Subspace-Based Face Recognition on an FPGA

Pablo Pizarro, Miguel Figueroa

Subspace-based face recognition on an FPGA

Pablo Pizarro, and Miguel Figueroa

Department of Electrical Engineering
Universidad de Concepción
{ppizarroj, miguel.figueroa}@udec.cl

Abstract. We present a custom hardware system for image recognition, featuring a dimensionality reduction network and a classification stage. We use Bi-Directional PCA and Linear Discriminant Analysis for feature extraction, and classify based on Manhattan distances. Our FPGA-based implementation runs at 75MHz, consumes 157.24mW of power, and can classify a 61×49-pixel image in 143.7μs, with a sustained throughput of more than 7,000 classifications per second. Compared to a software implementation on a workstation, our solution achieves the same classification performance (93.3% hit rate), with more than twice the throughput and more than an order of magnitud less power.

1 Introduction

During the last few decades, automatic face recognition has evolved greatly. This development involved increased algorithmic complexity, which translates into long computation time and high energy consumption. Software implementations of most highly effective methods require the performance of a state-of-the-art workstation to operate in real time. Their cost, size and power requirements preclude their use in embedded and portable electronic systems. This has motivated the development of custom hardware implementations of face recognition algorithms, which can exploit the parallelism available in silicon integrated circuits, achieving high performance with much smaller die area and power than general-purpose microprocessors.

To accomplish high performance, neural network based on linear subspaces are normally used, mainly because VLSI technology favors architectures that feature regular computation and local or structured communication. Often the net- work is trained (and retrained) offline in software and the coeficients are then transfered onto the chip, greatly simplifying the implementation.

This paper presents a custom hardware implementation of an image classification algorithm for face recognition. The algorithm uses Bi-Dimensional Principal Components Analysis (BDPCA) and Linear Discriminant Analysis (LDA) for dimensionality reduction and feature extraction, and a Manhattan distance metric to perform the classifcation. We implemented our design on a Xilinx Spartan 3 XC3S1000 FPGA and tested it with the Yale database of face images. Our chip, compared to a floating-point software implementation of the algorithm, achieves the same classifcation hit rate of 93.3% and classifies an image in half the time (143.7μs), while consuming only 157.24mW of power.

2 Design

2.1 Algorithm and testing

To select a suitable algorithm to implement, we analyzed different designs and assessed their classification performance and hardware cost. We considered four algorithms for feature extraction: Eigenfaces (PCA), Fisherfaces (PCA+LDA), BDPCA and BDPCA+LDA [1,3,5]. Both PCA and BDPCA project the images, for these tests onto a 24-dimensional space, while LDA projects to 14 dimensions. For classification Euclidean and Manhattan distances are tested.

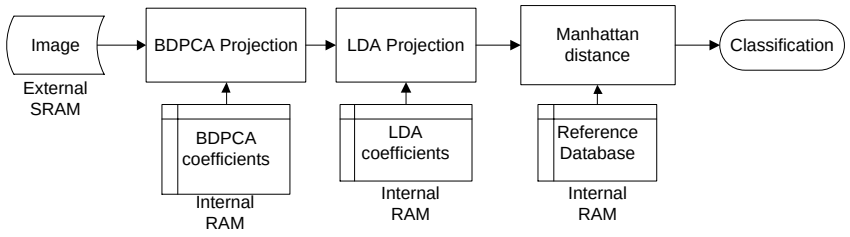
We used the Yale database as input data, which consists of 15 subjects, with 11 images of each, showing variations in lighting and facial expressions and details. The images are 8-bit grayscale, and were centered and resized to a resolution of 61×49 pixels using the Matlab Image Processing toolbox. We use 5 images of each subject for training, and 6 of each subject for testing. The procedure is performed 3 times, changing the selection for training and test images. We computed the mean classification hit rate and standard deviation for each experiment.

Fisherfaces achives the best recognition rate (97.8%) with lowest standard deviation. However, BDPCA+LDA achieves the second best performance (93%) with a memory usage 40 times lower than Fisherfaces and 3-4 times fewer arithmetic operations. The classification has a better performance using Manhattan distance. Based on the results, we chose to implement our hardware solution with BDPCA+LDA for feature extraction, and Manhattan distance for classification.

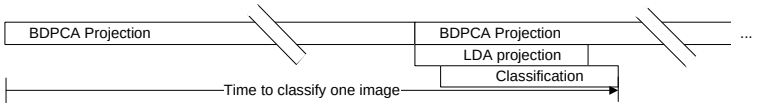
2.2 Architecture

Fig. 1(a) depicts the architecture of our image classifier. The classification is performed in three steps. First, the chip reads the image from external RAM and performs BDPCA dimensionality reduction. The second stage arranges the resulting matrix as a row vector and projects it onto the LDA feature space. Finally, the classifier computes the Manhattan distance between the vector and a database of reference face images in feature space, selecting the class based on the smallest distance. The BDPCA and LDA projection matrices and the classifier reference vectors are small, heavily reused, and require fast access, therefore they are stored in on-chip RAM. These matrices and reference vectors are calculated offline in a computer and are then transfered to the chip

Fig. 1(b) illustrates the three steps scheduled in a pipelined fashion. BDPCA projection is the slowest stage because it is the most computationally intensive, and is further limited by the external RAM access time. When the projected matrix is available, LDA projection starts concurrently with BDPCA projection for the next image. Distance computation and classification starts as soon as the first results from LDA are available, thus the two stages largely overlap.



(a) Block diagram



(b) Pipeline

Fig. 1. Architecture of the image classifier

2.3 Hardware platform

We implemented the architecture on a Xilinx Spartan 3 XC3S1000 FPGA chip with 24 18-bit embedded multipliers and 24 18-Kbit block RAM modules [4]. Because the FPGA does not feature floating-point hardware, we wrote our Verilog HDL descriptions using fixed-point arithmetic. We tested the system on a Digilent Nexys FPGA development board [2], which contains a Spartan 3 chip and an external 16 MByte RAM. The RAM interfaces to the FPGA through a 16-bit data bus, allowing us to read two pixels from the test image per each 80ns memory cycle. Currently, this memory bandwidth limits the performance of our classifier, as discussed in Section 3.

2.4 Implementation of BDPCA

Fig. 2 shows our pipelined implementation of the BDPCA algorithm. Module (M) reads the test image from external RAM, 2 pixels per memory cycle. Module (1) stores the 61×4 eight-bit coefficient matrix W_c in internal RAM and streams its elements at a rate of two 4-word rows per cycle. Modules (2) and (3) perform the multiplies and adds to implement the first projection (multiplication by W_c), streaming 4 words per cycle to the next stage. Module (4) stores the 49×6 eight-bit coefficient matrix W_r in internal RAM and streams it at 2 words per cycle to module (5), which performs the multiplies of the second projection (multiplication by W_r), streaming eight 16-bit words of the result per cycle. Finally, module (6) accumulates the results using a register file stored in internal

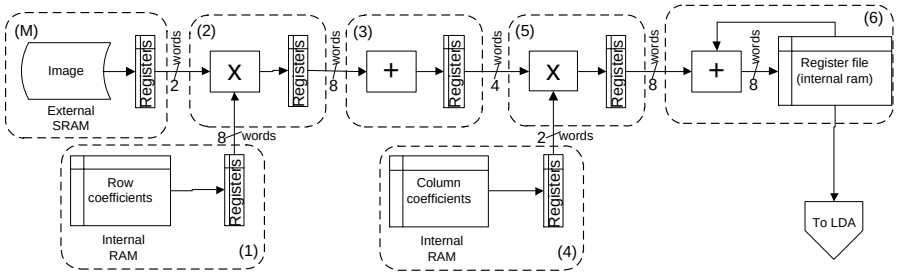


Fig. 2. Implementation of BDPCA

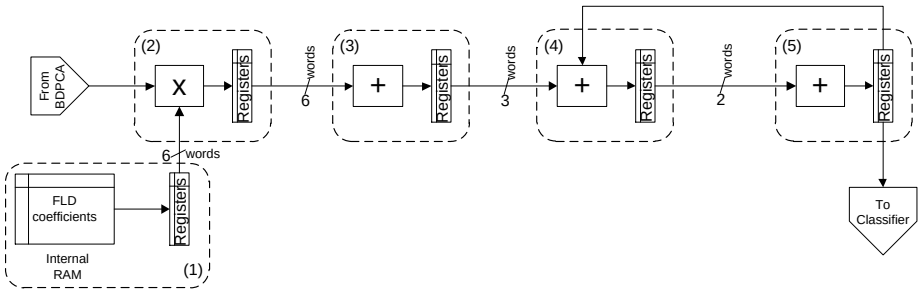


Fig. 3. Implementation of LDA

RAM, thus finishing the BDPCA projection. The resulting 4×6 matrix of 16-bit coefficients is stored in the register file, which is available to the LDA projector.

All 6 modules overlap their operation in a pipelined fashion. Because of limited external RAM bandwidth, the module processes 2 pixels of the input image every 7 clock cycles, with an initial latency of 13 cycles. With a faster memory, the module could process 2 pixels every 3 clock cycles with a latency of 6 cycles.

2.5 Implementation of LDA

Fig. 3 shows the LDA projector. As Fig. 1(b) shows, the module starts when BDPCA results are available in the register file. The block reads the 4×6 resulting matrix from BDPCA as a 24-element row vector. Module (1) stores the 14×24 LDA matrix using 16-bit coefficients, and streams it at a rate of 6 coefficients per cycle. Module (2) multiplies the 6 coefficients by 6 elements of the BDPCA vector, generating 6 results that are accumulated in a 3-stage pipeline by modules (3) to (4). Without this pipeline, the accumulation becomes the critical path of the circuit and limits the clock frequency. The module outputs 32-bit results, which are then used by the classifier.

As with the BDPCA block, all operations in the LDA projector are pipelined. The block produces a 32-bit element of the output vector in the feature space every 8 clock cycles, with an initial latency of 6 cycles.

Table 1. Implementation results at 75[MHz]

Classification hit rate			93.3%
Estimated speed	Limited by RAM bandwidth	1 image	143.7 μs
		throughput	7,045 images/second
	Not limited by RAM bandwidth	1 image	62.5 μs
		throughput	16,447 images/second
Estimated power consumption		Quiescent	99.41 mW
		Dynamic	57.83 mW
		Total	157.24 mW

2.6 Implementation of the classifier

The classifier, shown in Fig. 4, starts operating when the LDA projector produces its first result. Module (1) stores the reference database 14-element vectors, and streams four 32-bit coefficients per cycle. Modules (2) and (3) compute parallel subtractions, absolute values and accumulations to implement the Manhattan distance on 4 vectors simultaneously. Module (4) implements a Loser-Take-All (LTA) circuit that selects the smallest distance to classify the image. All operations are performed with 32-bit precision. Because the classifier overlaps its operation with the LDA projector, it outputs its result only 10 cycles after the LDA block finishes.

3 Experimental results

We synthesized a gate-level implementation of the classifier from our Verilog HDL code using the Synopsys Synplicity hardware compiler and mapped it onto a Xilinx Spartan 3 FPGA using the Xilinx place-and-route tool. The circuit achieves maximum throughput with a clock frequency of 75Mhz, limited by the access time of the external RAM. Table 1 shows the performance results for our implementation. We repeated the experiment of Section 2.1 with the Yale database on the FPGA, comparing its classification performance to the software implementation. Our fixed-point circuit achieves a hit rate of 93.3%, slightly better than the software for this particular dataset, but the difference is negligible

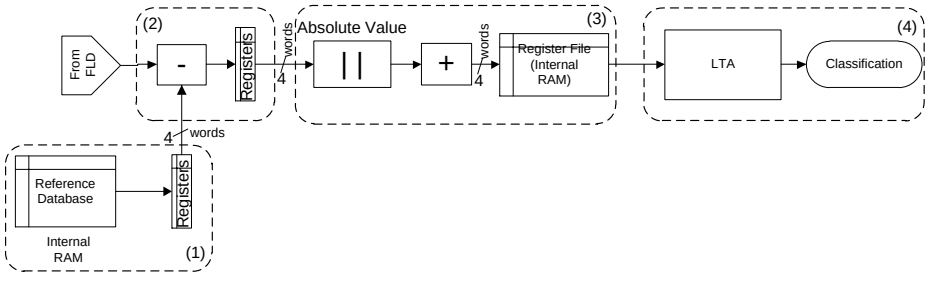


Fig. 4. Implementation of the classifier

(0.3%). The circuit classifies one image in 143.7μs, with a sustained throughput of 7,045 classifications per second. This is more than twice the throughput achieved by the software version running on a state-of-the-art workstation. Removing the limitation imposed by the external RAM, the performance more than doubles. We also estimated the power consumption using the Xilinx Xpower tool, showing that the circuit dissipates 157.24mW.

The utilization of combinational logic, flip-flops and RAM is less than 40%, the circuit is currently limited by the hardware multipliers (we use 22 out of 24 available on the chip). If we used a more complex classifier (e.g. an RBF network), we would need to reuse the multipliers with an increase in logic complexity.

4 Conclusions

We have described the architecture and implementation of a custom hardware digital implementation of a face recognition algorithm. Our solution uses BDPCA and LDA for feature extraction, and Manhattan distance for classification. We selected these algorithms based on their measured performance and their implementation cost in hardware. Our architecture makes use of parallelism and pipelined execution to maximize throughput limited by hardware resources. The resulting implementation on a Xilinx Spartan 3 FPGA achieves 93.3% classification hit rate on several experiments using the Yale database, which is equivalent to a floating-point software version. Compared to software running on a state-of-the-art microprocessor, our circuit performs more than twice as fast, with a reduction of more than one order of magnitude in power.

Acknowledgments

This work was funded by CONICYT Grant PFB-0824. Simulation and synthesis were done with EDA software donated by Synopsys.

References

1. K. Delac, M. Grgic, and P. Liatsis. Appearance-based statistical methods for face recognition. In *ELMAR, 2005. 47th International Symp.*, pages 151–158, 2005.
2. Digilent. *Digilent Nexys Board Reference Manual*, 2007.
3. M. Turk and A. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Procs. CVPR '91., IEEE Computer Society Conference on*, pages 586 –591, June 1991.
4. Xilinx. *Spartan-3 Generation FPGA User Guide*, 2008.
5. W. Zuo, D. Zhang, J. Yang, and K. Wang. BDPCA plus LDA: a novel fast feature extraction technique for face recognition. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(4):946 –953, Aug. 2006.