Implementation of Nonlinear Template Runner Emulated Digital CNN-UM on FPGA

Z. Kincses* Z. Nagy† P. Szolgay‡
*†Department of Image Processing and Neurocomputing, University of Pannonia, Hungary
*e-mail: kincsesz@vision.vein.hu
†e-mail: nagyz@almos.vein.hu
‡Analogic and Neural Computing Laboratory, Computer and Automation Institute of HAS, Budapest, Hungary
e-mail: szolgay@sztaki.hu

***Abstract*— In the original CNN paradigm template values are defined as constants but several complex tasks can be efficiently solved by using nonlinear weights between the CNN cells. Unfortunately programmable nonlinear weights can not be implemented by using present day analog VLSI technology. In this paper a new emulated digital CNN-UM architecture will be presented which makes it possible to use zero and first order nonlinear templates during emulation. The new architecture is based on the Falcon emulated digital CNN-UM architecture and implemented on FPGAs. The computing precision of the architecture is configurable and the area/speed/accuracy tradeoffs are investigated.**

***Index Terms*— Reconfigurable architectures, Cellular neural networks, Nonlinear CNN template, Field programmable gate arrays**

I. Introduction

THE Cellular Neural Network (CNN) was invented in 1988 [1]. The nonlinear CNN paradigm was developed by Roska and Chua [2]. In case of nonlinear CNN the template values are defined by a nonlinear function of input variables (nonlinear B) and the output variables (nonlinear A). Most of image processing problems can be solved using linear CNN templates but using nonlinear CNN templates several complex image processing tasks can be solved more easily such as histogram generating [3], Hamming distance computing [3], grayscale skeletonization [4]. It is notably useful when the nonlinear CNN has just a nonlinear feed-forward template, because less time is required to solve the problem. The linear CNN has several implementations the software, the emulated digital VLSI (ASIC/FPGA) and the analog VLSI implementation. However several studies proved the effectiveness of the nonlinear CNN templates it is not supported on the recent CNN implementations. Analog VLSI implementation of a programmable nonlinear CNN is difficult by using present day technologies, so the only way is using software simulation, but it has even lower performance than in the case of linear CNN. Emulated digital implementations can be very efficiently used in the emulation of linear CNN arrays [5]. Additionally the flexibility of the FPGA implementation can be exploited to handle nonlinear CNN templates. In the next section two classes of nonlinear templates will be introduced. After a brief introduction of the Falcon emulated digital CNN-UM architecture the required modifications are described to make it possible to use nonlinear CNN templates on this architecture.

II. Types of Nonlinearity

In case of nonlinear CNN some template values are defined by a nonlinear function. This value depends on the difference of the currently processed cell and the value of the cell belongs to the actual template element. Investigating the CNN Template Library two types of nonlinear templates were defined. These are the zero and the first order nonlinear templates. Classifications of the different nonlinear templates are shown in Table I.

In case of the zero order nonlinear templates the nonlinear function contains horizontal segments only as shown in Fig1. Zero order nonlinear templates are used for example grayscale contour detection [6].

In case of the first order nonlinear templates the nonlinearity contains straight line segments as shown in Fig1. These kinds of nonlinear templates are used for example global maximum finding [7]. Naturally there are nonlinear templates where the template values are defined by two or more different nonlinearities, for example grayscale diagonal line detection [8].

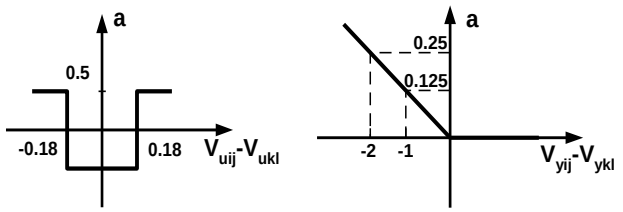

Fig. 1. Zero and first order nonlinearity.

TABLE I
CLUSTERING OF NONLINEAR TEMPLATES

ZERO ORDER NONLINEAR TEMPLATES	FIRST ORDER NONLINEAR TEMPLATES
Contour Extraction	Gradient Intensity Estimation
Game of Life DTCNN	Shortest Path (Explore)
Grayscale Diagonal Line Detector	Gradient Detection
Grayscale Line Detector	1-D Array Sorting
Grayscale Mathematical Morphology	Global Maximum Finder
Grayscale Skeletonization (Selection)	Hamming Distance Computation (Min. Distance)
Hamming Distance Computation (Differences)	Grayscale Skeletonization (Replacement)
Histogramm Generation	Tresholded Gradient
J-Function of Shortest Path (Minimum Selection)	J-Function of Shortest Path (Increased J-Function)
Local Maxima Detector	DepthClassification
Majority Vote Taker	Nonlinear Wave Metric Computation (Current Filling)
Median Filter	Spike Generation 4
Parity Counting	
Sorthest Path (Select)	

III. THE FALCON ARCHITECTURE

The original Falcon architecture has four main parts. These are the Memory Unit, the Template Memory, the Mixer, and the Arithmetic Unit. The Memory Unit stores a three line wide belt of the processed picture supposing nearest neighborhood templates. The Mixer stores the neighborhood of the currently processed cell. These elements are required to decrease the I/O bandwidth of the processor. The Template Memory stores the template values. And finally the Arithmetic Unit calculates the new state value of the cell. The architecture of the original Falcon processor is shown in Fig. 2.

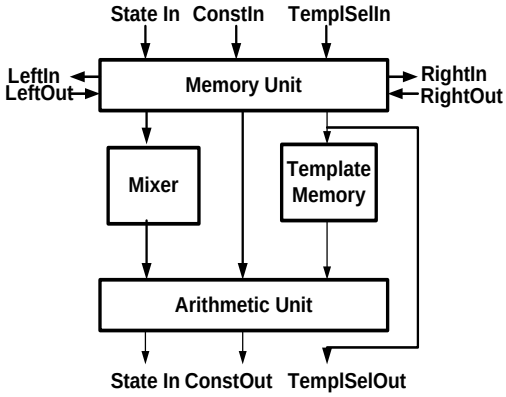

Fig. 2. The original Falcon architecture.

IV. THE MODIFIED FALCON ARCHITECTURE

To implement the nonlinear template runner emulated digital CNN-UM architecture the original Falcon structure was modified as follows. The Memory, the Mixer and the ALU are the same as with the original Falcon processor. But the Template memory was changed to be able to handle the nonlinear templates and their nonlinearity. These changes will be introduced in the next two subsections. As known the actual nonlinear template values are defined by the nonlinear function of the difference of the currently processed cell and the value of the cell belongs to the actual template element. So not only are the data from the Memory is needed but also the data from the Mixer are required to define the actual nonlinear template value so the outputs of the mixer was also connected to the Template memory.

V. ZERO ORDER NONLINEAR TEMPLATE MEMORY

In the original Falcon processor the template operations are performed row-wise. A RAM belongs to every column of a template in the Template memory. The actual template values are read out from the RAMs and transmit to the input of the ALU. In the Template memory of the modified Falcon processor also a RAM belongs to every column of the template but the values of the segments of the nonlinearity are stored in the RAMs. So in case of n segments the RAMs are n times larger. In the example the nonlinearity was partitioned into four segments as shown in Fig. 3. The segments are loaded into the RAMs as shown in Table II. We act upon this way because the two MSB bit of the difference which mentioned above is used to address the RAMs. In general the number of segments is power of two and more MSB bits are required.

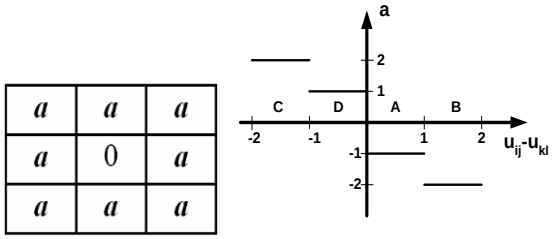

Fig. 3 Example zero order nonlinearity.

TABLE II
THE CONTENT OF THE RAMS

Address (Segments)	RAM1	RAM2	RAM3
0(A)	-1	-1	-1
1(B)	-2	-2	-2
2(C)	2	2	2
3(D)	1	1	1
4(A)	-1	0	-1
5(B)	-2	0	-2
6(C)	2	0	2
7(D)	1	0	1
8(A)	-1	-1	-1
9(B)	-2	-2	-2
10(C)	2	2	2
11(D)	1	1	1

The zero order Falcon processor is shown in Fig. 4. It contains three RAMs where the values of the segments of the nonlinearity are stored. The three subtractors are used to compute the difference for addressing as mentioned above. The task of the Shift registers is to timing the data from the Mixer and the Memory.

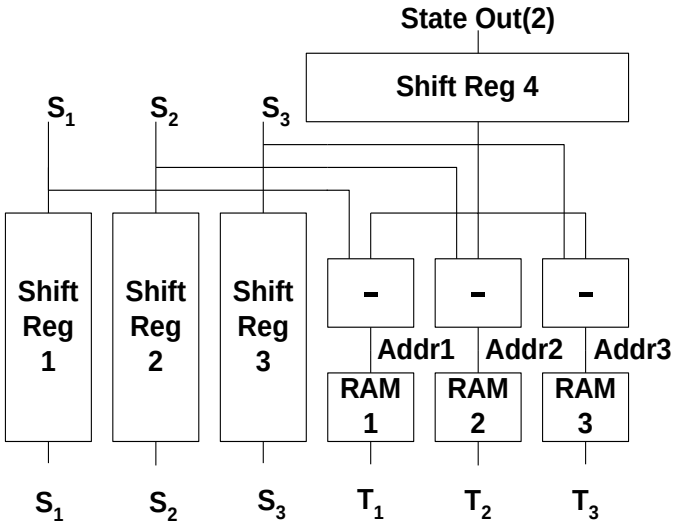

Fig. 4. The zero order nonlinear template memory.

VI. FIRST ORDER NONLINEAR TEMPLATE MEMORY

In this case the nonlinear characteristic is defined by a set of pice-wise linear function. So in the case of the Template memory of the first order Falcon processor six RAMs are required. The RAM1, RAM2 and RAM3 store the gradient of the function in the section of nonlinearity. The RAM4, RAM5 and RAM6 store the constant shift of the function in the actual section of nonlinearity. The readout of these RAMs is the same as the case of the zero order nonlinear template memory. To get the nonlinear template value the constant value should be added to the product of the address used for readout and the adequate gradient value. So in this case three adders and three multipliers are required, which raise the latency of the Template memory. So the Shift registers are longer to eliminate this additional latency. The first order nonlinear template memory is shown in Fig. 5.

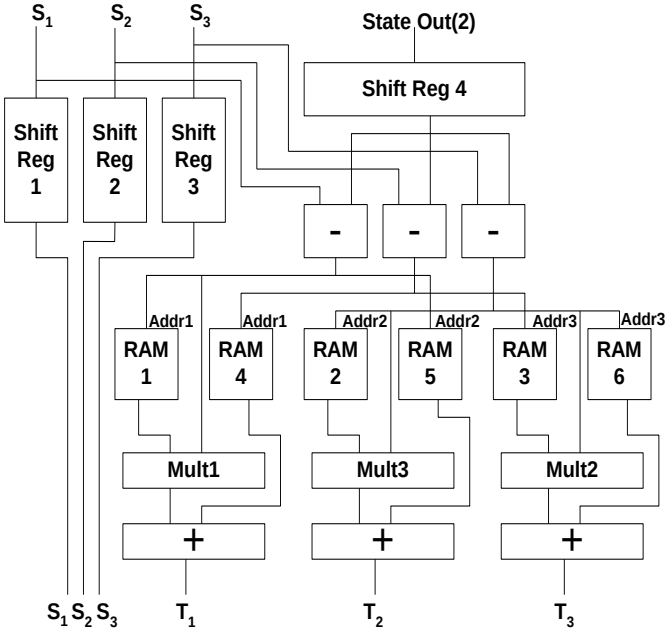

Fig. 5. The first order nonlinear template memory.

VII. TESTING

A. The implementation of the Falcon processor on the FPGA

The Falcon processor was implemented on the Celoxica RC203 development platform board for testing [9]. This board contains a Virtex-II 3000 FPGA [10] chip 4 Mb SRAM and connects to the computer with a parallel port. To implement this processor on the FPGA the Handel-C high level hardware description language and DK Design Suite [9] were used from Celoxica Inc. To build a real image processing system interfaces are needed to the memory, the parallel port and the video in and out. To store the partial results of the calculation the processor can access the ZBT memory through the ZBT interface. The work of the Memory Arbitration unit is to decide which unit can use the memory. The FIFOs match the bit width of the ZBT memory and the functional elements. The Parallel Port interface is used to connect to the parallel port and to configure the processor. The VGA interface is used to connect the monitor as the output of the processor and the Camera interface is used to connect the Camera as the input of the processor. These interfaces are part of the Platform Abstraction Layer (PAL) API from Celoxica. The real operable system on FPGA is shown in Fig. 6.

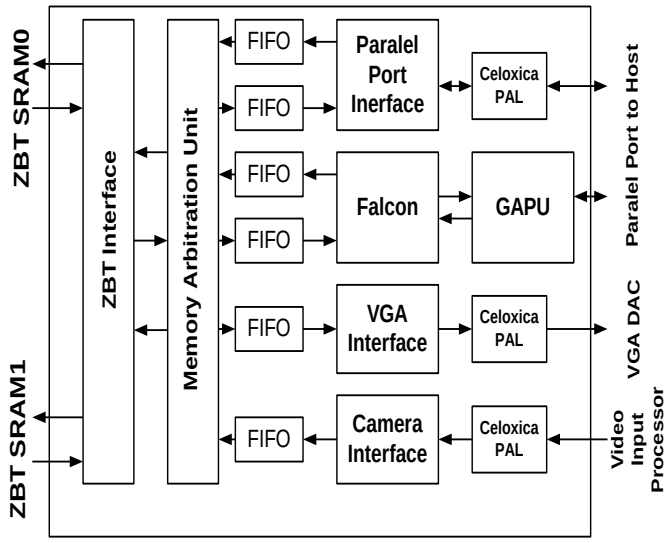

Fig. 6. The real operable system on FPGA XC2V1000/XC2V3000.

B. Area requirement

Similarly to the original Falcon processor the precision and the number of fraction bits of the state value, the constant value (which is determined by the feed forward equation) and the template value also can be configured in the case of the zero and first order nonlinear Falcon processor. The large number of possible configurations does not make it possible to investigate all cases. We just want to represent how the area requirement changes in case of different bit width and precision of data. In this test the precision of gij is set to 10 bit and the number of fraction bits is set to 6. The precision of the template values are set to 9 and the number of fraction bits is set to 7, which seems to be enough for most applications. The area requirements of the zero (A) and first (B) order processor in case of different state width are shown in the Fig. 7.

The investigation shows that the general resource requirement of the zero and the first order Falcon processor depends on the precision of the state value linearly. Although the requirement of the dedicated resource of these processors also depends on the precision of the state value. If the precision of the template values are changed, a similar behavior can be observed. In the next step it was investigated how many zero (A) and first (B) order Falcon processor can be placed on different kind of FPGAs if the precision of the state value is increased from 4 to 36. The results are shown in Fig. 8.

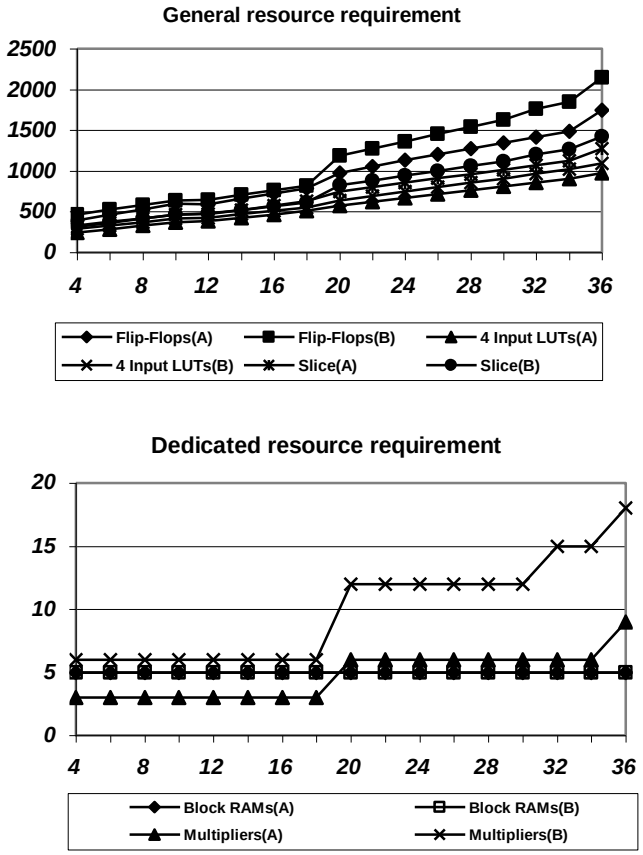

Fig. 7. The area requirement of the zero (A) and first (B) order Falcon processor (template width: 9 bit, constant width:10 bit)

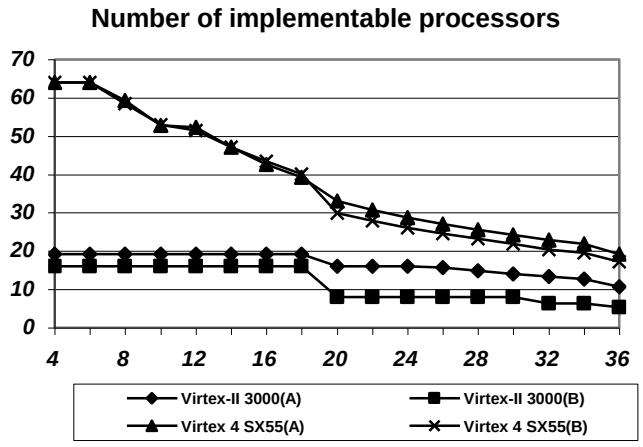

Fig. 8. The number of implementable zero (A) and first (B) order Falcon processors on different FPGAs. (template width: 9 bit, constant width:10 bit)

C. The speed of the processors

The performance of the zero and first order Falcon processors are compared to the software simulation. During the comparison 18 bit state and 9 bit template precision is used. The performance of the software simulation was measured on an Athlon64 3200+ processor running on 2GHz clock frequency. The measured computing performance of this processor was 2 million cell iteration/s in the case of zero order nonlinear templates while 1.5 million cell iteration/s was measured in the first order case.

In the case of Virtex-II 3000 FPGA nineteen zero order processors can be implemented. The maximum clock frequency of the processor is limited to 133MHz by the speed of the memories on our RC203 card. The cumulative computing performance of these processors is 842 million cell iteration/s, which is 421 times faster than the software simulation. If the currently available largest FPGA the

Virtex-4 SX55 is used 39 processors can be implemented and 400MHz clock frequency can be achieved. The resulting computing performance of the processor array is 5.2 billion cell iteration/s, which is 2600 times faster than the software simulation.

Implementation of the first order nonlinear Falcon processor requires additional flip-flop and LUT resources compared to the zero order case. However these elements can be packed more densely and in our test case smaller amount of slices is required. This makes it possible to implement 16 processors on the Virtex-II 3000 and 40 processors on the Virtex-4 SX55 FPGA. The maximum clock frequency of the first order nonlinear processor is not affected by the higher order nonlinearity thus 133MHz and 400MHz clock frequency can be achieved on the Virtex-II 3000 and Virtex-4 SX55 respectively. The computing performance of the first order processors is similar to the zero order case. However the performance of the conventional microprocessor is smaller in this case thus our processors provide higher speedup. Finally the computing performance of the zero (A) and first (B) order Falcon processors was compared to the software simulation in case of Virtex-II 3000 and Virtex 4 SX55. This is shown in Fig. 9.

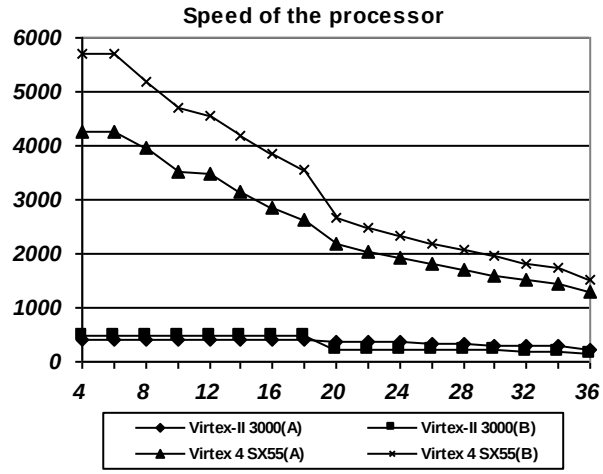

Fig. 9. The comparison of the computing performance of the zero (A) and first (B) order Falcon processors on different FPGAs to the software simulation.

VIII. CONCLUSION

The original Falcon processor was redesigned to be able to handle the zero and first order nonlinear template and its nonlinearity. To compute the zero and the first order template elements the template memory is significantly modified. In the zero order case the area requirement of the new processor does not increase significantly while in the first order case additional dedicated resources (MULT18x18) are required. The new architectures are implemented on our RC203 prototyping board and 421 times speedup is measured compared to an AMD Athlon64 3200+ microprocessor. Using the currently available largest FPGA even 2600 times higher performance can be achieved which makes our architecture ideal for real-time image processing tasks.

REFERENCES

[1] Chua, L.O., Yang, L., Cellular Neural Networks: Theory, IEEE TRAMs. on Circuits and Systems, 1988, (35): 1257-1272.

[2] T. Roska and L. O. Chua, "Cellular Neural networks with nonlinear and delay-type template elements and non-uniform girds," Int. J. Circuit Theory and Applications, vol. 20, pp. 469-481, 1992

[3] P. L. Venetianer, K. R. Crounse, P. Szolgay, T. Roska, and L. O. Chua, "Analog Combinatorics and Cellular Automata – Key Algorithms and Layout Design using CNN", Proceedings of the International Workshop on Cellular Neural Networks and their Applications CNNA, 1994, pp. 249-256, Rome.

[4] P. L. Venetianer, F. Werblin, T. Roska, and L. O. Chua, "Analogic CNN Algorithms for Some Image Compression and Restoration Tasks", IEEE Transactions on Circuits and Systems, 1995, Vol. 42, No. 5.

[5] Nagy, Z., Szolgay P., Configurable Multi-Layer CNN-UM Emulator on FPGA, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2003, (50): 774-778.

[6] T. Boros, K. Lotz, A. Radványi, and T. Roska, "Some Useful New Nonlinear and Delay-type Templates", Research report of the Analogical and Neural Computing Laboratory, Computer and

Algorithmic Research Institute, Hungarian Academy of Science, 1991, (MTA SzTAKI), DNS-1-1991, Budapest.
[7] L. O. Chua, T. Roska, T. Kozek, and Á. Zarándy, “The CNN Paradigm – A Short Tutorial ”, Cellular Neural Networks, T. Roska, and J. Vandewalle, editors, John Wiley and Sons, New York, 1993, pp. 1-14.
[8] CNN Software Library, ANALOGIC Computers LTD, Budapest, 2000.
[9] Xilinx Inc. Homepage: www.xilinx.com
[10] Celoxica Homepage: www.celoxica.com