Distribution Unlimited
PCA TRIPS / HPEC-SI Demo

- Frequency Domain Pulse Compression Demonstration is Complete Using C / VSIPL API, Implemented on TRIPS Simulator
  - Same C / VSIPL Code As Run on the RAW Hardware with NO Modifications; Used Same Java-Based GUI for Display

- Validated the Output from the TRIPS Simulator and Demonstrated the GUI Display; Demonstrations Planned for DARPA and the Navy / MDA

- Desire to Continue to Run Similar Demonstration Benchmarks (Using a High Level API, such as VSIPL or VSIPL++) on Other Non-Conventional Computing (NCC) Architectures
  - Initially Using Simulator Tools, if Necessary
  - Ultimately on the Actual Hardware
Lockheed Martin Has Initiated Discussions With IBM Systems & Technology Group to Investigate Porting CodeSourcery’s Sourcery VSIPL++ API to the Cell Broadband Engine for DoD Applications
PCA to DoD Transition Plan

- Lockheed Martin MS2 is Writing Real-Time Embedded Signal Processing Application Code Using Industry Standard APIs (MPI, VSIPL, VSIPL++, etc.) for Next Generation Shipboard Ballistic Missile Defense (and Other Applications)

- PCA Architectures May Provide a Significant Advantage in Size, Weight, Power, Cost, etc. for DoD Applications; They Need to be Demonstrated

- Industry Standard APIs and Middleware on PCA and Non-Conventional Computing Architectures Will Provide the Portability Necessary to Transition Mainstream PowerPC Applications to NCC Architectures for DoD

- Demonstrations for Other Lockheed Martin Businesses and DoD Program Managers Will Provide Metrics for Improving SWEPT
Acknowledgements

- Raw and TRIPS work presented was funded by DARPA IPTO on the Polymorphous Computing Architectures (PCA) Program

- Material contained herein is the opinion of the author, and does not imply endorsement by the US Government.

- Thanks to USC-ISI East, Steve Crago, Matt French, Karandeep Singh and Jinwoo Suh, for Support in Utilizing the Raw Processor Board and the TRIPS Simulation Environment in the USC-ISI Laboratory.