Sharing inter-module channels

We note that in all the networks described above, there are two networks: a request network that routes access requests from the processors to the memories, and a response network that routes replies from the memories to the processors. We can merge these two networks so that there is only one pair of channels going between any two modules. We can think of this network as being complementary to the previous network that dilated the inter-module channels.

In order to share the inter-module channels, we need an additional set of routing nodes that reduce the 64 channels of the Butterfly topology to 32 channels. We put these 2x1 merger nodes just before the out-going channels we get the network organization shown in figure 4.4. We call this network Shared-Butterfly. The inter-module channels are 16 bits wide, therefore we only consider topologies Shared-Butterfly.16.16, Shared-Butterfly.16-2dil.16 and Shared-Butterfly.32.16.

In this network the sharing of channels between access requests and responses makes this network susceptible to message deadlock. It is possible to have situations where message buffers get filled up and a circular wait situation gets created. In order to eliminate the possibility of deadlock, we assume that there are very large buffers at the memory units and at the processors. These large buffers guarantee that messages will be drained out of the network either at the processors or the memories. This eliminates the possibility of deadlock. The MIT Alewife machine uses a similar network architecture in which channels are shared between requests and replies. The technique to avoid deadlocks is also similar; when the buffers fill up, messages are emptied from the network into local memory, effectively simulating infinite buffers [A$^+$91].

Cross-bar

To establish an upper bound on the best performance achievable, we also consider the networks Xbar.8.8, and Xbar.32.8. Both networks connect the processors and memories to the inter-module channels using 32 by 32 cross-bars. In the first case both the input and output channels of the cross-bar are 8 bits wide. In the latter case the channels that connect to the processors and memories are 32 bits wide. The inter-module channels remain 8 bits wide as before.

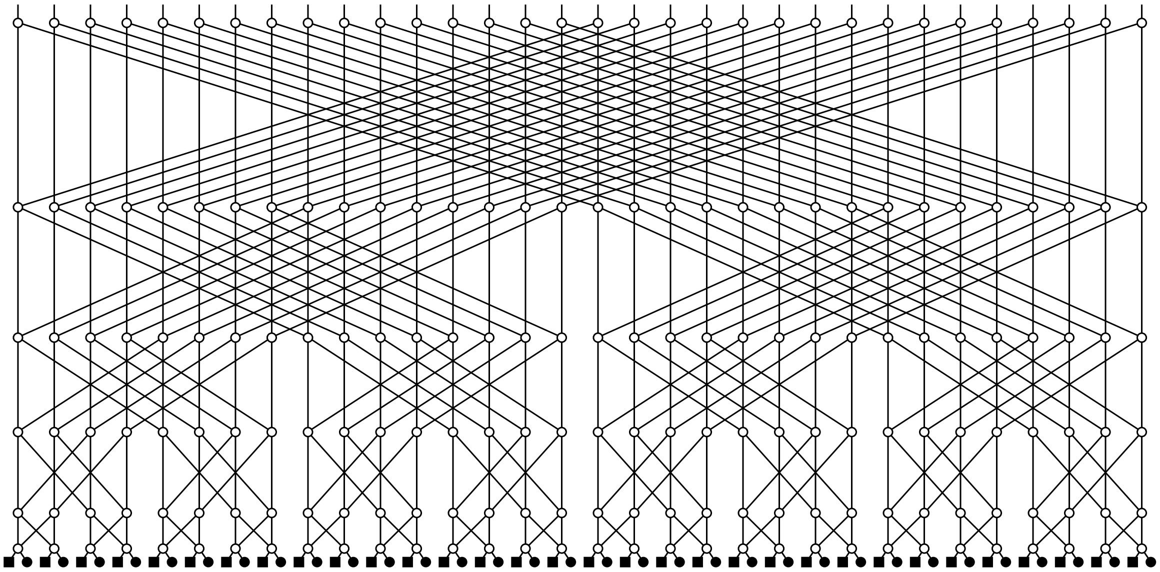

Figure 4.4: Intra-Module Network: Shared-Butterfly

4.3.2 M=64 Modules

We now consider the case where the packaging modules are smaller. We assume that each packaging module is capable of holding 16 processors, with a total of 64 modules for a 1024 processor system. The interconnection between the modules is a complete graph.

To make the comparisons fair, we keep the bisection width the same as before. Since there are 64 packaging modules, a bisection width of $B = 8192$ implies that pin-count per module is equal to $4B(M-1)/M^2 = 504$, for a total of $64 * 504 = 32256$ pins. Each pair of modules is connected by a bundle of 8 wires. Each bundle has 4 channels and each channel will be 2 bits wide.

The first intra-module network we consider is Butterfly.8.2 and is shown in figure 4.5. We build a butterfly on 16 inputs but with a degree 8 fan-out in the last column instead of degree 2. The degree 8 fan-out is required to generate enough channels to connect to all the modules. To balance the bandwidth at the last column of switching nodes, we have to provide 8-bit wide intra-module channels.

We also consider the network Butterfly.32.2, which has 32 bit wide intra-module channels.

To establish an upper bound, we also consider the network Xbar.32.2 in which a cross-bar is used to connect 16 processors to 64 inter-module channels.

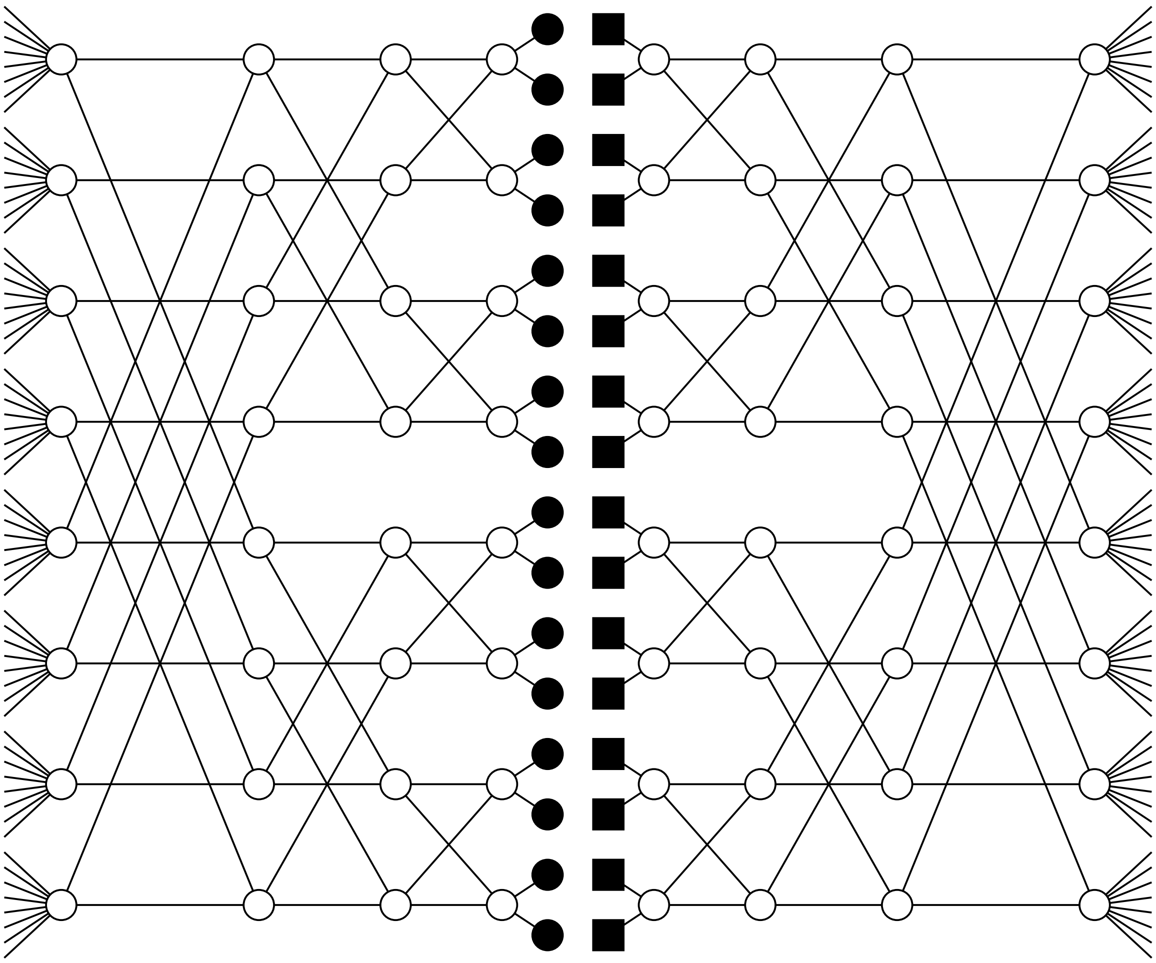

Figure 4.5: Intra-Module Network: Butterfly.8.2

4.3.3 M=16 Modules

Next, we consider the case where the packaging modules are larger. We assume that each packaging module is capable of holding 64 processors, requiring 16 modules for a 1024 processor system. The interconnection between the modules is a complete graph as before.

We derive this network by partitioning a butterfly with 256 inputs in a manner similar to the earlier case Section 4.3.1. Since we have to build a network on 1024 processors, we connect 4 processors to each input of the 256 input butterfly, using a 2 level tree, as shown in figure 4.6.

As before, we hold the bisection width constant at 8192. Since there are 16 packaging modules, a bisection width of $B = 8192$ implies that pin-count per module is equal to $4B(M-1)/M^2 = 1920$, for a total of $16 * 1920 = 30720$ pins. Each pair of modules is connected by a bundle of 128 wires. Each bundle has 4 channels and each channel is 32 bits wide.

Since the inter-module channels are 32 bits wide, we get a uniform network if we make the intra-module channels also 32 bits wide. This is the first network we evaluate and as per our naming conventions, we call this network Butterfly.32.32. We also consider the network Butterfly.128.32, with 4 times wider intra-module channels.

To establish an upper bound, we also consider the network Xbar.32.32 in which a cross-bar is used to connect 64 processors to 16 inter-module channels.

4.4 Simulation Results

4.4.1 Open-network model

Figures 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, and 4.13 show the results for the different networks. The latencies are round-trip latencies, i.e., the time taken for the access request to go from the processor to the memory, make the access and return to the processor. The x-axis is the normalized load expressed as a percentage of the peak load.

We see that in all the open-network graphs, as the applied load increase, the latency initially increases gradually and as the network nears saturation, the latency increases rapidly. The curves extend along the x-axis until the point at which the corresponding generation rate no longer yielded a stable latency. We defer detailed discussion of these

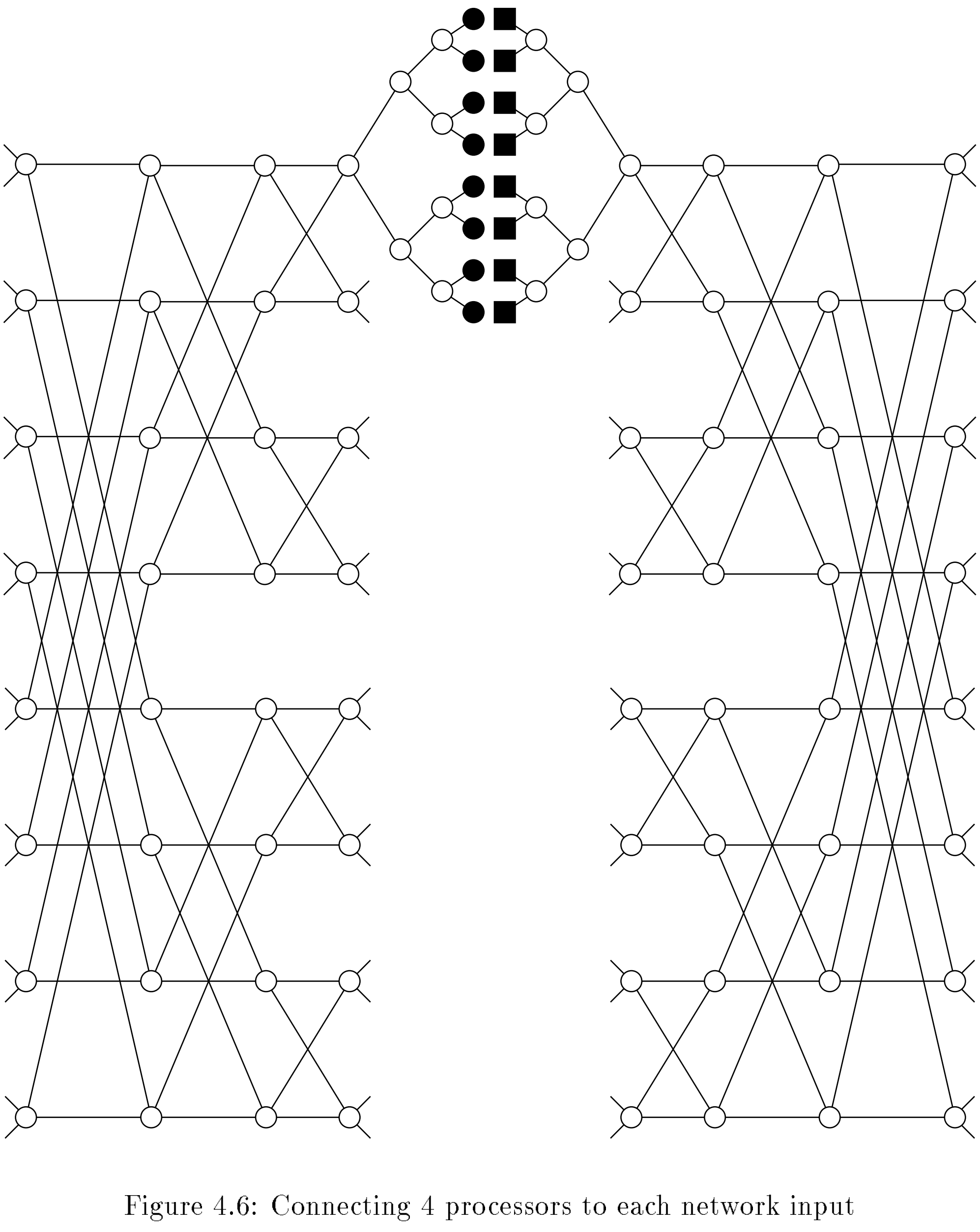

Figure 4.6: Connecting 4 processors to each network input

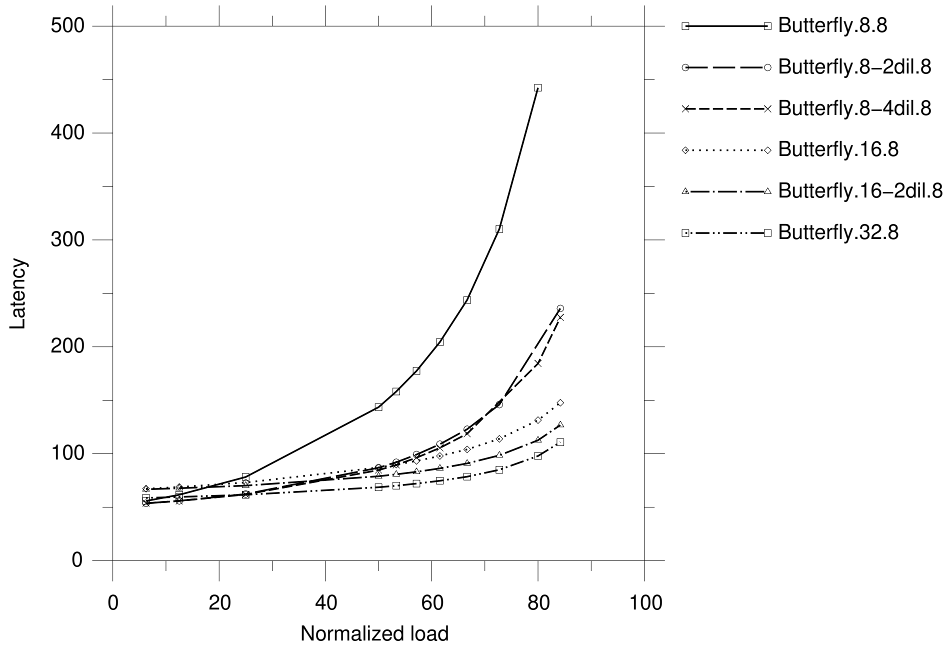

Figure 4.7: Open-Network Model; Butterfly Networks

results to section 4.4.3.

4.4.2 Multithreading work-load model

The performance metric under this work-load model is the average processor utilization, as a function of number of threads and of the rate at which threads make memory accesses. Based on the bisection width of the networks we can see that the peak bandwidth that each network can support is 8 bits per processor per cycle in each direction. This corresponds to a mean access rate of one every 16 cycles, since each access is 128 bits long. We consider mean access intervals of both 16 and 32 cycles.

A mean access interval of 16 cycles can potentially saturate the bisection and corresponds to running the network at 100% load. Since message insertion is linked with the delivery of responses, injecting messages with the mean access interval = 16 does not pose a problem. The processors themselves merely slow down until the injection rate matches the response rate. As the number of threads per processor increases, the processors become increasingly latency tolerant and are able to get more out of the network.

A mean access interval of 32 cycles corresponds to running the network at about 50% load. In this case, since the network is lightly loaded, the processors should be able

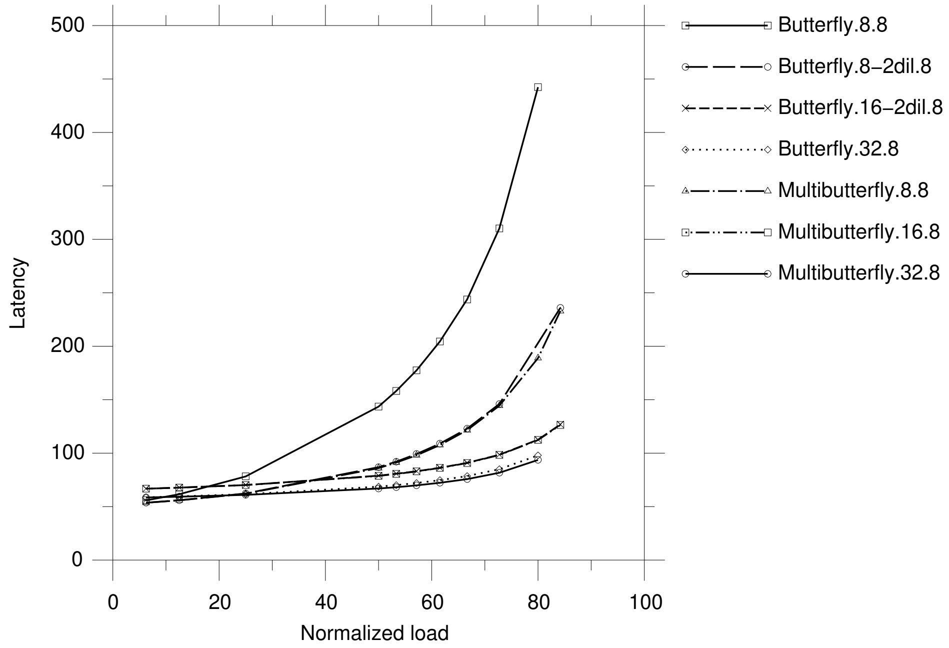

Figure 4.8: Open-Network Model; Multibutterfly Networks

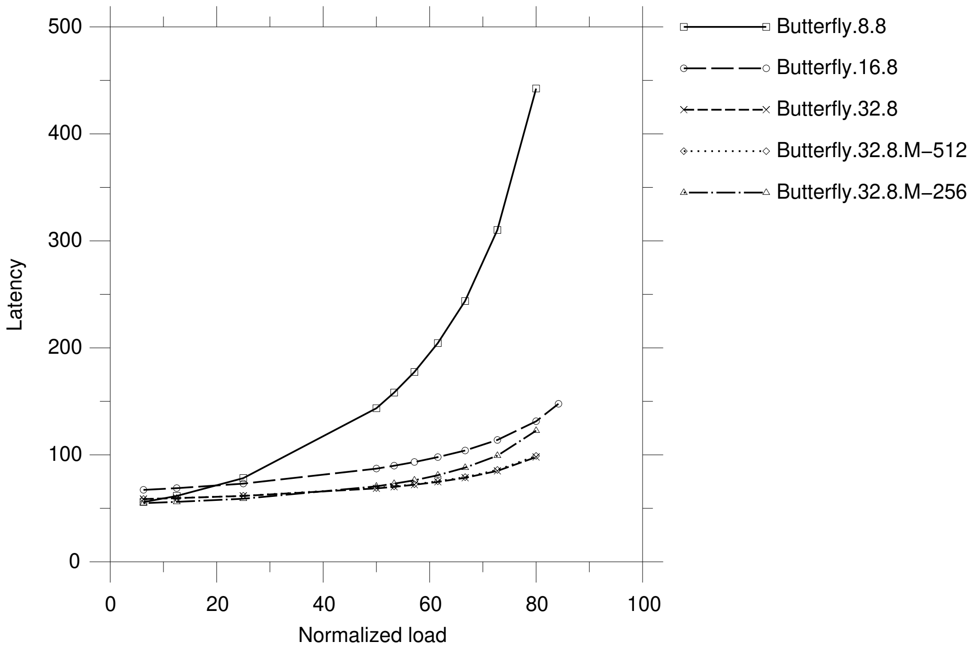

Figure 4.9: Open-Network Model; Fewer Memories

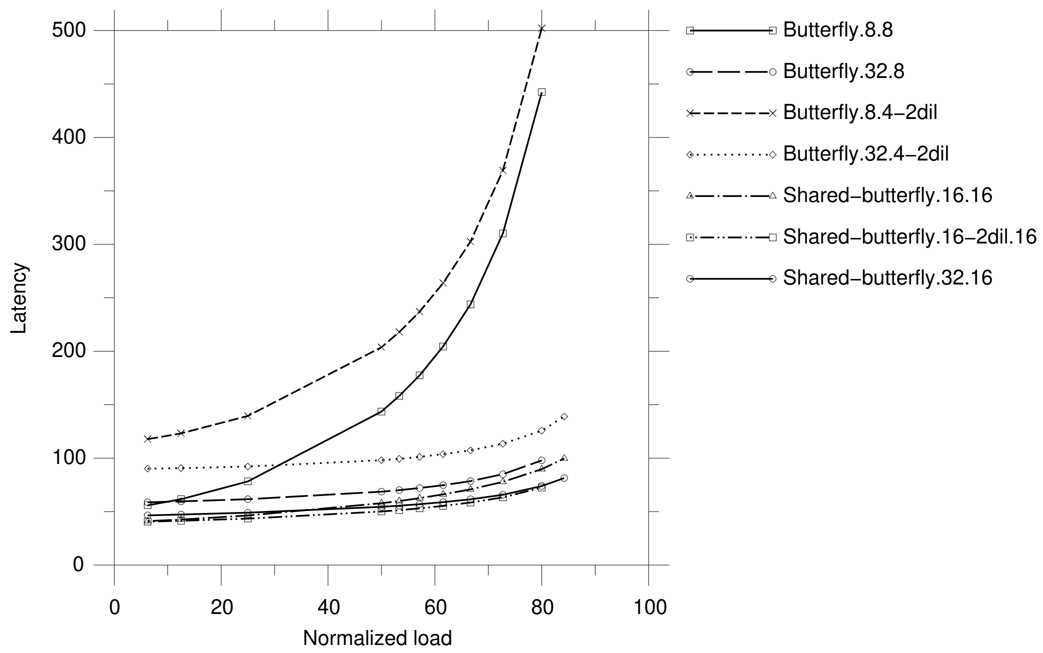

Figure 4.10: Open-Network Model; Sharing/Dilating Inter-module Channels

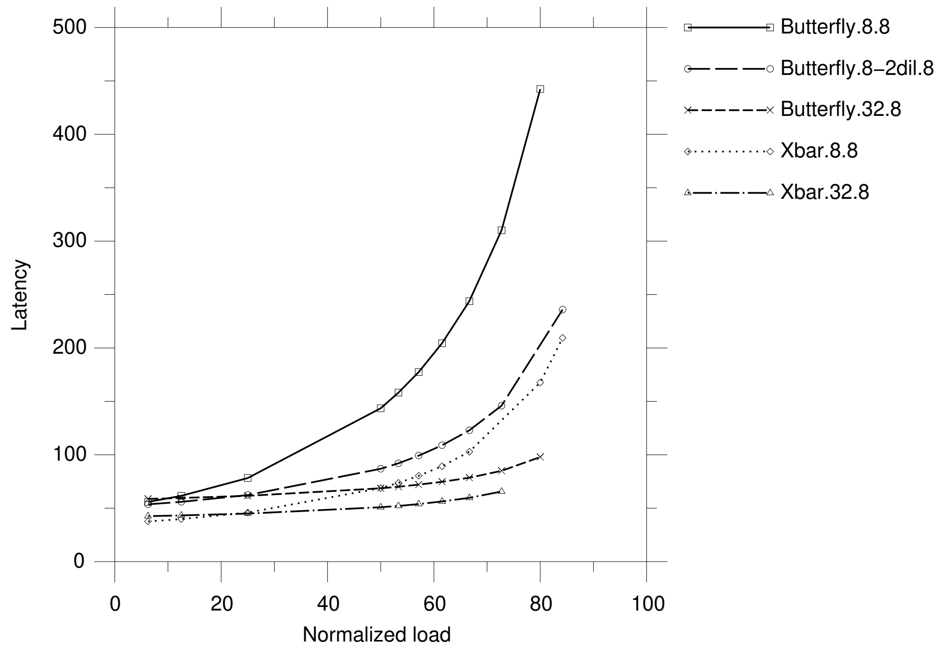

Figure 4.11: Open-Network Model; Cross-bar Networks

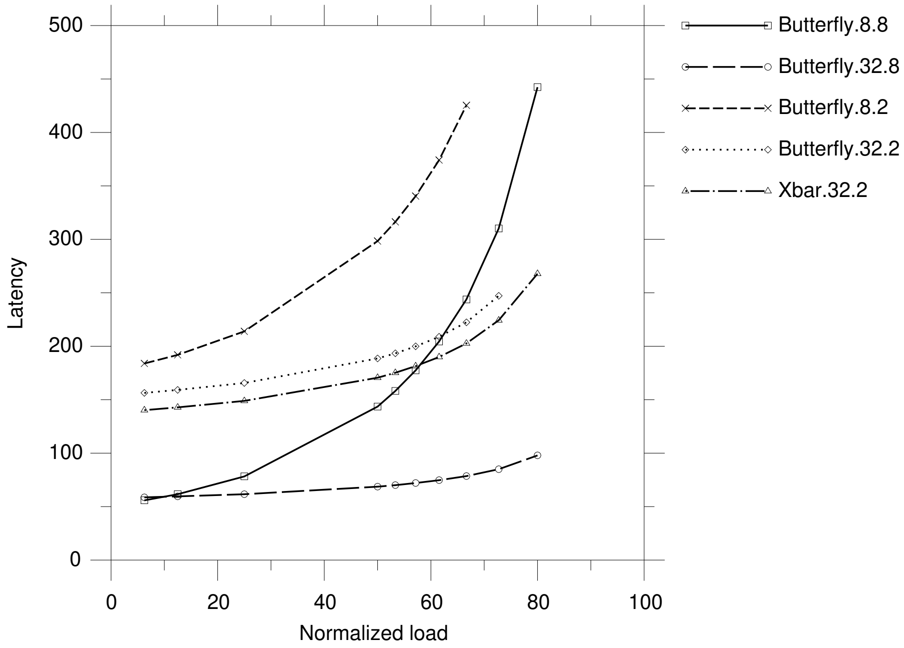

Figure 4.12: Open-Network Model; 64 Module Networks

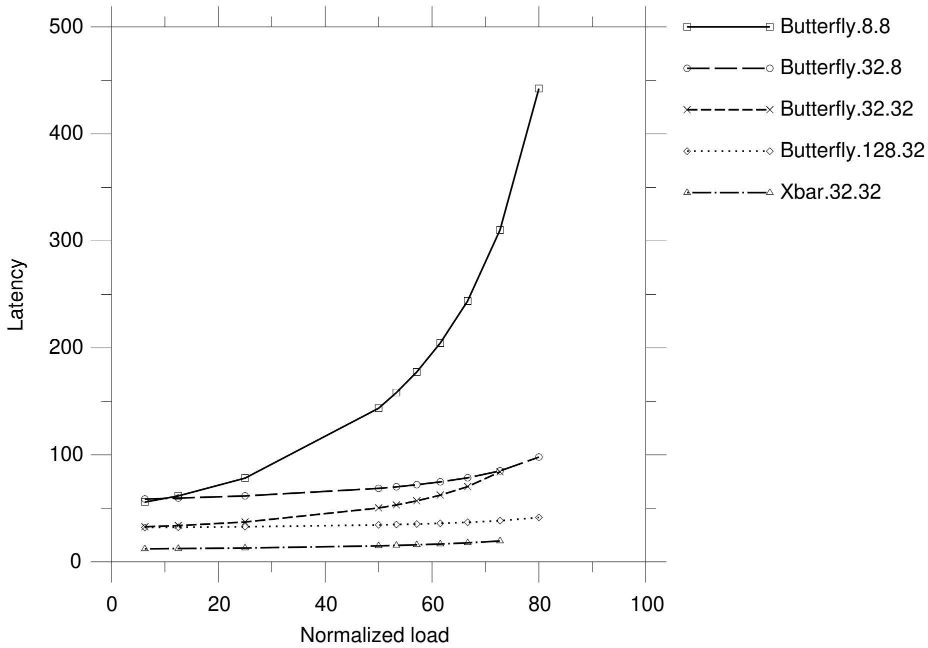

Figure 4.13: Open-Network Model; 16 Module Networks

to achieve almost 100% utilization if they can successfully hide the network latency using multithreading.

Figure 4.14 and 4.15 show the processor utilizations for the different networks when the mean access interval is 16 cycles. The graphs shows five lines corresponding to multithreading factors of 1, 2, 4, 8 and 16.

Figures 4.16 and 4.17 show similar curves for a mean access interval is 32 cycles.

The graphs for both geometric and periodic access patterns are similar to each other, except that the utilizations for the periodic model are higher. This is to be expected because, under the geometric model, there is a greater variance in inter-access times, and if many short inter-access intervals occur together, the instantaneous load increases and the processor utilization gets reduced.

4.4.3 Discussion of results

Dilation versus wider channels

The data corresponding to Butterfly.8.8, Butterfly.8-2dil.8 and Butterfly.8-4dil.8 in figures 4.7, 4.14 and 4.15 indicate that dilation of channels helps in improving performance. This improvement results from the reduced contention for the intra-module channels. Increasing the dilation from 2 to 4 does not seem to have much of an effect, indicating that a dilation factor of 2 is sufficient to eliminate most of the intra-module contention.

As opposed to dilation, widening the channels within a module affects the network performance in three different ways. First, it reduces the contention for intra-module channels. Secondly, it increases the effective memory service rate. Dilation does not increase the effective memory service rate because we do not dilate the channels to the memory units(Section "Dilation" in 4.3.1). Thirdly it increases message latency because of the non-uniformity in channel widths[3]. It is clear that the first effect has a positive influence on network performance. The second effect also helps improve performance because conflicts between accesses at the memories have less of an effect. The third effect hurts network performance because it increases the latency.

When the load is small (in the open-network case) or there are few threads (multithreading), the performance of the networks with wider channels is slightly worse than

[3]When a message transits through a node whose output channel is wider than its input channel, the message suffers a larger delay because it must be entirely buffered before it can be forwarded(Section 3.4).

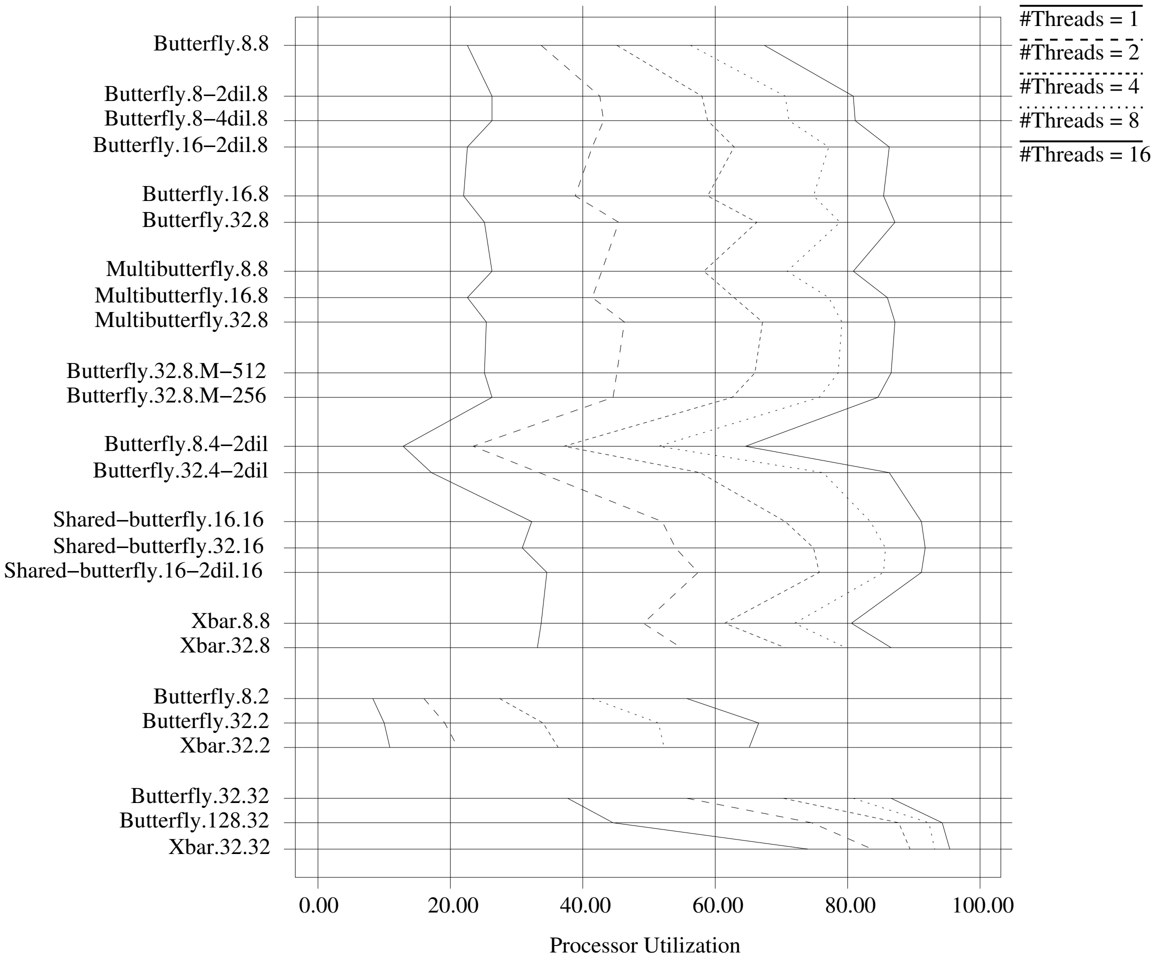

Figure 4.14: Periodic Access Model, access interval = 16 cycles

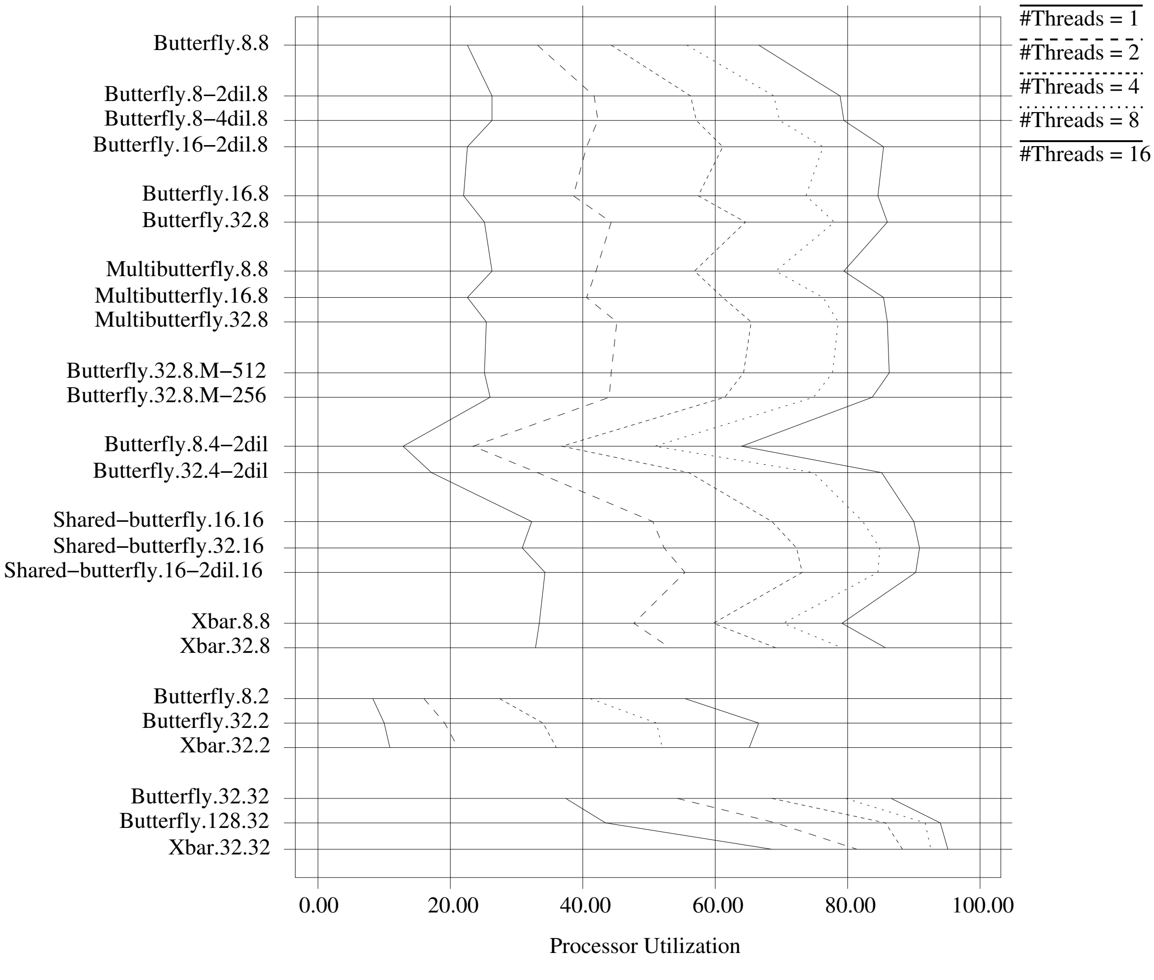

Figure 4.15: Geometric Access Model, mean access interval = 16 cycles

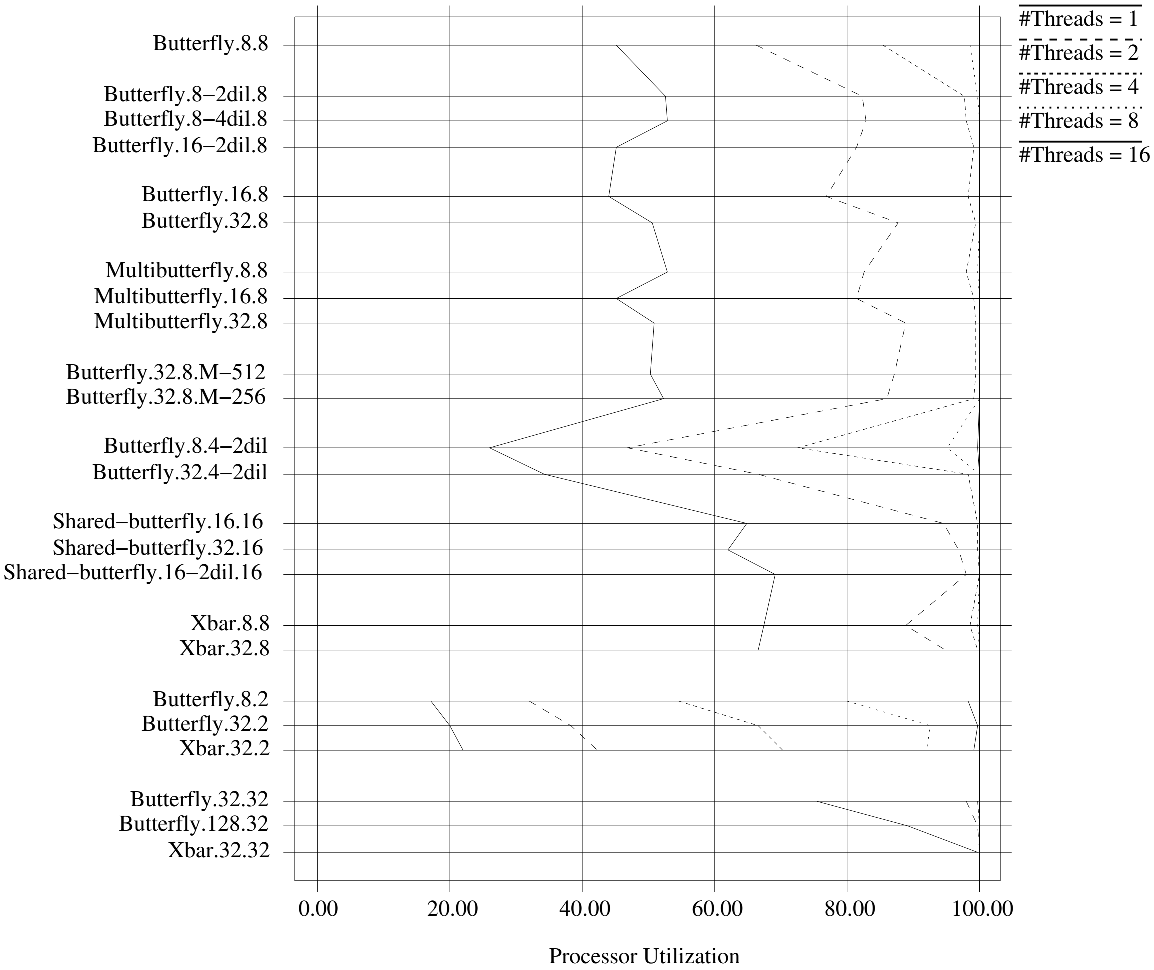

Figure 4.16: Periodic Access Model, access interval = 32 cycles

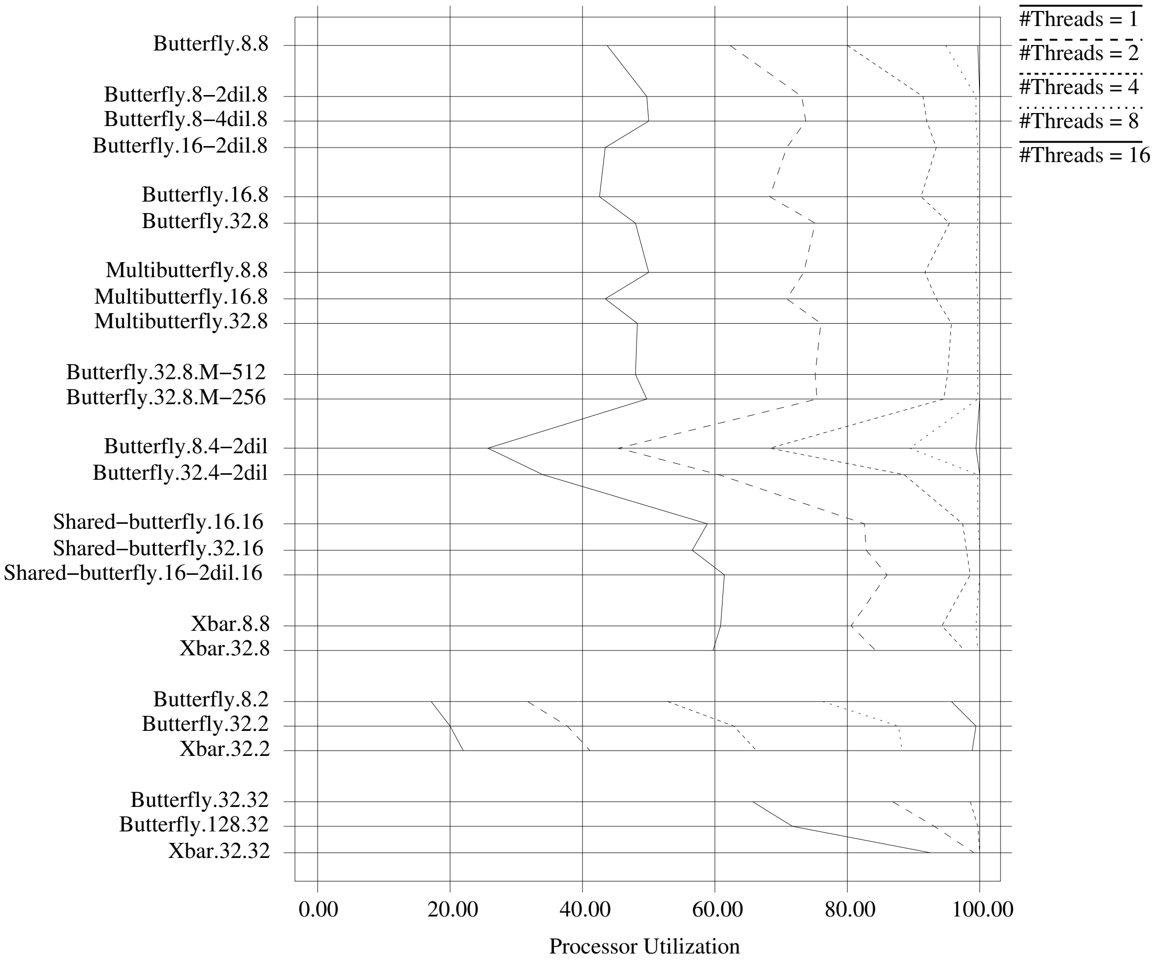

Figure 4.17: Geometric Access Model, mean access interval = 32 cycles

Butterfly.8.8. This is because of the third effect mentioned above. But, when the load on the network is increased, the effects of reduced contention and the higher memory service rate become more predominant. In the long run we can expect all memories to receive approximately the same number of requests because the access distribution is random, but in the short run the access distribution can be uneven. A higher memory service rate helps to reduce the effect of this non-uniformity.

Combining dilation and widening of channels does not seem to help much either, versus simply widening the channels as seen by comparing Butterfly.16-2dil.8 and Butterfly.32.8.

We can also try to compare the effects of dilation versus widening channels analytically using queueing theory. Dilation corresponds to using multiple servers with the same service rate, while widening channels corresponds to increasing the service rate of a single server. If we model the traffic between a pair of routing nodes using a simplistic queueing model consisting of poisson arrival streams and exponential service times, we can show that both dilation and wider channels have comparable performance when the load factor is high, but wider channels are better at lower loads. It is difficult to extend the analysis from a single pair of nodes to the entire network. Further, we need to take into account a variety of factors such as constant service times, bounded queues, packet-switched routing in some nodes due to non-uniform channel widths, contention at the memory units, stalling of processors when all threads have outstanding accesses, etc.

Multibutterfly versus Dilation

The multibutterfly networks require the same number of wires as the 2-dilated networks. The difference between them is the randomized wiring between stages in the multibutterfly. Figure 4.8 and the graphs corresponding to the multithreading workload indicate that the multibutterfly has approximately the same performance as a 2-dilated butterfly. Specifically, Multibutterfly.8.8 has approximately the same performance as Butterfly.8-2dil.8, and Multibutterfly.16.8 has the same performance as Butterfly.16-2dil.8. Simulations of large multibutterflies have shown that they perform better than dilated butterflies [LLM90, KU91], but the multibutterflies in our networks are too small to display any noticeable difference in performance. Since we use multibutterflies only for the intra-module network, they only have 32 inputs and five stages. For networks of this size, there is hardly any benefit from using randomized connections between stages.

Fewer memories

Butterfly.32.8.M-256 has the same amount of memory bandwidth as Butterfly.8.8, since the channel width to memory is 4 times as wide and there are 4 times as fewer memory units. Similarly Butterfly.32.8.M-512 has the same memory bandwidth as Butterfly.16.8. Comparing these pairs of networks illustrates that using fewer memory units with wider channels, results in a significant improvement in performance. The reason for this improvement in performance is the increased memory service rate at each memory unit. Even though a reduction in the number of memory units increases the load on each unit, temporary non-uniformities in the load distribution have less of an effect due to the higher service rate.

Instead of comparing networks based on equal memory bandwidth, we can also compare them based on identical memory channel width. We see that the performance of Butterfly.32.8 is indistinguishable from that of Butterfly.32.8.M-512 and Butterfly.32.8.M-256. The effect of the lower memory bandwidth is not felt since network bandwidth is the main limiting factor. Figure 4.9 indicates a slight increase in latency as the load increases because of increased contention for the fewer memory units, but this increase in latency is small and does not appear to affect the processor utilizations under the multithreading workload model.

Dilating versus sharing inter-module channels

The graphs indicate that dilating the inter-module channels causes a significant drop in performance, while sharing the inter-module channels improves performance over the base network. The inter-module channels are the bottlenecks as far as network throughput is concerned. A network that achieves a higher utilization of the inter-module channels will, therefore, achieve higher throughput. When we configure the available inter-module bandwidth as a pair of narrower channels (Butterfly.32.4-2dil), it is more difficult to achieve high channel utilization because we need twice as many independent messages to keep all the wires busy. In contrast, when we share the channels we need only half as many messages. This explains the difference in performance. However, sharing the inter-module channels between the forward and the return networks makes the network susceptible to deadlock and we need to devise some mechanism to avoid deadlock (We assumed that large buffers were available at the processors and the memory units).

Cross-bar

The intra-module cross-bar network completely eliminates the possible contention for links within a module. It also has a lower minimum latency because we assume that it imposes only a single cycle pipeline delay as opposed to one cycle per stage in a multi-stage network.

In figure 4.11 we see that the graph for Xbar.8.8 is approximately parallel to that of Butterfly.8-2dil.8 indicating that both networks have similar performance with respect to contention. The graph for the cross-bar network is lower because of its lower minimum latency. This difference does not show up under the multithreading workload (figures 4.14 and 4.15) indicating that the effect of this difference in latency is small. The comparison between Xbar.32.8 and Butterfly.32.8 is similar. This indicates that providing extra local bandwidth either through dilation or wider channels is successful in eliminating almost all the contention for the intra-module channels.

Multithreading

It can be seen that all networks achieve significant improvement in processor utilization as the number of threads is increased. This demonstrates that multithreading is successful in masking the access latencies. Increasing the number of threads per processor corresponds to operating the network farther down the x-axis on the load-latency graphs of the open-network model. Though this increases the latency experienced by the messages, the processor is able to better overlap the latency with useful computation. It is important to note that even when the number of threads is small(2-4), there is a significant improvement in processor utilization in relation to the utilization corresponding to a single thread. As we increase the number of threads further, the processor utilizations improve, but the marginal increase per added thread diminishes because increasing the number of threads also increases the loading of the network which increases latency.

Performance at low network loads

The graphs corresponding to an inter access time of 32 cycles (figure 4.16 and 4.17) show that at high levels of multithreading, it is possible to get processor utilizations of almost 100%. Since the load never gets so high as to make the latencies increase dramatically, multithreading can completely hide the network latency. For most of the networks we evaluated we can see that about 4 threads can achieve this (The networks corresponding to $M = 64$ needed about 16 threads to achieve 100% utilization since their latencies were high). Naturally, as we scale the machine to more processors we expect that we will need more threads, but we expect that the number of threads will not grow faster than $\log N$ since the network latencies under cut-through routing will grow slower than $\log N$.

We also see that at low levels of multithreading, the effects due to non-uniform channel widths are more pronounced. Since the network load is small, the increase in latency due to non-uniform channel widths dominates the effects due to reduced contention and increased memory service rates.

4.4.4 Choosing module size

We can also compare the results of the cases $M = 16$ and $M = 64$ with $M = 32$, but in doing so we should keep in mind that the costs of these networks are not the same, because they use different numbers of modules and different pin-counts per module. Since all the networks have the same peak throughput, the difference in cost vectors illustrates a trade-off between the number of modules and the pin-count per module. Reducing the number of modules by a factor of 2 requires increasing the pin-count per module by the same factor of 2, in order to maintain the same peak throughput.

Our results in comparing the performance across different module sizes are similar to the results of Cypher [Cyp90] in that the performance of the network is proportional to the product of the number of modules and the pin-count per module. This is to be expected since the cost model used in this chapter is similar to Cypher's pin-requirements model. However, the network design problem we consider here is substantially different from the one considered by Cypher.

Although the costs are different for different module sizes, since the peak throughput is the same, we can compare the fraction of peak throughput that each of these networks achieves. We see that using larger modules leads to better performance than using smaller modules. When we increase the module size by a factor of 2, we have a complete graph on half as many modules. This means that number of channels leaving a module drops by a factor of 2. Also the pin-count per module increases by a factor of 2, making the inter-module channels 4 times as wide. We already observed that we were able to improve performance when we organized the same number of inter-module wires as fewer wider channels. The same effect is observed in a more pronounced manner when we vary the module size. This explains the difference in performance between $M = 64$, $M = 32$, and $M = 16$.

4.5 Summary

We started with a simple model of the packaging hierarchy that consisted of only two levels and used a simple network cost model that had only two parameters; the number of top-level modules and the top-level pin-count. We fixed the network cost by fixing both the parameters and compared different networks in terms of their performance. We showed that for a fixed pin-count, the best network at the top level of wiring was a complete graph interconnect where each module was connected to every other module. At the lower level the packaging hierarchy, we examined several alternatives. We did not exhaustively examine all the possible choices at the lower level. However, we demonstrated that simple butterfly networks with wider channels at the lower level were able to match the performance of the best possible network (cross-bar), indicating that exploiting *packaging locality* can lead to better networks. Our results, therefore, argue in favor of designing networks on the basis of the availability of packaging resources, rather than designing networks first and packaging them later.

Chapter 5

Model 2: Pin-counts and bundles

In this chapter, we consider a model that is slightly more complex than model 1. In the previous chapter, we saw that the complete graph topology had the maximum performance for a fixed pin-count per module. However, the complete graph also had the maximum number of bundles per module of all the topologies considered in the previous chapter. In many packaging technologies, a complete graph interconnect may be considered expensive because it uses many bundles. For example, if the technology used in the interconnection of modules is cables with connectors at each end, we may have fixed per-cable and per-connector costs, independent of the total number of wires in the cables and connectors. Under such a packaging technology a network in which the wires leaving a module are organized into a smaller number of thicker bundles, will be cheaper than one in which the same number of wires are organized as a larger number of thinner bundles. In order to model this difference in costs, we include the number of bundles per module in the cost vector of Model 2.

Recall that model 2 is based on the same two level hierarchy as model 1 (inter-module and intra-module connections), but the cost vector has three dimensions: the number of modules, pin-count and number of bundles per module. As in model 1, we ignore the costs of the intra-module network.

5.1 Inter-module Networks

Under model 1, we compared the performance of different inter-module networks after fixing the cost vector. Similarly, under model 2, we first compare networks for a fixed cost vector (Section 5.1.1). However, this approach does not illustrate the trade-offs between the different components of the cost vector. To examine these trade-offs, we compare the cost vectors of different networks of the same estimated performance (peak throughput). For this comparison we shall hold the number of modules constant and compare networks in terms of pin-count and number of bundles. We shall see that reducing the number of bundles per module requires each module to have a larger pin-count (Section 5.1.2).

5.1.1 Fixing all three parameters

We showed in Chapter 4 that for a fixed pin-count per module, the peak throughput of a network is inversely proportional to h, the average number of hops made by a message over the inter-module network. This result is valid even under model 2, but under model 2 we also fix the number of bundles per module. Therefore, to maximize the performance, we have to minimize the number of hops subject to the bundling restriction. Clearly, if the number of bundles per module is greater than or equal to $M-1$, we can minimize h by building a complete graph interconnect between the modules, as we did in Chapter 4. The problem we consider in this section is: How do we minimize h, given a tighter restriction on the number of bundles?

A related problem that has received much attention in graph theory is the (Δ, D) graph problem [BDQ86]. The (Δ, D) graph problem consists of finding the largest graph in which the node degree is Δ and the diameter is D[1]. In the graph theoretic sense, the number of bundles per node is the same as its degree, and the diameter of a graph is usually an upper bound on h. While the (Δ, D) problem is concerned with finding the graph with the largest number of vertices for a fixed D, we are interested in the complementary problem, namely finding the smallest h for a fixed number of modules. For most of the interesting graphs (graphs that have small D for fixed Δ and a given number of nodes), h is usually quite close to D. Therefore, a solution to the (Δ, D) problem and a solution to the problem we are interested in, are closely related.

However, it is well known that the (Δ, D) problem is difficult [BDQ86]. For many values of Δ and D, the (Δ, D) problem is still open. We can show by means of a simple counting argument that the number of nodes in a graph with degree Δ and diameter D, cannot exceed $1 + \Delta + \Delta(\Delta-1) + \Delta(\Delta-1)^2 + \cdots + \Delta(\Delta-1)^{D-1} = \frac{\Delta(\Delta-1)^D - 2}{\Delta - 2}$. This

[1]The diameter of a graph is the maximum distance over all pairs of vertices, where distance between two vertices is defined as the shortest path between them.

bound is due to E. F. Moore (circa 1958). For some small values of Δ and D there exist graphs that achieve the Moore bound, but for other values, the largest known graphs are much smaller than the Moore bound. In some of these cases, researchers have been able to prove that the largest known graphs are optimal, but for a large number of other cases, we do not know if larger graphs exist.

The apparent complexity of the (Δ, D) problem suggests that the problem we are interested in is also complex. Therefore, instead of trying to find a network that achieves the *maximum* performance, we shall try to find a network that is close to the optimal.

One possibility is to search for a network whose performance is off by a small constant factor from the optimal. Consider a network on M nodes with diameter D and degree Δ. Let the optimal degree be Δ_{min}, when the diameter is D, and the optimal diameter be D_{min} when the degree is Δ. In other words, (M, D, Δ_{min}) and (M, D_{min}, Δ) satisfy the Moore equation. Further, let us assume that $\Delta = c\Delta_{min}$ where c is a small constant. For a fixed pin-count and number of bundles, we have already argued that, to a first approximation, the performance is inversely proportional to the diameter. Therefore, the network with parameters (M, D_{min}, Δ) is optimal, because any other network on M nodes and degree Δ has a greater diameter. Further, the performance of the optimal network is greater than the performance of the network with parameters (M, D, Δ) by a factor of D/D_{min}, the inverse ratio of the diameters.

Since (M, D_{min}, Δ) and (M, D, Δ_{min}) satisfy the Moore equation, $M \approx \Delta^{D_{min}}$ and $M \approx \Delta_{min}^{D}$. Therefore, $D/D_{min} \approx \log \Delta / \log \Delta_{min}$. But we assumed that $\Delta = c\Delta_{min}$. Therefore, $\frac{D}{D_{min}} \approx 1 + \frac{\log c}{\log \Delta_{min}}$. This implies that so long as Δ is within a small constant factor of Δ_{min}, the performance of the network will be very close to the optimal network with the same pin-count and number of bundles (Network with parameters (M, D_{min}, Δ)).

One family of networks for which $\Delta = c\Delta_{min}$ is the family of products of complete graphs. A product graph is defined as follows: Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The product of G_1 and G_2, denoted by $G_1 \times G_2 = (V, E)$ where $V = V_1 \times V_2$ and $E = \{((u_1, u_2), (v_1, v_2)) | ((u_1 = v_1) \wedge (u_2, v_2) \in E_2) \vee ((u_2 = v_2) \wedge (u_1, v_1) \in E_1)\}$. We call a product of two complete graphs a 2-hop network. In general an n-hop network is a product $G_1 \times G_2 \times \cdots \times G_n$, where each G_i is a complete graph. For a fixed total number of nodes, the number of bundles needed by an n-hop network is minimized when each of the G_is is of the same size, i.e., a complete graph on $\sqrt[n]{M}$ nodes. We use the term n-hop network to refer to a product graph of complete graphs because the diameter of the graph is n. We can think of the nodes in an n-hop network as being labeled using a vector of length n; $(m_0, m_1, \ldots, m_{n-1})$. Any pair of nodes whose labels differ in exactly one of the m_is are connected by an edge. Therefore, we can go from source node $(s_0, s_1, \ldots, s_{n-1})$ to destination node $(d_0, d_1, \ldots, d_{n-1})$ via the node: $(s_0, s_1, \ldots, s_{n-1})$, $(d_0, s_1, \ldots, s_{n-1})$, $(d_0, d_1, \ldots, s_{n-1})$, ..., $(d_0, d_1, \ldots, d_{n-1})$. When the number of hops is equal to 1, we have a complete graph, and when the number of hops is $\log M$, the resulting topology is a hypercube.

In an n-hop network, the degree of each node is $n(\sqrt[n]{M} - 1)$, which can be shown to be approximately n times the degree of the optimal network as specified by the Moore equation. Therefore, the performance of the n-hop intermodule network is close to the optimal for small values of n. In fact, we can prove that when $n = 2$, an n-hop network is optimal, based on the value of h rather than the network diameter.

Formally we have the following claim:

Claim 5.1 *When we have M modules with P pins each, the pins being organized as no more than $2(\sqrt{M} - 1)$ bundles, and each module contains an equal number of processors, the inter-module network that achieves the maximum peak throughput is a 2-hop network in which each edge corresponds to $P/2(\sqrt{M} - 1)$ wires. The peak throughput of this network is $\frac{MP}{4(1-1/\sqrt{M})}$*

For a fixed M and P, we have already proved that the peak throughput is bounded by $MP/2h$. The average number of hops is $h = \sum_{i=0}^{\infty} ip_i$, and in the case of a 2-hop network, $p_0 = 1/M$, $p_1 = 2(\sqrt{M} - 1)/M$, and $p_2 = 1 - p_0 - p_1$. Therefore, for a 2-hop network $h = 2(1 - 1/\sqrt{M})$. Further, the load on each edge of a 2-hop network is the same implying that making all the edges have an equal number of wires implies that the network achieves the upper bound on the peak throughput: $\frac{MP}{2h} = \frac{MP}{4(1-1/\sqrt{M})}$.

We shall now show that it is not possible to build a network with the same number of modules, pins and bundles such that it has a larger peak throughput. Assume that such a network is possible. Clearly this network should have a smaller average number of hops. We cannot reduce the average number of hops by increasing p_0, since p_0 is fixed at $1 - 1/M$. We cannot increase p_1 above $2(\sqrt{M} - 1)/M$, because this would require increasing the node degree beyond $2(\sqrt{M} - 1)$. Therefore, we cannot reduce h below that of the 2-hop network, implying that we cannot build a network with greater peak throughput than the 2-hop network for the above set of parameters. $\square$

Other networks that have node degree to within a constant factor of the optimal are de Bruijn graph and Kautz graphs [BP89]. A de Bruijn network with diameter n, on M nodes has node degree $2\sqrt[n]{M}$, which is closer to the optimal than an n-hop network for $n \geq 3$ and large M. Kautz graphs are even closer to the optimal than de Bruijn graphs. However, the difference in performance between the n-hop network and the other graphs is unlikely to be significant, since all networks are already extremely close to the optimal. In this chapter, we primarily consider the 2-hop and the 3-hop networks for purposes of evaluation. Some of the advantages of n-hop networks over de Bruijn and Kautz networks are symmetry and regularity. Besides, for the sizes of networks we are concerned with in this chapter, the differences in pin-counts and number of bundles, between the n-hop networks and de Bruijn and Kautz graphs are minimal.

5.1.2 Comparing cost for fixed performance

We note that the peak throughput of a 2-hop network on M modules and P pins per module is $\frac{MP}{4}(1 - 1/\sqrt{M})$, while that of a complete graph with the same number of modules and pins, is $\frac{MP}{2}(1 - 1/M)$. This means that the 2-hop network has approximately half the performance of a complete graph when we keep the pins per module constant. If we wish to design a 2-hop network with the same performance, we have to approximately double the pin-count. Therefore, we see that there is a trade-off between the pin-count and the number of bundles. We have obtained a reduction in the number of bundles from $M - 1$ to $2(\sqrt{M} - 1)$ by doubling the pin-count per module. This trade-off is illustrated by table 5.1. Since the bisection width is a good approximator of network performance, the table holds the bisection width constant across the different topologies.

We can clearly see from the table that as the number of bundles decreases, the pin-counts increase. Intuitively we can see the reason behind this trade-off. As we reduce the number of bundles we also increase the number of hops that messages need to make. As the number of hops increases, we need more wires in order to achieve the same network capacity resulting in an increase in pin-counts.

In this chapter we consider two different values for M, the number of modules; M=32 and M=64, corresponding to a limit on the number of processors per module at 32 and 16 respectively. For detailed analysis, we choose the complete graph, 2-hop and 3-hop intermodule networks. We choose these networks because we believe that at the higher

Inter-Module Topology	Bisection Width	Module Pin-count	Bundles Per Module	Total #Bundles
Complete graph	B	$(4B/M)\frac{M-1}{M}$	$M-1$	$M(M-1)/2$
2-hop Network	B	$(8B/M)\frac{\sqrt{M}-1}{\sqrt{M}}$	$2(\sqrt{M}-1)$	$M(\sqrt{M}-1)$
3-hop Network	B	$(12B/M)\frac{\sqrt[3]{M}-1}{\sqrt[3]{M}}$	$3(\sqrt[3]{M}-1)$	$3M(\sqrt[3]{M}-1)/2$
n-hop Network	B	$(4nB/M)\frac{\sqrt[n]{M}-1}{\sqrt[n]{M}}$	$n(\sqrt[n]{M}-1)$	$nM(\sqrt[n]{M}-1)/2$
2-hop de Bruijn	B	$8B/M$	$2\sqrt{M}$	$M\sqrt{M}$
3-hop de Bruijn	B	$12B/M$	$2\sqrt[3]{M}$	$M\sqrt[3]{M}$
n-hop de Bruijn	B	$4nB/M$	$2\sqrt[n]{M}$	$M\sqrt[n]{M}$
Butterfly	B	$4B\log M/M$	4	$2M$
CCC	B	$6B\log M/M$	3	$3M/2$
Hypercube	B	$2B\log M/M$	$\log M$	$M\log M/2$
d-dim Toroid	B	$dB\sqrt[d]{M}/M$	$2d$	Md
3-dim Toroid	B	$3B/M^{2/3}$	6	$3M$
2-dim Toroid	B	$2B/\sqrt{M}$	4	$2M$
Ring	B	B	2	M

Table 5.1: Characteristics of topologies used to connect M packaging modules