A VLSI Interconnect Structure for Neural Networks

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

Technical Report No. CS/E-88027
August 1988

Abstract

The Connectionist or Neural Network (CNN) computation model has constraints that limit the suitability of VLSI interconnect architectures for supporting efficient emulation. This paper presents a set of metrics that can be used for analysis of both CNN and physical system interconnect. It also provides an introduction to a range of CNN models and looks at what each requires. Finally, different possible implementation strategies are considered and a solution proposed that offers good performance for many CNN models.

Table of Contents

CHAPTER 1

Introduction

The connectionist/neural network (CNN) computational model is one with many simple processors calculating a state function based on their inputs and transmitting the new state to other processors. The communication requirements are for many short messages transmitted with minimal, predictable delays. If systems to emulate these networks are to be effectively built, it will require the design of communication architectures that provide the needed capabilities for minimal cost and complexity, but with maximal speed and fault tolerance. This paper proposes a solution architecture, the Augmented Broadcast Hierarchy (ABH).

ABH is capable of supporting the requirements of CNN computations, yet can be implemented with current VLSI silicon design capabilities. In this paper, ABH is contrasted with other alternative architectures that have been proposed. The areas of comparison include speed, cost, reliability, and resource allocation/contention.

1. Introduction

The use of large numbers of communicating processors requires the development of appropriate interconnection architectures. This paper addresses one aspect of this problem, how best to meet the communication needs of a system consisting of many processing nodes together with a *connectionist* or *neural network* computational paradigm, when restricted to single wafers of silicon[1] and near term future VLSI capabilities.

The computational model used in this paper is that of a set of simple processors, highly interconnected both for input and output, each of which combines its inputs, according to a node update function, to generate a new output. Each node then transmits its new output state to those other processors that require it as an input. It is necessary to have an interconnection topology with sufficient bandwidth for the large number of potential messages, as well as capable of providing the required degree of fan-in and fan-out for each processing node. The remainder of this paper develops the problem in more detail, presents a solution architecture, and shows why it is optimal.

In comparing alternate interconnection architectures and picking an optimal solution, the metrics used here 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 minimal computational modules. For evaluating the speed of a network, the time required for one *wave* of node computations and transmission to be fully propagated through the system will be used. This measure of speed will be more completely defined in Chapter 2. Complexity of design is of concern because additional complexity increases the

[1] Although this paper explicitly refers to silicon, any similar medium such as GaAs could be substituted in an actual implementation.

required area and leads to reduced fault tolerance. Finally, fault tolerance is critical because VLSI silicon is essentially a faulty medium and any large system must have built in capabilities to surmount those faults that will occur.

2. Background

Recently, the use of a *connectionist* model has gained popularity as an alternative computational paradigm for computer systems that display cognitive behavior. Connectionist models are based on a simplistic view of the structure of the brain's neural networks [AnH81, RuM86]. CNN systems 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 [FeB82].

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 [FeB82], "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.

Connection models are well suited to solving a variety of problems. Some of the applications developed are the NETtalk speech synthesis system developed by Sejnowski and Rosenberg [SeR86], speech recognition [PPP88], pattern recognition [Bal85], loan application ranking [PPP88], and natural language parsing [CoS84, PoW84, Sel85]. Problems with soft constraints, or where limits can be bent, are also ideal for the connectionist paradigm.

3. Communication Problem

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. As will be shown later, the optimal solution is to multiplex the physical communication channels. 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 a result, it is critical that the interconnection topology be chosen carefully.

There are a number of different interconnect topologies that have been proposed for linking multiprocessor systems. These include grid, or nearest neighbor; crossbar, or global interconnect; hypercube and its variations; shared memory; and various switching systems that provide virtual full interconnect by trading delay for complexity. All of these either

[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.

have insufficient fan-out to support the communication graphs of connectionist models, do not scale well, or require excessive VLSI area and complexity to implement as will be shown in Chapter 4.

Dally [Dal86] has shown that, since there is a fixed amount of bandwidth available for interprocessor communication, the problem is how best to allocate it between local and non-local communications. The major point of his work is that a system with a low *degree*, i.e. a four, or fewer, dimensional folded torus, performs better than higher degree structures such as the hypercube. This better performance is due to the reduction in the number of long, slow connections. With appropriate forwarding capabilities at the intermediate nodes, a system has more flexibility when it does not dedicate communication bandwidth to long connections.

This paper presents an architecture, the Augmented Broadcast Hierarchy (ABH), that has been optimized for the large fan-out, highly localized communications of connectionist networks. The large number of messages that a node transmits, when a state change occurs, are best supported by a broadcast method as will be shown in Chapters 3 and 4, where broadcast communications are compared with point-to-point for cost and degree of contention. Since the particular CNN models we are interested in have relatively sparse interconnection matrices, it is not practical to have one large broadcast region spanning the entire system. Thus the idea of multiple, overlapping regions arranged in a hierarchical manner to optimize bandwidth and required address space. Finally there is the need for a few long distance connections. A portion of the bandwidth must be reserved and a method provided for these long distance messages. Chapter 4 will present a technique for supporting a few long connections.

The use of a hierarchical organization for communication is not a new concept. The phone system is a well known example of a communication hierarchy that provides local access with a short address, the local phone number, and access to larger regions with longer addresses, area and country codes together with the local number. The *cm** multiprocessor computer developed at CMU, used a hierarchical bus structure to let a process access increasing amounts of memory with an incremental time delay. The Altera company uses two levels of buses, local and global, in their EP180 programmable logic devices to decrease contention on the global bus. There are many more possible examples, too many to include all of them here. ABH is different because it uses a hierarchical medium for *broadcast* messages, to provide a large communication fan-out, instead of for *point-to-point* messages. A precise characterization of ABH is provided in Chapter 4.

4. Summary

The major problem when implementing connectionist networks in silicon is determining how to best provide the needed communication paths. Connectionist networks are characterized by having large fan-in and fan-out and potentially requiring an excessive amount of bandwidth. This paper proposes a solution to this problem, ABH, and compares it with the other solutions that have been proposed.

The organization of the remainder of this paper is as follows. Chapter 2 provides some elementary graph theory and definitions such as locality that are required for the remainder of the paper. Chapter 3 characterizes the common connectionist models and creates a set of "idealized" networks that will be used later to compare interconnection architectures. Chapter 4 shows how multiplexing of communication lines is required for large scale implementations, presents the possible interconnection architectures, eliminates the poorer alternatives and ranks the remaining ones using graph theory and area-time based arguments based on the requirements established in Chapter 3. Chapter 5

summarizes the paper and provides future directions.

CHAPTER 2

Definitions

This chapter provides an introduction to the graph theory and definitions required for the analyses in the following chapters. Graphs are defined that represent both the CNN model and the physical processor layout. The concept of mapping a CNN to a physical system is presented along with a method for ranking mappings. Finally, some characteristics of graphs are given that can be used for delineating families of graphs. These characteristics can also be used to predict the quality of the mappings that are possible for a family of graphs.

1. Introduction

Before the results of this paper can be properly presented, it is necessary to define the terminology used. Since the topic to be discussed is interconnection architectures and the communication requirements between processors, the basic terminology used comes from graph theory.

This paper is concerned with the communication and interconnect requirements of CNNs. In most of the analyses of this paper, the computation that is performed by a CN and any learning function of the network is ignored. The calculations of area requirements in Chapter 4 assume a simple $\sum - \prod$ computation. In this model, each input to a CN is multiplied by the appropriate weight factor, the resulting products are summed, and a *thresholding* function is applied to the result.

The use of a simplistic computational model is sufficient because a more complex CN will either be slower or require additional area. Thus, the model used here overestimates the communication costs in terms of time and area required so that any limits derived will be more stringent than necessary for a "real world" implementation.

The precise computations used in a CNN model only impact the architecture by setting lower limits on the required functionality of the physical processors and upper limits on the number of CNN nodes that can be effectively emulated by each physical processor. The major impact is from the number and the physical dispersement of the CNs that are inputs to and outputs from each node. This interconnection pattern can be represented as a *directed graph* with CNN nodes as *vertices* and the dependencies between them as *edges* in the graph.

Definition 1: A *directed graph* $G(V,E)$ is a set of vertices V and a set of edges E, where $E \subset V \text{x} V$. That is, each element of E is an ordered pair (i,j) where $i,j \in V$. The edge (i,j) is different from the edge (j,i) and is said to *originate* with vertex i and *terminate* with vertex j. □

For an example of a graph see figure 2.1, where the set of vertices 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) is an outgoing edge from vertex a and an incoming edge to vertex d. In this paper the term *node* is used as a synonym for vertex.

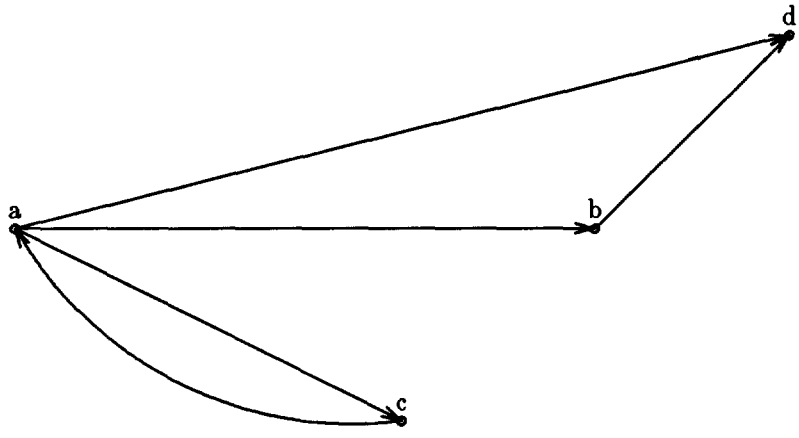


Figure 2.1 A Directed Graph

Definition 2: A *connection matrix* is an NxN matrix where the i,j entry represents the edge from node i to node j in the related directed graph. □

Figure 2.2 shows the connection matrix for the graph of figure 2.1. Note that a 0 in a given position indicates the absence of a connection and a 1 indicates its presence. It is possible to use values other than 0 and 1, with the magnitude of the i,j entry indicating some attribute of the connection between i and j.

Definition 3: A *c-graph* is the directed graph of a CNN where the CNs are represented by vertices and the connections between them are represented by corresponding edges. □

In a c-graph, the existence of the edge (p,q) implies that the output of CN p is an input to CN q. The edge (p,p) is not included in the c-graph, even if the past state of node p is an input to its update function, since the model is only of interprocessor communication. In general, c-graphs are asymmetrical, that is the existence of edge (p,q) does not imply the existence of edge (q,p). Asymmetry is not a requirement. Several CNN models, such as Hopfield networks [Hop82], have symmetrical graphs.

	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 2.2 A Connection Matrix

Definition 4: A *p-graph* is a directed graph representing the physical processor interconnect of a system. The vertices represent physical processing nodes and the edges represent the communication channels between them. □

Although the communication channel between any two PNs (physical nodes) p and q can be bi-directional, and in most physical systems we are considering, in a p-graph this situation is represented by including both edges (p,q) and (q,p). This convention is used because the connection may consist of a pair of physically separate conductors or a single conductor together with switching circuitry to turn it around. Also, more general models are possible than if only bi-directional connections were allowed. The existence of local memory in, or connections between multiple CNs assigned to, node p is not represented by the edge (p,p), again because the model is of interprocessor communication only.

2. The Mapping Problem

While it is possible to simulate or emulate a CNN with a single processor system, enhanced performance requires the use of multi-processor systems. This need can be readily seen by considering the computations involved in updating a CNN consisting of 10^6 CNs with each CN connected to only 1% of the other CNs. With one of the simplest of the proposed computational models, the $\sum - \prod$ model mentioned earlier, there are a total of 10^{10} multiplications and 10^6 additions for a single network update, ignoring the time required to record the new state for each node. For a *real−time* system that needs to perform thousands of such network updates per second, a solution other than the use of a single processor must be found.

While the easiest and fastest method of storing the data for a uni-processor system is by having a single array of CN states that is referenced for each computation, this is not feasible for a multiple processor system. With every CN computation accessing the same structure, even with a sparse matrix, there will be a large amount of memory access contention. The solution is either to use a message passing multi-processor system or a variation on the Ultra architecture developed at NYU [GGK83]. A typical shared memory multiprocessor system would not be appropriate, because of the high locality of memory references.

In order to effectively emulate a CNN on a message passing multi-processor system, it is necessary to partition the network and have each PN responsible for emulating a subset of the CNs. Two possible ways to partition a c-graph and to assign it to a p-graph are matrix splitting and graph embedding.

Matrix splitting consists of assigning a section of the connection matrix to each PN. The PNs then calculate the partial sums for the CNs they are assigned and forward these values to designated summing PNs, such as those on the diagonal, for inclusion in the final computations. After the summing PNs have completed their computations, they send new CN state values back to the original PNs and the cycle repeats.

A disadvantage to this approach is that, with each CN's state distributed over multiple PNs, there is the potential for data incoherency. That is, there are multiple copies of each CN, with each possibly in a different state at the same time. To solve this problem requires synchronizing to insure that all copies of each CN are in the same state before updating calculations are performed. A second problem with the matrix partitioning approach 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 that section are disrupted.

Matrix partitioning is primarily of value in situations where PNs are powerful, so they are able to update multiple CN states easily; the system is synchronous, to reduce the

probability of incoherent states; the connection matrix is uniformly dense, sparse connections reduce the efficiency of this approach and uniformity allows balanced use of all PNs; and interprocessor communication costs are expensive, so fewer, but longer, messages are preferable. Examples of message based multi-processor systems that would be appropriate for this approach are the BBN Butterfly and the Intel iPSC, with synchronizing messages broadcast system wide between node updates.

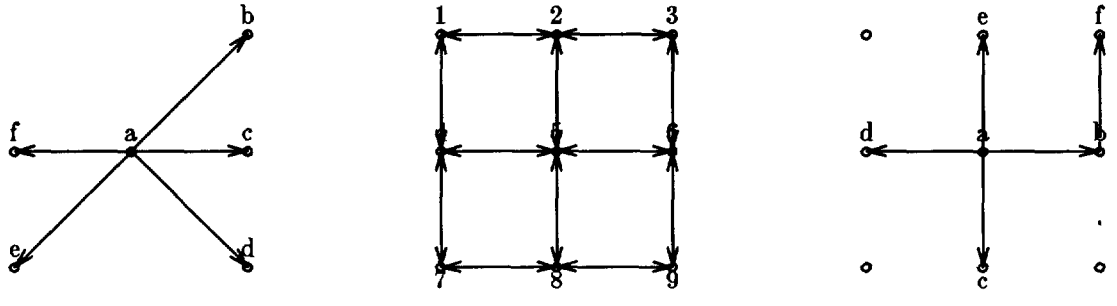
The second way to partition a c-graph is to assign one or more CNs to each PN and map each c-graph edge to the corresponding p-graph *path*. When a set of CNs has more *external* co-incident edges than the equivalent PN, either a group of PNs can be considered as a unit to provide the needed *fan-in/fan-out* or p-graph edges can be multiplexed and each c-graph edge mapped to a path in the p-graph. All intermediate PNs on a given path would only provide message forwarding, with no processing of messages not intended for them.

The primary thrust of this paper is to consider problems related to VLSI design, in particular the design of wafer-scale integrated CNN emulators, so processors are restricted in power and size, while the connections between them are fast and inexpensive, although constrained in number. That is, the design specification requires maximal performance for the cost, ruling out the use of large complex processors connected with an intelligent interprocessor communication structure. For these reasons, together with fault tolerance considerations, the matrix partitioning approach will not be considered further.

3. Mapping Metrics

Since the goal is to find optimal mappings of c-graphs to p-graphs, it is necessary to define a measure of mapping *goodness* so that mappings may be compared. If each edge in the p-graph is assigned a *cost*, where the exact definition of cost is not specified, but might include such factors as power loss or transit time, then it is possible to sum the costs resulting from a particular mapping to provide a quantitative rating for that mapping.

Definition 5: A *path* in a graph is a sequence of edges $e_1, e_2, \cdots, e_n$ such that if $e_r=(i,j)$ and $e_{r+1}=(k,l)$ then $j=k$. The *beginning* and *end* of a path are the vertices v_1 and v_m where $e_1=(v_1,j)$ and $e_n=(k,v_m)$. As used in this paper, paths are restricted to include no



c-graph c p-graph p $M:c \rightarrow p$

Figure 2.3 A mapping example.

loops. That is, no two edges of a path have the same starting or ending vertex. □

Definition 6: The *length*, $l(i,j)$, of a path is the number of edges it contains. □

Definition 7: A *mapping* $M:C \rightarrow P$, where C is a c-graph and P is a p-graph, is a function of $V_c \rightarrow V_p$ and $E_c \rightarrow P_p$ (P_p is the set of all paths in P) such that if $(i,j) \in E_c$ then either $M(i) = M(j)$ or $M(i,j) = p \in P_p$ and $M(i)$ is the beginning of p and $M(j)$ is its end. □

In other words, a mapping from a c-graph to a p-graph is an assignment of each CN to a PN. In addition, each edge connecting two CNs either disappears (when both CNs go to the same PN), or is mapped to a path in the p-graph that connects the destination PNs and preserves the direction of the connection. An example of such a mapping is shown in figure 2.3. Vertex a has a fanout of five so that path {(5,6),(6,7)} is used to connect a and f, while all the other edges of c are mapped directly to edges of p.

Definition 8: The *cost* of a mapping M: C -> P is $\sum_{e \in E_C} c(M(e))$ where e is an edge from E_C, the set of edges of C, and c is a function, that assigns a non-negative value to each path of P. □

If each edge traversed is assigned a cost of 1, then the mapping in figure 2.3 has a total cost of 6. This is the lowest possible mapping cost given the graphs c and p. A poorer mapping is {(a,1), (b,7), (c,8), (d,9), (e,6), (f,3)}. It would have a cost of 2 + 3 + 4 + 3 + 2, or 14, twice the cost of the minimal mapping.

It is also possible to assign a cost to PNs based on size and complexity. Supporting more CNs with each PN will increase the area required, so the cost function of a mapping could include a PN cost function to show the effect of varying the CN/PN ratio. In the work presented here, it is assumed that there is an upper limit to the number of CNs per PN. All mappings that do not exceed this limit are given the same cost. This simplifies the calculations of cost and is sufficient for the purposes of this paper.

The optimal, or target, mapping is the one with smallest cost. To restate this in another way, the best mapping of a c-graph to a p-graph is the one in which each CN is placed to minimize the cost of the paths to all other CNs to which it is connected.

4. Graph Measures

There are a variety of measures of graphs that can be used to predict the capabilities of the systems they represent. The measures defined in this section are used in Chapter 4 to characterize some of the CNN models that have been proposed. These values indicate the amount of communication and the ease of mapping the c-graphs to different p-graphs.

Definition 9: 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 $\frac{E}{V^2-V}$ where E is the number of edges and V is the number of vertices in G. A *sparse graph* is one with a low density. □

For example, the graph of figure 2.1 and 2.2 has a density of 5 / 12.

Definition 10: The *degree of a vertex* v of a graph G is the total number of edges incident to it. The number of edges leaving v is its *fan-out* (divergence) and the number of edges entering v is its *fan-in* (convergence). □

Definition 11: The *maximum degree of a graph* G is the degree of the vertex in G with maximal degree. □

Definition 12: The *average degree of a graph* G is the average of the degrees of its vertices or $\frac{2E}{V}$ where E is the number of edges and V is the number of vertices in G. □

Definition 13: The *reachability function* of a graph, $R(n)$, indicates the number of nodes that can be reached and how far they are from a given node. Let $r_q(n)$ be the number of nodes in the graph reachable by paths of length n from node q. Then $R_q(n)$ is the total number of nodes reachable in n steps, $R_q(n)=\sum_{i=1}^{n} r_q(i)$. $R(n)$ for the graph is the average such $R_q(n)$ for all nodes q, or $R(n) = \frac{1}{N}\sum_{q\in V}\sum_{i=1}^{n} r_q(i)$ for V the set of all nodes in the graph and N the size of V. □

For regular graphs, and ignoring boundary conditions, $R(n)=R_q(n)$ for all nodes q. For example, rectangular grids have $R(n) = \sum_{i=1}^{n} 4i$. Any connected graph of N nodes has *diameter* n' where $R(n') = N$ and $1 \leq n' \leq N$. For a fully connected graph, i.e. one with a connection between every two nodes, $n' = 1$.

Definition 14: The *locality function* is the inverse of the reachability function, $L = R^{-1}$. That is, $L(R(n)) = n$. □

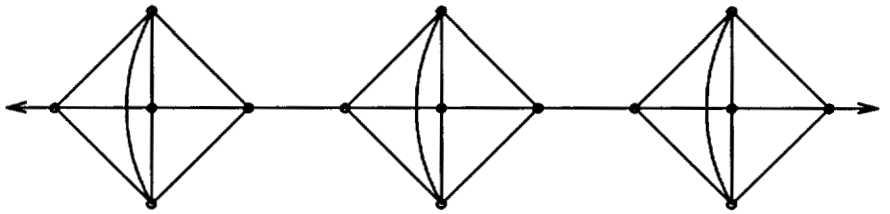
In other words, the locality of a graph is the number of steps required to reach some number of nodes. For example, in the above mentioned rectangular grid, $L(24) = 3$ since $\sum_{i=1}^{3} 4i = 24$.

Figure 2.4 shows three different graphs of degree four. Although they all have the same degree, they have widely varying reachability functions and can be mapped easily to different architectures. For example, consider infinite graphs of these types. Graph A maps readily to a line, both graphs A and B map readily to a plane, and graph C is not generally mapable.

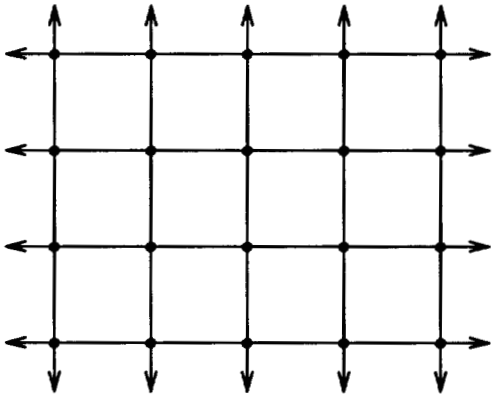
The relative reachability and locality of the c-graph that is being mapped and the p-graph it is being mapped to place a lower limit on the mapping cost. Even with an optimal mapping, as long as the number of CNs per PN is held constant, the ratio of the average degree of the c-graph, and consequently its reachability, compared with the average degree of the p-graph determines the number of p-graph edges, of length greater than one, that must be used. In other words, when every PN has fewer outgoing edges than the CNs that are mapped to it, some edges must be multiplexed. There are additional costs from these added p-graph edges. Topologies with a larger amount of interconnect, or equivalently a larger average degree, show better speed and performance in supporting arbitrary c-graphs. This topic, how graph characteristics limit implementation and mapping optimizations, is developed more fully in Chapters 3 and 4.

5. Networks and Layers

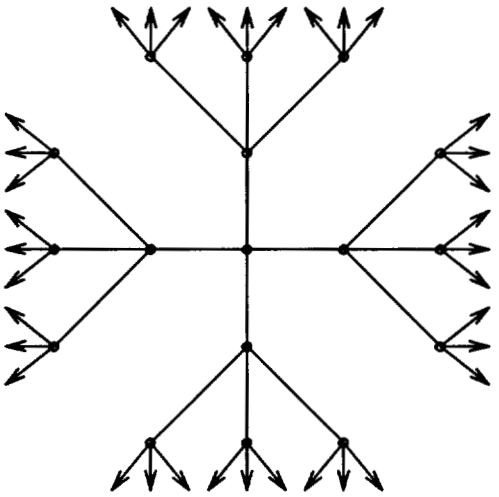
Although the functionality of a CNN is not of major interest here, its logical structure can provide insights into the communication requirements. Most neural network models are organized into groups of CNs called *layers*. Figure 2.5 shows the organization of a typical *feed-forward* network consisting of three layers. As is apparent from this figure, and the c-graph generalizations of Chapter 3, the amount of inter-layer and intra-layer interconnect directly impacts the overall communication requirements of the network being



A. Constant rate of increase



B. Linear rate of increase



C. Exponential rate of increase

Figure 2.5 Constant fanout, varying locality

emulated. Since each layer can be considered as a separate entity, in the remainder of this

paper most analyses will be of a single layer and its inputs and outputs.

6. Summary

This chapter has introduced the concept of a graph as a way of representing the interconnection and communication structure of CNNs and the physical systems used to emulate them. Specific definitions included the *c-graph* or connection network graph, the *p-graph* or physical system graph, a *connection matrix* as a representation of a networks interconnect, the *density* of a graph as a measure of the relative number of edges it contains, and the *degree* of a graph as indicating the number of connections to each vertex. The idea of a *mapping* from a c-graph to a p-graph was presented together with a method of measuring the relative *cost* of such mappings. In addition, *locality* and *reachability* were defined to help with the intuitive concept of similarity of structure of two graphs and *layers* were shown as a means of reducing a network graph to simpler subgraphs.

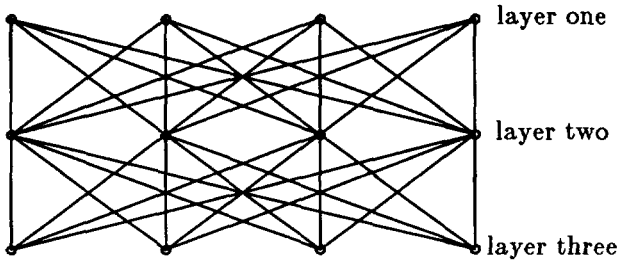


Figure 2.5 Three layer feed-forward network

CHAPTER 3

CNN Characteristics

The CNN computation model implies a set of characteristics that must be taken into account in designing a physical system for efficient CNN emulation. These characteristics include: asynchronous communication, large degree graphs, short messages, and predictable message delays. In this chapter these characteristics are developed along with some of their implications. The final result is several abstract c-graph models, distilled from the variety of current CNN proposals, that are used in later chapters.

1. General C-graph Characteristics

In addition to the graph measures introduced in Chapter 2, there are other characteristics of the communication requirements of CNNs. This section introduces several of them and shows what the ranges of expected values are. Later in this chapter, specific CNN models will be considered using this metrics.

1.1. Transmission Frequency

The first area to consider is the temporal spacing of messages. Many of the current CNN simulators and accelerators sequentially step through the nodes and calculate a new value for all before starting a new pass. This is in contrast to many of the current mathematical, and biological models which are asynchronous. That is, each node is continually recalculating its output value based on the current state of its inputs. Each calculation introduces a computation delay to the propagation of the input through the system, in addition to the communication delays from the message transit time. Changes in network state move through the system in a series of self-timed *waves*. Not all nodes in a given region or layer fire at the same time, so in determining the required bandwidth the expected percentage of nodes firing, not the maximum possible number, should be used. Also, when a node has recently changed its output, it is unlikely that it will change again for some time. It usually takes a change in multiple inputs to modify the output state. These various delays provide a leveling in the number of messages in the system at a given time. Peaks tend to be dampened out due to differential recovery intervals.

Since, in an asynchronous model, a node may receive update messages at any time, it is necessary to specify if the computation in progress should be restarted in reaction to new inputs. For example, consider a $\sum - \prod$ node, as described earlier, that has summed all input-weight products to some point when an input, that was already factored in, changes. Should a new calculation immediately commence, or should the current one run to completion? This question is dependent on the particular method used for computation in a system. Implementations using analog computation, or that calculate a new output state by modifying the current one by the amount of the change in the input received, are not as susceptible to this problem as those that redo the entire calculation.

It is also important to determine the frequency of changes in the output state of the nodes. This frequency is dependent on the precision used for the state transmission, the old

state of the node, and the particular step function used to generate the output. If only one or two bits are transmitted, then only gross changes in state result in new messages. When more precise values are sent, a much smaller change in state will generate a message and each node will transmit more often, resulting in more messages in the system at any given time.

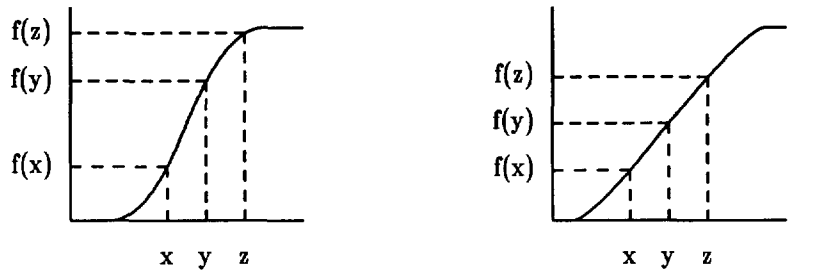
The previous state is important, as shown in figure 3.1-a, where a sigmoid function is used for illustrative purposes. A small change, from state x to y, causes a large change in the value to be transmitted, while the same amount of change, from state y to z, does not affect the output as much and no new message results. This type of behavior will occur whenever the node state is in a *critical* region, that is dependent on the particular function being used. A more linear function, as is used in figure 3.1-b, does not have as steep a slope in the critical region, so similar amounts of change in output are generated for both changes in node state, possibly resulting in three transmissions instead of two.

1.2. Delay Patterns

Communication delay includes the time required for wire transit, as well as for relaying a message through any intermediate nodes. In addition, using serial transmission adds an increase in time when sending longer messages. The amount of delay is dependent on the internode bandwidth. If a message sixteen bits long is sent over a communication channel that is *bit serial*, it will take four times as long as when it is sent over one that has four parallel lines, ignoring the time required for *packing* and *unpacking* the message at each end.

Definition 3.1: The *delay* between two nodes i and j, $d(i,j)$, is the amount of time from the start of transmission by node i to the end of receipt of the message by node j. □

The absolute amount of delay in a network is critical in determining if the throughput is sufficient to meet the requirements of the task the system is designed to perform. The magnitude of this delay should not impact the functionality of the underlying computational model. That is, using a faster clock in a system, for both communication and computation subsystems, will not change anything but the overall performance. It is possible that changing the speed of communication, relative to the computation speed, will



a - Steep critical region

b - Shallow critical region

Figure 3.1 Output function variations

affect the convergence of the CNN model. Insufficient research has been done in this area to determine how sensitive the various models are.

In addition to the absolute magnitude of the delay in a network, its distribution, *filially*, *temporally*, and *spatially*, may well impact the performance of the underlying CNN model.

Definition 3.2: The *temporal variance* in delay between two nodes i and j, is $d(i,j)$ at time t_1 *minus* $d(i,j)$ at time t_2. □

In other words, it is the difference in the time required for a message to travel between the same pair of nodes at two different times. Temporal delay variance may be caused by differences in routing of different messages or by contention for the resources required for the communication. Mike Rudnick and Steve Neighorn of OGC have empirically shown that large temporal delay variances adversely impact convergence of Hopfield type networks. Unfortunately, not enough research has been performed to adequately characterize the resilience of various CNN models to temporal delay or to determine the absolute maximum that can be tolerated by each.

Definition 3.3: *Filial variance* in delay is $d(i,j) - d(i,k)$. □

That is, filial delay variance is the difference in the amount of time required for a new state change to propagate to two different children of a cell. Filial variance may be caused by messages having to travel different distances or by either of the causes listed above for temporal variance. As Conwell has demonstrated [Con87], the Hopfield model is able to tolerate a limited amount of filial variance, as long as the variance is small compared to the update frequency of the network. Larger variances slow down convergence and may lead to non-convergence in some pathological circumstances. Variances that are large enough relative to the computation time to be significant, are equivalent to adding nodes to some paths that exist only to relay the messages. Since these new nodes only appear on some links, they are equivalent to removing symmetry from the Hopfield model, so the proofs of convergence no longer apply. No one yet has determined the maximal amount of filial variance that can be tolerated or how other CNN models perform in its presence.

Biological models, on the other hand, often display a wide range of filial variance. A neuron is normally connected to other neurons at varying distances. Also, the presence or absence of different neurotransmitters can change the, relatively large, delays involved in transiting synaptic gaps.

Definition 3.4: *Spatial variance* in delay is $d(i,j) - d(k,l)$. □

This final form of delay variance is simply a difference in communication speeds in different parts of the system. It can be caused by any of the above reasons, as well as by different connectivity or computational requirements in various parts of the system. No research results have been published on its impact on network performance, but cursory inspection of the models leads to the conclusion that it is not as serious a potential problem as the other two types of delay variance.

In light of the lack of knowledge about the effects of these types of variance on network convergence, and the fact that most CNN simulations to date have been restricted to no unexpected variance and still show convergence problems in some circumstances, this paper will assume that all introduced variances should be minimized or eliminated wherever possible. That is, only those delays that are inherent in or required by the CNN model should be kept.

1.3. Message Length

Unlike the messages in typical computer networks, interCN messages are short. All that needs to be transmitted is the identity of the sending CN, the intended receiver, and the new state. Researchers have shown that a few bits of information are sufficient in most cases. This characteristic implies the use of packet networks rather than dedicated or switched communication circuits between CNs. It also reduces the need for buffer space in the communication system.

1.4. Fanout Requirements

The potentially most damaging requirement of CNNs is their large degree. Some of the neurons in the human brain have 10^4 to 10^5 connections and, as is detailed later in this chapter, many CNN models call for full interconnect, either within a layer or between each pair of adjacent layers. This high degree requirement is a result of the need to correlate many divergent inputs. Models such as low level image processing only compare near-by values and have a much reduced fanout requirement.

This is a drastic mismatch with 2D VLSI, with its limited number of wires, and 3D VLSI is not sufficiently better, as is detailed in Chapter 4. As also shown there, the area required for interconnect is $O(n^3)$ of the number of connections required. Any p-graph must support mappings from high degree c-graphs and the only way to do this effectively is via high virtual degree. Unfortunately, using virtual fan-out implies increased possibilities for contention and delays.

2. Typical C-graphs

A wide variety of CNN Models have been proposed as solutions to different problems. Two general classes are those networks consisting of a single layer of uniformly interconnected nodes, such as the "Hopfield networks", and networks with multiple distinct layers. This section will describe a number of CNNs and their interconnect requirements.

The networks proposed by Hopfield [Hop82] consist of a single layer of nodes with each node connected to every other node (some connections have weights of zero, but this can change during learning). Since this implies the existence of n^2 total connections and an average fanout degree of n, it is obvious that networks of this nature do not scale for large values of n. They may be used as a subregion in a network, such as a single layer, and will be included here for that reason and as a limiting case.

Another model that is uniform throughout is the *alpha* model. This network is not a formal CNN model, but is based on a crude approximation of the structure of the cortical region of the brain and is used for comparative purposes in this paper. The name comes from the method of creating the c-graph, because the probability of a connection between any two nodes is α^{-d}, where d is the distance between the two nodes (using p-graph path length as a measure of distance). That is, two adjacent nodes are likely to be connected, while two nodes that are on opposites sides of the graph are not.

Another model based on the brain is the Lynch-Granger network model which comes from analysis of the structure of the piriform olfactory cortex. [LGL87] Their model is a layered model with the first layer getting inputs from the outer world. At most 10% of the inputs of any given node are from external sources and each external source contacts a random distribution of the first layer, reaching maybe 10% of the nodes in it. The second layer is fed from the first and generates feedback to it, with each feedback node again randomly reaching about 10% of the nodes in the first layer. In addition there are small inhibitory regions in the first layer, with each inhibitory node reaching all excitatory nodes within a limited radius. This produces classical on-center, off-surround, feature extraction capability

similar to that detailed in the work of Carpenter and Grossberg. [CaG86]

Due to its current popularity, the back propagation CNN model [RHW85] must be included in this list. It consists of a number of layers with every node in a given layer receiving data from every node in the preceding layer and transmitting to every node in the succeeding layer, similar to the network pictured in figure 2.5 During learning it is necessary for the *errors*, or differences between the values output and those expected, to be propagated back through a complementary network. Hammerstrom [Ham88] has shown that reduced interconnect is possible in some circumstances, but the more general model will be used for comparisons here.

The NeoCognitron [FMI83] is an interesting variation on the idea of a feed forward network because the nodes in a layer are grouped into smaller regions with intraregion inhibitory signals. Also, each node in a layer only receives inputs from a subset of the prior layer and transmits to a subset of the subsequent layer.

The general ART model [Gro86] has full interlayer and intralayer interconnect with each node receiving from the remainder of its layer and all of the preceding layer. Transmissions are to the entire succeeding layer as well as to all of the other nodes in the same layer. Localized versions have been proposed [RyW87] that resemble the NeoCognitron more closely, but here again the more general model is used for comparison.

3. Abstract CNN Model

The above CNN descriptions, while capturing the essence of the interconnection requirements of the various models, are not adequate for formal analysis. This section will attempt to define a set of abstract measures that are. As suggested in Chapter 2, the use of a single layer as the region of comparison will clarify the analysis.

Consider a layer consisting of N nodes with a total of I inputs and O outputs to the layer. Then each node p is connected to $0 \leq I_p \leq I$ inputs, $0 \leq O_p \leq O$ outputs, and $0 \leq N_p \leq N$ other nodes in the layer. It will be assumed that all nodes in the layer have the same values for I_p, O_p and N_p. While simplifying the analysis, this assumption still allows the development of limits that are valid in the more general case.

With this definition, the above CNN models can be reduced to three cases. The first consists of the NeoCognitron, Lynch-Granger, and alpha models where $0 < O_p \ll O$, $0 < I_p \ll I$, $0 < N_p \ll N$. The second is the feed forward networks like Back Propagation (without the feed back of learning). In this class $N_p = 0$, $O^p = O$, and $I_p = I$. The final group consists of the ART and Hopfield networks where $N_p = N$, $O_p = O$, and $I_p = I$. These three cases will be subsequently referred to as the **alpha**, **bp**, and **art** classes for consistency.

4. Summary

In this chapter the general requirements of CNN models have been presented together with how each requirement may impact p-graph models. Then the interconnect requirements of a variety of current CNN models were given. Finally an abstract model was distilled that detailed their interconnect requirements in a parameterized fashion.

CHAPTER 4

Physical Interconnection Alternatives

This chapter begins by showing why multiplexing of communication lines is required for large scale implementations. Then a variety of possible interconnection architectures, including the solution proposed by this paper (ABH), are presented and ranked by how well they meet the CNN requirements introduced in Chapter 3. Finally, a limited subset of the architectures are selected for further study.

1. Why Multiplex

As introduced in Chapter 3, the CNN computational model places constraints on any VLSI emulation. These include asynchronous nodes, large degree, bursty short messages, and predictable message delays. One obvious solution to all of these is having isomorphic c-graphs and p-graphs, with a mapping that assigns each CN to a unique PN and each c-graph edge going to the equivalent p-graph edge. Then, each node is separate and can fire at its own rate and messages can traverse the unshared links with only wire transit delays. Since this solution so clearly fits all the requirements, what problems does it have and why does this paper reject it?

The major problem with the above proposed *direct* implementation of c-graphs is creating a physical system with the appropriate interconnect. There are three possible alternatives: provide connections between each pair of nodes, set up a limited number of wires from each node to some interconnection area where selected pairs are joined by switches or fused links, or to lay out the VLSI design with the exact connections needed. Before addressing each of these options, consider the area required for interconnect in general.

The area needed by the connections between nodes is the product of the wire pitch, or wire width plus the required interwire spacing, times the number of wires. Assume the physical layout is a grid of PNs. For any ***uniform*** distribution of sources and destinations, the number of connections between any two PNs is equal to the number of wires that cross any vertical *bisector* of the grid along any horizontal row. This is a simple function of the number of PNs that a connection can be made to, times the probability that a given connection is long enough to cross the boundary, summed over all nodes on one side of the boundary, times two to take into account the two directions that a connection may take. A formula capturing this statement is:

$$C = 2\left(\sum_{i=1}^{\infty}\sum_{j=0}^{\infty} f(i+j) + 2\sum_{i=1}^{\infty}\sum_{j=0}^{\infty}\sum_{k=1}^{\infty} f(i+j+k)\right) \qquad 4.1$$

This reduces to:

$$C = 2\sum_{i=0}^{\infty} i\,(f(i) + (i+1)f(i+1)) \qquad 4.2$$

Where C is the number of connections and $f(x)$ gives the probability of a connection between two PNs x distance apart. Setting $f(x)$ to allow connections between all nodes within a fixed distance k,

$$f(x) = 1 \ \ (for \ x \leq k)$$

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

yields

$$C_k = 1/3 \ (k^3 - 6k^2 - k) \qquad 4.3$$

Since k, or the radius of the circle of connections, is a function of the number of nodes connected to, it is apparent that the area required is $O(n^3)$ in the number of connections. The same limit is developed in a different way by Hammerstrom in [Ham86]. With the *total interconnect* solution implying the existence of n connections, it obviously will not scale for very large connection networks (VLCNs). Thus one of the other two alternatives must be used to limit the rapid growth of area as the number of nodes increases.

The use of fused links is proposed by Raffell, Mann, and others at MIT in [RMB86]. This is a reasonable solution, as long as the degree of the c-graph is limited to a relatively small number of connections, on the order of tens or maybe a few hundred, and the required connections are known before the final manufacturing steps. It suffers from the problems of scaling and the cost of custom effort for each system. The use of programmable switches to make the connections, while solving the problems of flexibility and custom effort, also does not scale well and, in addition, requires much more area for the implementation of the switches and the circuitry to set them.

The only remaining alternative is laying out the actual connections needed when designing the p-graph. Due to the cost, both in time and effort, of laying out random or semi-random wires, this option is only available when the c-graph exhibits a high degree of regularity. Also, since the above calculation of the area required for interconnect assumes maximal locality, if the c-graph has very many edges mapped to long wires in the physical system the area required will increase even faster than shown in equation 4.3. As in the above option (fused links), direct suffers from being limited to systems with limited c-graph degree. CNN systems with thousands of connections between nodes are not feasible with direct implementation. The final problem is that, with the existence of some long wires between nodes, it may be necessary to add *repeaters* to maintain signal strength and speed for an additional increase in cost and area.

On the other hand, multiplexing of communication lines is no panacea. While it is easier to design and implement and scales much better than direct; multiplexing requires synchronization or contention resolution algorithms because of the competition for a shared resource, places more reliance on each line so the overall design is less fault tolerant, leads to the use of digital rather than analog communications which impacts the possible implementation of some CNN models, and can lead to unpredictable delay patterns due to contention. For these reasons and because non-multiplexed systems do not scale for VLCNs, it is necessary to create a multiplexed architecture that minimizes the effects of these problems.

2. Shared Interconnect Options

Assuming multiplexed communication, the possible interconnect architectures that have been proposed by other researchers are: switched, grid, high-dimensional, and bus. The *switched* class consists of all architectures where the set of PNs are connected via a

switching circuit. *Grid* is a two dimensional network with nearest neighbor connections and some forwarding mechanism to pass messages on to more distant nodes, it also includes the *torus* structure. By *high-dimensional*, I mean a grid with additional connections to non-adjacent nodes, this category includes such architectures as the hypercube. Finally, a *bus* architecture has all nodes connected with a single shared communication channel. Each category has its own advantages and disadvantages that will be examined in determining how best to allocate the fixed amount of potential bandwidth and silicon area to get the maximal level of system performance.

2.1. Switching Networks

Switching networks have historically been developed to connect multiple processors and memories together. Most provide the capability of a virtual *crossbar*, or full interconnect, but with reduced cost and complexity. Examples of switching networks include the shuffle-exchange, fat trees, and hashnet. The main drawbacks to the use of switching networks are the delay in transitioning them and the area required. The next three sections discuss how architectures based on the above examples would perform as connectionist emulators.

2.1.1. Shuffle Exchange

Shuffle exchange is included here as an example of a multistage permutation network. Stone in [Sto81] provides a comparison of a variety of similar switching networks. Figure 4.1 shows a small shuffle-exchange network. Any input can be connected to any output for the price of the time required to transit log n layers. The problems are: it is not possible for a given node to receive inputs from multiple other nodes in one time step, wiring the PNs to the switch itself (since the switch is a single large node connected to all other nodes, the interconnect area grows as shown in equation 4.3), the switch is a critical element and may not be fault tolerant, and the switch itself requires a large area.

2.1.2. Fat Trees

Fat Trees are an interconnection topology that exhibits good theoretical behavior [Lei85]. The original work was done as part of the effort in designing a communication network for the Connection Machine at MIT. See figure 4.2 for an example of a fat tree network. 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.

In conclusion, using an implementation that closely follows the one proposed by Leiserson, 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.

2.1.3. Hashnet

Scott Fahlman of CMU developed what he called the Hashnet as part of the design of the NETL system [Fah79, Fah80b]. Figure 4.3 shows how a hashnet system might appear. 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[Fah80a]. 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 CNN models do not map well to this alternative.

2.2. Grid

Since a grid is simply the PNs themselves laid out by rows and columns with connections between adjacent nodes, it provides both layout and routing simplicity. The torus is equivalent to a grid with opposing edges linked together (as shown in figure 4.4). A more common implementation of a torus is folding it so that all connections are the same length rather than having long wires between the edges. The large number of messages in a CNN presents a real problem for a grid system with excessive contention and inconsistent delays. In addition, there is the need for routing protocols and storage buffers and increased delays from message relaying. Even with these drawbacks, the simplicity of the grid is such that it remains as a viable alternative.

2.3. High Dimension Topologies

There have been many different high dimensional topologies proposed, for a comparison of the effectiveness of different elements of the class see [ReG87]. All are essentially grid layouts with each node connected to its neighbors and some set of other nodes determined by the defining algorithm. For example, the hypercube (as shown in figure 4.5) has each node connected to $\log n$ other nodes. While this added interconnect speeds messages that are for further away nodes, it may be wasted to some extent. As Dally shows in [Dal86], it is more efficient to have all the bandwidth available for all messages rather than have some dedicated for routes that may not be used. There is not sufficient gain in reducing the cost of forwarding messages through intermediate nodes to offset the loss of bandwidth that can't be used because it is to nodes far away from the routes required.

2.4. Bus

For small numbers of processors, a bus interconnect may be optimal. It provides the ability to send a given message to multiple destinations or to a single one with no difference in cost. All nodes have equal access to all other nodes. The problem is that it does not scale with the number of nodes. For any *fair* contention resolution protocol, each node with a message to send be able to transmit only during $1/n$ (where n is the number of transmitting nodes) of the time. This is obviously not adequate for networks with millions of nodes. In addition, each node on the bus must have dedicated access for transmission long enough for a message to propagate through the entire system, again totally infeasible for large networks. As the following section will show, however, there are ways to use the idea of a bus to create a good solution to the CNN emulation problem.

3. Why Broadcast

The above comparisons of interconnect architectures show that the major problem with all implementations is creating a p-graph with sufficient degree to support effective

mappings from a variety of c-graphs, but without the need to dedicate large amounts of area to rarely used, dedicated long connections. The solution proposed here is to use broadcast of messages as a way of creating large virtual fan-out by connecting a PN to all the other PNs within some surrounding area. Broadcast provides flexibility of connections, reduces memory requirements, shortens the message length, and depending on the implementation it can be more fault tolerant.

In programming CNN systems, one problem is adding new connections when they are needed. Most learning algorithms have the destination node adding the connection, not the node of origin. Unfortunately, when any point-to-point (PTP) communication technique is used, the transmitting node determines what nodes will receive a given message. Broadcast, using come-from addressing, solves this problem since any *listening* node can choose to accept a message or to ignore it.

When a PTP communication scheme is used, the sending node must maintain tables of addresses to send messages to. In addition the CNN computational model requires the receiving node keep tables of the addresses of incoming messages so it can assign the appropriate weight to the connection. The use of broadcast reduces the first requirement since only a single value need be stored to indicate the region to be broadcast to, instead of needing that the individual destination addresses be stored.

Similarly, the message length can be reduced by the use of come-from addressing. Typically in a PTP system each message must have a destination address or routing information in addition to the message itself. Unless some scheme is used to build the source destination as the message passes through intermediate nodes, as suggested by Leiserson in [Lei85], CNN models require that it be sent also to help with the weight assignment problem. Modifying the message when passing it through the network adds complexity to the intermediate nodes in any case. Come-from addressing allows dropping the destination address completely since every message is sent to every node in the region.

The transmitting node need generate only one message and specify the region it should go to, rather than having to generate a different message for each destination. This reduces the complexity of the transmitting circuitry and reduces the required area. At the same time, the number of destination nodes that can be reached grows as the area of the region transmitted to or as the square of its diameter.

Depending on the specific implementation used (as discussed later in this paper), broadcast can be made more fault tolerant than PTP. There are multiple copies of each message traveling around the network, not one, so if a node fails to forward a message it is possible to have the broadcast *flow* around the fault. This, while possible with PTP, is much more difficult and requires additional complexity in the routing nodes.

Unfortunately, broadcast is very inefficient when a large region is transmitted to in order to reach a small percentage of the nodes in it. Using a collection of regions organized into a hierarchy reduces this problem as shown in the next section.

4. The Broadcast Hierarchy Solution

When a CN updates its output, it transmits the new value to all of the nodes that it is connected to. No matter what the interconnection architecture is, this transmission of identical messages to many nodes is essentially a broadcast function. Designing the physical system around this fact is the basis of the Broadcast Hierarchy (BH) as proposed here.

The Broadcast Hierarchy (BH) is an interconnection topology 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 4.6. The following are some characteristics of BH that make it attractive for VLSI implementation:

(1) Messages are broadcast, this provides the effect of large fan-out plus all the advantages of broadcast listed above.

(2) The hierarchical organization allows for "Huffman" like encodings of addresses with resultant savings in memory area and message length.

(3) In addition, having multiple broadcast regions functioning in parallel reduces contention, since each node need only broadcast to the regions containing its destination nodes. The interconnect architecture allows the use of a variable length encoding of addresses with a resultant decrease in memory area requirements.

(4) BH efficiently emulates connection networks which exhibit locality of communication.

(5) 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.

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

A two dimensional BH 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.

There are two problems with a pure BH architecture. They are the long path problem and the high cost of crossing region boundaries. All CNN models defined in Chapter 3, with the possible exception of the alpha model, require some messages be sent to further nodes. There is no way, for example, to map a back propagation network to a BH system without requiring that many messages be sent via the highest layer. This negates all advantages of BH and comes back to the scaling problem inherent in any bus architecture. There are two instances of this problem. The first is when a node must transmit to a few

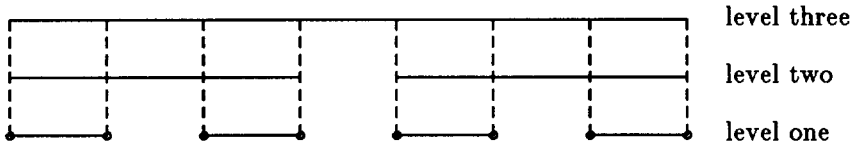


Figure 4.6. One-Dimensional Broadcast Hierarchy

nodes that are outside of its lower level broadcast region, and the second is when it must send to many. To solve both of these problems the Augmented Broadcast Hierarchy (ABH) was created.

5. The Augmented Broadcast Hierarchy

ABH, like its name indicates, is a BH system with some added features: 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 potential communication bandwidth then to require that the highest layer of the network be used. This is especially important when most of the CNs are in this situation, they would all require the top layer or with a limited PTP structure the many messages to nearby nodes can be handled with a lower broadcast level and the few long distance connections use the PTP system.

When a network has an interconnect similar to the alpha model, each CN transmits to most of the nodes in a circular region centered on it. This is fine for those nodes that are in or near the center of their broadcast regions, for those on the edge it requires the use of a higher level when it will only be used to reach a limited subset of the possible targets. Overlapping broadcast regions of similar size reduces this problem and is the architecture of choice for models that exhibit locality in this nature. Another possible addition is the creation of special nodes, physically located on the edges of broadcast regions, that periodically broadcast the entire state of one region to the other. This approach supports many layered models, but suffers from wasted bandwidth and unpredictable delays in relaying messages. Overlapping regions and relayed broadcasts provide different cost/benefit ratios and the choice of which to use depends on the c-graph being emulated.

Since the PTP network structure is being added to the BH system to increase flexibility, it should be based on a grid or torus interconnect. As shown above in the high dimensional discussion, dedicating bandwidth to links that may not be used can be wasteful and with ABH most messages are being sent with the broadcast medium.

6. Implementation Possibilities

The next question to consider is how to implement an ABH system. There are two possible solutions: physical and virtual interconnect structures. In a virtual broadcast hierarchy system (VBH) the interconnection structure is that of a grid or torus with messages being generated and sent by a broadcast algorithm. The main advantage is that VBH retains the simplicity of the grid architecture while gaining from the use of broadcast. In addition, it is possible to add fault tolerance fairly easily to such a system.

The physical broadcast hierarchy (PBH) system has a dedicated physical interconnect that supports the communication architecture. A variety of possible structures have been considered. They range from buses implemented in a higher level metal over the region to fat tree like layouts. The latter, as shown in figure 4.7 and described in more detail in [], is the option that will be assumed in this paper when necessary to have a physical layout for determining message transit characteristics. Messages are sent to the central broadcast node either through the PTP system or by a concentrator tree. They then are sent down the broadcast tree to each node in the region.

In comparing the two possible implementations, VBH is potentially more flexible since it does not have to be restricted to fixed broadcast regions. With appropriate routing algorithms, it is possible to have a different region for each node, removing the need for overlapping regions and directly supporting "edge detection" and other nearest neighbor type algorithms. The comparative performance relative to contention will be looked at in

the next chapter. Also to be determined there is the reproducibility of delay.

7. Summary

This chapter started by showing that the growth rate of the area required and the problems of laying out custom circuits for every CNN restricts the *direct* emulation of c-graphs to small networks. Then the possible interconnect options were introduced and compared by considering the requirements of CNNs. Broadcast was shown to gain much and ABH was developed as a solution that solves the problems posed. Now it is necessary to determine which of a few alternatives best supports the actual message traffic, as opposed to the interconnect requirements. The architectures that will be considered further are: **grid** or **torus**, **VBH**, and **PBH** (both with virtual concentrator trees and physical ones).

CHAPTER 5

Summary and Future Directions

This paper has introduced some of the problems inherent in emulating CNN models with VLSI based systems. The major one is the disparity between the fanout available with VLSI and the high degrees of c-graphs. Multiplexing was shown to be the only scalable way of building systems with more than strictly local connections. A variety of interconnect architectures were considered and the use a partially broadcast, partially point-to-point, network (ABH) was shown to be preferable.

The work that remains to be done includes simulation of the actual message traffic over time of the various CNN models. This information could then be analysed relative to the capabilities of a variety of p-graphs to see what the impact of congestion is on network performance.

Common CNN models need to be characterized more fully and a determination made of which p-graphs support which c-graphs and what variations are possible.

Finally implementation questions such as designability, fault tolerance, power requirements, potential speed, and heat dissipation requirements need to be answered for the remaining architectures.

Bibliography

[AnH81] Anderson, J. A. and Hinton, G. E., in *Parallel Models of Associative Memory*, J. A. Anderson and G. E. Hinton (ed.), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1981, pp. 9-48.

[Bal85] Ballard, D. H., "Cortical Connections and Parallel Processing: Structure and Function," Tech. Rep. 133, Computer Science Department, Rochester, NY, January 1985.

[CaG86] Carpenter, G. and Grossberg, S., "Neural Dynamics of Category Learning and Recognition: Structural Invariants, Reinforcement, and Evoked Potentials," in *Pattern Recognition and Concepts in Animals, People, and Machines*, M. L. Commons, S. M. Kosslyn and R. J. Herrnstein (ed.), Erlbaum Associates, Hillsdale, NJ, 1986.

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

[CoS84] 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.

[Dal86] Dally, W. J., "A VLSI Architecture for Concurrent Data Structures," Thesis, Pasadena, CA, 1986.

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

[Fah80a] Fahlman, S. E., *Design Sketch for a Million-Element NETL Machine*, Carnegie-Mellon University Department of Computer Science, Pittsburgh, PA, 1980.

[Fah80b] Fahlman, S. E., "The Hashnet Interconnection Scheme," CMU-CS-80-125, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, June 2, 1980.

[FeB82] Feldman, J. A. and Ballard, D. H., "Connectionist Models and Their Properties," *Cognitive Science*, vol. 6(1982), pp. 205-254.

[FMI83] Fukushima, K., Miyake, S. and Ito, T., "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-13, 5 (Sept/Oct 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.

[Gro86] Grossberg, S., *The Adaptive Brain (I and II)*, Elsevier/North-Holland, Amsterdam, 1986.

[Ham86] Hammerstrom, D., "A Connectivity Analysis of Recursive, Auto-Associative Connection Networks," Tech. Report CS/E-86-009, Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon, August 1986.

[Ham88] Hammerstrom, D., *The Connectivity Analysis of Simple Association - or - How Many Connections Do You Need?*, Proc. Neural Information Processing Conference, Denver, CO, November 1988.

[Hop82] 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.

[Lei85] Leiserson, C. E., "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Trans. on Computers*, vol. C-34, 10 (October 1985), pp. 892-901.

[LGL87] Lynch, G., Granger, R., Larson, J. and Baudry, M., Cortical encoding of memory: Hypotheses derived from analysis and simulationof physiological learning rules in anatomical structures, Unpublished Manuscript, 1987.

[PoW84] 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.

[PPP88] Proceedings of the ICNN, San Diego, CA, July 1988.

[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.

[ReG87] Reed, D. A. and Grunwald, D. C., "The Performance of Multicomputer Interconnection Networks," *IEEE Computer*, June 1987, pp. 63-73.

[RHW85] 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.

[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.

[RyW87] Ryan, T. W. and Winter, C. L., *Variations on Adaptive Resonance*, Proceedings of the ICNN, San Diego, CA, June 1987.

[SeR86] 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.

[Sel85] 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.

[Sto81] Stone, H. S., "Parallel Processing with the Perfect Shuffle," *IEEE Transactions on Computers*, vol. C-20, No 2(February 81), pp. 153-161.