AFRL-IF-RS-TR-2006-315
Final Technical Report
October 2006

STANDARDIZATION OF OBJECT ORIENTED EXTENSIONS TO VECTOR SIGNAL AND IMAGE PROCESSING LIBRARY (VSIPL)

Georgia Tech Applied Research Corporation

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO FINAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Rome Research Site Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-IF-RS-TR-2006-315 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
STANLEY LIS Work Unit Manager	JAMES A. COLLINS, Deputy Chief Advanced Computing Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCT 2006	Final	May 05 – May 06

4. TITLE AND SUBTITLE

STANDARDIZATION OF OBJECT ORIENTED EXTENSIONS TO VECTOR SIGNAL AND IMAGE PROCESSING LIBRARY (VSIPL)

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-05-1-0217

5c. PROGRAM ELEMENT NUMBER
63755D

6. AUTHOR(S)

Daniel P. Campbell

5d. PROJECT NUMBER
HPEC

5e. TASK NUMBER
SI

5f. WORK UNIT NUMBER
06

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Georgia Tech Applied Research Corp
Centennial Research Building
Georgia Institute of Technology
Atlanta GA 30332

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFTC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-315

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 06-730

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Vector Signal and Image Processing Library (VSIPL) is an industry standard Application Programming Interface for embedded signal processing tasks. The High Performance Embedded Computing Software Initiative (HPEC-SI) program is a collaborative program to establish extensions to the VSIPL specification to support Object Oriented elements of the C++ programming language, and encapsulated support for data parallel processing. The program goals include the simultaneous threefold improvement in software portability, threefold improvement in developer productivity, and fifty per cent improvement in software performance compared to standard practices. This report describes the efforts of the Georgia Tech Research Institute in support of the HPEC-SI program during the period from May 2005 through April 2006. These efforts included development of functional prototypes, and organizational strategies, participation in the HPEC-SI Applied Research and Development Working Groups, dissemination of program results to outside organizations via conference presentations and internet tools, and maintenance of a parallel computing software testbed for program participants.

15. SUBJECT TERMS

VSIPL C++ Object Oriented High Performance Software

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Stanley Lis
U	U	U	UL	17	19b. TELEPHONE NUMBER (*Include area code*)

Table of Contents

List of Figures

List of Tables

Table of Acronyms

API	Application Programming Interface
DARPA	Defense Advanced Research Program Agency
DRI	Data Reorganization Initiative
FFT	Fast Fourier Transform
GTRI	Georgia Tech Research Institute
HPEC-SI	High Performance Embedded Computing Software Initiative
MPI	Message Passing Interface
MSI	Morphware Stable Interface
PCA	Polymorphous Computing Architectures
SPMD	Single Program Multiple Data
VSIPL	Vector Signal & Image Processing Library

Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-05-1-0217. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

1. Introduction

During the period of 4 May 2005 – 3 May 2006, the Georgia Tech Research Institute (GTRI) engaged in the project "Standardization of Object Oriented Extensions to VSIPL," in support of the High Performance Embedded Computing Software Initiative (HPEC-SI) Program. The Vector Signal and Image Processing Library (VSIPL) [1] is an industry standard Application Programming Interface for embedded signal processing tasks. The High Performance Embedded Computing Software Initiative (HPEC-SI)[2] program is a collaborative program to establish extensions to the VSIPL specification to support Object Oriented elements of the C++ programming language, and encapsulated support for data parallel processing. GTRI contributed to the HPEC-SI forum objectives by assisting in the development of Object Oriented VSIPL standards, co-chairing the HPEC-SI Development Working Group, and assisting in the dissemination of technical designs and ideas.

2. Tasks Completed

GTRI provided specific standards prototypes for the Working Group for discussion and implementation. GTRI served on the Technical Advisory Board of the HPEC-SI program. GTRI served as co-chair for the HPEC-SI Development Working Group. GTRI maintained the website [2] for the HPEC-SI effort, which served as a collection point for information about the effort, as well as information presented at the meetings, for the forum members.

GTRI attended working group meetings in June 2005, August 2005, December 2005, and April 2006 in support of program efforts. At these meetings, GTRI participated in discussions vetting the proposed VSIPL++ and parallel VSIPL++.

GTRI attended the High Performance Computing Modernization Program Users Group Conference in Nashville, TN, in June 2005, to present progress on the HPEC-SI Development Working Group. The abstract of the technical paper submitted for that conference is included below:

> The High Performance Embedded Computing Software Initiative (HPEC-SI) program is developing a unified computation and communication Application Programming Interface (API) and framework for high performance signal processing tasks on parallel computers. The goal of the program is to address the high cost of software in Department of Defense (DoD) systems by improving the portability and productivity of signal processing application development threefold, while improving performance by one half compared to current practices. This paper describes the motivation for the HPEC-SI program, its goals and approaches, and progress of the HPEC-SI Working Groups in extending the Vector, Signal, and Image Processing Library (VSIPL) standard to C++ and transparent operation in parallel computing systems. The C++ extension to VSIPL is described, and highlights of its advantages are considered. This paper also examines results from the Demonstration Working Group, and describes requirements and plans developed by the Applied Research Working Group for data parallel extensions to VSIPL and describes Development Working Group progress so far in developing parallel VSIPL.

GTRI continued development and maintenance of the HPEC-SI Program website, including meeting notes and presentations for each of the Working Group meetings during the project period, conference presentations, and prototype applications. In conjunction with this, GTRI continued to maintain email reflectors for use by HPEC-SI program participants.

GTRI made a parallel computing cluster available for use by HPEC-SI program [3] participants as a testbed for VSIPL++, parallel VSIPL++, and other parallel computing systems. The cluster is a fifty node Beowulf style cluster with 104 compute processors of varying types. Several HPEC-SI program participants were given login access and have used the testbed. Software support, including the installation of software and tools useful to the HPEC-SI program was begun.

GTRI maintained a VSIPL++ User's Guide [4]. The Guide is intended to complement the VSIPL++ Specification Document and serve as an introduction and clarification of the Specification for application programmers. The guide contains elaborations and examples from portions of the VSIPL++ Specification that the HPEC-SI Working Groups find to be difficult or confusing for new VSIPL++ application programmers. The choices of topics draw heavily from the experiences of the projects undertaken by the Demonstration Working Group, as well as the vetting and clarification of the VSIPL++ Specification by the Development Working Group.

GTRI continued its advisory role on the Technical Advisory Board of the HPEC-SI program, as well as serving as the co-chair of the Development Working Group.

3. Results

Parallel VSIPL++ Specification

GTRI participated in the conceptual design, detailed specification, vetting, and verification of the Parallel VSIPL++ specification [5], as well as the consideration and adoption of the specification by the VSIPL Forum. GTRI was not the primary author of the Parallel VSIPL++ specification, but nevertheless was actively involved in the listed aspects of its development.

Parallel VSIPL++ is primarily focused on the productive facilitation of Single Program Multiple Data (SPMD) style data parallelism on a low latency parallel computing system. In common parallel computing practice, in the absence of Parallel VSIPL++, the process of splitting vectors and matrices into pieces, distributing the pieces, redistributing, and combining the pieces during and after communication are mostly static and well established processes. Nevertheless, these steps typically require a large number of instructions to fully specify, and typically embed a large amount of information about the platform configuration directly in the source code of the software. Use of middleware standards and libraries such as the Message Passing Interface (MPI) and the Data Reorganization Interface (DRI) have mitigated this problem, but the communication and organization of parallel data remains a large source of lines of code that are not directly related to algorithm specification, and are heavily platform configuration dependent. These problems significantly have traditionally hindered the productivity and portability of parallel software. Parallel C++ VSIPL addresses these problems by adding a data distribution map argument to the constructor of blocks and views. Initial data distribution is instantiated by the data objects, and mathematical operators are responsible for data collection and

communication to and from data objects. The preferred method of specifying the mapping argument is via reference to an external distribution declaration that can be read at run time, but other mechanisms are supported. This approach addresses some of the problems with current methods of data parallel programming. Data distribution configuration information can be collected into one location per data object, and decoupled from algorithmic specification; and communication functionality will be abstracted away from the algorithmic specification of the application, and encapsulated by the math operators. These improvements improve the productivity of signal processing application development by reducing the number of lines of code required to achieve common tasks, and improve the portability of applications by significantly reducing the amount of rewrite required in order to deploy an application to a new platform configuration.

The Parallel VSIPL++ Specification is a document which defines the Parallel VSIPL++ API in terms relative to the VSIPL++ Specification [6]. It is relatively concise, and describes the additional functionality required to support Parallel applications using VSIPL++. The primary additions are the addition of a `map` object type, and updated functionality for the various `view` data types. The map object type is the primary mechanism for describing the data distribution of VSIPL++ view objects. Maps support block, block-cyclic, and cyclic data distributions, with various controls for sizes of blocks and degree of cyclicity. The map type also provides various support services, such as mechanisms to access the indices of a view that are on the local node, or within a particular subblock. The updates to the view type allow the distribution of views over a set of processors or nodes, as described by the map type provided at the initialization of the view. A variety of additional support services are defined for the view type for proper and intuitive behavior within the context of a parallel application, including, for example, obtaining local subviews, identification of the local region of a view, etc.

The full text of the Parallel VSIPL++ Specification can be obtained electronically at http://www.hpec-si.org/spec-par-1.0-final.pdf

VSIPL++ User's Guide

GTRI led the development of a User's Guide for VSIPL++. The Guide was created by soliciting example application source code from the various HPEC-SI program participants, collecting them, and editing into a consistent format. An initial draft of the User's Guide was delivered to HPEC-SI program participants in June 2005, and a subsequent draft was delivered in December 2005.

The purpose of the User's Guide is to serve as a complement to the VSIPL++ Specification for application developers. The VSIPL++ specification fully defines the behavior of VSIPL++ implementations, and is focused on compactness and formal correctness rather than ease of reading. The Guide seeks to clarify important aspects of VSIPL++ application development.

The Guide is presented in the form of several illustrative examples that have been developed by VSIPL Forum members during the development and demonstration of the VSIPL++ Specification. Each example is in the form of VSIPL++ C++ source code, along with

descriptions, and in some cases equivalent VSIPL C source code for contrast. Each illustrates an element of VSIPL++ application development that users have found needing clarification.

The examples that are included in the VSIPL++ User's Guide are summarized in Table 1, below. The authors that have contributed examples and text to the Guide are summarized in Table 2, below. The VSIPL++ User's Guide is under continuous evaluation and expansion as new examples become available, and revisions to the VSIPL++ Specification are made. The most current version of the VSIPL++ User's Guide can be obtained electronically at http://www.hpec-si.org/VSIPL++%20User_s%20Guide%20Draft%20v0.2pdf. Table 3 and Table 4 capture specific examples that are in the current draft of the VSPL++ User's Guide.

Vector Add
No-Copy vector reference to matrix
Using User Defined Blocks
Simple FFT Example
Comparing VSIPL to VSIPL++ Simple Pulse Compression Case Studies
Importing and Exporting User Allocated Memory to VSIPL++ View Objects
Synthetic Aperture Radar VSIPL++ Example

Table 1 - VSIPL++ User's Guide Examples

Jules Bergmann
Susan Emeny
Randall Judd
Rick Pancoast
David Leimbach
Sharon Sacco
Brian Sroka

Table 2 - VSIPL++ User's Guide contributing authors

```
#include <vsip/initfin.hpp>
#include <vsip/matrix.hpp>
#include <vsip/vector.hpp>
#include <vsip/domain.hpp>
#include <iostream>

// display a row of the matrix with tabs.
std::ostream & operator << (std::ostream & os, vsip::Vector <> v) {
  int idx = 0;
  int size = v.size();
  for ( ; idx < size; ++idx)
    os << v.get(idx) << '\t';

  return os;
}

// Prints out a matrix row-wise
std::ostream & operator << (std::ostream & os, vsip::Matrix <> m) {
  int idx = 0;
  int size = m.size(1); /* m.size(1) is the size of a dimension of the matrix
*/
  for( ; idx < m.size(1); ++idx)
    os << m.row(idx) << std::endl;

  return os;
}

int main () {
  vsip::vsipl

  // defaults to scalar_f.
  vsip::Matrix<> m0 (4, 4, 0.0f); //4x4 Matrix with 0s

  // Show the contents of the matrix.
  std::cout << m0 << std::endl;

  // Each row_type is a reference to a row in the Matrix
  vsip::Matrix<>::row_type v00(m0.row(0));
  vsip::Matrix<>::row_type v01(m0.row(1));
  vsip::Matrix<>::row_type v02(m0.row(2));
  vsip::Matrix<>::row_type v03(m0.row(3));

  // Throw in some values diagonally.
  v01.put(1, 1.0f);
  v02.put(2, 2.0f);
  v03.put(3, 3.0f);

  // Show the original matrix
  std::cout << std::endl << m0 << std::endl;
}
```

Table 3 - No-Copy Vector Reference to Matrix (1.2)

VSIPL

```
void pulseCompress( int decimationFactor, vsip_cvview_f *in, vsip_cvview_f *ref,
vsip_cvview_f *out) {
  vsip_length savedSize   = vsip_cvgetlength_f(in);
  vsip_length savedStride = vsip_cvgetstride_f(in);
  vsip_length size = vsip_cvgetlength_f(in) / decimationFactor;

  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);
  vsip_cvview_f *tmpView1 = vsip_cvcreate_f(size, VSIP_MEM_NONE);
  vsip_cvview_f *tmpView2 = vsip_cvcreate_f(size, VSIP_MEM_NONE);
  vsip_cvputlength_f(in, size);
  vsip_cvputstride_f(in, decimationFactor);
  vsip_ccfftop_f(forwardFft, in, tmpView1);
  vsip_cvmul_f(tmpView1, ref, tmpView2);
  vsip_ccfftop_f(inverseFft, tmpView2, out);
  vsip_cvputlength_f(in, savedSize);
  vsip_cvputstride_f(in, savedStride);

  vsip_cvalldestroy_f(tmpView1);
  vsip_cvalldestroy_f(tmpView2);
  vsip_fft_destroy_f(forwardFft);
  vsip_fft_destroy_f(inverseFft);
}
```

VSIPL++

```
void pulseCompress(int decimationFactor,  const vsip::Vector< std::complex<float> >
      &in, const vsip::Vector< std::complex<float> > &ref
const vsip::Vector< std::complex<float> > &out) {
  int size = in.size() / decimationFactor;
  vsip::Domain<1> decimatedDomain(0, decimationFactor, size);

  vsip::Fft<vsip::Vector, vsip::cscalar_f, vsip::cscalar_f, vsip::fft_fwd> forwardFft
       ((vsip::Domain<1>(size)), 1.0);
  vsip::Fft<vsip::Vector, vsip::cscalar_f, vsip::cscalar_f, vsip::fft_inv, 0,
       vsip::SINGLE, vsip::by_reference> inverseFft ((vsip::Domain<1>(size)),
       1.0/size);

  inverseFft( ref * forwardFft( in(decimatedDomain) ), out );
}
```

Table 4 - Pulse Compression Comparison Example (1.5.5)

Parallel VSIPL++ for tiled architectures

GTRI participated in several discussions at HPEC-SI Working Group meetings, as well as additional meetings in support of defining possible approaches for Parallel VSIPL++ for tiled architectures. Examples of targeted tiled architectures include those developed under the DARPA Polymorphous Computing Architectures program, the IBM/Sony/Toshiba Cell Broadband Engine, and commodity multicore general purpose processors. The HPEC-SI

participants expressed a desire to leverage the software results obtained on the DARPA PCA program, therefore GTRI developed proposals for approaches that are based on integrating the Morphware Stable Interface [7], designed under the DARPA PCA program, into development flows of the HPEC-SI program. GTRI organized a joint meeting of the HPEC-SI and PCA programs for the purpose of allowing participants from each program to interact.

The baseline Morphware development flow is shown in Figure 1, below. This development flow is based around a two level compilation process. The higher level compiler translates programs from one of several input languages into a virtualized, abstract, architecture neutral intermediate language. This is then compiled by one of several platform specific backend systems into an executable for a particular platform. GTRI summarized and proposed several alternative approaches to augmenting the Morphware approach to support VSIPL++, and analyzed the impacts of the approaches. Two of the approaches were most widely considered by HPEC-SI program participants to be the most appropriate approaches for including tiled processor support in VSIPL++, and are discussed.

The first favored approach is to use the existing Morphware development system as an implementation method for VSIPL++. This approach is depicted in Figure 2, below. This approach was viewed favorably primarily due to speed and ease of implementation. It is reasonable to believe that VSIPL++ implementations delivering adequate performance for VSIPL++ applications can be created quickly using this approach, because the stream input languages are well suited to the functionality specified by VSIPL++. The main detractions from this approach are that the encapsulation of Morphware created by this approach is likely to impose a performance penalty, and remove the dynamic flexibility improvements achieved by Morphware. In addition, the small size of the Morphware community limits the robustness of some of the elements of the toolchain. Reliance on this approach may negatively impact the achieved results of VSIPL++ using this approach.

The second favored approach is to augment the VSIPL++ specification to include elements of a stream abstraction, and to then integrate VSIPL++ with one or more Morphware source languages. This approach is depicted in Figure 3, below. The primary disadvantage to this approach is the level of engineering effort that the group expected to be required in order to implement. This approach requires significant augmentation of one more compilers in order to implement. The primary advantage to this approach is that the relatively tight integration of VSIPL and Morphware high level compilers should reduce the performance impact of unnecessary abstraction barriers, and should allow the use of dynamic reconfiguration of applications based on runtime conditions.

The HPEC-SI working groups seemed to reach consensus that the appropriate approach for integrating tiled processor support into VSIPL++ would be to implement the layered approach first, and the integrated approach later. This would allow rapid demonstration of the concepts and an early opportunity for users, application developers, hardware providers, and development tool compilers to work with the API, while not preventing the more flexible and higher performing approach in the future.

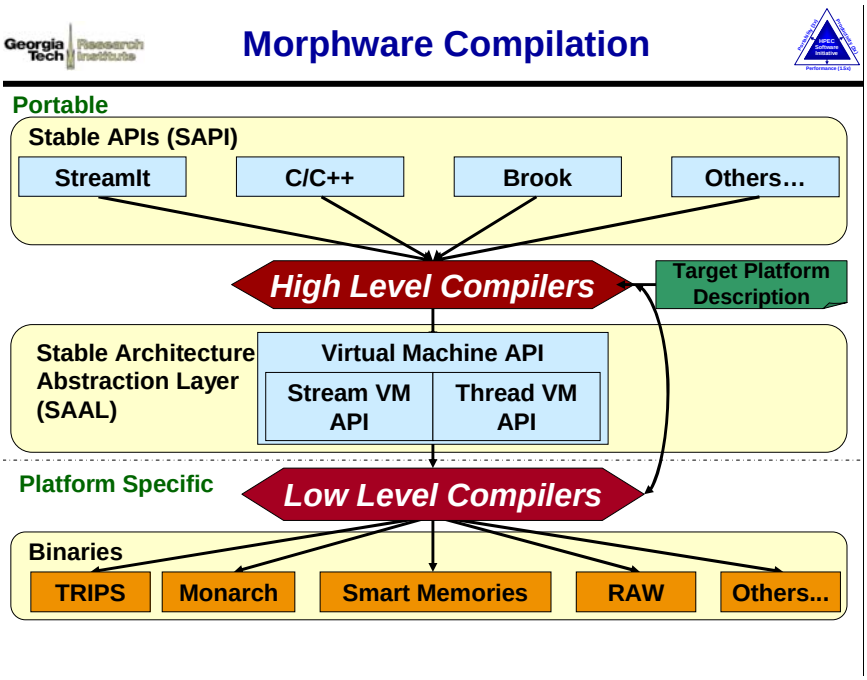

Figure 1 - Baseline Morphware Development

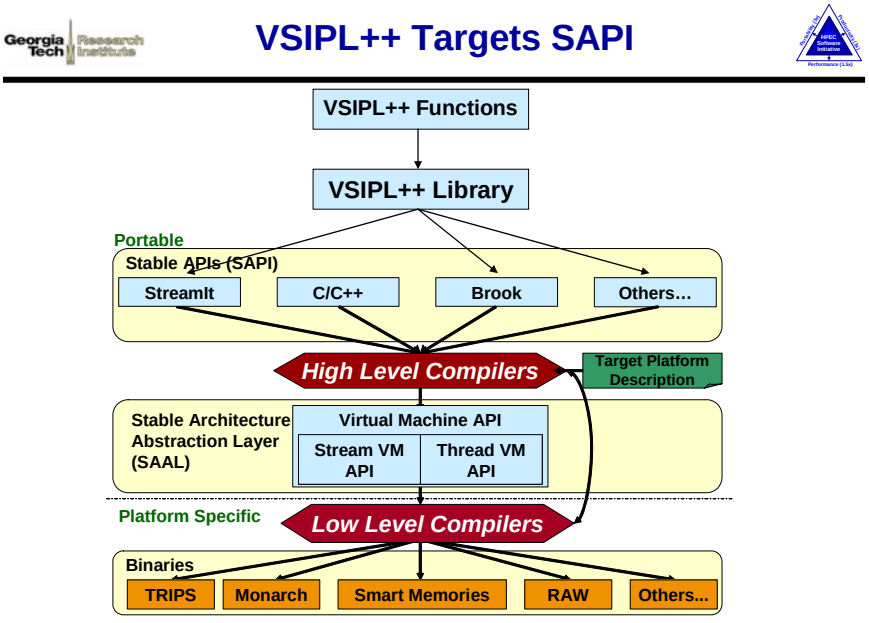

Figure 2 - VSIPL++ Encapsulates Morphware

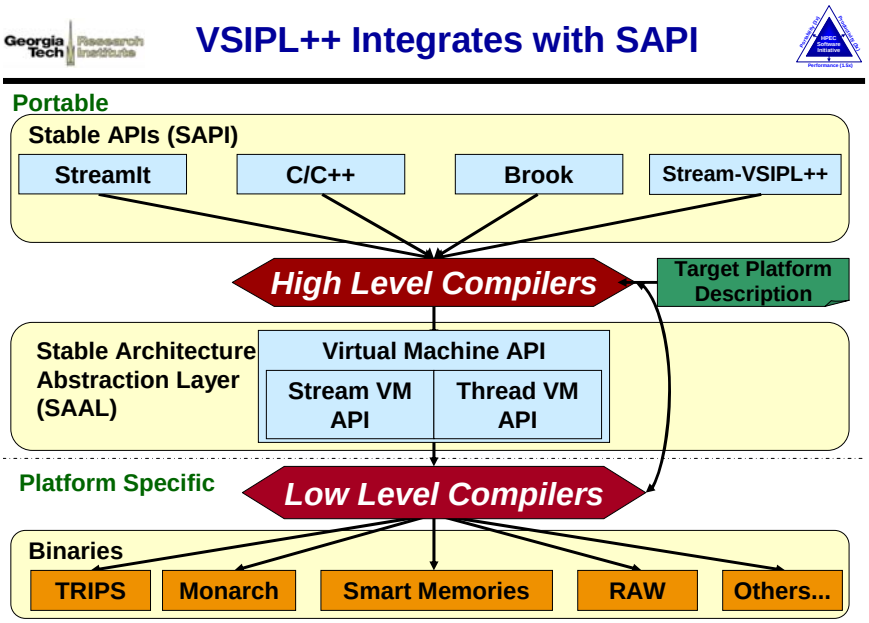

Figure 3 - VSIPL++ Integrated with Morphware Source Languages

4. Conclusions

During the period of performance of the subject contract, the Georgia Tech Research Institute contributed to the High Performance Embedded Computing Software Initiative program as a technical participant, and as a member of the technical advisory board. GTRI developed functional prototypes and software strategies, assisted in the dissemination of program results via conference presentation and internet tools, continued to provide a parallel software testbed for program participants, and continued development of a user's guide for VSIPL++ application development. GTRI also participated in technical advisory planning for the HPEC-SI program.

5. References

1. Schwartz, D.A., Judd, R.R., Harrod, W.J., Manley, D.P., "VSIPL 1.2 API" available electronically at http://www.vsipl.org/VSIPL_1p2_1.pdf

2. HPEC-SI Website, http://www.hpec-si.org

3. The Georgia Tech Research Institute Parallel Software Test and Evaluation Center – https://pastec.gtri.gatech.edu

4. Bergmann, J., Emeny, S., Judd, R., Pancoast, R., Leimbach, D., Sacco, S., Sroka, B., "VSIPL++ User's Guide", available electronically at http://www.hpec-si.org/VSIPL++%20User_s%20Guide%20Draft%20v0.2.pdf

5. CodeSourcery, LLC, "VSIPL++ Specification – Parallel Specification", available electronically at http://www.hpec-si.org/spec-par-1.0-final.pdf

6. CodeSourcery, LLC, "VSIPL++ Specification 1.01", available electronically at http://www.hpec-si.org/spec-1.01-final.pdf

7. Campbell, D.P., Cottel, D.M., Judd, R.R., Richards, M.A., "Introduction to Morphware – Software Architecture for Polymorphous Computing Architectures", Georgia Institute of Technology Library, Call Number QA 76.758.I579X 2004