Accelerating An IR Automatic Target Recognition Application with FPGAs

Jack Jean, Xuejun Liang, Brian Drozd, and Karen Tomko
Department of Computer Scicence and Engineering
Wright State University, Dayton, OH 45435, USA
{jjean, xliang, bdrozd, ktomko}@cs.wright.edu

An infrared automatic target recognition (IR ATR) application is accelerated with an FPGA co-processor board. The board features and the application are first stated. The FPGA design is then described. The achieved performance is reported and analyzed at the end.

The FPGA co-processor board used is a Giga Operations's G900 board which is PCI-bus based and hosted on a 180 MHz Pentium PC. On the board there are 8 XMODs, each containing 2 XC4020 FPGA chips, 256 KBytes of SRAM, and 2 MBytes of DRAM. It is possible to broadcast or multicast an FPGA configuration file to those XC4020s in 0.37 seconds.

The application of the IR ATR program is to locate and identify ground vehicles based on a single IR image frame. The G900 board is used to accelerate the bottleneck of the program which is to locate the region of interests (ROIs) by applying a set of templates through the whole image. Because the computation involved is different from a previous work on mapping ATR to FPGA [1], we were not able to adopt techniques used in that work.

To locate the ROIs six template pairs are applied through the whole image except the image boundary. Each template pair contains one target template and one for background template. Each template contains a pattern of exactly 30 pixels in a relatively large area, say, of size 20 by 50. (Different templates have different sizes.) Each image is partitioned into five strips of different heights and each template needs to be rescaled across strip boundary.

The surrounding area of an image pixel is tested against a template pair to see if it is an ROI. The computation involved is as follows.

1. Compute the mean of those 30 test points associated with the background template.
2. Test points whose values are larger than the mean are considered "hot" (otherwise "cold"). Compute four different sums, one for target hot values, one for target cold, one for background hot, and one for background cold.
3. Based on the four sums compute the correlation.
4. If the correlation is greater than or equal to a constant threshold, the pixel is a ROI.

The FPGA design has the following features.

1. All the floating point computations have been replaced with full-precision integer operations.
2. The original code uses one byte per image pixel. The FPGA used 5 bits per pixel. The recognition results were considered comparable judging from the image output files and the no. of ROIs.
3. Six XC4020 chips are used, one per template pair.
4. Each template pair is evaluated on two neighboring pixels in parallel.
5. The original C code uses several 2-D and 4-D arrays which require many array index computations. Those computations are removed with the FPGA design.

Only the locating of ROIs is implemented in the FPGA design. A host program running on the PC does everything else as required in the original source code. To overlap the FPGA computation and the host machine computation, the host program uses two threads. When the child thread is waiting for the FPGA signal of computation completion for one strip, the parent

thread can go ahead and process ROIs identified for the previous strips.

The FPGA design in an XC4020 contains several components:

1. An SRAM controller: it has four functions. (a) It receives contiguous image data from the host program and stores them in the SRAM on the XMOD. Note that data are stored redundantly as shown at the right so that two neighboring pixels can be evaluated in parallel. (b) For each single pixel location, it allows the (almost random) access of 60 test points. (c) Store the pixel location into SRAM if the location is an ROI. (d) Allow the host program to read back all the ROIs.
2. A buffer: it stores the locations of those 60 test points associated with a template pair.
3. A computation unit: it contains three parts: (a) two TEMPERATURE units, each containing an accumulator to compute the sum of 30 background image values, a buffer to store those 30 values, comparators and adders to compute HOT and COLD values.) (b) a multiplexer and a small unit called CONVERT, and (c) an ASSERT unit to evaluate if the correlation is greater than or equal to a threshold.

0	1
1	2
2	3
…	…
100	101
101	102
…	…

Memory data arrangement: Each memory address is for two pixels.

The ASSERT unit is time-multiplexed to process the outputs of the two TEMPERATURE units. This arrangement saves one CONVERT unit and one ASSERT unit at the expense of requiring the multiplexer. In addition, a 33 MHz clock is used for the ASSERT unit while the rest of the chip runs with a 16 MHz clock. The whole Round 0 design takes 704 CLBs per chip which is 89.8% of the total CLBs on a XC4020 chip.

The execution times for the computation of ROIs are summarized in Table 1. Note that the execution time with FPGA does not include the time to initialize the board (2.31 seconds) and the time to load the bit file (0.37 seconds). On G900, the execution time contains the following three parts that take 0.655 (= 0.075 + 0.018 + 0.562) seconds.

1. *Image data transfer from PC to XMODs*: about 0.075 seconds. The data transfer bandwidth is about 6MB/sec for write and 4MB/sec for read from G900.
2. *Other data transfer*: about 0.018 seconds. (1) The sending of templates to XMODs: less than 10 Kbyes (2) The reading of results from XMODs: Assume that the number of pixel locations that are ROIs is less than 10%. Since each location takes 2 bytes. There are about 60 (= 300 x 0.1 x 2) KBytes to read.
3. *Real computation*: The computation time is mostly overlapped with the reading of image data from SRAM to FPGA chips. For SRAM, it takes one clock cycle to read 2 bytes. For an image frame of 300 K pixels, each pixel requires the reading of 60 bytes. So it takes 9 M (= 60 x 300 K / 2) clock cycles, which takes 0.562 seconds with a 16 MHz clock.

Therefore the timing results obtained with the FPGA board are consistent with the above timing analysis.

Acknowledgments

This research is supported by DARPA under Air Force contract number F33615-97-1-1148, an Ohio State investment fund, and an Ohio State research challenge grant. Xilinx Inc. donated an FPGA design tool and FPGA chips on XMODs. The authors would like to thank Mr. Jim Hilger with the US Army Night Vision & Electronics Sensors Directorate for his introduction to the IR ATR application and his help and comments on the design.

Reference

[1] J. Villasenor et al., "Configurable Computing Solutions for Automatic Target Recognition," in IEEE Symposium on FCCM, pp. 70--79, 1996.

Image No.	*Time On PC*	*Time With FPGA*	*Speedup*
1	13.106	0.602	21.2
2	12.980	0.614	21.1
3	13.187	0.622	21.2
4	13.016	0.617	21.1

Table 1 : Execution times in second