Pipelined Gabor Filter Using Cordic Algorithm

MUKTHA S PATIL

ELECTRONICS AND COMMUNICATION , TONTADARYA COLLEGE OF ENGINEERING

Abstract

This project presents the improvisation of Gabor filter design using verilog HDL. This project details important enhancement made to the sizing problem and the coding style that synthesizable. The main characteristics of the proposed approach were to replace the parallel MAC to a serial MAC where the convolution matrix takes place. The CORDIC algorithm is a technique that can be used to compute the value of trigonometric functions. CORDIC differs from other methods of computation because it can be easily implemented using only addition, subtraction and bit shift operations. It is not as fast as table-based methods, but it can use significantly less chip area, making it desirable for application where area is more important than performance.

Key Words: Gabor, cordic, pipeline.

1. INTRODUCTION

The CORDIC algorithm is a technique that can be used to compute the value of trigonometric functions. CORDIC differs from other methods of computation because it can be easily implemented using only addition, subtraction and bit shift operations.
It is not as fast as table-based methods, but it can use significantly less chip area, making it desirable for application where area is more important than performance

This document shows the suggested format and appearance of a manuscript prepared for SPIE journals. Accepted papers will be professionally typeset. This template is intended to be a tool to improve manuscript clarity for the reviewers. The final layout of the typeset paper will not match this template layout.

2. Algorithm

The steps for CORDIC algorithm are:

- Get the angle and store it in Angle. Store the pre-calculated arctan values in an array.
- Assign X = 0.607252935 (i.e., X=T), Y=0
- Find X' and Y'
- If sign of Angle is positive then X = X - Y' Y = Y + X'
- else (If sign of Angle is negative) X = X + Y' Y = Y - X'
- Repeat steps (3) and (4) till the Angle approaches 0
- Print .Value of cos = . X
- Print .Value of sin = . Y
- Exit

2.1 CORDIC Hardware

A straight forward hardware implementation for CORDIC arithmetic[9] is shown
below in figure 2.2. It requires three registers for x, y and z, a look up table to store
the values of arctan(2-i) and two shifter to supply the terms 2−ix and 2−iy to the

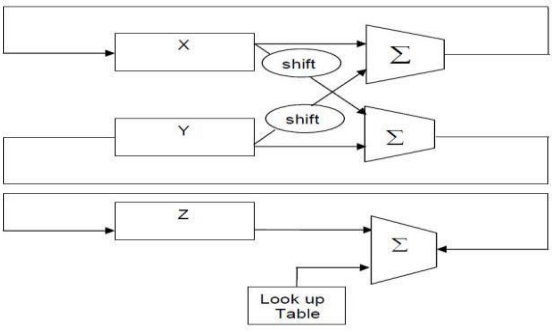


Figure 2.1: Hardware elements needed for the CORDIC method

It has a total of 13 stages, including 11 stages CORDIC iterations, and 2 stages adjustment before them. The architecture of the iterations is depicted in figure 2. 1. Values are point fixed to 11-bit, where MSB for sign bit and the lower 10-bit on behalf of the decimal bits, and range within [-1,1). But to ensure accuracy, before iterations, the input X0, Y0 are left-shifted 12-bit, meanwhile Z0 is left-shifted 4-bit. After 11 times

iterations, the Zout is close to 0. Take the high 11-bit of X11 and Y11, we can get the corresponding cosine and sine values.

MAIN USES

- REALIZATION OF ROTATIONS
- CALCULATION OF TRIGONOMETRIC FUNCTIONS
- CALCULATION OF INVERSE TRIGONOMETRIC FUNCTION tan□1(a=b)
- CALCULATION OF pa2 + b2, etc.
- EXTENDED TO HYPERBOLIC FUNCTIONS
- DIVISION AND MULTIPLICATION
- CALCULATION OF SQRT, LOG, AND EXP
- FOR LINEAR TRANSFORMS, DIGITAL FILTERS, AND SOLVING LIN. SYSTEMS

MAIN APPLICATIONS: DSP, IMAGE PROCESSING, 3D GRAPHICS, ROBOTICS.

3. Memory arrange

The memory block was used to store the image pixel. Discrete image convolution (or non-recursive FIR filtering) is one

of the most critical steps in digital image processingDue to computation flow transformation and
memory-logic integration, we were able to exploit the excessive bandwidth inherent in memory and achieve the fine-grain parallelism of computations. Estimations show that the architecture is
suitable for a real-time TV and HDTV picture convolution with very large kernels and can be implemented on a single VLSI chip. Algorithms involved in image recognition, correlation, enhancement, usually demand convolution by dynamic range kernels, with sizes varying from 2 x 2 and 3 x **3** (for the first order methods) up to 16x 16 or 24x24

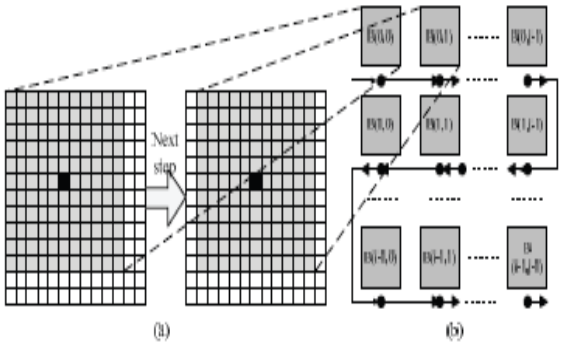


Figure 3 matrix convolution

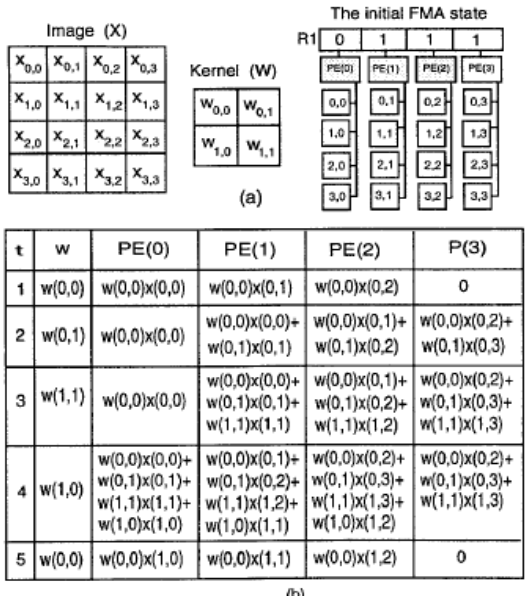
The Figure. 3 illustrates the matrix convolution to calculate the enhanced image pixel as Eq. 3 assuming Wg=11. In Figure. 3(a), each small box represents a pixel of fingerprint image. The gray 11x11 array is the convolution matrix, the central black box of which is the current pixel that should be convoluted. The Figure. 3(b) shows the sequence of pixels to be convoluted, and the gray boxes represent the overlapping convolution matrixes. In fact the "Z" shape memory [13]reading method can reduce the times of visiting the bus to get image pixel, since each adjacent matrix has 10x11 pixels same with each other.

3.1 The system operation

We assume that before processing, the first half of image is written into the FMA in a way that **x(i,j)** is stored in memory cell, Qi,j. Also, we assume that the kernel parameter *K* is set on register R1 through the following masking of the register bits:

$$R1(i) = \begin{cases} 1 & \text{if } 0 \le i \le (M-K) \\ 0 & \text{otherwise} \end{cases}$$

Figure 3.1(a) illustrates the pixel distribution in the memory array and initial state of the register R1 for a simple example of 4 x 4 image and 2 x 2 kemel. Let us assume for simplicity that all *B* registers in the **PES** be initially null.



t	w	PE(0)	PE(1)	PE(2)	P(3)
1	w(0,0)	w(0,0)x(0,0)	w(0,0)x(0,1)	w(0,0)x(0,2)	0
2	w(0,1)	w(0,0)x(0,0)	w(0,0)x(0,0)+ w(0,1)x(0,1)	w(0,0)x(0,1)+ w(0,1)x(0,2)	w(0,0)x(0,2)+ w(0,1)x(0,3)
3	w(1,1)	w(0,0)x(0,0)	w(0,0)x(0,0)+ w(0,1)x(0,1)+ w(1,1)x(1,1)	w(0,0)x(0,1)+ w(0,1)x(0,2)+ w(1,1)x(1,2)	w(0,0)x(0,2)+ w(0,1)x(0,3)+ w(1,1)x(1,3)
4	w(1,0)	w(0,0)x(0,0)+ w(0,1)x(0,1)+ w(1,1)x(1,1)+ w(1,0)x(1,0)	w(0,0)x(0,1)+ w(0,1)x(0,2)+ w(1,1)x(1,2)+ w(1,0)x(1,1)	w(0,0)x(0,2)+ w(0,1)x(0,3)+ w(1,1)x(1,3)+ w(1,0)x(1,2)	w(0,0)x(0,2)+ w(0,1)x(0,3)+ w(1,1)x(1,3)
5	w(0,0)	w(0,0)x(1,0)	w(0,0)x(1,1)	w(0,0)x(1,2)	0

Figure 3.1. Illustration **of** the convolution **process on example of** 4 x 4 image and 2 x 2 kernel

The convolution begins with reading the memory cells row $Q(i), k \le i \le (M)$ to the **PES** and sending the coefficient, w(O,O), to the register *Rw* (step 0). We assume that all *B* registers in the **PES** are initially null. In the next consecutive n clock cycles the Rw shifts 2 bits to the right broadcasting its **LSB** values to all the **PES.** During this time, each **PE** in the array multiplies its own pixel by the broadcasted values and adds the product to the partial sum, taken from its register *B.* Since **PES** with select lines ***c(i)*** = 0 compute zero products, the intermediate results will be changed only in the ***(A4*** - *K)* **PES,** shown by grey patterns. The first row in Figure 3.1(b) depicts the results saved in the registers B of the **PES** after the first computation phase. With each new coefficient, the register R1 is circularly shifted one bit to the right, and the computationalprocess is repeated. Thus during the *K* phases, the ***(A4*** - *K)* **x** ***(N*** - *H)* partial sums calculated in the first phase are iteratively moved along the rows *(K* - 1) times; each time accumulating the result computed at the FM they visit. When a new element of the kernel row enters the array, we read the next row of the Q cells and compute the results without shifting them between the **PES** (see the third row in Figure.3.1(b). Then we move in the opposite direction implementing the zigzag scan. As Figure 3.1(b) shows, the accumulation process is dynamic; unlike those of other architectures. The sum computed in the **PE(0)** in phase *t* = 1 travels the *(M - K)* **PES** in the zig-zag direction each time accumulating a new partial product. Generally, after *(K* x *H)* phases, all the kernel values are processed and all the *(M - K)* convolution results are available simultaneously in registers *B* of the **PES.** These results are then saved in the output buffer *2,* and the process is repeated again ***N*** - *H* times starting with reseting of registers *B.*

3.2 Convolution between image pixel and coefficient kernel

In this chapter it discusses on the method of convolution between coefficient kernels with the image data in the memory. It discusses on the flow of which image data will be
Convolute with the kernels and the next data to be convolute. This methodology was very important as it will determine on how the control logic unit need to control the flow of the data and the arithmetic process. Figure 2.3 may better explain on the flow of the convolution between image pixel and coefficient kernels.
From the figure 2.3, it well explains on how the matrix convolution took place. Firstly the pixel D11 is convolute with the kernel. The value of D11 after convolution is

D11=(D00xW11)+(D01xW12)+(D02xW13)+(D10xW21)+(D11xW)+(D12xW23)+(D20xW31)+(D21xW32)+(D22xW33).

The next pixel is
D12=(D01xW11)+(D02xW12)+(D03xW13)+(D11xW21)+(D12xW22)+(D13xW23)+(D21xW31)+(D22xW32)+(D23xW33).

The sequence of next pixel to be convolute is show in figure 3.2.

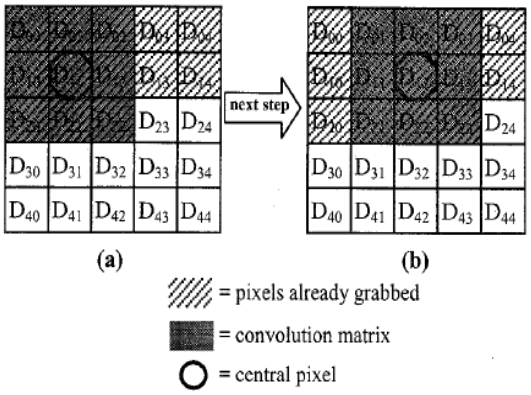
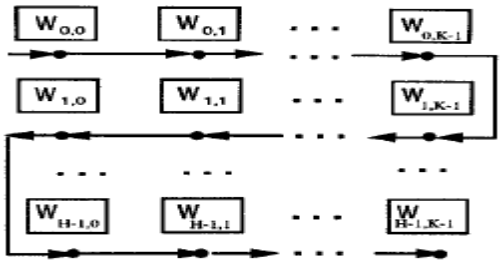


Figure3.2convolution sequence



Gabor Filter

In image processing, a Gabor filter, named after Dennis Gabor, is a linear filter used for edge detection. Frequency and orientation representations of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. The Gabor filters are self-similar: all filters can be generated from one mother wavelet by dilation and rotation.

Its impulse response is defined by a harmonic function multiplied by a Gaussian function. Because of the multiplication-convolution property (Convolution theorem), the Fourier transform of a Gabor filter's impulse response is the convolution of the Fourier transform of the harmonic function and the Fourier transform of the Gaussian function. The filter has a real and an imaginary component representing orthogonal directions. The two components may be formed into a complex number or used individually.

Gabor filters offer both frequency and orientation selective properties. It's appropriate to use Gabor filters as band-pass filters to remove the noise and preserve true ridge/valley structures of fingerprint. Making use of the frequency and orientation properties of the fingerprint image, the Gabor filter is defined by a cosine function multiplied by 2-D Gaussian function.

A Gabor filter is linear filter whose impulse response is defined by a harmonic function multiplied by Gaussian function. The Fourier transform of a Gabor filter's impulse response is the convolution of Fourier transform of harmonic function and the Fourier function of Gaussian function. This is the formula of the complex Gabor function:

$$g(x, y) = s(x, y)\, w_r(x, y)$$

4.2 The complex sinusoid carrier

The complex sinusoid is defined as follows,

$$s(x, y) = \exp(j(2\pi(u0\, x + v0\, y) + P))$$

where (u0, v0) and P define the spatial frequency and the phase of the sinusoid
respectively. We can think of this sinusoid as two separate real functions, conveniently allocated in the real and imaginary part of a complex function.

The real part and the imaginary part of this sinusoid are

$$\begin{aligned} \mathrm{Re}\,(s(x,y)) &= \cos\left(2\pi(u_0\, x + v_0\, y) + P\right) \\ \mathrm{Im}\,(s(x,y)) &= \sin\left(2\pi(u_0\, x + v_0\, y) + P\right) \end{aligned}$$

The parameters u0 and v0 define the spatial frequency of the sinusoid in Cartesian coordinates. This spatial frequency can also be expressed in polar coordinates as magnitude F0 and direction ω0:

$$\begin{aligned} F_0 &= \sqrt{{u_0}^2 + {v_0}^2} \\ \omega_0 &= \tan^{-1}\left(\frac{v_0}{u_0}\right) \end{aligned}$$

$$\begin{aligned} u_0 &= F_0 \cos\omega_0 \\ v_0 &= F_0 \sin\omega_0 \end{aligned}$$

Using this representation, the complex sinusoid is

$$s(x, y) = \exp\left(j\left(2\pi F_0\left(x\, \cos\omega_0 + y\, \sin\omega_0\right) + P\right)\right)$$

4.3 The Gaussian envelope

The Gaussian envelope looks as follows

$$w_r(x, y) = K\, \exp\left(-\pi\left(a^2\, {(x - x_0)_r}^2 + b^2\, {(y - y_0)_r}^2\right)\right)$$

Where (x0, y0) is the peak of the function, a and b are scaling parameters of the
Gaussian, and the r subscript stands for a rotation operation such that

$$(x-x_0)_r = (x-x_0)\cos\theta + (y-y_0)\sin\theta$$
$$(y-y_0)_r = -(x-x_0)\sin\theta + (y-y_0)\cos\theta$$

So the general function of Gabor filter can be represent as below

$$G(x,y,\theta,f_0) = \exp\left\{-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2}+\frac{y_\theta^2}{\sigma_y^2}\right)\right\}.\cos(2\pi f_0 x_\theta)$$

$$\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \begin{bmatrix} \sin\theta & \cos\theta \\ -\cos\theta & \sin\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

where θ is the ridge orientation respected to vertical axis, $f\theta$ is the selected ridge frequency in xθ – direction, σx and σy are the standard deviation of Gaussian function along the xθ and yθ axes respectively and the [xθ, yθ] are the coordination of [x,y] after a clockwise rotation of the Cartesian axes by an angle of (90-θ). Referring to the function in (1), this function can be decomposing into two orthogonal parts, one parallel and the other perpendicular to the orientation φ .

$$h(u,v;\varphi,f) = \exp\{-\frac{1}{2}\quad[\frac{u\varphi 2}{\delta u2}+\frac{v\varphi 2}{\delta v2}]\quad\}\cos(2\pi f u\varphi) \quad \text{.....(1)}$$

$$u_\Phi = u\cos\varphi + v\sin\varphi$$
$$v_\Phi = -u\sin\varphi + v\cos\varphi$$

Where φ and f are local ridge or valley's orientation and frequency respectively. δu and δv are the space constants of the Gaussian envelope along u and v axes, respectively, which both are set to 4.0 based on empirical data. To facilitate the computation and raise the memory efficiency, we use [-1,1) to represent [$-\pi$, π) for the input angle of CORDIC module. As a result, after some transformation, Eq. 1 can be modified as blow:

$$h(u,v;\varphi,f) = \exp\{-\frac{1}{32}r^2\}\cos(2l) \quad (2)$$

$$l = (fu)\cos\varphi + (fv)\sin\varphi$$

$$r^2 = u^2 + v^2$$

In digital signal processing, the output of the signal was the convolution between the input signals with the filter coefficient. So mainly, the digital filter circuit was the circuit to convolute the input signal with the filter coefficient. In digital image processing, the image was presented in matrix form or in pixel. So basically the convolution involves the matrix convolution-convolution between image pixels with coefficient kernel. Difference filter have difference method of filtering or sampling input signal. For this filter, it implemented a memory base architecture for real-time convolution with variable kernels[7]. Firstly the input data which was in pixel format will enter the filter and store it in the memory. The size of the memory was depending on the pixel size. If the pixel was 16x16 then the memory size would be 16x16 too. It means that every memory location will store for value for 1 image pixel. After the image had been stored in the memory then it would start the convolution process.

4.4 Configurable Pipelined Gabor Filter (CPGF)

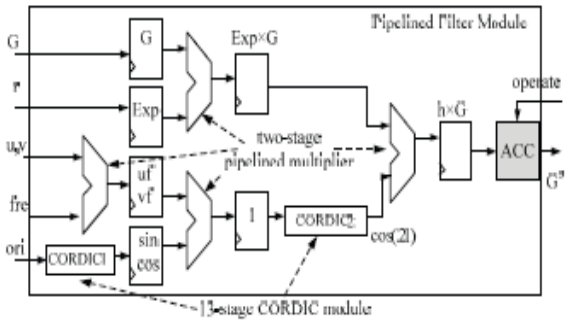
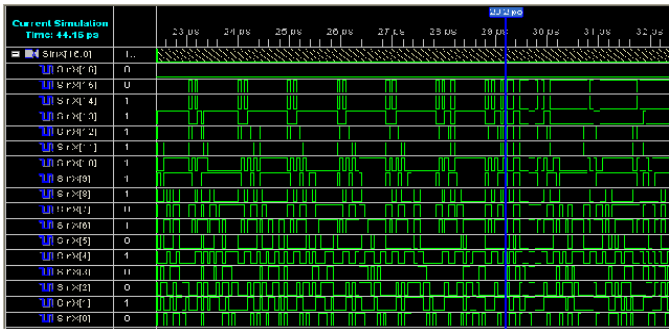


Figure 4. Pipelined Gabor Filter

It has counters to generate the address of the local frequency, orientation and image gray value. Configuration Register is used to store configuration information, such as the width and height of fingerprint image, convolution kernel size and so on. Second, the Controller can control the convolution process by using a FSM with three states.

The first state orders ACC (Accumulator) to accumulate. The second state let the ACC maintain the original value, because of the halt of the pipeline. The last State leads the ACC to writes back its value and then clear itself. Third, the PFM is dedicated hardware implementation of the matrix convolution described in Eq. 2 and 3. It reads in the required data from Register File, address of which is given by the Controller. Synchronization is achieved by controlling the sequence of the various addresses. Obviously, the critical path is the one involving with two CORDIC modules, thus several pipeline stages are employed to reduce the critical delay. Moreover, to exploit parallelism, the PFM

combines uf and vf while the CORDIC1 is calculating the sine and cosine of local orientation. And the multiplication of Exp and G will be performed in parallel with the calculation of cos(2l) in the CORDIC2. Exponential function to generate Exp results is implemented by look-up table, for the "r" of Eq. 2 has only a few possible values. Moreover since the Convolution kernel here is generated during the process rather than stored in ROM as a fixed kernel, the Flexibility is obtained. In fact, for different fingerprints or different area of a single fingerprint, we can use the most appropriate kernel to do the convolution to reduce the noise and enhance the ridge details. Furthermore before the enhancement we can configure the kernel size freely to achieve the best enhancement effect without any additional costs.



Simulation of sine30 result

Chapter 5: Integrated design testing

Block diagram

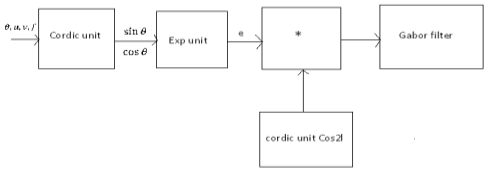
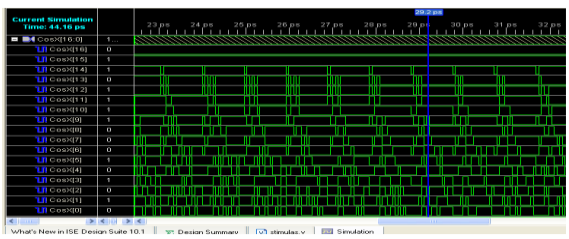


Figure 5.1 integrated block diagram

Figure 5.1 shows the integrated blocks of gabor filter. This block diagram consists of cordic unit , exponent unit , multiplier and is supplied with another cordic model and gabor filter. The inputs to the cordic model are θ,u,v,f where θ is angle and u,v are the orientations of the image and f is frequency . the output of the cordic unit is the sin and cos after some specified iterations (some iterations are given in the chapter 2) then these values are given to the exponent unit it generates the exponent of cordic function . output of the exponent is multiplied with another cordic function . the output of the multiplier is the desired gabor values.

Simulation Results



Simulation of cos30 result

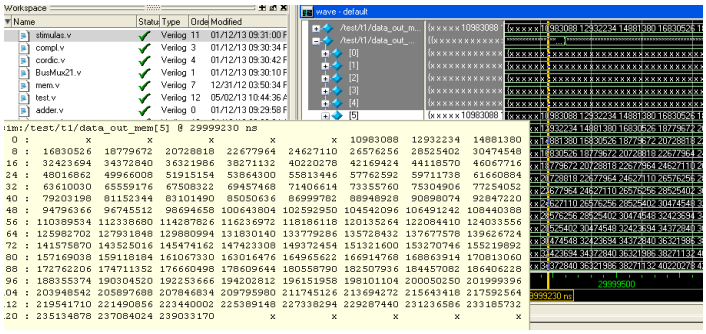


Figure 5.2 Gabor filter coefficients

Conclusion

Fingerprint recognition is an important biometric technique for personal identification. Before the minutiae extraction, fingerprint image enhancement is an essential step to ensure that the performance of an automatic fingerprint recognition system will be robust with respect to input fingerprint images with different quality. However, the delay caused by the high computational complexity of Gabor filter is the main obstacle for application. As a step with highly computational complexity, Gabor filtering accounts for a main part of the total processing time, more than 83% , among all the procedures of enhancement. So the hardware implementation of the Gabor filter is necessary and the recognition process will be significantly accelerated to satisfy the real-time requirement. Proposed a implementation of Gabor filter with a fixed 3x3 convolution kernel, which lacks the flexibility to adapt to the change of the local frequency and orientation of fingerprint image. Moreover the 3x3 kernel is too small to filter the noise. Set frequency to

1/6, and provided 8 options of orientation, which results in 8 5x5 filters. However their work is lack of accuracy, since in the area around the minutiae the frequency and orientation could be quite different from the other region. Reduced options of filters could reduce the complexity, but it has a bad effect on extracting feature points.

I implemented our novel Gabor filter with Verilog HDL and synthesize it by using Synopsys Design Compiler. The proposed design only occupies a small area of 63.8k equivalent gate count, but it can achieve the maximum operating frequency of 250MHz at SMIC 0.13um worst process corner. So, at most it only costs 31.7 ms that a fingerprint image of 256x256 pixels is enhanced by the proposed Gabor filter.

This paper presents a novel implementation of Gabor Filter for fingerprint image enhancement. The evaluation results show that the proposed design yields better performance, compared to other previous works. Moreover, its hardware cost is only 63.8k gate. Therefore, it is very suitable for the embedded fingerprint recognition system.

I presented a new memory-based architecture for two dimensional image convolution. Due to the convolution flow transformation and memory-logic integration, we were able to obtain a fine-grain computational parallelism and exploit the excessive bandwidth available within the memory array. The preliminary results show that the architecture ensures feasible solutions for the HDTV image convolution not only by parameterizable kernels but also by kernels **of** very large sizes (up to 19 *x* 19 coefficients). Future research will be dedicated to detailed chip design.

In future, maybe this filter can be run using full version software. Other aspect is to change the coefficient. This filter is reconfigurable filter. The coefficient can be change to suit to the implementation. This filter uses the previous work on Gabor algorithm. If later someone come up with improve Gabor coefficient than the coefficient can simply stored into the ROM of the filter.

References

[1] Y. W. Lin Hong, Anil Jain, “Fingerprint Image Enhancement: Algorithm and Performance Evaluation” , IEEE Transaction on Pattern Analysis and Machine Intelligence,vol. 20, NO. 8,(1998).

[2] A. H. A. Razak and R. H. Taharim, “Implementing Gabor Filter for Fingerprint Recognition Using Verilog HDL”, International Colloquium on Signal Processing & Its Applications. Pp.423-427, (2009).

[3] Lopez M. Canto E., Fons M., “Hardware-Software Co-design of a Fingerprint Image Enhancement Algorithm”, IEEE Industrial Electronics, (2006).

[4] Vincent Considine,“CORDIC Trigonometric Function Generator for DSP”, ICASSP, (1989).

[5] J. Volder, \The CORDIC trigonometric computing technique," IRE Trans. Electron. Compute., vol.EC-8, pp. 330{334, 1959.

[6] Pramod K. Meher, Javier Vallas, Tso-Bing Jung, K Sridharan and Koushik Maharatna, *50 Years of CORDIC: Algorithm, Architectures, and Applications*, IEEE Transactions on Circuits and Systems –I : Regular papers, VOL.56, NO.9, Sept- 2009.

[7] Vikas Kumar, *FPGA implementation of DFT using CORDIC algorithm*, Master’s thesis, Electronics and Communication Engineering Department Thapar University, Patiala, June 2008

[8] Tanya Vladimirova and Hans Tiggeler, *FPGA implementation of sine and cosine generator using the CORDIC algorithm*, University of Surry, Guilgford surrey, Gu2 5XH.

[9] B.Lakshmi and A.S. Dhar "CORDIC Architectures: A Survey" in *Hindawi Publishing Corporation*, VLSI Design,

[10] Gabor, D., 1946. Theory of communication. J. IEE 93, 429– 457.

[11] Daugman, J.G., 1985. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by twodimensional visual cortical filters. J. Optical Soc. Amer. 2 (7), 1160–1169.

[12] L. L. Jianwei Yang, Tianzi Jiang,Yong Fan, "A modified Gabor filter design method for fingerprint image enhancement," 2003.

[13] K. S. Vasily G. Moshnyaga, Keikichi Tamaru, "A Memory based architecture for real-time convolution with variable kernels," 1998.

[14] A. P. Arrigo Benedetti, Nello Scarabottolo, "Image Convolution o FPGAs:the implementation of a multi FPGA structure," 1998.