An Improved FPGA Implementation of CNN Gabor–type Filters

Evren CESUR
Electronics and Communications
Engineering Department
Yildiz Technical University
Istanbul, Turkey, 34349
Email: ecesur@yildiz.edu.tr

Nerhun YILDIZ
Electronics and Communications
Engineering Department
Yildiz Technical University
Istanbul, Turkey, 34349
Email: nerhuny@yildiz.edu.tr

Vedat TAVSANOGLU
Electronics and Communications
Engineering Department
Yildiz Technical University
Istanbul, Turkey, 34349
Email: tavsanav@yildiz.edu.tr

***Abstract*— In this paper, a new Cellular Neural Network (CNN) structure for implementing two dimensional Gabor–type filters is proposed over our previous design. The structure is coded in VHDL and realized on a state of the art Altera Stratix IV 230 FPGA. The prototype supports Full–HD 1080p resolution and 60 Hz frame rate. One dedicated processor is used for each Euler iteration, where time step is taken as the same as optimum step size, and 50 iterations are implemented. The input/output, control, RAM and communication blocks of the realization are taken from our second generation real time CNN emulator (RTCNNP-v2).**

I. Introduction

As well known, impulse response of Gabor filters are complex exponential functions modulated by a Gaussian [1]. Considering low–pass characteristics of Gaussian filters, Gabor filters have band–pass characteristics. However, there are some difficulties in the realization of FIR Gabor filters, such as complex valued calculations, large template requirement for high quality factor, etc. The difficulties have been overcame by proposing Gabor–type filters realized with CNN (Cellular Neural Networks), which has similar properties with Gabor filters [2]. Also a digital implementation is given in [3].

An alternative digital implementation of Gabor–type filters was proposed in [4]. In this paper, an improvement over [4] is proposed, which drastically reduces the number of multipliers required for FPGA implementation. The input/output, control, RAM and communication structures are the same as our real–time CNN emulator (RTCNNP-v2) [5]. Filter coefficient are programmable at runtime over an RS232 interface, hence there is no need to reconfigure the FPGA for different Gaussian–type and Gabor–type filters.

In this paper, first, a brief mathematical overview of the Gabor–type filters is presented. Second, resource optimization methods over our previous design are proposed. Third, the structure of the FPGA implementation is given. Finally implementation results are discussed.

II. Mathematical Overview

The mathematical model of the ij'th linear CNN cell is

$$\frac{dx_{ij}(t)}{dt} = \boldsymbol{A} \circledast \boldsymbol{X}_{ij}(t) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{1}$$

where i and j are the index variables of the neural cell, x_{ij} is the state, $\boldsymbol{A}$ is the feedback and $\boldsymbol{B}$ is the feed–forward cloning templates, $\circledast$ is the template–dot–product operator, $\boldsymbol{X}_{ij}$ is the output image at the ij'th cell and $\boldsymbol{U}_{ij}$ is the translated masked input image [6].

The equation (1) can be rewritten as

$$\begin{aligned}\dot{x}_{ij}^R + j\dot{x}_{ij}^I = {} & \boldsymbol{A}_R \circledast \boldsymbol{X}_{ij}^R - \boldsymbol{A}_I \circledast \boldsymbol{X}_{ij}^I \\ & + j\left(\boldsymbol{A}_R \circledast \boldsymbol{X}_{ij}^I + \boldsymbol{A}_I \circledast \boldsymbol{X}_{ij}^R\right) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij}\end{aligned} \tag{2}$$

if $\boldsymbol{U}_{ij}$ and $\boldsymbol{B}$ are real and $\boldsymbol{X}_{ij}$ and $\boldsymbol{A}$ are complex valued, which yields

$$\dot{x}_{ij}^R = \boldsymbol{A}_R \circledast \boldsymbol{X}_{ij}^R - \boldsymbol{A}_I \circledast \boldsymbol{X}_{ij}^I + \boldsymbol{B} \circledast \boldsymbol{U}_{ij} \tag{3}$$

$$\dot{x}_{ij}^I = \boldsymbol{A}_R \circledast \boldsymbol{X}_{ij}^I + \boldsymbol{A}_I \circledast \boldsymbol{X}_{ij}^R. \tag{4}$$

In [2], Gabor–type CNN filter templates are given as

$$\boldsymbol{A} = \begin{bmatrix} 0 & e^{-j\omega_{y0}} & 0 \\ e^{j\omega_{x0}} & -(4+\lambda^2) & e^{-j\omega_{x0}} \\ 0 & e^{j\omega_{y0}} & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

or

$$\boldsymbol{A}_R = \begin{bmatrix} 0 & \cos(\omega_{y0}) & 0 \\ \cos(\omega_{x0}) & -(4+\lambda^2) & \cos(\omega_{x0}) \\ 0 & \cos(\omega_{y0}) & 0 \end{bmatrix}$$

$$\boldsymbol{A}_I = \begin{bmatrix} 0 & -\sin(\omega_{y0}) & 0 \\ \sin(\omega_{x0}) & 0 & -\sin(\omega_{x0}) \\ 0 & \sin(\omega_{y0}) & 0 \end{bmatrix}.$$

where ω_{x0} and ω_{y0} are the center frequencies of x and y axes, respectively, and 2λ is the bandwidth.

Gabor–type CNN equation can be discretized by applying forward Euler approximation with optimum step size [7] and we obtain

$$x_{ij}^R(n+1) = \frac{1}{4+\lambda^2}\left\{\widetilde{\boldsymbol{A}}_R \circledast \boldsymbol{X}_{ij}^R(n) - \boldsymbol{A}_I \circledast \boldsymbol{X}_{ij}^I(n) + \boldsymbol{B} \circledast \boldsymbol{U}_{ij}\right\} \tag{5a}$$

$$x_{ij}^I(n+1) = \frac{1}{4+\lambda^2}\left\{\widetilde{\boldsymbol{A}}_R \circledast \boldsymbol{X}_{ij}^I(n) + \boldsymbol{A}_I \circledast \boldsymbol{X}_{ij}^R(n)\right\} \tag{5b}$$

where

$$\widetilde{\boldsymbol{A}}_R = \begin{bmatrix} 0 & \cos(\omega_{y0}) & 0 \\ \cos(\omega_{x0}) & 0 & \cos(\omega_{x0}) \\ 0 & \cos(\omega_{y0}) & 0 \end{bmatrix} \tag{6}$$

978-1-4244-9474-3/11/$26.00 ©2011 IEEE

is a real surround template [6]. The discrete Gabor–type CNN equations can be rewritten in matrix form as

$$\mathbf{X}_{i,j}(n+1) = \frac{1}{4+\lambda^2}\left\{\mathbf{G}_x\mathbf{X}_{i-1,j}(n) + \mathbf{G}_x^t\mathbf{X}_{i+1,j}(n) \right. \\ \left. + \mathbf{G}_y^t\mathbf{X}_{i,j+1}(n) + \mathbf{G}_y\mathbf{X}_{i,j-1}(n) + \lambda^2\boldsymbol{U}_{ij}\right\} \tag{7}$$

where

$$\mathbf{X}_{i,j}(n) = \begin{bmatrix}\boldsymbol{X}_{ij}^R(n)\\ \boldsymbol{X}_{ij}^I(n)\end{bmatrix}$$

and

$$\mathbf{G}_x = \begin{bmatrix}\cos(\omega_{x0}) & -\sin(\omega_{x0})\\ \sin(\omega_{x0}) & \cos(\omega_{x0})\end{bmatrix}$$
$$\mathbf{G}_y = \begin{bmatrix}\cos(\omega_{y0}) & -\sin(\omega_{y0})\\ \sin(\omega_{y0}) & \cos(\omega_{y0})\end{bmatrix}$$

are Givens rotations. The stability of the system can be proved using unique property of the Givens rotations as shown in [4].

Defining

$$\alpha_x = \frac{\cos\omega_{x0}}{4+\lambda^2}, \beta_x = \frac{\cos\omega_{y0}}{4+\lambda^2}, \alpha_y = \frac{\sin\omega_{x0}}{4+\lambda^2}$$
$$\beta_y = \frac{\sin\omega_{y0}}{4+\lambda^2}, b = \frac{\lambda^2}{4+\lambda^2}$$

the discrete Gabor–type CNN equations can be reordered as

$$\begin{bmatrix}\boldsymbol{X}_{i,j}^R(n+1)\\ \boldsymbol{X}_{i,j}^I(n+1)\end{bmatrix} = \begin{bmatrix}\alpha_x & -\alpha_y\\ \alpha_y & \alpha_x\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_{i-1,j}^R(n)\\ \boldsymbol{X}_{i-1,j}^I(n)\end{bmatrix} \\ + \begin{bmatrix}\alpha_x & \alpha_y\\ -\alpha_y & \alpha_x\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_{i+1,j}^R(n)\\ \boldsymbol{X}_{i+1,j}^I(n)\end{bmatrix} \\ + \begin{bmatrix}\beta_x & -\beta_y\\ \beta_y & \beta_x\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_{i,j-1}^R(n)\\ \boldsymbol{X}_{i,j-1}^I(n)\end{bmatrix} \\ + \begin{bmatrix}\beta_x & \beta_y\\ -\beta_y & \beta_x\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_{i,j+1}^R(n)\\ \boldsymbol{X}_{i,j+1}^I(n)\end{bmatrix} + \begin{bmatrix}b\boldsymbol{U}_{ij}\\ 0\end{bmatrix}. \tag{8}$$

III. Resource Optimization

The structure proposed in [4] is given in Fig. 1 where 16 multipliers and 10 adders are required for each iteration. In general, the number of multipliers and block RAMs are the primary concerns of an FPGA design. In the new structure, multiplication and addition operations are reordered in order to reduce the number of multipliers. This modification enables the number of multipliers to be reduced by half, from 16 to 8. Fig. 2(a) and 2(b) corresponds to the computation of real and imaginary parts of the output, respectively. Note that Fig. 2(a) and Fig. 2(b) are almost the same, except for the additional input shown in Fig. 2(a). This similarity can be exploited to further reduce the number of multipliers, i.e., from 8 to 4, by reusing the structure in Fig. 2(a) to calculate also the imaginary part. To compensate for the multiplexing, processor clock frequency should be twice the input frequency or dual edge should be used which is easily achieved in most cases.

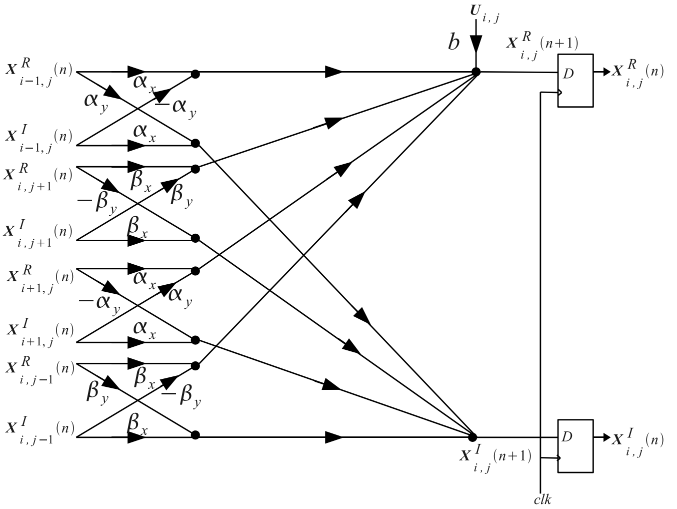
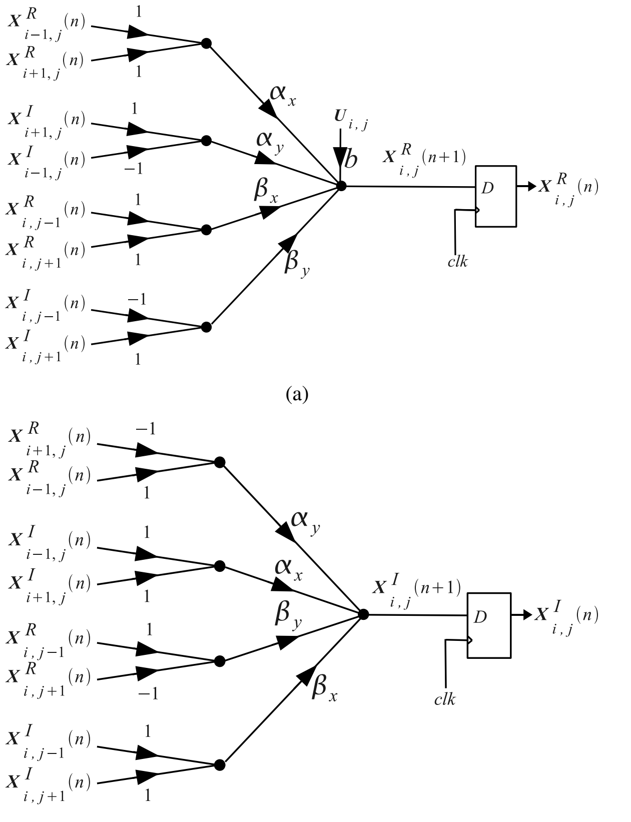


Fig. 1. Structure of the old Gabor–type filter.



(a)

(b)

Fig. 2. Structure of the new Gabor–type filter: (a) real and (b) imaginary output calculation parts.

IV. FPGA Realization

The proposed structure is realized on an Altera Stratix IV 230 FPGA development kit. A daughter card provides DVI input and output. Simplified block diagram of the system is given in Fig. 3. Input/output, control, RAM and communication structures of the new implementation are the same as that of our CNN processor structure RTCNNP-v2 [5]. Input block translates standard DVI signals to horizontal and vertical control signals designed for RTCNNP-v2. Gabor Filter process the data and its output is converted back to original DVI signals. Serial receiver and transmitter blocks are responsible for the communication and the serial programming of the coefficients.

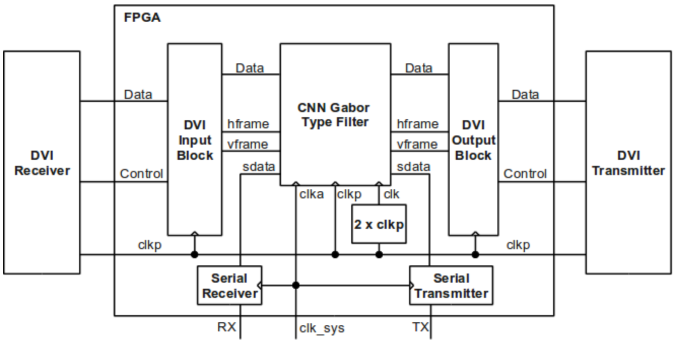


Fig. 3. Simplified block diagram of the FPGA implementation.

The internal structure of the CNN Gabor–type filter block is given in Fig. 4, where BPU is responsible for computing the parts of the state equation which are independent of the iterations. Each APU is responsible for computing one iteration. The system is fully pipelined and there is approximately two lines of latency between consequent APUs.

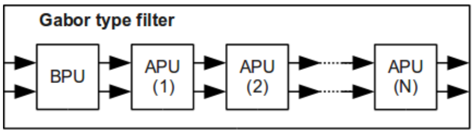


Fig. 4. Internal structure of the CNN Gabor–type filter block

Each APU has its own boundary generator, buffer RAMs, local control and computational sub–blocks (Fig. 5). Local control generates all control signals from *hframe* and *vframe* synchronization signals, which show starting and ending points of the lines and frames respectively [5].

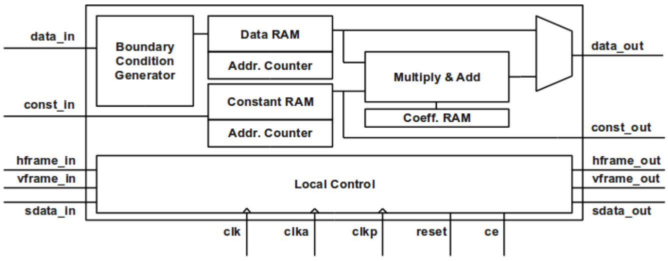


Fig. 5. Block diagram of the CNN Gabor–type processor.

The video test system has 1920 × 1080 resolution, 60 Hz frame rate (Full–HD 1080p@60Hz) and 148.5 MHz pixel frequency. Each processor has an additional two stage multiplexing pipeline that require 297 MHz processing clock derived from the pixel frequency. After synchronization, the odd and even clock signals are used to calculate real and imaginery parts of the state, respectively.

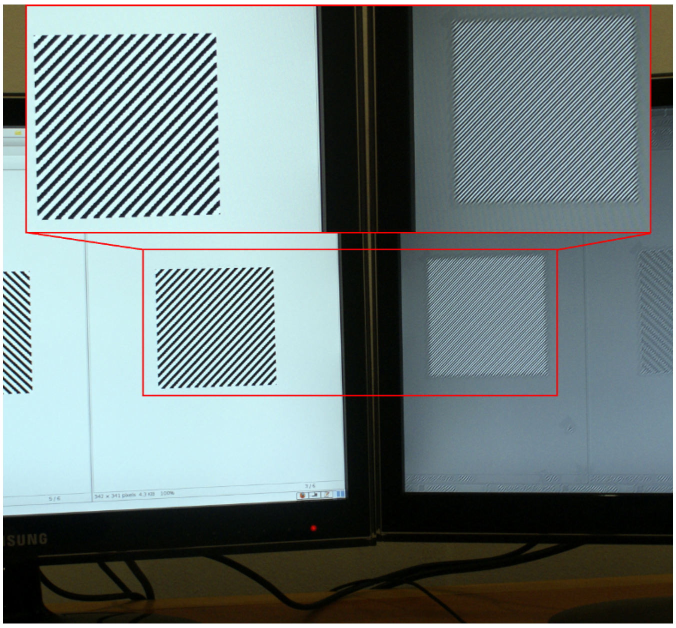


Fig. 6. Realization results: monitors on the left and right sides are the input and output video streams, respectively.

The number of APUs, and consequently the number of Euler iterations are chosen to be 50 for the test implementation. The fix point bit lengths are chosen to be 1.15 for filter coefficients, 1.7 for the real and imaginery states, and 2.22 for the BPU constant. Only one multiplier is needed to compute the BPU result.

Parameters of the system are chosen to be $\omega_{x0} = \pi/4$, $\omega_{y0} = \pi/4$ and $\lambda = 0.1$ for a specific case. In this case the dependent parameters are computed as $\alpha_x = 0.1763$, $\beta_x = 0.1763$, $\alpha_y = 0.1763$, $\beta_y = 0.1763$ and $b = 0.0025$, and the system is programmed accordingly, which yields the result given in Fig. 6. The parameters are chosen in order to tune the center frequency of the filter to the third harmonic of the 45 degree oriented striped image. The resource usage percentages are given in Table I, for the Altera Stratix IV 230 FPGA.

V. Conclusion

In our previous work, an alternative CNN Gabor–type filter realization method was proposed, coded in VHDL and the simulation results were given. In this paper, a new optimization technique is proposed and the system is re–coded for compatibility with our real time CNN emulator RTCNNP-v2 [5]. A test system is realized using auxiliary structures of the RTCNNP-v2 such as input/output blocks and local control. The test system is capable of processing video signals with 1920×1080 video resolution and 60 Hz frame rate (Full–HD 1080p), in real–time.

The filter parameters are easily programmed over a serial interface without the need to reconfigure the FPGA. The

TABLE I

RESOURCE USAGE OF THE PROTOTYPE.

Blocks	Combinational ALUTs	Dedicated Logic Registers	Block Memory (Kb)	DSP 18-bit Elements
50 APUs	13472 (7.4%)	20321 (11.2%)	871 (67.6%)	202 (15.5%)
Glue logic	613 (0.4%)	381 (0.2%)	0 (0%)	0 (0%)
Total	14085 (7.8%)	20702 (11.4%)	871 (67.6%)	202 (15.5%)

number of processors are configurable before synthesis, hence resource and power optimization is possible during the design according to the needs of a specific application.

The Gabor–type filters are suitable for real–time applications such as optical character recognition (OCR), facial recognition or license plate recognition and any other system that requires two dimensional band–pass filters. The realized Gabor–type filter is the fastest FPGA implementation to date and the only one that is reported to work on Full–HD 1080p 1920 × 1080 streaming video signals. The speed is not only provided with the high–end Altera Stratix IV FPGA structure, but also with the optimization and redesign of the architecture.

ACKNOWLEDGMENT

This research was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under project number 108E023.

REFERENCES

[1] D. Gabor, "Theory of Communication," *J. Inst Elect. Eng.*, Vol. 93, pp. 429-457, 1946.

[2] B.E. Shi, "Gabor-type image filtering with cellular neural networks," *IEEE International Symposium on Circuits and Systems 'Connecting the World', ISCAS*, Atlanta, US, May 1996.

[3] O.Y.H. Cheung and P.H.W. Leong and E.K.C. Tsang and B.E. Shi, "A Scalable FPGA Implementation of Cellular Neural Networks for Gabor-type Filtering," *International Joint Conference on Neural Networks, IJCNN*, Vancouver, BC, October 2006.

[4] 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.

[5] N. Yildiz, E. Cesur and V. Tavsanoglu, "A New Control Structure For The Pipelined CNN Processor Arrays," *12th International Workshop on Cellular Nanoscale Networks and Their Applications, CNNA*, Berkeley, CA, February 2010.

[6] L.O. Chua and T. Roska, "Cellular Neural networks and Visual Computing," *Cambridge University Press*, UK, 2002.

[7] E. Saatci and V. Tavsanoglu, "On the optimal choice of integration time-step for raster simulation of a CNN for gray level image processing", *IEEE International Symposium on Circuits and Systems, ISCAS*, Arizona, US, August 2002.