2-D IMPLEMENTATION OF DIGITAL GABOR FILTER DESIGN USING VERILOG

SUSHANTH K J[1] & SUSHANTH K J[2]

[1,2]Dept. of Electronics and Communication Sri Jayachamarajendra College of Engineering, Mysore, INDIA
E-mail : Sushanthkj@gmail.com & shankarsjce@gmail.com

Abstract - Fingerprint or Face Image enhancement using Gabor filter is one of highly computational complexity in fingerprint verification process. Gabor filter has a complex valued convolution kernel and a data format with complex values is used. So implementing Gabor filter is very significant in fingerprint verification process. Designing Gabor filter will help enhancing the quality of fingerprint image. In fingerprint recognition, Gabor filter optimally capture both local orientation and frequency information from a fingerprint image. By tuning a Gabor filter to specific frequency and direction, the local frequency and orientation information can be obtained. Thus, it is suited for extracting texture information from images. This paper presents the implementation of 2-D Gabor Filter design using Verilog HDL. This paper details important enhancement made to the 2D -Digital Gabor filter to minimize the sizing problem and the coding style that synthesizable. The intention is to study, analyze, simplify and improvise the design synthesis efficiency and accuracy while maintaining the same functionality. The result provides area efficiency architecture for the effective design.

Keywords **-** Digital filter, digital design, face recognition, Gabor filter, MAC, verilog HDL, Xilinx

I. INTRODUCTION

The Gabor filter (Gabor Wavelet) represents a band-pass linear filter whose impulse response is defined by a harmonic function multiplied by a Gaussian function. Thus, a bidimensional Gabor filter constitutes a complex sinusoidal plane of particular frequency and orientation modulated by a Gaussian envelope. It achieves an optimal resolution in both spatial and frequency domains.

Even though the design might compromise the speed, but the area consumption was reduced. The speed of serial design can be overcome by operate at a higher frequency.

In this article we are represent the 2-D digital gabor filter in MAT LAB, and design and implementation in verilog using Xilinx 10.1

1. Representation of 2-D gabor filter in MAT LAB.

A. Digital Gabor Filter

Gabor Filter was designed by transforming the designinto verilog using *xilinx 10.1*. The target device is Spartan 3Afamily. The figure shown below is the summary of the synthesized design. It can be seen that the utilization of the resource of the device exceeded 100%. This particular point was where the improvement needed to be done to achieve an effective and efficient design.

- **Texture analysis :** Texture is a fundamental property of natural images, and thus it is of much interest in the computer vision and computer graphics. Regular repetition of an elements or pattern of surface.
- **Frequency domain:** Gabor filter can be represented in frequency domain by using, the 2-D Gaussian functions.

Formally, a 2D Gabor filter in the spatial domain is defined by the following expression:

$$\psi_{v,\theta}(x,y) = \frac{f_u^2}{\pi\gamma\eta} e^{-\left(\frac{f_u^2}{\gamma^2}x'^2 + \frac{f_u^2}{\eta^2}y'^2\right)} e^{j2\pi f_u x'},$$

where x′ = x cos Θv +y sin Θv, y′ = Θx sin Θv +ycosΘv, and the parameters fu and Θv are defined as fu = fmax/2(u/2) and Θv = vΘ/8. As we can see, Gabor filters represent Gaussian kernel functions modulated by a complex plane wave whose center frequency and orientation are defined by fu and Θv, respectively. The parameters and Θ(theta) determine the ratio between the center frequency and the size of the Gaussian envelope.

Mainly 2-D gabor filter designed by using two aspects

- Filters in frequrncy domain:

- Filters in space domain:

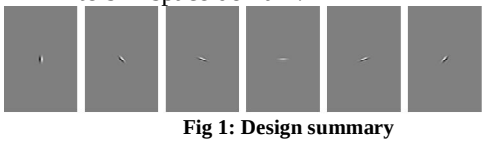

Fig 1: Design summary

Basically there were 3 major parts in the filter: CLU, ALU and MEMORY [4]. The 'convolution' signal indicates the operation of the filter. If the signal is high then the convolution process takes place. If it is low then the filter receives image input and stores it to the memory based on the input location. The data enters the filter pixel by pixel. The

'PIXEL_X' and 'PIXEL_Y' signal gave the address of the memory location.

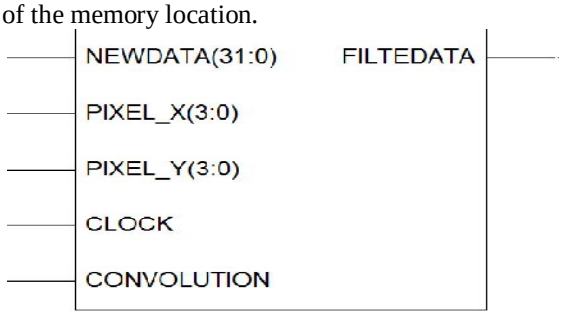

Fig 2: Top level

B. Arithmetic Unit

The main part of the digital gabor filter that is doing the convolution process. This is where the Gabor coefficient is stored[4]. It consists of 3 parts: ROM, DECODER and MAC. The ROM has 16 address locations but only 9 of it are used to store the coefficient. The MAC divided into 2 parts: multiplier and adder. The multiplier has 9 parallel multipliers. So the multiplication will be done in the same time as to speed up the convolution process. The adder consists of 8 adder connected in sequence. The adder is to sum up all the 9 multiplier outputs. Since the design uses 9 parallel multipliers and 8 adders, the design is significantly large. Both multiplier and adder use Xilinx IP cogen floating_point V3.0. This IP Cogen is generated from the Xilinx library.

II. METHODOLOGY

The focus of this work is not to design a new digital Gabor filter but to improve the design so it can be implemented on the device. As an ASIC designer, there are three major factors needed to be considered, maximization of speed, minimization of area and power consumption. In this work, minimization of area consumption will be the main priority.

A. Design

The design of the new multiplication-accumulation unit must be done precisely[6]. This is due to the sensitivity of the transition in a single data path. Below is the design flow of the filter.

Firstly, when the convolution signal is '0' the input data which is in pixel format will enter the filter and stored in the memory. The size of the memory depends on the pixel size. If the pixel is 16x16 then the memory size will be 32x32 too. It means that every memory location will be stored for value for 1 image pixel[4][10].

When the convolution signal is triggered to '1' the convolution process starts. The controller will read the image that is stored in the memory and send the data to the arithmetic unit. The controller will call the data from the determined memory location. In arithmetic unit there is also a ROM which will permanently store the coefficient kernel value. The value of kernel will also be called by the control into the convolution circuit. When both data has entered the convolution circuit the process of multiplication and accumulation will take place.

Only one series of data will be convoluted at a time. The counter will count for 9 convolution operation before giving out the result of filtered image.

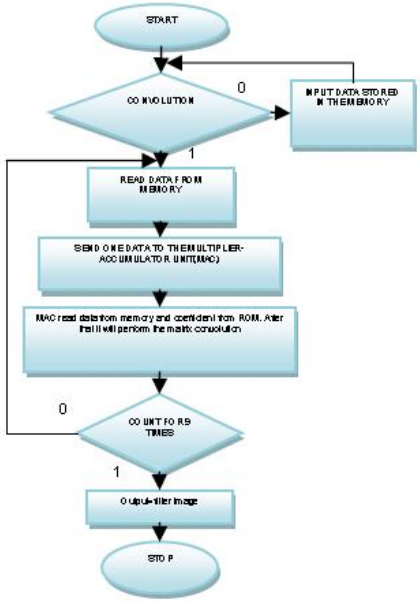

Fig 3 : Gabor filter flow

The reason why count for 9 consecutive cycle stands for the 9 coefficient kernel value. This will also be the result of the filter.

III. RESULT AND DISCUSSION

MAT LAB RESULTS:

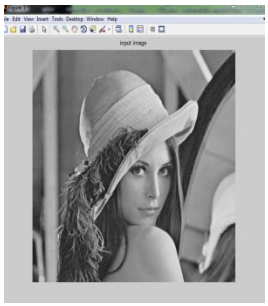

Fig 3: Test image

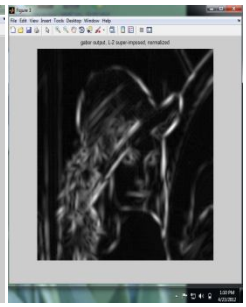

Fig 4: Gabor image

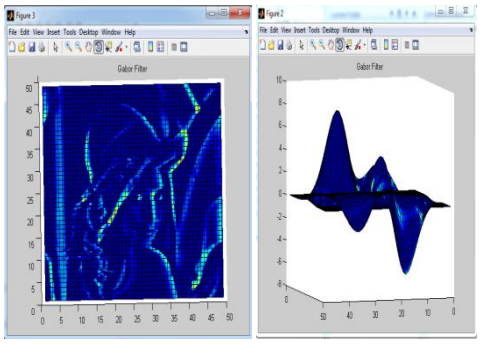

Fig 5: Gabor filter with 3-D view

The above MAT LAB results are just design and representation of digital gabor filter.

1. Design and implementation of 2-D gabor filter in verilog.

Redesigning the gabor filter in verilog using Xilinx 10.1 software, the code was then synthesized. The summary of the design was shown in figure 6. From the summary, the numbers of warnings were reduced from 92 to 21 warnings. The warnings generated are related to the incomplete if and else statement which a latch might be generated. In this summary, the target device Spartan3-S200 was used. This device contains large resources suitable for a design such as this. The numbers of Slices, Slice Flip Flops and LUTs were reduced.

RESULTS IN VERILOG:

- Below table values and waveform window from the Xilinx are the expected results when implemented in verilog

FIG 6: Design Summary

A. Top level

The above figure 7 shows the schematic view of the top level filter. There were 6 input pins and one output pin on the top level. New data stands for an unfiltered 32-bits image data. Pixel-X and Y hold the position of the memory when the write memory occurred. Clock and reset pins indicates the generated clock with 40ns period and reset button for the filter. The 'convolution' signal is to indicate the operation of the filter. If the signal is high then the convolution process takes place. If it is low then the filter receives image input and stores it to the memory based on the input location.

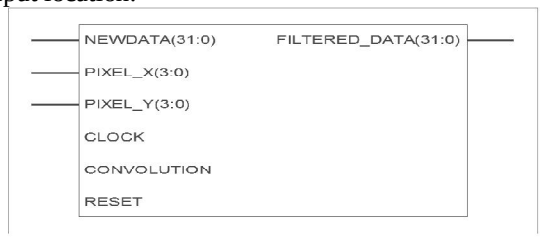

Fig 7: Toplevel

Output result for the top level filter is real convoluted data of filter is 0.006764772(3BDDAB06) but the expected result in figure 8 was 0.006764705(3BDDAA75). The difference was 0.00000068. The error was only 0.001%. This new design verifies that even though the multiplication and accumulation were design in serial, it can still give and maintain the same result from previous parallel design. It took 222 cycles to finish the convolution process in serial design.

IMAGE	COEFF	IMAGE*COEFF		SUM OF IMAGE*COEFF	
1	0.006737943	0.006737943	3BDCC9F6		
2	1.29E-05	2.57E-05	37D7F06A	0.006763677	3BDDA1D5
3	-4.00E-08	-1.21E-07	B402751D	0.006763556	3BDDA0D1
4	2.36E-07	9.43E-07	357D0CF5	0.006764499	3BDDA8BA
5	4.41E-08	2.07E-07	345E42E6	0.006764706	3BDDAA77
6	1.45E-12	8.69E-12	2D16D7D0	0.006764706	3DDDAA77
7	-1.36E-11	-9.51E-11	AED12154	0.006764705	3BDDAA75
8	2.65E-14	2.12E-13	2A6FF220	0.006764705	3BDDAA75
9	8.53E-17	7.68E-16	265D5A7A	0.006764705	3BDDAA75
		0.006764705	3BDDAA75		

Fig 8: Real convolution data

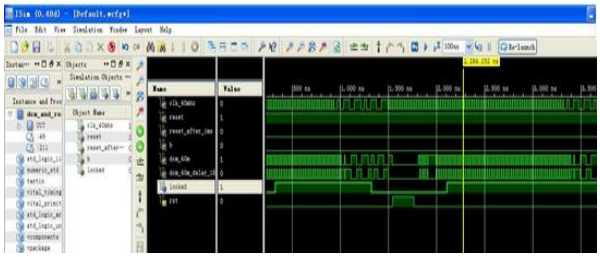

Fig 9: Verification of top level filter

B. Controller (CLU)

The control logic unit functions as controller for the data flow in the filter. It gives instruction to the other blocks to do their job. Basically, it gives the memory address to read data to the MEMORY and give address of coefficient to the ALU.

This CLU will only generate the address location when the 'START' signal is high. This signal indicats the convolution process that has taken place but if the signal is low, it indicates that the writing of image data into the memory takes place.

This CLU contains only 2 different blocks. One is the counter for the coefficient and memory address, and the other one is the counter decoder. The design of the counter gives the relationship between the coefficient and memory address. When the coefficient address was counted up until 9, the memory address for Y- direction will count a plus one. And the X-direction address must wait until Y-direction counts until 16 then it counts a plus one.

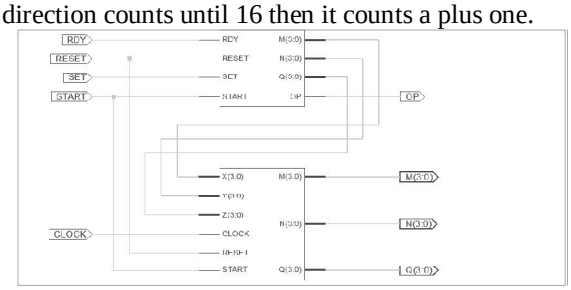

Fig 10: Controller

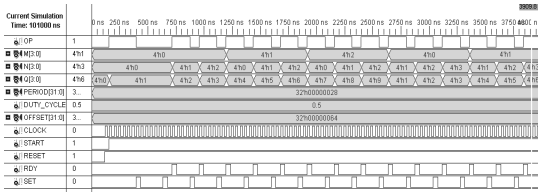

Fig 9: CLU Verification

C. Memory

The memory block is used to store the image pixel. The decoder only decodes address for Y-direction only. The clock was removed from the decoder so the decoded Y-direction can arrived at the same clock cycle with the X-direction. The adress for X-direction is supplied directly from the CLU or from the filter input. The image input is also connected directly from the filter input. The writenable signal indicates whether the operation is a write data or read data.

From the figure 10, first the 'WRITENABLE' signal is high to indicate the writing process is taking place. Then the signal goes low to read the data in the memory. The memory will give the output on the same clock cycle as the address location enters.

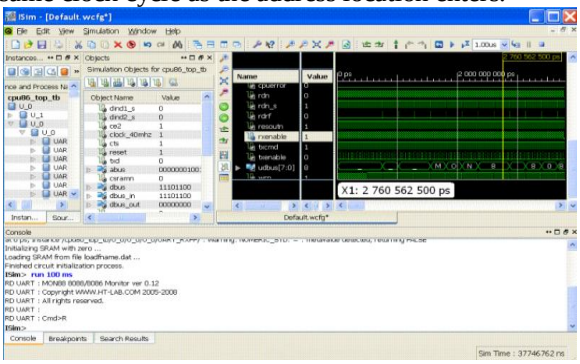

Fig 11: Verification memory unit

D. Arithmetic (ALU)

This is the main part of this work, arithmetic unit. This is where the convolution process takes place. It consists of two parts: the ROM and the MAC. The ROM is used to store the 9 coefficient values that are needed to convolute with the image while MAC consists of a buffer, a multiplier, an adder and a counter. The crucial part of this design was to make sure that the convolution process happened align with the correct image data and coefficient. The 'CONVO' signal plays important role to ensure there was no mismatch of data read.

From the figure 11 below, the 'CLOCK' and the 'CONVO' both were connected to the ROM and MAC. When the 'CONVO' went from low to high, the convolution process starts. The feedback 'READY' and 'SET' were sent to the CLU indicates convolution process completed. The CLU then will push the 'CONVO' signal from high to low before the next convolution takes place. These processes take 9 complete convolutions before sending the convoluted data out.

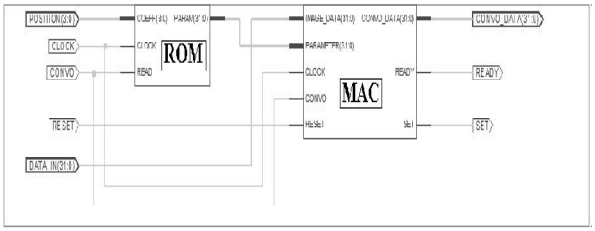

Fig 12: Arithmetic

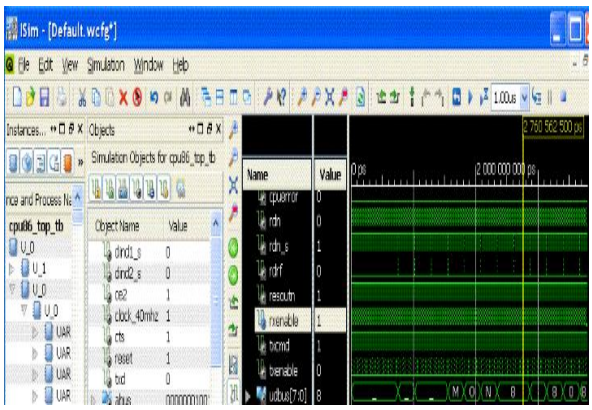

Fig 13: Verification of ALU

The buffer was used to hold the 'CONVO' operation for 1 cycle before the multiplier. The intention was to wait for the correct data sent from the memory for the convolution process. The Multiplier and the adder are connected in series. The design was done in such a way to lessen the area consumption of the filter. After 9 consecutive multiplications and additions, Since the design is a single data path also known as pipeline, the multiplication and the addition will take a longer period of time. The total cycles required for convolution for this design is 222 clock cycles with a time period of 40ns per cycle.

IV. CONCLUSION

The design and implementation of digital Gabor Filter has successfully reached. The area of the design has been significantly reduced while the function of the filter is perfectly maintained.

The numbers of slices used from previous design reduce from 5759 slices to 1625 slices. This significant change is due to the reduction of multiplier and adder used in the multiplication and accumulation unit. The enhancement made in the multiplication-accumulation unit has been proven effectively reliable and functional.

By adjusting the memory and the controller unit, the functionality of a complete and correct digital Gabor Filter is obtained. Even though, the precision of this Gabor Filter is 0.001% away from the calculated data. By minimizing the area, The result provides area efficiency architecture for the effective design.

REFERENCES

[1] A.D. Jepson and D.J. Fleet, "Image Segmentation" , Internet Draft 2007.

[2] Dennis Dunn, William E. Higgins, and Joseph Wakeley, Texture Segmentation Using 2-D Gabor Elementary Functions, IEEE TRANSACTIONS ON PAITERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 16. NO. 2, FEBRUARY 1994 : pp. 130-149.

[3] Razak, A.H.A. Taharim, R.H. "Implementing Gabor Filter for Fingerprint Recognition using verilog HDL," IEEE explorer , March 2009.

[4] P. H. W. L. Ocean Y. H. Cheung, Eric K.C. Tsang,Bertam E.SHi, "Implementing Of Gabor-type Filters on Field Programmable Gate Arrays," 2005.

[5] Heikkilä M, Pietikäinen M & Schmid C (2009) Description of interest regions with local binary patterns. Pattern Recognition 42(3): 425.436.

[6] Heisele B, Ho P, Wu J & Poggio T (2008) Face recognition: component-based versus global approaches. Compter Vision and Image Understanding 91(1.2): 6.21.