RASSP Technology Insertion into the Synthetic Aperture Radar Image Processor Application

Junius Pridgen, Richard Jaffe, William Kline
Lockheed Martin Advanced Technology Laboratories
Camden, NJ 08102
jpridgen@atl.ge.com, rjaffe@atl.ge.com, wkline@atl.ge.com

Abstract

This paper describes the development on RASSP Benchmark 1 and 2 of the synthetic aperture radar (SAR) image processor using the RASSP design environment. The overall process flow developed by Lockheed Martin's Advanced Technology Laboratories, as applied on the SAR processor, is illustrated. Results from using executable specifications; parametric cost estimating tools and VHDL-based performance modeling for architecture tradeoffs; hardware/software codesign; virtual prototyping for architecture verification; software generated by autocode; and VHDL-based, top-down hardware development are shown. This paper discusses the implementation strategy and lessons learned on the RASSP benchmark activities.

1: Introduction

Lockheed Martin Advanced Technology Laboratories (ATL) designed, fabricated, and tested a real-time synthetic aperture radar (SAR) signal processor under the RASSP Benchmark Program. Lincoln Laboratory defined the functional, performance, physical constraints, and interface requirements for the SAR processor [1]. The SAR processor is to form images of the Earth surface in real time from an airborne platform (Amber UAV). The airborne usage places maximum size (10.5„ by 20.5„ by 17.5„), weight (60 pounds), and power (500 watts at 28 volts DC) requirements on the SAR processor hardware. The continuous formation of images for up to three polarizations while maintaining processor accuracy (error less than -103 dB relative to maximum output signal power) requires approximately 750 MFOPS of processing power and 80 MBytes of memory. The exact processing power and memory requirements are dependent upon the hardware/software implementation.

2: Architecture Design

The architecture design process was started by capturing and analyzing the requirements of the SAR signal processor. The SAR signal processor processes radar-pulse data from up to three of four polarizations (HH, HV, VH, and VV) to produce the output images. The computational operations performed by the SAR signal processor are:

- PRI detection - detection of the start of pulse data and the extraction of ancillary navigation and radar data
- Video-to-baseband I/Q Conversion - modulation of input samples by (-1)n followed by finite impulse response (FIR) filtering
- Range compression - vector multiplication, Discrete Fourier Transform (DFT), and vector multiplication
- Azimuth compression - DFT, vector multiplication, and inverse DFT.

The Ascent Logic RDD-100 tool was used to decompose the SAR processor requirements, assign the requirements, and create specifications to document the hardware, software, and firmware. Several candidate architectures were evaluated with respect to meeting current requirements, allowing for future upgrades, and minimization of cost and risk. The candidate architectures represent different combinations of COTS and custom components and are based upon several different processors, including the Intel i860, the Analog Devices ADSP 21060-2, and the Sharp LH9124. An architecture trade-off matrix was created comparing eight architectural candidates and served as the basis for making the architectural selection. Performance simulations were used to determine processor and interconnect utilization in each candidate architecture. Matlab was used to evaluate processing accuracy for different combinations of fixed-point and floating-point number representation. The Lockheed Martin PRICE Systems' parametric cost estimating tools were used to compute development, production, and life-cycle costs of the SAR processor. The selected SAR signal processor architecture, shown in Figure 1, contains four major architectural elements:

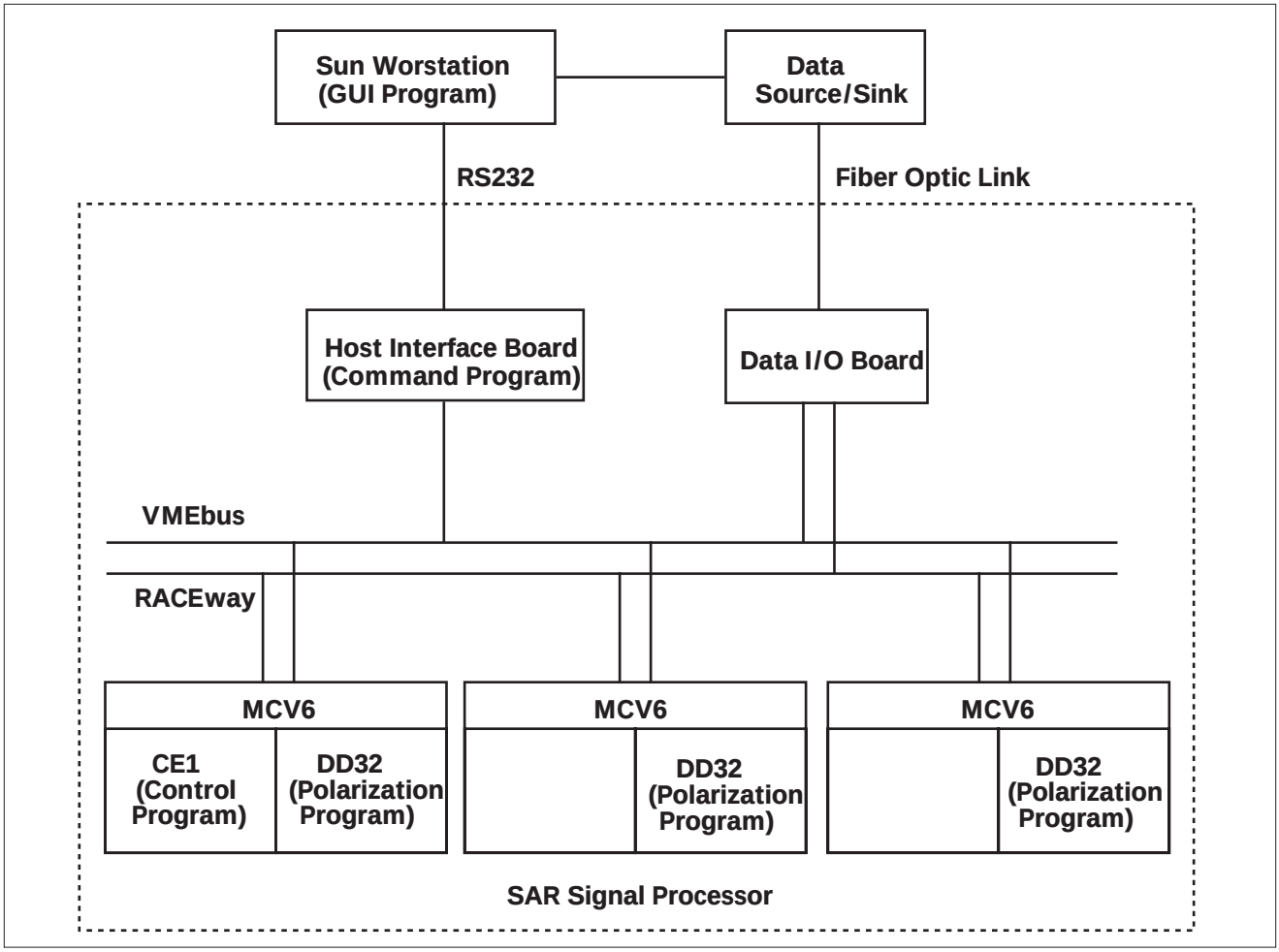

Figure 1. SAR architecture.

- Mercury Computer Systems MCV6 Processor Boards (VME 6U format) will perform the bulk of the signal processing. Each MCV6 can have up to two daughtercards. The SAR signal processor design uses a single daughtercard containing four Analog Devices ADSP21062 SHARC processor chips with 32 MBytes of DRAM for the processing of each polarization.

- A 68040 based single board computer (SBC), the Motorola MVME162, serves as the host interface and controls the SAR signal processor operation. When the SAR signal processor is in stand-alone mode, the 68040 boots the SAR signal processor and controls its operation.

- A custom designed Data I/O Board interfaces the SAR signal processor to the radar data source and sink, and performs front end signal processing functions. This includes synchronizing operation based upon the preamble sent with each pulse, performing video-to-baseband I/Q conversion, FIR filtering, and keeping track of pulse, polarization, and frame boundaries.

- An interconnect network consisting of Mercury Computer Systems Raceway acts as a high-bandwidth, point-to-point network for data transfer, and the VMEbus performs control operations.

The main risk with the chosen architecture was that the ADSP21062 SHARC processors were not yet available when the choice was made. Therefore, a risk mitigation plan was established to use dual i860 daughtercards for the signal processing if the SHARC based daughtercards were not available at the time of system integration. A single polarization design requires two i860 boards, each with two daughtercards, and a three polarization design requires five i860 boards, each with two daughtercards.

The above sizing estimates assume that the computational-intensive FIR filter operation is being performed in

dedicated FIR hardware on the Data I/O Board. The decision to include FIR filtering hardware on the Data I/O Board was made during the architecture tradeoff analysis phase of the design process. The FIR hardware reduces the computational requirements on the signal processors to the point where a single cluster of four SHARCs can be used per polarization being processed. Without modifications to any hardware, the number of FIR taps can be increased from the required 8 up to 64, which will more than meet the possible required future enhancement of 48 taps.

3: Virtual Prototype

The network performance of candidate architectures was modeled at the data-packet level to evaluate system sizing (number of processors), software mapping to processors alternatives, and performance of the interprocessor communication network. Five seconds of SAR time was simulated in 18 to 28 minutes, depending upon architecture and software mapping.

It is essential to select an appropriate abstraction level to achieve accurate, yet efficient, performance simulations that permits rapid exploration of a large number of alternative software mappings and hardware architectures. Simulation times of less than a half-hour allow investigation of multiple design options per day.

The next level of virtual prototype effort built upon the performance models by adding function to time and space. A data field added to the network token allowed actual data passing between the models, and the high level pseudo-code modeling of the software running on the processors used in the performance simulation was replaced with processors working at the DSP math library calls. The data I/O board and host I/F board were described in behavioral level VHDL code to model the function and interface to the MIT Lincoln Laboratory executable requirement testbench. Testbench models were created using scaled data sets for efficient debugging of the VHDL code. The generation of a full three-image data set took 14 hours of simulation time running Mentor's QuickVHDL on a Sun SPARC 10 with 256 MBytes of RAM. Images from the virtual prototype showed a processor accuracy of -127 dB. Over 4,600 lines of executable VHDL code were written to implement the virtual prototype.

The next level of VHDL modeling for the custom Data I/O board described the COTS components at the behavioral level with emphasis on interface behavior rather than internal chip structure, and the two new FPGAs in synthesizeable VHDL at the RTL level. Functional operation of the FPGAs and the Data I/O board was first debugged by using a VHDL test bench to exercised the Data I/O board and FPGAs through their various operation modes. Then final system level verification of the Data I/O board design was confirmed by operation of the detailed Data I/O board VHDL model in the virtual prototype of the SAR system. The model for the Data I/O board includes 2800 lines of executable VHDL code.

4: Software Design

Figure 1 identifies the software programs running on the processing elements and the major hardware elements.

- A graphical user interface (GUI) running on the Sun workstation accepts commands and displays status and data as requested by the user.
- A command program (CP) running on the Motorola host interface board performs initialization, BIT, and state control of the SAR signal processor.
- Polarization program running on the Mercury signal processor boards performs the image processing algorithms.
- Control program running on the Mercury signal processor board controls the operation of the polarization program and data movement to and from the Data I/O board.

The software layout on the processors was first prototyped using Processing Graph Method (PGM). The PGM graphs that correspond to the SAR application were developed, and the PGM graphs partitioned to correspond to the functions of the Data I/O board, control program, polarization program, and the command program. The interface data between all these functions that correspond to the software data dictionary was mapped to PGM graph variables (GV) and graph instantiation parameters (GIP) and queues.

Prior to integration with hardware, all of the software functionality was simulated in the Processing Graph Simulation Environment (PGSE). The Command Program was designed using Object-Oriented Analysis and Design Methodology (OOA/OOD). In this methodology the graphs, GVs, GIPs, and queues were handled as objects. Associated with each object was an action or state diagram that shows the interaction of these objects. Automatic code generation was used to create the Ada software using the OOA/OOD tool.

The interface between the CP and the polarization programs was developed using the PGM CP_Callable library as a guideline. This formally defined set of library functions was used to functionally decompose the interface software functional requirements among the software architectural elements. By using the same function specifications as the CP_Callable library for the application the CP program could be verified by simulation.

In order to control the polarization program, a control/graph manager program was developed to direct the data transfers between the Data I/O Board and the polarization processors using the RACEway. The VME interface is used to connect to the host processor, which sends messages to the graph manager as it executes the CP_Callable library functions. The VxWorks and Mercury Computer operating systems are used to obtain all the system functions needed to implement the message-passing and shared-memory functions required by the control and graph manager.

The polarization programs are executed on the parallel polarization processors. These programs consist of a set of tasks or threads that are supported by the Mercury Computer operating system. The tasks are *range*, *corner turning*, and *Azimuth,* which are implemented as software threads with different execution priorities. The coordination with the control program is handled by the status and command threads which have been integrated with the Azimuth computation thread.

5: Hardware Development

The hardware design for the SAR signal processor consisted of the mechanical design of the chassis and the design of the data I/O Board. All other boards in the SAR signal processor are COTS components which required no new hardware design. Hence, the hardware design discussion focuses on design of the Data I/O Board shown in block diagram form in Figure 2.

The Hot Rod module (HRC-500FS Fiber Optic Interface Card) provides separate fiber optic interfaces for the input radar data and output image data. The Hot Rod daughtercard interfaces to the rest of the Data I/O Board through two 40-bit wide buses — one for transmit and one for receive. The input data rate of 4.56 Megawords and output data rate of 6.83 Megawords are both within the 11.25 Megawords maximum of the Hot Rod. The Hot Rod has an internal loopback mode that connects the transmit side back to the receive side. The loopback mode is used during Data I/O Board test.

The Hot Rod Interface FPGA performs a number of operations on the radar data in addition to controlling the Hot Rod. The Hot Rod Interface FPGA contains logic that looks for the preamble that defines the beginning of a radar pulse. Once the start of a pulse is detected, the odd and even data samples are extracted along with bit serial data defining polarization and auxiliary radar data. The 12-bit odd and even pulse samples are modulated by (-1)n before being written into the Input FIFO. The auxiliary radar data is written to the AUX FIFO. The number of words, pulses, and image frames in both the receive and transmit channels are counted as part of the I/O control.

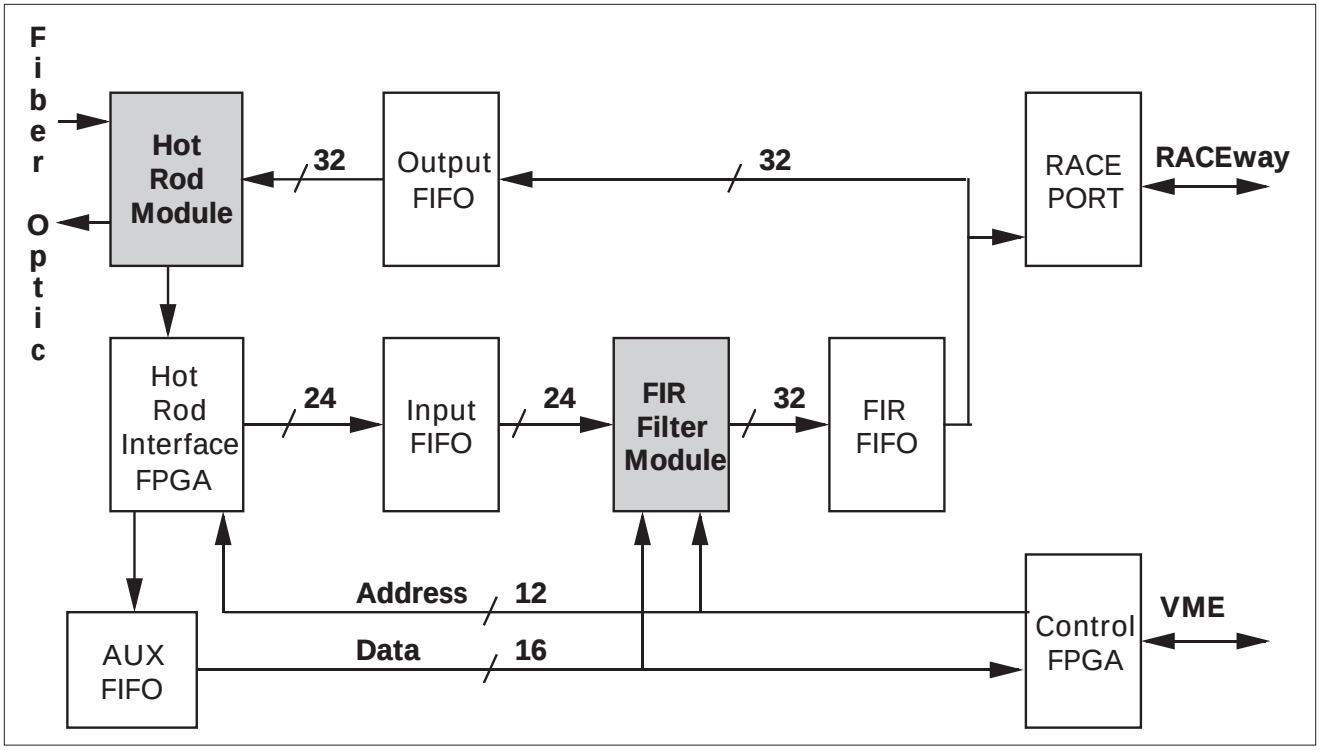

Figure 2. Data I/O board block diagram.

The FIR filter daughtercard is capable of simultaneously processing I and Q channel data at a 5 MHz input rate. The input data is 12-bit twos complement, the filter coefficients are 23-bit twos complement, and the output is 32-bit twos complement in both I and Q. Each channel uses two Plessey PDSP16256 Programmable FIR Filter chips configurable with up to 64 taps, with one FIR chip processing the most significant portion of the filter coefficients, and the other processing the less significant portion of the filter coefficients. The 16K deep FIR FIFO buffers received data until it is sent out of the RACE interface.

The RACE port interface contains two ASICs designed by Mercury Computer Systems, Inc. The RACE interface sends the received radar data to the appropriate signal processor, and receives output image data from the signal processors. The 4k-deep output FIFO provides data buffering between the RACE interface and the Hot Rod transmit port.

The Control FPGA controls the operation of the FIR filter and the FIFOs, keeps track of the status of all FIFOs, and implements the VMEbus interface.

Test considerations played an important part in the design of the Data I/O Board and associated FPGAs. One testability issue is the lack of JTAG scan on a majority of the COTS components. The design approach taken was to add JTAG scan bus transceivers to signal paths between non-JTAG devices. A second testability issue is the presence of asynchronous interfaces. The approach taken is to design these interfaces so that they can be tested synchronously, and to add test modes that force synchronous operation. Test modes included in the FPGA design enable bypass of either the PRI detection logic or the FIR Filter. FIR bypass is significant in that it allows testing of the Data I/O motherboard without the FIR Filter daughterboard in place. FPGA test modes reduce the cycle time of high modulo counters, allowing testing of higher order counter bits. The various test modes, which are part of the VHDL model, are included in the data I/O functional simulations.

Synopsys was used to synthesize logic for the two FPGAs using the AT&T ORCA cell library. NeoCAD was then used to map these cells to the ATT2C15 FPGA programmable logic cells (PLCs), place the cells, and route the interconnect between the cells. Static timing analysis (using NeoCAD) after mapping, placement, and routing was performed to identify any nets not meeting timing and speed specifications. In some cases, it was necessary to go back and modify the VHDL description to meet timing and speed specifications. A typical modification was the introduction of pipelining on paths not meeting speed. Once the FPGA design was completed, the logic and timing was back-annotated into the board level VHDL model and functionality was reverified.

Detailed schematics of the Data I/O Board design were captured using Mentor Graphics Design Architect. The schematics could not be finished until the FPGA design was completed with all pin assignments finalized. Once the schematics were finalized, placement and routing using Mentor Boardstation were performed for the Data I/O motherboard and FIR filter daughterboard. The board layout parasitics were back-annotated into the VHDL model to confirm that functional and timing specifications are met.

6: Integration and Test

When this paper was written, the Data I/O Board was just released to fabrication and the COTS hardware will be delivered shortly. The testing of the hardware and integration of hardware and software will be reported on at the conference.

7: Conclusion

The RASSP team has defined and demonstrated an approach for applying VHDL to model full computing systems that contain upwards of hundreds of processor elements. A central theme is the promotion of true hardware/software codesign through the independent specification of the software and the hardware, so as to support the rapid exploration of various software applications and mappings upon many architectural candidates. Reduction of virtual prototype simulation time to less than a half-hour for relatively complex applications, such as the SAR, allows investigation of multiple design options per day.

The team also demonstrated VHDL-based hierarchical design, with simulation and testbench data from higher levels of the virtual prototype used to verify RTL level descriptions of custom boards and FPGAs.

References

1. Zuerndorfer, B and G. A. Shaw, “SAR Processing for RASSP Application,,, Proceedings of 1st Annual RASSP Conference, Arlington, VA, 15-18 August 1994, pp 253 - 268.