

Benchmarking an MIMD Neural Network Processor

Karl Mathia¹, Jeff Clark², Brad Colbert, and Richard Saeks

Accurate Automation Corporation
7001 Shallowford Road
Chattanooga, Tennessee 37421

Abstract

Although artificial neural networks are increasingly recognized in academic and engineering communities as powerful tools for complex problem solving tasks, their use in time-critical applications often demands high performance, high cost hardware systems. Many real-world applications of neural networks eventually will require specialized neural network hardware (NNH) to achieve adequate performance at reasonable cost. This paper describes a multiple instruction multiple data (MIMD) neural network processor (NNP) and a set of NNH benchmarks. The benchmarks are applied to the NNP in a single and multiprocessor environment, and the NNP performance is compared against that of a high performance multiprocessor workstation.

1 Introduction

Due mostly to their learning capability, artificial neural networks (ANNs) are increasingly recognized in academic and engineering communities as powerful tools for complex problem solving tasks. Unfortunately, their use in time-critical applications often demands high performance, and therefore high cost hardware systems [1][4]. Such ANN applications eventually will require specialized neural network hardware (NNH) to achieve adequate performance at reasonable cost. The neural network processor (NNP³) by Accurate Automation Corporation, discussed below, is one realization of such specialized NNH. The design efforts for NNH is still in a relatively early stage, and researchers are experimenting with a variety of possible realizations. Analog, digital, and hybrid techniques are candidates for the implementation of neural network chips, neural network processors, and complete neurocomputer solutions.

The intrinsic parallelism of ANNs, i.e. local processing in numerous simple processing elements ('neurons') and the storage of learned knowledge in distributed memory ('connection weights'), makes the implementation on *dedicated parallel* hardware an obvious choice. The two main avenues for the development of NNH are currently

- *General purpose* hardware platforms (currently only digital) with a certain degree of flexibility for the implementation of a variety of ANN paradigms and learning algorithms, and
- *Special purpose* neural network processors or chips (analog, digital, or hybrid), specialized for the implementation of ANN architectures with high speed and efficiency.

The resulting variety of available neural network hardware makes the choice of the 'best' NNH a highly problem dependent task. A standardized set of benchmarks for comparing different NNH platforms is needed to facilitate this decision making process. A set of benchmarks was proposed for this purpose in [3] and [5]. In this paper variations of these benchmarks are applied to the NNP. The NNP performance is compared against that of a common Pentium based computer and that of a high performance Silicon Graphics multiprocessor workstation.

1. E-mail: kmathia@accurate-automation.com, mathiak@syc.pdx.edu.

2. Jeff Clark is now with Hughes Training Inc., Phoenix, AZ. E-mail: jclark@acm.org.

3. NNP[®] is a registered trademark of Accurate Automation Corporation.

2 The Neural Network Processor

The concept behind the development of the AAC neural network processor was to provide efficient NNH for processing a broad variety of feedforward ANNs and recurrent ANNs with high speed at reasonable cost [7]. The NNP's flexibility is based on its connection-based architecture and instruction set and allows the implementation of almost any connection topology. The resulting digital *special purpose* NNH consists of common electronic components and is optimized for certain neural processing tasks, i.e. its hardware resembles the local processing and distributed memory characteristics of ANNs. In theory, local processing in artificial neural networks is performed by *artificial neurons*. Neurons compute the output y for a given activation a . The activation is the inner product of input vector \mathbf{x} and vector of connection weights \mathbf{w} of that neuron, thus

$$y = g(a) = g(\mathbf{w}^T \cdot \mathbf{x}) , \quad (1)$$

where g represents the usually nonlinear neuron *activation function* (or *transfer function*). Neural networks can consist of a large number of neurons, and are organized in one or more layers of parallel neurons. All neurons perform a mapping as defined in Equation 1, but each with its distinct weight vector \mathbf{w} . The weights are adjusted during a learning process such that a particular problem is solved.

Multiplying one scalar input x with one scalar weight w in Equation 1 is called a *connection*. Connections and transfer functions are the most expensive computations in neural network implementations [5]. Because the number of neurons in ANNs can be large, and because the number of weights grows exponentially with the number of neurons, a NNH system must efficiently compute these connections. Computational speed can be further increased by an efficient implementation of neuron transfer functions. These two considerations were the driving force behind the NNP design. A multiply-accumulate unit and the storage of transfer functions in lookup tables are the basis of the NNP optimization (Figure 1). 16-bit fixed point arithmetic further increases computational speed. Efficiency is increased even more by limiting the instruction set to the minimal set of instructions necessary to program neural networks. The cornerstone of an NNP assembler program is the multiply-accumulate command, which multiplies one weight value with one input value and adds the product to the existing activation value. Instruction pipelining allows the completion of one command per clock cycle [7]. In addition to the above features, computational speed can be substantially increased by using the NNP in a multiprocessor environment. Up to eight NNPs in parallel form a multiple instructions multiple data (MIMD) NNH system, whose performance increases approximately linearly with the number of NNPs.

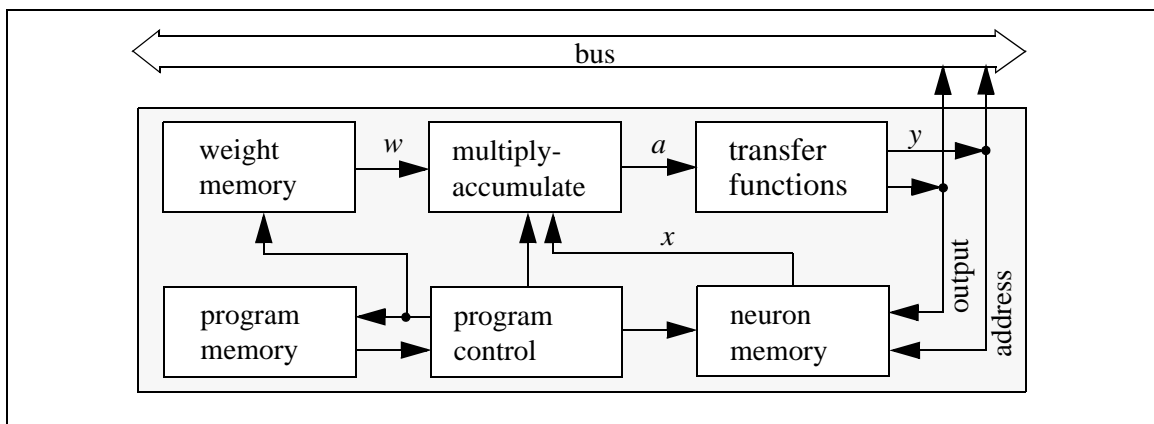


Figure 1 Block diagram of a single neural network processor.

3 Benchmarks for Neural Network Hardware

Despite the increasing interest in using neural networks for (often time critical) real world applications, the problem of benchmarking neural network hardware has rarely been addressed. A set of standardized NNH benchmarks is needed to facilitate the comparison of NNH system. *Connections-per-second* (CPS) were originally used to benchmark the computational speed of ANN software simulators on various computer systems, where the data format used was not a factor [5]. Unfortunately, data sheets provided by NNH manufacturers often follow this example and do not specify the data format used for benchmarking. For example, the CPS benchmark could state 1-bit connections, which is the least expensive and therefore fastest data format. Furthermore, the *theoretical* peak performance instead of the *realizable sustainable performance* is often stated. Considering all issues involved in hardware design, it is clear that a single number, for example CPS, cannot provide all information needed. In the following we define two benchmarks which, together with already existing measures, give a broader picture about the performance of NNH systems.

The numerical precision required for any application of digital hardware is highly problem dependent. This also applies to ANN applications, where the computational cost of connections grows with the numerical precision used. This computational cost can be expressed in a benchmark by normalizing the CPS with respect to the data format being processed [3]. The performance of neural network implementations can also depend on the ANN size. In theory, neural processing is perfectly parallelized and the data throughput of ANNs is independent of the network size, i.e. the number of neurons and weights. Of course, this is not (yet) realizable at reasonable cost, and the network size indeed affects the computational speed of ANNs. This can be expressed by normalizing CPS with respect to the number of weights in the network. Following the same line as in [3], we define two NNH benchmarks:

1. *Connection-bytes-per-second* (CBS). With the same notation as in Equation 1, we define

$$CBS = bytes(w) \cdot bytes(x) \cdot CPS \quad . \quad (2)$$

For generality of the CBS benchmark, the data format of both input x and weight w must be considered. In a multiplication both factors are processed bit by bit. The associated computational cost is expressed by the product in the above definition.

2. *Connection-bytes-per-second-per-weight* (CBSW). The computational speed of NNH is related to the network size by normalizing CBS with respect to the number of weights in the network,

$$CBSW = \frac{CBS}{N_w} \quad . \quad (3)$$

(N_w is the total number of weights in the network.)

For example, the computation of a 16-bit connection on the NNP involves the multiplication of two 16-bit numbers, which requires the processing of four bytes. The computational cost of one CPS on the NNP is therefore equivalent to four CBS, 1 CPS = 4 CBS. The above discussion also applies to ANN learning, although benchmarks for ANN performance during learning are not considered here.

4 Benchmark Tests

The objective of the benchmark tests was to measure the *realizable sustainable performance* (RSP) of the AAC NNP. As is shown below, for the NNP the RSP approaches the theoretical peak performance, which can be credited to the NNP's optimization for neural processing tasks. Although data I/O between

NNP and host computer is an important factor in applications, here only the on-board neural processing performance is considered. The continuous Hopfield network [2] has been chosen to benchmark the NNP for several reasons: the continuous Hopfield network (CHN) is a well known ANN paradigm, it is easy to implement, and its computational complexity is similar to that of a feedforward network with bias terms, e.g. a multilayer perceptron (MLP) [6]. When 'unfolding' the CHN in time (Figure 2), it is clear that each MLP layer is equivalent to one CHN iteration. This holds when all MLP layers have the same number of neurons. Thus, CHN benchmarks also apply to MLPs. CHNs with $2^n, n = 1, \dots, 8$, neurons were tested. The dynamics of a discrete-time implementation of continuous Hopfield networks is given by

$$\mathbf{x}_{k+1} = \mathbf{g}(\mathbf{W} \cdot \mathbf{x}_k + \mathbf{u}), \quad (4)$$

where \mathbf{x} is the state vector, \mathbf{W} is the weight matrix, \mathbf{u} is the external input vector, and \mathbf{g} is a vector of sigmoidal transfer functions.

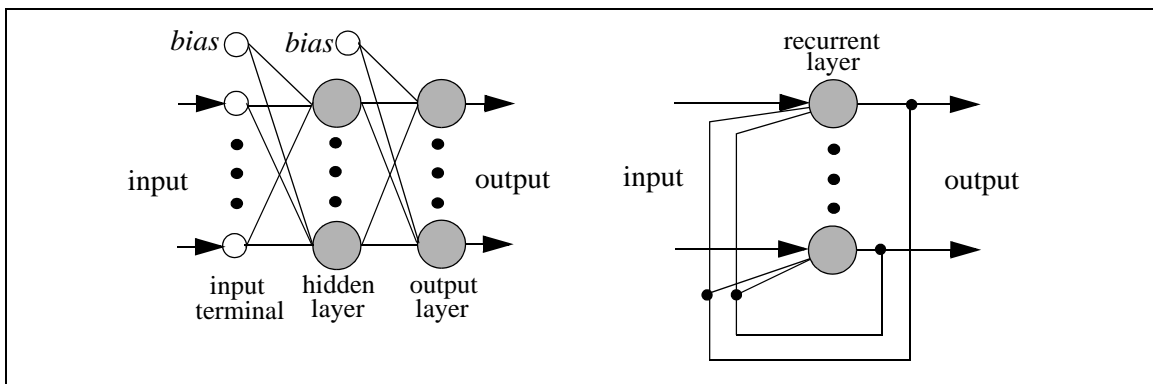


Figure 2 ANN architectures with similar computational complexity: multilayer perceptron (left), and Hopfield network (right).

For a direct comparison the CHNs were also coded in the C-programming language. Weight matrices and input and state vectors were implemented as indexed arrays in dynamically allocated memory. The data format was 32-bit floating point numbers (1 CPS = 4*4 CBS = 16 CBS). The programs ran on two computers: a common Pentium based computer and a Silicon Graphics Inc. Onyx multiprocessor machine. The Pentium has a 90 MHz clock rate and 16 MB RAM. MS-DOS 6.22 and Borland C++ 4.0 (all optimization switches on) were used. The SGI Onyx has two R4400 CPUs and two R4010 FPUs, 64 MB RAM, and runs at a 200 MHz clock rate. The Irix 5.3 operating system and the Irix C-compiler 3.19 (optimization switch -O3) were used. 64-bit arithmetic is used, independently from the data format specified by the program. This makes benchmarking a highly problem dependent task, and the SGI data listed below must be interpreted with care. The CBS and CBSW benchmarks obtained by the Onyx using one and two processors are included in the results listed below. All interprocessor communication was implemented with the shared memory/semaphore model using System V system calls.

5 Benchmark Results

The benchmarking results as a function of the network size, i.e. number of neurons, are listed in Figure 3 and Figure 4. One, two, and four NNPs, as well as the SGI Onyx were benchmarked using continuous Hopfield networks of various sizes. Note that one or two NNPs do not provide sufficient weight memory for 256 neurons. On the other hand, at least four neurons are needed for the implementation on four parallel NNPs. Figure 3 shows the million CBS (MCBS) measured, and that even a single NNP outperforms the SGI Onyx. The MCBS-curves illustrate that multiple NNPs in parallel do not significantly increase perfor-

mance of smaller networks, due to the computational overhead involved. The same holds for the SGI Onyx, where program control overhead (e.g. for-loops) and communication overhead for two processors (e.g. semaphores) limit the CBS for smaller networks. For larger networks the maximum realizable, sustainable performance (RSP) is being approached on all systems. The maximum RSP of the NNP in a single and multiprocessor environment almost reaches the theoretical peak performance, which is a result of its optimization for neural processing tasks (Table 1). The theoretical peak performance for the SGI Onyx was considered one FLOP per connection, which gives $1 \text{ CPS} = 4 \times 4 \text{ CBS} = 16 \text{ CBS}$ for 32-bit floating point numbers.

Table 1: Performances in million connection-bytes-per-second (MCBS) for a CHN.

System	Theoretical peak performance	Realizable sustainable peak performance	% of theoretical peak performance
1 NNP	140	134.6 (128 neurons)	96.1
2 NNPs	280	268.4 (128 neurons)	95.8
4 NNPs	560	547.8 (256 neurons)	97.8
Pentium 90	(unknown)	28.5 (256 neurons)	(unknown)
SGI Onyx (1 Processor)	1600 ^a	131.1 (256 neurons)	8.2
SGI Onyx (2 Processors)	3200 ^a	184.4 (256 neurons)	5.8

a. Based on SGI's measure of 100 MFLOPS (1 processor), and 200 MFLOPS (2 processors).

Figure 4 shows the observed million CBSW (MCBSW) of the CHN implementations. Note that a small number of neuron per NNP slows down the overall network performance. This is due to the relatively high computational overhead compared to the actual neural processing on each NNP. Figure 5 is a different representation of the results shown in Figure 3. It shows the performance increase (MCBS) as a function of the number of NNPs. The NNP specification predicts a linear increase until the data throughput saturates at $(f + 1)/4$ NNPs, where f is the average fan-in into each neuron. Figure 5 shows that this linear specification is met (note the logarithmic scale).

6 Concluding Remarks

Two benchmarks for neural network hardware were defined and applied to a digital neural network processor. The results were compared to that of a common Pentium based computer and to that of a high-performance Silicon Graphics multiprocessor workstation. The comparison demonstrates that specialized neural network hardware can be efficient, small in size and available at reasonable cost, and that it has great potential for many real world applications.

7 References

- [1] C. Cox, K. Mathia and R. Saeks, "Learning Flight Control and LoFLYTE", *Proc. Wescon/95*, San Francisco, pp. 720-723, November 1995.
- [2] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sciences USA*, Vol. 81, pp. 3088-3092, 1984.
- [3] E. van Keulen, S. Colak, H. Withagen, and H. Hegt, "Neural Network Hardware Performance Criteria," *Proc. IEEE Int. Conf. Neural Networks*, Orlando/Florida, Vol. 3, pp. 1885-1888, 1994.
- [4] K. Mathia and K. Priddy, "Real-Time Geometrical Approximation of Flexible Structures Using Neural Networks," *Proc. IEEE Int. Conf. Syst., Man and Cyb.*, Vancouver, B.C., Vol. 3, pp. 2099-2102, October 1995.
- [5] G. Rogers, J. Solka, J. Ellis, and H. Szu, "A Neurocomputing Benchmark for Digital Computers," *Proc. SIMTEC-WNN '92*, Clear Lake, Texas, pp. 425-430, November 1992.

- [6] E. Rumelhart and J.L. McClelland (eds.), *Parallel Distributed Processing*, Vol. 1, Chapter 8, MIT Press, Cambridge/Massachusetts, 1986.
- [7] R. Saeks, K. Priddy, K. Schnieder, and S. Stowell, "On the Design of an MIMD Neural Network Processor", *Proc. World Cong. Neural Networks '94*, San Diego/California, pp. 590-595, June 1994.

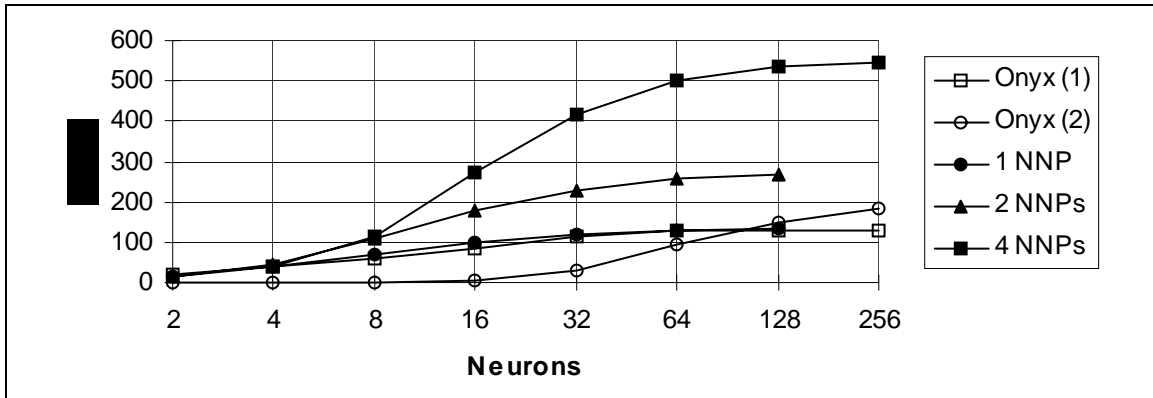


Figure 3 MCBS as a function of network size (number of neurons).

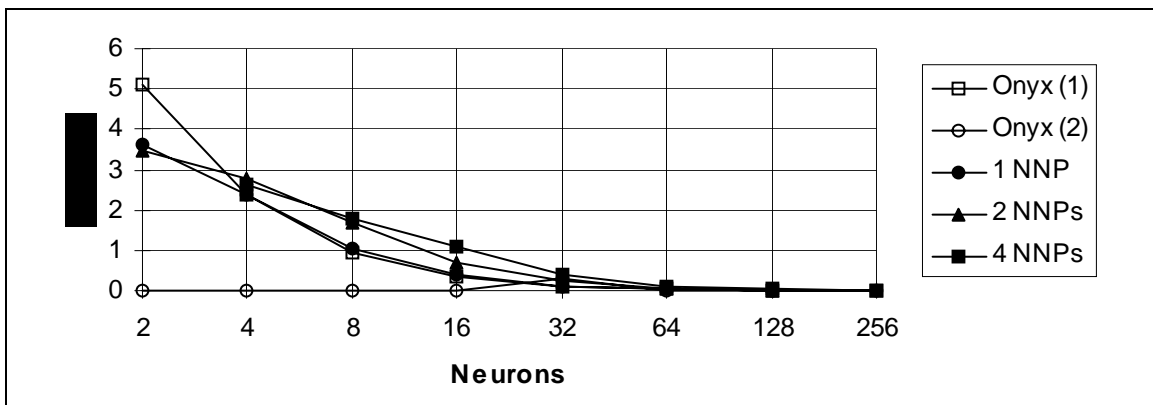


Figure 4 MCBSW as a function of network size (number of neurons).

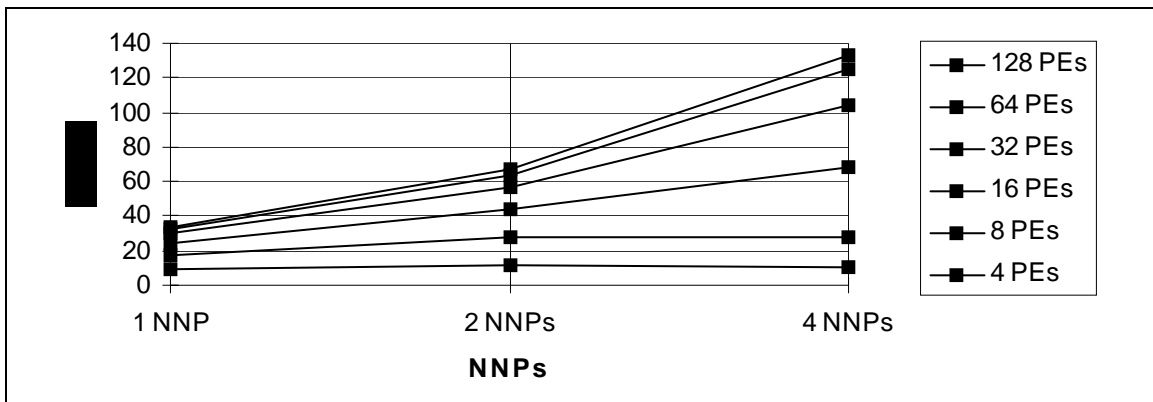


Figure 5 MCBS as a function of the number of parallel NNPs (PE = processing element, or neuron).