AFRL-IF-RS-TR-2003-290
Final Technical Report
December 2003

HETEROGENEOUS EMBEDDED REAL-TIME SYSTEMS ENVIRONMENT

Integrated Sensors, Incorporated

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J469

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

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

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-290 has been reviewed and is approved for publication.

APPROVED: /s/

MARTIN J. WALTER
Project Engineer

FOR THE DIRECTOR: /s/

JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	DECEMBER 2003	Final Aug 97 – Jun 03

4. TITLE AND SUBTITLE
HETEROGENEOUS EMBEDDED REAL-TIME SYSTEMS ENVIRONMENT

5. FUNDING NUMBERS
C - F30602-97-C-0259
PE - 62301E
PR - D002
TA - 01
WU - P6

6. AUTHOR(S)
Cosmo Castellano and James Graham

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Integrated Sensors, Incorporated
502 Court Street, Suite 310
Utica New York 13502

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTC
26 Electronics Parkway
Rome New York 13441-4514

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2003-290

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Martin J. Walter/IFTC/(315) 330-4102/ Martin.Walter@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
The primary objective of the H-RTExpress effort is to provide enabling technology and domain specific tools that will quickly and effectively move systems targeted for heterogeneous architectures containing adaptive and embedded computing nodes from concept to implementation. The approach to accomplishing this objective is to build an integrated graphical development environment that will provide the capability to allow direct mapping and execution of sequential MATLAB code into heterogeneous embedded parallel computing systems containing both embedded processing and adaptive computing nodes comprised of field programmable gate array (FPGA) technology. The benefit of such an approach is that it provides an easy to use software development environment that significantly reduces the software development cost associated with building applications for these heterogeneous hardware systems. H-RTExpress allows the user to develop their application in a high level language (MATLAB) and frees the user from having to worry about data distribution and communication, since the tool automatically handles these details for the user. The H-RTExpress development environment leverages existing technology developed under previous DARPA/ITO funding as well as ISI commercial product technology RTExpress and Annapolis Microsystems COREFIRE.

14. SUBJECT TERMS
H-RTExpress, Domain Specific Tools, MATLAB, Heterogeneous Embedded Parallel Computing Systems, Enabling Technology

15. NUMBER OF PAGES
74

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Program Summary and Results

1.1 Problem Description

Heterogeneous computing architectures provide the ability to match unique hardware capabilities with specific system application requirements. For example a Field Programmable Gate Array (FPGA) processor can be used as a Fast Fourier Transform (FFT) accelerator while a general purpose processor such as a PowerPC can be used to implement a complex decision tree. The overall system performance is optimized in terms of throughput and cost by utilizing the strengths of a variety of compute element types. Typically, heterogeneous computing architectures are required to meet the demanding needs of sensor applications such as Space-Time Adaptive Processing (STAP), Automatic Target Recognition (ATR), and Synthetic Aperture Radar (SAR). A primary drawback to utilizing a mixed hardware environment is the level of effort required to develop optimum requirements/hardware mappings and generation of the overall system control software. Often the time required to implement the first iteration of the requirements/hardware mapping exceeds the budgeted project timeline and the application designer has little or no feedback on the quality of decisions made. The system control software includes not only the message/data interface among the system components but also start up and initialization of each of the individual hardware assets. The application designer is required to research the specifications of each of the hardware types utilized and develop unique I/O interfaces for each.

During this research and development investigation, which spanned six years, H-RTExpress added inclusion of Mercury Raceway PowerPCs to the earlier RTExpress work and operation on i860 processors with Hughes message passing interface (MPI) software. Support was added for the CSPI two-level multicomputer (PowerPC) with Myrinet and the network of Sun Workstations on Ethernet running Solaris. As the i860 architecture was becoming obsolete, a switch was made to support SHARC boards, loaned from AFRL, and a demonstration of that hardware mix was accomplished. The MPI/Pro implementation of MPI from MPI Software Technology, Inc. was utilized, and an effort to produce a real-time version of MPI, or MPI/RT, was started by Mississippi State University and later terminated.

H-RTExpress began with support for the MathWorks MATLAB software, which was at version 4.2, and followed the MATLAB software through two updates to versions 5.2 and 5.3.

The user interface was improved from the text based Target Balancing Tool (tbt) in RTExpress, to the graphic user interfaces (GUI) in the tools, mapit, splitm, editm, and bedit. Additional graphic performance monitoring tools were also added to the system.

The GUI tools in the H-RTExpress environment provide the ability to map MATLAB code segments onto heterogeneous processing nodes. A user is able to define and view a graphical representation of their target hardware architecture and software application, and then using point and click techniques, graphically map the software to hardware elements. The graphical representation of the hardware architecture is a coarse level representation. It provides multiple levels to view the hardware architecture. It includes processor nodes, switches, buses, I/O, and host information. A user can click on any of the architecture components to get more detailed information about the component.

Techniques and methods have been developed for launching the MATLAB application components on the heterogeneous hardware architecture. Part of this launching process is generating the executable code for the target architecture. For the embedded processors, parallel C code is generated and then compiled. For the adaptive computing nodes, preprogrammed library cores are utilized, and a graphic data-flow tool to create core library elements has been integrated into the environment.

The final stages of this program concentrated on field programmable gate arrays (FPGA). Support for the System Level Applications of Adaptive Computing (SLAAC) FPGA board from USC/ISI was added and demonstrated along with demonstrated support for the Annapolis WILDSTAR FPGA VME board.

The initial concept for RTExpress™ was funded under an Air Force Research Laboratory contract. The development of the RTExpress™ environment was funded under DARPA/ITO BAA 95-19. H-RTExpress was used to support other DARPA programs, such as the Preprocessor For UHF/VHF SAR program, contract DAAH01-99-C-R026, Power Aware Computing and Communication program, DARPA/ITO BAA 99-37, contract F30602-00-C-0150, and the Symbiotic Communications, SYCO, subcontract #370020SC.

The remainder of this report concentrates on the present state of the H-RTExpress software as demonstrated at ISI and installed at AFRL, Rome Research Site. Acknowledgement is given to past Principle Investigators, Rich Besler, Milissa Benincasa, and Diane Brassaw, and to past and present team contributors, Ron Adair, James Graham, John Ivory, Gary Kapps, and Adriana Kane.

Under this contract ISI let and administered subcontracts with Lockheed Martin Corporation (LMCO) and Northrop Grumman. LMCO developed WILDSTAR™ VHDL FFT Cores and Northrop Grumman Electronic Systems (NGES) developed an enhanced Micro-Accelerator FPGA Daughter Card. These developments are described separately in Appendix I and Appendix II, respectively.

1.2 Program Objectives

The objective of this effort is to provide enabling technology and domain specific tools to quickly and effectively move systems targeted for heterogeneous architectures from concept to implementation. An important part of such a tool suite is optimizing the selection of the "best" hardware to meet specific algorithm requirements. The tool suite must also address rapid changes in hardware capabilities, which tend to quickly obsolete systems that do not include portable software, and thus lead to large software reinvestment.

Objectives for the integrated software development environment include the following features:

- An integrated graphical user interface (GUI) environment for end-to-end application development on heterogeneous processor architectures in the system engineering domain. The GUI environment must provide both hardware and software graphical visualization.

- Techniques and notation for specifying variable data precision in conjunction with MATLAB® algorithm description.

- Methods for mapping MATLAB® code segments onto heterogeneous processing compute elements consisting of both embedded and adaptive computing hardware.

- Methods for launching applications across heterogeneous processor architectures.

1.3 Processing Overview

This effort produced an integrated development tool suite and run-time environment that provides the capability to quickly and efficiently move from system concept to deployable system utilizing various hardware architectures including heterogeneous architectures. The development environment supports a variety of hardware types and also provides the capability to be expanded to support new products, as they become available. The run-time environment provides the ability to launch the applications across a heterogeneous set of processing elements and provide the required system initialization and synchronization. The run-time environment also supports inter processor communication which allows efficient data distribution across components of the overall hardware architecture.

The tools developed to satisfy the development and runtime environment requirements are collectively call H-RTExpress™ since they are based on RTExpress™, which is a commercial product produced by Integrated Sensors Inc. RTExpress™ currently meets many of the desired program objectives for homogeneous architectures and it was used as a springboard to meet the needs of heterogeneous systems.

1.3.1 Development Environment

The H-RTExpress™ development environment is used to map system requirements to individual compute elements and then generate the required executables and run scripts. The user must supply the system level requirements in the form of MATLAB® scripts. Using the development environment the user then performs the following three step to produce a executable software system:

1) Subdivide the processing requirements into processing groups. There are several possible reasons for doing this. Processing groups may be defined so that multiple groups can operate in parallel, or a processing group may be defined to contain a section of the requirements which is applicable to a unique type of compute element, or a processing group may be defined to encapsulate a system interface. The H-RTExpress™ environment supports either no subdivision or any combination of the possibilities listed. The subdivision of requirements at this point is independent of the actual hardware the system will be implemented with.

2) Build a description of the target system hardware architecture. This step requires the user to identify compute element types and quantities. The H-RTExpress™ environment supports the concept of a hardware resource library so that a system administrator can build the actual low level description of node types and the general user can then select from a predefined list of resources. The general user also has the ability to create and/or modify a compute element descriptor as required. This step is independent of the specification of processing requirements described in step 1. In fact, this step can be used to create several hardware configurations which can be used to implement the same system requirements.

3) Map the processing groups specified in Step 1 to the hardware compute elements described in Step 2. This step allows the user to specify the number and type of compute elements assigned to each processing group. Within a processing group the compute element type must be homogeneous. This step also allows the user to specify the type(s) of parallelization to be used. The supported forms of parallelization are data parallelization, task parallelization, and round robin. A processing group can only support one type of parallelization at a time but the collective system can support any combination of parallelization methods.

Once these steps have been completed the development environment is then used to generate the correct load images for each of the compute elements and the required scripts to launch the application.

There are two H-RTExpress™ features which need to be introduced here. These are FPGA processor and variable data precision support.

1.3.1.1 FPGA Support

For this program, a heterogeneous architecture consisting of general-purpose processors and FPGA compute elements was selected. Within this architecture definition four different FPGA compute elements were considered. Within the H-RTExpress™ environment an FPGA compute element actually consists of a pair of processors, a general purpose processor which provides the MPI interface to the “compute element” and an FPGA accelerator. This matches the model used by each of the FPGA vendors, Annapolis Micro Systems (WILDSTAR™ and FIREBIRD™), University of Southern California/Information Sciences Institute (USC/ISI) East (SLAAC-2), and Northrop Grumman (Mercury FPGA Accelerator). The H-RTExpress™ tool set does not provide the capability to convert MATLAB® script into images that can be loaded into the FPGA devices. While there are several programs in progress which are looking into MATLAB® to VHSIC (Very High Speed Integrated Circuits) Hardware Design Language (VHDL) or C to VHDL translations, there are no mature tools available for use in the H-RTExpress™ development environment. The H-RTExpress™ tool set does provide a growth path to easily couple this type of tool into the code generation section when they become available. The development environment instead supports a library of predefined FPGA cores defined for each FPGA type. The user is able to develop and add cores to the library. The library of cores represents a predefined list of functions, which can be executed on the FPGA node.

This list also defines the required inputs and output formats for the supplied function. Using the H-RTExpress™ environment the user will be able to implement functions (i.e. FFT, FIR filter, etc.) on the FPGA once a core for those functions is available. For the WILDSTAR™ and FIREBIRD™ FPGAs the COREFIRE™ tool from AMS was used to generate a number of cores. Standard VHDL tools were used to generate cores for the SLAAC-2 and Northrop Grumman accelerator cards.

1.3.1.2 Variable Data Precision Support

One of the key advantages of using FPGA accelerators is the ability to perform reduced precision operations where possible to reap the resulting performance benefits. General-purpose processors provide some flexibility in precision selection but the underlying hardware often cannot take full advantage of a fixed-point data size or support mixed precision modes. Support in H-RTExpress™ for variable data precision was limited to the selection of single or double precision floating point on a general-purpose processor. Data precision for the FPGA devices, on the other hand, is limited by the core design for the FPGA devices. An FPGA library of cores may be used to provide a selection of data precision or support mixed data precision. For the general purpose processor, the user is able to specify the floating-point selection on a line-by-line basis in the MATLAB® script. For the FPGA compute element the precision is defined within the core specification where a core or set of cores represents a function callable from a MATLAB® script.

1.3.2 Runtime Environment

The runtime environment is used to load each compute element in the hardware architecture and control startup, system initialization, and termination of the application processing. Application startup consists of bringing each compute element to a known state, loading any required runtime kernels, and loading each compute element with its required executable. System initialization requires synchronization of all of the compute elements, distribution of required interfacing and/or application initial data, and initiation of the application. Termination requires synchronization of the compute elements, logging of system information as required, release of acquired system assets, and termination of the application on each compute element. The H-RTExpress™ runtime environment automatically provides each of these functions based on the type of hardware compute element selected. For FPGA compute elements this functionality is encapsulated in a library generated for each board. The H-RTExpress™ development tool set automatically links in the appropriate library as directed by a user's selection of an FPGA hardware type.

1.4 Summary of Program Accomplishments

H-RTExpress™ incorporated the strengths of RTExpress™, which was developed under a rapid prototyping DARPA initiative. These strengths allow the system designer to quickly and efficiently move from a MATLAB® based development environment to an embedded platform hardware implementation. H-RTExpress™ also provides a complete run time environment that allows the user to load and execute the software design on the heterogeneous environment. In addition H-RTExpress™ offers a variety of performance monitoring tools so the designer can evaluate and fine tune process and hardware mapping decisions.

H-RTExpress™ significantly improved the capability of the system designer GUI interface (mapit). Hardware systems can now be defined in terms of vendor supplied units (VME boards, PC boxes) instead of individual compute elements. This reduces the end user burden when required to build hardware platform description since vendor specifications can be used to build a library of hardware options. The end-to-end application development can be visualized and controlled from the GUI interface.

H-RTExpress™ provides a very modular approach for the incorporation of unique hardware processors such as FPGA boards. The FPGA interface is layered so that addition and deletion of supported platforms is performed when the application code is linked. This means that the same version of the H-RTExpress™ can support a variety of hardware configurations. For H-RTExpress™ to support a particular FPGA processor a software interface layer must be written which interfaces the H-RTExpress™ calls to the correct sequence of vendor API calls as shown in

Figure 1. Using a layered interface approach H-RTExpress™ is able to easily adapt to the individual interface requirements of hardware incorporated into platform and provide a consistent interface to the user program.

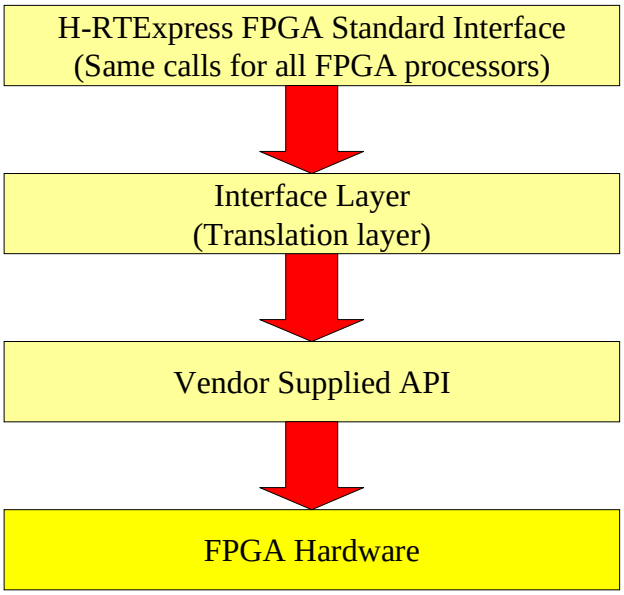


Figure 1 Layered FPGA Interface Design

H-RTExpress™ also demonstrated techniques and methods for specifying data precision from the MATLAB® specification level. H-RTExpress™ demonstrated the capability to operate in a mixed data precision mode both in terms of floating point and fixed point computations. This initial work limited the variety of data precision that could be specified but a software structure was created that could be expanded to increase the depth of coverage.

2 Concept Evaluation and Technical Results

2.1 Overview

This section provides detailed descriptions of the unique features of H-RTExpress™ and the results of demonstrations used to evaluate the use and performance of those features.

2.2 H-RTExpress™ Development Environment

The H-RTExpress™ software development environment is illustrated in Figure 2.

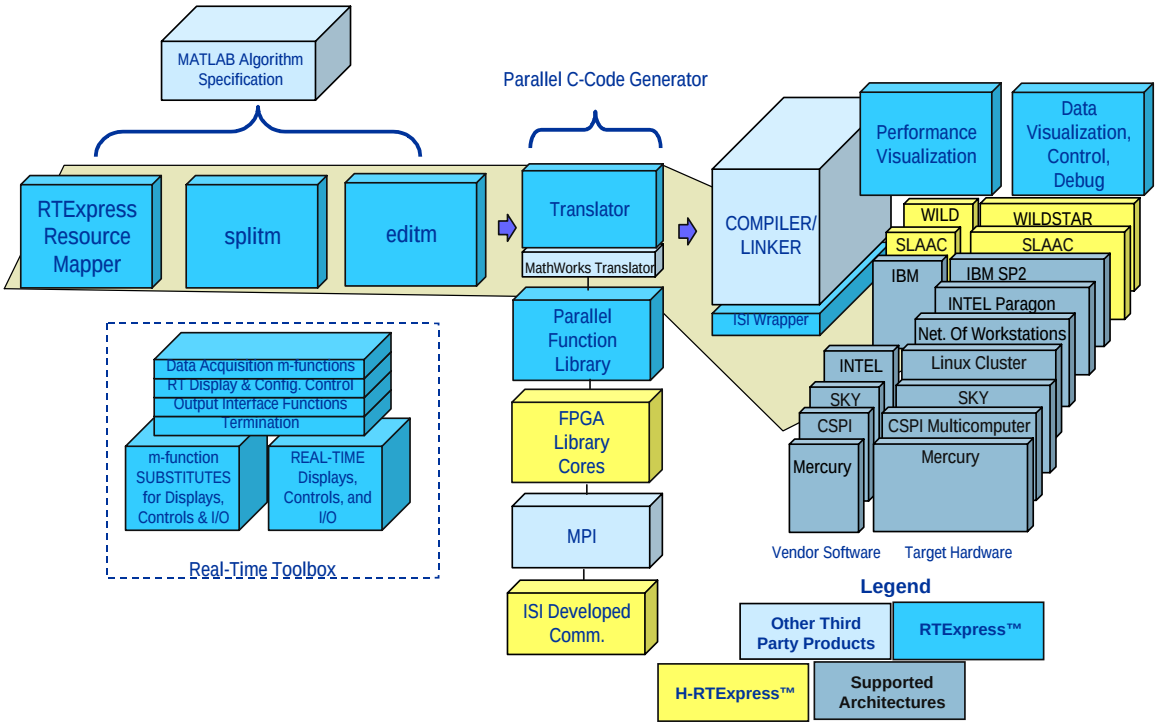


Figure 2 H-RTExpress™ Environment

All of the existing features of RTExpress™ are maintained in H-RTExpress™, and support for both the Annapolis Micro Systems WILDSTAR™ and USC/ISI SLAAC-2 Adaptive Computing Systems boards have been added. Support for the Annapolis Microsystems FIREBIRD™ PCI board (BAA-99-37) and for a Northrop Grumman Mercury RACE++ FPGA Daughter Card (BAA 97-06 ECP) has also been added.

The H-RTExpress™ Environment utilizes four main user tools: splitm (MATLAB® editor and splitting); editm (MATLAB® editor); Mapit (resource mapper); and bedit (hardware specification). A visualization of the tool windows is shown in Figure 3. In addition to other third party products, such as the MathWorks MATLAB® Translator, MSTI's MPI Pro, and a standard compiler and linker, H-RTExpress™ includes the ISI Real-Time Toolbox functions and parallel function library that brings MATLAB® functions to a parallel implementation. Where available, vendor supplied libraries are used, as is the case with the Mercury Computer Scientific Application Library (SAL).

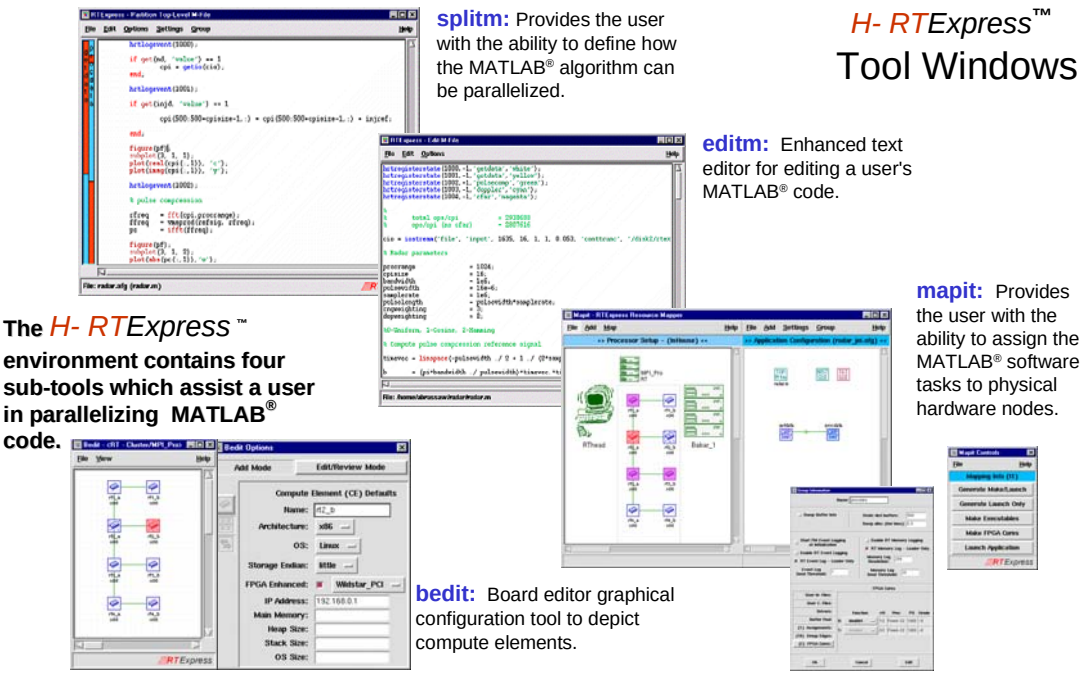
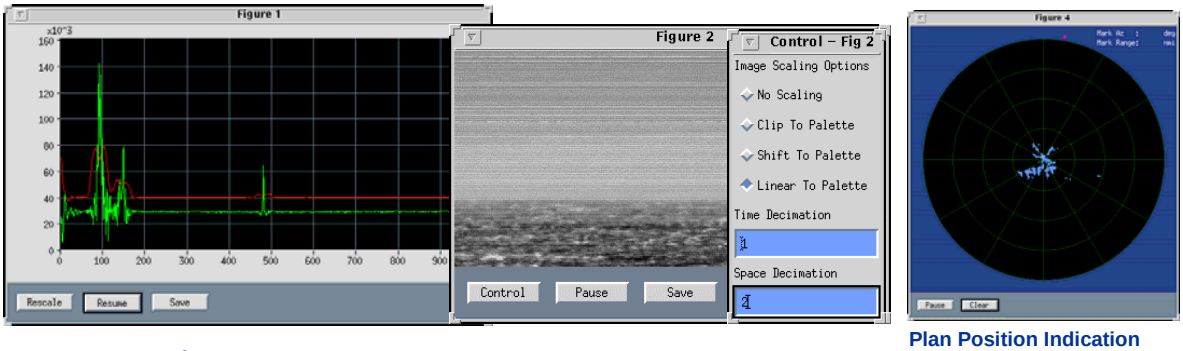


Figure 3 H-RTExpress™ Tool Windows

Real time data visualization and user defined buttons and input and output data boxes are available in the H-RTExpress™ Real-Time Toolbox graphic user interface. Some of the features are shown in Figure 4.



Emulates MATLAB® plot with enhancements for real-time operation
Data Pause/Save to disk for User Analysis
Variable scaling
Time Decimation to control I/O Loading

Emulates MATLAB® "image" with enhancements for real-time operation
Data Pause/Save to disk for User Analysis
Variable scaling
Time/Space Decimation to control I/O Loading

Plan Position Indication (PPI)

Figure 4 H-RTExpress™ Data Visualization

The H-RTExpress™ environment provides real-time performance monitoring, shown in Figure 5, as the application is executing on the target architecture, plus post-mortem instrumentation (for debugging and analyzing the parallel application after it has been executed). Graphic visualization, not shown in Figure 5, is also supplied for real-time program memory utilization, and real-time program event monitoring.

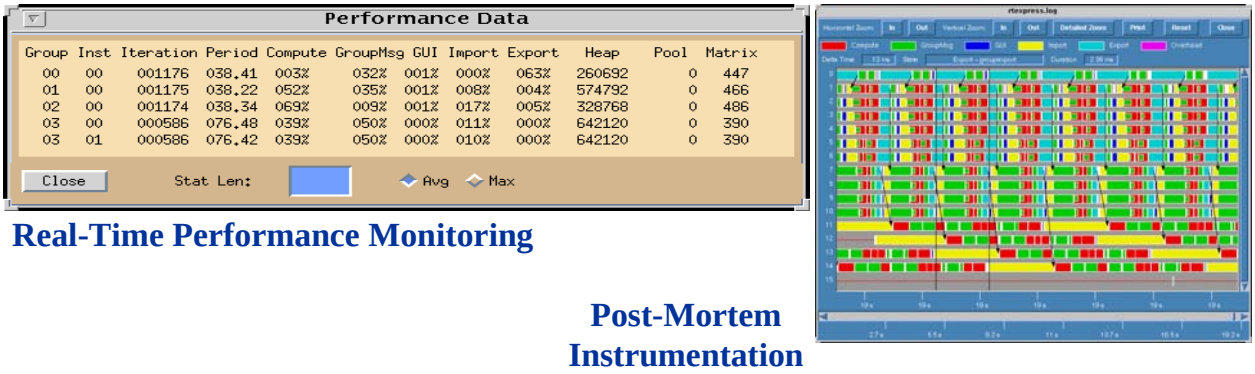


Figure 5 H-RTExpress™ Performance Monitoring

2.2.1 H-RTExpress™ FPGA Model

The FPGA is viewed as an adjunct processor to a general-purpose processor (GPP) by H-RTExpress™, as shown in Figure 6. The GPP associated with an FPGA communicates to other processors via the Message Passing Interface (MPI) standard, and is responsible for all data conversion and control to and from the FPGA. H-RTExpress™ allows the programmer to select a function from the FPGA library to be called from the MATLAB® language. An FPGA library element consists of the FPGA Wrapper (C program) and FPGA core, or set of cores. The FPGA cores are built and tested for specific devices on the appropriate board. The C wrapper executes on the GPP and uses the vendor supplied API to interface to the FPGA accelerator. This process is illustrated in Figure 7.

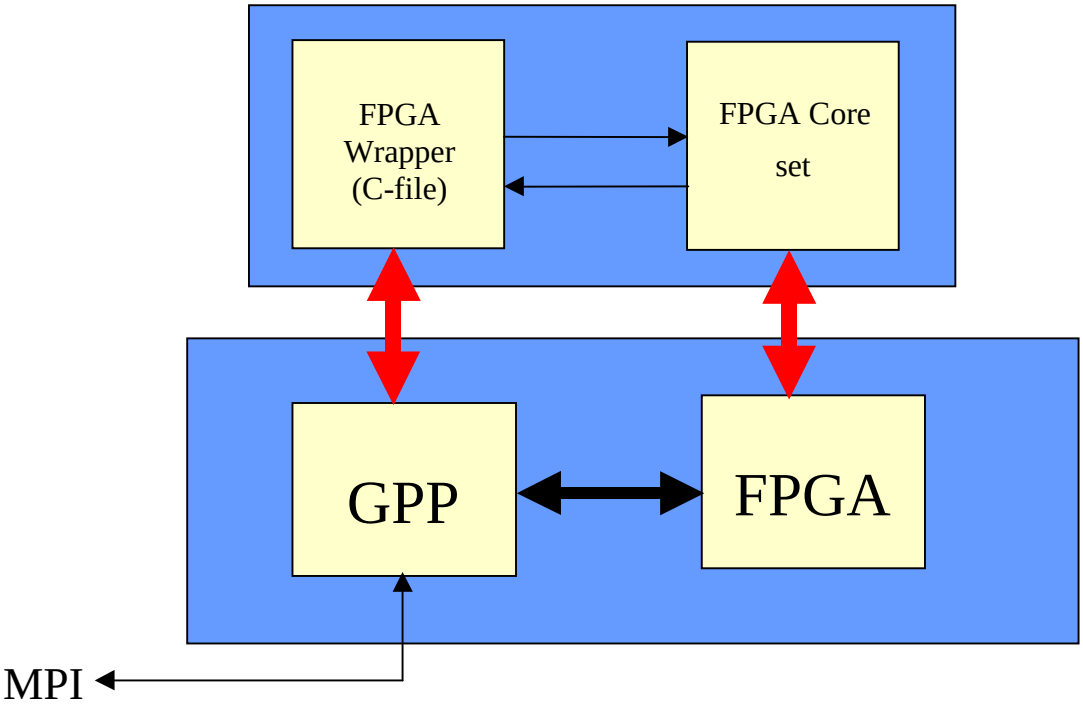


Figure 6 H-RTExpress™ FPGA Model

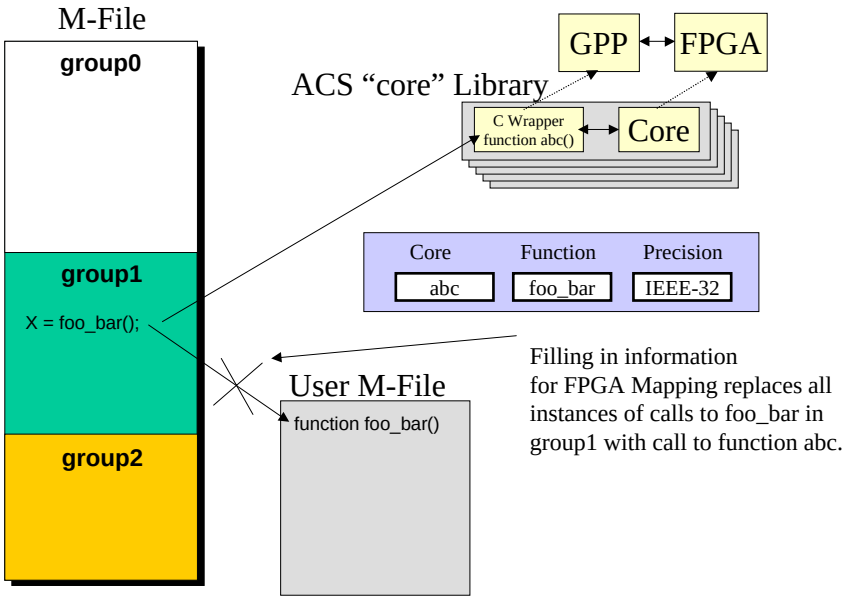


Figure 7 Splitting M-File and Mapping to FPGA

2.2.2 H-RTExpress™ Tools

The H-RTExpress™ user may start with a top-level MATLAB® m-file, and using "splitm" take the m-file and split it into logical processing groups. Splitm as shown in Figure 8, allows the programmer to define various groups, and graphically select lines from the m-file and associate them with the defined groups. The splitm tool creates separate m-files for each processing group. The "editm" tool, seen in Figure 9, is very much like splitm, however, it is only concerned with editing m-files, and like splitm, has color-coded presentation to highlight aspects of the MATLAB® language. Built-in help for MATLAB® and H-RTExpress™ Real-Time Toolbox functions is included.

An additional feature of H-RTExpress™ is that individual lines may be selected, using either tool, and through the menu pull-down, or a keyboard shortcut and tagged as an exception to the default precision. The project precision is selected from the settings window of the mapit GUI. This is the default precision for all lines in the MATLAB® script unless an exception is noted. H-RTExpress™ denotes precision exceptions for subsequent processing by the addition of a token (%_p%) at the end of the line(s) selected. The token denotes a line will be executed with the opposite (single vs. double) precision selected as the project default. Through this mechanism, as seen in Figure 10, H-RTExpress™ users may use the performance enhancement of single precision while still utilizing double precision for those portions of their programs that require the additional word size.

• The H-RTExpress™ splitm tool provides the capability to specify the functional groupings of a top level MATLB® m-file
• splitm currently supports the following parallel models
  - •Data parallel
  - •Task parallel
  - •Pipeline
  - •Round Robin
  - •Mixed Mode
• Once a m-file has been functionally decomposed using splitm, it will be graphically represented in the H-RTExpress ™ Resource Mapper tool

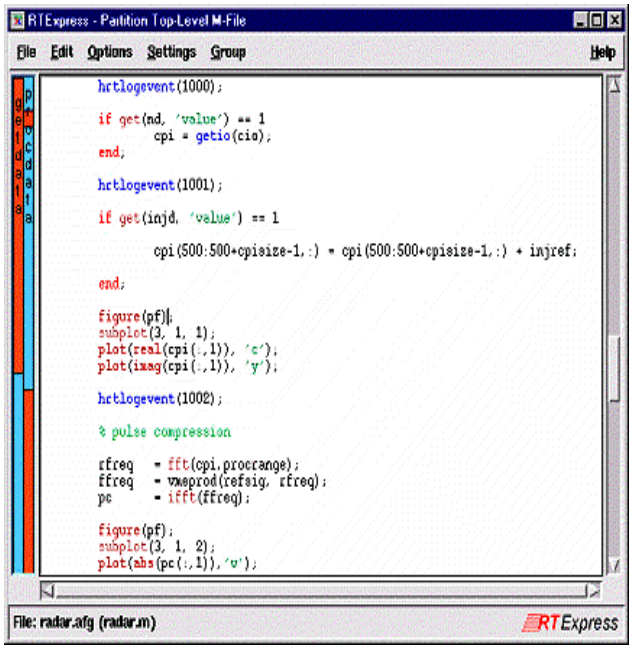


Figure 8 H-RTExpress™ Splitm

• The H-RTExpress™ editm tool provides an enhanced text editor for a user's MATLAB m-file

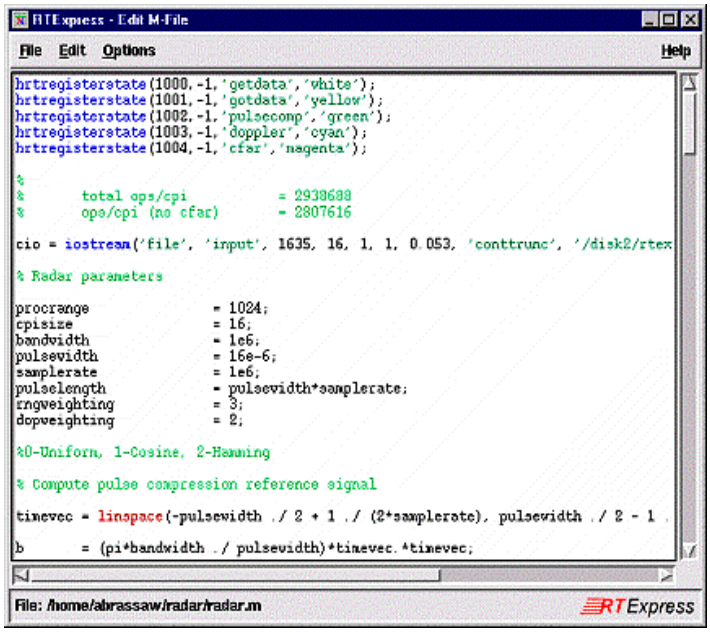


Figure 9 H-RTExpress™ Editm

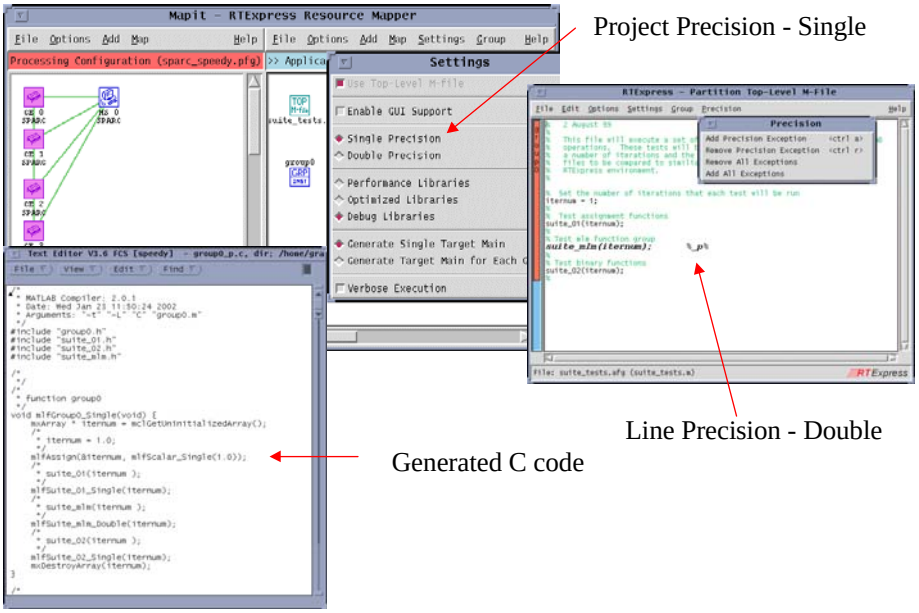


Figure 10 Variable Data Precision Control

Using the H-RTExpress™ Mapit Resource Mapping tool shown in Figure 11, the programmer makes a relationship between processing groups previously defined using splitm and processing resources, such as a general-purpose processor or an FPGA board. A mapping configuration file holds information on the application configuration and processor configuration files and the mapping relations between them. The application configuration also contains information about the top-level m-file, any m-files called, or any C-program files called. If the processing group is assigned to an FPGA enhanced compute element, the application configuration also contains information about the FPGA core to load and initialize. The tool produces the standard "make" files and a startup script to launch the parallel program.

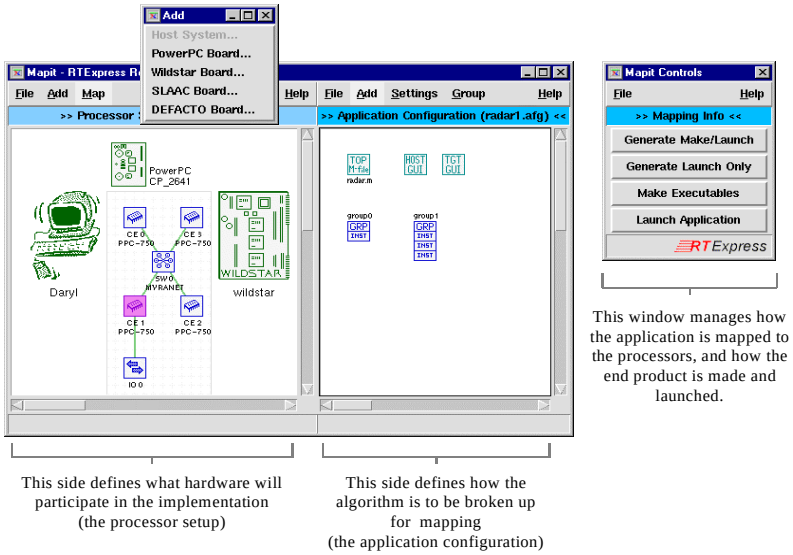


Figure 11 H-RTExpress™ Resource Mapping

If the process group is assigned to an FPGA enhanced compute element, double clicking the group icon (see Figure 12) brings up the FPGA library pop-up menu for specification of an existing FPGA function. The FPGA library is extensible, and for this project, several cores were added.

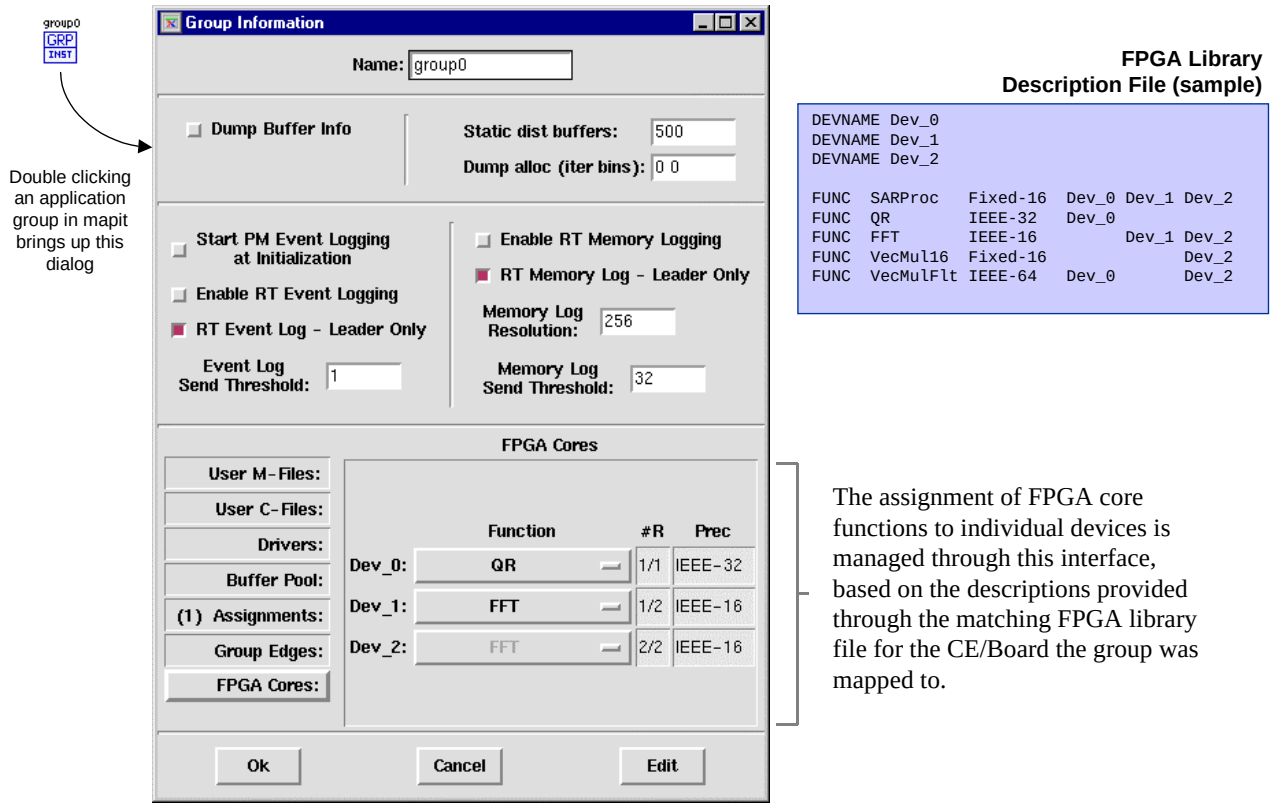


Figure 12 H-RTExpress™ Library Selection

The processor configurations (left window of mapit GUI) are built using the H_RTExpress™ board editor (bedit). bedit allows the user to build hardware architectures from graphical components or load predefined hardware architectures. This tool can be used to create a library of hardware architecture boards. The diagram in Figure 13 shows a four-node cluster, which has a single element that contains a FIREBIRD™ FPGA accelerator. The popup shown in Figure 13 is obtained by double clicking the upper right compute element.

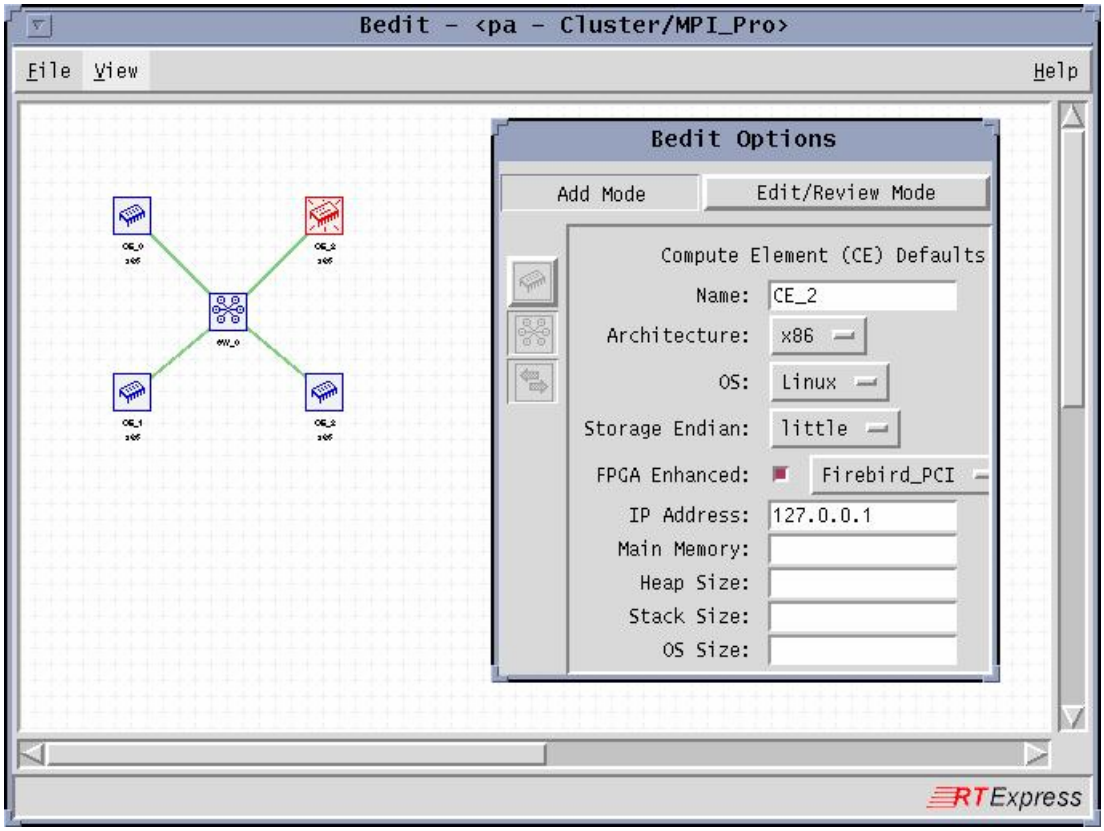


Figure 13 H-RTExpress™ Board Editor

2.2.3 H-RTExpress™ Extendibility

There are several features in the overall H-RTExpress™ design which allow the user to extend the environment to cover a wider range of applications. The user is able to include additional MATLAB® scripts as well as files written in C. In the same manner the FPGA core library is user extensible. Each library element consists of a C-wrapper file and one or more FPGA cores that are associated with it. The cores are built and tested for a specific function and FPGA hardware. One of the tools available to aid the user in development of a library of FPGA cores is COREFIRE™ built by Annapolis MicroSystems which is described in the next section. This tool was used to develop the FIREBIRD™ FPGA cores for the Coherent Sidelobe Cancellor demonstration described in Section 2.4.4

2.2.3.1 AMS COREFIRE™

The COREFIRE™ Design Suite provides a large collection of cores and an application builder that enable users to describe and build FPGA designs without using hardware design languages such as VHDL or Verilog. The COREFIRE™ Design Suite also includes an application debugger that enables users to load and interact with their designs in the target environment without writing application programs. As the central component of the COREFIRE™ Design Suite, the COREFIRE™ Application Builder features a unique drag-and-drop capability for combining core components into ready-to-run programs. See
Figure 14 for an overview of the COREFIRE environment. An extensive array of these core components are available in the COREFIRE™ Module Libraries, along with Annapolis Micro Systems, Inc. board-specific support packages.

In summary, some of the benefits of COREFIRE™ include:

a) Works from High Level, Data Flow Concept of the Design
b) Combines GUI Design Entry and Debug Tools with Tested, Optimized COREFIRE™ IP Cores
c) Drag and Drop High and Low Level Modules
d) COREFIRE™ Modules Incorporate Years of Application Development Experience - Highly Optimized and Tested
e) COREFIRE™ Tools and Modules are Intelligent
f) Modules Automatically Handle Synchronization
g) Manage Clocks and Other Low Level Hardware Signals
h) Guarantee Correct Control by Design
i) Modules "Know How" to Interact With Each Other
j) Board Support Packages Incorporate Hardware Details of the Boards - Invisible to Users
k) Supports Conversion Between Data Types - Bit, Signed and Unsigned Integers, Single Precision Floating Point, and Integer and Floating Point Complex Data Types
l) Provides Java Files & Host Code to Run & Debug the FPGAs
m) Works with all WILD™ Virtex™, Virtex™ E and Virtex™ II FPGA processor and I/O boards

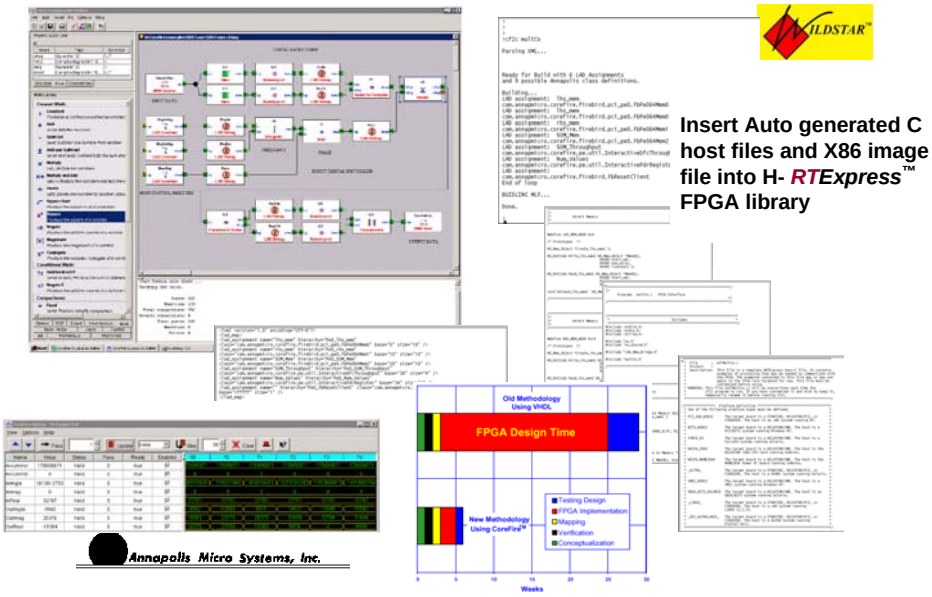


Figure 14 AMS COREFIRE™ Development Environment

An ISI generated utility reads the XML map file from COREFIRE™ and creates a series of files to facilitate insertion of the newly generated FPGA core into the H-RTExpress™ core library. The basic operation of COREFIRE™ to C (cf2c) is to read the XML file which contains the host interface definitions, picking up the user's name and address for each item and the COREFIRE™ name for the item. Based on the COREFIRE™ name, the appropriate template is selected. That template is appended to the *.c file (lower-case name) and its associated *.h file (lower-case-name). The template contains various key words that are replaced with the user's interface item name and local address. The mlf*.c file becomes the C-wrapper that is a callable function from the H-RTExpress™ MATLAB®, and it uses the interface functions that were created by this process. The user must edit the main portion of the mlf file to produce the desired behavior, however, all of the procedures to support each interface item have been created to expedite the process of creating the wrapper file. The new FPGA function is then entered into the FPGA Library Description File for use by H-RTExpress™.

2.3 Run-time Environment

To execute a user application on a heterogeneous target hardware a run time environment is required to facilitate launching the application, supporting communication across the environment, and allowing an orderly termination when the application ends. When an application is launched each compute element must be initialized, an executable loaded, and the start of program execution synchronized with all other compute elements. Similarly when an application terminates system resources allocated during program execution must be released and hardware devices should be placed in a known state. During execution all compute elements, including the FPGA devices, may be required to transmit and/or receive data and/or command information. In a heterogeneous environment each of these requirements may involve a unique set of procedures for each node type utilized. The H-RTExpress™ run-time environment provides mechanisms to support each of these requirements.

2.3.1 Startup

H-RTExpress™ uses the Message Passing Interface (MPI) communications library as the underlying mechanism to communicate between processing nodes. As described before an FPGA device is viewed as an adjunct to a general-purpose processor (GPP). A GPP node fully supports MPI. Within MPI the utility MPI_Init is utilized to synchronize all processing nodes used in the application at startup. An FPGA enhanced processing node (combination of GPP and FPGA device) is responsible for initializing its associated FPGA device prior to calling MPI_Init. When launch scripts are built for an application by the H-RTExpress™ Development Environment tools, a configuration file, node.cfg, is built which contains user selectable run time parameters for the H-RTExpress™ environment. Among these parameters is an indicator that a group consists of FPGA enhanced nodes and that the user has mapped an FPGA based function to the group. A second configuration file was also created for the group which contains information required to initialize the FPGA device. This information includes but is not limited to: number of cores, identity of devices to be loaded, names and paths to cores, minimum, maximum, and nominal clock frequency settings. This file is parsed by the run-time environment and the associated FPGA device is initialized using a board specific initialization module that is part of the H-RTExpress™ run-time library. Once the FPGA device is initialized the processing node synchronizes with the remaining processing nodes using MPI_Init. The startup procedure is summarized in Figure 15.

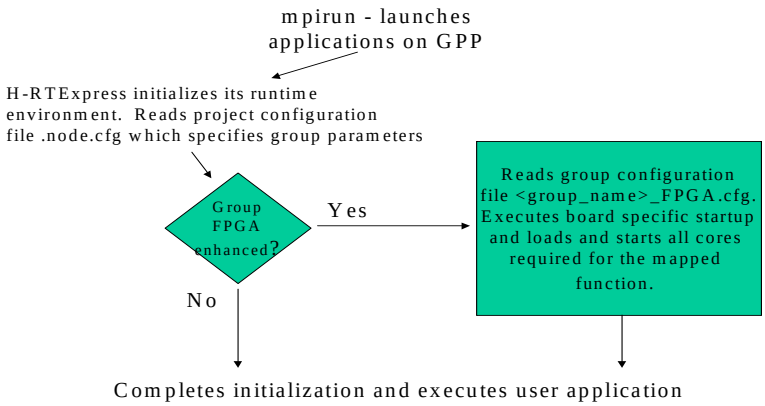


Figure 15 FPGA Group Start-Up

2.3.2 FPGA Wrappers

Figure 16 is a generalized form of the C wrapper required for each function performed in an FPGA device. A C wrapper that executes on the GPP is required to interface the specific FPGA device to the MPI based environment and provide the API required by the function design (control and/or data sequence, control parameters, data format). Depending on the FPGA device and/or function design all of the following steps may not be required.

a) Format input data:

Parameters and/or data are passed to the C wrapper within mxArray structures. These structures contain the data itself if the parameter is a scalar, otherwise a pointer to the data is passed within the structure. The data can be real or complex and either single or double precision floating point. The FPGA device and function design will determine the required data format translation.

b) Perform required control/setup:

As a function of the FPGA design any required setup and/or control must occur. The FPGA device design will also specify the required sequencing of control and/or data. This processing in addition to being design specific will also be device specific in its implementation.

c) Transfer data to FPGA:

The device design will specify the required input control and/or data format. The device will specify how the data is to be transferred, FIFO, DMA, registers, etc.

d) Synchronize data completion:

Device design will specify how and if any synchronization is required to determine the availability of results from the FPGA device. The FPGA device itself will determine how that notification occurs (polling, DMA transfer, interrupt, etc). The H-RTExpress™ run-time environment does not currently support a mechanism to post for data completion.

e) Transfer data from FPGA:
The device design will specify the required output data and its format. The device will specify how the data is transferred (FIFO, DMA, registers, etc).

f) Format return parameters:

All data and status is returned to the MATLAB® processing environment within mxArray structures. As on input the output data must match the expected precision of the MATLAB® processing.

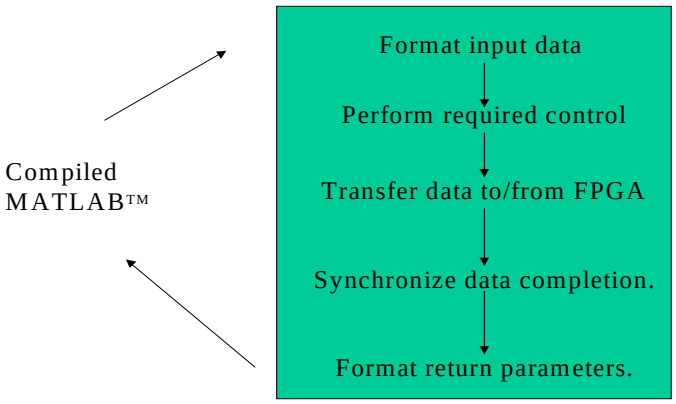


Figure 16 "C" Wrapper Functional Flow

2.3.3 Shutdown

As with Startup, the MPI communications library module MPI_Finalize is utilized to synchronize shutdown of all processing nodes used in the application. An FPGA enhanced processing node is responsible for terminating its associated FPGA device prior to calling MPI_Finalize. A processing node has already determined during initialization that its associated FPGA device is being utilized. A board specific termination module that is part of the H-RTExpress™ run-time library is called to perform the required device shutdown. Once the FPGA device is shutdown the processing node synchronized with the remaining processing nodes using MPI_Finalize. The system shutdown processing is summarized in
Figure **17**.

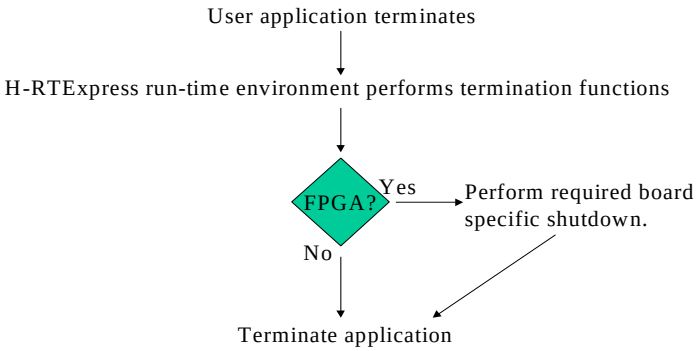


Figure 17 FPGA Group Shutdown

2.3.4 Supported Platforms

2.3.4.1 SLAAC-2

The SLAAC-2 VME daughter card shown in Figure 18 was designed and manufactured by the University of Southern California/Information Sciences Institute (USC/ISI) East. This card was developed under the Systems Level Applications of Adaptive Computing (SLAAC) project sponsored by DARPA TTO Adaptive Computing Systems.

The SLAAC-2 processor is actually two "independent" SLAAC-1 bit-file compatible accelerators on a single 6U PMC mezzanine card. This mezzanine plugs into a modified CSPI 2641 PowerPC baseboard, the M2641S. Each of the FPGA accelerators features one user-programmable Xilinx 4085 devices, two user-programmable Xilinx 40150 devices, and ten 256Kx18 100MHz ZBT synchronous SRAMs. A block diagram of the SLACC-1 node is shown in Figure 19

Figure 18 SLAAC-2 VME Daughtercard

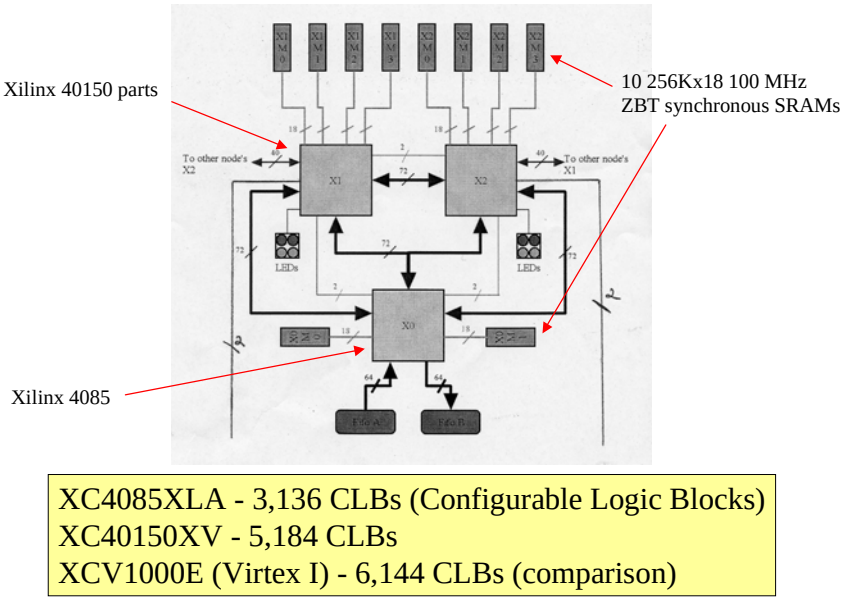


Figure 19 SLAAC -1 FPGA Accelerator

The 6U VME CSPI M2641S baseboard contains two 300 MHz Motorola PowerPC 603e processing elements each with 256 Mbytes of SDRAM. The M2641S uses Myrinet as its external interface.

Each PowerPC interfaces with its associated FPGA accelerator through a pair of FIFOs (one for input and one for output) as shown in Figure 20. For this project CSPI software version 1.4 which incorporates VxWorks version 5.3.2 and Tornado 1.0.1 was used with SLAAC-2 API version 1.2.

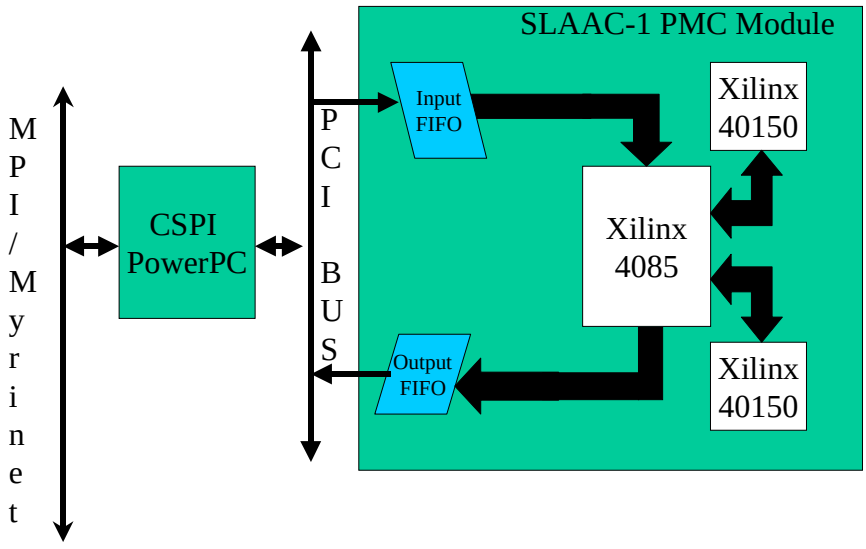


Figure 20 SLAAC-1 Element Interfaces

2.3.4.2 AMS WILDSTAR™ – VME

The WILDSTAR™ FPGA board, see Figure 21, is produced by Annapolis Micro Systems, Inc (AMS). It is a 6U VME card which contains up to three Virtex™ or Virtex™ E FPGA Processing Element (XCV400 to XCV2000E). The board used for this project contained three Virtex™ XCV1000E speed grade 4 parts. The board can also support 2 to 40 MBytes of Synchronous ZBT SRAM. The board used for this project contained 20 Mbytes of SRAM. The board can also accept up to two PMC daughtercards. Available daughtercards support Race™, Race++™, Front Panel Data Port (FPDP), WILDSTAR™ Data Port (WSDP™) and Myrinet interfaces. Additional daughtercards are available which support an i960 processor and a high speed A/D interface. A block diagram of the WILDSTAR FPGA processor is shown in Figure 22.

Figure 21 WILDSTAR™ FPGA Processor

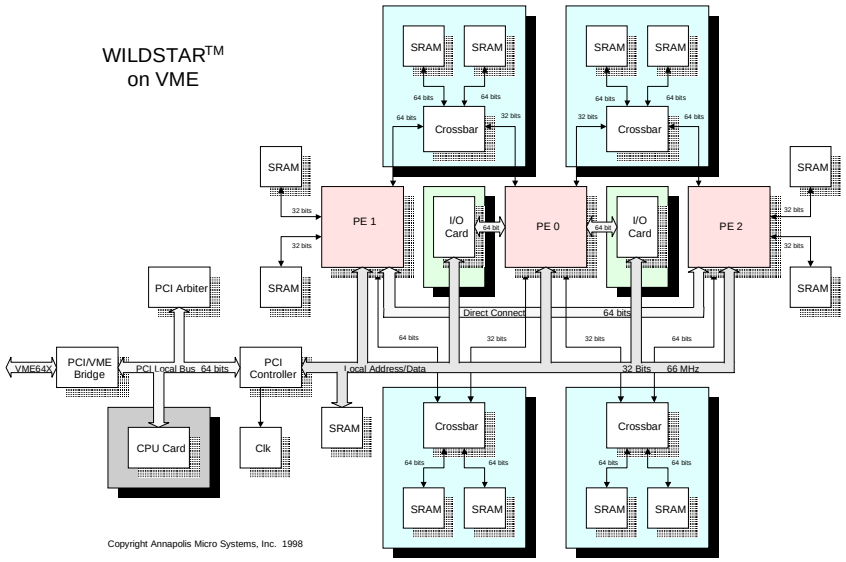


Figure 22 WIDSTAR™ Block Diagram

The AMS WILDSTAR™ API Version 3.0.0 and PCI Version 3.0.0 were used for this project.

2.3.4.3 AMS FIREBIRD™

The FIREBIRD™ PCI card, shown in Figure 23, is also produced by AMS. It is a PCI card which contains a single Virtex™ E FPGA Processing Element ranging from XCV1000E to XCV2000E . The board used for this project contains a Virtex™ XCV2000E speed grade 8 part. The board can also support 9 to 36 MBytes of Synchronous ZBT SRAM in 5 Memory Banks. The board used for this project contained 36 Mbytes of SRAM. The board is PCI Bus - Rev 2.2 Compliant and can support the following interface options:

5V Board - 32/64 Bit, 33 MHz, 5V or 3.3V Slot
3.3V Board - 32/64 Bit, 33/66 MHz, 3.3V Slot
Automatic 32/64 Bit PCI Bus Recognition

The board used for this project was the 3.3V board, which was configured for a PCI interface, using 64 bits and a 66 MHz data clock. The board can accept a single PMC daughter card and will support the same daughter cards available for the WILDSTAR™. The board used for this project had an FPDP-E daughter card installed which contained a single Virtex XCV2000E speed grade 6 part.

Figure 23 AMS FIREBIRD™ For PCI

The AMS WILDSTAR™ API Version 5.0.0, Driver Version 1.2.5, and PCI Version 2.8.0 were used for this project.

2.3.4.4 Northrop Grumman Micro Accelerator

The Northrop Grumman Micro Accelerator is a Type B Race++™ Mercury daughtercard. A detailed description of the accelerator is contained in

Appendix II - Description of Northrop Grumman Micro Accelerator.

2.3.5 Variable Data Precision

2.3.5.1 Definition

This project examined techniques and notation for specifying variable data precision in conjunction with MATLAB® algorithm description. The problem was first addressed by dividing the data precision definition into two categories, fixed point and floating point. Fixed point representation was then limited to FPGA devices. Floating point representation was limited to IEEE 32 bit (single precision floating point) and 64 bit (double precision floating point). RTExpress™ supports application execution entirely as either single or double precision floating point. This project expanded that capability to allow the user to specify the precision for an individual line of MATLAB®. Using this definition the precision of a function and/or variable can be controlled. As described before the precision select is accomplished via annotations, which can be added by the user in either the splitm or editm tools. A summary of the Data Precision Control model is shown in Figure 24.

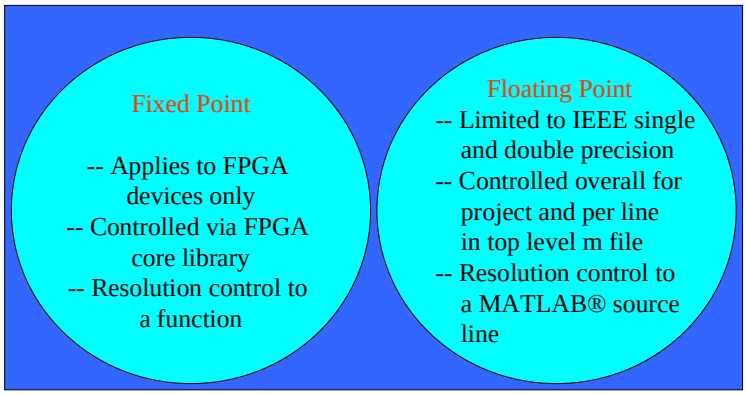


Figure 24 Data Precision Control Summary

2.3.5.2 Implementation

2.3.5.2.1 User M files

Once a user MATLAB® script (m file) has been annotated it is passed through a set of utilities to transition from MATLAB® to compilable C code. There are two utilities that process the precision annotations. These are postpass2 and parse. Postpass2 processes the MATLAB® script to append the desired precision selected to each invoked functions. There are some functions where the selected precision does not effect its processing so these are left unchanged. For those modules where the precision select does matter the precision is appended to the name to create a unique identifier. If function foo is to be preformed using single precision postpass2 will create an m script with the call to foo_single. If foo is to perform using double precision then postpass2 will create an m script with a call to foo_double. Since the precision is controlled to a MATLAB® line there could be calls to foo_single and foo_double within the same module.

If foo is a user MATLAB® file then it must be compiled in both single and double precision format, see Figure 25. The general format of the resultant C function is shown in Figure 26 and Figure 27. The function is compiled in two formats, single precision when PRECISION_SINGLE is defined and double precision when PRECISION_DOUBLE is defined. The top code section, Figure 26, defines prototypes for functions defined at the end of this module which will implement the actual requirements of the function. The second code section defines the interface routines for the function. These defined the module prototypes required for other functions to interface to this function. These interfacing routines also handle all the required input data format conversion as well as any memory allocation/deallocation required because of data formatting. Data is only reformatted on input since it cannot be determined what the required data format for functions that will use these outputs. When input data is reformatted the original inputs may be released depending whether they are temporary or permanent. The last code section, Figure 27, is the actual implementation of the function. All calls to other functions from this section must preserve the required data precision. The output objects are then linked with the completed project.

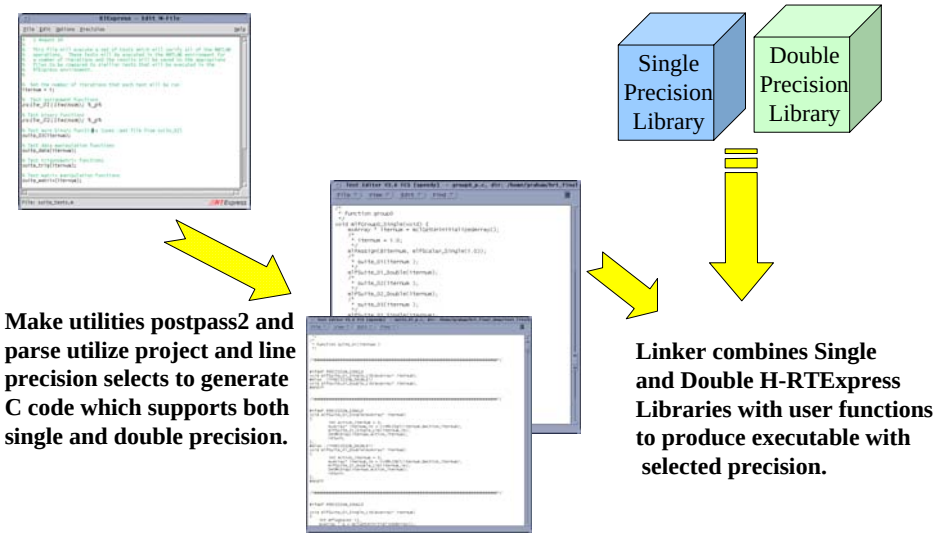


Figure 25 Floating Point Implementation

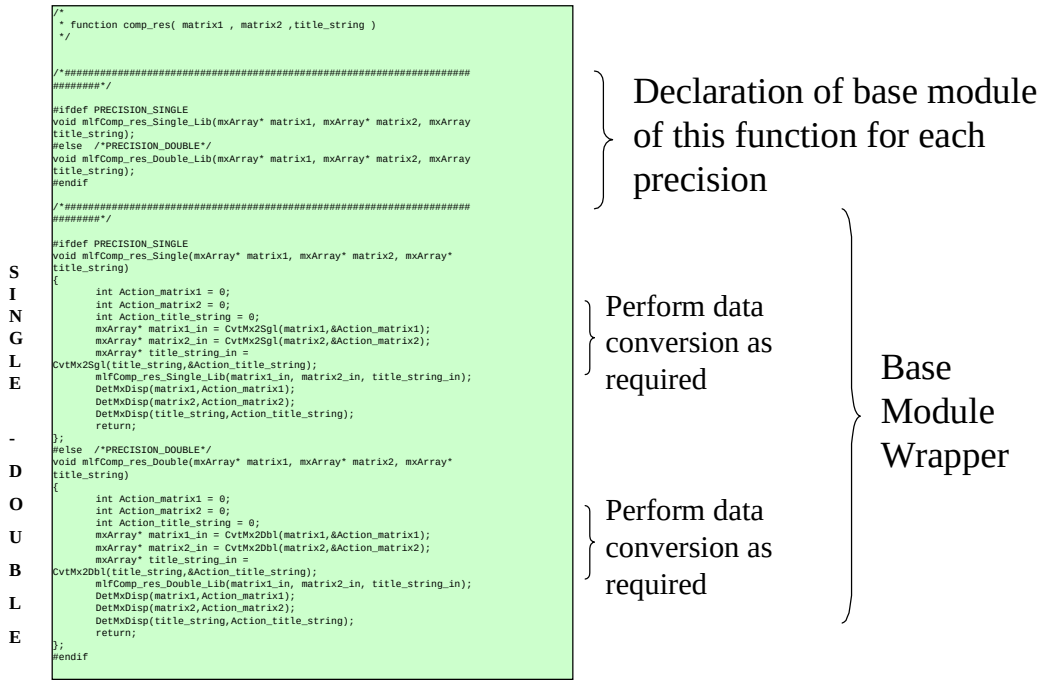


Figure 26 C Wrapper Format - Part 1

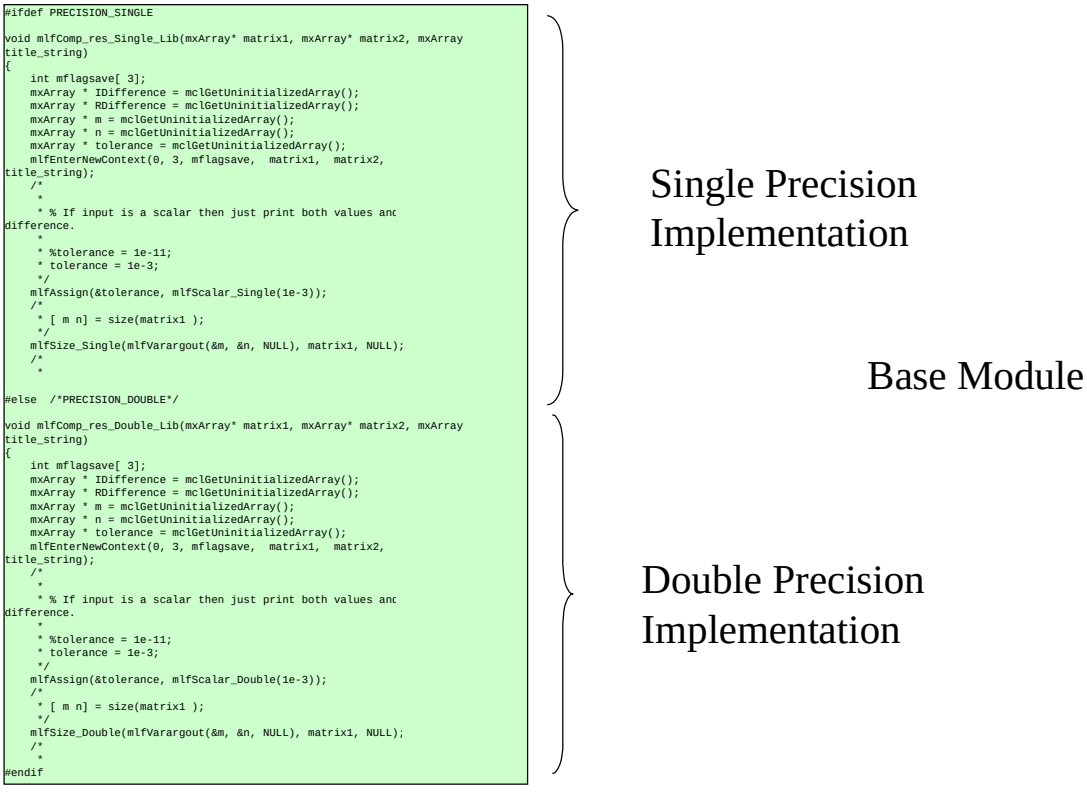


Figure 27 C Wrapper Format - Part 2

2.3.5.2.2 Libraries

The H-RTExpress™ libraries must be built in much the same manner as the user m files. In this case the C routine is modified directly. For the current H-RTExpress™ library, 271 out of the 401 functions were modified. For some functions there is no difference in the processing required for single and double precision. In this case the function is only compiled once and added to the run time library. Objects for each function must be created with the single and double precision defines set. Each of these outputs is added to the H-RTExpress™ library.

2.4 Demonstration of Technical Results

2.4.1 SLAAC-2 Interface

2.4.1.1 Purpose

This demonstration showcases the capabilities of H-RTExpress™ that provide the ability to map, load, and execute user applications to a heterogeneous environment consisting of PowerPC and SLAAC-2 FPGA enhanced compute elements.

2.4.1.2 Hardware Description

This demonstration requires the following hardware:

a) SLAAC-2 daughter card mounted on a CSPI 2164S 6U VME base card.
b) CSPI 2164 6U VME PowerPC card
c) 6U VME rack
d) SPARC Workstation with Myrinet Interface
e) Myrinet switch and cables

The SLAAC-2 node (a) and CSPI PowerPC board (b) are mounted in the VME rack (c) and connected to the SPARC host (d) via the Myrinet switch (e).

2.4.1.3 Test Description

A MATLAB® script is written which calls the function FifoTest, see Figure 28. The function FifoTest is implemented in an FPGA core and performs a loopback test utilizing the input and output fifos of the SLAAC-2 compute node, see Figure 20. The FPGA core was generated by USC/ISI East as part of their debugger/verification software supplied with the SLAAC-2 processor. For this test data written to the input FIFO is echoed to the output FIFO by the program loaded into X0. The top level m file was divided into two groups as shown in Figure 28. The first group was assigned to one of the CSPI PowerPC nodes to invoke the FIFO test and display the results while the second group which performed the loop back test was assigned to one or two of the SLACC-2 compute nodes. The mapit GUI for the test shown in Figure 29 depicts the assignment of processing groups to hardware compute elements. The group information menu for group 2, which contains the SLACC-2 FPGA enhanced compute elements, is shown in Figure 30. This figure shows the assignment of the fifo test core to the group. The test was then executed. A pass/fail flag is sent back from each of the SLAAC-2 nodes so that the size of the results matrix at the end of the test indicates the number of FPAG nodes tested. In addition a xterm window can be used to log into the individual processors under test to verify operation of the test on the node itself.

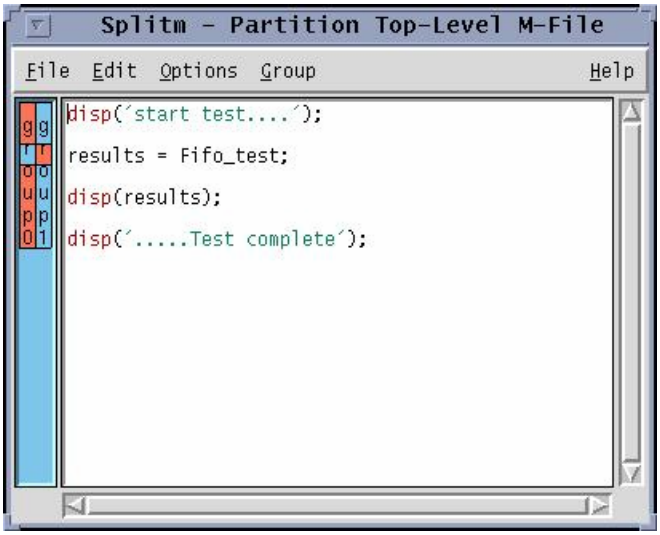


Figure 28 SLAAC-2- FIFO Test Top Level MATLAB® Script

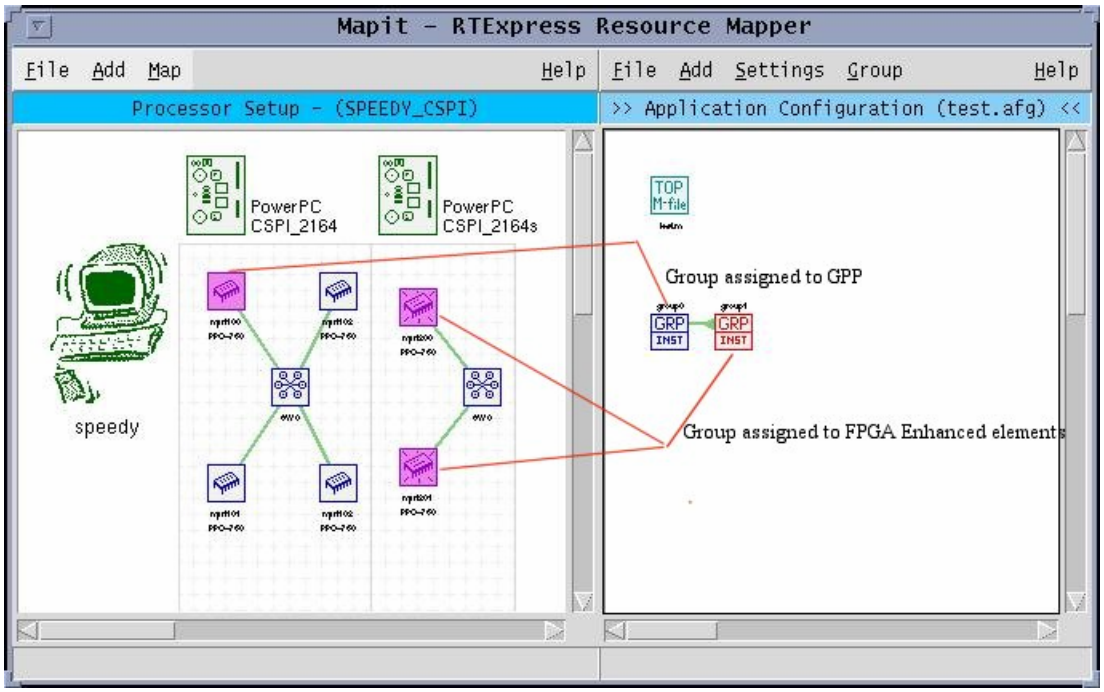


Figure 29 SLAAC-2 FIFO Test mapit setup

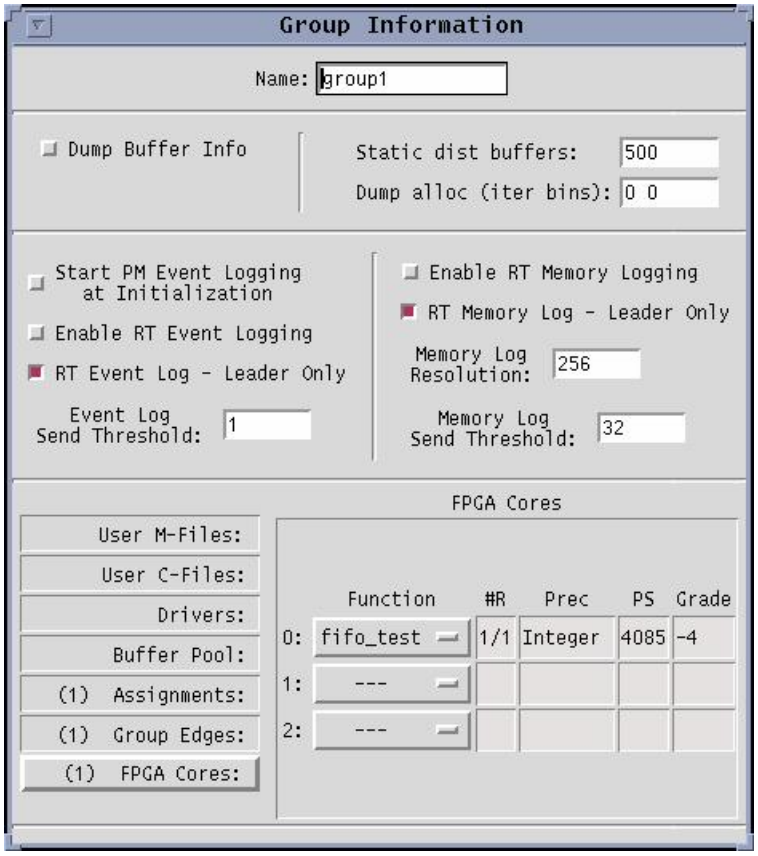


Figure 30 SLAAC-2 FIFO Demo Group Information Menu

2.4.1.4 Results

We successfully demonstrated the capability to assign portions of the overall function design to one or more SLAAC-2 processing nodes. The test was successfully initialized, completed, and terminated in an orderly fashion. The extensibility of the FPGA functional library was also successfully demonstrated since additional FPGA node tests were added to the SLAAC-2 FPGA core library. These functions included the following:

a) A memory verification test in which a checksum is computed for each of the ten memories on the SLAAC-2 daughter card after they have been initialized. The checksum is passed to the host node where it is verified.
b) Interface verification test that verified data integrity over each of the crossbar connections.

Figure 31 shows the ASCII descriptor file of the SLAAC-2 Core library. All three functions described above are included. The top portion of the file indicates there are three individual FPGA devices, which can be utilized on the node and their names. The lower table contains an entry for each library core. The fields for each entry are the following:

1) Func : This is the MATLAB® function name used to invoke the C wrapper file “mlf<Func>.c” which will provide the core interface.
2) Prec : Data precision implemented in the core.
3) Fname : Base name of the FPGA image file to be loaded. If the function requires more than one core to be loaded the files loaded are Fname_<dev>.extension.
4) Clock : These three fields represent the default clock speed selection and the minimum and maximum clock setting respectively.

5) PartSize : Identifies the FPGA type. In the case of the SLAAC-2 processor there is mode than one type of FPGA device on the daughter card so only the X0 part is listed.
6) SpeedGrade : Identifies the FPGA device speed grade. As with PartSize in the case of multiple speed grade parts on the board only the speed grade of the first device is listed.
7) Library : Identifies additional libraries which must be linked with the application to support the FPGA board API. Each FPGA processor has its own library which encapsulates the vendor supplied API. Additional libraries may be required to support the vendor API.
8) LEDs : Provides support for debugging LEDs supported by the FPGA processor.
9) Devs : Lists the FPGA devices utilized for the function. For the SLAAC-2 library the fifo test requires only the first device while the remaining two tests, memory checksum and x-bar test, requires cores to be loaded in all three FPGA devices.
10) Notes : Provides space for additional information supplied by the core developer.

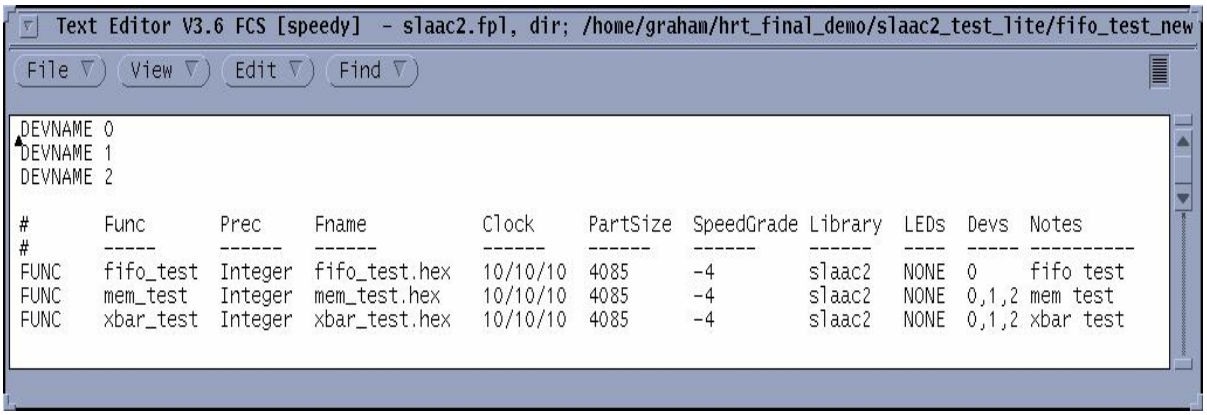


Figure 31 SLAAC-2 Core Library

2.4.2 Variable Data Precision

2.4.2.1 Purpose

This demonstration showcases the capabilities of H-RTExpress™ which provide the ability to define the precision used to compute an individual line of MATLAB® in the top level m-file. The precision select is limited to single or double precision IEEE floating point.

2.4.2.2 Hardware Description

The demonstration required a SPARC Workstation running Solaris 2.6

2.4.2.3 Test Description

The test was preformed using a portion of a test suite developed for RTExpress™. The top level m file called three different test suites, suite_01, suite_mlm,and suite_02. Suite_01 tests basic assignment statements, suite_mlm tests transcendental functions, and suite_02 tests basic math functions. Within the suite_mlm test there is a section, which test the exp function. The test suites perform a number of tests, collect the results, and then compare the results to results generated by executing the same scrip in MATALB® on a SPARC host. If the computed results are not within a tolerance of the truth values than an error is indicated to the user. For this demonstration the tolerance for the exp test results was set such that the test would fail if it was executed in single precision and pass if it was executed in double precision. All other tests pass in both single and double precision.

2.4.2.4 Results

The following cases were executed and the test results for the exp test recorded in the following table. The results of the test for the correct precision are indicated in the last column. The criterion used was the exp test should pass if it was executed in double precision and fail if it was executed in single precision.

System Precision	Precision of line calling suite_mlm	Exp test results	Test Results
single	single	fail	PASS
single	double	pass	PASS
double	single	fail	PASS
double	double	pass	PASS

The test results along with examination of the C source files indicated that the tool was processing each line with the desired precision.

2.4.3 Challenge Problem – AMS WILDSTAR™

2.4.3.1 Purpose

This demonstration showcases the application of H-RTExpress™ to a real world problem. Lockheed Martin Government Electronic Systems was tasked with providing a real world problem for evaluation of the H-RTExpress™ environment. Two candidates were proposed. The first candidate was an electronic countermeasures analysis (ECMA) system and the second was a coherent sidelobe cancellation (CSLC) system. The second candidate was chosen since the ECMA system posed implementation and data sharing problems since many of the techniques and data used for analysis were either classified or proprietary.

A top-level block diagram of the Coherent Sidelobe Canceller Algorithm is shown in Figure 32. The algorithm processes N channels of complex data and is divided into two main sections, Weight Calculation and Weight Application. Weight Calculation subsamples the input data stream to generate a complex set of weights which are applied to the full bandwidth input data to suppress sidelobes. Weight Application is just as the names describes the application of the complex weights generated by Weight Calculation to the input data stream. For the H-RTExpress™ demonstration only the Weight Application processing was implemented since the Weight Calculation was considered company proprietary by LMCO.

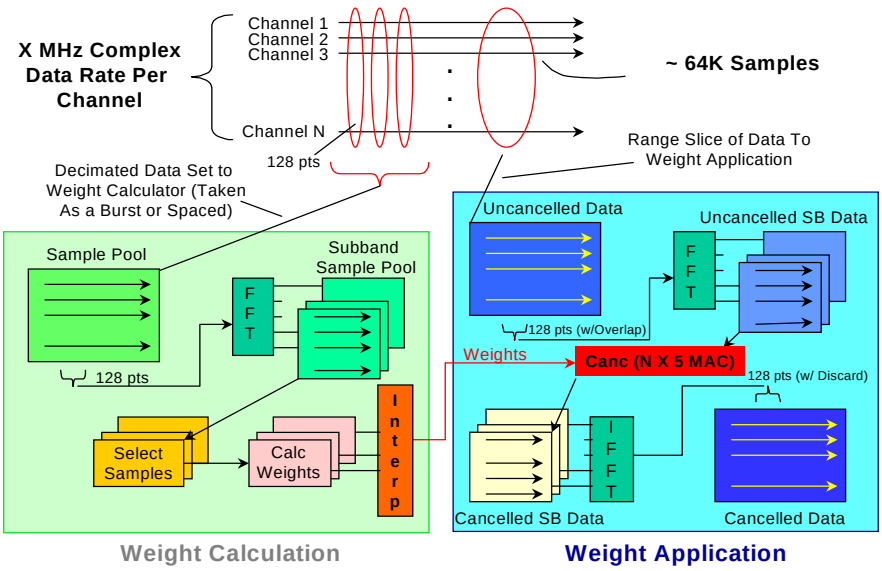


Figure 32 Coherent Sidelobe Canceller Algorithm System Block Diagram

The Weight Application algorithm processes four channels of input data and two sets of frequency domain weights to perform coherent sidelobe cancellation. The algorithm shown in Figure 33 produces two channels of output data. The cancellation algorithm uses subband processing, small FFTS on overlapping segments of range. The input data subbanding and FFT processing is shown in Figure 34. The subbanding process generates 128 point vectors from the input vector, overlaping each vector by 16 samples. For the 8192 point input vector this will create 73 128 point vectors as input to the FFT processing. The weights were adaptively derived form the same data set prior to cancellation by the Weight Calculation processing. The output data inverse FFT and subband processing is shown in Figure 35. After the inverse FFT the center 112 (128 – 16) data samples are taken from each inverse FFT output. The resulting vector contains 8176 points (Vector length (8192) – Overlap (16)).

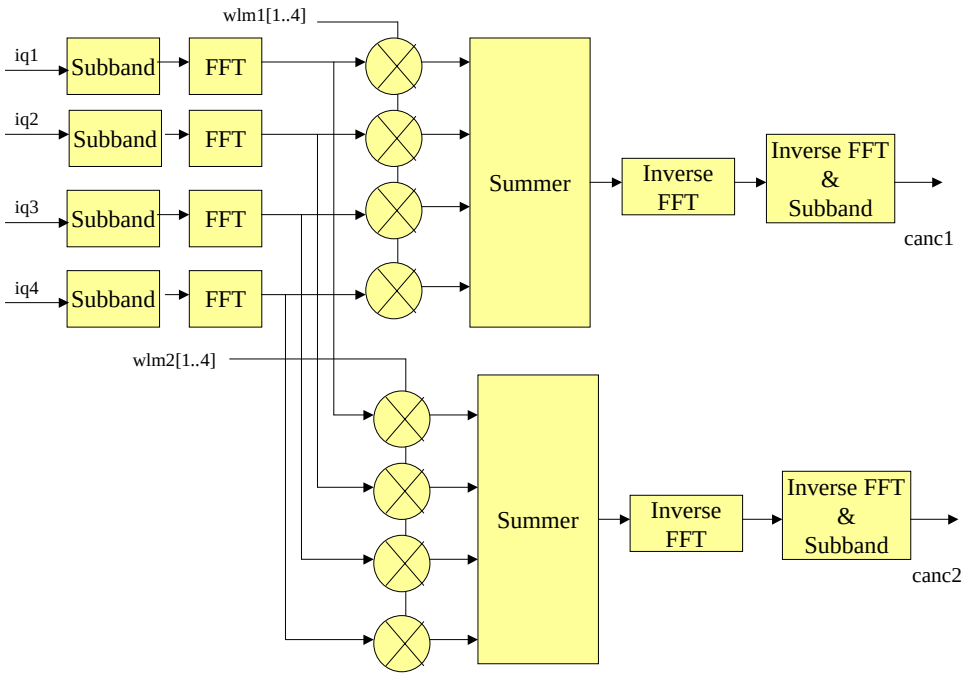


Figure 33 CSLC Algorithm Processing Block Diagram

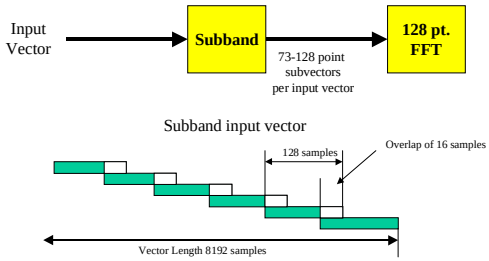


Figure 34 Input data subbanding and forward FFT

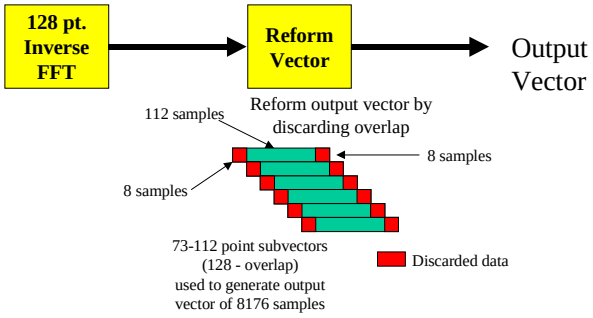


Figure 35 Inverse FFT and Regeneration of Output Vector

The CSLC algorithm was originally to be implemented using a heterogeneous environment, which consisted of PowerPCs on a Myrinet 2167 VME board and a WILDSTAR™ VME FPGA processor. The algorithm was partitioned such that the FPGA processor performed the forward and inverse FFTs and the PowerPCs performed the remaining parts of the Weight Application processing. Data transfer between the FPGA and PowerPC compute elements was via a Myrinet interface. The WILDSTAR™ VME processor was purchased with a Myrinet I/O daughtercard. The proposed Myrinet based processing flow is shown in Figure 36. However the discovery of the following two interfacing issues caused the final implementation to be changed.

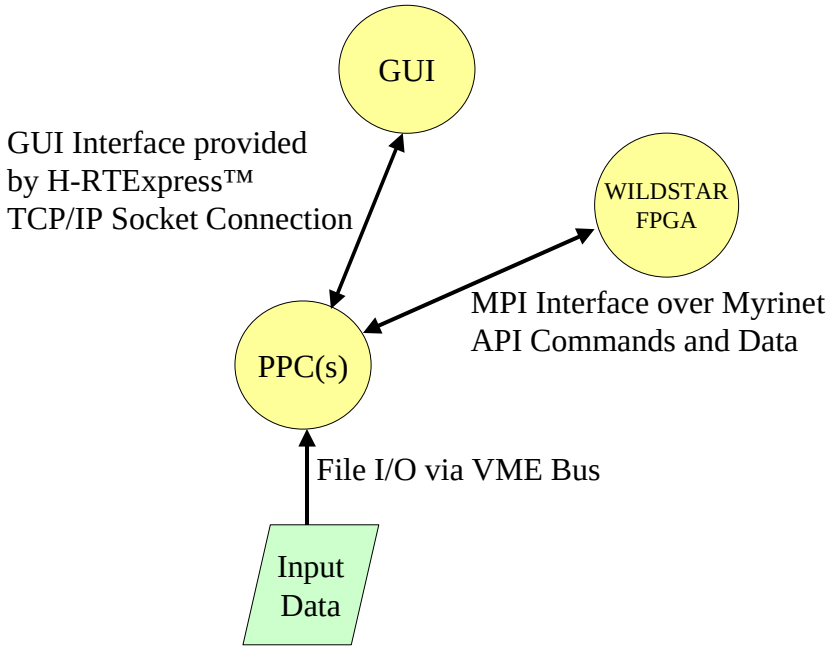


Figure 36 Proposed CSLC Demonstration Interfaces

2.4.3.1.1 Myrinet Interface

CSPI provides an MPI (Message Passing Interface) Application Program Interface (API) that is based on MPI Software Technology Inc.'s (MSTI) BDM/Pro™. BDM/Pro™ provides a low level interface to the Myrinet fabric. BDM/Pro™ consists of two parts. The first is a host library, which is linked with applications running on the host processor, and the second is the Myrinet Control Program (MCP) which executes on the LANai network processor. Each compute element on the Myrinet network contains a LANai network processor. The network processors monitor the Myrinet bus for messages for its associated host processor, send message from the host processor, and respond to configuration message requests from the network-controlling node.

The MCP for the WILDSTAR™ Myrinet Interface card is from Myricom which does not support the message extensions of BDM/Pro. CSPI maintains BDM/Pro™ independent of Myricom. A LANai Software Development package was downloaded from Myricom and an attempt was made to modify the AMS MCP. Progress was made in modifications of the MCP to receive BDM/Pro™ messages but a complete solution would have required changes to AMS cores resident on the myrinet interface daughter card. This effort was terminated since these cores are considered AMS proprietary and changes to the cores would have required reallocation of project resources and funds.

2.4.3.1.2 PowerPC API

Originally the WILDSTAR™ API was to be ported to the PowerPC. This would enable the CSPI PowerPC Compute element paired with the WILDSTAR™ node to communicate directly with the FPGA processors via the

VME bus. It was discovered however that the CSPI 2167 design does not extend the VME bus onto the board other than for power and ground connections. This means that the PowerPC compute elements on the CSPI 2167 board have no means to access the VME bus. The PowerPC port was terminated since the PowerPC compute element has no access to the FPGA.

2.4.3.1.3 Final Demonstration Hardware Environment and Interfaces

Since the Myrinet FPGA interface was not an option a decision was made to use Mercury compute elements and dedicate the CSPI system to the SLAAC-2 processors. Since a PowerPC based API was not available the WILDSTAR™ API was executed on the SPARC host and a communication protocol was defined such that the PowerPC could send and receive interface commands and results via an interface process executing on the host. The data flow for the API interface is shown in Figure 37. Normally H-RTExpress™ would interface to an FPGA processor by linking in an API support package specific to the processor that translates the generic application program requests to the required vendor supplied API call(s). Since there is no direct path to the WILDSTAR™ control registers from the controlling PowerPC an intermediate level of communication was added. In this case WILDSTAR™ API requests from the PowerPC originating program were relayed via file I/O to a host based task which invoked the correct API call. All data required for the API call and /or results from the API call was also relayed via the interfacing file. Handshaking was used for all command and data transfers to ensure synchronization between the asynchronous tasks. Because of the API design a command from the PowerPC to the WILDSTAR™ must be completed (i.e. receipt of the API response or a verification message) before processing continued.

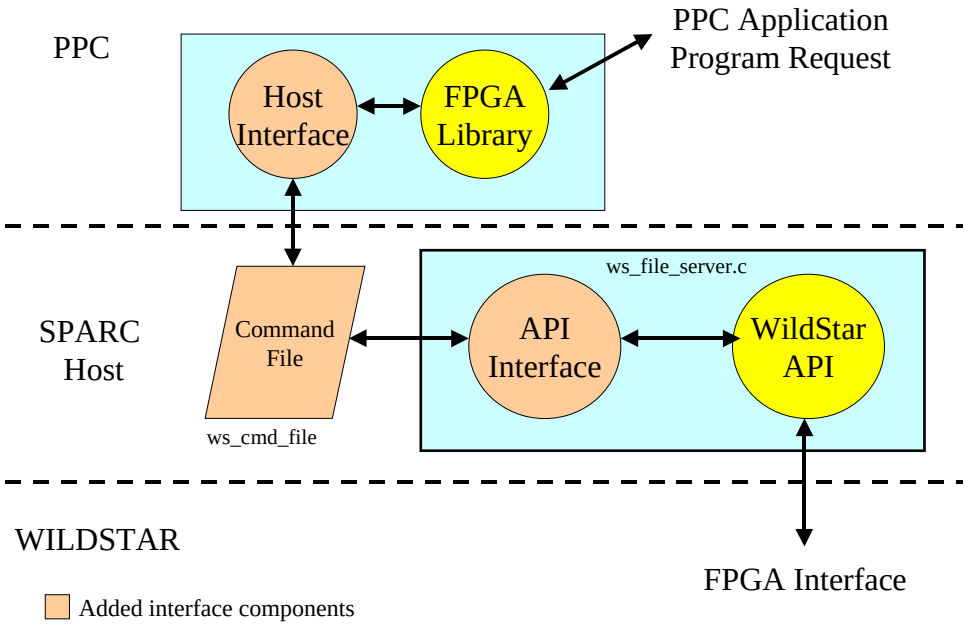


Figure 37 WILDSTAR™ Enhanced API

2.4.3.2 Hardware Description

This demonstration requires the following hardware:

a) Mercury 6U/9U VME chassis with SPARC 20 VT host.
b) Mercury MCH9 (9U) baseboard with 8 P2A16B daughter cards. Each daughter card contains 2 PowerPC 603e (200 MHz) processors and 16 MB of RAM,
c) AMS WILDSTAR™ VME card

2.4.3.3 Test Description

The weight application processing was divided such that a PowerPC compute element performed all operations except the forward and inverse FFTs. Lockheed Martin provided a WILDSTAR™ core that performed a fixed point (16 bit) 128-point FFT. The core could be controlled to perform either a forward or inverse FFT. A description of the core interface and control is included in Appendix I – Lockheed Martin FFT Core Description. An earlier version of H-RTExpress™ was used to build this demonstration, which included all the features of the final version of H-RTExpress™ except the new mapit interface. The mapit GUI used is shown in Figure 38. Only two of the PowerPC compute elements are assigned, one to the GUI interface node (tgtgui) and the other to the top-level m file. The top level M file calls functions that are implemented as user C files. These user C files provide the interface to the FFT cores implemented on the WILDSTAR™ compute element. Once the process starts the PowerPC continually processes the same input vector until the operator terminates.

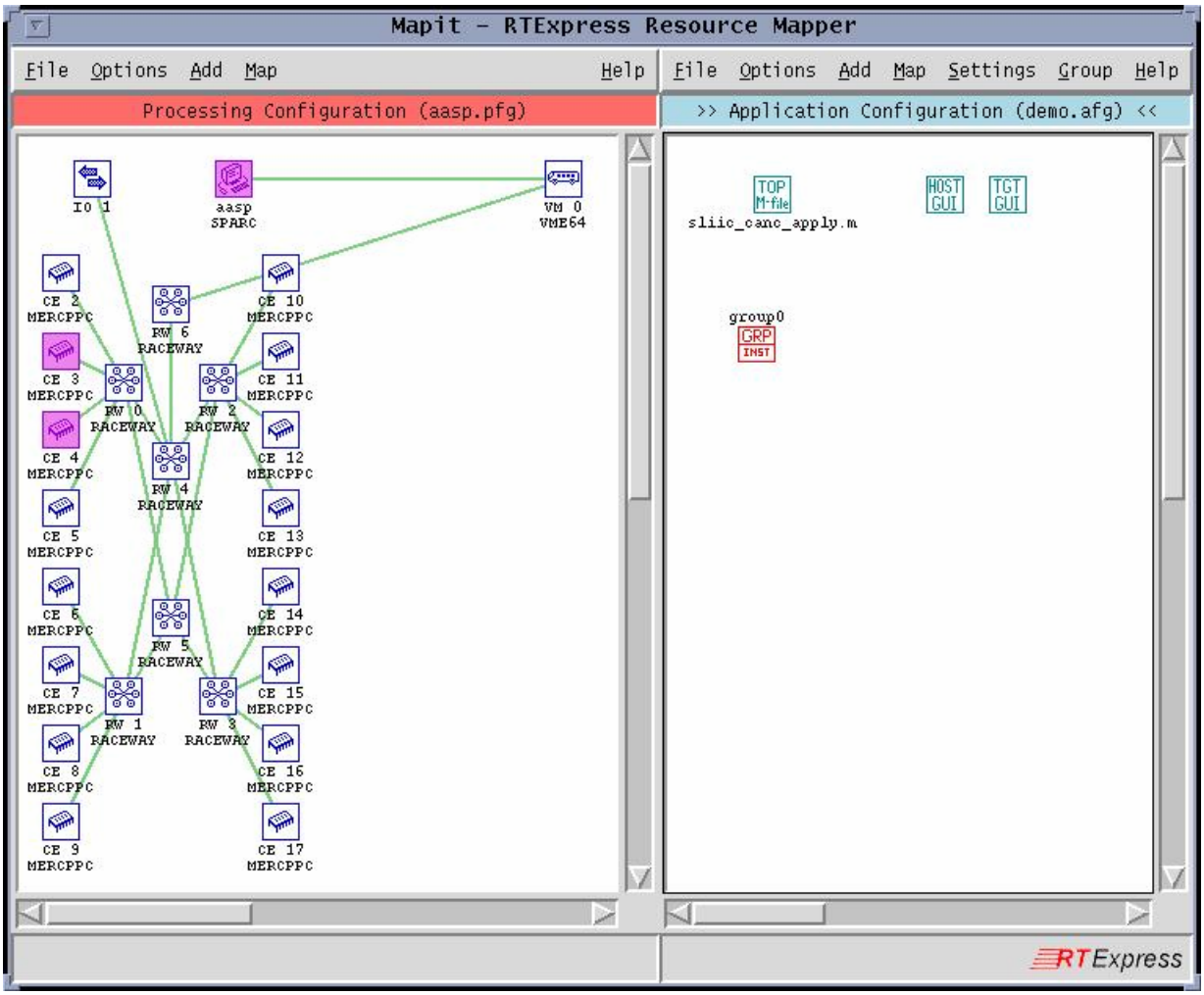


Figure 38 CSLC Process Mapping

2.4.3.4 Results

Figure 39 shows an amplitude plot of one of the forward FFTs performed on a subbanded input vector.
Figure 40 is a plot of the two channel output baseline data generated by MATLAB®. These results were obtained

by executing the algorithm in MATLAB® (double precision) on a SPARC workstation.
Figure 41 is a plot of the same variables plotted in
Figure 40 except the data was processed on a PowerPC in single precision IEEE floating point with the FFT being preformed on the WILDSTAR™ processor using a fixed 128 point (16 bit) FFT.
Figure 42 is a plot of the error between the MATLAB® baseline and the PowerPC/WILDSTAR™ processed data. Statistics of the error signal were computed and are summarized in the following table.

SIGNAL/PARAMETER	MINIMUM DIFFERENCE (DB)	MAXIMUM DIFFERENCE (DB)	AVERAGE DIFFERENCE (DB)
Channel 1	0.124	-44.490	-11.213
Channel 2	-1.243	-56.697	-15.091

Table 1 Summary of Fixed Point FFT Output Signal parameters

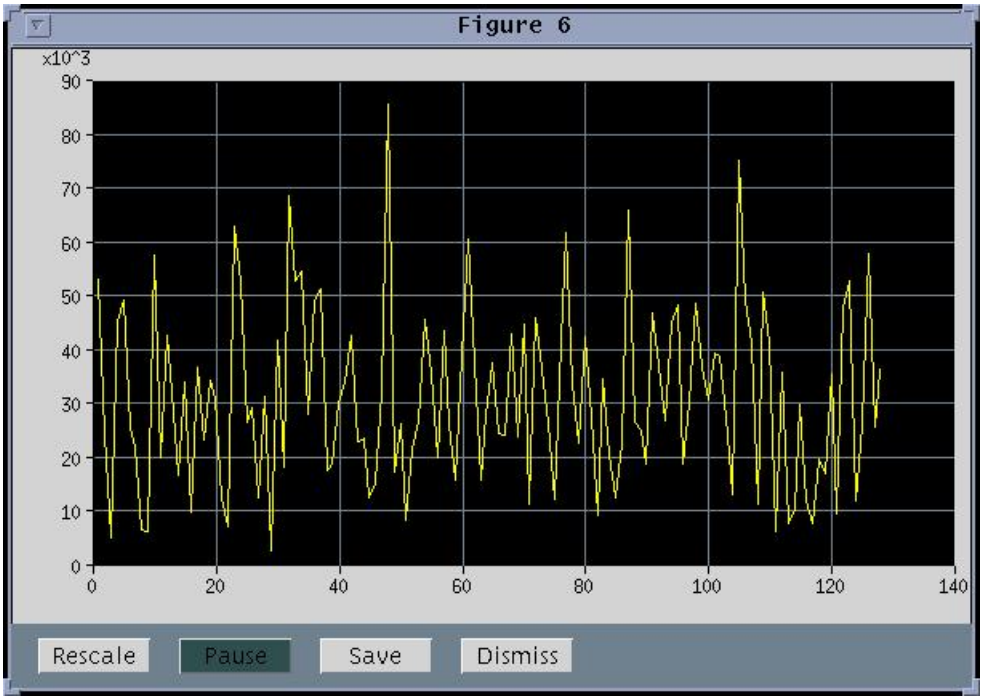


Figure 39 Forward FFT (Fixed Point FFT) of Subbanded Data

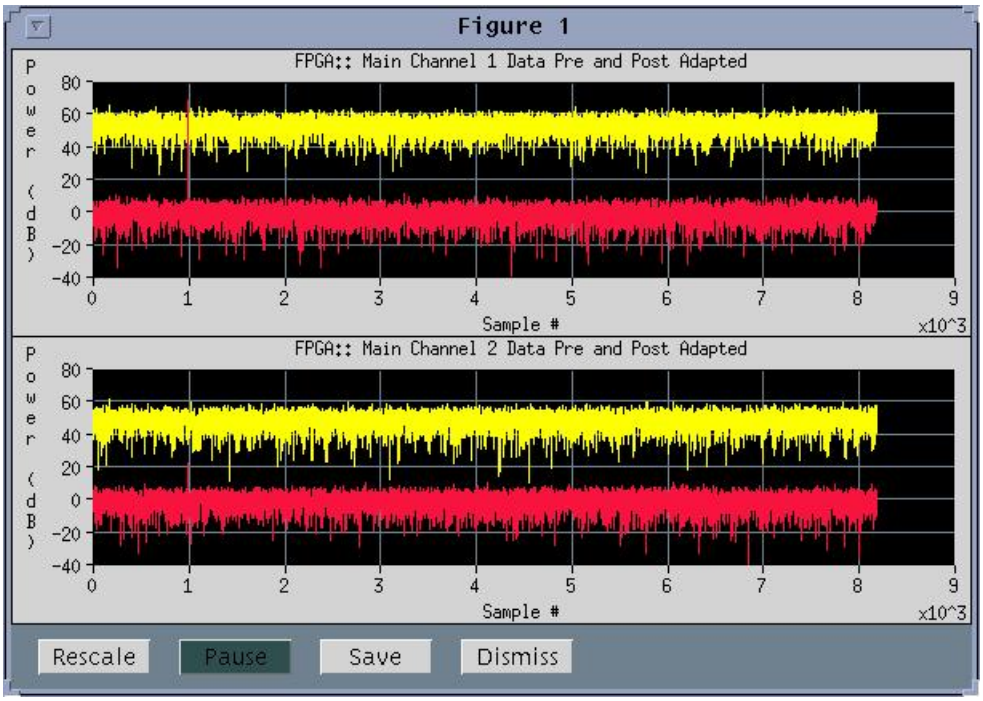


Figure 40 MATLAB® Processed Pre and Post Adapted Data

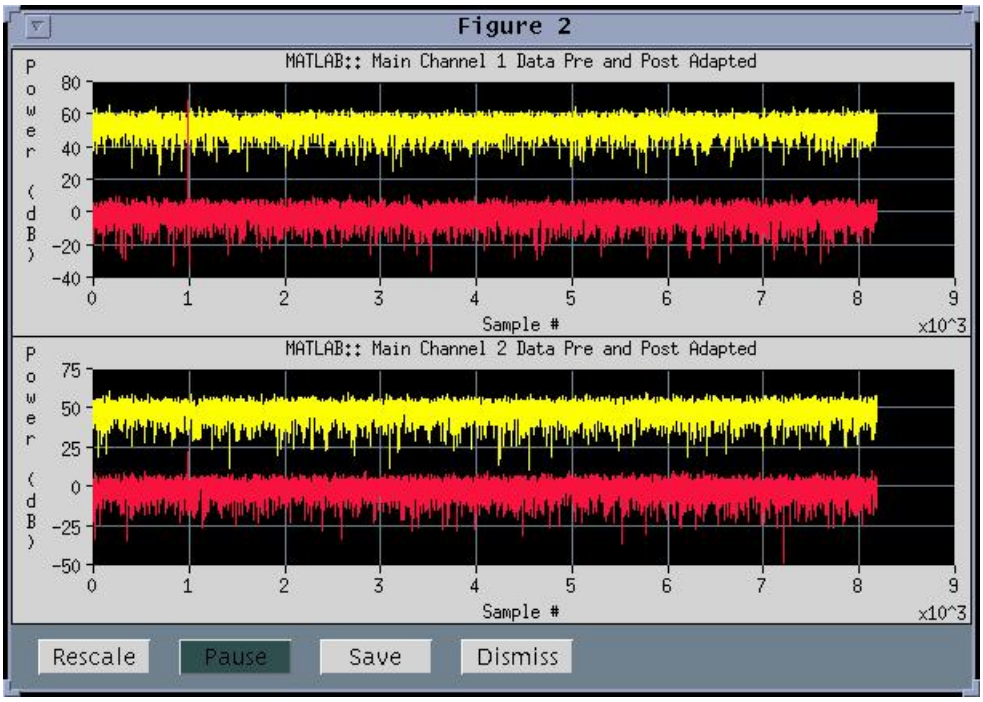


Figure 41 WILDSTAR™ Processed Pre and Post Adapted Data (Fixed Point FFT)

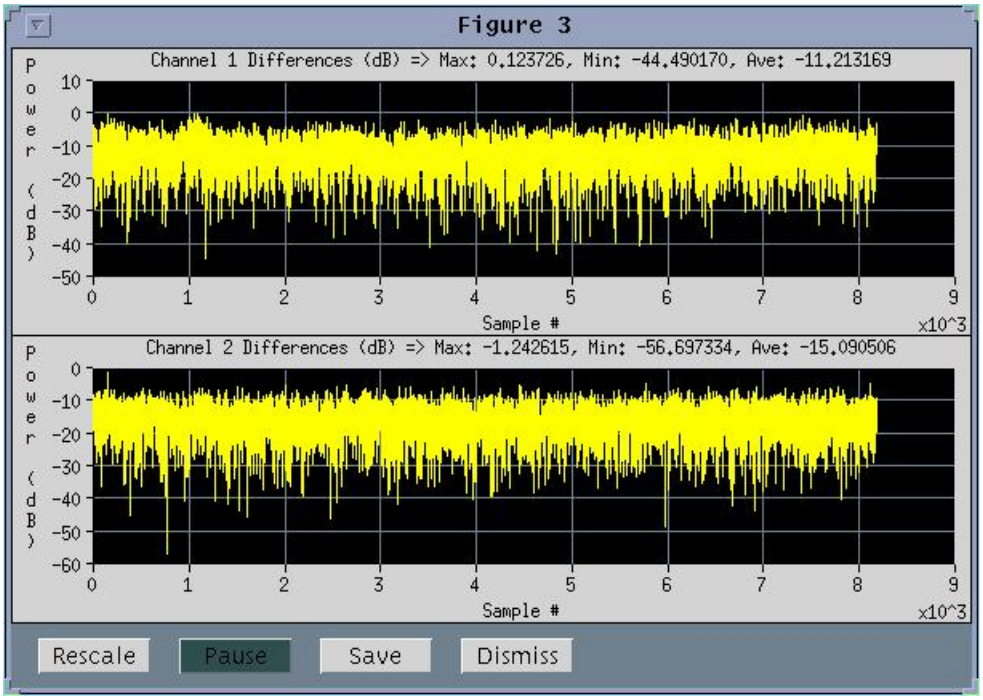


Figure 42 Differences between MATLAB® and WILDSTAR™ (Fixed Point FFT) Processed Data

2.4.4 Challenge problem – FIREBIRD™

2.4.4.1 Purpose

The same demonstration described in Section 2.4.3 was repeated with the following exceptions:

a) The test was targeted to a Linux environment with used an AMS PCI FIREBIRD™ as the FPGA target device.
b) The FIRBIRD™ performed the same function (FFT) except that a DMA interface was used to pass data to/from the FPGA board.
c) The FFT core performed all computations in single precision floating point.
d) The WILDSTAR™ Enhanced API was not required since the host processor was also a compute element.

2.4.4.2 Hardware Description

The demonstration requires the following hardware:

a) AMS FIREBIRD™ PCI FPGA card with a FPDP (Virtex™ 2000E) daughter card.
b) PC workstation running Linux 6.2

The FIREBIRD™ PCI card is mounted in the PC Workstation.

2.4.4.3 Test Description

The same procedure was followed for the FIREBIRD™ version of the test as for the WILDSTAR™ version. The mapit interface used to define the application setup is shown in Figure 43.

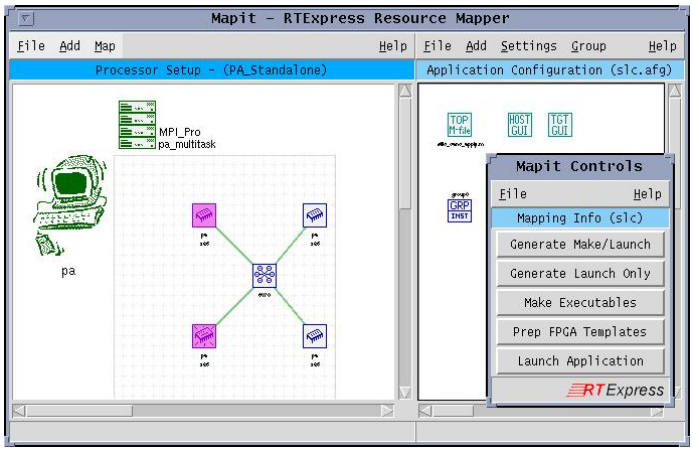


Figure 43 FIREBIRD™ Based Challenge Application

2.4.4.4 Results

Figure 44 is a plot of the same variables plotted in
Figure 40 except the data was processed on a PowerPC in single precision IEEE floating point with the FFT being preformed on the FIREBIRD™ processor using a floating point 128 point FFT. Figure 45 is plot of the error between the MATLAB® baseline and the PowerPC/FIREBIRD™ processed data. The same statistics as computed for the fixed point FFT are listed in the following table along with the fixed point results. The advantage of the floating point processing over the fixed point processing is approximately 68 db.

SIGNAL/PARAMETER	MINIMUM DIFFERENCE (DB)		MAXIMUM DIFFERENCE (DB)		AVERAGE DIFFERENCE (DB)	
	Fixed Pt. FFT	Floating Pt. FFT	Fixed Pt. FFT	Floating Pt. FFT	Fixed Pt. FFT	Floating Pt. FFT
Channel 1	0.124	-64.609	-44.490	-120.479	-11.213	-79.065
Channel 2	-1.243	-71.604	-56.697	-121.484	-15.091	-83.154

Table 2 Summary of Fixed and Floating Point FFT Output Signal Parameters

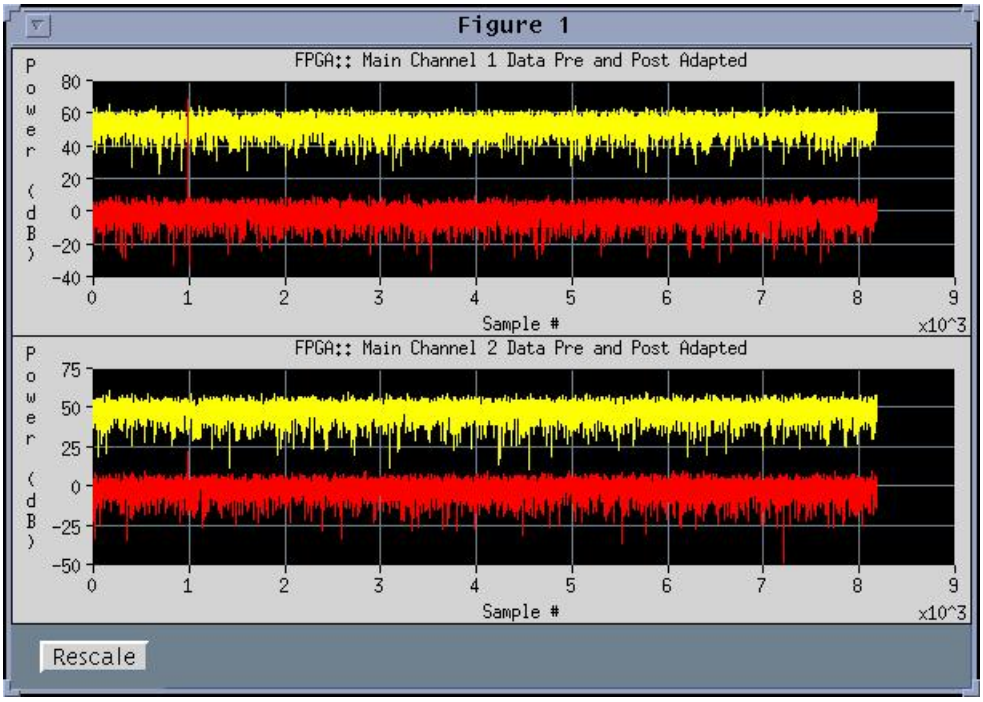


Figure 44 FIREBIRD™ Processed Pre and Post Adapted Data (Floating Point FFT)

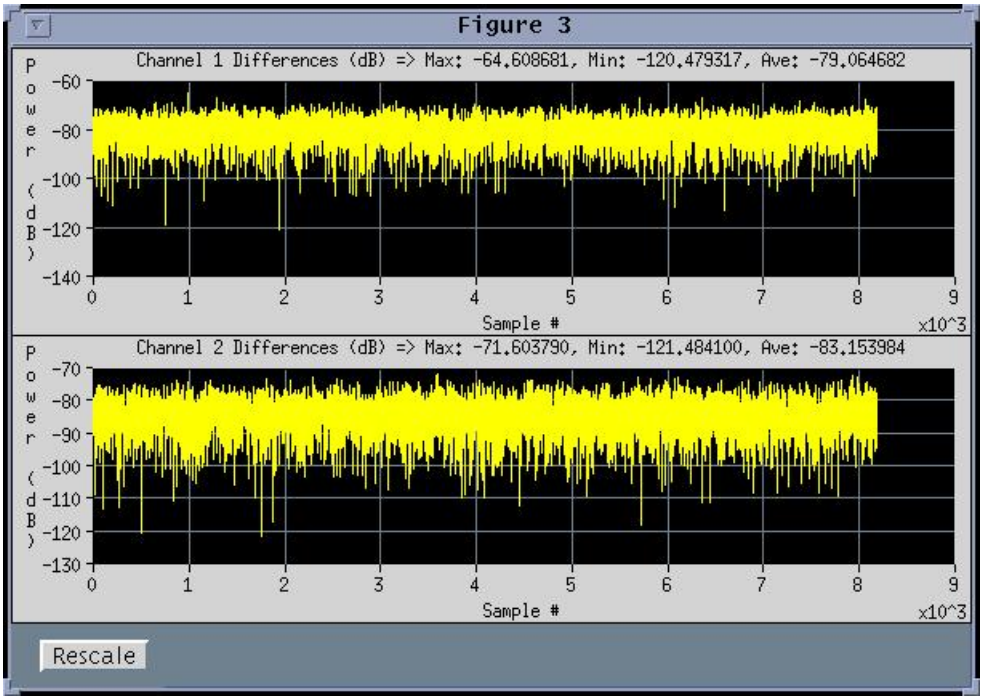


Figure 45 Differences between MATLAB® and FIREBIRD™ (Floating Point FFT) Processed Data

2.4.5 Northrop Grumman Micro Accelerator

A description of the demonstration using the Northrop Grumman Micro Accelerator is included in Appendix II- Northrop Grumman Micro Accelerator Summary.

3 Summary

3.1 Accomplishments

Extended the supported architectures in H-RTExpress to include support for the Mercury Raceway PowerPC nodes

- Prior to this H-RTExpress only supported the Mercury i860 nodes. This was the first step under this program to support multiple node types within a single box

Integrated MATLAB® 5.1 into H-RTExpress.

- Initially, H-RTExpress supported MATLAB® 4.2, but needed to be modified to support the changes that were made in 5.1 and its associated MATHWORKS C Translator
- Integration of MATLAB® 5.2 followed later, and on ISI funding, supported integration of MATLAB 6

Extended the supported architectures in H-RTExpress to include support for the CSPI Two-Level Multicomputer

splitm and editm tools

- Replaced the Target Balancing Tool (tbt) with a graphical editor
- Provided the programmer capability to functionally decompose the top level MATLAB source
- Supported the same parallel paradigms from tbt
- Point and Click selection of source code lines and assignment to processing groups
- Added help information for highlighted functions during edit

Board Editor (Bedit)

- New TCL/Tk tool added to development environment
- Allowed user to build library of hardware boards based on vendor specifications
- Supported definition and use of board templates

mapit tool

- Incorporated board level descriptions of hardware platform
- Greatly enhanced ability to transition the same application to different hardware platforms
- Supported heterogeneous hardware architectures (specifically FPGA devices)

Variable Data Precision

- Defined techniques and methods to use mixed data precision types within an application
- Defined and implemented a method for specification of floating point precision from MATLAB® level specification on a line by line basis.
- Modified and tested existing H-RTExpress™ libraries to support mixed floating point precision
- Defined and implemented C wrapper format for functions executed on an FPGA compute element

FPGA support

- Defined and implemented a layered software approach for inclusion of FPGA interfaces
- Implemented and tested procedures which initiate and load FPGA applications
- Implemented and tested procedures which close and terminate an FPGA application
- Added interface libraries for four different FPGA elements
- Added support for the USC/ISI SLAAC and Annapolis VME Wildstar
- Through other programs, added support for the Annapolis PCI Firebird and PCI Wildstar

The value of the H-RTExpress™ environment has been proved by its utilization on other ongoing DARPA programs such as Power Aware Signal Processing Environment (PASPE) for PAC/C (Contract #F30602-00-C-0150) and the Symbiotic Communications (SYCO) program (Subcontract #370020SC). Both programs used this environment to develop and implement applications on a variety of heterogeneous target systems.

3.2 Recommendations for Continuing Efforts

3.2.1 Parallel FPGA Mapping

The current model supported in H-RTExpress™ for the FPGA is that of an adjunct processor to a general-purpose processor. This means that there is a one-to-one mapping of general-purpose processor and FPGA compute element. The data transfer rate of this interface is often limited by the bus speed of the host processor. FPGA board vendors also provide very high-speed interfaces between their own products. One example of this is the WILDSTAR™ Data Port (WSDP™) interface supplied by AMS, which is capable of transfer rates up to 800 MB/sec. The ability to map a set of MATLAB® requirements to a network of FPGA processors interconnected with a very high-speed data interface such as the WSDP™ interface would be very advantageous. Support for this type of mapping would require changes to the current FPGA model supported in H-RTExpress™ and the addition of a model for the inter-FPGA communications. This task could be accomplished in a staged approach by first defining a limited model and then expanding that model as each stage is completed.

3.2.2 FPGA based message router

Several of the current FPGA board designs include multiple FPGA parts (i.e. SLAAC-2, WILDSTAR™). The current H-RTExpress™ programming model only allows a single function to be assigned to an FPGA board. A function may require multiple FPGA devices but together they only support a single function. FPGA boards are generally designed such that one of the FPGA devices handles all of the I/O for the board. In order to map more than one function to a board a message router is required which would be resident in the I/O FPGA. A routing table would be required for the FPGA so that incoming and outgoing message could be sent to the correct destination. This router while having a generic design would also be board specific since each FPGA board generally has unique I/O characteristics.

3.2.3 Multiple Data Precision

Support for more types of data precision could be added. However a more fruitful effort would be an analysis of the benefits of the MATLAB® version 6 upgrade in terms of supporting additional data precision. The front-end techniques developed under this program are still valid but the underlying software structure may make use of the variable mxArray description to simplify and streamline the required data representation and conversion processing.

3.2.4 FPGA core development tools

A significant number of FPGA designs generated using COREFIRE™ from AMS have been created and transitioned into H-RTExpress™. Some development has been done to modify the COREFIRE outputs for inclusion into the H-RTExpress™ environment. This work could be enhanced to further automate the generation of a C wrapper template for a generated function. Other FPGA development tools such as AccelFPGA™ from AccelChip and the Xilinx System Generator for SIMULINK™ could be studied to determine how and if they could be integrated into the H-RTExpress™ environment.

3.2.5 Additional FPGA Hardware support

The current effort addresses four different FPGA boards. Additional boards such as Catalina Research Inc.'s Cheetah accelerator card could be addressed.

3.2.6 Reconfiguration

An area that has not been addressed is reconfiguration of FPGA devices. The current H-RTExpress™ model assumes that once an application is loaded that it will remain resident until the system application is terminated. One of the potential strengths of the FPGA is to dynamically reload all or a portion of the FPGA device to time-multiplex tasks it performs. The rate tasks are swapped in and out would vary based on the application. Cases where reconfiguration would be helpful would be to redefine the task of the FPGA based on mode, or swap FPGA functions when one task becomes idle, or to vary the FPGA task executed based on a defined quality of service. There are several subtasks that could be addressed. These include the logistics of swapping FPGA functions, specification of the transition control (when, why), and identifications of measures of performance (i.e. when does the time to transition the FPGA cause loss of system performance).

4 Commercialization

ISI is currently integrating several H-RTExpress™ features into its commercial product RTExpress™. These features include:

a) The new mapit and bedit tools which support board level hardware architecture description.
b) The enhanced mapit features, which allow seamless transitions from one architecture type to another.
c) mapit support for FPGA enhanced compute nodes. This includes the generation of launch scripts to include the FPGA based nodes, support of core libraries and an interface to allow the user to map FPGA based functions to specific devices within the FPGA board.
d) Run-time environment changes that support launching an application and its orderly termination on a heterogeneous hardware environment.

In the future ISI plans to implement support for multiple data precision. Currently the RTExpress™ product is being ported to support MATLAB® 6.0 and compiler version 2.2. This upgrade will provide enhanced capability and a much more efficient compiler. The multiple data precision implementation needs to be researched in light of the upgrade.

ISI also plans to provide the ability to incorporate a MATLAB® to executable FPGA core tool so that users will be able to directly map MATLAB® source lines to an FPGA device. The structure of the H-RTExpress™ environment should allow an easy transition to such an FPGA develop tool once they reach maturity.

5 Appendix I – Lockheed Martin FFT Core Description

<SEE ATTACHED>

APPENDIX I

LOCKHEED MARTIN

ENHANCED

MICRO ACCELERATOR

DAUGHTERCARD

PROJECT

Included in this appendix is Rev. A (7/25/01) of the H-RTExpress™ AMS WILDSTAR VHDL FFT Cores memo written by Lockheed Martin.

H-RTExpress Annapolis Micro Systems WildStar VHDL FFT Cores

SUBJECT: H-RTExpress Wildstar VHDL Core Description

ABSTRACT: This memorandum describes the functional characteristics and interface requirements of the VHDL FFT Cores for the Annapolis Micro Systems (AMS) Wildstar VME Reconfigurable Computing Engine.

If the cores described in this memorandum are improved or modified as dictated by changing functional requirements, corrections, funding, and schedule, the changes may be reflected in revisions of this document.

Initiated:_____ Initiated:_____

W. J. Branham, Firmware RI — D. Koch, CM Engineer

SPY-1F TIP Firmware Development — Software Configuration Management

---

1.0 OVERVIEW

1.1 Purpose

The purpose of these VHDL FFT Cores is to support the controlled demonstration and evaluation of the H-RTExpress utility's capability to generate a real time executable image from a Matlab application. These cores implement a fixed point FFT and IFFT on the AMS Wildstar VME Reconfigurable Computing Engine (RCE).

1.2 Introduction

The VHDL cores implement a complex 16 bit fixed point FFT and IFFT function on the AMS Wildstar VME RCE. These cores were built for execution on the Xilinx XCV-1000 FPGA parts. The Wildstar VME RCE has 3 Processing Elements (PE) designated as PE0, PE1, and PE2. Core image 'pe0_1000.m68' should only be downloaded and executed on the PE0. Core image 'pex_1000.m68' can be downloaded and executed on PE1 and PE2 of the card.

2.0 FUNCTIONAL DESCRIPTION

2.1 FFT

Each VHDL FFT core has two modes of operation: 1) FFT, and 2) IFFT. Data loaded into the core in FFT mode should be in a 32 bit format as shown in Figure 46. Internally the each 16 bit integer is sign extended to 23 bits. Not more than 128 points will be processed at a time by the FFT engine.

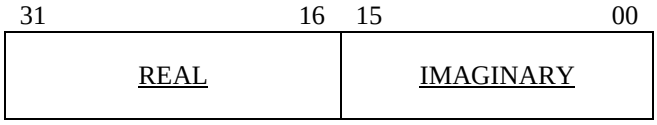


Figure 46 Data format for FFT core input

2.2 IFFT

All input data must be divided by 128 prior to downloading into the IFFT core. Internally the core multiplies the data by 128 before processing it. All resultant/output data will be returned in a 32 bit integer format as specified in figure 1 above. No more than 128 points will be processed by the IFFT.

3.0 AMS WILDSTAR C API

AMS has defined and implemented a C API and driver to interface the application to the VHDL core running on the Wildstar RCE. Reference the AMS Wildstar Reconfigurable Conputing Engine User's Manual, section 6.

4.0 TEST DRIVER

4.1 Test Driver Interface

The 'wsdbg.exe' is a quick and dirty test driver written in C for the UNIX environment, that employs the AMS Wildstar C API and Driver to communicate and control the target system. It provides capability to perform PE programming, data download and upload, and block(ie. internal to FPGA) memory reads and writes. The test driver is invoked by entering 'wsdbg< CR >'.

4.2 Test Driver Commands

The test driver is a command line oriented, case insensitive utility suitable for limited debugging within a Unix C Shell. Table 1 below defines the currently supported test driver commands.

Table 3 Test Driver Command Definition

help	Prints a list of the supported test driver commands
exit	Ends session and exits the test driver
dbg	Prints test driver internal test point information
Jou	Opens a journal/logging file to record the test session transactions
open	Open a Wildstar VME board in slot [1 - ?]
close	Close a Wildstar VME board in slot[1 - ?]
dcnfg	Display the hardware configuration of the Wildstar board
cnfg	Configure the M Clock of board in slot[1 - ?] to N Mz. At present only the '-f' switch is supported. Clock frequency is in floating for format
reset	Resets the board in slot[1 - ?]
pgmpe	Program the PE[0 – 2] of board in slot[1 - ?] with the image specified by file spec
dnld	Download a complex vector specified by file spec onto board at slot[1 - ?] and PE[0 – 2] starting at address[80(H) – FF(H)]. The file must be ASCII format and the vectors must be fixed point decimal integer format starting at the specified address.
upld	Upload a specified number of complex points to file spec, starting at [80(H) – FF(H)] from the board in slot[1 - ?], PE[0 – 1].
rper	Read 1 memory location at address[80(H) – FF(H)] from slot[1 - ?], PE[0 – 2].
Wper	Write 1 memory location at address[80(H) – FF(H)] of slot[1 - ?], PE[0 – 2]. The location will be loaded with the specified 32 bit word

6 Appendix II - Description of Northrop Grumman Micro Accelerator

<SEE ATTACHED>

APPENDIX II

NORTHROP GRUMMAN

ENHANCED

MICRO ACCELERATOR

DAUGHTERCARD

PROJECT

1 Introduction and Background

Northrop Grumman Electronic Systems (NGES) is under contract to develop and demonstrate an enhanced Micro-Accelerator FPGA Daughter Card. The daughter card can be configured by the user as an algorithm accelerator or preprocessor within a Raceway-based COTS processing system, and is used to perform multi-GFLOPS/GOPS processing functions for SAR, STAP, GMTI, EO, and other military sensor processing applications. The daughter card will provide 10X to 50X improvements in processing size and weight for stressing preprocessing functions such as FFT, QR decomposition, and radar and EO filtering. NGES has accomplished this task by upgrading its existing micro-accelerator daughter card design.

This paper is the Final Report for the Enhanced Micro Accelerator Daughtercard program and project.

2 Applicable Documents

The following documents of the issue shown, form part of this report to the extent specified herein.

2.1 Acronyms Used

The following is a definition of acronyms used in this report.

ACS	Adaptive Computing Systems
BIT	Built In Test
CPLD	Complex Programmable Logic Device
DARPA	Defense Advanced Research Projects Agency
DCI	Digital Controlled Impedance
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
FPGA	Field Programmable Gate Array
ISE	Integrated Software Environment
ISI	Integrated Sensors, Incorporated
MCOS	Mercury Computer Operating System
MCS	Mercury Computer Systems
NGES	Northrop Grumman Electronic Systems
NGNS	Northrop Grumman Norden Systems
OS	Operating System
PQFP	Plastic Quad Flat Pack
SSRAM	Synchronous Static Random Access Memory
TQFP	Thin Quad Flat Pack
VHDL	VHSIC Hardware Description Language
VHSIC	Very High-Speed Integrated Circuits

2.2 Contract Number

The program contract numbers for this project are listed as follows:

Northrop Grumman Electronic Systems Contract Number:	ISI01-105
Prime Contract Number:	F30602-97-C-0259

2.3 Specifications

Component Specifications and Manufacturers Data Sheets, Various, Dated Sept 2001 or later
Printed Wiring Board, Board Specific Design Requirements, Dated September 13, 2001

2.4 Drawings

Drawing Title	Drawing Number
Schematic Diagram, Micro Accelerator Daughter Card	819R612
Printed Wiring Board, Micro Accelerator Daughter Card	819R613
Circuit Card Assembly, Micro Accelerator Daughter Card	819R614
Mercury Computer Systems Type “B” Daughter Card	DWGNO

2.5 Standards

American National Standard for Raceway Interlink, Document ANSI/VITA 5.1-1999

2.6 Application Notes

Mercury Computer Systems:

Raceway I/O Products Guide TC-RW-IOP-311, Dated November 01, 2000
Race++ Racetrack Overview PowerPoint Presentation, Dated May 14, 2001

Xilinx, Inc.:

Xilinx Virtex-II Platform FPGA Handbook, Document UG003 V1.3, Dated December 3, 2001

Sanmina Corporation

Buried Capacitance Design Guide, Document Number 95-06-001

2.7 Design Reviews

Northrop Grumman Electronic Systems:

Concept Review, Dated April 25, 2001
Tradeoff Studies Review, Dated May 22, 2001
Preliminary Design Review, Dated June 29, 2001
Electrical Design Review, Dated October 2, 2001
Produceability Review, Dated October 24, 2001
Final Design Review, Dated October 25, 2001
Design Verification Review, Dated November 26, 2001
Design Demonstration Review, Dated February 12, 2002

Mercury Computer Systems

NGNS Racetrack Review I.doc Dated August 30, 2001
NGNS Racetrack Review II.doc Dated October 11, 2001

2.8 List Of Test Equipment Required

The following is a list of Test Equipment and Revision needed to support the Micro Accelerator development and operational environment.

- Personal Computer (description of machine used during Micro Accel development):
  Hardware:
  1.5 GHz Pentium Based Processor,
  30 Byte Hard Drive
  2 Byte RAM (needed for Xilinx FPGA development only)
  CDROM drive (software installation support)
  Software:
  Windows NT OS
  Xilinx ISE Version 5.1 with Service Pack 1 (Needed for FPGA Image download support)
  Xilinx Chipscope Version 5.1 (Needed for Design Development and Verification Support)
  Visual Elite V_HDL Development Tool, Version 2.0.2 (Needed for Design Development Only)
  Synplicity Synplify Version 7.1 (Needed for Design Development Only)
  Model Technologies ModelSim PE Version 5.6A (Needed for Design Development Only)
  Cypress Warp Version 5.2 (Needed for CPLD Design Development Only)

- Test Set
  Tracewell Systems 8 slot VME64X chassis, P/N 580-6021-F00-00 (Contains Integrated Power Supply)
  DY4 SVME179 Single Board Computer (or Equivalent) running MCS VxWorks driver package
  MCS MCJ6 Motherboard with Firmware Revision 2D
  With MCS Power PC 750 Dual Processor Node in slot A/B

- Test Software
  Mercury Computer Operating System (MCOS) Version 5.6
  MCS C compiler (for compiling Test Driver Program Only)
  MCS VxWorks Runtime Package driver related to MCOS 5.6
  Tornado 2 Latest Version used to build VxWorks Boot image for DY4 processor.
  Test Driver Program: Nacc.c
  Test Driver Program Test Files: TBD

- Test Equipment
  Northrop Grumman Norden Systems designed Micro Accel Interface Cable, P/N Engineering Design. (Design information attached)
  Volt/Ohm Meter (Typical, For Design Debug Only)
  Xilinx Parallel II Serial Interface Cable Model DLC5 and JTAG Header with Flying Leads
  Cypress UltraISR Programming Cable P/N 37KISR.03
  Hewlett Packard Logic State Analyzer Model 1660ES (Typical, For Design Debug Only)
  350 MHz Bandwidth Oscilloscope (Typical, For Design Debug & Support Only)

3 Design and Development

This section discusses the design and development aspects for the Enhanced Micro Accelerator FPGA Daughter card.

3.1 Concept Definition And Requirements

3.1.1 Concept Definition

As stated in the introduction, the concept of this project centered around the development of a Raceway based hardware architecture and design, such that various Radar, EO, and/or IR compute intensive algorithms could be remapped into this hardware, and their respective processing “accelerated”, providing benefit to the overall system.

3.1.2 Requirements

The requirements for this design were established fairly early within the project cycle, and are listed below for reference.

- Hi Performance FPGA based Accelerator Daughter Card Design
  - Compatible with Mercury Type B daughtercard slots
  - Design upgrade from current Micro Accelerator card
  - Convective cooled design
- Reconfigurable per computation algorithm required
- Upgrade the design to the latest FPGA technology
- Design to best commercial practices
- Upgrade pathway for all components to industrial temp grade and technology expansion where possible
- Interface upgrade from one Raceway 1.0 port to two Race++ ports
- Capability to download FPGA programming data via Race++ port
- Increase in FPGA to local memory bandwidth and size
  - Provides higher efficiency in data-rate intensive computations

3.1.2.1 Requirements Impact

Note that several of these requirements imposed design constraints, as discussed below.

The requirement to fit the design into an MCS Type B daughtercard imposed the most severe restrictions, dictating component height and location constraints on the overall design.

For example, a review of the Type B daughter card design "rules" indicates:

- all components located on the "B" (secondary) side are restricted to a height of .050";
- all components located on the "A" (primary) side are restricted to a height of .170";
- the Printed Wiring Board thickness is limited to a "maximum" of .062", to fit within adjacent module pitch requirements in the intended MCS chassis environment;
- Total power dissipation in a standard MCS chassis environment (with standard air flow and standard module pitch) is limited to approximately 17 Watts per daughter card board.

When considering an increase in FPGA to local memory bandwidth and size, it was decided to provide as much local memory as could practically fit on the PC board, in addition to components that could provide a very high data communications bandwidth. Since there was no "application" design intended for this project, NGNS felt that a reasonable target for memory usage and bandwidth would be to be able to fit the NGNS 1 BOPS FFT ASIC onto the design, at least in terms of memory requirements for the FFT engine. The FFT ASIC required 12 RAMS, and so set the goal for this design.

A consideration of having to provide a latest technology reconfigurable compute node precluded the use of one time programmable (ROM based) FPGAs, and indicated the use of reprogrammable (RAM based) FPGA technology. At the time of initial design, Xilinx Inc. and Altera were offering competing products in terms of large RAM based FPGAs. Xilinx, however, came out ahead, offering a new family of FPGAs in the Virtex II family. The Xilinx Virtex-II XC2V6000 was chosen based on availability, an upgrade pathway to larger devices, built-in "hard" multipliers, and NGNS previous experience with Xilinx toolsets and devices.

3.2 Architecture

During concept phase, several architectural approaches were evaluated to fulfill the design requirements. The one that was chosen was driven by several factors:

- Previous Micro Accelerator programming experience indicated that a design that "separated" memory addressing and data movement into more than one FPGA was difficult to design, operate, and program for operation at high clock speeds. A design that was able to incorporate these two areas (local memory addressing and data) into one FPGA would make design implementations easier.
- The architecture provides a good fit in the sense that it provides a relatively easy VHDL design environment: all processing areas are contained within one programmable part. Competing accelerators in the industry typically have more than one smaller FPGA with crossbars to connect memories to FPGAs and/or FPGAs to FPGAs.
- The architecture provides a more flexible and adaptable processing node, in that either a coprocessor architecture (two processing nodes, bi directional data on each I/O port, which is typical of MCS computation nodes) or a data flow architecture (one processing node, data input on one I/O port, data output on the other) can be implemented, depending on the application requirements.
- The FPGA compute node can be split into two unique and different applications, again providing application flexibility.
- A balancing of power dissipation versus logic density/performance is obtained by the use of a programmable clock circuit. Raceway interface areas operate at the maximum Raceway bandwidth as provided by the incoming Raceway clock. Very dense logic designs within the FPGA can be clocked at a "slower" rate, and less dense designs can be clocked at a higher rate, in order to balance power dissipation and the operational environment.
- In order to maintain a high module I/O bandwidth, both Race++ I/O ports are implemented in MCS supplied FPGA devices, capable of providing sustained maximum I/O bandwidth with minimal overhead, as well as off the shelf performance.
- To provide low latency and high bandwidth to the local memory, the use of Synchronous Static RAM (SSRAM) was chosen. The Micro Accelerator design is intended to provide hardware acceleration functions for digital signal processing applications (FFT, Q-R factorization, matrix operations, etc). Most DSP functions require non sequential memory accesses; Static RAMs in general provide higher performance for this DSP type of switching among random address locations as compared to dynamic RAMs (DRAMs), in that SSRAMs do not require delayed access times due to pipeline delays, and can maintain bandwidth throughput while providing true random access memory location data.

3.2.1 Block Diagram

The functional block diagram for the Enhanced Micro Accelerator FPGA Daughtercard is shown in Figure AII-1. Key points of this diagram are:

- One Xilinx FPGA, a Virtex-II XC2V6000 as a compute node
- 12 banks of independently accessible 512K x 32 SSRAMs
- Race 1 Clock Recovery/Reprogrammable regeneration
- Single Chip Bi-directional FIFO on each Race++ path.
- Local Boot PROM for multiple versions of Xilinx FPGA image storage
- Configuration manager for download of Xilinx image over Raceway, and storage of image on Micro Accelerator.
- Local 2 phase switching power supply for FPGA V_{CORE} voltage of 1.5Volts. This is required because of the large amounts of core current needed when the FPGA is programmed for hi logic density or hi speed applications.

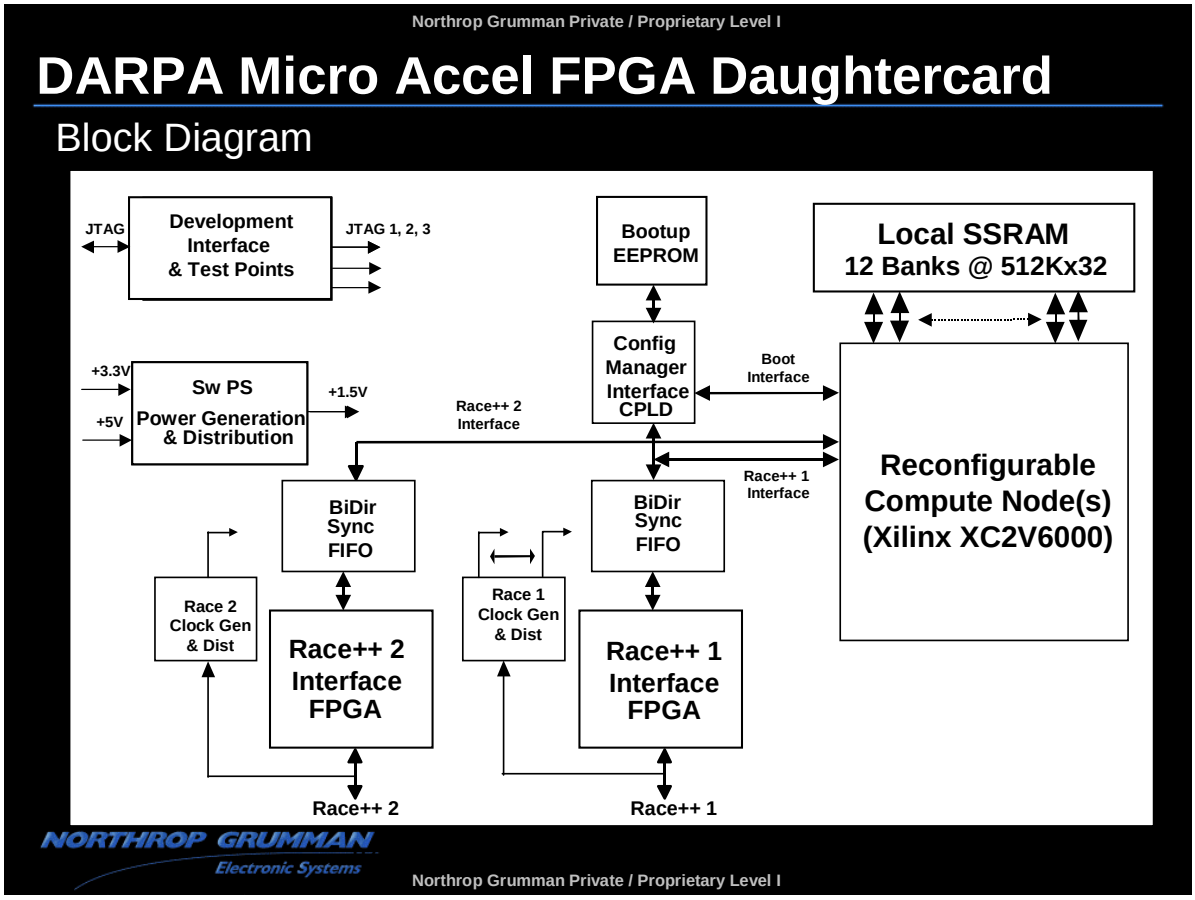


Figure AII-47 Enhanced Micro Accelerator FPGA Daughtercard Functional Block Diagram

3.2.2 Component Selection

Component selection was done with the following guidelines.

- All device packaging had to "fit" within the allocated height limitations dictated by the MCS Type B daughtercard limitations.
- Each component had to provide an upgrade pathway to industrial temp components, and to the "next generation" devices that would eventually come on to the market place.
- All major components required at least one alternate or backup vendor to supply parts. In all cases except for the programmable devices (Xilinx FPGA, Cypress CPLD, MCS Race++ interface FPGA), this is true.

With this in mind, Figure AII- 48 lists the major components selected for use in the Enhanced Micro Accelerator FPGA Daughter card. Figure AII-1 provides a detailed block diagram for the Micro Accelerator.

DARPA Micro Accelerator CDR

P/L; Power; SPL Status; 1 of 2

P/N	DESCRIPTION	QTY.	EST POWER Watts	NG P/L	SPL	
05083C104MAT	CAPACITOR, 0.10 UF	145		No		No
06035C471KAT	CAPACITOR, 470 PF	1		Yes		No
0612YC105KAT	CAPACITOR, 1.0 UF	12		Yes		No
07032	INDUCTOR, 1UH	2	0.48	No		No
74LVT244WM	74LVT244	2	0.0363	Yes	Yes	
89030801 77MSDA	CONN_JACK	2		No		No
A54SX32TQ176I	RIC-ERWO_TQ176	2	3.40065	No		No
AM29LV065DU120REI	29LV065-120	1	0.005	No		No
BAT54A	DIODE	1		Yes	Yes	
C0805C105K8RAC	CAPACITOR, 1.0 UF	5		Yes	Yes	
CRCW06031001F	RESISTOR, 1K, 1/16W	2		Yes	Yes	
CRCW06033320F	RESISTOR, 332, 1/16W	1		Yes	Yes	
CRCW06034751F	RESISTOR, 4.75K, 1/1	28		Yes		No
CY2309SI-1H	CY2309	1	0.115	No		No
CY37512VP208-66NI	MICRO_ACCEL_PQ208	1	0.4	No		No
CY7B994V-5AI	CY7B994V-5	1	0.825	No		No
CY7C43684AV-7AC	CY7C43684AV-7	2	0.396	No		No
Si4442DY	TRANSISTOR	4	0.64	No		No
JL00178-003	CAPACITOR, 330 PF	1		Yes	Yes	
JL00178-009	CAPACITOR, 1000 PF	3		Yes	Yes	
JL00178-015	CAPACITOR, 3300 PF	1		Yes	Yes	
JL00179-005	CAPACITOR, 0.01 UF	1		Yes	Yes	
JL00180-004	CAPACITOR, 0.033 UF	1		Yes	Yes	
JL00180-010	CAPACITOR, 0.1 UF	1		Yes	Yes	
JL00195-011	CAPACITOR, 4.7UF	1		Yes	Yes	
JL00196-011	CAPACITOR, 15UF	1		Yes	Yes	
JL00219-001	RESISTOR, 10, 1/16W	5		Yes	Yes	
JL00219-034	RESISTOR, 22.1, 1/16	91		Yes	Yes	

Assumptions:
1) SSRAM: 6 Memories accessed at 25% duty cycle (R or W) per process at F_{max}; 6 Memories in ZZ mode
2) EEPROM duty cycle = 5% .
3) Actel FPGA estimated typical power
4) CPLD: estimated power
5) Xilinx FPGA: estimated power

DARPA Micro Accelerator CDR

P/L; Power; SPL Status; 2 of 2

JL00219-069	RESISTOR, 51.1, 1/16	16		Yes	Yes	
JL00219-130	RESISTOR, 221, 1/16W	3		Yes	Yes	
JL00219-147	RESISTOR, 332, 1/16W	3		Yes	Yes	
JL00219-193	RESISTOR, 1K, 1/16W	26	0.26	Yes	Yes	
JL00219-226	RESISTOR, 2.21K, 1/1	3		Yes	Yes	
JL00219-258	RESISTOR, 4.75K, 1/1	2		Yes	Yes	
JL00219-318	RESISTOR, 20K, 1/16W	1		Yes	Yes	
JL00219-339	RESISTOR, 33.2K, 1/1	1		Yes	Yes	
JL00219-373	RESISTOR, 75K, 1/16W	1		Yes	Yes	
LRC-LRF3W-01-R006-F	RESISTOR, .006, 3W	2	0.44	No		No
LTC1629IG-PG	LTC1629	1	0.29	No		No
MBRS320	DIODE	2	0.46	No		No
MMS-112-02-L-DH	CONN_JACK	2		No		No
MT55L512Y36PT-6	MT55L512Y36-7.5	12	3.02445	No		No
T510E108M004AS4115	CAPACITOR, 1000 UF	2	0.0006	No		No
T510X337K010AS	CAPACITOR, 330UF	2	0.0662	No		No
TNPW06037151BT-9	RESISTOR, 7.15K, 1/1	1		No		No
TNPW06038061BT-9	RESISTOR, 8.06K, 1/1	1		No		No
VTSR1601222G	RESISTOR_NETWORK, 2.	3	0	Yes	Yes	
XC2V6000-FF1152C	MICRO_ACCEL_FF1152	1	8	No		No
(SW_PS CLAD)			0.17			
	Totals		19.0092	**		

** Power consumption is application controlled

October 23, 2001

NORTHROP GRUMMAN Electronic Systems

Figure AII- 48 Micro Accelerator Parts List

3.2.3 PC Board Design And Development

Another challenging area of the Enhanced Micro Accelerator FPGA Daughtercard design was in the area of the PC Board: component layout, PC board layering stackup, and signal routing.

Referencing Figure AII-49, to properly design a daughtercard that will fit in a given MCS chassis among other "standard" modules, the MCS Type B daughtercard imposes height and thickness restrictions on the PC board and components. The suggested nominal thickness for a Type B daughtercard should be .062", along with "side A" component height restrictions of .170", and "side B" restrictions of .050".

In order to accomplish all of the architectural goals previously discussed, it was decided that the largest Xilinx Virtex II device should be used in the design, in order to provide the most application flexibility. The largest Xilinx device that could provide all the necessary memory and Race interfaces, and provide a future upgrade pathway, was the XC2V6000 in an 1152 fine pitch Ball Grid Array package. In order to provide PC board connections for this package, the PC board layering definition shown in Figure AII- 50 was established. This layering definition provided the following: a "symmetrical stackup such that, from a center point reference, both "sides" were identical in manufacture; a thickness of .075 inches; 3 power planes and 3 ground planes, of which 2 were structured to provide "buried capacitance"; 8 routing layers; one "buried resistor" layer. Note that the planned thickness of .075 required a further restriction of component heights on the "A" side of {.170 – (.075 - .062) =} .157"

The buried capacitance layers were needed because the PC board area left after component placement, did not leave sufficient room for an adequate number of standard IC decoupling capacitors, along with the desired placement of these capacitors. The buried capacitor layers provide high frequency (> 500 MHz) signal decoupling, in addition to power and ground planes for proper signal referencing. It is expected that these buried capacitance layers will provide the "desired" additional decoupling for high signal quality.

A buried resistor layer was used for a similar reason as buried capacitors: component layout so restricted the remaining area of the PC board, certain signal series termination resistors were placed on an internal PC board layer, so as to not occupy board space.

Also, in order to not increase number of layers used for signal routing, which would have increased the PC board thickness, a technique of routing 2 signals between FPGA balls spaced at 1 mm was developed. In effect, this provided a method of routing all signals onto 8 layers, as compared to a normal routing of approximately 10 layers if this technique were not used.

Also, since both sides of the PC board contained SSRAM devices, another technique was developed such that for SSRAM devices that occupied the same space but on opposite sides of the PC board, their power and ground connection vias were "shared" in the connection method to the respective power or ground layer. This reduced the number of board vias required, thus providing more room for signal routing.

All signal layers were designed to have a characteristic impedance of 50 ohms. This impedance is what the Virtex II FPGA is designed to "impedance match" to, such that the FPGA I/O drivers were programmed to use Digitally Controlled Impedance (DCI) drivers that would match to this impedance, and thereby maintain signal transmission quality. DCI is a new Xilinx technology that was exploited in this design.

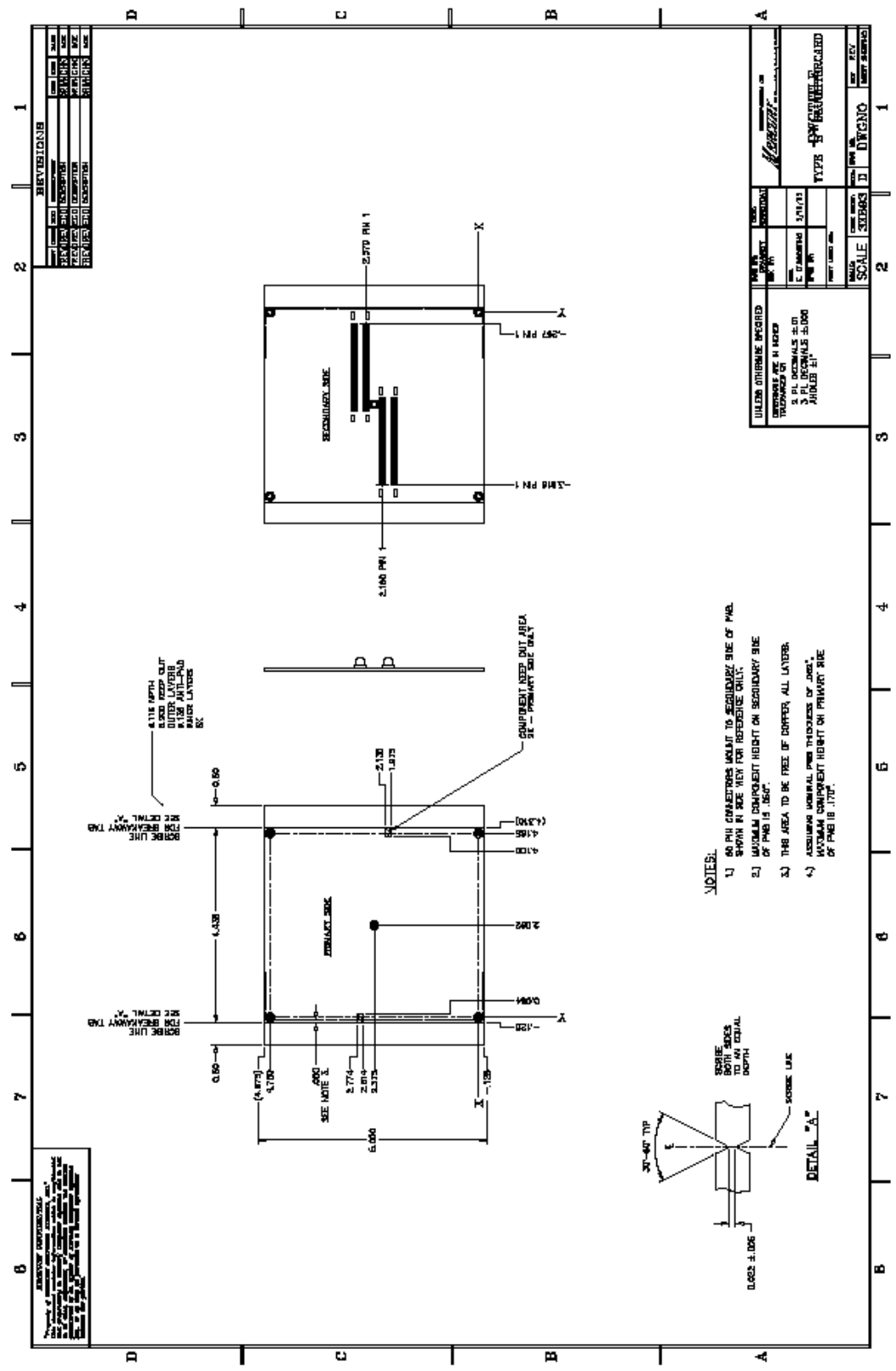


Figure AII-49 Mercury Computer Systems Type B Daughtercard Mechanical Definition

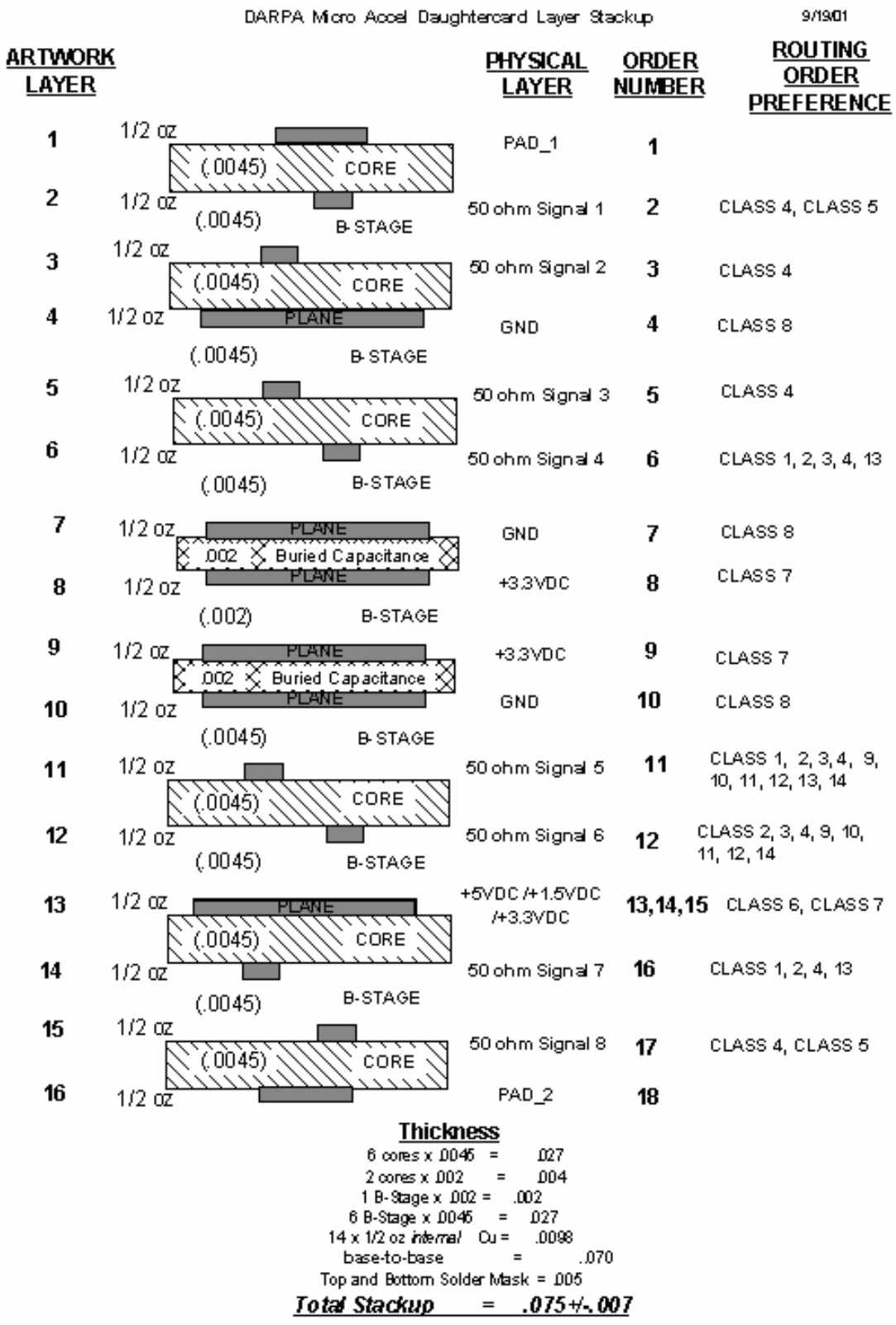


Figure AII- 50 Micro Accelerator Daughtercard PC Board Layering Definition & Stackup

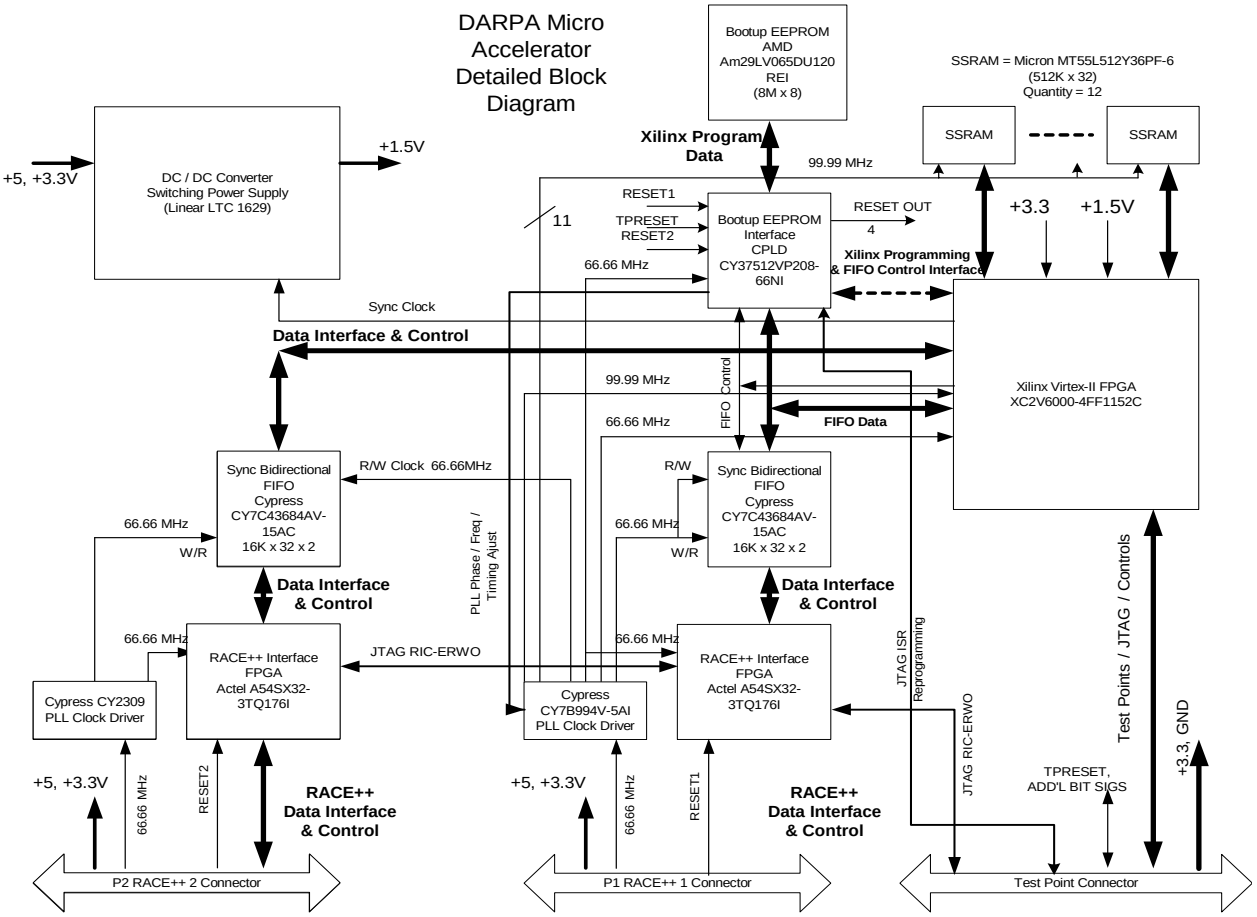


Figure AII-51 Enhanced Micro Accelerator Daughtercard Detailed Block Diagram

3.2.4 Daughtercard Assembly

Figure AII-52 shows the results of all the development aspects discussed so far. As indicated, there are components on both sides of the PC board. The Virtex II FPGA is physically located on the top side, but the interconnections extend to the bottom side, allowing signal access to all 8 routing layers. The FPGA is surrounded by the 12 local SSRAM components, and when viewed "through the board", will occupy the same "footprint" when placed on top of each other. The local 1.5V switching power supply occupies both top and bottom surfaces. The Race++ interface FPGAs occupy both top and bottom surfaces.

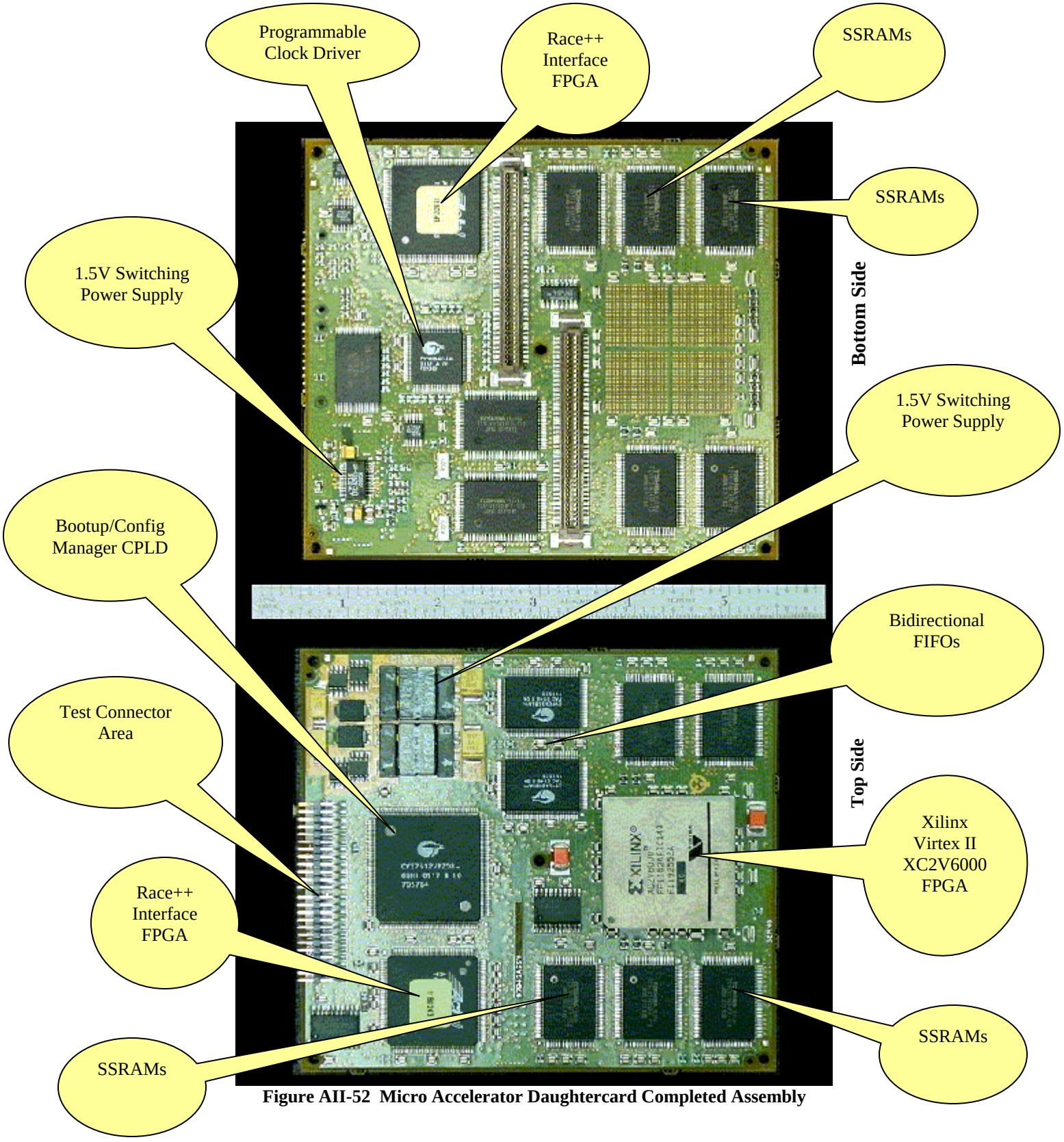


Figure AII-52 Micro Accelerator Daughtercard Completed Assembly

3.2.5 Test Program Development

This section provides a description of the test program that was developed for the Micro Accelerator, which allows each Race++ port to perform Master Write and Master Read transactions.

3.2.5.1 Test Chassis Setup

Figure AII-53 is a picture of the test chassis used during development and debug of the Micro Accelerator. As indicated, the chassis is a Tracewell 8 slot VME64 backplane, with integrated power supply. The VME processor module used is a DY-4 SVME179 single board computer. The MCJ6 motherboard consists of a dual compute element PowerPC 750 daughtercard, and the Micro Accelerator daughtercard.

The figure also shows the position of the daughtercard when mounted on the MCJ6 motherboard: left hand side, MCJ6 (or MCH6) slot C, D position.

3.2.5.2 Test Design Block Diagram

In order to successfully prove that each Race++ port could respond to Master Reads into the Micro Accelerator, and execute Master Writes from the Micro Accelerator into the Raceway network, it was decided that each port would need the capability of bi-directional data transfer (coprocessor configuration). AII-6 represents a block diagram of the high level test environment developed for debug of the Micro Accelerator daughtercard. It indicates that the Virtex II FPGA is split into two processing nodes; each one is capable of interfacing to the Race++ port in the coprocessor configuration. Each node contains an interface to 6 of 12 SSRAM memories, and supplies an internal Xilinx 256 point FFT processor, based on the Xilinx Core Gen component. A memory map for this design is shown in Table 4. Each “function” is mapped to a Raceway address, and logic is designed to interpret this address and corresponding data and react appropriately.

A more detailed block diagram of the Raceway interface function developed for use within the FPGA is shown in Figure AII- 55. Note that this figure depicts only ½ of the FPGA interface; the entire design is repeated for the second Race++ port. Both ports are configured for bi-directional data transfer. This design is written in VHDL, and uses Xilinx CoreGen components.

Table 4 Race Translated Micro Accelerator Memory Map

Note: address map changed to accommodate CPLD synthesis constraints.

Race++ Slave Address Region (Word 0 bits 31 – 29 of RIC-ERWO Slave Write Cycle Protocol) bits	Race++ Slave Address Region (Word 1 bits 27 – 3) of RIC-ERWO Slave Write Cycle Protocol) hex	Width/Alignment Nibble Word 1 Slave Address bits 31 – 28 (Race provided)	Micro Accelerator Region	Local Mapping "Regional" Translated Address hex (Addr 24:0)	Comments
000	0000000 → 3FFFFE8 Accessible in increments of 4 bytes only.	1101 = 4 byte accessing	Flash PROM	000000 → 7FFFFC	EEPROM is 8 bits wide. Race++ is 32 bits wide. Race 32 bits are mapped into EEPROM 8 bits
000	4000000 → 40000F8	1101 = 4 byte accessing	CPLD Registers	800000 → 80001F	32 registers addressable in 32 bit words only
000	4000100 → 40FFFF8	1101 = 4 byte accessing	Blank Space	800020 → 81FFFF	Spare Memory Space. Not Mapped to components
000	4100000 → 47FFFF8	1101 = 4 Byte Accessing	FPGA Register Space	820000 → 8FFFFF	917503 32 bit registers
000	4800000 → 4BFFFF8	1101 = 4 byte accessing	SRAM 1	900000 → 97FFFF	Addressable as 32 or 64 bit words. Accommodates (512K x 32) SRAM bytes.
000	4C00000 → 4FFFFF8		SRAM 2	980000 → 9FFFFF	
000	5000000 → 53FFFF8		SRAM 3	A00000→ A7FFFF	
000	5400000 → 57FFFF8		SRAM 4	A80000 → AFFFFF	
000	5800000 → 5BFFFF8		SRAM 5	B00000 → B7FFFF	
000	5C00000 → 5FFFFF8		SRAM6	B80000 → BFFFFF	
000	6000000→ 63FFFF8		FFT RAM	C00000 → C7FFFF	
000	6400000 → FFFFFFF		No memory functions.		Not mapped to components.

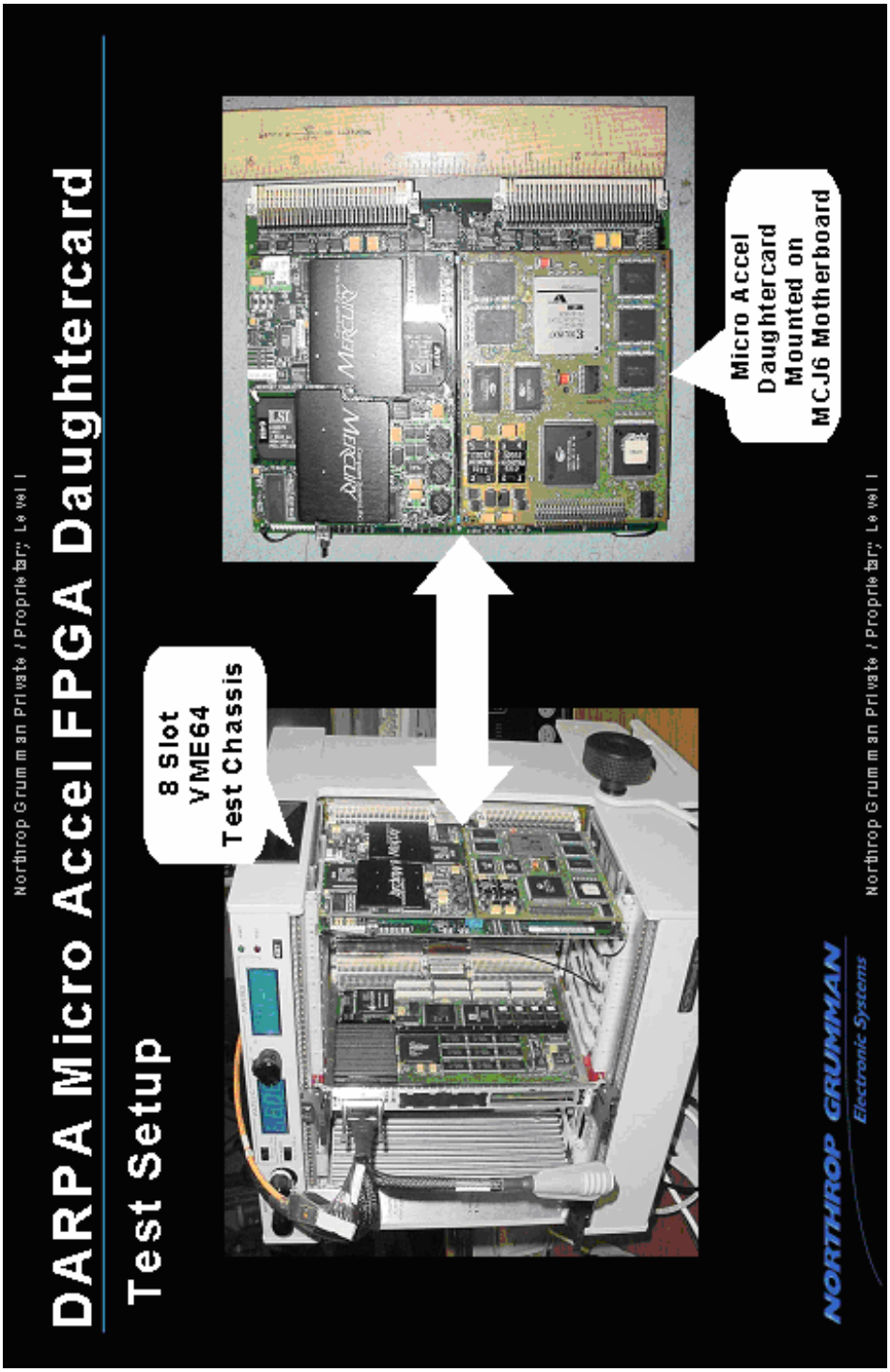
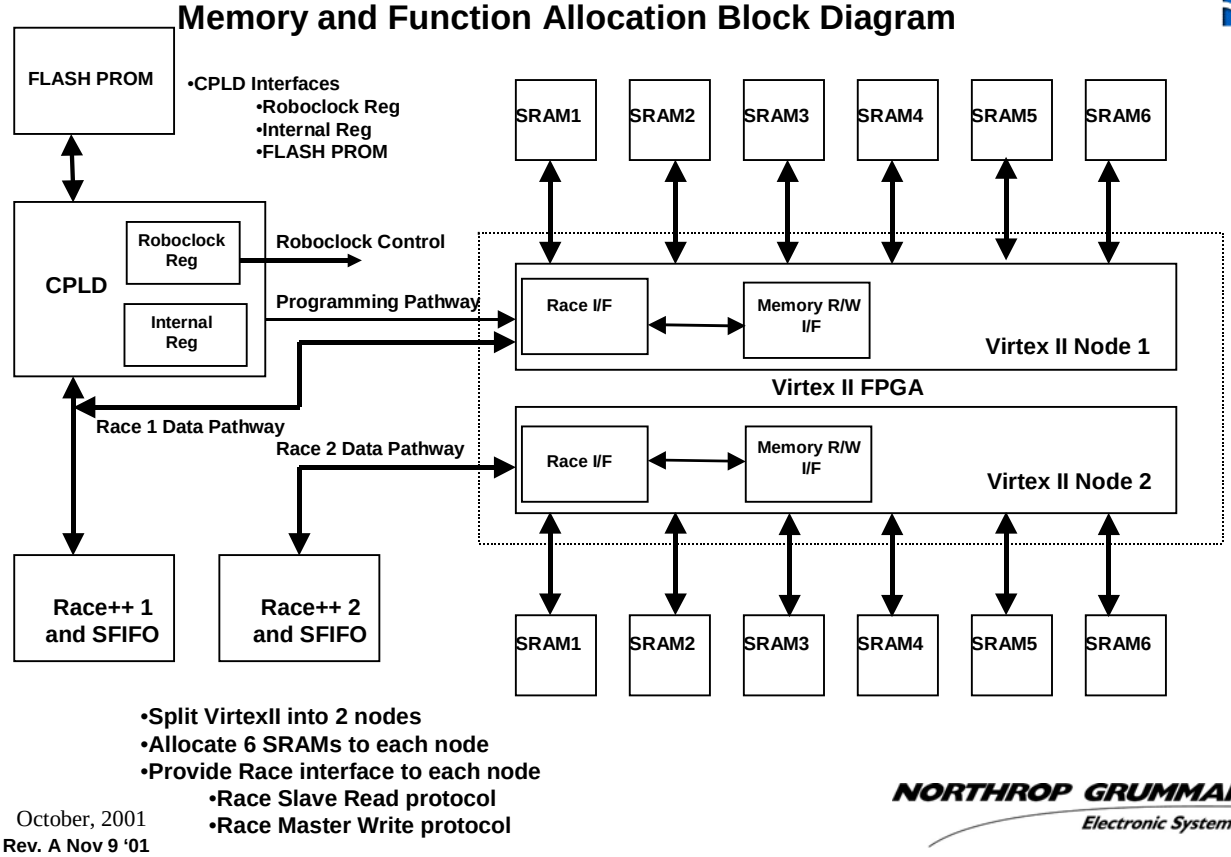


Figure AII-53 Micro Accelerator Test Chassis and Module Test Setup

DARPA Micro Accelerator DVT

Memory and Function Allocation Block Diagram



- •Split VirtexII into 2 nodes
- •Allocate 6 SRAMs to each node
- •Provide Race interface to each node
  - •Race Slave Read protocol
  - •Race Master Write protocol

October, 2001
Rev. A Nov 9 '01

NORTHROP GRUMMAN
Electronic Systems

Figure AII-54 Test & Design Verification Block Diagram

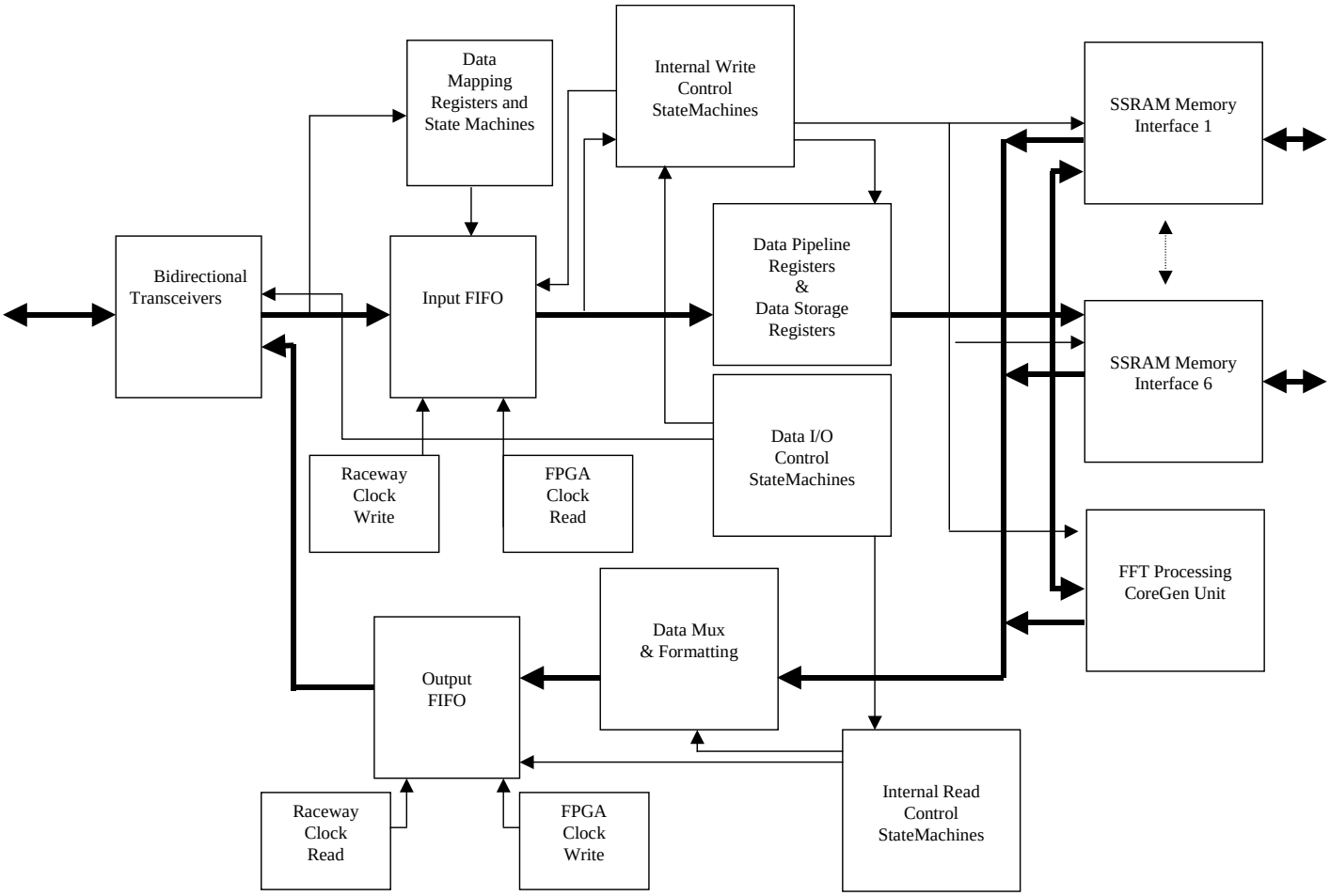


Figure AII- 55 Test Design Detailed Block Diagram

3.2.5.3 VHDL Hardware Design And Development

The intent of the test design is to provide a Master Write and Master Read capability for the individual Race++ ports on the Micro Accelerator Daughtercard. The test design provides a memory-mapped interface wherein each register or SRAM memory on the Micro Accel is mapped into a section of Raceway space, and is located and defined by an offset from the base address for each Race++ port on the Micro Accelerator. Referring to AII-55: 32 bit Raceway data is presented to the FPGA interface via bi-directional I/O registers that are default set as input drivers. Logic detects and decodes the input stream, and if the data is mapped to an FPGA designated location that is correctly memory mapped, the FPGA will respond appropriately to this Master Read input data by writing a register, writing a block of memory etc., or by initiating a Master Write cycle. State Machines provide internal strobe control and generation, data flow control, and data formatting. All "external" interfaces are driven in synchronization with the Raceway derived clock, and all "internal" FPGA processes are driven in synchronization with the higher speed FPGA clock; FIFOs provide the clock domain decoupling.

Tools used for this development include:

- Visual Elite Version 2.0.2 for VHDL Source Development
- Synplicity Synplify Version 7.1 for logic synthesis
- Xilinx ISE Version 5.1 Service Pack 1 for logic place and route and device programming via JTAG
- Xilinx Chipscope Version 5 for internal FPGA debugging.
- Xilinx CoreGen for development of FPGA Intellectual Property
    - FIFOs, FFT Processors, SSRAM Memory Interfaces
- Model Technologies ModelSim PE Version 5.6 for simulation and logic debug.

3.2.5.4 Test Software DESIGN AND Development

The software used to test the accelerator daughter-card was developed in the C programming language for Mercury's MCOS release 5.6. The development environment consisted of a SUN Microsystems Ultra 80 workstation, hosting Mercury's development environment and a Tornado II development environment from WindRiver Systems. The test VME chassis consisted of an SVME-179 single board computer from DY-4 Systems Inc., and a(n) MCJ6 Race++ motherboard fitted with a PPC-750 daughtercard (dual 375MHz PowerPC node, with 32MB per node) and the accelerator daughter-card.

The DY-4 single board computer (SBC) was booted via the network (Ethernet) using the Ultra-80 workstation as the boot host. Once booted, the SBC automatically executes a boot script, which is used to configure the Mercury Compute environment and start the remote server process. It is via the remote server process that users logged into the Ultra-80 workstation can load/execute programs on the Mercury compute nodes. As shown in Figure AII-56 Software Development Environment the DY-4 SBC serial port is directly connected to the serial port of the Ultra-80 workstation. This allows a program such as SUN's "tip" or Kermit (available open-source code) to be used as the DY-4 SBC console.

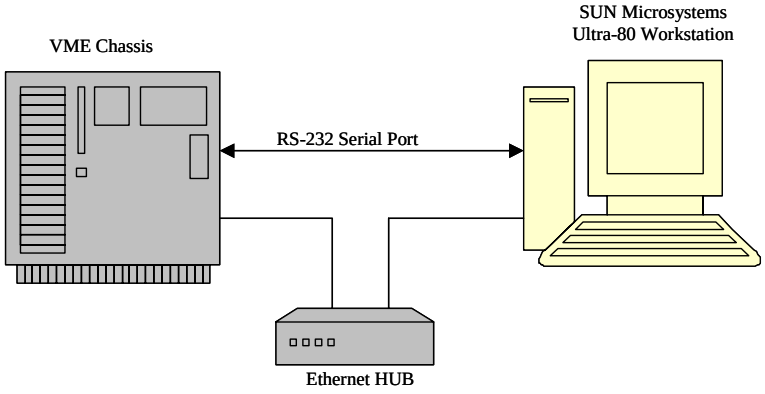


Figure AII-56 Software Development Environment

The test software design approach was based on using Mercury's dx-transfer facility. Two shared memory buffers (SMB) were created locally on one compute node serving as the master node. Under program control, the first SMB is populated with commands and associated data intended for the accelerator daughter-card. This SMB is used as the source endpoint for dx_start. The second SMB is used as the results buffer, which is used by the accelerator as the destination endpoint for output data. For added flexibility, the test software was designed to obtain command/data intended for the accelerator daughter-card form ASCII formatted text files. This approach allows for rapid "what-if" scenarios without modifying the test application software, but simply editing the text file(s).

Since the accelerator daughter-card is a race++ capable device, each raceway port (RP1 and RP2) is treated as a separate device. Control of each device is under application program control. The test application performs a device attach, specifying the named ports in the configuration file for the mercury system, NG_ACC_1 for raceway port 1 and NG_ACC_2 for raceway port 2. The test software also allows the accelerator daughter-card to signal the controlling compute node, after results are DMA'ed to the results SMB. This feature is implemented using mercury's mailbox facility.

4 Program Results

The following section describes the results of the Micro Accelerator program, including debug, demonstration, and performance observations and results.

4.1 Hardware Performance

4.1.1 Power Consumption

Initial power consumption measurements were taken as a function of an MCJ6 without, and then with, a Micro Accelerator Daughtercard, and are derived from the +5Volt supply current, and based on the following assumptions.

Each MCJ6 motherboard receives +5V, and generates +3.3V for distribution to daughtercards etc via a DC/DC converter. Assuming an efficiency ratio E_{FF} of 0.9, I_{OUTPUT} of a given supply is obtained by

$I_{OUTPUT\ 3.3V} = (E_{FF} \times 5V)/3V \times I_{INPUT} = 1.36 \times I_{INPUT\ 5V}$

$I_{OUTPUT\ 1.5V} = (E_{FF} \times 5V)/1.5V \times I_{INPUT} = 3 \times I_{INPUT\ 5V}$

Conditions to determine Power Consumption with results are given below:

Table 5 Micro Accelerator Daughter Card Power Dissipation

Condition	5V Current Measured (Amps)	3.3V Current Derived (Amps)	3.3V Power Watts, Calculated	1.5V Current Amps, (Derived	1.5V Power Watts, Calculated
"Bare" MCJ6 (without daughtercards)	2.97				
MCJ6 with One Micro Accel	3.8	1.08	3.56		
MCJ w/ Micro Accel w/ FPGA Programmed	4.4	1.08	3.56	1.8	2.7

Conclusions: The Micro Accelerator daughtercard dissipates approximately 3.5 Watts without any FPGA program loaded. The FPGA dissipates approximately 2.7 Watts with the Design Verification test program (described in 3.2.5.2) loaded, which occupies approximately 17% of the FPGA. These values appear to be within estimated power dissipation expectations.

4.1.2 Hardware Known Issues And Resolution

4.1.2.1 CPLD

During design debug, it was determined that several CPLD related functions did not operate as intended, namely the Roboclock adjustment feature, and Flash PROM read & write interface. Design errors were uncovered, requiring various corrections for both that ultimately resulted in a compiled design that would no longer "fit" into the pinout assignment of the CPLD. Although these features have been successfully redesigned and debugged via simulation, the complete fitting and programming of the entire device is not completed because of an overuse of available logic resources within the CPLD.

Resolution of this problem will require some or all of the following:

- Insertion of a new and larger CPLD into the same footprint that exists on the Micro Accel PC board, if available from the manufacturer.
- Redesign of existing functions to simplify operations, and/or elimination of functions or reduction in their complexity to obtain a proper CPLD fit.
- Re-pinout of the CPLD to accommodate these redesigned functions. Note that this will require redesign of the PC board.

Currently, the use of the CPLD is to provide daughtercard setup defaults for the Roboclock and reset circuitry. The Roboclock frequency control can be "partially" adjusted by way of commanding a Master Read function, with the Roboclock frequency adjustment control register address as the destination. Presently, a completely new setting will require the reprogramming of the CPLD, such that new Roboclock setup parameters are provided upon powerup.

The CPLD is also used to provide a Switching Power Supply Sync function, should future performance testing determine that the Switching Power Supply needs this.

4.1.2.2 FPGA

The Race++ interface within the FPGA has been debugged and will accommodate Master Read block data transfers to any location within any of the 6 SRAMs attached to the specified Race++ port. The design currently supports a Race clock speed of 66 MHz, and an FPGA clock speed of 66 MHz, and is designed to support FPGA clock speeds up to 133 MHz.

However, performance testing of this interface is required, specifically exercising the various FIFO Empty and Full conditions that will arise, depending on Race and FPGA clock speeds.

Also, as a result of the CPLD problems mentioned above, the FPGA Slave Parallel Reconfiguration mode has not been tested, since this involves correct Write and Read control of the Flash PROM via the CPLD. The reconfiguration mode currently used is via the JTAG interface, which requires the NGNS designed Test Cable .

4.1.2.3 SSRAM

During daughtercard debug, it was discovered that all SSRAM V_{IO} pins were connected via the PC board to 0 Volts (Ground). In order to properly interface to the FPGA, these power pins must be connected to +3.3V.

Further observation indicates that, at least for daughtercard S/N 001, because of the amount of time the board was powered on in this (incorrect) state, and due to logic level defaults set within the FPGA (data contention), it is suspected that several (if not all) of the SRAMs have become damaged, and will require replacement and ECN installation to correct this problem.

Consequently, only ½ of 1 SRAM (16 of 32 bits) has been verified to be fully operational.

4.1.2.4 FFT Processor

Regarding performance in this area, an examination (via Chipscope) of the internal FPGA RAM data needed for proper operation of the FFP processor, indicated that the RAMs themselves are not getting loaded with the proper input data (RAM writing is incorrect).

No further examination of this area was conducted, as the program was out of time and money.

4.2 Software Performance

The test application software has been successfully demonstrated with the accelerator daughter-card. Executing on a user specified compute node, the test application software, successfully transmits commands and data to the accelerator via raceway ports 1 or 2. Similarly, the test application software successfully received and displayed data DMA'ed from the accelerator.

4.2.1 Software Known Issues And Resolution

Perhaps the only major issue with the test applications software is the mailbox signaling has not been successfully integrated with the accelerator daughter-card. Additional time is needed to complete this integration.
At the moment, the test application software interacts only with raceway port 1 or 2, which is determined at compile time. A few modifications to the code should be put in place to attach to both ports and allow the user to specify which port the commands/data are intended.

4.3 Demonstration

4.3.1 Setup And Testing Performed

Referring to AII-9 this setup was used to demonstrate the work accomplished to date. The demonstration conducted showed the system ability to read and write ½ of one SRAM component from Race++ port 1.

5 Conclusions

The design and development of the Micro Accelerator Daughtercard described herein fulfills the requirements of the Northrop Grumman Contract in terms of design features, and requirements, flexibility, local memory quantity and projected bandwidth, upgradeability, form, fit, function, and power dissipation.
Further evaluation of the daughtercard in terms of high speed clock performance, full memory interface testing, dual port Raceway I/O bandwidth, and software performance should be conducted to confirm design issues. As mentioned previously, these design issues include:

- a high performance PC board layering structure designed to provide high frequency decoupling;
- Raceway I/O FPGAs designed to supply full Race++ bandwidth with minimal overhead,
- A VHDL designed section of hardware designed to handle the full clock extremes that the board is capable of producing;
- A flexible clock controller designed to provide minimum and maximum speed clocks to compute elements,
- A capability of storing and programming the FPGA through an on board program data store.

The demonstration of the daughtercard was also successful. A Logic State Analyzer was used to show that a Master Read and Master Write could be executed by both Race ++ ports.

Micro Accel Daughtercard
Test Plug Adapter Wiring Definition
CPLD & FPGA JTAG Interface

Pin Orientation When Looking INTO Connector Edge

24	22	20	18	16	14	12	10	8	6	4	2
23	21	19	17	15	13	11	9	7	5	3	1

Wiring Definition Connector / (Board Signal) / Pin Connection			
JA -1 / No Connect /		JB -1 / No Connect /	
JA -2 / TP_FS_SEL /		JB -2 / FPGA_TP_7 /	Flying Lead
JA -3 / No Connect /		JB -3 / No Connect /	
JA -4 / Test Clock /		JB -4 / FPGA_TP_6 /	Flying Lead
JA -5 / No Connect /		JB -5 / No Connect /	
JA -6 / FPGA_DONE /	Flying Lead	JB -6 / CPLD_TP_0 /	Flying Lead
JA -7 / FPGA_M1 /	GND	JB -7 / No Connect /	
JA -8 / TP_DIS4_EN /		JB -8 / FPGA_TP_4 /	Flying Lead
JA -9 / GND /	GND + 3 Flying Leads	JB -9 / FPGA_M0 /	+3.3V
JA -10 / No Connect /		JB -10 / CPLD_TP_1 /	Flying Lead
JA -11 / RACE_TCK /		JB -11 / +3.3V /	+3.3V + 2 Flying Leads
JA -12 / TP_S0_ENABLE /		JB -12 / FPGA_TP_2 /	Flying Lead
JA -13 / RACE_TMS /		JB -13 / R1_CLKLOCK /	
JA -14 / USER_RESETN /	N.O. Push Button to GND	JB -14 / FPGA_TP_3 /	Flying Lead
JA -15 / CPLD_TMS /	Flying Lead	JB -15 / FPGA_TP_1 /	Flying Lead
JA -16 / RACE1_TDI /		JB -16 / FPGA_TP_0 /	Flying Lead
JA -17 / FPGA_TDO /	Flying Lead	JB -17 / FPGA_TP_5 /	Flying Lead
JA -18 / INT_RACE2_TDO /		JB -18 / CPLD_TP_0 /	Flying Lead
JA -19 / CPLD_TDI /	Flying Lead	JB -19 / No Connect /	
JA -20 / FPGA_TCK /	Flying Lead	JB -20 / CPLD_TP_2 /	Flying Lead
JA -21 / CPLD_TDO /	Flying Lead	JB -21 / +1.5V /	Flying Lead
JA -22 / CPLD_TCK /	Flying Lead	JB -22 / CPLD_TP_1 /	Flying Lead
JA -23 / FPGA_TMS /	Flying Lead	JB -23 / PROM_READY /	
JA -24 / FPGA_TDI /	Flying Lead	JB -24 / CLOCK_TEST_EN /	

Figure AII-57 Micro Accel Daughtercard Test Plug Adapter Cable Wiring Definition