A VLSI Interconnect Strategy for Biologically Inspired Artificial Neural Networks

James L. Bailey

B.S., Oregon State University, 1973
M.S., Brigham Young University, 1986

A dissertation presented to the faculty of the
Oregon Graduate Institute of Science & Technology
in partial fulfillment of the requirement for the degree
Doctor of Philosophy
in Computer Science & Engineering

January 1993

The dissertation "A VLSI Interconnect Strategy for Biologically Inspired Artificial Neural Networks" by Jim Bailey has been examined and approved by the following Examination Committee:

Dan Hammerstrom, Thesis Advisor
Associate Professor

David Maier
Professor

Todd K. Leen
Assistant Professor

Alan J. Coppola
Adjunct Assistant Professor

Acknowledgements

I would like to acknowledge the Office of Naval Research and Tektronix, Inc. for their financial support. My manager at Tektronix, Steve Palmquist, has helped me in many ways while finishing this dissertation.

During the time for this research I have been supported and encouraged by my thesis advisor Dan Hammerstrom and I appreciate all of his efforts. I have also been assisted by numerous friends and fellow students who have reviewed my writings and suggested improvements and new research directions.

I would like to acknowledge the support and encouragment I have received over the years from my wife Janice. Sabrina, Sean, and Stephanie, my children, have helped me stay focused and have provided a reason to continue when I needed one. To them I offer a special thank you.

Table of Contents

Table of Figures

Table of Tables

Abstract

Current interconnection architectures are not adequate to support the communications requirements of Artificial Neural Networks based upon Neurophysiological models. For ANN models, direct implementation has a cost in required area which scales as the cube of the number of connections per node. A system of one million nodes, each connected to one thousand others, would require 40 times as much silicon if implemented as a series of direct wires as it would with multiplexed interconnect.

This thesis further shows that using a broadcast communication paradigm improves cost-performance results by at least a factor of $N^{1/2}$ over point-to-point. Broadcast also allows for fewer messages, shorter messages, easier implementation, and can be implemented either with a physical broadcast interconnection structure or as a virtual model imposed upon a point-to-point physical interconnection architecture. This research lays the theoretical foundations for development of broadcast as an effective communications paradigm for ANN implementations.

In support of the primary results of this thesis, methods of analyzing target models and interconnection architectures are developed. Other proposed interconnection architectures are compared with the proposed broadcast solutions and shown to be inadequate for these network models.

In addition, results are given which show the effectiveness of broadcast for implementing ANN models ranging from artificial models such as feed-forward layered networks to olfactory piriform cortex to mammalian hippocampus to abstract neurophysiological models. It is shown that a complete rat hippocampus could be implemented in a single eight inch wafer with a .3 micron technology.

CHAPTER 1

Introduction

The current research interest in Artificial Neural Networks (ANNs) began in the mid 1980s with publications by Hopfield [Hop82], Hinton [Hin84], Sejnowski [SeR86], and Rumelhart and McClelland [RuM86]. Since then, the annual International Joint Conference on Neural Networks as well as dedicated journals have been filled with the results of research on training algorithms, applications, and implementation techniques. During the same time, knowledge of the organization and functionality of the central nervous system has increased, as shown by publications of Shepherd [ShB79], White [Whi89], Lynch [Lyn86], and Koch and Segov [KoS89]. Granger *et alia* [GAL89, GAA90], Bower [Bow90a], Van Essen and Anderson [VaA90], Rolls [Rol90], and others have merged these two lines of research to show how models based on observed neurophysiology can be abstracted and applied to problems such as pattern matching and input characterization.

This dissertation presents results aimed at solving what we believe is the key problem facing large scale ANN implementations, whether based on mathematical or neurophysiological models. This problem is how to provide for the immense amount of communication that is an inherent part of all of these models. In the mammalian brain, for example, more than 10^8 neurons are each communicating with 10^3 to 10^5 other neurons. Because the focus of this work is on the communications problem, questions of which algorithms to use or what problems to solve are not addressed.

The large degree of interconnection within ANN models is well matched to a broadcast representation, rather than a message-by-message point-to-point model. Broadcast

models are difficult to physically implement in VLSI silicon. This dissertation proposes solutions to the implementation problem and shows how they support large ANN models, especially ones based on neurophysiology, better than any proposed alternative.

This dissertation is organized into three logical sections. First, a means is presented for concise specification and understanding of the communications requirements of ANN models. Next, a new family of broadcast interconnection architectures is proposed and compared to existing point-to-point architectures. Finally, some rules are given to select and tune a specific architecture for a proposed ANN based upon its connectivity requirements and size. As a demonstration of the design decisions required and the methods proposed, a prototype architecture is given for a stylized VLSI hippocampus.

1.1. Background

Several names have been used for the computational networks discussed in this dissertation. They include *Parallel Distributed Processing* (PDP), *Connectionist Networks* (CNs), and *Artificial Neural Networks* (ANNs). The acronym ANN is used in this work to indicate the goal of implementing systems abstracted from biological neural networks.

The research here focuses on the communication and interconnection problems of supporting messages among thousands or hundreds of thousands of processing nodes with a high percentage of interconnections. The degree of interconnectivity varies, by model, from 5% to 100%. Each node, thus, typically communicates with hundreds to thousands of other nodes. The computational model utilizes a set of simple processing nodes, each of which applies a weighting function to its inputs and calculates a new output state. Then the nodes each *fire* and broadcast their updated state to all connected nodes, much as a neuron in the brain transmits across its synapses. The interconnection architecture must provide sufficient bandwidth while allowing each node to maintain its input and output connections to other

nodes. In addition, required temporal behavior of the computational model must be supported.

ANN models have been applied to many problems. Neurophysiological models have shown the ability to solve pattern matching problems [HoT89, Rol89], process visual and audio data [Mea86, MeM], and differentiate between classes of inputs [Bow90b, LGL89]. Mathematically derived ANN models are being used for similar classes of problems. As simulations and applications grow in size, specialized hardware is required to provide adequate real-time performance. While some commercial applications, such as commodity trading [BeW91], can use systems the size and cost of workstations, others require smaller, less expensive custom designs. Examples of this latter group of applications include speech or handwriting recognition and process control. Unfortunately, while silicon can readily provide the necessary analog or digital computational capacity, it does not readily support the required connectivity between computational nodes.

Many people are working toward physical implementations of ANNs. Some solutions have proven cost and performance effective for small networks. As demonstrated in Chapter 3, current designs are not able to adequately support large network models without severe performance or cost penalties. Computational models, such as Back Propagation [RuM86], ART [CaG87], and Boltzmann Mechanics [HSA84], perform well with small networks, but as network size increases, the time required to learn a set of patterns and select the appropriate one increases faster than the rate of increase in network size. As shown by Hinton, [Hin87] for example, the learning performance of Back Propagation is $O(w^3)$, for w the number of weights in the system. A new approach, both at the physical implementation and algorithmic computational levels, is needed for applications that require significantly more nodes than current models or architectures can manage.

Biological neural networks have characteristics that make them good candidates for solving large, difficult problems. They converge to solutions, despite being four or five orders of magnitude larger than any current ANN implementations [ShB79]. They respond in real time, even though the underlying biochemical devices are orders of magnitude slower than electronic devices [FeB82]. Learning takes place in biological systems in the presence of large amounts of noise. For these reasons, the possibility of implementing neurophysiologically based ANNs is of interest for solving problems where current approaches are not adequate. Unfortunately, the size of neurophysiological ANN models makes their implementation via traditional methods unacceptably slow and costly.

Before implementations based on biological models can be architected, a method of concisely describing the connectivity between neurons is required. These descriptions must abstract the necessary temporal and topological behavior of the network without including unnecessary detail of the mechanisms involved. Shepherd [ShB79, She90], White [Whi89], Van Essen [Van85], and Braitenberg [Bra89] are all neuro-biologists who have studied and described the circuitry of biological neural networks. In this dissertation, their descriptions of neural connectivity are further abstracted to interconnection graphs.

CMOS VLSI silicon is a well understood medium with readily available cost and performance parameters. For these reasons, it was chosen as the hypothetical implementation technology for this research. Also, although a variety of alternative semiconductors are currently in use or under research development, none provide a topology significantly different from silicon. Multi-layer designs and three dimensional layout do not change the conclusions presented here, because of the exponential growth rate of the area needed for interconnect. Moving to alternatives that are further out, such as optical, electro-optical, or biologically grown systems, might change some of the conclusions. However, since these are not yet commercially viable alternatives, they were not considered.

In comparing alternative interconnection architectures and picking an optimal solution, the metrics used in this dissertation include cost, area, speed, complexity, and fault tolerance. Cost and area are combined because the major determinant of yield, and thus production cost, is the area of the minimum computational modules. The speed of a network is determined by the time required for messages to reach their destinations. Complexity of design is of concern because added complexity increases the required area, test time and difficulty, and leads to reduced fault tolerance. Fault tolerance is critical because VLSI silicon is a faulty medium and any large system must overcome those faults that occur. Fault tolerant capability can either be designed in as part of the computational model or provided transparently beneath it by the physical implementation.

1.2. Thesis Organization

This introductory chapter is followed by Chapter 2, which provides the definitions and metrics used in the remainder of the dissertation.

Chapter 3 shows that current interconnection architectures cannot meet either the desired response times or the communication requirements of large ANN models. The work of Dally is referenced because of its significance for classical interconnection architectures for general purpose parallel computers. While his conclusions about the benefits of reducing network dimensionality remain valid for ANN models, the implications for implementation decisions are different, because of the increased volume of message distribution.

Chapter 4 contains a method of classification of ANN models based on their interconnect patterns. This method is then used to describe both mathematically and neurophysiologically derived ANN models. Three families of ANN models are described based on their interconnection graphs. The implications of these graphs for implementation is also given.

Chapter 5 provides an overview of current research into multiprocessor interconnection architectures. For each of several several selected architectures, its ability to support ANN networks is described. The use of broadcast structures is shown as a good solution for improving the cost-performance ratio of implementations. A family of broadcast architectures is then introduced.

Chapter 6 contains an analysis of message counts and delays for four interconnection architectures from Chapter 5: grid, torus, physical broadcast, and virtual broadcast. The area required for connectivity and the time required for message propagation are calculated in terms of the number of messages entering the system from external sources or being generated within the system.

Chapter 7 is a summary and generalization of the results of the earlier sections. It is divided into two major sections: a set of rules or heuristics for selecting the right interconnection architecture for a given model, and descriptions of some possible implementations showing how the rules could be applied.

Chapter 8 provides a summary of the results and the conclusions of this research. In addition, possible topics for further research are suggested.

CHAPTER 2

Definitions and Terminology

This chapter defines the terms used in the remainder of this dissertation. Communication requirements are described as graphs, so a brief introduction to graph theory is provided. Metrics used to compare alternative implementations or models are also defined.

2.1. Graph Definitions

This work focuses on a key problem of large ANN implementations: their communication or interconnection requirements. The computation performed by a network node and the ANN model itself is ignored, except where it defines the communication requirements.

Ignoring the details of the ANN models permits this research to concentrate on interconnect requirements. Where it is necessary to make assumptions about a computational model, the model used is a simple sum of products node that multiplies its inputs by weights and sums them before executing a threshold function (see Figure 2.1). More complex nodes require additional time or area to function, so these analyses overestimate communication costs as a percentage of the required time and area and derive limits more stringent than necessary.

The choice of computational model affects the architecture by defining minimum requirements for physical processors and setting upper bounds on the number of ANN nodes that can be effectively emulated by each physical processor. The major constraint on the interconnection architecture is the number and physical placement of the inputs to or outputs from each node. This interconnection pattern can be represented as a *directed graph* with CNs

as *nodes* and the connections between them as *edges* in the graph.

Definition 2.1: A *directed graph* $G(N,E)$ is a finite set of nodes N and a set of edges E, where $E \subseteq N \times N$. Thus, each element of E is an ordered pair (i,j) where $i,j \in N$. Edges are directed so the edge (i,j) is different from the edge (j,i) and is said to *originate* with node i and *terminate* with node j.

For an example of a directed graph, see Figure 2.2, where the set of nodes is $\{a,b,c,d\}$ and the set of edges is $\{(a,b),(a,c),(a,d),(b,d),(c,a)\}$. The arrow heads on the edges indicate their direction. For example, (a,d) originates with node a and terminates with node d. If

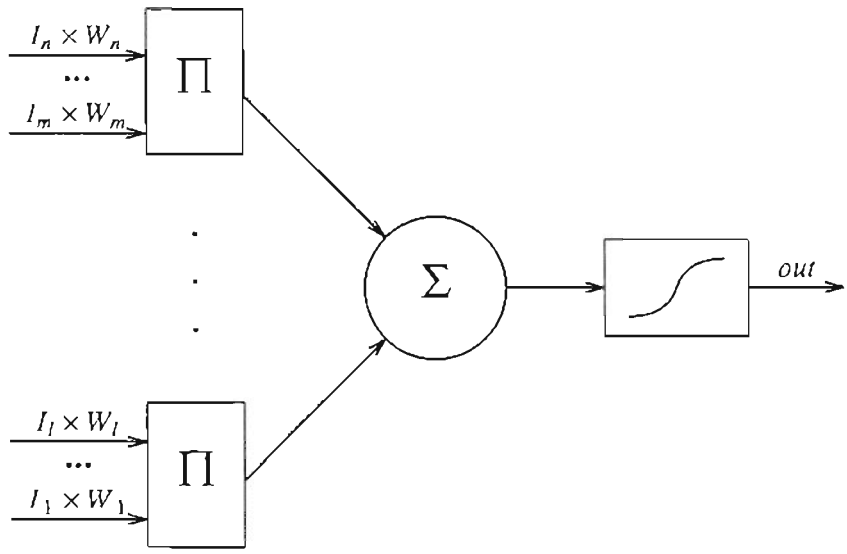


Figure 2.1 Sum of Products Node

Each input, I_x, is multiplied by the corresponding weight, W_x. The resulting products ($\prod$) are summed ($\sum$) before a sigmoid function is applied.

edges are present in both directions between a pair of nodes, both need to be explicitly shown, as are (a,c) and (c,a). This paper will use the term *connection* as a synonym for *edge* where necessary to conform with ANN notational conventions.

Definition 2.2: A *connection matrix* is a *NxN* boolean matrix that describes the edges of a corresponding graph and where column indices represent the originating nodes and the row indices represent the terminating nodes in the related directed graph. A "0" in a given position indicates the absence of a connection. A "1" indicates its presence.

Figure 2.3 shows the connection matrix for the graph of Figure 2.2.

Although connection matrices are defined here as boolean matrices, in general it is possible to have connection matrices with values other than "0" and "1". For such non-

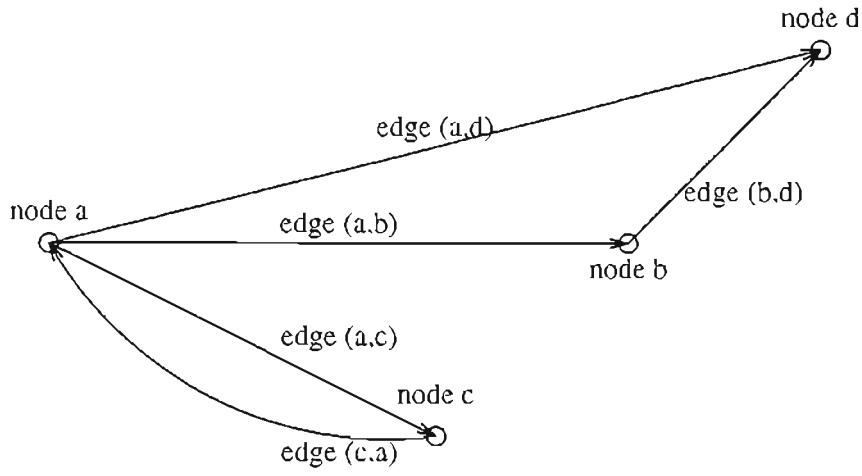


Figure 2.2 A Directed Graph

The direction of the arrows indicate the direction of information flow from source node to destination node.

boolean matrices, the magnitude of an entry indicates the value of some associated attribute of the edge.

Definition 2.3: A *communication graph* (*c-graph*) is the directed graph of an ANN where the CNs are represented by nodes and the connections between them are represented by corresponding edges.

In a c-graph, the existence of the edge (p,q) implies the output of p is an input to q. The edge (p,p) is not explicitly included in the c-graph, even if the past state of node p is an input to its update function, since the graph is intended to show only interprocessor communication. Because of the restriction on edges of the type (p,p), the diagonal entries of connection matrices are all zero. In general, c-graphs are asymmetrical; the existence of edge (p,q) does not imply the existence of edge (q,p). Asymmetrical ANN models include NeoCognitron [FM83], Back-Propagation [RuM86] in non-learning mode, and neurophysiologically derived systems where synapses are uni-directional and neurons may or may not contact their inputs. Asymmetry is not universal however; symmetry is required for the proof of convergence of Hopfield networks [Hop82].

---

	a	b	c	d
a	0	1	1	1
b	0	0	0	1
c	1	0	0	0
d	0	0	0	0

Figure 2.3 A Connection Matrix

This matrix shows the connections of the directed graph of Figure 2.2.

Definition 2.4: A *physical graph* (*p-graph*) is a directed graph representing the physical processor interconnect of a system. The nodes represent physical processing nodes (PNs) and the edges represent communication channels or connections between them.

The communication channel between any two PNs, p and q, can be bi-directional. In a p-graph, such a bi-directional connection is represented by explicitly including both edges (p,q) and (q,p). The convention of directed edges was adopted for p-graphs because a physical connection often consists of two physically separate conductors with different potentials for faults. In addition, more general architectural models may be represented with such a convention. As the intent is to only capture interprocessor communication, the existence of local memory in node p, or a connection between two CNs assigned to it, is not represented by the edge (p,p).

Definition 2.5: A *path* in a graph G is a sequence of edges $e_1, e_2, \cdots, e_m$, for $e_i \in E_G$ and $m \geq 1$, such that if $e_r=(i,j)$ and $e_{r+1}=(k,l)$ then $j=k$, for $1 \leq r < m$. The beginning and end of a path are the nodes n_1 and n_m of N_G respectively, where $e_1=(n_1,j)$ and $e_n=(k,n_m)$. Paths may not include cycles. That is, no two edges of a path may have the same beginning or ending node.

Definition 2.6: The *length*, $l(p)$, of path p is the number of edges p contains.

Definition 2.7: The *empty path* for node n, $\varepsilon(n)$, is defined as the zero length path starting and ending with node n.

Definition 2.8: The *degree* of a node n is the total number of edges incident to n. The number of edges leaving n is its *fan-out* (divergence) and the number of edges entering n is its *fan-in* (convergence). The degree of a graph is the average of the degrees of the nodes within it.

2.2. Mappings of CNs to PNs

Given a c-graph C and a p-graph P, it is necessary to perform some mapping or assignment of the CNs of graph C to the PNs of P. This section defines possible ways of describing such mappings and metrics for evaluating them.

The computation of the ANN algorithm can either be done on a single processor or divided among multiple processors. On a single-processor computer, improved performance can be achieved by modifying the ANN algorithm or using a faster processor. These areas of study are outside the focus of this work and are not considered further.

The alternative considered in this research is to share the computational load among multiple processors. Two ways to partition the computation, both of which are based on the c-graph, are *matrix splitting* and *graph embedding*. Depending on the ANN model, it is possible to define other partitionings, such as having one PN execute some function f, pass the results to a second PN which performs g, and on to a third which does h, where the computation in each CN is $h \cdot g \cdot f(i)$ on input i. The design and effectiveness of such an approach is dependent on the particular ANN model or algorithm. The option of dividing the computation of a single CN among multiple PNs is considered under the matrix splitting model.

Matrix splitting is a method of dividing the communication and computation of the ANN among the PNs based on the connection matrix. In Figure 2.4, three divisions of a connection matrix are illustrated. For example, if the matrix is divided by columns, as in Figure 2.4 A, the PN assigned to column i would perform the same calculation for all CNs, using i as an input. If the matrix was divided by rows, as in Figure 2.4 B, the PN with row j would execute the algorithm of CN j with all inputs it receives. The case of row-based partitioning, for multiple rows per PN, is covered later under *graph embedding*. Finally, Figure 2.4 C shows a hybrid approach where each nodes performs a calculation on multiple inputs for each of

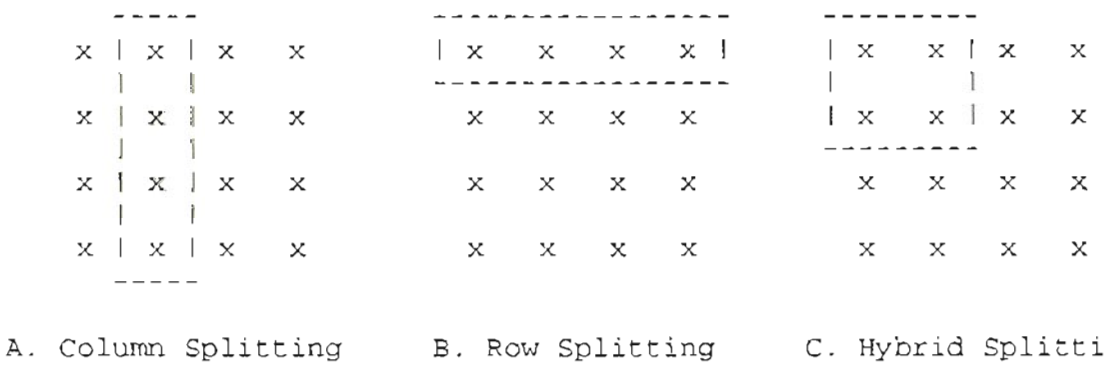


Figure 2.4 Alternative Ways of Performing Matrix Splitting

several nodes. Hybrid splitting differs from the other two alternatives since each CN and its inputs is split among multiple PNs. In such a split mapping, the intermediate results from each PN would be forwarded to secondary nodes, such as those on the diagonal, for completion of the computation.

Several problems exist with matrix splitting of types A and C. If each row, or set of rows, is divided so that no PN contains an entire row, the potential exists for data incoherency. With multiple copies of each CN, different copies could be in different states at the same time. Reducing data incoherency requires synchronizing all copies to the same state before any updating calculations are performed. A second problem is decreased fault tolerance. The loss of a PN deletes the computation of an entire section of the matrix and all rows and columns containing the missing section are disrupted.

Matrix splitting is primarily of value in situations where PNs are powerful enough to update multiple CN states easily; where the system is synchronous to reduce the probability of

incoherent states; and where interprocessor communication costs are expensive, so fewer messages are preferable, even if longer. Examples of message-based multi-processor systems that would be appropriate for this approach are the Connection Machine [Hil85], and the Intel iPSC [Jac91], with synchronizing messages broadcast system-wide between node update cycles.

Another circumstance where matrix splitting would be of potential value is when the matrix is sparse with locally dense regions. Two examples of c-graphs with such matrices are multi-layer feed-forward networks and the visual processing regions of the brain as shown in Figure 4.2. Given such a locally dense matrix, assigning a processor to each dense region could reduce the interprocessor communication requirements.

Graph embedding is a second way to partition a c-graph. In this approach, one or more CNs are assigned to each PN and each c-graph edge is mapped to a corresponding p-graph *path*. When a set of CNs has more *external* co-incident edges than the equivalent PN, either a group of PNs can be considered as a unit to provide the needed connectivity or p-graph edges can be multiplexed. All intermediate PNs on a given path provide message forwarding.

Definition 2.9: A *mapping* $M{:}C \rightarrow P$, where C is a c-graph and P is a p-graph, is a function of $N_c \rightarrow N_p$ and $E_c \rightarrow P_p$ (P_p is the set of all paths in P) such that for $(i,j) \in E_c$ then $M((i,j))$ is a path with beginning $M(i)$ and ending $M(j)$. Note that if $M((i,j))$ is the empty path $\varepsilon(n)$, then $M(i) = M(j) = n$. That is, the edge (i,j) is best represented internally to PN (n).

Figure 2.5 shows the example mapping of a c-graph C to a p-graph P. In this example, node a of c-graph C is connected to five other nodes so path (5,6),(6,3) of p-graph P is used to connect nodes a and f, while all the other edges of C are mapped directly to edges of

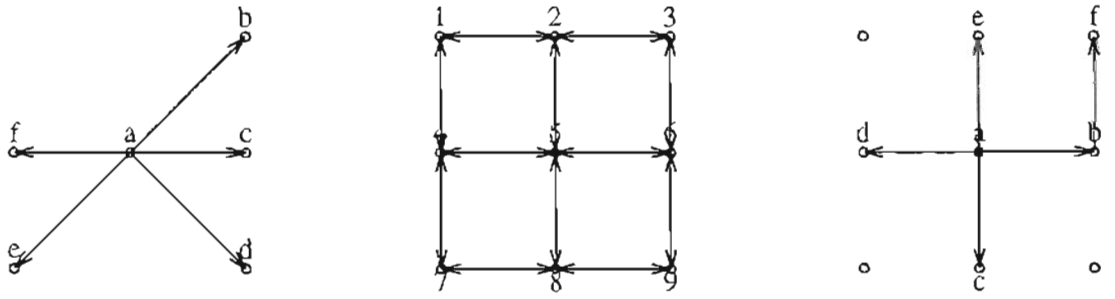


Figure 2.5 Example Mapping M of C-Graph C to P-Graph P

P. Also, note that edge (5,6) is used twice to provide the required connections (a,b) and (a,f).

Definition 2.10: The *dilation* of a mapping M, of c-graph C to p-graph P, is the length of the paths in P that edges of C are mapped to by M. The *maximum dilation* is $\max(l(M(e)))$ for $e \in E_C$. The *average dilation* is $\frac{1}{|E|}\sum_{e \in E_C} l(M(e))$, where $|E|$ is the cardinality of E.

Definition 2.11: The *communication cost*, with regards to some cost function k, of a mapping M: C -> P is $\sum_{e \in E_C} k(M(e))$ where k assigns a non-negative value to each path of P.

If the cost function k is defined such that each edge traversed is assigned a cost of 1, then the mapping in Figure 2.5 has a total cost of 6. This is the mapping with the lowest possible communication cost given the two graphs c and p and with each CN mapped to a distinct PN. An example of a mapping with a higher cost under the same function k is M' that maps node a to 1, b to 7, c to 8, d to 9, e to 6, and f to 3. M' also maps edge (a,b) to path (1,4),(4,7); (a,c) to path (1,4),(4,7),(7,8); (a,d) to path (1,2),(2,5),(5,8),(8,9); (a,e) to path (1,2),(2,3),(3,6); and (a,f) to path (1,2),(2,3). M' would have a cost of 2 + 3 + 4 + 3 + 2, or 14, more than twice the cost of the mapping M.

Each mapping also has a *computational cost*. This cost is determined by the amount of computation each PN performs for its assigned CNs. The computational capabilities of each PN place an upper bound on the number of CNs which can be mapped to a single PN. In the absence of any computational constraints, the optimal mapping, from a communication cost perspective, would be to assign all CNs to a single PN.

The *optimal mapping*, as used here, is the one with the smallest communication cost that satisfies the computational constraints of the PNs. This definition of an optimal mapping allows PNs of the p-graph to have unequal computational tasks.

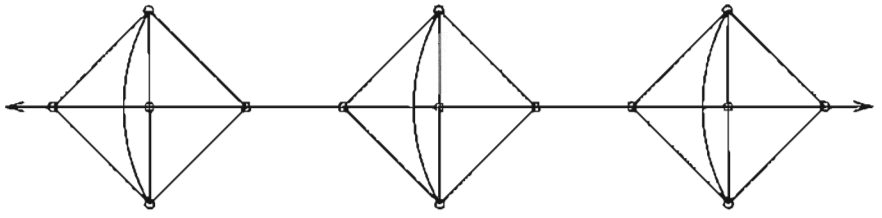
2.3. Graph Measures

The graph measures defined in this section are used in Chapter 4 in describing example ANN models. The Chapter 4 descriptions include the degree of communication of each class of c-graph and the resulting difficulty of finding mappings to potential p-graphs which meet cost-performance goals.

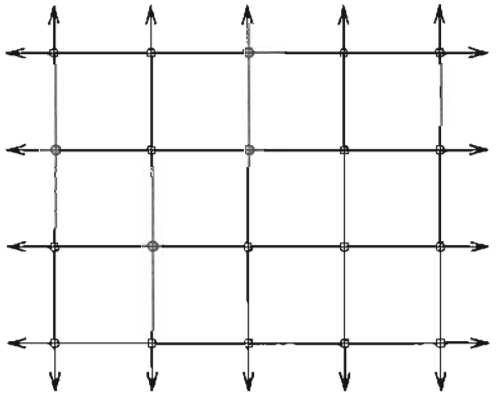
Definition 2.12: The *density* of graph G is the ratio of the number of edges in G to the number of edges in a *fully connected* graph with the same number of nodes or $\frac{|E|}{|N|^2-|N|}$. $|N|^2 - |N|$ is the denominator of this formula instead of $|N|^2$ since edges of the form (p,p) are not allowed.

For example, the graph of Figures 2.2 and 2.3 has a density of 5 / 12.

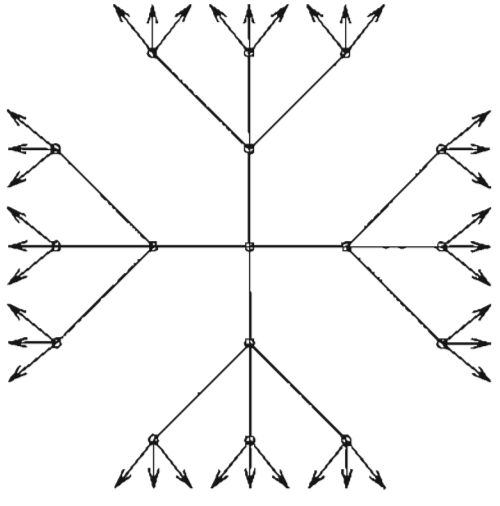
Figure 2.6 shows three different graphs of degree four. In this figure, each line represents a pair of directed edges. Although all three graphs have the same degree, they can not be mapped easily onto the same architecture without dilation, given a one-to-one limit on the number of CNs per PN. A graph of type A maps readily to a one-dimensional, nearest-neighbor connected p-graph, with no paths of length greater than three. Graphs of type A and B map readily to a two-dimensional grid of nearest-neighbor connected nodes, with no paths



A. Constant Rate of Increase of Reachability



B. Linear Rate of Increase of Reachability



C. Exponential Rate of Increase of Reachability

Figure 2.6 Three Graphs with Different Reachability Functions

of length greater than two. Graphs of type *C* have a dilation factor that increases with size

when mapped to such a one-dimensional or two-dimensional p-graph, or even to a three-dimensional version of a nearest-neighbor p-graph.

The three graphs of Figure 2.6 can be characterized by the rate at which new nodes can be reached by paths of increasing length starting at any initial point. This rate is the *reachability function* of the graphs as defined in Definition 2.13. For graph A, the number of new nodes encountered at each step follows either the sequence 4,2,6,2,6,2,6,... or 4,4,4,..., depending on the choice of starting node. In both cases, the average number of new nodes that can be reached by paths of any given length from an arbitrary starting node is 4. Graph B follows the function 4,8,12,16,... or $4n$. For graph C, the reachability function is 4,12,36,108,... or $4\times3^{n-1}$. Thus, these graphs are labeled as constant, linear, or exponential in Figure 2.6, since the rate of new connections divided by fan-out is respectively 1, n, and 3^{n-1}.

Definition 2.13: The *reachability function* of a node q, $r_q(l)$, is the number of unique nodes that can be reached by paths of length l beginning at q. The reachability function R of a graph G is the average of the node reachability functions over all nodes in the graph or $R(l) = \frac{1}{|N_G|} \sum_{q \in N_G} r_q(l)$.

A *regular graph* is one with the same topology in all of its parts. For such graphs $R(l) = r_q(l)$ for all $q \in N$. For example, rectangular graphs, such as B in Figure 2.6, have $R(l) = 4l$. Graph A of Figure 2.6 is not regular, but graph C is and $R_C(l) = 4 \times 3^{l-1}$. In both of these cases, the lack of regularity at the edge of the graph is ignored to provide simplicity in the formulas.

Reachability provides a useful means of characterizing graphs. When mapping a c-graph to a p-graph, the crucial determinant of communication performance is the dilation caused by the mapping. If all messages resulting from a particular computation must reach their destinations before the next computation can be performed, even the presence of a single

long path slows down the system.

Another measure which predicts how well a c-graph maps to a p-graph is their relative degree of *locality*. The concept of locality is intended to capture the intuitive idea of having all connections for a given node be to *nearby* nodes. It is related to the usage in operating systems theory where the locality of memory references of a program is used to determine how well it will perform in a given virtual memory computer system [DeJ73]. Here, the concept of locality is used to capture how well the interconnect requirements of a c-graph will be supported by a given p-graph.

Definition 2.14: For a graph G and $S \subset N_G$, the *destination set* of S is D_S where $D_S = \{n \in N_G : m \in S \text{ and } (m,n) \in E_G\}$.

For any c-graph C, it is possible to construct a series of *locality graphs* L_p where p represents some partitioning of the nodes of C. Consider some partitioning of the nodes of C into subgraphs. For each such subgraph define a node of L. Then there exists an edge (i,j) of L if i and j represent subgraphs of C and the subgraph associated with j contains nodes within the destination sets of the nodes in the subgraph associated with i. Note that some graph theoreticians refer to L as a homeomorphism of C and the partitions of C as its kernels.

Definition 2.15: A graph G exhibits *locality* if there exists a partitioning P of N_G into non-trivial subsets such that L_P is sparse.

Although locality is not a computable metric, it is possible to say that one graph has a higher degree of locality than another if the first is readily partitionable into sets with a sparser resulting locality graph. The c-graphs in sections 4.3.1 and 4.3.2 demonstrate such a comparative level of locality.

The relative reachability and locality of the c-graph being mapped and the p-graph to which it is mapped place a lower limit on the mapping cost. When every PN has fewer

outgoing edges than the CNs mapped to it, some edges must be multiplexed with a resultant penalty in performance. Topologies with a larger average degree show better speed and performance in supporting arbitrary c-graphs. The way in which c-graph characteristics limit implementation and mapping optimizations is developed more fully in Chapter 5.

2.4. Networks and Layers

Although the functionality of an ANN, other than node and edge activity, is not of major interest here, its logical structure can provide insights into communication requirements. Many artificial neural network models are organized into groups of CNs called *layers*.

A *layer* is a subset of a c-graph. The nodes within a layer are either computationally or physically identifiable in some manner as belonging to a distinct set. They may be connected by some type of intralayer connections, for example, lateral inhibition. A layer performs a transform on its inputs before providing them as inputs to the next layer or layers.

Once a graph has been partitioned into layers, its edges may be classified as intralayer or inter-layer.

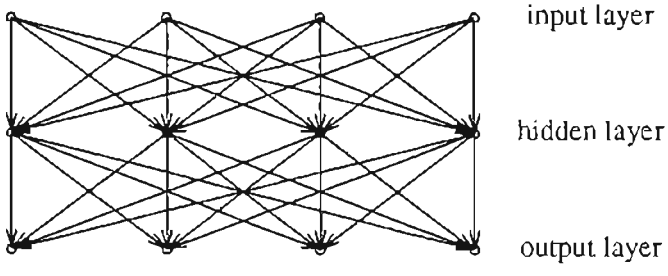


Figure 2.7 Three Layer Feed-Forward Network

Definition 2.16: An *intra-layer* connection is an edge of a c-graph between nodes in the same layer. An *inter-layer* connection is an edge of a c-graph between nodes in different layers. It is said to be between the two layers containing its endpoints.

In a multiple-layer model, the *input layer* receives external inputs to the system. The *output layer* transmits results to the external world. Intermediate or *hidden layers* are not externally visible. Figure 2.7 shows the organization of a typical *feed-forward* network consisting of three layers. A *feed-forward* system is one where inputs enter one layer, pass to its successor, and continue on with no cycles or *feed-back* connections.

In most of the models of this thesis, external inputs and outputs are ignored. It is assumed that some means of connecting to each node can be provided without affecting the bandwidth requirements of the system. In an actual physical implementation, input and output requirements are a problem because of packaging limits.

CHAPTER 3

The Need for Multiplexed Interconnect

This chapter presents two fundamental results of this dissertation: the area required for directly wiring connections grows as the cube of the number of interconnected nodes, and the cost savings from multiplexing or sharing interconnect more than compensates in total cost-performance for the resulting communication delays. In addition, it explains why special architectures are needed for cost-effective implementations of ANN models.

3.1. Background

One problem with the effectiveness of using ANNs to solve real-world problems has been poor cost-performance. Due to performance problems, network sizes have, in most cases, been restricted to a few hundred nodes. In addition to limiting the size of problems, the lack of appropriate hardware has kept researchers from empirically investigating the effects of increasing size on network algorithms.

A solution to the performance part of this problem is the use of enhanced or specialized computer systems. The benchmark program *NETalk*, a system that learned to generate speech from text [SeR86], originally required more than twelve hours of VAX 11-780 time for the necessary learning trials. With the use of a specialized computer system, the Adaptive Solutions CNAPS, the learning time is reduced to six seconds [Ham92b].

Computational solutions proposed have included supercomputers, networks of simple microprocessors, vector co-processor boards, and specialized systems based on fast numeric processors. Unfortunately, steadily increasing network sizes require larger and faster emula-

tors. The introduction of models based on neurophysiological research only serves to accelerate this trend. In addition, the development of commercial applications requires inexpensive emulation systems.

In designing custom VLSI circuits to support large ANN networks, a range of possible implementations is available. Very fast, dedicated processors tuned for the required computations can be used alone or in small groups to process an entire network. This *network processor model* is the approach used by SAIC and HNC in their *neurocomputer* products. At the other end of the spectrum are systems directly implementing ANNs in silicon with a processor, usually analog, for each *neuron*. This approach is used by researchers at Bell Labs, Synaptics, Lincoln Labs, JPL, Cal Tech, and elsewhere. An example of an intermediate solution is the CNAPS system from Adaptive Solutions, where multiple PNs can each emulate one or more CNs as required.

The network processor model, while effective for small systems, fails to support large networks adequately. As the number of nodes, N, in a network increases, the computational and memory requirements increase at least linearly. The communication requirements increase from order N to order N^2. The use of general purpose multiprocessor systems, such as the BBN Butterfly, the Intel Hypercube, or the Connection Machine, are limited to research situations or very expensive applications because of their high cost.

Eliminating the network processor model and the use of general purpose multiprocessor systems, because of cost and performance issues, leaves multiprocessor systems specifically designed for ANN emulation as the only cost effective approach that may be able to support large networks. Two possible alternatives for such systems exist: the traditional *direct implementation* approach and the *virtual implementation* proposed here. In a direct implementation, there is no connection multiplexing, therefore, the c-graph and p-graph are

topologically equivalent. In a virtual implementation, this restriction is removed.

Direct implementation matches the connectivity requirements of highly localized primary sensory processing as exemplified by the visual processing chips of Mead *et alia* [MeM]. Direct implementation is only effective for ANN models with a *limited connection radius*, because of the $O(n^3)$ area requirement to be shown in Result 3.1. A limited connection radius model is any model which can be mapped with minimal constant dilation to a p-graph where each PN is only connected to physically adjacent PNs. Unfortunately, direct implementation does not support the greater degree of interconnect required by neurophysiological and associative ANN models.

The virtual implementation approach proposed in this research multiplexes connections and potentially allows for the mapping of multiple CNs to a single PN. In addition to reducing the problem of increasing network size and supporting networks with a high degree of interconnect, this approach also delivers better cost-performance ratios for some ANN models.

3.2. Scaling Problem

A general heuristic for communication systems design is that systems are more flexible when the binding of scarce resources is delayed as late as possible, preferably dynamically during execution. The earlier a partitioning of resources is performed, the more likely circumstances arise that invalidate it, overloading one section and underutilizing another. This concept of delayed partitioning is critical to the architectures presented here, since message traffic varies from instant to instant depending on which nodes fire and in what sequence.

In his thesis [Dal86], Dally applied this law to interconnection architectures. He showed better performance of systems with a lower dimensionality of interconnect and concluded that given a fixed area available for interconnect, it is better to use the area to increase

the communication between adjacent processors and have them forward messages to further nodes than to dedicate silicon real estate to direct connections between non-adjacent processors.

In the cerebral cortex of the brain, the volume dedicated to interconnect exceeds the volume dedicated to computation. All computation is performed within a thin layer of neurons on the surface of the brain, while much of the inner volume is taken up by white connective or communication tissue. Shepherd [ShB79] estimates the area of the human cerebral cortex as $1.2 \times 10^3\ cm^2$ and that computation requires less than 1/4 of this area, with the remainder primarily dedicated to communication. Given an average thickness of 5 mm, the cortex volume is approximately $6 \times 10^5\ mm^3$ with $1.5 \times 10^5\ mm^3$ for computation. The total volume of the adult human brain approximates a sphere of diameter 12 *cm*, that has volume $9 \times 10^5\ mm^3$. Thus, the portion that is primarily interconnect is about 6 times as large as the portion that is computation.

In designing an artificial implementation of neural networks in silicon, three alternatives exist for setting up the interconnect. The first is to make exactly those connections which are necessary for the functioning of the network. This approach is useful for specialized, regular, limited connectivity models. For large, non-regular ANN models, it requires a complete prior understanding of all possible configurations, does not support multiple models, provides no recovery from damaged connections, and is impossible until tools exist for laying out large numbers of pseudo-random wires.

The second alternative is to provide all potentially required connections. Providing all possible connections is the current approach for many analog chips and works well for designs of limited size. Due to excessive area requirements, providing all connections is not feasible for many models as shown later in this chapter.

The third alternative is to multiplex metal lines, providing *virtual* connectivity. Each physical wire of the system is shared by one or more ANN connections as necessary. Virtual connectivity reduces the number of required wires, but introduces communication delays. This approach is used in conventional multicomputer systems as well as custom digital chips such as CNAPS.

Currently both of the first two alternatives are being used for direct VLSI implementations. The first, implementing the required connections only, approach is used with models that have short-length internode connections. That is, each node communicates only with nodes within a small physical radius. Required connections only is the model used by Mead *et alia* in building the *Silicon Retina* chip [MeM]. The other, more general, model provides for all possible connections. This second approach is being used by Jackel *et alia* in the Intel ETANN associative network chips [GrV87, JHG], and by Alspector and Allen in their Boltzman Machine chip [AlA87]. The two approaches result from differences in the ANN algorithms being implemented.

These two approaches perform differently as the number of nodes in the network is increased. If each PN is connected only to its nearest neighbors, then the size of the system can be increased with few problems. Each row of nodes added to an existing system only adds communication costs to nearby nodes. Those nodes further than the radius of connection from the new nodes are not affected as illustrated by imagining a planar layout of PNs with new PNs added along one edge. New connections are made only between the added PNs and the previous border PNs, which were not fully connected before. For this reason, systems with short radius connections do not have scaling problems when network size increases.

Unfortunately, the total interconnect model does not scale well. Figure 3.1 shows how a Hopfield Network chip may be laid out with an $O(N^2)$ area requirement for N connec-

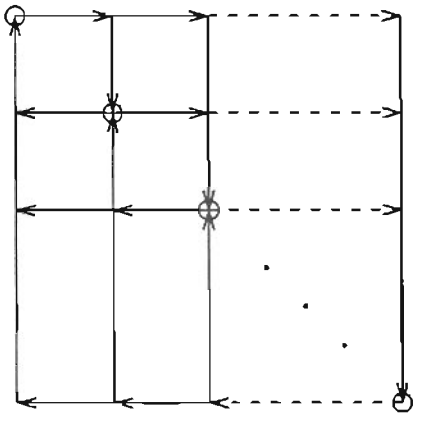


Figure 3.1 A Possible Hopfield Network Layout
The circles in this figure represent PNs placed along the diagonal. The horizontal arrows are outgoing connections from each PN and the vertical arrows are incoming, summing connections.

tions. In this figure, horizontal lines transmit new output states and vertical wires sum them. Thus, one vertical line is *shared* among a number of connections. If the computation performed is not a sum of inputs, this approach does not work. Consider a PN computing a function with two or more classes of inputs: it requires a vertical summing wire for each class. When the computational model requires N discrete inputs, N vertical wires for each computing node are needed. Such a model requires $O(N^3)$ area for interconnect. Due to the $O(N^3)$ growth rate, global interconnect is not feasible for large networks. Other possible layouts have a similar problem as is shown in the next paragraphs.

Result 3.1 shows how the area required for connections grows with less than global interconnect. The area required for interconnect can be calculated by determining the number of wires that cross an arbitrary dividing line between two adjacent nodes, multiplying this number by the area required for a single wire, and dividing by the number of separate layers available for implementing a specific connection:

Area = wire count × wire width × wire length / implementation layer count

Since wire width and implementation layer count are constants for a given technology, the area required for connections is dependent on the number of connections or wires between two adjacent nodes.

Result 3.1: The number of connections crossing a vertical bisector between two horizontally adjacent PNs is $(d^3 + 2d)/3$ when all nodes distance d apart are connected. This derivation assumes an infinite rectangular grid of PNs and a routing of connections with vertical runs traversed before horizontal runs.

Argument:

The approach is to sum all the connections from the line of processors with the same "y" coordinate, then add in all connections from processors with greater "x" coordinates and greater "y" coordinates. Next add in all connections from processors with greater "x" coordinates and lessor "y" coordinates. Finally, sources with lessor "x" coordinates mirror those described above, so the final sum is multiplied by two.

A formal restatement is:

$$C = 2\left(\sum_{i=1}^{\infty}\sum_{j=0}^{\infty} f(i+j) + 2\sum_{i=1}^{\infty}\sum_{j=0}^{\infty}\sum_{k=1}^{\infty} f(i+j+k)\right)$$

This reduces to:

$$C = 2\sum_{i=0}^{\infty} i\,(f(i) + 1/2\,(i+1)f(i+1))$$

where C is the number of connections and $f(x)$ gives the probability of a connection between two PNs *Manhattan* distance x apart. Defining $f(x)$ to allow connections between all nodes within a fixed distance d of either of the two nodes.

$$f(x) = 1 \quad (for\ x \le d)$$

$$f(x) = 0 \quad (otherwise)$$

yields

$$C_d = (d^3 + \frac{2d)}{3}$$

In Result 3.1, the value of d is a function of the number of nodes interconnected. This result shows the area required for interconnect is $O(r^3)$ where r is the degree of the graph.

The third alternative for interconnect, shared connections between nodes, is illustrated by the following example computation of areas required. It is more fully covered in Chapter 5.

To see the possible gains in layout density due to multiplexing of interconnect, compare the area and response times of two equivalent systems of 10^6 CNs with 10^3 connections for each, for a total of 10^9 connections. The *direct* implementation has one CN per PN and every connection between CNs is explicitly present as a physical wire. The *virtual* implementation has 16 CNs per PN and connections are made only to nearest neighbors with messages relayed to further destinations.

Both implementations assume all PNs are physically laid out in a hexagonal grid. In addition, wire widths and interwire spacing are each 1 μ, three levels of metal are available, and effective memory cell area is 50 μ^2. The processor area required for computation is ignored. One bit of state is transmitted between nodes and weights are four bits. These specific values are not critical; rather a measure of relative performance is intended.

Placing the PNs in a hexagonal pattern reduces both the bandwidth required and the diameter of the interconnect region over the use of a square pattern because of the increase in the number of adjacent nodes. Each PN both transmits to and receives from its six neighbors. Any messages that need to be routed to a non-adjacent PN are relayed by intermediate nodes using a routing algorithm based on the destination address of the message.

Communication delays in a direct connect system are due solely to the length of the interconnect lines. Message length is minimal, because each node can readily identify the source of a message for weight assignment and no addresses are needed for routing. The delay in transmitting a one bit message is equal to the time for the signal to transit the wire.

The best case from an implementation viewpoint is a system with every PN connected to the nearest 10^3 PNs. For a hexagonal grid layout of PNs, the minimal radius of such a fully interconnected group of PNs is 18 since

$$\sum_{i=0}^{n} 6i \geq 1000$$

has a minimal solution of $n = 18$.

The total processor area is then $\pi r^2 \times 10^6$, for a processor radius r; the total interconnect area is $2/3 \times l \times 10^9$, where l is the average length of an interprocessor connection; and $l = 37/3 \times 2r$, since l is the length of an average connection spanning a circular region of radius 18 PN diameters. Solving this set of equations yields $r = 5.2 \times 10^3 \mu$, PN area of $8.6 \times 10^7 \mu^2$ and a total chip area of $8.6 \times 10^{13} \mu^2$ or 86 square meters. The area required for memory is only 2×10^{11} μ^2, showing how the interconnect requirements dwarf the processor requirements. The worst case delay is over a wire $18 \times 2r$ μ long with a delay of 10^{-2} *nsec per* μ, or 1.9×10^3 *nsec*, and the average delay is 1.3×10^3 *nsec*.

This analysis shows direct interconnection is not feasible for silicon implementations of large associative networks with even moderate connectivity requirements. The next paragraphs present a similar calculation for the area requirements and speed performance of a system with shared interconnect.

Assuming sixteen CNs in each PN, each CN requires 10^3 weights, 16 local addresses and 984 global addresses, given an optimal mapping. Using the routing information as the address, a global address requires 19 bits, because each of five intermediate PNs requires three bits to select

the routing direction and the destination PN needs four bits to pick the correct CN. A local address is 4 bits.

The source address in each message is used to determine which weight to apply. Each CN requires two sets of addresses, one for destinations and one for weight selection. The total memory required is thus 41520 *bits per CN*. This predicts a PN area of 3.3×10^7 μ^2. The complete system would require 2.1×10^{12} μ^2, 2.1 square meters. This is one-fortieth the area of the direct implementation. Each PN only communicates with adjacent PNs so total wire area is 7×10^4 PNs $\times$ 3 wires each $\times$ wire length of $6 \times 10^3\mu \times 1\mu$ wide or $1.3 \times 10^8\mu^2$.

A message sent between two PNs consists of the routing information and one bit of state, for a total of 20 bits. The expected path length can be calculated by solving the equation:

$$\sum_{i=1}^{n} 6i \geq (1000 - 16) / 16$$

The smallest such integral n is 5, so the maximum number of intervening links expected is 5 and the average is 3.7, assuming PNs are uniformly distributed within the area of a circle with radius 5. All connections are to nearest neighbors, so communication latency is the time required to relay the message through intermediate PNs. PN diameter is is $\sqrt{3.3 \times 10^7}$ μ. With a wire transit time of 10^{-2} nsec per μ per bit, this equals a minimal relay time of 57 nsec per node per bit.

A rough approximation of the affect of contention on the system is the number of messages in the system divided by the number of wires available to carry them. Assuming a 10% firing percentage for the CNs and six wires in each direction between adjacent PNs yields a multiplier of 525:

$$\frac{984\ \textit{messages per CN} \times 16\ \textit{CNs per PN} \times 10\%\ \textit{firing rate} \times 62500\ \textit{PNs}}{6\ \textit{wires per PN} \times 62500\ \textit{PNs} / 2}$$

This contention factor greatly underestimates the delays due to contention since it assumes message traffic is uniformly distributed over all wires in the system, both temporally and

physically. Thus, these numbers are order of magnitude approximations only.

The average time for a 20 bit message to reach its destination CN is 1.1×10^6 nsec and the worst case time is 3.0×10^6 nsec This delay is three orders of magnitude slower than in the direct implementation and an order of magnitude slower than biological neural systems.

Both direct and virtual implementation examples suffer from requiring optimal mappings of c-graph to p-graph since otherwise some connections would be to remote PNs. Both also ignore edge discontinuities. They are thus for comparative purposes only and are not indicative of the actual costs involved, other than as order of magnitude guesses. With these caveats, the calculations above show a 40 times improvement in area required, due to the exchange of interconnect area for address storage, and a speed degradation of 10^3, due to contention for the limited number of wires between processors.

This tradeoff is in the right direction, but systems need to be much smaller and faster before actual implementations are feasible. As is shown in Chapter 5, the use of broadcast techniques further reduces the area required by a factor of two and reduces communication delays by an order of magnitude.

3.3. Multiplexing Advantages and Disadvantages

In addition to reducing interconnect area, multiplexing of communication lines provides more flexibility in use. By changing the addresses stored in the PNs, new connections can be made or changed dynamically, while the system is running. This flexibility allows a given physical system to support a variety of different ANN models.

Another benefit of virtual implementations is the possibility for a regular p-graph to support a highly irregular c-graph. This regularity of p-graphs allows for simplified design of the physical system since a small region can be designed and tested and then replicated or *tiled* to form the complete system. In addition to simplifying design, this feature allows a single

architecture to support multiple ANN models and connectivity patterns may be allowed to change during execution.

While the concept of sharing wires does not force the use of digital communication, analog multiplexing is more difficult and would probably not be used in practice. Digital communication has the added advantages of increased robustness and easier debugging. Also, digital communication does not preclude the use of analog computation.

Multiplexing interconnect is not a panacea. It has a variety of problems that need to be solved before very large network emulation systems can be built. These problems include decreased fault tolerance, increased complexity in addressing and routing algorithms, increased power consumption and heat generation, and increased contention problems.

Decreased fault tolerance can be avoided by using multiple instances of structures that might fail. For example, three communication lines might be used with required agreement among any two. Alternatively, a final manufacturing processing step can be used to remove flawed connections and substitute working ones [Col87, RMB86]. Another possible solution is to use intelligent routing to detect and send messages around flaws or only assign CNs to working PNs. Unfortunately, each of these approaches increases the amount of area required and the cost of the final product. May [May88] investigated the reliability of multiplexed implementations and the behavior of the resulting ANN models in the face of a variety of flaws. His results showed most faults occur in the memory of the processors and do not significantly impact system performance or reliability. His results match our expectations, since current defect densities are on the order of five to ten per square inch as compared to millions of transistors in the same area. Since much of the area of the system consists of memory for storage of weights, addresses, and CN states, the loss of a few bits should be tolerable and rarely have a major impact. In addition, defect density varies with circuitry type with metal typically having a low defect rate.

Multiplexing interconnect requires increased complexity in addressing and routing algorithms. When multiple messages are sent across a single communication link, routing information is required to insure they are properly delivered and leads to a packet structure to separate data from addresses. Complexity is added to PNs because it is necessary to decode arriving packets to determine their destinations. A factor offsetting the problems of communication complexity is the speed differential between communication and computation. With chips being clocked at 50 to 100 megahertz, a clock cycle is less than 2×10^{-8} seconds. Matching biological systems requires computational updates less than once every 10^{-4} seconds. The difference, a factor of four orders of magnitude, should provide sufficient time to resolve message traffic problems.

Power consumption and heat generation are increased for a multiplexed system because of the increased complexity of the PN circuitry. Analyses of two possible PN architectures show system requirements are acceptable [Mea91, RuH88]. The added PN complexity is potentially offset by the elimination of the long wires required in a direct implementation.

Contention is a problem due to the potential for increased temporal delay variance and increased overall system delay.

Simulation results [Con87, RuH88] indicate delay repeatability and magnitude are critical in ANN models only when they are large relative to the CN update speed. Since communication times are orders of magnitude faster than computation times, a reasonable level of contention should be supportable. In addition, systems can be designed with increased bandwidth or the use of broadcast to reduce contention.

In conclusion, the use of multiplexed interconnect provides critical benefits in exchange for solvable problems.

CHAPTER 4

Model Networks

This chapter presents several classes of ANN models, including both mathematical and neurophysiological ones. Example c-graphs from these classes will be used in Chapter 7 to illustrate the effectiveness of mappings to alternative p-graph architectures. In addition, these presentations use the definitions of Chapter 2 and illustrate their effectiveness for describing c-graphs.

Mathematical ANN models are presented first because their smaller size and more precise definitions makes them more tractable. Increasing the size of any of the mathematical models to where the architectures of this thesis would be required for effective implementations is not feasible because of issues with learning algorithms and network resolution times. For this reason, their inclusion in this work is primarily illustrative.

The latter half of this chapter covers biological or neurophysiological ANN models. These models are drawn from attempts to describe the functioning of brain subregions. Element counts and communication paths are extrapolations from descriptions of limited biological observations. Further neurophysiological research could refine or change some of the assumptions or descriptions. Information from multiple sources has been combined to reduce the likelihood of later invalidation of these analyses.

4.1. The General Model

As shown in Chapter 2, all connectionist models, whether of artificial or biological origin, are examples of directed graphs called c-graphs. These c-graphs consist of nodes representing the computational units, or neurons, and edges symbolizing the internode connections. Many

ANN models consist of logically grouped subsets with some connection pattern between them. Drawing from the ANN nomenclature, these subgroups of nodes are termed *layers*, as defined in Chapter 2.

4.2. Restricted C-Graphs

Layered graphs can be logically divided into several subclasses by their interconnect requirements. In Figure 2.6 no node in the input layer is directly connected to the output layer. The graph in Figure 2.6 is thus an example of a *restricted inter-layer connection* c-graph. Many of the ANN models proposed to date are c-graphs with restricted inter-layer connections.

Definition 4.1: A *restricted inter-layer connection c-graph*, or *restricted c-graph*, is one where there exists an ordering of the layers such that all connections are between adjacent layers. In addition, any restricted c-graph consists of at least three layers. If a c-graph is not restricted, it is termed an *unrestricted c-graph*.

Whenever referring to restricted c-graphs, the terms "predecessor layer" and "successor layer" refer to the layers preceding and succeeding a given layer according to an ordering under which the graph can be shown to be restricted.

The benefits of restricted c-graphs over unrestricted c-graphs include decreased potential degree, both maximal and average, and decreased network density. Increased ease of implementation is also a benefit of restricting c-graphs, as is further discussed in Chapter 7.

The maximum degree of a node in any layer of a restricted c-graph is $2(i + j + k - 1)$ where i is the number of nodes in the predecessor layer, j is the number of nodes in the layer itself, and k is the number of nodes in the successor layer. This value is calculated by counting the number of nodes to which a connection could be made and multiplying by two, to reflect the potential for both inputs to a node and outputs from it.

In calculating limits on the density of a restricted c-graph, both intra-layer and inter-layer connections must be considered. Since the intra-layer limit on connections is $n\ (n-1)$ for a layer of size n, to maximize the number of intra-layer connections in a graph it is necessary to have a single node in all layers but one, with the remaining nodes concentrated in the final layer. A three layer example of such a graph would have $(N-2)(N-3)$ intra-layer connections for a graph size of N. If the three layers are of equal size, the number of possible connections is $3(N/3)(N/3-1) = N(N-3)/3$, approximately one third of the maximum possible with unequally sized layers.

The inter-layer connection count of a layered graph is maximized when the nodes are divided into two equally sized layers. Such a two-layered graph with full inter-layer connectivity has $2(N/2)^2 = N^2/2$ inter-layer connections. For a network of three layers, this same maximum connection count can be achieved by placing half the nodes in the middle layer with the other half split between its successor and predecessor layers. All networks of more than three layers, or with three layers and a division other than as above, will have fewer inter-layer connections because of the restrictions on nodes within non-adjacent layers not being connected.

Adding these two limits together, for a graph divided into two equal sized layers, yields the expected density of 1 since there could be $\frac{N^2}{2}$ inter-layer connections and $\frac{N^2-2N}{2}$ intra-layer connections for a total of N^2-N connections. Increasing the number of layers decreases the density and degree of the graph. Lower density of the c-graph makes it easier to implement in a planar technology.

4.3. Restricted C-Graph Variations

Given the preceding definition of a restricted c-graph, the next step is to parameterize the possible variations of interconnect and see how different ANN models may be classified. The three classes of inputs within a single layer are: *feed-forward* inputs from the predecessor layer,

feed-back inputs from the successor layer, and *local* or intra-layer connections. In addition to these incoming connections, each node may have the corresponding outgoing connections.

4.3.1. Full Feed-Forward, No Intralayer Connection Networks

The network shown in Figure 2.7 is an example of a restricted c-graph with no intra-layer connections and full feed-forward inter-layer connections. This figure is representative of ANN models such as back propagation [RuM86]. The maximal density of such a network is approximately 1 / 4. This can be demonstrated by considering the division of the network into two layers of N / 2 nodes and an inter-layer connection total of N^2 / 4. Since, by definition, restricted c-graphs must have at least three layers with no direct connections between non-adjacent layers, this is a limit which can only be reached by graphs of three layers with 1/2 of their nodes in the middle layer and the remainder evenly split between the input and output layers, as shown in Section 4.2.

ANN models with restricted c-graphs, full feed-forward inter-layer connections and no intra-layer connections clearly exhibit locality since the partitioning into layers provides destination sets which are unequal. If there are n_i nodes in layer i, then nodes in layer $i-1$ have a fan-out of n_i. Similarly, the reachability function for nodes in layer $i-1$ is $n_i, n_{i+1}, \cdots$. The dilation of such c-graphs under mappings to possible p-graphs is a function of the number of nodes in each layer, as will be shown in Chapter 7.

4.3.2. Partial Feed-Forward, Partial Intralayer Connected Networks

Networks, such as the one shown in Figure 4.1, are examples of c-graphs with partial inter-layer connectivity in a feed-forward direction, no feed-back connections, and partial intra-layer connectivity. In this figure, the inter-layer connections are not shown, but in most models of this type, nodes within a layer which share common inputs and outputs are connected together. Examples of ANN models which fit into this class include Neo-Cognitron [FMI83] and ART

[CaG87]. The density of the interconnect in such graphs is determined by the percentage of the feed-forward and local connections.

If f is the probability of a feed-forward inter-layer connection between two nodes of a three-layered c-graph and c is the corresponding probability for intra-layer connections, the maximal interconnect is $fn_1n_2 + fn_2n_3 + 2cn_1(n_1 - 1) + 2cn_2(n_2 - 1) + 2cn_3(n_3 - 1) < f(\frac{N}{2})^2 + c(N-2)^2$, where n_i is the number of nodes in layer i and N is the total number of nodes in the graph. The inequality results from substituting in the maximal values for each type of connectivity in a three layered network and realizing that both maximums can not exist in the same graph. For large values of N, this provides a graph density in the limit of $\frac{f}{4} + c$.

Under the extension of locality suggested in Chapter 2, it can be seen that networks with partial feed-forward connectivity have a higher degree of locality than those with full feed-forward connectivity, as in Section 4.3.1, since the destination sets are a smaller proportion of the

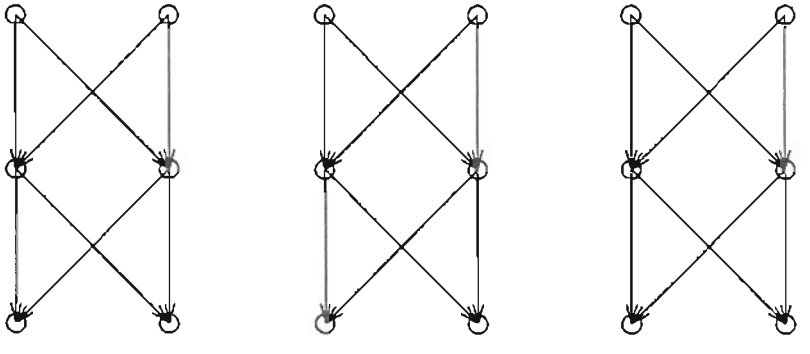


Figure 4.1 Three Layer Partially Connected Feed-Forward Network

graph. Maximal fan-out for a given node in layer i is $fn_i n_{i+1} + cn_i$. Because of their increased locality and decreased degree and fan-out, these networks are easier to implement than those of the previous section.

4.3.3. Restricted C-Graph Summary

Restricting connections in these graphs to between *adjacent* layers simplifies the task of implementing them in planar technologies. The degree of the network is limited by the restriction of potential connections. This restriction directly reduces graph density and the area required for interconnect. The increase in the degree of locality also aids in determining p-graph architectures with sufficient interconnect capability for effective implementations.

4.4. Unrestricted C-Graphs

Unlike ANN models, those derived from neurophysiological research tend to have a more general pattern of interconnect, though a layered structure may be often observed. If no restrictions are made on the possible destinations for connections from a node, then even if the graph appears to be divisible into definite layers, it is possible to have full interconnect, no locality, and a graph density approaching one.

Although biologically derived neural networks do not have restricted c-graphs, they do have a low probability of connections and exhibit locality. Even in regions immediately surrounding the axon of a neuron, not every nearby neuron makes contact. One classical paper on the subject [Utt55], gives probability functions for different axon and dendrite branching patterns and shows an exponential reduction in the probability of a connection between two neurons with increasing distance between them.

Nature has provided neither blueprints nor mathematical statements of the interconnect, or even indicated which paths are important for biologically derived models. The communication paths used in the central nervous system range from the release of chemicals that modify neuron

behavior over a large region to individual synaptic junctions on a single site of a neuron. Researchers have been able to estimate the gross circuitry of some cortical areas using techniques such as neuron staining, observing the efferent impact of lesion damage, tuning the variables of computer simulations until performance matches experimental results, measuring electrical activity with test probes, and intuitive analysis. While detailed information is lacking on individual neuron-to-neuron circuits, enough is known to start drawing conclusions about probable individual connections. Models based upon knowledge of specific cortical areas, such as the olfactory piriform cortex, have shown results that correlate with experimental results on living organisms. Such correlations imply the accuracy of the model at capturing some level of functionality. Although it is not yet possible to be certain about the c-graphs of brain regions, the combination of general knowledge of its connectivity or structure and working abstracted ANN models indicates the potential value of developing p-graph architectures capable of supporting our current understanding.

The selection of biological models presented in this paper is not all inclusive; examples were selected to illustrate the suggested approach. Also, as this thesis is not a study in neurophysiology, biological aspects were simplified. As stated earlier, the computational functionality of models is not considered except where it affects or defines the communication requirements.

4.4.1. Olfactory Piriform Cortex

As pointed out in Shepherd [ShB79], the olfactory cortex is probably the best place to start a study of the cortex. It is the earliest, or most primitive, region of the cortex and has the simplest structure. Granger *et alia* [GAL89, GAA90, LGL89] have developed computer simulations that mimic the olfactory system's ability to generalize upon and differentiate signals. Bower [Bow90a, Bow90b] has also designed a model by reverse engineering the olfactory cortex. His model shows signal patterns similar to those observed in biological studies.

In terms of the definitions of this paper, olfactory piriform cortex can be considered as a system with input via the lateral olfactory tract (LOT) and outputs to the entorhinal cortex and, thus, to the hippocampus. It can be divided into layers where each layer consists of those nodes sharing local connections.

The LOT is comprised of between 5×10^4 to 6×10^4 axons. Estimates place the number of pyramidal cells in piriform cortex higher than the number of LOT inputs, but of the same order of magnitude. Taking numbers from Shepherd [She90], and rounding for simplicity, gives an estimate of 5×10^4 inputs and 1×10^6 processing nodes. Of the processing nodes, half are

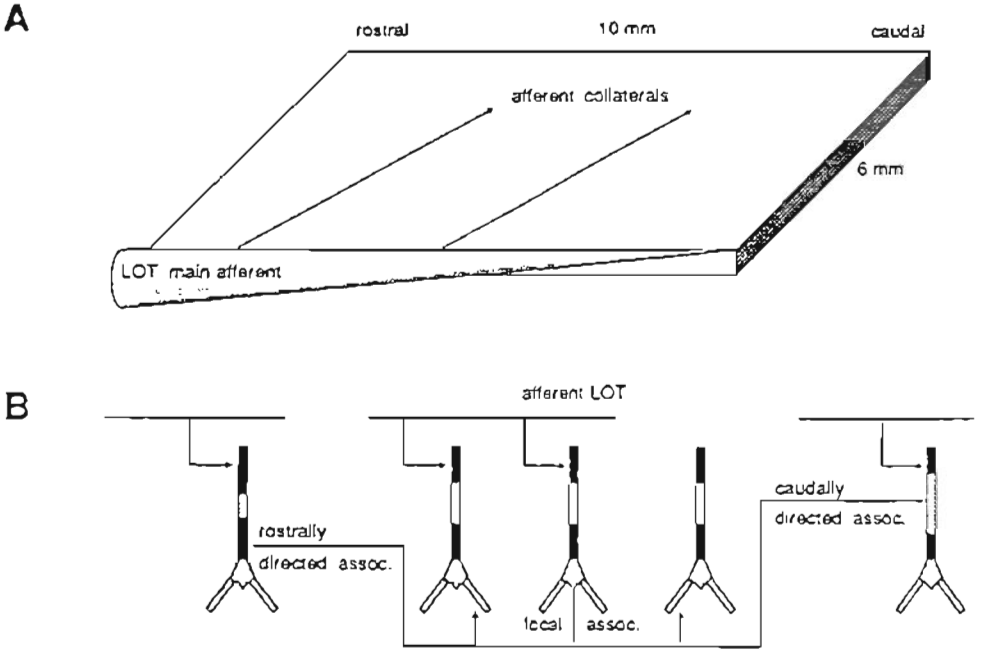


Figure 4.2 Piriform Cortex

(This figure is taken from Bower [Bow90a].)

pyramidal cells that provide outputs from the olfactory cortex, and the other half are locally connected inhibitory and excitatory cells.

The concentration of inputs from the LOT tapers off from rostral to caudal cortex as shown in Figure 4.2-A. The probability of a given input connecting with a particular pyramid cell ranges from about 10% at the maximum to 0.5% at the minimum.

Compensating for the decreased number of external inputs toward the caudal end of the piriform cortex, pyramidal cells axons provide increased numbers of connections. The number of excitatory connections per pyramidal cell stays approximately the same throughout the entire region. In addition to providing associative inputs within the layer itself, these pyramidal axons provide outputs from the piriform cortex to the remainder of the brain.

The other connections present within the piriform cortex are localized to small patches. Each pyramidal cell is connected to a group of nearby stellate cells that generate inhibitory signals. One class of inhibitory signals is local to the patch and provides a *winner-take-all* functionality where only a single pyramidal cell within a patch fires for any given input. The other is rostrally directed and appears to provide for periods of recovery between the pulses of input.

Following the model shown in Figure 4.3, piriform cortex can be considered as a series of layers. Layers closer to the input side receive a higher proportion of their data from the system inputs. Those closer to the output side receive a lower proportion of system inputs and a higher proportion of inputs from previous layers. The probability of a pyramid node in a layer receiving a given system input ranges from a high of 10% to a low of 0.5%. Each pyramid node receives about 10% of the total number of inputs available from both system inputs and other layers.

Each layer consists of 1/2 pyramid nodes and 1/2 non-pyramid nodes. Each non-pyramid node in a layer receives inputs from and transmits output to all pyramid nodes in the layer. Pyramid nodes within a layer are not connected to each other. Pyramid nodes provide output

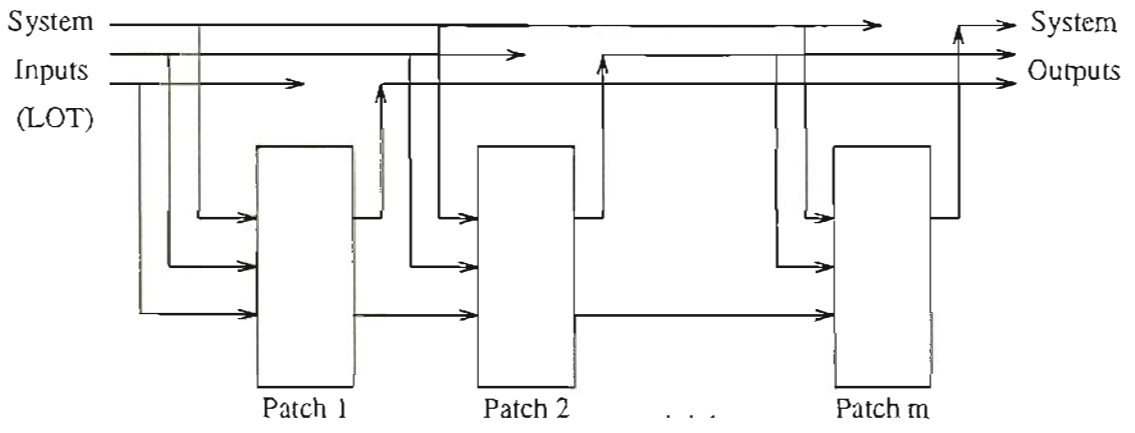


Figure 4.3 Schematic Abstraction of Piriform Cortex

from a layer to successor layers and to the external world.

Such a model demonstrates locality since layers on the input side do not contain destination nodes for nodes in the layers on the output side. Also, non-pyramid cells do not have connections outside their layers. With the restrictions on intra-layer connections, such that pyramid cells are not connected, only 1/2 of the possible intra-layer connections can be made. Given a division into m equal sized layers, $(\frac{N}{2m})^2$ is the total intra-layer connection count. If external inputs are treated as a type of intra-layer connection for simplicity, then $\frac{N}{2}$ of the nodes each receive $\frac{N}{10}$ of the possible intra-layer connections for a total of $\frac{N^2}{20}$ intra-layer connections. With these two components, maximal density is $\frac{10+m}{20m}$ or roughly $\frac{1}{20}$.

4.4.2. Hippocampus

Like the olfactory cortex, the hippocampus is an early, more primitive region of the brain. The model of communication routing presented here is abstracted from the work of Shepherd [ShB79], Rolls [Rol89], and Squire *et alia* [SSA89].

One reason for studying the hippocampus is that it is a primitive region of the brain and hence simpler than other brain regions. Another reason is the variety and power of the functions that have been attributed to it. Experiments with rats have shown that hippocampal neurons fire to indicate spatial location [O'K89]. Other rat studies have demonstrated that hippocampal lesions inhibit the ability to create general associations between different odors [EOW90]. In humans, amnesia has been correlated with hippocampal damage [SSA89]. The common thread in the results sections of these studies has been the conclusion that the hippocampus enables associations between different concepts or experiences. The hippocampus is also believed to provide a critical step in enabling long term memory.

Figure 4.4 shows the regions of the hippocampus and the general flow of data. The major source of input data is the entorhinal cortex. Information flows via the alvear pathway to region CA1 of the hippocampus and via the perforant pathway to region CA3. In addition, the mossy fiber output of the dentate gyrus provides indirect inputs to CA3. The primary input to the dentate gyrus is also the perforant pathway.

Figure 4.4 shows the basic circuitry of the hippocampus. The entorhinal cortex is the primary source of external inputs to it. It is estimated there are 200,000 layer II cells of the entorhinal cortex which project to the hippocampus, each making approximately 18,000 synapses.

The dentate gyrus, or layer 1 of Figure 4.5, consists of roughly 1,000,000 granule cells. Each granule cell receives about 3700 inputs. Estimates for the number of contacts between a given entorhinal and granule cell range from 1 to 10. This results in a maximal probability of

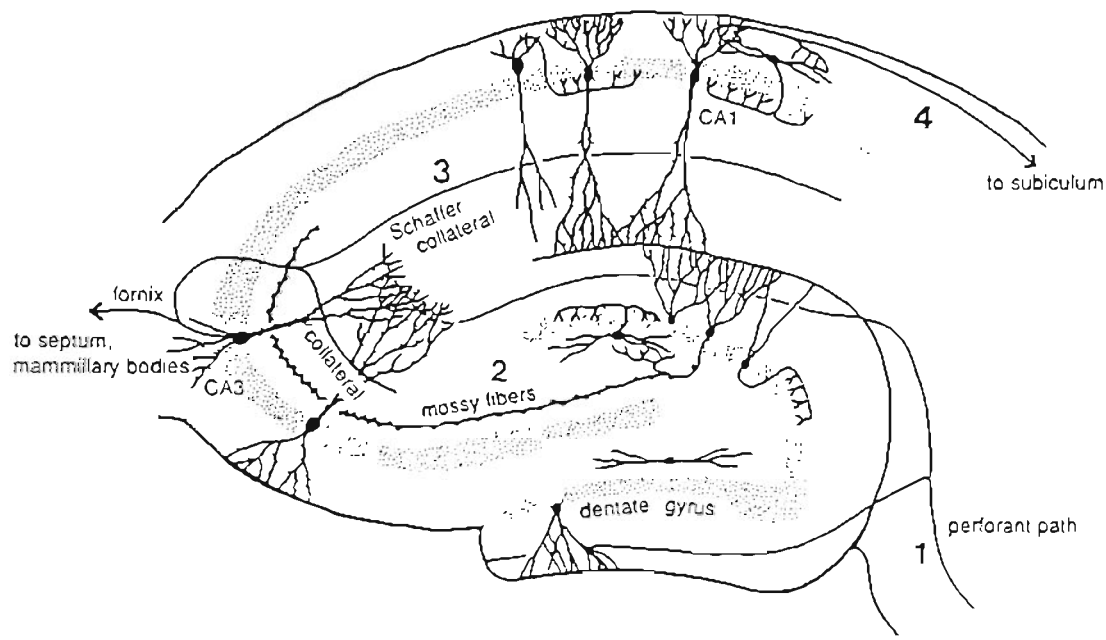


Figure 4.4 Hippocampus

(This figure is taken from Rolls [Rol90].)

contact of a particular granule cell by a given entorhinal cell of .02. Each granule cell receives inputs from 400 to 3700 different entorhinal cells. Separate regions of entorhinal cells project to different regions of layer 1. The precise boundaries and sizes of these regions, both source and destination, are not known.

The layer 1 granule cells have local feedback via 3500 basket cells and 20,000 associative cells. The basket cells provide inhibitory feedback within relatively small patches of about 200 granule cells each. These inhibitory local connections provide a winner-take-all behavior of the granule cells on a patch by patch basis. The associative cells have widely divergent axons

making contact throughout layer 1.

In addition to local connections with the basket and associative cells, granule cells from layer 1 project to layer 2 (CA3) pyramid cells via the mossy fibers. There are about 180,000 layer 2 pyramid cells and each mossy fiber connects to approximately 14 of them, so roughly 80 layer 1 inputs are made to each layer 2 pyramid cell.

Layer 2 has extensive intra-layer connectivity and any layer 2 cell may be connected to any other layer 2 cell. The probability of a given connection is roughly 5%. In addition to the local connections, layer 2 receives input from the entorhinal cell axons, with each layer 2 pyramid cell receiving approximately 23,000 inputs. Rolls postulates that the combination of sparse, strong inputs from the mossy fibers and dense, weak entorhinal inputs results in finely tuned pattern separations. The mossy fibers provide the initial orientation of the separation space and the entorhinal inputs then differentiate between similar mossy inputs. The total number of synapses on each pyramid cell is on the order of 10,000 with 3/4 of them devoted to local or intra-layer connections.

The output of layer 2 is both to layer 3 and to the external world. There are about 1.5 layer 3 cells for each layer 2 pyramid cell. The projection of layer 2 onto layer 3 is topographically organized. Cells in different regions of layer 2 have different destination sets in both layers 2 and 3. Cells on the input side have limited intra-layer connections within layer 2 and project to the most distal extent of layer 3. Cells central to layer 2 have major intra-layer connections that are widely dispersed within the layer and project to nearer portions of layer 3. Finally, the layer 2 cells near the border between the two layers initiate connections that terminate close to their sources in both layers.

Layer 3 does not contain significant numbers of intra-layer connections. It is considered to further classify the output of layer 2.

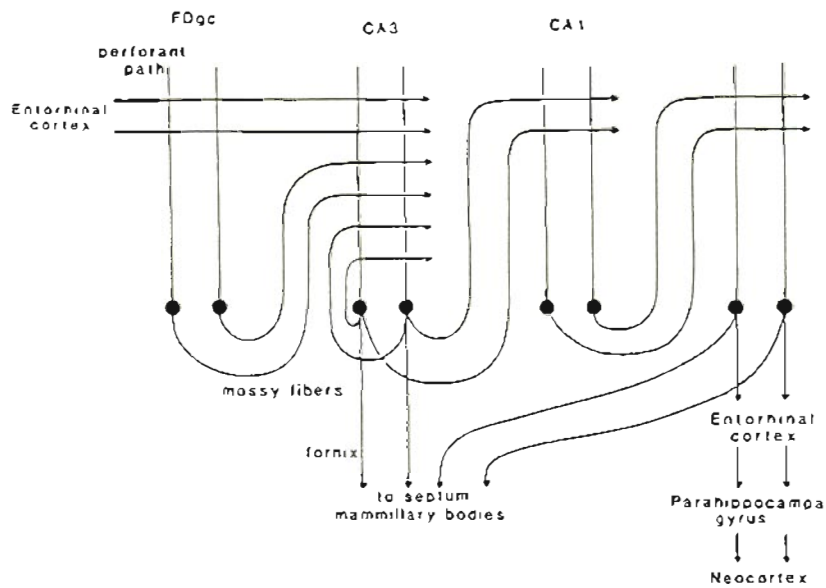


Figure 4.5 Abstract of Hippocampal Connectivity

(This figure is taken from Rolls [Rol90].)

From Figure 4.5 it can be seen there is a clear organization of the hippocampus into layers. It is not strictly a restricted c-graph because both layers 1 and 2 receive external inputs, and layers 2 and 3 both provide external outputs. The lack of connections from layer 2 to layer 1 and from layer 3 to layers 1 and 2 show a degree of locality. The subdivision of the layers into patches, with inter-layer connections being made from nodes in a patch in one layer only to nodes in a single patch in the next layer, provides a greater degree of locality and allows for a more effective implementation as will be shown in Chapter 7. In addition, the sparseness of connectivity of the hippocampus also significantly aids implementation efforts.

There is an inter-layer connectivity rate of 2% between the external source and layer 1. The inter-layer connectivity rate between layers 1 and 2 is approximately 0.01% (14 connections

out of a possible 18,000). From the external source to layer 2, the inter-layer connectivity rate is 10%. Finally, inter-layer connections are again made at a rate of 10% between layers 2 and 3.

Layer 1 is divided into 5000 patches of 200 nodes each with dense interconnect within each patch. In addition, the associative cells provide connections between the patches. The large number of regions leads to a low degree of intra-layer interconnect, less than 0.02%. Layer 2 has an intra-layer connectivity rate of 5% and layer 3 has no intra-layer connections.

Thus the system has a hierarchy of structures with layers containing patches. Combining the various inter-layer and intra-layer connectivity rates yields an overall density of less than 1%. Although nodes individually have high degree, the sharing of sources and destinations within a patch provides a high degree of locality and a relatively low reachability.

4.4.3. Primate Visual Cortex

The third neurobiologically derived model is of primate visual cortex. The numbers and connection organization estimates given here are from Van Essen [Van85, VaA90]. Unfortunately for the purposes of this paper, even less is known about exact cell populations and interconnect patterns for visual cortex than for either piriform cortex or hippocampus.

The functional organization of visual cortex is a series of interconnected regions, or layers, with a general feed-forward information flow through them. The number and organization of the layers differs by species. Figure 4.6 shows the layers and their inter-layer connection patterns for the macaque.

For inter-layer connections, the general pattern is to have each layer broken into a number of smaller patches. Each patch then receives inputs from multiple discrete patches in the originating layers. Each patch also has a dense intra-layer connection pattern. Patches within the same layer are distinct with no connections between them.

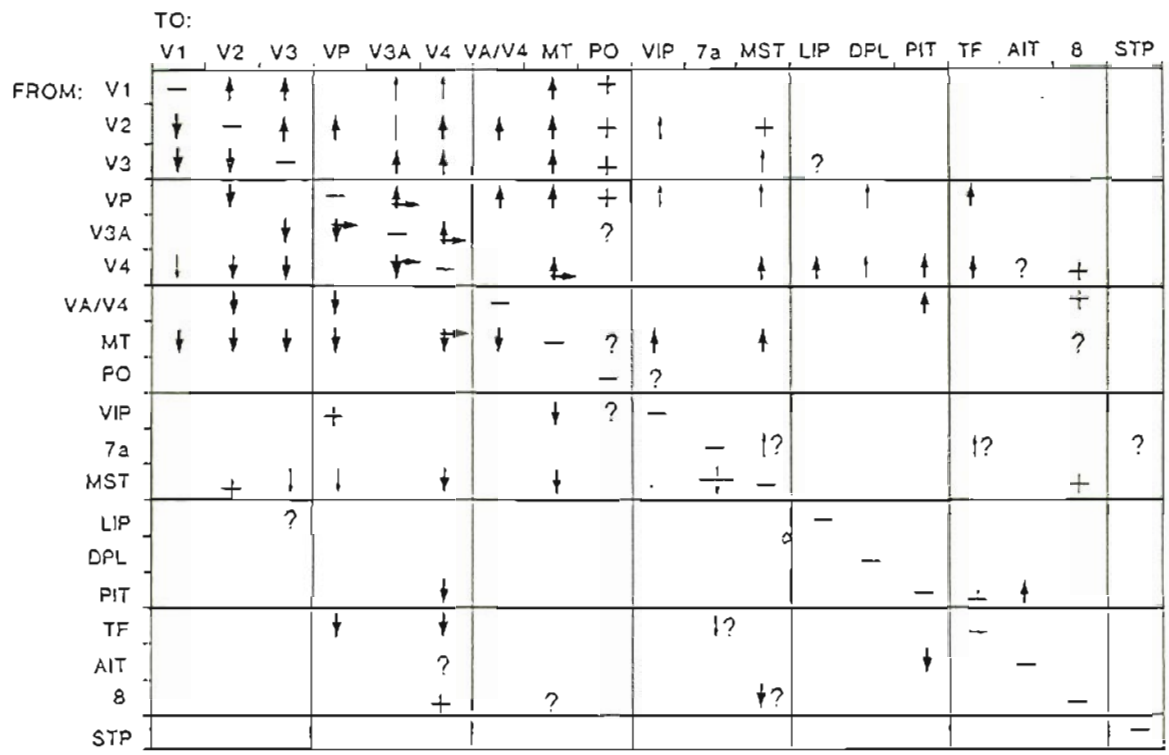


Figure 4.6 Macaque Visual Regions and Their Interconnections

(This figure is taken from Van Essen [Van85].)

The model shows approximately twenty layers with a fairly clear flow pattern through them, although there exist connections in both directions and the layers are not restricted in inter-layer connectivity. Inputs to a layer are to restricted patches within it. Intralayer connections are also restricted to the same patches, but are locally dense.

Given the independence of the subregions of each layer of the system as described above, the potential density and node fan-out is constrained significantly more than would be for an unrestricted system of twenty layers as is shown in the following derivation.

Result 4.1: If a partitioning of each layer of a c-graph G into r distinct sets of nodes $s_1, s_2, \cdots, s_r$ exists such that the source and destination sets are also distinct, maximal fan-out and network density are both reduced by a factor of r.

Argument:

This can be shown for intra-layer connections by considering a layer of size n divided into r equal sized patches. Constrain all intra-layer connections to exist only within a patch. Each node in the layer can then connect to only n/r nodes. The resulting n^2/r intra-layer connections compare to the n^2 possible without constraints.

The inter-layer density is equivalently reduced. Consider l layers as described above. If each node can only connect to l patches of size n/r, then the maximum number of inter-layer connections is $(l-1)n/r$ per node. The system total number of inter-layer connections is $l(l-1)n^2/r$. Adding these two values, results in a density of $1/r$.

In summary, visual cortex can be considered as a c-graph with a fair degree of locality under the partitioning into patches. It is not a restricted c-graph because each layer receives inputs from multiple other layers. This inter-layer connectivity is constrained though, with each layer connecting to a restricted subset of the other layers. The division of the layers into patches significantly reduces the density and fan-out of the system.

4.4.4. Abstract Neurobiological Models

Braitenberg has developed a theoretical interconnection model based on his studies of the architectonics of the brain [Bra89]. While limited direct biological evidence exists to either support or challenge this model, it provides intriguing possibilities for interconnection limits.

For cortical connections, suppose a random interconnection scheme is chosen. Divide the cortex into $N^{1/2}$ patches each containing $N^{1/2}$ cells. In the human, each patch would have approximately 10^5 pyramidal cells and would be about 1 *mm* in diameter. This diameter is about the same size as the spread of a large pyramidal cell, so interconnections between cells of a patch are readily possible. Braitenberg suggests each patch is densely interconnected and receives inputs via one axon from each other patch. Figure 4.7 shows a schematic of such an interconnec-

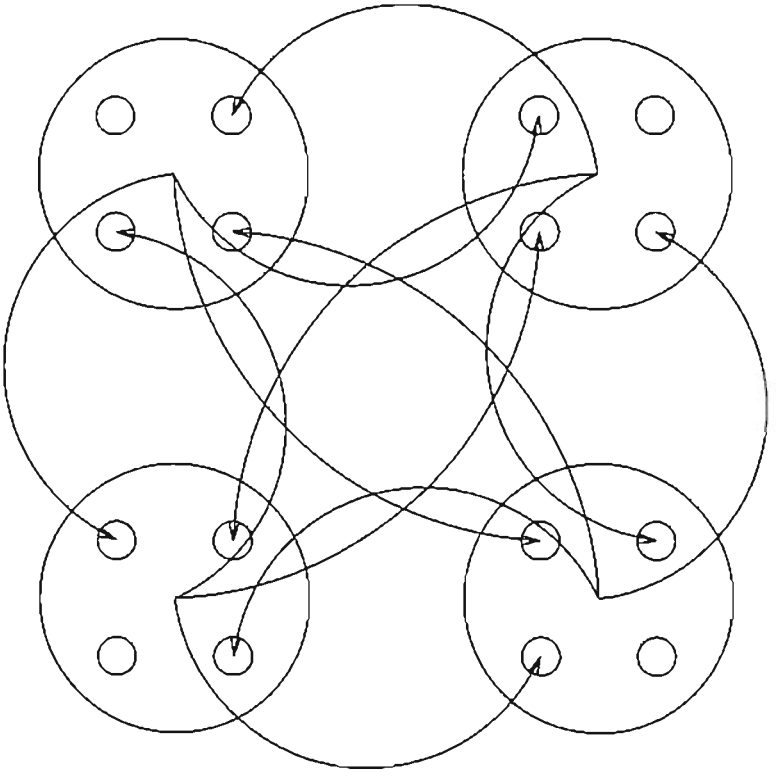


Figure 4.7 Abstract Pseudo-Cortical Connectivity Patterns

tion pattern.

The degree of a typical node would be $(p + 1) N^{1/2}$ where p is the probability of a connection between any two nodes within the same patch. Given a total system size of N cells, the total density possible is $(p + 1)N^{3/2} / N^2$ or $(p + 1) / N^{1/2}$. Since p is a probability, its maximal value is 1, and the limit on the density of a system of this type is $2 / N^{1/2}$. Compared to the potential of N^2 connections for a fully connected system, such a model is sparsely connected. But this system exhibits no locality and any node can update any other node in at most $1 / p$ steps.

Under the natural partitioning of the system into patches, as shown in Figure 4.7, there is no locality since the homeomorphic graph is fully connected. In a biological system the communication can be provided by having a neuron in each patch which has a long axon that diverges at its end and contacts the destination patch neurons. For a silicon based system, this lack of locality is a major implementation problem and requires a specialized interconnection architecture as will be shown in Chapter 7.

4.5. Summary

This chapter has introduced a set of c-graph models that will be used in Chapter 7 to illustrate the effectiveness of alternative p-graph architectures. The concepts of density and locality have been applied to these models to show how their restrictions on connectivity affect the overall number and pattern of connections. Results have been shown which relate density to the number and organization of the layers in a graph and to the presence of patches within layers.

CHAPTER 5

Interconnection Architectures

This chapter reviews historical architectures for multiprocessor systems and shows how different constraints can be used to improve implementations of neurophysiological ANN models. The first section presents several different interconnection architecture, or p-graph, classifications. It also describes how well systems from each class can support ANN implementations. A broadcast interconnection structure is then introduced as a new architectural approach with improved cost-performance ratios. The final section contains a limited selection of architectures suggested for implementation building blocks.

5.1. Architecture Classifications

A number of different classification schemes have been applied to multi-processor computer systems. This section provides a summary and shows how readily different classes of architectures may support ANN implementations.

5.1.1. Instruction and Data Stream Counts

One method of classifying multiprocessor systems is by the number of simultaneous independent processes running on the system and the number of independent data sets being worked on. Of the four possible architectures, single or multiple data by single or multiple instruction, two are of limited interest for ANN implementations. The first, a Single Instruction Single Data (SISD) system is equivalent to a monoprocessor computer with no parallelism. This approach was eliminated in Chapter Three because the lack of parallelism limits the potential for supporting large emulations. The second, a Multiple Instruction Single Data (MISD) system,

may be considered as a cascaded series of processors working on a single information stream. The systolic architectures of Kung *et alia* [AKM85] as implemented in the Intel iWarp system, are also examples of this approach. While providing higher levels of performance than SISD, MISD is still limited by its communications bandwidth.

The two remaining classes, Multiple Instruction Multiple Data (MIMD) and Single Instruction Multiple Data (SIMD), are more applicable to ANN implementations. On an MIMD, system each processor supports separate process and data streams. Communications exist between processes, but they are not tightly linked. Examples of MIMD systems include the NYU Ultra, BBN Butterfly, and Intel iPSC.

In contrast, on SIMD systems, multiple processors are executing the same instruction at the same time, but on different data sets. Commercial examples of SIMD systems are Thinking Machines' Connection Machine and Adaptive Solutions' CNAPS neurocomputer.

SIMD and MIMD have been used for ANN emulations. They also are each feasible for physical implementations of ANN models. MIMD has some added benefits for emulating neurophysiological ANN models. One problem with many SIMD systems is their need for global synchronization, which is hard to provide for large implementations. Also, the MIMD model more closely approximates the organization of the brain where multiple neurons are simultaneously performing different functions on independent data streams.

5.1.2. Communication Type

A second categorization of architectures is into shared memory versus message passing. In shared memory systems, processors communicate by reading from and writing to a common store. In message passing systems, messages are constructed and sent by the communication system to destination processors, where they are unpacked and their information extracted.

5.1.2.1. Shared Memory Systems

With common memory shared between multiple processors, the potential exists for memory access contention and race conditions when two or more processes access a single memory location. One common way of reducing memory and bus contention is to use a memory caching scheme. Caches can be designed to listen for memory writes by other processors that invalidate their contents, and to either discard or replace the old data. To keep common memory contents current, and support bus snooping hardware, writes to the cache need to write through to main memory. Such designs require a single bus for memory references to facilitate cache coherency, so they are limited by memory bus bandwidth in relation to cache size and locality of program or data references.

An alternative to the use of caches to reduce the impact of bus and memory bandwidth limits is used in the NYU Ultra [GGK83]. This design uses a combinatory switching network to route memory references from the processors. The *serialization principal* was developed as part of this project and states "the effect of simultaneous actions by the PE's [processing elements] is as if the actions occurred in some (unspecified) serial order". If two references to the same memory location are detected, they are combined according to type. If both are reads, they are joined into a single read that has two back paths from the detecting switch node. If two writes collide, one is discarded. Since it was a race as to which write would be last, the choice to discard can be made randomly. If a read and a write to the same location are detected, the read returns the value of the write and the write continues on through the routing system to the memory destination. To further reduce memory bandwidth requirements, and to implement the algorithms above, the switching network and memory are augmented by the capability to perform a *fetch and add* operator. This can be used as a form of *test and set* and allows many algorithms to be executed in parallel which would otherwise require the serialization of critical sections. Although this design significantly reduces memory contention and traffic, the performance of the

system is ultimately limited by the bandwidth of the processor to memory switch. Increasing the number of processors to support larger problems causes an increase in both switch delay and complexity, since switch size is $O(n\log n)$ for n processors, and system size is ultimately limited.

A third design option is the use of a hierarchy of memories. [WiM88] Here, each processor has its own local memory, as well as semi-local memories shared among a few processors. Global data can reside in a common store. This architecture is actually a generalization of the two memory levels found in cache systems. While this approach reduces bus traffic, no hardware provision is made to maintain data coherency. In this behavior, the cm* system is similar to message passing architectures.

While all of these shared memory designs are effective for the system sizes and the classes of problems for which they are intended, they do not support efficient implementations of ANN models. In both mathematical and neurophysiological ANN models, the ratio of computation to communication is lower than in symbolic processing models. Also, the communication includes a high degree of replication of results to multiple destination processors. If a shared memory system is used, contention may occur between processors accessing common sections of memory. In addition, shared memory systems do not scale well to the number of processors required to implement ANN models. Seitz shows that the maximum effective number of processors in a shared memory system is less than 10^3 [Sei90]. His argument is that, while a saturated bus can be replaced by a switching network, even for switching networks that are $\log_2 N$, the latency of traversing them constrains their effective maximum size.

5.1.2.2. Message Passing Systems

Message passing models can be compared by their message granularity and routing mechanism. In this context, granularity is a measure of the minimal message length and its associated overhead. When message transmission is a major expense in a system, it restricts the

degree to which a problem may be partitioned between separate processors. Systems with minimal overhead can be used for finer-grained jobs, such as extracting the parallelism from a compare loop or performing an FFT. ANN models also require fine-grained communications.

The spectrum of granularities within multiprocessor systems runs from the Intel iPSC-1, which used an ethernet packet as its smallest message size, to the proposed MIT J-Machine, which would execute messages analogously to instructions. The J-Machine model can effectively support partitioning of programs with 5μs between memory references, while the iPSC-1 requires minimal tasks on the order of 10ms [Dal90b].

The next sections focus on message passing architectures, demonstrating how the spectrum of ANN implementations can further be constrained by type and topology of interconnection.

5.1.3. Direct Versus Indirect Routing

Whether the system is shared memory or message passing, message routing is required. The different approaches to routing messages can be partitioned into two general classes: direct and indirect. With a direct network, connections are made between the processing nodes themselves. In an indirect connected system, the processing nodes are attached to a separate network of switching nodes. Shared memory architectures are examples of indirect networks with the simplest switching network being a shared bus. Message passing systems can be built with either approach.

5.1.3.1. Indirectly Connected or Switching Networks

A wide variety of indirect connection, or switching, network, alternatives have been proposed in the computer architecture literature. Most provide the equivalent of *cross-bar* or the potential for a connection between every pair of source and destination nodes. Although a message can be sent from each source node to each destination node, switching networks of n

processors are not usually implemented to support n simultaneous connections since it would require the n^2 cost of a true cross-bar system. The main drawbacks to the use of switching networks are the delay in traversing them and the added area required.

The general form of switching networks is $N = k^n$ processing nodes connected by n stages or layers of k^{n-1} switching nodes of size $k \times k$. The difference between alternative architectures is the definition of the switching nodes and the topology of the wires between them. Stone provides a comparison of switching networks that have been shown to be computationally equivalent [Sto81]. In all of these architectures, any input can be connected to any output. The communication delay is the time required to transit the n layers.

While it is possible to design a system where sources may broadcast to multiple destinations, it is not possible for a single destination to receive inputs from multiple sources at one time. For ANN implementations, the communication system must cycle multiple messages through the switch for each network update. Either the switch is built as a separate physical entity with wires run from the processing nodes, or it is superimposed upon the nodes themselves. Both possibilities present design problems. As shown by Dally, the bisection cost for switching networks connecting N processors is order N [Dal90b]. That is, for all switching network topologies, a vertical cut through the switch would sever N wires. The area of the layout is dependent on the number of nodes and does not vary with the choice of either k or n.

Fat Trees are an interconnection topology in which the processors are wired as the leaves of a complete binary tree [Lei85]. The internal nodes of the tree are routing processors. Available bandwidth between routing nodes increases toward the root and decreases towards the leaves. In an optimal system, the bandwidth increases appropriately so no messages are lost or delayed and the root is not a bottleneck.

A Fat Tree is a synchronous system with all messages moving one bit at a time up the trunk until they reach the first common ancestor of the source and destination processors. The time required for message transmission is equal to the number of inner nodes encountered, plus the length of the message multiplied by the delay for a single bit to move between two adjacent processors. All messages are sent during the same period of time, so best case delay is equal to worst case delay.

Fat Trees, as originally designed, are not good candidates for emulating ANNs. They require equal numbers of routing nodes as processing nodes, have high wiring costs, and each message is delayed as long as the worst case. An asynchronous Fat Tree is possible, but would require increased bandwidth or conflict resolution techniques. The idea of a tree as an indirect routing system is presented later during the discussion of possible physical broadcast implementations.

Fahlman developed the Hashnet concept as part of the design of the NETL system [Fah79, Fah80a]. Hashnet introduced the concept that, with a sufficient number of layers in the interconnection network, it is not necessary for all possible paths to be physically present. Fahlman showed the requirements for a million node system based on a Hashnet to be a 960 x 960 switching network time-shared 1024 ways [Fah80b]. Time sharing of the switching network is proposed as an alternative to increased area in interconnect and switching nodes. Although Hashnet itself is not a good choice for an ANN emulator due to excessive transition delays and the cost of connecting wires between the PNs and the switch, the idea of time-sharing a switching network between inputs is one way to reduce cost-performance ratios.

5.1.3.2. Directly Connected Networks

Examples of directly connected networks include trees, hypercubes, cube-connected cycles, grids, and tori. Each has a p-graph which describes its interconnect. This section briefly

describes these architectures and lists their benefits and costs.

Trees provide $O(N^{1/2})$ delays for $O(N)$ layout area [Maz87]. They are readily laid out following the H-tree pattern shown in Figure 5.1 and originally developed by Horowitz and Zorat [HoZ81]. Fault tolerance is a problem, since the loss of the central node splits the tree into two disjoint subtrees. Also, H-trees are not efficient for VLSI implementations, since they have an increasing percentage of unused space with increasing size.

One popular architecture for commercial message passing systems is the hypercube. Each node is connected to log n destinations as shown in Figure 5.2. The primary problem with hypercubes is the larger number of connections required per node as system size increases. Also, increasingly long wires are needed to lay out larger systems, resulting in increased power consumption and message transit times.

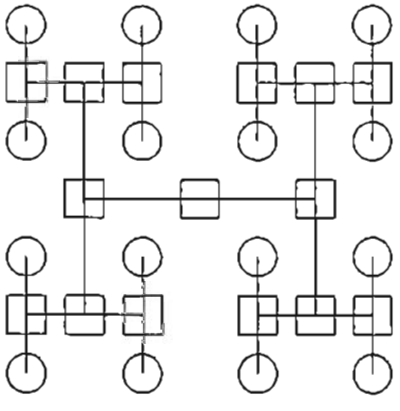


Figure 5.1 H-Tree Layout Pattern

A variation of the hypercube is cube-connected cycles. Each vertex of a boolean hypercube of size 2^m is replaced by a ring of m vertices. The benefit of this design is the restriction of each node to degree three. Its primary problem is that it requires $O(N^2 / \log^2 N)$ area for layout [Maz87].

The final topology considered here is the mesh or torus. A mesh is a regular array of processing nodes, each connected to its nearest neighbors. Nodes on the edges are not fully connected. That is, for a rectangular mesh, right edge nodes are not directly connected to left edge nodes and top row nodes are not directly connected to bottom row nodes.

If opposite edges of a mesh are directly connected, a torus results. If only one pair of opposing edges are connected, the resulting figure is a cylinder. To avoid the problem of long

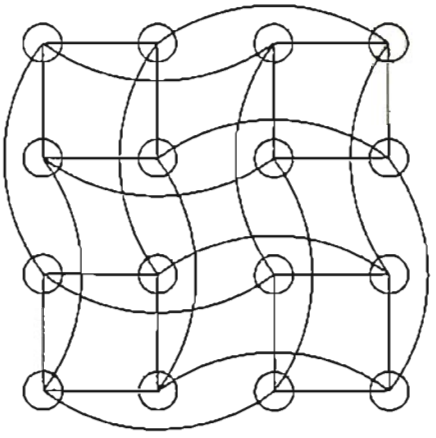


Figure 5.2 Dimension Four Hypercube

wires, the normal VLSI implementation of a torus is folded as in Figure 5.3. This folding increases the internode wire length to twice the length used in a mesh, but results in all wires being of the same length, rather than some wires spanning the width or height of the underlying mesh.

Tori, meshes, and hypercubes are all members of the more general class of k-ary n-cubes, as defined by Dally [Dal90a]. In this notation, k is the radix and indicates the number of nodes in

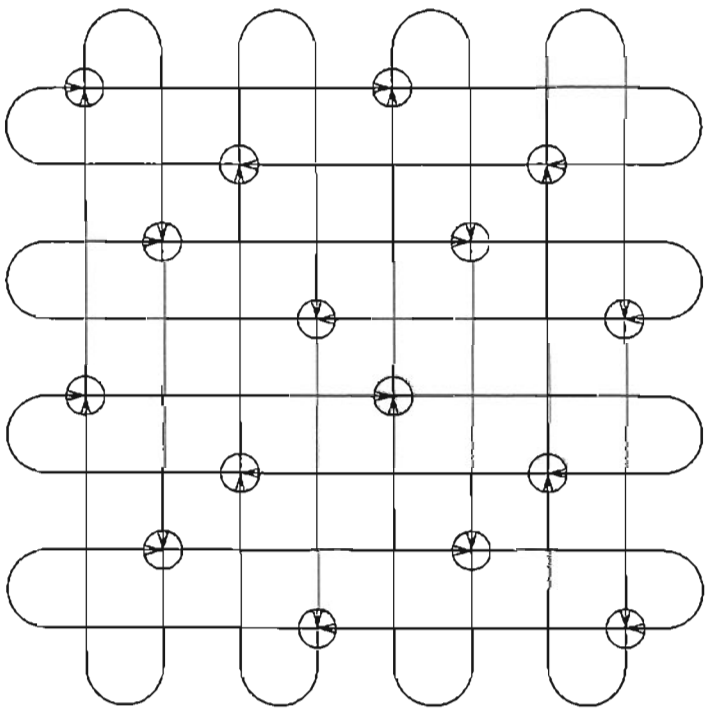


Figure 5.3 Folded Torus

each dimension while n is the dimension of the system. A torus is a k-ary 2-cube and a hypercube is a 2-ary n-cube. For $n = 1$, the resulting topology is a ring of k nodes. A mesh is considered to be a disconnected torus in this viewpoint.

Low dimensional k-ary n-cube networks can be wired with complexity $O(N)$ compared to the $O(N^2)$ required for multistage networks. It is possible to vary the choice of n and k to minimize network latency since networks with large k have a greater internode bandwidth for a fixed bisection width and networks with large n have a decreased diameter or number of intermediate nodes a message must be forwarded through.

5.1.4. Point-to-Point versus Broadcast Communications

When examined from the viewpoint of communication requirements, the primary characteristic of ANNs is the high degree of their c-graphs. As shown in Chapter 3, it is necessary to multiplex some p-graph edges to get an area-efficient implementation of a large c-graph. The mapping of c-graph to p-graph is simplified when the degree of the p-graph is increased. By using a broadcast architecture, it is possible to increase the /fIvirtual fan-out/fR of each PN of a p-graph of N nodes to N.

The architectures described earlier are all designed for a point-to-point (PTP) communication protocol. In PTP, each processor communicates with a single destination at a time. Messages are created with sufficient information to allow them to be routed to destination nodes and for destination nodes to be able to determine message sources. For computational models with a limited degree of simultaneous communication, PTP is the appropriate model. However, for ANN models, benefits accrue from using broadcast instead.

A major problem with ANN implementations is finding a p-graph with sufficient degree to support effective mappings for a variety of c-graphs, but without the need to dedicate large amounts of area to rarely used long connections. One solution is to use message broadcast to

create large *virtual fan-out*. In this usage, virtual fan-out refers to connections which are not physically present in the p-graph, but which are made to appear present by the use of broadcast connections. Broadcast provides flexibility of connections, reduces memory requirements, and shortens message length.

In programming ANN systems, one problem is adding new connections when they are needed. Most learning algorithms require the destination node, not the source node to add the connection. When a PTP communication technique is used, the transmitting node determines which nodes receive each message. Broadcast, using *come-from addressing*, solves this problem since any *listening* node can choose to either accept or ignore any message.

Come-from addressing allows each message packet to contain only identification of the originating node. All messages are broadcast with no need for routing information. Nodes receive a message packet and either accept it, if they need input from the source, or discard it based on the source address in the packet.

When a PTP communication scheme is used, the sending node must maintain address tables of PNs to which to send messages. The receiving node must also keep tables of sources, so it can assign the appropriate weight to each connection. The use of broadcast replaces the need for tables of destination PNs with a smaller list of destination broadcast regions, unless a node exists in only one region and no table is needed. Broadcast does not, in general, change the requirement for source tables. Thus, the use of broadcast can reduce the required address space in the PNs by at least half. In addition, if the ANN model has a dense matrix, weights of zero can be applied to all unused sources. Then a trade-off can be made between connection source addresses in the destination nodes versus extra space for weights.

Similarly, message length can be reduced by broadcast. In PTP systems, each message must contain a destination address or routing information. Come-from addressing omits the des-

tination address, since every message is sent to every node in a region and no routing is required. ANN models still require the source address be sent for weight assignment. It is possible to have the sequence of messages define their sources using a *slotted* broadcast protocol, where n broadcast intervals are allocated for n nodes and node i always broadcasts in interval, or slot, i. Slotted protocols eliminate all need for addresses in message packets. In cases where source addresses must be sent, it is possible to use shorter addresses, as long as each source within a broadcast region is unique. With these variations, messages can be reduced to data plus source address in most cases and to data alone in the rest.

The transmitting node need generate only one message. It is not necessary for multiple distinct messages to be created. Sending a single message reduces the complexity of the transmitting circuitry and reduces its area.

Unfortunately, broadcast is inefficient when a small percentage of the nodes in a region are destinations for any one message. In Chapter 7, required graph densities are given for the effective use of broadcast. Using a collection of regions organized in an overlapping or hierarchical structure can reduce the problem for some ANN models.

Figure 5.4 shows an overlapping broadcast structure. This feed-forward network has three layers where first layer nodes use region A to transmit to second layer nodes. Second layer nodes receive their inputs in region A, but use region B to transmit their output to the third layer nodes. If a single region were used for this system, it would have twice as many source and destination nodes as either A or B and the efficiency of the communication system would be halved. Also, the delay due to serialization of messages would be doubled.

5.1.4.1. Augmented Broadcast Architectures

Figure 5.5 shows an example of a one dimensional broadcast hierarchy. This approach is appropriate for systems where message traffic decreases with increasing distance from the source.

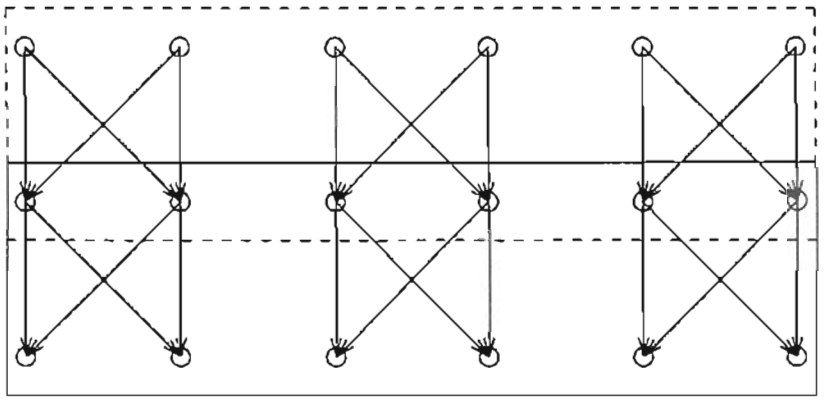


Figure 5.4 Overlapping Broadcast Structure Implementing a Feed-Forward Network

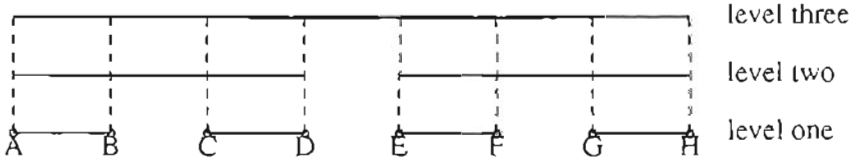


Figure 5.5 One-Dimensional Broadcast Hierarchy

For example, node A uses a level one connection to communicate with node B, but has to compete with three nodes when it uses a level two connection to reach node C. Finally, eight nodes compete for bandwidth on the level three connection which A must use to communicate with E. By seldom using the higher levels, congestion is reduced. If every message from node A must reach node E, no benefit is derived from having the hierarchy, from the perspective of node A.

In summary, the use of broadcast provides a large effective fan-out and reduces message length and PN memory requirements. ANN implementations can be more flexible, since the c-graph is not directly implemented and connections can be added as necessary. The use of multiple overlapping regions can further reduce the memory requirements, since addresses can be made shorter. In addition, overlapping or hierarchical regions provide for added parallelism in communication and may reduce communication contention, but at the cost of increased system complexity.

Two major potential problems exist with a broadcast architecture: the presence of long paths and the high cost of crossing region boundaries. Many ANN models require some messages be sent to far nodes. If only broadcast is used, these messages must be sent to all nodes within the smallest region containing both the source and the furthest destination. Sending messages destined for a few nodes to many nodes negates most advantages of broadcast and brings back the scaling problem inherent in a bus architecture. That is, remote connections reduce the effectiveness of broadcast and create a increased need for bandwidth, which is then poorly utilized.

Similarly, if a node outside the primary broadcast region must be reached, a broadcast is required to all the nodes in the second region containing the additional node. Such a secondary broadcast again increases required bandwidth and reduces the efficiency of usage of the broadcast system.

Augmented broadcast is a modified broadcast system that addresses these two problems. It has a PTP structure for the occasional long paths and overlapping regions or relayed messages to blur region boundaries. When only a few messages must be transmitted to far nodes, it is more efficient to reserve a limited portion of the communication bandwidth for them than to use global broadcast. The determination of what percentage of communication to reserve for broadcast

versus PTP depends on the density of the c-graph.

One possible class of c-graphs has a unique destination set for each node such that when mapped to a nearest neighbor planar p-graph, each CN connects to all PNs within some fixed radius. If the only interconnect is a set of rectangular, non-overlapping broadcast regions, severe problems may result. While no problem exists for those CNs mapped to PNs near the center of the physical broadcast regions, CNs mapped to PNs near an edge of a region must broadcast into the adjoining region(s) as well. Broadcasting into multiple regions causes decreased effective utilization of available bandwidth. The overlapping broadcast regions of an augmented broadcast architecture are intended to reduce this problem.

Another possible architectural variation is the use of relay nodes. Each relay node is connected by a PTP system to distant sources and broadcasts into its destination region messages received from its sources. This interconnection is similar to some brain models where axons travel a long distance before branching out to connect with many neurons at their destinations.

5.1.4.2. Broadcast Implementation Possibilities

Broadcast may be implemented in two ways: *virtual* and *physical*. In a virtual broadcast system, the physical communication system may be any PTP architecture. Here, the virtual broadcast and PTP communication of an augmented system can share the same basic communication structure. Messages are sent by the originating node and forwarded to all nodes within some region. One possible way to implement such a model would be with a *forward count* in each message that is decremented every time the message is forwarded until it reaches zero. Such an architecture would require a routing algorithm, for example messages received horizontally are forwarded in the opposite horizontal direction and messages received vertically are forwarded in all non-receipt directions, to avoid the problem of duplicate messages being received by a node. One example of virtual broadcast is the SIMD system to be described in Chapter 6, which

is based on a torus.

The primary advantage of virtual broadcast is it retains the simplicity of the underlying architecture while gaining the benefits of broadcast. Virtual broadcast is potentially more flexible than physical broadcast since it is not restricted to fixed regions. With appropriate routing algorithms, it is possible to have a different region for each node, removing the need for overlapping physical regions and directly supporting nearest neighbor algorithms. The comparative performance of virtual to physical broadcast is determined by the size of the regions and the amount of contention, as shown in Chapters 6 and 7. In addition to the absolute delay magnitudes introduced by contention, delay reproducibility may be a problem.

A physical broadcast system has a dedicated physical interconnection structure which implements the communication architecture. A variety of possible structures are available, ranging from buses to rings, trees, and stars. For small regions with relatively few nodes, a bus provides the simplest interconnect. With the use of a slotted algorithm for sharing a bus, no address information need be transmitted since the slot index indicates the transmitting node. In VLSI implementations, the energy to drive a wire and the speed with which it can be driven are determined by its capacitance, limiting potential bus size. These problems can be solved in some architectures by using either a hierarchical or pipelined bus design.

When repeaters are added to a bus, it is effectively changed to a ring with each node driving the wire between itself and the next node. It can be debated whether the virtual broadcast architecture in Chapter 6 is a series of physical broadcast rings or a torus supporting virtual broadcast. In either case, this approach has each node in a ring forwarding a value to its successor node each step with all nodes completely updated every n steps, for a ring of length n. Such a ring architecture also supports a slotted protocol where each message is uniquely labeled by its slot and address requirements are eliminated from each message packet. It will require $\max(m, n)$

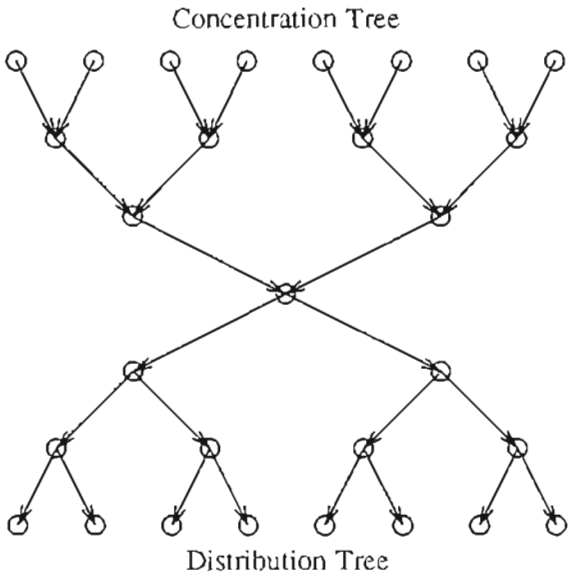


Figure 5.6 Broadcast Tree

steps for all messages to reach all nodes where m is the number of messages and n is the number of nodes.

Another topology is a star network where each node in turn transmits its new state to a central hub, which then updates all the destination nodes. This model is adapted from the original Aloha network, which was the first slotted broadcast architecture. One way of wiring such a star is using a tree as shown in Figure 5.6. Some characteristics of a possible tree architecture are described by Rudnick [RuH88]. This approach also has the benefit of readily supporting inputs external to the region.

5.2. Effective P-Graph Architectures for Supporting Large ANNs

In the preceding sections of this chapter a number of different p-graph architectures have been described. This section summarizes the earlier findings and suggests a small set of architectures which are effective for supporting large ANN systems. These suggested architectures will be compared in Chapters 6 and 7 to show how they fair in supporting potential message loads.

A few generalizations can be made about which p-graph architectures are effective candidates for large-scale silicon ANN implementations. Because of the problem of propagating clocks across a large area, it is likely for synchronized regions to be limited in size. Message-passing architectures are better able than shared-memory architectures to scale in size to support systems of 10^3 to 10^6 independent processors. Direct networks, where the connections are between the PNs themselves are preferable to indirect networks where there is a separate switching network providing the connectivity. The only feasible use of an indirect network is directly overlaid on top of a grid of processors. Not using an overlaid approach allows for denser PNs, but at the cost of increased communication delays from the length of the connections to the switch.

The range of implementation possibilities is reduced to k-ary n-cubes and broadcast, both virtual and physical. Before considering implementations based on these remaining options, it is necessary to characterize them by cost and performance.

The measure used here for performance of an interconnection network is the number of messages that can be communicated in a fixed time period. The number of messages which can be sent is a function of how many can be sent in parallel, or the bandwidth of the system, and the time required for a single message to reach its destination. The delay, or transmission time, can be further broken down into wire transit time and internode relay time. The wire transit time is dependent on the length of the wires, while the internode relay time is related to node size, com-

plexity, and routing mechanism. Distance traveled is used here because it is a good approximation of the delay time. Another contributing factor affecting performance is the degree of contention in the system caused by multiple messages competing for a single communication resource.

The cost of a given architecture is a function of the physical area required for wires and nodes plus any redundancy needed to circumvent faults. Another factor is the degree of regularity of the design. More regular designs are easier to lay out, have fewer flaws from the design process, and are easier to test.

5.2.1. Performance Characteristics

Dally [Dal90a] shows the time required to transit a message is $t = D + L / W$ where D is the average diameter of the p-graph, L is the message length in bits, and W is the channel width in bits. Given this formula, the lowest latency occurs when $D \approx L / W$. For a system of 10^6 nodes, he shows the optimal design is a 16-ary 5-cube.

In Dally's arguments, graph diameter is a good measure of the distance a message must travel, because all wires are held to be equal in length. This result does not hold when the entire system is mapped to two-dimensional silicon with limited numbers of layers. Higher-dimensional systems suffer the added cost of longer wires. Given the short message length of ANN models, it is not a good design decision to dedicate a long communication structure to a single message at a time.

For a square mesh, the average message distance is $2/3\ (N^{1/2} - N^{-1/2})$ [Sei90]. For a torus, the average distance is $2\ N^{1/2}$ links, but since a link is twice the length of a mesh link it is actually $4\ N^{1/2}$. For a broadcast tree, all distances are the same and for a square layout of $N = 2^{2k}$ in size are $2k - 1$ to the central node and $2k - 1$ back for a total distance of $4k - 2$, which is equal to $\sqrt{N} - 2$. The first two designs can be laid out in area equal to N processing nodes. The third requires additional space for the inner switching nodes, but as shown in [RuH88] the additional

space needed is less than 15%.

5.2.2. Comparison with Dally

Dally first proposed the idea of using a fixed cross section for comparing performance of different interconnection architectures in his dissertation [Dal86]. Since then he has published a number of following papers where he has expanded upon the concept of k-ary n-cube networks as optimal solutions for many communication problems [Dal90a, Dal90b]. Since publication of these results, most research in the area has used them as a starting position. Some of the results in this paper disagree with some of his conclusions. Since his work is so well respected, this section was added to explain where the two models differ and why the results are different.

The standard message in Dally's work is taken to be a multiple-byte object that is split into a number of individual packets named *flits*. In ANN models, a message consists of a source address, a destination address, and a numeric value. In many models, the source and destination address may be reduced to a single value that is a routing address from source to destination and uniquely identifies the source. With messages reduced to the size of a single packet or flit, the problem of deadlock disappears, no resolution algorithm is required, and there is no need for the *virtual channel* concept Dally uses to support *wormhole routing*. Another advantage of having no possibility for deadlock is messages can be supported on a mesh instead of requiring a torus with single direction messages. Another advantage of the short message length in ANN models is *store and forward* becomes a viable alternative, since the required memory for message storage in a node is significantly less and it is feasible to support a larger buffer size.

Another difference between the two models is while Dally also uses a probabilistic model for communication, the actual pattern of message transmission is dependent upon the computational algorithm being used and the timing differences between the processors. In an ANN system, the message flow is more predictable since it is a part of the definition of a network, both in

routing and firing frequency. Because of the robustness of ANN models, if a problem exists with excessive messages, it is possible to not deliver some percentage without significantly degrading the system performance. Different researchers have shown varying levels of continued functioning in the presence of flaws, but deleting 1% to 5% of the connections in a network has been shown to be supportable.

The sizes of the networks used in the models are comparable. While Dally has not created any systems with 10^6 nodes, his calculations include such networks sizes. In the actual systems he is designing, the number of nodes is closer to 10^3. The ANN systems under consideration in this paper start with networks of this size and scale on up to 10^6 or greater.

One major difference is that the model used here is of a planar silicon implementation while Dally is using a hierarchical approach with chips, boards and systems. His results thus are supported by the assumption of nearly equal length wires in a k-ary n-cube. On a planar layout, this assumption is not true since some lines are orders of magnitude longer than others. The presence of long wires causes systems with larger n to have a slower performance since the long runs require more time for message transit. The work here agrees with Dally on the value of low dimensionality networks and the idea of using bisection width as the measure of cost rather than wire fan-out count.

In a typical computer system, a delay in a message transmission only delays the computation at the destination end. In an ANN model, it may cause the network to behave differently if the delay variation is of the temporal type as defined in Chapter 3. Finally, with the typical firing pattern of ANNs, the messages come in waves that tend to saturate the network, which then settles down into a quiescent mode. The time required for all messages in a set to be delivered defines the scaling factor between computational time and communication time. In more conventional computer systems, messages are typically independent objects and the latency for a single

one to transit the system is the critical measure. Pipelining approaches are thus useful for ANN models and useless for conventional ones.

The previous paragraph illustrates one difference between ANN and the more classical communication models of Dally. In the classical model, there is a simulated clock with messages containing some time-stamp. In ANN models, as described here, time is its own representation. The messages are sent in real-time and there is no attempt to provide an external synchronizing function.

In conclusion, the models here are more supportive of store and forward, do not deal well with high-dimensional graphs, require a flushing of the messages in a burst, and allow some messages to be lost in times of high network saturation.

CHAPTER 6

Communication Cost-Performance Tradeoffs

To select between alternate interconnection architectures, knowledge of their cost-performance behavior under typical ANN message loads is needed. For VLSI silicon, cost is a function of design complexity and area required, because larger devices are more difficult to manufacture and have a higher probability of containing significant flaws. Performance is measured here by the time required for message delivery. An optimal system design is one where the cost and performance of the communication subsystem matches that of the computational subsystem. Otherwise, either the communication subsystem could be implemented less expensively or it constrains the system speed.

Since the work of Mead *et alia* [MeR79] and Thompson [Tho79], VLSI architectures and algorithms have been compared on the basis of the area versus the time required. This research follows the example of Mead and uses $Area \times Time$ as the final comparator. The other alternative used in the literature places a higher value on time by using $Area \times Time^2$ as in Thompson and Seitz [Sei90]. This presentation also follows the example of Mead in using the terminology of cost-performance rather than the equivalent area-time notation of Seitz.

In this chapter, an abstract model of message sources and destinations is defined, followed by a statement of the assumptions used in the analyses. In comparing interconnection architectures, only two of the four possible communication scenarios need to be considered: messages originating outside the region for internal destinations, and messages that originate and terminate within the region. Messages originating within the region with external destinations are equivalent to messages of external origin with internal destinations. Since this work focuses

on message loads within a region and in general it is not possible to predict loads due to routing externally sourced and destined messages, the fourth possibility is ignored. Under each message model, the cost and performance of supporting a uniformly distributed message load is calculated for physical broadcast on dedicated broadcast structures and virtual broadcast and PTP on mesh and torus.

The final section of this chapter summarizes the results and describes their significance. It also shows when each alternative architecture would be preferred.

6.1. Communication Model Definition

The model p-graph used in this chapter is a square grid of processing nodes with a communication subsystem superimposed upon it. A square grid is used because it is the simplest design for regular tiling of a plane. The only regular figures that support a tiling of the plane are triangles, rectangles, and hexagons. The complex shapes of other possibilities for tiling, coupled with a lack of significant benefits, removes them from consideration. Increasing perimeter length increases the opportunity for routing of interconnect. Perimeter length to area is maximal for a triangle and minimal for a hexagon; for regular examples of these figures the ratios are: triangle, 4.5 to 1; square, 4 to 1; and hexagon, 3.7 to 1. The number of adjacent neighbors of a node determines the number of minimal cost connections. All non-adjacent connections must cross at least one intervening node. Given these considerations, a square shaped PN provides an intermediate level of functionality on all counts, supports simpler cost and performance calculations, and is easier to lay out and fabricate.

External inputs can be from a variety of sources. Some, such as light for the Silicon Retina [Mea86], come from a third dimension and do not require any interconnection support within a module. At the other end of the continuum are inputs, as shown in Figure 6.1, routed to their destinations via the interconnect capabilities of the implementation process itself, or in

VLSI, the multiple layers of poly and metal. Intermediate between these two possibilities are systems where some inputs are to the periphery of the module while others are conveyed externally to internal locations. An example of such an interconnection is a chip with both internal and peripheral contact points. The most conservative design constraint is to assume all external messages enter across a single edge of the module. Running external wires to central pads is equivalent to increasing the number of layers in the technology and does not significantly affect the calculations, since message load can increase geometrically with model size and the number of layers is constant. Allowing three dimensions for interconnection is a possible generalization, but it does not significantly change the final conclusions since cross section in three dimensions increases as $N^{2/3}$ while message possibilities increase as N^2 [Sei90].

Figure 6.1 illustrates the node layout and communication model used to determine boundary communication requirements. This figure has twelve possible sources and sixteen potential destinations for a given message. Four points are shown where messages may cross the boundary. This representation accurately replicates the case of two communicating submodules within a VLSI system. In addition, it is an effective abstraction of the more general instance of inputs coming from a variety of sources at different distances or delay rates and crossing an interface boundary into a stand-alone module.

In the following arguments, S represents the number of sources, D the number of destinations, M_i for $1 \le i \le S$ the number of messages generated by each source during a single communication interval, and Q the number of queues or interface crossing points. If each source is sending a message to each destination, $\forall i\ M_i = D$, the total number of messages is DS, and the average number passing through each queue DS/Q.

The maximal value of D is the number of nodes in the module, or N. For a square module, the length of each side is $\sqrt{N}$ nodes. Each processing node has area A, so the total area

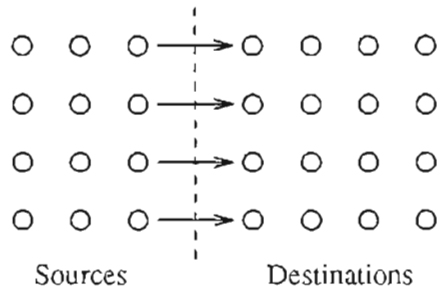


Figure 6.1 External Inputs Entering a System.

of the module is AN and each side is $\sqrt{AN}$. B represents the bandwidth of the interconnect. For a value of $B = 1$, a single message transits a wire of length $\sqrt{A}$ in time T. When both communication and computation time are used together, T_m is message time and T_a is algorithmic computational time, otherwise T refers to communication time. A wire of bandwidth $B = 1$ is defined to have a physical width of W. The units of W and A depend on the implementation technology.

Since the computational capability and layout of a processing module are not of concern here, all modules are assumed to be of some unknown, identical design. All messages from one module to another are defined to originate in the center of the first and to terminate in the center of the second. Thus, T_m is the time required for two adjacent nodes to transmit a single message.

Each source node generates all messages resulting from a particular computation in a single burst, i.e., all M_i messages from node i are created during the same time interval and queue up if the communication channels are unable to absorb them immediately. The pattern of when a particular node i fires and where messages are sent is determined by the ANN model implemented. There are no ordering constraints on messages. The only possible interaction between

two messages is competition for routing and only then if they share some subpath between source and destination.

Messages are assumed to be of unit length. Allowing for bit serial transmissions only marginally changes the relative performance of the architectures, based on message length and address requirements. Dally has shown that serial messages may be routed using the *e*-cube, or dimensional order, method without deadlock [Dal90b]. For these reasons, the use of unit length messages, while greatly simplifying the arguments, does not introduce significant error into the analysis. A more precise definition of bandwidth and time is thus: a single message of length L transits a communication path of bandwidth B in time T_m. Such a path has length $\sqrt{A}$ and width W. Note that under this model a message of length kL would take time kT_m.

Another simplifying assumption made in the calculations of message transit time and area required is that all wires run either north, south, east, or west with no angles other than 90 degrees. The cost of this simplification is a potential increase in wire lengths by a maximum factor of $\sqrt{2}$.

6.2. Interface Communication Requirements

Communication bottlenecks may occur at two locations in a system. One is at interfaces with the external world or between modules. The other is on internal communication paths. For layered networks, these two locations are equivalent to constraints upon inter-layer and intra-layer communication potentials.

In this section, the requirements to support inter-layer or external world communication are determined. If a layer, or other set of processing nodes, is to perform a useful computation, it is necessary to have inputs and outputs. These may be system inputs and outputs, i.e., those from the external world, or they may be messages between subparts of the system. For a single module, as considered here, these two classes of message sources are equivalent.

While initially only the bandwidth at the interface and the required length of a single side of the module are of concern, it is necessary that the internal distribution mechanism be able to absorb messages as rapidly as they cross the interface. If not, messages must be stored at the interface or delays propagated back to the sources. Potential message-storage capacity needs to be infinite to indefinitely support a faster rate of message generation than internal distribution. Thus, the maximum interface bandwidth is equal to the maximal rate at which messages are distributed to their ultimate destinations within the module.

If the rates at which messages are created and consumed are both deterministic, it is possible to create a system with constant queue length and no excess system capacity [Kle76]. In such a system, the rate of message consumption is equal to the rate of message generation. In theory, it is possible to arbitrarily increase the number of queues, or their bandwidth, to raise the message absorption rate of the communication system to whatever is needed. The primary constraint limiting such growth is the cost, or module side length, required for a given queue capacity at the interface and the area required internally for the communication interconnect to distribute messages as they are received. The other ultimate limit on the message consumption rate is the rate at which PNs can remove messages from the communication system.

An alternative way to solve the problem of excessive messages is to slow down the rate at which new messages are generated to match the consumption rate. Such a slowing may be done by defining the time used for computation, T_a, as a multiple of the communication time, T_m. Such an approach reduces system throughput instead of increasing system cost. These two options, slowing the computation to reduce message generation rates and increasing system cost to obtain greater bandwidth and message consumption rates, are the parameters of system optimization. For this reason, systems can be compared by the area required for communications times the time needed to deliver messages.

Unfortunately, not all implementations are deterministic in message creation or consumption. Congestion can cause distribution delays of arbitrary length. Various ANN models have different, often probabilistic, rates of message creation. In addition, the spatial pattern of message generation, i.e., which nodes fire during a given time interval, can temporarily overload some queues while leaving others idle. Given these behaviors, it is necessary to design in excess capacity to avoid the possibility of arbitrarily long message queues. Systems with more flexible communication structures, such as broadcast architectures, do not require as much excess capacity as systems based on dedicated communication links. The following arguments assume a deterministic world for the sake of simplicity.

6.3. Interface Bandwidths

This section considers the potential bandwidth or message consumption rate of the entire module as constrained by the internal bandwidth and interconnection architecture.

As considered in Chapter 3, the system with highest message throughput is a direct connect implementation with each destination and source physically connected, requiring SD wires. If the computation model limits a source node i to sending messages only to a subset of the possible destinations, the number of wires is $\sum_{i=1}^{S} M_i$ or $S\overline{M}$, where $\overline{M}$ is the average message fan-out of the source nodes. The space required for wires defines the minimum perimeter possible for the system. The total perimeter is SDW/z where z is the number of layers provided by the technology. Due to the excessive cost of direct connect as an implementation architecture, as shown in Chapter 3, it is not considered further.

The system with the minimal interface area requirements is a broadcast architecture capable of absorbing one message per cycle. Such a system requires only a single queue from the external world. To supply messages to this queue requires either a concentration tree, as shown in Figure 5.6, or some equivalent mechanism. Since the wires to route messages to the interface

are external to the module under consideration, their cost is included in the module where the messages are generated and is therefore ignored here.

With a physical broadcast system, each message is sent to all destinations, so the total number of messages is reduced to S. The result is a serialization of messages and $T_a^B = ST_m^B$. If the number of messages that could be delivered in a single time step were increased to bandwidth B^B, then the length of a computational cycle would decrease so $T_a^B = ST_m^B/B^B$. Note the presence of the superscript B, for broadcast, is used to differentiate variables when comparing results with other architectures, such as mesh, with superscript M.

An architecture intermediate between direct connect and physical broadcast for both performance and cost is a mesh. In a mesh architecture, each node is connected to the four adjacent nodes (North, South, East, and West). Messages enter the system through the nodes closest to the system boundary and are routed to their destinations through intermediate nodes as required. Mesh provides a higher total bandwidth at the interface than broadcast because $Q^M = \sqrt{N}$ compared to $Q^B = 1$. This increase in bandwidth is offset by the possibility of message congestion within the system, as covered later. Mesh does not replicate messages as does broadcast, so it is necessary to transfer $S\overline{M}$ messages per computational cycle and $T_a^M = \frac{S\overline{M}T_m^M}{B^M Q^M}$, where B^M is the bandwidth per node at the interface for mesh. Comparing T_a^B to T_a^M and deleting common factors gives a performance ratio of $\frac{T_m^B}{B^B}$ for broadcast to $\frac{T_m^M \overline{M}}{B^M \sqrt{N}}$ for mesh which reduces to $\frac{T_m^B}{T_m^M} \times \frac{B^M}{B^B} \times \frac{\sqrt{N}}{\overline{M}}$. Given comparable bandwidths and system timings, i.e. $B^B \approx B^M$ and $T_m^M \approx T_m^B$, if $\overline{M} > \sqrt{N}$, mesh performs worse than broadcast. Otherwise, it performs better by the ratio of $\sqrt{N} / \overline{M}$. This result indicates that for systems where the fan-out per message is less than the square root of the number of potential destinations, mesh is the preferred implementation for performance. Torus is not considered separately here because it has equivalent message

absorption rates at the system boundary. It will be covered later when describing message distribution with the module and message consumption.

The primary benefit of a broadcast architecture is the reduction in the messages generated each computational cycle. Each source sends a single message, no matter how many destinations it is intended for. A tantalizing dream is to combine a reduction in message count with the simplicity of a mesh type nearest-neighbor architecture, hoping for optimal cost performance to result. For this reason, and as a model which can be compared to both mesh and physical broadcast, virtual broadcast is considered as an alternative communication architecture.

Virtual broadcast is defined as a communication architecture rather than an interconnection architecture because it is a message routing technique, not a p-graph wiring technique. Virtual broadcast may be implemented on any p-graph. In this chapter, it is described on both mesh and torus. For a PTP communication architecture, each node sends out a specific message to each destination and this message is routed by intermediate nodes according to some protocol. In contrast, for a virtual broadcast system, each node sends out a single generic message which is forwarded by each node it encounters up to some limit, again defined by the routing protocol.

In virtual broadcast, each message is propagated through the entire destination network, visiting each node at least once. The preferred routing architecture minimizes duplicate copies of messages. Interestingly, such a network performs at essentially the same speed as the physical broadcast tree, i.e., the added value of the additional parallel paths is not realized. To prove this assertion, consider the two modules pictured in Figure 6.2. The physical architecture is a series of loops, one for each column in the layout. As explained earlier, such a topology is actually a cylinder with inputs into one of the unconnected ends. A cylinder is used because it provides the same approximate level of throughput as PTP on a mesh, but with reduced congestion. The physical broadcast system accepts a single message every cycle and retransmits it to all destinations.

The virtual broadcast system accepts $\sqrt{N}$, in this case four, messages at a time. Since these messages must be propagated to all the nodes in the first column, it takes four cycles for the leftmost loop to update all nodes. Then each node in each column shifts its current value to the next column to the right and the leftmost column accepts a new set of messages. Thus virtual broadcast accepts $\sqrt{N}$ messages every $\sqrt{N}$ communication cycles and physical broadcast accepts a single message each cycle. At the end of S cycles, for S divisible by $\sqrt{N}$, each has potentially transmitted a message from each source to all destinations. The differences between the two implementations is cost of interconnection, complexity of design, and propagation time.

In summary, the nature of the internal interconnection architecture determines the maximal rate at which messages are accepted by the system at the interface. Direct implementation with a wire for every message provides maximal bandwidth at the interface, but requires maximal area. In comparing PTP on a mesh and broadcast, if both have the same internal bandwidth and system clocks, their performance has the ratio of $\sqrt{N}/M$, where N is the total number of nodes in

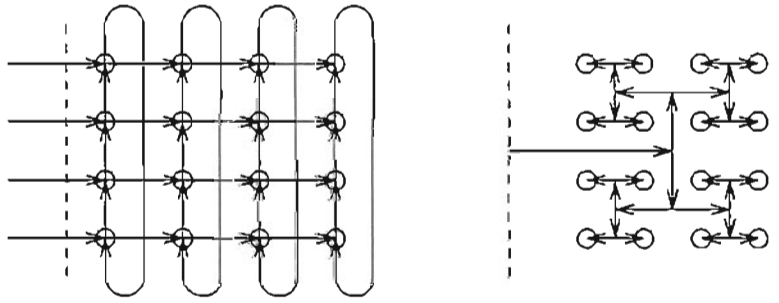


Figure 6.2 Comparison of Virtual and Physical Broadcast

the region and $\overline{M}$ is the average number of destinations per message.

6.3.1. Internal Support for Interface Requirements

The previous section compared system bandwidths for fixed internal interconnect bandwidths. In this section, the internal distribution of message loads is shown. The architectures considered are the same with the addition of a one-dimensionally connected mesh or cylinder. All arguments assume messages are presented to one side of a square grid of N nodes continuously. The destinations for these messages are randomly distributed throughout the region with equal probabilities.

With the elimination of direct connect as an alternative, the easiest architecture to characterize is physical broadcast. In Chapter 5, several different implementation possibilities were discussed. The model used in this chapter is a concentration tree feeding to a single point and back via a distribution tree as shown in Figure 5.6. Long branches in the tree are broken into multiple pipeline stages of length $\sqrt{A}$. Such a tree can be considered as a pipeline which accepts 2^i inputs every 2^i cycles and generates output to 2^j nodes, one set of outputs occurring each cycle after the pipeline has been filled. An alternative view is that it accepts an input from each of the system sources every 2^i cycles, and delivers it to all destinations $2^i + p$ cycles later, where p is the time required to transit the pipeline.

Mesh, both with individual message routing and with virtual broadcast, has the problems of message congestion and route contention. The routing algorithm used is to complete all required row traversals, then make any necessary column traversals. To calculate the effect of message congestion on required area, the following paragraphs calculate the required width of each internal communication path to support a continuous input of m messages into each row every time interval. These calculations assume all messages are uniformly distributed among the possible destinations.

With m messages entering each row of a mesh-based system via the n nodes on one edge, the number that cross each horizontal wire within a row is a monotonically decreasing function from a maximum of m entering the source side to $mq(n)$ at the opposite side of the system. Here $q(i)$, for $1 \le i \le n$, is a function such that $\sum_{i=1}^{n} q(i) = 1$ and $q(i)$ represents the probability that a given message will have a destination in column i. Specifically, for the uniform distributions assumed here, the horizontal bandwidth into each node of column i is $m(n-i+1)/n$ since m messages enter the leftmost node of each row and $1/n$ are consumed in each column. The total horizontal bandwidth for each row is then $m(n+1)/2$.

Vertical capacity is more complex because of the possibility of congestion due to the interactions of messages entering a column from different rows. If a message is waiting to enter the vertical system from a horizontal path and the next available slot is taken by a message coming from the vertical path, the new message must wait. In turn, the following horizontal message is delayed and horizontal throughput decreased.

Two alternatives are considered here for the vertical architecture. One has independent paths in both directions, the other is a loop, or torus, with all messages traveling in a single direction. The two architectures are separately analyzed.

A layout of a series of adjacent loops is more predictable in its behavior, so it is the first one to be discussed. As defined in Chapter Five, an interconnect structure is not properly a torus unless it is *looped* in both x and y directions as shown in Figure 5.3. Consider a single column of n nodes with each node capable of accepting messages from the left, or input side, and from its predecessor in the loop each cycle. Each node is also defined to be able to forward messages to the right, or output side, as well as to its loop successor during the same cycle. A message enters a node on one cycle and exits it on the next, if an open path exists in the desired direction. Otherwise the message is stored until the route is available. Since a finite amount of storage is

available, a new message only enters a node after another message has exited.

To build a system that continues to accept messages every cycle without requiring infinite storage capacity, requires all messages to be routed to their destinations without any delay from congestion. This requires that all nodes have bandwidth nm since nm messages enter the system in each cycle, and it is possible for all messages to be intended for a single destination. Fortunately, ANN models can be considered to have a uniform distribution of messages across all the nodes within a given region. Thus, the average number of messages consumed per cycle per PN is m/n.

For a series of loops, the vertical bandwidth between nodes must be $m(n-1)/2n$, to avoid congestion, with m messages entering the system on each row each time interval. Given n nodes per loop, the total vertical bandwidth is $m(n-1)/2$. Appendix A has the derivation of these formulas as Results 6.1-a and 6.1-b.

One generalization is that a message is destined for a node in column i with probability $q(i)$. Under such a distribution, all nodes within a column would be equally probable destinations, but nodes in different columns would have varying likelihoods of being destinations. Such a distribution matches the olfactory piriform cortex model described in Chapter Four where the probability of a connection with a LOT neuron decreases with the anterior location of the piriform neuron.

The next PTP interconnection architecture is a true mesh with separate paths in each vertical direction. With a mesh, there are no direct connections between nodes at the top and bottom of the system. Again, the expected horizontal bandwidth declines with the distance from the source edge. Unlike the case for a cylinder, a uniform distribution of destinations within columns leads to a non-uniform pattern of required vertical bandwidths.

Given m/n messages moving into each column from each row, the required communication width ranges from a maximum of m in the center to a minimum of $2m(n-1)/n^2$ at the top and bottom of the column. The actual function is $2jm(n-j)/n^2$, $1 \leq j \leq n$, where j is the index of the edge from either the top or bottom of the column. Results 6.2-a and 6.2-b in Appendix A show the derivations of these formulas. Given $n-1$ edges in each column, the total vertical communication bandwidth is $m(n-1)(n+1)/3n$.

In summary, for PTP on a mesh interconnection with m messages entering each row at one edge, the horizontal message load per edge is m in the worst case. The load per edge with a uniform distribution of message destinations declines from m across the first edge to m/n across the final one. The vertical message load differs depending on architecture. For both mesh and cylinder, the worst case is $m(n-1)$ messages per edge and occurs only with a single destination for all messages entering the system. The expected number of messages crossing each edge is uniform for the loop case and is equal to $m(n-1)/2$ for a message interjection rate of m per row every time period. For a mesh architecture, the load is uneven and ranges from $2m(n-1)/n^2$ across the top and bottom edges to m in the center following the function $2mj(n-j)/n^2$ where j is a row index. These message load formulas are used in the next section to calculate the area and time requirements for each architecture.

6.3.2. Edge Based Cost-Performance Trade Offs

Two alternatives exist for balancing message creation and consumption rates. One is to increase the communication bandwidth until messages are consumed at the same rate as they are created. The other option is to slow the message creation rate and reduce the required bandwidth. In this section, formulas are given that express the relationship between bandwidth cost and performance for each of the alternative architectures. As before, all messages originate externally and enter the module through a single side.

These calculations assume S distinct message sources with $\overline{M}$ messages per source. To allow direct comparisons between the architectures, the $N = n^2$ destination PNs are laid out in an array of $2^k \times 2^k$. For all models, B represents the bandwidth or number of messages that cross a p-graph edge each time interval.

For a physical broadcast system, implemented as shown in Figure 5.6, with B messages entering a single channel from the external world each time cycle, the minimal time required to distribute S input messages is $(S/B + 3n/2)T_m$ where T_m is the time for a message to travel a distance of $\sqrt{A}$. This formula results from considering the message distribution tree as a pipeline of $3n/2$ wires of length $\sqrt{A}$. Such a pipeline has $n/2$ edges, of length $\sqrt{A}$, from the edge of the module to the central root of the tree and n such edges in each path to the leaf PNs.

Building such a physical broadcast system would require $(3n^2-2n)BW\sqrt{A}$ area for wires. A single wire of length $n\sqrt{A}$ runs from the edge to the root and the tree contains $3(n^2-n)$ wires of length $\sqrt{A}$ for a total wire length of $(3(n^2 - n) + n)\sqrt{A} = (3n^2 - 2n)\sqrt{A}$. To support B messages per cycle, each wire is BW wide.

A virtual broadcast system consisting of a $n \times n$ cylinder, unconnected at the left and right sides, with the capability of accepting B messages at each leftmost edge node each time cycle, can distribute S messages in time $(\frac{2S}{B} + 3n)T_m$. It takes S/nB steps for S messages to enter n nodes, B at a time. Each such step takes time $2n$ as explained earlier. Once the final messages enter the first column, it takes an additional $3n$ steps for them to fully propagate throughout the system. Such a communication system requires an area equal to $\frac{n(3n-1)BW\sqrt{A}}{2}$. This area calculation follows from the earlier analysis of a PTP loop system where total vertical bandwidth is $Bn(n-1)/2$ and total horizontal bandwidth is $Bn(n+1)/2$, in these formulas B has replaced m as the number of messages entering each row each time interval. Vertical wires are of length $2\sqrt{A}$ and horizontal wires are of length $\sqrt{A}$. Adding the bandwidth values, multiplied by wire

widths and lengths, yields the stated result.

A PTP system built on a cylinder can process $S\bar{M}$ messages in time $\left\lceil \frac{2S\bar{M}}{nB} + \frac{3n}{2} \right\rceil T_m$, where B is the bandwidth, or number of messages per cycle, into each edge node. This is the expected time given a uniform distribution of sources and destinations. Message delivery time has two components. First is the $\frac{S\bar{M}}{nB}$ required for all messages to cross the left-hand boundary into the system. Vertical edges are of length $2\sqrt{A}$ so the time is doubled. Added to this is the average time of $3n/2$ for the final message to reach its destination. The wire cost for such a mesh system would be $\frac{n(3n-1)BW\sqrt{A}}{2}$, identical to the virtual broadcast system.

Since time steps in a mesh are 1/2 of those of a connected mesh, a simple mesh can process $S\bar{M}$ messages in time $\left\lceil \frac{S\bar{M}}{nB} + n \right\rceil T_m$. The results in the previous section on bandwidths predict a system wire cost of $\frac{(n+1)(5n-2)BW\sqrt{A}}{6}$. Again, this wire cost formula represents the addition of the total bandwidth times the width and length of a single wire.

Table 6.1 summarizes these results. The performance column shows how long it takes to transfer one full set of messages, $\bar{M}$, from each of S sources to their destinations in terms of time T_m. The second column is the cost for wire area in terms of the bandwidth B, wire width W, and wire length $\sqrt{A}$. Finally, the right column is the product of the *continuous communication* portion of the performance times the cost. In this last column, the common factors $\sqrt{A}$, W, S, and T_m have all been omitted to show more clearly how the alternatives compare.

The *continuous communication* time used in the calculation of the cost × performance column is the time for all messages to enter the region. This is the first term in the performance formulas. The *variable* portion of the time is how long it takes for the last set of messages to

Architecture	Performance ($\times T_m$)	Cost ($\times BW\sqrt{A}$)	Cost × Performance($\times WST_m\sqrt{A}$)
Physical Bcast	$\frac{S}{B} + \frac{3n}{2}$	$3n(n-2)$	$3n(n-2)$
Virtual Bcast	$\frac{2S}{B} + 3n$	$\frac{n(3n-1)}{2}$	$n(3n-1)$
Cylinder	$\frac{2S\overline{M}}{nB} + \frac{3n}{2}$	$\frac{n(3n-1)}{2}$	$(3n-1)\overline{M}$
Mesh	$\frac{S\overline{M}}{nB} + n$	$\frac{(n+1)(5n-2)}{6}$	$\frac{(n+1)(5n-2)\overline{M}}{6n}$

Table 6.1 Performance and Cost for Edge Based Message Loads

propagate to their final destinations. For a system in steady state, as soon as one set of messages have entered the system, the next set are sent in a pipelined manner. For this reason, the variable time is not a factor in ongoing performance.

Table 6.1 shows a true mesh as costing 5 / 9 as much as a cylinder. As stated in the derivation section, a mesh requires a non-uniform layout with more bandwidth in the center of columns. All alternatives scale $O(n^2)$ for cost. With n^2 nodes in the destination region, communication costs increase at the same rate as destination nodes given a fixed number of external sources. Virtual broadcast is slower than physical broadcast by a factor of 2 and its cost is one half as much, duplicating the earlier intuitive demonstration of cost-performance equivalence between virtual and physical broadcast.

The variable portion of the system update time is within a factor of three for all architectures. The continuous term is better for physical broadcast by the ratio of $\overline{M} / n$. If the average number of connections per source node is less than the square root of the number of potential destinations, a PTP message-routing system is better than a broadcast one. Otherwise, broadcast

requires less time with an order of n better performance for total connection. This result can be understood by considering the effect of n paths into the region for PTP versus the single path for broadcast times the $\overline{M}$ message count multiplier for PTP.

6.4. Internal Communication Requirements

The next area of concern in designing a communication system is the possibility of delays caused by contention on interconnection paths supporting messages both initiated and terminated within the module. For the physical broadcast system, the cost of supporting internal messages is equal to the sum of the costs of the input and output trees. For PTP architectures, the added degree of freedom of message sources can significantly affect minimal bandwidth requirements.

For the analyses in this section, all messages are defined to be uniformly distributed both in source and destination. S is the number of sources, m is average messages per source, and $n^2 = 2^k \times 2^k$ total nodes are in the system.

A binary tree implementation of physical broadcast has an upper bound of $(s/b + p)t$ time to distribute s messages to all destinations, where s is the number of message sources, b is the bandwidth, p is the number of wires a message must transit going from a leaf through the root and back out to a leaf, and t is the time for a single message to transit the longest wire in the tree. For a shared source and destination space of $2^k \times 2^k$ nodes, with full pipelining and message repeaters a node length apart on all long wires, $p = 2^{k+1} - 2$, $t = T_m$ and the upperbound becomes an equality. Replacing 2^k by n to provide results more directly comparable with the other architectures yields a system update time of $(S/B + 2(n-1))T_m$. For full interconnect, with n^2 nodes transmitting values, the update time is $(n^2/B + 2(n-1))T_m$.

The wire cost to build such a system is $6(n^2 - n)BW\sqrt{A}$. Both the concentration and distribution trees contain $3(n^2 - n)$ wires. The width of each wire is BW since B simultaneous

messages are being transmitted, and their length is $\sqrt{A}$ because of the full pipelining.

The cost and performance of physical broadcast can be compared to virtual broadcast implemented on a torus with a single update cycle of $2(n^2 - 1)T_m$, regardless of the number of message sources or destinations. During the first $n - 1$ time intervals, input messages are cycled vertically through the source column. Each node then shifts its message to the adjacent horizontal node. The next $n - 1$ steps are a repeat of the first, distributing the second set of messages to all nodes of the newly receiving column. This procedure continues until all columns and nodes have received all messages. The total number of steps is $n^2 - 1$ because each step allows a new node to receive a given message and a total of n^2 nodes are in the system. Since this system is torus based all wires are twice the length of a single node and the total time required is $2(n^2 - 1)\ T_m$. In this model, every node transmits a value each time interval. If a node's state has not changed since the last transmission, it repeats its old value.

The wire cost for a virtual broadcast system of this nature is that of the underlying torus, or $4n^2\ W\ \sqrt{A}$. A torus requires $2n^2$ wires, n columns of n nodes with wires in both dimensions. Each wire is of width W, since a single message is transmitted each time cycle and is of length $2\sqrt{A}$. Note that a torus as described here differs from the cylinder of the previous section by having both vertical and horizontal edges connected.

PTP routing algorithms such as mesh and torus must support a total message load of $S \times \overline{M}$, with the worst case for an array of $n \times n$ nodes of $n^2(n^2 - 1)$ messages. Each node i generates a total of $0 \le M_i \le n^2 - 1$ messages with probability p_i, where p_i is the probability of node i firing. Worst case for message load is $\forall i\ M_i = n^2 - 1$ and $p_i = 1$. For a uniform distribution of sources and destinations, it is possible to define m as the number of messages generated and consumed by each node each cycle. If each node broadcasts and receives the same number of messages, $m = \frac{1}{n^2} \sum_{i=1}^{n^2} M_i\ p_i$.

For a torus with uniform distribution of message sources and destinations, all messages can be delivered in time $\left\lceil \frac{m(n-1)}{nB} + 2n \right\rceil T_m$. This formula results from the assumption of perfect routing of messages with all nodes forwarding messages in a pipelined fashion. Within a torus, all nodes are topologically equivalent. Pick an arbitrary node and consider how long it takes to distribute its m messages. The local column gets m/n with the remaining $m(n-1)/n$ being forwarded to the next column. This latter activity takes $m(n-1)/nB$ time steps if B messages can be transmitted in each step. The expected time for the last message to be delivered is n time steps of length $2T_m$. The worst case for the last message is $2(2n-1)T_m$. Such a system requires a total bandwidth of $B(n-1)/2$ and a total system wire cost of $2n^2(n-1)BW\sqrt{A}$. The derivations of these formulas are also in Appendix A as Result 6.3.

For a mesh, all messages can be delivered in time $\left\lceil \frac{m(n-1)}{nB} + n \right\rceil T_m$. Here, m is the expected number of messages generated or consumed by each node and B is the internode bandwidth. This result may be determined analogously to the torus above. The wire cost is $\frac{2n(n-1)(n+1)BW\sqrt{A}}{3}$ as shown in Result 6.4 of Appendix A.

Table 6.2 summarizes these results. The organization of the table and its definitions are the same as those of Table 6.1 except the last column in Table 6.2 explicitly includes the factor S, or the number of message sources, since the virtual broadcast row does not require it and the others do.

All communication times shown in the performance column in Table 6.2 have a variable and a constant term. The variable term is a function of the number of messages sent and the constant term is the inherent time required for the communication system, considered as a pipeline, to empty. An alternative phrasing is the constant term is the time required for the last message to

Architecture	Performance ($\times T_m$)	Cost ($\times W\sqrt{A}$)	Cost × Performance	Constant Delay
Physical Bcast	$\frac{S}{B} + 2n$	$6Bn(n-1)$	$6Sn(n-1)$	$2n$
Virtual Bcast	$0 + 2(n^2 - 1)$	$4n^2$	$8n^2(nsup\,2 - 1)$	$2(n^2 - 1)$
Torus	$\frac{m(n-1)}{nB} + 2n$	$2Bn^2(n-1)$	$2mn(n-1)^2$	$2n$
Mesh	$\frac{m(n-1)}{nB} + n$	$\frac{2}{3}Bn(n^2-1)$	$\frac{2}{3}m(n-1)^2(n+1)$	n

Table 6.2 Performance and Cost for Internally Generated Message Loads

be delivered. The variable term for virtual broadcast in Table 6.2 is 0 because the communication delay is constant, it does not depend on the number of messages to be delivered. The cost × performance values in Table 6.2 only include the variable performance term for all architectures but virtual broadcast since it is the only part of the communications time dependent on the bandwidth.

Both broadcast architectures have a cost $O(n^2)$, while the PTP communication systems have cost $O(n^3)$. This $O(n)$ difference in cost is due to the increase in the number of potential message sources with increasing region size as opposed to the earlier results shown in Table 6.1 that assumed a fixed number of sources.

Virtual broadcast is thus better than physical broadcast for cases where most nodes are firing. If the bandwidth is set to one for physical broadcast, the implementation costs are similar. Physical broadcast does better when fewer than all nodes are firing at a time. When all nodes are potential sources, $S = n^2$ and for a bandwidth of one, the two have cost-performance ratios differing by 3/4.

For most internal routing systems with high message loads, physical broadcast is the preferable architecture. If all nodes are broadcasting, then virtual broadcast is directly comparable.

6.5. Summary

The three classes of messages supported by a given communication subsystem are external to internal, internal to internal, and internal to external. For messages crossing the boundary in either direction, Table 6.1 summarizes cost and performance for each of the considered interconnect architectures. All are $O(n^2)$ in cost with a constant factor ranging from one to six separating them. Both PTP-based systems are slower than broadcast by the ratio of $\overline{M} / n$, so they perform better only for systems with few destinations per source.

For systems with messages both originating and ending internally, Table 6.2 summarizes the cost-performance equations. Here both broadcast systems are $O(n^2)$ in cost compared to $O(n^3)$ for the PTP systems.

Virtual broadcast has a cost and performance that is dependent on system size only. Physical broadcast does better than virtual broadcast when the number of message sources decreases and poorer when dealing with full interconnect. Both PTP systems are slower by a factor of the average message load, so would be rejected on the basis of higher cost and slower performance.

The next chapter presents generalizations of these results, including some possible tradeoffs that can be made to better tune real systems. It also includes examples showing the cost and performance of possible implementations of artificially and neurophysiologically based network designs.

CHAPTER 7

Implications and Examples

This chapter combines the various results developed in this thesis and shows some of their implications. The first half of it summarizes results in three sections. The initial section is a parameterization by density of connection and CN firing rate of the cost and performance formulas in Chapter 6. The next section is a discussion of potential trade-offs between computation and communication. The last section is a collection of hints for selecting between PTP and broadcast for a given implementation.

The second half of the chapter contains descriptions and analyses of example implementations. These are based on the ANN models described in Chapter 4. The first model considered is a simple three layer feed-forward network. This is followed by a review of a possible implementation of piriform cortex. The third example is a proposed silicon version of a stylized rat hippocampus. The last section is an overview of some of the problems and issues to be solved before implementation of visual or cerebral cortex models is possible.

7.1. ANN Interconnection Theory

This section summarizes a theory of ANN interconnection. It is based on results of the earlier chapters. The goal is to assist system designers in determining which interconnection architecture to use when implementing a particular ANN model.

7.1.1. General Cost-Performance Formulas

The formulas developed in Chapter 6 were based on a fixed message load each communication cycle. It is possible to represent such a system load as the product of two probabilities, c

and f, times the number of sources, s, and the number of destinations, d. In this representation, c is the probability of a connection existing between two nodes in a given network architecture and f is the probability of a message being transmitted over a connection during any one communication cycle. For a direct implementation of a c-graph, $c \times d$ is the number of wires required per source node and $100 \times f$ is the percentage utilization of those wires during each cycle. Although f is defined in terms of the probability of a given node firing, because of the large number of nodes and the uniform distribution of their destinations, it is valid to use f as a multiplier on the total potential message count.

One important characteristic of the ANN models being considered here is a numerically uniform distribution of inputs. That is, all inputs are to equivalent numbers of nodes and all nodes in a layer receive the same number of inputs. The spatial and temporal distribution of connections may be either *uniform* or *structured*. For a *uniform*, or homogeneous, model each portion of a layer is equally likely to contain nodes receiving a given input. The ultimate example of a homogeneous network is every node in the source layer connected to every node in the destination layer as shown in Figure 2.7. In *spatially structured*, or *uneven*, models, while all nodes receive the same number of inputs, each source node has outputs to a limited area of the destination layer. In the terms of Chapter 2, spatially structured models typically exhibit a higher degree of locality than uniform models.

The cost and performance of interconnection architectures are related by the bandwidth of the communication subsystem. Cost is directly proportional to bandwidth since bandwidth determines the physical area required for each set of wires. The total time to transmit a set of messages is the product of the number of messages sent times the time required for each such transmission divided by the number that can be transmitted simultaneously:

$$total\ time = message\ count \times transmission\ time / bandwidth$$

Increasing bandwidth increases the number of messages which can be sent in parallel and reduces

total time. The performance of a system is measured by the time required for a single update cycle. Tables 6.1 and 6.2 show how cost and performance are related for four different interconnection options.

For both the uniform and structured models, broadcast provides identical cost and performance results. With broadcast systems, only the number of source messages to be transmitted affects the time required. Neither the quantity nor distribution of destinations is of importance, since all nodes receive all messages. The only exception to this rule is ANN models where destination regions are sufficiently partitionable to allow separate broadcast systems to be effectively implemented for each region. With such a partitioning, the potential exists for a reduction in the height of the distribution tree and a resulting reduction in the time required for messages to be delivered. If both the source and destination regions are partitionable even greater performance improvements are possible. The expected message load for a broadcast system during each computational interval is the number of potential inputs times the probability of each one firing or $s \times f$. The cost of using broadcast is efficiency of message distribution, since messages are sent to nodes that do not need them.

PTP architectures are affected by both the total message count and the patterns of source nodes firings and message destinations. For uniform systems, all parts of a layer are equally likely to contain destinations for a given input. The expected message load into all regions is equally affected by the firing or non-firing of a given source and is $c \times f \times s \times d$. For non-uniform systems, while the same number of messages must be processed, different parts of the layer may receive unequal numbers of messages during a given time interval. Since all nodes receive the same number of messages, to avoid contention delays, it is necessary to design non-uniform systems to support a message load of $c \times s \times d$.

The conclusions in the preceding two paragraphs match the results shown in Tables 6.1 and 6.2 where the number of destinations ($c \times d$) affects the message load for PTP, but not for broadcast. It also emphasizes that the critical value for broadcast performance is the probability of node firings or f. In other words, the *spatial* distribution of messages is crucial for PTP while their *temporal* distribution is crucial for broadcast.

Table 7.1 is derived from Table 6.2 by substituting the c and f notation in place of the S and m values originally used, for the number of messages sources and the average number of messages a node transmits. It clearly shows f, the probability of a node firing during a particular communications cycle, is the crucial measure for a broadcast system. For PTP systems, it is necessary to provide bandwidth from each node in proportion to the potential number of messages, even when the node does not transmit an update and the bandwidth remains idle. Thus c, the probability of a connection of the system, is the critical measure for PTP systems.

Architecture	Performance (T_m)	Cost ($W\sqrt{A}$)	Cost × Performance	Constant Delay
Physical Bcast	$\frac{fn^2}{b} + 2n$	$6bn(n-1)$	$6fn^3(n-1)$	$2n$
Virtual Bcast	$0 + 2(n^2-1)$	$4n^2$	$8n^2(n^2-1)$	$2(n^2-1)$
Torus	$\frac{cn(n-1)}{b} + 2n$	$2bn^2(n-1)$	$2cn^3(n-1)^2$	$2n$
Mesh	$\frac{cn(n-1)}{b} + n$	$\frac{2}{3}bn(n^2-1)$	$\frac{2}{3}cn^2(n-1)^2(n+1)$	n

Table 7.1 Performance and Cost for Internally Generated Message Loads

Note that the constant portion of all methods except virtual broadcast has order n. This factor represents the average distance the final message must travel. For virtual broadcast there is a constant communication time, regardless of the number of messages to be sent. In all other architectures, there is a pipeline effect where the constant portion of the communication delay represents the filling or draining of the pipeline and the variable portion of the formula represents the number of messages that transit it.

While both broadcast systems require message times of order n^2, it is a constant value for virtual broadcast but variable for physical. Thus physical broadcast can gain from the use of increased bandwidth and is better in cases when less than all nodes are firing each time interval.

The major observation to be drawn from Table 7.1 is that, while all interconnection architectures have order n^2 performance, the two PTP examples are more expensive in cost by a factor of n. Thus the cost of PTP rises faster by $N^{1/2}$ than either broadcast alternative, where N is the number of nodes.

7.1.2. Computation versus Communication

Varying the size and performance of processing nodes can provide an improved match between T_a and T_m for a given network update algorithm. As shown in Chapter 6, T_m is dependent upon the communications bandwidth of the system and ranges from a maximum when one message is sent at a time to a minimum when all messages are sent in parallel. T_a is a function of the ANN update algorithm and the computational capabilities of the PNs. Both T_m and T_a can be varied by changes in the complexity and area of the PN and communication system.

The area of each PN may be logically divided into three subparts. The first contains the communication support system: buffers for storing messages during routing, circuitry implementing the routing algorithm, line drivers to transmit messages, receiving buffers, and the portion of the physical interconnect within the PN boundaries. The second major subsection of a

processing node is memory to store connection weights, source addresses for messages to allow weight selection, and destination addresses. The final subsystem is the computational unit: adders, multipliers, comparators, control logic, and registers for intermediate results.

It is possible to vary the ratios of these three subsections as well as total PN area to achieve different performance criteria. At one extreme is a PN consisting of a minimal computational subsystem with no parallel processing capacity, memory for a single CN, and a communication system able to transmit and receive one message at a time. The other limit is a PN with sufficient parallel capabilities to perform all computations simultaneously and enough bandwidth to transmit updating output to all destinations without delays.

If the time for computational update is proportionately excessive compared to the communication time, multiple instances of the computational subsystem can be implemented in parallel. The communication and memory systems would be marginally increased in area due to the increased number of paths to the multiple processing units, but the major impact would be in the percentage of area devoted to computation. Such an approach is taken to its ultimate limit by the analog computational designs of Mead *et alia* [Mea86] and Jackel *et alia* [GrV87, JHG], where all inputs are processed in parallel in a single cycle.

Similarly, the bandwidth of the communications system can be increased by providing multiple drivers and wires to allow messages to be transmitted in less time and to support multiple simultaneous messages. The limit to this approach is a direct connection system with no multiplexing of wires between multiple message sources or destinations and wire area as the limiting factor in system design.

Finally, it is possible to reduce the total message load in the system by having multiple CNs implemented in each PN. Once a PN is designed with communication and calculation capabilities which can be shared, implementing multiple CNs with it requires adding memory and a

longer total communication and computation cycle. Depending on the temporal sparseness of the ANN, the affect on the system may either be increased area from the added memory needs or reduced performance from the serialization of functions.

Such a multiplexing of nodes can reduce communication costs and delays in three ways. First, if k CNs per PN require less space than k instances of a single CN per PN, total system area would decrease, reducing overall wire lengths and communication times. This reduction in area occurs when either communication or computation resources within a PN are shared by two or more CNs. Experience with the CNAPS architecture indicates this benefit is of limited value since storage of CN state requires > 70% of PN area for a single CN per PN [Ham92a]. Thus the maximal benefit from combining CNs onto a single PN is < 30% savings in area.

The second way that the multiplexing of nodes can reduce communication delays is, it is only necessary for one copy of a message to be sent to any PN containing multiple destinations for it. For systems with a uniform distribution of message sources and destinations, the expected message load scales inversely to the CN per PN ratio. A third effect of multiple CNs per PN is a reduction in the message count from having CNs with common local communication mapped to the same PN. In this situation, the internal wiring of the PN needs to support the local connections, but the external communication system does not.

7.1.3. Heuristics for Choosing an Interconnect

In designing a system, an architect seeks a balance between different parts to match implementation with function. This also holds when deciding which interconnection architecture to use. The works by Dally, referenced in Chapter 2, and Leighton [Lei92], study how different interconnection topologies support different problems. The conclusions drawn in this thesis differ from theirs because the overriding design problem when implementing ANN models is the large number of messages sent in each time interval. This difference in the computational models

constrains the communication requirements and structures. In terms of Tables 6.2 and 7.1, their studies focused on how to reduce the constant term of the communications time while this work focuses on the variable term.

This section assumes a rectangular array of PNs with some p-graph representing the interconnection topology between them. A mapping of a single CN to each PN is assumed. The p-graphs considered include PTP and broadcast as shown in Table 7.1.

One of the first concerns when choosing the interconnection architecture for any ANN model is its locality of communication. The locality of a graph is the relative size of the destination sets under a partitioning of it into distinct sets of nodes. Another concern is the dilation factor of c-graph edges from a mapping to a given p-graph. The dilation is a result of the reachability of the c-graph and its locality contrasted with the equivalent measures for the target p-graph.

Trivially, when a c-graph maps to a two-dimensional grid of PNs with a constant dilation factor, a nearest neighbor PTP architecture is adequate. For other dilation functions, the choice of p-graph depends upon message firing rates.

Consider a c-graph of N CNs. Assign them under some mapping to a contiguous, minimal area, two-dimensional array of PNs. Given the earlier definitions of c and f, cfN^2 messages are generated each communication cycle for a PTP system. For a broadcast system, the equivalent number of messages is fN. Superficially the lower number of messages for broadcast indicates its preferential use except for systems where $cN \leq 1$. One problem with this conclusion is that in a PTP system, multiple paths allow messages to travel in parallel, but in a broadcast system all messages are sent in serial. A second problem is the c-graph and mapping may be such that messages travel short distances with the PTP architecture and system wide with broadcast. A third problem with always using a broadcast system is overlapping communication requirements.

Serialization of messages under broadcast as compared to message contention and throughput under PTP was the topic of Chapter 6. PTP has a penalty of n, for a system of n^2 nodes, in area cost when compared with broadcast. Ignoring this factor and only considering message delivery times, broadcast is better than PTP when $f \leq c$ as shown in Table 7.1.

The two problems with using broadcast are that messages may be sent further than necessary and nodes that do not need a message may receive it anyway. The effective result of using broadcast is to provide full connectivity from one region of a graph to a second, possibly identical, region. Any node in the source region can transmit a message to any node in the second using the broadcast interconnect. No explicit provision is made within a single broadcast structure for connections from nodes in the first region to any other part of the graph, although such connections may occur via augmenting PTP connections, other overlapping broadcast structures, or message forwarding by the nodes within the destination region of the broadcast structure. In essence, the graph is partitioned into multiple distinct subgraphs, contrasted with PTP systems where some pathway normally exists between any two nodes.

The following discussion assumes the source and destination regions to be the same. This assumption is made to simplify the phrasing of the arguments. It is a simple argument to extend these results to systems with separate source and destination regions. Again, this argument is concerned with a single broadcast structure, not with a complete system.

When using broadcast, a major problem to solve is how to select an optimal size for the broadcast regions in order to obtain good cost and performance measures. The larger a broadcast region is, the more likely that it contains the destinations for all messages generated within it. Obviously, a single broadcast region spanning the entire graph contains all sources and destinations. Unfortunately, such a single region requires the serial transmission of all messages. If the utilization factor is high, i.e., most nodes require most messages, a single broadcast structure may

be appropriate. Otherwise, it is a poor usage of interconnect to have many messages being distributed more widely than necessary. The limits of utilization can be seen in Table 7.1 where the variable term of fn^2 / b is compared to the fixed term of $2n$. For systems with $f/b \leq n^{1/2}$, the fixed delay of communications is significant relative to the variable cost. But the fixed delay is determined by system size while the variable delay is a result of node firing rates.

The solution for the problem of wasted bandwidth from sending messages to nodes that do not need them is to use multiple overlapping broadcast regions. Again the trade-off is between size and utilization. If size is minimized so each region is of minimal diameter d, fixed shape, contains m nodes, and overlaps to any arbitrary amount, then each node is contained in m distinct regions each with fm messages to be transmitted every communication interval. The fact of each node only appearing in m regions can be understood by imagining a $k \times l = m$ rectangle being sequentially placed on the planar array of nodes so that a marked node occupies positions $1,1\ 1,2\ \cdots\ 1,l\ 2,1\ 2,2\ \cdots\ 2,l\ \cdots k,l$. If instead, each region overlaps by $d/2$ and is of size $4m$, then only four regions contain any node. Each of these new regions has message traffic of $4fm$. Total message load in the system has decreased since previously each node received fn^2 messages and now each one receives $16fm$.

Some ANN models allow a better mapping where some set α of CNs have a common destination set β. An example of such a model is the feed-forward network shown in Figure 5.4. This figure shows one mapping of such c-graphs to broadcast regions.

For broadcast architectures, a *slotted* protocol saves time when the message firing rate f is greater than the ratio of message length to message plus address length. Under a slotted protocol, each node is assigned a particular slot for its transmission. When a node does not fire, its slot contains no new information. Such a protocol allows the elimination of source addresses from messages since the slot position provides equivalent information. So, if the increase in informa-

tion content due to deleting addresses is greated than the decrease in content from empty slots, the system is more effective. The other benefit of a slotted protocol is it reduces the required space and complexity of PNs by replacing source address decoders by a simple counter. If c is large enough, the counter can index into an array of input values and remove the need for tables of source addresses.

A final benefit of broadcast is the simplification of mappings. All nodes which share common source and destination broadcast regions are equivalent. Mapping a CN to one of many equivalent PNs is simpler than selecting an optimal PN choice.

One way of considering the difference between broadcast and PTP is that PTP is designed for sparse spatial connections while broadcast is better for systems with sparse temporal connections. That is, PTP is intended for situations where each node is connected to very few other nodes. Broadcast is for situations where each node is connected to many other nodes, but the firing rate for each node is low, or when the communications subsystem is faster than the computation subsystem.

7.2. Example Implementations

As mentioned before, this section of the chapter contains the analyses of several different implementations. These examples have been chosen to show how the results described in this thesis are applicable to both current and future ANN implementations.

7.2.1. Feed-Forward Networks

Much current ANN research is based on simple feed-forward networks. Although the Back Propagation model suffers from $O(w^3)$ learning times [Hin87], where w is the number of weights in the system, and is thus not feasible for networks of the sizes considered in this work, other ANN models are being developed that have better learning performance. Before moving on to consider neurophysiologically inspired models, it is instructive to see how the results of this

work predict the performance of simple feed-forward networks.

Consider a system of 2^{10} nodes per layer and three layers. Also, define full inter-layer connections, so $c_i = c_o = 1$ for the central layer and no intra-layer connections, so $c_l = 0$. While the firing rate f depends on the problem set and the network inputs, a value of $f = 0.1$ is not unreasonable. The total number of messages in the center layer are then 0.1×2^{21} for a PTP architecture. For broadcast the serialized message count is 0.1×2^{10}.

Such a system could be laid out as three adjacent 32×32 squares of nodes. This provides 32 paths into and out of the system for PTP and 1 path for broadcast or $(.1{\times}2^{10} + 3{\times}2^{9})t \approx 3.2 \times 2^{9}t$ and PTP $(.1{\times}2^{16} + 3{\times}2^{9})t \approx 15.8 \times 2^{9}t$ for some clock rate t. Thus broadcast requires one-fifth the time of PTP. Given a clock rate of 10^{-11}, one complete communication update could be performed every 1.7×10^{-8} second for more than 5×10^{7} updates per second for broadcast.

With $O(n^3)$ area requirements for PTP and $O(n^2)$ ones for broadcast, the communications system for broadcast would require 1/32 of the area needed for PTP. If the interconnect is on the order of 15% of the entire system area as indicated by Means [Mea91] and Rudnick [RuH88], then with identical computational areas, the total space for PTP would be 5.6 times that of broadcast. Thus the final cost performance benefit of using broadcast is 5 times the speed for 1/5th the area or 25 times as good. As these results are based on the formulas of Table 7.1, they share the same assumptions of interconnect type.

7.2.2. Olfactory Piriform Cortex

In Chapter 4, layer II olfactory piriform cortex was described as having several neurophysiological models that are well enough defined to make it a good candidate for potential silicon implementation. To quickly summarize the interconnect, between 10^4 and 10^5 pyramidal cells form winner-take-all patches of 50 to 100 cells each. Inputs enter via the LOT and traverse the

patches from rostral to caudal. LOT fibers are randomly replaced by pyramidal axons from the patches until the LOT is completely changed by the caudal end of the system.

Means [Mea91] has described three possible implementations of layer II piriform cortex following the Lynch *et alia* model [GAL89, GAA90]. Means determined a direct implementation of a 10^4 neuron system with 10^3 inputs using analog computation would require $47cm^2$. By using a broadcast architecture with each patch contained in a single digital PN, the area required is reduced to $6.9cm^2$. In the direct model 95% of the area is required for communication wires. This percentage is reduced to 30%, including the area required for address decoders, in the broadcast implementation.

Performance is better for the analog PTP model under non-learning conditions where it is 20 times as fast as the digital system. The digital system would learn faster, but exact calculations for how much better were not provided.

Means' final conclusion is, because of scaling issues, the use of broadcast and digital computation with multiplexed PNs is preferable to the use of PTP communication and analog computation. A six inch wafer could hold 19,000 neurons with the analog model, and 35,000 with the digital one. Going to an eight inch wafer, the numbers increase to 24,000 and 56,000 respectively. Again, the primary benefit of broadcast architectures is their ability to scale better than PTP ones. The conclusions drawn by Means match those of this thesis: a direct implementation performs faster than a multiplexed broadcast one, but costs significantly more for interconnect, especially as network sizes increase.

Means' thesis was an application of the concept of broadcast as defined in the research reported here to a particular neurobiological ANN model. His calculations of the area required for computation and communication affirm the conclusion in Chapter 3 that direct implementation of large ANN models is not effective. While he does not consider locality in his work, it is

the presence of patches within piriform cortex with a low resulting message rate that makes the virtual, multiplexed system perform as well as it does.

In conclusion, it is possible to effectively implement the Lynch-Granger model in silicon using a broadcast interconnection architecture.

7.2.3. Hippocampus

The model of this section is drawn from the biological description of Section 4.4.2. The order of magnitude of the number of neurons and their interconnect has been preserved, but the actual counts have been rounded to values that simplify the design description. Again, the intent is to show how the required interconnect could be designed, not to define a biologically exact computational model.

Figure 4.5 shows schematically how the connections are made. The following description follows the flow of information from the external world to entorhinal cortex, then to dentate gyrus, CA3, CA1, and back to the external world. Given the speed differential between biological cells and silicon CNs, the design is for 200 CNs per PN. Based upon its approximate neuron count of 2×10^5, entorhinal cortex would consist of 20 patches. Each such patch would contain of 10^4 CNs or 50 PNs. These patches would be designed as broadcast concentration trees with their output directed to broadcast distribution patches within dentate gyrus and CA3. For an estimated firing rate, f, of 0.01, each such patch would generate 100 messages each update cycle.

The number and size of the patches came from the layering which is believed to exist in the biological model. The size of each patch preserves a balance between communication and computation delays for broadcast interconnect and shared PNs.

Dentate gyrus would be split into 50 similar patches. Each one would consist of 2×10^4 CNs or 100 PNs. Each PN would emulate 200 granule cells, one basket cell and 4 associative cells in a winner-take-all group. As described earlier, the use of multiplexed PNs to emulate

competitive groups reduces system message counts. Each patch would have inputs from one entorhinal patch and 20% of the total population of associative cells. The 20 entorhinal patches would each connect to 5 dentate gyrus patches. Each simulated granule cell would receive a mix of 1000 entorhinal inputs out of the 10,000 possible in its patch and 100 associative inputs from the 400 available. Each simulated associative cell would receive inputs from all of the granule cells in its patch. The assignment of inputs would be on a pseudo-random basis within each patch. With $f_{entorhinal} = 0.01$ and $f_{associative} = 0.05$, each patch receives 120 inputs each time interval. With winner-take-all clumps of 200 cells each, the number of outputs per patch would be 100 for the granule cells and 20 for the associative cells.

The CA3 layer would consist of 20 patches of 100 PNs each. Here, each PN would be simulating 200 pyramidal cells. Each CA3 patch would receive inputs from 3 of the dentate patches. In addition to the dentate gyrus inputs, each patch also would receive inputs from one of the entorhinal patches in a pattern similar to the dentate gyrus. Every CN would receive 20 dentate, 500 entorhinal, and 1500 local associative inputs. Given a $f = 0.01$ for all entorhinal, granule, and pyramidal cells, the total number of inputs for each broadcast region of CA3 would be 300 dentate, 100 entorhinal, and 200 local messages per cycle. The total number of outputs per cycle would be the 200 local messages. These local messages are then transmitted to the CA1 area.

The final area is CA1. Its organization would be 20 patches of 150 PNs. Again, each PN would support 200 pyramidal cells. Each CA1 patch would receive inputs from one CA3 patch.

The structure described here would have a quick communications cycle with the slowest broadcast trees being the CA3 regions with 600 messages per cycle. Given a pipelined structure, the total time required would be less than 1000 communications clock cycles. A system performance rate of 1000 updates per second would require total communication time of $10^6 t_{com}$, t_{com}

is determined by the time required for a message to travel the width of a PN. With a processing clock rate of 10^{-11}, approximately 10^6 computation steps are possible per update. Given 200 CNs per PN, this reduces to 50,000 computational steps per CN or about 100 steps per input per update.

The primary requirements for area would be the memory needed for storage of inputs. Since the number of external world inputs to the entorhinal region are not defined, it is ignored here. Setting the storage size to eight bits per input: dentate gyrus would require 9×10^9 bits, CA3 would require 6.4×10^{10} bits, and CA1 would require only 4.8×10^8 bits. This yields a total memory requirement for the system of about 8×10^{10} bits of data. Area required is approximately 50λ per memory bit [MeC80]. As stated earlier, communication requires about 15% of total system area. Extrapolating from the experience with the CNAPS architecture and realizing the high degree of sharing of computational resources implies a computational cost of < 5% of total area. Combining these figures yields a area of 60λ per bit or a total area of $5 \times 10^{12}\lambda$. A value for λ of 0.1μ is considered feasible with CMOS technologies, so the entire system would require $5 \times 10^{10}\mu^2$ or 500 cm^2 and could be built on a wafer 25 cm in diameter.

Thus it would be possible to create a total hippocampus in silicon, but only with a 0.2μ design process and 10 inch wafers. Such a system would have a system update rate of 1000 cycles per second, almost one full order of magnitude faster than the equivalent biological model.

7.2.4. Cortex

For both visual cortex, as described in Chapter 4 and the hypothetical model of cerebral cortex developed by Braitenberg, the proposed solution is to split the system into many small densely intraconnected patches. These patches correspond to the subregions or columns of the visual cortex and to the $\sqrt{N}$ regions of Braitenberg.

From Braitenberg, each patch would have $n = \sqrt{N}$ external inputs, each connected to every cell in a patch. In addition, n local sources are each connected to $n - 1$ local destinations. Such a collection of densely connected patches maps well to a broadcast style interconnect where a total of $2n$ messages are distributed to the n destinations within each patch.

Additionally, each node requires a single long distance connection to another patch. This connection could be implemented by a single PTP connection from each node to one other node in a different patch. The receiving node would then broadcast two values, its own internally generated one and the one received from the far node. Such a PTP connection would require nominally a grid interconnect to support the N messages being transmitted between a total of N nodes.

A similar approach could be used for visual cortex where each region receives multiple inputs from a maximum of 20 other regions. In addition to the connections from these external sources, each region is densely connected with every node receiving input from each other node of its region.

For both of these models, the use of many small broadcast regions allows fewer messages to be transmitted over long distances. It also allows each region or patch to be performing its communications in parallel with all other patches. The lack of locality of the Braitenburg model is compensated for by providing an augmenting PTP communication system. While it is possible to architect such systems, the example of hippocampus given earlier shows the requirements to store state make it impossible with current VLSI technology to implement an entire brain on a single wafer.

7.3. Summary

The examples in this chapter have shown it is possible to conceptualize p-graphs to support a wide variety of ANN models. Where c-graphs have a high degree of locality, either virtual or physical broadcast regions can effectively be used to readily support them with an $O(N^{1/2})$

improvement in communication cost-performance over PTP architectures. For c-graphs with less locality, PTP connections or overlapping virtual or physical broadcast regions can be added to provide the necessary augmentation.

CHAPTER 8

Conclusions and Future Directions

This chapter is a summary of the major results presented in this dissertation and some directions for further research.

8.1. Results

The research reported in this dissertation shows the effectiveness of treating ANN models as broadcast in nature and designing physical interconnection architectures around this characteristic. Previous work in interconnection architectures has focused on systems with single messages being sent from a given source at any one time.

The concepts defined in Chapter 2, such as reachability and locality, capture essential characteristics of ANN model topology. While other researchers have commented on dilation as a problem in performing a mapping from a problem graph to a physical system, they have only considered fan-out as a contributing factor. Here it is shown that a second-order effect, reachability, is more significant than simple fan-out in predicting dilation. The degree of locality or structure in a graph predicts how well it can be supported by a broadcast architecture.

Another aspect of the definition phase was in the choice of terminology. The idea of using c-graphs versus p-graphs as a model for ANN interconnect provided a simplicity and led to the use of such measures as density of interconnect. The concept of a mapping, with its cost based on the dilation factor, was a natural development, given this basic model.

It is common knowledge that a cross-bar switch can be built to allow n nodes to pair-wise communicate at an $O(n^2)$ cost. This research showed supporting full ANN interconnect is $O(n^3)$.

That the same cubic rate of growth is true in a less than globally interconnected system was surprising, but significant given the large degree of the nodes involved.

This rapid rate of growth in the silicon area required for connections leads to the conclusion that it is not feasible to provide direct connections, except for the most limited cases of nearest neighbor architectures. This conclusion is in direct contradiction with the designs of many research groups.

If direct connection is not feasible, then multiplexed connections are required, which leads to the use of multiplexed PNs and digital communications. Computation can still be performed in an analog manner, but digital communications allows for more efficient multiplexing of wires. While the analyses described in this dissertation are all oriented toward the use of time-division multiplexing, some form of frequency-division multiplexing is not ruled out by any of the assumptions or conclusions.

Once the c-graph and p-graph are seen as not needing to be identical, because of virtual connections and multiplexing of PNs, the problem of how to map one to another needs to be solved. This paper briefly touches upon this topic, with the discussion of matrix splitting versus graph embedding. In general, mapping of one graph to another has been shown to be NP-complete. The difficulty of solving this problem is reduced by the use of broadcast regions which make multiple PNs equivalent.

Examining the c-graphs of ANN models led to the insight that broadcast might be an effective implementation technique. Testing the feasibility of this idea required the development of methods of determining when broadcast was preferable to PTP and how to determine the effectiveness of an interconnection architecture.

The concepts of physical broadcast, with the example of a broadcast tree, and virtual broadcast came from the conclusion that PTP architectures were inadequate to support the mes-

sage load of ANN models. The loop model of virtual broadcast is simple to implement and provides a low cost, effective and elegant solution to many problems. Other concepts, which came from the study of broadcast regions, include addressing techniques which assign an address to each PN in a subregion so overlapping regions support unique PN addresses of minimal length, the use of come-from addressing to add flexibility to designs while reducing message length, and the use of slotted protocols to further reduce message length while simplifying PN decoding design. Because of problems with locality, overlapping regions and augmented broadcast were invented as enhancements to the basic concept.

Virtual broadcast is a communication architecture instead of an interconnection architecture like physical broadcast. It can be implemented on any PTP p-graph simply by defining a new routing mechanism for each node and having a single generic message sent by each CN rather than individual messages targeted to specific destinations. For systems with dense interconnections, it is a very powerful concept.

Virtual broadcast can be implemented as described in Chapter 6, where all nodes within some region receive a set of messages. Alternatively, it could be implemented using any PTP architecture, but with a *forward count* in each message that is decremented when the message is forwarded to a new node. Adding such a refinement would allow for virtual broadcast regions that could be dynamically modified in size or shape. Given the flexibility of virtual broadcast and its ability to be built upon any PTP architecture, it is a significant tool for implementation of ANN models.

The results of Chapters 6 and 7 show the cost of implementing broadcast is $O(N)$ compared to the $O(N^{3/2})$ cost of PTP. Both PTP and broadcast architectures have a performance time $O(N)$, which reflects the N size of the networks. Virtual broadcast can be designed to have a minimal cost implementation for a fixed communication delay of the same order of magnitude as

the other options.

While this research was analytical and no simulations were performed, the Adaptive Solutions CNAPS chip is a commercial product that resulting in part from this research. Broadcast interconnect has thus been shown to be effective for inexpensive silicon implementations and allows the CNAPS chip to get n^2 connections in n clocks. Efficiency of the CNAPS system is high because most of the emulated networks are dense.

In addition to providing the theoretical foundation for a commercial system, this research has resulted in several patents:

"Neural-Model, Computational Architecture Employing Broadcast Hierarchy and Hypergrid, Point-to-Point Communication." Dan Hammerstrom, Patent No. 4,983,962, issued January 8, 1991

"Neural-Model Computational System with Multi-dimensionally Overlapping Broadcast Regions," Dan Hammerstrom and Jim Bailey, Patent No. 4,918,617, issued April 17, 1990.

"Neural-Model, Information-Handling Architecture and Method" Dan Hammerstrom and Jim Bailey, Patent No. 4,796,199, issued Jan. 3, 1989.

8.2. Further Directions

In Chapter 7, simple examples of implementations have been presented. Each of these could be simulated to show how it would perform in relationship to other implementation possibilities. Such simulations would require more accurate estimations of node complexity and message generation patterns. A related issue is tracking further neurophysiological research to find better models for message loads and interconnection graphs.

The possibility of a virtual-broadcast system based on message relay counts as mentioned above has interesting theoretical possibilities for support of ANN models where the probability of a connection declines with the distance from the source node. How such a system could be built

and how well it would perform are two more possible research directions. Solving these problems includes dealing with issues of message congestion, routing algorithms, robustness in the presence of faults, and which ANN models could be well supported by such an architecture.

The use of a tree structure to support physical broadcast was proposed. Another topic to consider is what other structures are feasible. For example, how many nodes could a bus based system support and would a hierarchy of buses perform better than a broadcast tree? More exploration of pipelining in broadcast is merited to determine how it could be implemented, what the cost is, does the pipeline need to be flushed between communication cycles, and how much of the interconnect can be laid down over the top of PNs.

8.3. Summary

In summary, this research has opened up a new area of interconnection architectures for the new, important computational model of ANNs, and examined the most efficient techniques thus for presented for implementing ANNs in standard silicon. It has explored some of the consequences and benefits of these architectures. The initial development of supporting theories has been done. New research directions are indicated that further develop on these ideas.

Bibliography

[AlA87] Alspector, J. and Allen, R. B., "A Neuromorphic VLSI Learning System," *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, Cambridge, MA, 1987.

[AKM85] Arnould, E., Kung, H. T., Menzilcioglu, O. and Sarocky, K., "A Systolic Array Computer," *Proceedings of 1985 IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1985.

[BeW91] Bergerson, K. and Wunsch, D., "Commodity Trading Model Based on a Neural Network - Expert System Hybrid," *International Joint Conference on Neural Networks*, vol. I(1991), .

[Bow90a] Bower, J., "Reverse Engineering the Nervous System: An Anatomical, Physiological, and Computer Based Approach," in *An Introduction to Neural and Electronic Networks*, S. Zornetzer, J. Davis and C. Lau (ed.), Academic Press, 1990.

[Bow90b] Bower, J., "Piriform Cortex and Olfactory Object Recognition," in *Olfaction: A Model System for Computational Neuroscience*, J. Davis and H. Eichenbaum (ed.), MIT Press, 1990.

[Bra89] Braitenberg, V., "Some Arguments for a Theory of Cell Assemblies in the Cerebral Cortex," in *Neural Connections; Mental Computations*, L. Nadel, L. Cooper, P. Culicover and R. Harnish (ed.), MIT Press, 1989.

[CaG87] Carpenter, G. A. and Grossberg, S., "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and*

Image Processing, 1987.

[Col87] Cole, B., "Inova Brings Wafer-Scale Integration to Market," *Electronics*, January 8, 1987, pp. 91-95.

[Con87] Conwell, P. R., *Effects of Connection Delays in Two State Model Neural Circuits*, Proceedings of the ICNN, San Diego, CA, June 1987.

[Dal86] Dally, W., "A VLSI Architecture for Concurrent Data Structures," Ph.D. Dissertation, California Institute of Technology, 1986.

[Dal90a] Dally, W., "Performance Analysis of K-Ary N-Cube Interconnection Networks," *IEEE Transactions on Computers*, vol. 39, 6 (June 1990), .

[Dal90b] Dally, W., "Wire-Efficient Communication Networks," in *VLSI and Parallel Computation*, R. Suaya and G. Birtwistle (ed.), Morgan Kaufmann, 1990.

[DeJ73] Denning, P. J. and Jr., E. G. C., *Operating Systems Theory*, Prentice Hall, Inc, 1973.

[EOW90] Eichenbaum, H., Otto, T. A., Wible, C. G. and Piper, J. M., "Building a Model of the Hippocampus in Olfaction and Memory," in *Olfaction: A Model System for Computational Neuroscience*, J. Davis and H. Eichenbaum (ed.), MIT Press, 1990.

[Fah79] Fahlman, S. E., *NETL: A System for Representing and Using Real-World Knowledge*, MIT Press, Cambridge, MA, 1979.

[Fah80a] Fahlman, S. E., "The Hashnet Interconnection Scheme," CMU-CS-80-125, Carnegie-Mellon University, Pittsburgh, PA, June 1980.

[Fah80b] Fahlman, S. E., *Design Sketch for a Million-Element NETL Machine*, Carnegy-Mellon University, Pittsburgh, PA, 1980.

[FeB82] Feldman, J. A. and Ballard, D. H., "Connectionist Models and Their Properties," *Cognitive Science*, 1982.

[FMI83] Fukushima, K., Miyake, S. and Ito, T., "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, 5 (September/October 1983), pp. 826-834.

[GGK83] Gottlieb, A., Grishman, R., Kruskal, C. P., McAuliffe, K. P., Rudolph, L. and Snir, M., "The NYU Ultracomputer - Designing a MIMD Shared Memory Parallel Computer," *IEEE Transactions on Computers*, vol. C-32(February 1983), pp. 175-189.

[GrV87] Graf, H. P. and Vegvar, P., "FAM 22.1: A CMOS Associative Memory Chip Based on Neural Networks," *IEEE International Solid-State Circuits Conference*, 1987, pp. 304-305,437.

[GAL89] Granger, R., Ambros-Ingerson, J. and Lynch, G., "Derivation of Encoding Characteristics of Layer II Cerebral Cortex," *Journal of Cognitive Neuroscience*, vol. 1, 1 (Winter 1989), .

[GAA90] Granger, R., Ambrose-Ingerson, J., Anton, P., Whitson, J. and Lynch, G., "Computational Action and Interaction of Brain Networks," in *An Introduction to Neural and Electronic Networks*, S. Zornetzer, J. Davis and C. Lau (ed.), Academic Press, 1990.

[Ham92a]

Hammerstrom, D., Personal Communication, 1992.

[Ham92b]

Hammerstrom, D., A Highly Parallel Digital Architecture for Neural Network Emulation, Unpublished Manuscript, 1992.

[Hil85] Hillis, W. D., *The Connection Machine* , MIT Press, 1985.

[Hin84] Hinton, G., "Distributed Representations," CMU-CS-84-157, Carnegie-Mellon University, Pittsburgh, PA, October 1984.

[HSA84] Hinton, G. E., Sejnowski, T. J. and Ackley, D. H., "Boltzmann Machines: Constraint Satisfaction Networks that Learn," CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, PA, May 1984.

[Hin87] Hinton, G., "Connectionist Learning Procedures," CMU-CS-87-115, Carnegie-Mellon University, Pittsburgh, PA, June 1987.

[Hop82] Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79(April 1982), pp. 2554-2558.

[HoT89] Hopfield, J. and Tank, D., "Neural Architecture and Biophysics for Sequence Recognition," in *Neural Models of Plasticity: Experimental and Theoretical Approaches*, J. Byrne and W. Berry (ed.), Academic Press, 1989.

[HoZ81] Horowitz, E. and Zorat, A., "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI." *IEEE Transactions on Computers*, vol. 30, 4 (April 1981), .

[JHG] Jackel, L. D., Howard, R. E., Graf, H. P., Straughn, B. and Denker, J. S., "Artificial Neural Networks for Computing," *Journal of Vacuum Science Technology*, , pp. 61-63.

[Jac91] Jackson, D., "A Decomposition of Back Propagation Artificial Neural Networks for Multicomputers," Masters Thesis, Oregon Graduate Institute, 1991.

[Kle76] Kleinrock, L., in *Queueing Systems, Volumes 1 and* 2, Wiley, New York, 1975-1976.

[KoS89] C. Koch and I. Segev, eds., *Methods in Neuronal Modeling*, MIT Press, 1989.

[Lei92] Leighton, F. T., in *Introduction to Parallel Algorithms and Architectures: Arrays - Trees - Hypercubes*, Morgan Kaufmann, 1992.

[Lei85] Leiserson, C. E., "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing." *IEEE Transactions on Computers*, vol. 34, 10 (October 1985), .

[Lyn86] Lynch, G., *Synapses, Circuits, and the Beginnings of Memory*, MIT Press, 1986.

[LGL89] Lynch, G., Granger, R., Larson, J. and Baudry, M., "Cortical Encoding of Memory: Hypothesis Derived from Analysis and Simulation of Physiological Learning Roles in Anatomical Structures," in *Neural Connections; Mental Computations*, L. Nadel, L. Cooper, P. Culicover and R. Harnish (ed.), MIT Press, 1989.

[May88] May, N., "Fault Simulation of a Wafer-Scale Neural Network," Tech. Report CS/E-88-020, Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon, May 1988.

[Maz87] Mazumder, P., "Evaluation of On-Chip Static Interconnection Networks," *IEEE Transactions on Computers*, vol. 36, 3 (March 1987), .

[Mea86] Mead, C., *Analog VLSI and Neural Systems*, California Institute of Technology, Pasadena, CA 91125, 1986.

[McM] Mead, C. A. and Mahowald, M. A., "A Silicon Model of Early Visual Processing," *Neural Networks*, , pp. 91-97.

[MeR79] Meade, C. A. and Rem, M., "Cost and Performance of VLSI Computing Structures," *IEEE Journal of Solid-State Circuits*, vol. 14, 2 (April 1979), .

[MeC80] Meade, C. A. and Conway, L., in *Introduction to VLSI Systems*, Addison Wesley, 1980.

[Mea91] Means, E., "Designs for a Cortically-Inspired Neurocomputer Architecture," Masters Thesis, Oregon Graduate Institute, 1991.

[O'K89] O'Keefe, J., "Computations the Hippocampus Might Perform," in *Neural Connections; Mental Computations*, L. Nadel, L. Cooper, P. Culicover and R. Harnish (ed.), MIT

Press, 1989.

[RMB86] Raffell, J., Mann, J., Berger, R., Soares, A. and Gilbert, S., "A Generic Architecture for Wafer-Scale Neuromorphic Systems," *Proceedings of the ICNN*, San Diego, CA, June 1986.

[Rol89] Rolls, E., "Functions of Neuronal Networks in the Hippocampus and Neocortex in Memory ," in *Neural Models of Plasticity; Experimental and Theoretical Approaches*, J. Byrne and W. Berry (ed.), Academic Press, 1989.

[Rol90] Rolls, E., "Principles Underlying the Representation and Storage of Information in Neuronal Networks in the Primate Hippocampus and Cerebral Cortex," in *An Introduction to Neural and Electronic Networks*, S. Zornetzer, J. Davis and C. Lau (ed.), Academic Press, 1990.

[RuH88] Rudnick, M. and Hammerstrom, D., "Physical Broadcast Structure," Tech. Report CS/E-88-018, Oregon Graduate Center , April 1988.

[RuM86] D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1 and 2, Bradford Books/MIT Press, Cambridge, MA, 1986.

[Sei90] Seitz, C. L., "Concurrent Architectures," in *VLSI and Parallel Computation*, R. Suaya and G. Birtwistle (ed.), Morgan Kaufmann, 1990.

[SeR86] Sejnowski, T. J. and Rosenberg, C. R., "NETtalk: A Parallel Network that Learns to Read Aloud," JHU/EECS-86/01, The Johns Hopkins University, 1986.

[ShB79] Shepherd, G. and Brain, T. S. O., Oxford University Press, Second Edition 1979.

[She90] Shepherd, G., "Computational Structure of the Olfactory System," in *Olfaction: A Model System for Computational Neuroscience*, J. Davis and H. Eichenbaum (ed.),

MIT Press, 1990.

[SSA89] Squire, L., Shimamura, A. and Amaral, D., "Memory and the Hippocampus," in *Neural Models of Plasticity; Experimental and Theoretical Approaches*, J. Byrne and W. Berry (ed.), Academic Press, 1989.

[Sto81] Stone, H. S., "Parallel Processing with the Perfect Shuffle," *IEEE Transactions on Computers*, vol. 20, 2 (February 1981), .

[Tho79] Thompson, C. D., "Area-Time Complexity for VLSI," *Caltech Conference on VLSI*, January 1979.

[Utt55] Uttley, A., The Probability of Neural Connexions, 1955.

[Van85] Van Essen, D., "Functional Organization of Primate Visual Cortex," in *Cerebral Cortex*, vol. 3, E. Jones and A. Peters (ed.), Academic Press, 1985.

[VaA90] Van Essen, D. and Anderson, C., "Information Processing Strategies and Pathways in the Primate Retina and Visual Cortex," in *An Introduction to Neural and Electronic Networks*, S. Zornetzer, J. Davis and C. Lau (ed.), Academic Press, 1990.

[Whi89] White, E., *Cortical Circuits; Synaptic Organization of the Cerebral Cortex Structure, Function, and Theory*, Birkhauser, 1989.

[WiM88] Winsor, D. C. and Mudge, T. N., "Analysis of Bus Hierarchies for Multiprocessors," *Computer Architecture News*, vol. 16, 2 (May 1988), .

Appendix A

Derivations of Results

This appendix contains the derivations of some of the formulas of Chapter 6.

Result 6.1-a: For a uniform distribution of destinations within a system of n rows and n connected columns, with m messages entering the system via each row each time interval, the expected vertical message load within a column is $m(n-1)/2n$ per edge. A connected column is a column of nodes which forms a loop.

Argument:

There are x messages that enter the column from each row each time interval. Of these, x/n will be destined for each node within the column, because of the uniform distribution of destinations. Number the nodes of the column sequentially from 1 to n following the message routing order; i.e. messages pass from node k to $k+1$ and from n to 1. All nodes are topologically equivalent so the choice of which node is assigned number 1 may be arbitrarily made. Choose the pair of nodes n and 1, then calculate the incremental bandwidth required between them for the contribution from each row in the system. The numbering was started with an arbitrary node, so this result can be generalized to the edge between any pair of adjacent nodes.

For any node i, of the x messages received by it, $x(n-1)/n$ will be targeted for other nodes in the loop and x/n will be for internal

consumption. First consider node 1; none of the messages it received and passed on to node 2 will return via the edge between node n and itself since all nodes in the loop have been visited by a given message packet when it reaches n. If any messages remained, they were not for targets in the column and would not have entered it in the first place. Next consider messages which entered the column via node 2; only messages destined for node 1 will remain in packets which enter via 2 and have visited nodes 2, 3, ... n. Thus node 2 will contribute x/n messages to the edge between n and 1. Similarly for node i; it will contribute the $x(i-1)/n$ messages intended for nodes 1, 2, ... $i-1$ to the edge between nodes n and 1. Finally, node n will contribute its entire forwarded packet of $x(n-1)/n$ messages. Summing these contributions gives an expected number of messages crossing the edge of $x\sum_{j=1}^{n}((j-1)/n) = x/n\sum_{j=1}^{n-1} j = x(n-1)/2$.

If m messages enter the system via a row and there are n columns with equal likelihood of being a destination, m/n messages must be injected into each column from each row each time interval. Substituting m/n for x provides the desired formula.

Result 6.1-b: For the system described in Result 6.1-a, the total message load to be handled by a column is $m(n-1)/2$.

Argument:

Each column has n edges. The load per edge is $m(n-1)/2n$ by Result 6.1-a. The total load per column is thus $m(n-1)/2$.

Result 6.2-a: For a mesh connected set of nodes and for a column of n nodes with each accepting m/n messages per time interval and a uniform distribution of message destinations among the nodes in the column, the expected message load across the edge between node j and $j + 1$ is $2mj(n - j)/n^2$. Here j is an index indicating the distance of the node from the top end of the column.

Argument:

Define x as the number of messages that enter the column via each row. Number the nodes in the column beginning with 1 at the top and ending with n at the bottom. Assume the column has reached a steady state where x messages are entering via each node each time cycle. If the distribution of destinations is uniform, each node will consume x/n messages that enter it from the horizontal dimension each cycle out of the total of x messages consumed by it from all sources.

Choose an arbitrary pair of nodes j and $j + 1$ where $1 \leq j < n$. Consider the messages that enter the system below a line dividing the two nodes each time interval. By the uniform distribution of destinations, with j nodes above the line, j/n of the messages will be for destinations above it and $(n - j)/n$ for ones below. Similarly, j/n of the messages originating below the line are for destinations above it and $(n - j)/n$ for ones below. If jx messages enter the system above the line and the probability for any message is $(n - j)/n$ that it is destined for a node below the line, $jx(n - j)/n$ messages will need to cross the line. Similarly, $(n - j)xj/n$ will need to cross the line in the opposite direction. The expected message load across the edge is thus the sum of

these two values or $2xj(n-j)/n$. Substituting the value m/n for x provides the desired formula $2mj(n-j)/n^2$.

Result 6.2-b: For a mesh, as in Result 6.2-a, with a uniform distribution of destinations both horizontally and vertically and m messages entering each row every time interval, the total expected load in each column is $m(n-1)(n+1)/3$.

Argument:

There are $n-1$ edges with a load per of $2mj(n-1)/n^2$. Thus the total bandwidth required is $\sum_{j=1}^{n-1} 2mj(n-1)/n^2 = \frac{2m}{n^2}\sum_{j=1}^{n-1} j(n-j) = \frac{2m}{n^2}\left[\sum_{j=1}^{n-1} jn - \sum_{j=1}^{n-1} j^2\right] = \frac{2m}{n^2}\left[\frac{n^2(n-1)}{2} - \frac{n(n-1)(2n-1)}{6}\right] =$ $m(n-1)(n+1)/3n$.

Result 6.3: For a torus, with uniform distribution of message sources and destinations, the expected bandwidth for each row or column is $\frac{B(n-1)}{2}$ with a total system wire cost of $2n^2(n-1)BW\sqrt{A}$.

Argument:

By Result 6.1-a, the expected message load per edge in a loop of uniform probability destinations is $x(n-1)/2$ where x messages enter via each node during each time interval. The rows of a full torus form such a loop with B messages being injected by each node and B messages leaving as they reach their destination column. Here B is the bandwidth for messages to leave each node. Thus the row edge cost is $B(n-1)/2$.

By analogous argument, the column edge load is the same value. Although the routing is non-symmetric, B messages enter each row and

column of the system each time interval. With symmetric topologies, analogous arguments can be applied.

Thus the entire row cost is n rows of n edges with $B(n-1)/2$ each or $\frac{n^2B(n-1)}{2}$. The same number of columns with the same edge loading gives a total column cost of $\frac{n^2B(n-1)}{2}$. Each wire is of length $2\sqrt{A}$ and width W for a total wire cost of $2n^2(n-1)BW\sqrt{A}$.

Result 6.4: The wire cost for a two-dimensional unconnected mesh is $\frac{2n(n-1)(n+1)BW\sqrt{A}}{3}$.

Argument:

The message traffic in a single row of n nodes with each producing m messages can be generalized to two dimensions in a manner analogous to the torus argument above. Messages are fed into the first dimension as they are created. The consumption in the first dimension is the injection into the columns. The final consumption of messages by the destination nodes occurs from the second dimension. Thus each row has a message load across its internode boundaries that follows the function $2Bi(n-i)/n$ where $1 \le i \le n-1$ is the index of the boundary from an edge or a total row bandwidth of $\frac{B(n-1)(n+1)}{3}$.

Each node is injecting B messages into the vertical columns, because of the uniform distribution of destinations. Thus each column has the same function defining its bandwidths. Total expected bandwidth is the same in each dimension and for the system is

$\frac{2n(n-1)(n+1)B}{3}$. The wire cost is $W\sqrt{A}$ times this, yielding the desired formula.

Biographical Sketch

I was born in Eugene, Oregon on November 29, 1951. My undergraduate education was at Eastern Oregon State College and Oregon State University, where I received a BS in Mathematics in 1973. I obtained a MS in Computer Science from Brigham Young University in 1986.

I have worked as a programmer and software engineering manager for fifteen years in a variety of different companies and fields. During the research reported in this dissertation I received two patents:

"Neural-Model Computational System with Multi-dimensionally Overlapping Broadcast Regions," Dan Hammerstrom and Jim Bailey, Patent No. 4,918,617, issued April 17, 1990.

"Neural-Model, Information-Handling Architecture and Method" Dan Hammerstrom and Jim Bailey, Patent No. 4,796,199, issued Jan. 3, 1989.

My publications include:

"A Monitor for the Transparent Interfacing of a Microcomputer to a Host System", Masters Thesis, Brigham Young University

"A Program for Mapping CNNs to Physical Architectures", Oregon Graduate Institute Techinical Report CS/E-88026

"Why VLSI VLCN's Require Multiplexing", with D. Hammerstrom in Proceedings of First International Conference on Neural Networks, 1987

"Silicon Association Cortex", with D. Hammerstrom, J. Mates and M. Rudnick in "An Introduction to Neural and Electronic Networks"