Scalable Network Based FPGA Accelerators for an Automatic Target Recognition Application

Ruth Sivilotti, Young Cho, Wen-King Su, Danny Cohen and Brian Bray*
Myricom, Inc. http://www.myri.com
*Sandia National Laboratories

Extended Abstract

Image processing, specifically Automatic Target Recognition (ATR) in Synthetic Aperture Radar (SAR) imagery, is an application area that can require tremendous processing throughput. In this application, data comes from high bandwidth sensors, where the processing is time-critical. There is limited space and power for processing the data in the sensor platforms or in battlefield groundstations. DoD's strong push for using commercial-off-the-shelf (COTS) technology, the very high non-recurring engineering (NRE) costs for low volume ASICs, and evolving algorithms limit the feasibility of using custom special purpose hardware. In addition, a scalable system is required as the different sensor platforms have different image pixel rates and different mission requirements have different target recognition throughput needs per pixel.

In this paper, we will describe an ATR algorithm implementation using FPGA accelerators. We will first describe the ATR algorithm that was implemented, the implementation on a single FPGA, how the FPGA nodes are connected to make a scalable system, and compare the performance to current scalable microprocessor-based implementations.

Sandia National Laboratories' real-time SAR ATR systems use a hierarchy of algorithms to reduce the processing demands for the processing of images, while yielding a high probability of detection (Pd) and a low false alarm rate (FAR). The first algorithm step is a Focus of Attention (FOA) algorithm that runs over a down-sampled version of the entire image to find regions of interest that are of approximately the right size and brightness. The regions of interest are then extracted and processed by the Indexing stage which further reduces the data stream, target hypothesis, orientation estimations, and target center locations. The surviving hypotheses have the full resolution data sent to an identification executive that schedules multiple identification algorithms and then fuses the results of the multiple identification algorithms.

The algorithm that we have implemented using FPGA accelerators is an indexing algorithm called Second-Level Detection (SLD). It is used for finding targets in-the-clear, not for camouflage, concealment or deception (CC&D) scenarios. The SLD task is to take the extracted imagery (an image chip), match it against the list of provided target hypotheses and return the hit information for each target hypothesis which consists of the top two angle matches (if any) and the corresponding pixel location. The image chip is a 64x64 region of byte data and has a 21x21 search region. SLD is basically a binary silhouette matcher that has a bright mask and a surround mask that are mutually exclusive. The bright mask and surround reside in a 32x32 region, each are valid for approximately 5-10% of the region. The first step of the computation is called the shapesum, and it consists of adaptively estimating the illumination (energy under the bright mask) for each pixel location assuming that it is the target at that orientation and location. If the energy is too little or too much then no further processing for that pixel location for that template match is required. The next step is the threshold calculation which is to determine what is really a bright pixel and a surround (or dark) pixel. This consists of dividing the shapesum by the number of pixels in the bright mask and subtracting a template specific constant ("bias"). The pixels under the bright mask that are greater than or equal to the threshold are counted, if this count exceeds the minimum bright pixel count threshold, the processing continues. Now the pixels under the surround mask that are less than the threshold are counted. If this count exceeds the minimum surround pixel count threshold there is a hit. The quality of the hit is the average of the percent of bright and surround pixels that were correct. A 200MHz 603ev PowerPC microprocessor with a 256KB L2 cache running optimized code for this algorithm with the VxWorks operating system can achieve approximately 700 template matches per second (varies based on templates and data).

Sandia's current ATR systems are based on heterogeneous two-level multicomputers, microprocessors and DSP chips that are linked by Myrinet in a system area network (SAN) configuration. Myrinet is a scalable high bandwidth network based on intelligent network interfaces and nonblocking crossbar switches. The first level of processing is the network interface, and the second level of processing has been microprocessors and DSPs, and now includes FPGAs. Instead of the microprocessor-based indexing executive sending image chips and the corresponding SLD matching tasks to microprocessors via Myrinet, an indexing executive could send SLD matching tasks via Myrinet to the FPGA nodes.

The FPGA node consists of an ORCA-40 for computation, a Myrinet network interface (LANai 4.3 and SRAM) and a small FPGA with some flash memory for booting. The computation FPGA has no memory directly attached to it; the computation FPGA must access the memory for template data and image data from the LANai. The LANai has 512KB of memory attached which is used for network message data, program space, template storage, and image data. A single template requires 270 bytes, and there are 72 templates for each target configuration. The initial implementation supports 6 target configurations per node. Additional target configurations are supported by spreading out the templates amongst the processing nodes.

The correlations are mapped to a linear systolic pipeline. A high degree of parallelism is exploited. In addition to computing an entire row of correlation results in parallel (21 pixel locations), the FPGA computes the address calculations, data loading, and correlations in parallel. Short inter-register paths allow the design to run at 40MHz, which is limited by the clock rate at which external memory can be fetched. In the microprocessor implementation there are three phases of the computation, thus there is plenty of opportunity for reconfiguration. The first stage consists of the accumulation of 8-bit data into a 16 bit accumulator, the second and third stages consist of comparing 8-bit values and incrementing a counter. In the FPGA implementation, the second and third stage were optimized to be performed in parallel. Instead of reconfiguration for each stage, there is clever design of the processing element so no reconfiguration is necessary, and hence no time is spent on reconfiguration instead of computing. The compact processing element consists of one 8-bit data input register, one 8-bit accumulator, one 8-bit adder/subtracter, and two 8-bit counters. Twenty one of these processing elements along with the address generator reside on the FPGA while the divide operation is concurrently done in software in the LANai. There are 1024 (32*32) pixel locations in a template with 21 lines and two processing steps and some overhead (32*32*21*2 + ~5K overhead @40MHz) for a rate of 833 template matches per second. This was simulated and measured performance is slightly better than the baseline 200MHz PowerPC microprocessor.

Additional performance has been achieved by several additional optimizations. These optimizations are similar to the ones that a microprocessors uses: exploiting the sparseness of the templates and the stopping of the computation for a given test as soon as a processing stage fails (early outs). Since the correlation are in a linear pipeline we can not exploit all the sparseness in a given row, however it is easy to eliminate calculation for that row if the template is entirely empty for that row, and hence track the actual size of each template. To maximize this possibility, tall narrow templates are transposed into short wide ones and rotated image data is used for correlation. Another optimization is for early outs on an entire row where the entire row does not pass the shapesum test. After these optimizations a FPGA node processes 1300-1600 template matches per second. This variation is due to the number of all empty rows that exist in a given template and the number of early outs. On average with these optimizations, a FPGA node has greater than 2X the performance over the embeddable high performance PowerPC microprocessor.

Currently the bottleneck is the divide which is still being done in software in the LANai. We are currently moving the divide into the FPGA and we are expecting approximately 2000 template matches/sec, which is almost 3X over a high performance microprocessor implementation.

Four FPGA nodes easily fit on a single 6U VME board, yielding an impressive ~6000 template matches per second (8000 is expected with the divide performed in the FPGA). Currently, this is greater than 8X compute density for a single PowerPC node or 2X an expensive quad PowerPC board. Performance could be further increased if one were to reimplement the design with a larger FPGA even without adding more external memory bandwidth. There is at least a factor of 21 of unexploited parallelism since rows are processed sequentially. Most of the memory fetches are common to the next row of computation, however by computing multiple rows at the same time, the chance that the shapesum early out will occur decreases. Doubling the hardware on-chip might give close to 2X performance improvement, but 21X hardware increase will give only around 8X performance increase unless memory bandwidth is increased.

Conclusion

In this paper we have described not just a single accelerator node FPGA design, but a scalable FPGA system. This system is scalable by using an embeddable high performance networking technology (Myrinet) that has a smart network interface. This network interface contains a microprocessor core that is user programmable, so the network interface can be used for local control of the FPGA accelerators. This is an excellent example of two-level multicomputing where the second-level (the FPGA) can contain no control capability, and it is entirely dependent on the first-level (the smart network interface). We have demonstrated the scalable FPGA system for an automatic target recognition application. For that application and a comparable number of VME boards, we have an achieved a 8X speed-up over a high performance single board computer, and a 2X speed-up over a high performance quad multicomputer, while using a moderate-sized FPGA device, ORCA-40. We expect further planned design enhancements to increase the computational density advantage by another 33%. Even further performance advantages can be had by exploiting the remaining parallelism the design provides by using currently available, larger FPGA devices.

To achieve high performance there were eight main themes: (1) use a proper system architecture (scalable); (2) have a proper decomposition between hardware and software, do not try to execute in FPGAs the complex but not computationally intensive parts of modules that are best left to software in host microprocessors; (3) use dense packaging to achieve high performance for a given volume, microprocessors typically have a non-trivial amount of support chips; (4) have short inter-register paths for a fast clock so achievements in parallelism are not all lost due to microprocessor clock rate advantages; (5) have a design that can take advantage of high degrees of parallelism; (6) have a design that can adapt to the dynamic variances in the data to eliminate excess computation, just like in microprocessor software; (7) do not neglect start and finish overheads; and (8) minimize reconfiguration since most current FPGA devices reconfigure too slowly.