2007 International Symposium on Field-Programmable Custom Computing Machines

Discrete-Time Cellular Neural Networks in FPGA

J. Javier Martínez-Álvarez, F. Javier Toledo-Moreo, J. Manuel Ferrández-Vicente
Dpto. Electrónica, Tecnología de Computadoras y Proyectos
Univ. Politécnica de Cartagena Cuartel de Antiguones, Pl. Hospital, 1, 30202 Cartagena Spain
jjavier.martinez@upct.es

Abstract

This paper describes a novel architecture for the hardware implementation of non-linear multi-layer cellular neural networks. This makes it feasible to design CNNs with millions of neurons accommodated in low price FPGA devices, being able to process standard video in real time.

1 Introduction

Since Chua and Yang proposed the cellular neural network in 1988, a wide field of research has spread on its applications and implementation. A Cellular Neural Network (CNN) is a bioinspired non-linear cellular processor array suitable to be implemented on electronic circuits. These nets are a powerful analogue computer, able to solve complex array signal processing problems.

In this paper, a new architecture is proposed for the hardware implementation of a Discrete-Time Cellular Neural Network. It allows lower cost implementation of bioinspired models in a shorter design cycle.

2 DTCNN model for FPGA implementation

Our proposed equivalent Discrete-Time (DTCNN) model is based in eq. (1) and (2), detailed in [1]:

$$X_{ij}[n] = \sum_{k,l \in Nr(ij)} A_{kl}[n-1] Y_{kl}[n-1] + \sum_{k,l \in Nr(ij)} B_{kl}[n-1] U_{kl} + I_{ij} \tag{1}$$

$$Y_{ij}[n] = \frac{1}{2}\left(|X_{ij}[n]+1| - |X_{ij}[n]-1|\right) \tag{2}$$

where I, U, Y and X denote bias, input, output and state variable of each cell, respectively. B is the non-linear weights template for the inputs and A is the corresponding non-linear template for the outputs of the neighbouring cells. Non-linearity means that templates can change over time. Taking into account the typical target applications, the size of the templates is 3×3.

As the infinite feedback loop of eq. (1) and (2) is unfeasible, it must be constrained in a range. Clearly, the accuracy of the approximation depends on the number of iterations considered. Looking for the worst case value in typical video processing applications, simulations have been carried out with different inputs and templates. These simulations have consisted of several processing algorithms: edge detections, blur, sharpness, gaussian, etc. The correlation coefficient has been used as a measure of how closely the output of original analogue and proposed discrete models are related. The results reveal that 10 iterations or stages are enough to obtain data practically identical to those of the original model (correlation coefficients around $9.997e{-1}$).

The hardware implementation of the discretization arises two difficulties: the network associates one cell to each data input, which entails networks with a very large number of neurons in typical CNN-based applications; and all of the cells work in parallel, generating simultaneously their contributions to the system output. For video processing applications, this means that using a CNN to process 8-bit 640×480-pixel images would require an FPGA with more than 2,400,000 pins and a huge area to accommodate more than 300,000 neurons. In practice, this is unfeasible with present off-the-shelf FPGAs.

To break these area difficulties the solution is time-multiplexing, using one functional unit to execute multiple operations. So an N-cell CNN can be folded in just one-cell CNN, keeping the area consumed to a minimum. With this approach, the computation of a CNN is equivalent to a single neuron which shifts along the input array.

However, two problems rise with this solution. On the one hand, the reduction is at expense of increasing the computation time by the same ratio N. Once again,there exists a limit for the size of the CNN, now fixed by the maximum FPGA clock frequency and by the desired performance. Following the same example of 640×480-pixel video, with the folded structure and considering a 10-stage cell, it is necesary to execute more than 3 million convolutions with each template in the cell per image. Considering 40 ns for each convolution (as stated in Section 2.1), it takes 120 ms per image, hardly 8 frames per second. If a higher frame rate is desired, the number of convolutions must be reduced,

0-7695-2940-2/07 $25.00 © 2007 IEEE
DOI 10.1109/FCCM.2007.25

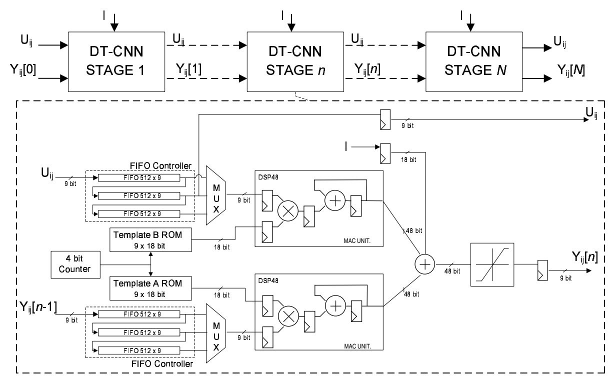

Figure 1. Stages pipeline and architecture.

at the expense of losing accuracy or making the CNN array smaller.

On the other hand, the iterations that each cell must compute demand the storage of the output of each stage. For video applications, due to the size, it is not possible to store each iteration output in the FPGA internal memory, so that an external memory devices must be used. This complicates the system design and adds new constraints, like the memory timing specifications.

To overcome these problems a novel architecture is proposed, where the CNN array is folded in just one cell, but this cell is unfolded in as many stages as iterations are required. Instead of executing the N successive iterations on the same hardware cell, N hardware cells have been pipelined; the area consumed by the cell is N times bigger than in the fully-folded version, but the computations are N times faster. Besides, the problem of the data storage between stages is overcome, since now the system can work on data streaming. Instead of using external memory to store the full output of each stage, internal buffers store just the data required for the computations of each stage.

As shown in Fig. 1, the stage has two input ports and two output ports: the input U_{ij} and the output from previous stages $Y_{ij}[n–1]$; and the output data $Y_{ij}[n]$ and the input data, which is delayed as many cycles as the stage latency requires in order to ensure the data synchronization. The presence of the input data in an output port makes it possible the pipelining of stages all working with the same input data, which allows the convergence of information and thus to build networks where one output can be function of multiple inputs, far more than the 3×3 template. In each stage, the inputs can be adjusted by different templates A and B. It allows designing powerful non-linear CNNs.

2.1 DTCNN cell implementation

The cell has been designed using low level RTL description and component placement tecnhiques.The Xilinx Virtex-4 devices have been the hardware platform selected. As shown in Fig. 1, two DSP48 slices perform the multiplication by the templates. The FIFOs, based on the BlockRAM, manage the data stream between stages. The cell works internally with 48-bit resolution, while the input and output data are 9-bit resolution and the templates 18-bit. A summary of timing information and the occupied resources when implementing a 10-stage cell in a Xilinx XC4VSX35 is included in Table 1. The pixel clock frequency implies that the cell can process 640×480 video data at 78 fps or 1024×1024 at 23 fps. The cell hardly occupies the 11% of the XC4VSX35. Therefore a 9-layer CNN, making up an almost 3 million neurons CNN, can be implemented with nine of the proposed cells in the mentioned FPGA.

Table 1. Summary of timing and used resources for a 10-stage cell.

Resources	Units	% Used
Area (slices)	1500	9.8
Flip flops	1720	5.7
DSP48	20	10.5
BlockRAM	20	10.5
Max. internal freq. (MHz)	410	-
Max. pixel Clk. (MHz)	24.11	-

If the number of iterations is reduced, the length of the cell pipeline is shortened and the number of layers of the CNN can be increased in the same ratio. In the limit point, when just one iteration is considered for approximating the discrete model to the analogue neuron, the cell consists of one stage and the number of layers can be increased by 10. This means a 95-layer CNN with 30 million neurons implemented on a low price FPGA. With a bigger device, the CNN can be even bigger. For example, an XC4VSX55 can accommodate a 50 million neurons CNN, which can process 640×480 78 fps video in real time. If a bigger CNN is still desired, the architecture proposed has been designed regarding the interconnection with other FPGA devices in pipeline. This greatly facilitates the implementation of CNN with hundreds or thousands of millions of neurons in a multi-FPGA system. These characteristics confer the proposed architecture a great advantage against other implementations in ASIC (e.g [2]) and FPGA (e.g. [3]), where the number of cells is considerably smaller.

Acknowledgement: This research has been funded by MTyAS of Spain, IMSERSO RETVIS 150/06.

References

[1] J.J. Martínez, F. Toledo, and J. M. Ferrández. Implementation of a discrete cellular neuron model (DT-CNN) architecture on FPGA. In *Proc. 2nd SPIE Int. Conf. on bioengineered and Bioinspired Systems*, volume 5839, pages 332–340, 2005.

[2] R. Carmona, F. Jimenez, R. Dominguez, S. Espejo, and A. Rodriguez. CMOS realization of 2-layer CNN universal machine chip. In *Proc. 7th IEEE Int. Work. on Cellular Neural Networks and Their Applications*, pages 444–451, 2002.

[3] Z. Nagy and P. Szolgay. Configurable multilayer CNN-UM emulator on FPGA. *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, 50(6):774–778, 2003.