SEE: a Concept for an FPGA based Emulation Engine for Spiking Neurons with Adaptive Weights

HEIK H. HELLMICH and HEINRICH KLAR
Faculty of Electrical Engineering and Computer Science
Technical University Berlin
Einsteinufer 17, Sekr. EN 4, D-10587 Berlin
GERMANY

Abstract: - We present an FPGA based architecture of a digital acceleration platform for the simulation of spiking neurons, called Spiking Neural Network Emulation Engine (SEE), that is capable to account for up to 2^{19} neurons with more than $3 \cdot 10^3$ weights each. This distributed memory architecture tackles the main bottle-neck of reduced memory bandwidth during the simulation of large networks of spiking neurons. With this approach an effective parallelisation of neuron modelling is achieved by providing multiple channels to the weight memory. Especially, the consideration of dynamical synapses where synaptic weights have to be adapted throughout the whole simulation time benefit from the proposed SEE architecture. The targeted programmable neuron model utilised by SEE is based on synaptic weights with several adaptation rules. It is evaluated that the currently implemented numerical integration method which is necessary for the neuron state computation can be accelerated by a factor of more than 100 compared to a software implementation running on a stand-alone PC.

Key-Words: - Spiking neural network, Simulation acceleration, FPGA based emulation, Numerical integration

1 Introduction

Spiking neural networks (SNNs) or pulse-coded neural networks (PCNNs) are examined for two reasons. Firstly to understand and reproduce the spike or pulse processing in the brain. Secondly to use the results of this research for technical systems that primarily perform vision tasks, e. g. different image features are separated by different phases of spikes of a neuron group representing these features. The simulation acceleration of large SNNs has to face five main problem classes [1]: calculation steps, communication resources, load balancing, storage capacity and memory bandwidth. The execution time of calculation steps can be reduced by introducing parallelism, pipelining and event lists that reduce the number of neurons and synapses involved in spike generation. Communication resources and load balancing arise to a problem during parallel processing and can lead to longer simulation times because of additional communication overhead between processing elements (PEs) and unbalanced loads among the PEs [2]. The storage capacity determines the size of the network that can be simulated on a digital acceleration platform without involving the time consuming communication channel to a host computer for transferring network parameters. However, the main reason for poor simulation performance of large SNNs is the reduced memory bandwidth [3] because the most limiting sequential part during the simulation is the data transfer between weight memory and PEs [4]. For this reason, a digital acceleration platform is needed which on the one hand tackles this main bottle-neck problem by providing a distributed memory architecture and on the other hand offers programmability for the control software and for the simulation accelerating hardware implementation by field-programmable gate arrays (FPGAs). The basic architecture of the platform which incorporates these two aspects is described in the following and is called Spiking Neural Network Emulation Engine (SEE).

2 SEE architecture

Most commercially available FPGA based prototyping platforms or the academic rapid-prototyping system RAPTOR2000 [5] do not offer the necessary IO bandwidth to external memory devices and storage capacity for an efficient and fast simulation of large SNNs. Any digital accelerator system with (IO bounded) off-chip memory is almost useless compared to any DSP or PC system which provides comparable or better performance at a smaller price, shorter time-to-market and higher availability [4]. A promising approach is the BEE rapid-prototyping system [6], which is equipped with up to 2400 IO pins. The major drawback of this platform is that the communication effort between a central event list and a central network topology unit to the 20 FPGAs that the BEE incorporates is very high. However, today's FPGA technology offers single FPGA devices that present more than 1000 usable IO pins [7] and therefore allows a more compact digital acceleration platform in order to optimise the communication resources.

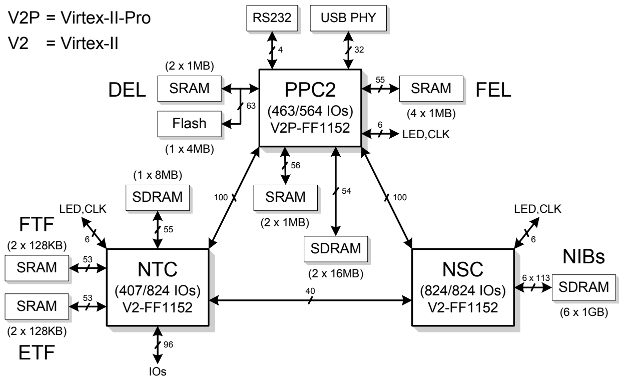

Fig. 1: Overview of the SEE platform.

The SEE, based on three FPGAs (see Fig. 1), is intended as hardware/software (HW/SW) experimental platform for the simulation of large SNNs that accomplishes compact usage of FPGA resources and multiple channels to the weight memory. Latter is especially needed for spiking neuron models that utilise dynamical synapses because weight values have to be adapted throughout the whole simulation time. This approach is therefore more sophisticated than a recently introduced approach of a digital acceleration system considering synaptic plasticity [8] where a central system controller manages the data transfer to the external weight memory via one single channel. The three FPGAs within SEE in Fig. 1 undertake the tasks of simulation control (PPC2), network topology computation (NTC) and neuron state computation (NSC).

2.1 Simulation Control

The central FPGA is a Virtex-II-Pro device [7] that incorporates two software programmable PowerPC cores and is responsible for the simulation control. This task includes the initialisation of the network via an external serial interface, the monitoring and visualisation of network parameters that can be deposited in the connected SDRAM and the management of the event lists stored in the SRAM devices. For the simulation two event lists are necessary: the dynamic event list (DEL) and the fire event list (FEL). The DEL includes all excited neurons that receive a spike or an external input stimulus. The FEL stores all firing neurons that are in a sending state and the corresponding time values when the neuron enters the receiving state again. The memory size of the DEL of 2 MB determines the maximum number of neurons that can be treated during a simulation run. Neurons as well as time values are coded as 4 byte data. Considering the worst case scenario that all neurons in the network can be excited at the same time, the maximum number of simulatable neurons within SEE leads to 512 K (2^{19}). The FEL has to provide a storage capacity which is double in size compared to the DEL because of the additional time value for each neuron.

2.2 Network Topology Computation

The FPGA in charge of the NTC performs two tasks: the definition of the presynaptic activity of each excited neuron by generating a topology vector (receptive field) and the determination of postsynaptic neurons in case an excited neuron has entered the sending state (projective field). For these tasks two tag memories are required that are realised by dedicated SRAMs: the fire-tag-field (FTF) and the excitation-tag-field (ETF). The FTF marks each firing neuron with a tag (1 bit) and is needed for the generation of the topology vector. The ETF tags all firing neurons and all neurons presented in the DEL. This tag-field is required for determining the postsynaptic neurons that have to be added to the DEL because they have become excited by a spiking neuron and are not present in the DEL yet. Currently, three regular connection schemes are implemented: feedforward point-to-point, 4-nearest-neighbour and 8-nearest-neigbour connections.

2.3 Neuron State Computation

The FPGA carrying out the NSC has dedicated communication channels to the event lists and the NTC module providing the topology vectors that are necessary for updating the membrane potentials of each excited neuron. The simulation algorithm is realised with an event-driven approach [9] and does not utilise fixed time steps. A simulation event is the time where a neuron changes from receiving to sending state or vice versa. Reading the FEL the next time can be determined when a neuron stops to fire. Within this time interval a numerical integration is performed to determine if one of the excited neurons starts to fire (*next-spike phase*). The smaller time represents the next simulation event and the whole network has to be updated to this time (*update phase*). The NSC unit is connected to six SDRAM modules that provide a high storage capacity (1 GB each) and an 8 byte wide data bus. Thus, six dedicated channels to the weight memory are available in order to achieve effective parallelisation of the numerical integration process. The parameters for each neuron are stored in a neuron information block (NIB). Each NIB consists of a header that is coded as 8 byte data, and the membrane potential and presynaptic weight values that are coded as 4 byte data values. The header contains information about number of presynaptic weights and the input stimuli value, if present. The NIB of each neuron is accessed by indirect addressing through a pointer table which is stored at the beginning of each SDRAM module. Hence, each neuron occupies a storage of 16 bytes incorporating the header, the membrane potential and a 4 byte pointer needed for indirect addressing. This causes a memory requirement of 8 MB (2^{23}) within the total weight memory of 6 GB ($6 \cdot 2^{30}$) for the maximum size of 512 K (2^{19}) neurons. This leads to $(6 \cdot 2^{30} - 2^{23})/2^{19}/4 = 3068$ weights per neuron when the maximum number of neurons is considered.

3 Spiking Neuron Model with Adaptive Weights

The targeted programmable spiking neuron model is a non-leaky integrate-and-fire neuron (IFN) model presented in [10]. The function of the membrane potential a_K is illustrated in Eq. 1,

$$a_K(t) = a_K(t_0) + \int_{t_0}^{t} \left[i_K + \sum_{i \in N_s} W_{KL_i}(t) \right] dt \qquad (1)$$

where t_0 is the time of the last simulation event, i_K represents the external input stimulus, N_S is the number of neurons sending a spike, and W_{KL} represents the corresponding presynaptic weight value. The external input stimulus is a grey pixel value that is normalised to values between 0 and 1. The membrane potential a_K stays constant if the neuron does not receive any external stimulus or any spikes. According to Eq. 1, neurons receiving only a constant input current will always fire regularly. This is in contrast to a leaky IFN model where the product of input current and leaky-resistor have to be greater than the firing threshold so that the neuron fires periodically [11]. The dynamics of the spiking neuron model evolves from the time course of the synaptic weights and the emission of spikes by the presynaptic neurons. The utilised adaptation rule during network simulations for the synaptic weights is,

$$W_{KL}' = -\gamma \cdot W_{KL} + \begin{cases} \mu \cdot \left(a_K - \frac{\theta}{2} \right) & X_K = 0 \wedge X_L = 1 \\ 0 & else \end{cases} \qquad (2)$$

where γ is the decay constant, μ is the gain factor, θ is the constant firing threshold, and X_K and X_L represent the status of the post- and presynaptic neuron, respectively. If the postsynaptic neuron is in a receiving state ($X_K = 0$) and the presynaptic neuron in a sending state ($X_L = 1$) the exponential decay of the weight will be affected. Depending on the membrane potential of the postsynaptic neuron the weight acquires a potentiative ($a_K > \theta/2$) or a depressive effect ($a_K < \theta/2$) for the duration of the pulse width t_d. This kind of weight adaptation achieves a synchronised firing for neurons stimulated by similar external inputs and connected laterally in a 4-nearest-neighbour connection scheme [10] even if the membrane potentials of all neurons are initialised with random values at the start of the simulation.

3.1 Resolution Analyses

For the mapping of the software implementation to the SEE, resolution analyses were performed in order to investigate the network performance according to the transformation from a floating-point representation to a fix-point representation of model parameters (a_K, i_K, W_{KL}, μ, γ, θ, ...) and time parameters. Only arithmetic operations with fix-point data result in fast and optimised FPGA implementations. Fix-point numbers are represented by an integer and a fractional part: (i.f). Since the firing threshold is set to 1, it is sufficient to represent the integer part of model parameters by 2 bits. For the resolution quality of a fix-point representation two quality measures have to be explored: the spiking frequency and the standard deviation of spike time distances between connected neurons. Simulations have revealed that a fractional part of 14 bits for model parameters is needed to achieve the same spiking frequency as a floating-point implementation. However, to cope with equal spike time distances between connected neurons only a fractional part of at least 18 bits for model parameters were regarded to be sufficient. This leads to the current implementation that model parameters are represented by fix-point number of 20 bits (2.18) and time parameters by 32 bits (14.18).

3.2 Numerical Integration Method

In order to provide a high degree of flexibility for necessary changes to the neuron model we decided to pursue a numerical and not an analytical implementation to determine the next spike time of neurons, according to Eq. 1 and Eq. 2. The primary criterion for the choice of a numerical method is the computational efficiency. An additional complication arises that each neuron's firing influences other connected neurons. An integration with fixed time steps or an first-order-accurate numerical method, e. g. Euler, can artifactually synchronise neurons at a sacrifice of network dynamics [12]. The software implementation has revealed the computational efficiency concerning execution time of the Bulirsch-Stoer integration method [13] compared to a 4th-order Runge-Kutta integration method. The Bulirsch-Stoer integration method incorporates two arithmetic operations [14]: a modified-midpoint integration (MMID) and a polynomial extrapolation (PZEXTR).

The MMID calculates from the function value *y(x)* at the point *x* the new function value *y(x+H)* at the point *x+H* by a sequence of *nstep* substeps each of size *h*. Therefore, *H* represents the integration interval and *nstep* the number of integration steps within that interval. Besides the first integration step, the gradient of the current intermediate function value k_m and the last intermediate function value k_{m-1} is needed to calculate the next intermediate function value k_{m+1} or the final new function value $y_{nstep} \approx y(x+H)$ as shown in Eq. 3 and Eq. 4 [14]:

$$k_{m+1} = k_{m-1} + 2 \cdot h \cdot f(x + m \cdot h, k_m) \tag{3}$$

$$y_{nstep} = \frac{1}{2} \cdot [k_{nstep} + k_{nstep-1} + h \cdot f(x + H, k_{nstep})] \tag{4}$$

Finally, the PZEXTR performs an extrapolation of the new function value y_i by calculating the two variables $Q_{i,k}$ and $D_{i,k}$ [13],

$$\begin{aligned} y_i &= Q_{i,0} = D_{i,0}, \qquad i = 0, \ldots, 7 \\ Q_{i,k} &= xq_i \cdot (D_{i,k-1} - Q_{i-1,k-1}) \\ D_{i,k} &= xd_i \cdot (D_{i,k-1} - Q_{i-1,k-1}) \end{aligned} \tag{5}$$

where the variable *i* is the index for different new function values based on *y(x)* and determined by the MMID. The extrapolated function value is then defined by a summation stated in Eq. 6:

$$y_{ext,i} = \sum_{k=0}^{i} Q_{i,k} \tag{6}$$

4 SEE performance evaluation

The ultimate goal of SEE is to represent a flexible digital simulation platform for large SNNs that provides a speed-up factor ($F_{SPEED\text{-}UP}$) of at least 10 compared to state-of-the-art stand-alone PCs. This allows a more intense observation of neural network dynamics by multiple simulation runs. In the following, the performance evaluation of SEE compared to a software implementation running on a PC (2.4 GHz Pentium-4, 1 GB RAM) is described in order to demonstrate the superiority of the SEE architecture and to justify the further HW/SW implementation. Software simulations were performed with single layer network architectures that were stimulated by a grey-value image that differed in pixel size (32 x 32, 48 x 48 and 64 x 64). The neurons were laterally connected by a 4-nearest-neighbour (n = 4) and a 8-nearest-neighbour (n = 8) connection scheme. Each network was simulated for 1000 ms and the values used for the model parameters in Eq. 2 are summarised in Table 1:

γ	μ	θ	t_d	$W_{KL}(t=0)$	$a_K(t=0)$
0.1	0.3	1	1 ms	0.12	rand[0,...,1]

Table 1: Neuron model parameter values.

4.1 Software simulation profiling

Profiling during the software simulations revealed that the simulation execution time of sparsely connected neurons is dominated by the numerical integration process. In Fig. 2 the number of events N_{EVENT} for the 4-nearest-neighbour connection scheme is shown that increases with increasing network size. For networks with 8-nearest-neighbour connections N_{EVENT} is less than 10 % higher. The number of performed numerical integrations per neuron N_{BSSTEP} is slightly above N_{EVENT}. At each simulation event a numerical integration will be performed by each neuron in the *update phase* but only a numerical integration will be conducted by excited neurons in the *next-spike phase* ($N_{EVENT} \leq N_{BSSTEP} \leq 2 \cdot N_{EVENT}$).

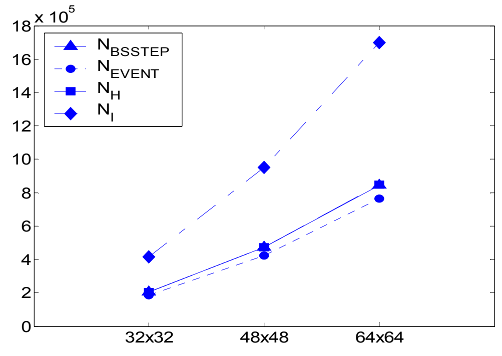

Fig. 2: Numerical integration steps (n = 4).

Further, it was noticed that the number of integration interval changes per neuron N_H is almost identical to N_{BSSTEP} which confirms that a chosen integration interval *H* had not been readjusted. The number of substep-divisions per executed numerical integration N_I is nearly twice as N_{BSSTEP} which indicates that two subdivisions into substeps *h* within *H* were necessary. According to Fig. 2, the average number of integration interval changes per neuron H_{AVG} and the average number of necessary substep-divisons within an integration interval I_{AVG} can be specified as:

$$H_{AVG} = N_H / N_{BSSTEP} \cong 1 \tag{7}$$

$$I_{AVG} = N_I / N_{BSSTEP} \cong 2 \tag{8}$$

In Fig. 3 the number of neurons (N_{NEURON}) and weights (N_{WEIGHT}) are visualised by the left axis while the required software simulation time T_{SW} for the numerical integration process (about 70 % of the total software simulation time) is shown by the right axis.

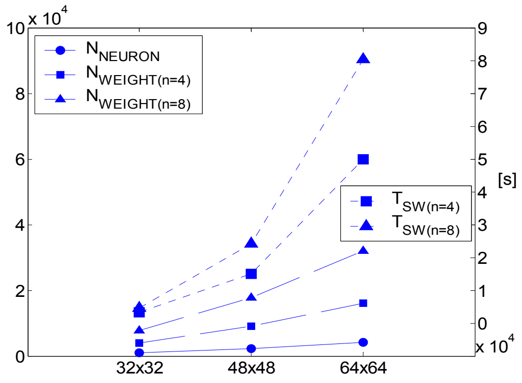

Fig. 3: Software simulation time.

The software simulation time does not increase linearly with increasing number of neurons and weights and takes more than 22 h (80331 s) for a 64 x 64 network with an 8-nearest-neighbour connection scheme.

4.2 SEE simulation time

The simulation time of SEE depends on several parameters: the total number of executed numerical integrations during the simulation ($N_{BSSTEP} \cdot N_{NEURON}$), the number of parallel usable memory channels N_{IO}, the operating frequency of the acceleration platform T_{CLK}^{-1}, and the duration of the numerical integration for each neuron T_{NEURON} that depends on the number of presynaptic weights n and is measured in clock cycles. The simulation time T_{SEE} can therefore be estimated as follows:

$$T_{SEE} = N_{BSSTEP} \cdot \frac{N_{NEURON}}{N_{IO}} \cdot T_{CLK} \cdot T_{NEURON}(n) \qquad (9)$$

The duration of the numerical integration for each neuron can be stated as in Eq. 10,

$$T_{NEURON} = H_{AVG} \cdot \sum_{i=0}^{I_{AVG}-1} [T_{MMID}(n,i) + T_{PZEXTR}(n,i)] + t_{SDRAM} \qquad (10)$$

where t_{SDRAM} represents the clock cycle latency for accessing the NIB in the SDRAM, and T_{MMID} and T_{PZEXTR} are the required clock cycles necessary for the execution of MMID and PZEXTR, respectively. The duration of the HW module within the NSC-FPGA (see Fig. 1) that realises the MMID operation is given by,

$$\begin{aligned} T_{MMID} &= t_{MMID} + ceil(n/2) \cdot (nstep+1) \\ &= t_{MMID} + ceil(n/2) \cdot [2 \cdot (i+1) + 1] \end{aligned} \qquad (11)$$

where t_{MMID} represents the clock cycle latency before the first data appears to be valid at the output of the pipelined HW module. The derivation of the synaptic weights (see Eq. 2) required by the MMID incorporates two parallel multiplications (γ-term and μ-term) that require 4 clock cycles each. The total number of clock cycles for the derivation leads to 6 which sums up to 12 clock cycles for t_{MMID}. The term *n/2* in Eq. 11 accounts for the fact that according to the 8 byte wide data bus of the SDRAM modules two synaptic weights are read at the same time.
The duration of the HW module that implements the PZEXTR operation is given by,

$$\begin{aligned} T_{PZEXTR} &= t_{PZEXTR} + ceil(n/2) \\ &\quad + i \cdot [ceil(n/(2 \cdot t_{PZEXTR})) \cdot t_{PZEXTR}] \end{aligned} \qquad (12)$$

where t_{PZEXTR} is the latency of 4 clock cycles resulting from the pipelined multiplication (see Eq. 5). Considering Eq. 7, Eq. 8, the worst case SDRAM latency that can vary between 4 and 10 clock cycles depending on access changes of pages or banks ($t_{SDRAM} = 10$), and the time latencies of the current pipelined HW implementations ($t_{MMID} = 12$, $t_{PZEXTR} = 4$) T_{NEURON} can be determined by Eq. 10 for the two different simulated connection schemes:

$$T_{NEURON}(4) = 66 \qquad (13)$$

$$T_{NEURON}(8) = 86 \qquad (14)$$

The intention of the pipelined HW implementation for arithmetic operations is to ensure an operating frequency of 100 MHz ($T_{CLK} = 10 \cdot 10^{-9}$ s). When sparsely connected network architectures are simulated ($n = 4$, $n = 8$) the presynaptic weight values and the corresponding integration variables (k_m, k_{m-1}, $Q_{i-1,k}$, y_i and $y_{ext,i}$) can be stored in the internal RAM blocks provided by the FPGA device. This leads to the fact that all six memory channels of the NSC-FPGA can be used in parallel ($N_{IO} = 6$). Applying Eq. 9, the simulation time of SEE for the numerical integration can be evaluated and is listed in Table 2 for different network architectures and sizes.

n	N_{NEURON}	N_{BSSTEP}	T_{SW}	T_{SEE}	$F_{SPEED-UP}$
4	32 x 32	204637	3251 s	24 s	135
	48 x 48	473434	15053 s	120 s	125
	64 x 64	846168	49946 s	382 s	130
8	32 x 32	213514	4591 s	32 s	143
	48 x 48	550944	24252 s	182 s	133
	64 x 64	980053	80331 s	576 s	139

Table 2: Estimated acceleration by SEE.

It can be seen that the numerical integration process can be accelerated by a factor of more than 100 compared to a software implementation running on a stand-alone PC.

5 Conclusion

We have presented a promising architecture for a digital acceleration platform, called SEE, for the simulation of large SNNs. The performance evaluation compared to a software implementation has revealed that for the numerical integration process an acceleration by two orders of magnitude is obtainable. The simulation and design environment of SEE allows the verification of the functionality of implemented HW and SW modules, but the simulation under real-time conditions of such a highly parallel operating digital system is not feasible. This fact and the promising SEE architecture encourages the essential next development step of designing the printed circuit board (PCB) sketched in Fig. 1 in order to confirm the SEE performance.

Further, densely connected network architectures, e. g. full-connected, are currently under evaluation. This leads to the scenario that the integration variables have to be sourced out to the SDRAM modules in order to handle the increasing number of synaptic weights. In this case, three memory channels are used in parallel for each numerical integration process ($N_{IO} = 2$) in order to accelerate the MMID where always two function values are read (k_m and k_{m-1}) and one function value is stored (k_{m+1} or y_{nstep}).

References:

[1] M. Schäfer, T. Schönauer, C. Wolff, G. Hartmann, H. Klar, and U. Rückert, Simulation of Spiking Neural Networks: Architectures and Implementations, *Neurocomputing*, Vol. 48, 2002, pp. 647-679.

[2] K. Mohraz, U. Schott, and M. Pauly, Parallel Simulation of Pulse Coded Neural Networks, *Proceedings of the 15th World Congress on Scientific Computation, Modelling and Applied Mathematics (IMACS)*, Vol. 6, 1997, pp. 523-528.

[3] A. Jahnke, U. Roth, and T. Schönauer, Digital Simulation of Spiking Neural Networks, *Pulsed Neural Networks* (Editors: W. Maass and C. M. Bishop), MIT Press, ISBN 0-262-13350-4, 1998.

[4] L. M. Reyneri, Implementation Issues of Neuro-Fuzzy Hardware: Going Toward HW/SW Codesign, *IEEE Transaction on Neural Networks*, Vol. 14, No. 1, 2003, pp. 176-194.

[5] M. Porrmann, U. Witkowski, H. Kalte und U. Rückert, Implementation of Artificial Neural Networks on a Reconfigurable Hardware Accelerator, *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (PDP)*, 2002, pp. 243-250.

[6] C. Chang, K. Kuusilinna, B. Richards, and R. W. Broderson, Implementation of BEE: a Real-time Large-scale Hardware Emulation Engine, *International Symposium on Field Programmable Gate Arrays (FPGA)*, 2003, pp. 91-99.

[7] Xilinx, Virtex-II/-II-Pro Datasheets, http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp.

[8] N. Mehrtash, D. Jung, H. H. Hellmich, T. Schönauer, V. T. Lu und H. Klar, Synaptic Plasticity in Spiking Neural Networks (SP^{2}INN): A System Approach, *IEEE Transactions on Neural Networks*, Vol. 14, No. 5, 2003, pp. 980-992.

[9] M. Mattia, and P. Del Giudice, Efficient Event-Driven Simulation of Large Networks of Spiking Neurons and Dynamical Synapses, *Neural Computation*, Vol. 12, 2000, pp. 2305-2329.

[10] A. Heittmann, U. Ramacher, D. Matolin, J. Schreiter, and R. Schüffny, An Analog VLSI Pulsed Neural Network for Image Segmentation using Adaptive Connection Weights, *International Conference on Artificial Neural Networks (ICANN)*, 2002, pp. 1293-1298.

[11] W. Gerstner, and W. M. Kistler, *Spiking Neuron Models - Single Neurons, Populations, Plasticity*, Cambridge University Press, ISBN 0-521-81384-0, 2002.

[12] M. V. Mascagni, and A. S. Sherman, Numerical Methods for Neuronal Modelling, *Methods in Neuronal Modelling: From Ions to Networks* (Editors: C. Koch and I. Segev), MIT Press, 2nd Edition, ISBN 0-262-11231-0, 1998.

[13] J. Stoer, and R. Bulirsch, *Numerische Mathematik II,* Springer Verlag, 4th Edition, ISBN 3-540-67644-9, 2000.

[14] Numerical Recipes, Books On-Line, Chapter 16, http://www.nr.com/nronline_switcher.html.