Adapting Parallel Backprojection to an FPGA Enhanced Distributed Computing Environment

Albert A. Conti, Ben Cordes, Miriam Leeser, Eric Miller
{aconti, bcordes, mel, elmiller}@ece.neu.edu
Dept. Electrical and Computer Engineering
Northeastern University, Boston, MA

Richard Linderman
Richard.Linderman@rl.af.mil
AFRL/IF, Rome, NY

Introduction

Synthetic Aperture Radar (SAR) is a widely used method for generating radar images in applications ranging from remote sensing of the environment to target identification. Unfortunately, SAR generates a large amount of data for image reconstruction, and processing speeds on conventional computers are typically slow. Backprojection is a highly parallel algorithm for SAR reconstruction that maps well to distributed computing environments and reconfigurable logic. We have developed an implementation of backprojection for SAR on a Heterogeneous High Performance Cluster (HHPC). Our implementation uses multiple nodes of the cluster and the COTS FPGA boards available at each node. The resulting implementation runs a factor of 26 times faster than our efficient serial implementation and can reconstruct images from SAR data in a matter of seconds.

The HHPC used in our system is part of the Distributed Center at the Air Force Research Labs in Rome, NY. It is a two chassis, 48 node heterogeneous HPC from HPTi, which marries the benefits of Beowulf cluster computing with the reconfigurability of FPGAs. Each node is a Linux box with dual Xeon processors and an Annapolis Wildstar II FPGA board with two Virtex6000 FPGAs. The nodes are interconnected with Myrinet 64-bit PCI cards that provide a maximum bisectional bandwidth of 12 GB/s. The system is designed for a throughput of 422 GFLOPS from the Beowulf cluster, and 34 FIR Tera OPS from the FPGAs.

Implementation

The backprojection algorithm is well suited for parallelism in term of both the input (radar projections) and the output (pixels in the image). After an analysis of the processing and file I/O requirements of the algorithm and an experimental examination of the HHPC's file system and communication capabilities, we decided on the optimal extent of parallelism at the software and hardware levels.

At the software level, the target image is split into equal partitions, each of which is processed by the hardware at a single node. Projection data are distributed such that only time samples that contribute to pixels within each node's respective partition are loaded into the memory of that node. Once all the input data have been loaded in the memory of each node, projections are sequentially uploaded to the FPGA boards. The hardware iterates through the projection data and then downloads its portion of the target area to the local host PC.

The MPICH parallel computation library is used to control the multiple nodes of the HHPC. MPICH implicitly links in the GM libraries, which enable the use of the Myrinet interconnect to pass data between nodes. Our system runs on a single master node that is responsible for file I/O and controlling the computation on each of the slave nodes. Once the slave nodes have downloaded result images from the FPGA board, the data are collected by the master node, merged, and written out to a file. This accumulation and output stage takes 40% of the total system run time. Later versions of the MPI library, which were unavailable to us on the HHPC, support features such as parallel file I/O that could be used to improve these bottlenecks.

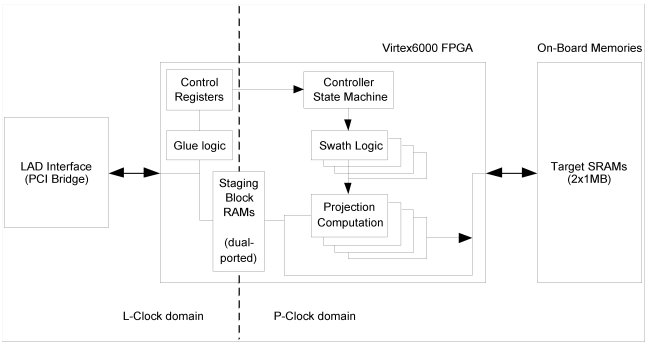

Figure 1. Block Diagram of Hardware On FPGA Device

The hardware on the FPGA board (see Figure 1) includes memories to hold a portion of the target image plus all the projection data needed to calculate the new image. Glue logic connects the on-board PCI interface to the logic on the chip, and a series of control registers handle commands and status information. A monolithic state machine reads the control registers and sequences the flow of data through the computation units. Swath logic (see below) computes a time index into the projection data based on the pixel in the target image that is currently being considered. Intermediate target images are read from the on-board SRAM, accumulated with the data from the projections that are currently in memory, and written back to another SRAM. The SRAMs switch read/write roles at each processing step, resulting in a "ping-pong" effect.

Each pixel in the final SAR image is a coherent superposition of the contributions from every radar return in a given data set. Determining which value in a given return contributes to a specific pixel is carried out via a mapping function. This function reflects the round trip travel time of a pulse of energy back and forth between the radar (at a known position along its flight path) and the location on the ground corresponding to the pixel. In our system, this mapping of pixel value to temporal index into a radar

return array is carried out by the processing unit labeled "Swath Logic."

Optimization

We explored several different avenues of parallelization to improve speedup. Since the contribution of a projection to the target image is independent of all the other projections, we can pipeline multiple projection computation units together to increase performance. Adding this functionality provides a 40% runtime performance gain, but an overall system performance gain of only 10% over a system that processes a single projection at a time. The performance boost is small due to the dominance of file I/O.

Similar effects can be achieved by processing multiple pixels simultaneously; more memory bandwidth is required, but our current design does not saturate the available memory ports. Both of these avenues of parallelization quickly reach diminishing returns, since as the level of parallelization increases, the portion of the overall run time that is due to this computation becomes smaller and smaller.

The pixel-to-time mapping function can either be computed or read out of a table. Reading from a table is slightly faster but requires more coefficients to be uploaded to the FPGA and more on-chip memories to be used. Moving from table lookup to computation logic did not noticeably affect performance; see Figure 2, where the "1K Swath SRAM" and "1K Swath Logic" lines are nearly overlapping. However, using logic instead of a table greatly eased the implementation of the multiple-projection optimization above.

The number of samples of projection data that are uploaded correlates directly to the resolution of the final image. Also, as the number of samples increases, the hardware solution shows better performance when compared to the software implementation. Our tests showed that the resolution does not significantly improve after 4k samples per projection, and the corresponding increased coefficient upload times are not a good tradeoff.

Results

Figure 2 shows a runtime performance comparison of multiple parallel hybrid solutions. The results are normalized to the runtime of an efficient serial software solution processing the same amount of data, which is listed as the point (0,1) for all four series of data. The fastest and most scalable solution computes the swath array in logic and processes four projections in parallel. The falloff in performance gains as the number of processor nodes increases is primarily due to the dominance of file I/O on the run time. This necessary data transfer to and from disk is the main barrier to further acceleration with the current HHPC platform. Improved file I/O of the cluster would result in better performance.

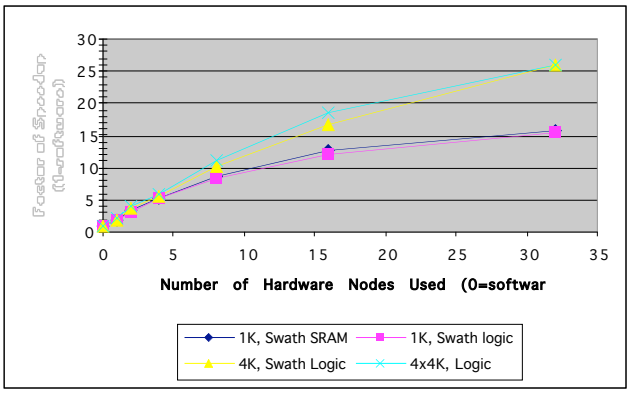

Figure 2. System Performance Across Several Implementations of the Parallel Solution

Acknowledgement

This publication made possible through support provided by DoD High Performance Computing Modernization Program (HPCMP) Programming Environment and Training (PET) activities through Mississippi State University under the terms of Contract No. N62306-01-D-7110.

References

[1] M. Soumekh, "Synthetic Aperture Radar Signal Processing with MATLAB Algorithms". John Wiley and Sons, New York, 1999. ISBN 0-471-29706-2

[2] S. Coric, M. Leeser, E. Miller, and M. Trepanier, "Parallel-Beam Backprojection: an FPGA Implementation Optimized for Medical Imaging". *Tenth ACM International Symposium on Field-Programmable Gate Arrays*, February, 2002.