Architectures for Efficient Face Authentication in Embedded Systems *

Najwa Aaraj†, Srivaths Ravi‡, Anand Raghunathan‡, and Niraj K. Jha†
† Department of Electrical Engineering, Princeton University, Princeton, NJ 08544
‡ NEC Laboratories America, Princeton, NJ 08540
†{naaraj, jha}@princeton.edu ‡{sravi, anand}@nec-labs.com

Abstract

Biometrics represent a promising approach for reliable and secure user authentication. However, they have not yet been widely adopted in embedded systems, particularly in resource-constrained devices such as cell phones and personal digital assistants (PDAs). In this paper, we investigate the challenges involved in using face-based biometrics for authenticating a user to an embedded system. To enable high authentication accuracy, we consider robust face verifiers based on principal component analysis/linear discriminant analysis (PCA-LDA) algorithms and Bayesian classifiers, and their combined use (multi-modal biometrics). Since embedded systems are severely constrained in their processing capabilities, algorithms that provide sufficient accuracy tend to be computationally expensive, leading to unacceptable authentication times. On the other hand, achieving acceptable performance often comes at the cost of degradation in the quality of results.

Our work aims at developing embedded processing architectures that improve face verification speed with minimal hardware requirements, and without any compromise in verification accuracy. We analyze the computational characteristics of face verifiers when running on an embedded processor, and systematically identify opportunities for accelerating their execution. We then present a range of targeted hardware and software enhancements that include the use of fixed-point arithmetic, various code optimizations, application-specific custom instructions and co-processors, and parallel processing capabilities in multi-processor systems-on-chip (SoCs).

We evaluated the proposed architectures in the context of open-source face verification algorithms running on a commercial embedded processor (`Xtensa` from Tensilica). Our work shows that fast, in-system verification is possible even in the context of many resource-constrained embedded systems. We also demonstrate that high authentication accuracy can be achieved with minimum hardware overheads, while requiring no modifications to the core face verification algorithms.

1 Introduction

Embedded systems are ubiquitously used to capture, store, manipulate, and access data of a sensitive nature (*e.g.*, personal appliances such as cell phones, PDAs, smart cards, portable storage devices), or perform safety-critical functions (*e.g.*, automotive and aviation electronics, medical appliances). Such systems face some of the most demanding security concerns. They frequently operate in physically insecure environments, while the small form factor of devices such as cell phones and PDAs lends them to loss and theft. Furthermore, increasing programmability and networked nature of these devices make them tough to secure against various software attacks. While recent advances in embedded system security have addressed issues such as secure communication, secure information storage, and tamper resistance (protection from physical and software attacks) [1, 2, 3, 4], objectives such as user-to-device authentication have often been overlooked, placing a premium on the overall security of the system.

Currently, most solutions for user authentication use surrogate representations of a person's identity, such as passwords/personal identification numbers (prevalent in electronic access control) and token cards (prevalent in banking, corporate network, and government applications). These approaches suffer from several drawbacks, including insufficient security and inconvenience to users [5, 6]. *Biometrics*, which refer to the automatic recognition of people based on their distinctive physiological (*e.g.*, face, fingerprint, iris, retina, hand geometry, voice) and behavioral (*e.g.*, signature, gait) characteristics, could form a component of effective user identification solutions, because they intrinsically and reliably represent the individual's bodily identity [7]. Biometric characteristics cannot be lost or forgotten; they are quite difficult to copy, share, and distribute; and they require the person being authenticated to be physically present at the time and point of authentication.

In embedded systems such as mobile phones and PDAs, acquisition of voice and face is naturally possible due to the presence of microphone and camera. While the accuracy of authentication systems based on face and voice is lower than alternatives such as fingerprint and iris, voice and face biometrics come at a significantly lower cost. This prompts us to investigate their applicability in authenticating users to embedded systems. In this work, we specifically focus on face verification, and the challenges associated with their deployment in resource-constrained embedded systems.

One of the main challenges in deploying robust face verification algorithms comes from the limited processing capabilities of embedded systems. Since any authentication system involves two phases, namely, enrollment (when distinguishing characteristics of the user are extracted and stored as a mathematical model) and verification (when a device actually verifies the identity of a user against the enrolled model), both enrollment and verification can be time-consuming when high-accuracy face verifiers based on PCA-LDA and Bayesian algorithms are used. Therefore, our objective is to provide accurate and fast authentication through low-overhead modifications to the embedded SoC architecture. Our contributions include the following:

- We provide a comprehensive analysis of the computational characteristics of robust face verification algorithms such as PCA-LDA and Bayesian classifiers, while running on an embedded processor. We identify performance hotspots and other opportunities for optimizing their execution.
- Based on our performance analysis, we propose various hardware/software enhancements to improve both enrollment and verification times. Software enhancements include the conversion of floating-point to fixed-point arithmetic operations and the use of code optimizations such as loop unrolling and code re-ordering. We present hardware optimizations for both uniprocessor and multiprocessor systems. For uniprocessor embedded systems, we propose an architecture, wherein the processor is augmented with custom instructions and/or co-processors to accelerate the core kernels of face authentication. We also address the multiprocessor embedded SoC scenario, which is beginning to see practical application with the emergence of products such as NEC Electronics's MP211 application SoC for cell phones. Here, we observe that the latent parallelism of the architecture can be further exploited to provide improved authentication times. A specific application of this architecture is to make the deployment of multi-modal face biometric solutions (wherein multiple face verification algorithms are employed to improve authentication accuracy) feasible with minimum performance penalties.

*Acknowledgments: This work was supported by NSF under Grant No. CCR-0310477.

3-9810801-0-6/DATE06 © 2006 EDAA

- We perform our experimental evaluations in the context of a testbed featuring a state-of-the-art embedded processor (Xtensa [8]). We use popular, open-source implementations of face authentication algorithms and show that both enrollment and verification times can be sped up significantly with minimal overheads, while maintaining good authentication accuracy.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 details the computational characteristics of various kernels used in face verification. Section 4 presents architectures for efficient face verification, while Section 5 details the experimental results. Section 6 concludes.

2 Related Work

In this section, we briefly survey work related to face verification on embedded systems.

Face verification and/or recognition (matching image data to one or more persons in a database) is a well-researched problem, and various techniques (geometric, template, hybrid, 2-D or 3-D, *etc.*) have been proposed in the literature. Of these, PCA or the most expressive features method [9], LDA or Fisherfaces method [10, 11], independent component analysis (ICA) approach [12], elastic bunch graph matching (EBGM) method [13], Bayesian classifiers [14], *etc.*, have been widely recognized as effective techniques for performing face verification or recognition. A detailed survey of many of these techniques can be found in [15, 16].

Researchers have traditionally focused on improving the accuracy of face recognition systems. While solutions have thus emerged for overcoming specific problems such as illumination, face expression variations, noise in the image data, *etc.*, very little attention has been paid to the question of improving the efficiency of these systems. This is becoming a major concern, especially since face verification solutions are being considered for deployment in battery-powered embedded systems. One such effort is described in [17], wherein, a novel architecture is proposed that exploits the presence of an embedded FPGA in the SoC to accelerate various image and speech processing kernels. Other works focus on tuning the image pre-processing and face recognition algorithms to the needs of the end system. For example, various algorithmic design considerations have been made in [18] in order to reduce the complexity of face recognition. Commercially, various face verification (recognition) solutions are emerging for mobile devices. These include products such as the OKAO face recognition sensor [19] and FaceIt ARGUS for Motorola's cell phones [20]. The effectiveness of these solutions has not yet been widely studied/reported.

Another important trend is the usage of multiple biometrics (multimodal biometrics) to improve the authentication capabilities of face verification systems. Recent studies such as [21] use speaker identification to augment face verification in the context of a handheld device. The framework, however, completely relies on the transmission of image and audio data to an external server, so that the computational requirements of supporting multi-modal authentication on a PDA can be circumvented.

3 Computational Characteristics of Face Verification

In this section, we analyze the computational characteristics of two of the most robust face authentication algorithms: PCA-LDA and Bayesian based (Sections 3.1 and 3.2, respectively). We also examine a multimodal system that combines these algorithms (Section 3.3).

3.1 PCA-LDA

PCA-LDA based authentication employs PCA to minimize the dimensionality of the face image, while using LDA to find a subspace that minimizes differences between various images of the same user and emphasizing differences with images of other individuals. Authentication proceeds according to the overall flow chart shown in Figure 1 in two phases (i) a one-time enrollment or training phase, and (ii) the actual authentication or verification phase, which occurs whenever the user presents himself/herself to the device.

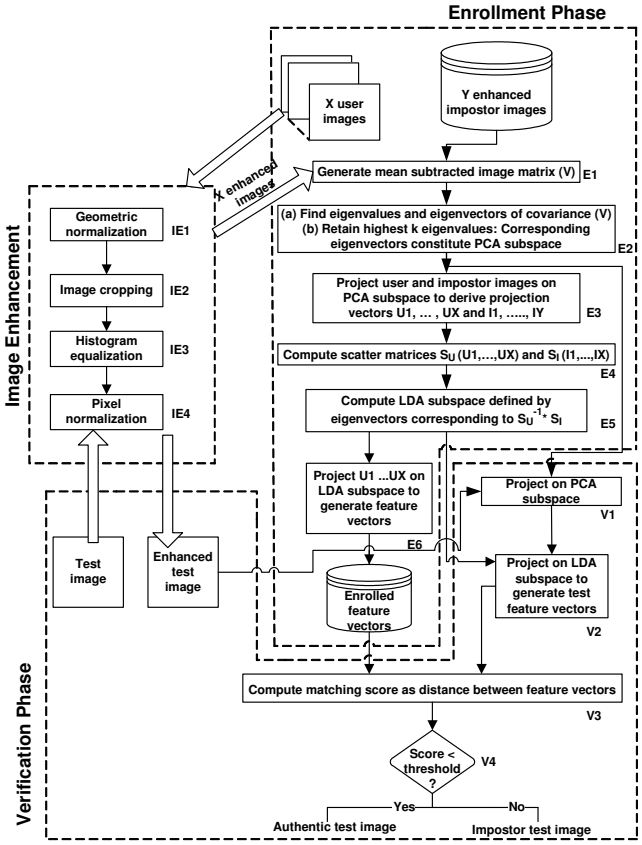

Figure 1: Flowchart of PCA-LDA based authentication

During enrollment, X number of images of the user is taken and presented to the algorithm. Each image is then enhanced and standardized to an $N \times M$ size, using the image enhancement (IE) step shown in the figure (discussed later). These are then used in combination with an existing impostor image database (containing say Y number of images) to derive the statistical features that are characteristic of the user images. The main steps of subspace derivation include: (E1) Computing a mean subtracted image matrix V, where each column of V represents the difference between image data (user or impostor) and the average of all the user and impostor images' data. All the user and impostor images are represented in this matrix, and thus, this matrix will have $(X+Y)$ columns and $N*M$ rows. (E2) Finding the covariance of matrix V, and the corresponding eigenvectors and eigenvalues. The eigenvectors corresponding to the largest k eigenvalues here constitute the basis of the PCA subspace. (E3) Projecting the data corresponding to the user and impostor images on the PCA subspace to derive the corresponding projection vectors (say vectors $U1, \cdots, UX$ and $I1, \cdots, IY$), each having k components. (E4) Computing the scatter matrices between projections of the user images (called within-class scatter matrix S_U) and between projections of the impostor images (called between-class scatter matrix S_I). (E5) Finding the eigenvectors of matrix G given by $S_U^{-1} \times S_I$, which constitutes the basis of the LDA subspace. (E6) Determining the user's feature vectors as a projection of vectors $(U1, \cdots, UX)$ on the LDA subspace.

Verification of a user's identity proceeds according to steps V1-V4 in the figure. We obtain the user's image, apply the image enhancement step, and project the resulting images on the PCA and LDA subspaces to obtain the corresponding feature vector. A distance measure is then computed between this feature vector and the user's enrolled feature vectors to yield a matching score. Comparison with a set threshold is then used to decide the identity of the user.

We analyzed the computational requirements of running the PCA-LDA based authentication on a 100MHz Xtensa embedded processor. For our experiments, we used five user images (X=5) and two mean impostor images (Y=2) during enrollment, while using one test image for verification. We found that enrollment takes 26.75 sec., while verifi-

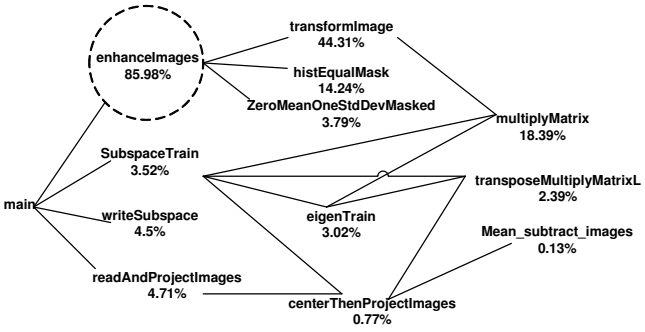

Figure 2: Function call graph for the enrollment phase

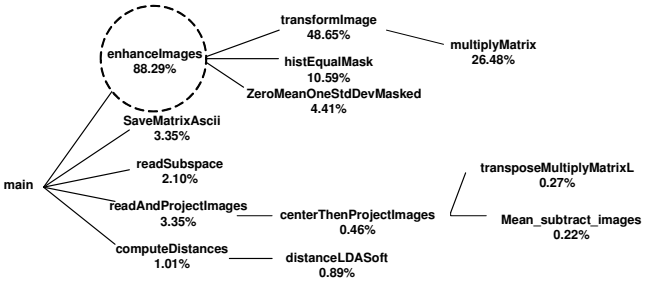

Figure 3: Function call graph for the verification phase

cation takes 5.21 sec. The profiles for enrollment and verification are shown in Figures 2 and 3, respectively. They reveal that nearly 85.98% of enrollment and 88.29% of verification time are spent in the image enhancement step.

Let us examine the image enhancement step (Figure 1) in more detail. It includes the following steps. (IE1) Geometric normalization, where a $(W * L)$ image is standardized to an image size of $(N * M)$. This is done by first generating a $3 * 3$ transform matrix that specifies the amount of translation, rotation, scaling, and reflection needed. The corresponding matrix values are derived based on the eye coordinates of the captured image. Using the transform matrix, the pixels of the source image can be interpolated to derive the standard size image. (IE2) Image cropping, where a standard elliptical mask is used to crop the image such that a selected region of the face (from forehead to chin and cheek to cheek) remains visible. (IE3) Histogram equalization, where an elliptical mask is used to equalize the histogram of the unmasked part of the image, and (IE4) Pixel normalization, where the pixel values are scaled to have a mean of 0 and a standard deviation of 1.

We extracted a profile of a single image enhancement run (see Figure 4). It shows that the dominant function in image enhancement is the function *transformImage* used in geometric normalization, which takes 51.6% of the time. Nearly 25% of that time is spent in matrix multiplication (function *MultiplyMatrix*), while 19.5% of the time is used to carry out linear interpolation (function *InterpLinear*). This analysis motivates us to optimize these functions so as to improve both enrollment and verification times.

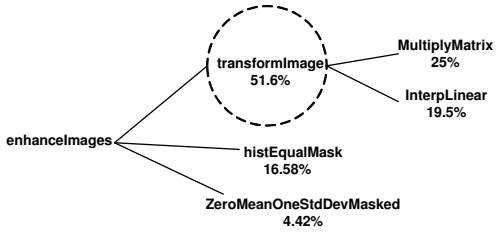

Figure 4: Function call graph for the image enhancement phase

3.2 Bayesian Authentication

Figure 5 shows the complete flowchart for Bayesian authentication. As with the PCA-LDA based approach, the authentication process includes the enrollment and verification phases.

During enrollment, we obtain X images from a user, and use an available database of Y impostor images. Based on specified parameters $N1$ and $N2$, we obtain $N1$ intrapersonal difference images and $N2$ interpersonal difference images (step E1). An intrapersonal image refers to the difference image between two user images. An interpersonal image refers to the difference image between a user image and an impostor image. We then derive the PCA subspaces for the intrapersonal images and interpersonal images (called intraSubspace $S1$ and extraSubspace $S2$, respectively) (step E2), as explained in Section 3.1.Finally, we compute the mean of the user images (denoted mean image U) and impostor images (denoted mean image I) (step E3). During verification, we obtain a test image T from the user, enhance it, and then compute the difference images given by $D1 = T - U$ and $D2 = T - I$ (step V1). We then project $D1$ ($D2$) on subspaces $S1$ and $S2$, and generate the maximum-likelihood distances C (E) and D (F) (step V2). Next, we compute distances $A = C + D$ and $B = E + F$. If $B/A > 1$ and A is less than a specified threshold, the user is authenticated (step V3).

We analyzed the computational requirements of running Bayesian authentication on a 100MHz Xtensa embedded processor. For our experiments, we used three user images (X=3) and two mean impostor images (Y=2) during enrollment, and one test image for verification. We found that enrollment takes 23.15 sec., while verification takes 6.61 sec. Execution time profiles reveal that nearly 61.05% of enrollment and 69.59% of verification time are spent in the image enhancement step, making it the performance hotspot that must be targeted for optimization.

We also identified opportunities for parallelism in both enrollment and verification. During enrollment, computation of the PCA subspaces for the user class (step E2(a)) and impostor class (step E2(b)) are independent, time-consuming tasks. Each subspace computation takes 4.34 sec., which can benefit from any parallelism in the underlying architecture. Similarly, during verification, the projection of difference images $D1$ and $D2$ onto the PCA subspaces $S1$ and $S2$ can be split into two parallel tasks. Each task takes nearly 0.62 sec.

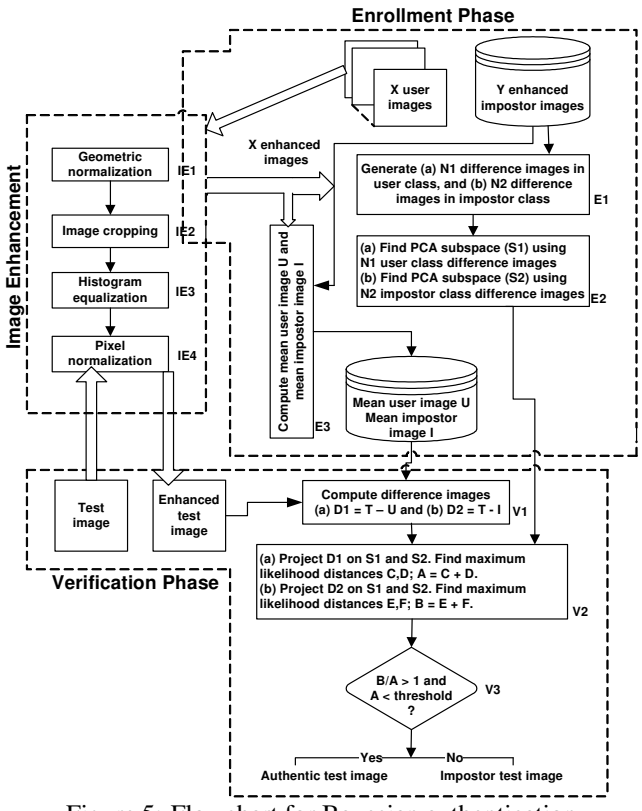

Figure 5: Flowchart for Bayesian authentication

3.3 Multi-modal Face Biometrics

PCA-LDA and Bayesian face authentication approaches can be combined as a part of a single, multi-modal authentication system, as shown

in Figure 6. Both algorithms then share the user image acquistion and image enhancement processes during both enrollment and verification. Verification proceeds by generating matching scores K and B from each verifier, and fusing them at this level. Score fusion is through a "Simple Sum of Scores" rule, which can be applied after the individual scores are normalized. For normalization, we used the tanh normalization method [22], which has been shown to be highly effective in practice. The various factors used in the tanh estimator were estimated empirically, and are shown in the figure. We omit further details for brevity.

In such a system flow, we can clearly identify kernels that are independent, time-consuming, and hence, parallelizable. Subsequent to image enhancement, enrollment for PCA-LDA and Bayesian approaches can occur in parallel. The corresponding computations take 3.75 sec. and 9.35 sec., respectively. Similarly, steps PLV1-PLV2-PLV3 and BV1-BV2-BV3 in verification can occur in parallel [consuming 0.61 sec. and 1.05 sec. (before conversion to fixed-point arithmetic), respectively].

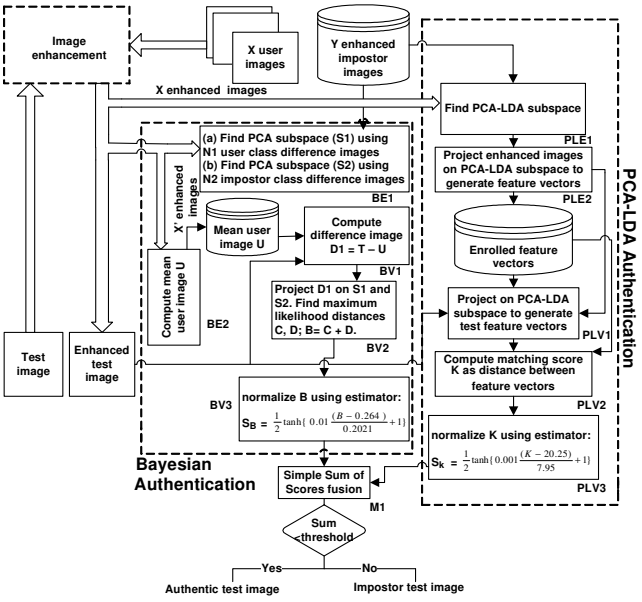

Figure 6: Multi-modal system flow graph

4 Architectures for Efficient Face Authentication

In this section, we present various harware/software architectural enhancements that optimize the performance of PCA-LDA, Bayesian, and multi-modal authentication. We describe optimizations relevant to both uniprocessor and multiprocessor systems. In this context, we use the generic SoC template shown in Figure 7, which includes two processors $p1$ and $p2$, each with its private memory and communicating using a shared memory. The processors we consider in our work are the extensible Xtensa processors, whose instruction set can be customized to accelerate fine-grained hotspots in the application. Large-granularity kernels can be accelerated by adding co-processing hardware outside the processor core.

4.1 Optimizations for Uniprocessor Systems

In this section, we describe the various optimizations used to optimize the execution of the face verification algorithms given in Section 3 for a uniprocessor system. Since performance analysis revealed that significant portions of execution time are spent in the image enhancement step, our optimizations target this step (though the software optimizations were applied to the entire algorithm), and include the following:

- *Software optimizations:* By examining the performance profiles of image enhancement at a fine-grained level, we observed that a high percentage of execution time is spent in 32-bit floating-point operations. Since the embedded processor does not include a floating-point unit, and hence, emulates their operations, we transformed the operations from floating-point to fixed-point arithmetic (the alternative solution of adding a floating-point unit is too expensive for the gains that can be obtained) and were able to speed the algorithm up to 2.07 sec. (from 4.6 sec.). We found that judicious use of two representation formats (Q20 and Q16) for fixed-point arithmetic enabled us to avoid any overflow/underflow problems. The use of fixed-point operations also enabled us to add custom instructions (see hardware optimizations below) that can accelerate their computation (a property of the Xtensa design process). We further identified code snippets that were amenable to optimizations such as loop unrolling and replacement of select divisions with shift operations. This allowed us to reduce the execution time to 1.65 sec.

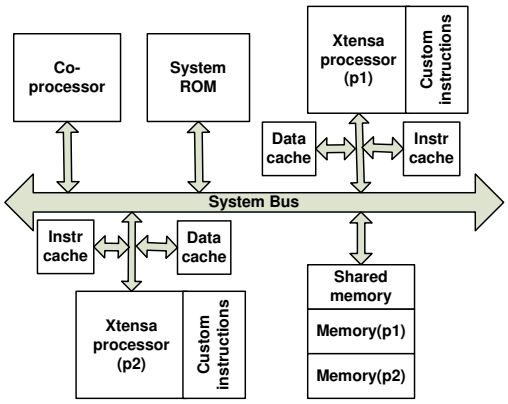

Figure 7: Generic architectural model

```
fp31 interpLinear(Image img, fp31 x, fp31 y, int c)
{
    fp31 xfrac = SUB(x , FLOOR(x));
    fp31 yfrac = SUB(y , FLOOR(y));
    double K = fixed_to_float(x);
    double L = fixed_to_float(y);
    int xLower = INT_FLOOR(K);
    int xUpper = INT_CEIL(K);
    int yLower = INT_FLOOR(L);
    int yUpper = INT_CEIL(L);

                                        Part 1

    fp31 valUpper, valLower, value, value1;
    valUpper = ADD( MULT(ie(img,xLower,yUpper,c),SUB(FPONE,xfrac)) ,
               MULT(ie(img,xUpper,yUpper,c),(xfrac)) );
    valLower = ADD( MULT(ie(img,xLower,yLower,c),SUB(FPONE,xfrac)) ,
               MULT(ie(img,xUpper,yLower,c),(xfrac)) );
    value = MULT(valLower, SUB(FPONE,yfrac));
    value1 = MULT(valUpper,yfrac);
    return ADD(valeur,valeur1);

                                        Part 2

}
```

Figure 8: Original code of function *interpLinear*

- *Hardware optimizations (custom instructions):* The performance profile of the image enhancement phase shown in Figure 4 shows that a significant percentage of execution time is spent in functions interpLinear and histEqualMask (this is true even after the software optimizations are carried out). Figure 8 shows a code snippet that outlines the operations performed in function interpLinear. The function takes a pixel of an input image, and outputs a value for the pixel's intensity that is interpolated based on the intensity values of the pixels in its neighbourhood. Logically, the function consists of two kernels of computation (i) the first part (labeled Part 1) performs a sequence of subtraction, ceiling, and floor operations on the input pixel co-ordinates to determine the pixel's neighbours, and (ii) the second part (labeled Part 2) is a series of multiplication-addition operations that computes the desired interpolated intensity value. We can accelerate the performance of function interpLinear by executing code snippets Parts 1 and 2 by adding the custom hardware shown in Figure 9 into the execution stage of the Xtensa processor's pipeline. At the application level, the custom hardware can be accessed through two new instruction calls: INTERP1 and MULT_TIE. Note that the MULT_TIE instruction is implemented such that it can be used to

accelerate function histEqualMask as well. In the absence of the new custom instructions, one call to interpLinear takes 2571 cycles, with 228 and 2343 cycles spent in Parts 1 and 2, respectively. With the new custom instructions, the computations corresponding to each part of this function execute in one clock cycle. This custom instruction-based optimization reduces the execution time of the image enhancement step to 0.97 sec.

- *Hardware optimizations (co-processors):* Further acceleration is feasible by using a co-processor that can perform two coarse-grained tasks in parallel to master processor $p1$, namely, generation of the transform matrix (part of steps IE1, IE2) and generation of the elliptical mask needed by steps IE3 and IE4. Figure 10 shows the timeline of operations for master processor $p1$ and the co-processor. A specified address in shared memory, called *lock*, guarantees memory consistency between $p1$ and the co-processor. While $p1$ is reading the image from the input file, the co-processor generates the transform matrix by reading the small portion of the image needed to determine the eye co-ordinates. Processor $p1$ can access the transform matrix only after *lock* is set to 1. In the meantime, the co-processor begins the task of mask generation. The co-processor implementation reduces processing time of image enhancement to 0.565 sec.

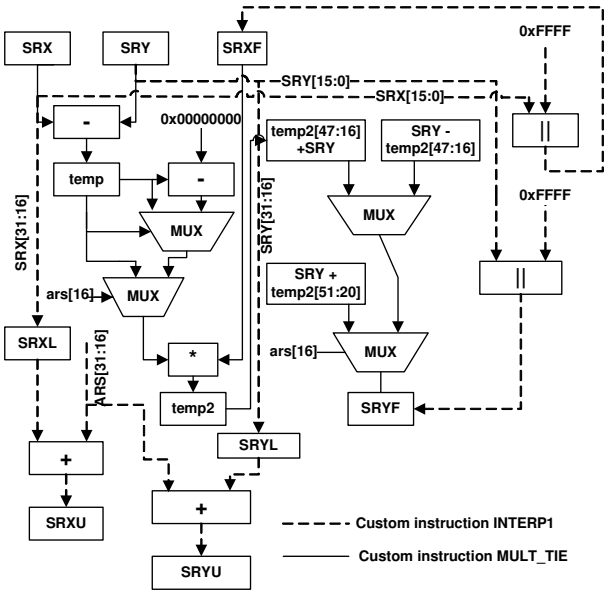

Figure 9: Register-transfer level description of custom instructions added for accelerating image enhancement

4.2 Optimizations for Multiprocessor Systems

The optimizations described above for uniprocessor systems are also applicable to multiprocessor systems. In this section, we specifically consider the two-processor scenario, and detail additional optimizations that can exploit the available architectural parallelism.

4.2.1 Speeding up Bayesian Authentication

Section 3.2 identified steps in Bayesian enrollment and verification, which were parallelizable. Figure 11 shows the timeline of operations, wherein Bayesian verification benefits from executing tasks V1(a) and V2(a) on one processor, while simultaneously executing tasks V1(b) and V2(b) on another processor. Processor $p1$ reads the image and stores it in shared memory so that it can be accessed by both processors $p1$ and $p2$. Maximum-likelihood distance computations can then proceed in parallel, after which processor $p1$ decides whether the user is authenticated or not (step V3). Again, synchronization between the two processors is achieved by setting, reseting, and polling a common memory location *lock*. On a single processor, Bayesian verification of an enhanced test image takes 1.63 sec. (after the uniprocessor software optimizations), while Bayesian verification in the two-processor scenario takes only 0.821 sec.

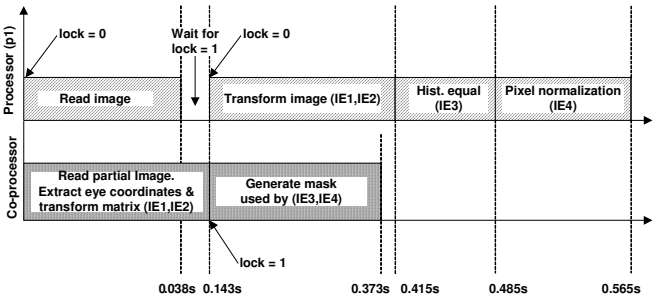

Figure 10: Timeline of operations in a uniprocessor system with a co-processor present

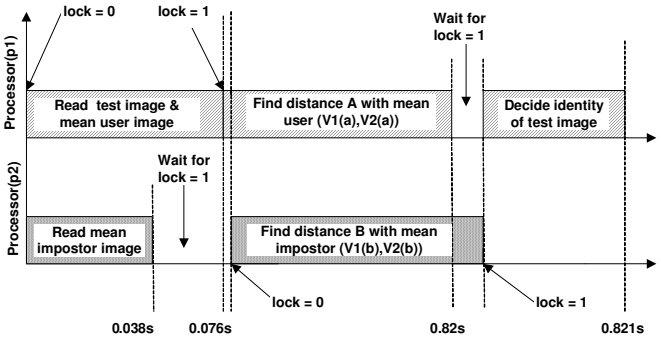

Figure 11: Bayesian verification in a two-processor system

4.2.2 Multi-modal Biometrics

The coarse-grained parallelism between the two authentication algorithms, as identified in Section 3.3, allows for the implementation of the scheme outlined in Figure 12. The enhanced test image is read by the two processors $p1$ and $p2$, which will run the PCA-LDA and Bayesian verification steps. Since processor $p1$ computes normalized LDA distances in 0.47 sec., it waits until processor $p2$ computes the Bayesian distance. At this point, $p2$ releases the synchronization variable *lock* so that fusion of scores can be performed by $p1$. Overall, multi-modal verification on a dual processor takes only 0.824 sec. to complete, while taking 2.062 sec. on a uniprocessor.

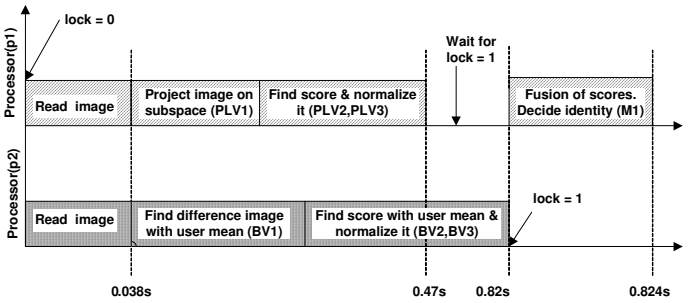

Figure 12: Multi-modal verification in a two-processor system

5 Experimental Results

We next present our experimental results.

Our experimental setup is as follows. We designed our authentication system using the open-source face recognition software suite provided by Colorado State University [23]. We used the grayscale FERET Database provided by NIST [24] for various experiments and the open-source software XnView [25] for image format conversions.

We evaluated the proposed architectural enhancements in the context of the Xtensa platform [8]. The base processor considered in our experiments is a five-stage embedded RISC processor, with 32 KB instruction and data caches. The area of the base processor is 402,667 grids. The

Xtensa software development toolkit was used for cross-compilation, instruction set simulation, and profiling. The custom instructions designed in our work are specified in a hardware description language called Tensilica Instruction Extension (TIE), and added to the base processor using the Xtensa processor generator. In the following subsections, we present various area and performance results for our architectural modifications, and also discuss trade-offs between accuracy and performance.

5.1 Performance and Area Results

In Section 4, we presented various software and hardware optimizations for the uni-modal and multi-modal systems. Figure 13 provides the execution times at various stages of the optimization process for enrollment and verification using PCA-LDA and Bayesian algorithms. We can see that the enrollment and verification phases in PCA-LDA are sped up by factors of 4.8X and 5.0X, respectively, while enrollment and verification in Bayesian are sped up by factors of 2.3X and 3.0X, respectively. Among the various optimizations, the use of fixed-point arithmetic and custom instructions provide the best accceleration. In the two-processor scenario, Bayesian enrollment and verification are further sped up to 7.08 sec. and 1.791 sec., respectively. For the multi-modal scenario, we found that enrollment and verification times on a two-processor system can be reduced to 11.38 sec. and 1.794 sec., respectively.

For the image enhancement step, we evaluated the area overhead of adding custom instructions (INTERP1 and MULT_TIE) to the base Xtensa processor. The area overhead is 19,386 grids (only 4.81% of the base processor), where 4 grids are equivalent to one gate using NEC's 0.18μ CB-12 CMOS standard cell library. We also evaluated the energy consumption of the image enhancement algorithm running on the customized processor by applying the energy macro-model proposed in [26]. The energy consumed is 28.27mJ, thus achieving an energy-delay product reduction of 5.12X over the base processor core.

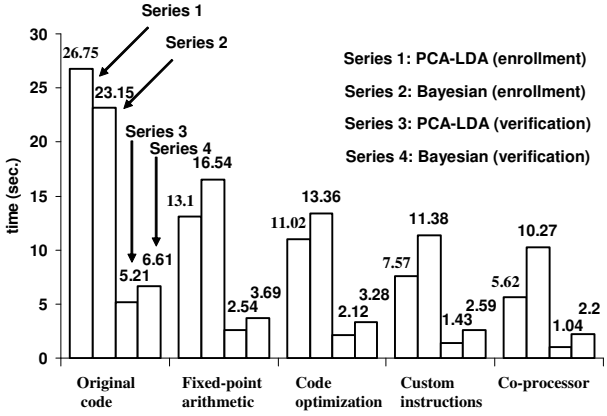

Figure 13: Speedup achieved at each step of the optimization process

Table 1: Accuracy and performance characteristics

Algorithm	*#Img*	*EER(%)*	*En. (sec) B.O.*	*Ver. (sec) B.O.*	*En. (sec) A.O.*	*Ver. (sec) A.O.*
PCA-LDA	8	31.2	42.80	5.25	11.98	1.44
	7	21.8	37.33	5.23	10.33	1.44
	6	23.5	31.84	5.23	8.97	1.44
	5	7.6	26.75	5.21	7.57	1.40
	4	8.2	21.4	5.21	5.44	1.40
Bayesian	3	7.2	23.15	6.61	6.61	2.59
	5	6.3	35.25	13.38	7.63	3.78
Multi-modal		5.8	26.75	11.38	5.65	1.78

5.2 Accuracy and Performance Tradeoffs

For the various authentication algorithms, we computed the false accept rate (FAR) and false reject rate (FRR) by varying the acceptance threshold. We computed the equal error rate (EER) (the point where FAR equals FRR) of the system and we found that the use of five user images for PCA-LDA enrollment yields the best EER (7.6%).

Table 1 summarizes the EERs (column 3) for the various authentication algorithms (column 1), and the execution times [before optimizations (B.O.) and after optimizations (A.O.)] for the enrollment and verification phases (columns 3-6). Column 2 indicates the number of user images used in enrollment. The results show that we can obtain EERs of 7.6% in PCA-LDA, 6.3% in Bayesian, and 5.8% in multi-modal systems with reasonable execution times.

6 Conclusion

In this paper, we introduced various architectural enhancements that enable the deployment of uni-modal and multi-modal authentication algorithms. The proposed enhancements include software optimizations, hardware additions in the form of custom instructions and co-processors, and exploitation of parallel processors (if available). We evaluated the proposed enhancements in the context of a commercial embedded procesor, and showed that both enrollment and verification can be performed efficiently in the system.

References

[1] *Intel(R) Wireless Trusted Platform: Security for Mobile Devices*. Intel Corp., 2004.

[2] S. Hattangady and C. Davis, *Reducing the Security Threats to 2.5G and 3G Wireless Applications*. Texas Instruments Inc., 2002.

[3] R. York, *A New Foundation for CPU Systems Security*. ARM Limited, 2003.

[4] P. Kocher, R. B. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a new dimension in embedded system design," in *Proc. Design Automation Conf.*, pp. 753–760, June 2004.

[5] D. V. Klein, "Foiling the cracker: A survey of, and improvements to, password security," in *Proc. Wksp. USENIX Security*, pp. 5–14, July 1990.

[6] I. Armstrong, "Passwords exposed: Users are the weakest link," in *http://www.scmagazine.com*, June 2003.

[7] A. Jain, R. Bolle, and S. Pankanti, eds., *Biometrics: Personal Identification in Networked Society*. Boston, MA: Kluwer Academic, 2002.

[8] *Xtensa Application Specific Microprocessor Solutions - Overview Handbook*. Tensilica Inc. (http://www.tensilica.com), 2001.

[9] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Conf. Computer Vision & Pattern Recognition*, pp. 561–586, June 1991.

[10] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Analysis & Machine Intelligence*, vol. 19, pp. 711–720, July 1997.

[11] Y. Li, J. Kitller, and J. Matas, "Face verification using client specific Fisherfaces," in *Proc. Int. Conf. Statistics of Directions, Shapes & Images*, pp. 63–66, Sept. 2000.

[12] C. Havran, L. Hupet, and J. Czyz, "Independent component analysis for face authentication," in *Proc. Knowledge-Based Intelligent Information & Engineering Systems*, pp. 1207–1211, Sept. 2002.

[13] L. Wiskott, J.-M. Fellous, and N. Krüger, "Face recognition by elastic bunch graph matching," *Pattern Analysis & Machine Intelligence*, vol. 19, pp. 775–779, July 1997.

[14] B. Moghaddam, C. Nastar, and A. Pentland, "A Bayesian similarity measure for direct image matching," in *Proc. Int. Conf. Pattern Recognition*, pp. 350–358, Aug. 1996.

[15] S. Z. Li and A. K. Jain, *Handbook of Face Recognition*. Springer, 2005.

[16] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, pp. 399–458, Dec. 2003.

[17] M. Borgatti, F. Lertora, B. Foret, and L. Cali, "A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O," *IEEE J. Solid-State Circuits*, vol. 38, pp. 521–529, Mar. 2003.

[18] J. Yang, X. Chen, and W. Kunz, "A PDA-based face recognition system," in *Proc. Wkshp. Applications of Computer Vision*, pp. 19–23, Dec. 2002.

[19] *OKAO Vision Face Recognition Sensor*. OMRON, 2005.

[20] R. Naraine, *Face Recognition, via Cell Phones*. Internetnews.com, 2002.

[21] T. Hazen, E. Weinstein, R. Kabir, A. Park, and B. Heisele, "Multi-modal face and speaker identification on a handheld device," in *Proc. Wkshp. Multimodal User Authentication*, pp. 120–132, Dec. 2003.

[22] A. K. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," *to appear in Pattern Recognition*.

[23] "Evaluation of face recognition algorithms, http://www.cs.colostate.edu/evalfacerec."

[24] "The FERET database, http://www.itl.nist.gov/iad/humanid/feret/."

[25] "Xnview - free graphic viewer, http://www.xnview.com."

[26] Y. Fei, S. Ravi, A. Raghunathan, and N. K. Jha, "Energy estimation for extensible processors," in *Proc. Design Automation and Test in Europe Conf.*, pp. 10682–10687, Mar. 2003.