Artificial Neural Networks in Hardware: A Survey of Two Decades of Progress[*]

Janardan Misra[†]
HTS Research Lab
151/1 Doraisanipalya, BG Road, Bangalore 560076, India
Email: janardan.misra@gmail.com
Indranil Saha
Computer Science Department
University of California, Los Angeles, CA 90095, USA
Email: indranil@cs.ucla.edu

Abstract

This article presents a comprehensive overview of the hardware realizations of Artificial Neural Network (ANN) models, known as Hardware Neural Networks (HNN), appearing in academic studies as prototypes as well as in commercial use. HNN research has witnessed a steady progress for more than last two decades, though commercial adoption of the technology has been relatively slower. We study the overall progress in the field across all major ANN models, hardware design approaches, and applications. We outline underlying design approaches for mapping an ANN model onto a compact, reliable, and energy efficient hardware entailing computation and communication and survey a wide range of illustrative examples. Chip design approaches (digital, analog, hybrid, and FPGA based) at neuronal level and as neurochips realizing complete ANN models are studied. We specifically discuss, in detail, neuromorphic designs including spiking neural network hardware, cellular neural network implementations, reconfigurable FPGA based implementations, in particular, for stochastic ANN models, and optical implementations. Parallel digital implementations employing bit-slice, systolic, and SIMD architectures, implementations for associative neural memories, and RAM based implementations are also outlined. We trace the recent trends and explore potential future research directions.

Keyworld: Hardware neural network, neurochip, parallel neural architecture, digital neural design, analog neural design, hybrid neural design, neuromorphic system, FPGA based ANN implementation, CNN implementation, RAM based implementation, optical neural network.

1 Introduction

Hardware devices designed to realize Artificial Neural Network (ANN) architectures and associated learning algorithms especially taking advantage of the inherent parallelism in the neural processing are referred as Hardware Neural Networks (HNN). Although most of the existing ANN applications in commercial use are often developed as software, there are specific applications such as streaming video compression, which demand high volume adaptive real-time processing and learning of large data-sets in reasonable time and necessitate the use of energy-efficient ANN hardware with truly parallel processing capabilities. Specialized ANN hardware (which can either support or replace software) offers appreciable advantages in these situations as can be traced as follows [195]:

- Speed: Specialized hardware can offer very high computational power at limited price and thus can achieve several orders of speed-up, especially in the neural domain where parallelism and distributed

[*]This is an earlier (and extended) version of the article: *J. Misra and I. Saha: "Artificial neural networks in hardware: A survey of two decades of progress". Neurocomputing 74(1-3): 239-255 (2010).*

[†]Corresponding author.

computing are inherently involved. For example, Very Large Scale Integration (VLSI) implementations for Cellular Neural Networks (CNNs) can achieve speeds upto several teraflops [115] (2000), which otherwise is a very high speed for conventional DSPs, PCs, or even work stations.

- Cost: A hardware implementation can provide margins for reducing system cost by lowering the total component count and decreasing power requirements. This can be important in certain high-volume applications, such as ubiquitous consumer-products for real-time image processing, that are very price-sensitive.

- Graceful Degradation: An intrinsic limitation of any sequential uni-processor based application is its vulnerability to stop functioning due to faults in the system (*fail-stop* operations). Primary reason is the lack of sufficient redundancy in the processor architecture. As some recent studies [329] (2007) suggest, even with the advancement and introduction of the multi-core PC processors architectures, the need for having effective fault-tolerant mechanisms is still present. In contrast to this parallel and distributed architectures allow applications to continue functioning though with slightly reduced performance (*graceful degradation*) even in the presence of faults in some components. For those ANN application which require complete availability or are safety critical, fault tolerance is of utmost importance and in this respect parallel hardware implementations offer considerable advantage.

Mapping highly irregular and non-planar interconnection topology entailing complex computations and distributed communication on regular two dimensional surfaces poses significant challenge for the (VLSI) HNN designers. Also since hardware constraints (especially analog components) may introduce computational errors, degradation of learning and lack of accuracy in results become a major challenge while designing HNNs. These errors can divert the trajectory of the learning process, generally increasing the number of cycles required to achieve convergence. Non-linearity of activation functions poses yet another challenge while designing a compact hardware. To address these challenges wide spectrum of technologies and architectures have been explored in the past. These include digital [172, 134, 31] (1992,1995,2003), analog [214, 39] (1989,2004), hybrid [278, 186] (1999,2004), FPGA based [281, 233, 2] (2005,2007,2009), and (non-electronic) optical implementations [222, 300, 176] (1996,2000,2007). At this point it is important to add that, for practical purposes, a HNN realizing an ANN model alone is not sufficient by itself and a fully operational system would demand many other components e.g., for sensor acquisition, for pre and post processing of inputs and outputs etc.

Although not as widespread as ANNs in software, there do exist HNNs at work in real world applications. Examples include Optical Character Recognition, Voice Recognition (Sensory Inc. RSC Micro controllers and ASSP speech recognition specific chips), Traffic Monitoring (Nestor TrafficVision Systems), Experiments in High Energy Physics [72] (1993) (Online data filter and Level II trigger in H1 electron-proton collision experiment using Adaptive Solutions CNAPS boards), adaptive control, and robotics. See Table 1 for more examples.

With the advent of these technologies need of having timely surveys has also been felt. There are indeed several surveys which have appeared from time to time in the past. We will briefly discuss these surveys next.

Related Surveys: [214] (1989) by Mead, [172, Part IV] (1992) by Kung, and [105] (1994) by Glesner and Poechmueller are some early references on the VLSI implementations of the ANN models. Lindsey and Lindbad [195, 196] (1994,1995) present one of earliest detailed overviews of the field covering most of electronic approaches as well as commercial hardware. Heemskerk [122] (1995) presents an overview of Neurocomputers built from accelerator boards, general purpose processors, and neurochips coming out from both industries and academia upto mid 90s. Ienne et al. [135] (1996) present a survey of digital implementations by considering two basic designs: Parallel systems with standard digital components and parallel systems with custom processors. They also discuss their experience with running a small ANN problem on two of the commercially available machines and conclude that most of the training times are actually slower or only moderately faster than on a serial workstation. Aybay et al. [22] (1996) lay out a set of parameters, which can be used to classify and compare digital neurocomputers and neurochips. Moerland and Fiesler [221] (1997) present an overview of some of the important issues encountered while mapping an ideal ANN model onto a compact and reliable hardware implementation, like quantization and associated weight discretizations, analog nonuniformities, and nonideal responses etc. They also discuss hardware

Applications	Examples (HNN Types)
High Energy Physics	[72] (1993) (Digital-neurochip)
Pattern Recognition	[313] (2005)(FPGA), [31] (2003)(Digital)
Image/Object Recognition	[7, 21] (1984,2005) (RAM based), [268] (2008) (Optical)
Image Segmentation	[310] (1997) (FPGA), [279] (2002) (Digital), [103] (2006) (FPGA)
Generic Image/Video Processing	[158] (1994) (RAM based), [54, 119] (1991,2004) (Analog), [179] (1994) (Optical), [198](FPGA)
Intelligent Video Analytics	[169, 321] (2003) (FPGA), [260, 327] (2004) (Hybrid)
Finger Print Feature Extraction	[218] (2003) (Analog)
Direct Feedback Control	[197] (2002) (Analog)
Autonomous Robotics	[90] (2003) (Digital), [30] (2004) (FPGA), [16, 17] (2005,2006) (Hybrid), [247] (2007)(DSP)
Sensorless Control	[187] (2006) (FPGA)
Optical Character/Handwriting Recognition	[160] (2005) (Digital)
Acoustic Sound Recognition	[75] (2006) (DSP)
Adaptive Signal Processing	[151]
System identification	[85]
Adaptive Control	[85]
Real-Time Embedded Control	[53] (1995) (Digital)
Audio Synthesis	[44] (1994) (Analog)
Assignment Solver	[130] (2003) (Analog), [311] (1992) (Digital)
Olfactory Sensing	[166] (2007)

Table 1: Examples of HNN Applications

friendly learning algorithms. Sundararajan and Saratchandran [295] (1998) discuss in detail various parallel implementation aspects of several ANN models (Back Propagation (BP) based NNs, ART NN, recurrent NN etc) using various hardware architectures including scalable general purpose parallel computers and MIMD (multiple instruction multiple data) with MPI interface. Individual chapters discuss reviews, analysis, and experimental case studies, e.g., on implementations for BP based NNs and associated analysis of network and training set parallelisms. Burr [43, 42] (1992,1991) presents techniques for estimating chip-area, performance, and power consumption in the early stages of architectural exploration for HNN designs. These estimation techniques are further applied for predicting capacity and performance of some of the neuro architectures. Hammerstrom [114] (1998) provides an overview of the research done in the digital implementations of ANNs till late 90s. Reyneri [256] (2002) presents an annotated overview of the ANNs with “Pulse Stream” modulations including a comparative analysis of various existing modulations in terms of accuracy, response time, power, and and energy requirements. Zhu and Sutton [330] (2003) survey Field Programmable Gate Array (FPGA) based implementations of ANNs discussing different implementation techniques and design issues. Based upon the purpose of reconfiguration (prototyping and simulation, density enhancement, and topology adaptation) as well as data representation techniques (integer, floating point, and bit stream arithmetic) it provides taxonomy for classifying these implementations. Reyneri's survey on neuro-fuzzy hardware systems [257] (2003) is an important paper discussing various technological aspects of hardware implementation technologies with a focus on hardware/software co-design techniques. Diasa et al. [74] (2004) is one of the the latest surveys with specific focus to commercially available hardware. Schrauwen and D'Haene [281] (2005) provide a brief overview of some of the recent FPGA based implementations of Spiking Neural Networks (SNN). Another more recent article by Maguire et al. [207] (2007) also presents a detailed overview of FPGA based implementations of SNN models and brings out important challenges ahead. Bartolozzi and Indiveri in [26] (2007) provide a comparative analysis of various hardware implementations for the spiking synaptic models. Smith [292] (2006) surveys digital and analog VLSI implementation approaches for neuronal models with or without explicit time. Probably the most recent survey of the field with very interesting critical historical analysis of the major developments and limitations of digital, analog, and HNN approaches is presented by Hammerstrom and Waser in [112] (2008). Also Indiveri et al. [137] (2009) present a survey of the recent progress in the field of neuromorphic designs and discusses challenges ahead for augmenting these systems with cognitive capabilities. Some of the HNN topics have found wider audience and there are specialized volumes on these topics. An edited volume by Austin [20] (1998) provides a detailed glimpse on the RAM based HNN designs. Similarly another edited volume [238] (2006) by Ormoindi and Rajapakse presents a recent update on FPGA based ANN implementations including foundational issues, various implementations, and lessons learned from a large scale project. An edited volume by Valle [303] presents discussions on various approaches to build smart adaptive devices.

Even though there exist several reviews and edited volumes on the subject, most of these either focus on specific aspects of HNN research or may not be so recent. This paper attempts to survey on all major HNN design approaches and models discussed in literature and in commercial use. Primary objective is to review the overall progress in the field of HNN over last two decades across all major ANN models, hardware design approaches, and applications. We cover these topics by including most of the important works which have appeared in the literature with an optimistic perspective. However, owing to space limitations, there are topics, which will not be covered in this survey including *hardware friendly learning algorithms* (e.g., perturbation learning [148] (1992), constructive learning [291] (1993), cascade error projection learning [79, 80] (1995, 2000), and local learning [52] (2004) with its special case of spike based Hebbian learning [138] (2007)), *HNN designs focused on specific ANN models* (e.g., MLP with back propagation [171, 69] (1994, 2000), radial basis function networks [306, 321, 84] (1994, 2003, 2004), and Neocognitron [245] (2001)), and *neurocomputers* [105, 294] (1994,1996).

Rest of the paper is organized as follows: Issues related to the parameters used for evaluating an HNN system are highlighted in Section 2. Section 2 also presents discussion on difficulties in HNN classification. Section 4 deals with different electronic approaches to implement a single neuron, whereas Section 5 provides a presentation on complete HNN models available as chips. CNN implementations are covered in Section 7. Neumorphic systems including implementations for spiking NNs are covered in Section 8. A discussion on optical neurocomputers appears in Section 10. Finally Section 12 concludes the article by outlining some of the possible future research directions.

2 Evaluation Parameters and Classification

An ANN is generally specified in terms of the network topology, activation function (AF), learning algorithm, number and type of inputs/outputs, number of processing elements (neurons) and synaptic interconnections, number of layers etc. For a hardware implementation, in addition, specifications may include the technology used (analog, digital, hybrid, or FPGA), data representation (fixed/floating-point), weight storage, bits of precision, programmable or hardwired connections, on-chip learning or chip-in-the-loop training, on-chip or off-chip transfer function, *e.g.*, look–up table, and degree of cascadability.

Based upon these parameters, various figures of merit are derived to indicate the resultant hardware performance. The most common *performance ratings* include

- *Connections-Per-Second (CPS)* for processing speed: Rate of multiplication/accumulate operations and transfer function computation during testing phase. This indicates how well the specific algorithm suits the architecture.
- *Connection-Updates-Per-Second (CUPS)* for learning speed: Rate of weight changes during learning, involving calculation and update of weights. This measures how fast a system is able to perform input-output mappings.
- *Synaptic Energy* Average energy required to compute and update each synapse. Measured as $WCPS$ (watt per connection-per-second) or *J per connection* [256] (2002).

Sometimes normalizing the CPS value by the number of weights on the chip (*CPSPW*, or *CPS per weight*) can be a better way to indicate the processing power of the chip [105] (1994). Similar ratings could be defined for CUPS. Although CUPS normally refers to BP learning, its value could be given for other algorithms as well. However note that an algorithm such as Boltzmann learning [116] (1996) may only need a few passes through the training set as compared to perhaps 1000's of epochs for BP. So a Boltzmann chip may have a lower CUPS value than a BP chip, and yet accomplish the learning in a shorter time. Keulen et al. [159] (1994) propose an improved measure that also accounts for accuracy by defining bit connection primitives per second: CPPS = $b_i \times b_w \times CPS$, with b_i and b_w denoting input and weight accuracy in bits respectively. For RBF, instead of these, *pattern presentation rate* is actually used as a performance parameter.

Hardware constraints, such as weights/ states precision, finite arithmetic/ quantization effects caused by discrete values of the channel length, width of MOS transistor geometries, and AF realization play a major role in HNNs. Cornu and Ienne [65] (1994) introduce the notion of algorithmic efficiency for performance measurement and evaluation of digital neurocomputers. Algorithmic efficiency is defined as a measure of the effect of the hardware constraints on the convergence properties of various ANN models to be simulated on a neurocomputer. They argue that comparing relative speeds in MCUPS is not sufficient and instead estimate global speedup of a neurocomputer as a product of its raw hardware speedup (corresponding to MCUPS) and the algorithmic efficiency (w.r.t. a specific ANN model).

The non-linearity associated with the AFs represents one of the major bottlenecks in digital VLSI implementation of ANNs, involving large overheads in time and silicon area. Possible solutions include use of look–up tables [113, 235] (1990,2003) and piecewise linear approximating functions [10] (1997). In case of look–up table, table size again imposes an upper bound on the number of bits. A statistical study by Holt and Hwang [129] (1993) on the precision requirements for a two layer MLP with BP learning showed that under certain assumptions (e.g., uniformly distributed input variables) a fixed point encoding of 16 bit is sufficient and at least 12 bits might be essential. A mathematical analysis of the effect of limited precision in analog hardware for weight adaptation in on-chip learning for single layer feed-forward neural networks is discussed by Annema et al. [14] (1994). The analysis is further applied for a worst-case estimation of the minimum size of the weight storage capacitors. Vollmer and Strey [308] (1999) present the experimental analysis on the precision requirements under RBF and other training algorithms and conclude that RBF requires at least 20 bits of precision as compared to 16 bits for BP. Bieu [29] (1998) presents several upper and lower bounds for the number-of-bits required for solving a classification problem using neural networks. These bounds are in turn used to devise ways for efficiently building the hardware implementations. Use of 1st and 2nd order Taylor interpolation also provides relatively high accuracy (up to 16-20 bits) even with very small look-up tables (256 words).

For large scale neural network, synaptic storage density is very important, and memory optimization plays an important role. However there is a trade-off between the memory size and power consumption in the memory - one transistor DRAM has the highest density, but consumes more power than SRAM, as DRAM memory cells need to be refreshed due to leakage current, on the other hand six transistor SRAM consumes the least power, but achieves density which is factor 4 worse than one transistor DRAM.

Hardware Neural Network Classification

Neural network hardware is becoming increasingly difficult to classify in a way that the classification yields useful comparative information for practical purposes. Primary source of difficulty arises from the multitude of characteristics associated with any such hardware implementation both arising from chosen hardware as well as underlying ANN model. As mentioned before, Aybay et al. [22] (1996) list several classification attributes including transfer function characteristics: on-chip/off-chip, analog/digital, threshold/look-up table/computation; cascadibility, clock and data transfer rates. Based upon these attributes several HNN chips and designs were classified. Though such a classification covers wide range of attributes, extracting information for practical purposes using comparative analysis is relatively difficult. For these reasons, we do not attempt here to present another classification, though instead structure the discussion under several themes - starting with a discussion on basic neuronal level hardware designs, then progressing to the chip level approaches for various ANN models, followed by a discussion on several parallel implementation of specific ANN models including CNN, and finally focusing discussion on specific approaches including neuromorphic designs, and optical neurocomputers. In Table 2, we present an overview of the examples of various HNN implementations across wide range of ANN models. Forthcoming sections provide further details on these.

3 Hardware Friendly Neural Algorithms

HNNs involve a variety of implementations using digital, analog, optical, and hybrid techniques. A common factor is the mapping of neural network algorithms onto a reliable, compact, low cost, and fast hardware, optimizing certain constraints like accuracy, space, power, and processing speed. The design aspect is governed by a balance of these criteria. In this framework, hardware friendly learning algorithms offer significant advantages in design and overall reduction in the manufacturing cost albeit at the slight compromise on the

HNN / ANN	Digital	Analog	Hybrid	Neuromorphic	FPGA	Optical
MLP	[318](1990)				[96](2000)	
RBF	[82](1997)	[306](1994) [84](2004)			[321](2003)	
SOFM	[267](1994) [82](1997)	[106](2003)				
Feed Forward Network	[171](1994) [82](1997)				[187](2006) [233](2007)	[150](1993) [275](1995)
Spiking NN	[279](2002)			[194](2005) [276](2001)	[30](2004) [276](2001) [281](2005)	
Pulse Coded NN		[240](1999)		[240](1999) [162](1999)	[206](2003) [125](2003)	
CNN	[273](1995) [272](1999)	[118](1992) [161](1995) [17](2006)	[16](2005)		[198](2005) [246](2007) [2](2009)	[300](2000)
Associative Memory		[266](1991)	[123](2002)		[313](2005)	[1](1987) [144](1989)
Recurrent NN		[39](2004)				[322](1990)
Stochastic NN					[68](1993) [24](1994) [233](2007)	

Table 2: ANN-HNN Table

algorithmic performance (e.g., convergence rate). Some of the widely applied algorithms in HNN designs are discussed below:

- **Perturbation Algorithms:** The general idea is to obtain a direct estimate of the gradients (in multilayer BP networks) by a slight random perturbation of some network parameters, using the forward pass of the network to measure the resulting network error. These onchip training techniques not only eliminate the complex backward pass but are also likely to be more robust to any non-idealities occurring in hardware. The two main variants of this class of algorithms are node perturbation and weight perturbation [148] (1992). Their main disadvantage lies in the sequential weight update calculation as opposed to the parallel calculations in the conventional BP algorithms.

- **Local Learning Algorithms:** The implementation of a learning rule can be greatly simplified if it uses information that is locally available [242] (1993), thereby minimizing the amount of wiring and communication involved. Several local learning algorithms, that avoid a global back propagation of error signals, have been designed. An example is an antiHebbian learning algorithm that is suitable for optical neural networks [249] (1993). The weight updates in this algorithm depend only on the input and output of that layer and one global error signal. Although not a steepest descent rule, it is still guaranteed that the weights are updated along the descent direction. A promising approach is taken in the stochastic *Alopex* (ALgorithm Of Pattern EXtraction), based on the correlation between the changes of the individual weights and the network's error measure [220] (1996). The main advantage of this approach is that the weights can be updated synchronously and no modeling of the multipliers and AFs is needed. Chen et al. [52] (2004) present two novel Monte Carlo sampling-based Alopex for training neural networks, combining the sequential Monte Carlo estimation and Alopex-like procedure for gradient-free optimization, and the learning proceeds within the recursive Bayesian estimation framework.

- **Networks with Heaviside Functions:** The design of a compact digital neural network can be simplified considerably when Heaviside functions are used as AFs instead of differentiable sigmoidal functions. One of the earliest examples of such a learning rule is Madaline2 [314] (1990). It is based on node perturbation, but the training error is minimized by investigating the effect of an inversion of the activation value of a neuron. If this inversion reduces the Hamming error on the output neurons, the incoming weights of the inverted neuron are adapted with a perceptron training algorithm to reinforce this inversion. There exist constructive algorithms [291] (1993), which gradually build a

Heaviside network by adding neurons and weights. These algorithms are often based on the perceptron algorithm that is used to adapt the weights of the freshly added neurons.

- **Cascade Error Projection Learning:** Duong [79] (1995) presented a new learning algorithm, quite suitable for VLSI implementations of ANNs. The algorithm was termed as Cascade Error Projection (CEP), which is based upon cascade correlation architecture developed earlier by Fahlman and Lebiere [83] (1990), which incrementally adds new neurons as hidden units during training phase. A detailed mathematical analysis of CEP was presented in [80] (2000). CEP involves fewer iterations and is more tolerant of low resolution in the quantization of synaptic weights. Therefore, CEP learns relatively quickly and the circuitry needed to implement it is also simple. At this point is useful to distinguish between the number of weight bits stored (usually 16-32 bits) which affects training and the number of bits (usually the MSBs) effectively used in multiplication (usually 8-16 bits) to reduce the complexity of HW (or SW) multipliers.

- **Cellular Neural Network:** Chua and Yang [59, 58, 56] (1988,1993) introduced CNN as an regular array of locally interconnected analog processing elements, or cells, operating in parallel, whose dynamic behavior is determined by the cell connectivity pattern (neighborhood extent) and a set of configurable parameters. CNN by its very design is a circuit oriented architecture and is conceptually suitable for hardware implementation. This is of special interest for VLSI implementation owing to the sparse local connectivity present in a CNN.

4 Hardware Approaches to Neuronal Design

The transmission of signals in biological neurons through synapses is a complex chemical process in which specific neurotransmitter substances are released. Their effect is to change the electrical potential in the receiving cell by changing the Osmotic and ionic equilibrium across the cell membrane. If this potential reaches a threshold, the neuron fires. Artificial neuron models attempt to reproduce this phenomena at varying levels of abstractions [121] (2008).

In this section we describe the basic structure of digital and analog neurons used for HNN implementations and briefly discuss the implementations of spiking neurons and their synaptic dynamics. An analog implementation is usually efficient in terms of chip area and processing speed, but this comes at the price of a limited accuracy of the network components. In a digital implementation, on the other hand, accuracy is achieved at the cost of efficiency (e.g., relatively larger chip area, higher cost, and more power consumption). This amounts to a trade off between the accuracy of the implementation and the efficiency of its performance.

It is also important to add at this point that the HW designs to be discussed throughout this paper involve significant manual ad-hoc steps, which is a time-consuming and expensive operation and a major factor in increasing *'time-to-market'*. We will have bit more to say on this in the conclusion section.

4.1 Digital Neuron

In a digital neuron, synaptic weights are stored in shift registers, latches, or memories. Memory storage alternatives include one, two or three transistor dynamic RAM, or four or six transistor static RAM [105] (1994). Adders, subtracters, and multipliers are available as standard circuits, and non-linear AFs can be constructed using look-up tables or using adders, multipliers etc. A digital implementation entails advantages like simplicity, high signal-to-noise ratio, easily achievable cascadability and flexibility, and cheap fabrication, along with some demerits like slower operations (especially in the weight $\times$ input multiplication). Also conversion of the digital representations to and from an analog form may be required since usually input patterns are available in analog form and control outputs also often required to be in analog form.

In a recent work Muthuramalingam et al. [227] (2007) discuss in detail issues involved with the implementation of a single neuron in FPGA including serial versus parallel implementation of computational blocks, bit precision and use of look-up tables. Hikawa [125] (2003) describes digital pulse-mode neuron which employs piecewise-linear function as its AF. The neuron is implemented on a FPGA rendering the piecewise-linear function programmable and robust against the changes in the number of inputs. In [284, 67] (1991,1994), Daalen et al. demonstrate through experiments how linear and sigmoid AFs can be generated in a digital stochastic bit stream neuron. The AF of the neuron is not built in the hardware explicitly, rather it is generated by the interaction of two probability distributions. Different AFs can be generated by controlling the distribution of the threshold values provided to each neuron.

Skrbek [290] (1999) presents an architecture and overview of *shift-add neural arithmetic*, for an optimized implementation of multiplication, square root, logarithm, exponent and nonlinear AFs at neuronal level for fast perceptron and RBF models. Functions are linearly approximated, for example, 2^x is be approximated as $2^{int(x)}(1 + frac(x))$ where $int(x)$ calculates the integral part of x and $frac(x)$ is its fractional part. Shift operation calculates $2^{int(x)}$, whereas linear approximation $(1 + frac(x))$ approximates remaining $2^{frac(x)}$. Further an FPGA based implementation for the shift-add arithmetic is discussed involving only adders and barrel shifters.

4.2 Analog Neuron

In an analog neuron weights are usually stored using one of the following: resistors [108] (1987), charge-coupled devices [3] (1990), capacitors [224] (1992), and floating gate EEPROMS [128] (1990). In VLSI, a variable resistor as a weight can be implemented as a circuit involving two MOSFETs [316] (2000). However, discrete values of channel length and width of the MOS transistors may cause quantization effect in the values of the weight. The scalar product and subsequent nonlinear mapping is performed by a summing amplifier with saturation [332] (1992).

In the analog domain the characteristic nonlinear functionality of neuronal AF can sometimes be captured directly (*e.g.*, above saturation level current and voltage characteristics of transistors), yet a coherent set of all the basic elements is difficult to achieve. As the AFs used in software ANN implementations cannot be easily implemented in VLSI, some approximation functions are instead considered to act as AFs [316] (2000). Also analog neuron implementations benefit by exploiting simple physical effects to carry out some of the network functions [214] (1989). For example, the accumulator can be a common output line to sum currents. Analog elements are generally smaller and simpler than their digital counterparts. On the other hand, obtaining consistently precise analog circuits, especially to compensate for variations in temperature and control voltages, requires sophisticated design and fabrication.

In analog modeling, signals are typically represented by currents [306] (1994) and/or voltages [128] (1990) which work with real numbers. Current flow is preserved at each junction point by Kirchhoff's Current Law, and during multiplication various resistance values can be used for the weighting operation of the signal. Thus a network of resistors can simulate the necessary network connections and their resistances are the adaptive weights needed for learning. Besides, the non linear voltage-current response curve of field effect transistors (FETs) makes them especially suitable for simulating neuronal AFs. However, the encoding of signals as voltages makes certain operations like addition rather difficult to implement as compared to multiplication and threshold activation. Also a major problem with this representation scheme is that before performing any operation a signal needs to be held constant for some time. The current which flows between the source and sink depends linearly on the potential difference between them, and the proportionality constant is determined by the stored charge. Learning involves weight updates corresponding to changes in the amount of charge stored. Even if the power source is disconnected, the magnitude of the weights remains unchanged. A different approach [54] (1991), with charged coupled devices (CCDs), is used to store the charge dynamically.

The main challenges for analog designs are the synapse multiplier over a useful range and the storage of the synapse weights. Moreover, there are some characteristics inherent to analog computation like the spatial non-uniformity of components (which are particularly troublesome when the training of the network is done off-chip, without taking these component variations into account) and non-ideal responses (that particularly affect the implementation of a linear multiplication and nonlinear AF, like the standard sigmoid).

There are also attempts for designing digitally programmable analog building blocks for ANN implementations. Almeida and Franca [9] (1996) propose a synapse architecture combining a quasi-passive algorithmic digital to analog converter providing a 7-bit bipolar weight range and on-chip refreshing of the analog weight followed by a four quadrant analog-digital multiplier with extended linear range. Hamid et al. [111] (2005) discuss an approach of including the effect of Deep Sub-Micrometer (DSM) noise in MOSFETs for circuit-level and architecture-level simulations. They show that that DSM noise has the potential to be exploited for probabilistic neural computation architecture hardware implementation. For example, they tested the effect of a noisy multiplier on the performance of Continuous Restricted Boltzmann Machine (CRBM) [48, 49] (2002,2003) and result demonstrate that stochastic neuron implemented using noisy MOSFET can produce performance comparable with that of a "perfect" CRBM with explicit noise injected into it.

4.3 Silicon Implementation of Spiking Neuron and its Synaptic Dynamics

Actual communication between biological neurons happens by short electrical pulses, which are known as action potentials or spikes. Integrate and fire (I&F) neuron model is among the simplest models with spiking dynamics. An I&F neuron model can handle continuously time varying signals, support synchronization, and is computationally powerful as compared to non spiking neuron models [100] (2002). Leaky I&F model and its generalization as spike response model, non-linear I&F model, Hodgkin-Huxley model, Mihalas-Niebur model, and Morris-Lecar model are among the better known extensions of the basic I&F model. Networks of I&F neurons exhibit a wide range of capabilities including feature binding, segmentation, pattern recognition, onset detection, and input prediction [202] (2001). We will next briefly discuss some of the representative hardware implementations (generally using mixed-mode circuits) for I&F model and some of its extensions since they are often used while designing neuromorphic systems as discussed later in the Section 8.

For adequately realizing an I&F model in hardware, it is necessary that the realized hardware can set at least an explicit threshold to define occurrence of a spike and implements spike-frequency adaptation. One of the early designs meeting these requirements was proposed by Schultz and Jabri in [282] (1995) which can set an explicit threshold voltage and can realize spike-frequency adaptation. Because these spiking neuron models are capable of generating potentially varied functionalities, their detailed hardware realizations naturally tend to consume relatively larger silicon area and power. For example, Rasche and Douglas [253] (2000) describe an analog implementation of Hodgkin-Huxley model with 30 adjustable parameters, which required $4mm^2$ area for a single neuron. Therefore in order to be able to build larger neuromorphic systems using these models, it is necessary that these designs are optimized for area and power requirements. Schaik [276] (2001) presents a circuit design for generating spiking activity. The circuit integrates charge on a capacitor such that when the voltage on the capacitor reaches a certain threshold, two consecutive feedback cycles generate a voltage spike and then bring the capacitor back to its resting voltage. The size of the presented circuit is small enough that it can be used in designing the larger systems on a single chip. Later, Indiveri and Fusi [138] (2007) present a design employing 20 transistors and 3 capacitors for the leaky I&F model with average power consumption in the range of $[0.3\text{–}1.5]\mu$W.

Models by Izhikevich [146] (2003) and Mihalas and Niebur [217] (2009) are one of the recent attempts to define computationally simpler models of a spiking neuron having biological accuracy for spiking and bursting activity. The silicon realization of the Izhikevich's model has been presented in a recent work by Wijekoon and Dudek [315] (2008). However since Izhikevich's model does not land itself directly to parametric biological interpretation, it is bit difficult to integrate in larger neuromorphic designs. Folowosele et al. [91] (2009) on the other hand present the hardware realization of a simplified Mihalas and Niebur's model in terms of switched capacitor circuits fabricated using $0.15um$ CMOS technology, which could be used in larger neuromorphic systems.

There have also been concentrated efforts in modeling and implementing temporal dynamics of synaptic (ionic) current in a biological neuron enabling learning of neural codes and encoding of spatiotemporal spike patterns. Synaptic circuits implementing synaptic dynamics operate by translating presynaptic voltage pulses into postsynaptic currents injected in the membrane of the target neuron, with a gain corresponding to the synaptic weight. Briefly the implementations for the synaptic models can be classified as presented by Bartolozzi and Indiveri in [26] (2007):

- Multiplier Synapse: For models representing synaptic information in terms of mean firing rates, synapse is usually modeled as a multiplier circuit.

- Pulsed Current-Source Synapse: Synapse is implemented in analog form using transistors operating in subthreshold region such that an output pulsed current from the synapse circuit is generated for the duration of the input voltage spike given to it digitally. The underlying model represents synaptic information in terms of mean firing rates. See [214, 95] (1989, 2000).

- Reset-and-Discharge Synapse: Using 3 p-EFT transistors and a capacitor such implementation can give rise to a postsynaptic excitatory current (EPSC), which can last longer then the input voltage spike and decays exponentially with time. See [181] (1994).
- Linear Charge-and-Discharge Synapse: It is a variant of reset-and-discharge synapse, where first the input voltage spike decreases linearly as the postsynaptic excitatory current increases exponentially. After this, input voltage pulse increases to a reference power supply voltage and at the same time postsynaptic current decreases. See [19] (2004).
- Current-Mirror-Integrator Synapse: It is a variant of linear charge-and-discharge synapse where 2 transistors and a capacitor form a current mirror integrator circuit. In contrast to linear charge-and-discharge synapse, postsynaptic excitatory current increases in a sigmoidal fashion and later decreases in a hyperbolic fashion with respect to time. See [139] (2000).
- Log-Domain Integrator (LDI) Synapse: It is another variant of linear charge-and-discharge synapse which utilizes the logarithmic relationship between subthreshold MOSFET gate-to-source voltage and the channel current. The resultant synaptic circuit works like a linear low-pass filter. However the circuit area is relatively larger as compared to other models. See [216] (2004).
- Diff-pair Integrator (DPI) Synapse: Destexhe et al [73] (1998) proposed a macroscopic model for synaptic transmission and the linear summation property of postsynaptic currents, for which Bartolozzi and Indiveri [26] (2007) propose a VLSI synaptic circuit - the diff-pair integrator - that implements this model as a log-domain linear temporal filter and supports synaptic properties including short-term depression to conductance based EPSC generation. The synaptic circuit uses six transistors and a capacitor and effectively works same as low-pass linear filter. However unlike LDI synapse, DPI synapse can give rise to exponential dynamics for both excitatory as well as inhibitory postsynapptic currents.

For further details, reader is suggested to refer to [26] (2007), where authors present an overview and comparative analysis of existing synaptic circuits proposed in the literature, e.g., [131, 19, 18] (2006,2004), including their own DPI circuit.

5 Hardware Neural Network Chips

This section provides an overview of HNNs implemented as chips, also known as neurochips, realizing complete ANN models. These include digital neurochips, analog neurochips, hybrids, neuromorphic implementations, FPGA-based neurochips, RAM based neurochips, and neurochips for neural associative memories. A general-purpose neurochip is capable of implementing more than one neural algorithm for a particular application, while a special-purpose neurochip models a particular neural algorithm for many applications.

Typically an activation block, performing the weight $\times$ input multiplication and their summation, is always on the neurochip, whereas other blocks, involving neuron state, weights, and activation function, may be on/off the chip and some of these functions may even be performed by a host computer. Neuron states and weights can be stored in digital/analog form, and the weights can be loaded statically or updated dynamically.

5.1 Digital Neurochips

The majority of the available digital chips use CMOS technology. There are several categories of digital chips, like bit-slice, single instruction multiple data (SIMD), and systolic arrays. The advantages of digital technology include the use of well-understood fabrication techniques, RAM weight storage, and flexible design. The biggest challenge for designers is the synapse multiplier, which normally is the slowest element in the network processing.

In case of conventional bit-slice architectures, a processor is constructed from modules, each of which processes one bit-field or "slice" of an operand. They provide simple and cheap building blocks (typically single neurons) to construct networks of larger size and precision. An example is Micro Devices' MD1220

Neural Bit Slice [66] (1990), one of the first commercial HNN chips. It has eight neurons with hard-limiting thresholds and eight 16-bit synapses with 1-bit inputs. With bit-serial multipliers in the synapse, the chip provides a performance of about 9 MCPS. Other examples of slice architectures are the Philips' Lneuro chip [211] (1992) and the Neuralogix' NLX-420 Neural Processor[234] (1990). Slice architectures generally include off chip learning.

In case of SIMD, each of the multiple PEs run the same instruction simultaneously, but on different data sets [215] (1991). For a better match with ANN requirements one has to turn to programmable systems, and most such designs are SIMD with minor variations. Instructions are often horizontally encoded, that is, each field of the instruction word directly configures a part of the PE[1]. The two features, *viz.*, no address/issue logic and reduced instruction decoding, render the implementation suitable for the resources required by general ANNs. In Adaptive Solutions' N64000 [113] (1990) with 64 PEs, each PE holds a 9 × 16-bit integer multiplier, a 32-bit accumulator, and 4 KB of on-chip memory for weight storage. Kim et al. [160] (2005) propose a high performance neural network processor based on the SIMD architecture that is optimized for image processing. The proposed processor supports 24 instructions, and consists of 16 Processing Units (PUs) per chip. Each PU includes 24-bit 2K-word Local Memory and one PE.

In case of systolic array based designs, each PE does one step of a calculation synchronously with other PEs and then passes its result to the next processor in the pipeline, thus making the architecture very suitable for implementing efficient synapse multiplier. For example, in Siemens' MA-16 [40] (1993), fast matrix-matrix operations (multiplication, subtraction, or addition) are implemented with 16-bit elements for 4 × 4 matrices. The multiplier outputs and accumulators have 48-bit precision. Weight storage is off-chip RAM and neuron transfer functions are off-chip via look–up tables. Generally systolic arrays are application specific processing arrays for problems displaying a large amount of fine-grained parallelism, and thus they are well matched to ANNs having low data bandwidth and potentially high utilization ratio of the processing units. Their disadvantage lies in the high complexity of the system controlling and interfacing the array with a host system. Some further examples of systolic architectures for HNNs include vector processor arrays [60] (1992), common bus architecture [105] (1994), ring architecture [134] (1995), and TORAN (Two-in-One Ring Array Network) architecture [11] (1999). Eppler et al. [82] (1997) presented a cascadable, systolic processor array called Simple Applicable Neural Device (SAND), designed for fast processing of neural networks. The neurochip may be mapped on feed-forward networks, RBF, and Kohonen feature maps. The chip is optimized for an input data rate of 50 MHz, 16 bit data and could be considered having low cost at the time of its design. The performance of a single SAND chip that uses four parallel 16 bit multipliers and 40 bit adders in one clock cycle is 200 MCPS. In early nineties, Wang proposed an analog recurrent neural network [311] (1992) based on the deterministic annealing network for solving the assignment problem. However, that analog implementation required mapping the massive number of interconnections and programming the parameters. Later Hung and Wang [130] (2003) presented the digital realization of the same by mapping it to a one dimensional systolic array with ring interconnection topology. A scaled down version was realigned using FPGA based devices. Interestingly, they demonstrate that regularities in the data for the assignment problem could be used to eliminate the need of multiplication and devision operations.

Apart from the above, other digital HNN designs also exist. Bagging [38] (1996) is a technique for improving classification performance by creating ensembles. Bagging uses random sampling with replacement from the original data set in order to obtain different training sets. It is observed that bagging significantly improves classifiers that are unstable in the sense that small perturbations in the training data may result in large changes in the generated classifier. Bermak and Martinez [31] (2003) present a 3D circuit implementation of bagging ensembles for efficient pattern recognition tasks. Individual classifiers within the ensemble are decision trees specified as threshold networks having a layer of threshold logic units (TLUs) followed by combinatorial logic elements. The proposed architecture supports a variable precision computation (4/8/16–bit) and configurable network structure w.r.t. number of networks per ensemble or the number of TLUs and inputs per network.

[1]In a horizontally encoded instruction set, each field in an instruction word controls some functional unit or gate directly, as opposed to vertical encoding where instruction fields are decoded (by hard-wired logic or microcode) to produce the control signals. A horizontally encoded instruction allows operation level parallelism by specifying more than one independent operations and thus in a single cycle multiple operations can be performed simultaneously. Because an architecture using horizontal encoding typically requires more instruction word bits it is sometimes known as a very long instruction word (VLIW) architecture [89] (1983). These architectures are especially suitable for HNN implementations.

In self-organizing feature map (SOFM) the capability of calculating the exact equation of the learning rule and the distance required by a PE has a direct bearing on the chip area. In particular, it becomes too large when large number of PEs are to be considered. Rueping et al. [267] (1994) present a digital architecture based on the idea that restriction on the learning algorithm may simplify the implementation. In this architecture the Manhattan Distance and a special treatment of the adaptation factor are used to decrease the necessary chip area so that a high number of PEs can be accommodated on a single chip. The hardware is extendable and advantageous to realize map sizes of 10×10 in one chip with only 28 pins. With binary data, even higher performance (> 25 GCPS for a 50×50 map) can be achieved.

Recently, Dibazar et al. [75] (2006) discuss Texas instrument's TMS320C6713 DSP Starter Kit (a floating point DSP processor) based implementation of a Dynamic Synapse Neural Network model for acoustic sound recognition in noisy environments. The developed hardware achieves an accuracy of 90% for classification and localization task for gunshot recognition.

5.2 Analog Neurochips

Some of early fully developed analog chips include Intel's ETANN and Synaptic's Silicon Retina. Intel's *Electrically Trainable Analog Neural Network* (ETANN) 80170NX [128] (1990) is an elaborate analog chip with 64 fully connected neurons. It is a general-purpose neurochip where analog non-volatile weights are stored on-chip as electrical charge on floating gates, and Gilbert-multiplier synapses provide four-quadrant multiplication. ETANN does not support on-chip learning and only a chip-in-the-loop mode using a host computer is used so that at the end of the learning phase weights could be downloaded on the chip. The chip is reported to achieve a calculation rate of 2 GCPS, accuracy of 4-bits with a 64-bit bus, and 10,240 programmable synapses. ETANN chips can be cascaded to form a network of upto 1024 neurons with upto 81,920 weights, by direct-pin/bus interconnection. The Mod2 Neurocomputer [226] (1992) is an early design employing 12 ETANN chips for real-time image processing. Later many other systems utilized these ETANN chips including MBOX II [44] (1994), an analog audio synthesizer with 8 ETANN chips.

Competition based ANNs such as Kohonen SOFM often need calculating distances between input vectors and the weights. An analog implementation for an SOFM generally results into a compact circuit block that accurately computes the distances. Common measures for calculating the distances include the Euclidean distance function and the Manhattan distance function. Circuits for calculating distance by the method of Euclidean distance function have been presented by Lanbolt et al. [178] (1992) and Churcher et al. [61] (1993). In early 90s, Churcher et al. [61] (1993) presented circuits for calculating Euclidean distance measure. Later, Gopalan and Titus [106] (2003) provide an analog VLSI implementation of a wide range of Euclidean distance computation circuit which can be used as part of a high-density hardware implementation of a SOFM.

Liu et al. [197] (2002) present a mixed signal CMOS feed-forward chip with on-chip error-reduction hardware for real-time adaptation. The chip was fabricated through MOSIS in Orbit 2μm n-well process and weights were stored in capacitors targeting oscillating working conditions. The implemented learning algorithm is a genetic random search algorithm, known as Random Weight Change (RWC) algorithm, which does not require a known desired neural-network output for error calculation and is thus suitable for direct feedback control. In experiments, the RWC chip, as a direct feedback controller, could successfully suppress unstable oscillations modeling combustion engine instability in real time. Nonetheless, volatile weight storage remains an issue limiting the possible applications.

Ortiz and Ocasio [239] (2003), on the other hand, present a discrete analog hardware model for the morphological neural network, which replaces the classical operations of multiplication and addition by addition and maximum or minimum operations.

Milev and Hristov [218] (2003) present an analog-signal synapse model using MOSFETs in a standard 0.35-μm CMOS fabrication process to analyze the effect of the synapse's inherent quadratic nonlinearity on learning convergence and on the optimization of vector direction. The synapse design is then used in a VLSI architecture consisting of 2176 synapses for a finger-print feature extraction application.

Brown et al. [39] (2004) describe the implementation of a signal processing circuit for a Continuous-Time Recurrent Neural Network using subthreshold analog VLSI in mixed-mode (current and voltage) approach, where state variables are represented by voltages while neural signals are conveyed as currents. The use of current allows for the accuracy of the neural signals to be maintained over long distances, making this architecture relatively robust and scalable.

Bayraktaroglu et al. [28] (1999) discuss - ANNSyS - a system for synthesizing analog ANN chips by approximating on-chip training to provide the starting point for 'chip-in-the-loop training'. The synthesis system is based on SPICE circuit simulator and a silicon assembler and designed for analog neural networks to be implemented in MOS technology.

5.3 Hybrid Neurochips

Hybrid Chips combine digital and analog technologies in an attempt to get the best of both. For example, one can use analog internal processing for speed with weights being set digitally. As an example, consider the hybrid Neuro-Classifier from the Mesa Research Institute at University of Twente [210] (1994), which uses 70 analog inputs, 6 hidden nodes, and one analog output with 5-bit digital weights achieving the feed-forward processing rate of 20 GCPS. The final output has no transfer function, so that multiple chips can be added to increase the number of hidden units. Similarly [186] (2004) presents a hardware efficient matrix-vector multiplier architecture for ANNs with digitally stored synapse strengths.

Cortical neurons [77, 4] (1991,1994) whose major mode of operation is analog can compute reliably even with the precision limitation of analog operations owing to their organization into populations in which a signal at each neuron is restored to an appropriate analog value according to some collective strategy. Douglas et al. [78] (1994) describe a hybrid analog-digital CMOS architecture for constructing networks of cortical amplifiers using linear threshold transfer function. Romariz et al. [261] present a hybrid architecture for neural co-processing in which a fixed set of analog multipliers and capacitors (analog memory) emulates multilayer perceptions through digitally-controlled multiplexing, thus preserving parallelism partially without direct analog implementation of the whole structure.

A hybrid architecture with on-chip learning has been presented in [278] (1999). The overall circuit architecture is divided into two main parts with regard to their operating modes, *viz.*, analog and digital. The analog ANN unit executes the neural function processing using a charge based circuit structure. It is composed of a 20 neuron layer, each with 10 bit vector inputs. The *winner-takes-all* unit is devoted to the task of selecting one neuron as the winner on the criterion of the best degree of match between the stored pixel pattern and the current input vector. On the other hand, the units for error correction, circuit control and clock generation are kept purely digital.

5.4 FPGA Based Implementations

Reconfigurable FPGAs provide an effective programmable resource for implementing HNNs allowing different design choices to be evaluated in a very short time. They are low cost, readily available, and reconfigurable offering software like flexibility. Partial and online reconfiguration capabilities in the latest generation of FPGAs offer additional advantages. However the circuit density using FPGAs is still comparably lower and is limiting factors in the implementation of large models with thousands of neurons.

Krips et al. [169] (2002) present an FPGA implementation of a neural network meant for designing a real time hand detection and tracking system applied to video images. Yang and Paindavoine [321] (2003) present an FPGA based hardware implemented on an embedded system with 92% success rates of face tracking and identity verification in video sequences.

Maeda and Tada [206] (2003) describe an FPGA realization of a pulse density NN using the simultaneous perturbation method [205, 204] (1995,1997) as the learning scheme. The simultaneous perturbation method is more amenable to a hardware realization than a gradient type learning rule, since the learning rule requires only forward operations of the network to modify weights unlike the BP present in the gradient type rule. Pulse density NN systems are also robust against noisy conditions.

In contrast to Custom VLSI, the FPGAs are readily available at a reasonable cost and have a reduced hardware development cycle. Moreover, FPGA-based systems can be tailored to specific ANN configurations. For example, Gadea et al. [96] (2000) present the implementation of a systolic array for a multilayer perceptron on a Xilinx Virtex XCV400 FPGA of a pipelined on-line BP learning algorithm. Huitzil and Girau [103] (2006) map the integrate-and-fire LEGION (Local Excitatory Global Inhibitory Oscillator Network) spiking neural model for image segmentation [299, 310] (1997) onto Xilinx Virtex XC2V1500FF896-4 device. However multiplication is bit costly using FPGAs since each synaptic connection in an ANN requires a single multiplier, and this number typically grows as the square of the number of neurons. Mordern FPGAs, e.g., Xilinx' Virtex II Pro [319] (2007) with embedded IBM PowerPC cores and Altera's Stratix III [293] (2007), though can have hundreds of dedicated multipliers.

In a relatively recent work Himavathi et al. [126] (2007) have used layer multiplexing technique to implement multilayer feed-forward networks into Xilinx FPGA XCV400hq240. The suggested layer multiplexing involves implementing only the layer having the largest number of neurons. A separate control block is designed, which appropriately selects the neurons from this layer to emulate the behavior of any other layer and assigns the appropriate inputs, weights, biases, and excitation function for every neuron of the layer that is currently being emulated in parallel. Each single neuron is implemented as a look-up table. To assess the effectiveness of the design a flux estimator for sensorless drives [304] (1998) was used for testing with reported 50% decrese in the number of neurons though adding an speed overhead of 17.7% because of the control block.

Another recent study Rice et al. [258] (2009) reports that a FPGA based implementation of a neocortex inspired cognitive model can provide an average throughput gain of 75 times over software implementation on full Cray XD1 supercomputer. They use the hierarchical Bayesian network model based on the neocortex developed by George and Hawkins [99] (2005). Their hardware-accelerated implementation on the Cray XD1 uses Xilinx Virtex II Pro FPGAs with off–chip SRAM memory and software implementation uses 5 dual core 2.0 GHz Opteron processors.

An important problem faced by designers of FPGA based HNNs is to select the appropriate ANN model for a specific problem to be implemented using optimal hardware resources. Simon Jothson and others provide interesting insights in [154] (2005) for this purpose. They carried out a comparative analysis of hardware requirements for implementing four ANN models onto FPGA. The selected models include MLP with BP and RBF network as classical models, and two SNN models - leaky integrate and fire (LIF) and spike response model. These models were then analyzed on a benchmark classification problem for FPGA hardware resources. The results of the study suggest that LIF SNN model could be the most appropriate choice for implementation for non-linear classification tasks.

FPGA Implementations of Stochastic ANN Models: Practical hardware implementations of large ANNs critically demand that the circuitry devoted to multiplication is significantly reduced. One way to reduce it is to use bit-serial stochastic computing [97] (1969). This uses relatively long, probabilistic bit-streams, where the numeric value is proportional to the density of "1"s in it. For example, a real number $r \in [-1, 1]$ is represented as a binary sequence such that probability of a bit getting set to 1 is $(r+1)/2$. The multiplication of two probabilistic bit-streams can be accomplished by a single two-input logic gate. This makes it feasible to implement large, dense, fully parallel networks with fault tolerance. Even though stochastic computation is simple, it may not always be efficient (see [256] (2002) for comparison.)

Most of the stochastic ANN models have been implemented in hardware using FPGAs [68, 24, 187, 233] (1993,1994,2006,2007). Daalen et al. [68] (1993) describe an FPGA based expandable digital architecture with bit serial stochastic computing to carry out the parallel synaptic calculations. Authors discuss that fully connected multi-layered networks can be implemented with time multiplexing using this architecture. FPGAs have also been used to implement stochastic computation with look–up table based architecture for computing AF [24] (1994). Li et al. [187] (2006) discuss FPGA implementation of a feed forward network employing stochastic techniques for computing the nonlinear sigmoid AFs. Further it is used to design a neural-network based sensorless control of a small wind turbine system. Nedjah and Mourelle [233] (2007) describe and compare the characteristics of two Xilinx VIRTEX-E family based FPGA prototype architectures implementing feed-forward fully connected ANNs with upto 256 neurons. The first prototype used traditional adders and multipliers of binary inputs while the second instead has stochastic representation of the inputs with corresponding stochastic computations. They compare both prototypes in terms of space requirements, network delays, and finally the time $\times$ area factor. As expected, stochastic representation reduces space requirements to a good extent though resulting networks are slightly slower compared to binary models.

5.5 Other Implementations

Szabo et al. [297] (2000) suggest a bit-serial/parallel neural network implementation method for pre-trained networks using bit-serial distributed arithmetic for implementing digital filters. Their implementation of a matrix-vector multiplier is based on an optimization algorithm, which utilizes CSD (Canonic Signed Digit) encoding and bit-level pattern coincidences. The resulting architecture can be realized using FPGA or ASIC and can be integrated into automatic neural network design environments. The suggested matrix multiplier structure is useful for both in MLP designs as well as cellular neural networks (CNNs).

5.5.1 Implementations for Associative Neural Memories

Basic operation of an Associative Neural Memory (ANM) is to map between two (finite) pattern sets using threshold operation. Palm et al. [241] (1993) studied a very simple model of a neural network performing this task efficiently, where the input, output, and connection weights are binary. Ruckert et al. [266, 123] (1991, 2002) thereafter designed VLSI architectures for this model using analog, digital, and mixed signal circuit techniques. Digital architecture is based on a 16-Kbit on-chip static RAM, a neural processing unit, a coding block including input/output logic, and an on-chip controller providing 12 instructions for synchronizing, controlling, and testing the modules. The learning rate estimated to be 0.48 GCUPS. The test chip contains a 16 neuron $\times$ 16 synapse matrix using 1.2-μ CMOS technology. The designed chips can be scaled up, for example, upto 4000 neurons, each having 16,000 inputs. Cascading such chips would further enlarge the design. Willshaw et al. [317] (1969) define a type of ANM model called Correlation Matrix Memory (CMM), where output pattern is a label associated with the most similar stored pattern to the input. Justin et al. [313] (2005) discuss an FPGA based implementation of the pipelined binary-CMM with on-board training and testing for high-performance pattern recognition tasks. For an accessible reference on various ANM models the reader is referred to the edited volume by Hassoun [120] (1993) - Part IV discusses implementations of several ANM models including an optical implementation.

5.5.2 RAM Based Implementations

First introduced by Bledsoe and Browning [32] (1959), RAM based NN (RNN) (also known as weightless NN) [20, 200] (1998,1999) consists of PEs (neurons), which have only binary inputs and outputs and no weight between nodes. Neuronal functions are stored into look-up tables, which can be implemented using commercially available RAMs. Unlike other neural network models, they can be trained very rapidly and can be implemented using simple hardware. Instead of adjusting weights in the conventional sense the RNNs are trained by changing the contents of the look-up tables. RNNs have found applications including as a class of methods for building pattern recognition systems. [20, 200] (1998,1999) provide detailed overview on RNNs.

Aleksander et al. [7] (1984) provide the first hardware realization of a general purpose image recognition system - WISARD, based on RAM circuits. In [62] (1992), hardware implementation of the probabilistic RAM networks is presented, as well as the learning algorithm. Kennedy and Austin [158] (1994) describe a SAT (Sum And Threshold) processor; a dedicated hardware implementation of a binary neural image processor. The SAT processor is specifically aimed at supporting the Advanced Distributed Associative Memory (ADAM) model. ADAM essentially is a two layered binary weighted neural network aimed at recognizing and extracting features from images. Austin et al. further design C-NNAP (Cellular Neural Network Associative Processor) [21] (1995), which is a MIMD array of ADAM based processors to provide a distributed solution to the object recognition problems.

6 Parallel Implementations of Specific ANN Models

In this section we provide an overview of the mapping of some popular ANN models onto parallel hardware architectures. These can also be categorized as special-purpose neurocomputers (ref. Section 9). The neural algorithms considered here comprise BP in multilayer perceptron (MLP) [270] (1986), RBF network [223] (1989). Although most neural learning algorithms typically involve a lot of local computations, the output of an unit usually depends on the output of many other units. Hence, in the absence of a judicious mapping, a parallel implementation can often spend the majority of its runtime in communication instead of actual computation [288] (1990).

The key concepts of an efficient mapping are load balancing, and minimizing inter-PE communication and the synchronization between them. Schoenauer et al. [280] (1998) examine basic strategies to map neural networks on parallel computers. Based upon load balancing, inter-PE communication and synchronization, and scalability, they discuss strategies for mapping with neuron-parallelism, synapse-parallelism, and input pattern parallelism.

6.1 Back-propagation in MLP

There can be several forms of parallelism in the BP algorithm [236] (1992). For *neuron level* parallelism, node-level calculations are viewed as matrix-vector products, and each row of the matrix is mapped onto a processor, while for the *synapse level* parallelism each column of the matrix is mapped onto a processor. When the learning is in batch-mode, a replica of the whole network and a partition of the training set patterns are mapped on each processor. Each replica evaluates partial weight changes, that are then summed. In *layer forward-backward* parallelism, the learning is in batch-mode and the forward and backward phases for different training patterns can be pipelined.

When each processor simulates a single neuron or a connection, the hardware implementation is penalized by the amount of communication required. *Data partitioning* has been implemented on the GF11 [318] (1990), an SIMD parallel computer consisting of 556 processors capable of 20 MFLOPs (mega floating-point operations per second) each. The NETTALK benchmark (203-60-26 with 12022 training patterns) provides 900 MCUPS with 356 processors. A modified version of BP has been implemented on the IPSC hypercube [36] (1989). Each replica is trained i times in batch-mode using randomly selected examples. The weight changes evaluated after i iterations are sent to a master that calculates the mean. The mean changes are transmitted back to nodes that update weights, and the cycle restarts. A speed-up of 28.92 for $i = 32$ and 12.78 for $i = 1$ were obtained with 32 nodes, where speedup for n nodes is defined as the ratio of the time for 1000 iteration on one node and the time for 1000 iterations on n nodes. Although the system allows fast simulations, the user is constrained to batch-learning.

In *network partitioning*, the BP algorithm is viewed as a sequence of matrix-vector products that are usually performed in a systolic fashion. A linear array of cascaded neural chips, HANNIBAL [232] (1995), each with four PEs having local RAM storing 256 16-bit weights, has been used to implement the algorithm in a pipeline. When a row of the weight matrix is mapped on each processor [174, 155] (1989,1994), the elements of the input vector circulate through the processor network. As the i^{th} element of the vector reaches the j^{th} processor, it is multiplied by the $(i, j)^{th}$ element of the matrix and the result is added to a partial sum. On termination, the j^{th} processor holds the j^{th} element of the resulting vector. Since the elements of the vector are transmitted unchanged, the computation can be easily overlapped with the communication. On the other hand, when a column of the weight matrix and an element of the input vector are mapped on each processor, the intermediate sums are moved between processors. In this case, the computation and communication aspects cannot be overlapped [173] (1988). Typically when the sizes of the different layers are not approximately equal, the mapping is not very efficient.

Feed-forward networks with BP learning have been mapped on p-processor hypercube [171] (1994) involving an embedding of $\sqrt{p} \times \sqrt{p}$ processor grid. Nodes of each layer are equally distributed among the diagonal processors, while weights are equally distributed among all processors. This hypercube architecture employs special partitioning scheme called *checker–boarding*, which allows concurrent non interfering communication among processors. The communication aspect has been further explored in the context of vertical slicing (neuron parallelism) [5, 163] (1997,1996). A communication-intensive batch version of the algorithm has been implemented on Transputer arrays [248] (1989) as a sequence of matrix-matrix products [92] (1987). Mapping of BP to various multiprocessor network topologies, like ring, hypercube, binary tree and extended hypercube, have been evaluated [170] (1996). Second-order gradient-based learning has been developed on pipeline and ring configurations [213] (1997).

Acierno [69] (2000) utilizes parallelism at the synaptic, neuronal, and training set levels to map the BP algorithm to SIMD and ring-connected MIMD architectures. Communication involves all-to-one and one-to-all broadcasting. Each processor is assumed to know all the training patterns and handles one hidden neuron. Neuron parallelism is used to evaluate the activation values and error terms of the hidden neurons, while synapse parallelism is used to evaluate the output node activations. The mapping is suitable for classification problems, where output layer is smaller than other layers.

6.2 Radial Basis Function

RBF networks are conceptually much simpler to interpret and use, as compared to MLPs. While RBF networks do not generalize that well, they can be trained very quickly especially if implemented in hardware. The particular basis functions used for simulations can vary, but the signum (*i.e.*, step function) or exponential functional are the favorites for the existing hardware. There are also numerous variations of the training algorithms. IBM's ZISC036 (Zero Instruction Set Computer) chip [132] (1994) is one such example. This chip holds 36 prototypes, and can be easily cascaded to increase the available prototypes. The vectors contain sixty four 8 bit elements, and the output classes can vary from 1 to 16383. The distance norm is either the Manhattan block or the largest difference. This eliminates the need for multipliers, and hence achieves greater speed and reduced circuit complexity. The basis functions are signum with radii ranging from 0 to 16383, while the learning algorithm is modified version of RCE. Maria et al. [209] (1994) discuss how RBF networks can be efficiently implemented on 1D and 2D systolic arrays. Their proposed algorithms has been useful for implementation in the MANTRA machine having 2D grid and the SMART Neuro-computer with 1D ring. A parallel analog processor chip design for a RBF based portable kernel classifier has been discussed by Verleysen et al. in [306] (1994).

A major factor that accounts for the advantages of RBFs or quadratic NNs (QNNs) [191] (1992) over MLPs is related primarily to their closed-boundary discrimination property.Fakhraie et al. [84] (2004) demonstrate the feasibility of the idea of using deep-submicron CMOS technology to implement closed-boundary discriminators and efficient function-approximation building blocks. They showed that all important properties of QNNs in function approximation, pattern classification, and closed-boundary-region forming are maintained in scaled submicron technologies. Yang and Paindavoine [321] (2003) describe a model that allows detecting the presence of faces, to follow them, and to verify their identities in video sequences using a RBF network. It also describes real time application using three hardware implementations on embedded systems based on FPGA, ZISC chips, and DSP TMS320C62. The success rates and processing speeds for images size of 288×352 for face tracking and identity verification were, respectively, 92% and 14 images/s (FPGA), 85% and 25 images/s (ZISC), and 98.2% and 4.8 images/s (DSP).

6.3 Neocognitron

The Neocognitron [93, 199, 286] (1988,1997,2007) has a hierarchical structure oriented toward modeling the human visual system. There are different layers composed of sets of planes, each plane consisting of a group of cells. A two-dimensional image pattern applied at the first layer passes through the successive planes/layers to the output layer, where it is recognized. The model is found to be position and size independent. A parallel implementation of the Neocognitron has been made on a hypercube computer NCUBE [145] (1990). Another approach to implementing incremental reinforcement of synaptic weights in the Neocognitron has used a binary counter for digital storage of variable weights [141] (1990). Here FETs, ideally suited for integrated circuit (IC) implementation, are controlled to provide variable weights. A mapping of the model on a star topology is made in [245] (2001), using a parallel virtual machine with a linear speedup. Parallelism is exploited at the planar level.

7 CNN Implementations

Chua and Yang [59, 58, 56] (1988,1993) introduced CNN as an regular array of locally interconnected analog processing elements, or cells, operating in parallel, whose dynamic behavior is determined by the cell connectivity pattern (neighborhood extent) and a set of configurable parameters. CNN by its very design is a circuit oriented architecture and is conceptually suitable for hardware implementation. After the inception of CNN, their implementations in hardware have attracted substantial interest covering different types of CNN models differing in interaction type (e.g., linear, non linear, dynamic, or delay), modes of operation (e.g., dense time versus discrete time, oscillating type versus dynamic), and grid topology (e.g., planar, polygonal, circular etc.) There exist analog [118, 161] (1992,1995), digitally programmable [273, 272] (1995,1999), hybrid [16] (2005), FPGA [228, 198, 230, 246] (2003,2005,2007), as well as optical [300] (2000) implementations for CNN. CNN implementations can achieve speeds upto several tera flops and are ideal for the applications which require low power consumption, high processing speed, and emergent computation e.g., real-time image processing [115] (2000). We will only briefly cover some of the recent representative implementations here. For further details readers may look into the detailed overview [265] (2003) and monographs [57, 115, 264] (2002,2000).

Rodriguez-Vazquez et al. [260] (2004) discuss ACE16k, a mixed-signal SIMD-CNN ACE (Analogic Cellular Engine) chips as a vision system on chip realizing CNN Universal Machine (CNN-UM) [263] (1993). ACE16k is designed using 0.35μm CMOS technology with 85% analog elements. Its design incorporates several advancements over its predecessor ACE4k chip [192] (2002) including the use of local analog memories and ACE-BUS enabling it to process complex spatio-temporal images in parallel through a 32-bit data bus working at 120 MBPS with peak processing speed of 330 GOPS. The ACE16k chip consists of an array of 128×128 locally connected mixed-signal processing units operating under SIMD mode. Yalcin et al. in [320] (2005) discuss the spatio-temporal pattern formation in ACE16k and Carranza et al. [45] (2005) present design of a programmable stand-alone system ACE16k-DB for real-time vision pre-processing tasks using ACE16k together with Xilinx XC4028XL FPGA. ACE16k chips have been used in commercial **Bi-i** [327] (2005) speed vision system developped by AnaLogic Computers Ltd and MTA-SZTAKI. Also there exist many recent topographic, sensory, and Cellular Wave Architectures and corresponding hardware implementations based upon CNN-UM. Zarandy et al. [167] (2005) present a brief overview of these implementations. An FPGA based emulated-digital CNN-UM implementation using GAPU (Global Analogic Programming Unit) as discussed by Voroshazi et al. [309] (2008) is a recent work in this direction. They discuss design of an extended Falcon architecture using GAPU. Falcon was earlier proposed as a reconfigurable multi-layer FPGA based CNN-UM implementation employing systolic array architecture by Nagy and Szolgay in [229] (2003). In its original design, Falcon could compute result of only one iteration (e.g., only one image in a vedio sequence) so in [309] (2008) a high level embedded control and arithmetic logic block (GAPU) is used which could support several interation together. Actual design of the GAPU employs Xilinx MicroBlaze arcchitecture. The comparitive tests revealed that in comparision to software based implmentation using Intel core2 Duo T7200 processor with optimized C++ code, the FPGA based hardware implementation could acheive 47 times speed up in time.

Arena et al. [16, 17] (2005,2006) discuss design of a CNN-based analog VLSI chip for real-time locomotion control in legged robots. The analog chip core solves the gait generation task whereas digital control modulates the behavior to deal with sensory feedback. An experimental six cell CNN chip is designed using a switched capacitors in CMOS AMS 0.8- μm technology.

Anguita et al. [13] present implementations of fixed-template CNN's with reduced circuit complexity using single-polarity signals reducing the number of transistors required for signal replication and simple current-mode circuits to implement the output pseudo-linear function and application specific network parameter configurations Experimental results for a CCD-CNN chip prototype with a density of 230 cells per mm^2 are also reported.

One of the more recent works include the design of a stochastic bit-stream CNN model by A. Rak et al. [2] (2009), which is implemented using FPGA. Also Ho et al. [127] (2008) suggest design of a CNN simulator using graphics processing unit (GPU) [124] (2008) consisting of high performance parallel graphics accelerators, by parallezing the CNN computations so that they can be executed concurrently.

8 Neuromorphic HNNs

Neuromorphic refers to a circuit that closely emulates the biological neural design. The processing is mostly analog, although outputs can be digital. Examples include Silicon Retina [214] (1989) and Synaptic Touchpad [296]. Another important category of neuromorphic HNNs is Pulse Coupled Neural Networks (PCNNs) [240, 162] (1999). These have been designed after the mammalian visual system, and further implemented in hardware. Like many other NN models, PCNNs can perform image preprocessing, such as edge finding and segmentation. The time series output is invariant to scaling, rotation and translation. A compact architecture for analog CMOS hardware implementation of voltage-mode PCNNs is presented by Ota and Wilamowski in [240] (1999), which shows inherent fault tolerance and high speed compared to its software counterpart.

An important aspect of neuromorphic designs is the address event representation protocol (AER). There has been a considerable effort to create larger neuromorphic neural networks with point-to-point pulse/spike communication between neural assemblies. AER is used to emulate the point-to-point connections for SNNs of considerable size. They are now quite popular in the neuromorphic community. This work was initiated by Mahowald [208] (1994) and Mortara [225] (1994). Over the last years AER has been perfected by Boahen [34] (2004) and a large AER neuromorphic network system in hardware for visual processing has been presented in Serrano-Gotarredona et al. [107] (2005), claimed to be the most complex neuromorphic pulse communication network yet. In a more recent work Bamford et al. [25] (2008) discuss design of a distributed and locally reprogrammable address event receiver, which could allow for arbitrarily large axonal fan-out.

Selective attention is a mechanism used to sequentially select and process only relevant subregions of the input space, while suppressing other irrelevant inputs arriving from other regions. By processing small amounts of sensory information in a serial fashion, rather than attempting to process all the sensory data in parallel, this mechanism overcomes the problem of flooding limited processing capacity systems with sensory inputs. Indiveri [139, 140] (2000,2003) presents a 2-D neuromorphic hardware model called attention chip, which implements a real-time model of selective attention, for sequentially selecting the most salient locations of its inputs space. It is implemented on an analog VLSI chip using spike-based representations for receiving input signals, transmitting output signals and for shifting the selection of the attended input stimulus over time. Experiments were carried out using a 8×8 grid, demonstrating how the chip's bias parameters could be used to impose different behaviors of the system. Also we should add the recent convolusion chip by Serrano Gotarredona et al. [283] (2006), which can implement many classical NN computations, specifically feature-maps.

The silicon retina are an important class of neuromorphic hardware with a potential to have commercial success beyond pure research. The earliest electronic retina was proposed by Fukushima et al. [94] (1970) in 1970 itself and was subsequently integrated onto an ASIC by Mahowald [208] (1994) in early nineties. Besides spatial contrast/derivative retina, later focus has been turned towards temporal contrast/derivative retina [168, 188] (2002,2007). However unlike the spatial contrast retina they do not communicate with their neighbors to attain a collective computation. Recent and relatively popular studies on the design of a neuromorphic model for mammalian retina include those by Boahen's group [324, 325, 326, 33] (2004,2006). In these studies both outer and inner retina were modeled such that outer retina model performs linear band-pass spatiotemporal filtering and inner retina model performs high-pass temporal filtering and can realize non linear temporal frequency adaptation as well as contrast gain control [324] (2004). The presented model was fabricated as actual chip having 90×60 photoreceptor, 3.5×3.3 mm^2 surface area using 0.35μm-CMOS technology [325] (2004). As authors report, the chip has photoreceptor density only 2.5 times sparser that the human cone density. However in contrast to actual mammalian retina, such designed retina chip does not respond at high temporal frequencies (10 Hz and above) [326] (2006).

Another important topic of Neuromorphic hardware are the silicon cochleae. The network aspect is somewhat weak for them, although the sensor nodes do have connections to one neighbour but more in the manner of a processing chain than a network. Initial work in this direction was reported by Lyon and Mead [201] (1988) and J. Lazzaro and Mead [180] (1989). Recent improvements have been reported by Sarpeshkar et al. [274] (1998) and Chan et al. [47] (2007).

Indiveri et al. [137] (2009) present current state of the are in the field of neuromorphic engineering [177] (1998) and discuss the challenges for designing cognitive–neuromorphic systems.

8.1 Spiking Neural Network Hardware

Spiking (or pulsed) ANNs (SNNs), a class of ANNs, model neurons on a level relating more closely to biology and have attracted attention in many bio-sensing areas including image processing applications [194] (2005) and olfactory sensing [166] (2007). They incorporate computation of membrane potentials, synaptic time delays, and dynamical thresholds, in addition to the prevalent synaptic weighting, postsynaptic summation, static threshold, and saturation. A SNN model synchronizes by taking into account the precise timing of spike events. A noteworthy characteristics of SNNs is that they have been proven to be computationally more powerful than classical ANN models with sigmoidal neurons [203] (1997). However, computing large networks of complex neuron models is a computationally expensive task and leads to longer execution delays even with high-performance workstations [149] (1997). Hardware implementations of a single spiking neuron model has been discussed in the Section 4.3. We next consider relatively recent efforts on designing low power compact VLSI architectures for large scale implementations of SNN models.

Schoenauer et al. [279] (2002) present a neuro-processor, called NeuroPipe-Chip, as part of an accelerator board, which approaches real-time computational requirements for SANNs in the order of 10^6 neurons. For a simple SNN benchmark network for image segmentation, the simulation of the accelerator suggested nearly two orders of magnitude faster computation time than a 500 MHz Alpha workstation and a performance comparable to dedicated accelerator architecture consisting of 64 high-performance DSPs. The NeuroPipe-Chip comprising 100 K gate equivalents is fabricated in an Alcatel five-metal layer 0.35-μm digital CMOS technology. To improve the speed of computations weight caches are used to accumulate all weighted spikes occurring in one time slot. To further speed up the performance, the NeuroPipe-Chip design was augmented with additional on-chip inhibition unit, which would apply equally distributed negative potential to a large set of spikes. Floriano et al. [90] (2003) also demonstrate the usefulness of hardware SANNs in designing embedded microcontrollers for autonomous robots which can evolve the ability to move in a small maze without external support. More recently, Bellis et al. [30] (2004) report using an FPGA based implementation of SNN for building collaborative autonomous agents.

Ros et al. [262] (2006) present a HW/SW codesign approach, where the spike response model for a neuron is implemented in hardware and the network model of these neurons and the learning are implemented in software with a support for an incremental transition of the software components into hardware. Neuronal synapses are modeled as input-driven conductances and various stages of the temporal dynamics of the synaptic integration process are executed in parallel. Multiple PEs process different neurons concurrently. Effectiveness of the proposed architecture is tested with a prototype system using FPGA board and a host computer interacting with each other using PCI bus on a real–time visual data with a time resolution of $100\mu s$. Similarly Zou et al. [331] (2006) also present real–time simulation architecture for networks of Hodgkin–Huxley spiking neurons using a mix of analog circuits and a host computer.

Vogelstein et al. in [307] (2007) describe a mixed signal VLSI chip with on–chip learning for emulating larger SNN models. The experimentally designed chip consists of 60×40 array of I&F neuron with reconfigurable synaptic connectivity allowing arbitrary number of synaptic connections to exist between neurons. The synaptic connections are actually implemented using digital RAM enabling reconfiguration of these connections and associated parameters (e.g., conductance value, post synaptic address) on-the-fly. The actual neuron and its membrane dynamics are implemented in an analog VLSI using a conductance based modeling. The chip has an area of $9mm^2$ with 645μW of power consumption on 10 MCUPS activity. The chip was demonstrated to emulate attractor dynamics observed in the neural activity in rat hippocampal "place cells" [328] (1998).

Koickal et al. [165] (2009) present a spike–timing based reconfigurable single chip architecture for neuromorphic designs. The presented architecture uses only one type of event block designed as an analog circuit, which can be configured to model the functionality of a leaky I&F neuron, a summing exponential synapse, a spike time dependent learning window, and for adaptively generating a compensating current at the neuron input so that neuron firing synchronizes with the timing of a target signal. The configurable event block uses a programmable capacitor array designed earlier by the same authors in [164] together with an operational transconductor, and a comparator and occupies an area of $0.03mm^2$.

Spiking models have also received a lot of attention in the context of learning rules. Traditional ANNs process real-numbers that are inspired by average spiking frequency of real neurons. A fully represented spike train from a neuron, however, can potentially convey much more information content. Some neurophysiological experiments investigating synaptic change, i.e. learning, for example, indicate that relative spike timing of single spike pairs influences direction and magnitude of change of synaptic efficacy, i.e., average spiking frequencies are insufficient to describe the learning behavior of real neurons. This was implemented in neuromorphic on-chip learning synapses by Hafliger et al. [110] (1996) and recently advanced by Fusi et al. [95] (2000) and Chicca et al. [55] (2003). The latest publications by these groups [136, 109] (2006) also describe network experiments with those synapses.

Attempts to realize (multiplier-less) SNN models include works of Chen et al. [50, 51] (2006) and of Ghani et al. [101] (2006). A recent article by Maguire et al. [207] (2007) presents a detailed overview of conventional simulation based approaches to implement SNNs and further details various FPGA based implementations of SNNs.

9 Neurocomputers

Neurocomputers [105] (1994) are defined as stand-alone systems with elaborate hardware and software. They are intended for large scale processing applications like high throughput Optical Character Recognition. The Siemens' *Synthesis of Neural Algorithms on a Parallel Systolic Engine* (SYNAPSE) 1 is an early general-purpose neurocomputer [252, 251] (1993,1994), using eight MA-16 systolic array chips in a pipelined array of 16 and controlled by two MC68040 processors. It resides in its own cabinet and communicates via Ethernet to a host workstation. Storage of weights are off–chip. Peak performance is of the order of 3.2 billion multiplications (16x16-bits) and additions (48-bits) per sec., at 40 MHz clock rate with 5.1 MCPS. Several ANNs like BP and Hopfield nets have been mapped onto SYNAPSE 1.

Adaptive Solutions' *Connected Network of Adaptive Processors* (CNAPS) [212] (1991) is a general-purpose digital neurocomputer that uses VME boards in a custom cabinet running from a UNIX host via an ethernet link. N6400 chips can be cascaded as a linear array. Upto two boards with 1-4 chips each, give a total of 512 PEs. Information is broadcast between the blocks in an SIMD mode by two 8-bit buses. Software includes a C-language library, assembler, compiler, and a package of ANN algorithms including BP and Kohonen's SOFM. One of the advantages of the CNAPS architecture is its scalability to the incorporation of additional N6400 chips.

Another approach to dealing with the personal computer is to work with it in partnership using HNN Accelerator cards, which reside in the expansion slots and are used to speedup the neural computations. These are cheaper than fully dedicated neurocomputers and are usually based on HNN chips, but some just use fast DSPs for multiply and accumulate operations. Examples include IBM ZISC ISA and PCI Cards [105] (1994). An California Scientific's CNAPS accelerators run with the popular BrainMaker neural network software using either 4 or 8 chips (16 PE/chip), to give a total of 64 or 128 PEs. Upto 2.27 GCPS speeds can be achieved, depending on transfer rates of particular machines. The Intel MMX Pentiums [142] (2007) have built-in parallel processing. MMX Pentiums have four extra PEs with instruction set to execute in parallel on the individual bytes, words (16 bits), or doublewords (32 bits) contained in vector registers of 64 bits. Each of the PEs can do a 32-bit integer multiplication and addition operations in one clock cycle in parallel.

The Intel SSE (Streaming SIMD Extensions) instruction set for Pentium III processors offer significant extensions of MMX Pentiums with expanded set of SIMD instructions with 32-bit (single precision) floating point support and an additional set of 128-bit vector registers to perform SIMD and FPU operations at the same time. A further SSE2 extension supports double precision floating point arithmetic. SSE4 [143] (2007) is yet another major recent enhancement, adding a dot product instruction and many additional integer and bitwise instructions.

A parallel neurocomputer architecture, based on a configurable neuroprocessor design, is suggested in [294] (1996). The suggested neuroprocessor dynamically adapts its internal parallelism to the required data precision for achieving an optimal utilization of the available hardware resources by encoding a variable number of different data elements in one very long I/O data word.

10 Optical Neural Networks

In this section we provide a brief overview on optical neural networks (ONNs), designed on the principles of optical computing. Optical technology (see [46] (2003)) utilizes the effect of light beam processing that is inherently massively parallel, very fast, and without the side effects of mutual interference. Optical transmission signals can be multiplexed in time, space, and wavelength domains, and optical technologies may overcome the problems inherent in electronics. The results range from the development of special-purpose associative memory systems through various optical devices (*e.g.*, holographic elements for implementing weighted interconnections) to optical neurochips. Optical techniques ideally match with the needs for the realization of a dense network of weighted interconnections.

Optical technology has a number of advantages for making interconnections, specifically with regard to density, capacity and $2D$ programmability. One of the early ONN design using optical vector matrix product processor or crossbar interconnection architecture is discussed in [88, 86] (1987). Similarly a spatial coding method of dealing with input/output patterns as 2D information is used to develop a neural network system with learning capabilities in [144] (1989). However, the lack of efficient optical switches and high capacity erasable optical memories has been the cause of a bottleneck in the growth of ONNs. Typically such optical switch or spatial light modulator is designed as a set of movable mirrors, called a Deformable Mirror Device (DMD), which is inherently difficult to design on large scale.

Hopfield networks are widely used for exemplifying optical implementations. An optical 2D NN has been developed [322] (1990) using a liquid-crystal television (LCTV) and a lenslet array for producing multiple imaging under incoherent illumination. Multilayer feedforward/feedback networks have also been optically implemented with the threshold function getting evaluated electronically [150] (1993) or approximately realized by optical devices [275] (1995). In the second approach, the architecture employs LCTVs to implement the inputs and the weights, while liquid crystal light valves are used to implement the nonlinear threshold [222] (1996).

An example of optical neurocomputer is the Caltech "Holographic Associative Memory" presented in [1] (1987). The goal of the system is to find the best match between an input image and a set of holographic images that represent its memory. Neurons are modeled by non-linear optical switching elements (optical transistors) that are able to change their transmittance properties as the brightness of a light beam changes. Weighted interconnections are modeled by holograms, which are able to record and reconstruct the intensity of light rays. A 1 inch planar hologram, produced on a tiny photographic film, can fully interconnect 10,000 light sources with 10,000 light sensors making 100 million interconnections. The whole system, consisting of a set of lenses and mirrors, a pinhole array, two holograms and an "optical transistor", is realized as an optical loop.

As Lange et al. [179] (1994) discuss, both electronic and optical technology could be useful to solve problems in real time image processing applications. They fabricated an optical neurochip for fast analog multiplication with weight storage elements and on-chip learning capability. The chip can hold upto 128 fully interconnected neurons. They have also developed the "artificial retina chip", a device that can concurrently sense and process images for edge enhancement or for feature extraction. Applications of these optical devices are in the domains of image compression and character recognition. Silveira [287] (2003) presents recent review on various issues and design approaches related to these optoelectronic NN implementations.

Burns et al. [41] presented an experimental implementation of a modest layered network where the input layer is optical and the output layer is electronic. Direct optical input systems use a sensor array for image capture, with subsequent electronic processing. They also presented a simple technique to significantly reduce photo-induced charge leakage of the neuron activations stored dynamically on capacitors.

Lack of effective programmability is one of the major limitation in optical implementations. Burns et al. [41] (1994) describe an optoelectronic design to overcome this limitation using a combination of optics and electronics with high fan-in and temporal multiplexing of the weights. The layered network design consists of electronically controlled optical input layer using spatial light modulation with subsequent electronic processing, though the multiplication of input pattern with interconnection weight was still carried out using software. Their design significantly reduced the photo-induced charge leakage of the neuron activations stored dynamically on capacitors.

Skinner et al. [289] (1994) proposed an optical implementation of a feed-forward ANN using Kerr-type nonlinear optical material which has ultra-fast response time and allows both weighted connections and non-linear neuron processing to be implemented using only thin material layers separated by free space. This layered network can process both forward calculation signals and backward error propagation simultaneously.

Tokes et al. discuss in [300] (2000) an optical CNN device, also known as Programmable Optical Array/Analogic Computer (POAC), which is based on modified Joint Fourier Transform Correlator and Bacteriorhodopsin as a holographic optical memory. Later Moagar-Poladian and Bulinski [219] (2002) presented a type of reconfigurable optical neuron, in which weights can be dynamically changed. Once a weight is set, it is memorized for a period of few days. The optical neuron comprises a photoelectret as the recording medium of the weights and an optical nonlinear crystal with transverse Pockels effect. First described in 1906 by the German physicist Friedrich Pockels, Pockels effect is a linear electrooptical effect, in which, the application of an electric field produces a birefringence which is proportional to the field. To achieve, transverse Pockels effect, the electro-optic crystal is used in the transverse mode, i.e. the optic axis is set perpendicular to the direction of propagation of the light. Shortt et al. [285] (2005) demonstrate a bipolar matrix vector multiplier based optical implementation of the Kak neural network [157] (1994) . For this, the CC4 algorithm was modified on the training phase for implementing N-Parity problem. First proposed by Kak and Tang [298] (1998), CC4 is a corner classification training algorithm for three-layered feed-forward neural networks. A very recent work in this direction include optical implementation of SNN using a thin film of electron-trapping material by Pashaie and Farhat [244] (2007).

Llata and Rivera [175, 269, 176] (2003,2005,2007) have proposed design of vision system based upon a CMOS image sensor and a hybrid optoelectronic hardware architecture called optical broadcast neural network (OBNN). An OBNN processor classifies input patterns using Hamming classification using a set of reference patterns. The input signals are sent in their temporal order to an array of PEs for computing weight updates by means of an global optical broadcasting, thus taking advantage of fast optical communication as well as electronic computational processing. The downside of the architecture is that it is sensitive to rotation, translation, and scaling of the input images. To overcome these limitations, recently in [268] (2008), they extend the design by introducing PCNN preprocessor stage, which converts an 2D input image into a temporal pulsed pattern. These pulses are then applied as inputs to the OBNN processor. The combined system is reported to achieve the rate of 10^4 classifications per second on binary input images of size 128×128 pixels.

Articles by Yu and Uang [323] (2003) and by Ahmed et al. [23] (2004) are interesting reviews on the later advancements in the design and implementation of ONNs. [23] (2004) has additional discussion on the design of a portable POAC and optical template library.

11 Neuro-Fuzzy Hardware

Fuzzy systems are another important class of design models for intelligent systems, which can represent linguistic knowledge and perform reasoning by means of rules. However, in contrast to ANN models, fuzzy systems do not have a mechanism to automatically acquire new rules or adapt the existing ones. In recent years, there has been substantial research to combine these two approaches primarily to overcome the limitations of the ANN models, namely, the inability to process linguistic information and extract the learned knowledge. Such combination has been achieved in two different ways: one by introducing the fuzzification into the neural-network structure (e.g., at neuronal level or in aggregation function), which gives rise to fuzzy neural-networks [237] (2006) and secondly by providing the fuzzy systems with learning ability by means of neural-network algorithms, i.e., neural-network-driven fuzzy reasoning techniques [152, 147] (1993,2000). Neuro Fuzzy systems (NFSs) have found applications in many areas including [151, 250, 85] (1997,2003,2006): adaptive control, real-time embedded control (e.g., CINTIA [53] (1995)), robotics, adaptive signal processing, pattern recognition, and system identification.

Adaptive-Network-based Fuzzy Inference System (ANFIS) [152, 243] (1993,2005) is one of the earliest and very well known example of ANN based fuzzy inference systems. An ANFIS system is a fuzzy inference system whose parameters are trained by means of neural-network training algorithms. This training process adjusts the parameters of the fuzzy system, such as membership functions (MFs), strength of the rules, consequents, etc. The algorithm is composed of a forward pass which is carried out by a least squares estimator (LSE) process, followed by a backward pass which is carried out by a BP algorithm. In a recent work, Echanobe et al. [81] (2008) present a restricted model of ANFIS with reduced complexity to make it amenable for hardware implementation using "Altera" Stratix II (EP2S15) FPGA, achieving efficiency upto 29 M inferences per second.

Generating well-approximated and smooth nonlinear functions for digital neuro-fuzzy hardware (NFH) usually requires relatively large storage or the use of complex circuits that occupy large silicon areas and drastically reduce system operation speed. Here Analog designs offer natural advantage with several existing analog CMOS implementations for NFSs [231, 312] (2003,2006). These implementations primarily vary with the number of fuzzy rules, fabrication technology, power consumption, degree of programmability, I/O interface and delay. See Table 1 in [312] (2006) for relative comparison. Most of these designs use Gaussian membership function. In a recent work, Basterretxea et al. [27] (2007) have argued that smoothness of the nodal functions may affect the general properties and capabilities of NFSs such that smooth, nonlinear AFs and MFs are capable of compressing more information than simple peice-wise linear functions as triangles, trapezoids, step functions, bounded ramps, etc. They conduct extensive experimental studies to determine the effect of non linear activation and membership functions on the performance of ANFIS with BP/RBF neural architectures.

Owing to the increasing speeds in general purpose hardware as well as high degree of flexibility offered by the software, HW/SW codesign approaches have been recently favored for implementing neuro-fuzzy hardware [257, 71, 81] (2003,2008). Reyneri [257] (2003) presents interesting comparative analysis of different approaches for implementation of NFSs concluding that HW/SW codesign can often outperform homogeneous solutions based either on HW or SW. In particular, he suggests that to take full advantage of HW/SW solutions, it would be desirable for all the parts of the adaptive NFS to be integrated in a single chip. Taking this idea further Campo et al. [71, 81] (2008) present a system-on-a-programmable-chip (SOPC) based implementation of an special type of ANFIS architecture termed piecewise multilinear ANFIS with four layers. An SOPC contains a full FPGA board and a core processor on a single chip. An Alteras Excalibur family based SPOC with 32-bit ARM922T processor is used for the implementation. Both off-line learning based high speed parallel architecture and a pipeline architecture for online learning are proposed. It is also important to add the work on unification of common ANN frameworks and Fuzzy systems by Reyneri [255] (1999). Reyneri introduced weighted RBF (or $NF-n$) as a unified paradigm for the NFSs and demonstrated how common ANN and Fuzzy models could be expressed as WRBF ($NF-n$). As stated in [254] (2001) - “NF unification allows to mix together the various paradigms within the same project or system and to implement all of them on similar pieces of HW (or even on a single one)”.

12 Conclusions and Discussion

HNN research and applications have witnessed a slow and incremental progress in last two decades. Even though ANN hardware has been there for more than last two decades, the rapid growth in general purpose hardware (microprocessors, DSPs, etc.) did not let most of these implementations to outperform to the extent of becoming commercially successful. Nonetheless, novel application areas have steadily started appearing, e.g., embedded microcontroller for autonomous robots [90, 259] (2003,2007), autonomous flight control [15, 98] (1999,2006), proposed silicon model of the cerebral cortex - neurogrid [35] (2007) (also see [153] (2007)), and silicon retina [33, 326] (2006). In recent years several special issues dedicated to HNN implementations have been published [271, 193, 12] (1999,2003,2004) as well as a steady stream of Ph.D theses have appeared [37, 301, 104] (2002,2007,2008) indicating the growing interest in the area.

However in spite of the presence of expressive high-level hardware description languages and compilers, efficient neural-hardware designs still demand ingenious ways to optimally use the available resources for achieving high speed and low power dissipation. Judicious mapping of ANN models onto parallel architectures, entailing efficient computation and communication, is thus a key step in any HNN design and there is a need to design tools for automatically translating high level ANN models onto hardware [102, Sec. 5.5] (2006).

As noted in [280] (1998), digital neurohardware tends to be more algorithm specific requiring a good knowledge about algorithms as well as system design that eventually results into a high time-to-market as compared to conventional hardware. In this respect general-purpose hardware seems more user-friendly, offering more flexibility with uniform programming interfaces, and can therefore profit more from advances in technology and architectural revisions. However, many of the applications involving ANNs often demand computational capabilities exceeding of workstations or personal computers available today. For example, a typical real-time image processing task may demand 10 teraflops [2], which is well beyond the current capacities of PCs or workstations today. In such cases neurohardware appears attractive choice and can provide a better cost-to-performance ratio even when compared to supercomputers because many aspects of user friendliness vanish for the supercomputers which are also relatively expensive.

Since currently many ANN applications use networks with less than 10^4 neurons and/or inputs and only need occasional training, software simulation is usually found to be sufficient in such situations. But when ANN algorithms develop to the point where useful things can only be done with $10^6 - 10^8$ of neurons and $10^{10} - 10^{14}$ of synapses between them [76, 70] (2008), high performance neural hardware will become essential for practical operations. It is important to add that such large scale neural network hardware designs might not be a distant reality as is apparent from the recent work of Schemmel et al. on wafer-scale integration of large SNN models [87, 277] (2008).

[2]Assuming the frame-rate of 100 fps, frame size of 1280×1024 pixels with 3 bytes per pixel, and average number of basic imaging operation having computational complexity of $\mathcal{O}(N)$, (N is the frame-size) with 10^5 such operations to be performed on each frame.

It is observed that presently it is not always possible to exploit the entire parallelism inherent in the ANN topology along with a good cost-performance ratio, mainly due to the cost associated with the implementation of the numerous interconnections, control and mapping involved. In this scenario, optical implementations add a different dimension. Multi-Chip Modules or Wafer-Scale Integration hold further promise for implementing such large networks. IBM cell processor [156] (2005) with nine processor cores or its recent variant QS22 PowerXCell 8i [133] (2008) with their powerful vector processing capabilities hold good promise for highly parallel large scale ANN implementations or their fast emulations for comparative analysis. Also using 3D VLSI packaging technology [6] (1998), large number of synaptic connection could possibly be realized in small space. 3D VLSI classifier [31] (2003) as discussed before in Section 5.1 is an example at hand.

CMOS/nanowire/nanodevice ("CMOL") technology [190, 189] (2005,2008), which combines both CMOS and nanotechnology, is one of the important emerging technologies with high potential for large scale HNN implementations. The basic idea of CMOL circuits is to combine the advantages of CMOS technology including its flexibility and high fabrication yield with the high potential density of molecular-scale two-terminal nanodevices. Neuromorphic Mixed-Signal CMOL Circuits (known as "CrossNets") [302, 183, 184, 185, 182] (2005,2006,2007) are the first results of an active research by K. Likharev's Nanotechnology Research Group at Stony Brook University. In a "CrossNet", CMOS subsystem realizes the neuron core, whereas crossbar nanowires play the roles of axons and dendrites (connections), and crosspoint latching switches serve as elementary (binary-weight) synapses enabling very high cell connectivity (e.g., 10^4) in quasi-2D electronic circuits.

Molecular technology is another relatively new approach for possible hardware implementation. It combines protein engineering, biosensors, and polymer chemistry in the efforts to develop a molecular computer [63] (1988). The computation uses the physical recognition ability of large molecules, like proteins, which can change their shape depending on the chemical interactions with other molecules. Molecular computing is still in its infancy. The major problem is to develop appropriate technology that would allow for construction of bio equivalents of transistors. However inherently parallel generalization and adaptation capabilities perfectly match the needs of neural networks implementations. Research in this direction appears promising [64, 117] (2000) and, in the future, molecular computers with neural architectures appear to have a potential to become a reality. In a recent work Alibart et al. [8] report feasibility of designing a hybrid nanoparticle-organic device, a nanoparticle organic memory FET, which uses the nanoscale capacitance of the nanoparticles and the transconductance gain of the organic transistor to mimic the short-term plasticity of a biological synapse. Dan Ventura [305] (2001) presents an interesting discussion on the possibility designing quantum neural computing devices utilizing quantum entanglement effects.

HNN models hopefully will have the respected place in coming years when industry will face demands imposed by ubiquitous computing with learning and autonomous decision making capabilities e.g., autonomous robotics and assistive technologies. These applications demand dealing with large amounts of real-time multimedia data from interacting environment, using lightweight hardware with strict power constraints, without letting the computational efficiencies go down. The DARPA initiative [70] (2008) - 'Systems of Neuromorphic Adaptive Plastic Scalable Electronics' - towards building cognitive-neuromorphic systems [137] (2009) is indicative of such emerging directions.

References

[1] Y. S. AbuMostafa and D. Psaltis. Optical neural computers. *Scientific American*, 255:88–95, 1987.

[2] Gyorgy Cserey Adam Rak, Balazs Gergely Soos. Stochastic bitstream-based CNN and its implementation on FPGA. *International Journal of Circuit Theory and Applications*, 37(4):587–612, 2002.

[3] A. J. Agranat, C. F. Neugebauer, and A. Yariv. A CCD based neural network integrated circuit with 64k analogprogrammable synapses. In *International Joint Conference on Neural Networks (IJCNN)*, pages 551–555, 1990.

[4] B. Ahmed, J. C. Anderson, R. J. Douglas, K. A. Martin, and C. Nelson. Polyneuronal innervation of spiny stellate neurons in cat visual cortex. *Comparative Neurology*, 341(1):39–49, 1994.

[5] E. Ahmed and K. Priyalal. Algorithmic mapping of feedforward neural networks onto multiple bus systems. *IEEE Transactions on Parallel and Distributed Systems*, 8:130–136, 1997.

[6] SF Al-Sarawi, D. Abbott, and PD Franzon. A review of 3-D packaging technology. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging*, 21(1):2–14, 1998.

[7] I. Aleksander, W. V. Thomas, and P. A. Bowden. WISARD: a radical step forward in image recognition. *Sensor Review*, 4(3):120–124, 1984.

[8] F. Alibart, S. Pleutin, D. Guérin, C. Novembre, S. Lenfant, K. Lmimouni, C. Gamrat, and D. Vuillaume. An Organic Nanoparticle Transistor Behaving as a Biological Spiking Synapse. *Advanced Functional Materials*, 20(2):330–337, 2009.

[9] A. P. Almeida and J. E. Franca. Digitally programmable analog building blocks for the implementation of artificial neural networks. *IEEE Transactions on Neural Networks*, 7(2):506–514, 1996.

[10] H. Amin, K. M. Curtis, and B. R. Hayes-Gill. Piecewise linear approximation applied to nonlinear function of a neural network. In *IEE Proceedings of Circuits, Devices and Systems*, pages 313–317, 1997.

[11] H. Amin, K. M. Curtis, and B. R. Hayes-Gill. Two-ring systolic array network for artificial neural networks. In *IEE Proceedings of Circuits, Devices and Systems*, pages 225–230, 1999.

[12] D. Anguita, I. Baturone, and J. Miller, editors. *Special issue on hardware implementations of soft computing techniques: Applied Soft Computing*, volume 4(3), Aug 2004.

[13] M. Anguita, FJ Pelayo, I. Rojas, and A. Prieto. Area efficient implementations of fixed-template CNN's. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 45(9):968–973, 1998.

[14] A.J. Annema, K. Hoen, and H. Wallinga. Precision requirements for single-layer feed forward neural networks. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 145 – 151, Turin, Italy, 1994.

[15] Annon. The intelligent flight control: Advanced concept program final report. Technical report, The Boeing Company, 1999.

[16] P. Arena, L. Fortuna, M. Frasca, and L. Patane. A CNN-based chip for robot locomotion control. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(9):1862–1871, 2005.

[17] P. Arena, L. Fortuna, M. Frasca, L. Patane, and M. Pollino. An autonomous mini-hexapod robot controlled through a CNN-based CPG VLSI chip. In *10th International Workshop on Cellular Neural Networks and Their Applications*, pages 1–6, 2006.

[18] J. Arthur and K. Boahen. Learning in silicon: Timing is everything. *Advances in neural information processing systems*, 18:75, 2006.

[19] JV Arthur and K. Boahen. Recurrently connected silicon neurons with active dendrites for one-shot learning. In *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, volume 3, pages 1699–1704, 2004.

[20] J. Austin, editor. *RAM-Based Neural Networks.* World Scientific, Farrer Road, Singapore, 1998.

[21] J. Austin, S. Buckle, J. Kennedy, A. Moulds, R. Pack, and A. Tumer. The Cellular Neural Network Associative Processor (C-NNAP). *Associative processing and processors*, pages 284–299, 1997.

[22] I. Aybay, S. Cetinkaya, and U. Halici. Classification of neural network hardware. *Neural Network World*, 6(1):11–29, 1996.

[23] Ahmed Ayoub, Szabolcs Tks, and Lszl Orz. Evolution of the programmable optical array computer (POAC). In *Proceedings of IEEE International Workshop on Cellular Neural Networks and their Applications*, pages 64–69, Budapest, Hungary, July 2004.

[24] S. L. Bade and B. L. Hutchings. FPGA-based stochastic neural networks - implementation. In *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines Workshop*, pages 189–198, Napa, CA, April 1994.

[25] S.A. Bamford, A.F. Murray, and D.J. Willshaw. Large developing axonal arbors using a distributed and locally-reprogrammable address-event receiver. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1464–1471, 2008.

[26] C. Bartolozzi and G. Indiveri. Synaptic dynamics in analog VLSI. *Neural Computation*, 19(10):2581–2603, 2007.

[27] Koldo Basterretxea, José Manuel Tarela, Inés del Campo, and G. Bosque. An experimental study on nonlinear function computation for neural/fuzzy hardware design. *IEEE Transactions on Neural Networks*, 18(1):266–283, 2007.

[28] S Bayraktaroglu, A. S. Ogrenci, G. Dundar, S. Balkr, and E. Alpaydin. ANNSyS: An analog neural network synthesis system. *Neural Networks*, 12:325–338, 1999.

[29] Valeriu Beiu. How to build VLSI-efficient neural chips. In E. Alpaydin, editor, *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems*, pages 66 – 75, Tenerife, Spain, 1998.

[30] S. Bellis, K.M. Razeeb, C. Saha, K. Delaney, C. Mathuna, A. Pounds-Cornish, G. de Souza, M. Colley, H. Hagras, G. Clarke, V. Callaghan, C. Argyropoulos, C. Karistianos, and G. Nikiforidis. FPGA implementation of spiking neural networks - an initial step towards building tangible collaborative autonomous agents. In *Proceedings of the IEEE International Conference on Field-Programmable Technology*, pages 449–452, 2004.

[31] Amine Bermak and Dominique Martinez. A compact 3-D VLSI classifier using bagging threshold network ensembles. *IEEE Transactions on Neural Networks*, 14(5):1097–1109, 2003.

[32] W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Proceedings of Eastern Joint Computer Conference*, volume II, pages 225–232, Boston, 1959.

[33] K. Boahen. Neuromorphic microchips. *Special Editions*, 16(3):20–27, 2006.

[34] Kwabena Boahen. A burst-mode word-serial address-event channel-i: Transmitter design. *IEEE Transactions on Circuits and Systems I*, 51(7):1269–1280, July 2004.

[35] Kwabena Boahen. Neurogrid: Emulating a million neurons in the cortex. In *Grand Challenges in Neural Computation*, page 6702, 2006.

[36] J. Bourrely. Parallelization of a neural learning algorithm on a hypercube. In F. Andre and J. P. Verjus, editors, *Hypercube and Distributed Computers*, pages 219–229, North-Holland, 1989. Elsevier Science B. V.

[37] D. Braendler. *Implementing Neural Hardware with On Chip Training on Field Programmable Gate Arrays*. PhD thesis, Swinburne University of Technology, Melbourne, Australia, 2002.

[38] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[39] B.E. Brown, X. Yu, and S.L. Garverick. Mixed-mode analog VLSI continuous-time recurrent neural network. In *Proceedings of International Conference on Circuits, Signals and Systems*, pages 104 – 108, 2004.

[40] N. Bruels. MA16 - programmable VLSI array processor for neuronal networks and matrix-based signal processing, user description. Technical Report 1.3, Siemens AG, Corporate Research and Development Division, Munich, Germany, October 1993.

[41] D. C. Burns, I. Underwood, A. F. Murray, and D. G. Vass. An optoelectronic neural network with temporally multiplexed grey-scale weights. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 3–7, 1994.

[42] James B. Burr. Energy, capacity, and technology scaling in digital VLSI neural networks. NIPS'91 VLSI Workshop, May 1991.

[43] James B. Burr. Digital neurochip design. In K. Wojtek Przytula and Viktor K. Prasanna, editors, *Parallel Digital Implementations of Neural Networks*, pages 223–281. Prentice Hall, Upper Saddle River, NJ, USA, 1992.

[44] Camalie. Box 2 - an analog audio synthesizer: Architecture and procedures guide, 1994.

[45] L. Carranza, F. Jimenez-Garrido, G. Linan-Cembrano, E. Roca, S.E. Meana, and A. Rodriguez-Vazquez. ACE16k based stand-alone system for real-time pre-processing tasks. In *Proceedings of SPIE*, volume 5837, page 872, 2005.

[46] H. John Caulfield. Optical computing. In Ronald G. Driggers, editor, *Encyclopedia of Optical Engineering 1:1*, pages 1613–1620. CRC Press, 270 Madison Avenue, New York, NY, USA, 2003.

[47] V. Chan, S. Liu, and A. van Schaik. AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems I*, 54(1):48–59, 2007.

[48] H. Chen and A. Murray. A Continuous Restricted Boltzmann Machine with a hardware-amenable learning algorithm. In *Proceedings of the International Conference on Artificial Neural Networks*, Lecture notes in computer science, pages 358–363. Springer, 2002.

[49] H. Chen and A.F. Murray. Continuous restricted boltzmann machine with an implementable training algorithm. In *IEE Proceedings - Vision Image and Signal Processing*, volume 150, pages 153–159. The Institution of Electrical Engineers., 2003.

[50] Y. Chen, S. Hall, L. McDaid, O. Buiu, and P. Kelly. On the design of a low power compact spiking neuron cell based on charge coupled synapses. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1511–1517, Vancouver, Canada, 2006.

[51] Y. Chen, S. Hall, L. McDaid, O. Buiu, and P. Kelly. A silicon synapse based on a charge transfer device for spiking neural network applications. In *Proceedings of the 3rd International Symposium on Neural Networks(ISNN)*, pages 1366–1373, Chengdu, China, 2006.

[52] Z. Chen, S. Haykin, and S. Becker. Theory of monte carlo sampling-based alopex algorithms for neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 17 – 21, 2004.

[53] Marcello Chiaberge and Leonardo M. Reyneri. Cintia: A neuro-fuzzy real-time controller for low-power embedded systems. *IEEE Micro*, 15(3):40–47, 1995.

[54] A. M. Chiang and et al. A CCD programmable image processor and its neural network applications. *IEEE Journal of Solid State Circuits*, 26:1894–1901, 1991.

[55] E. Chicca, G. Indiveri, and R. Douglas. An adaptive silicon synapse. In *Proceeding of the IEEE International Symposium on Circuits and Systems (ISCAS)*, volume V, pages 81–84, Bangkok, 2003.

[56] L. O. Chua and T. Roska. The CNN paradigm. *IEEE Transactions on Circuits and SystemsI: Fundamental Theory and Applications*, 40:147–156, 1993.

[57] L.O. Chua and T. Roska. *Cellular Neural Networks And Visual Computing: Foundation And Applications.* Cambridge University Press, Trumpington Street, Cambridge, UK, 2002.

[58] LO Chua and L. Yang. Cellular neural networks: Applications. *IEEE Transactions on Circuits and Systems*, 35(10):1273–1290, 1988.

[59] LO Chua and L. Yang. Cellular neural networks: Theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272, 1988.

[60] Jai Hoon Chung, Hyunsoo Yoon, and Seung Ryoul Maeng. A systolic array exploiting the inherent parallelisms of artificial neural networks. *Microprocess. Microprogram.*, 33(3):145–159, 1992.

[61] S. Churcher, A. F. Murray, and H. M. Reekie. Programmable analogue VLSI for radial basis function networks. *Electronic Letters*, 29:1603–1605, 1993.

[62] T. G. Clarkson, C. K. Ng, D. Gorse, and J. G. Taylor. Learning probabilistic RAM nets using VLSI structures. *IEEE Transactions on Computers*, 41(12):1552–1561, 1992.

[63] M. Conrad. The lure of molecular computing. *IEEE Spectrum*, 23:55–60, 1988.

[64] M. Conrad and K.-P. Zauner. Molecular computing with artificial neurons. *Communications of the Korea Information Science Society*, 18(8):78–89, 2000.

[65] T. Cornu and P. Ienne. Performance of digital neuro-computers. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 87–93, 1994.

[66] Micro Devices Corp. MD1220 neural bit slice, data sheet, March 1990. Lake Mary.

[67] M. Daalen, T. Kosel, P. Jeavons, and J. Shawe-Taylor. Emergent activation functions from a stochastic bit stream neuron. *Electronic Letters*, 30(4):331–333, February 1994.

[68] Max V. Daalen, Peter Jeavons, and John Shawe-Taylor. A stochastic neural architecture that exploits dynamically reconfigurable FPGAs. In Duncan A. Buell and Kenneth L. Pocek, editors, *IEEE Workshop on FPGAs for Custom Computing Machines*, pages 202–211, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[69] A. d'Acierno. Back-propagation learning algorithm and parallel computers: The CLEPSYDRA mapping scheme. *Neurocomputing*, 31:67–85, 2000.

[70] DARPA. Systems of Neuromorphic Adaptive Plastic Scalable Electronics (SyNAPSE). *http://www.darpa.mil/dso/solicitations/BAA08-28.pdf (Last Accessed on Jan 20, 2010)*, 2008.

[71] I. del Campo, J. Echanobe, G. Bosque, and J. M. Tarela. Efficient hardware/software implementation of an adaptive neuro-fuzzy system. *IEEE Transactions on Fuzzy Systems*, 16(3):761–778, June 2008.

[72] B. Denby. The use of neural networks in high energy physics. *Neural Computation*, 5:505–549, 1993.

[73] A. Destexhe, Z.F. Mainen, and T.J. Sejnowski. Kinetic models of synaptic transmission. *Methods in neuronal modeling*, pages 1–25, 1998.

[74] Fernando Morgado Diasa, Ana Antunesa, and Alexandre Manuel Motab. Artificial neural networks: a review of commercial hardware. *Engineering Applications of Artificial Intelligence*, 17:945–952, 2004.

[75] Alireza A. Dibazar, Abhijith Bangalore, Hyung O. Park, Sageev T. George, Walter M. Yamada, and Theodore W. Berger. Hardware implementation of dynamic synapse neural networks for acoustic sound recognition. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2015 – 2022, Vancouver, BC, Canada, 2006.

[76] M. Djurfeldt, M. Lundqvist, C. Johansson, M. Rehn, ö Ekeberg, and A. Lansner. Brain-scale simulation of the neocortex on the IBM Blue Gene/L supercomputer. *IBM Journal of Research and Development*, 52(1):31–41, 2008.

[77] R. Douglas and K. Martin. A functional microcircuit for cat visual cortex. *Phisiology*, 440:735–769, 1991.

[78] R. J. Douglas, M. A. Mahowald, and K. A. C. Martin. Hybrid analog-digital architectures for neuromorphic systems. In *International Conference on Neural Networks*, pages 1848–1853, 1994.

[79] Tuan A. Duong. Cascade error projection: an efficient hardware learning algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 175–178, Perth, Australia, 1995.

[80] Tuan A. Duong and Allen R. Stubberud. Convergence analysis of cascade error projection: An efficient hardware learning algorithm. *International Journal of Neural System*, 10(3):199–210, 2000.

[81] J. Echanobe, I. del Campo, and G. Bosque. An adaptive neuro-fuzzy system for efficient implementations. *Information Sciences*, 178(9):2150–2162, 2008.

[82] W. Eppler, T. Fischer, H. Gemmeke, T. Becher, and G. Kock. High speed neural network chip on PCI-board, 1997.

[83] Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in neural information processing systems 2*, pages 524–532, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[84] S. M. Fakhraie, H. Farshbaf, and K. C. Smith. Scalable closed-boundary analog neural networks. *IEEE Transactions on Neural Networks*, 15:492 – 504, 2004.

[85] Arash Fanaei and Mohammad Farrokhi. Robust adaptive neuro-fuzzy controller for hybrid position/force control of robot manipulators in contact with unknown environment. *J. Intell. Fuzzy Syst.*, 17(2):125–144, 2006.

[86] N. H. Farhat. Optoelectronic analogs of self-programming neural nets: Architectures and methodologies for implementing fast stochastic learning by simulated annealing. *Applied Optics*, 26:5093–5103, 1987.

[87] J. Fieres, J. Schemmel, and K. Meier. Realizing biological spiking network models in a configurable wafer-scale hardware system. In *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 969–976. IEEE World Congress on Computational Intelligence, 2008.

[88] A. D. Fisher, W. L. Lippincott, and J. N. Lee. Optical implementations of associative networks with versatile adaptive learning capabilities. *Applied Optics*, 26:5039–5052, 1987.

[89] Joseph A. Fisher. Very long instruction word architectures and the ELI-512. In *Proceedings of the 10th annual international symposium on Computer architecture*, pages 140–150, New York, NY, USA, 1983. ACM.

[90] Dario Floreano, Nicolas Schoeni, Gilles Caprari, and Jesper Blynel. Evolutionary bits'n'spikes. In *Proceedings of the eighth international conference on Artificial life*, pages 335–344, Cambridge, MA, USA, 2003. MIT Press.

[91] F. Folowosele, A. Harrison, A. Cassidy, A.G. Andreou, R. Etienne-Cummings, S. Mihalas, E. Niebur, and T.J. Hamilton. A switched capacitor implementation of the generalized linear integrate-and-fire neuron. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2009.

[92] G. Fox, S. Otto, and A. Hey. Matrix algorithm on a hypercube matrix multiplication. *Parallel Computing*, 4:17–31, 1987.

[93] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1:119–130, 1988.

[94] K. Fukushima, Y. Yamaguchi, M. Yasuda, and S. Nagata. An electronic model of the retina. *Proceedings of the IEEE*, 58(12):1950–1951, 1970.

[95] S. Fusi, M. Annunziato, D. Badoni, A. Salamon, and D.J.Amit. Spike-driven synaptic plasticity: theory, simulation, VLSI implementation. *Neural Computation*, 12:2227–2258, 2000.

[96] R. Gadea, J. Cerda, F. Ballester, and A. Macholi. Artificial neural network implementation on a single FPGA of a pipelined on-line back-propagation. In *Proceedings of the 13th International symposium on system synthesis*, pages 225–230, 2000.

[97] B. R. Gaines. Stochastic computing systems. *Advances in Information Systems Science*, 2:37–172, 1969.

[98] Wang Geng, Sheng Huanye, and Lu Tiansheng. Development of an embedded intelligent flight control system for the autonomously flying unmanned helicopter sky-explorer. In *Embedded Systems Modeling, Technology, and Applications*, pages 121–130. Springer Netherlands, 2006.

[99] D. George and J. Hawkins. A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 3, pages 1812–1817, 2005.

[100] W. Gerstner. *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press, Trumpington Street, Cambridge, UK, 2002.

[101] A. Ghani, T. Martin McGinnity, and L.P. Maguire. Area efficient architecture for large scale implementation of biologically plausible spiking neural networks on reconfigurable hardware. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 1–2, Madrid, Spain, 2006.

[102] Bernard Girau. Activity report - neuromimetic intelligence. Technical report, INRIA, Project Team CORTEX, 2006.

[103] Bernard Girau and Cesar Torres-Huitzil. Massively distributed digital implementation of an integrate-and-fire legion network for visual scene segmentation. *Neurocomputing*, 70(7-9):1186–1197, 2007.

[104] Massimiliano Giulioni. *Networks of spiking neurons and plastic synapses: implementation and control.* PhD thesis, Universit degli Studi di Roma 'La Sapienza', Italy, 2008.

[105] M. Glesner and W. Poechmueller. *Neurocomputers: An Overview of Neural Networks in VLSI.* Chapman and Hall, London, 1994.

[106] A. Gopalan and A. H. Titus. A new wide range euclidean distance circuit for neural network hardware implementations. *IEEE Transactions on Neural Networks*, 14:1176–1186, 2003.

[107] Rafael Serrano Gotarredona, Matthias Oster, Patrick Lichtsteiner, Alejandro Linares Barranco, Rafael Paz Vicente, Francisco Gamez Rodraguez, Havard Kolle Riis, Tobi Delbrck, and Shih Chii Liu. AER building blocks for multi-layer multi-chip neuromorphic vision systems. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1217–1224. MIT Press, Cambridge, MA, 2006.

[108] H P Graf, L D Jackel, R E Howard, B Straughn, J S Denker, W Hubbard, D M Tennant, and D Schwartz. VLSI implementation of a neural network memory with several hundreds of neurons. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 182–187, Woodbury, NY, USA, 1987. American Institute of Physics Inc.

[109] P. Häfliger. Adaptive WTA with an analog VLSI neuromorphic learning chip. *IEEE Transactions on Neural Networks*, 18(2):551–572, 2007.

[110] P. Häfliger, M. Mahowald, and L. Watts. A spike based learning neuron in analog VLSI. *Advances in neural information processing systems*, 9:692–698, 1996.

[111] NH Hamid, AF Murray, D. Laurenson, S. Roy, and B. Cheng. Probabilistic computing with future deep sub-micrometer devices: a modeling approach. In *IEEE International Symposium on Circuits and Systems*, pages 2510–2513, 2005.

[112] D. Hammerstrom and R. Waser. A Survey of Bio-Inspired and Other Alternative Architectures. *Nanotechnology: Information Technology-II*, 4:251–285, 2008.

[113] Dan Hammerstrom. A VLSI architecture for high-performance, low-cost, on-chip learning. In *Proceedings of the International Joint Conference on Neural Networks*, pages 537 – 544, San Diego, Ca, 1990.

[114] Dan Hammerstrom. Digital VLSI for neural networks. *The handbook of brain theory and neural networks*, pages 304–309, 2003.

[115] M. Hänggi and G.S. Moschytz. *Cellular Neural Networks: Analysis, Design, and Optimization.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[116] L.K. Hansen, L. N. Andersen, U. Kjems, and J. Larsen. Revisiting boltzmann learning: parameter estimation in markov random fields. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages 3394–3397, 1996.

[117] D. Haronian and A. Lewis. Elements of a unique bacteriorhodopsin neural network architecture. *International Journal of Neural Systems*, 10(3):191–197, 2000.

[118] H. Harrer, JA Nossek, and R. Stelzl. An analog implementation of discrete-time cellular neural networks. *IEEE Transactions on Neural Networks*, 3(3):466–476, 1992.

[119] Reid R. Harrison. A low-power analog VLSI visual collision detector. In Sebastian Thrun, Lawrence Saul, and Bernhard Scholkopf, editors, *Advances in Neural Information Processing Systems 16.* MIT Press, Cambridge, MA, 2004.

[120] Mohamad H. Hassoun, editor. *Associative Neural Memories.* Oxford University Press, Inc., New York, NY, USA, 1993.

[121] S. Haykin. *Neural Networks: a Comprehensive Foundation.* Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 2008.

[122] J. Heemskerk. Overview of neural hardware, 1995. In: Neurocomputers for Brain-Style Processing. Design, Implementation and Application.

[123] A. Heittmann and U. Ruckert. Mixed mode VLSI implementation of a neural associative memory. *Analog Integrated Circuits and Signal Processing*, 30(2):159–172, 2002.

[124] M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming.* Morgan Kaufmann, San Francisco, CA, USA, 2008.

[125] Hiroomi Hikawa. A digital hardware pulse-mode neuron with piecewise linear activation function. *IEEE Transactions on Neural Networks*, 14(5):1028– 1037, 2003.

[126] S. Himavathi, D. Anitha, and A. Muthuramalingam. Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization. *IEEE Transactions on Neural Networks*, 18(3):880–888, 2007.

[127] T.Y. Ho, P.M. Lam, and C.S. Leung. Parallelization of cellular neural networks on GPU. *Pattern Recognition*, 41(8):2684–2692, 2008.

[128] M. Holler, S. Tam, H. Castro, and R. Benson. An Electrically Trainable Artificial Neural Network (ETANN) with 10240 “Floating Gate” Synapses,. *IEEE Computer Society Neural Networks Technology Series*, pages 50–55, 1990.

[129] Jordan Holt and Jeng-Neng Hwang. Finite precision error analysis of neural network hardware implementations. *IEEE Transactions on Computers*, 42(3):281–290, 1993.

[130] Donald L. Hung and Jun Wang. Digital hardware realization of a recurrent neural network for solving the assignment problem. *Neurocomputing*, 51:447–461, April 2003.

[131] KM Hynna and K. Boahen. Neuronal ion-channel dynamics in silicon. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3614–3617, 2006.

[132] IBM. Ibm microelectronics ZISC zero instruction set computer: Preliminary information. In *Supplement to Proceedings of World Congress on Neural Networks*, pages 31 – 39, San Diego, CA, June 1994.

[133] IBM. IBM Cell Broadband Engine technology, 2008.

[134] P. Ienne. Digital hardware architectures for neural networks. *Speedup Journal*, 9(1):18–25, 1995.

[135] Paolo Ienne, Thierry Cornu, and Gary Kuhn. Special-purpose digital hardware for neural networks: an architectural survey. *J. VLSI Signal Process. Syst.*, 13(1):5–25, 1996.

[136] G. Indiveri, E. Chicca, and R. Douglas. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Tran. on Neural Networks*, 17:211–221, Jan. 2006.

[137] G. Indiveri, E. Chicca, and R.J. Douglas. Artificial Cognitive Systems: From VLSI Networks of Spiking Neurons to Neuromorphic Cognition. *Cognitive Computation*, 1(2):119–127, 2009.

[138] G. Indiveri and S. Fusi. Spike-based learning in VLSI networks of integrate-and-fire neurons. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2007, pages 3371–3374, 2007.

[139] Giacomo Indiveri. Modeling selective attention using a neuromorphic analog VLSI device. *Neural computation*, 12(12):2857–2880, 2000.

[140] Giacomo Indiveri. A neuromorphic VLSI device for implementing 2-D selective attention systems. *IEEE Transactions on Neural Networks*, 12(6):1455–1463, 2003.

[141] R. M. Inigo, A. Bonde, and B. Holcombe. Self adjusting weights for hardware neural networks. *Electronics Letters*, 26:1630–1632, 1990.

[142] Intel. Intel pentium processors with MMX^{TM} technology, 2007.

[143] Intel. Intel Streaming SIMD Extensions 4 (SSE4) Instruction Set Innovation, 2007.

[144] M. Ishikawa, N. Mukouzaka, H. Toyoda, and Y. Suzuki. Optical association: A simple model for optical associative memory. *Applied Optics*, 28:291–301, 1989.

[145] T. Ito, K. Fukushima, and S. Miyake. Realization of neural network model Neocognitron on a hypercube parallel computer. *International Journal of High Speed Computing*, 2:1–10, 1990.

[146] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[147] Kuo R. J., Wu P. C., and Wang C. P. Fuzzy neural networks for learning fuzzy if-then rules. *Applied Artificial Intelligence*, 14:539–563, 2000.

[148] M. Jabri and B. Flower. Weight perturbation: An optimal architecture and learning technique for analog VLSI feeforward and recurrent multilayer networks. *Neural Computation*, 3(4):546–565, 1991.

[149] A. Jahnke, T. Schoenauer, U. Roth, K. Mohraz, and H. Klar. Simulation of spiking neural networks on different hardware platforms. In *Proceedings of ICANN*, pages 1187–1192, 1997.

[150] J. -S. Jang, S. G. Shin, S. W. Yuk, S. Y. Shin, and S. Y. Lee. Dynamic optical interconnections using holographic Lenslet arrays for adaptive neural networks. *Optical Engineering*, 32:80–87, 1993.

[151] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.* Prentice Hall, 1997.

[152] Jyh Sing Roger Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665–684, 1993.

[153] Christopher Johansson and Anders Lansner. Towards cortex sized artificial neural systems. *Neural Networks*, 20(1):48–61, 2007.

[154] Simon Johnston, Girijesh Prasad, Liam P. Maguire, and T. Martin McGinnity. Comparative investigation into classical and spiking neuron implementations on FPGAs. In *ICANN (1)*, pages 269–274, 2005.

[155] S. R. Jones, K. M. Sammut, and J. Hunter. Learning in linear systolic neural network engines: Analysis and implementation. *IEEE Transactions on Neural Networks*, 5:584–593, 1994.

[156] J.A. Kahle, M.N. Day, H.P. Hofstee, C.R. Johns, T.R. Maeurer, and D. Shippy. Introduction to the Cell multiprocessor. *IBM Journal of Research and Development*, 49(4/5):589, 2005.

[157] Subhash Kak. New algorithms for training feedforward neural networks. *Pattern Recogn. Lett.*, 15(3):295–298, 1994.

[158] J. V. Kennedy and J. Austin. A hardware implementation of a binary neural image processor. In *Proceedings of the IV International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 178–185, Torino, Italy, 1994.

[159] E. V. Keulen, S. Colak, H. Withagen, and H. Hegt. Neural network hardware performance criteria. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1885–1888, 1994.

[160] D. Kim, H. Kim, H. Kim, G. Han, and D. Chung. A SIMD neural network processor for image processing. *Advances in Neural Networks*, 3497:665–672, 2005.

[161] P. Kinget and MSJ Steyaert. A programmable analog cellular neural network CMOS chip for high-speed image processing. *IEEE Journal of Solid State Circuits*, 30(3):235–243, 1995.

[162] J. M. Kinser and T. Lindblad. Implementation of pulse-coupled neural networks in a CNAPS environment. *IEEE-NN*, 10(3):584–597, 1999.

[163] H. Klapuri, T. Hamalainen, J. Saarinen, and J. Kaski. Mapping artificial neural networks to a tree shape neurocomputer. *Microprocessors Microsystems*, 20:267–276, 1996.

[164] TJ Koickal, LC Gouveia, and A. Hamilton. A programmable time event circuit block for reconfigurable neuromorphic computing. *Lecture Notes in Computer Science*, 4507:430–437, 2007.

[165] T.J. Koickal, L.C. Gouveia, and A. Hamilton. A programmable spike-timing based circuit block for reconfigurable neuromorphic computing. *Neurocomputing*, 72(16-18):3609–3616, 2009.

[166] T.J. Koickal, A. Hamilton, S.L. Tan, J.A. Covington, J.W. Gardner, and T.C. Pearce. Analog VLSI circuit implementation of an adaptive neuromorphic olfaction chip. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(1):60–73, 2007.

[167] kos Zarndy, Pter Fldesy, Pter Szolgay, Szabolcs Tks, Csaba Rekeczky, and Tams Roska. Various implementations of topographic, sensory, cellular wave computers. In *IEEE International Symposium on Circuits and Systems - ISCAS (6)*, pages 5802–5805, Piscataway, N.J., USA, 2005. IEEE Computer Society.

[168] J. Kramer. An on/off transient imager with event-driven, asynchronous read-out. In *IEEE International Symposium on Circuits and Systems*, volume II, pages 165–168, Phoenix, AZ, USA, 2002.

[169] M. Krips, T. Lammert, and A. Kummert. FPGA implementation of a neural network for a real-time hand tracking system. In *Proceedings of 1st IEEE International Workshop on Electronic Design, Test and Applications*, pages 313–317, 2002.

[170] J. M. Kumar and L. M. Patnaik. Mapping of artificial neural networks onto message passing systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26:822–835, 1996.

[171] V. Kumar, S. Shekhar, and M. B. Amin. A scalable parallel formulation of the back-propagation algorithm for hypercubes and related architectures. *IEEE Transactions on Parallel and Distributed Systems*, 5(10):1073–1090, 1994.

[172] S. Y. Kung. *Digital Neural Networks.* Prentice-Hall, Upper Saddle River, NJ, USA, 1993.

[173] S. Y. Kung and J. N. Hwang. Parallel architectures for artificial neural nets. In *Proceedings of International Conference on Neural Networks*, volume II, pages 165–172, San Diego, CA, 1988.

[174] S. Y. Kung and J. N. Hwang. A unified systolic architecture for artificial neural networks. *Journal of Parallel Distributed Computing*, 6:358–387, 1989.

[175] H. Lamela, M. Ruiz-Llata, and C. Warde. Optical broadcast interconnection neural network. *Optical Engineering*, 42:2487–2488, 2003.

[176] Horacio Lamela and Marta Ruiz-Llata. Optoelectronic neural processor for smart vision applications. *Imaging Science Journal*, 55(4):197–205, 2007.

[177] Tor Sverre Lande, editor. *Neuromorphic Systems Engineering: Neural Networks in Silicon.* Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[178] O. Landolt, E. Vittoz, and P. Heim. CMOS self biased euclidean distance computing circuit with high dynamic range. *Electronics Letters*, 28:352–354, 1992.

[179] E. Lange, Y. Nitta, and K. Kyuma. Optical neural chips. *IEEE Micro*, 14:29–41, 1994.

[180] J. Lazzaro and C. Mead. A silicon model of auditory localization. *Neural Computation*, 1:41–70, 1989.

[181] J. Lazzaro and J. Wawrzynek. Low-power silicon neurons, axons and synapses. *Silicon implementation of pulse coded neural networks*, pages 153–164, 1994.

[182] J. H. Lee. *CMOL CrossNets ad Defect-Tolerant Classifiers.* PhD thesis, Stony Brook University, NY, 2007.

[183] Jung Hoon Lee and Konstantin Likharev. CMOL crossnets as pattern classifiers. In *Proceedings of 8th International Work-Conference on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems*, volume 3512 of *Lecture Notes in Computer Science*, pages 446–454, Berlin, Heidelberg, Germany, 2005. Springer.

[184] Jung Hoon Lee and Konstantin Likharev. In situ training of CMOL crossnets. In *Proceedings of the International Joint Conference on Neural Networks*, pages 5026–5034, 2006.

[185] Jung Hoon Lee, Xiaolong Ma, and Konstantin Likharev. CMOL crossnets: Possible neuromorphic nanoelectronic circuits. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 755–762. MIT Press, Cambridge, MA, 2006.

[186] T. Lehmann, E. Bruun, and C. Dietrich. Mixed analog/digital matrix-vector multiplier for neural network synapses. *Analog Integrated Circuits and Signal Processing*, 9(1):55–63, 2004.

[187] H. Li, D. Zhang, and S.Y. Foo. A stochastic digital implementation of a neural network controller for small wind turbine systems. *IEEE Transactions on Power Electronics*, 21(5):1502–1507, Sept. 2006.

[188] P. Lichtsteiner, C. Posch, and T. Delbruck. An 128x128 120dB 15us-latency temporal contrast vision sensor. *IEEE Journal of Solid State Circuits*, 43(2):566–576, 2007.

[189] Konstantin K. Likharev. CMOL: Second life for silicon? *Journal of Microelectronics*, 39(2):177–183, 2008.

[190] Konstantin K. Likharev and D. B. Strukov. *CMOL: Devices, Circuits, and Architectures*, chapter 16, pages 447–477. Springer, Berlin, Heidelberg, Germany, 2005.

[191] G. S. Lim, M. Alder, and P. Hadingham. Adaptive quadratic neural nets. *Pattern Recognition Letters*, 13:325–329, 1992.

[192] G. Linan, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vazquez. ACE4k: An analog I/O 64x64 visual microprocessor chip with 7-bit analog accuracy. *International Journal of Circuit Theory and Applications*, 30(2-3):89–116, 2002.

[193] B. Linares-Barranco, A. G. Andreou, G. Indiveri, and T. Shibata, editors. *Special Issue on Neural Networks Hardware Implementation: IEEE Transactions on Neural Networks*, volume 14(5), 2003.

[194] T. Lindblad and J. M. Kinser. *Image Processing Using Pulse-Coupled Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[195] C. Lindsey and T. Lindblad. Review of hardware neural networks: A user's perspective. In *Proceedings of 3rd Workshop on Neural Networks: From Biology to High Energy Physics*, pages 195–202, Isola d'Elba, Italy, September 1994.

[196] C.S. Lindsey and T. Lindblad. Survey of Neural Network Hardware. In *Proceedings of the 1st International Conference on Applications and Science of Artificial Neural Networks*, volume 2492, pages 1194–1205. the Society of Photo-Optical Instrumentation Engineers (SPIE), 1995.

[197] Jin Liu, Martin A. Brooke, and Kenichi Hirotsu. A CMOS feed-forward neural network chip with on-chip parallel learning for oscillation cancellation. *IEEE Transactions on Neural Networks*, 13:1178–1186, 2002.

[198] JC Lopez-Garcia, MA Moreno-Armendariz, J. Riera-Babures, M. Balsi, and X. Vilasis-Cardona. Real time vision by FPGA implemented CNNs. In *Proceedings of the 2005 European Conference on Circuit Theory and Design*, volume 1, pages 281 – 284, 2005.

[199] D. R. Lovell, T. Downs, and A. C. Tsoi. An evaluation of the neocognitor. *IEEE Transactions on Neural Networks*, 8(5):1090–1105, September 1997.

[200] Teresa B. Ludermiry, Andre de Carvalhoz, Antonio P. Bragax, and Marclio C. P. de Souto. Weightless neural models: A review of current and past works. *Neural Computing Surveys*, 2:41–61, 1999.

[201] R. F. Lyon and C. Mead. An analog electronic cochlea. *IEEE Trans. on Acoustics, Speech, and Sig. Proc.*, 36(7):1119–1133, July 1988.

[202] W. Maass and C.M. Bishop. *Pulsed Neural Networks*. The MIT Press, Cambridge, MA, USA, 2001.

[203] Wolfgang Maass. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 211–217, Cambridge, MA, USA, 1997. The MIT Press.

[204] Y. Maeda and R. J. P. de Figueiredo. Learning rules for neuro-controller via simultaneous perturbation. *IEEE Transactions on Neural Networks*, 8:1119–1130, 1997.

[205] Y. Maeda, H. Hirano, and Y. Kanata. A learning rule of neural networks via simultaneous perturbation and its hardware implementation. *Neural Networks*, 8(2):251–259, 1995.

[206] Y. Maeda and T. Tada. FPGA implementation of a pulse density neural network with learning ability using simultaneous perturbation. *IEEE transaction on Neural Networks*, 14(3):688–695, 2003.

[207] L. P. Maguire, T. M. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin. Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomput.*, 71(1-3):13–29, 2007.

[208] M. Mahowald. *An Analog VLSI System for Stereoscopic Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[209] N. Maria, A. Guerin-Dugue, and F. Blayo. 1d and 2d systolic implementations for radial basis function networks. In *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 34–45, Turin, Italy, 1994.

[210] P. Masa, K. Hoen, and H. Wallinga. A high-speed analog neural processor. *IEEE Micro*, pages 40–50, 1994.

[211] N. Mauduit, M. Duration, and J. Gobert. Lneuro 1.0: A piece of hardware LEGO for building neural network systems. *IEEE Transactions on Neural Networks*, 3:414–422, 1992.

[212] H. McCartor. Back propagation implementation on the Adaptive Solutions CNAPS neurocomputer chip. In R. Lippmann and et al., editors, *Advances in Neural Information Processing Systems*, pages 1028–1031. Morgan Kaufmann, 1991.

[213] S. McLoone and G. W. Irwin. Fast parallel off-line training of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 8:646–653, 1997.

[214] Carver Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, Boston, MA, USA, 1989.

[215] R. W. Means and L. Lisenbee. Extensible linear floating point SIMD neurocomputer array processor. In *Proceedings of International Joint Conference on Neural Networks*, volume I, pages 587–592, Seattle, Washington, July 1991.

[216] P. Merolla and K. Boahen. A Recurrent Model of Orientation Maps with Simple and Complex Cells. *Advances in Neural Information Processing Systems*, pages 995–1002, 2004.

[217] S. Mihalas and E. Niebur. A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. *Neural Computation*, 21(3):704–718, 2009.

[218] Momchil Milev and Marin Hristov. Analog implementation of ANN with inherent quadratic nonlinearity of the synapses. *IEEE Transactions on Neural Networks*, 14(5):1187 – 1200, 2003.

[219] G. Moagar-Poladian and M. Bulinski. Reconfigurable optical neuron based on the transverse pockels effect. *Journal of Optoelectronics and Advanced Materials*, 4(4):929–936, 2002.

[220] P. D. Moerland and E. Fiesler. Hardware friendly learning algorithms for neural networks: An overview. In *Proceedings of the Fifth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 117–124, Lausanne, Switzerland, February 1996.

[221] P. D. Moerland and E. Fiesler. Neural Network Adaptations to Hardware Implementations. In E. Fiesler and R. Beale, editors, *Handbook of Neural Computation*, pages E1.2:1–13, New York, NY, USA, 1997. Institute of Physics Publishing and Oxford University Publishing.

[222] P. D. Moerland, E. Fiesler, and I. Saxena. Incorporation of liquidcrystal light valve nonlinearities in optical multilayer neural networks. *Applied Optics*, 35:5301–5307, 1996.

[223] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281 – 294, 1989.

[224] T. Morishita, Y. Tamura, T. Otsuki, and G. KANO. A BiCMOS analog neural network with dynamically updated weights. *IEICE TRANSACTIONS on Electronics*, 75(3):297–302, 1992.

[225] A. Mortara and E. A. Vittoz. A communication architecture tailored for analog VLSI artificial neural networks: intrinsic performance and limitations. *IEEE Trans. on Neural Networks*, 5:459–466, 1994.

[226] ML Mumford, DK Andes, LL Kern, U.S.N.W. Center, and C. Lake. The Mod 2 neurocomputer system design. *IEEE Transactions on Neural Networks*, 3(3):423–433, 1992.

[227] A. Muthuramalingam, S. Himavathi, and E. Srinivasan. Neural network implementation using FPGA: Issues and application. *International Journal of Information Technology*, 4(2):2–12, 2007.

[228] Z. Nagy and P. Szolgay. Configurable multilayer CNN-UM emulator on FPGA. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 50(6):774–778, 2003.

[229] Z. Nagy and P. Szolgay. Configurable multilayer CNN-UM emulator on FPGA. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 50(6):774–778, 2003.

[230] Z. Nagy, Z. Voroshazi, and P. Szolgay. An emulated digital retina model implementation on FPGA. In *9th International Workshop on Cellular Neural Networks and Their Applications*, pages 278–281, 2005.

[231] R. Navas, F. V. Verd, and A.R. Vzquez. Neuro-fuzzy chip to handle complex tasks with analog performance. *IEEE Transactions on Neural Networks: Special Issue on Neural Networks Hardware Implementations*, 14(5):1375–1392, 2003.

[232] D. Naylor, S. Jones, and D. Myers. Backpropagation in linear arrays – A performance analysis and optimization. *IEEE Transactions on Neural Networks*, 6:583–595, 1995.

[233] Nadia Nedjah and Luiza de Macedo Mourelle. Reconfigurable hardware for neural networks: binary versus stochastic. *Neural Computing and Applications*, 16(3):249–255, 2007.

[234] Neurologix. NLX420 data sheet, 1992. Neurologix, Inc., 800 Charcot Av., Suite 112, San Jose, Ca. USA.

[235] B. Noory and V. Groza. A reconfigurable approach to hardware implementation of neural networks. In *Proceedings of Canadian Conference on Electrical and Computer Engineering*, pages 1861–1864, 2003.

[236] T. Nordstrom and B. Svensson. Using and designing massively parallel computers for artificial neural networks. *Journal of Parallel Distributed Computing*, 14:260–285, 1992.

[237] Sung-Kwun Oh, W. Pedrycz, and Ho-Sung Park. Genetically optimized fuzzy polynomial neural networks. *IEEE Transactions on Fuzzy Systems*, 14(1):125–144, 2006.

[238] A. R. Ormondi and J. C. Rajapakse, editors. *FPGA Implementations of Neural Networks*. Springer-Verlag, Dordrecht, Germany, 2006.

[239] J.L. Ortiz and C.T. Ocasio. Analog hardware model for morphological neural networks. In *Proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence*, pages 40 – 44, Anaheim, CA, USA, 2003. ACTA Press.

[240] Y. Ota and B. M. Wilamowski. Analog implementation of pulse-coupled neural networks. *IEEE Transactions on Neural Networks*, 10:539–544, 1999.

[241] G. Palm, F. Kurfess, F. Schwenker, and A. Strey. Neural associative memories. *Associative Processing and Processors*, pages 284–306, 1997.

[242] F. Palmieri, J. Zhu, and C. Chang. AntiHebbian learning in topologically constrained linear networks: A tutorial. *IEEE Transactions on Neural Networks*, 4:748–761, 1993.

[243] M. Panella and A.S. Gallo. An input-output clustering approach to the synthesis of ANFIS networks. *IEEE Transactions on Fuzzy Systems*, 13(1):69–81, 2005.

[244] Ramin Pashaie and Nabil H. Farhat. Optical realization of bio-inspired spiking neurons in the electron trapping material thin film. *Applied Optics*, 46(35):8411–8418, 2007.

[245] L. M. Patnaik and R. N. Rao. Parallel implementation of neocognitron on star topology: Theoretical and experimental evaluation. *Neurocomputing*, 41:109–124, 2001.

[246] G.E. Pazienza, J. Bellana-Camanes, J. Riera-Babures, X. Vilasis-Cardona, M.A. Moreno-Armendariz, and M. Balsi. Optimized cellular neural network universal machine emulation on FPGA. In *Proceedings of the 18th European Conference on Circuit Theory and Design*, pages 815–818, 2007.

[247] G.E. Pazienza, X. Ponce-Garcia, M. Balsi, and X. Vilasis-Cardona. Robot vision with cellular neural networks: a practical implementation of new algorithms. *International Journal of Circuit Theory and Applications*, 35(4):449 – 462, 2007.

[248] A. Petrowski, L. Personnaz, G. Dreyfus, and C. Girault. Parallel implementations of neural network simulations. In F. Andre and J. P. Verjus, editors, *Hypercube and Distributed Computers*, pages 205–218, North-Holland, 1989. Elsevier Science B. V.

[249] D. Psaltis and Y. Qiao. Adaptive multilayer optical networks. In E. Wolf, editor, *Progress in Optics*, volume 31, pages 227–261, Amsterdam, 1993. Elsevier Science.

[250] Babuska R. and Verbruggen H. Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control*, 27:73–85, 2003.

[251] U. Ramacher. Neurocomputers: Towards a new generation of processors. *Siemens Review*, 61(3):26–29, 1994.

[252] U. Ramacher, W. Raab, J. Anlauf, U. Hachmann, and M. Wesseling. SYNAPSE-1 - A general purpose neurocomputer. Siemens AG Proprietary Information Manual, Corporate Research and Development Division, Siemens AG, Munich, February 1994.

[253] C. Rasche and R. Douglas. An improved silicon neuron. *Analog Integrated Circuits and Signal Processing*, 23(3):227–236, 2000.

[254] Leonardo Maria Reyneri. Design and Codesign of Neuro-Fuzzy Hardware. In *Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks: Bio-inspired Applications of Connectionism-Part II*, pages 14–30. Springer-Verlag London, UK, 2001.

[255] LM Reyneri. Unification of neural and wavelet networks and fuzzy systems. *IEEE Transactions on neural networks*, 10(4):801–814, 1999.

[256] L.M. Reyneri. On the performance of pulsed and spiking neurons. *Analog Integrated Circuits and Signal Processing*, 30(2):101–119, 2002.

[257] L.M. Reyneri. Implementation issues of neuro-fuzzy hardware: going toward HW/SW codesign. *Neural Networks, IEEE Transactions on*, 14(1):176–194, Jan 2003.

[258] K.L. Rice, T.M. Taha, and C.N. Vutsinas. Scaling analysis of a neocortex inspired cognitive model on the Cray XD1. *The Journal of Supercomputing*, 47(1):21–43, 2009.

[259] Patrick Rocke, Brian McGinley, Fearghal Morgan, and John Maher. Reconfigurable hardware evolution platform for a spiking neural network robotics controller. In Pedro C. Diniz, Eduardo Marques, Koen Bertels, Marcio Merino Fernandes, and João M. P. Cardoso, editors, *ARC*, volume 4419 of *Lecture Notes in Computer Science*, pages 373–378, Berlin, Heidelberg, Germany, 2007. Springer.

[260] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and SE Meana. ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(5):851–863, 2004.

[261] A.R.S. Romariz, P.U.A. Ferreira, J.V. Campelo, M.L. Graciano, and J.C da Costa. Design of a hybrid digital-analog neural co-processor for signal processing. In *Proceedings of the 22nd EUROMICRO Conference*, pages 513 – 519, 1996.

[262] E. Ros, E.M. Ortigosa, R. Agis, R. Carrillo, and M. Arnold. Real-time computing platform for spiking neurons (RT-spike). *IEEE transactions on Neural Networks*, 17(4):1050, 2006.

[263] T. Roska and LO Chua. The CNN universal machine: an analogic array computer. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40(3):163–173, 1993.

[264] T. Roska and A. Rodriguez-Vazquez. *Towards the Visual Microprocessor: VLSI Design and the Use of Cellular Neural Network Universal Machines.* John Wiley & Sons Ltd., Chichester, West Suzzex, England, 2000.

[265] T. Roska and J. Vandewalle. *Cellular neural networks*, chapter 39, pages 1075–1092. The circuits and filters handbook. CRC Press Boca Raton, FL, second edition, 2003.

[266] Ulrich Ruckert. An associative memory with neural architecture and its VLSI implementation. In *Proceedings of Hawaii International Conference on System Sciences*, pages 212–218, 1991.

[267] S. Rueping, K. Goser, and U. Ruckert. A chip for self-organizing feature maps. In *Proceedings of Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 26–33, 1994.

[268] Ruiz-Llata and H. Lamela-Rivera. Image identification system based on an optical broadcast neural network and a pulse coupled neural network preprocessor stage. *Applied Optics*, 47:47–10, 2008.

[269] M. Ruiz-Llata and H. Lamela-Rivera. Image identification system based on an optical broadcast neural network processor. *Applied optics*, 44(12):2366–2376, 2005.

[270] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Distributed representations. *Parallel Distributed Processing*, pages 77 – 109, 1986.

[271] F. M. Salam and T. Yamakawa, editors. *Special Issue on Micro-Electronic Hardware Implementation of Soft Computing: Neural and Fuzzy Networks with Learning*, volume 25, 1999.

[272] M. Salerno, F. Sargeni, and V. Bonaiuto. A dedicated multi-chip programmable system for cellular neural networks. *Analog Integrated Circuits and Signal Processing*, 18(2):277–288, 1999.

[273] F. Sargeni and V. Bonaiuto. A fully digitally programmable CNN chip. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 42(11):741–745, 1995.

[274] R. Sarpeshkar, R.F. Lyon, and C. Mead. A low-power wide-dynamic-range analog VLSI cochlea. *Analog integrated circuits and signal processing*, 16(3):245–274, 1998.

[275] I. Saxena and E. Fiesler. Adaptive multilayer optical neural network with optical thresholding. *Optical Engineering*, 34:2435–2440, 1995.

[276] A.V. Schaik. Building blocks for electronic spiking neural networks. *Neural Networks*, 14:617–628, 2001.

[277] J. Schemmel, J. Fieres, and K. Meier. Wafer-scale integration of analog neural networks. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 431–438. IEEE World Congress on Computational Intelligence, 2008.

[278] A. Schmid, Y. Leblebici, and D. Mlynek. A mixed analog digital artificial neural network with on chip learning. *IEE Proceedings - Circuits, Devices and Systems*, 146:345, 1999.

[279] T. Schoenauer, S. Atasoy, N. Mehrtash, and H. Klar. NeuroPipe-Chip: A digital neuro-processor for spiking neural networks. *IEEE transaction on Neural Networks*, 13(1):205–213, 2002.

[280] T. Schoenauer, A. Jahnke, U. Roth, and H. Klar. Digital neurohardware: Principles and perspectives. In *Proceedings of Neuronal Networks in Applications*, pages 101–106, Magdeburg, Germany, 1998.

[281] Benjamin Schrauwen and Michiel D'Haene. Compact digital hardware implementations of spiking neural networks. In J. Van Campenhout, editor, *Sixth FirW PhD Symposium*, page on CD, 2005.

[282] SR Schultz and MA Jabri. Analogue VLSI 'integrate-and-fire' neuron with frequency adaptation. *Electronics Letters*, 31(16):1357–1358, 1995.

[283] A. Serrano-Gotarredona, T. Serrano-Gotarredona, A. J. Acosta-Jiménez, and B. Linares-Barranco. An arbitrary kernel convolution AER-transceiver chip for real-time image filtering. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3145–3148, Kos, Greece, 2006.

[284] J. Shawe-Taylor, P. Jeavons, and M. Daalen. Probabilistic bit stream neural chip: Theory. *Connection Science*, 3(3):317–328, 1991.

[285] A. Shortt, J. G. Keating, L. Moulinier, and C. N. Pannell. Optical implementation of the Kak neural network. *Inf. Sci. Inf. Comput. Sci.*, 171(1-3):273–287, 2005.

[286] H. Shouno. Recent Studies Around the Neocognitron. *Lecture Notes In Computer Science*, pages 1061–1070, 2007.

[287] Paulo E. X. Silveira. Optoelectronic neural networks. In Ronald G. Driggers, editor, *Encyclopedia of Optical Engineering 1:1*, pages 1887 – 1902. CRC Press, 270 Madison Avenue, New York, NY, USA, 2003.

[288] A. Singer. Implementations of artificial neural networks on the connection machine. *Parallel Computing*, 14:305–315, 1990.

[289] S. R. Skinner, J. E. Steck, and E. C. Behrman. Optical neural network using kerr-type nonlinear materials. In *Proceedings of Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pages 12–15, 1994.

[290] Miroslav Skrbek. Fast neural network implementation. *Neural Network World*, 5:375–391, 1999.

[291] F. J. Smieja. Neural network constructive algorithms: Trading generalization for learning efficiency? *Circuits, Systems, and Signal Processing*, 12:331–374, 1993.

[292] L.S. Smith. Implementing neural models in silicon. *Handbook of Nature-Inspired And Innovative Computing: Integrating Classical Models with Emerging Technologies*, page 433, 2006.

[293] Stratix. Stratix III FPGA device family overview, 2007.

[294] A. Strey and N. Avellana. A new concept for parallel neurocomputer architectures. In *Proceedings of EuroPar'96*, pages 470–477, Lyon, France, August 1996.

[295] N. Sundararajan and P. Saratchandran. *Parallel Architectures for Artificial Neural Networks: Paradigms and Implementations.* IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[296] Synaptics. Synaptic touch pad, last accessed at Nov 22, 2009.

[297] T. Szabo, L. Antoni, G. Horvath, and B. Feher. A full-parallel digital implementation for pre-trained NNs. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 2, page 2049, 2000.

[298] K. W. Tang and S. Kak. A new corner classification approach to neural network training. *Circuits, Systems, and Signal Processing*, 17(4):459–469, 1998.

[299] David Terman and DeLiang Wang. Global competition and local cooperation in a network of neural oscillators. *Physica D*, 81(1-2):148–176, 1995.

[300] Sz. Tokes, L. Orzò, Gy. Vr, and T. Roska. Bacteriorhodopsin as an analog holographic memory for joint fourier implementation of CNN computers. Technical Report DNS-3-2000, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary, 2000.

[301] O. Turel. *Devices and Circuits for Nanoelectronic Implementation of Artificial Neural Networks.* PhD thesis, Stony Brook University, NY, 2007.

[302] Ö. Türel, J.H. Lee, X. Ma, and K.K. Likharev. Architectures for nanoelectronic implementation of artificial neural networks: New results. *Neurocomputing*, 64:271–283, 2005.

[303] M. Valle. *Smart Adaptive Systems on Silicon.* Kluwer Academic Publishers, 3300 AA Dordrecht, The Netherlands, 2005.

[304] P. Vas. *Sensorless Vector and Direct Torque Control.* Oxford University Press, USA, 1998.

[305] D. Ventura. On the utility of entanglement in quantum neural computing. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1565–1570, 2001.

[306] Michel Verleysen, Lean luc Voz, and Jordi Madrenas. An analog processor architecture for a neural network classifier. *IEEE Micro*, 14:16–28, 1994.

[307] R.J. Vogelstein, U. Mallik, J.T. Vogelstein, and G. Cauwenberghs. Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Transactions on Neural Networks*, 18(1):253, 2007.

[308] U. Vollmer and A. Strey. Experimental study on the precision requirements of RBF, RPROP and BPTT training. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, pages 239 – 244, London, UK, 1999.

[309] Z. Voroshazi, A. Kiss, Z. Nagy, and P. Szolgay. Implementation of embedded emulated-digital CNN-UM global analogic programming unit on FPGA and its application. *International Journal of Circuit Theory and Applications*, 36(5-6):589–603, 2008.

[310] DeLiang Wang and David Terman. Image segmentation based on oscillatory correlation. *Neural Computation*, 9(4):805–836, 1997.

[311] Jun Wang. An analog neural network for solving the assignment problem. *Electronic Letters*, 28(11):1047 – 1050, 1992.

[312] Weizhi Wang and Dongming Jin. Neuro-fuzzy system with high-speed low-power analog blocks. *Fuzzy Sets and Systems*, 157(22):2974–2982, 2006.

[313] Michael Weeks, Michael Freeman, Anthony Moulds, and Jim Austin. Developing hardware-based applications using PRESENCE-2. In *Perspectives in Pervasive Computing*, pages 469 – 474, Savoy Place, London, 2005.

[314] B. Widrow and M. A. Lehr. Thirty years of adaptive neural networks: Perceptron, Madaline, and Backpropagation. *Proceedings of the IEEE*, 78:1415–1442, 1990.

[315] J.H.B. Wijekoon and P. Dudek. Compact silicon neuron circuit with spiking and bursting behaviour. *Neural Networks*, 21(2-3):524–534, 2008.

[316] B. M. Wilamowski, J. Binfet, and M. O. Kaynak. VLSI implementation of neural networks. *International Journal of Neural Systems*, 10(3):191197, 2000.

[317] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960 – 962, 1969.

[318] M. Witbrock and M. Zagha. An implementation of back-propagation learning on GF11, a large SIMD parallel computer. *Parallel Comput.*, 14:329–346, 1990.

[319] Xilinx. Virtex-ii pro and virtex-ii pro x platform fpgas: Complete data sheet, 2007.

[320] ME Yalcin, JAK Suykens, and J. Vandewalle. Spatiotemporal pattern formation in the ACE16k CNN chip. In *In proceedings of the IEEE International Symposium on Circuits and Systems*, pages 5814–5817, 2005.

[321] F. Yang and M. Paindavoine. Implementation of an RBF neural network on embedded systems: Real-time face tracking and identity verification. *IEEE Transaction on Neural Networks*, 14(5):1162–1175, 2003.

[322] F. T. S. Yu, T. Lu, X. Yang, and D. A. Gregory. Optical neural network with pocketsized liquidcrystal televisions. *Optics Letters*, 15:863–865, 1990.

[323] Francis T. S. Yu and Chii Maw Uang. Optical neural networks. In Ronald G. Driggers, editor, *Encyclopedia of Optical Engineering 1:1*, pages 1763–1777. CRC Press, 270 Madison Avenue, New York, NY, USA, 2003.

[324] KA Zaghloul and K. Boahen. Optic nerve signals in a neuromorphic chip I: Outer and inner retina models. *IEEE Transactions on Biomedical Engineering*, 51(4):657–666, 2004.

[325] K.A. Zaghloul and K. Boahen. Optic nerve signals in a neuromorphic chip II: Testing and results. *IEEE Transactions on Biomedical Engineering*, 51(4):667–675, 2004.

[326] K.A. Zaghloul and K. Boahen. A silicon retina that reproduces signals in the optic nerve. *Journal of Neural Engineering*, 3(4):257–267, 2006.

[327] A. Zarandy and C. Rekeczky. Bi-i: a standalone ultra high speed cellular vision system. *IEEE Circuits and Systems Magazine*, 5(2):36–45, 2005.

[328] K. Zhang, I. Ginzburg, B.L. McNaughton, and T.J. Sejnowski. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *Journal of Neurophysiology*, 79(2):1017, 1998.

[329] Lei Zhang, Yinhe Han, Huawei Li, and Xiaowei Li. Fault tolerance mechanism in chip many-core processors. *Tsinghua Science & Technology*, 12(Supplement 1):169 – 174, 2007.

[330] Jihan Zhu and Peter Sutton. FPGA implementations of neural networks - a survey of a decade of progress. In *Field-Programmable Logic and Applications*, volume 2778, pages 1062–1066, 2003.

[331] Q. Zou, Y. Bornat, J. Tomas, S. Renaud, and A. Destexhe. Real-time simulations of networks of Hodgkin–Huxley neurons using analog circuits. *Neurocomputing*, 69(10-12):1137–1140, 2006.

[332] J.M. Zurada. Analog implementation of neural networks. *IEEE Circuits and Devices Magazine*, vol.8, (no.5):36–41, 1992.