

A Software Methodology for Real Time Target Recognition

Wim Bohm

Colorado State University

and

Steve Heistand, David Caliga and Jeff Hammes

SRC Computers, Inc.

Abstract

This paper describes a software methodology for creating high performance FPGA configurations for a significant Automatic Target Recognition Application, Probing. The goal of the particular application under study is to recognize the SRC logo at varying distances and angles at video frame rate, which cannot be achieved using a microprocessor. Probes are pairs of points straddling an edge of a target. Probesets define a silhouette. Using a Perl script, probesets can first be viewed, scaled and rotated, and later be translated into C. Using the MAP C compiler, which is a component of the SRC Carte™ Programming Environment, the resulting C program is translated into an FPGA hardware configuration code. The code executing on the FPGA hardware runs at video camera frame rate. This is an impressive speedup compared to a C implementation of the same algorithm executing on an Intel P4 Xeon® microprocessor.

Introduction: SRC MAP and MAP C

In a SRC MAPstation™ the CPU is connected to a Compact MAP processor, Figure 1, via a high-speed memory interface. The MAP processor contains two FPGAs that are accessible to the user, and a control FPGA that handles DMAs between the six on-board memories and the CPU's memory. Each memory can be accessed simultaneously by the control FPGA and one of the two User FPGAs. The whole system is programmed in C or Fortran. The user controls the partitioning of a code between the CPU and the MAP by putting a MAP-targeted routine in its own source file and compiling it to hardware configurations with the MAP C compiler.

Probing

An interesting test case for the MAP processor is an Automatic Target Recognition (ATR) application. Probing is an ATR technique suited to recognize rigid objects. A simpler version of the algorithm presented here was used as part of an ATR system developed to recognize particular vehicles at various angles but at a fixed distance, developed at Colorado State University [1,2]. In this case we want to recognize varying sized logos at varying distances. Figure 2 shows the image with a logo (with the crosshairs) and two decoys.

A **probe** is a pair of input image pixels and an associated boolean result. Typically, a probe returns true when the absolute value of the difference between pixel values exceeds a threshold, answering the question: "Is one pixel on one side of an edge of interest of an object and the other on the other side?" A **probeset** is a set of probes arranged along the significant edges of an object, defining its silhouette as well as its internal structure. When the proper probeset is placed over an object of interest, one expects a high percentage of the probes to return true. If the probesets are sufficiently detailed and the objects sufficiently distinct, then no other object will score as high.

In our case we want to recognize the SRC logo at a range of distances and angles in real time, i.e. at the frame rate of a video camera. Figure 3 shows a logo and a fitting probeset generated from the original by the Perl script. The irregular alignment of the probes is caused by a number of factors, including compensation for camera characteristics, and the integer math performed on the probeset pairs.

Probesets can be viewed and transformed (scaled or rotated) using a Perl script. The Perl script also compiles the probesets into C code that performs the necessary point accesses, subtractions and comparisons. By compiling the generated C code with the MAP C compiler, the probesets are hardwired into FPGA configurations.

Recognizing objects of varying sizes and angles

The "initial" probeset has 50 probes and fits a logo in a 40x40 pixel window of the 640x480 pixel gray scale input image. The distance between the probe points is about six pixels in the initial probeset. This allows objects of slightly varying size to be recognized. In order to cope with objects that vary much more in size two techniques are employed. First to recognize smaller objects, three additional probesets are created from the initial one, each fitting a window of about 5/6 the size of the previous window. This results in windows of 40x40, 33x33, 28x28, and 24x24, and allows logos of all sizes from 24x24 to 40x40 to be recognized. Secondly, to recognize larger logos the input image is downsampled two, four, and eight times. The combination of the two techniques provides a range of 24x24 to 320x320.

To recognize logos at various angles the four probesets are rotated two times to the right and two times to the left. This allows logos at angles from -30 degrees to +30 degrees to be recognized. We now have twenty probesets, each with

50 probes. The goal is to perform the calculations for all 1,000 probes in parallel on the FPGAs.

Probing Performance and Analysis

A pseudo code version of the Probing algorithm is given below.

```

for downsampling 1, 2, 4 and 8 {
  for each 40x40 pixel window in image {
    for all probe_sets {
      score = 0;
      for each probe in probe_set {
        if hit(probe) score++;
      }
      winning_score_in_window = maximum_score;
    }
    gather winning scores > threshold
  }
  suppress overlapping winning scores;
  return image with winning scores marked
}

```

An exhaustive search through the probesets for all possible sizes and angles in all possible placements in an image is a costly computation. In a 680x480 image, 408,000 (640x480 + 320x240 + 160x120 + 80x60) placements need to be checked (actually, we overshoot by 40 pixels to detect targets right on the edge) with 1,000 (20x50) probes at each placement, totaling 408 million probes. Each probe requires six operations, three memory accesses (two pixels, one probe threshold), a subtract, a compare, and an add. This comes to 2.45 Giga operations per image, just for the inner loop of the program.

A C implementation of the algorithm on a 2.8 GHz Intel P4 Xeon microprocessor with 512 KB cache and 4 GB memory runs slightly over 3 seconds, two orders of magnitude slower than the required frame rate (30 frames per second). Whereas the microprocessor executes the algorithm sequentially, the MAP exploits several levels of parallelism. The two inner loops (for all probesets ... for each probe) are completely unrolled by the Perl script, so that all 6,000 operations in these loops are executed in parallel. A delay queue mechanism allows a new window to be produced every clock cycle. Downsampling by factors one and two are executed concurrently on the two User FPGAs in the MAP, followed by downsampling factors four and eight. Downsampling mode one accesses 307,200 40x40 pixels windows and downsampling mode four accesses 19,200 windows. This comprises most of the work. The gathering and suppressing of the winning scores is done in a short sequential loop. Using almost all of the logic space of both VP100 Virtex chips, clocked at 100 MHz, the MAP can operate at over 100 frames a second, three times the required rate, 300 times faster than the microprocessor.

Conclusion

We have introduced the MAP hardware and a two-stage software methodology, creating C programs from probesets using Perl and FPGA configurations from C codes using the

MAP C compiler. An automatic target recognition code created according to this methodology runs at frame rate on the MAP. This technology allows real-time (frame rate) object recognition of rigid targets.

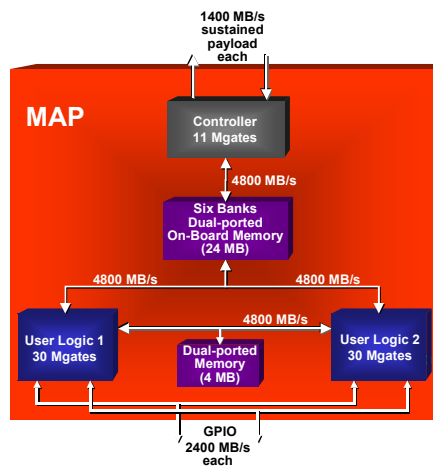


Figure 1: Compact MAP Processor

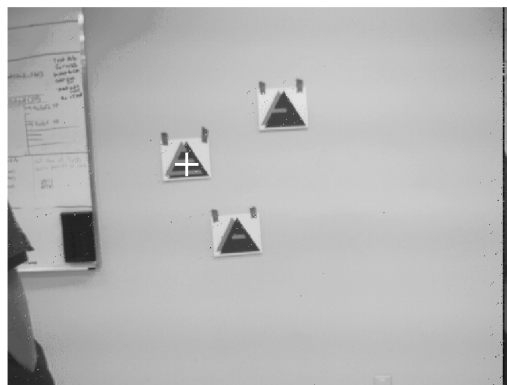


Figure 2: (Noisy) SRC logo and decoys

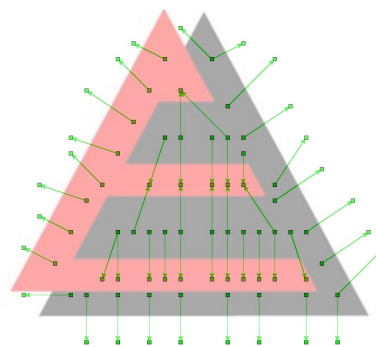


Figure 3: Performance results

References

[1] J.R Beveridge et.al., A Coregistration Approach to Multisensor Target Recognition, in Reconnaissance, Surveillance and

Target Acquisition for the Unmanned Ground Vehicle, pp 231-265, Morgan Kaufman, 1997.

- [2] Compiling ATR Probing Codes for Execution on FPGA Hardware, IEEE Symposium on Field-programmable Custom Computing Machines, Napa Valley, CA, April 21-24, 2002.