Design of a Modified Gabor Filter with Vedic Multipliers Using Verilog HDL

Naheean Rahim, Shamayla Islam, and Iqbalur R. Rokon

***Abstract*—Gabor Filters are widely used in all kinds of image processing. Gabor Filters include a memory, a controller and an arithmetic logic unit. The Gabor Filter designed in this project has a RAM type Memory, but a few changes were made in the Controller and the Arithmetic Logic Unit (ALU). The Arithmetic Logic Unit had a new type of multiplier called a Vedic Multiplier. So building a Gabor Filter with Vedic Multipliers is something that we have introduced in this paper. Using Vedic Multipliers, our filter was made faster without affecting the functionality of filter. The project included two phases where we did simulation of the Verilog Codes and synthesis of the whole Gabor Filter. For simulation and coding modules with Verilog HDL, we used ModelSim-Altera 6.5b (Quartus II 9.1) Starter Edition. For synthesis of the units and examining RTL schematic diagrams, we used Xilinx ISE 9.2i.**

***Index Terms*—Gabor filter, MAC, vedic multipliers, verilog HDL, Xilinx.**

I. Introduction

Image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image processing techniques involve treating the image as a two dimensional signal and applying standard signal processing techniques to it [1].

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 [2].

Some of the implementations of Gabor Filters include fingerprint recognition, palm print recognition and facial recognition. It is also used in biomedical applications like processing kidney images and ultrasound images.

The Gabor Filter has been modified in a lot of ways in recent years like making it area efficient or trying to make it faster by bringing changes in the Multiplier and Accumulator unit (MAC). For instance, parallel multipliers were exchanged with series multipliers in the MAC to make it more area efficient whereas, Vedic Multipliers have not been used before.

Manuscript received December 1, 2014; revised February 2, 2015.

Naheean Rahim and Shamayla Islam are with the Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh (e-mail: naheean_rahim@yahoo.com, shamayla_91@hotmail.com).

Iqbalur Rahman Rokon is with North South University, Dhaka, Bangladesh. He was with VLSI Chip Research and Development (R&D), Emulex Corporation, California, USA (tel.: +88-01726246189; e-mail: irahman@northsouth.edu).

A. Gabor Filter

The contents of our version of the Gabor Filter include three blocks inside the Top Level block. The three parts are the Control Logic Unit (CLU), the Arithmetic Logic Unit (ALU), and the Memory. The Memory is a RAM type memory, the CLU is simply a controller and the ALU contains the new addition to our design which is the Vedic Multiplier.

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 [3].

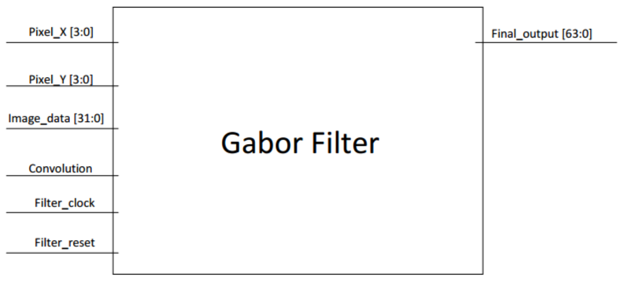


Fig. 1. Top level diagram.

B. Arithmetic Logic Unit (ALU)

The ALU has three blocks: the MAC block, the ROM block, and the Convolution Signal Buffer. Inside the MAC block resides three more blocks: the Data Control Buffer, the Vedic Multiplier, and the Accumulator.

Vedic Mathematics has been used in this work by using a Vedic Multiplier. The speed of MAC greatly depends on the multiplier. It enables parallel generation of intermediate products, eliminates unwanted multiplication steps with zeros and scaled to higher bit levels using Karatsuba algorithm with the compatibility to different data types. MAC is an extensible block using the Vedic Multiplier module plays an important role in computing especially digital signal processing.

C. Vedic Multiplication

Vedic mathematics is mainly based on 16 Sutras (formulae) which deals with various branches of mathematics like arithmetic, algebra, geometry, etc [4], [5]. Jagadguru Shankaracharya Bharati Krishna Teerthaji Maharaja (1884-1960) comprised all this work together and gave its mathematical explanation. After extensive research in

Atharva Veda Swamiji constructed 16 sutras (formulae) and 16 Upa sutras (sub formulae). Vedic mathematics deals with several basic as well as complex mathematical operations. The methods of Vedic mathematics are extremely simple and very powerful. One of the methods include the Urdhva Triyagbhyam.

The work has proved the efficiency of Urdhva Triyagbhyam Vedic method for multiplication which strikes a difference in the actual process of multiplication itself. It enables parallel generation of intermediate products, eliminates unwanted multiplication steps with zeros and scaled to higher bit levels using Karatsuba algorithm with the compatibility to different dat a types. These Multipliers have an important effect in designing arithmetic, signal and image processors. Many mandatory functions in such processors make use of multipliers (for example, the basic building blocks in Fast Fourier transforms (FFTs) and multiply accumulate (MAC) are multipliers) [6].

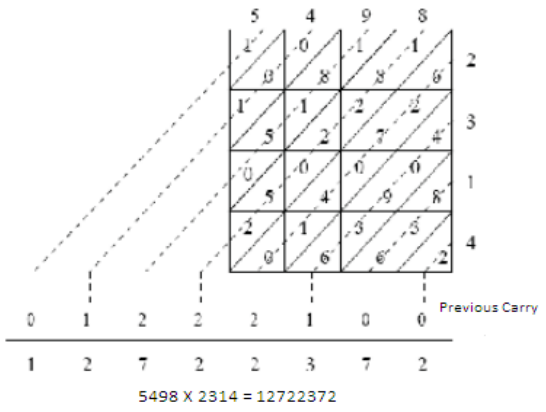


Fig. 2. Vedic multiplication.

Urdhva - Triagbhyam Sutra is a general multiplication formula applicable to all cases of multiplication. It literally means "Vertically and Crosswise". To Illustrate this multiplication scheme, let us consider the multiplication of two decimal numbers (5498×2314). The conventional methods already known to us will require 16 multiplications and 15 additions. An alternative method of multiplication using Urdhva - Triagbhyam Sutra is shown in Fig. 2.

The numbers to be multiplied are written on two consecutive sides of the square as shown in the figure. The square is divided into rows and columns where each row/column corresponds to one of the digit of either a multiplier or a multiplicand. Thus, each digit of the multiplier has a small box common to a digit of the multiplicand. These small boxes are partitioned into two halves by the crosswise lines. Each digit of the multiplier is then independently multiplied with every digit of the multiplicand and the two-digit product is written in the common box. All the digits lying on a crosswise dotted line are added to the previous carry. The least significant digit of the obtained number acts as the result digit and the rest as the carry for the next step. Carry for the first step (i.e., the dotted line on the extreme right side) is taken to be zero [6].

II. Previous Works

The Gabor Filter has previously been designed to be area efficient. No previous record of redesigning the MAC was found where the digital Gabor Filter was made faster. Vedic Multipliers have previously being used in MAC units that resides inside the Arithmetic Logic Unit.

III. Methodology

The focus of this work is to improve the design of a digital Gabor Filter where the maximization of speed will be the main priority.

A. Design Flow

The design flow of the Gabor Filter is shown in Fig. 3.

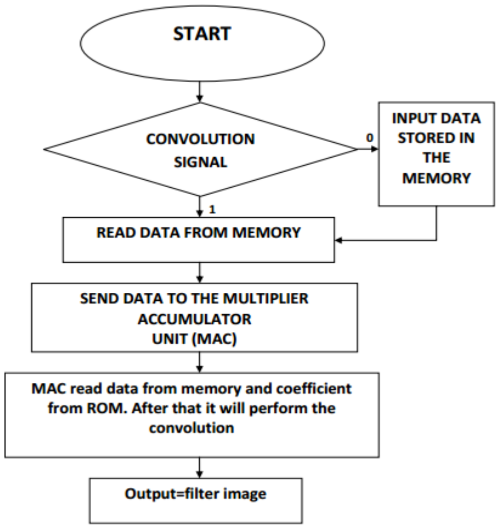

START
CONVOLUTION SIGNAL
0
INPUT DATA STORED IN THE MEMORY
1
READ DATA FROM MEMORY
SEND DATA TO THE MULTIPLIER ACCUMULATOR UNIT (MAC)
MAC read data from memory and coefficient from ROM. After that it will perform the convolution
Output=filter image

Fig. 3. Design flow of the gabor filter.

When a data is entered, it enters pixel by pixel into the Memory block. The memory location where it is stored is given by Pixel_X and Pixel_Y. The "Convolution" signal indicates the operation of the filter. If this signal is high (1), it means convolution has started. If this signal is low (0), the data entered is written in the memory location. When the "Convolution" signal is high, the Controller will read the image that is stored in the memory and send the data to the MAC unit of the Arithmetic Logic Unit. The Controller will call the data from the determined memory location. The Arithmetic Logic Unit consists of a ROM which is holding the Kernel Coefficient value. The address of Kernel Coefficient will also be generated by the Controller. The Kernel Coefficient is sent to the MAC unit from the ROM. Both the image data and Kernel Coefficient enter the MAC and multiplication and accumulation process starts taking place. Only one series of data will be convoluted at a time [7]. Since there are 9 Kernel Coefficients, there will be 9 convolution operations. Therefore, 9 image data and 9 Kernel Coefficients convolute and accumulate. The result is the filtered output.

The contents of the three blocks Control Logic Unit, Arithmetic Logic Unit and Memory are discussed below.

IV. Experimental Results and Discussion

After designing the Gabor Filter using Verilog HDL in the ModelSim-Altera 6.5b (Quartus II 9.1) Starter Edition, the code was then synthesized in Xilinx ISE 9.2i.

A. *Top Level*

Fig. 1 shows the schematic view of the top level filter. There were six input pins and one output pin on the top level. “Image_data” stands for an unfiltered 32-bit image data. “Pixel_X” and “Pixel_Y” contained the location of the memory where the “Image_data” is to be written. Filter_clock and Filter_reset pins indicate the generated clock and reset button for the filter. The “Convolution” signal indicates the operation of the filter. If the signal is high (1), the convolution process takes place. If the signal is low (0), the memory in the filter receives image input and stores it on the location specified by “Pixel_X” and “Pixel_Y”. Fig. 4 below shows the detailed block diagram which shows the inner structure of the Gabor Filter.

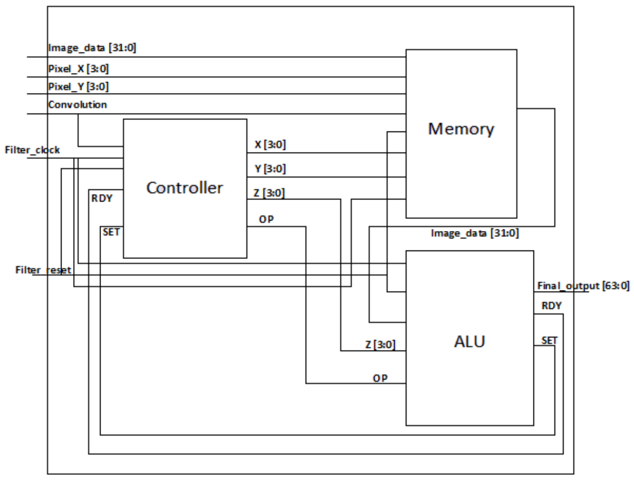


Fig. 4. Detailed top level block diagram.

Pixel_X	Pixel_Y	Image_data	Convolution	Final_output
0000	0000	1	0	-
0000	0001	2	0	-
0000	0010	3	0	-
0001	0000	4	0	-
0001	0001	5	0	-
0001	0010	6	0	-
0010	0000	7	0	-
0010	0001	8	0	-
0010	0010	9	0	-
-	-	-	1	3BDDA263
-	-	-	1	3BDDA177
-	-	-	1	3BDDA93E
-	-	-	1	3BDDAAEB
-	-	-	1	3BDDAAF2
-	-	-	1	3BDDAB02
-	-	-	1	3BDDAB03
-	-	-	1	3BDDAB06
-	-	-	1	3BDDAB06

Fig. 5. Output result of the filter.

IMAGE	COEFF	IMAGE*COEFF		SUM OF IMAGE*COEFF	
1	0.006737943	0.006737943	3BDCC9F6		
2	1.29E-05	2.57E-05	37D7E06A	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	2D18D7D0	0.006764706	3BDDAA77
7	-1.36E-11	-9.51E-11	AED12154	0.006764705	3BDDAA75
8	2.65E-14	2.12E-13	2A6EE220	0.006764705	3BDDAA75
9	8.53E-17	7.68E-16	265D5A7A	0.006764705	3BDDAA75
		0.006764705	3BDDAA75		

Fig. 6. Real convolution data.

From Fig. 5 the output result for the filter is 0.006764772(3BDDAB06) but the expected result in Fig. 6 was 0.006764705(3BDDAA75). The difference was 0.00000068. The error was only 0.001%. This new design verifies that even though a Vedic Multiplier was implemented in the MAC unit of the filter, the functionality of the Gabor Filter has not been jeopardized.

B. *Controller (CLU)*

The Control Logic Unit (CLU) is the controller which controls the data flow in the filter. It instructs the other blocks to perform their jobs. It provides the memory address from where data is to be read in the Memory and also provides the address from where the Kernel coefficient is to be read from the ROM in the ALU.

Fig. 7 shows the Block Diagram of the Controller. The “RDY” and “SET” signals are inputs to the Controller which are outputs of the ALU. These feedbacks from the arithmetic unit are used to control the operation “OP” of the arithmetic unit. When the “OP” signal is high, the convolution process at the ALU starts. When the “OP” signal is low, the convolution process stops.

The “OP” was designed this way to control accurate series data sent to the arithmetic unit so there won’t be any mismatch of data [7]. The “SET” signal turns off the “OP” signal and the “RDY” signal turns on the “OP” signal. It is a continuous process. Initially, the data is being entered and written to the Memory. When the “START” signal in the Controller goes high, only then the Controller generates “X”, “Y”, and “Z”. “X” and “Y” are address locations in the Memory from where data is to be read and “Z” carries the address from where the Kernel Coefficient is to be read.

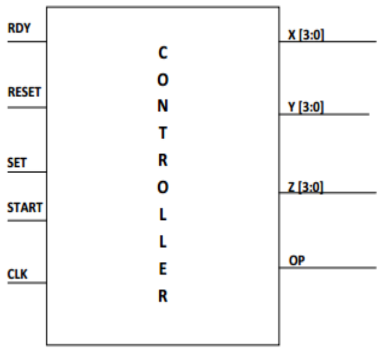


Fig. 7. Block diagram of the controller (CLU).

X	Y	Z	OP	RDY	SET	START	RESET
0000	0000	0000	1	0	0	1	0
0000	0000	0000	0	0	1	1	0
0000	0001	0001	1	1	0	1	0
0000	0001	0001	0	0	1	1	0
0000	0010	0010	1	1	0	1	0
0000	0010	0010	0	0	1	1	0
0001	0000	0011	1	1	0	1	0
0001	0000	0011	0	0	1	1	0
0001	0001	0100	1	1	0	1	0
0001	0001	0100	0	0	1	1	0
0001	0010	0101	1	1	0	1	0
0001	0010	0101	0	0	1	1	0
0010	0000	0110	1	1	0	1	0
0010	0000	0110	0	0	1	1	0
0010	0001	0111	1	1	0	1	0
0010	0001	0111	0	0	1	1	0
0010	0010	1000	1	1	0	1	0
0010	0010	1000	0	0	1	1	0

Fig. 8. Verification of the controller.

C. *Memory*

The Memory block is where the image pixels are being stored. Fig. 9 shows the Memory block diagram. Initially, “Image_data” of 32 – bit is being entered along with

"Pixel_X" and "Pixel_Y" which are 4 bit data providing the location of the memory where this new data is going to be stored. The "Convolution" signal will indicate if the data is going to be read or written in the memory. The "X" and "Y" signals are the signals sent by the controller carrying the location of the memory from which data is to be read. It also has a "RESET" button. The output of the Memory block is the image data we send to the Arithmetic Logic Unit for convolution.

The Memory is a 16 by 16 block and therefore we have 256 locations. Each location is able to store 32 – bit data. The memory is RAM type. When the "Convolution" signal is low, data is being written in the memory in the location provided by "Pixel_X" and "Pixel_Y". When the "Convolution" signal is high, data is being read from the memory from the location provided by the 4- bit signals X and Y. Fig. 10 below shows the verification of the Memory block.

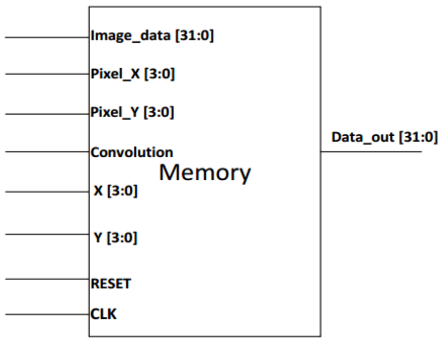


Fig. 9. Block diagram of memory.

Pixel_X	Pixel_Y	Image_data	Convolution	X	Y	Data_out
0000	0000	1	0	-	-	-
0000	0001	2	0	-	-	-
0000	0010	3	0	-	-	-
0001	0000	4	0	-	-	-
0001	0001	5	0	-	-	-
0001	0010	6	0	-	-	-
0010	0000	7	0	-	-	-
0010	0001	8	0	-	-	-
0010	0010	9	0	-	-	-
-	-	-	1	0000	0000	1
-	-	-	1	0000	0001	2
-	-	-	1	0000	0010	3
-	-	-	1	0001	0000	4
-	-	-	1	0001	0001	5
-	-	-	1	0001	0010	6
-	-	-	1	0010	0000	7
-	-	-	1	0010	0001	8
-	-	-	1	0010	0010	9

Fig. 10. Verification of the memory.

D. Arithmetic Logic Unit (ALU)

The Arithmetic Logic Unit is the most important part of this design. The 32 – bit image data enters the Arithmetic Logic Unit from the Memory. The 4 bit Kernel Coefficient address location (Coeff_add) enters the ROM in the ALU from the Controller. The OP signal enters the ALU from the Controller. The CLK and RESET are also inputs to the ALU.

When all the inputs enter the ALU, the image data enters the MAC unit straight away. The coefficient address sent to the ALU by the controller enters the ROM to generate the coefficient address. When the OP signal goes high, the coefficient address is generated and sent to the MAC. The OP signal enters the ROM and the Convolution Signal Buffer simultaneously. The Buffer is used to delay the OP signal by one clock cycle so that it can enter the MAC at the same time as the Kernel Coefficient. Then convolution takes place inside the MAC. The convolved data is our final output. This block also generates a SET and RDY signal which is the driving force of the whole filter. These two signals operate the OP signal for further convolutions.

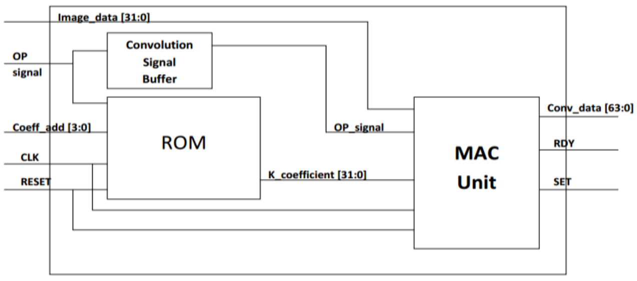


Fig. 11. Block diagram of Arithmetic Logic Unit.

Image_data	Coeff_add	OP Signal	K_Coefficient	Conv_data
1	0000	1	3BDCC9F6	3BDDA263
2	0001	1	37586D14	3BDDA177
3	0010	1	B32BCC77	3BDDA93E
4	0011	1	347D6730	3BDDAAEB
5	0100	1	333D6876	3BDDAAF2
6	0101	1	2BCC11C1	3BDDAB02
7	0110	1	2D6F40F4	3BDDAB03
8	0111	1	28EEB0D7	3BDDAB06
9	1000	1	24C4B03C	3BDDAB06

Fig. 12. Verification of the Arithmetic Logic Unit.

Fig. 13 shows the detailed description of the MAC unit where the Vedic Multiplier lies. The inputs in MAC are the 32-bit image data signal, 32-bit Kernel coefficient signal, and the OP signal. All these enter the Data Control Buffer. The Buffer is used to control the data that enters the Vedic Multiplier. The Buffer holds the data till the OP signal goes high. When the OP signal goes high, only then the image data and the Kernel Coefficients are released and allowed to enter the Vedic Multiplier. At that same positive edge clock cycle when the convolution signal goes high, the signals enter and the convolution also takes place.

The OP signal entering the Vedic Multiplier controls the SET signal. When the OP signal goes high, the SET signal waits one clock cycle to become high. The SET signal now goes to the controller to turn off the OP signal.

The 64-bit multiplied output (mul_out) from the Vedic Multiplier now enters the Accumulator. The "mul_rdy" signal (which is also the SET signal) enters the Accumulator and controls the accumulation. When the "mul_rdy" signal goes high in the Accumulator, accumulation takes place and eventually we get a 64 – bit filtered output. When the "mul_rdy" signal goes high, the RDY signal waits four clock cycles to become high. This RDY signal enters the controller from the ALU.

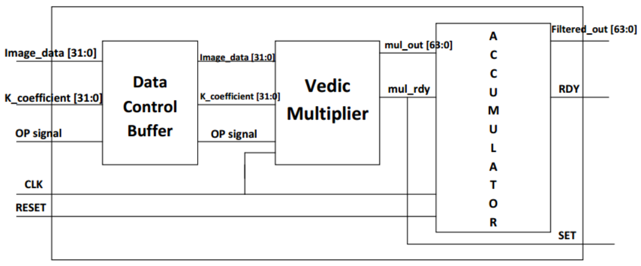
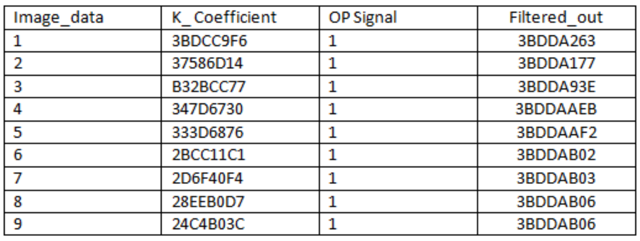


Fig. 13. Block diagram of the MAC unit.

When RDY signal enters the controller, it makes the OP signal high. As one signal influences the other continuously, the whole process continues till there is no data left for convolution. The verification of this is shown in Fig. 14.



Image_data	K_Coefficient	OP Signal	Filtered_out
1	3BDCC9F6	1	3BDDA263
2	37586D14	1	3BDDA177
3	B32BCC77	1	3BDDA93E
4	347D6730	1	3BDDAAEB
5	333D6876	1	3BDDAAF2
6	2BCC11C1	1	3BDDAB02
7	2D6F40F4	1	3BDDAB03
8	28EEB0D7	1	3BDDAB06
9	24C4B03C	1	3BDDAB06

Fig. 14. Verification of the MAC unit.

V. CONCLUSION

Our project proposed the design of Multiplier and Accumulator (MAC) Unit in the Gabor Filter using the techniques of Ancient Indian Vedic Mathematics that have been modified to improve performance. The speed of MAC depends greatly on the multiplier. This work has proved the efficiency of Urdhva Triyagbhyam – Vedic method for multiplication which strikes a difference in the actual process of multiplication itself. Our project shows that design of MAC unit using Vedic multiplication is efficient in terms of speed compared to conventional multiplication. This modified Gabor Filter has a lot of future scopes as it can be used for Cancer Detection, Brain Tumor Detection, Video Processing and even extracting Satellite Images.

We have successfully designed a Gabor Filter with Vedic Multipliers using Verilog HDL. First of all, we designed the main module (top module) and then we designed the modules of the CLU, ALU, and Memory. After designing these modules, we had to further design the modules that are needed to run CLU, ALU and Memory. The main and important part of our modified Gabor Filter was the Vedic Multiplier which was hard to design because we had access to the 2 bit by 2 bit Vedic Multiplier which we had to instantiate several times to obtain the 32 bit by 32 bit one.

ACKNOWLEDGMENT

Authors would like to thank the Almighty for giving them the opportunity to do this work. They would also like to thank Dr. Kazi Abdus Salam, Chairman of the ECE Department at North South University.

REFERENCES

[1] Image processing. [Online]. Available: http://en.wikipedia.org/wiki/Image_processing
[2] Gabor Filter. [Online]. Available: http://en.wikipedia.org/wiki/Gabor_filter
[3] A. H. A. Razak and R. H. Taharim, "Implementing gabor filter for fingerprint recognition using Verilog HDL," *IEEE Explorer*, March 2009.
[4] G. G. Kumar and V. Charishma, "Design of high speed Vedic Multiplier using Vedic mathematics techniques," *International Journal of Scientific and Research Publications*, vol. 2, issue 3, March 2012.
[5] J. Sriskandarajah, "Secrets of ancient maths: Vedic mathematics," *Journal of Indic Studies Foundation*, California, pp. 15-16.
[6] V. K. Karthik, Y. Govardhan, and V. K. Reddy, "Design of multiply and accumulate unit using Vedic multiplication techniques", vol. 4, issue 6, June 2013.
[7] M. F. M. M. Idros, S. A Razak, A. H. A. Zoolfakar, and A. S. Al-Junid, "Improvisation of gabor filter design using Verilog HDL," Faculty of Electrical Engineering, Universiti Teknologi Mara, Malaysia.

Naheean Rahim is a graduate student from the Department of Electrical and Computer Science Engineering, North South University, Dhaka, Bangladesh. He was born on the 9th of April, 1990, in the city of Dhaka, Bangladesh. He is willing to pursue his master's on solid state electronics from North American University. His research interests strongly lie within the fields of VLSI and system-on-chip design (SoC).

Shamayla Islam is a graduate student from the Department of Electrical and Electronic Engineering, North South University, Dhaka, Bangladesh. She was born on the 25th of January, 1991, in the city of Dhaka, Bangladesh. She is willing to pursue her master's on electronics engineering from a North American University. Her research interests lie strongly within the fields of VLSI and computer architecture.

Iqbalur Rahman Rokon is a faculty member at North South University, Dhaka, Bangladesh. He was a former Sr. engineer from VLSI Chip Research and Development (R&D), Emulex Corporation, California, USA.