Advances in Radio Science, 3, 287–291, 2005
SRef-ID: 1684-9973/ars/2005-3-287

Advances in Radio Science

Digitalsimulator für pulscodierte neuronale Netze

H. H. Hellmich and H. Klar

Technische Universität Berlin, Fakultät für Elektrotechnik und Informatik, Institut für Technische Informatik und Mikroelektronik, Einsteinufer 17, Sekr. EN 4, D-10587 Berlin, Germany

Zusammenfassung. Die Simulation von grossen pulscodierten neuronalen Netzen (PCNNs) für die Evaluierung einer biologisch motivierten Bildverarbeitung ist auf Einprozessor-Systemen (PCs oder Workstations) immer noch sehr zeitineffizient. Den Flaschenhals während der Simulation bildet der sequentielle Zugriff auf den Gewichtsspeicher zur Berechnung der Neuronenzustände. Es wird ein Digitalsimulator basierend auf feld-programmierbaren Gate-Arrays (FPGAs) vorgestellt, der dieses Flaschenhals-Problem durch eine verteilte Speicherarchitektur und eine erhöhte Speicherbandbreite angeht und zusätzlich eine getrennte Berechnung von Neuronenzuständen und Netzwerktopologie vorsieht. Somit ist es möglich, den Parallelitätsgrad des Simulationsalgorithmus zu erhöhen. Die momentane Implementierung mit einer Taktfrequenz von 50 MHz verspricht einen Beschleunigungsfaktor von etwa 30 für eine spärliche Vernetzungsstruktur (Vierer- und Achternachbarschaft) im Vergleich zu einem PC mit einer 2,4 GHz CPU und 1 GB RAM Arbeitsspeicher.

1 Einleitung

Die Simulation von PCNNs wird primär aus 2 Gründen durchgeführt. Zum einen möchte man die Pulsverarbeitung von Neuronen im menschlichen Gehirn reproduzieren und verstehen, und zum anderen möchte man die dadurch erlangten Erkenntnisse verwenden, ausgefeilte technische Systeme für unterschiedliche Anwendungsgebiete, insbesondere in der Bildverarbeitung, zu realisieren. Bei der Simulationsbeschleunigung von PCNNs gibt es 5 zu beachtende Flaschenhals-Probleme (Schäfer et al., 2002): Rechenoperationen, Kommunikation, Lastverteilung, Speicherkapazität und Speicherbandbreite. Der Hauptgrund einer ineffizienten Simulationsperformanz bei grossen PCNNs ist die reduzierte Speicherbandbreite (Jahnke et al., 1998), da der Datentransfer zwischen dem Gewichtsspeicher und den Prozessor-Elementen (PEs) der grösste sequentielle Teil bei der Simulation ist (Reyneri, 2003). Aus diesem Grund ist ein Digitalsimulator für die Simulationsbeschleunigung notwendig, der dieses grösste Flaschenhals-Problem anhand einer verteilten Speicherarchitektur angeht. Desweiteren stellt der Digitalsimulator eine ausreichende Flexibilität durch programmierbare Software (SW) und rekonfigurierbare Beschleunigungs-Hardware (HW) anhand von FPGAs bereit.

Correspondence to: H. H. Hellmich
(hellmich@mikro.ee.tu-berlin.de)

2 Neuronenmodell mit adaptiven Synapsen

Der Digitalsimulator ist primär für das Neuronenmodell von Heitmann und Ramacher (2003) ausgelegt, bei dem es sich um ein Integrate-and-Fire Modell ohne Leckwiderstand handelt. Die Gleichung für das Membranpotental a_K ist definiert als,

$$a_K(t) = a_K(t_0) + \int_{t_0}^{t} \left[i_K + \sum_{i \in N_S} W_{KLi}(t) \right] \mathrm{d}t \,, \qquad (1)$$

wobei t_0 der Zeitpunkt des letzten Simulationsereignisses ist, i_K repräsentiert den externen Eingangsreiz (Grauwert-Pixel des Eingangsbildes, der auf Werte zwischen 0 und 1 normiert ist), N_S ist die Anzahl der Puls-sendenden presynaptischen Neuronen und W_{KLi} stellt den jeweiligen Gewichtswert der Synapse dar. Im Gegensatz zu einem Integrate-and-Fire Neuron mit Leckwiderstand, das nur periodisch feuern kann, wenn das Produkt von konstantem Eingangsstrom und Leckwiderstand grösser als der Schwellwert ist (Gerstner und Kistler, 2002), feuert ein Neuron nach Gl. 1 stets periodisch, wenn ein konstanter Eingangsstrom anliegt. Die neuronale Dynamik des Neuronenmodells basiert auf der Adaption der Synapsengewichte W_{KL} und der Pulsaktivität der presynaptischen Neuronen. Die verschiedenen Adaptionsregeln beeinflussen die zeitliche Ableitung der Synapsengewichte. Die in den Simulationen verwendete Adaptionsregel für die Synapsen lautet wie folgt,

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

wobei γ die Abklingkonstante, μ den Verstärkungsfaktor und θ den neuronen-spezifischen konstanten Schwellwert darstellen. Die Variablen X_K und X_L repräsentieren jeweils den Status des post- bzw. presynaptischen Neurons. Wenn sich das postsynaptische Neuron im Empfangszustand

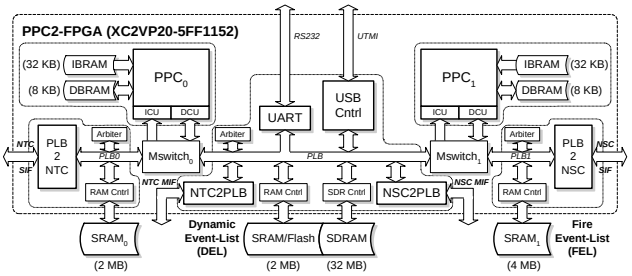

Abbildung 1. FPGA für die Simulationskontrolle.

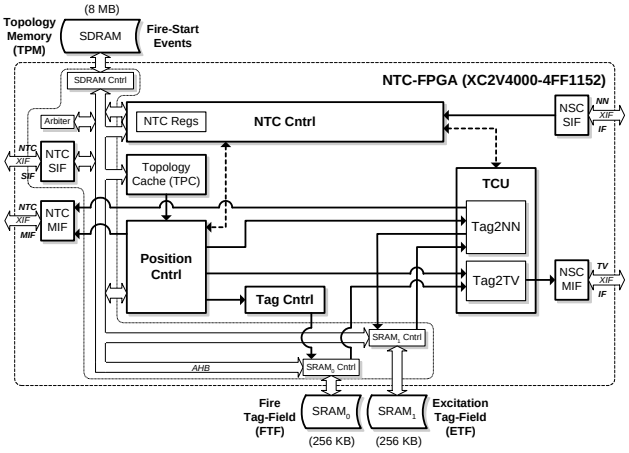

Abbildung 2. FPGA für die Netzwerktopologieberechnung.

(X_K=0) und das presynaptische Neuron sich im Sendezustand (X_L=1) befindet, wird das exponentielle Abklingverhalten der Synapse beeinflusst. Abhängig vom Membranpotential des postsynaptischen Neurons findet ein verstärkender Effekt ($a_K > \theta/2$) oder ein abschwächender Effekt ($a_K < \theta/2$) für die Pulsdauer t_d statt. Durch diese neuronale Dynamik werden Neuronen, die von einem ähnlichen externen Eingangsreiz stimuliert werden, zum synchronen Pulsen animiert und können zur Flecken-Detektion einer visuellen Szene eingesetzt werden (Heitmann und Ramacher, 2003).

3 Architektur des Digitalsimulators

Die Notwendigkeit für eine neuartige Architektur eines Digitalsimulators beruht auf der Tatsache, dass existierende akademische und kommerziell erhältliche Prototypen-Plattformen für die Simulation von grossen PCNNs nicht geeignet sind. Entweder stellen existierende Prototypen-Plattformen keine ausreichende Speicherbandbreite und Speicherkapazität zur Verfügung (sog. Rapid-Prototyping-Plattformen), oder aber eine ausreichende Speicherbandbreite wird nur in Kombination mit einer Überkapazität an FPGA Ressourcen bereitgestellt (sog. Emulations-Plattformen), die nicht effizient genutzt werden können (Krupnova und Saucier, 2000). Aus diesem Grund stehen bei der Architektur des Digitalsimulators 3 Eigenschaften im Vordergrund: 1. eine kompakte und effiziente Nutzung von FPGA Ressourcen, 2. ausreichende Speicherbandbreite für eine effiziente Parallelisierung des Simulationsalgorithmus, und 3. Autarkie durch ausreichende Speicherkapazität, um den zeitintensiven Datentransfer während der Simulation mit einem Host-Rechner zu vermeiden. Der Digitalsimulator basiert auf 3 FPGAs, die jeweils eigenständige Simulationsaufgaben übernehmen: Simulationskontrolle, Netzwerktopologie- und Neuronenzustandsberechnung (Hellmich und Klar, 2004).

3.1 Simulationskontrolle

Die Simulationskontrolle wird vom PPC2-FPGA in Abb. 1 anhand von 2 eingebetteten PowerPCs durchgeführt, die bei Virtex-II-Pro FPGAs bereitgestellt werden (Xilinx, 2004). Die Simulationskontrolle besteht aus 3 Aufgaben: 1. Konfiguration des Netzwerks vor dem Simulationsstart über externe serielle Schnittstellen, 2. Überwachung von Netzwerkparametern, und 3. Verwaltung von Ereignislisten. Während der Simulation müssen 2 Ereignislisten verwaltet werden: die dynamische Ereignisliste (DEL) und die Feuer-Ereignisliste (FEL). Die DEL beinhaltet alle erregten Neuronen, die zum betrachteten Simulationszeitpunkt einen Puls oder einen externen Reiz empfangen. Die FEL speichert alle Neuronen, die einen Puls senden und sich somit im Sendezustand befinden, und den dazugehörigen Zeitpunkt, an dem das Neuron wieder in den Empfangszustand gelangt. Die maximale Anzahl an Neuronen, die während einer Simulation berücksichtigt werden können, wird von der Speicherkapazität der DEL (2 MB) bestimmt und beträgt $2^{21}/4 = 2^{19}$, da ein Neuron mit 4 Bytes codiert wird. Aufgrund des zusätzlich abzuspeichernden Zeitwertes muss die FEL im Vergleich zur DEL eine zweifache Speicherkapazität bereitstellen.

3.2 Netzwerktopologieberechnung

Das NTC-FPGA, welches für die Netzwerktopologieberechnung zuständig ist (siehe Abb. 2), operiert in 2 Phasen: Topologie-Vektor-Phase und Topologie-Aktualisierungs-Phase. Für die Berechnungen sind 2 gleich grosse Markierungsspeicher (Tag-Felder) notwendig: das Feuer-Tag-Feld (FTF), in dem alle feuernden Neuronen markiert sind, und das Erregungs-Tag-Feld (ETF), das alle erregten und feuernden Neuronen markiert.

In der Topologie-Vektor-Phase wird durch Lesen des FTFs die presynaptische Aktivität aller Neuronen bestimmt, die sich in der DEL befinden. Dazu wird ein Bit-Vektor als Topologie-Vektor an die Neuronenzustandsberechnungseinheit gesendet. In der Topologie-Aktualisierungs-Phase wird je nach aufgetretenen Puls-Stop- oder Puls-Start-Ereignis das FTF aktualisiert. Bei Puls-Start-Ereignissen werden durch Lesen des ETFs die postsynaptischen Neuronen bestimmt, die durch den startenden Puls erregt werden und sich noch nicht in der DEL befinden.

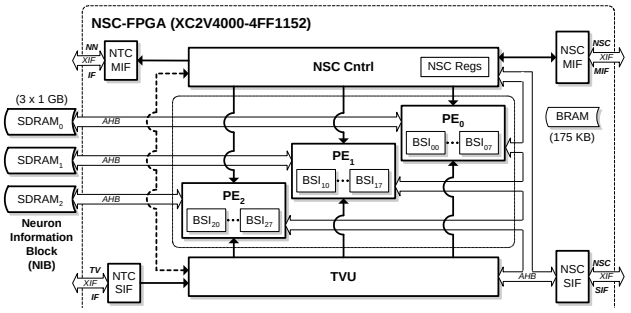

Abbildung 3. FPGA für die Neuronenzustandsberechnung.

3.3 Neuronenzustandsberechnung

Die Neuronenzustandsberechnung wird vom NSC-FPGA durchgeführt (siehe Abb. 3). Eine Topologie-Vektor-Einheit (TVU) verteilt dabei die eingehenden Topologie-Vektoren an die 3 vorhandenen PEs. Jede PE beinhaltet mehrere numerische Integratoren (BSIs) und hat einen eigenen Datenkanal zu einem SDRAM Modul, wo Neuronen-Informations-Blöcke (NIBs) abgelegt sind. Im Gegensatz zu anderen Beschleunigungs-Plattformen (Mehrtash et al., 2003) ist somit ein effizienterer Zugriff auf den Gewichtsspeicher möglich, da 3 Datenkanäle parallel genutzt werden können. Ein NIB beinhaltet neben neuronen-spezifischen Parametern, wie dem Membranpotential und dem Schwellwert, alle Synapsengewichte der presynaptischen Neuronen mit synapsen-spezifischen Parametern, wie Abklingkonstanten und Verstärkungsfaktoren. Für jedes Neuron wird ein Informationsblock, der Membranpotential, externen Eingangsreiz und Sprungadresse beinhaltet, von 16 Byte vorgesehen und jeder presynaptischer Gewichtswert, inklusive synapsen-spezifischen Parametern, wird mit 4 Byte codiert. Daraus ergibt sich bei einer Speicherkapazität aller 3 SDRAM Module von insgesamt 3 GB eine maximale Anzahl von $(3{\cdot}2^{30}{-}2^{19}{\cdot}16)/4{=}803{\cdot}10^6$ adaptiven Synpasen, die vom Digitalsimulator berücksichtigt werden können.

Das NSC-FPGA operiert ebenfalls in 2 Phasen: Neuronen-Puls-Phase und Neuronen-Aktualisierungs-Phase. Nach dem Lesen der FEL ist der nächste Zeitpunkt bekannt, an dem ein Neuron aufhört zu pulsen (Puls-Stop-Ereignis). In der folgenden Neuronen-Puls-Phase wird ermittelt, ob vor diesem Puls-Stop-Ereignis ein Neuron einen Puls generieren wird (Puls-Start-Ereignis). Dabei muss für alle Neuronen, die Pulse von anderen Neuronen empfangen, nach Gl. 1 eine numerische Integration durchgeführt werden. Bei Neuronen, die lediglich einen externen Eingangsreiz erhalten, lässt sich der Zeitpunkt des Puls-Start-Ereignisses ohne numerische Integration ermitteln ($t{=}(\theta{-}a_K)/i_K$). In der Neuronen-Aktualisierungs-Phase werden schliesslich alle Membranpotentiale und Synpasengewichte mittels der numerischen Integration auf den Zeitpunkt des neu ermittelten Simulationsereignisses aktualisiert. Während eines Simulationsdurchlaufs werden somit maximal 2 numerische Integrationen pro Neuron durchgeführt. Die durchzuführende numerische Integration erfolgt nach dem Bulirsch-Stoer Verfahren (Stoer und Bulirsch, 2000) und besteht im wesentlichen aus 2 arithmetischen Operationen: der Modified-Midpoint Integration (MMID) und der Polynomextrapolation (PZEXTR). Die MMID berechnet für jeden Funktionswert $y(t_0)$ innerhalb eines Integrationsintervalls H einen neuen Funktionswert $y(t_0{+}H)$. Dafür werden $nstep$ verschiedene Zwischenwerte berechnet, indem H in gleichgrosse Subintervalle $h{=}H/nstep$ unterteilt wird. Für die Berechnung des nächsten Zwischenwertes k_{m+1} ist die Ableitung des momentanen Zwischenwertes k_m und der Funktionswert des letzten Zwischenwertes k_{m-1} notwendig. Die Formeln für die Berechnung der Zwischenwerte und des neuen Funktionswertes y_{nstep} sind der Gl. 3 zu entnehmen,

$$\begin{aligned} k_{m+1} &= k_{m-1} + 2 \cdot h \cdot f(x + m \cdot h, k_m) \\ y_{nstep} &= \tfrac{1}{2} \cdot \left[k_{nstep} + k_{nstep-1} + h \cdot f(x + H, k_{nstep})\right] \end{aligned} \tag{3}$$

Die PZEXTR führt eine Extrapolation der neu ermittelten Funktionswerte durch, indem nach Gl. 4 die beiden Variablen $Q_{i,k}$ und $D_{i,k}$ berechnet werden,

$$\begin{aligned} &y_i = Q_{i,0} = D_{i,0}; i = 0, \cdots, 7 \\ &Q_{i,k} = \left[y_i/(y_{i-k} - y_i)\right] \cdot \left(D_{i,k-1} - Q_{i-1,k-1}\right) \\ &D_{i,k} = \left[y_{i-k}/(y_{i-k} - y_i)\right] \cdot \left(D_{i,k-1} - Q_{i-1,k-1}\right) \end{aligned} \tag{4}$$

wobei die Indexvariable i für verschiedene Funktionswerte gilt, die durch die MMID ermittelt wurden. Der extrapolierte Funktionswert $y_{ext,i}$ ergibt sich dann aus der Summe nach Gl. 5,

$$y_{ext,i} = \sum_{k=0}^{i} Q_{i,k} \tag{5}$$

4 Abschätzung der Performanz

Für die Leistungsabschätzung des Digitalsimulators wurden 5 Eingangsschichten mit unterschiedlicher Neuronenanzahl (32·32; $\cdots$; 96·96) simuliert. Die Neuronen dieser Schichten sind lateral im Rahmen einer Vierernachbarschaft (n=4) oder einer Achternachbarschaft (n=8) miteinander verbunden. Jede Simulation wurde bis zu einer Netzwerkzeit von 500 ms durchgeführt. Die Membranpotentiale aller Neuronen wurden mit zufälligen Werten zwischen 0 und 1 inititalisiert (θ=1; t_d=1 ms) und für alle Synapsen, die mit dem Wert 0, 12 initialisiert wurden, wurde die Adaptionsregel nach Gl. 2 angewendet (γ=0,1; μ=0,3).

4.1 Software Simulationszeiten

Während der SW Simulationen, die auf einem Linux-PC (2,4 GHz Pentium-4; 1 GB RAM) durchgeführt wurden, wurden 4 Variablen protokolliert: die Anzahl der aufgetretenen Simulationsereigniszeiten (N_{EVENT}), die Anzahl der durchgeführten Integrationen pro Neuron (N_{BSSTEP}), die Anzahl der Integrationsintervalländerungen (N_H) und die Anzahl der notwendigen Integrationsschrittunterteilungen pro Integration (N_I). In Abb. 4 sind diese Variablen für die

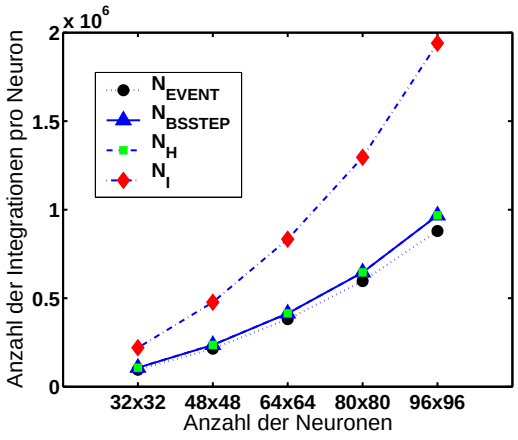

Abbildung 4. Durchgeführte Integrationen pro Neuron.

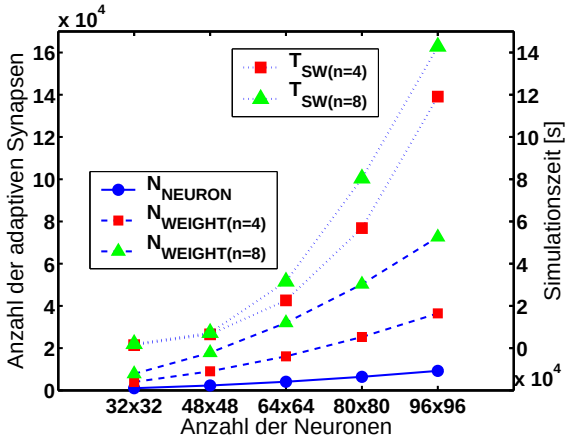

Abbildung 5. Synapsen, Neuronen und Software Simulationszeiten.

5 Eingangsschichten mit einer Vierernachbarschaftsverbindung dargestellt. Man erkennt, dass für jede durchgeführte Integration nahezu keine Integrationsintervalländerung notwendig war (H_{AVG}=$N_H/N_{BSSTEP}\simeq$1). Die Betrachtung der Variablen N_I verdeutlicht, dass durchschnittlich 2 Integrationsschrittunterteilungen pro Integration notwendig waren (I_{AVG}=$N_I/N_{BSSTEP}\simeq$2). Die SW Simulationszeiten T_{SW} berücksichtigen nur die Zeit, die für die Durchführung der numerischen Integration notwendig war (etwa 70% der gesamten Simulationszeit). In Abb. 5 sind die SW Simulationszeiten, die mit zunehmender Anzahl der Neuronen (N_{NEURON}) und Synapsen (N_{WEIGHT}) exponentiell ansteigen, dargestellt. Die Simulation einer 96·96 grossen Eingangsschicht mit einer Achternachbarschaftsverbindung dauert somit mehr als anderthalb Tage (142 834 s).

4.2 Simulationszeiten des Digitalsimulators

Die Simulationszeiten des Digitalsimulators T_{DS} lassen sich nach folgender Formel abschätzen,

$$T_{DS} = N_{BSSTEP} \cdot \frac{N_{NEURON}}{N_{IO}} \cdot T_{CLK} \cdot T_{NEURON}(n) \quad (6)$$

wobei N_{NEURON} der Anzahl der zu simulierenden Neuronen entspricht, N_{IO} ist die Anzahl der parallel nutzbaren Datenkanäle, T_{CLK} ist die Taktperiode und T_{NEURON} entspricht der Taktanzahl, die für die numerische Integration eines Neurons benötigt wird. Diese Taktanzahl ist von der Anzahl n der presynaptischen Gewichte abhängig und kann wie folgt berechnet werden,

$$T_{NEURON} = H_{AVG} \cdot \sum_{i=0}^{I_{AVG}-1} [T_{MMID}(n,i) + T_{PZEXTR}(n,i)] + t_{SDRAM} \quad (7)$$

wobei t_{SDRAM} die Latenzzeit in Takten für den Zugriff der NIBs im SDRAM ist. T_{MMID} und T_{PZEXTR} stellen die Anzahl der Takte dar, die für die MMID und die PZEXTR notwendig sind. Die notwendige Taktanzahl für die MMID ist in Gl. 8 angegeben,

$$T_{MMID} = t_{MMID} + \text{ceil}(n/2) \cdot [2 \cdot (i+1) + 1)] \quad (8)$$

wobei t_{MMID} der Latenzzeit in Takten entspricht, bevor das erste gültige Datum aus der Pipeline des implementierten HW Moduls erscheint. Der Term $n/2$ in Gl. 8 kommt dadurch zustande, dass die Datenbusbreite der SDRAM Module 8 Bytes beträgt und somit stets 2 Synapsengewichte gleichzeitig gelesen werden können. Die notwendige Taktanzahl für die PZEXTR ist in Gl. 9 beschrieben,

$$T_{PZEXTR} = t_{PZEXTR} + \text{ceil}(n/2) + i \cdot [\text{ceil}(n/(2 \cdot t_{PZEXTR})) \cdot t_{PZEXTR}] \quad (9)$$

wobei t_{PZEXTR} wieder der Latenzzeit in Takten entspricht, die aufgrund der Pipeline des HW Moduls resultiert. Anhand Gl. 8 und Gl. 9 kann nun durch Einsetzen der entsprechenden Latenzzeiten (t_{SDRAM}=10, t_{MMID}=12, t_{PZEXTR} = 4) die Berechnungsdauer eines Neurons in Takten mittels Gl. 7 für eine Vierer- (n=4) und Achternachbarschaft (n=8) angegeben werden (H_{AVG}=1, I_{AVG}=2),

$$T_{NEURON}(n=4) = 66 \qquad T_{NEURON}(n=8) = 86 \quad (10)$$

Für eine Taktfrequenz von 50 MHz (T_{CLK}=20 ns) und der parallelen Nutzung von 3 Datenkanälen (N_{IO}=3) lassen sich mit der durch die SW Simulation ermittelten Anzahl an durchgeführten Integrationen pro Neuron (N_{BSSTEP}) die Simulationszeiten des Digitalsimulators mit Hilfe von Gl. 6 abschätzen. Die resultierenden Beschleunigungsfaktoren F berechnen sich aus dem Verhältnis von T_{SW} zu T_{DS} und sind in der Tabelle 1 zusammengefasst.

5 FPGA Prototypen-Platine

Die realisierte FPGA Prototypen-Platine in Abb. 6 zeigt die 3 FPGAs, auf denen die Architektur des Digitalsimulators beruht. Das PPC2-FPGA hat Zugang zu den beiden Ereignislisten DEL und FEL, das NTC-FPGA hat Zugriff auf die beiden Tag-Felder FTF und ETF, und das NSC-FPGA steuert die 3 SDRAM Module an, um die entsprechenden NIBs aktualisieren zu können. Die 250×150×2 mm grosse 10-lagige

Tabelle 1. Abgeschätzte Simulationszeiten des Digitalsimulators.

n	N_{NEURON}	N_{BSSTEP}	T_{SW}	T_{DS}	F
	32·32	98 717	1405 s	45 s	31,2
	48·48	222 365	6527 s	226 s	28,9
	64·64	420 299	22620 s	758 s	29,8
	80·80	721 463	65277 s	2032 s	32,1
	96·96	926458	119109 s	3757 s	31,7
	32·32	107 276	1990 s	63 s	31,6
	48·48	235 863	7263 s	312 s	23,3
	64·64	413 861	31 548 s	972 s	32,5
	80·80	645 694	80 378 s	2370 s	33,9
	96·96	967 572	142 834 s	5113 s	27,9

Prototypen-Platine (2 Masse-, 2 Versorgungs- und 6 Signal-Lagen) verwendet mehr als 350 Entkoppelkapazitäten und muss 5 verschiedene Versorgungsspannungen (1.5 V; 1.8 V; 2.5 V; 3.3 V und 5 V) bereitstellen.

6 Schlussfolgerungen

Es wurde eine Digitalsimulator für grosse PCNNs vorgestellt, dessen Flexibilität auf der Programmierbarkeit von SW und auf der Rekonfigurierbarkeit von beschleunigender HW basiert. Bei einer Taktfrequenz von 50 MHz verspricht der Digitalsimulator einen Beschleunigungsfaktor von etwa 30 für eine spärliche Vernetzungsstruktur (Vierer- und Achternachbarschaft) im Vergleich zu einem konventionellen PC. Die Architektur des Digitalsimulators ist durch eine verteilte Speicherarchitektur, eine effiziente FPGA-Nutzung, eine hohe Speicherbandbreite und ausreichende Speicherkapazität charakterisiert. Die realisierte FPGA-Prototypen-Platine befindet sich zur Zeit in der Testphase, um die Funktionalität des Digitalsimulators zu überprüfen und die abgeschätzten Simulationszeiten zu validieren.

7 Danksagung

Wir danken W. Maurer und D. Backhaus von der CAD-UL GmbH für die Unterstützung bei der Entflechtung und Bestückung, sowie der Rinde GmbH für die Herstellung der Prototypen-Platine.

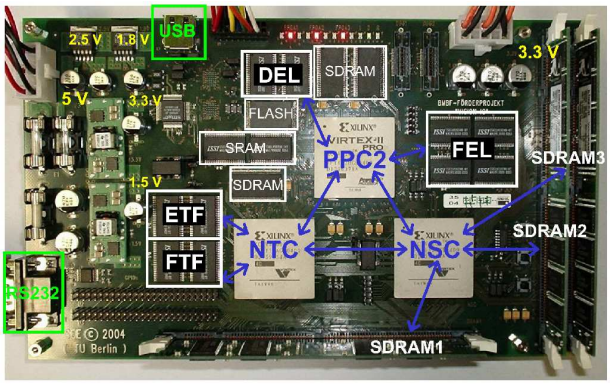

Abbildung 6. FPGA-Prototypen-Platine des Digitalsimulators.

Literatur

Gerstner, W. und Kistler, W. M.: Spiking Neuron Models – Single Neurons, Populations, Plasticity, Cambridge University Press, ISBN 0-521-81384-0, 2002.

Heittmann, A. und Ramacher, U.: Correlation-Based Feature Detection Using Pulsed Neural Networks, IEEE 13th Workshop on Neural Networks for Signal Processing, 479–488, 2003.

Hellmich, H. H. und Klar, H.: An FPGA based Simulation Acceleration Platform for Spiking Neural Networks, IEEE 47th Midwest Symposium on Circuits and Systems, 2, 389–392, 2004.

Jahnke, A., Roth, U. und Schönauer, T.: Digital Simulation of Spiking Neural Networks, Pulsed Neural Networks, Herausgeber: Maass, W. und Bishop, C. M., MIT Press, ISBN 0-262-13350-4, 1998.

Krupnova, H. und Saucier, G.: FPGA-Based Emulation: Industrial and Custom Prototyping Solutions, 10th Int. Conf. on Field Programmable Logic and Applications, 68–77, 2000.

Mehrtash, N., Jung, D., Hellmich, H. H., Schönauer, T., Lu, V. T. und Klar, H.: Synaptic Plasticity in Spiking Neural Networks (SPINN): A System Approach, IEEE Trans. on Neural Networks, 14-5, 980–992, 2003.

Reyneri, L. M.: Implementation Issues of Neuro-Fuzzy Hardware: Going Toward HW/SW Codesign, IEEE Trans. on Neural Networks, 14-1, 176–194, 2003.

Schäfer, M., Schönauer, T., Wolff, C.,Hartmann, G., Klar, H. und Rückert, U.: Simulation of Spiking Neural Networks: Architectures and Implementations, Neurocomputing, 48, 647–679, 2002.

Stoer, J. und Bulirsch, R.: Numerische Mathematik II, Springer Verlag, ISBN 3-540-67644-9, 2000.

Xilinx: Virtex-II/II-Pro Datenblätter, www.xilinx.com/xlnx/xweb/xil_publications_index.jsp, 2004.