A Compact and Efficient Fingerprint Verification System for Secure Embedded Devices

Shenglin Yang
UCLA Dept of EE
Los Angeles, CA 90095
+1-310-267-4940
shengliny@ee.ucla.edu

Kazuo Sakiyama
UCLA Dept of EE
Los Angeles, CA 90095
+1-310-267-4940
kazuo@ee.ucla.edu

Ingrid M. Verbauwhede
UCLA Dept of EE
Los Angeles, CA 90095
+1-310-794-5209
ingrid@ee.ucla.edu

***Abstract*--Creating a biometric verification system in a resource-constrained embedded environment is a challenging problem. This paper describes an efficient fingerprint verification module, which is part of an embedded device called ThumbPod. The whole fingerprint verification algorithm runs on a 50MHz fixed-point processor. As the result of our SW/HW optimizations, we achieve 55.6% and 60.0% execution time reduction for the minutiae extraction and the matching, respectively, compared to a traditional implementation reference. The complete process finishes in less than 5 seconds.**

I. INTRODUCTION

ThumbPod [1] is a biometrically secure embedded device, in which the complex and computation intensive biometric process module is executed locally on a 50MHz fixed-point LEON-2 processor. In a traditional distributed system application involving resource-limited embedded devices, system partitioning between the server and the embedded device is usually based on computation distributing. However, ThumbPod requires a partitioning that also takes security into consideration. Because of the unique and sensitive biometric data, we need to perform the full biometric processing locally on the ThumbPod instead of offloading it to the server. ThumbPod consists of four basic subsystems: data collection, minutiae extraction, matching and communication. The first three parts take care of the biometric processing and verification, while the communication part transmits the matching result, a yes/no answer, to the server.

II. RELATED WORK

Traditional fingerprint verification systems use the central server to store the template fingerprint. However, this can cause critical biometric information leakage. Some systems try to decentralize the fingerprint template into a storage device such as smart card [7], [8]. Although this provides higher security for the fingerprint matching process and the biometric storage, the minutiae extraction process running on the card reader and the transmission of the input fingerprint information still can lead to disclosure of the precious biometric data. What is unique to our proposed method is that both minutiae extraction and matching are executed locally on the embedded device, gaining maximum security in the system.

This work is sponsored by the NSF, account no CCR-0098361 and the Langlois foundation.

III. MINUTIAE EXTRACTION

The starting point for the algorithm used to extract minutiae of the fingerprint image is taken from NIST Fingerprint Image Software [3]. Fig. 1 shows the basic steps of it.

The fundamental step in the minutiae extraction process is deriving a directional ridge flow map. To get this map, the fingerprint image (256×256 pixels) is first divided into a grid of blocks (8×8 pixels). For each block, there is a surrounding window (24×24 pixels) centered by this block and the surrounding window is rotated incrementally. A DFT analysis is conducted at each orientation and the number of orientations is set to 16, creating an increment in angle of 180°/16, i.e. 11.25°. Within an orientation, the pixels along the rotated row of the window are summed together, forming a vector of 24 row sums. The 16 orientations produce 16 vectors of row sums (Fig. 2). The resonance coefficients produced by convolving each of the 16 row sum vectors with four different discrete waveforms are analyzed and the dominant ridge flow direction for the block is determined by the orientation with the maximum waveform resonance calculated from (1):

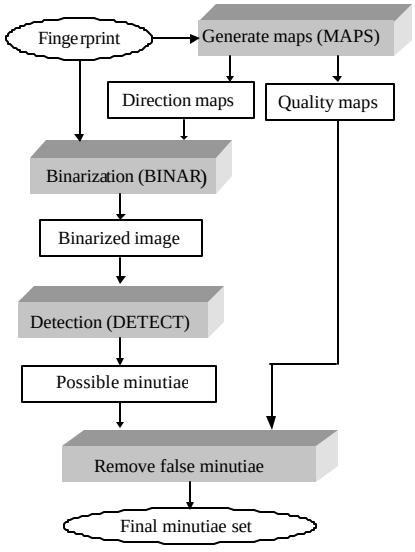

Fig. 1. NIST minutiae extraction flow.

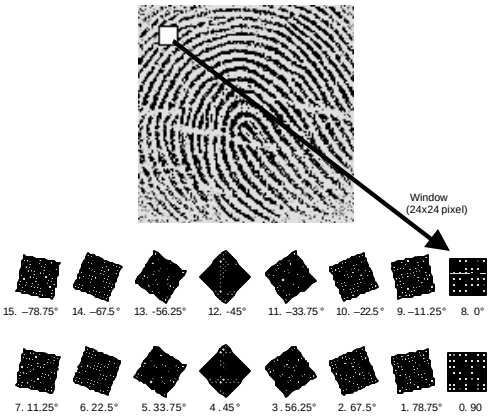

Fig. 2. Window rotation.

$$E(k,q)=\left|\sum_{n=0}^{23}\text{row\_sum}(n,q)W^{kn}\right|^2 \quad , \quad W=\exp\left(\frac{-jp}{16}\right) \qquad (1)$$

$(k=1, 2, 3, 4)$

Each pixel is assigned a binary value based on the ridge flow direction associated with the block to which the pixel belongs. Following the binarization, the detection step methodically scans the binary image of a fingerprint, identifying localized pixel patterns that indicate the ending or bifurcation of a ridge. After all the minutiae candidates are pointed out, the final step is then to remove false minutiae from these candidates and keep the true ones.

NIST Fingerprint Image Software is based on floating point computations. However, the processor used in ThumbPod only supports fixed-point calculation. Therefore before porting the software to ThumbPod, fixed-point refinement is needed. Using the refined algorithm, the minutiae set can be detected. An example is presented in Fig. 3.

IV. FINGERPRINT MATCHING ALGORITHM

After two minutiae sets, which are for the input fingerprint image and the template fingerprint image respectively, are extracted, the matching algorithm can be described.

In the algorithm flow shown in Fig. 4, the first step is to find out the correspondence of these two minutiae sets. A minutia, N, can be described by a feature vector: $N=(x,y,j,t)$, where (x,y) is its coordinate, j is the local ridge direction and t is the minutia type. However, x, y and j cannot be directly used for matching since they are dependent on the

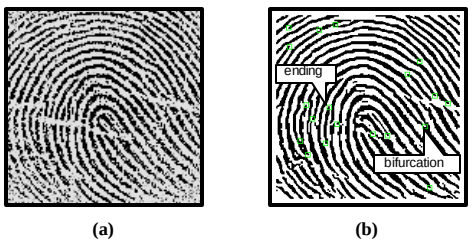

Fig. 3. Original image and minutiae in the binarized image.
(a) Original image (256×256 pixels).
(b) Binarized image and minutiae points.

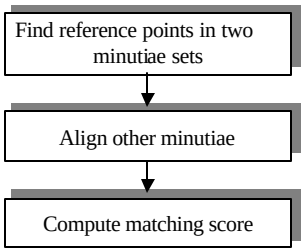

Fig. 4. Matching algorithm flow.

rotation and translation of the fingerprint. To solve this problem, we construct a rotation and translation invariant feature:

$$M=(d_1,d_2,q_1,q_2,j_1,j_2,n_1,n_2,t,t_1,t_2) \qquad (2)$$

Fig. 5 indicates the details of this local feature, where n is the number of ridges between neighbors. Assume $M_I(i)$ and $M_T(j)$ are the local feature vectors of the i-th minutia of the input fingerprint and the j-th one of the template fingerprint, respectively. A similarity level of these two minutiae can be defined as:

$$sl(i,j)=\begin{cases}1-\dfrac{|M_I(i)-M_T(j)|_W}{A}, & if\ |M_I(i)-M_T(j)|_W<A(W)\\ 0, & otherwise\end{cases} \qquad (3)$$
$$i=1,2\ldots p \quad j=1,2\ldots q$$

where p and q are the total numbers of minutiae in the input fingerprint and the template fingerprint, respectively. $|M_I(i)-M_T(j)|_W$ is the weighted distance between two local feature vectors. $A(W)$ is the threshold, which is related to the weight vector W. In this paper, we set $W=(1,1,8,8,8,8,3,3,1/3,1/3,1/3)$ and $A(W)$ = 55. By thoroughly searching $sl(i,j)$, one minutiae pair (b_1,b_2) can be reached so that $sl(b_1,b_2)=\max_{i,j}(sl(i,j))$.

The next step is to align the other minutiae by converting them into a polar coordinate system based on the corresponding pair (b_1,b_2). For minutia N, the polar coordinate is $M^p=(r,q,j)$, where

$$r=\sqrt{(x-x_b)^2+(y-y_b)^2}$$
$$q=diff(\arctan\left(\frac{y-y_b}{x-x_b}\right),j_b) \qquad (4)$$
$$j=diff(j,j_b)$$

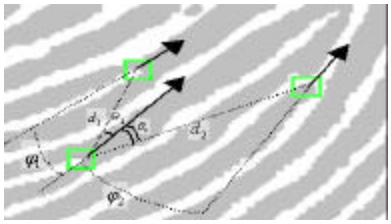

Fig. 5. Local feature of minutia.

Function $diff(\ )$ is the difference between two angles. Based on the aligned minutiae sets, the matching level of each minutia in the input fingerprint and each one in the template fingerprint can be calculated, following

$$ml(i,j)=\begin{cases}1-diff\_total/16, diff\_total<Bg\\ 0, otherwise\end{cases} \quad (5)$$

where $diff\_total=\left|M_I^p(i)-M_T^p(j)\right|_{W^p}$, Bg is a bounding box. In this paper $Bg=(8,p/6,p/6)$ and $W^p=(1,8,8)$.

To avoid one minutia being used more than once for matching, $ml(i,j)$ is set to zero if there is any k that makes $ml(i,k)>ml(i,j)$ or $ml(k,j)>ml(i,j)$. Afterwards, the final matching score can be achieved by:

$$Ms=100\times\frac{\sum_{i,j}ml(i,j)}{\max(p,q)} \quad (6)$$

V. System Analysis

Using the algorithm described above the fingerprint verification module is implemented on ThumbPod. The sensor used for fingerprint scanning has a relatively small area (13×13 mm^2), so the performance is strongly dependent on which part of the finger is captured. To eliminate this problem we introduce a two-template system in ThumbPod. The Thumb Pod system is tested with live-scan fingerprints. The image set consists of 10 fingerprints per finger from 10 different fingers, forming a total 100 fingerprint images. Each fingerprint is compared with every two other fingerprints and the two match scores are ported into a decision engine in order to get the final matching result. Totally, 7,200 decisions are made for the matched case and 81,000 decisions for the mismatched case. We have achieved 0.5% FRR (False Rejected Rate) and 0.01% FAR (False Accepted Rate).

VI. High-speed Optimization

Implementing the fingerprint verification module on a 50MHz LEON-2 embedded processor requires not only the performance, but also high speed and low power consumption. In this paper, we investigate both software and hardware optimization techniques to achieve this goal.

Software optimization aims to reducing the cycle number of the whole process. To get better performance, the first step is to find out the hot-points of the system. The TSIM SPARC simulator is used for profiling [2].

A. Profiling of the Minutiae Extraction

Fig. 6(a) shows the profiling result of the minutiae extraction. The execution time of the image binarization and the minutiae detection are 11% and 12% of the total, respectively. They are not considered a system bottleneck. However, the direction map deriving step (MAPS) occupies 74% of the total execution time. Therefore, the detailed algorithm for it is checked to speedup this process. Fig. 6(b) shows the instruction-level profiling of MAPS.

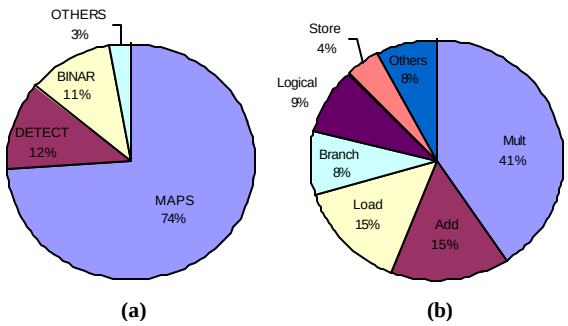

Fig. 6. (a) Profiling of the execution time for the minutiae extraction; (b) Instruction-level profiling of MAPS.

The numbers of instructions for multiply (Mult) and addition (Add) sum up to 56% of the total MAPS processing due to the repetitive DFT calculations for creating the direction map. Based on the profiling results, software optimization and hardware acceleration are considered for the DFT calculations in the direction map-deriving step.

B. Software Optimization for Minutiae Extraction

Studying the pattern of a fingerprint, we find that the neighboring blocks tend to have similar directions. An example of a direction map is shown in Fig. 7. This characteristic can be used to significantly reduce the number of DFT calculations. For instance, the first direction data, upper left in Fig. 7, is calculated using the same method as the original program. When deciding the direction of the right data, the DFTs for q = 4, 5, 6 are calculated first because the result is most likely to be q = 5. If the total energy for q = 5 is greater than both its neighbors (q = 4, 6) and a threshold value (E_{TH}), the direction of q = 5 is considered as the result. Otherwise, q is incremented or decremented until the total energy for q could have a peak with a greater value than E_{TH}. In other words, if the three conditions in (7) are met, the calculation of a direction value is finished. The execution speed as well as the matching error rate is measured when E_{TH} is changed from 10M to 35M. The results are shown in Fig. 8, from which it is found that when E_{TH} is larger than 20, the error rate is within an acceptable range.

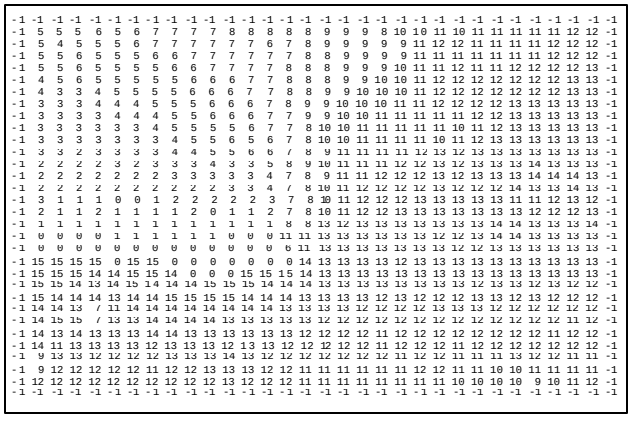

Fig. 7. Example of *Direction Map*. "-1" is no-direction because zero-padding in the image.

$$\sum_{k=1}^{4} E(k,q) > \sum_{k=1}^{4} E(k,q-1) \ [when q = 0, q-1 = 15]$$
$$\sum_{k=1}^{4} E(k,q) > \sum_{k=1}^{4} E(k,q+1) \ [when q = 15, q+1 = 0] \qquad (7)$$
$$\sum_{k=1}^{4} E(k,q) > E_{TH}$$

C. DFT Accelerator for Minutiae Detection

The software optimization reduces the number of DFT calculations and results in a significant speedup of the minutiae extraction. However, there are still a large number of DFT calculations, even if E_{TH} is set to a proper value. Therefore, DFT hardware acceleration is needed in addition to the software optimization.

The final specification of the accelerator only implements MAC computations for sine and cosine part separately (Fig. 9). In the multiply operation, Canonic Signed Digit (CSD) is used for saving hardware resources. The energy calculation part is not included because it needs a square operation of 16 bits data, which requires a general multiplier. As a result, the execution time of the minutiae detection is reduced to about 4 seconds as shown in Fig. 10.

D. Software Optimization for Matching

Compared to the minutiae extraction, the matching algorithm is much faster. However, this does not mean that there is no need to consider the high-speed optimization of this part. Thinking about extending the system from a one-to-one

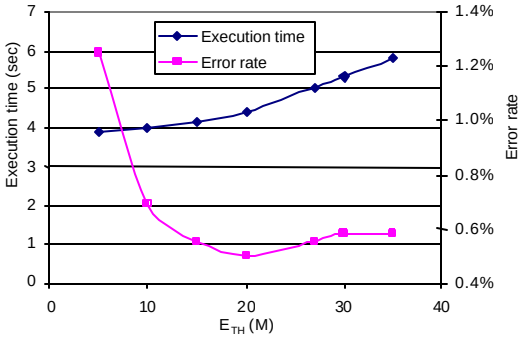

Fig. 8. Relation between E_{TH} and performance.

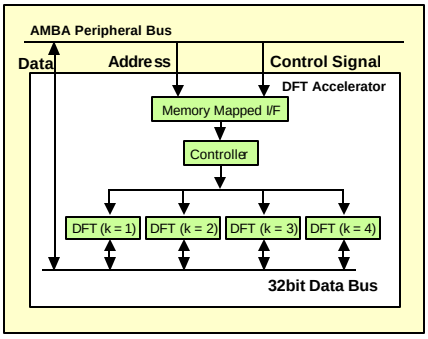

Fig. 9. Block diagram for the memory-mapped DFT accelerator.

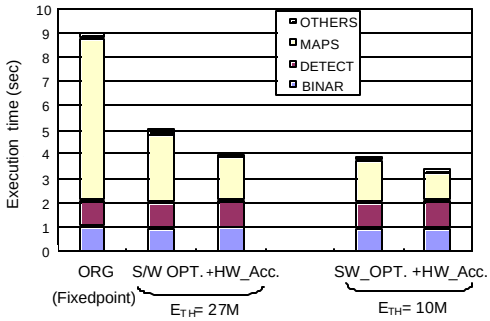

Fig. 10. Reduction of the execution time for the minutiae detection.

verification system to an identification system based on a big database, the matching process needs to be used a large number of times. In that case the speed for one-to-one fingerprint matching becomes critical in the system.

Analysis of the profiling result of the matching algorithm shows that a large part of the computation (52.2%) is used for finding the reference pair for the input image and the template image. The reason for this is that when trying to find out which pair is the reference pair, thorough search for each (i, j) pair is conducted, where $i = 1...p$, $j = 1..q$, p and q are the number of minutiae in input fingerprint and template fingerprint, respectively. In total, the similarity level $sl(i, j)$ needs to be calculated $p \times q$ times. To obtain all of these $sl(i, j)$, the local feature vector M for each minutia in the input fingerprint as well as the template fingerprint is required. However, detailed study of one typical case shows that 89% of the $sl(i, j)$ are zero, which means these pairs have totally different neighborhoods and by no means can become the reference pair. In the process of calculating the local feature vector M, the most time consuming part is finding the angles (q_1, q_2, j_1, j_2) between the minutia and its neighborhood. To make the matching system more efficient, for those (i, j) pairs whose $sl(i, j)$ are zero, an earlier decision should be made. A modified algorithm is implemented. The basic idea is that before calculating the local feature vector, one additional module called "Pre-Checking" is added, as shown in Fig. 11.

As mentioned before, the weighted distance between each pair of minutiae is $|M_I(i) - M_T(j)|_W$. In the Pre-Checking module, we define $W = W_d = (1,1,0,0,0,0,0,0,0,0,0,0)$, which means only the distance information is needed in this procedure. Only if the weighted distance $|M_I(i) - M_T(j)|_{W_d}$ is within a pre-set threshold $M_{TH} = A(W_d)$, the computation of the complete local feature vector is necessary.

Both the reduction of the computation time after adding the Pre-Checking module and the result degradation depend on the value of the threshold M_{TH}. The relationship between M_{TH} and the performance is presented in Fig. 12. In order to get high accuracy with small number of instruction cycles, $M_{TH} = 20$ is an appropriate choice, which leads to 26.1% reduction in the computation time.

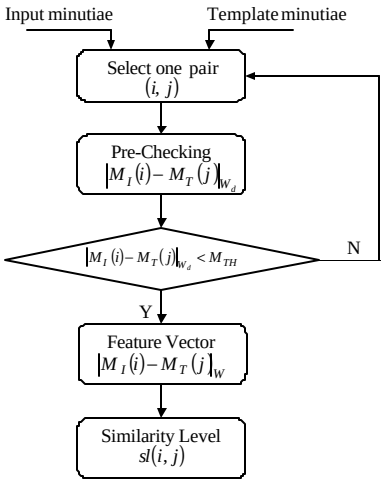

Fig. 11. Pre-checking process.

During the regular process of setting flags to the possible multiple-used matched pairs, one comparison loop with a size of $p \times q \times (p+q)$ is used, where p and q are the number of minutiae in the input and the template fingerprints, respectively. For a sample case where $p = 37$ and $p = 39$, the instruction cycle number to finish this process is 1.4M, which is 38.9% of the whole matching process. As mentioned before, $ml(i, j)$ is calculated from the local feature distance of the *i*-th minutia in the input fingerprint and the *j*-th minutia in the template fingerprint. For most of the pairs (i, j), the local feature vector is so different that it contributes nothing to the overall matching score. Based on this characteristic, the process of marking possible multiple-used $ml(i, j)$ can be optimized. Whenever the $ml(i, j)$ is zero, all the remainder comparison steps are skipped and the algorithm jumps to the next pair. After the above optimizations, the total cycle number is 1.34M. Hence the execution time for matching is reduced to 26.80ms (Fig. 13).

VII. CONLUSION

This paper describes an efficient fingerprint verification system, which includes both minutiae extraction and matching. The whole process is performed on a 50MHz fixed-point LEON-2 processor. By implementing the optimized minutiae extraction and matching algorithms, as well as the DFT hardware accelerator, we successfully reduce the execution time for the minutiae extraction from 9.04s to 4s, and for the matching process from 67.17ms to 26.80ms. This means that the whole process finishes in less than 5 seconds.

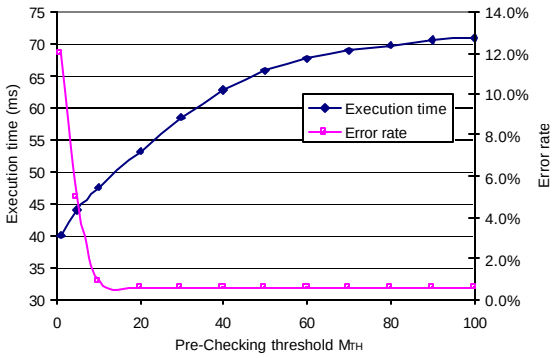

Fig. 12. Relation between Pre-Checking threshold M_{TH} and performance.

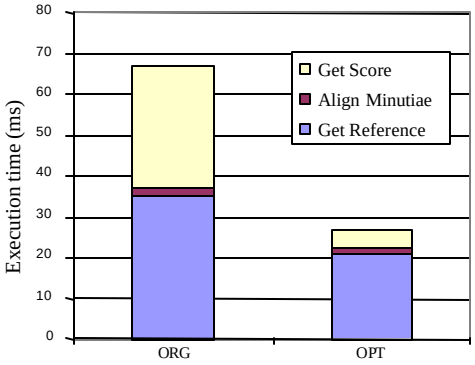

Fig. 13. Reduction of the execution time for matching

ACKNOWLEDGMENT

The Authors would like to thank all the teammates in the ThumbPod project [1]. We also thank Gaisler research for providing the Leon-2 Sparc core and for support in setting up the simulation environment.

REFERENCE

[1] http://www.ThumbPod.com
[2] http://www.gaisler.com
[3] User's Guide to NIST Fingerprint Image Software (NFIS). NISTIR 6813, National Institute of Standards and Technology.
[4] X. Jiang and W.-Y. Yau, "Fingerprint minutiae matching based on the local and global structures," *Proceedings of International Conference on Pattern Recognition (ICPR 2000)*, Sept. 2000, pp. 6038-6041.
[5] A. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, No. 4, Apr. 1997, pp. 302-314.
[6] J. W. Crenshaw, *Math Toolkit for Real-Time Programming*, CMP Books, Lawrence, Kansas, 2000.
[7] M. Mimura, S. Ishida and Y. Seto. "Fingerprint verification system on smart card," *International Conference on Consumer Electronics 2002*, pp.182-3. Piscataway, NJ, USA.
[8] Y. Gil, D. Moon, S. Pan and Y. Chung, "Fingerprint verification system involving smart card," *ICISC 2002: 5th International Conference*, Seoul, Korea, Nov. 2002.