Implementing CASTLE emulated digital processor array and interfacing with physical test environment into the ALADDIN system

P.Tóth, Zs.Vörösházi

Image Processing and Neurocomputing Department
Veszprém University

Egyetem u. 10, H-8200 Veszprém
Hungary
tothpeter@lab.analogic.sztaki.hu,voroshazi@freemail.hu

P.Jónás, P.Szolgay,
T.Hidvégi*, P.Keresztes*, I.Sütő*
Analogic and Neural Computing Laboratory
Computer and Automation Research Institute of
Hungarian Academy of Sciences
Lágymányosi u. 11, H-1111 Budapest
*Automation Department, Széchenyi University
*Egyetem tér 1. Győr, Hungary

Abstract. – **The architecture and the physical test environment of the CASTLE emulated-digital CNN-UM chip, which were integrated into the Aladdin hardware and software system and used for CNN research are introduced. The CASTLE processor array was designed by have 2*3 processing elements (2 rows / 3 columns) with 12/6/1-bit controllable accuracy, and with an arithmetic accelerated by pipe-lining technique. Moreover, cascading of this architecture supports the building up of arbitrary size arrays. Due to these advantageous features it will be used in problems where the analog VLSI approaches have some difficulties. The CASTLE architecture was implemented 0.35µ by using CMOS VLSI technology (2 polysilicon + 3 metal layers).**

I. INTRODUCTION

The CASTLE emulated digital CNN implementation (the operation of the regular structured analog-dynamical processor arrays emulating with digital processor elements) focus on the solution of some problems occurred in the analog VLSI implementations of a Cellular Neural Network - Universal Machine (CNN-UM) [1],[2],[3]. The precision, the usage of nonlinear template operations and the implementation of multilayer structures are some of the problems to be solved by an emulated digital CNN-UM solution. The operational speed of the CASTLE should not be drastically smaller than an analog CNN-UM.

The CASTLE processor array was implemented using a 0.35µ CMOS technology with variable accuracy and with a 1ns/ virtual CNN cell/ iteration speed, supposing 12-bit accuracy. It was designed to interface into Aladdin System where the hardware-software environment provides the same high-level support to develop analogical CNN algorithms that are generally used in case of the analog VLSI CNN-UMs.

II. The derivation of CASTLE computational model

The original dynamical state equation of the CNN model (1) from Chua-Yang has the following form:

$$\dot{x}_{ij}(t) = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}\, y_{kl}(t) + \sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}\, u_{kl} + z_{ij}, \quad (1)$$

where, u_{kl}, x_{ij} and y_{kl} indicate the input, state and output variables (voltage values), respectively, while **A** means feedback, **B** is feed-forward template matrices, and Z_{ij} indicates the bias (constant current value). The domain of sum-operations $S_r(i,j)$ represents the r radius neighbourhood of cell (i,j).

The architecture of CASTLE processor was introduced as a systolic type implementation of forward EULER integration formula [4] of the former CNN equation:

$$x_{ij}(n+1) = (1-h)x_{ij}(n) + h\left(\sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}\, y_{kl}(n) + \sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}\, u_{kl} + z_{ij}\right), \quad (2)$$

In order to simplify the digital computations in the next step we try to eliminate all possible variables with the help of the (FSR) Full Signal Range model (3.):

$$x_{ij}(n+1) = \begin{cases} 1 & ha \quad v_{ij}(n) > 1 \\ v_{ij}(k) & ha \quad |v_{ij}(n)| \le 1 \\ -1 & ha \quad v_{ij}(n) < -1 \end{cases}, \quad (3)$$

$$v_{ij}(n) = (1-h)x_{ij}(n) + h\left(\sum_{C(k,l)\in S_r(i,j)} A_{ij,kl}\, x_{kl}(n) + \sum_{C(k,l)\in S_r(i,j)} B_{ij,kl}\, u_{kl} + z_{ij}\right)$$

By using this solution we consider that state variables equal to the corresponding output variables and truncate the state values: the absolute value of the state variables are not larger than 1.

A single iteration step of the forward EULER integration in the CASTLE processor is executed in two phases. In the first phase (4) the variables g_{ij} are calculated from the input variables u_{kl} and from the bias values z_{ij}, which are produced by the dimensionless time-step, h. The h and *(1-h)* constant coefficients are included in the modified **A1** and **B1** template matrices.

$$g_{ij} = \sum_{C(k,l)\in S_r(i,j)} B1_{ij,kl}\, u_{kl}(n) + hz_{ij}, \quad (4)$$

In the second phase (5.) the new values of the state variables are calculated:

$$x_{ij}(n+1) = \sum_{C(k,l)\in S_r(i,j)} A1_{ij,kl}\, x_{kl}(n) + g_{ij}, \quad (5)$$

So the iteration scheme was simplified to the 3*3 convolution and an additional term, which can also be performed easily using emulated-digital technique.

III. The architecture of the CASTLE processor array

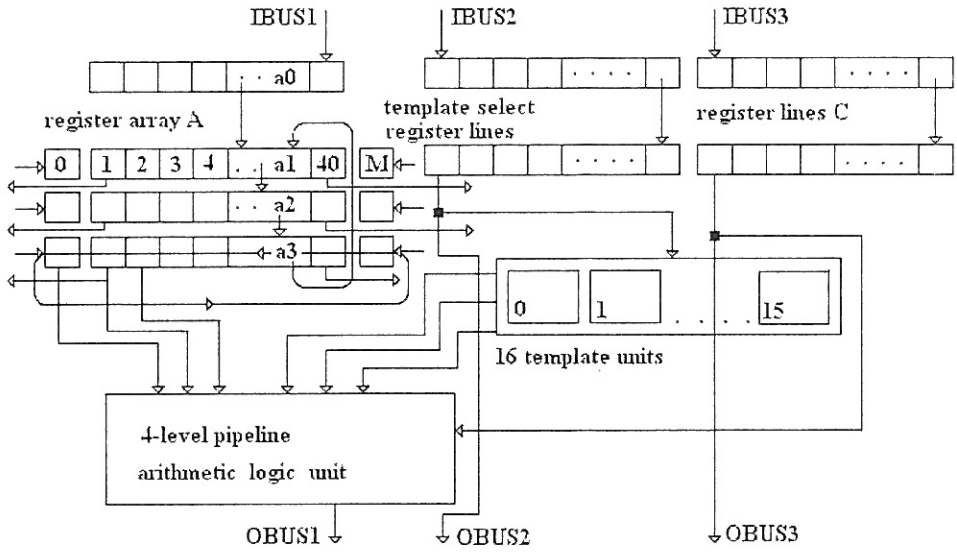

Fig. 1.. The register-transfer level architecture of a CASTLE processor element

A given processor element (Fig.1) in the array is ordered to a given number iteration step of the numerical integration of state variables, and it processes a 40-variable-wide, or a 80-variable-wide column of the state matrix depending on 12-bit or 6-bit mode, respectively. It can process a 40-binary-variable-wide column in logical (1-bit) mode.

A FIFO-type register line **a0** receives serially a 40-cell section of the input or state values of the actual input row via bus **IBUS1.** A cell represents a single state in 12-bit mode, and two states in 6-bit mode. In the logical mode 10-binary state-values enter parallel into the processor. The values of the state variable are serially filled into the register line. If the filling of a new line is completed, the new line enters into register line **a1**. Simultaneously, **a3** receives the content of **a2**, and a2 receives the content of a1.

So the register lines a1, a2 and a3 contain not only a part-row of the state variables, but the upper and lower neighbouring part-rows, too. Edge state-values belong to the rows, and serve the communication with the neighbouring processors. These edge registers are: a1(0), a2(0), a3(0), a1(M), a2(M) and a3(M).

An important characteristic of the CASTLE, that *one of 16-template matrices* can be assigned to each state value. The addresses of the template memory enter simultaneously with the state values via **IBUS2**. An additive variable is received via **IBUS3.** A different additive constant and one of the 16 template units can be ordered to each state value. The additive variables are hz_{ij} values in the first phase, and g_{ij} values in the second phase.

According to the Euler integration formulas (4) and (5), the *4-level pipeline arithmetic unit* [5],(Fig.2.) calculates the sum of products of 9 operand-pairs and the constant, and executes the limitation. In logical mode, the logic unit processes 10 states during a single clock cycle.

The input string of template-select addresses and the additive variables enter without any modification into the next row of processors.

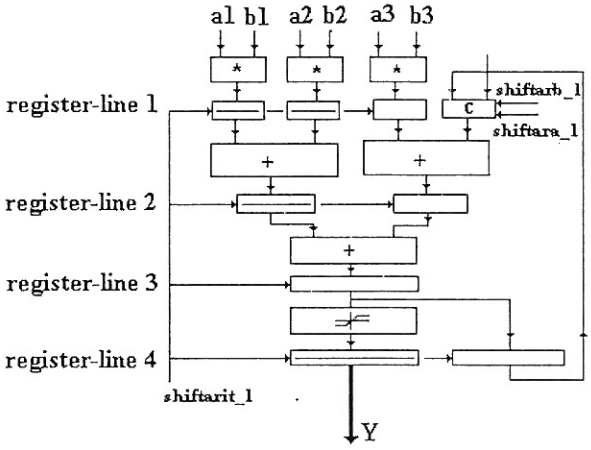

Fig.2. The 4-level pipeline arithmetic unit

A 2x3 array of CASTLE processor is shown in Fig.3. Except for the left and right processors, the interconnections are obvious. If the horizontal communication between the subarrays was solved in the same way as at the inner edges of the processors, the number of I/O ports would be extremely high. Consequently two bidirectional I/O buses serve this communication in time-multiplex mode.

Using a two phase clock signal, the *Timing and Control Unit* generates the register-tranfer signals. The output signals of unit *Front-End-Pointer* select the row of processors in which the front (the first line) of the state matrix or the end (last line of the state-matrix) is under processing. The processors of selected rows generate the top and bottom rows of the boundary states with a special control sequence. The left and right boundary states are generated also with a special control sequence, which is based on the positions of the processors. If a processor is assigned to be one of the left-edge or one of the right-edge processors of the array, the dedicated sequence of the

control does not invoke communication, but generates boundary states from the inner register-lines of the processor.

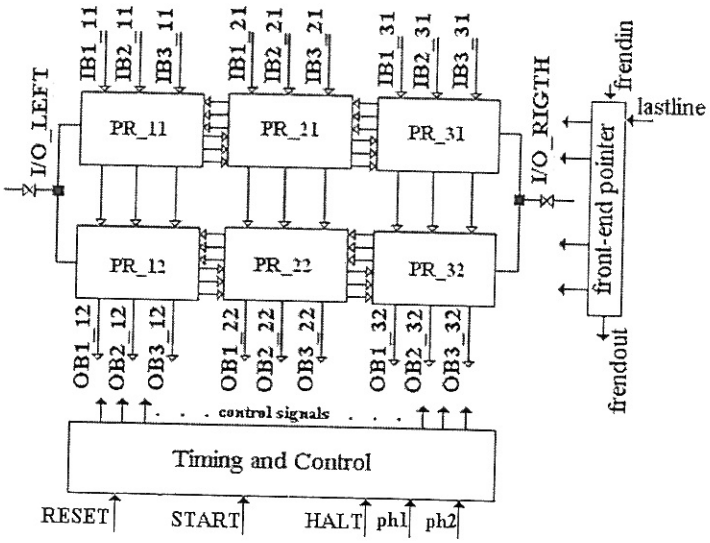

Fig.3. The 2 x 3 CASTLE array

IV. HARDWARE AND SOFTWARE INTERFACE COMPONENTS OF CASTLE ARRAY

A. Hardware and software interface components of CASTLE array

The CASTLE processor array is used with hardware and software support of Aladdin Pro System [6],[7]. This system includes three well-separable parts. The first part is the PC and the software, the second part of the system is the DSP card, and the third one is the platform with CASTLE chip. (Fig. 4.)

The software system can be programmed under a low-level AMC (Analogic Macro Code) or a high-level ALPHA language. The programmer surface of the system is called CNNedit. The program which was created under CNNedit is ready to run under CNNrun program. The CNNrun is in fact the running surface of the Aladdin.

The task of DSP card establishes the contact between the Aladdin system and CNN-UM chip. The DSP card (SPM6020) contains a type of TMS320C6202 chip from Texas Instruments and a PCI/104 interface. So we need another adapter-card which converts this PCI/104 into normal PCI interface. The DSP card includes an extended bus which is called platform-bus. With the help of this bus we can attach the hardware to DSP.

The Aladdin's operation system is called COS (Chip Operation System) runs on DSP. The task of the COS is to process the ABC code which was downloaded earlier into the memory of the DSP card. The operation of COS is similar to the interpreter because it executes the instructions row by row. If it is necessary the COS turns to the PC, otherwise it sends data to the CNN chip.

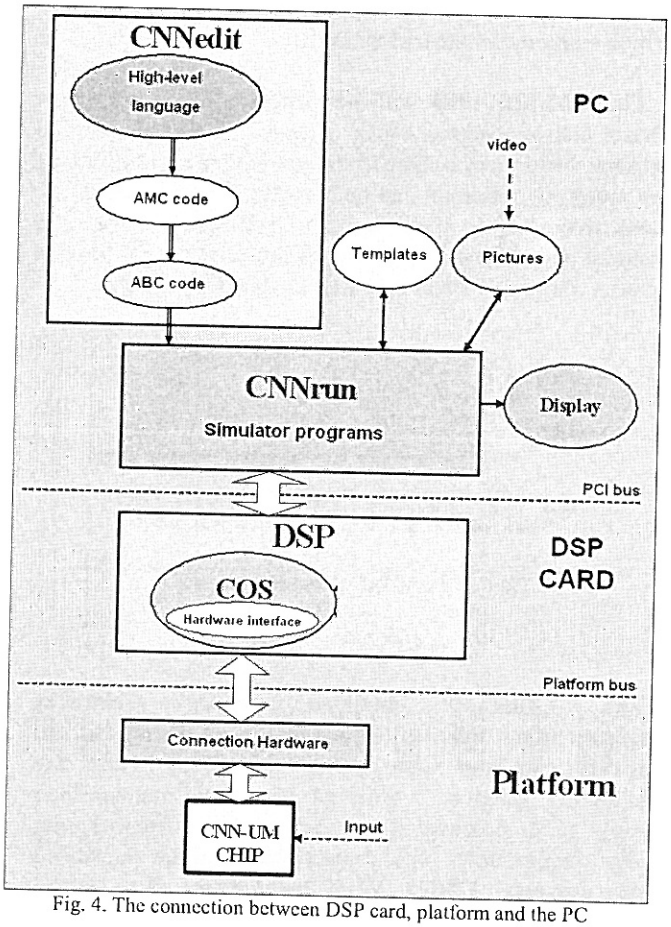

Fig. 4. The connection between DSP card, platform and the PC

B. CASTLE PLATFORM

The important part of the platform is the Xilinx FPGA. The main control functions of the chip were implemented in FPGA which programmable through JTAG port. Both of the CASTLE chip and the programmable FPGA work on 3.3V power supply. The block diagram of the platform is shown in Fig. 5.

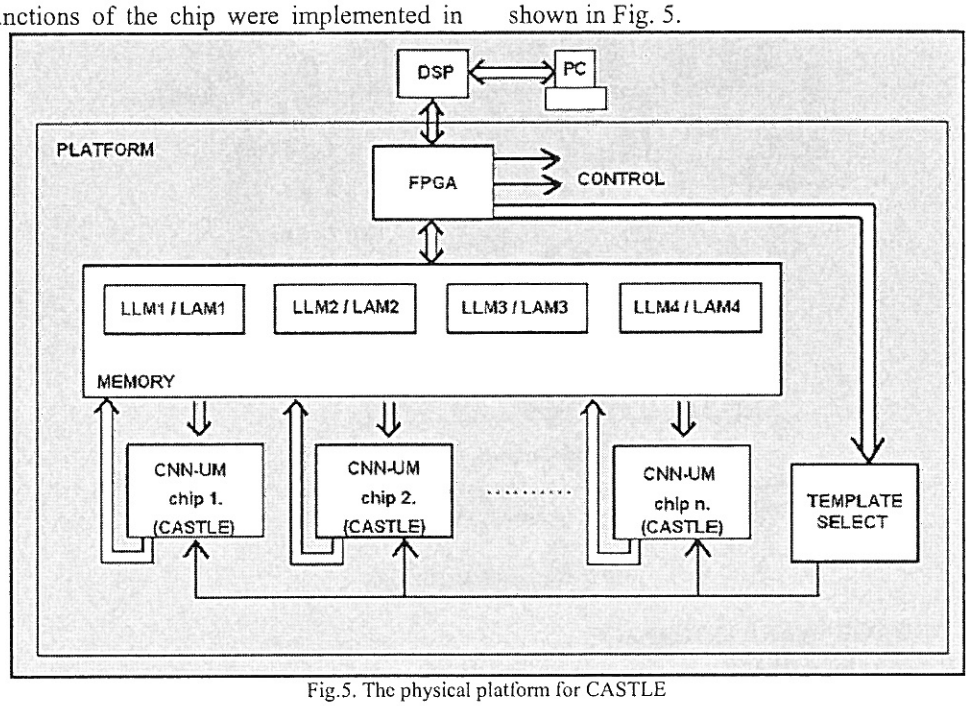

Fig.5. The physical platform for CASTLE

The LLMs/LAMs are built up from FIFO storage units. The number of these local logical and analog memory elements depend on the Aladdin development system, which supports 4 LAM/LLM units.

The CASTLE chip and the platform work with two-phase non-overlapped clock signals. These phases are presented in Figure 6. Due to this method the possibility of the incorrect operation can be avoided. The external CLK clock frequency is approximately 100 MHz but the chip contains an internal frequency-multiplicator so it can achieve about 200 MHz operational speed.

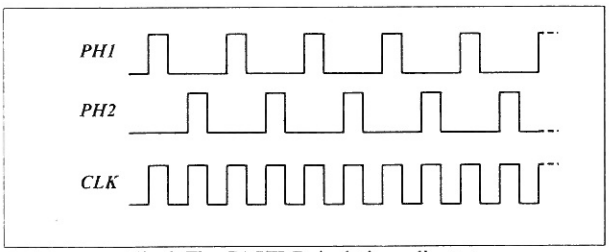

Fig 6. The CASTLE clock times diagram

V. CONCLUSION

The CASTLE emulated digital CNN-UM implementation includes the advantageous features of the software simulator and analog approaches. Only the operational speed is 2 order of magnitude smaller than analog VLSI, however it is more versatile, flexible, and easily configurable. The design time can be decreased using universal CMOS VLSI technology. Due to the surface optimalization (total chip area is only ~0.8 mm^2) CASTLE has smaller dissipated power (~0.8W), and it can be integrated directly into the digital environment building up hardware platform.

The developed CASTLE architecture is capable of carrying out complex image processing tasks in real-time. It could process a video stream with 240x320 resolution, in 25fps, at 12-bit accuracy. According to the cascading mechanism of the CASTLE chip (in horizontal or vertical direction) we can reach larger processing capability. The first Platform will contain 3 CNN-UM chips, in a row.

The dynamically scalable accuracy of the computation can be set by an external control signal to a given analogical algorithm. If we set bitvector mode, the accuracy is 12 bit or 6 bit, each of which can be set independently. If we choose binary mode, the accuracy is 1 bit. At the 12-bit accuracy this architecture provides 1ns / virtual CNN cell/ iteration speed.

The CASTLE chip and the hardware environment are being in production phase, we are planning to show some interesting examples running on this architecture, in the oral presentation.

VI. ACKNOWLEDGMENT

This work was sponsored by Info-Communications Technologies Applications, Ministry of Education, Hungary (grant no IKTA4-002/2001).

VII. REFERENCES

[1] T.Roska and L.O.Chua, "The CNN Universal Machine: an analogic array computer", IEEE Transactions on Circuits and Systems-II Vol.40, pp. 163-173, March, 1993.
[2] L.O.Chua and L.Yang, "Cellular neural networks: Theory and Applications", IEEE Trans. On Circuits and Systems, Vol.35, pp. 1257-1290, 1988
[3] A. Rodrigez –Vazquez, R. Dominguez – Castro, S. Espejo, "Challenges in Mixed-Signal IC Design of CNN

Chips in Submicron CMOS", Proc. of the IEEE CNNA'98, London pp: 13, April, 1998.
[4] Péter Keresztes, Ákos Zarándy, Tamás Roska, Péter Szolgay, Tamás Bezák, Timót Hidvégi, Péter Jónás, Attila Katona "An emulated digital CNN implementation" Journal of VLSI Signal Processing Systems Kluwer Academic Publishers, Vol. 23.pp.291-303, 1999.
[5] T. Hidvégi, P. Keresztes, P. Szolgay. "An accelerated digital CNN-UM (CASTLE) Architecture by using the pipe-line technique, Proc. of the IEEE CNNA'02 Frankfurt, 2002
[6] Á. Zarándy, "ACE Box: High-performance Visual Computer based on the ACE4K Analogic Array Processor Chip", Proc. of ECCTD'01, pp. I-361-I-364, Espoo, 2001
[7] Aladdin Pro. System, Analogic Comp. Inc. Budapest, 2002