Implementation of an Embedded DoD VSIPL Application on the DARPA Polymorphous Computing Architectures (PCA) Raw Processor

High Performance Embedded Computing Workshop 2005

MIT Lincoln Laboratory

22 September 2005

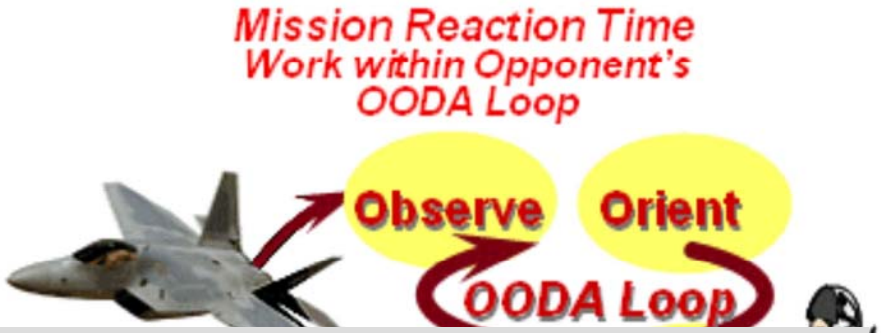
Joe Cook / Louis Morda / Rick Pancoast - Lockheed Martin MS2

Steve Crago / Jinwoo Suh - USC-ISI

LOCKHEED MARTIN

Maritime Systems & Sensors

DARPA is Developing Polymorphous Computing Architectures (PCA)

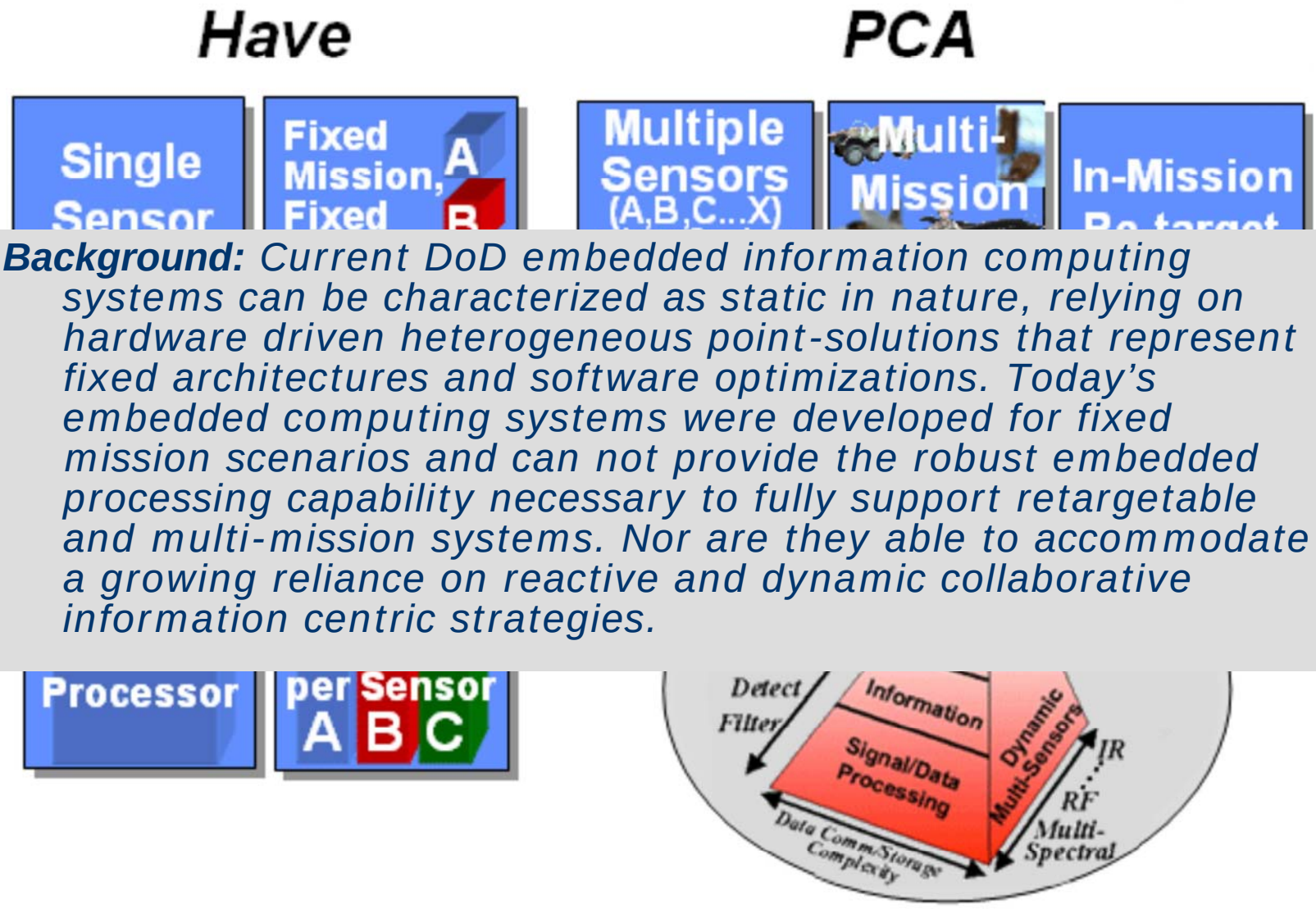


Mission: *Enable reactive multi-mission and in-flight retargetable embedded information computing systems that will reduce mission computing payload adaptation, optimization, and verification times from months and years to minutes. Develop processing architectures that can reconfigure and adapt to mission requirements.*

Software First – Hardware Last

Extended Sensor System Service Life

DARPA is Developing Polymorphous Computing Architectures (PCA)



Background: *Current DoD embedded information computing systems can be characterized as static in nature, relying on hardware driven heterogeneous point-solutions that represent fixed architectures and software optimizations. Today's embedded computing systems were developed for fixed mission scenarios and can not provide the robust embedded processing capability necessary to fully support retargetable and multi-mission systems. Nor are they able to accommodate a growing reliance on reactive and dynamic collaborative information centric strategies.*

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 Begun to Implement One of Our Standard Processing Benchmarks, Radar Pulse Compression, on the PCA Raw Chip Architecture from MIT, Using a C VSIPL API***

Different Approaches to Using VSIPL With PCA Architectures:

- ***Run the VSIPL / VSIPL++ API on the Host Processor, and Call Assembly or Microcoded Functions (Primitives) that are Optimized for the Particular Architecture of Interest (PCA, FPGA, etc.)***
  - ***This is the Approach Selected for the Pulse Compression VSIPL / RAW Demonstration, Since it is Probably the Easiest to Implement and we Have the Necessary Primitive Functions***
- ***VSIPL++ Calls on the Host Invoke an Optimized C++ Compiler for the Attached PCA Architecture***
  - ***This appears to be the approach that UT Austin is taking with the TRIPS C compiler***
- ***C++ / VSIPL++ is Compiled and Optimized Directly for the PCA Architecture of Interest, Viewed as a "Parallel Processor" that Exploits Parallel VSIPL++ Optimizations***
- ***There Are Probably Other Approaches That Will Evolve***

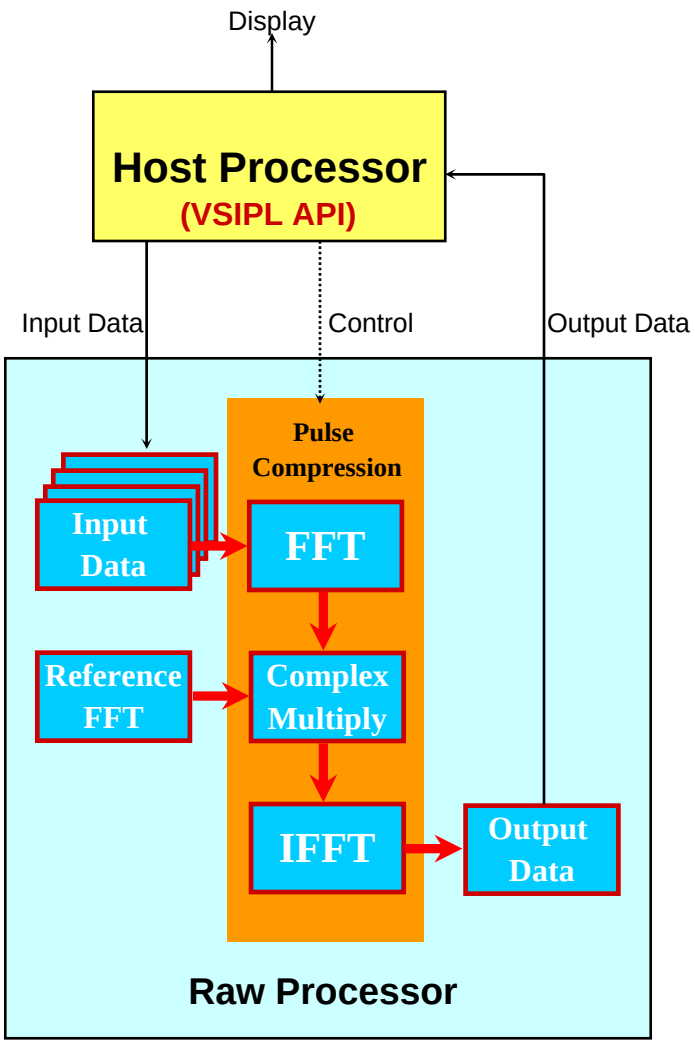
AMP Phase 2 VSIPL / Raw Demo Plan

- *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]*
- *Develop a C / VSIPL Generic Implementation (Using Randy Judd's C - VSIPL Reference Library) of the MATLAB Pulse Compression Algorithm [Complete]*
- *Develop a C Program that will Execute on the Host Processor, and Execute a Streaming VSIPL (Wrapper) Algorithm (e.g Pulse Compression) on the Raw Processor on the Handheld Board [VSIPL Running on Raw Simulator - VSIPL on Raw In-Progress]*
- *Provide a Demonstration GUI in Java using Ptolemy Ptplot [Prototype GUI is Running]*
- *Morph the Environment to Switch to a Threaded Algorithm Process (e.g. Integrated Radar Search & Track Tracking Algorithm) on Raw; [In-Progress]*
- *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 Using VSIPL Pulse Compression With PCA Raw Architecture:

- ***Run the VSIPL / VSIPL++ API on the Host Processor, and Call Assembly or Microcoded Functions (Primitives) that are Optimized for the Raw Architecture***
- ***Input Data is Plotted on the Host Processor then Downloaded to the PCA Raw Processor Chip***
- ***Graphic on Host Processor Indicates which Raw Tiles are Executing which VSIPL Pulse Compression Function Primitives***
- ***Output Data is Uploaded to the Host Processor from the PCA Raw Chip then Plotted on the Host Processor***
- ***The Above Cycle Repeats to Represent Real-Time Repetitive Radar Pulse Compression Operations***

AMP Phase 2 Demo Approach



– Data Sets Verified by Running Through a MATLAB Pulse Compression Algorithm Offline

– Data Transferred From Host Processor to Raw Processor

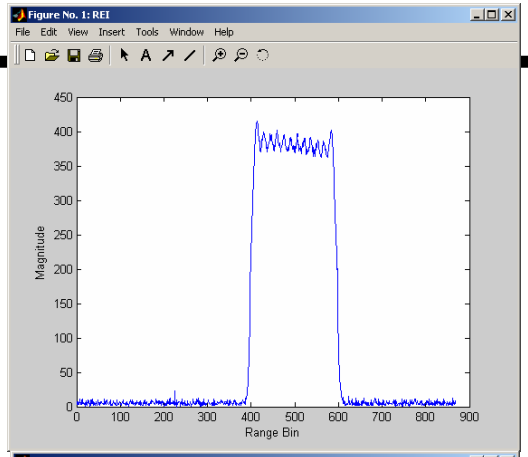
–1K FFT, Complex Multiply, and IFFT implemented on Raw Chip; Invoked via VSIPL / VSIPL++ API in the Host Processor

– Pulse Compression based upon existing Raw Primitives: FFT, complex multiply by a realistic Reference Function, & IFFT

– Morph from a Streaming Pulse Compression Process to a Threaded Track Process

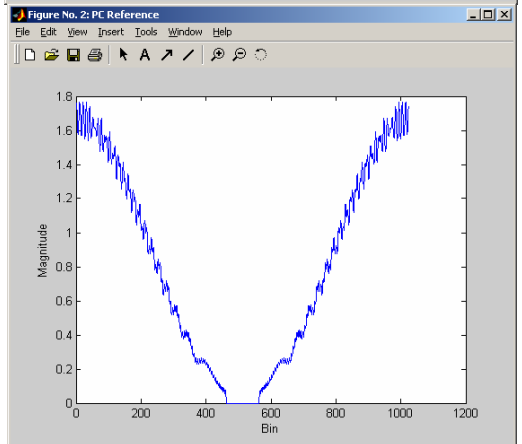
MATLAB VSIPL / Raw Data Sets

- **Pulse Compression**
- **Input (MatLab)**



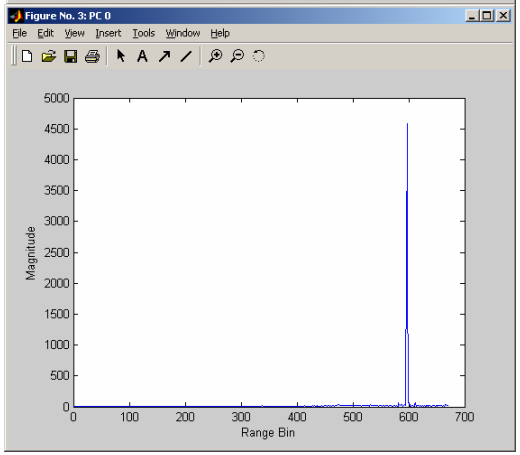
- 1 KHz PRF (1ms PRI)
- 20 MHz sampling rate
- 870 samples
- Echo (Envelope Shown)
  - *10 µs pulse*
  - *Linear FM chirp up*
  - *200 samples*
- Pulse Shifted to Simulate Object Range Movement

- **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)**

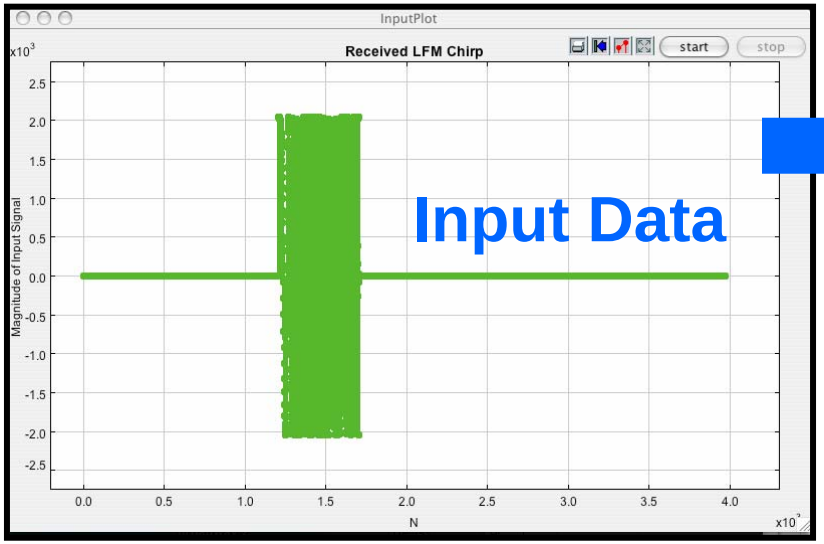


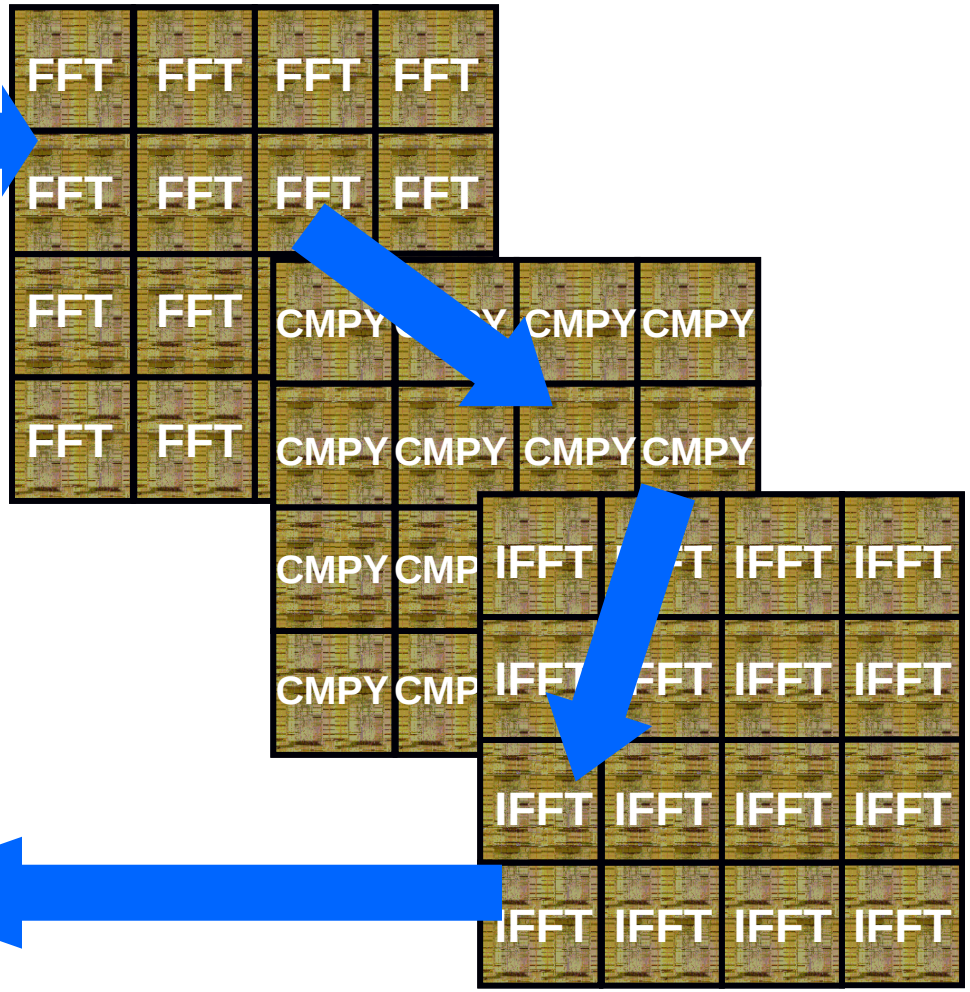
- 671 samples out of Pulse Compression
- Peak Indicates Detection / Range
- Range Shifts In Accordance With Input Pulse Shift

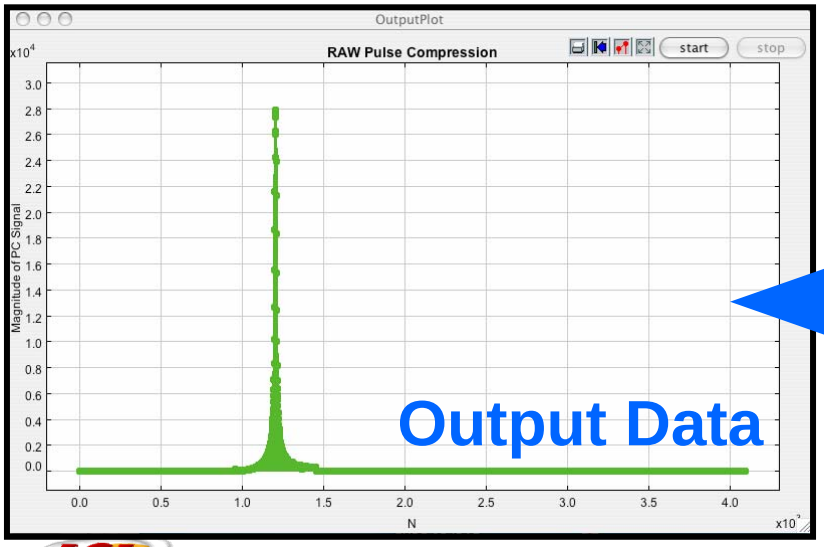
Information Sciences Institute

LOCKHEED MARTIN
Maritime Systems & Sensors

AMP VSIPL / Raw Demo Display







Information Sciences Institute

LOCKHEED MARTIN
Maritime Systems & Sensors

AMP VSIPL / Raw Demo Code

void pulseCompress(in, ref, out)

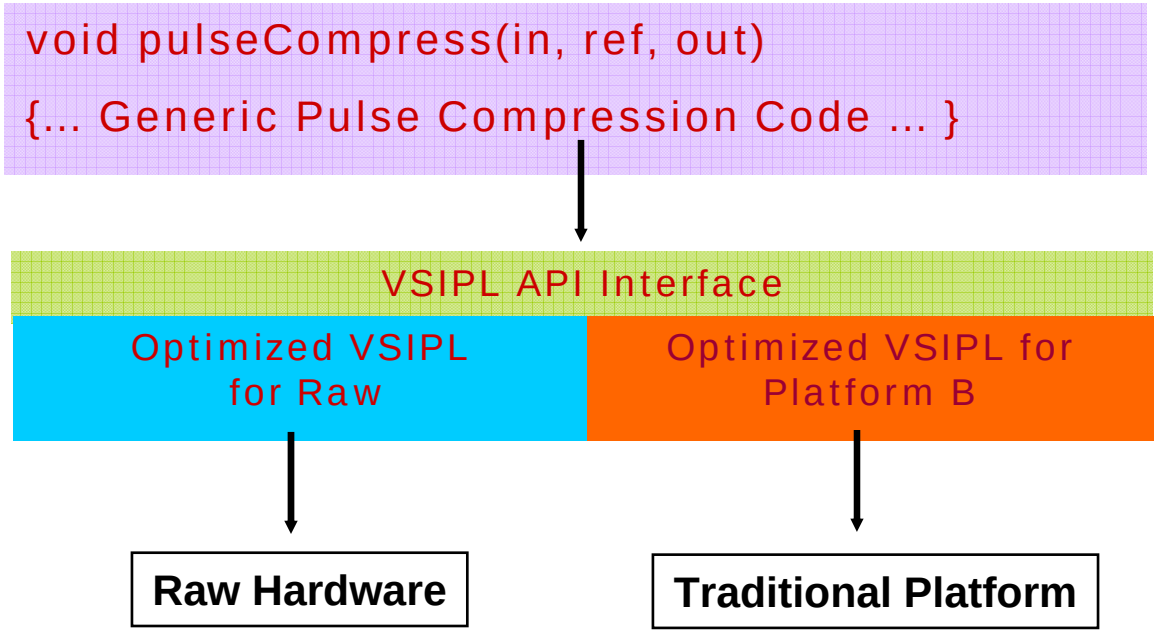
{... Generic Pulse Compression Code ... }

```
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

AMP VSIPL / Raw Demo Code

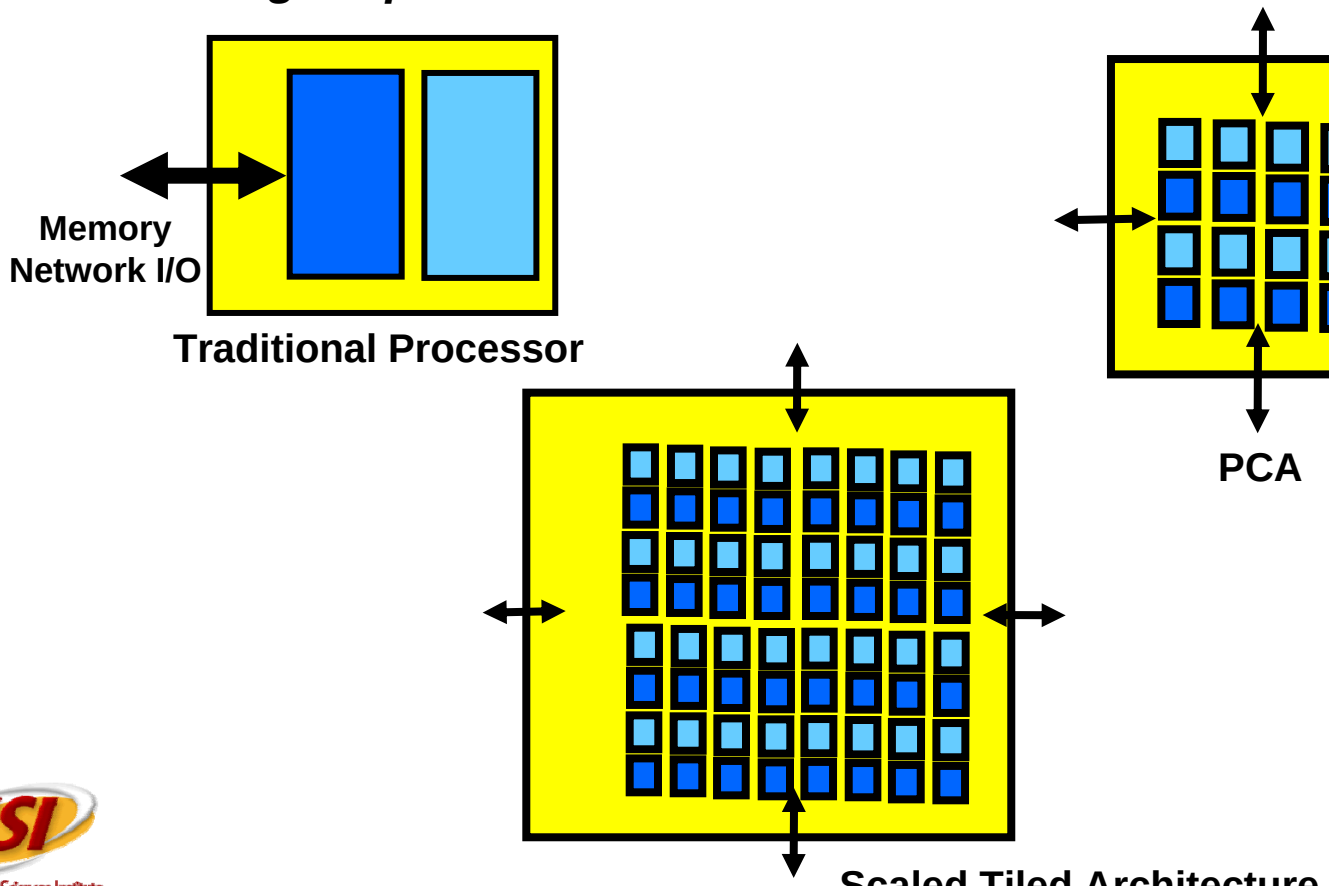


Common VSIPL interface allows Application Designer to develop "Write Once, Use Anywhere" Code

Reduced Portability Costs, and Increasing Platform Options

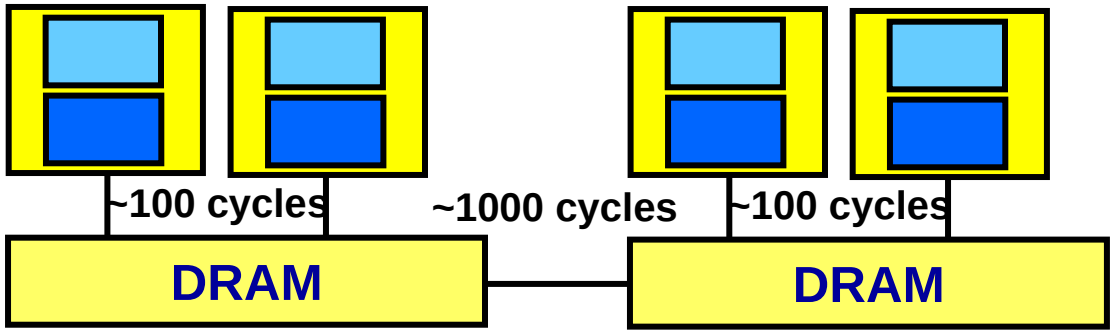
I/O Bandwidth Issues

- *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*

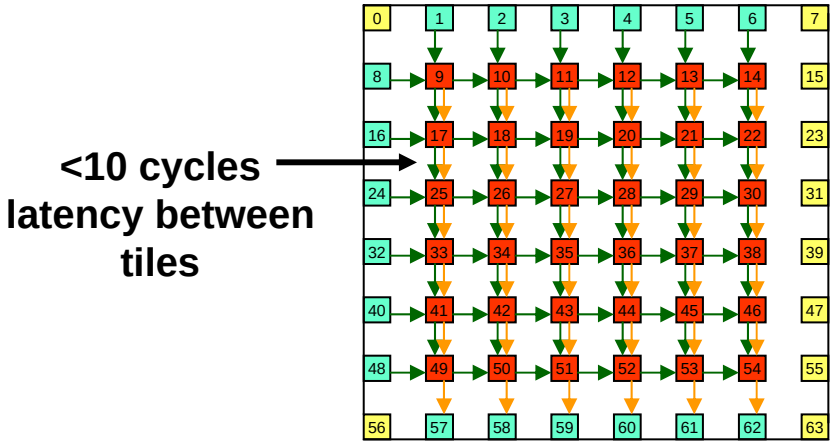


Interprocessor Communication

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



Traditional Multiprocessor



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 Planned for 4Q CY2005 Using C / VSIPL API, Implemented on Raw***
- ***Morph the Environment to Switch to a Threaded Algorithm Process on Raw Using a Demonstration GUI***
- ***More compiler/software problems to be solved to optimize for PCA architectures***
- ***Validate the Output from Raw and Demonstrate for USC-ISI, DARPA and the Navy / MDA; Develop a DoD Transition Plan***

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***
- ***Industry Standard APIs and Middleware on the PCA Architectures Will Provide the Portability Necessary to Transition Mainstream PowerPC Applications to PCA Architectures for DoD***
- ***Demonstrations for Other Lockheed Martin Businesses and DoD Program Managers Will Provide Metrics for Improving SWEPT***

Acknowledgements

- ***Work presented is 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 and Jinwoo Suh, for Support in Utilizing the Raw Processor Board in the USC-ISI Laboratory.***