Physical Broadcast Structure[1]

Mike Rudnick and Dan Hammerstrom
Dept. of Computer Science/Engineering
Oregon Graduate Center
Beaverton, Oregon

Technical Report No. CS/E-88-018
October 1988

Abstract

We present the design of an interconnect architecture that solves the inter-node communication problem posed by the emulation of large neural network models on silicon based neurocomputers [RuH88]. Our architecture is a hybrid using analog computation and multiplexed digital interconnect. Our long term goal is to emulate a million nodes, each with a thousand connections, on a single wafer.

For communication performance efficiency, we found it necessary to place constraints on the connectionist models emulated. First, a model must use only local information: each CN's (Connection Node) update function can use information only from its inputs and internal state. Second, the model must have communication locality: each CN's output must project predominantly to nearby CNs for any reasonable planar mapping. Both constraints provide a balance between physical communication resources (metal interconnect) and network communication requirements (bandwidth and latency); they serve to limit global connectivity.

Our interconnect system is the Augmented Broadcast Hierarchy[2] (ABH). ABH has two parts. The first is a domain oriented broadcast communication system for efficient local communication. The second is a point to point message delivery system for more remote communications.

This paper discusses the design and implementation of the broadcast portion of ABH, the Physical Broadcast Structure (PBS). The PBS is a high speed, pipelined, domain oriented,

tree-based communication structure. CNs are grouped into regions called domains. Each domain has a PBS for local message traffic.

A single PBS consists of two k-ary trees called a *dual tree* — a concentrator tree and a broadcast tree. The trees are connected at their leaves and roots. Their leaves are the CNs within a domain. The concentrate tree delivers messages from the CNs to the root. The broadcast tree delivers copies of each message from the root to all CNs within the domain. We use an adaptation of Leiserson's fat-tree to maintain throughput despite the geometric increase of interlevel line length with tree level.

SPICE time and power simulation results are presented. A CMOS PBS prototype chip is described that has been fabricated and tested.

Table of Contents

Figures

1. Introduction

The purpose of this report is to describe and analyze a single physical broadcast structure (PBS). PBS is intended to efficiently handle local inter-node communication as a part of the Cognitive Architecture Project (CAP) [HMT86]. We document the motivation for each PBS design decision. The goal of the PBS design is to maximize PBS cost/performance in the context of CAP operation. A synopsis of this report appears in [RuH88]

§ 2 provides a brief overview of CAP [BBB, HMT86], and § 3 presents the context in which PBS functions. § 4 presents PBS in detail, including a motivational evolution of its design. § 5 presents speed and power consumption analyses, including propagation delay speed and power simulation results. § 6 and 7 discuss an overview of fault tolerance and layout issues. § 8 discusses the implementation of a VLSI CMOS chip, called the TBH Test Chip, that demonstrates the PBS design. § 9 proposes areas for future research and speculative applications of current and future technology to PBS designs. Appendix I is the microarchitectural specification for the TBH Test Chip, and appendix II shows the calculation of the numerical results.

2. The Cognitive Architecture Project

The CAP goal is to design and build a neurocomputer capable of emulating connectionist models with a million nodes and a thousand connections per node. VLSI CMOS silicon was chosen because its functional density, simplicity, reliability, maturity, availability, and low cost are unmatched by other technologies. It lends itself well to implementing massively parallel, regular structures. We believe that the adaptive character and inherent fault tolerance of neural network models will largely compensate for the implementation faults intrinsic to CMOS process technology.

A major problem with massively parallel silicon neurocomputers is connectivity. CMOS is planar (two dimensional), currently providing 2 or 3 levels of metal interconnect, thus limiting connectivity. Neural network models tend to be high-dimensional with large numbers of interconnections. It is this connectivity mismatch that the augmented broadcast hierarchy design (described below) attempts to overcome.

2.1. Overview of Connectionist Interconnect Architectures & Issues[3]

A spectrum of possibilities exist for massively parallel connectionist architecture interconnect structures. They range from direct interconnect, or zero multiplexing, to fully multiplexed interconnect. When a connectionist model is mapped to a planar silicon surface using direct interconnections, the minimal area required is $O(n^2)$, where n is the number of connections per node [Ham86]. Achieving even this lower bound requires optimal mapping and a maximal degree of spatial locality (defined later) in the model being mapped. Because it does not scale well, direct interconnect is undesirable for large, highly connected neural network models.

Multiplexing — sharing connections between multiple nodes — reduces interconnect area in proportion to the degree of multiplexing. It is particularly appropriate for silicon based systems, because of the speed differential between silicon circuits and biological neurons; silicon circuits are four to seven orders of magnitude faster than neurons. Hence, the same message

[3] Portions of this section come from an unpublished summary of CAP efforts in interconnection architectures written by Mike Rudnick, Jim Bailey, and Dan Hammerstrom.

traffic can be handled by sending many messages quickly over one communication channel (metal) as by sending one message over each of many slow channels (axon processes).

Using connectionist models with a high degree of spatial locality reduces interconnect costs - most of a node's connections will be to "nearby" nodes. Local broadcast communication allows shorter node addresses and efficiently handles the large fan-out of local connections. Point to point communication is used for the relatively few remaining long-distance connections and for network input and output.

The CAP architecture's interconnect system is the Augmented Broadcast Hierarchy (ABH). ABH has two parts. The first is a domain oriented broadcast communications system to efficiently handle local connections. The second is a point to point message delivery system for longer connections. CAP has developed two alternate ABH broadcast domain designs, the physical broadcast structure (PBS) and the virtual broadcast structure (VBS). PBS uses a physical interconnect structure to broadcast messages. A concentrate tree funnels all messages to a central location. From there a broadcast tree delivers a copy to each node in the domain.

VBS uses a grid interconnect structure. Where in PBS the broadcast regions are formed by physical communication channels, in VBS they are formed by routing algorithms. VBS provides greater flexibility and reduced interconnect complexity, but performance is reduced because of increased contention and message delay.

A broadcast domain consists of a local region of nodes. When a node in the domain changes state it generates a message containing its identifier and new state. A broadcast structure delivers the message to all nodes in the domain. Nodes "connected" to the sending node record the new state. Other nodes ignore the message.

Broadcast domains may overlap in a variety of ways. A simple yet powerful approach is to rank them into levels by size so that each level's domains are larger, and hence include more nodes, than those of previous levels. A node broadcasts at the lowest level sufficient to reach most of its connections, leaving the relatively few remaining long-distance connections to the point to point communication mechanism.

Benefits of the broadcast domain approach include: Issuing a single message notifies all nodes in the domain of a state change. Messages are shorter since they do not include destination addresses. Finally, addresses are shorter because they only have to be unique within the local broadcast domain.

3. Neurocomputer Design — Architectural Issues

3.1. Processing Parallelism

This section introduces a number of definitions.

Definition 1: A *c-graph* is the directed graph representing a connectionist model to be emulated. The c-graph's directed edges represent inter-node connections, while vertices represent the model's nodes (sometimes called neurons, units, processing elements, etc.).□

Definition 2: A *connection node* (CN) is a c-graph vertex. Two CNs, each connected to the other, require two inter-CN connections, one for each direction.□

Definition 3: A *p-graph* is the directed graph representing the physical network of interconnected processing nodes — the neurocomputer per se.□

Definition 4: Each *processing node* (PN) is a vertex in the p-graph. A PN represents a single neurocomputer processing node, which may host multiple CNs.□

Definition 5: *Virtualization granularity* refers to the number of CNs being emulated by a single PN.□

In a sense, virtualization granularity is the opposite of parallelism. For a connectionist model of fixed size, the greater the parallelism in an architecture, the smaller the virtualization granularity. Possible values of virtualization granularity for a particular neurocomputer range from the size of the c-graph (all CNs on one PN) to 1 (each CN is has its own PN).

Processing parallelism refers to the degree to which computation occurs in several places at once. Both uniprocessor and single DSP (digital signal processor) neurocomputer designs are representative of the low parallelism end of the spectrum. Each has a single PN hosting all CNs in the c-graph. At the massive parallelism extreme is the so-called "direct" implementation, which has an individual PN for each CN. Existing multi-DSP shared memory designs fall toward the low end of the parallelism spectrum, with a virtualization granularity near the size of the c-graph. CAP lies towards the high end of the parallelism spectrum, having a virtualization granularity in the 4 to 64 range, with 16 CNs/PN often used as a typical value. Figure 1 plots several neurocomputer architectures on a degree-of-parallelism scale calibrated in both number of CNs/PN and $\log_2$(virtualization granularity). Degree of parallelism may be affected by both the size of the c-graph to be emulated and the maximum size c-graph that can be run on an architecture. Figure 1 assumes a $2^{20} = 1{,}048{,}576$ CN c-graph is to be emulated and that each architecture listed can run such a c-graph. Note that the CAP architecture is many orders of magnitude above conventional multi-DSP designs.

CAP's choice of virtualization granularity is motivated by the intrinsic trade-off between minimizing silicon area and maximizing computational speed. Each CN consists of its state memory, including weights, input values, current activation level, and current output value. Each PN contains hardware to perform weight modification (learning), communicate with other PNs, and compute new CN activation and output values. Large virtualization granularities correspond to implementations using minimal silicon area, but are slow because there are few copies of the computational hardware. Small virtualization granularities correspond to implementations that are fast, but use lots of area because there are many copies of the computational hardware. They also use more inter-PN communication structures, and consequently more silicon area, because there are more PNs. After a preliminary analysis of virtualization granularity CAP chose 16 CNs/PN [Bai].

3.2. Communication Mechanisms

After deciding to use a low virtualization granularity, the next step is to choose the inter-PN communication mechanisms. The decision is important since the part of a silicon based neurocomputer that scales worst is the inter-PN connections. The decision is difficult because this architectural design space is largely unexplored.

3.2.1. Communication Parallelism & Multiplexing

We first examine parallelism from a top-level system design perspective. The equation

$$\textit{processing parallelism} \times \textit{processor throughput} \times \textit{average fan-out} \quad (1)$$

$$= \textit{communication parallelism} \times \textit{communication channel throughput}$$

says total messages generated as a result of processing should equal total communication throughput. Analogously,

$$node\ updates \times average\ fan\text{-}out = message\ deliveries \tag{2}$$

where a node update represents the update of a single CN's output, and average fan-out is the average number of edges leaving each CN in the c-graph. Equation (2) represents the c-graph's view of communication; the virtualization granularity is 1 CN/PN and maximum parallelism is present. In the case with less parallelism, some update messages may be destined for CNs residing in the same PN. In that case the following inequality holds

$$node\ updates \times average\ fan\text{-}out \geq message\ deliveries \tag{3}$$

where message deliveries means messages delivered over the communication systems. These equations assume enough communication channel bandwidth is present to handle the maximum message load. They provide a conceptual context for the following discussion. Although CNs are the ultimate source and destination of message, from the ABH and PBS viewpoint the PN is the communication source and destination.

Communication parallelism should be proportional to processing parallelism, but the problem is not quite so simple. Instantaneous processor throughput will, in general, not equal instantaneous communication channel capacity, either spatially or temporally. CN update activity, and therefore message generation activity, is a statistical function; it varies across both space and time.

A PN's *instantaneous load* is the number of messages its CNs generate during a single computation cycle which are destined for CNs contained in other PNs. Instantaneous load is a function of time. Assuming no PN output queues, communication resources must be able to handle both spatial and temporal maximum load. The greater the variance in the message generation rate for each individual PN, the greater the proportion, on average, of unused communication resources. Consequently, average utilization of communication resources is lowered. The resulting reduction in utilization is called the *maximum instantaneous load problem.* Unused communication resources are wasted communication resources.

A partial solution is to reduce the maximum instantaneous load by spatial averaging. Consider a region of PNs sharing a common communication structure, and the peaks and valleys in each PN's graph of the number of messages generated by that PN versus time. On average as the size of the region increases, each PN's peaks and valleys will average out against those of other PNs. The larger the region the smoother the plot of maximum instantaneous load. In effect, multiplexing superimposes the load of all CNs within the region. Such a solution assumes the message generation behavior of each CN is asynchronous and uncorrelated. While CN updates are asynchronous, the degree to which they are uncorrelated within a region is in large part dependent upon the nature of the connectionist model and the size of the region. Although potentially helpful, spatial averaging is certainly not the whole solution. Handling the maximum instantaneous load problem remains a topic for further research.

The problem is illustrated by examining two candidate communication structures, direct interconnect and a global bus, which represent design extremes.

Direct interconnect is an interconnect design where the neurocomputer has a virtualization granularity of one CN/PN. A physical wire runs between a pair of PNs if and only if there is an edge in the c-graph between some CN hosted by the first PN and a CN hosted by the second PN. Direct interconnect has high communication parallelism and zero multiplexing. Consequently, direct has high communication cost (large area used for communication) and low

communication channel utilization; it suffers from the maximum instantaneous load problem. Direct's message latency scales well, but its silicon area cost of implementation scales poorly [BaH86]. Direct scales so poorly for nonlocal c-graphs that it was rejected as a communication alternative early in the CAP design process.

Note that if the connectionist model emulated can be mapped so that each CN has only a few connections, and those to near neighbors, direct interconnect works well; the cost in area of communication resources is small and hence not significant. In effect, the maximum instantaneous load problem ceases to be a problem. The silicon retina of Carver Mead is such a c-graph [MeM88]. Direct interconnect of such models also scales well since the connections are both few and local.

A *global bus* is a communication structure where all CNs communicate over a single shared bus having sufficient bandwidth to handle a fixed portion of the maximum instantaneous message load. The fixed portion must, of course, be greater than the average load, else the message backlog and delivery latency become unbounded.

The global bus solution has relatively low communication parallelism, maximum communication channel multiplexing, and maximum communication channel utilization by virtue of performing the maximum possible spatial message load averaging. The global bus solution suffers from two problems. First, it is difficult, or impossible, to meet the bandwidth requirements with a silicon implementation. Second, the global bus fails to capture the spatial locality inherent in the connectionist models under consideration. A CN connected only to CNs sitting in adjacent PNs would send its update messages to every PN on the wafer. Such a bus is impractical because of line driving problems.

3.2.2. The Necessity of Spatial Locality

We now examine why spatial locality is essential for massively parallel VLSI neurocomputer architectures.

Consider a triple, (C,P,M), consisting of a c-graph, C, a planar p-graph, P, and a mapping, M, of the c-graph into the p-graph. For such a triple, *spatial locality* is simply the average inter-CN distance among connected CNs, where the metric used is the number of p-graph edges used to implement a CN connection. Further, it is assumed for each CN connection that the shortest p-graph path will implement that connection [4]. The greater a c-graph's spatial locality, the potentially cheaper are its inter-node connection costs in a silicon implementation.

The global bus communication structure does not take advantage of spatial locality. Whether a message is generated for delivery to a CN located in a PN next door or a PN on the other side of the wafer, the communication cost is the same.

The direct interconnect communication structure is improved by increased spatial locality, but is still too costly for large networks having large receptive fields. Given the context stated in [Ham86], Hammerstrom has shown for a triple, (C,P,M), where the p-graph is identical to the c-graph, the mapping M is optimal, and the triple has maximum locality, that total metal interconnect area, A, is related to receptive field size, r, by the equation

[4] We informally refer to the c-graph itself as having spatial locality. For a more formal treatment of spatial locality see [Bai].

$$A = O(r^3) \tag{4}$$

This result shows why direct interconnect is infeasible as a candidate communication structure for neurocomputers designed to emulate large connectionist networks. Unless the c-graph has small, local receptive fields such as those found in Mead's silicon retina [MeM88], they rapidly become unwieldy.

The CAP design is predicated on emulating models with moderately high spatial locality. Despite the architectural desirability of connectionist models with extremely high spatial locality, a certain degree of global connectivity appears necessary for networks to perform functions such as association. We call such models, for example [Lyn86], silicon cortex when they are emulated or implemented in silicon.

3.2.3. The Message Generation Characterists of Models

Little is known about either the spatial or temporal characteristics of message generation in connectionist models. The lack of papers in the connectionist literature on this topic is striking. The only paper we are aware of that addresses the issue of message generation characteristics, which is so crucial for neurocomputer design, is Robert Conwell's paper presented at the 1987 ICNN [Con87]. Both the temporal and spatial message generation characteristics of connectionist networks must be known before informed decisions can be made about massively parallel neurocomputer architectural tradeoffs.

Tolerance of the model to message latency is another important issue to be explored. Conwell found that a simple McCulloch-Pitts network, when run with fixed message delays, resulted in the network still converging [Con87]. Rudnick and Neighorn [RuN87] obtained empirical results indicating that for a grid based VBS communication scheme, even a simple Hopfield network may take an arbitrarily long time to converge.

3.2.4. Broadcast Communication

Definition 7: A *communication domain* is a local group of PNs sharing a common communication mechanism.□

Communication domains enable short-range message delivery to use cheap, short-range communication mechanisms, leaving the more expensive long-haul message delivery mechanisms to carry only long-range messages. Hierarchical, overlapping communication domains allow us to take advantage of spatial locality.

A CN's *receptive field* is the group of CNs from which it requires input. Likewise, a PN's receptive field is the group of PNs from which it requires input. A CN's *fan-out* is the group of CNs to which it provides input. Likewise, a PN's fan-out is the group of PNs to which it provides input.

When a CN changes its output state, a message is sent to all the nodes in its fan-out. A potentially efficient way to implement such duplicated fan-out communication is via broadcast. A message sent over a broadcast communication channel is sent only once. Each PN on the channel receives a copy of the message. Those PNs which host a CN having a receptive field containing the originating CN deliver a copy of the message to the "listening" CN. All other PNs on the channel will "ignore" the message.

Definition 8: A *broadcast domain* is a communication domain in which all messages sent by any PN in the domain are received by all PNs in the domain.□

When a CN generates a new message, deliveries of that message may be divided into two kinds — deliveries within the local broadcast domain and deliveries outside the local broadcast domain. All the deliveries within the local broadcast domain may be accomplished by a single broadcast of the message. For c-graphs with moderately high spatial locality and depending upon the size of the broadcast domain, such short-range messages constitute the majority of messages to be delivered [Bai]. Deliveries destined for PNs lying outside the local broadcast domain are handled by a separate point to point communication system where a copy of the message is sent to individual, explicitly specified destination CNs by a routing protocol. Using a cheap, short-haul message delivery system such as the broadcast domain for high fan-out short-range messages leads to an efficient and scalable neurocomputer communication design.

The augmented broadcast hierarchy solution provides each CN with a hierarchy of broadcast domain sizes. Such a collection of broadcast domains, where higher level domains are composed of lower level domains, is called a *broadcast domain hierarchy.* Which domain is used by each CN will depend upon the locality of that CN's fan-out in the p-graph. The more localized the fan-out, the lower the broadcast domain used. ABH allows the majority of a CN's high-locality connections to be handled by a single, efficient message broadcast mechanism. As we will see below, there are also implementation reasons for using hierarchy.

4. The Physical Broadcast Structure

In the physical broadcast structure (PBS) there is a physical, dedicated communication structure to handle the broadcast for each domain in the design. This section details a particular design implementing a single broadcast domain. For comparison purposes, unless explicitly stated otherwise, all communication structure examples used in §4 are bit serial.

4.1. An Introduction to PBS

Definition 9: A broadcast domain is denoted by D. The *size of a domain,* $|D|$, is the number of PNs it contains. A *domain at level i* of the hierarchy is denoted by H_i, for i = 1, 2, ...□

Thus, H_1 is the smallest possible domain, while H_5 denotes a fifth level domain in the hierarchy. A CAP neurocomputer with a virtualization granularity of 16 CNs/PN and a factor of four increase in domain size for each level in the hierarchy could emulate a million-node c-graph with an eight level hierarchy.

4.2. Hierarchy - An Electrical Circuit Motivation

A simple implementation could use a single bus connecting each PN in the domain. Such a domain bus communication structure has several problems. As domains become larger, the bus slows down or requires more power. Each PN has both a receiver and driver attached to the bus, which increases bus capacitance and creates a large bus load. An H_5 domain has 1K PNs. An H_8 domain has 64K PNs. An H_5 broadcast bus would have capacitance of

$$C_{bus} = C_{bus\ metal} + 1024 C_{driver} + 1024 C_{reciever}$$

The per-receiver capacitance will be approximately three times the gate capacitance for a single minimum sized FET. The per-driver capacitance will vary with the size of the driver, which in turn will vary with the bus load and the speed requirements for bus operation. Even if we are willing to pay the speed and associated power penalty, each PN will require an extremely large driver even to drive the bus slowly, resulting in a large silicon area. Finally, if

the bus is relatively long, dependent upon both the number of PNs in the broadcast domain and the size of an individual PN, there may be enough line resistance to cause an R-C driving problem. In that case, no matter how big the drivers are, the bus will be slow. Thus, as broadcast domains become large, a single bus structure is inadequate for use as the domain broadcast communication structure.

Definition 10: A *tree bus* is a tree structure in which each node consists of a local bus connecting that node and its children. Mead and Conway [MeC80] use the term hierarchical bus for such tree structures. Tree bus is used here to avoid confusion with domain hierarchies.□

Using a tree bus structure overcomes the line driving problem, but raises the question of what the interlevel fanout should be.

Definition 11: Mead and Conway [MeC80, Chapter 8.5] define α, the *branching ratio*, as a tree bus' interlevel fan-out factor.□

Definition 12: A *switch node* (SN) is a tree-node junction in the PBS dual tree communication structure (described later).□

They show that a minimum propagation delay occurs in such a communication structure for $\alpha = e = \sim 2.7$. For a cost function weighing both delay and area equally, the minimum occurs at $\alpha = e^2 = \sim 7.4$. The CAP group chose a branching ratio of $\alpha = 4$, falling between the speed and speed-area optimization points.

A branching ratio of 4 seemed natural and worked well. It turns out there are two reasons for such a choice. The first reason relates to the assumption that the silicon area devoted to inter-PN communication will be dwarfed by the silicon area devoted to the PNs themselves. Each CAP CN requires a relatively large silicon area for memory to support 1000 connections, where each connection includes both a weight and CN address. Furthermore, each PN requires a large area for CN output level calculation circuitry. Depending upon required real-time operating speed, anywhere from one copy of the calculation circuits to one copy for each CN hosted by the PN will be present in the PN. Finally, CAP uses a virtualization granularity of 16 CNs/PN, which means 16 times the memory for a single CN will be present in each PN. For these reasons, PNs will be relatively large. In a three metal process, if two layers are dedicated for inter-PN Manhattan routing, leaving one metal layer and poly for intra-PN routing, then the only part of the tree bus requiring dedicated silicon area will be the circuits implementing a tree bus nodes. For these reasons, PN area will dominate SN area (see the section on silicon area for a more detailed analysis). Therefore, trading increased SN area for increased communication speed is a good trade-off. A branching ratio of $\alpha = 4$, which is on the propagation delay optimized side of the speed-area optimization point, is an appropriate trade-off.

The second reason derives from layout considerations. To allow for a simple and hierarchically regular layout, the size of a broadcast region in each dimension, in units of PNs, must be integral. Since there are two dimensions for silicon layout, the ratio of the size of domains in adjacent levels of the hierarchy must be one of 4, 9, 16, ... Conveniently, $\alpha = 4$ is one of the choices.

4.3. Counting Switch Nodes and Processing Nodes

Throughout this section the number of PNs on a wafer, N_{PN}, is restricted to an integral power of α, the branching ratio.

Theorem 1: If the number of PNs on a wafer, N_{PN}, is an integral power of α, then all PNs can be included in a single domain, $H_{\log_\alpha(N_{PN})}$, where $\log_\alpha(N_{PN})$ is the level of the domain in the domain hierarchy. Such a domain would, of course, be of size N_{PN}. The total number of SNs in such a domain is

$$N_{SN} = 2 \times \sum_{i=1}^{\log_\alpha(N_{PN})} \left(\frac{N_{PN}}{\alpha^i} \right) \tag{5}$$

Proof: The number of SNs across the wafer on level l of each tree of a PBS dual tree structure is $\frac{N_{PN}}{\alpha^l}$. A PBS domain of size N_{PN} has $\log_\alpha(N_{PN})$ levels. Finally, each PBS consists of two trees, hence the factor of two in equation (5).□

Corollary 1: An H_i PBS may be thought of as α individual H_{i-1} PBSs plus two SNs connecting them at the i^{th} level. Hence equation (5) also serves as an upper bound of N_{SN} for single coverage of all PNs on a wafer by domains.□

Corollary 2: If the number of PNs on a wafer, N_{PN}, is an integral power of α and single coverage of the wafer is accomplished using H_h PBS dual trees, then the number of SNs across the wafer is

$$N_{SN} = 2 \times \sum_{i=1}^{\log_\alpha(N_{PN})} \left(\frac{N_{PN}}{\alpha^i} \right) - 2 \times \sum_{i=\log_\alpha(N_{PN})-h}^{\log_\alpha(N_{PN})} \left(\frac{N_{PN}}{\alpha^i} \right) \tag{6}$$

The first term in equation (6) is identical to the right hand side of equation 5 in theorem 1, and represents the number of SN in a single PBS dual tree of height $\log_\alpha(N_{PN})$. The second term represents the number of SNs in the upper levels of a height $\log_\alpha(N_{PN})$ dual tree needed to connect the height h dual trees via the technique of corollary 1 into a single $\log_\alpha(N_{PN})$ dual tree covering the entire wafer. Equation (6) may be simplified to

$$N_{SN} = 2 \times \sum_{i=1}^{h} \left(\frac{N_{PN}}{\alpha^i} \right) \tag{7}$$

□

4.4. Silicon Area

The following area numbers are based on our experience with the TBH Test Chip as described in a later section. The size of a single SN on the TBH Test Chip was 965 λ by 956 λ for λ = 1.5 μ with a 3 μ process, giving a single SN area of $2.1 \times 10^6 \mu^2$. Given a professional design and layout, the SN area figure would likely decrease by more than a factor of two. Balancing this improvement is the need for asynchronous protocol hardware and/or anti-synchronization-failure hardware which together might increase the size of an SN by a factor of four. Note that assuming an RS-232 style asynchronous protocol, each SN needs access to clocking signals in order to generate an asynchronous protocol message bit-stream. Each SN can take its clock from an adjacent PN, eliminating the space otherwise taken by clock generation circuitry. The alternative is to use an asynchronous handshake protocol which requires two additional handshake lines for each inter-SN connection.

The resulting expected SN area is $A_{SN} = 4 \times 10^6 \mu^2$, for a level 1 SN in a PBS dual tree. Scaling A_{SN} from a 3μ process to a 1.25μ process yields $A_{SN} = .69 \times 10^6 \mu^2$. A leaf node SN

contains only a single two-bit message data buffer; higher level nodes will contain larger data buffers. However, since the data buffer is a small part of the total SN circuitry and the number of higher level switch nodes decreases by a factor of α per dual tree level, the area requirements due to the increased buffer sizes are not significant.

CAP's estimate of the size of a single PN is $A_{PN} = 12.25\times10^6\mu^2$ [BaH86] based on a 1.25 μ process. The estimate includes only the memory needed for the PN (it does not include the computation and control circuitry) using a DRAM optimized process requiring $50\mu^2$ per DRAM cell. Therefore, total PN area will be greater than $12.25\times10^6\mu^2$ for a 1.25 μ process.

From equation 7 the number of SNs, given the number of PNs is an integral power of α, for single coverage of PNs by H_5 broadcast domains is

$$N_{SN} = 2 \times \sum_{i=1}^{5} \left\lceil \frac{N_{PN}}{\alpha^i} \right\rceil \Bigg|_{\substack{\alpha = 4 \\ N_{PN} = 64K}} = 43648 \tag{8}$$

Therefore, the formula for the fraction of a wafer's silicon area consumed by SNs is

$$r_{\frac{A_{SN,w}}{A_{total,w}}} = \frac{A_{SN,w}}{A_{SN,w} + A_{PN,w}} = \frac{A_{SN} \times N_{SN}}{A_{SN} \times N_{SN} + A_{PN} \times N_{PN}} = .0361 = 3.6\% \tag{9}$$

$A_{SN,w}$ is the total area of all SNs on the wafer, while A_{SN} is the area of a single SN, and N_{SN} is the number of SNs on the wafer. $A_{PN,w}$ is the total area of all SNs on the wafer, while A_{PN} is the are of a single PN, and N_{PN} is the number of PNs on the wafer. Likewise, $A_{total,w}$ is the total area of the wafer. Details of the numerical calculations performed in equations (8) and (9) are presented in Appendix II. Equation (9) says, for single coverage of PNs by broadcast domains, PN silicon area will dominate PBS communication area. Because $r_{\frac{A_{SN,w}}{A_{total,w}}}$ is so small, trading increased SN area for increased communication speed is a good trade-off. These figure are for a simple, bit serial, non-fat-tree PBS.

4.5. Pipelining the Hierarchy

Once both domain and electrical hierarchy has been decided upon, the next choice is whether to use circuit switching communication or message routing. In circuit switching communication, a message is passed by first establishing a complete circuit from originator to destination. The circuit may be always present as in a dedicated point to point line, or may be established dynamically on demand. Once the circuit is established, the message is passed straight through to the destination. With message routing, the message goes from the originator to the destination via a series of routing hops. At each hop a routing algorithm determines where to send the message next.

4.5.1. Dual Trees

Definition 13: The *dual tree* design uses two trees, one for collecting messages, the other for broadcasting messages. The collecting tree is called the *concentrate tree*. The tree broadcasting the messages is called the *broadcast tree*.□

The two trees are connected at the roots and leaves. Each leaf of the concentrate tree is the transmit port of a PN, while the corresponding leaf of the broadcast tree is the receive port of the same PN. Hence the two trees are connected at their leaves by the PNs in the

domain. The concentrate tree and broadcast tree are also connected at their roots. The root connection is where the concentrate tree delivers messages it has collected to the broadcast tree.

The path taken by any particular message is as follows. The message is originated by some CN changing its output state. The CN delivers its name (CN address), new state value, and broadcast domain level to its host PN. The host PN assembles the information (minus the domain id) into a broadcast message. The PN then queues the message for transmission to the concentrate tree. When the channel is available, the message is delivered to the bottom interior node (the PN itself is one of the concentrate tree's leaves) of the concentrate tree corresponding to the specified domain. The message then flows up the concentrate tree, one level at a time until it reaches the root. There the message is passed to the root of the broadcast tree. As the message flows down the broadcast tree, a copy is sent to each child of each node at each level of the tree.

Figure 2 is a diagram of the PN and SN connections for a H_3 dual tree with a branching factor of $\alpha = 2$ ($\alpha = 2$ was used instead of $\alpha = 4$ in order to simplify the diagram). The 8 PNs run across the middle of the diagram. The concentrate tree is above the PNs, and the broadcast tree is pictured inverted below the PNs. The root of the concentrate tree delivers messages to the root of the broadcast tree. For a branching factor of four and a tree height of l, the number of copies of the message delivered at the PN leaves constituting level zero of the broadcast tree is 4^l.

A *message stream* is a sequence of messages flowing between two connected SNs in adjacent levels of either a concentrate tree or a broadcast tree. Each SN in a concentrate tree combines its input message streams into a single output message stream according to a priority scheme (described below).

The main motivation for using a dual tree structure is noncrossing paths. Noncrossing paths provide simplicity in SN design and pipelining, which increases message throughput. The increased flexibility allows the incorporation of fat-trees as presented later.

4.5.2. Concentrate Tree Contention, Message Priority, and Contention Resolution

Definition 14: A *message time slice* is the time it takes a PN receive port to receive a complete message.□

Message contention occurs in the concentrate tree because up to α incoming message streams may vie for a SN's single outgoing channel. To resolve the problem, during each message time slice the message stream with the highest priority is passed up to the next level in the concentrate tree. The broadcast tree has no contention problem since each SN has only one input channel.

There are several options for determining which incoming message should have the highest priority. A fixed position priority scheme may be used where each incoming channel is assigned a fixed priority. As long as the highest priority channel has messages waiting, other channels must wait. If all incoming channels continually have messages pending, only the highest priority channel's messages will get through, occupying 100% of the outgoing channel's bandwidth. When the highest priority channels have no messages pending, the remaining channels contend for the outgoing channel. The channel with the highest priority among the contending channels wins and sends its message. A fixed position priority scheme guarantees the lowest possible message delivery latency for the highest priority PN within the domain at the expense of a higher, and potentially unbounded latency for lower priority PNs. Such a scheme imposes a

total order on PNs within a particular broadcast domain with respect to priority.

Another option is an equal priority scheme which amounts to round robin polling. In effect, each message stream currently flowing through an SN in the concentrate tree gets an equal slice of that SN's output channel's bandwidth.

A PN's *fair share* of message bandwidth, with respect to a particular broadcast domain, is the number of messages per unit time that the broadcast domain is capable of delivering divided by the number of PNs in the domain with pending messages. With an equal priority scheme, and because message flow is arbitrated on a level-by-level basis, if a domain is saturated (broadcasting at maximum throughput) and the message load is spatially unbalanced, a hot splotch (like a hot spot only bigger) will get less than its fair share of throughput. This is because equal priority is round robin at each SN, rather than round robin across the domain. The TBH Test Chip used an equal priority scheme.

A generalization of the equal priority scheme is the fixed bandwidth slice priority scheme. Each incoming channel gets a fixed portion of the outgoing channel's bandwidth. As an example, assume there are four incoming channels labeled c_1 through c_4 for a branching factor of α=4, with bandwidth priority ratios $c_1 = 1 / 2$, $c_2 = c_3 = c_4 = 1 / 6$. If all input channels to a particular SN always had messages pending, out of every six messages passing through the SN three would come from c_1, and one each from c_2, c_3, and c_4.

Finally, the priority assignments, or bandwidth sharings, need not remain fixed. For example, priorities can vary adaptively according to recent load, or whatever other locally computable function is desired, or even random. This is an area requiring future investigation.

Choice of contention resolution method impacts message latency characteristics. The method chosen must meet the message latency requirements of the connectionist model to be emulated. Further research is needed.

4.5.3. Balancing Propagation Delay

Definition 16: An *interlevel flit*[5], (flit for short) is the smallest portion of a message that can be passed between two connected SNs, which, of course, must be on adjacent levels of the tree. An interlevel flit is denoted by $f_{i,C}$ for a concentrate tree and $f_{i,B}$ for a broadcast tree. In each case, subscript i represents the lower of the two levels.□

Definition 17: The time for a flit of the currently active message to go from one level to the next is the *flit-time*. For both concentrate trees and broadcast trees flit-time is denoted by $t_{f,i}$, where i is the lower of the two levels involved. Flit-time consists of two parts. The first is *flit propagation delay*, or flit inter-SN transfer time, denoted by t_{fd}. *The second is flit process time*, or flit intra-SN transfer time, denoted by t_{fp}.□

Flit process time is roughly constant throughout the dual tree. Flit propagation delay varies with the height of the SN in the tree. Because of the two-dimensional nature of VLSI layout and for a branching factor of $\alpha = 4$, the length of an internode line at each level in the communication tree is about a factor of two longer than that of the next lower level and a factor of two less than that of the next higher level. Bakoglu and Meindl [BaM85] have shown that by employing repeaters, propagation delay can be held to $\Theta(L)$ where L is the length of

[5] This is an adaptation of Dally's flit terminology [Dal86]. Dally's flit is the atomic unit of information transferred between nodes in his wormhole routing scheme.

the line to be driven. Therefore, in the limit it takes about twice as long to transfer a bit from level i+1 to level i+2 as from level i to level i+1; flit propagation delay doubles at each level up the tree. In order to maintain average throughput, we use an adaptation of Leiserson's fat-tree idea [Lei85]. At each level up the tree the number of bit lines in each inter-SN communication channel is doubled, thus maintaining per bit throughput despite the doubling of flit propagation delay. In effect, flits become larger moving up a concentrate tree and smaller moving down a broadcast tree.

Definition 18: *Multiplexing ratio*, r_m, is the ratio of the number of bit lines for a flit at level i+1 to the number of bit lines for a flit at level i. A multiplexing ratio of $r_m = 2$ makes the propagation delay at each level of a PBS dual tree approximately equal.□

Definition 19: A *PBS fat-tree* consists of a concentrate or broadcast tree where the number of parallel data lines between SNs on levels l and $l+1$ is $N_l = (r_m)^l$, where r_m is the multiplexing ratio.□ Figure 3 diagrams an H_3 PBS fat-tree with a branching factor of $\alpha = 2$.□

In a pure fat-tree, the higher the level in the tree, the more parallelism there is in each interlevel interconnect. At the bottom level of a tree, the flit size is $N_0 = 1$ bit, and a flit transfers in time t_d. At the second level the flit size is $N_1 = 2$ bits, each interlevel connection consists of two lines, and each flit takes time $2t_d$ to transfer, with an average propagation delay throughput of one bit per time t_d. At level i the flit size is $N_i = r_m^i = 2^i$ bits, and each flit takes time $(r_m)^i \times t_d$ to transfer, with an average propagation delay throughput of one bit per time t_d.

For short metal lines, propagation delay increases sublinearly. It is only in the limit that propagation delay increases linearly. Therefore, the fat-tree paradigm is not invoked until interlevel lines become long. Consequently, hybrid trees are used where the bottom levels of the tree is non-fat. Figure 3 illustrates a pure fat- tree implementation, not the hybrid tree that would actually be used. Hybrid fat-trees show up later in the section on PBS performance analysis.

Using the fat-tree design means all tree levels will have the throughput of the slowest level. However, throughput would be limited by the slowest level in any case. The fat-tree approach also means there is a granularity sensitivity to the size of a message. If the height of the dual tree is h, then in a pure fat-tree design a flit of 2^{h-1} bits will be transferred at the top level every 2^{h-1} bit time units, thus naturally constraining the message size to a power of two. If messages are of size 2^i for i < h-1 and i integral, then multiple messages can be transferred together at the higher levels of the dual tree to fill out the 2^{h-1} bit top level flit. If messages are of size 2^i for $i > h-1$, then multiple top level flits will be required to transfer a single message. Finally, if messages are of size 2^{h-1}, then one message exactly fills a top level flit.

A SN at concentrate tree level i in a height h fat-tree contains two buffers, each of size 2^i bits, a receive buffer and a transmit buffer (see figure 4). The receive buffer builds a single 2^i bit flit from two 2^{i-1} bit flits received from level $i-1$. When the receive buffer is filled, the 2^i bit portion of the current message is moved to the transmit buffer where it becomes level i's flit, $f_{i,C}$. The transmit buffer holds the 2^i bit flit of the current message while it is being sent to level i+1. All concentrate tree SNs work this way, except the root node at level h, which has 2^{h-1} bit flits in both receive and transmit buffers.

The concentrate tree's root node receives a 2^{h-1} bit flit and immediately moves it to the transmit buffer. The root node's transmit buffer holds the 2^{h-1} bit flit while it is being sent to the broadcast tree's root node. Therefore, the concentrate tree and broadcast tree root nodes must be no farther distant than the distance between the level $h-1$ SN and the root node,

which is the level h SN.

The broadcast tree's SNs work in a fashion similar to the concentrate tree's SNs, with two exceptions. In the broadcast tree, messages (as flits) flow down the tree instead of up the tree. Receive buffers at level i receive 2^i bit flits from the next higher level, move the flit to the transmit buffer where it is sent to the next lower level of the tree as two consecutive 2^{i-1} bit flits. The second difference is that each SN's current flit is broadcast to all α of that SN's children. In this way, broadcast is accomplished.

It is possible to support messages of size $s \leq 2^{h-1}$ that are not an integral power of two by buffering portions of multiple messages together to fill out the higher-level flit widths. If there are no other messages available to fill out a partially filled last flit of a message, that space will remain empty. We have not explored these alternatives, but our feeling is they would require additional overhead and unduly complicate the design.

4.6. Additional PBS Parallelism

Ideally, the PBS should be able to accept broadcast messages as fast as the PN hardware generates them. A balance is needed between the message generation rate and the message delivery rate for each broadcast domain. As mentioned earlier, little work has been done on message generation characteristics for connectionist models. So, as a working reference point we assume the PBS must be able to deliver one broadcast message every 100 ns. We further assume a uniform message size of 32 bits. We call the delivery of one 32 bit message every 100 ns the standard reference design goal.

The standard reference design goal requires that one bit be delivered to all PNs in the domain every ~3.2 ns. Such rapid throughput is unrealistic for a single dual tree PBS. However, it can be achieved either by using multiple dual trees in a single PBS broadcast domain system, or by using a single dual tree modified so that each inter-level channel is replaced by wider bit-paths, resulting in a bottom-level flit consisting of more than a single bit.

For the standard reference design goal, a four-bit PBS (bottom level flit is 4 bits) will be able to handle the message throughput, thus allowing 12.9 ns for delivery of each 4-bit bottom level flit. Four separate PBSs serving a single domain will also have the same average message throughput. Compared to a four-bit PBS, four separate PBSs serving a single domain are more difficult to layout. They also incur greater control overhead and therefore require more silicon area, but degrade more gracefully with faults.

4.7. PBS Efficiency

Definition 20: Consider, in a particular PBS domain D, a single CN changing state and its fan-out. *Broadcast efficiency*, $E_{CN,D}$, is the proportion of PNs in the domain which host CNs to receive that update. Broadcast efficiency can vary from zero, where no PNs in the domain "listen" for the efferent CN's updates, to one, where all PNs in the domain "listen".□

Definition 21: *Domain coverage efficiency*, denoted E_D, is the average broadcast efficiency for all CNs in all PNs in domain D.□

The formula for domain coverage efficiency is

$$E_D = \frac{\sum_{i=1}^{N_{CN}} E_{i,D}}{N_{CN}} \tag{10}$$

where N_{CN} is the number of CNs in domain D.

What broadcast domain size is optimal? There is an implicit trade-off between average PBS network load across the wafer and point to point network load across the wafer. If a domain is too large, coverage efficiency drops and the PBS saturates. If it is too small, too many messages are sent via the point to point mechanism. In addition, what is optimal varies for each c-graph. It depends upon the average fan-out of the c-graph, the locality of the c-graph, and how good an embedding can be found. Point to point messages require destination addresses for routing. PBS messages require source addresses for use by the receiving PNs. Because only messages destined for CNs residing outside the local PBS domain must be sent via the point to point network, as larger PBS domains are used the point to point network load will decline and fewer destination-addresses may need to be maintained. The cost incurred for these benefits is decreased PBS domain broadcast efficiency.

For a CN of a particular locality, there is a trade-off between broadcast domain size and coverage efficiency. Figure 5 pictures the trade-off mentioned above for a c-graph to be emulated on a CAP neurocomputer. It shows a plot of typical coverage efficiency versus broadcast domain size. The inverted-S plot is characteristic of c-graphs having good locality and a p-graph mapping which captures that locality. The plot indicates that broadcast domains of small size have maximum coverage efficiency, E_{max}. At some some point S_1, coverage efficiency starts to decrease as broadcast domain size increases. From S_1 on, coverage efficiency decreases more or less steadily until it starts asymptotically approaching zero at S_2. At some point in the curve we move from using the broadcast domain communication system to using the point to point communication system for message delivery. The optimal broadcast domain size will lie somewhere between S_1 and S_2.

5. PBS Performance Analysis

This section presents PBS long line drive and propagation delay characteristics in terms of speed and power.

5.1. Flit Propagation Delay Simulation

Interlevel propagation delay, t_{lp}, determines how long a line can be used before repeaters are needed. Several SPICE II runs were made to characterize interlevel propagation delay. In all runs both a SPICE level two MOS transistor model and process parameters for a MOSIS nonscalable 3 micron process were used. The transistor model and process parameters are those used for the TBH Test Chip.

A second order Π R-C model was used to approximate a long R-C line and is shown in figure 9. A first order Π model lumps all resistance into a single line resistor with all capacitance divided evenly between two line capacitors, one before and one after the resistor. A second order Π model evenly divides resistance into two serial line resistors with half the line capacitance placed between the two resistors and half of the remaining capacitance placed before and after the two resistors. A second order Π model for long R-C lines is more accurate than a first order model.

For driver sizes of 1X, 10X, and 100X, figure 6 plots flit propagation delay, t_{fd}, versus tree transfer level. Each driver is a single inverter. Each load consists of the Π R-C model with a minimum-sized inverter as the receiver in an SN. t_{fd} is measured as the time from when the input voltage crosses the midpoint of the full-swing voltage range to when the output voltage crosses the midpoint of its range. The input voltage is defined as a piecewise linear voltage source switching in two nanoseconds. Being a voltage source, its transition is independent of the size of the driver (and therefore the load) it is driving. The input midpoint transition is used to initiate measuring, since driving the driver is not properly included in propagation delay. Interlevel connections are run in metal two using the parameters listed in Table 1. Figure 9 shows the test circuit used in these simulations and the equations used to derive power dissipation.

A one Farad capacitor in parallel with a 100 megohm resistor is placed between the Vdd voltage source and all Vdd circuit terminals (the driver and receiver p transistors). The change in voltage across the capacitor is proportional to the integral of the current that flows through the capacitor over the time interval of interest, t_{fd}. The 100 megohm resistor is big enough compared to the capacitor so that effectively all current delivered flows through the capacitor. From the final charge on the capacitor, the total power consumed by the test circuit and shown in the power graphs is calculated. Worst case power, figure 7, incorporates the following assumptions:

- each line and driver combination run at the maximum speed possible for that combination — 100% duty cycle
- a worst-case bit-stream is transmitted (alternating 0's and 1's)

Figure 7 shows power dissipation versus tree level for transmitting a worst-case bit stream. Low level lines in the PBS dual trees have the shortest delay times; they send the most bits per unit time. As lines become longer, R-C effects tend to dominate, effectively limiting throughput. As a result, and despite the use of large drivers, power dissipation peaks for transfers between levels two and three. Long lines have more capacitance, and much slower throughput. Figure 7 is useful for calculating worst-case power for a full fat-tree configuration running at 100% duty cycle.

Figure 8 shows the worst-case power dissipation due to line driving for sending a single bit at each level. Figure 8 can be used to calculate overall power dissipation in cases where some or all of the inter-level lines are running at a duty cycle of less than 100% by simply scaling power dissipation by duty cycle.

It is important to keep in mind that Figures 7 and 8 represent only the driver and line

parameter	value	units
PN size	3350	microns
metal2 R	.03	Ω/□
metal2 C	$.14\times10^{-4}$	$picofarads/\mu^2$

Table 1: SPICE parameters for power and propagation delay simulation.

components of PBS dual tree power dissipation.[6]

5.2. Simulation Interpretation

The notation $L_{j,i}$, denoting the level i to level j interconnect, will be used throughout this section. The first subscript always denotes the *to* level, while the second subscript denotes the *from* level. If $i < j$, then $L_{j,i}$ indicates a transition in a concentrate tree. If $i>j$, then $L_{j,i}$ indicates a broadcast tree.

5.2.1. Speed

Figure 6 shows that the 10X and 100X propagation delay curves are similar while the 1X driver curve is considerably slower. The differences in the 10X and 100X driver time curves do not become significant until extremely long line lengths are reached and R-C characteristics dominate. Thus, even replacing the line driver with a voltage source would not show a major improvement for longer line lengths.

Equation (9) shows an increase in the size of PBS dual tree line drivers has little impact on the total area needed for a CAP neurocomputer. Therefore, a speed-area tradeoff in favor of higher speed PBS drivers is appropriate. Figure 6 shows little can be gained from driver sizes greater than about 10X. Assuming 12 ns for propagation delay driving a $L_{5,6}$ PBS line with a 10X driver, and adding another 2 ns for the first stage of a two-stage driver gives a total send-receive delay of 14 ns. Such a inter-level transfer could occur during one phase of a standard two phase CMOS non-overlapping clock[7], and corresponds to roughly a 35 MHz clock rate. For a symmetric, two phase clock there will be another 14 ns during phase two for intra-SN transfers. Therefore, propagation delay is not a problem when using 10X drivers until we get to level H_6, where fat-trees must be used to maintain throughput. It is interesting to note that the $L_{5,6}$ line is approximately the knee of the 10X plot in Figure 6.

Alternatively, a 1X (minimum sized) driver could be used with fat-tree throughput maintenance and 1X repeaters employed above $L_{2,3}$, maintaining the same throughput as the 10X design considered above. Multi-bit flits would begin at $L_{2,3}$ rather than at $L_{5,6}$ as with 10X drivers. Since flit size, and therefore the number of drivers per channel, increases exponentially with level, it is not clear whether 1X drivers and repeaters use more or less area than 10X. As will be seen in the power analysis, the 1X design uses less power. The drawback is more complexity in the design and layout because fat-tree channels are invoked at a lower level in the dual trees.

5.2.2. Power

Only the broadcast portion of a dual tree contributes significantly to power dissipation; for each flit cycle, and assuming broadcast domain saturation, every line in the broadcast tree is active. In the concentrate tree, even if there are flits pending on every line, only one path up

---

[6] Our goal with the PBS design is to prove feasibility. Since the propagation power component of total power dissipation was the least know at the start of the PBS design process, we calculated only the power needed to drive the long, inter-SN lines.

[7] In the CAP design each PN has its own independent local clock, which is running asynchronous to all the others, but at about the same rate. Both this and the potentially large inter-SN distances to be traversed implies that inter-SN transfers must be asynchronous. For the PBS architectural analysis, the asynchronous nature of the inter-PN transfer is ignored in order to focus on higher-level architectural issues. For a real PBS implementation, some sort of asynchronous protocol is needed for each set of flit lines and will increase the SN size. Whether sufficient allowance was made in the SN to PN size comparisons of § 4.4 remains to be determined.

the tree, the path corresponding to the current message being sent, actually has bits flowing on it at any particular time. This is actually an over-simplification since newly generated messages having priority less than the highest priority message will move up the concentrate tree until blocked.

The following power calculations assume messages are, on average, composed of roughly equal portions of randomly distributed 0's and 1's. Such a message model produces a power figure between best-case and worst-case. Since most of the power in CMOS circuits goes to charge/discharge capacitance, a worst-case bit stream is composed of alternating 0's and 1's. A best-case bit stream is composed of all 0's or all 1's and dissipates no power for line driving. The inter-level communication model used here is a single line dissipating power, using a single line asynchronous communication protocol.

5.3. Propagation Power for H_i

Table 2 shows power dissipation figures for PBS communication structures for varying domain sizes. Detailed calculations are presented in Appendix II. The assumptions made are presented below.

As one moves up the broadcast tree, the power consumed at each level decreases until the fat-tree throughput maintenance level is encountered. Repeater usage also starts at the same level as the fat-tree mechanism. From that point on, power consumption per level remains constant. There is a factor of four reduction in the number of logical connections at each level up the tree, but repeaters and the fat-tree mechanism each add a factor of two per level to power consumption, resulting in a net change in power consumption per level of zero.

For each domain size, propagation delay power is calculated for both 10X and 1X drivers. Figure 8 provides the numbers needed. All designs use the following parameters:

- 35 MHz clock (14 ns $L_{0,1}$ flit propagation delay period)
- messages composed of random 0's and 1's
- MOSIS 3 μ non-scalable process technology

The 1X design has the following design parameters:

- 1X sized inter-level line drivers
- repeaters at $L_{3,4}$ and higher
- PBS fat-tree throughput maintenance at $L_{3,4}$ and higher

The 10X design uses the following design parameters:

- 10X inter-level line drivers
- repeaters at $L_{5,6}$ and higher
- PBS fat-tree throughput maintenance at $L_{5,6}$ and higher

5.4. Propagation Power for Wafers

The 1X design gives a total wafer power dissipation due to PBS propagation delay for dual coverage by H_5 broadcast domains using a 3 μ process technology and containing 64K PNs, of 13.7 Watts (.107 Watts/H_5 $\times$ 64 H_5/single coverage $\times$ 2 single coverage/dual coverage); the 10X design dissipates 25.9 Watts (.202 Watts/H_5 $\times$ 64 H_5/single coverage $\times$ 2 single coverage/dual coverage). These figures represent only the power consumed to drive the inter-level lines. The intra-SN logic power dissipation, which is the other component of total PBS

Propagation Power		
Broadcast Domain	Driver Size	
	1X	10X
H_1	0.228 mW	0.525 mW
H_2	1.26 mW	2.77 mW
H_3	5.6 mW	12.1 mW
H_4	24.6 mW	49.9 mW
H_5	107 mW	202 mW
H_6	464 mW	819 mW
H_7	2,000 mW	3,320 mW
H_8	8,600 mW	13,500 mW

Table 2: Propagation power for PBS broadcast domains using 3μ MOSIS technology.

power dissipation, has not been estimated, but we expect it to be manageable since the SN logic is relatively standard.

These figures are artificial since it is impossible to get even a small fraction of 64K PNs on a 3 μ process technology wafer. They represent the propagation power that *would* be dissipated were it possible to fabricate such a large wafer. Consequently, power density is a more useful figure than total power. For the 3 μ technology, propagation power density is 1×10^{-11} Watts/μ^2 for 1X drivers and 1.6×10^{-11} Watts/μ^2 for 10X drivers.

Using $1/\alpha^2$ as the factor by which dynamic power dissipation scales as the process technology is scaled down by a factor of α [WeE85], propagation power was figured for two additional wafers. Note that a dynamic power scaling factor of $1/\alpha^2$ is for ideal scaling. Ideal scaling assumes supply voltage, V_{DD}, scales as $1/\alpha$; such a large decrease in V_{DD} is unlikely. Power density remains constant for ideal scaling, thus the propagation power density figures presented above for propagation power apply to all technologies considered here.

A 6" wafer using a 1.25 μ process technology can implement a square array of 1024 PNs. Such a wafer may be covered by two sets of four H_4 broadcast domains arranged so as to overlap. For 1X drivers, the total PBS propagation power consumed by such a wafer is 34.2 mW (see Appendix II for details). This figure is not directly comparable to the corresponding 3 μ technology's figure of 13.7 W because the size of the p-graphs are different.

An 8" wafer using a .1 μ process technology (near the limit of process technology improvement) can implement 399,360 PNs, each with a virtualization granularity of 16 CNs/PN. Using dual overlapping H_5s for broadcast domains, the wafer would contain 780 H_5 domains. For 1X drivers, total propagation power consumed by such a wafer would be $\sim$ 93 mW (again, see Appendix II for details).

6. Fault Tolerance

Fault tolerance is of considerable importance to any wafer scale integration level architecture, since faults are unavoidable in silicon process technologies. In conventional CMOS

VLSI, die yield decreases as chip size increases — the larger the chip fabricated, the smaller the yield.

6.1. Faults & PBS

A *damaged region* is the portion of a PBS dual tree impaired due to a process fault. In general, each CAP wafer will have a number of damaged regions. Damaged regions may be isolated from each other or overlapped.

There are two types of PBS dual tree faults. If the fault occurs in a concentrate tree, the portion of the broadcast domain prevented from broadcasting is the damaged region. If the fault occurs in a broadcast tree, the portion of the broadcast domain prevented from receiving broadcast messages is the damaged region. In general, the PNs (and CNs) in the damaged region may be completely functional except for being isolated from the network because of the PBS fault.

Faults occurring in the PBS communication structures are generally more costly than faults occurring in the PNs. For example, a fault disabling a bottom level (level L_1) concentrate tree SN may prevent the α PNs using that SN from broadcasting messages, and hence effectively prevent the CNs they host from participating in network computations. If there is multiple broadcast domain coverage of those PNs (e.g., dual coverage), the affected region of the network might still be able to participate in network computations, but possibly in an aberrant fashion.

The usefulness of contributions from the resulting isolated PNs would be doubtful and could even be a hindrance to network function. Similarly, such PNs may continue to communicate via the point to point network. To prevent unwanted interference from PNs in damaged regions, an autism function should be included in all PNs. If a PN detects that it is in a damaged region, it should disconnect itself from the network. Further research is needed.

A worst-case PBS fault is damage to one of a dual tree's root SNs, or their interconnection. Such a fault would result in the entire domain, including all of its PNs and the CNs they host, becoming a damaged region — a high cost if the domain happens to be an H_5 with 1K PNs and 16K CNs. Such a worst-case fault is equivalent to a fault occurring in a critical portion of each of the 1K PNs in the broadcast domain.

6.2. PBS Fault Amelioration

There are at least two ways to reduce the impact of fabrication faults on the PBS communication structures by adding redundancy. First, more conservative design rules may be used for SNs and the metal runs connecting them. Second, redundancy may be added to the PBS by duplicating components.

6.2.1. Conservative Design Rules for PBS

More conservative design rules reduce the probability that any particular fabrication defect will result in a fault. The tradeoff is increased silicon area for PBS and a slight decrease in performance. The performance decrease is due to increased capacitance, especially for the longer inter-SN lines. Since SN area is a small proportion of total wafer area (see equation (9)), substantial relaxation of design rules may be accommodated with only a modest increase in total silicon area.

No attempt to characterize the expected reduction in faults due to relaxation of design rules for SNs and inter-SN connections has been done. FaultSim, a tool for simulating faults

on a wafer scale CAP neurocomputer has been developed, and may be used to characterized the impact of such faults. Refer to Norman May's thesis for further information [May88].

6.2.2. Redundancy for PBS

The second method to reduce the high cost of PBS fabrication faults is to add redundancy through duplication of components. Either fine grain intra-SN redundancy may be added at the device and gate level, or coarse grain redundancy may be added at the SN and inter-SN connection level. We briefly discuss the latter option.

Figure 10 shows an H_3 concentrate tree with redundant SNs at levels L_2 and L_3. For each pair of SNs serving an identical function, one is labeled with a prime. For example, SN $C'_{2,0}$ and $C_{2,0}$ are two such SNs. For each such pair of SNs, one is redundant.

Coarse grain redundancy works as follows. At system configuration time a global initialization mode is entered. All SNs to receive redundant inputs mark all such input ports as NOT OK. Then a known test pattern is sent from each PN. Performing such an action requires global and possibly external coordination and control. A SN at level L_i which sees the expected pattern on a redundant port marks the corresponding port as OK. A SN that never sees the expected pattern fails to mark the corresponding input port as OK. This process repeats for each redundant path into receiving SNs. When initialization mode is terminated, if there are multiple OK ports, one of them is chosen for use. If there is only passes, it is used. If none pass, that part of the system is a damaged region.

Coarse grain redundancy adds complexity to the design of each redundant SN and requires a global configuration process. However, the increased design complexity cost for all redundant SNs is paid only once — repeated use of the design incurs few additional design costs. Silicon area and power dissipation costs are related to how often redundant SNs are used. We expect redundant SNs will only be used in the upper portions of PBS dual trees, where the cost of a lost SN is greatest, and consequently the associated area cost will be acceptable.

7. Layout Considerations

Strict adherence to a regular, hierarchical layout structure will allow WSI (Wafer Scale Integration) layout using conventional, manual, hierarchical layout tools such as Magic [Ous]. Such optimism results from the fact that all PBS broadcast domains employ a regular hierarchical structure.

All PBS communication structures can be constructed using a recursive, bottom-up approach. The following methodology makes several assumptions. First, a standard PN cell exists. Second, metal 2 and metal 3 are reserved for PBS. Manhattan routing allows maximum regularity using a minimum number of dedicated metal levels. Finally, mask steppers capable of placing a unique mask section at each step position across a wafer with better than one λ registration accuracy already exist [Col87].

The following methodology assumes Magic as the layout tool; other layout tools could be used. First an H_1 concentrate tree is manually laid out, complete with connections to PN transmit ports. Note that the SN-PN metal runs must be a part of the H_1 concentrate tree cell. Next, to layout a H_2 concentrate tree four H_1 concentrate tree cells are arrayed using the Magic array command. The four are connected by manually adding the level L_2 SN and the metal runs to connect the new SN (the root of the H_2 concentrate tree) to the $\alpha = 4$ H_1 subtree roots. Thus, the H_2 concentrate tree standard cell is completed. H_3, H_4, and H_5 concentrate

tree standard cells can likewise be recursively constructed. PBS broadcast tree cells can be constructed just as easily. To complete each H_i, $i = 1, 2, \ldots$. PBS dual tree, simply instantiate an H_i concentrate tree, an H_i broadcast tree, and connect their roots. For each dual tree the only manual layout was placement of the root SN of each concentrate and broadcast tree, the metal lines connecting each root to its $\alpha = 4$ children, and the metal lines connecting the two roots.

Once standard cells for each H_i dual tree have been built, instantiate an array of the desired size of standard PNs. Then instantiate a H_i dual tree wherever a broadcast domain is desired. For regular broadcast domain structures, such as single coverage of a wafer by H_i dual trees, three Magic array commands will suffice — one to array the PNs, and one for each of the H_i dual tree arrays.

As long as the layout is regular and standard across the wafer, a hierarchical layout can be performed using the Magic array command. Such an approach simplifies layout to the point where, even for a complete wafer, the PBS layout can be done by hand. However, a silicon compiler style hierarchical layout tool may be able to overcome the wasted space problem described below.

The following problems may be encountered. First, Magic may not work correctly for such large structures. Such a problem, while troublesome, is not a flaw in the methodology presented above, only a limitation of Magic. Second, the existence of fault amelioration techniques may complicate the above methodology. However, for a single, regular dual tree design, the PBS fault amelioration techniques discussed in § 6 add only minor complications. Third, some areas forbidden to PBS routing may be required for power, ground, and point to point cross-overs. Such forbidden areas must be avoided at each level of the PBS hierarchical layout, but should present no significant problem provided the forbidden areas are regular, that is, constitute a uniform and repeating pattern. Finally, the layout scheme as described does not easily provide for overlapped domain coverage.

A simple application of the layout procedure described above suffers from a wasted space problem. Every SN throughout the dual tree must occupy occupy some space. To accommodate the SNs, space in the layout of the standard PN cell must be reserved. The reserved space occurs in all PNs whether or not SNs will actually occupy the space. For example with an $\alpha = 4$ branching factor, two of every four reserved spaces will be occupied by level L_1 SNs in the concentrate and broadcast trees. An additional two of every $\alpha^2 = 16$ reserved spaces will be occupied by level L_2 SNs, and so on up the dual tree. The net effect is that some of the reserved space is wasted. The reserved space requirement also complicates the standard cell layouts for the H_is, since only one SN can occupy any particular reserved space.

8. The TBH Test Chip

As one of the class projects for the 1986-87 Oregon Graduate Center's Advanced VLSI Design class, a PBS demonstration chip was designed, fabricated using MOSIS scalable 3 μ processes technology, and tested. The chip is the PBS (or TBH) Test Chip.

In order to simplify the design, only the concentrate tree portion of an H_3 PBS dual tree structure was implemented. As further simplifications, a branching ratio of $\alpha = 2$ was used rather than the $\alpha = 4$ branching ratio anticipated for wafer implementations, and no fat-tree throughput enhancement was included. A total of seven concentrate tree SNs were implemented — four for the bottom interior level of the concentrate tree (level L_1), two for level L_2, and one for level L_3 (the root of the concentrate tree). To support this H_3 concentrate tree,

two sets of registers were implemented to emulate PNs with a virtualization granularity of 1 CN/PN. The "transmit PNs" consisted of a set of eight seven-bit register — three bits for a one-of-eight-PN destination address and four bits of data. The "receive PNs" consisted of eight four-bit registers and a three bit address decoder. Each message consisted of seven bits — a three bit destination address (go to addressing), and four bits of destination data.

Each SN contained two one-bit buffers, one for receiving bits from one of the SN's two children, and one for holding the bit being sent to the SN's parent. The concentrate tree had a sustainable throughput of one bit per clock cycle.

The chip worked at 5 MHz. Figure 11 is a photocopy of the TBH Test Chip layout. Roughly, the upper right quarter or the layout is the transmit PNs registers, the lower half is the concentrate tree, and the upper left quarter is the receive PN registers. In the concentrate tree, the upper complete row is the L_1 bottom level SNs, the bottom left and right corners are the L_2 second level SNs, and the bottom middle is the concentrate tree root (level L_3). It is clear from figure 11 that the SN layout is rather sparse compared with either the transmit or receive registers. The figures presented in § 4.4 on SN area and the total SN area to total wafer area ratio, $r_{\frac{A_{SN,w}}{A_{w,total}}}$, equation (9), were derived from the TBH Test Chip's SN layout area.

A simple fixed-by-position bandwidth sharing message priority scheme was employed where the higher numbered child always had first call for message delivery, but the lower numbered child could never be locked out for more than one message time.

Upon testing the chip, the TBH Test Chip team discovered a transient timing design flaw that interfered with the handshaking between the concentrate tree root and the receive PNs. By laser fusing a wire we were able to "fix" the design flaw by locking high the TAKEN handshake signal coming from the receive PNs to the concentrate tree root. After the microsurgery, the chip performed as expected.

See the microarchitecture design specification in Appendix I for more chip design details.

With respect to PBS layout issues, the inter-SN (between switch node) wiring layout was not as difficult as anticipated. Although we implemented an H_3 non-fat concentrate tree with a branching factor of $\alpha = 2$, it would have taken little additional time to implement a branching factor of $\alpha = 4$, the standard PBS branching factor.

9. Future Areas of Research

Many areas need further investigation. For example, what kinds and magnitudes of message latency result from different architectural tradeoffs such as choice of concentrate tree priority scheme, and how tolerant are different connectionist models to such latencies. A related need is the development of message generation models which both spatially and temporally characterize various connectionist networks. A partitioning of connectionist models by sensitivity to kind and distribution of message latency will result, and ultimately determine which connectionist models are most appropriate for emulation by which architectures. These results are vital not only for CAP style neurocomputers, but to all electronic, highly parallel neurocomputer designs.

Another important research area is the scaling problem. Currently, we know of no connectionist models that use even a tenth of a single 10^6 CN CAP neurocomputer. How can the size of connectionist networks be increased to successfully tackle more difficult problems such as speech recognition? Again, such research is not particular to the CAP neurocomputer

design; it is necessary in order to design very large connectionist models, of a size requiring the capabilities of the CAP architecture.

Another point needing further investigation is the PBS network versus VBS network tradeoffs. The figures to be compared are silicon area, power consumption, and message throughput. Perhaps most important is a comparison of the kind and magnitude of delay associated with each type of message delivery network.

As mentioned in § 6 and § 7, work needs to be done on PBS fault tolerance characteristics, fault amelioration techniques, and layout automation techniques and organization.

Provided high temperature superconductivity current densities are high enough, reducing the resistive component of the metal line RC characteristic to zero changes driving a long line to simply charging a capacitor in series with a small inductor. The result would be to increase PBS speed, and probably increase power dissipation too. The availability of practical high temperature superconductivity could also be helpful in electrically connecting multiple CAP wafers into a single mega-system.

Optical interconnect between wafers may allow wafer stacks containing many wafers to function as a entity. Employing such a technique, one can imagine tens and possibly hundreds of millions of CNs functioning as a single entity.

10. Conclusion

Connectivity is a significant problem facing any electrical neurocomputer design. The CAP silicon cortex approach solves the problem by requiring locality in the connectionist model to be emulated, by multiplexing communication, and by providing two separate mechanisms for message delivery: a point-to-point network optimized for delivery of long haul, individual messages, and the physical broadcast structure (PBS) optimized for delivery of high fanout, local messages.

The PBS efficiently emulates the dense, local interconnections characteristic of high-locality connectionist networks by use of a pipelined, domain oriented broadcast mechanism. Each broadcast domain is implemented by a pair of communication trees, a concentrate tree to collect messages and a broadcast tree to deliver them. The dual tree design is very flexible; it may be tailored to the fault tolerance, domain size, and bandwidth required by a wide range of possible CAP designs.

Our research has determined that the PBS approach to local interconnect emulation is feasible. The PBS has good speed, power dissipation, and silicon area characteristics. Further, it is appropriate to CAP's augmented broadcast hierarchy design.

This report documents some early steps in the achievement of CAP's long term goal of creating an inexpensive neurocomputer capable of emulating 10^6 CNs with 10^9 connections. Some problems in the design of WSI neurocomputers have been solved and others have been identified, hopefully to be solved by future research. Many steps and much research and engineering remain to be done.

Appendix I

Specification of the TBH Test Chip

Micro-Study for the
TBH Test Chip
August 1987

Sudarshan Cadambi, Charles Hong, Mike Rudnick
Class: CS&E 529
Professor: Dan Hammerstrom
Oregon Graduate Center

1. Introduction

The TBH (The Broadcast Hierarchy) chip implements a three level, 16 PN (Processing Node) communication structure. For each PN, only either the transmit or receive part of the PN is implemented. PNs are used to implement highly parallel VLCN (Very Large Connection Network) architectures.

The TBH test chip implements part of a broadcast domain communication structure. A broadcast domain is the group of PNs (and associated communication circuitry) interconnected via an instance of a broadcast communication structure. The higher the domain is in the hierarchy of broadcast domains, the larger the group of PNs connected by the broadcast domain.

A broadcast domain may contain a concentrate tree and/or a broadcast tree, or neither, depending upon the implementation. A concentrate tree is a tree with one-way communication paths running up the tree. The leaves of a concentrate tree are PNs. A broadcast tree is a tree with one-way communication paths running down the tree. Again, the leaves of the tree are PNs. A full concentrate-broadcast tree consists of one concentrate tree, one receive tree, and a communication structure connecting their roots. On the TBH test chip there is a concentrate tree, but no broadcast tree, only a global broadcast line.

There are 16 PNs on the TBH test chip. They are divided into 8 transmit PNs and 8 receive PNs. For the transmit PNs, only the transmit portion of each PN is implemented. Likewise a receive PN implements only the receive portion of a PN. The 8 transmit PNs are organized as a transmit buffer register file. Likewise the 8 receive PNs are organized as a receive buffer register file.

2. Functionality

2.1. PN Connections and Addresses

There are 8 unique PN addresses, 0 through 7. For any particular address, exactly one transmit PN and one receive PN will have that address. Each message sent consists of a 3-bit address followed by a 4-bit value. Whenever external data is loaded into a transmit PN, the low order (rightmost) three bits are the destination (receive) PN address and the high order (leftmost) four bits are the value. Each message is transmitted via the broadcast hierarchy communication structure (in this case the concentrate tree and global broadcast line) to the indicated receive PN.

The broadcast hierarchy communication structure transmits only one message at a time. Since multiple messages may be awaiting transfer, the communication structure determines which message has highest priority and completes transmission of that message first.

The TBH Test Chip uses a bandwidth slice prioritization scheme. Each switch, after having received the last bit of the last message, makes its priority choice according to the following scheme: n messages are taken from the higher address child followed by m messages from the lower address child. If a message is not available from the selected child, then message(s) from the other child are accepted.

For the TBH Test Chip, the n and m of the bandwidth slice prioritization scheme are both 1. This means when all the transmit PNs continuously have messages to transmit (ie, the communication structure is maximally loaded), each switch node will accept one higher priority (from the higher address

child) message for every lower priority (from the lower address child) message.

Once a node accepts a message it will complete the transfer of that message before accepting a new message. In general, once a message has completed transmission, there will be a group of messages waiting to be transmitted (ie, pending). The next message to be transmitted will depend upon both the current state of the concentrate tree (ie, which part of its n high then m low cycle each switch node is in), the sending address of the messages waiting to be transmitted, and when each message first requested transmission relative to when the other pending messages first requested transmission.

2.2. Concentrate Tree Switch Addresses

For monitoring purposes the concentrate tree switches are assigned addresses. The switch addresses run from 0 through 6. Switch addresses 0 through 3 correspond to the first level of concentrate tree switches with 0 being the switch connected to the two lowest address transmit PNs and 3 being the switch connected to the two highest address transmit PNs. The second level of switches correspond to switch addresses 4 and 5, while the single switch in the last (third) level corresponds to switch address 6.

3. External Interface

There are three parts to the external interface - loading the transmit buffers, reading the receive buffers, and monitoring the transmit hierarchy tree nodes and global receive line. Unless explicitly stated otherwise, all external lines are to be valid at the rising edge of phase 1. Please refer to figure 1.

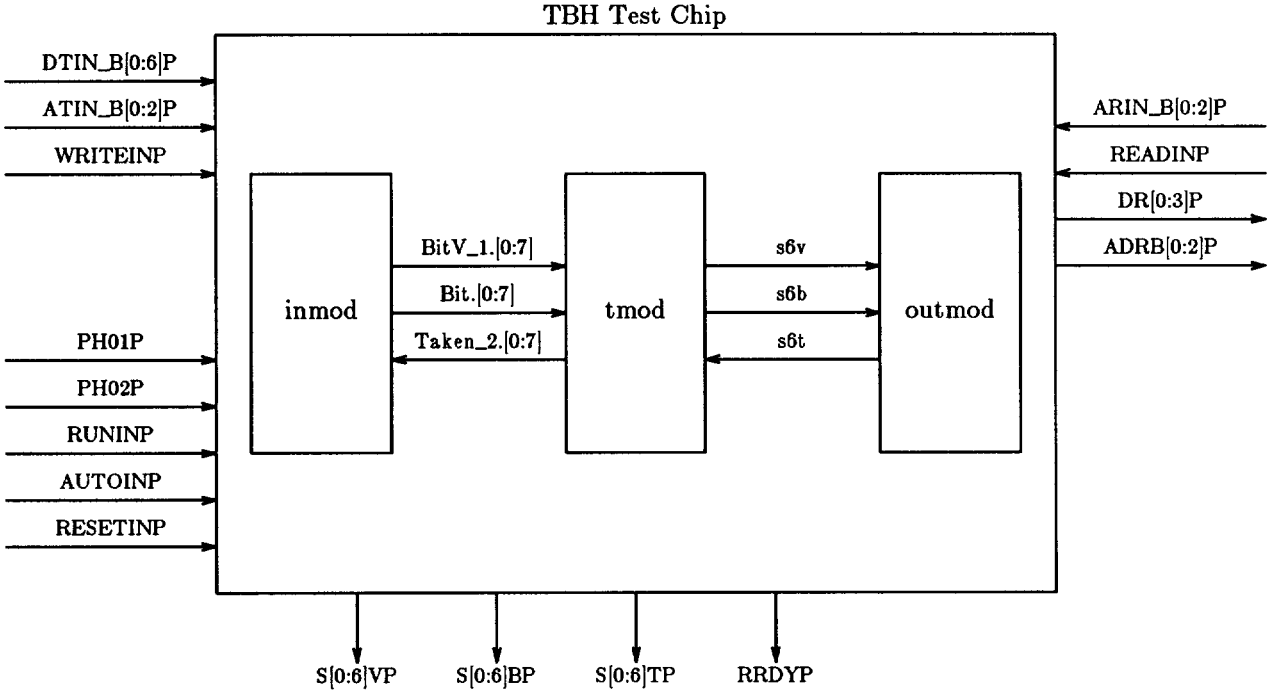


Figure 1: TBH Test Chip Overview ([0:6] is replaced by a digit 0 through 6)

The operating sequence for the TBH Test Chip is:

1) reset the chip
2) load the transmit PNs
3) run the chip, monitoring concentrator tree traffic and received addresses
4) read out received values from the receive PNs

Appendix I

3.1. Transmit PN Buffer Input

DTIN_B <6:0> - PN data

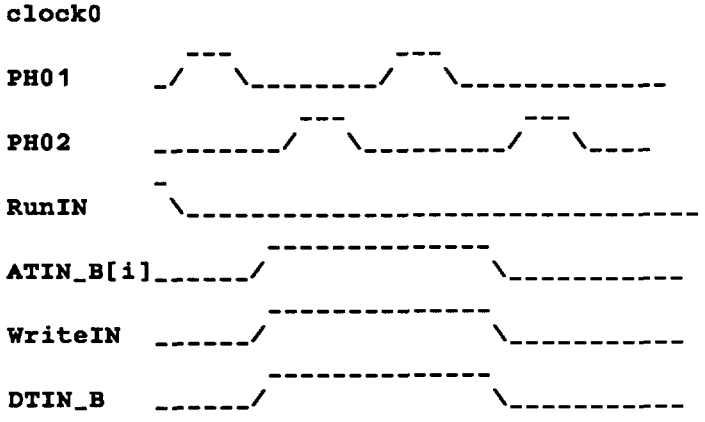
DTIN_B <6:3> - "value"
DTIN_B <2:0> - "address"

ATIN_B <2:0> - transmit PN address

WriteIN - write-enable control signal for writing data to transmit buffer

DTIN_B <6:0> are the input pins for a message. The 3 low order lines (DTIN_B <2:0> are for inputing the receive PN address and the 4 high order lines (DTIN_B <6:3>) are for inputing the value.

ATIN_B <2:0> are the input pins for the addresses of the 8 transmit PN registers.



3.2. Receive PN Buffer Output

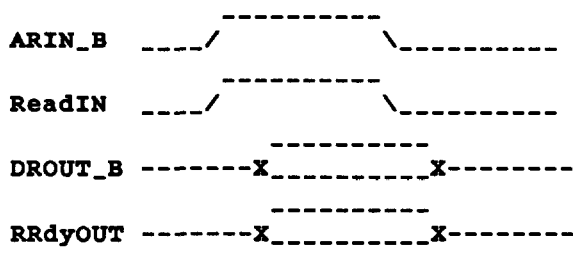
DROUT_B <3:0> - receive PN's data

ARIN_B <2:0> - receive PN's address

RRdyOUT - receive buffer ready
ReadIN - read receive buffer data request

DROUT_B <3:0> are the output pins for reading message from a receiving PN's Register. The address of the receiving PN's Register is specified by the 3 address pins - ARIN_B <2:0>.

RRdyOUT signals to the outside that the receiving Register which is addressed by ARIN_B is ready for being read.



3.3. Debug, Monitor, and Control

ADRB2:0 - received message address
RUNINP - run TBH Test Chip (ie, send pending messages)
AUTOINP - run TBH Test Chip continuously on same transmit PN data
RESETINP - reset TBH Test Chip
PH01 - phase one clock
PH02 - phase two clock

for x = 0, 1, ..., 6:

SxVP - switch data valid
SxBP - switch data (bit)
SxTP - switch data taken

ADRB2:0 is the 3 bit address shift register of outMOD. The value on the pins is the current contents of the shift register and can be monitored realtime.

RUNINP high causes all valid messages in the transmit PNs to contend for transmission. RUNINP must be high by the leading edge of phase 1 for the TBH Test Chip to be active (run) during that clock cycle.

AUTOINP asserted causes the valid data in the transmit PNs to be sent repeatedly. That is to say that once a message has been sent, that same message is immediately requeued for transmission. This mode allows the concentrate tree prioritization scheme to be tested.

RESETINP high causes any messages currently being transmitted to abort and resets all logic to the initial state with the exception that the data contained in the transmit and receive PNs is set to zero. RESETINP takes precedence over all other signals. RESETINP should be asserted for one full clock cycle (or longer).

The highest level concentrate tree switch (Switch6) uses RUNINP to start and stop the flow of pending messages. This means that at any particular time when message flow is in a stopped state (ie, RUNINP low, and the current bit completed), several pending transmit PN messages may be in various states of working their way up the concentrate tree to become the current message.

PH01 and PH02 are phase one and phase two of a non-overlapped 12 MHz (84 ns) clock.

SxVP, SxBP, SxTP (x = 0, 1, ..., 6) are concentrate tree monitor lines. Each line shows the current (realtime) state of the valid (V), data (bit, or B), or taken (T) switch interconnect lines for one of the concentrator tree switches.

Appendix I

4. Microsimulation Modules

4.1. Overview

There are three modules in the TBH Test Chip microsimulator: a transmit PN register module, an interconnect structure module, and a receive PN register module.

The transmit PNs (inmod) serve to generate messages which contend to be transmitted via the concentrate tree and global receive line (tmod) to be received by the receive PNs (outmod).

The lines between the transmit PN register module and the tmod module, and between the tmod module and the receive PN module, plus the associated timing diagrams and interface protocol are described in the tmod section later in this document.

4.2. Transmit PN Buffer -- inMOD

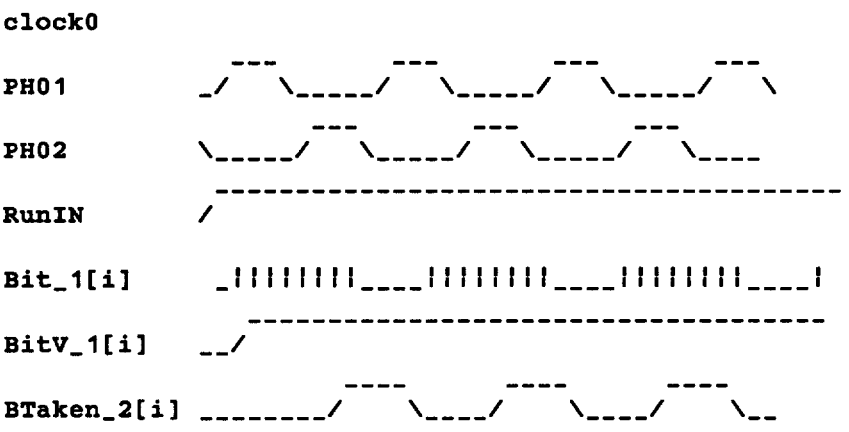
4.2.1. inMOD interface

The inMOD module interfaces internal to the tMOD module for shifting bit-data to the lowest level switches and external to the outside pins for writing data into the registers.

The inMOD module interfaces externally with the outside for inputting message to the registers. There are 7 data lines and 3 address lines. A write-enable (WriteIN) line control/enable the address decoder. The input lines and timing diagram have been described in section 3.

The inMOD module interfaces internally with the 4 lowest level switches of the tmod module. Every 2 registers interfaces to a switch [i] through its bit-data line (Bit_1[i]) , its data-valid line (BitV_1[i]) and its bit-taken line (BTaken_2[i]). Since there are 8 registers and 3 interface lines per register, there are 18 signals connected to the tmod module.

The timing diagram is shown as below:



4.2.2. inMOD functionality

The inMOD module (or the Transmit PN Buffer) consists of: a Register File (8 registers); 8 Counters; 8 sets of control circuits; a 3-bit address decoder.

Each of the 8 Registers is 7 bits wide: the more significant 4 bits (bit6 --- bit3) are for storing the value, and the less significant 3 bits (bit2 --- bit0) are for storing the address of the receiving PN register. The bit0 is closest to the tmod module, and the bit6 is fartherest to the

tmod module. The value and the address can be written, in parallel, through the 7 data input pins (DTIN_B) within one clock (CLK0) cycle, starting during PH2 and ending at PH1.

The 3-bit address decoder determines, by decoding the input address (ATIN_B), which register gets written. Only one register can be written in a clock cycle. And, the WriteIN needs to go low to change a new input address. The Transmit Register File is a "write-only" file from the external pins.

There is a 3-bit Counter corresponding to a register. The Counter counts from 0 to 6. When the ResetIN signal is asserted, all the Counters get reset to 0 and their control circuits generate no-valid-bit signals (BitV_1=low) to the tmod module. Whenever a new message gets loaded into a register, its Counter resets to 0 and outputs a bit-valid signal (i.e. BitV_1 goes high) to the tmod module. As long as there are valid bits waiting to be transmitted, the BitV_1 signal stays high. When a register receives a BTaken_2 high signal during PH2 (sending back from the tmod module), its Counter increments by one. At the same time, the bit-data in the register gets shifted 1 bit towards the lower order bit because its control circuit generates a "shift" signal during PH2. The bit0 data gets recirculated back to the bit6 so that when the AutoIN signal gets asserted (in addition to the RunIN) the message can be transmitted to the tmod module again and again.

Each Counter keeps track of how many bits are left in its register. If all 7 bits of a message have been transmitted to the tmod module, the BitV[i] signal goes low during the same PH2 when its Counter receives the last BTaken_2 signal from the tmod module. When the external ResetIN signal is asserted, all the control signals are set to zero.

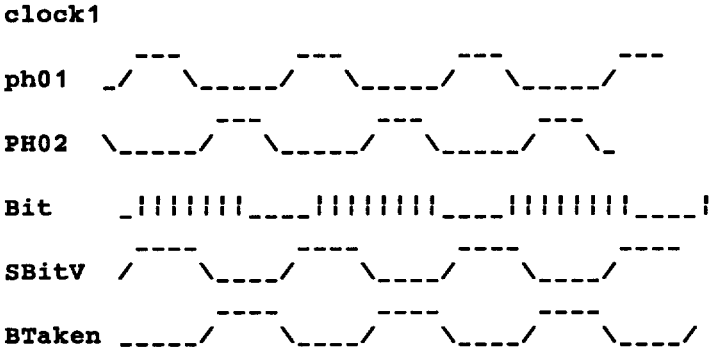
Additional information on the inter-module interface & timing diagram can be found in the treeMOD section.

4.3. Receive PN Buffer -- outMOD

4.3.1. outMOD interface

The outMOD module interfaces internally to the tMOD module and externally to the outside pins.

The outMOD module interfaces internally with the highest-level switch (S6) through 3 lines. The S6 transmits bit-data through the Bit line and bit-valid signal through the SBitV line. When the outMOD module receives a valid bit-data, a data-taken signal is sent back to the S6 through the BTaken line. The timing diagram is shown below:



4.3.2. outMOD functionality

The Receive PN consists of: a serial shift-in Register File (8 registers); a 3-bit address buffer; a Counter; an address decoder.

Each of the 8 receiving registers is 4 bits wide for storing the 4-bit value. All the registers are "read-only" from the external address (ARIN_B) pins.

The outMOD module receives bit-data from the highest-level switch or the broadcast transmitter, when RunIN is asserted. When SBitV is high, the bit-data (Bit) is trapped. At the same time, a data-taken signal is sent back to the broadcast transmitter (i.e. BTaken is high).

The first 3 arriving bits of a new message are recognized as an address and are shifted into the 3-bit address buffer. The address is decoded to select one of the 8 registers. The next 4 arriving bits are interpreted as value-bits and get shifted into the proper receiving register. The first 3 bits (i.e. the address of one of the 8 receiving registers) can be monitored by the 3 external (ADRB2:0) pins.

The modulo-7 Counter monitors the serial message traffic and does the bit-keeping for the Receive PN Buffer. At the beginning of a new message, the Counter is reset to 0. The Counter increments by one when a new bit-data has been trapped and the BTaken signal has been sent back to S6. At any time of operation, the ReadIN input can be asserted and the contents of any register can be read out by selecting the address through the 3 external address (ARIN_B) pins. The address is decoded combinationally and the 4 bits data come out of the 4 data pins (DROUT_B) in parallel along with the RRdyOUT status bit indicating valid data.

Please refer to the tMOD section for additional information on the inter-module interface signals & timing diagram.

4.4. Interconnect Structure Module - tmod

The TBH Test Chip interconnect module is called tmod. It simulates the operation of the concentrate tree and the single global receive line. A three level concentrate tree is used, and so is large enough to handle 8 transmit PNs. The bottom level of the concentrate tree has four switch nodes, each connected to two transmit PNs, one to a high address child and one to a low address child.

4.4.1. tmod Interface

The tmod module interfaces (internal to the TBH Test Chip) with both the inmod module and the outmod module.

4.4.1.1. tmod Interface Input Lines

Each of four bottom level concentrate tree switch nodes are connected to two transmit PNs via a set of three signal lines - Valid_1.x, Bit.x, and Taken_2.x, x = 0, 1, ..., 7. These signals run between inmod and tmod. These module interface signals are listed below:

Valid_1.x - transmit PN[x] has next bit ready

Bit.x - next data bit of transmit PN[x]'s message

Taken_2.x - switch has taken current bit (Bit.x)

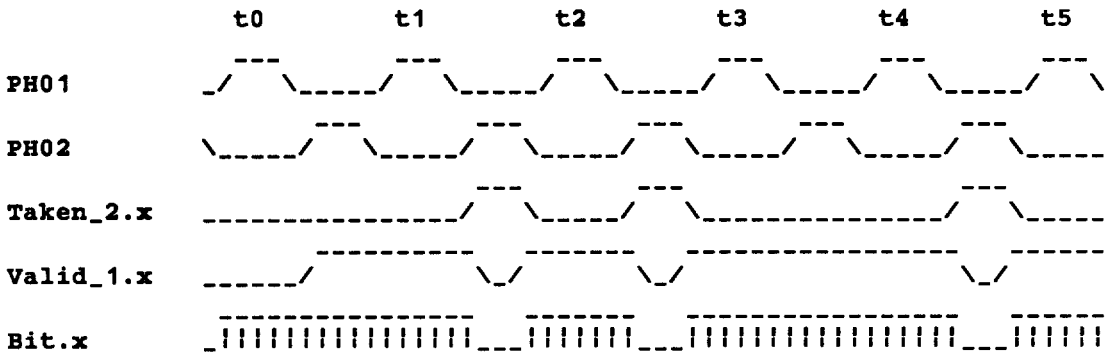
The Valid_1.x and Bit_1.x lines are inputs from the transmit PNs to the bottom level concentrate tree switches. The Taken_2.x lines are handshake lines from the bottom level concentrate tree switches to the transmit PNs.

Valid_1.x asserted means the transmit PN's next bit is ready. Taken_2.x asserted means the concentrate tree switch has accepted the current bit and is ready to accept the next bit of the message. The next bit is then placed on the Bit.x line and Valid_1.x is asserted.

Appendix I

Below is the timing diagram for three bits of a message from a transmit PN to a bottom level concentrate tree switch. The switch in this example does not take the first bit until the second clock cycle. This corresponds to the switch having been busy handling the end of some other message.

For all lines except Bit.x, high means high (Vdd!) and low means low (GND!). For Bit.x vertical "T"s means valid and low means not valid. Note that Valid_1.x must remain asserted until Taken_2.x is seen. This is because the switch node traps Bit.x with PH01.

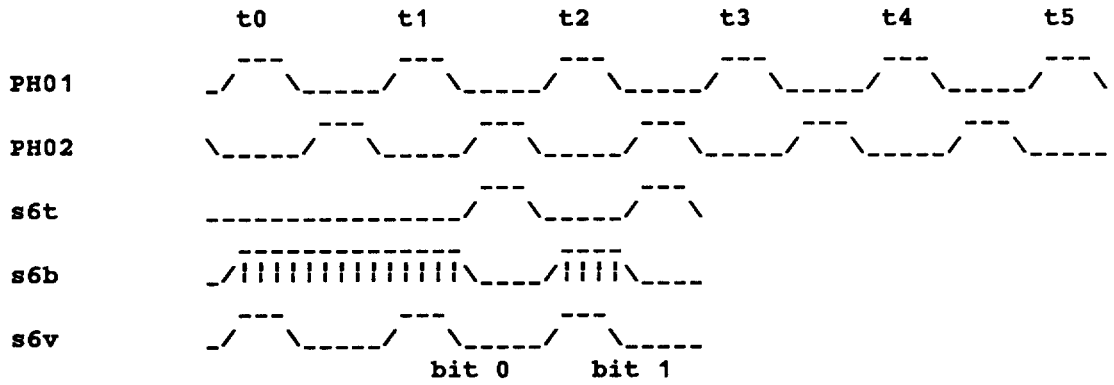


If Valid_1.x, Taken_2.x, and Bit.x (where x = 0, ... , 7 indexes one of the inmod PNs) are changed to syv, syt, and syb, respectively (where y = 0, ... , 6 indexes one of the tmod switch nodes and v indicateds the valid line, t indicates the taken line, and b indicates the bit, ie, data, line), the above timing diagram shows the synchronous handshake between two concentrate tree levels.

4.4.1.2. tmod Interface Output Lines

s6t	handshake and data between
s6b	top level concentrate tree
s6v	switch and receive PNs

Below is the timing diagram for two bits of a message from Switch6 to the receive PNs. The timing diagram is from the point of view of Switch 6 (in terms of propagation delay).



For all lines except s6b high means high (Vdd!) and low means low (GND!). For s6b, T's means valid and low means not valid.

4.4.2. tmod functionality

The tmod module is the interconnection communication structure between the transmit PNs and the receive PNs. It consists of a concentrate tree and a global receive line. Messages are generated by the transmit PNs, contend for transmission via the concentrate tree, and are ultimately received by the addressed receive PN.

As described earlier, the concentrate tree is a binary tree with a communication switch at each node. Each switch enforces a bandwidth slice priority scheme in deciding which of the potentially contending messages should be sent (given priority). Messages geting through to the top switch in the concentrate tree are sent along the global receive line to the receive PNs.

The highest level concentrate tree switch (Swith6) plays a pivotal function in the TBH Test Chip. Switch6 receives and implements the RUNINP line.

When the TBH Test Chip is running (ie, RUNINP asserted) and Switch6 sees RUNINP go low, Switch6 continues the current bit transfer (if any are in progress) to outmod (the receive PNs) and then blocks further bit transfers. In this way the TBH Test Chip is halted.

On RESETINP asserted, all concentrate tree switches will initialize to their powerup state.

5. Miscellaneous Helpful Information

Inmod signals valid as soon as data is written to a transmit register. The valid signal is not qualified by RUNINP. Tmod does not require this, but will accept it. When inmod sends the last bit of a message and AUTOINP is not asserted, inmod drops the valid line on PH02 (instead of at the beginning of the next PH01).

To load inmod PNs, the address and data lines must be valid before the beginning of PH02. In practice, asserting the address and data lines during PH01 works well. Also, WRITEINP and RUNINP must be asserted mutually exclusively. This means there must be at least one phase of null cycle after WRITEINP drops and before RUNINP is asserted. RUNINP must become true after PH02 drops and before PH01 starts. Finally, WRITEINP is a PH02 valid signal and must remain valid througout PH02.

In tmod, only the output from each switch is RUNINP qualified. This means when RUNINP drops each leaf node level switch will see the inmod register's valid line true (for those that are true) and load in a potentially extra bit as data. Because of this and in general, RUNINP must remain asserted during each individual run. RESETIN must be asserted for at least one complete cycle between individual runs.

Appendix II

Derivations of Formulas

1. Derivation of Number and Area of SNs

1.1. Derivation of Number of SNs — Equation (8)

As presented earlier, the number of SNs in a wafer implementation of a CAP neurocomputer containing an array of 64K PNs is

$$N_{SN} = 2 \times \sum_{i=1}^{5} \left(\frac{N_{PN}}{\alpha^i} \right) \Bigg]_{\substack{\alpha = 4 \\ N_{PN} = 64K}} = 43648 \tag{8}$$

Throughout this section $f(C)\Big]_{C=A}$ is interpreted to mean formula f(C) evaluated at the condition $C = A$.

A wafer with 64K PNs could be covered either by 1 H_8 broadcast domain, 4 H_7 broadcast domains, ... , or 16K H_1 broadcast domains. For equation (8), the assumption is made the wafer is covered by 64 H_5 PBSs in an 8 by 8 array.

There are two ways to calculate the number of SNs on a wafer. One can calculate the number of SNs in each H_5 broadcast domain and multiply by 64. Alternately, one can sum the number of H_1s, H_2s, H_3s, H_4s, and H_5s on the wafer for one side of a dual tree, and multiply the sum by two because there are two trees per dual tree. The first method produces

$$N_{SN} = 64 \times N_{SN,H_5} = 64 \times \sum_{i=1}^{5} N_{SN,H_5^i} \tag{11}$$

$$= 64 \times (512 + 128 + 32 + 8 + 2) = 43648$$

where N_{SN,H_5} is the number of SNs in each H_5, and N_{SN,H_5^i} is the number of SNs at level i of a single H_5 dual tree. Using the alternative approach yields

$$N_{SN} = 2 \times \sum_{i=1}^{5} \frac{N_{SN}^i}{2} = \sum_{i=1}^{5} N_{SN}^i \tag{12}$$

$$= 32768 + 8192 + 2048 + 256 + 64 = 2 \times 21824 = 43648$$

where N_{SN}^i is the number of SNs across the wafer at level i, for all H_5's on the wafer. Of course, equations (11) and (12) produce identical answers.

Appendix II

1.2. Derivation of SN Area — Equation (9)

$$r_{\frac{A_{SN,w}}{A_{total,w}}} = \frac{A_{SN,w}}{A_{SN,w} + A_{PN,w}} = \frac{A_{SN} \times N_{SN}}{A_{SN} \times N_{SN} + A_{PN} \times N_{PN}} = .0361 = 3.6\% \qquad (9)$$

The parts of equation (9) are derived as follows. The area for all SNs across a 64K PN wafer with the same assumptions as for equation (8), and scaling from a 3μ process to a 1.25μ process is

$$A_{SN,w} = N_{SN} \times A_{SN} = 43648 \times .69 \times 10^6 \mu^2 = 3.01 \times 10^{10} \mu^2$$

where the value for A_{SN}, the area for a single SN, was given in § 4.3. The area for all PNs across the wafer is

$$A_{PN,w} = N_{PN} \times A_{PN} = 65536 \times 12.25 \times 10^6 \mu^2 = 8.03 \times 10^{11} \mu^2$$

where the value for A_{PN}, the area for a single PN, was given in § 4.3. Therefore, the ratio of SN area to total area is

$$r_{\frac{A_{SN,w}}{A_{w,total}}} = \frac{A_{SN,w}}{A_{SN,w} + A_{PN,w}} = \frac{3.01 \times 10^{10} \mu^2}{3.01 \times 10^{10} \mu^2 + 8.03 \times 10^{11} \mu^2} = .0361 = 3.6\%$$

It is important to keep in mind the fact that the A_{PN} area estimate used in these calculations includes only space for PN memory, and no space for the calculation hardware or control circuitry. Therefore, A_{PN} may increase substantially when other PN circuitry is included, resulting in a corresponding decrease in $r_{\frac{A_{SN,w}}{A_{w,total}}}$.

2. Derivation of Propagation Power

2.1. Power for Individual H_i

This section shows the derivation of the propagation power figures shown in table 2. Due to the fact that only the highest priority message path in a concentrate tree is transmitting at any particular time, concentrate tree propagation power dissipation is always negligible. Hence, only the broadcast tree contribution to dual tree propagation power dissipation is considered.

The propagation power calculations are based on SPICE simulation runs using a 3 μ MOSIS process technology (see figure 12). In each case a single low (Gnd) to high (Vdd) data transition was simulated for various metal 2 line lengths. Each line was simulated using a second order Π RC model as depicted in figure 9. Each line has a driver (either a 1X or a 10X inverter) on one end and a receiver on the other end. The transition was defined as the time from when the signal driving the driver passed 50% of a full high to low swing (.5 Vdd) to when the output voltage of the receiver reached 50% of its high to low swing. This transition time resulted in a 35 MHz clock rate. Total power consumed during the transition was calculated by use of a 1 Farad current integrating capacitor connected between Vdd and the test circuit. The charge on the capacitor at the end of the transition run represents the total current that flowed and therefore the power consumed. Note that discharging the line causes no additional current to flow through the current integrating capacitor. In effect, the current

integrated by the capacitor represents the power for a pair of low to high and back to low again cycles. — one cycle to send a one (charge the line up) and one cycle to send a zero (discharge the line). Multiplying by the 35 MHz clock rate divided by 2 cycles/charge-discharge-cycle gives the propagation power to drive the line at a 100% duty cycle.

Repeaters and fat-tree throughput maintenance is used with $L_{3,4}$ and higher lines when 1X PBS drivers are used. The following equation recursively defines propagation power consumed by broadcast domains H_1 through H_8. All calculations are based on a 35 MHz clock rate and transmission of random data.

$$P_{H_i}\Big]_{1X} = \begin{cases} \alpha \times P_{L_{0,1}}\Big]_{1X} & \text{if } i = 1 \\ \alpha \left(P_{H_{i-1}}\Big]_{1X} + P_{L_{i-1,i}}\Big]_{1X} \right) & \text{if } 1 < i < 4 \\ \alpha \left(P_{H_{i-1}}\Big]_{1X} + 4^{i-3} \times P_{L_{2,3}}\Big]_{1X} \right) & \text{if } i >= 4 \end{cases} \tag{13}$$

As usual, α is the branching factor. $P_{L_{i-1,i}}\Big]_{1X}$ is the propagation power to drive a level $L_{i-1,i}$ line using 1X drivers. $P_{H_i}\Big]_{1X}$ is the propagation power for an H_i broadcast domain using 1X drivers. The first subequation in equation (13) serves to ground the recursion when $i = 1$. It represents the propagation power for a H_1 PBS with 1X drivers. The second subequation recursively defines propagation power for H_2 and H_3. The last subequation recursively defines propagation power for H_4 through H_8.

The equation for power to drive a single line is

$$P_{L_{i,i+1}}\Big]_{1X} = \frac{P^{0\rightarrow 1}_{L_{i,i+1}}\Big]_{1X} \times 35MHz}{4} \tag{14}$$

$P^{0\rightarrow 1}_{L_{i,i+1}}\Big]_{1X}$ is the power for driving the level i+1 to level i line in the broadcast tree from a low to high signal level. Values for $P^{0\rightarrow 1}_{L_{i,i+1}}\Big]_{1X}$ were obtained from SPICE simulations using a 3 $\times$ MOSIS non-scalable process. Since the line is being charged up to Vdd, $P^{0\rightarrow 1}_{L_{i,i+1}}\Big]_{1X}$ includes the power for discharging the line when the 1$\rightarrow$0 transition occurs. Half of the $\div$ 4 adjusts for the fact that virtually no additional power is consumed from Vdd to discharge the line on the 1$\rightarrow$0 data transition. The other half of the $\div$ 4 adjusts for the fact that the data is random, so a transition will occur on average every other bit-time.

Note for the *if* $i >= 4$ part of equation (13), the 4^{i-3} comes from the invocation of repeaters and fat-tree throughput maintenance at level 4 and above; it represents a factor of 4 increase in the term at each level above level 3. First, there is a factor of two per level of power increase due to the use of repeaters. In effect, longer lines are built out of $L_{2,3}$ lines with a repeater between each section. Second, there is an additional factor of two increase in propagation power due to the fact that the width of a flit doubles at each level above level 3; twice as many lines are being driven in parallel at each higher level.

Appendix II

Applying equation (13) yields the following propagation power figures for each PBS using 1X drivers.

$$P_{H_1}\Big]_{1X} = \alpha \times P_{L_{0,1}}\Big]_{1X} = 4 \times .65\times10^{-11} \div 4 \times 35MHz = .228mW$$

$$P_{H_2}\Big]_{1X} = \alpha \left(P_{H_1}\Big]_{1X} + P_{L_{1,2}}\Big]_{1X} \right)$$

$$= 4\,(.228mW + 1.0\times10^{-11}Watts \div 4 \times 35MHz) = 1.26mW$$

$$P_{H_3}\Big]_{1X} = \alpha \left(P_{H_2}\Big]_{1X} + P_{L_{2,3}}\Big]_{1X} \right)$$

$$= 4\,(1.26mW + 1.6\times10^{-11}Watts \div 4 \times 35MHz) = 5.6mW$$

$$P_{H_4}\Big]_{1X} = \alpha \left(P_{H_3}\Big]_{1X} + 4^{4-3} \times P_{L_{2,3}}\Big]_{1X} \right)$$

$$= 4\,(5.6mW + 4^1 \times 1.6\times10^{-11}Watts \div 4 \times 35MHz) = 24.6mW$$

$$P_{H_5}\Big]_{1X} = \alpha \left(P_{H_4}\Big]_{1X} + 4^{5-3} \times P_{L_{2,3}}\Big]_{1X} \right)$$

$$= 4\,(24.6mW + 4^2 \times 1.6\times10^{-11}Watts \div 4 \times 35MHz) = 107mW$$

$$P_{H_6}\Big]_{1X} = \alpha \left(P_{H_5}\Big]_{1X} + 4^{6-3} \times P_{L_{2,3}}\Big]_{1X} \right)$$

$$= 4\,(107mW + 4^3 \times 1.6\times10^{-11}Watts \div 4 \times 35MHz) = 464mW$$

$$P_{H_7}\Big]_{1X} = \alpha \left(P_{H_6}\Big]_{1X} + 4^{7-3} \times P_{L_{2,3}}\Big]_{1X} \right)$$

$$= 4\,(464mW + 4^4 \times 1.6\times10^{-11}Watts \div 4 \times 35MHz) = 2W$$

$$P_{H_8}\Big]_{1X} = \alpha \left(P_{H_7}\Big]_{1X} + 4^{8-3} \times P_{L_{2,3}}\Big]_{1X} \right)$$

$$= 4\,(2W + 4^5 \times 1.6\times10^{-11}Watts \div 4 \times 35MHz) = 8.6W$$

The propagation power for PBS broadcast structures with 10X drivers is similar to equation (13), but with two differences. The first is that numbers from SPICE runs for $P^{0\to1}_{L_{i,i+1}}\Big]_{10X}$ are used in place of those for $P^{0\to1}_{L_{i,i+1}}\Big]_{1X}$. The second difference is that fat-tree and repeater throughput maintenance begins at level 6 with 10X drivers instead of at level 4 as with 1X drivers because the 10X drivers can drive longer lines than the 1X drivers. Therefore, the recursive equation for propagation power with 10X drivers is

Appendix II

$$P_{H_i}\Big]_{10X} = \begin{cases} \alpha \times P_{L_{0,1}}\Big]_{10X} & \text{if } i = 1 \\ \alpha \left(P_{H_{i-1}}\Big]_{10X} + P_{L_{i-1,i}}\Big]_{10X} \right) & \text{if } 1 < i < 6 \\ \alpha \left(P_{H_{i-1}}\Big]_{10X} + 4^{i-5} \times P_{L_{4,5}}\Big]_{10X} \right) & \text{if } i >= 6 \end{cases} \quad (15)$$

Applying equation (15) yields the following propagation power figures for 10X drivers.

$$P_{H_1}\Big]_{10X} = \alpha \times P_{L_{0,1}}\Big]_{10X} = 4 \times 1.5\times10^{-11} Watts \div 4 \times 35MHz = .525mW$$

$$P_{H_2}\Big]_{10X} = \alpha\left(P_{H_1}\Big]_{10X} + P_{L_{1,2}}\Big]_{10X}\right)$$

$$= 4\,(.525mW + 1.9\times10^{-11} Watts \div 4 \times 35MHz) = 2.77mW$$

$$P_{H_3}\Big]_{10X} = \alpha\left(P_{H_2}\Big]_{10X} + P_{L_{2,3}}\Big]_{10X}\right)$$

$$= 4\,(2.77mW + 3\times10^{-11} Watts \div 4 \times 35MHz) = 12.1mW$$

$$P_{H_4}\Big]_{10X} = \alpha\left(P_{H_3}\Big]_{10X} + P_{L_{3,4}}\Big]_{10X}\right)$$

$$= 4\,(12.1mW + 4.3\times10^{-11} Watts \div 4 \times 35MHz) = 49.9mW$$

$$P_{H_5}\Big]_{10X} = \alpha\left(P_{H_4}\Big]_{10X} + P_{L_{4,5}}\Big]_{10X}\right)$$

$$= 4\,(49.9mW + 7.8\times10^{-11} Watts \div 4 \times 35MHz) = 202mW$$

$$P_{H_6}\Big]_{10X} = \alpha\left(P_{H_5}\Big]_{10X} + 4^{6-5} \times P_{L_{4,5}}\Big]_{10X}\right)$$

$$= 4\,(202mW + 4^1 \times 7.8\times10^{-11} Watts \div 4 \times 35MHz) = 819mW$$

$$P_{H_7}\Big]_{10X} = \alpha\left(P_{H_6}\Big]_{10X} + 4^{7-5} \times P_{L_{4,5}}\Big]_{10X}\right)$$

$$= 4\,(819mW + 4^2 \times 7.8\times10^{-11} Watts \div 4 \times 35MHz) = 3.32W$$

$$P_{H_8}\Big]_{10X} = \alpha\left(P_{H_7}\Big]_{10X} + 4^{8-5} \times P_{L_{4,5}}\Big]_{10X}\right)$$

$$= 4\,(3.32W + 4^3 \times 7.8\times10^{-11} Watts \div 4 \times 35MHz) = 13.5W$$

Using 10X drivers, an $L_{4,5}$ line is the longest that can be driven without using repeaters and still manage a 35 MHz bit rate. 10X driver lines at levels $L_{5,6}$ and above must use repeaters and fat-tree throughput maintenance to keep up with a 35 MHz clock. Finally, the 10X driver figures are sensible as compared to the 1X driver figures.

The 10X drivers overcome the R-C line by driving it harder, resulting in more power being burned up overcoming the line resistance.

2.2. Power for All SNs on a Wafer

We consider two wafer contexts: A 1.25 μ process technology and a 0.1 μ that may exist by the end of the century.

2.2.1. SN Power for 1.25 μ Wafer

Assuming a PN requires a square area .0035 meters on a side using a 1.25 μ process [personal communication from Jim Bailey, CAP project], each PN consumes an area of $1.225 \times 10^{-5} m^2$ per PN, for a PN supporting a virtualization granularity of 16 CNs/PN and each CN supporting 1K connections. The area on a 6" wafer is

$$A_{w_{6"}} = \pi r^2 = 3.14 \times (3 inches \times .0254 m/inch)^2 = .01824 m$$

The number of PNs that could fit on such a wafer is

$$N_{PN_{6"}} = A_{w_{6"}} \div A_{PN}\Big]_{1.25\mu} = \frac{.01824 m^2}{1.225 \times 10^{-5} m^2} = 1488\ PNs$$

As in the previous section, $A_{PN}\Big]_{condition\ X}$ means A_{PN} is evaluated under the stated condition (*condition X*). Ease of layout requires a regular structure, such as a square array of PNs. A square array of 1K PNs will fit on a 6" 1.25 μ process technology wafer. The remaining area can be used to support I/O and on-wafer diagnostic hardware or it can be left unused.

A 1K PN array is exactly the size of a single H_5. Such a PN array can also be covered by four overlapping H_4 domains. We calculate propagation power assuming two sets of overlapping H_4 domains. Note that the "odd" set (the set not edge-aligned with the PN array) will consist of one complete H_4 domain in the center of the wafer, four half H_4 domains on the edges of the PN array, and four quarter H_4 domains at the corners of the PN array.

The propagation power for a single H_4 domain with 1X drivers from table 2 is 24.6 mW. The entire 6" wafer will contain the equivalent of eight complete H_4 domains. The 24.6 mW per H_4 propagation power figure from table 2 was figured for a 3 μ MOSIS non-scalable technology. Since power scales as $\frac{1}{(scaling\ factor)^2}$ [WeE85], for a 1.25 μ technology propagation power will be approximately

$$P_{H_4}\Big]_{1.25\mu} = P_{H_4}\Big]_{3\mu} \times \left(\frac{1.25}{3}\right)^2 = 24.6 mw \times .174 = 4.27 mW$$

For an entire wafer containing eight H_4s, propagation power is

$$P_{H_{4's\ on\ a\ 6"\ wafer}}\Big]_{1.25\mu} = 8 \times P_{H_4}\Big]_{1.25\mu} = 8 \times 4.27 mW = 34.2 mW$$

2.2.2. Power for 0.1 μ Wafer

In this section we perform a similar analysis but use the following assumptions. An 8" wafer will be implemented using a .1 μ process technology. This represents the

expected limit of CMOS technology for the future due to fundamental limitations imposed by physics. We will continue to assume power scales as $1/\alpha^2$. The assumption is probably overly optimistic because the supply voltage used is unlikely to scale by $1/\alpha$ to less than the .5 volts assumed by ideal scaling.

A .1 μ CMOS technology will have an lineal scaling factor of $\frac{.1}{1.25} = .08$ as compared to a 1.25 μ technology. Therefore, area for a single PN will be

$$A_{PN}\Big]_{.1\mu} = A_{PN}\Big]_{1.25\mu} \times .08^2 = 1.225\times10^{-5}m^2 \times .0064 = 7.84\times10^{-8}m^2$$

The area of the 8" wafer will be

$$A_{w_{8"}} = \pi r^2 = 3.14 \times (4\ inches \times .0254\ m/inch)^2 = .0324\ m^2$$

The number of PNs that could fit on such a wafer is

$$N_{PN_{8"}} = A_{w_{8"}} \div A_{PN}\Big]_{.1\mu} = \frac{.0324m^2}{7.84\times10^{-8}m^2} = 415{,}384\ PNs$$

Since relative granularity of broadcast domains is much finer here than was the case with the 6" 1.25 μ wafer, most of the PNs on the 8" wafer can be productively used. We estimate 399,360 of the potential 415,384 PNs can be incorporated into broadcast domains, yielding 390 H_5 domains, each containing 1024 PNs. For two way overlapped and staggered coverage (as with the 6" wafer), there will be 780 H_5 domains in total.

The propagation power for a single H_5 domain with 1X drivers from table 2 is 107 mW (3 μ non-scalable MOSIS process). But since power scales as $\frac{1}{(scaling\ factor)^2}$, for a 0.1 μ technology propagation power will be approximately

$$P_{H_5}\Big]_{.1\mu} \times \left(\frac{.1}{3}\right)^2 = 107mw \times .00111 = .119mW$$

For the entire wafer (780 H_5s), propagation power is

$$P_{H_5{}^{w_8}}\Big]_{.1\mu} = 780 \times P_{H_5}\Big]_{.1\mu} = 780 \times 0.119mW = 92.8mW$$

Appendix II

References

[BBB] C. Bahr, J. Bailey, T. Baker, G. Beaver, D. Hammerstrom, K. Jagla, J. Mates, N. May, H. McCartor and M. Rudnick, An overview of the CAP project at OGC, Unpublished paper.

[BaH86] J. Bailey and D. Hammerstrom, "How to Make a Billion Connections," Tech. Report CS/E-86-007, Dept. of Computer Science/Engineering, Oregon Graduate Center, Beaverton, Oregon, July 1986.

[Bai] J. Bailey, "A VLSI Interconnect Structure for Neural Networks," Ph.D. Dissertation, Department of Computer Science/Engineering, Oregon Graduate Center. In Preparation.

[BaM85] H. B. Bakoglu and J. D. Meindl, "Optimal Interconnection Circuits for VLSI," *IEEE Transactions of Electron Devices*, vol. ED-32, 5 (May 1985), pp. 903.

[Col87] B. C. Cole, "INOVA Brings Wafer-Scale Integration to Market," *Electronics*, January 8, 1987.

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

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

[Ham86] D. Hammerstrom, "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.

[HMT86] D. Hammerstrom, D. Maier and S. Thakkar, "The Cognitive Architecture Project," *Computer Architecture News*, vol. 14, 1 (January 1986), pp. 9-21, ACM SigArch.

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

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

[May88] N. May, "Fault Simulation of a Wafer-Scale Neural Network," Masters Dissertation, Department of Computer Science/Engineering, Oregon Graduate Center, 1988.

[MeC80] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley Publishing Co., Reading, MA, 1980.

[MeM88] C. A. Mead and M. A. Mahowald, "A Silicon Model of Early Visual Processing," *Neural Networks*, vol. 1, 1 (January 1988), pp. 91-97.

[Ous] J. Ousterhout, *Magic Tutorial, version 4*, Computer Science Division, Electrical Engineering and Computer Sciences, Univerity of California, Berkeley, CA.

[RuN87] M. Rudnick and S. Neighorn, A Virtual Broadcast Hierarchy Simulation, OGC Architecture Class Project, CS&E 523, June 1987.

[RuH88] M. Rudnick and D. Hammerstrom, "An Interconnect Structure for Wafer Scale Neurocomputers," in *Proceedings of the 1988 Connectionist Models Summer*

School, D. Touretzky, G. Hinton and T. Sijnowski (ed.), 1988, pp. Morgan Kaufmann Publishers, Inc..

[WeE85] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, 1985.

Figures

Figure 1: Degree of Parallelism

Figure 1 shows the degree of parallelism for four architectural alternatives for neurocomputer design plotted on a log scale. The scale runs from 1 CN/PN to 1KK (2^{20}) CNs/PN. The left side of the log scale plots virtualization granularity in units of CNs per PN. The right side of the log scale plots $\log_2$(virtualization granularity).

In order to make the different architectural alternatives comparable, a sample problem (artificial neural network model) to be run of size $N_{CN} = 2^{20}$ CN nodes is postulated. The CAP design lies near the massively parallel end of the virtualization granularity scale while the alternative architectures lie near the zero parallelism end.

Note that the virtualization granularity figure characterizes only degree of parallelism, not size of network that can be run nor computational performance. Of course, while holding computational unit performance constant, the greater the degree of parallelism the greater the total performance.

Figure 2: H_3 Dual Tree with $\alpha = 2$

PNs run across the center of the diagram. The dual tree consists of two parts: the concentrate tree above the PNs (nodes labeled $C_{j,i}$) and the broadcast tree below the PNs (nodes labeled $B_{j,i}$). The broadcast tree is inverted for presentation in the diagram. The PBS dual tree has a branching factor of $\alpha = 2$ instead of the standard PBS branching factor of $\alpha = 4$. Using a branching factor of $\alpha = 2$ simplifies the diagram for illustration purposes and allows it to fit easily on a single page.

Message flow is as follows: A CN within some PN_i changes its output state. It generates a message and passes the message to PN_i. The PN then tries to pass the message to its concentrate tree SN (switch node) $C_{1,\lfloor i/2 \rfloor}$, where the message contends for $C_{1,\lfloor i/2 \rfloor}$'s output path. From there the message flows to $C_{2,\lfloor i/4 \rfloor}$, and then to $C_{3,\lfloor i/8 \rfloor} = C_{3,0}$, the root of the concentrate tree. At each $CN_{j,i}$ the message contends for transmission.

From the root of the concentrate tree the message moves to the root of the broadcast tree, $B_{3,0}$. This move happens without contention, as do succeeding moves down the broadcast tree (up in the diagram). From SN $B_{3,0}$, a copy of the message is broadcast to each of the next level broadcast SNs, $B_{2,0}$ and $B_{2,1}$. From each of these nodes the message is broadcast to each of their child nodes ($B_{1,0}$ and $B_{1,1}$ for $B_{2,0}$, and $B_{1,2}$ and $B_{1,3}$ for $B_{2,1}$). Finally, the message is broadcast from each of the bottom level broadcast tree SNs to their respective PNs (e.g., $B_{1,0}$ broadcasts to PN_0 and PN_1).

Figure 3: PBS Fat-Tree

A PBS fat-tree is shown with a branching factor of $\alpha = 2$ (as opposed to the usual PBS branching factor of $\alpha = 4$) and the standard PBS multiplexing ratio of $r_m = 2$. A multiplexing ratio of two says a connection at each level has twice as many data lines as does the connection at the next lower level. The doubling of the number of data lines at each level compensates for the fact that it takes twice as long at each level to send a single data bit as it does at the previous level. Taking twice as long at each level, in turn, is due to the fact that a line leaving level L_i is twice as long as a line

leaving level L_{i-1}.

The net effect is that the number of bits transferred per unit time is constant between any two connected nodes. This effect is called fat-tree throughput maintenance.

Figure 4: A Single Fat-Tree Switch Node

Figure 4 shows a single fat-tree level L switch node and the two data buffers inside the switch node. In a pure fat-tree, the size of each buffer is 2^L bits, which is equal to the size of the switch node's output flit for concentrate trees or input flit for broadcast trees. The concentrate tree SN receives each message from the next lower level as a series of pairs of consecutive flits. Each incoming flit is of size 2^{L-1} bits.

For concentrate tree, the time it takes an SN to receive two incoming $f_{C,L-1}$ flits is $2 \times t_{f,C,L-1}$. In this time it can send only a single output flit, $f_{C,L}$, because, of course, $2 \times t_{f,C,L-1} = t_{f,C,L}$.

Figure 5: Domain Coverage Efficiency vs Broadcast Domain Size

This plot shows a hypothetical domain coverage efficiency versus broadcast domain size for a c-graph with good locality. Domain coverage efficiency is maximum when the domain size is smallest. This will be true for all networks to be run which posses good locality, provided their p-graph mapping captures that locality.

Figure 6: Flit Propagation Delay vs Transfer Level

For driver sizes of 1X, 10X, and 100X, this graph shows flit propagation delay versus tree transfer level. Note that each curve is exponential, due to the resistance part of the long line RC characteristic. A large reduction in flit propagation delay is achieved by moving from 1X drivers to 10X drivers, while only a modest gain is achieved by moving from 10X to 100X drivers.

Figure 7: Worst Case Propagation Power vs Transfer Level

Figure 7 shows worst case power versus transfer level for each of three driver sizes: 1X, 10X, and 100X. Worst case power is based on a single line running as fast as possible at a 100% duty cycle (transferring a string of alternating zeros and ones).

In general the throughput for each driver size at each level is different, because the larger the driver the faster it can move the line from a state to the opposite state (e.g., from 1 to 0). At the higher transfer levels the power for all driver sizes becomes similar, because the resistance part of the long line RC characteristic limits how much current can flow through the driven end of the wire. Even a voltage source would show a tendency to converge to the "common" power figure.

Figure 8: Single Bit Power vs Transfer Level

Figure 8 shows power to send a singe bit (switch the line from 0 to 1) versus transfer level. While figure 6 shows a big gain in speed by moving from 1X drivers to 10X drivers, 8 shows only a modest rise in expended power in moving from 1X to 10X drivers.

Figures

Figure 9: Spice Test Circuits

Both the long line test circuit SPICE model (left side) and the Vdd charge integration circuit (right side) are shown.

The long line test circuit consists of a driver, which may be of size 1X, 10X, or 100X, driving a second order Π RC circuit model of a long line, with a 1X receiver on the other end of the long line. The values for the resistors and capacitors in the second order Π long line model are calculated based on the MOSIS 3μ non-scalable process technology.

All Vdd current supplied to the simulation circuit is provided by a 1 Farad capacitor in series with the Vdd supply voltage source. The capacitor serves to integrate the current flowing through it during the simulation, thereby measuring total charge that flows during simulated time, making for an easy calculation of average power based on average current supplied.

Figure 10: H_3 Concentrate Tree with Redundant Upper Level SNs

This diagram shows the concentrate portion of a H_3 non-fat dual tree where coarse grain redundancy has been used at the upper two levels (levels L_3 and L_2) as a fault amelioration technique. Levels L_0 and L_1 are as depicted in the concentrate tree portion of figure 3. Level L_2 consists of two redundant SNs, each of which consists of two SNs. Likewise, level L_3 consists of a single redundant SN consisting of two SNs.

Figure 11: TBH Test Chip Layout

This figure shows the layout of the TBH Test Chip. The seven node concentrate tree occupies the lower half of the chip layout. Four SNs are sandwiched across the layout just below the center. The remaining three SNs are at the bottom of the chip. Each SN is approximately 950 λ on each side, where $\lambda = 1.5\mu$.

Figure 12: SPICE Propagation Power Test Circuit

Diagram of circuit used as basis for SPICE simulation and used to calculate propagation power.

Figures

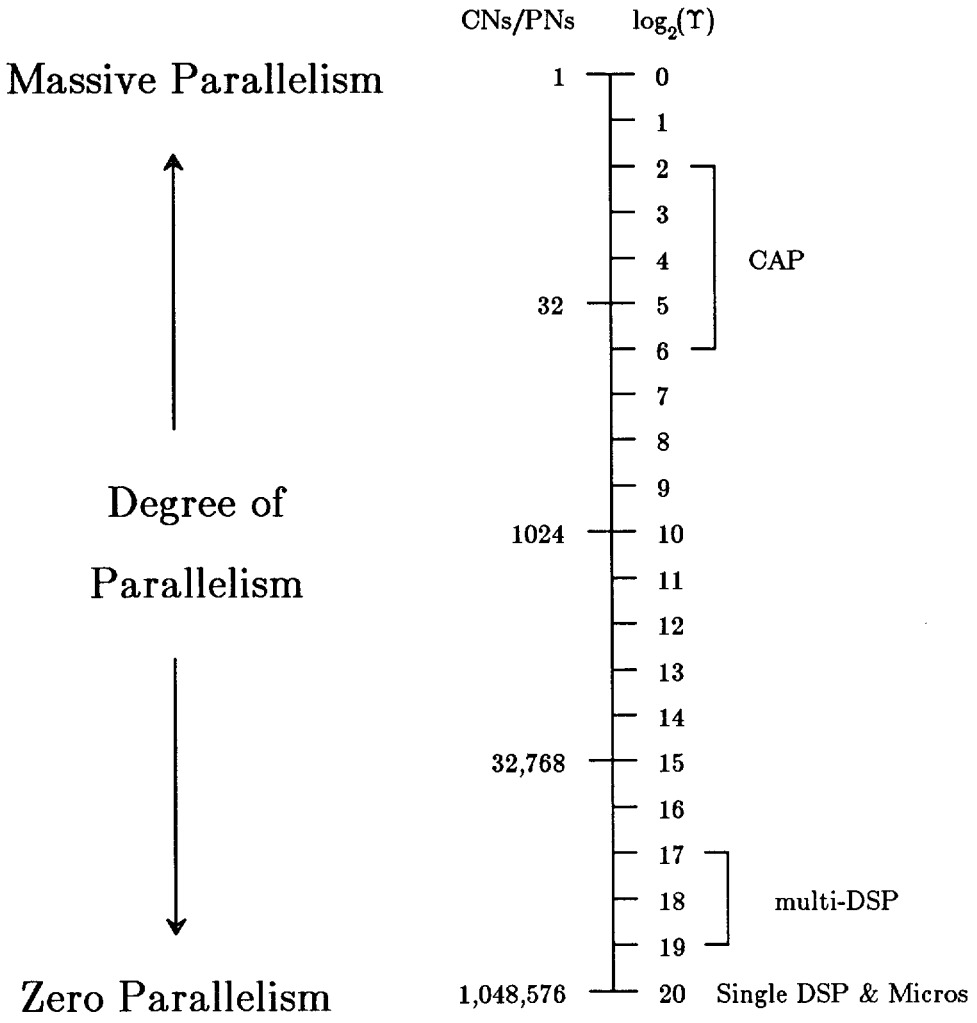


Figure 1: Degree of Parallelism

Figures

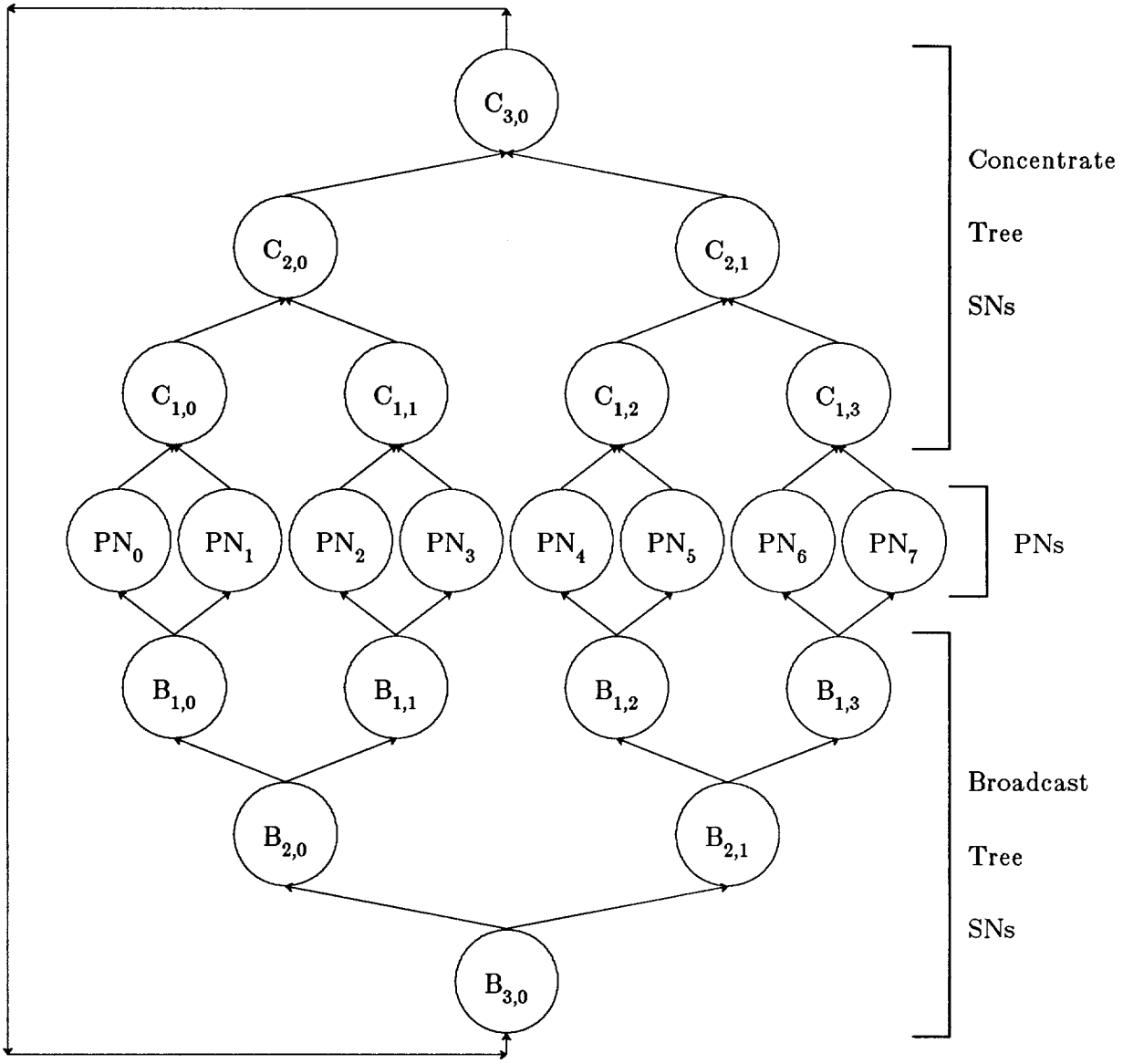


Figure 2: D_3 Dual Tree with $\alpha = 2$

Figures

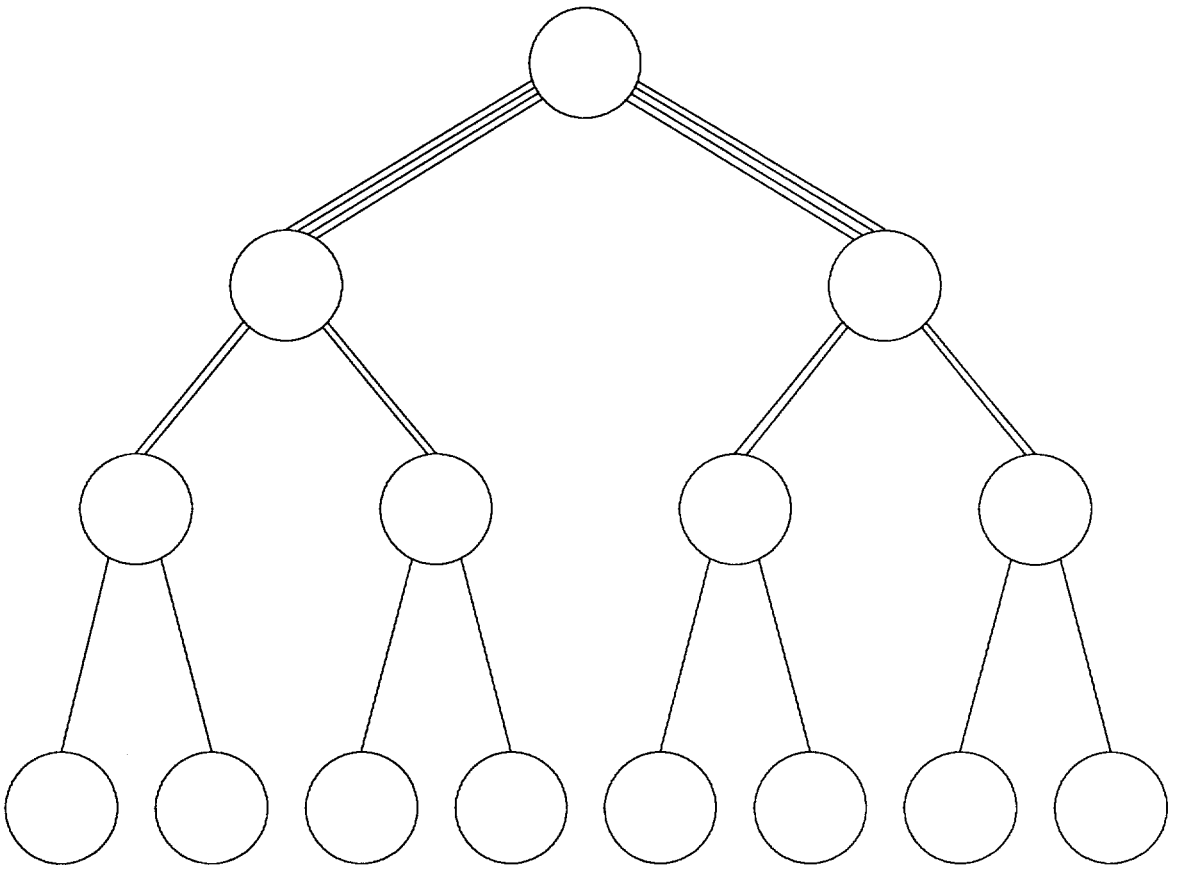


Figure 3: PBS Fat-Tree

Figures

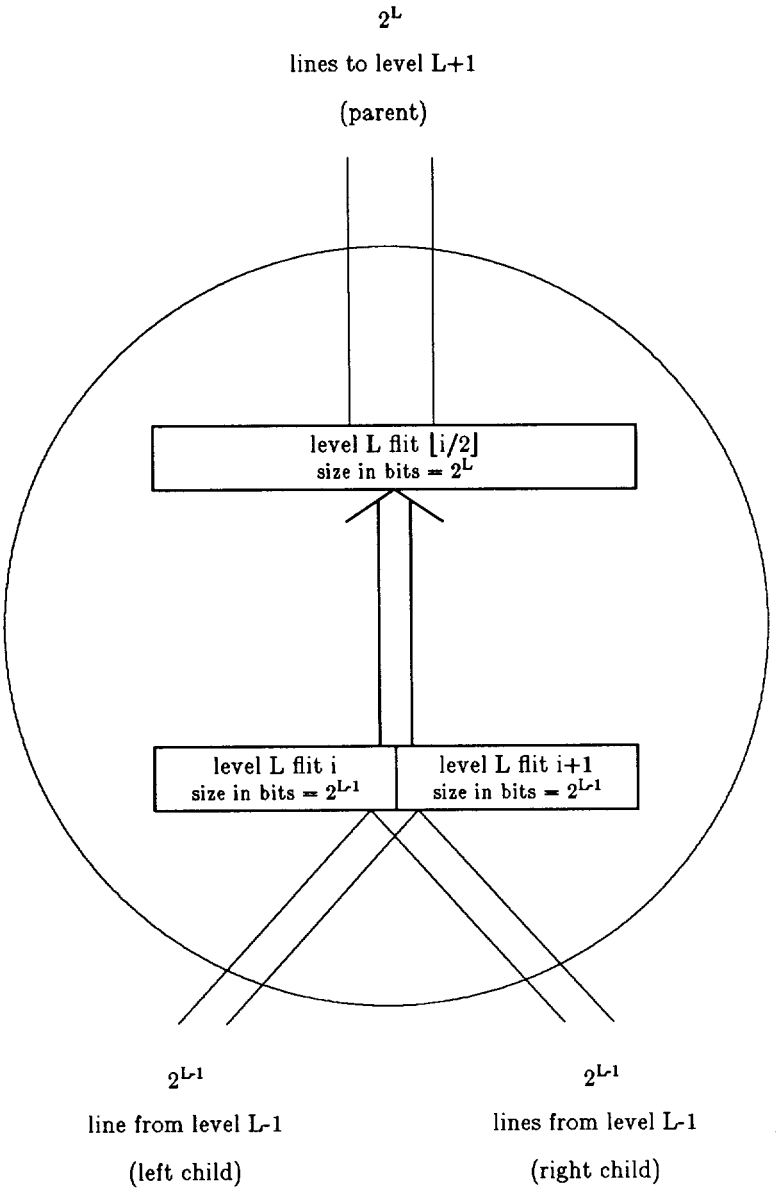


Figure 4: A Single Fat-Tree Switch Node

Figures

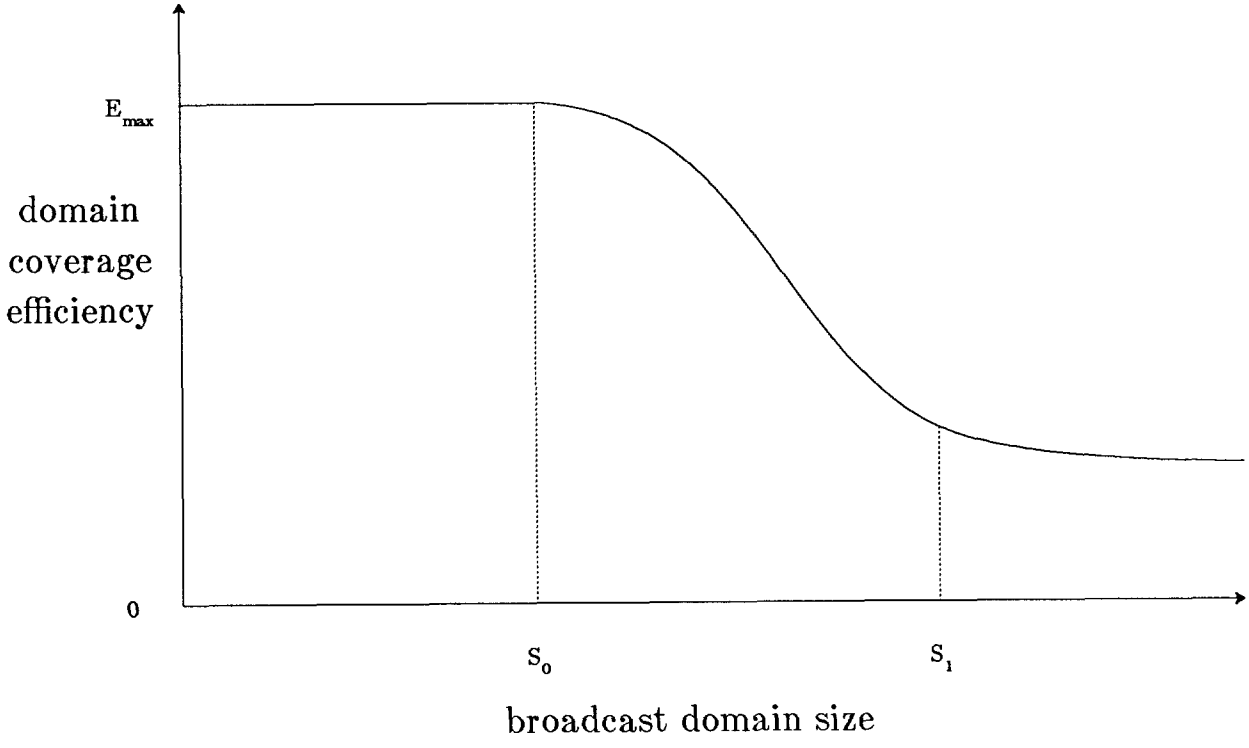


Figure 5: Domain Coverage Efficiency vs Broadcast Domain Size

Figures

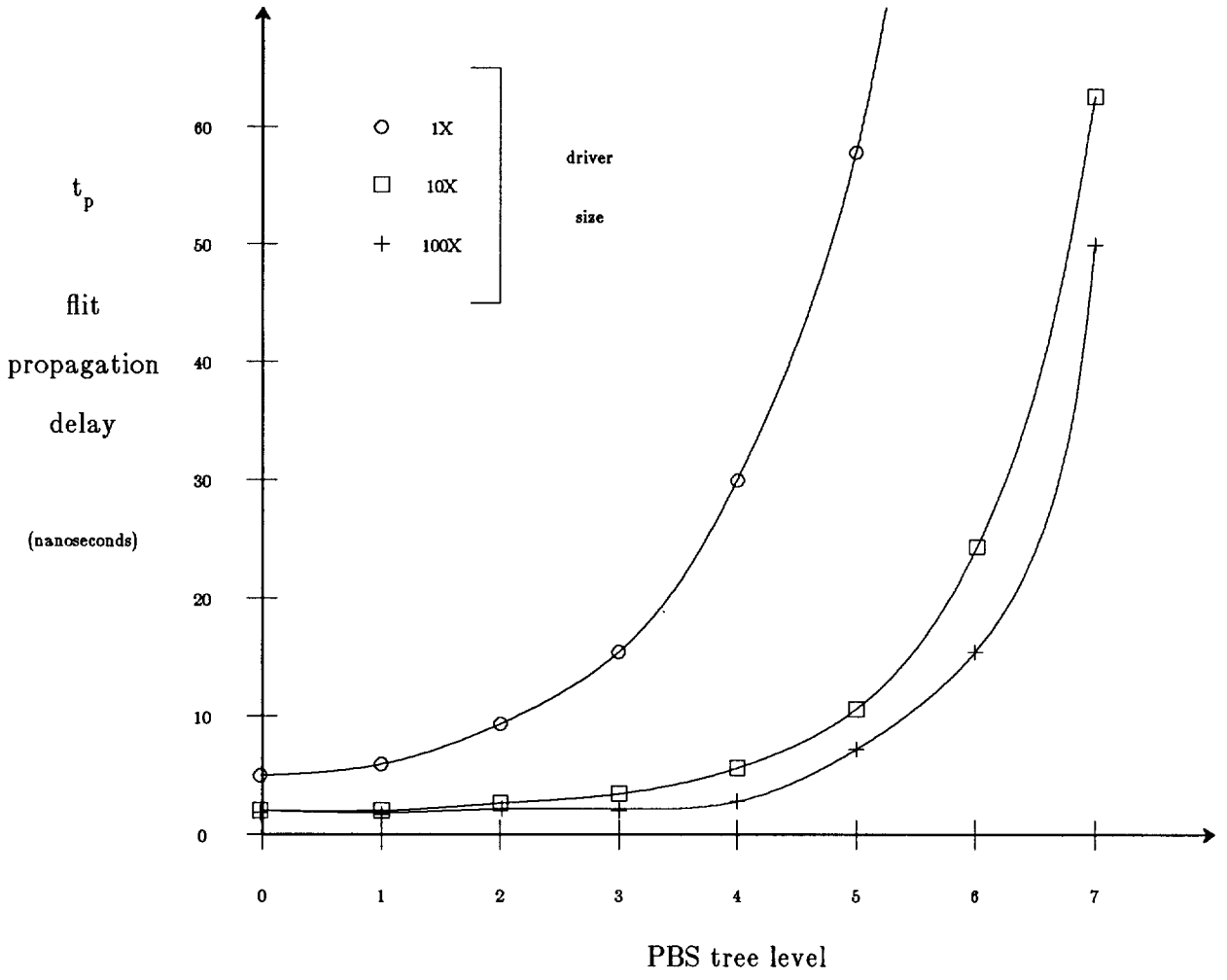


Figure 6: Flit Propagation Delay vs Transfer Level

Figures

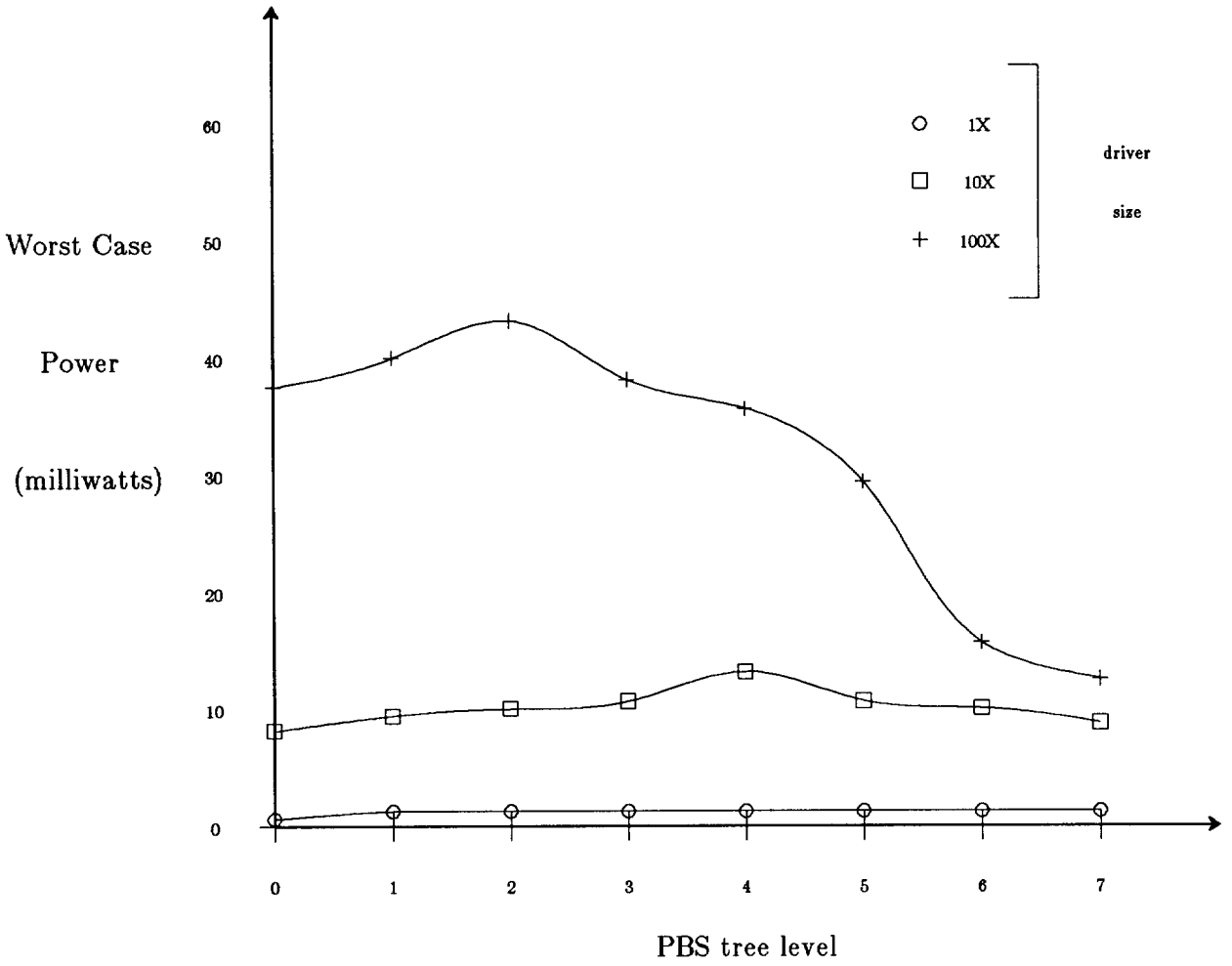


Figure 7: Worst Case Propagation Power vs Transfer Level

Figures

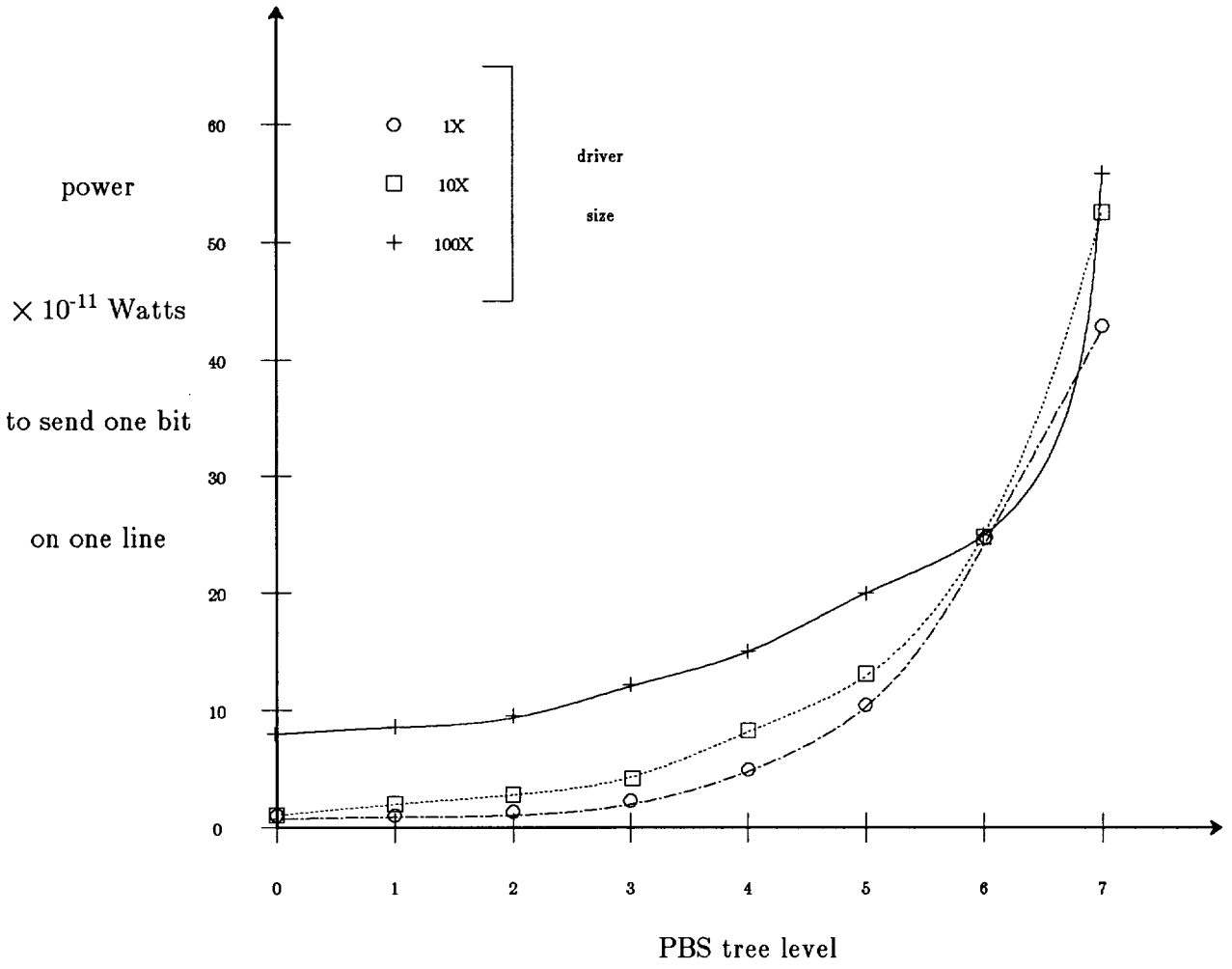
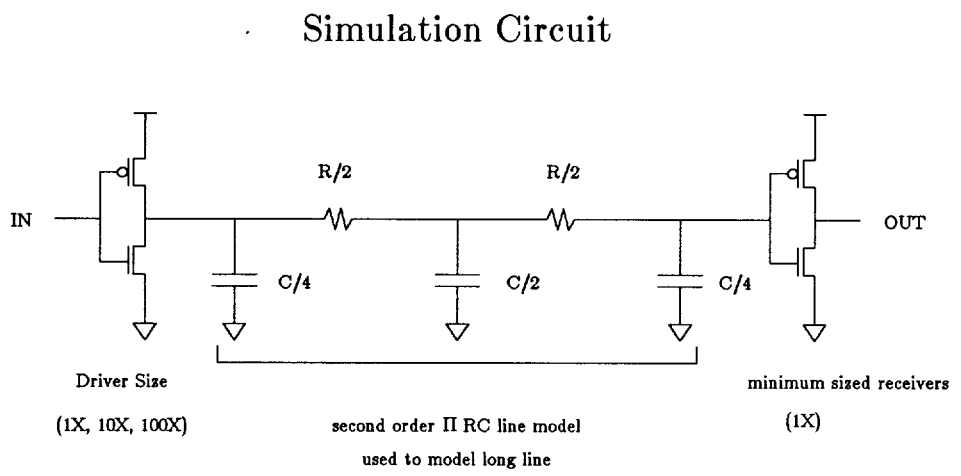


Figure 8: Single Bit Power vs Transfer Level

Figures



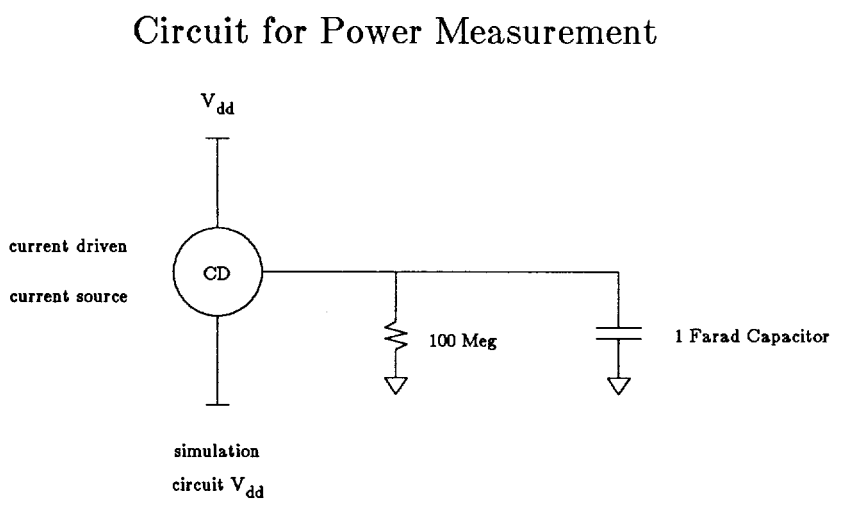


Figure 9: Spice Test Circuits

Figures

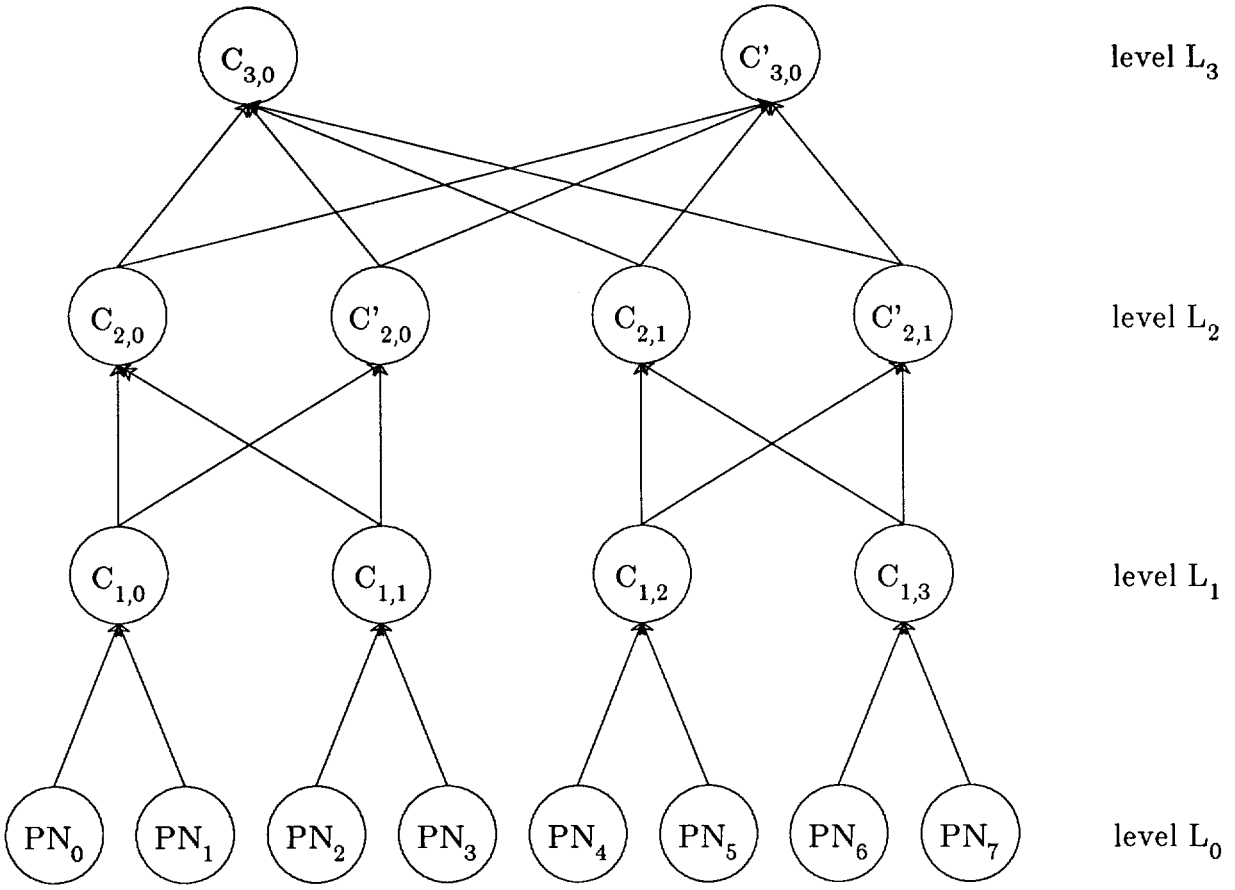


Figure 10: D_3 Concentrate Tree with Redundant Upper Level SNs

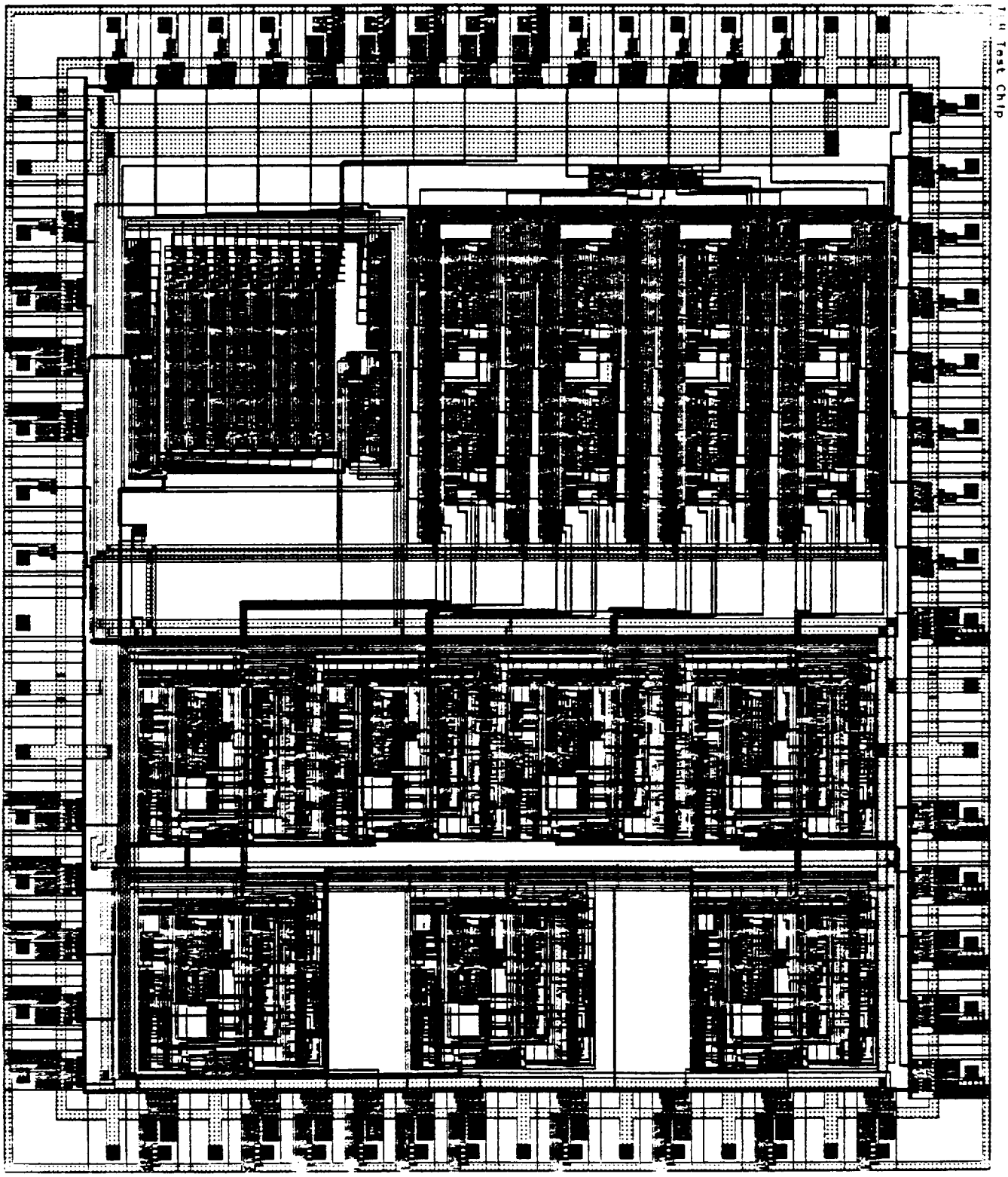


Figure 11: PBS Test Chip Layout

Figures

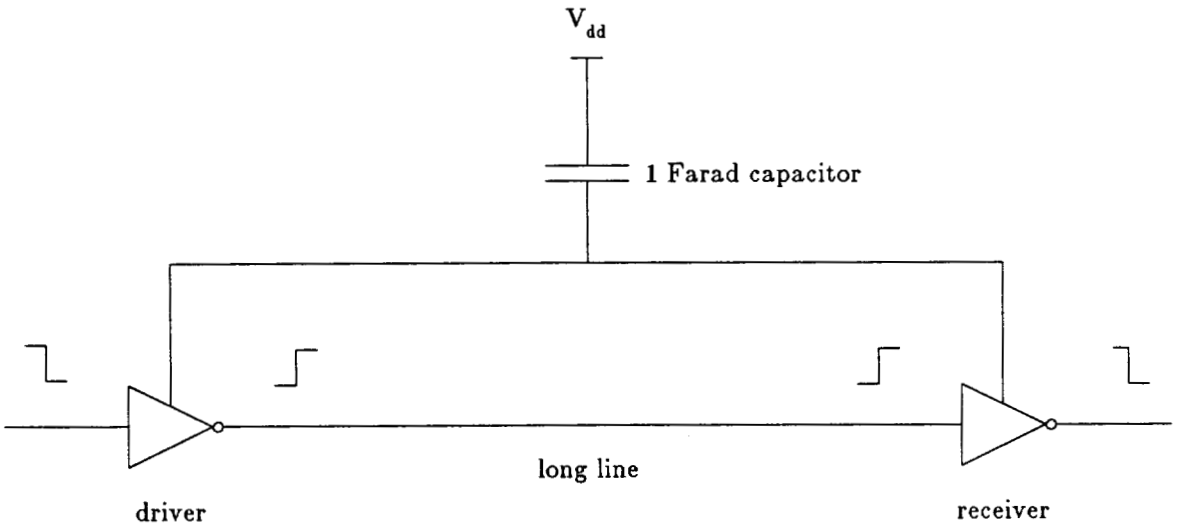


Figure 12: Spice Propagation Power Test Circuit