Implementation of the RASSP SAR Benchmark on the Intel Paragon*

Curtis P. Brown
Richard A. Games
John J. Vaccaro

The MITRE Corporation,
202 Burlington Road, Bedford MA, 01730-1420

Abstract

A software design process for mapping real-time applications onto massively parallel processors is described. The design methodology incorporates a software test bench used to evaluate the level of real-time performance the processing nodes are capable of delivering. The final integration step maintains the simple test bench interfaces and reduces the complexity of integrating the components to satisfy the timing requirements of the overall application. The process is applied to implement the RASSP SAR benchmark on an Intel Paragon. The initial implementation uses 12 Paragon GP nodes for a single polarization. Under OSF/1 this 12 node configuration satisfies all the real-time requirements. Under SUNMOS, a streamlined high performance operating system available on the Paragon, the throughput improves significantly with sustained processor utilization approaching 40%. A projected implementation of the RASSP SAR benchmark on the Embedded Touchstone suggests that all three polarizations can be processed (including I/O) using 14 out of its 16 MP nodes.

1.0 Introduction

High performance computing can play a significant role in the military's evolving seamless design methodology to rapidly produce systems with reduced life-cycle costs by providing homogeneous communication and scalable computing frameworks. Two related programs underway at ARPA are focusing on providing efficient and effective computing solutions for the Department of Defense. The Rapid Prototyping of Application Specific Signal Processors (RASSP) program, sponsored by the Electronic Systems Technology Office (ESTO), has a goal of improving the process of specifying, designing, manufacturing, and supporting complex digital signal processors. On the other hand, the Computing Systems Technology Office (CSTO) has initiated the Embedded Computing program whose goal is to make available commercial scalable high performance computing technology to real-time embedded applications. For example, the CSTO/Honeywell Embedded Touchstone project is packaging the commercial Intel Paragon massively parallel processor (MPP). The implementation of the RASSP synthetic aperture radar (SAR) image formation benchmark on the Intel Paragon explores the boundary between these two ARPA programs.

After providing a brief introduction to the Intel Paragon and Embedded Touchstone, this paper focuses on a software design process for mapping real-time applications onto MPPs. The process is applied to the RASSP SAR benchmark to obtain an initial implementation that uses 12 Paragon GP nodes for a single polarization. A projected implementation of the RASSP SAR benchmark on the Embedded Touchstone suggests that all three polarizations can be processed (including I/O) using 14 out of its 16 MP nodes. Finally, real-time scalability for the SAR application is discussed.

* This work was supported by the United States Air Force Electronic Systems Center and performed under MITRE Mission Oriented Investigation and Experimentation (MOIE) Project 74260 of contract F19628-94-C-0001, managed by Rome Laboratory/OCTS. This work was supported in part by a grant of HPC time from the DOD HPC Major Shared Resource Center, Wright-Patterson Air Force Base, on an Intel Paragon™. We wish to thank the Honeywell Corporation, Space Systems, Clearwater, Florida, and Sandia National Laboratory for the use of their Paragons. Paragon is a trademark of the Intel Corporation. All other product names, trademarks, and registered trademarks are the property of their respective holders.

The authors would also like to acknowledge Mark Flanzbaum who provided high performance library calls and other assistance.

2.0 Intel Paragon and the Embedded Touchstone

The Intel Paragon is a multiple instruction multiple data (MIMD) distributed memory MPP with nodes used for either service, computing, or I/O. Service nodes primarily compile, link and start jobs. Compute nodes are typically not time-shared and are used exclusively to execute application code. Nodes dedicated to I/O may have interfaces to RAID, a parallel file system, Ethernet, or HiPPI. The current generation GP node has 32 Megabytes (MB) of DRAM and two 50 MHz i860XP microprocessors. The next generation MP node has three i860XPs and 64 MB of DRAM.

The Paragon is delivered with a commercial version of the OSF/1 operating system. OSF/1 is a full featured micro-kernel-based operating system that provides system services such as virtual memory, threads and inter-process communication, and networking support for ftp and NFS [Zajcew, et al., 1993]. OSF/1 consumes up to 12 Mbytes of memory including the micro-kernel, OSF/1 AD server and system buffer space. Under OSF/1, one of the i860XPs at each node is dedicated to message handling and is unavailable to the application. An alternative operating system called SUNMOS (approximately 250 kbytes) has been developed by Sandia National Laboratory and the University of New Mexico [Maccabe et al., 1993]. SUNMOS provides minimal services but delivers higher performance (lower latency and higher throughput) and provides the application programmer access to the second i860XP at each node. Neither operating system has a preemptible kernel, which is a requirement for a real-time operating system, creating particular challenges in applying the current Intel Paragon for embedded real-time applications.

The nodes are interconnected in a two-dimensional mesh topology with routers at each node employing dimension-order wormhole routing. The nodes have a direct memory access interface to the mesh through a single ported network interface chip. The mesh has a theoretical capacity of 200 MB/s with OSF/1 achieving a sustained communication throughput of 80 MB/s and SUNMOS sustaining 160 MB/s. Intel supports the NX application programming interface (API) for message passing, and a subset of the API is available under SUNMOS [McCurley, 1993]. Applications which adhere to the subset are source compatible between OSF/1 and SUNMOS.

ARPA/CSTO has sponsored the Honeywell Embedded Touchstone program [Blitzer, 1993] to develop an embeddable version of the Intel Paragon containing MP-nodes with three i860XPs per node. A requirement of this packaging effort is that the Embedded Touchstone will execute the exact same software as the corresponding commercial system. Honeywell is developing a 16 node prototype (4.8 Gflop/s) that will occupy .5 cu. ft. and consume 560 watts of average prime power. Honeywell is currently working in collaboration with the Open Software Foundation-Research Institute (OSF-RI) on a real-time operating system that will run on the Embedded Touchstone. The physical parameters of the Embedded Touchstone make it an attractive target architecture for the RASSP benchmark. Assessing the limitation due to the current non real-time operating system is one objective of this work.

3.0 Real-Time Parallel Software Design Methodology

All engineering design endeavors are facilitated by a clear specification of the system requirements. Real-time embedded systems are characterized by functional, timing and physical requirements. Military systems have additional requirements based on life-cycle costs which encompass maintainability, reliability, supportability, extensibility and scalability. This section describes a software design approach that utilizes a test bench implemented on the target architecture as a means of assuring functional and timing correctness. The methodology is described in more detail in [Brown, et al., 1995].

The process starts with clear specifications of the functional and timing requirements and a high-level model of the target architecture. The specifications and model are used to perform standard data and control flow analysis that serves to identify the major computation and communication kernels of the application. The functional analysis includes memory requirements and operation counts that combine with timing specifications to yield operation throughput requirements. The throughput requirements and a knowledge of the system's capabilities can be used to get a rough estimate of the number of compute nodes that will be required for each task. A real-time test bench consisting of a data source, the function under test, and a data sink is implemented on the MPP to establish the actual level of real-time performance the system is capable of delivering on each of the identified kernels. After functional correctness is established, perhaps using specified ground truth data, the test bench facilitates the process of tuning the kernel to meet the timing requirements. This may be accomplished by including

vendor supplied library functions or even customized assembly language functions. If the timing requirements at a node cannot be satisfied, then the algorithm partition needs to be reevaluated.

After timing requirements at the node level have been satisfied, the same test bench is used to exercise collections of nodes as they are integrated, culminating with a complete system test. Maintaining the test bench interfaces in the final integration step usually guarantees timing compliance when nodes are integrated. This requires the use of dedicated processing resources to implement the more complicated global communication requirements found in applications. These additional nodes are primarily dedicated to packing and unpacking the messages involved in these global communication patterns, e.g., the "distributed corner turn" in the current two-dimensional processing example. This means that the nodes doing the processing are relieved of this duty (these activities would alternatively take place in the application code running on the compute processor) and so can deliver more useful work. This in turn reduces the number of compute nodes required. The test bench also facilitates rapid node reconfiguration to allow swift system integration and to encourage experimental (perhaps ultimately adaptive) approaches to algorithmic partitioning and mapping for real-time embedded systems.

4.0 RASSP SAR Benchmark

The progress on the RASSP program is being evaluated with a series of benchmark exercises. The initial exercise is a synthetic aperture radar (SAR) image formation problem that has moderate computational throughput and memory requirements. The embedded requirements are consistent with real-time processing on board an unmanned air vehicle (UAV). The RASSP SAR benchmark is described in [Zuerndorfer and Shaw, 1994].

The benchmark is a stripmap SAR where the maximum pulse repetition frequency (PRF) of 556 Hz induces a minimum pulse-to-pulse period of 1.8 ms. One foot (0.3 m) resolution is obtained in both range and crossrange with a 375 m range swath nominally at a range of 7.26 km. The radar parameters and imaging geometry result in a benign SAR image formation process. Range migration less than one resolution cell results in separable range and crossrange processing.

The RASSP timing requirements derive from the 556 Hz maximum PRF producing a 0.921 second/frame specification. This results in a 1.1 Gflop/s requirement to process all three polarizations. Latency is required to be less than 3 seconds. There is also an accuracy requirement that can be met with single precision floating point. The embedded requirements include a 60 pound payload and 500 watts of average prime power. This implies a 2 Mflop/s/watt solution.

4.1 Real-Time Intel Paragon Implementation

This section describes the application of the parallel software design process discussed in Section 3 to implement the RASSP SAR benchmark on the Intel Paragon. The functional and timing specifications are clearly stated in [Zuerndorfer and Shaw, 1994]. An executable specification in the form of sequential C code running on a SPARC workstation was also provided, along with ground-truth test data. Analysis of the SAR image formation algorithm begins by establishing the computational requirements of the three main computational kernels (1) video to baseband conversion, (2) range compression, and (3) azimuth compression.

Although radar pulses arrive in sequence, it is more efficient to buffer the pulses and to process them in blocks. The operations required for video to baseband conversion increase linearly with the number of pulses processed in each block, denoted by N_P. The required operation rate of 36 Mflop/s (based on the maximum PRF requirement of 556 Hz.) is independent of N_P since the time available also scales linearly with number of pulses.

The operation rate for the range compression FFT (assuming a complex pre-multiplier and a real post-multiplier) is again invariant with block size N_P, and for a range FFT size of 2048, the required performance is 72 Mflop/s.

The overlap and save fast convolution implementation of the azimuth compression results in transforms whose length (1024) is twice the frame size (512). The resulting 241 Mflop/s throughput requirement is based on the 512 pulse frame period of 0.92 seconds. In order to make the azimuth processing as efficient as possible, a corner turn operation is required (to obtain contiguous samples in memory for fast convolution processing).

Several alternative parallelization techniques exist (e.g., data parallel, pipelining, and round robin) and trade-offs are often driven by throughput and latency requirements. For instance, pipelining will improve throughput, but each stage of the pipeline will add latency. Although the data

parallel approach is a low latency solution, it necessitates more complicated data flow.

The latency for the RASSP SAR benchmark is defined as the time between the arrival of the last pulse used to form the frame until the first corresponding image pixel is output. Latency is minimized by a front-end (video to baseband and range compression) solution that processes blocks with a small number N_P of pulses (fine grain), so that when the last pulse of the frame arrives, most of the other pulses for the frame have already been processed. Granularity studies suggest lower bounds on N_P motivated by the desire to maintain adequate levels of processor utilization [Brown, et al., 1995]. The non real-time operating systems periodically preempt the application, causing the sustained utilization of the processor to drop as the block size is decreased. In the current implementation, a blocksize (subframe) of $N_P = 64$ pulses was chosen; it is a factor of the frame size (512), and is well out onto the flat part of the performance curve.

The front end was implemented with a linear pipeline: the video to baseband and range compression functions each comprised a separate pipeline stage. With some assembly level optimization, the video to baseband conversion can be accomplished (just barely) on a single node. The range compression was further parallelized using a "data-parallel pipelined" technique. Each range compression node passes the entire subframe of data but operates on only a fraction of it. This approach, although arguably less efficient, was motivated by a desire to keep the front-end pipeline linear thereby simplifying the corner turn operation. A three-stage data parallel pipeline was required to meet the specification of 72 Mflop/s. The azimuth compression back end is necessarily a frame based (coarse grain) process that is naturally implemented as a data-parallel process partitioned over range subswaths. As expected, the azimuth compression required six nodes (a single node delivers approximately 45 Mflop/s.)

Two particular constraints drove the design of the corner turn software: (1) OSF/1 consumes 12 of the available 32 MB of DRAM on each GP node requiring a two node solution to support buffering (a single overlap extended frame is 16 MB), and (2) the fine grain front end must be transitioned into the coarse grain back end. Since the front end is a simple linear pipeline (the benefit of the "data-parallel pipeline") the corner turn is not "distributed" (i.e., the two corner turn nodes do not have to exchange data). Each corner turn node is responsible for half of the range swath. The last range compression node sends its complete output to both corner turn nodes which operate on the appropriate data subset.

The single polarization solution using 12 GP nodes is shown in Figure 4-1. This solution is mapped to the mesh by a loader in a run-time library that is linked to the application code. In the current implementation each node receives the same program image and a switch statement in the application code determines the function performed at each node based on the logical node number. This switch currently respects (left to right) the pipeline configuration shown in Figure 4-1, and the resulting mapping does not appear to have any communication contention problems. It is also possible to load different programs on the respective nodes, but this is usually avoided because of the increased complexity of the software configuration and maintenance.

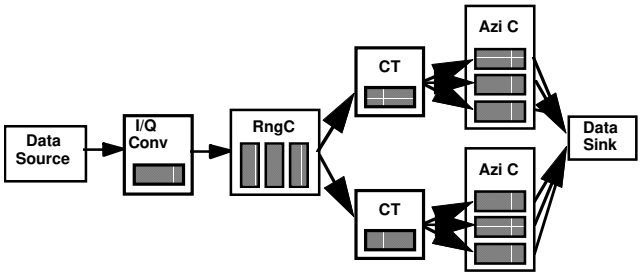

Figure 4-1. Single Polarization RASSP Benchmark Solution (GP Nodes)

Under OSF/1 the 12 node single polarization implementation achieves a worst-case throughput performance of 0.85 second/frame and a 1.16 second latency. Under SUNMOS the results improve to 0.71 second/frame throughput with a 1.02 second latency. These results are based on 15 minute runs to account for the non real-time behavior of the operating systems. The SUNMOS results correspond to end-to-end processing efficiencies approaching 40% of the theoretically available Mflop/s.

The implementation team had complementary skills and background. One staff member had a strong background in signal processing and SAR, but little experience with parallel machines. The other staff member had a strong background in programming parallel machines but no background in signal processing or SAR.

The overall effort took 9.5 staff months, although 4 staff months were devoted to background, the development of the test bench and other supporting infrastructure. The implementation and design iteration of the application code itself took 2.5 staff months, including .5 staff months dedicated to developing optimized assembly-level code.

Three staff months were spent testing and debugging. Debugging was more difficult than normal due to problems with system software on the Paragon and our use of remote machines. A more specific breakdown of activities is shown in Table 4-1.

Table 4.1. Software Development Level of Effort

Infrastructure: (4 staff-months)	
Background (ADTS/Paragon):	0.75 SM
Test-bench coding:	1.50 SM
Generating results:	0.25 SM
Support/analysis tools:	1.50 SM
Application: (5.5 staff-months)	
Preliminary design partitions:	0.5 SM
Initial functional coding:	0.5 SM
Design iteration:	1.5 SM
Testing and debugging:	3.0 SM

4.2 Embedded Touchstone Projection

The next generation (MP) nodes of the Paragon have three i860XP microprocessors. If all three i860s are available to the application programmer (supported under SUNMOS), it is estimated that the RASSP SAR benchmark requirement could be met with 4 MP nodes per polarization, with 12 MP nodes for all three polarizations. Input could be accommodated by interfacing the RASSP fiber input source to a HiPPI I/O node. An additional I/O node could be required for output. With this estimate, the functional, physical, and timing requirements of the RASSP SAR benchmark would be satisfied with the Embedded Touchstone prototype using 14 out of its 16 MP nodes.

4.3 Scalable Real-Time Processing

The issue of scalability for real-time parallel processing is discussed in [Brown, et al., 1994]. A SAR problem scales in two ways. The simplest way is to increase the range swath with the same PRF. This does not effect the resolution but simply increases the number of samples within each pulse. A more aggressive scaling involves an improvement in resolution (resolution is assumed to be equal in range and azimuth) for a fixed range swath. In the range dimension the number of samples per pulse increases inversely proportional to the resolution reduction. In the azimuth dimension the wider beamwidth required increases both the azimuthal correlation template and the PRF so that the crossrange computation rate also increases inversely with resolution. With this second scaling, computational requirements increase as the square of the resolution reduction.

The present front end (video to baseband conversion and range compression) limits the scalability of the current implementation. The "data-parallel pipelined" range compression quickly loses efficiency as the number of stages increases. A scalable solution requires a data-parallel front end combining both video to baseband conversion and range compression functions on a single node. With some further optimization we anticipate such a data-parallel front end would eliminate two out of the four GP nodes used in the current implementation. A data-parallel front end will also necessitate a distributed corner turn (i.e., the corner turn nodes have to gather data from the front end, format messages, exchange data between themselves, unpack messages, and then scatter data to the back end).

A notional scalable SAR architecture is shown in Figure 42. One feature of this architecture is that it dedicates a separate stage to the distributed corner turn. In this way its overhead and impact on real-time scalability can be explicitly determined. Preliminary assessments of the scalability of the data exchange phase of the corner turn is given in [Brown, et al., 1994]. Scalable corner turn solutions are required for both scalable signal processing implementations and their synthesis with data flow shell tools. This is the subject of future work.

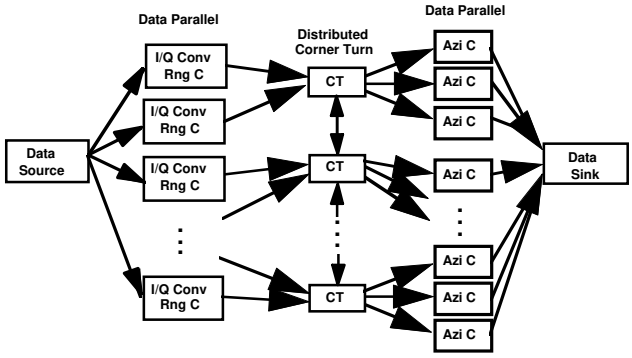

Figure 4-2. Scalable Real-Time SAR Processing

5.0 Conclusion

A parallel software design process for real-time embedded applications was applied to map the RASSP SAR

benchmark onto an Intel Paragon. A single polarization solution was demonstrated with 12 (GP) nodes, and projections to the Embedded Touchstone suggest that a 14 (MP) node system will support processing of all three polarizations including I/O. These solutions use the current non real-time OSF/1 operating system but still provide worst-case timing guarantees that meet the RASSP real-time constraints. Fully taking advantage of the higher performance SUNMOS operating system would reduce this node count, as would presumably a future real-time version of OSF/1. Future work will focus on the area of real-time scalability. A scalable corner turn implementation is critical for truly scalable signal processing and to facilitate application of software synthesis tools.

References

Brown, C. P., M. I. Flanzbaum, R. A. Games, and J. D. Ramsdell, 1994, *Real-Time Embedded High Performance Computing: Application Benchmarks,* MTR 94B145, The MITRE Corporation, Bedford, MA.

Brown, C. P., R. A. Games, and J. J. Vaccaro, 1995, *Real-Time Parallel Software Design Case Study: Implementation of the RASSP SAR Benchmark on the Intel Paragon*, MTR 95B*TBD*, The MITRE Corporation, Bedford, MA, *to appear*.

Blitzer, F., 1993, "Militarized Touchstone Program," *Proceedings of the 1993 IEEE National Aerospace and Electronics Conference*, Vol. 1, Dayton, OH, pp. 137–143.

Maccabe, B., K. S. McCurley, and R. Rissen, November 1993, *SUNMOS for Intel Paragon: A Brief User Guide*, ftp://ftp.cs.sandia/pub/sunmos/papers/ISUG94-1.ps.Z, Sandia National Laboratories, Albuquerque, NM.

McCurley, K. S., November 1993, *Intel NX Compatibility under SUNMOS*, Technical Report No. SAND 93-2618, Sandia National Laboratories, Albuquerque, NM.

Zajcew, R., P. Roy, D. Black, C. Peak, P. Guedes, B. Kemp, J. LoVerso, M. Leibensperger, M. Barnett, F. Rabii, D. Netterwala, January 1993, "An OSF/1 Unix for Massively Parallel Multicomputers," *Proceedings of the 1993 Winter USENIX Conference*, San Diego, CA, pp. 449–468.

Zuerndoerfer, B., and G. A. Shaw, 1994, "SAR Processing for RASSP Application," *Proceedings of the 1st Annual RASSP Conference*, ARPA, pp. 253–268.