How to Make a Billion Connections

Jim Bailey, Dan Hammerstrom

Oregon Graduate Institute
Department of Computer Science and Engineering
20000 NW Walker Rd
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 86-007
August 1986

How to Make a Billion Connections

Jim Bailey
Dan Hammerstrom
Oregon Graduate Center
19600 N.W. von Neumann Dr.
Beaverton, Oregon 97006

Technical Report No. CS/E-86-007
August 1986

Abstract

This paper addresses some of the problems involved in implementing a connection network consisting of a million nodes with a billion connections between them. It is organized into three general parts. The first explains what connection networks are, how they work and what some of the proposed implementations are. The second portion is a preliminary comparison of some of the interconnect strategies that have been suggested for parallel systems and how the connectionist model maps onto them. A set of metrics for analyzing different interconnection topologies are defined. They include *speed of communication, interconnect cost, locality of communication and degree of fault tolerance.* Several potential interconnect architectures are compared using the given metrics. The candidates studied fall into one of five general classifications: direct implementation of the connection network in silicon, a simplistic model such as a grid, a message routing model such as the hypercube, a shared memory model and a new proposed solution - the broadcast hierarchy. Preliminary results show that some interconnection structures are incompatible with most existing connection network models.

1. Introduction

Recently, the use of a *connectionist* model has gained popularity as an alternative computational paradigm for Artificial Intelligence systems that display cognitive behavior. Connectionist models are based on the structure of the brain's neural networks [2,35]. They consist of many simple processors that modify the strength of their interconnections to store data. Because of the origin of these models, they are expected to exhibit computational behavior similar to that of the brain. The most important of these behaviors is the ability to process an input and reach a conclusion in a few steps instead of the thousands of instructions of a typical sequential computer program. A human being, for example, can recognize an image flashed on a screen and respond in less than one second, using neurons that have firing times on the order of milliseconds [12].

The processing elements in a connectionist network do not individually solve the problem, it is by communication between them that the solution is generated. To quote Feldman [12], "The fundamental premise of connectionism is that individual neurons do not transmit large amounts of symbolic information. Instead they compute by being appropriately connected to large numbers of similar units." This is done by generating, in parallel, multiple competing hypothesis and then *relaxing* to a single *best-match* interpretation.

In addition to the interest of AI researchers and computer architects in these models, some researchers in neuroscience are now using them to analyze brain functions. One pertinent quote from Braitenberg's paper on Cortical Architectonics [4] reads: "The most significant development since the time of von Economo is the shift of emphasis from the brain considered as a collection of cells to the brain considered as a network of fibers, from the neuropathologist's view of the brain to that of the computer engineer; in short, from Nissl pictures to wiring diagrams." Another supporting comment is from Creutzfeldt [6], "In more general terms, the neocortex can be considered as a *cooperative network*, in which the activities of the individual elements interact with each other through inhibition."

Connection models are well suited to solving problems requiring differentiation between alternatives, such as context association. An interesting use has been the work by Hopfield and associates [26,27], where they have created a connection network that solves the traveling salesman problem, i.e., determines the shortest route that visits each of a group of cities. Problems with soft constraints, or where limits can be bent, are also ideal for the connectionist paradigm.

The initial design for computational elements of this nature, the perceptron, was suggested by Rosenblatt [33] and was based on the neural learning model conceived by Hebb [19]. Perceptron research generated a learning algorithm, the *Perceptron Learning Rule*, for single layers of processors. It was not possible to generalize this learning rule to systems with multiple layers of perceptrons, because in modifying weights on paths that consist of several perceptrons, one could not easily determine which contributed the most to the error response and thus how to change its input weights. The resulting limitations of single level perceptrons were well proven by Minsky and Papert [31], and further progress on computational systems based on neural networks waited for the development of models with learning methods for multiple layers.

The realization by Hopfield that it was possible to use a model based on the energy minimizing behavior of physical networks to determine the appropriate modifications of connections [25] has lead to much of the current interest in connectionist architectures. Hinton, Sejnowski and Ackley carried the energy model further with the *Boltzman machine*, a system that uses statistical probabilities from Boltzman mechanics to determine the connection weight adjustments [23]. The problem with this work is that the Boltzman machine, while able to demonstrate multi-level, recursive learning, is slow [7,13].

A major new development is the work by Rumelhart and Hinton at UCSD [34]. They have multiple-level parallel learning networks that learn by the back propagation of errors, in other words, back propagation provides a means to program systems containing multiple levels of perceptrons. Discrepancy at the highest level causes a modification of the input weight for that node. The error, modified by the previous weight, is passed back to the prior level. Weights are allocated by a process of taking derivatives of the continuous output functions. The advantage of this model is that it is deterministic and converges quickly. The back propagation algorithm was used in the NETtalk speech synthesis system developed by Sejnowski and Rosenberg [38].

Several researchers have developed other computing models, based on the general connectionist scheme, that are able to recognize patterns [3] and parse natural language input [5,32,39]. Grossberg has shown that much of the research results from human cognition studies can be explained by the use of analytic models based on formal models of neural networks. He has contributed several major insights to connectionist research, such as competitive learning via noise suppression, the use of on-center/off-surround neural models and STM (Short Term Memory) by adaptive resonance [17,21].

Most connectionist models share common characteristics [35]. Each model contains a set of processing units that calculate a function of their inputs; an activation rule that determines the output of each processing element based on a threshold comparison function; a relaxation rule that allows the network to resolve the competing hypotheses; a pattern of connectivity between the processing elements; and some method to modify the strengths, or weights, of the connections to allow the system to learn dynamically. The factors that differentiate the models are the choice between a continuous output function, as in Grossberg's work [17], and a discrete function, as used by Hopfield [25]; the degree of interconnect, from Hopfield's full crossbar to Hammerstrom's sparse networks [18]; the specific function computed; the choice between a sigmoid (and the degree of linearity of the sigmoid) or a threshold function for activation; and the knowledge representation paradigm, whether localized, as Feldman would have it [12] or distributed, as proposed by Hinton [24].

Our research is directed towards solving the problems inherent in building a system capable of emulating very large connectionist networks. It is apparent that the problems researchers are beginning to address, such as speech and vision, will require systems with many interconnections between the nodes. Preliminary results indicate that a system of 10^9 connections is close to the maximum that can be attained using a reasonable technology, so our target system is 10^6 nodes with 10^9 connections between them. Incidentally, it is not known how much *intelligence* a system with 10^9 connections will exhibit[1], or if it will be able to compete favorably with current AI systems.

A connection network containing 10^9 connections is significantly beyond the capabilities of any existing computer. Currently, the largest running simulator on a general purpose computer, at the University of Rochester [11], is able to support less than 4×10^4 connections and nodes using a 128-node BBN Butterfly. Thus, it is apparent that specialized processors, such as the one being developed at OGC, are necessary to support connectionist/neural network modeling research that requires systems with large numbers of connections. One such project is the ADAPT Mark IV system being developed by Hecht-Nielsen at TRW which will support networks with up to 5.5×10^6 connections [20].

The architectural issues being addressed by our project include trade-offs between non-local communication and local computation, the design of a VLSI processing element, the degree of fault tolerance of such a system, and an efficient interconnect topology, with

[1] A snail, for example, has roughly 10^6 neurons with 10^9 connections between them.

our ultimate goal being a wafer scale implementation. The purpose of this paper is to provide a preliminary answer to the question, "what is the best interconnect topology for emulating connectionist networks?"

In mapping a connectionist model to silicon[2], one obvious disparity is the difference in interconnect characteristics. Connection networks are characterized by large numbers of low bandwidth connections and VLSI systems by a few, high bandwidth connections. The only solution is to multiplex the physical communication channels. As shown later, the connectivity requirements of the connectionist model greatly influence implementation cost and speed, along with determining how best to multiplex the physical channels and how they should be structured. No matter what the computational paradigm of the connectionist network, there is the need for large quantities of short internode messages. As as result, it is critical that the interconnection topology be chosen carefully.

The communication requirements are critical to VLSI design since they greatly affect speed, reliability and cost. Speed is influenced by the time required to traverse long runs of conductors and pass through intermediate nodes that must buffer or otherwise process the message. Reliability is the resistance of the design to failures caused by faults in the system. Increasing the chip area causes an increase in the likelihood of faults. Increasing the fault tolerance of the system reduces the redundancy that must be built in to guarantee a given level of functionality. Since the cost of a wafer is essentially fixed, decreasing the amount of redundancy and interconnect area reduces the cost per unit of *useful* area.

The remainder of this paper addresses these problems and suggests a family of interconnection architectures that may be used to build a connectionist/neural network emulator. The approach taken is to define a set of metrics to use in comparing interconnection architectures and apply them to several potential candidates. First, it is necessary to provide some basic definitions and assumptions. These can be found in Section 2. They are followed, in Section 3, by the definitions of the metrics and, in Section 4, by the comparisons themselves. The paper concludes by presenting a summary of the results and suggesting the best interconnection architecture.

2. Definitions and Assumptions

A connection network is defined as a set of processing nodes together with the interconnections between them. The network that is perceived by the user is emulated, rather than physically implemented, in all the following models (except direct connect). This is much like a virtual memory system where the memory architecture the user sees is not necessarily the same as that of the underlying physical system. The virtual connectionist network is asynchronous in operation with each node functioning independently. By definition, its size is 10^6 nodes with 10^9 total connections, or 10^3 connections per node. The number of nodes is not as important to the analysis as the number of connections, since it is the connections and the associated weight and address tables that contribute the most to the required chip area and determine system functionality. Although the goal is to have a system that can emulate most models from the connectionist literature, a simpler target has been used in this preliminary study. Since the purpose of this paper is to consider the relative performance of different interconnection architectures, the functionality of the network is not critical. It is necessary that the network be able to function with a relatively sparse

[2]2-dimensional silicon has been chosen as a target medium because of its density, reliability, and ease of use. Our project can obviously benefit from advances in connectivity by 3D VLSI and optical communication developments, but such technologies are not considered here since our intent was to consider technology that will be readily available in the next few years.

interconnect.

Although the analyses presented here do not require the specification of the network functionality, understanding what a node is and how it functions will help in following the discussion. It is not intended that the computing model mirror the brain, therefore, a processing node can be considered the equivalent of either a single neuron or a large group of closely correlated neurons. Figure 1 illustrates one possible type of node, the Sigma-Pi. A Sigma-Pi node calculates the sum of the products of the weighted inputs and outputs a result based on a sigmoid function.

More precisely, a CN, or *Connection Node*, is a simple computing element with inputs $I_1 \cdots I_n$. What function is computed by the CN is dependant on the particular connectionist model being emulated. An input may be the output of a CN or it may come from the external environment. Each input, I_j, has a weight, W_j, associated with it. The function of a weight is similar to the facilation of the synapse. If two neurons correlate in their firings, the strength is increased, otherwise it is decreased. The weights may be statically programmed or modified dynamically by a learning algorithm. Since this paper deals with architectural issues and not with learning models, the meaning and derivation of the weights will not be considered further. The I_is, and in some models the W_is, are time varying. In addition, the computed funtion may be state dependent.

There are four different types of distance used in this paper. The first is Hamming distance and will be referred to as $distance_h$. The Hamming distance between two vectors is the number of positions in which they differ. The second type of distance, is the distance

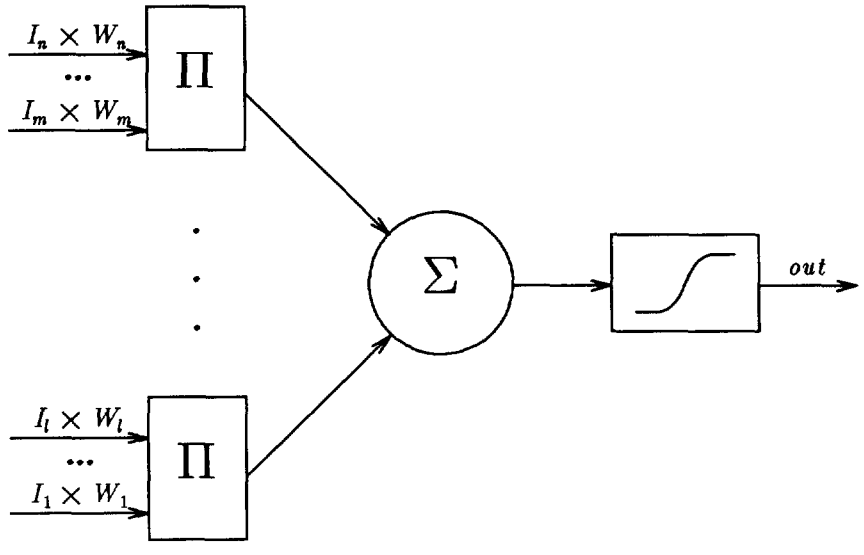

Figure 1. Sigma Pi Node

between two nodes in the connection network, $distance_c$. The connection network can be considered as a directed graph, where there is an edge from node i to node j when the output of CN_i is an input to CN_j. The distance$_c$ from a given CN to any other CN is the length of the shortest path starting at the source node and leading to the destination node, and determines the influence the source node has on the destination node. If there is no such path the distance is defined to be infinite. The distance$_c$ from CN_a to CN_b is not necessarily the same as the distance$_c$ from CN_b to CN_a.

The remaining two distances are related to the implementation and are $distance_g$, the distance in the interconnection graph and $distance_p$, the physical distance. It is necessary to map the connection network to a physical network. This is a two stage graph embedding problem where first the connection network must be embedded into some interconnection graph. While the connection network can be considered as a directed graph, the interconnection topology is defined to be a non-directed graph, since communication paths are assumed to be bi-directional. The second portion of the embedding problem is mapping the interconnection graph to silicon.

Each PN, or *Physical Node,* is an independent processor capable of standard arithmetical and logical computations together with sufficient memory to hold all required connection network state information. The state data includes the most recent value for each input, the weight assigned to each incoming link and the addresses of the CNs where outputs are sent. Some connectionist/neural network models require additional state bits, these are not allowed for here. A PN always accepts an update to the stored state. A physical model for this latter capability is a set of I/O-processors that listen to the communication lines and update a multi-ported associative memory. Weights are always assigned and stored on the incoming side, because it is the destination node that is able to correlate its firing with a given input to determine the correct value for a particular weight in most dynamic weight modification algorithms.

Locality is an important characteristic of neural networks. Since connections are expensive in terms of energy and space requirements, and in the genetic material required to describe differentiated wiring patterns, nature only uses the minimum number of connections required to perform the necessary functions with the necessary reliability. Braitenberg [4], when discussing his research on mouse brains, stated: "A set of 10 million neurons of various types are connected together by two sorts of fibers: intercortical, short-range (= a few hundred microns) and corticocortical, long-range connections." He later showed that most of the connections are short range. One common characterization of the brain [3,4,6] suggests that the probability of a connection is a function of the physical separation between two neurons.

Though hypothetical, it is our belief that the sparse connectivity of real neural systems results from:

(1) Redundancy of input data. Sensory input data (in particular auditory and visual) is both spatially and temporally redundant. Since neural systems reflect the environment in which they operate, we expect that significant connectivity reduction results from the influence of such correlations (see Hammerstrom [18]).

(2) Selective attention and limited short-term memory. Biological systems can only attend to a few simultaneous stimuli, thus limiting the degree of simultaneous interconnectivity.

(3) Abstraction. Higher order nervous systems, e.g. primates, have the ability to abstract or "chunk" groups of concepts into a new concept; this convergance of information performs an encoding that reduces inter-module bandwidth requirements [41].

(4) Functional decomposition. Despite the regularity and distribution of processing in neural systems, research has shown distinct functional areas within the human brain. The best example of this has been in the decomposition of the visual and auditory processing areas in primates.

In the interconnection network the distance$_g$ between two PNs can be represented as the number of communications links between them. In graph theoretic terms, it is the length of the shortest path between the two nodes. In assigning CNs to PNs, it is critical for performance that the CNs near each other in distance$_c$ are placed on PNs near in distance$_g$. To measure the average distance$_g$ between two CNs, it is necessary to define the locality of communication in both the connection and physical networks. Communication locality intuitively means the distance a typical message must travel, with a network exhibiting high locality having a lower typical distance than one with low locality. Since this intuitive definition is not precise enough to use in selecting an interconnect topology, a formal definition is now presented.

Definition: $R(n)$. Let $r_q(n)$ be the number of nodes in a graph reachable by paths of length n from some starting node q. Then $R_q(n)$ is the total number of nodes reachable in n steps, $R_q(n)=\sum_{i=1}^{n} r_q(i)$. $R(n)$ is the average such $R_q(n)$ for all nodes q. For regular graphs, such as those considered in this paper, and ignoring boundary conditions, $R(n)=R_p(n)$ for all nodes p.

Definition: $C(n)$. $C(n)=\sum_{i=1}^{n} \alpha^{-i} r(i)$, for some $\alpha \geq 1$.

Definition: L(k). $L=R^{-1}$, that is $L(R(n))=n$. In the following analyses, the definition of L has been modified such that $L=C^{-1}$.

The *reachability* of a connection network is $R(n)$ for the equivalent graph. Any connected graph, with N nodes, has a reachability of N in n_N steps for some $n_N \leq N$. For a fully connected graph, i.e., one with a connection between every two nodes, $n_N = 1$. For any infinite graph of regular degree, it is possible to specify a function for the number of nodes reachable in n steps. For example, a rectangular grid has a reachability function of $\sum_{i=1}^{n} 4i$. The *connectivity* function, $C(n)$, shows the number of nodes that can be *expected* to be *connected* to the starting node after n steps using an exponential function for the probability of each connection. The *locality* of a connection network is $L(n)$ for the equivalent graph, or the number of steps in the connection graph that are required to reach a specified number of nodes. Locality, connectivity and reachability can be defined for both physical and connection networks, in fact it is the relationship between locality$_g$ and locality$_c$ that determines the effectiveness of a particular interconnection architecture.

In the target connection network of 10^9 connections and 10^6 nodes, each CN is connected to 10^3 others, so, for each interconnection topology, it is necessary to calculate the number of steps required to reach 10^3 CNs or $L(10^3)$. Thus, once the reachability function is known the required number of interprocessor hops can be readily determined.

Although the characterization of the connectivity function with an exponential decay, α_g^{-d}, is simplistic, especially in light of the fact that neural networks appear to exhibit multi-modal connectivity distributions [6], it is suitable for the preliminary analyses presented here. Later research will require a more complex characterization of locality. Performing the following analyses while varying α_g from 1 to 2, and keeping the total number of connections per CN constant at 10^3, has shown that the relative rankings are not affected, although the architectures with increased reachability, Hyper-Cube and the

Broadcast Hierarchy, perform significantly better than grid for larger values of α_g. To provide one set of numbers for comparative purposes, a value of α_g of 1.25 has been chosen.

The effect of varying α_g is to vary the quality of the mapping from the *idealized* connection network to the interconnect graph. A value of 1 for α_g indicates an optimal mapping, one where the sum of $distances_g$ from each CN to all the CNs which it is connected to is minimal, since each CN is connected to the $nearest_g$ 10^3 CNs. A system with a larger value for α_g is less *good* i.e. the total $distance_g$ is non-minimal. Obviously, determining the optimum mapping is an NP-complete problem and for large networks less satisfactory mappings will be used. One area for future research is determining a value for α_g that reflects the typical mappings quality.

A formal restatement of these definitions is as follows.

Connection node - A node in the connection network, abbreviated CN.

Physical node - A node in the physical network, abbreviated PN.

$Distance_h$ - Hamming distance or the difference between two bit vectors. It is equal to the number of bit positions that differ. For example, 0010001 and 1010000 are Hamming distance two apart because the first and last bits are different.

$Distance_c$ - The length of the minimal path between two CNs in a connection network.

$Distance_g$ - The length of the minimal path between two nodes in a interconnection graph. Alternately, the least number of edges a message must traverse in traveling from one PN to another.

$Distance_p$ - The distance between two PNs in the physical network.

Locality - The number of steps required to reach a specified number of nodes.

Reachability - The number of nodes that can be reached from a given node within a specified number of steps.

Connectivity - The expected number of nodes reached within a specified number of steps using some probability function to calculate the odds of a given connection.

α_g - A constant for the reachability calculations in the interconnection graph and a measure of the effectiveness of the graph embedding of the connection network into the interconnection graph.

To reduce the analysis to one that can be adequately addressed within the scope of this paper, several simplifying assumptions have been made. Further research on interconnection problems will consist of relaxing these limitations and using statistical simulations to generate more precise values to replace the approximations used here.

The first set of premises relate to current VLSI technology. First, the available silicon area is infinite, so there is no need to worry about partitioning problems and the delays due to off-chip routing. Second, there are three levels of metal interconnect, this provides two levels for interprocessor routing and one level for the processor nodes themselves. Third, the minimal feature width is one micron with a pitch of one micron. Fourth, the clock speed of the chip is 20 MHz. Since all topologies will be considered using the same assumptions, the comparisons will be consistent; consequently, the precise details of these assumptions are not critical. The values used have been chosen to reflect the state of the art.

Even though connection networks are essentially asynchronous in operation, all physical systems described in this paper are assumed to be synchronous, though we suspect that a real implementation will consist of synchronous PNs operating asynchronously to one another, and that the connection networks themselves will have not inter-CN synchronization. Assuming synchronous networks simplifies the calculations of communication delays. In creating a physical system of this size, the problem of clock skew would be considerable. For the purposes of this paper, it is ignored.

The length of each message is assumed to be one bit to transmit a state change plus any bits required for source or destination addresses by the interconnect strategy. All data are sent serially, so the number of bits in a message impacts the time required for its transmission. The weights are small enough that they can be represented as four-bit quantities. Although a real system will require the ability to vary the size of the weights and outputs, to allow emulating different models, they are fixed here. A source address is necessary, except in the direct connection architecture, so that weights may be properly assigned to incoming signals. It is not necessary for this address to be global, it simply needs to be unique within a given PN.

For calculating the PN area, the memory required for connection tables and state preservation is used. All models have the same computation needs, so the area required for the processor arithmetic unit and state machine control is ignored; in any case, this area is much smaller than that required for connection memory and interconnect. There is no consideration of the amount of area required for line drivers.

Each PN is assumed to contain 16 CNs. Further research will be needed to establish the optimal value since it is dependent on the capabilities of the PN as well as the amount of contention in the network. Preliminary results indicate that varying the CN per PN value from 10 to 100 does not effect the relative rankings for this paper. One justification for choosing a number of this magnitude is that the network is resolving an input vector to the Hamming nearest learned vector. Since the two vectors are near, few of the nodes need to change; with a system that contains many learned vectors, the difference between any two is less than 20% [28]. In addition, the number of CNs per PN is limited on the low end by the increased interconnect required for more processors. The primary reason for decreasing this number is the desire for the maximum degree of parallelism in the network's execution, since a connectionist/neural network model is able to effectively use all available processors as noted by Fanty in his work on the Butterfly [11]. The absolute minimum of one CN per PN is infeasible as the direct implementation analysis shows.

The more PNs in the system, the more fault tolerant it is. We believe that VLSI connection networks will require no explicit fault correction capabilities. That is, the inherent fault tolerance of the connection network will be sufficient without the need to add extra fault correction hardware. This has yet to be proven and will be examined in later research.

In addition to saving silicon real-estate and increasing the degree of parallelism, the assignment of multiple CNs to each PN has a neurological basis. Many researchers have been struck by the organization of the brain into groups of tightly cooperating neurons. As expressed in Ballard's paper [3]: "To a coarse approximation, the cortex can be regarded as a two-dimensional sheet, a few millimeters in thickness. Within this sheet the anatomy can be usefully thought of as being organized into functional, overlapping columns of about 800 μm in diameter." Another researcher's statement of a similar observation is that "the columns represent a mode of intracortical fiber distribution that is encountered in all of the conventionally distinguished categories of cortex." [15]

The physical layout of the processors is assumed to be a tightly packed hexagonal array, unless stated otherwise. The hexagon grid is used because it allows the maximum number of direct connections between processors. Processors will be assumed to be adjacent unless the required interconnect consumes more space than the processors do. In that case, the distance between adjoining processors will be increased uniformly.

Accurate performance estimation must consider the effect of contention or arbitration of shared communication resources. To attempt to model the affect of contention on message delay, two additional factors are included in the calculations. The first is a *contention factor* and is the total number of messages in the system, or the number of CNs times the number of messages transmitted per CN times the percentage of CNs that are assumed to be firing at any given time (in the following calculations the number firing is set to 10% of the total number of CNs since our simulations show that 10% is a typical value for networks of this nature), divided by the number of potential paths, which is the number of PNs times the fanout degree of the interconnection graph divided by 2. The effect of the contention factor is to assume a random distribution of messages and determine the amount of delay encountered by each message on each physical link it has to travel. The second factor is a *serialization factor* and is the number of messages sent out of the PN divided by the fanout degree. This serialization factor presumes that a PN can simultaneously transmit on all outgoing wires and penalizes systems with fewer wires available or more messages to send.

3. Proposed Metrics

Speed and cost are the obvious choices for the quantities to be evaluated in comparing two different interconnection strategies. In any given design, there is a trade-off between the two. Designs with a greater degree of interconnect exhibit less communication latency, but at the price of increased area for the interconnect paths. The other factor used in these comparisons is the degree of fault tolerance. This set of metrics compares well with the characteristics chosen by Agrawal and his fellow researchers in defining a system for evaluating different multiprocessor configurations [1]; they chose to use communication latency, the required link count, the degree of fault tolerance and expandability.

There are other system constraints related to neural network emulation that are beyond the scope of this paper and are not considered here. An example of such a constraint is the required degree of temporal correlation of signals originating in divergent parts of the system.

3.1. Speed

Since the problem being addressed is how well a particular interconnect architecture allows the execution of a connectionist model, the measure to be used for emulation speed is the delay involved in making one update of the connection network. That is, the connection network requires T *cycles* to settle to a solution when given an input, where T is

proportional to the $distance_h$ between the input vector and the nearest stored vector. A *cycle* is defined as the length of time it takes from the moment a CN transmits a change of state until it receives an update from a destination CN. In other words, the value determined is twice the length of time it would take a message to travel from a given node to one of its destination nodes. The two destination nodes considered are the one furthest away and one that is placed at a $distance_g$ equal to the weighted average of all connection lengths.

The *communication latency* reported consists of two numbers - the average delay and the worst case delay. The worst case is not critical because of the asynchronous nature of the underlying model. In most of the architectures, there are two types of connections, those to CNs on the same PN and those to non-local CNs. The $distance_g$ between two CNs sharing the same PN is defined to be zero and there is assumed to be no delay for an update. The time required to compute the output function on the source and destination PNs is assumed to be constant across different architectures and is ignored in the analyses.

Communication delay has several components. First, and simplest to compute, is the time it takes the signal to travel the length of any physical links between two PNs. All messages are assumed to travel from center of PN to center of PN, and the time to transit a PN is taken to be equivalent to the time required to transit a wire equal to the PN diameter. The time required is due to the capacitance and resistance of the wire and is proportional to the distance between the nodes. The physical delay time for a 1μ wide metal wire of length 1μ is about 10^{-2} *nsec* [30]. This value was calculated by using current VLSI design rules and considering that resistance increases by the square of the decrease in width while capacitance remains constant.

The other factors that contribute to signal delay are the number of intermediate nodes that must process the signal, the degree of contention for the wires, the required message length and possible synchronization delays. The delay from buffering a message is assumed to be two clocks per bit, one to write to memory and one to read. With a system clock of 20Mhz, this translates into 10^2 *nsec*. Since pipelining is assumed, this is a one-time per PN delay.

3.2. Cost

The *cost* of implementing a particular architecture in silicon is represented by the required area. The area for a given architecture is the sum of the interconnect area, the processor memory area and the area of any nodes used only for communication. To simplify the calculations, there is no consideration of the cost of moving from one layer of metal to another or other routing problems.

A metal width of 1μ and a pitch of 1μ yields an area of $2\mu^2$ per micron of length. The area of a memory cell is taken to be 25 times the area of a minimal square of metal [14], or $50\mu^2$ per bit. To provide additional generality of the results, i.e. to provide easy scaling for new memory technologies, the variable M (for memory) will be used to represent this value in most calculations.

3.3. Fault Tolerance

VLSI is a faulty medium, so a design strategy that allows the proper functioning of a chip containing multiple defects is obviously better than one that is more vulnerable, therefore, the relative fault tolerance of each architecture will be described. It is difficult to determine precisely the degree of tolerance, since the connection model itself exhibits resistance to failure from isolated flaws. Connectionist/neural network models exhibit differing

degrees of fault tolerance. For wafer scale integration, a distributed data representation [24] will probably be required to provide the needed fault tolerance. For each design, the relative advantages and disadvantages are discussed. A more precise determination would include a probabilistic analysis, where the network is analyzed with varying fault parameters to determine the maximum allowable defect density for a given level of performance. Such analysis is expected to be part of further research in this area.

4. Analyses

The previously defined set of metrics is now applied to several candidate interconnect topologies.

4.1. Direct Implementation

The direct implementation strategy assigns one CN to each PN and creates physical links between the PNs to match the connection network's interconnections, i.e. the connection network and interconnection graphs are isomorphic. The physical model used here differs from the physical realizations of connection networks being researched by Hopfield *et al.* [26,27], who are also creating direct implementations, because their systems are analog and this one is digital and because they are assuming full interconnect.

As we show, a direct mapping of connection networks to silicon is infeasible. Also, since all connections are hard-wired, the system cannot be reconfigured for a new problem, making it impractical for most applications; not to mention the problems with CAD tools for developing such a system. Finally, with a direct connect system, the preloading of static weights into the system is difficult, leaving *in situ* learning as the only alternative. The analysis of such a scheme is thus provided as a standard to compare the multiplexed systems against, and because of the number of researchers who still feel that it is the best way to build connection networks.

The communication delays in a direct connect system are due only to the length of the interconnect lines, since there is no communication channel multiplexing or message routing through intermediate nodes. Message length is minimal, since each node can readily identify the source of a given message for weight assignment and no destination addresses are needed for routing purposes. The delay in transmitting a one bit message is therefore equal to the time it takes for the signal to transit the wire.

Fault tolerance is equivalent to that of the connection model, the only possible points of failure are the nodes themselves and the connections between them. With the number of nodes in the system, and the ability of the connectionist model to change the weights of other connections to adjust for invalid inputs, such a system should be quite tolerant of defects.

Since it is impossible to build a system with an α_g=1.25, because $L_{direct}(n)$, $\sum_{i=1}^{n} 6i\alpha_g^{-i} \geq 999$, has no solutions for $\alpha > 1.1$, and the rest of the comparisons use 1.25 for α_g; consider the best case from an implementation viewpoint, a system with every node connected to the $nearest_g$ 10^3 nodes, or $\alpha_g = 1.0$. With α_g set equal to 1.0, $L_{direct}(999)$=18.

The total processor area is $\pi r^2 \times 10^6$, 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 $l = \sqrt{2}/2 \times 18 \times 2r$, since l is the length of an average connection spanning a circular region of radius 18 PN diameters ($2r$). Solving this set of equations yields $r = 5.4 \times 10^3\mu$, PN area of $9.2 \times 10^7\mu^2$ and a total chip area of $9.2 \times 10^{13}\mu^2$ or 92 square meters. The area required for total PN memory is only 4 *bits per weight* $\times$ M μ^2 *per bit* $\times$ 10^3 *weights* $\times$ 10^6

PNs or $2 \times 10^{11}\mu^2$, showing how the interconnect requirements dwarf the processor requirements.

The worst case delay is over a wire $18 \times 2r\ \mu$ long with a delay of 10^{-2} *nsec per* μ, or 1.9×10^3 *nsec*, and the average delay is 1.4×10^3 *nsec*.

This analysis is limited in accuracy because the node of origin is assumed to be in the center of the system and there will be some nodes on the periphery. Also, it is impossible that three metal layers can be routed as closely as the calculations assume or to have this optimal of a mapping of CNs to PNs. For these reasons, and because of the use of approximate values in all computations, the results are only lower bounds of roughly the correct magnitude.

4.2. Grid or Nearest Neighbor

The consideration of this interconnection topology is similar to the direct connect, because the main purpose of the analysis is to provide a basis for comparison. Because of the lack of non-local connections, the grid should exhibit minimum interconnect cost and maximum communication delays.

One potential problem with having multiple CNs per PN is the required interprocessor communication bandwidth. When the definition of locality was first stated, there was some concern that the required bandwidth between two nodes was not bounded. The following theorem shows that the bandwidth is bounded and gives an upper bound.

Theorem: The bandwidth, B_{grid}, between any two adjacent nodes of an *infinite* two-dimensional grid, given that all messages travel the shortest path between any two points and are equally likely to take either of two paths of the same length, is bounded by

$$\beta \left(\frac{\alpha^3}{(\alpha-1)^4} + \frac{\alpha^2}{(\alpha-1)^2} + \frac{\alpha}{(\alpha-1)^2} + \frac{\alpha}{3(\alpha-1)^3} \right)$$

where β is the length of a message and α is the previously defined measure of connection locality.

Proof: First consider how many paths a message might travel. The number of possible equal length paths from node (x_1,y_1) to node (x_2,y_2) is $\frac{(x+y)!}{x!y!}$ where $x=|x_1-x_2|$ and $y=|y_1-y_2|$ A combinatorial argument for this result is, if the length of a path is $x+y$ and a path is exactly specified by the choice of which edges are vertical (alternatively, which are horizontal), then the number of paths would be $\binom{x+y}{x}$.

To determine the required bandwidth between any two nodes, pick an arbitrary node and label it (0,0) and assign the standard Cartesian coordinate system to the grid centered on that point. Then calculate the bandwidth over the interval [(0,0),(0,1)]. This bandwidth is equal to the sum of several component parts. There is the contribution from connections between nodes in the quarter plane $x,y>0$ and nodes in the quarter plane $x,y\leq 0$; plus connections between nodes in the quarter plane $x>0,y<0$ and nodes in the quarter plane $x\leq 0,y\geq 0$; plus connections between nodes on the line $x>0,y=0$ and nodes in the half-plane $x\leq 0$.

Formalizing the above statements about the number of paths and the sources of the messages (combining symmetrical terms) produces the following equation for the bandwidth B_{grid}.

$$B_{grid} = \beta \left(2 \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} \alpha^{-(i+j+k+l-1)} \frac{\frac{(k+l)!}{k!l!}}{\frac{(i+j+k+l)!}{(i+k)!(j+l)!}} + \sum_{i=1}^{\infty}\sum_{j=0}^{\infty} \alpha^{-(i+j-1)} \right.$$

$$\left. + 2 \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha^{-(i+j-1)} \frac{1}{\frac{(i+j)!}{i!j!}} + 2 \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sum_{k=1}^{\infty} \alpha^{-(i+j+k-1)} \frac{\frac{(j+k)!}{j!k!}}{\frac{(j+k+l)!}{(i+j)!k!}} \right)$$

Simplifying terms and realizing that the later factorial values are smaller than the first yields the following absolute limit.

$$B_{grid} < \beta \left(\sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} \alpha^{-(i+j+k+l-1)} + \sum_{i=1}^{\infty}\sum_{j=0}^{\infty} \alpha^{-(i+j-1)} + \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha^{-(i+j-1)} + \frac{1}{3} \sum_{i=1}^{\infty}\sum_{j=1}^{\infty}\sum_{k=1}^{\infty} \alpha^{(i+j+k-1)} \right)$$

This summation reduces to

$$B_{grid} < \beta \left(\frac{\alpha^3}{(\alpha-1)^4} + \frac{\alpha^2}{(\alpha-1)^2} + \frac{\alpha}{(\alpha-1)^2} + \frac{\alpha}{3(\alpha-1)^3} \right)$$

□

Corallary: B_{hex}, the bandwidth in an infinite hexagonal layout, is finite.

All PNs are placed on the silicon surface in a hexagonal pattern. Each PN both transmits to and receives from its six neighbors. Any messages that need to be routed to a non-local PN are relayed by the intermediate nodes using a routing algorithm based on the destination address of the message. Hillis, in his paper on the Connection Machine [22], has described such a routing algorithm. The probability of a connection between two CNs is α_g^{-d} where d is the number of links between the PNs on which they reside. For example, all CNs on the same PN are at a distance$_g$ of zero and thus connected.

There are sixteen CNs on a PN. Each CN requires 10^3 weights, 16 local addresses and 984 global addresses. A global address is 28 bits, eight hops with 3 bits per hop plus four for the destination PN, and a local address is 4 bits. A connection is addressed by the source address contained in the incoming update message so there are two sets of addresses required, one for the destinations and one to indicate which weight is being updated by an incoming message. The total memory required is thus $4 \times 10^3 + 2 \times (4 \times 16 + 984 \times 28)$ or 59232 *bits per CN*. This predicts a PN area of $9.5 \times 10^5 \times M$ or $4.7 \times 10^7 \mu^2$. The complete system would require $2.9 \times 10^{12} \mu^2$, or one thirtieth the area of the direct implementation.

A message sent between two PNs consists of the routing information, which functions as both source and destination address, and one bit of state, for a total of 29 bits. The expected path length can be calculated by finding $L_{grid}(984)$ or by solving the equation $\sum_{i=1}^{n} 6i\alpha^{-i} \geq (1000 - 16) / 16$ for n. The smallest integer n that is a solution is 8, so the maximum number of intervening links expected is 8 and the average is $\sqrt{2}$ / 2 times 8 or 5.6, assuming the PNs are randomly distributed within the area of the circle with radius 8. All connections are to nearest neighbors, so the communication latency is the time required to relay the message through intermediate nodes. With a delay per node of 10^2*nsec* plus a message length of 29 *bits* times the distance travel time of 69 *nsec per node per bit*, $\sqrt{4.7 \times 10^7}\ \mu \times 10^{-2}$ *nsec per* μ *per bit*, times the contention factor of 525 (984 *messages per CN* $\times$ 16 *CNs per PN* $\times$ 10% *firing rate* $\times$ 62500 *PNs* /(6 *wires per PN*

$\times$ 62500 *PNs* / 2)) and the serialization factor of 164 (984 *wires* / *fanout of* 6), there is an average cycle time of 9.6×10^8 *nsec*, $2 \times 29 \times 69 \times 5.6 \times 525 \times 164$ / 2, and a worst case time of 2.7×10^9 *nsec* ($2 \times 29 \times 69 \times 8 \times 525 \times 164$). This delay is five to six orders of magnitude greater than in the direct implementation and two to three orders of magnitude greater than neural systems.

4.3. Hypercube

The hypercube is an interconnection network in which, given N nodes in the system, a node can reach any other node in at most $O(lnN)$ steps. It has been shown to be optimal for a number of physical models [36,37] and is the basis of the Caltech Cosmic Cube, Intel iPSC, N-cube Hypercube and FPS T-series computers, since it provides an excellent trade-off between interprocessor communication and implementation cost. The standard implementation is a store and forward network with all the delays inherent in such a system. Charles Seitz and Bill Daly of Caltech have created a new model that uses virtual circuits; the details are not yet available.

The routing algorithm is best shown by an example. Consider a 3D hypercube with 8 nodes. The nodes can be numbered (in binary) as 000 001 010 011 100 101 110 111. Each node is connected to all nodes that have an address that differs in one bit position. Figure 2 graphically shows the connections of a 3D hypercube. A simple inductive argument shows that the number of nodes that can be reached in n steps in a cube of dimension m is $\binom{m}{n}$.

The physical implementation of a hypercube-based architecture is similar to the grid structure. There are sixteen CNs per PN and the probability of a connection is calculated in the same manner. The area of each PN is also determined in a similar way. To simplify the analysis, the PN layout is assumed to be a rectangular grid. This configuration is used so that the physical placement directly maps to the interconnect topology.

To provide full interconnect in a system of 6.25×10^4 PNs, a cube of order 16 is required. The interconnection graph has a branching factor of 16, therefore each node requires 16 output paths. $L_{cube}(984)$, the expected number of hops needed to reach at least 10^3 CNs is derived via the formula $\sum_{i=1}^{n} \binom{16}{i} \alpha_g^{-i} \geq 984 / 16$. The minimum integral value for n is 2.

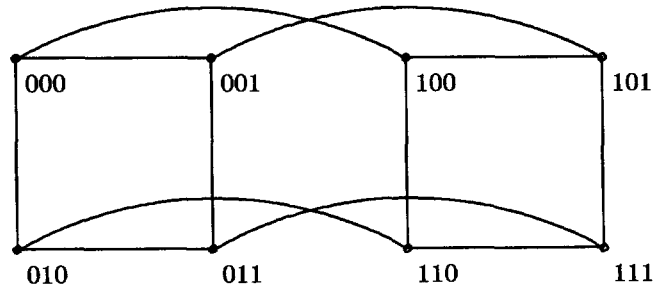

Figure 2. 3D Hypercube

A simple construction argument shows that each PN has the following set of 16 connections: 1,1,2,2,4,4,8,8,16,16,32,32,64,64,128,128 where each number is the length$_p$ in processor widths. In other words, each PN is connected to two immediate neighbors, two PNs that are two diameters away, etc. Because of the definition of locality$_g$, and the graph mapping assumptions, a CN on the PN at the end of each of these lines is equally likely to be connected to the given CN.

However, it is not necessary to use the entire 16-D hypercube to provide the needed reachability$_g$ for any given CN. Using a 14-D subcube eliminates some of the extremely long wires that skew the results. A 14-D cube has a longest physical connection of 64 PN diameters and an average length connection of 18, using the list of physical connection lengths in the preceding paragraph. Non-local addresses need to be 12 bits long, two hops at 4 bits each plus 4 bits for specifying the correct CN on the destination PN, and local addresses are 4 bits. The total CN area is memory plus two times the address space, one for incoming and one for outgoing messages, or $4 \times 10^3 + 2 \times (16 \times 4 + 984 \times 12) = 27744$ *bits*, multiplying by M yields $1.4 \times 10^6 \mu^2$. The per PN area is 16 times this, or $2.2 \times 10^7 \mu^2$, and total chip area is $1.4 \times 10^{12} \mu^2$.

The worst case delay is on a connection two hops long where each hop is 64 PN diameters in length with a contention factor of 225 (984 *messages* $\times$ 16 *CNs/PN* $\times$ 10% *firing* / (14/2)) and a serial factor of 70 (984 *messages* / 14 *wires*), or $2 \times 64 \times \sqrt{2.2 \times 10^7 \times 10^{-2}}$ *nsec per bit* $\times$ 13 *bits per message* $\times$ 225 $\times$ 70 $= 1.2 \times 10^9 nsec$. This gives a worst case *cycle* time of $2.5 \times 10^9 nsec$. Similarly, the average cycle time is $3.5 \times 10^8 nsec$.

The hypercube exhibits results a factor of two better, in both cost and speed, than the grid, since with its greater interconnect it is possible to reach more nodes directly instead of needing intermediate relays, requiring shorter addresses and thus less area and time. There is also less congestion with the hypercube since there are more potential paths. These benefits are offset to some degree by the cost of the longer paths required when mapping the interconnection graph to a planar surface.

4.4. Switching Networks

Several switching networks have been proposed for linking large numbers of processors together. They all provide global interconnect and have more capability than required for implementing our connectionist networks. The main drawbacks to the use of switching networks are the delay in transitioning the switches and the area required for them. The next three sections discuss how architectures based on the models of Fat Trees and Hashnet, two example switching networks, would perform as connectionist emulators.

4.4.1. Fat Trees

Fat Trees are an interconnection topology that exhibits good theoretical behavior [29]. The original work was done as part of the effort in designing a communication network for the Connection Machine at MIT. The essential idea is that the PNs are the leaves of a complete binary tree and that the internal nodes of the tree are routing processors. Available bandwidth between routing nodes increases as you move up the tree from leaves to root. In the optimal system, the bandwidth increases appropriately as one ascends the tree, so that no messages are ever lost and there is no resulting bottleneck at the root.

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 PNs. The time it takes for a message transmission is equal to the number of inner nodes

encountered plus the length of the message. All messages must be sent during the same period of time, so the best case delay is equal to the worst case delay.

The formula for calculating the expected number of links required, $L_{fat\ tree}(984)$, is $\sum_{i=1}^{n} 2^{i-1}\ \alpha_g^{-n} \geq 984/16$. This yields a value for n of 9. A link count of n implies 2^{n-1} internal nodes, so the time required to transit the switching network, ignoring all wire transit times, is $2^8\ nodes \times 14 bits \times 50 nsec\ per\ node$ or $1.8 \times 10^5 nsec$, for a cycle time of $3.6 \times 10^5 nsec$. To make this result comparable to the other results, the cycle time should be multiplied by a serialization factor of 984, the number of non-local messages, and a contention factor of 1. With this adjustment, the time for a complete cycle is $3.6 \times 10^8 nsec$. Although this delay appears to be an order of magnitude better than the other models, it should be remembered that it does not involve wire transit times. The time per node was reduced to one clock owing to the pipelining specified in the Fat Tree design.

The area required for a system based on a Fat Tree is $4 \times 10^3 + 2 \times (16 \times 4 + 984 \times 13)$ *bits per CN* or $1.5 \times 10^{12}\mu^2$ total PN area together with the area of the switching network. For a system with 6.25×10^4 PNs, $2^{16}-1$ switching nodes are required together with an equivalent number of wires. Without designing a switching node it is not possible to estimate the area in square microns.

In conclusion, using an implementation that closely follows the one proposed by Lieserson, Fat Trees are not good candidates for emulating connectionist/neural networks, because there are as many routing nodes as processing nodes, much interconnect is required, the synchronous operation requires that clocks be transmitted across the entire system, and each message is delayed as long as the worst case. An asynchronous Fat Tree is possible, but it would require increased bandwidth or conflict resolution techniques. It may be possible to generalize the Fat Tree model to better support connectionist/neural network emulations, but that would be a digression from the topic of this paper and has not been done.

4.4.2. Hashnet

Fahlman of CMU developed what he called the Hashnet as part of the design of the NETL system [8,10]. The principle behind the Hashnet as a connection topology is that, if you have a interconnection network with a sufficient number of layers, it is not necessary for all possible paths to be physically present. He states the requirements for a million node system communicating via a hashnet[9]. It would need a 960 x 960 switching network that was time-shared 1024 ways. The time sharing of the switching network was proposed as an alternative to increased area in interconnect and switching nodes. Unfortunately, communication intensive connectionist models do not map well to this alternative.

The time required for a cycle is at least $2 \times 41\ bits \times 50\ nsec\ per\ node \times 2^6 nodes$ or $2.6 \times 10^5 nsec$. Only 2^6 layers need be traversed since each PN contains 2^4 CNs. This value for the length of a cycle ignores all wire delay times getting from each node to the switch, going through the switch and going on to the destination node. Thus, the drawbacks to the use of hashnet are the time required to pass messages through 10^3 node transitions and the space required for the switching circuit.

4.5. Shared Memory

There are three approaches to designing a system that communicates via shared memory. The first uses a bus between processors and memory, the second uses a switching network as described above and the third is a compromise such as used in the NYU Ultra computer. The limitations on bus bandwidth rule out the first option and the second has been covered in section 4.4.

The NYU Ultra is designed as a system that provides each processing element with a virtual circuit to memory [16]. It has an *add_and_store* primitive that provides an efficient mechanism for implementing shared data structures, but is unfortunately of no help in emulating a connection network. For a system with N processors, the switching network has a bandwidth that is $O(N)$ and a latency that is $O(ln(N))$. Any conflicts within the switching network cause requests to be queued at the intermediate nodes and a serious degradation of the response time.

In emulating connectionist networks, a PN must check memory regularly to make sure that there have not been any changes that require the recalculation of the output function of one of its CNs. A system with 10^6 CNs, each one requiring the checking of 10^3 locations in memory, over a network with a limited bandwidth, will not provide acceptable response times.

4.6. Connection Machine

With the recent announcement of the Connection Machine (CM) as a commercial product, together with its name, the obvious question is why not use it to emulate connection networks. The answer is that it is a SIMD machine and does not map well to the data dependent routing required in connectionist systems. While it is trivial to imagine an emulation of a neural network on the CM, with each CN's output being recalculated at the same time using a common formula, which, incidently, eliminates any connection models that require different functions on different nodes, the transmission of updated outputs from each CN to many locations simultaneously is difficult. If the execution model where each CN with new output sequentially updates its destinations is used, the emulation becomes serial, not parallel, and much slower. Connection networks spend most of their time communicating, not computing.

The CM only provides for 4k bits of storage per PN. This is insufficient to support a network with a large number of connections. It is possible to imagine a CM that would have more local memory, but that does not solve the connection problem.

The final problem with the CM is that the communication strategy is for each chip containing 16 nodes to be fully connected, while all non-local messages transit a hypercube. In the above analysis of the hypercube, some problems with its use for long distance updates were presented. For these reasons, we believe that the current CM architecture will not scale well to emulate connection/neural networks with 10^9 connections.

4.7. The Broadcast Hierarchy[3]

The Broadcast Hierarchy (TBH) is an interconnection topology we have developed specifically for the communication requirements of connectionist networks. In its simplest configuration, it is a hierarchy of broadcast regions with each region at level $n+1$ including at least two level n regions. Two nodes communicate via the lowest common level. A one dimensional example is shown in Figure 3. The following are some characteristics of TBH that make it attractive for VLSI implementation:

(1) The interconnect architecture allows the use of a variable length encoding of addresses with a resultant decrease in memory area requirements.

(2) TBH efficiently emulates connection networks which exhibit locality of communication. Furthermore, the hierarchical structure can be adjusted to match locality_c and locality_g.

[3]Patent pending.

(3) There is no routing, this simplifies the design and fit of the layout and increases fault tolerance.

(4) No destination address is needed in messages. Each message consists of the source CN and the update value, all listening PNs accept a message if they have a connection to the source.

(5) The upper communication levels can function asynchronously. This allows the use of multi-access communication protocols.

(6) The use of overlapping regions provides a neural-like interconnect. For example, the broadcast of non-specific excitatory or inhibitory signals is common in neural networks.

(7) All nodes within a region are updated in parallel, leading to significant communication concurrency.

(8) Nodes may be loaded or examined by an external system via the top broadcast level.

(9) The use of wide planes of metal for buses allows for a simple layout, yet is within current technology. It also allows more levels of metal than the use of narrow wire runs and is less susceptible to defects and manufacturing faults.

A TBH system can be visualize as follows. Imagine a group of four PNs each of which is emulating 16 CNs. These four PNs are tied to a single bus; when a CN changes state, the new state is broadcast on the bus to all listening PNs. Now, assume that four groups of four PNs (16 PNs and 256 CNs) are connected to another bus. This interconnection structure can be repeated recursively for as many levels as desired. A processor broadcasts a particular CN state change on the bus that reaches all PNs containing the CNs that are connected to the CN whose state changed; no state change need be broadcast more than once. Locality guarantees that most CNs require only local broadcast, so consequently, though more PNs share the higher level buses, each PN requires relatively less global bandwidth on those buses, with most broadcasts being at the lower levels.

The performance and area analyses are similar to those of other architectures because they assume that questions such as communication protocols and physical layout are solved. Though a regular topology is not required, one is assumed to simplify the analysis. In calculating the performance of TBH, there are several parameters that need to be specified. These include the number of nodes that can communicate via the lowest level and the multiplier factor for each higher level. Table 1 shows some results for different values of these two parameters. They are held equal to each other in these calculations, but that is not necessary. The *expected* values refer to how many of the 10^3 target CNs would

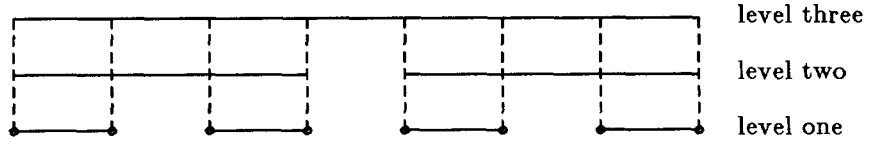

Figure 3. One-Dimensional Broadcast Hierarchy

be connected for the first time, at that level, using the same α_g approach as with the other topologies. The bottom line in the table, *max levels*, is the number of broadcast levels required to provide 100% reachability.

For the following calculations, a multiplier of four is used. Initial calculations indicate that four is a reasonable value for the multiplier, at least as far as these preliminary analyses are concerned, but further research is needed here also. The length of the address required is a function of the level on which the message is transmitted - a very important characteristic of TBH, since it reduces PN memory needs and leads to a variable length address encoding. The maximum length required is 8 bits for PN designation and 4 bits for naming the CN within the PN. Connections on level 0 do not require any interconnect. Levels 1, 2, 3 and 4 can be spanned with runs 2, 6, 14 and 30 times the width of an average PN. For a multiplier of 4, worst case communication takes place on level 4 and the time for a circuit is $2 \times 13\ bits \times 30 \times L\ \mu \times 10^{-2} nsec\ per\ \mu$ or $7.8 \times L\ nsec$, where L represents the width of a PN. The average delay takes place on level three and the time is $1.6 \times L\ nsec$. Only one address is needed since there is no routing and its length is related to the number of CNs reachable on the level being used.

Even though there is no need to keep the destination addresses, it is necessary to keep the addresses of the incoming data for connection weight assignment. Even so, the reduction in the area required for address bits significantly shrinks PN size. The area required for each CN is $4 \times 10^3 + 16 \times 4 + 32 \times 6 + 112 \times 8 + 384 \times 10 + 456 \times 12$ or $14464\ M$. With 16 CNs per PN, this is equivalent to $1.2 \times 10^7 \mu^2$ for a PN or $7.2 \times 10^{11} \mu^2$ for the entire system.

This value for PN size yields a worst case circuit time of $1.1 \times 10^7\ nsec$ and an average circuit time of $5.5 \times 10^5\ nsec$ after contention has been taken into account. The serialization factor is 1 since each message is only broadcast once on a level and the contention factor is the number of CNs on the level times the percent firing or worst case of 409.6 and average case of 102.4. Using multiple buses on each level would decrease the contention but increase the serialization factor, since contention would drop faster it would be worth doing.

multiplier	4	9	16
level		expected connections	
0	16	16	16
1	32	96	192
2	112	736	792
3	384	152	-
4	456	-	-
max levels	8	6	4

Table 1. TBH branching behavior.

Although loss of bandwidth due to arbitration is not considered here, it will be a concern in the design of a real system. Our preliminary plan is that arbitration will be synchronous at the lowest levels and will be asynchronous, requiring a multi-access collision detect mechanism, as in Ethernet, at the upper levels. Theoretical results have shown that the effective bandwidth using a multi-access collision detect protocol with an infinite number of processors is better than $1/e$ times the available bandwidth [40]. If the network exhibits enough locality, this should be sufficient for emulating connection/neural networks. A secondary use of the highest levels is to provide a means of supplying global inputs to the PNs, TBH is significantly better in this regard than all the other architectures considered.

There are several possible methods of implementing the interconnection buses. The first is to use a grid of horizontal and vertical buses for each region with forwarding drivers at each intersection. Different communication levels would use parallel buses, since contention would drop, there would be a net gain. This strategy is represented in the level spanning distances used above, since a message might need to travel the entire width and height of a region. With a PN size of $1.2 \times 10^7 \mu^2$, there is room for more than the 8 buses in each direction required for a multiplier of four, even if they are many times the minimal width.

A second alternative for implementing the interconnection is to extend the bus width until they are planes rather than metal strips. Doing so would increase the yield of the wafer and, since a plane is a regular surface, it would be possible to use more levels of planes than the three levels of metal proposed in this paper, even with current technology. The use of planes of metal is a new concept and will require more research to determine the appropriate techniques for accessing and driving them, as well as VLSI fabrication. Possible benefits of using planes include increased fault tolerance, no need for repeaters and faster response times. One drawback is that a ground plane would be required between each pair of signal planes.

The third implementation that is being considered is using light as the carrier medium at the top level. There are some interesting possibilities this raises, such as the use of multiple frequencies to increase the bandwidth, but much more research is needed to determine the feasibility. The range of possibilities demonstrate that the underlying architecture is implementation independent and will work with any broadcast medium.

The reliability of TBH is greater than most of the other topologies, since there is no dependence on intermediate nodes for message passing and, if one bus exhibits problems, the next higher one can be used with a slight cost in performance.

5. Conclusions

A summary of the results of the preceding analyses is presented in Table 2. Consider the areas in square meters: direct - 92, grid - 2.9, hypercube - 1.4 and TBH 0.7. A VAX 11/780 is said to have 2 m^2 of silicon. With current VLSI technology, a billion connection TBH system could be built on twenty-four 8-inch wafers.

This paper gives some absolute sizes of systems built with the different interconnect topologies. It should be realized that each approach differs enough in its implicit assumptions that it is the relative order of magnitude that is significant, not the precise values. Future developments in VLSI technology will reduce the area required for all of these systems. Current researchers are working on chips with submicron feature size and increasing the effective surface area with layers of devices. These developments, along with architectural improvements we envision, should allow the shrinking of a billion connection TBH such that it can fit on a single wafer. We believe that this technology is possible within the next 5 to 10 years.

Of all the topologies considered, TBH is the preferred one since the area required is half that of the next best alternative and the time delay is several orders of magnitude better. In fact, the results of the TBH analysis provides an affirmative answer to the question posed earlier: "Can a connection network with 10^6 nodes and 10^9 connections can be built within reasonable cost/performance restraints?"

One major conclusion of this paper is: the degree of locality of both the connection and physical networks is critical to any implementation. Architectures with a model of reachability different from those required by connectionist models were several orders of magnitude worse than an architecture designed with the connectionist model in mind. Further research is needed to establish a more comprehensive definition of locality and reachability so the preliminary results of this paper can be verified. Initial neural studies show a multi-modal distribution is appropriate; but what are the relative quantities of long and short connections and what are their range of lengths? Answering these questions will entail research into neural networks to determine precisely how they are interconnected and network simulation to better understand the trade-off between connectivity and function.

Another topic for future research is to determine how increasing the numbers of CNs per PN impacts execution speed. There should be some loss of speed due to decreased parallelism, but this may be offset by a reduction in area and a decrease in the average time required to update destination CNs. Other issues that need to be addressed include: how fault tolerant are the computational models, what are appropriate values of α_g to reflect the mapping problems, what are the weaknesses of TBH architectures, what are the correct values of TBH implementation parameters, what is the optimal bus structure for TBH, how to provide a good user interface and debugging aids, and how is a network emulator to be loaded and unloaded?

In conclusion, this paper has eliminated all connection topologies requiring routing functions from consideration as candidates in building a connection network emulator and has suggested The Broadcast Hierarchy as the only reasonable alternative. Our analyses have pointed out several research topics, such as a better characterization of locality, that must be studied before such an emulator can be built.

style	area	average message	worst case message
direct	$9.2 \times 10^{13}\mu^2$	1.4×10^3 *nsec*	1.9×10^3 *nsec*
grid	$2.9 \times 10^{12}\mu^2$	9.6×10^8 *nsec*	2.7×10^9 *nsec*
h-cube	$1.4 \times 10^{12}\mu^2$	3.5×10^8 *nsec*	2.5×10^9 *nsec*
TBH	$7.2 \times 10^{11}\mu^2$	5.5×10^5 *nsec*	1.1×10^7 *nsec*

Table 2. Summary of Results

6. References

[1] Agrawal, D. P., Janakiram, V. K. and Pathak, G. C., "Evaluating the Performance of Multicomputer Configurations," *IEEE Transactions on Computers*, May 1986, pp. 23-37.

[2] Anderson, J. A. and Hinton, G. E., "Models of Information Processing in the Brain," in *Parallel Models of Associative Memory*, J. A. Anderson and G. E. Hinton (ed.), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1981, pp. 9-48.

[3] Ballard, D. H., "Cortical Connections and Parallel Processing: Structure and Function," Tech. Rep. 133, Computer Science Department, Rochester, NY, January 1985.

[4] Braitenberg, V., "Cortical Architectonics: General and Areal," in *Architectonics of the Cerebral Cortex*, M. A. B. Brazier and H. Petsche (ed.), Raven Press, New York, 1978, pp. 443-465.

[5] Cottrell, G. W. and Small, S. L., "Viewing Parsing as Word Sense Discrimination: A Connectionist Approach," in *Computational Models of Natural Language Processing*, B. G. Bara and G. Guida (ed.), Elsevier Science Publishers (North-Holland), 1984, pp. 91-119.

[6] Creutzfeldt, O. D., "The Neocortical Link: Thoughts on the Generality of Structure and Function of the Neocortex," in *Architectonics of the Cerebral Cortex*, M. A. B. Brazier and H. Petsche (ed.), Raven Press, New York, 1978, pp. 357-383.

[7] Derthick, M., "Learning in Boltzmann Machines and Why it's Slow," CMU-CS-84-120, Computer Science Department, Carnegie-Mellon University, 1982.

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

[9] Fahlman, S. E., *Design Sketch for a Million-Element NETL Machine*, Carnegie-Mellon University Department of Computer Science, Pittsburgh, PA, 1980.

[10] Fahlman, S. E., "The Hashnet Interconnection Scheme," CMU-CS-80-125, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, June 2, 1980.

[11] Fanty, M., "A Connectionist Simulator for the BBN Butterfly Multiprocessor," Tech. Rep. 164, Dept. of Computer Science, University of Rochester, Rochester NY, January, 1986.

[12] Feldman, J. A. and Ballard, D. H., "Connectionist Models and Their Properties," *Cognitive Science*, vol. 6(1982), pp. 205-254.

[13] Feldman, J. A., "Energy and the Behavior of Connectionist Models," Tech. Rep. 155, Dept. of Computer Science, Univ. of Rochester, Rochester, NY, November 1985.

[14] Glasser, L. A. and Dabberpuhl, D. W., *The Design and Analysis of VLSI Circuits*, Addison-Wesley Publishing Co., Reading, MA, 1986.

[15] Goldman, P. S. and Nauta, W. J. H., "Columnar Distribution of Cortico-Cortical Fibers in the Frontal Association, Limbic, and Motor Cortex of the Developing Rhesus Monkey," *Brain Research*, vol. 122(1977), pp. 393-413, Elsevier/North-Holland Biomedical Press.

[16] 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.

[17] Grossberg, S., "How Does a Brain Build a Cognitive Code?," *Psychological Review*, vol. 87, 1 (January 1980), pp. 1-51.

[18] Hammerstrom, D., "A Connectivity Analysis of Recursive, Auto-Associative Connection Networks," Tech. Report CS/E-06-009, Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon, August 1986.

[19] Hebb, D. O., *The Organization of Behavior*, John Wiley, New York, 1949.

[20] Hecht-Nielsen, R., *Artificial Neural Systems Technology*, TRW Ranco Carmel AI Center, San Diego, CA, March 10, 1986.

[21] Hestenes, D., "How the Brain Works: the next great scientific revolution," *Proc. 3rd Workshop on Max. Entropy and Bayesian Methods in Applied Statistics*, Aug. 1-4, 1983.

[22] Hillis, W. D., "The Connection Machine (Computer Architecture for the New Wave)," A.I. Memo No. 646, MIT Artificial Intelligence Laboratory, Sept 1981.

[23] Hinton, G. E., Sejnowski, T. J. and Ackley, D. H., "Boltzmann Machines: Constraint Satisfaction Networks that Learn," Technical Report CMU-CS-84-119, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA 15213, May 1984.

[24] Hinton, G. E., "Distributed Representations," Tech. Rep. CMU-CS-84-157, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA 15213, 1984.

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

[26] Hopfield, J. J. and Tank, D. W., "Neural Computation of Decisions in Optimizing Problems," *Bio. Cybernetics*, vol. 52(July 1985), pp. 141-152.

[27] Hopfield, J. J. and Tank, D. W., "Computing with Neural Circuits: A Model," *Science*, vol. 233(8 August 1986), pp. 625-633.

[28] Kanerva, P., "Self-Propagating Search," CSLI-84-7, Cntr for the Study of Language and Information, Stanford Univ., Palo Alto, CA, March 1984.

[29] Leiserson, C. E., "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Trans. on Computers*, vol. C-34, 10 (October 1985), pp. 892-901.

[30] Mead, C. and Conway, L., *Introduction to VLSI Systems*, Addison-Wesley Publishing Co., Reading, MA, 1980.

[31] Minsky, M. and Papert, S., *Perceptrons*, The MIT Press, Cambridge, MA, 1969.

[32] Pollack, J. B. and Waltz, D. L., "Parallel Interpretation of Natural Language," *Proc. Intl. Conf. on Fifth Generation Computer Systems*, Tokyo, 1984, pp. 686-691.

[33] Rosenblatt, F., *Principles of Perceptrons*, Spartan, Washington, D.C., 1962.

[34] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., "Learning Internal Representations by Error Propagation," ICS Report 8506, Institute for Cognitive Science, La Jolla, CA, September 1985.

[35] Rumelhart, D. E. and McClelland, J. L., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Bradford Books/MIT Press, Cambridge, MA, 1985.

[36] Seitz, C. L., "Concurrent VLSI Architectures," *IEEE Transactions on Computers*, vol. C33(December 1984), pp. 1247-1265.

[37] Seitz, C. L., "The Cosmic Cube," *Communications of the ACM*, vol. 28(January 1985), pp. 23-33.

[38] Sejnowski, T. J. and Rosenberg, C. R., "NETtalk: A Parallel Network that Learns to Read Aloud," JHU/EECS-86/01, The Johns Hopkins Univ. Elec. Eng. and Comp. Sci. Tech. Rpt, 1986.

[39] Selman, B., "Rule-Based Processing in a Connectionist System for Natural Language Understanding," CSRI Technical Report No. 168, Computer Systems Research Institute, University of Toronto, Toronto, Ont., Canada, 1985.

[40] Tanenbaum, A. S., *Computer Networks*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[41] Wickelgren, W. A., "Chunking and Consolidation: A Theoretical Synthesis of Semantic Networks, Configuring in Conditioning, S-R Versus Cognitive Learning, Normal Forgetting, the Amnesic Syndrome, and the Hippocampal Arousal System," *Psychological Review*, vol. 86, 1 (1979), pp. 44-60.