A Cellular Network Architecture With Polynomial Weight Functions

Jens Müller, Jan Müller, Robert Braunschweig, and Ronald Tetzlaff

***Abstract*— Emulations of cellular nonlinear networks on digital reconfigurable hardware are renowned for an efficient computation of massive data, exceeding the accuracy and flexibility of full-custom designs. In this contribution, a digital implementation with polynomial coupling weight functions is proposed for the first time, establishing novel fields of application, e.g., in the medical signal processing and in the solution of partial differential equations. We present an architecture that is capable of processing large-scale networks with a high degree of parallelism, implemented on state-of-the-art field-programmable gate arrays.**

***Index Terms*— Cellular neural networks, field-programmable gate arrays (FPGAs), image processing, partial differential equations (PDEs), system-on-chip.**

I. Introduction

Since their introduction in 1988 [1], the cellular nonlinear networks (CNNs) have proved to be suitable for the image processing [2], medical signal processing [3], robot control [4], and solution of partial differential equations (PDEs) [5], [6], among others. Analog and mixed-signal implementations of the CNN universal machine [7] provide the exceptional computational performance of thousands of processing units on a single chip [8].

However, the precision of analog implementations is usually not sufficient for numerically sophisticated applications. Moreover, the design of these application specific integrated circuits (ASICs) is generally fixed and parameters like network size or data precision cannot be adjusted. Thus, the emulation of CNNs on reconfigurable digital devices, especially on field-programmable gate arrays (FPGAs), becomes attractive for prototyping and applications where flexibility and/or higher precision is required [9]–[12].

It has been shown that networks with nonlinear couplings are inevitable for numerous biologically motivated applications [3], [13] and especially for solving PDEs [6]. In [14], a full-custom mixed-signal chip with a polynomial-type CNN (PTCNN) has been proposed for the applications in EEG signal processing. This is considered the only practical implementation of a PTCNN so far.

In this contribution, we present an advancement of the recently introduced NERO architecture [10], [15], [16] to the TITUS architecture for the digital emulation of PTCNNs with an arbitrary polynomial order. In Section II, the underlying mathematical model is introduced. Section III is devoted to architecture modifications for the processing of nonlinear couplings. Some results of the implementation on Xilinx FPGAs are given in Section IV, and two application examples are discussed in Section V. Finally, the conclusion is drawn in Section VI.

II. PTCNN Model

A CNN is a regular system of processing elements (PEs) (cells) that are coupled to its neighbors in lateral and diagonal directions. In a common 1-neighborhood each cell is hence coupled to eight neighbors and to itself (also called 3×3 neighborhood). In a PTCNN, the couplings between the neighboring cells are represented by polynomial weight functions. Since a standard model has not yet been defined, we exclusively refer to feedback and feedforward terms of polynomial order $P \in \mathbb{N}^*$ [6], [17], which leads to a cell state defined by the differential equation

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{\substack{|k|\le 1\\|l|\le 1}} \sum_{p=1}^{P} a_{kl}^{(p)} \cdot y_{i+k,j+l}^{p}(t) + \sum_{\substack{|k|\le 1\\|l|\le 1}} \sum_{p=1}^{P} b_{kl}^{(p)} \cdot u_{i+k,j+l}^{p} + z. \quad (1)$$

To avoid confusion, y^p denotes the pth power of y, whereas in $a^{(p)}$, p is an index.

Using the forward Euler integration method, we obtain the discrete-time state equation of a PTCNN

$$x_{ij}(n+1) = \mathcal{N}\left(x_{ij}(n) + \sum_{\substack{|k|\le 1\\|l|\le 1}} \sum_{p=1}^{P} \hat{a}_{kl}^{(p)} x_{i+k,j+l}^{p}(n) + \hat{w}_{ij} \right) \quad (2)$$

with the nonlinear output function

$$\mathcal{N}(x) = \begin{cases} -1, & x < -1 \\ x, & -1 \le x \le 1 \\ 1, & x > 1 \end{cases} \quad (3)$$

and with the modified weights $\hat{a}_{0,0}^{(1)} = h \cdot (a_{0,0}^{(1)} - 1)$; $\hat{a}_{kl}^{(p)} = h \cdot a_{kl}^{(p)}$, $\forall (k,l,p) \neq (0,0,1), |k| \le 1, |l| \le 1, p \ge 1$; $\hat{b}_{kl}^{(p)} = h \cdot b_{kl}^{(p)}$, $\forall (k,l,p), |k| \le 1, |l| \le 1, p \ge 1$ and $\hat{z} = h \cdot z$. The modified offset term of the PTCNN is given as

$$\hat{w}_{ij} = \sum_{\substack{|k|\le 1\\|l|\le 1}} \sum_{p=1}^{P} \hat{b}_{kl}^{(p)} u_{i+k,j+l}^{p} + \hat{z} \quad (4)$$

replacing the lower term in (1). Applying the Horner scheme, the double sums in (2) can be rewritten as

$$\sum_{\substack{|k|\le 1\\|l|\le 1}} \sum_{p=1}^{P} \hat{a}_{kl}^{(p)} x_{i+k,j+l}^{p}(n) = \sum_{\substack{|k|\le 1\\|l|\le 1}} x_{i+k,j+l}(n) \cdot \Big[\hat{a}_{kl}^{(1)} + x_{i+k,j+l}(n) \big(\hat{a}_{kl}^{(2)} + \cdots + x_{i+k,j+l}(n) \hat{a}_{kl}^{(P)} \big) \cdots \Big]. \quad (5)$$

The double sum in (4) is written down similarly.

III. Hardware Architecture

A. NERO Architecture

A direct mapping of the cellular network structure to digital hardware is feasible and efficient [18], yet strongly limited by the available resources. For that reason, we designed the NERO architecture by mapping large-scale networks to medium- or small-scale processor

Manuscript received April 9, 2014; revised December 3, 2014; accepted January 25, 2015.

The authors are with the Faculty of Electrical and Computer Engineering, Institute of Circuits and Systems, Technische Universität Dresden, Dresden 01062, Germany (e-mail: jens.mueller1@tu-dresden.de; muellerj@iee.et.tu-dresden.de; robert-braunschweig@gmx.de; ronald.tetzlaff@tu-dresden.de).

Digital Object Identifier 10.1109/TVLSI.2015.2397449

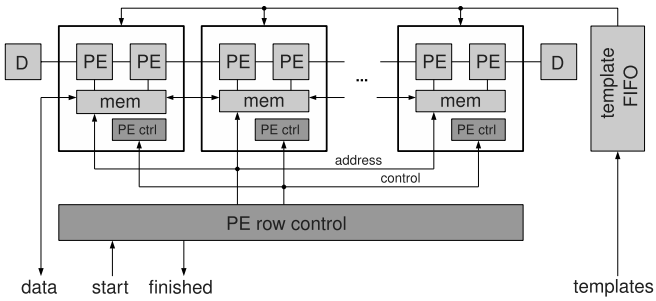

Fig. 1. Array architecture with a row of locally coupled PEs and additional PE dummies (D).

arrays, yet retaining the local couplings of the CNN paradigm and minimizing the number and length of interconnections between the PEs [10], [15]. To allow for many-iteration CNN operations like wave propagation and for complex algorithms with a number of successive operations on the same data, the complete input data is stored in local memory of the processor array.

In this architecture, the network is divided into vertical segments that are treated line-by-line and segment-by-segment by a row of n_{pe} PEs. Each PE is connected to its left and right neighbors, or to a dummy PE at either ends of the row, respectively. Fig. 1 shows the structure of the 1-D processor array with the main control and data paths. For an optimal hardware utilization (Section IV-A), two PEs share a common memory unit (mem) and a PE control unit. Both the PEs and the mems are coupled only locally to moderate the routing complexity. The left and right PE dummies (D) have no calculation core and are required only to provide data from the borders to the neighboring segments, or to ensure the network boundary conditions, respectively. The template values (weight coefficients) of the operation are stored in a template first-in-first-out that is designed as a ring buffer. The structure of the PEs, mems, and controlling modules are explained in [10] and [15].

B. TITUS—Extension for PTCNN

The modifications to NERO to process the PTCNN state equation (2) comprise the calculation core of the PE, the local controller unit, and the template FIFO. The novel architecture of the calculation core will be explained subsequently.

The utilization of the Horner scheme (5) exhibits some advantages over the conventional form of the PTCNN state equation (2)—the powers of the state variables do not have to be stored explicitly in registers, and the interconnections within the PE can be retained.

With an increasing polynomial order, each cell state computation easily requires dozens of MAC operations. Therefore, a further parallelization inside the PE becomes reasonable. Equation (5) permits a semiparallel or fully parallel computation for all neighbors, followed by an accumulation of the intermediate results. To avoid an unbalanced workload, the utilization of one, three, or nine DSP elements is favorable for the given 1-neighborhood. Thus, a trade-off between resource requirements and performance is inevitable.

For the targeted FPGA platform, the usage of three DSPs rendered the best choice since the on-chip DSP slices feature a three-stage pipeline that can be utilized efficiently to store three intermediate results. This configuration allows each DSP to calculate the three polynomials in a pipeline with single clock cycle latency and without any interruption.

The structure of the extended calculation core is shown in Fig. 2. It comprises three DSPs for the left, central, and right column, each processing the contribution of three neighboring cells to the state equation. Each DSP (shown in Fig. 3) calculates three elements of the Horner scheme

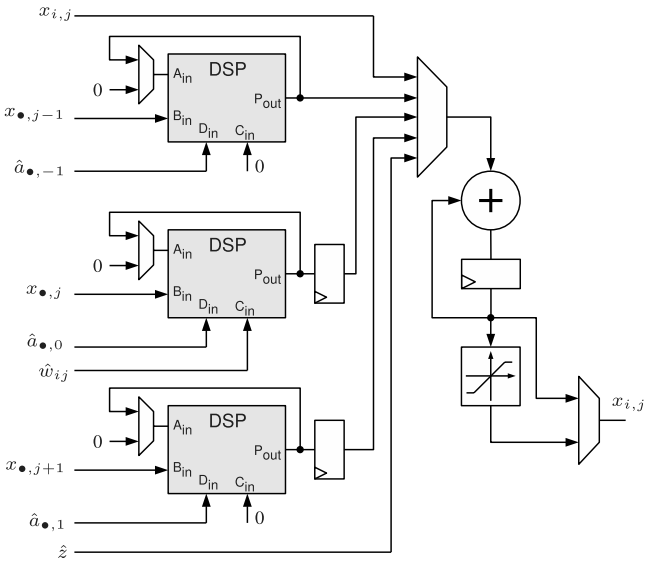

Fig. 2. TITUS calculation core for the emulation of a PTCNN—the place-holder (•) refers to all rows of the neighborhood.

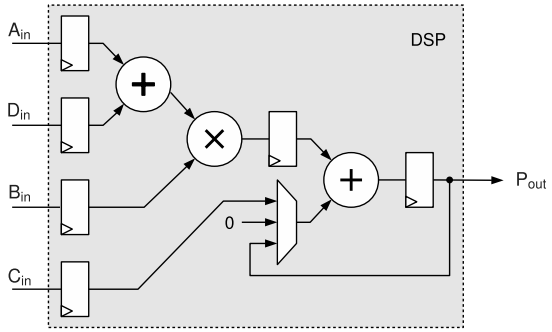

Fig. 3. Structure of the DSP core.

$$m_{k+1} = x \cdot (a + m_k) \tag{6}$$

with $m_0 = 0$.

The number of iterations of (6) depends on the selected polynomial order. The results are accumulated, added to $\hat{w}_{ij}$, and finally constrained to the range $[-1; 1]$ with the operator $\mathcal{N}$. This constraint can be bypassed for the calculation of the offset term $\hat{w}_{ij}$ that should be stored with the highest possible accuracy to prevent numerical problems. In order to provide the template values correctly, it is also necessary to extend the template FIFO module (Fig. 1) by replacing the single buffer with three buffers operating in parallel.

IV. Results

The architecture has been implemented on a Xilinx Virtex-6 FPGA and a Xilinx Zynq-7000 system-on-chip. The following specifications apply to the Xilinx Virtex-6 LX240T.

A. Resource Requirements

As explained in Section III-B, the modified architecture employs three DSPs for each PE. The utilization of lookup table and flip-flop resources depends on the data precision and the number of PEs. Only the data width for the highest precision of state and

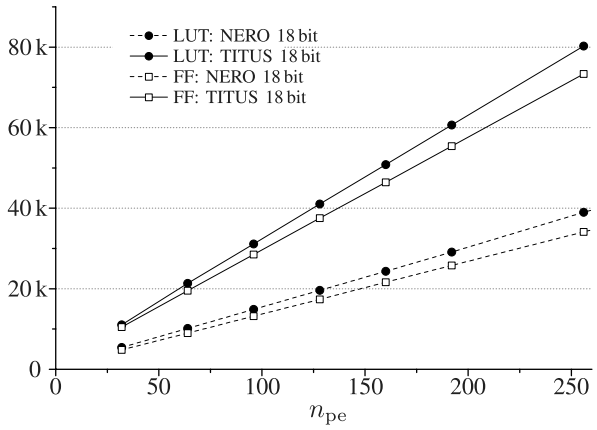

Fig. 4. Resource requirements depending on n_{pe} for standard CNN and PTCNN emulations at an 18-bit state representation. FF—flip-flop. LUT—lookup table.

input values is considered here

$$W_{\mathrm{state}} = W_{\mathrm{input}} = W_{\mathrm{data}} = 18 \text{ bit}. \tag{7}$$

The usage of a lower precision should be handled with care since in the calculation chain of the Horner scheme, each multiplication is restricted to 25- and 18-bit input precision. In Fig. 4, the resource utilization over n_{pe} is shown for TITUS and NERO, for the same state precision.

For each PTCNN cell, $4 \cdot W_{\mathrm{data}}$ cells of on-chip memory are allocated, leading to a total memory requirement of

$$c_{\mathrm{total}} = 4 \cdot M \cdot N \cdot W_{\mathrm{data}}. \tag{8}$$

With the given configuration of two PEs sharing one block-RAM (BRAM), each BRAM has to store an amount of

$$c_{\mathrm{bram}} = \frac{8 \cdot M \cdot N \cdot W_{\mathrm{data}}}{n_{\mathrm{pe}}}. \tag{9}$$

The network size is limited by the available BRAM size. For the given Virtex-6 LX240T FPGA, we successfully implemented a PTCNN of 256 $\times$ 256 cells using 256 PEs, with a data precision of $W_{\mathrm{data}} = 18$ bit.

B. Performance

The performance of the implementation is dominated by the execution time of two processes—the input/output transfers to/from the BRAMs and the actual computation of the state equation (2). Usually one strives to maximize the ratio of calculation time to I/O time. However, with the successive execution of several PTCNN operations within a complex algorithm (PTCNN subroutine), the time spent on the calculation increases, while the transfer time may remain constant, and thus this ratio grows.

The transfer time for either reading or writing a single array of state or input values is for a given clock frequency f_{clk}

$$t_{\mathrm{IO}} = \frac{M \cdot N}{\chi} \frac{1}{f_{\mathrm{clk}}} \tag{10}$$

where χ denotes the I/O parallelization factor, i.e., the number of data words that can be read in parallel from the DRAM in one clock cycle.

The time to compute n_{it} iterations of (2) is

$$t_c = \frac{M \cdot N \cdot T_{\mathrm{calc}} \cdot (n_{\mathrm{it}} + 1)}{n_{\mathrm{pe}}} \frac{1}{f_{\mathrm{clk}}} \tag{11}$$

wherein the additional iteration $(n_{\mathrm{it}} + 1)$ is caused by the computation of the offset term. T_{calc} denotes the number of clock cycles that are required to calculate the offset or one state iteration for a single cell, where

$$T_{\mathrm{calc}} = 6 + 3(P - 2) \tag{12}$$

for the emulation of a PTCNN.

TABLE I
COMPARISON OF TWO CONFIGURATIONS OF THE TITUS PTCNN WITH A MIXED-SIGNAL ASIC IMPLEMENTATION

	LAIHO et al. [14]	**TITUS a)**	**TITUS b)**
Technology	250 nm	40 nm [1]	40 nm [1]
Network size	72×72	72×72	256×256
Number of PEs	144	72	256
Clock frequency	n.a.	100 MHz	100 MHz
Max. data resolution	8 bit	18 bit	18 bit
Polynomial order	1, 2, 3	≥ 2	≥ 2
Discretisation method	Heun	Euler	Euler
Feedforward coupling	no	yes	yes
Feedback coupling	yes	yes	yes
Iteration time $P = 2$	166.4 µs	8.6 µs / 86.4 µs [2]	30.7 µs / 1013.8 µs [2]
Iteration time $P = 3$	166.4 µs	13.0 µs 90.7 µs [2]	46.1 µs 1029.1 µs [2]

[1] Implemented on Xilinx Virtex-6 LX240T FPGA
[2] Including data I/O with $\chi = 2$

C. Comparison

For comparison, only the work of [14] could be considered since it appeared to be the only hardware implementation of a PTCNN architecture so far. The mixed-signal cellular processor array with polynomial couplings of an order up to $P = 3$ has been developed for the analysis of brain activity. In Table I, the chip is compared with two different configurations of the TITUS architecture: 1) one with the same network size of 72×72 cells and 2) one with the largest possible quadratic network at 18 bit on the given FPGA.

V. EXAMPLES

A. Simulation of the Burgers' Equation

Following the approach given in [5] and [6], the well-known BURGERS' equation

$$\frac{\partial \varphi(s,t)}{\partial t} = -\frac{1}{2} \frac{\partial \varphi(s,t)^2}{\partial s} + \frac{1}{R} \frac{\partial^2 \varphi(s,t)}{\partial s^2} \tag{13}$$

has been simulated using the emulated PTCNN. With the spatial discretization step Δs to evaluate the approximate solution of the PDE at discrete points $\varphi(i\,\Delta s, t) \equiv x_i(t)$, we obtain the CNN state equation

$$\dot{x}_i(t) = \frac{x_{i-1}(t) - 2x_i(t) + x_{i+1}(t)}{R(\Delta s)^2} - \frac{x_{i+1}^2(t) - x_{i-1}^2(t)}{4\Delta s}. \tag{14}$$

If we assume $y_{ij}(t) = x_{ij}(t)$ for $-1 \leq x_{ij}(t) \leq 1$, a comparison of the coefficients with (1) delivers the following coupling weights:

$$\mathbf{A}^{(1)} = \begin{array}{|c|c|c|} \hline \frac{1}{R(\Delta s)^2} & \frac{R(\Delta s)^2 - 2}{R(\Delta s)^2} & \frac{1}{R(\Delta s)^2} \\ \hline \end{array}$$

$$\mathbf{A}^{(2)} = \begin{array}{|c|c|c|} \hline -\frac{1}{4\Delta s} & 0 & \frac{1}{4\Delta s} \\ \hline \end{array}$$

$$\mathbf{B}^{(0)} = 0, \ \mathbf{B}^{(1)} = 0, \ \mathbf{z} = 0. \tag{15}$$

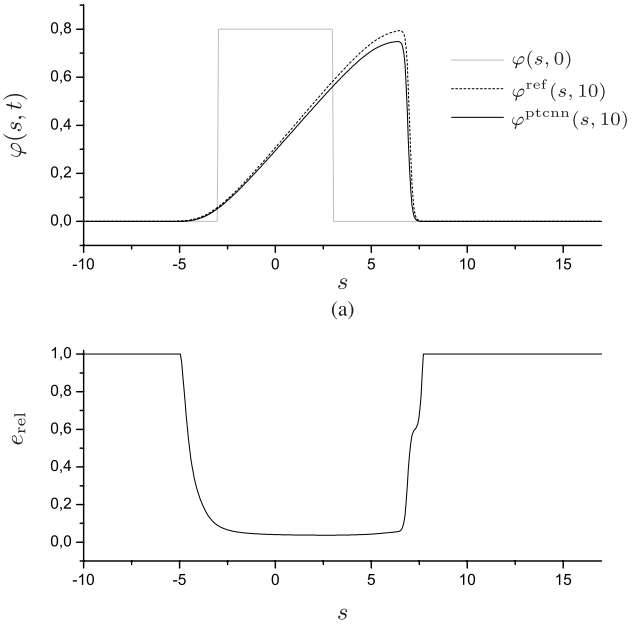

Fig. 5. Comparison of solutions of the discretized homogeneous BURGERS' equation at time $t = 10$. (a) Results obtained by the PTCNN with a data precisions of 18 bit; the simulation parameters are $R = 30$, $\Delta s = 0.06$, and $h = 0.01$. (b) Relative error of the PTCNN solution.

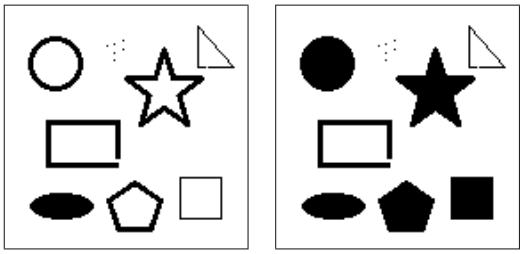

Fig. 6. Hole-filler template applied to a 128×128 input image. (a) Input image. (b) Reference output.

The computation has been performed in the spatial range $[-60, 60]$ on a network comprising 2000 cells, i.e., $\Delta s = 0.06$. As an example, the solution at time $t = 10$ was evaluated using a step size of $h = 0.01$, and is compared with a numerical 64-bit floating-point reference solution $\varphi^{\mathrm{ref}}(s,t)$ [Fig. 5(a)]. The relative error is defined by

$$e_{\mathrm{rel}} = \left| \frac{\varphi^{\mathrm{cnn}}(s,t) - \varphi^{\mathrm{ref}}(s,t)}{\varphi^{\mathrm{ref}}(s,t)} \right| \tag{16}$$

and shown in Fig. 5(b). The network output values are computed by 250 PEs at a clock of 100 MHz, resulting in a computation time of $t_c = 1.5$ ms.

The usage of an 18-bit data precision delivers an adequate approximation of the solution after the 1000 iteration steps. However, the relative error strongly increases when $\varphi(s,t) \to 0$, since very small values ($\varphi(s,t) < 2^{-17}$) are simply cutoff numerically and thus rounded to zero.

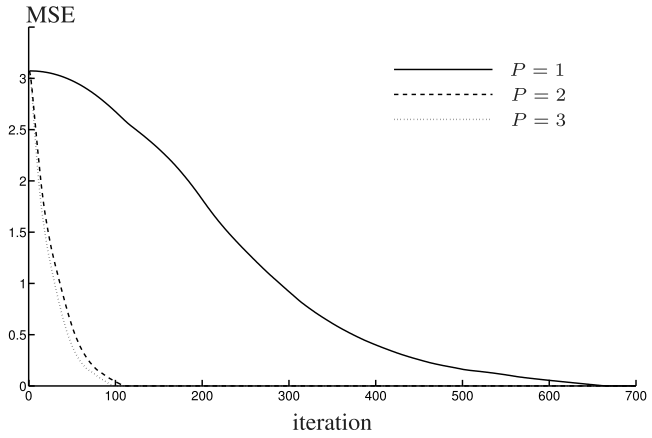

Fig. 7. Evolution of the MSE for the hole-filler template in the case of linear coupling weights ($P = 1$) and nonlinear coupling weights ($P = 2$ and $P = 3$).

TABLE II
RESOURCE UTILIZATION AND COMPUTATION TIME FOR THE HOLE-FILLER TEMPLATE WITH DIFFERENT POLYNOMIAL ORDERS

P	n_{pe}	n_{dsp}	T_{calc}	n_{it}	t_c
1	128	128	10	672	17.2 ms
2	128	384	6	119	1.8 ms
3	128	384	9	103	2.4 ms

B. Hole-Filler With PTCNN

Finally, we demonstrate the application of networks with linear and polynomial couplings to the popular hole-filler task on a 128×128 input image (Fig. 6). At first, we implemented a standard CNN ($P = 1$) of 128×128 cells with 128 PEs, processing the hole-filler template given in [19]. For comparison, we emulated a PTCNN with $P = 2$ and $P = 3$ of the same network size. The corresponding templates have been optimized by a genetic algorithm on a set of multiple training images [20].

A step size of $h = 0.1$ and a clock of $f_{\mathrm{clk}} = 100$ MHz were used throughout all measurements. Using the mean squared error (MSE)

$$\mathrm{MSE} = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} (y_{ij}(n) - \hat{y}_{ij})^2 \tag{17}$$

the network output y_{ij} at iteration n is evaluated in comparison with the reference output $\hat{y}_{ij}$, as shown in Fig. 6(b). In the linear case, at least 672 iterations are required to reach the steady state (MSE ≈ 0) corresponding to 17.2 ms. The number of iterations required to reach the steady state is significantly lower for polynomial couplings (119 for $P = 2$ and 103 for $P = 3$), resulting in a computation time of 1.8 and 2.4 ms, respectively. The evolution of the MSE is shown in Fig. 7.

It is evident that the usage of polynomial coupling weights hugely improves the performance of the given task. However, the step to $P = 3$ does not dramatically affect the required number of iterations and costs additional computational effort for each iteration (Table II). Thus, a polynomial order of two turns out to be most favorable for solving this task.

VI. CONCLUSION

General purpose architecture for a digital emulation of CNNs with polynomial couplings has been presented. Implemented on a

state-of-the-art FPGA, the system is capable of high-speed computation of PTCNN operations on large-scale networks. The proposed system is considered the first digital hardware implementation of a PTCNN so far. Applications for image processing and the simulation of PDEs have been discussed, some of which could not be realized in a CNN hardware before.

We are currently implementing an extension of the polynomial-weight architecture supporting on-chip optimizations of network parameters, and thus paving the way to a very efficient determination of problem-specific templates.

REFERENCES

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.

[2] L. Nicolosi, F. Abt, A. Blug, A. Heider, R. Tetzlaff, and H. Höfler, "A novel spatter detection algorithm based on typical cellular neural network operations for laser beam welding processes," *Meas. Sci. Technol.*, vol. 23, no. 1, p. 015401, 2012.

[3] F. Gollas, C. Niederhöfer, and R. Tetzlaff, "Toward an autonomous platform for spatio-temporal EEG-signal analysis based on cellular nonlinear networks," *Int. J. Circuit Theory Appl.*, vol. 36, nos. 5–6, pp. 623–639, Sep. 2008.

[4] P. Arena, L. Fortuna, M. Frasca, and L. Patané, "A CNN-based chip for robot locomotion control," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 9, pp. 1862–1871, Sep. 2005.

[5] T. Roska, L. O. Chua, D. Wolf, T. Kozek, R. Tetzlaff, and F. Puffer, "Simulating nonlinear waves and partial differential equations via CNN. I. Basic techniques," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 42, no. 10, pp. 807–815, Oct. 1995.

[6] F. Puffer, R. Tetzlaff, and D. Wolf, "Modeling nonlinear systems with cellular neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 6. May 1996, pp. 3513–3516.

[7] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 3, pp. 163–173, Mar. 1993.

[8] A. Rodriguez-Vazquez *et al.*, "ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 5, pp. 851–863, May 2004.

[9] J. J. Martínez, J. Garrigós, J. Toledo, and J. M. Ferrández, "An efficient and expandable hardware implementation of multilayer cellular neural networks," *Neurocomputing*, vol. 114, pp. 54–62, Aug. 2013.

[10] J. Müller, J. Müller, and R. Tetzlaff, "NEROvideo: A general-purpose CNN-UM video processing system," *J. Real-Time Image Process.*, pp. 1–12, Sep. 2014.

[11] Z. Nagy and P. Szolgay, "Configurable multilayer CNN-UM emulator on FPGA," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 6, pp. 774–778, Jun. 2003.

[12] N. Yildiz, E. Cesur, and V. Tavsanoglu, "Demonstration of the second generation real-time cellular neural network processor: RTCNNP-v2," in *Proc. 13th Int. Workshop Cellular Nanosc. Netw. Appl. (CNNA)*, Aug. 2012, pp. 1–2.

[13] T. Roska and L. O. Chua, "Cellular neural networks with nonlinear and delay-type template elements," in *Proc. IEEE Int. Workshop Cellular Neural Netw. Appl. (CNNA)*, Dec. 1990, pp. 12–25.

[14] M. Laiho, A. Paasio, A. Kananen, and K. A. I. Halonen, "A mixed-mode polynomial cellular array processor hardware realization," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 2, pp. 286–297, Feb. 2004.

[15] R. Braunschweig, J. Müller, J. Müller, and R. Tetzlaff, "NERO mastering 300k CNN cells," in *Proc. Eur. Conf. Circuit Theory Design (ECCTD)*, Sep. 2013, pp. 1–4.

[16] J. Müller, J. Müller, and R. Tetzlaff, "A new high-speed real-time video processing platform," in *Proc. 14th Int. Workshop Cellular Nanosc. Netw. Appl. (CNNA)*, Jul. 2014, pp. 1–2.

[17] F. Corinto, M. Gilli, and P. P. Civalleri, "On stability of full range and polynomial type CNNs," in *Proc. 7th IEEE Int. Workshop Cellular Neural Netw. Appl. (CNNA)*, Jul. 2002, pp. 33–40.

[18] J. Müller, R. Becker, J. Müller, and R. Tetzlaff, "CESAR: Emulating cellular networks on FPGA," in *Proc. 13th Int. Workshop Cellular Nanosc. Netw. Appl. (CNNA)*, Aug. 2012, pp. 1–5.

[19] K. Karacs *et al.*, *Software Library for Cellular Wave Computing Engines V.3.1*. Budapest, Hungary: Pázmány Péter Catholic Univ., 2010.

[20] B. Chandler, C. Rekeczky, Y. Nishio, and A. Ushida, "Adaptive simulated annealing in CNN template learning," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 82, no. 2, pp. 398–402, 1999.