Thinning Processor for 160x192 Pixel Array Fingerprint Recognition

Seung-Min Jung, *Member, KIMICS*

***Abstract*—A thinning algorithm changes a binary fingerprint image to one pixel width. A thinning stage occupies 40% cycle of 32-bit RISC microprocessor system for a fingerprint identification algorithm. Hardware block processing is more effective than software one in speed, because a thinning algorithm is iteration of simple instructions. This paper describes an effective hardware scheme for thinning stage processing using the Verilog-HDL in 160x192 Pixel Array. The ZS algorithm was applied for a thinning stage. The hardware scheme was designed and simulated in RTL. The logic was also synthesized by XST in FPGA environment. Experimental results show the performance of the proposed scheme.**

***Index Terms*— thinning, fingerprint, RTL, HDL, VLSI**

I. INTRODUCTION

The fingerprint is known to be the most representative biometrics for authentication of individual persons. Among all the biometrics, verification systems based on fingerprints are very popular both for historical reasons and for their proven performance in the field. Fingerprints are graphical flow-like ridges present on human fingers. The fact that fingerprints are unique to each person has been well established[1]. The recent expansion of trading by Internet and the needs to prevent unauthorized usage of private communication and information processing systems open new opportunities to authentication systems. In that sense, a portable fingerprint recognition system, especially a silicon-based sensor system could be an ideal candidate[1].

A fingerprint verification algorithm has two phases: enrollment and verification. In the off-line enrollment phase, an enrolled fingerprint image is preprocessed, and the minutiae are extracted and stored. In the on-line verification phase, the similarity between the enrolled minutiae and the input minutiae is examined. There are three steps involved in the verification process: 1) image preprocessing, 2) minutiae extraction, and 3) minutiae matching[2].

Fingerprint image obtained from a fingerprint sensor has the image distortion and noise in it. Gabor filter is strongly recommended to get improved image and reduce a noise level. It makes a smoothing image as shown in Fig. 1. It made the image more effective to extract the minutiae from it. Before it is applied to the image, frequency and direction of the image should be calculated to emphasize unique features and make ridge clear. A one-pixel wide skeleton image is generated by changing thick ridge pattern to one-pixel line, which is called thinning stage as shown in Fig. 1. Minutiae mean ridge ending or bifurcation of fingerprint images. After all the true minutiae of the image are detected, these features are stored in a pattern file.

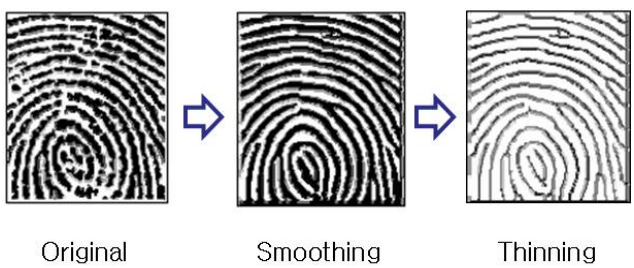

Fig. 1 Fingerprint image preprocessing stages

Fig. 2 shows a preprocessing of the conventional fingerprint identification. Fingerprint image noises need to be decreased, using a Gaussian low-pass filter, to correctly extract a print's ridge orientation and frequency [2]. Using a Gaussian low-pass filter eliminates some noise, but it is not enough. False estimation of orientation and frequency are found through disconnected ridges and noise dots. In order to solve this problem, an additional vector-type smoothing filter is needed. Moreover, the Gaussian low-pass filter is not a directional bandpass filter. Thus, there is a limitation in performance. A directional bandpass filter is essentially required in order to distinguish ridges from noise dots between ridges in corrupted but recoverable regions.

Manuscript received July 29, 2010; revised August 19, 2010; accepted August 27, 2010.

Seung-Min Jung is with the Department of Information Science and Telecommunications Eng., Hanshin University, Osan, 447-791, Korea (Tel: +82-31-379-0661, Mobile: +82-19-286-2571, Email: jasmine @hs.ac.kr)

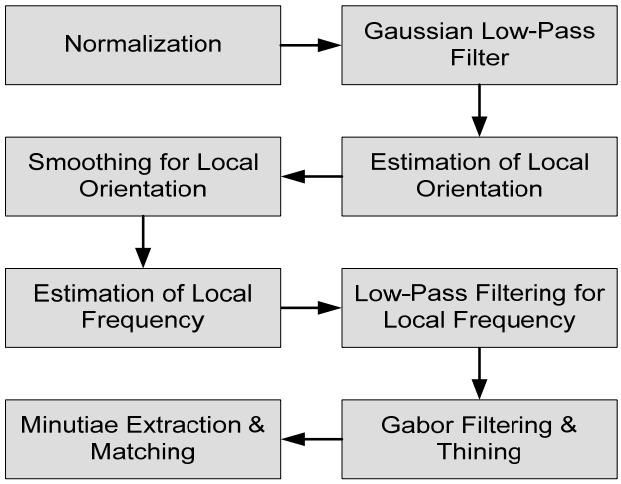

Fig. 2 Flowchart of general fingerprint recognition algorithm

Fig. 3 shows a conventional fingerprint authentication hardware and software system. Generally, the system needs a high performance microcontroller such as 32bit ARM7, or ARM9 CPU for completing the processing time below 1 second. All the results of algorithm instructions are compared with those of ARMulator, which is a software emulator of the ARM7 processor, at every algorithm cycle as shown in Fig. 4. The total clock number of an algorithm based on minutiae is 44.7M cycles and the operating speed was 1.12sec at 40MHz with a 160x192 sample image. GABOR filtering and thinning step occupies about 80% of total cycle. The thinning step occupies 39%(17.8M cycles). The processing time of thinning step is 0.445 second at 40Mhz. Minutiae detection time including matching is less then 3%. Hardware block processing is more effective than software in speed, because a thinning algorithm is iteration of simple instructions. This paper describes an effective hardware scheme for thinning stage processing using the Verilog-HDL in 160x192 Pixel Array. The ZS(Zhang and Suen) algorithm is applied for a thinning stage[3].

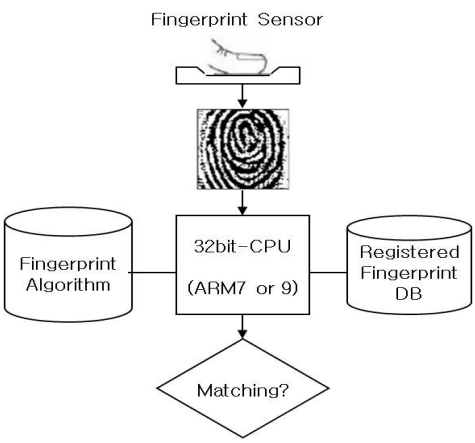

Fig. 3 Typical fingerprint identification system

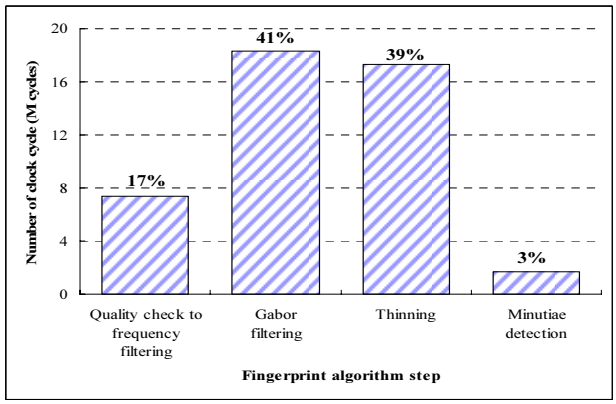

Fig. 4 Embedded 32-bit Microcontroller performance for algorithm

The paper is divided as follows: section II presents the overall thinning algorithm while section III followed by conclusions focuses on the proposed RTL implementation of tinning algorithm and simulation results.

II. THINNING ALGORITHM

In the past several decades, many thinning algorithms have been developed[3-6]. In order to reduce the quantity of information minimally, a thinning algorithm play an important role in recognition of fingerprint images. The well-proved thinning algorithms are ZS(Zhang and Suen). ZS parallel thinning algorithm performs sub-iteration step twice in 3x3 image pixel window as shown in Fig. 5. Here, P(i) and P1-8 are pixel binary image value. The value 1 means black and 0 means white.

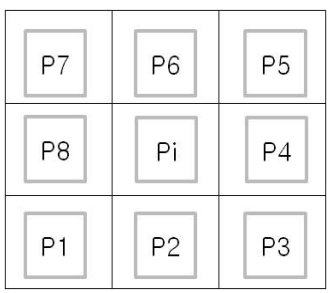

Fig. 5 P(3x3) : window pixel array

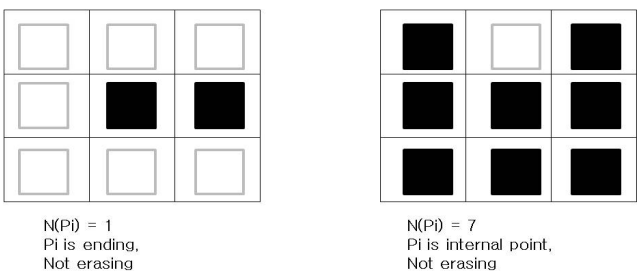

Fig. 6 Number of black pixel neighbor P(i)

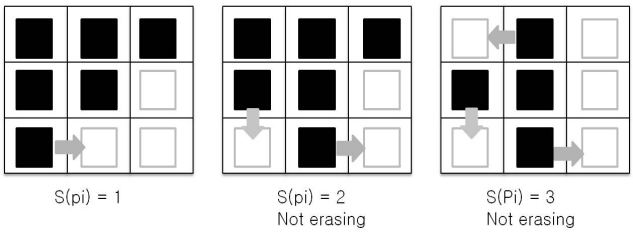

Fig. 7 Number of black to white change neighbor P(i)

The first step:

The pixels satisfied with following conditions are erased.

1. $2 \leq N(Pi) \leq 6$
2. S(Pi) = 1
3. P2*P6*P8=0
4. P4*P6*P8=0

Here, N(Pi) is the number of value 1 in 8-neighbor pixels of Fig. 6 and expressed in following equation (1).

$$N(Pi) = P1 + P2 + \cdots + P8 \qquad (1)$$

And, as shown in Fig. 7, S(Pi) means the number of 1 to 0 (1→ 0) patterns in 8-neighbor pixels.

The second step:

Condition 3 and 4 in the first step are replaced with the following conditions.

3'. P2*P4*P8=0

4'. P2*P4*P6=0

III. IMPLEMENTATION OF THINNING PROCESSOR

Iterations of 160x192 pixel array are performed for ZS algorithm in FPGA environment to verify proper operation. Fig. 8 shows the proposed thinning processor block diagram. The 160x192 couter generates 2 dimensional address for selecting fingerprint pixel array. The 3x3 pixel window gen block generates 8-neighbor pixels of P(i,j) for 3x3 window. The thinning stag1 and 2 blocks process first and second steps of ZS algorithm, respectively.

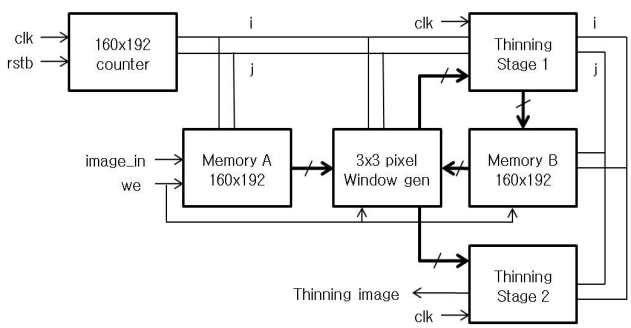

Fig. 8 Block diagram of thinning processor

```
1 `timescale 1ns/1ps
2 module thin_10 (p3x3,i, j, we, p_out, clk,);
3
4 output p_out;
5 input [0:8] p3x3;
6 input [cnt_width:0] i, j;
7 input clk, we;
8
9 reg [3:0] num_pixel_1;
10 reg [2:0] num_connect;
11 reg p468, p268 ;
12 reg p_out_tmp;
13 reg [cnt_width:0] j_tmp;
14
15 reg [0:data_width] mem2 [0:add_width];
16 reg [0:data_width] mem3 [0:add_width];
17
18 always @(posedge clk)
19 begin
20 num_pixel_1 = 0;
21 num_connect = 0;
22
23 if(p3x3[0] == 1) begin
24
25 if (p3x3[1] == 1) begin
26 num_pixel_1 = num_pixel_1 + 1 ; end
27 if (p3x3[2] == 1) begin
28 num_pixel_1 = num_pixel_1 + 1 ; end
29
30 if (num_pixel_1 >= 2 && num_pixel_1 <=6) begin
31 if (p3x3[1] == 1 && p3x3[2] == 0) begin
32 num_connect = num_connect +1; end
33 if (p3x3[2] == 1 && p3x3[3] == 0) begin
34 num_connect = num_connect +1; end
35
36 p268 = p3x3[2] & p3x3[6] & p3x3[8];
37 p468 = p3x3[4] & p3x3[6] & p3x3[8];
38
39 if (num_connect == 3'b1 &&
40 p268 == 0 && p468 == 0) begin
41 p_out_tmp = 0;
42 end else begin
43 p_out_tmp = 1;
44 end
45
46 assign p_out = p_out_tmp;
47
48 always @(posedge clk)
49 begin
50 j_tmp = j;
51 for (j_tmp = 0; j_tmp <= add_width ;
52 j_tmp = j_tmp+1) begin
53 if( j == j_tmp && i <= data_width) begin
54 mem2_tmp[i] = p_out_tmp;
55 mem2[j] = mem2_tmp; end
```

Fig. 9 RTL code of the thinning step 1

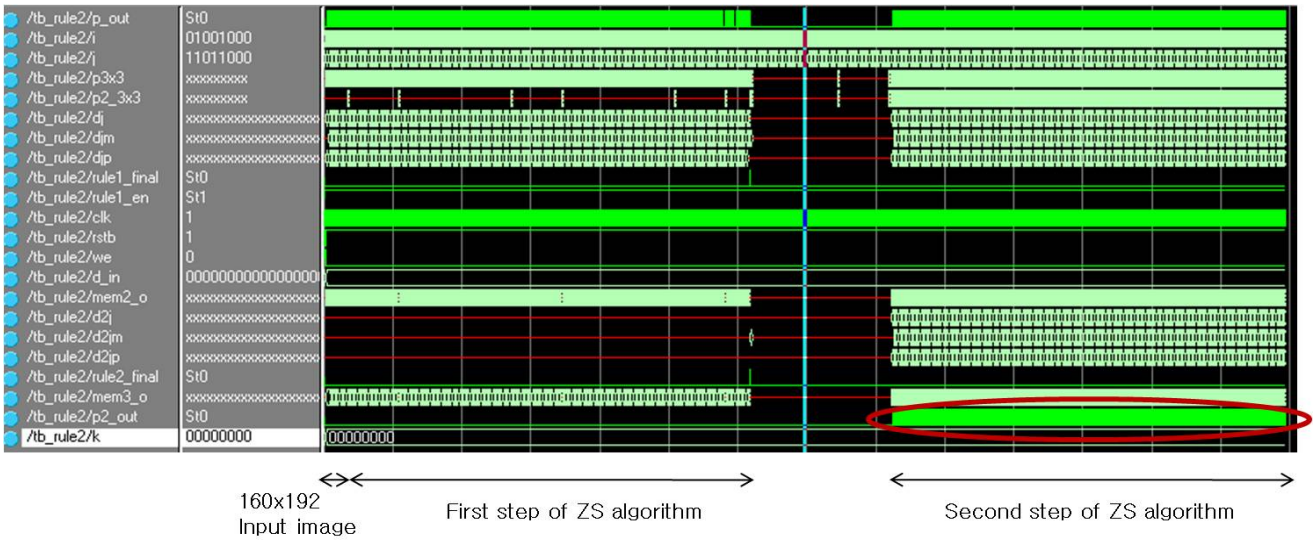

Fig. 10 Simulation result

Fig. 9 describes the RTL code of first step algorithm(Thinning stage 1 block in Fig. 8) by Verilog-HDL. The 9bit register 'p3x3' of line 5 describes 8-neighbor of P(i,j). Line 4 describes a serial output of first step processing and line 15-16 declares 160x192 memory, mem2 and mem3. Always-statement of line 18-44 describes 4 operations of ZS algorithm step1.

Fig. 10 shows a behavioral simulation result. The simulation was completed at 65000 clock cycles while step 1 and 2 processing. The result shows a valid operation of processor block. Fig. 11 shows the thinned image of 160x192 pixel sample. The left image is original sensor output data and right is simulation output of thinning processor. Partially, the results are not completed at bottom of center image, because of a various influences. It can be improved through the future works.

The logic was synthesized by XST(Xilinx synthesis tool) in FPGA environment. Fig. 12 shows schematic. It includes 6 top modules and sub modules of RTL code.

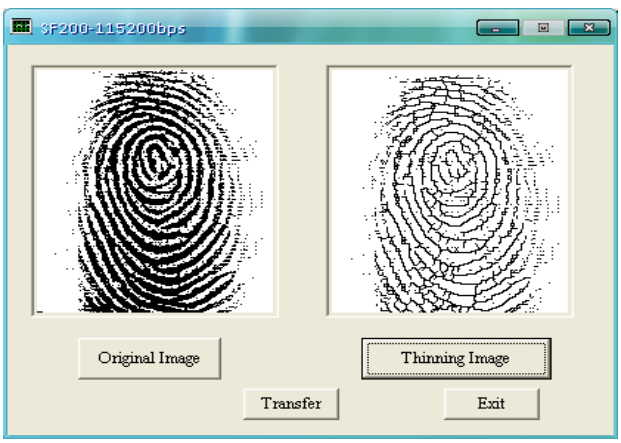

Fig. 11 160x192 sample image simulation result

The design was composed of 113,539 slices, 94,640 flip-flops, and 125,360 LUT with FPGA-XC4VLX200 including 2 memorys of 160x192. Synthesis report shows the thinning processor can be operated in minimum 14.3ns clock period. Maximum operating speed was 70MHz. The processing time of thinning step 1 and 2 is 16.2ms at 40MHz by total 65000 clock cycles. It means a thinning processing time can be reduced from about 400ms of a conventional system to 16.2ms, dramatically. Hardware burden will be relatively small on ASIC chip. This is one of the future works in this research. Fig. 13 describes the improved fingerprint system architecture including thinning processor. It can be implemented in a System on Chip(SoC). That is our future work and will be implemented in FPGA.

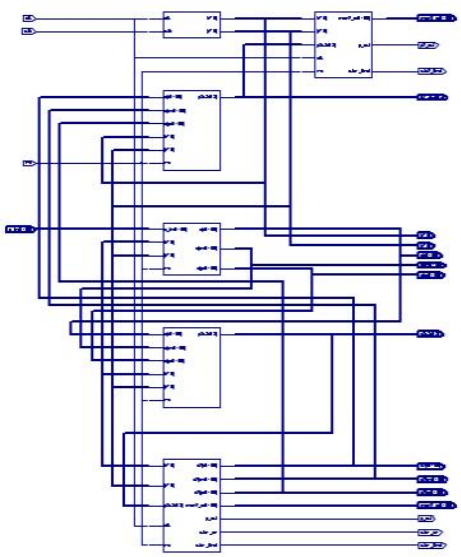

Fig. 12 Logic synthesis result

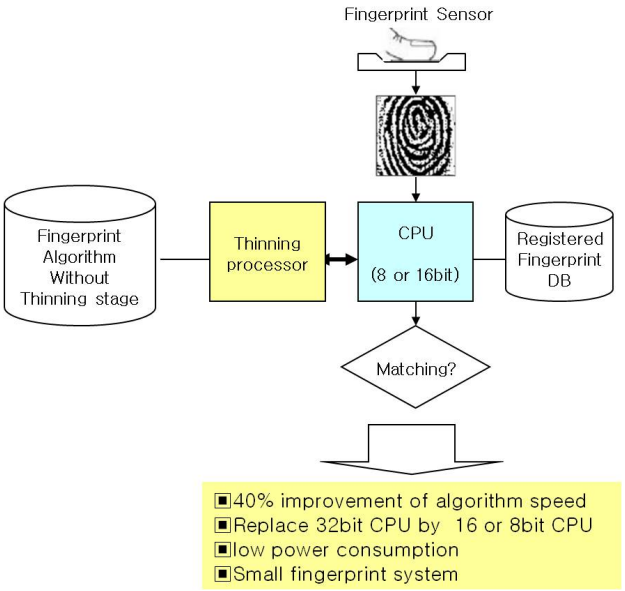

Fig. 13 Recommended fingerprint system

IV. CONCLUSIONS

This paper proposes an effective hardware scheme for thinning stage processing of a fingerprint identification algorithm based on minutiae with 40% cycle occupation of 32-bit RISC microprocessor. Conventional fingerprint authentication hardware and software system needs a high performance microcontroller such as 32bit ARM7, or ARM9 CPU for completing the processing time below 1 second. Hardware block processing is more effective than software one in speed, because a thinning algorithm is iteration of simple instructions. Proposed hardware scheme for thinning stage processing was designed using the Verilog-HDL in 160x192 Pixel Array. The ZS algorithm was applied for a thinning stage. The hardware scheme was designed and simulated in RTL. The logic was also synthesized by XST in FPGA environment. Synthesis report shows the thinning processor can be operated in minimum 14.3ns clock period. Maximum operating speed was 70MHz. The processing time of thinning step 1 and 2 is 16.2ms at 40MHz by total 65000 clock cycles. It means a thinning processing time can be reduced from about 400ms of a conventional system to 16.2ms.

In future work, it is planned to improve the output image quality and to optimize the architecture.

ACKNOWLEDGMENT

This work was supported by Hanshin University Research Grant.

REFERENCES

[1] Seung-Min. Jung, J.M. Nam, D.H. Yang, and M.K Lee “A CMOS Integrated Capacitive Fingerprint Sensor with 32-bit RISC Microcontroller,” IEEE Journal of Solid-State Circuits, vol. 40, no. 8, pp. 1745–1750, Aug. 2005.

[2] Lin Hong, Yifei Wan, and Anil Jain, "Fingerprint Image Enhancement: Algorithm and Performance Evaluation," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, pp. 777-789, August 1998.

[3] T. Y. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns”, Communications of the ACM, vol. 27, no. 6, pp. 236-239, Mar. 1984.

[4] Y. Y. Zhang, “Redundancy parallel thinning,” Pattern Recognition Letters, vol. 18, no. 1, pp. 27-35, Jan. 1997.

[5] Jun-Sik Kwon, “Obtaining 1-pixel Width Line Using an Enhanced Parallel Thinning Algorithm,” Journal of IEEK, Vol. 46- SP., No. 1, January 2009.

[6] Lei Huang, Genxun Wan, Changping Liu, “An Improved Parallel Thinning Algorithm,” Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), 2003.

Seung-Min Jung (M’02) Refer the (IJMICS) International Journal of Maritime Information and Communication Sciences, Vol. 6, No. 1, 2008. He is associate professor in the Department of Information Science and Telecommunications Engineering, Hanshin University, Osan, Korea.