NIPS*91 VLSI Workshop Summary

Organized by
Clifford Lau, Office of Naval Research
Jim Burr, Stanford University

February 8, 1992

1 Introduction

This is a report on the NIPS Post-Conference Workshop on VLSI Neural Networks and Neurocomputers, which was held at Vail, CO on December 6-7, 1991. [1] The objective of the two day workshop was to address the latest advances in VLSI implementation of neural nets, and the design of high performance neurocomputers. The following individuals participated in the workshop:

Clifford Lau	clau@charm.isi.edu
Jim Burr	burr@mojave.stanford.edu
Pat Shoemaker	shoe@cod.nosc.mil
Alice Chiang	alice@micro.ll.mit.edu
Tony Montalvo	tonym@sting.ece.ncsu.edu
Shaolin Li	shaolin@helmholtz.sdsc.edu
Nathan Zeldes	nzeldes@fab8.intel.com
Torsten Lehmann	torsten@eiffel.ei.dth.dk
Michael Chuang	chuang@micro.ll.mit.edu
Hans Peter Graf	hpg@neural.att.com
Andreas Andreou	andreou@hermes.ece.jhu.edu
Eduard Sackinger	edi@neural.att.com
Josh Alspector	josh@bellcore.com
Fin Martin	fmartin@sc9.intel.com
Chin Park	cpark@td2cad.intel.com
Tony Jayakumar	tony@flash.bellcore.com
Ron Benson	benson@hope.caltech.edu
Marwan Jabri	marwan@ee.su.oz.au
Hal McCartor	hal@asi.com
Alan Cooley	cooley@beno.css.gov
John Wawrzynek	johnw@cs.berkeley.edu
Tom Schwartz	-
Steve Anderson	-

[1] This summary may be obtained in latex (.tex) or postscript (.ps) format via anonymous ftp from archive.cis.ohio-state.edu:/pub/neuroprose/NIPS91/vlsi.summary.{tex,ps} .

The following people did not attend the workshop but have requested copies of the workshop summary:

Kelvin Wagner	kelvin@boulder.colorado.edu
Max van Daalen	max@cs.rhbnc.ac.uk

2 Issues for discussion

The plan was to address the following questions:

- How successful have VLSI implementations of neural nets been so far?
- What high performance neural net chips are available today?
- How are neural net chips different from DSP chips?
- Are dedicated neural net chips being used in real applications?
- What algorithms have been implemented? Which ones have not been? Why not?
- How important is on-chip learning?
- How much arithmetic precision is necessary?
- Which is more important, capacity or performance?
- What are the issues in constructing very large networks? What are the scaling limits?
- What is the appropriate architecture for high performance neurocomputers?
- What are the input/output devices for such neurocomputers?
- Can we achieve the equivalent of one TERAOPS sustained performance? How?
- What are the application areas for supercomputing neurocomputers?
- What application and programming support environment is necessary?

3 Workshop agenda

- Fri morning
  - New chips
  - Application survey: computation and capacity requirements
  - New neuromorphic circuits and structures (not discussed)
  - Synaptic cell area
  - Connection energy: decision and storage (not discussed)
  - Analog vs digital vs optical (not discussed)

- Fri evening
  - case studies

ASI CNAPS	Hal McCartor, Adaptive Solutions
RAP	John Wawrzynek, U.C.Berkeley and ICSI
CLC	Josh Alspector, Bellcore
LL CCD	Alice Chiang, MIT

- Sat morning
  - High performance neurocomputer attributes
  - TeraOps processor design
- Sat evening
  - Benchmarks
  - CAD design tools
  - Programming support environment

4 Chip and board level design issues

As indicated in the agenda, discussions on Friday centered on chip level and board level design issues. During the opening discussions, it was clear that VLSI neural net chip design is a series of trade-offs. At the device level, there are many possible ways and technologies to realize modifiable synapses and the sigmoidal nonlinearity. Device level issues were not intended to be a part of the workshop. At the chip and board level, however, there are issues such as special purpose chips versus general purpose chips, network size, expandability, and cascadability of the chips. At the small system level and high performance neurocomputer level, there are issues such bus interfaces, software support, commercial and scientific applications, and research environment.

Friday morning, we updated our list of reported neurochips, added a few applications to the DARPA CPS/connections graph, and presented and updated a table of synaptic cell areas. Friday evening, we heard from a number of researchers who have implemented large network machines.

5 New chips

Table 1 gives a summary of existing implementations. It is not meant to be exhaustive; it is meant to be a sampling of a wide variety of implementations. Analog, waferscale, neurocomputer, and biological entries are for comparison purposes. The two numbers in the *power* column are chip power dissipation and energy per connection.

6 Application survey

who	ref	type	tech	capacity (N, S)	accuracy (N, S)	speed CPS	power chip, syn	l	p	c	a
SNAP	TJS										
Chiang2	-	CCD	3μ CMOS	192, 6K	8-10, 6	1.9e9	8W, 4nJ	n	n	y	y
SAIC	TJS										
Motorola	TJS										
Intel/Nestor	TJS										
Synaptics	TJS										
Siemens MA16	[21]	digital	1.0μ CMOS	off-chip DRAM	16,16	7.2e8	-	y	n	y	n
Master Perturbator	MJ										
Bourke	MJ										
ANNA	[6]	D,A	0.9μ CMOS	64, 4096	3, 6	1e10	0.75W, 75pJ	n	y	y	y
Caltech	[1]	CCD	2μ CMOS	256, 64K	-	5e8	-	n	n	-	-
Chiang	[7]	CCD	3μ CMOS	144, 2016	8-10,6	1.44e9	2W, 1.4nJ	n	n	y	y
USC	[17]	cap	2μ CMOS	25, 2525	-	-	-	n	n	n	-
Matsushita	[18]	analog	2.2μ BiCMOS	64, 768	-	7.7e7	-	n	n	n	-
ETANN	[15]	analog	1.0μ EEPROM	64, 8192	8,8	8e9	2W, 250 pJ	n	n	y	y
NET32K	[11]	D,A	0.9μ CMOS	1024, 32K	-	3e11	160mW, 500fJ	n	n	y	-
ECLNN(CLC)	[2]	D,A	1.2μ CMOS	32, 1000	?,5	1e9	1W,1nJ	y	n	y	-
Mitsubishi	[4]	analog	1.0u CMOS	125, 10K	-	1e11	1.5W, 15pJ	y	n	y	-
JPL	[10]	analog	2.0μ CMOS	32, 1000	-	1e9	-	n	n	n	-
CNAPS(X1)	[13]	digital	0.8μ CMOS	-, 256K	8, 16	1e9	4W, 4nJ	y	y	y	y
Ouali	[19]	digital	2.0μ CMOS	-	-	-	-	n	n	y	-
Duranton	[9]	digital	1.6μ CMOS	-	-	8e8	-	y	-	y	-
Hirai	[14]	digital	1.2μ CMOS	6, 84	-	8.4e4	-	n	n	y	-
DNNA	[16]	stochastic	-	32, 1024	-	2e8	-	n	n	y	-
STONN	[24]	stochastic	-	20, 2000	-	5e8	-	n	n	y	-
TInMANN	[8]	stochastic	2.0μ CMOS	1, ?	-	-	-	y	n	y	-
Hitachi	[25]	digital wsi	0.8μ CMOS	540, 34K	-	1.4e8	-	n	n	y	-
CM1	[5]	mpp	-	-	-	3e6	-	y	y	-	y
Warp	[20]	systolic	-	-	-	17e6	-	y	y	-	y
AAP-2	[22]	array	-	-	-	18e6	-	y	y	-	y
SPRINT	[12]	systolic	-	-, 1e6	-	12e6	100W, 8μJ	y	y	-	y
neuron	[23]	-	-	1e11,1e15	8 bit?	1e16	1W, 0.1fJ	y	n	y	y

Table 1: Summary of existing implementations. In the right four columns, l = learning; p = programmable; c = cascadable; a = available.

application	connections	CPS
regular edge detection	50K	10^9
image segmentation	20K	$10^{10} - 10^{11}$
check reading	20K	400M
speech training	4M	10^{11} CUPS
speech preprocessing	100K	200M
speech recognition	40M	10^9

Table 2: Capacity and performance requirements of recent applications.

fets	type	$\lambda \times \lambda$	λ^2	λ^2/T	rel.area	who
1T	EPROM	8×8	64	64	1(6)	commercial
1T	DRAM	8×8	64	64	1(2)	[3]: 4 bit/cell
1T	DRAM	8×8	64	64	1(6)	commercial
4T	SRAM	12×20	240	60	4(24)	commercial
1T	DRAM	8×16	128	128	2(12)	MOSIS
2T	DRAM	13×31	403	200	6(36)	MOSIS
3T	DRAM	24×24	576	192	9(54)	MOSIS
6T	SRAM	32×36	1152	192	18(108)	MOSIS
7T	USC	40×60	2400	342	37	[17]
12T	ETANN	83×97	8036	671	125	[15]
42T	Mitsubishi	200×200	40000	952	625	[4]
200T	ECLNN	160×320	51200	256	700	[2]
1T	MIT CCD	12×10	120	120	2(12)	MOSIS
3T	MIT CAM	-	-	-	-	MIT
30T	Andreas	80×80	6400	213	100	JHU
30T	Bourke	106×113	11978	399	187	U.Sydney
30T	M.Pert	150×180	27000	900	422	U.Sydney

Table 3: Area of various synaptic cells. Numbers in parentheses are relative areas of 6-bit digital synapses.

We reviewed the CPS vs connections graph published in the 1988 DARPA report [23] to see how well the application domains defined there compared to capacity and performance requirements of recent applications.

We concluded the graph is still a good guide, with many of the more recent applications falling within its domains. A notable exception was vision, where weight sharing has been used to greatly reduce the number of physical connections required (see Table 2).

7 Synaptic cell area

We reviewed a table presented by Jim listing areas of various synaptic cells, and added some new entries (see Table 3). This table generated some lively discussion at the workshop. In a sense it is comparing apples and oranges, because the digital synapses are just memory cells, whereas most of

the analog synapses include learning circuits. Also, the digital cell area is quoted per bit whereas the analog cell stores the weight to full, albeit limited, precision.

Despite these differences, the relative area occupied by a single synaptic weight is an important parameter in computing the amount of silicon area required for weight storage in an application with a known network capacity.

A conclusion which can be drawn from this table is that analog implementations to date are better for low capacity, high performance networks, and digital implementations are better for high capacity, lower performance networks. This could change if someone were to build a time-multiplexed analog network which amortized each learning circuit over a cluster of synaptic weights.

8 Case studies

Friday afternoon, several researchers discussed specific implementations as background material for Saturday morning's TeraOps processor design.

8.1 ASI CNAPS

Hal McCartor of Adaptive Solutions (hal@asi.com) discussed the CNAPS chip and overall system design. The following is his contribution:

The current configuration for the CNAPS system is 4 CNAPS 1064 chips (64 processors per chip) on one board running at 20 MHz. The processors are fully programmable and operate as a massively parallel SIMD machine. Each of the 256 processors can perform a multiply and an add each clock cycle.

The system is supported by an assembler, a "C" compiler, a symbolic debugger, and a graphical user interface. There is support for calling the hardware from a normal "C" program or a scripting language using algorithm type, learning parameters, network size parameters etc. as arguments for the call.

8.2 RAP

John Wawrzynek (johnw@cs.berkeley.edu) discussed the ICSI RAP. The following is his contribution:

The Ring Array Processor (RAP) was designed for fast implementation of layered neural network algorithms [Beck, 1990; Bilmes and Kohn, 1990; Kohn and Bilmes, 1990; Morgan, 1990d; Morgan et al, 1990b]. This system was implemented in roughly one year (from conception to working prototype), and has since been in serious use for research into connectionist applications, particularly speech recognition. The RAP is a multi-DSP MIMD system with a low-latency ring interconnection scheme using programmable gate array technology and a significant amount of local memory (16 MB) per node. All inter-processor communication is handled by the ring. Multiple 4-processor boards serve as an array processor for a SPARC host. Theoretical peak performance is 128 MFlops/board, and peak ring communication speed is 64 MB/second/board. Test runs with up to ten boards (the largest configuration benchmarked) show a sustained throughput of roughly 10%

to 90% of this for algorithms of current interest. For a net dimension of sufficient size, forward propagation approaches 100% efficiency (with respect to 64 million multiply-accumulates/second/board for a 16 MHz clock), and ten RAP boards are as much as three orders of magnitude faster than a Sun SPARCStation-1 running the same benchmark (typically 50-120 Million Connection Updates Per Second, or MCUPS, and 200-570 Million Connections Per Second, or MCPS). Software development is aided by a Sun-based command interpreter, a Unix-compatible library (e.g., fread() and fwrite()), and an assembly language library of matrix and vector routines.

A reasonable design for a parallel system in which all processors need values from all other processors is a single broadcast bus (e.g., for forward propagation in a fully connected layered network). However, it is difficult to design a high-performance bus that supports more than a handful of processors. An alternative is a ring design, which is also more flexible for other related algorithms such as the backward phase of the back-propagation learning algorithm, for which the optimal storage of the weight matrix is in the opposite order. To avoid duplication of the weight matrix to get both orderings, one can use a circular pipeline or ring architecture to distribute partial sums to neighboring processors where local contributions to these sums can be added. When the processors must communicate with each other through the ring, hardware synchronization automatically occurs. A node that attempts to read before data is ready, or to write when there is already data waiting, will stop executing until the data can be moved.

The RAP DSP software is built in three levels. At the lowest level are hand coded assembler routines for matrix, vector and ring operations. An intermediate level consists of matrix and vector object classes coded in C++. A programmer writing at this level or above can view the RAP as a conventional serial machine. These object classes divide the data and processing as evenly as possible among the available processing nodes, using the ring to redistribute data. The top level of RAP software is the Connectionist Layered Object-oriented NEtwork Simulator (CLONES) environment for constructing ANNs [Kohn 1990].

8.3 ECLNN(CLC)

Josh Alspector (josh@bellcore.com) discussed his ECLNN chip. The following is his contribution:

We report learning measurements from a system composed of a cascadable learning chip, data generators and analyzers for training pattern presentation, and an X-windows based software interface. The 32 neuron learning chip has 496 adaptive synapses and can perform Boltzmann and mean-field learning using separate noise and gain controls. We have used this system to do learning experiments on the parity and replication problem. The system settling time limits the learning speed to about 100,000 patterns per second roughly independent of system size.

8.4 LL CCD: CCD's for PATTERN RECOGNITION

Alice Chiang (alice@micro.ll.mit.edu) discussed her CCD chips. The following is her contribution:

Recognition of a complex pattern in real time is an interesting, useful and challenging goal. One currently envisioned approach to this pattern recognition problem uses a multilevel, hierarchical, distributed computation and information neural network (NN) architecture. At the first processing level, processors will be used for signal conditioning such as noise removal, enhancing and/or skeletonizing an input image or Fourier transforming incoming speech. At the next level, neural

networks with local connections and replicated weights are used to generate feature maps of the processed input. Finally, the extracted feature map will be passed to higher level fully connected neural networks for object classification. Each of the above-mentioned processing level requires NN processors with large numbers of neurons, interconnections and interconnection computations/s, which poses very difficult VLSI design requirements, especially if the system has limited power and weight constraints. Since 1987, MIT Lincoln Laboratory has been developing high-speed, low-power, CCD-based neural network modules based on a time-multiplexed parallel-computation architecture. These modules share common features such as large numbers of time-multiplexed-parallel CCD computation elements, digitally stored connection weights and serial I/O buffers to support either multi-module ring-type or multi-level parallel, pipelined system architecture. In particular, CCD NN module consisting of 224 neurons, providing 1.9 billion connections/s, 6K interconnection storage has been developed and NN module consisting of 500 neurons, providing 5 billion connections/s and 100K interconnection storage can be developed in near future.

9 System and neurocomputer issues

The first discussion centered on the question, "What are the attributes of neurocomputers?" The term "neurocomputer" refers to a computer that mimics human information processing, with an architecture that is based on the abstraction of the computational principles in the brain. For that reason, neurocomputers must be sufficiently different from conventional computers to justify calling them by a different name. The driving force behind the need to design and and construct a high performance neurocomputer is the fact that there is often a claim that a "neural network chip can outperform a Cray" but yet never substantiated. Therefore our goal is to design a high performance neurocomputer with the equivalent of one TERAOPS, to show that indeed a neurocomputer can exploit its massive parallelism to achieve that kind of throughput rate.

If the neurocomputer were to mimic human information processing, it should have some attributes that are similar to that of the human being, attributes such as the ability to learn from experience, ability to remember things, recall from partial information, recognition, reasoning, understanding, and sensory motor data processing. It must also be small size and low power. It should have as a minimum the ability to process vision information and languages. The neurocomputer should not be constrained to the digital von Neumann kind of architecture, but should be flexible and innovative.

However, since a language for that kind of innovative neurocomputer does not yet exist, we are reduced to the current state of neurocomputers. It is felt that the following specifications, in the current terminology, would be considered a supercomputing neurocomputer:

- 10^{12} connections per second
- 2.5×10^{11} connection updates per second
- 10^{9} connection weights (synapses)
- 8 bit precision in numerical computations (16 bit preferred)
- Small size
- Low power

- Low cost

We started on the design of such a high performance neurocomputer. It was felt that the best digital architecture would be a mesh connected array. Such a nearest neighbor interconnected mesh would reduce the massive interconnections typically required in neural network realizations. Furthermore, almost all neural networks can be mapped onto a mesh connected array. Such an array, because of its regular geometry and next neighbor connections, is very expandable to larger size if the need arises. The processors would be neural network chips. The following are estimates of the number of chips required to build such a high performance neurocomputer:

1. RAP, ICSI
    - TMS320 DSP chip as processor
    - 16MCPS per processor
    - 4 processors per board
    - 64 MB per RAP board
    - 0.25 CPS/C - too low
    - Total 10,000 RAP boards
    - 25 watts per board
    - Total 250 KW power
2. Intel i860
    - 50MCPS per processor
    - Size of the array would be roughly 140 x 140
    - Total 20,000 chips
    - 50 KB memory per node for 1000 CPS/C
    - 40 KW power
3. CNAPS, Adaptive Solutions, Inc.
    - 25 MCPS per processor
    - 4 KB per processor
    - 6250 CPS/C - too high
    - 64 processors per chip
    - 4 chips per board
    - Total 1,000 boards with 4000 chips for 10^9 connections
    - 16 KW power (4W/chip)
4. Analog
    - Mitsubishi Boltzmann chip:
    - 300 neurons, 50K weights, 10^{11} CPS, 5W
    - 2 MCPS/C - much too high

- 10^9 connections require 20K chips, 100KW
- This gives over 10^{15} CPS.

5. standard DRAMs

   - 100ns access time
   - 4 bits/cycle: 5 MCPS/chip
   - 4 Mbit DRAM chips: 10 CPS/C - too low
   - 10^{12} CPS requires 200,000 chips
   - This gives 10^{11} connections

6. page mode DRAMs

   - Page mode access 100MHz
   - 4 bits/cycle: 50 MCPS/chip
   - 4 Mbit DRAM chips, page mode: 100 CPS/C - too low
   - 10^{12} CPS requires 20,000 chips
   - This gives 10^{10} connections

7. high speed SRAMs

   - 100MHz access, byte wide: 100 MCPS/chip
   - 1 Mbit SRAM: 128K weights, 780 CPS/C - just about right
   - 10^{12} CPS requires 10,000 chips
   - This gives 10^9 connections

8. CCD, MIT Lincoln Lab.

   - 500 KB per chip
   - 50 processors per chip
   - 20 MHz clock
   - 2000 CPS/C - a little too high
   - Total 2,000 chips
   - 2 KW power
   - This is beginning to look possible.

9. MOSIS digital 0.8μ CMOS chip using 32×32 SRAM (new design)

   - 100 MHz clock
   - 8×8 Multiply Accumulate
   - 100 KB memory on-chip
   - One processor per chip
   - 1000 CPS/C - just right for this application
   - Total 10,000 chips
   - 1 KW at 5 volt power supply

- This looks very possible too.

10. feedforward MOSIS analog

    - Orbit 2.0μ double poly floating gates: 16$\times$16 λ
    - 400K weights/chip
    - 4e11 CPS/chip (400K weights every 1μsec)
    - 1 MCPS/C - much too high
    - 400mW/chip
    - 2,500 chips for 10^9 connections
    - 10^{15} CPS
    - 1KW at 5 volt

11. Custom designed digital 0.8μ CMOS chip using 16$\times$16 SRAM

    - 100 MHz clock
    - 4 processors per chip, 8$\times$8 MAC
    - 400 KB memory per chip
    - 1000 CPS/C - just right
    - 2,500 chips
    - 1 KW at 5 volt, 40W at 1 volt (low voltage CMOS)
    - This looks quite possible.

12. Ideal feedforward 0.8μ analog

    - 8$\times$8 λ floating gates
    - 10M weights/chip
    - 1e13 CPS/chip (10M weights every 1μsec)
    - 1 MCPS/C - much too high
    - 10W/chip
    - 100 chips for 10^9 connections
    - 10^{15} CPS
    - 1KW at 5 volt

Although the chip design appears to be quite feasible, the workshop did not discuss the system level issues associated with the architecture of such a high performance neurocomputer. Certainly a computer is more than just a collection of chips. The mesh connected array is only one possibility. Furthermore, the issues of interconnection, communication protocol between chips, and routing of the activations and/or weights were not discussed. Many system integration issues remain open.

10 Supercomputing neurocomputer applications

In the afternoon session, discussion turned to application areas of such a high performance neurocomputer. Many potential applications were identified, such as vision processing, language and speech processing, weather modeling (which may require 16 bit precision), human genome classification project, brain modeling, and even stock market analysis and prediction. However, since such a high performance machine currently does not exist, it is premature to discuss its potential for application. Undoubtedly if and when such a machine becomes a reality, new applications will appear and be found to be in need of such a neurocomputer.

11 Benchmarks

Next, we turned to the problem of benchmarking for neurocomputers. The rationale is that we want to be able to compare the performance of neurocomputers so that the customers can objectively and intelligently choose among available alternatives. Hopefully the benchmark should be a problem that is independent of the neurocomputer realization, such as the FFT in conventional computer benchmarks. Benchmarking turned out to be a more difficult issue than it appears on the surface. Often it is difficult to separate the problem from its solution. In other words, neural network solutions are very often problem specific. It is generally agreed, however, that benchmarks should be developed at the board level and not at the chip level. This is because chips are only usable when they are put into a board.

One possible benchmark is the problem of convolution, which is often done in image processing. In this case, we can compare the time required to perform a convolution with 16 x 16 kernel over a 512 x 512 pixel image. Even in this case, the question arises as to whether the uploading and downloading of the data should be a part of the benchmark. It is clear that in order to have a reasonable set of benchmarks, it is necessary to develop a set of performance criteria. In other words, what is it that we are measuring and comparing?

Several possible performance parameters were discussed: time to complete task, accuracy of classification, size of training set, size of test set, % correct, % error, time to train the net, time to make a decision, processing throughput rate, cost, size, and power. It was felt that most likely the benchmarks should be involved in speech processing, vision, or optical character recognition. However, it was felt that problems such NETALK, ISOLET, and sonar were too small scale to really allow the power of neural networks to shine. Other databases such as the NIST character set are too difficult to use. In spite of the difficulty of identifying a set of benchmarks, it was felt that benchmarks were necessary.

11.1 Currently available benchmarks

After the workshop, Scott Fahlman (Scott_Fahlman@SEF-PMAX.SLISP.CS.CMU.EDU) sent Jim this email message in response to a query about accessing Nettalk:

```
The data set that you want is part of the collection of learning benchmarks
that we are maintaining at CMU.
```

If you just want to access this collection via FTP, here are the general instructions:

For people (at CMU, MIT, and soon some other places) with access to the Andrew File System (AFS), you can access the files directly from directory "/afs/cs.cmu.edu/project/connect/bench". This file system uses the same syntactic conventions as BSD Unix: case sensitive names, slashes for subdirectories, no version numbers, etc. The protection scheme is a bit different, but that shouldn't matter to people just trying to read these files.

For people accessing these files via FTP:

1. Create an FTP connection from wherever you are to machine "pt.cs.cmu.edu". The internet address of this machine is 128.2.254.155, for those who need it.

2. Log in as user "anonymous" with no password. You may see an error message that says "filenames may not have /.. in them" or something like that. Just ignore it.

3. Change remote directory to "/afs/cs/project/connect/bench". Any subdirectories of this one should also be accessible. Parent directories should not be. NOTE: you must do this cd as one atomic operation, not try to crawl in one directory at a time.

4. At this point FTP should be able to get a listing of files in this directory and fetch the ones you want.

5. The directory "/afs/cs/project/connect/code" contains public-domain programs implementing the Quickprop and Cascade-Correlation algorithms, among other things. Access it in the same way.

The exact FTP commands you use to change directories, list files, etc., will vary from one version of FTP to another. I'm not sure if there's currently any way to FTP files if you live on bitnet or csnet. I've heard for years about plans to add FTP capability between those nets and Arpanet, but I don't think it has happened yet. Once the collection is large enough, we may try to find some organization willing to make tapes for people at a reasonable cost.

Scott Fahlman
School of Computer Science
Carnegie Mellon University

12 Tools

We spent the last half of the Saturday afternoon session discussing tools for chip and network design.

12.1 Chip design

A number of public-domain or low-cost tools are available for chip design. The following list is not meant to be exhaustive, but represents tools actually used by one or more participants to design chips:

- Berkeley CAD suite
  - access: send email to vlsi@ucbvax.berkeley.edu
  - magic - VLSI layout editor
  - spice - transistor level circuit simulation
  - irsim - transistor level RC timing simulation (a descendant of rsim by Chris Terman of MIT, modified at Stanford, distributed with the Berkeley tools).
- Berkeley OCT tools
  - access: send email to vlsi@ucbvax.berkeley.edu
  - misII - logic synthesis
  - timberwolfe - standard cell place and route
- Berkeley Lager tools
  - access: send email to vlsi@ucbvax.berkeley.edu
  - timlager - tiling tool
  - flint - interactive floorplanner
- Caltech Analog suite
  - access: anonymous ftp to hobiecat.cs.caltech.edu
  - analog - device level simulator with nice user interface
  - wol - layout editor
  - wolcomp - layout composition tool
  - netcomp - netlist comparator used to compare analog netlists with wol and wolcomp layouts
- Tanner tools
  - access: call (818) 795-2696
  - ledit - layout editor, nice user interface, optimized for PCs
  - lvs - layout vs schematic comparator
  - gatesim - gate level simulator

- misc tools
  - access: ?
  - cande - CCD simulation
  - CaZM - spice rewrite from MCNC

12.2 Network design

We compiled a list of tools which participants have found useful in constructing and simulating neural networks and evaluating their performance. Two types of tools were identified: general and neural-net specific.

General tools include:

Tool	source	comments
Mathematica	Wolfram Research	(217) 398-0700
Matlab	the MathWorks, Inc.	
Labview		Mac only
Desire		
CNet	U. Washington	

Neural net tools include:

Tool	source	comments
NeuralWorks		Sun, Mac
Dynamind		PC only
Neuralist		
ICSIM	ICSI	
Aspirin,Migrane		
Planet		
Boolean	U. Toronto	
Cosmic	NASA	
Brainmaker		
Neural shell		PC only

13 Open issues

Throughout the workshop, it was clear that there were a number of open issues that needed resolution and discussion. These open issues include:

- Mapping of neural network to processor architecture.
- Mapping of the problem to the neural network approach (vice conventional DSP approach).
- Development of metric for measuring network performance.

- Development of benchmarks.
- Development of programming support environments (assembly language programming is generally not acceptable, high level language such as C is preferred.)
- Development of sensors and actuators to realize the potential of neural networks.
- Manufacturability of large scale neural network chips, boards, and systems.
- Fault tolerance of large scale network.
- Analog versus digital trade-offs, and design issues.
- Development of I/O devices to exploit the advantage of neurocomputers.

14 Summary

Due to the limited time available for the workshop, it was not possible to discuss in depth all the issues that we started out with. We were overly ambitious in terms of trying to come to resolution of many of the very difficult issues.

In terms of high performance neurocomputers, we found that it is quite feasible to design a neurocomputer to achieve the equivalent of one TERAOPS. Special custom design chips are most likely necessary. Many of the architecture and system integration issues remain unresolved, however. Some answers to the questions we set out in the beginning have been obtained, but many questions remain.

It is generally felt that the workshop was vastly successful. Many issues were brought up concerning the VLSI realization of neural nets during lively discussions. Everyone benefited from having participated in this workshop.

Application references

ASI CNAPS

RAP references

Beck, J. (1990). The Ring Array Processor (RAP): Hardware. *International Computer Science Institute Technical Report TR-90-048.*

Bilmes, J., and Kohn, P., (1990). Ring Array Processor (RAP): Software Architecture. *International Computer Science Institute Technical Report TR-90-050.*

Kohn, P., (1991). *CLONES: A Connectionist Layered Object-oriented NEtwork Simulator*, International Computer Science Institute, Technical Report In Prep.

Kohn, P. and Bilmes, J., (1990). Ring Array Processor (RAP): Software User's Manual Version 1.0. *International Computer Science Institute Technical Report TR-90-049.*

Morgan, N. (1990d). The Ring Array Processor (RAP): Algorithms and Architectures. *International Computer Science Institute Technical Report TR-90-047.*

Morgan, N., Beck, J., Kohn, P., Bilmes, J., Allman, E., and Beer, J. (1990b). The RAP: a Ring Array Processor for Layered Network Calculations. *Proc. of Intl. Conf. on Application Specific Array Processors*, pp. 296-308. IEEE Computer Society Press, Princeton, N.J.

ECLNN

LL CCD

References

[1] Aharon J. Agranat, Charles F. Neugebauer, and Amnon Yariv. A CCD based neural network integrated circuit with 64K analog programmable synapses". In *IJCNN International Joint Conference on Neural Networks*, pages II:551–555, 1990.

[2] Joshua Alspector. CLC - A cascadeable learning chip. NIPS90 VLSI Workshop, December 1990.

[3] Masakazu Aoki, Yoshinobu Nakagome, Masahi Horiguchi, Shin'ichi Ikenaga, and Katsuhiro Shimohigashi. A 16-level/cell dynamic memory. *IEEE Journal of Solid-State Circuits*, pages 297–299, April 1987.

[4] Yutaka Arima, Koichiro Mashiko, Keisuke Okada, Tsuyoshi Yamada, Atushi Maeda, Harufusa Kondoh, and Shinpei Kayano. A self-learning neural network chip with 125 neurons and 10K self-organization synapses. In *Symposium on VLSI Circuits*, pages 63–64, 1990.

[5] G. Blelloch and C. R. Rosenberg. Network learning on the Connection Machine. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 323–326, 1987. Milan, Italy.

[6] Bernhard Boser and Eduard Sackinger. An Analog Neural Network Processor with Programmable Topology. *IEEE Journal of Solid-State Circuits*, 26(12), December 1991.

[7] Alice Chiang, Robert Mountain, James Reinold, Jeoffrey LaFranchise, James Gregory, and George Lincoln. A programmable CCD signal processor. In *IEEE International Solid-State Circuits Conference*, pages 146–147, 1990.

[8] David Van den Bout and Thomas K. Miller III. TInMANN: The integer Markovian artificial neural network. In *IJCNN International Joint Conference on Neural Networks*, pages II:205–211, 1989.

[9] M. Duranton and J.A. Sirat. Learning on VLSI: A general purpose digital neurochip. In *IJCNN International Joint Conference on Neural Networks*, 1989.

[10] Silvio Eberhardt, Tuan Duong, and Anil Thakoor. Design of parallel hardware neural network systems from custom analog VLSI 'building block' chips. In *IJCNN International Joint Conference on Neural Networks*, pages II:183–190, 1989.

[11] Hans Peter Graf and Don Henderson. A reconfigurable CMOS neural network. In *IEEE International Solid-State Circuits Conference*, pages 144–145, 1990.

[12] A. J. De Groot and S. R. Parker. Systolic implementation of neural networks. In *SPIE, High Speed Computing II*, volume 1058, January 1989.

[13] Dan Hammerstrom. A VLSI architecture for high-performance, low-cost, on-chip learning. In *IJCNN International Joint Conference on Neural Networks*, pages II:537–544, 1990.

[14] Yuzo Hirai, Katsuhiro Kamada, Minoru Yamada, and Mitsuo Ooyama. A digital neuro-chip with unlimited connectability for large scale neural networks. In *IJCNN International Joint Conference on Neural Networks*, pages II:163–169, 1989.

[15] Mark Holler, Simon Tam, Hernan Castro, and Ronald Benson. An electrically trainable artificial neural network (ETANN) with 10240 "floating gate" synapses. In *IJCNN International Joint Conference on Neural Networks*, pages II:191–196, 1989.

[16] Max Stanford Tomlinson Jr., Dennis J. Walker, and Massimo A. Sivilotti. A digital neural network architecture for VLSI. In *IJCNN International Joint Conference on Neural Networks*, pages II:545–550, 1990.

[17] Bang W. Lee, Ji-Chien Lee, and Bing J. Sheu. VLSI image processors using analog programmable synapses and neurons. In *IJCNN International Joint Conference on Neural Networks*, pages II:575–580, 1990.

[18] Takayuki Morishita, Youichi Tamura, and Tatsuo Otsuki. A BiCMOS analog neural network with dynamically updated weights. In *IEEE International Solid-State Circuits Conference*, pages 142–143, 1990.

[19] J. Ouali and G. Saucier. Silicon compiler for neuro-ASICs. In *IJCNN International Joint Conference on Neural Networks*, pages II:557–561, 1990.

[20] Dean A. Pomerleau, George L. Gusciora, David S. Touretzky, and H.T. Kung. Neural network simulation at Warp speed: How we got 17 million connections per second. In *IEEE International Conference on Neural Networks*, pages II:143–150, 1988.

[21] U. Ramacher, J. Beichter, W. Raab, J. Anlauf, N. Bruls, M. Hachmann, and M. Wesseling. Design of a 1st generation neurocomputer. In U. Ramacher and U. Rucket, editors, *VLSI Design of Neural Networks*. Kluwer Academic, 1991.

[22] Takumi Watanabe, Yoshi Sugiyama, Toshio Kondo, and Yoshihiro Kitamura. Neural network simulations on a massively parallel cellular array processor: AAP-2. In *IJCNN International Joint Conference on Neural Networks*, pages II:155–161, 1989.

[23] Carol Weiszmann, editor. *DARPA Neural Network Study, October 1987 - February 1988.* AFCEA International Press, 1988.

[24] William Wike, David Van den Bout, and Thomas Miller III. The VLSI implementation of STONN. In *IJCNN International Joint Conference on Neural Networks*, pages II:593–598, 1990.

[25] Moritoshi Yasunaga, Noboru Masuda, Masayoshi Yagyu, Mitsuo Asai, Minoru Yamada, and Akira Masaki. Design, fabrication and evaluation of a 5-inch wafer scale neural network LSI composed of 576 digital neurons. In *IJCNN International Joint Conference on Neural Networks*, pages II:527–535, 1990.