The International Journal Of Engineering And Science (IJES)
|| Volume || 3 || Issue || 6 || Pages || 25-31 || 2014 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Design and Implementation of Gabor Type Filters on FPGA

[1,]Sunitha M K , [2,]Harsha B K
[1,2,]*Department of electronics and communication, CMRIT*

---ABSTRACT---

A Field Programmable Gate Array implementation of the Gabor-type filter is presented. The implementation uses the forward Euler approximation. In, this paper the details of the realization of Gabor filter architecture using the floating – point operations are discussed The paper presents, a brief mathematical overview of the Gabor-type filters which involves floating point arithmetic operation analysis results, the architectural details of the Gabor-type filter, and, the structure of the FPGA implementation is given. Finally implementation results are discussed. Simulation and Synthesis is done using Xilinx ISE design suite. Verilog HDL will be used as a description language for mapping algorithm in VLSI and hardware implementation on SPARTAN-3E FPGA.

KEYWORDS: *Gabor filter, FPGA, Euler approximation, floating point arithmetic.*

I. INTRODUCTION

In image processing, a Gabor filter, named after Demis Gabor, is a linear filter used for the edge detection. Frequency and orientation representation 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 literature, FIR [3]-[6] and CNN – based [7] digital implementation methods for Gabor filters are available, where [3]-[6] demonstrated graphics processing unit (GPU) and application – specific integrated circuit implementations respectively However there are some difficulties in the realization of FIR Gabor-filters using fixed-point operations, such as accuracy, large template requirements for high quality factor etc. The difficulties have been overcome by Gabor-type filters realized with CNN using floating point operations, which has similar properties to Gabor-filters. This paper deals with mathematical overview of the Gabor-type filter, which involves floating point arithmetic operation analysis, results and the architectural details of the Gabor-type filter are presented and the structure of the FPGA implementation is given. Finally implementation results are discussed

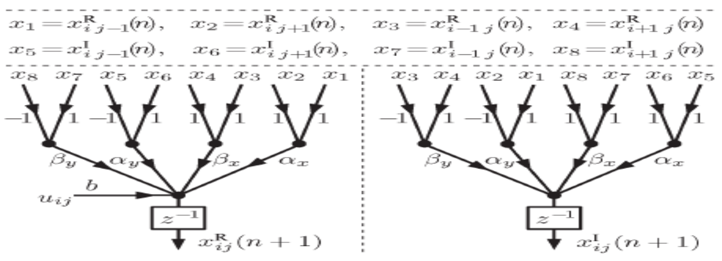
II. CNN – BASED GTF ARCHITECTURE

A DT-CNN-based GTF structure was proposed in [2] as an implementation of [1] whose signal-flow diagram is shown in figure 1. below.

$$\alpha x = \frac{\cos \omega_{x_0}}{4+\lambda^2}, \beta x = \frac{\cos \omega_{y_0}}{4+\lambda^2} \qquad (1)$$

$$\alpha y = \frac{\sin \omega_{x_0}}{4+\lambda^2}, \beta y = \frac{\sin \omega_{y_0}}{4+\lambda^2}, b = \frac{\lambda^2}{4+\lambda^2} \qquad (2)$$

Where, ω_{x_0} and ω_{y_0} are the center frequencies of x and y axes respectively
λ= wavelength of the sinusoidal factor.


$x_1 = x^R_{i\,j-1}(n)$, $x_2 = x^R_{i\,j+1}(n)$, $x_3 = x^R_{i-1\,j}(n)$, $x_4 = x^R_{i+1\,j}(n)$
$x_5 = x^I_{i\,j-1}(n)$, $x_6 = x^I_{i\,j+1}(n)$, $x_7 = x^I_{i-1\,j}(n)$, $x_8 = x^I_{i+1\,j}(n)$
β_y α_y β_x α_x
u_{ij} b
z^{-1}
$x^R_{ij}(n+1)$
$x^I_{ij}(n+1)$

.

Fig.1 DT-CNN-based GTF structure signal flow graph [22]

The signal flow graph of the above figure1 is as shown as a logical block diagram of one CNN – based GTF computation unit in figure 2. A DT-CNN –based GTF structure forms the basis for the logical block diagram of figure 2 which is used in this paper for the design and implementation of the GTF architecture.

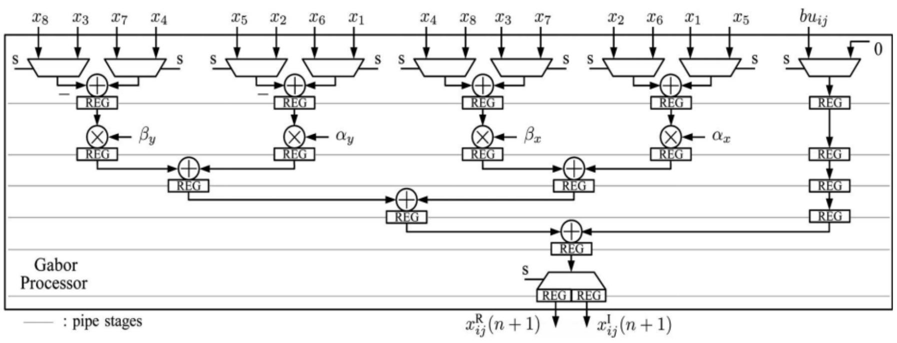


Fig.2. Logical block diagram of one CNN-based GTF computation unit

Two different approaches are used in the logical realization of the fig 2: direct implementation and resource sharing. In the former, a computation tree is implemented directly and, in the latter, considering the similarity of the two diagrams; one computation tree is realized and multiplexed for the computation of real and imaginary parts (fig. 2)

The multiplexing halves the number of multiplier and adder resources; however, twice the pixel frequency is required for computation. The input bu_{ij} is also multiplexed with zero, as it is only required for the calculation of the real part. All adders and multipliers are registered to form a pipeline. The latency of the processor is three pixel clock cycles as there are six pipe stages. Finally, the real and the imaginary parts of the resulting values are demultiplexed to their respective output registers.

III. RESULTS AND DISSCUSSION

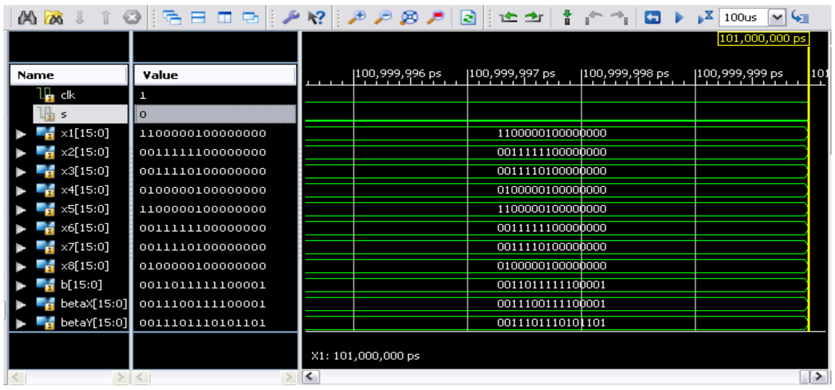
3.1 SIMULATION RESULTS

The architecture was written with Verilog HDL and synthesized by Xilinx ISE 14.2. After synthesizing this Verilog HDL code, tests were done with MODELSIM. The following are the simulation results obtained.

3.1.1 Simulation and synthesis results for select line‘s’ zero

The below window represents the input values for select line zero. The calculated floating-point binary numbers for each multiplexer output is shown. Thus indicating, for the GTF architecture the results are accurate, matching both the manual and the synthesis results exactly.

Fig3. by forcing the clock as ‘1’, period of ‘100µs’ and select line ‘s’ zero, the results of the multiplexer and de-multiplexer pipeline stages of the GTF architecture are obtained representing the real part of the architectural design.

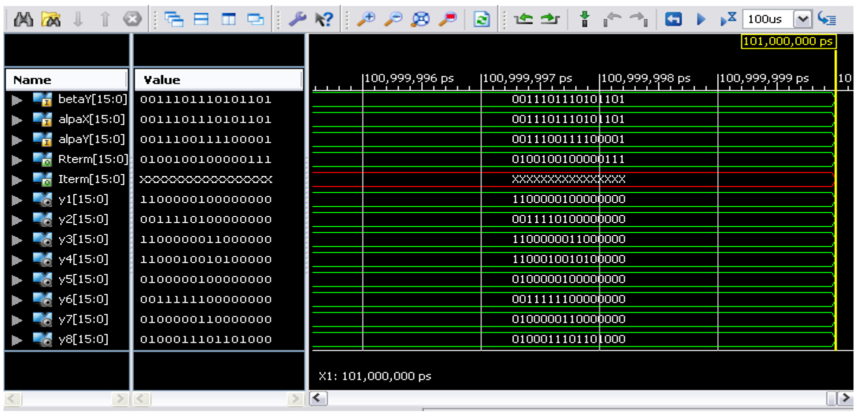


Name	Value
clk	1
s	0
x1[15:0]	1100000100000000
x2[15:0]	0011111100000000
x3[15:0]	0011110100000000
x4[15:0]	0100000100000000
x5[15:0]	1100000100000000
x6[15:0]	0011111100000000
x7[15:0]	0011110100000000
x8[15:0]	0100000100000000
b[15:0]	0011011111100001
betaX[15:0]	0011100111100001
betaY[15:0]	0011101110101101

X1: 101,000,000 ps

Fig. 3 representing input values for the real part of the system with select line zero

Fig 4 represents the boundary conditions alpha, beta, lambda values of real term with select line‘s’ zero. Also resulted multiplexed registered outputs and the demultiplexed real term [15:0] is obtained

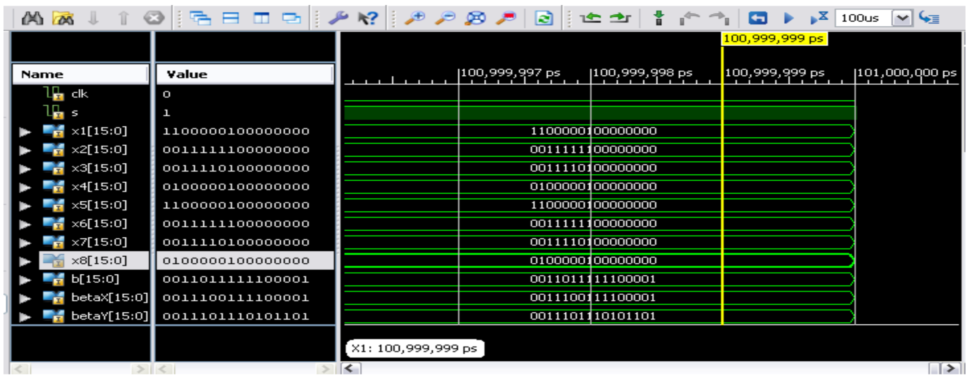


Name	Value
betaY[15:0]	0011101110101101
alpaX[15:0]	0011101110101101
alpaY[15:0]	0011100111100001
Rterm[15:0]	0100100100000111
Iterm[15:0]	XXXXXXXXXXXXXXXX
y1[15:0]	1100000100000000
y2[15:0]	0011110100000000
y3[15:0]	1100000011000000
y4[15:0]	1100010010100000
y5[15:0]	0100000100000000
y6[15:0]	0011111100000000
y7[15:0]	0100000110000000
y8[15:0]	0100011101101000

Fig. 4 representing the boundary conditions and registered outputs [15:0] real term of the system with select line zero

3.1.2 Synthesis and simulation results for select line‘s’ one: This represents the imaginary value of the demultiplexed output. Below window represents the input values for the imaginary term of the multiplexer and de-multiplexer.

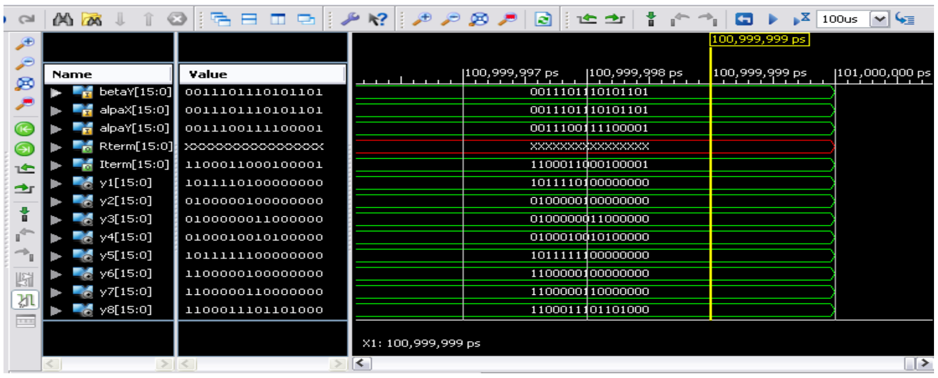
Fig. 5 : by forcing the clock as ‘1’, period of ‘100µs’ and select line ‘s’ one, the results of the multiplexer and de-multiplexer pipeline stages of the proposed architecture are obtained representing the imaginary part of the architectural design.



Name	Value
clk	0
s	1
x1[15:0]	1100000100000000
x2[15:0]	0011111100000000
x3[15:0]	0011110100000000
x4[15:0]	0100000100000000
x5[15:0]	1100000100000000
x6[15:0]	0011111100000000
x7[15:0]	0011110100000000
x8[15:0]	0100000100000000
b[15:0]	0011011111100001
betaX[15:0]	0011100111100001
betaY[15:0]	0011101110101101

Fig 5 representing the real part of the system with select line one

Fig 6 represents the boundary conditions alpha, beta, lambda values of imaginary term with select line‘s’ one. Also resulted multiplexed registered outputs and the demultiplexed real term [15:0] are obtained**.**



Name	Value
betaY[15:0]	0011101110101101
alpaX[15:0]	0011101110101101
alpaY[15:0]	0011100111100001
Rterm[15:0]	XXXXXXXXXXXXXXXX
Iterm[15:0]	1100011000100001
y1[15:0]	1011110100000000
y2[15:0]	0100000100000000
y3[15:0]	0100000011000000
y4[15:0]	0100010010100000
y5[15:0]	1011111100000000
y6[15:0]	1100000100000000
y7[15:0]	1100000110000000
y8[15:0]	1100011101101000

Fig. 6 represents the boundary conditions alpha, beta, lambda and imaginary term

3.2. FPGA IMPLEMENTATION RESULTS

In order to form Gabor type filter structure shown in fig 2, multipliers and adders are needed. The structure is formed by blocks shown in fig 1. The logical block diagram of fig 1 is depicted in fig 2 This architecture was written with VHDL and synthesized by Xilinx ISE 14.2. After synthesizing this VHDL code, tests were done with MODELSIM. When this process unit is synthesized, placed and routed for XC3S400 FPGA, a 50-MHz clock rate can be achieved. The architecture can cover 186 (1%) slice flip flops and four mult 18×18 (20%) on FPGA.

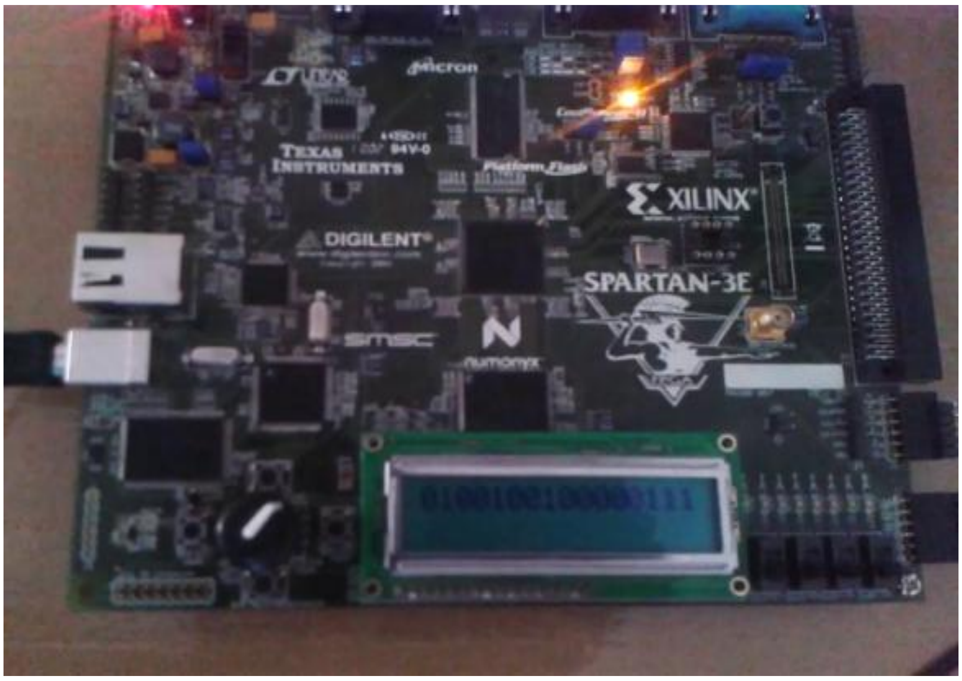


Fig. 7 FPGA implementation of GTF architecture with the hardware resulting in real part of the demultiplexer with select line‘s’ zero.

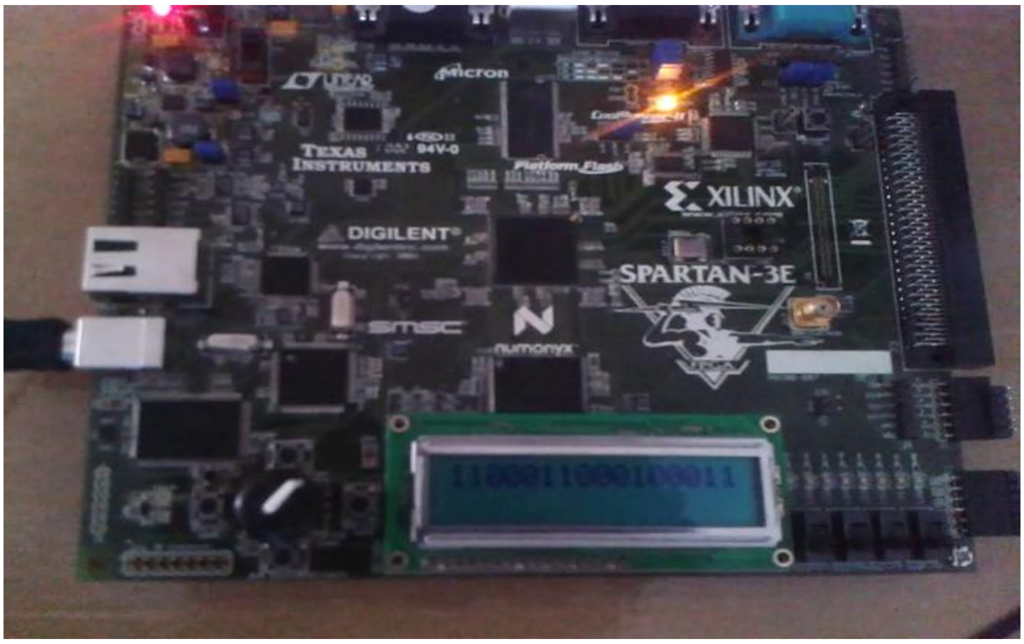


Fig. 8 FPGA implementation of GTF architecture with the hardware resulting in imaginary part of the demultiplexer with select line ‘s’one.

The fig. 9 below shows the top module of the system architecture.

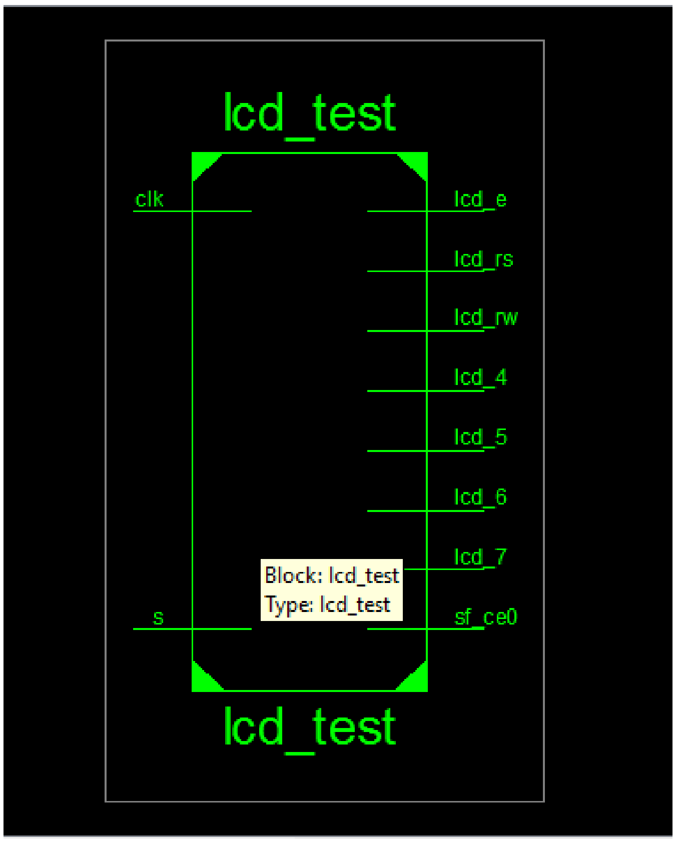


Fig.9 Top module technology schematic

The Fig. 10 below shows the internal architecture of the top module

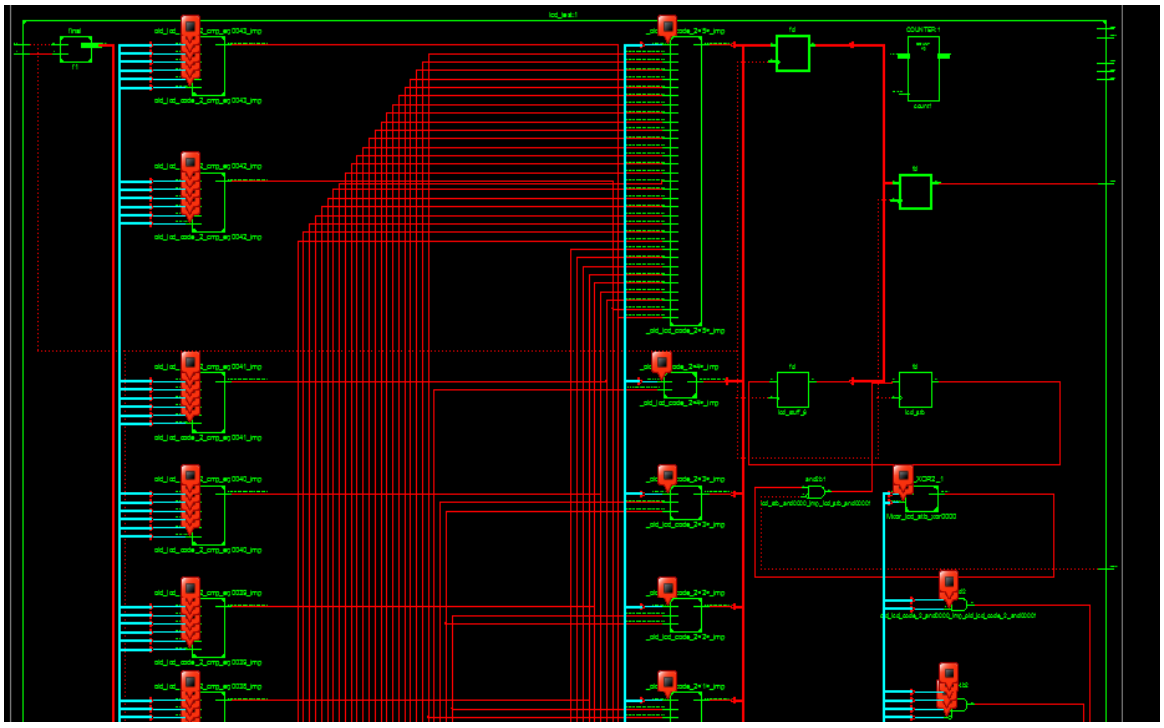


Fig. 10 The internal architecture of top module

The fig. 11 shows the schematic diagram of the GTF architecture.

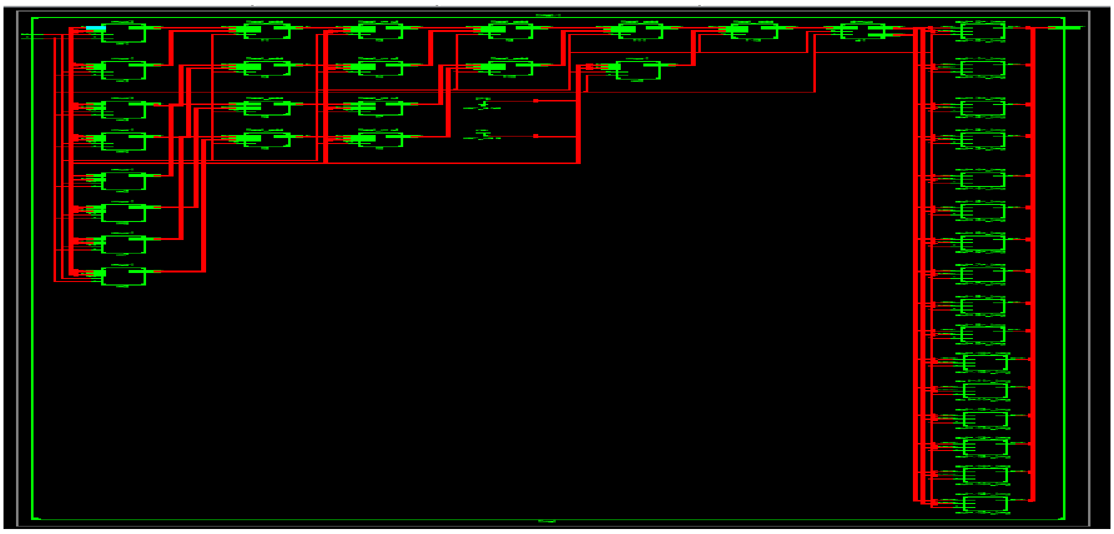
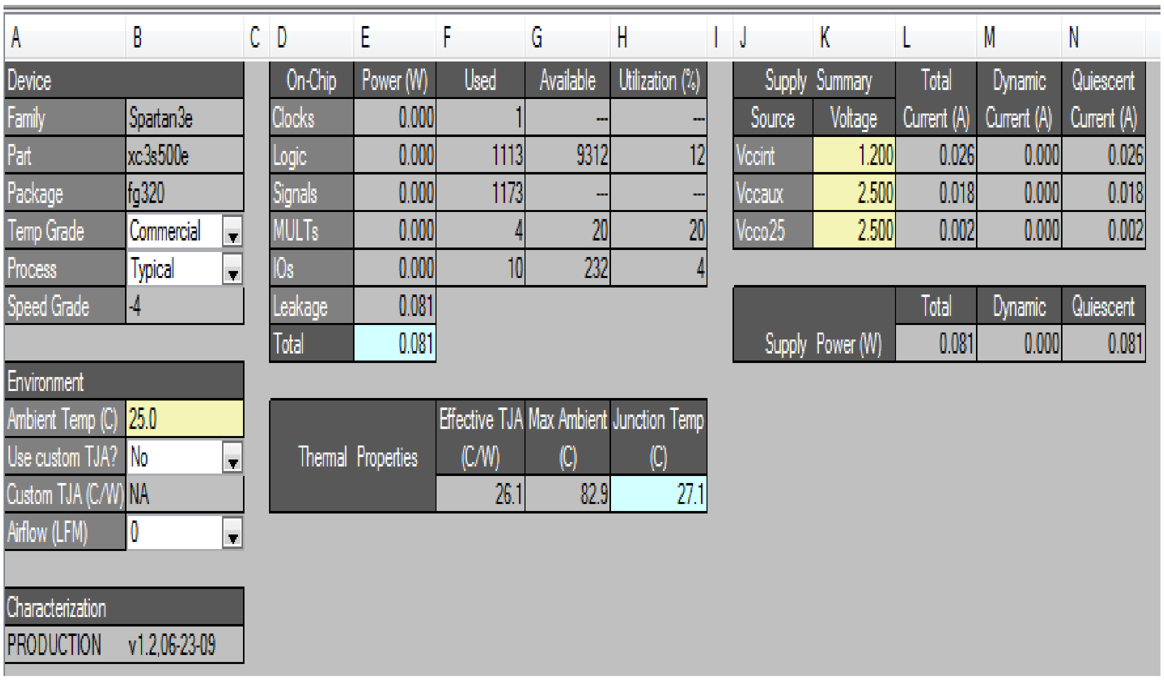


Fig. 11 The schematic diagram of the GFT architecture

The fig. 12 shows the power analysis results for the desugned architecture.



A	B
Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4
Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	1113	9312	12
Signals	0.000	1173	---	---
MULTs	0.000	4	20	20
IOs	0.000	10	232	4
Leakage	0.081			
Total	0.081			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	82.9	27.1

Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.026	0.000	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.081	0.000	0.081

Fig 12 Power analysis

IV. CONCLUSION

In this paper, a brief mathematical overview of the Gabor-type filters is presented; floating point arithmetic operation analysis results and the architectural details of the Gabor-type filter are given. The structure of the FPGA implementation is given. Finally implementation results are discussed. A CNN- Gabor type filter realization method was proposed coded in VHDL and the simulation results are obtained resulting in accuracy.

Digital implementation of Gabor-type filters was proposed, which drastically reduces the number of multipliers required for FPGA implementation. The Gabor filters are suitable for a real-time application such as optical character recognition (OCR), facial recognition or license plate recognition and any other system that requires two-dimensional band-pass filters.

ACKNOWLEDGEMENTS

I am extremely pleased to present myself this dissertation after getting innumerable movements and hard time with a never-ending coordination. I sincerely acknowledge offering my words of thanks to our beloved and respected Mr. Harsha Assistant professor Mrs. Indumathi Head of Electronics and communication department I sincerely acknowledge offering my words of thanks to my beloved and respected parents for their constant support and encouragement.

REFERENCES

[1]. E. Saatci, E. Cesur, V. Tavsanoglu, and I. Kale, "An FPGA implementation of 2-D CNN Gabor-type filter," in Proc. 18th ECCTD, Aug. 2007, pp. 280–283.

[2]. E. Cesur, N. Yildiz, and V. Tavsanoglu, "An improved FPGA implementation of CNN Gabor-type filters," in Proc. IEEE ISCAS, May 2011, pp. 881–884

3]. X. Wang and B. Shi, "GPU implementation of fast Gabor filters," in Proc. IEEE ISCAS, May 30–Jun. 2, 2010, pp. 373–376.

[4]. E. Norouznezhad, A. Bigdeli, A. Postula, and B. Lovell, "Robust object tracking using local oriented energy features and its hardware/software implementation," in Proc. 11th ICARCV, Dec. 2010, pp. 2060–2066.

[5]. Y. Cho, S. Bae, Y. Jin, K. Irick, and V. Narayanan, "Exploring Gabor filter implementations for visual cortex modeling on FPGA," in Proc. Int. Conf.FPL, Sep. 2011, pp. 311–316.

[6]. J. Liu, S.Wang, Y. Li, J. Han, and X. Zeng, "Configurable pipelined Gabor filter implementation for fingerprint image enhancement," in Proc. 10thIEEE ICSICT, Nov. 2010, pp. 584–586.

[7]. O. Cheung, P. Leong, E. Tsang, and B. Shi, "A scalable FPGA implementation of cellular neural networks for Gabor-type filtering," in Proc.IJCNN, 2006, pp. 15–20.

[8]. E. Saatci, E. Cesur, V. Tavsanoglu and I. Kale, "An FPGA implementation Of 2-D CNN Gabor-type filter," 18th European Conference on Circuit Theory and Design, ECCTD, Seville, Spain, August 2007.

[9]. J. Yang, L. Liu, T. Jiang, and Y. Fan. A modified Gabor filter design method for fingerprint image enhancement. Pattern Recognition Letters, 24:1805–1817, 2003

[10]. V. Kyrki, J.-K.Kamarainen, and H. Klviinen. "Simple Gabor feature space for invariant object recognition". Pattern Recognition Letters, 25, 2004

[11]. Ajay Kumar, Member, IEEE, and Grantham K. H. Pang, Senior Member, IEEE, "Defect Detection in Textured Materials Using Gabor Filters"

[12]. H. Sari-Sarraf and J. S. Goddard, "Vision systems for on-loom fabric inspection," IEEE Trans. Ind. Applicant., vol. 35, pp. 1252–1259, Nov/Dec. 1999.

[13]. S. Kim, M. H. Lee, and K. B.Woo,"Wavelet analysis to defects detection in weaving processes," in Proc. IEEE Int. Symp. Industrial Electronics, vol. 3, July 1999, pp. 1406–1409

[14]. L. Shen and L. Bai, "A review of Gabor wavelets for face recognition," Patt.Anal. Appl.9:pp.273-292, 2006

[15]. M. Turk and A. Pentland, "Eigenfaces for recognition," J.Cognitive. Neuroscience, 3(1): 71-86, 1991

[16]. P, Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection" IEEE Trans. Pattern Analysis and Machine Intelligence, 19(7): 711-720, 1997

[17]. B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," IEEE Trans. Pattern Analysis and Machine Intelligence, 18(8): 837-842, 1996

[18]. S. Arivazhagan, L. Ganesan, and S. P. Priyal, "Texture classification using Gabor wavelets based rotation invariant features," Pattern recognition letters, 27(16): 1976-1982, 2006

[19]. M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with Gabor wavelets," Proc. Int'l Conf. Automatic Face and Gesture Recognition, 200-205, 1998

[20]. Z. Zhang, M. Lyons, M. Schuster, and S. Akamatsu, "Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron," Proc. Int'l Conf. Automatic Face and Gesture Recognition, 454-459, 1998

[21]. Joni-Kristian Kamarainen, Ville Kyrki, and HeikkiKälviäinen, "Invariance Properties of Gabor Filter-Based Features", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 15, NO. 5, MAY 2006

[22]. EvrenCesur, NerhunYildiz, and VedatTavsanoglu, "On an Improved FPGA Implementation of CNN-Based Gabor-Type Filters", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 59, NO. 11, NOVEMBER 2012