levels of the packaging hierarchy, the number of pins per module is a more important factor than the number of bundles. We have already evaluated the networks for a complete graph inter-module network in the previous chapter, and here we consider a 2-hop network for $M = 32$, and 2-hop and 3-hop networks for $M = 64$.

Interestingly, it turns out that any n-hop inter-module network can be generated by appropriately partitioning a butterfly. A butterfly network of size $N = 2^{(n+1)x}$ can be partitioned into 2^{nx} modules so that the interconnection between the modules is a n-hop network. A $2^{(n+1)x}$ butterfly has $(n+1)x$ columns. We separate these columns into $(n+1)$ groups of x columns each and shuffle the rows such that the connections within each group form butterflies of size 2^x. Next we partition the network into 2^{nx} modules such that each module has 2^x rows and $(n+1)x$ columns. The edges that connect between the modules form a n-hop network. Note that this construction only applies when the number of processors is of the form $2^{(n+1)x}$, and the number of modules is 2^{nx}. This construction can be easily extended to the case where the number of processors is not of the form $2^{(n+1)x}$, by building a network of size $2^{(n+1)x}$, partitioning it into 2^{nx} modules and connecting multiple processors to each network input.

We can also build an n-hop network when the number of modules M is not of the form 2^{nx}, but the numbers in table 5.1 will not apply directly. The table assumes that the n-hop network topology is a n-fold product of complete graphs of size $\sqrt[n]{M}$. If the nth root of M is not an integer, the sizes of the complete graphs that form the product will not be equal. The number of bundles and the pin-count will be slightly larger than that shown in the table.

Since 32 is not a perfect square, we use the topology $K_4 \times K_8$, where K_i denotes a complete graph on i nodes and x denotes graph product. Figure 5.1 shows this graph. For the sake of simplicity, the figure only shows one of the 4 K_8s. The actual topology, will have 3 more K_8s, one on each node of the K_4s. In this topology, there are 10 edges per node, for a total of 160 bundles (in comparison to 31 bundles per node or a total of 496 bundles in a complete graph on 32 modules).

When M=64, we use the 2-hop network $K_8 \times K_8$, and the 3-hop network $K_4 \times K_4 \times K_4$. Figure 5.2 shows the 2-hop network. The figure shows only one of 8 complete graphs. The number of bundles per module is 14, or a total of 448 (in comparison to 63 bundles per module, or a total of 2016 in a complete graph). Figure 5.3 shows the 3-hop network. For the sake of simplicity, the figure does not show all the edges. This network

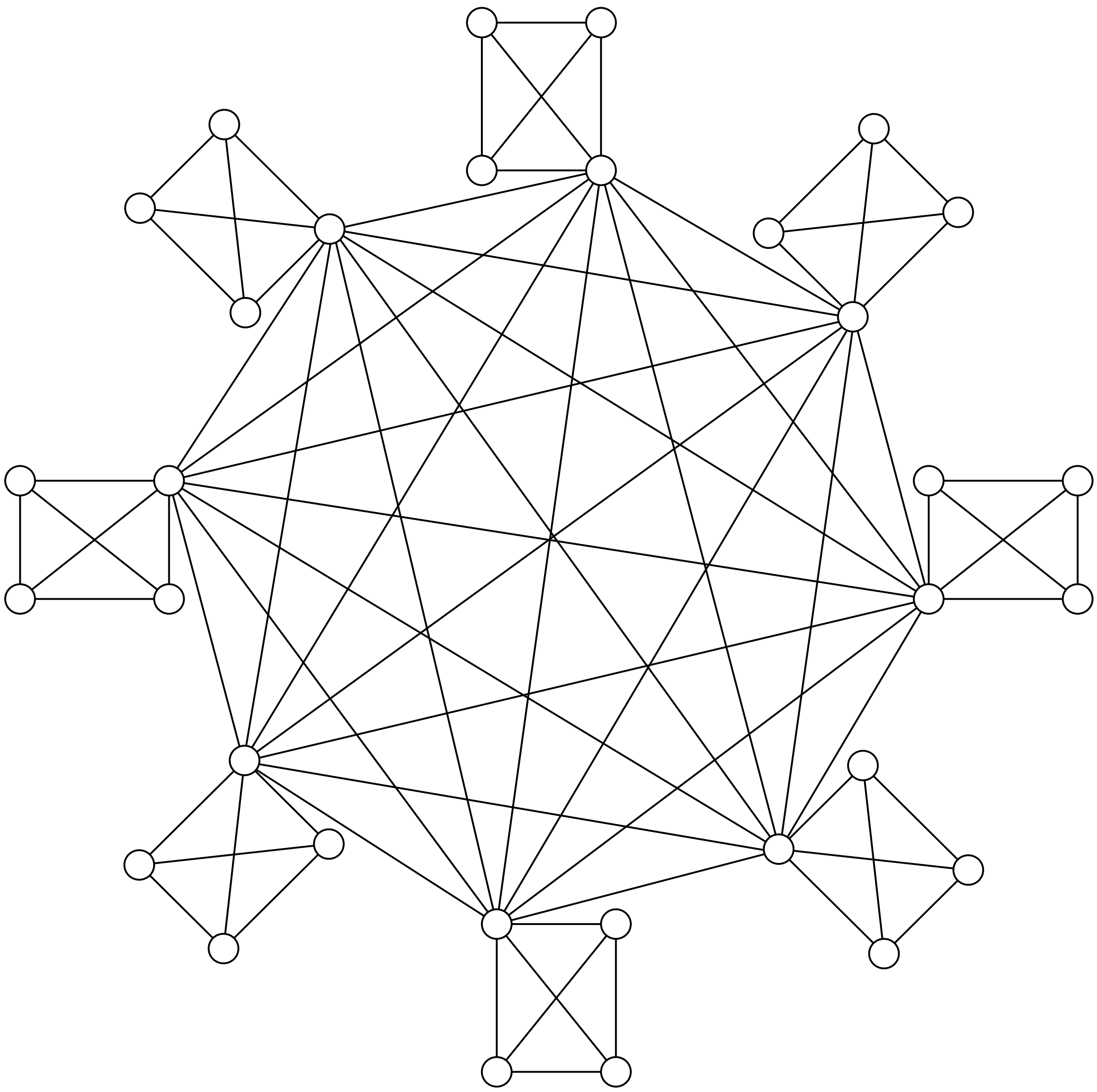

Figure 5.1: $K_4 \times K_8$ 2-hop Network (Only one of the 4 complete graphs on 8 nodes is shown)

Inter-Module Topology	Module Pin-count	#Bundles per Module	Total #Bundles
M=32			
Complete graph	992	31	496
$K_4 \times K_8$	1664	10	160
M=64			
Complete graph	504	63	2016
$K_8 \times K_8$	896	14	448
$K_4 \times K_4 \times K_4$	1152	9	288

Table 5.2: Pin-counts and number of bundles required

requires 9 bundles per module, or a total of 288 bundles. Table 5.2 shows the pin-counts and number of bundles required by the inter-module topologies considered in this chapter. All networks have a bisection width of 8192 as before. The complete-graph topologies are presented for comparison.

5.2 Intra-module Networks

5.2.1 M=32 Modules (2-hop inter-module network)

As in the previous chapter, we consider two separate networks, one for access requests and one for the responses, and each edge in the inter-module topology corresponds to 4 different channels, two in each direction. As before we fix the bisection width B at 8192. In order to achieve this bisection width, the edges of the K_4 will have to be 256 bits wide (4 64-bit wide channels), and the edges of the K_8 will be 128 bits wide (4 32 bit wide channels). This inter-module network requires 1664 pins per module as opposed to 992 for a complete graph.

Basic 2-hop network

The intra-module network of the basic 2-hop network is shown in figure 5.4. The processors are connected to the nodes on the left. The inter-module network is a $K_4 \times K_8$ and is generated by partitioning a 256-input butterfly as described in section 5.1. A 256-input butterfly leaves us with 8 inputs per module. Since there are 32 processors per module and only 8 inputs, we have to connect 4 processors to each input. This is done by building a

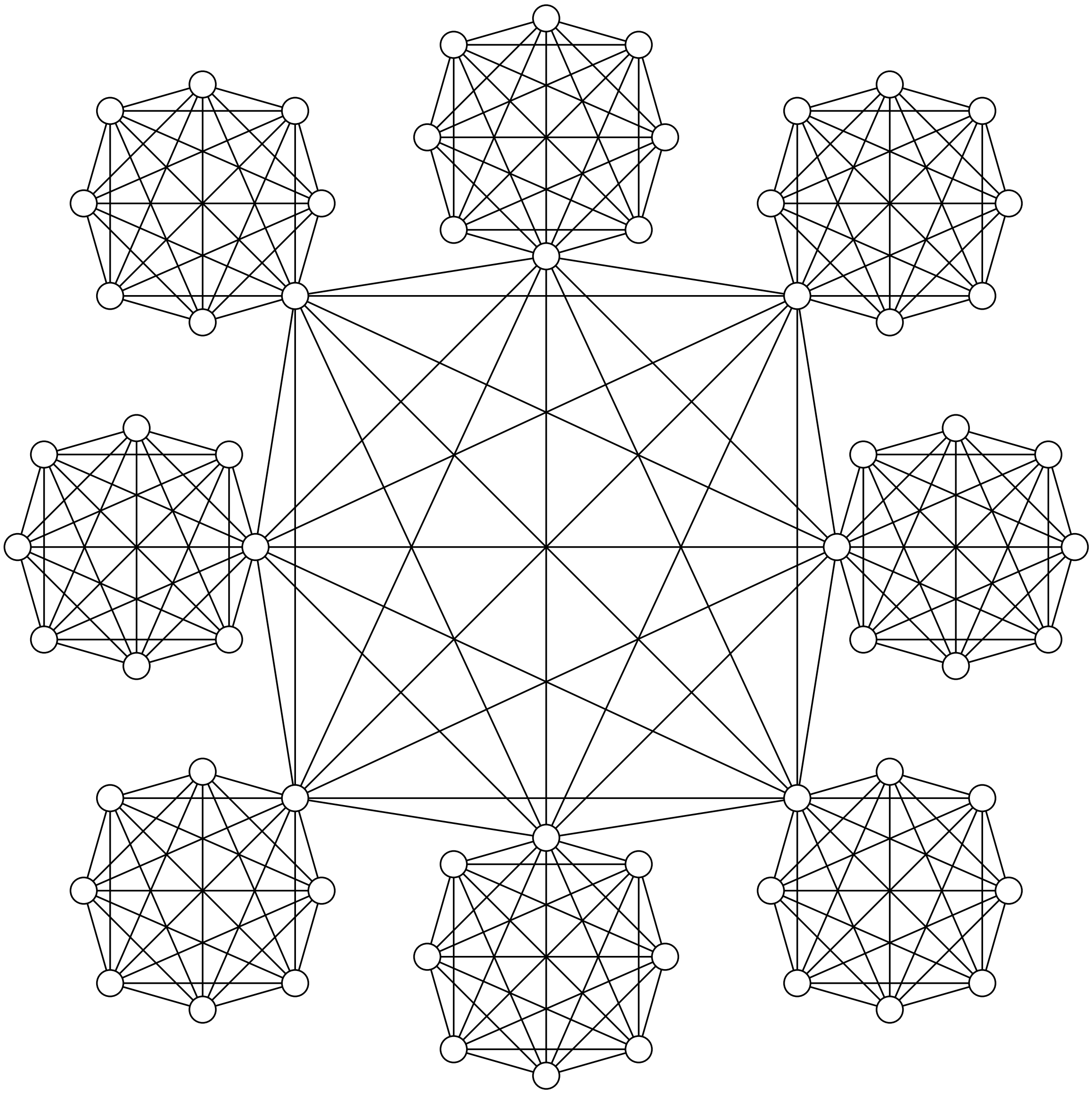

Figure 5.2: $K_8 \times K_8$ 2-hop Network (Only one of the 8 complete graphs on 8 nodes is shown)

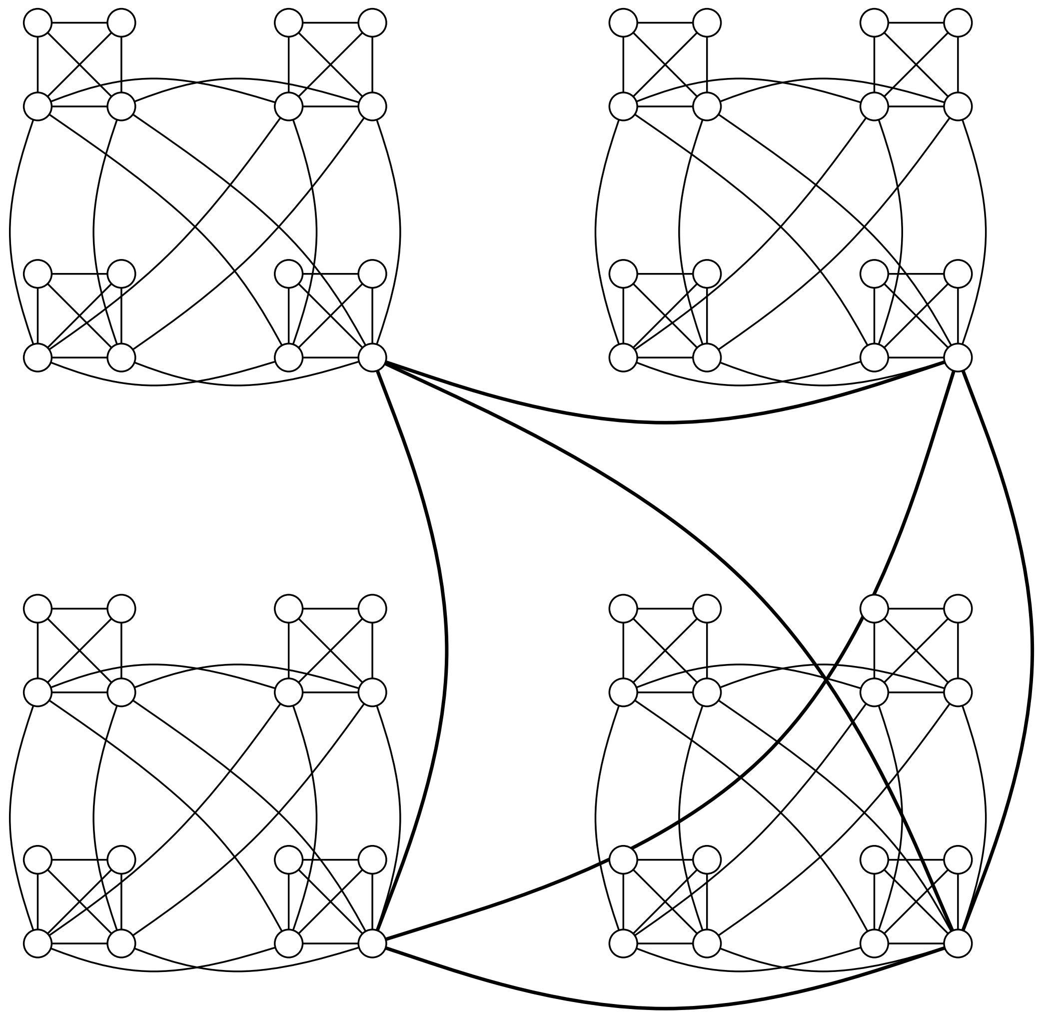

Figure 5.3: $K_4 \times K_4 \times K_4$ 3-hop Network (Only 2 of 4 complete graphs, and 1 of 16 complete graphs are shown)

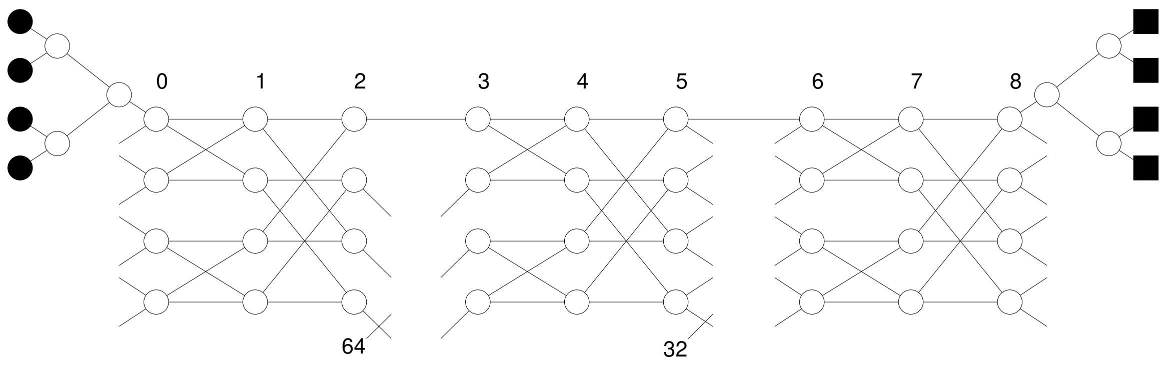

Figure 5.4: Intra-Module Network: 2-hop network, M=32 Modules

2 level tree as shown on the top node. The memories are also connected in a similar fashion. Processors and memories are paired as in the network topologies of the earlier chapter, but this is not illustrated pictorially.

The channels between the column 2 and 3 correspond to K_4. They connect between modules that differ in their least significant 2 bits. The channels between column 5 and 6 correspond to K_8 and connect between modules that differ in their most significant 3 bits. All channels are 32 bits wide except those between columns 2 and 3, which are 64 bits wide.

As per our naming convention we call this network *Butterfly(2-hop).32.64-32* since the intra-module channels are 32 bits wide and there are two kinds of inter-module channels, which are 64 and 32 bits wide.

Widening channels and dilation

Similar to the networks defined for the complete graph inter-module topology, we define the network *Butterfly(2-hop).64.64-32*. This network is the same as the previous network except that all the intra-module channels are 64 bits wide, including those that connect to the processors and memories.

We also define the network *Butterfly(2-hop).32-2dil.64-32* with a degree 2 dilation of the intra-module network.

Fewer memory units

In the basic 2-hop network we used a 2 level tree to connect 4 memories to each network input. Since the required memory throughput is only 8 bits per cycle, we can

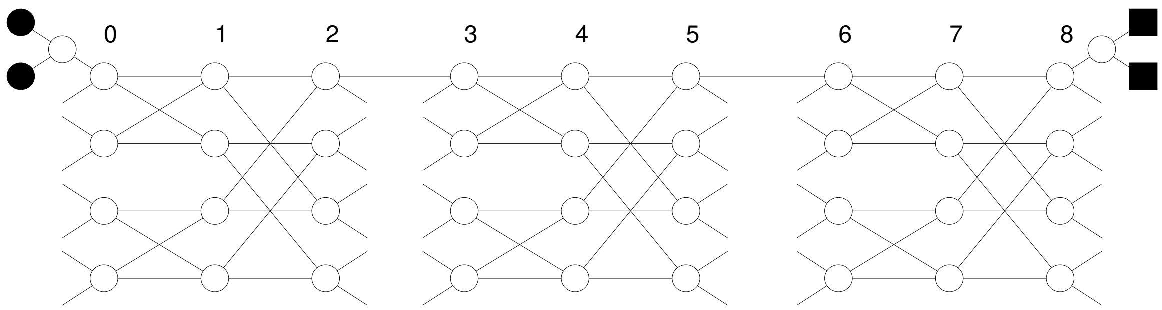

Figure 5.5: Intra-Module Network: 2-hop network, M=64 Modules

replace the 2 level tree by a single memory unit when the channels are 32 bits wide. We call this network *Butterfly(2-hop).32.64-32.M-256*. We also evaluate *Butterfly(2-hop).64.64-32.M-256* where the intra-module channels are 64 bits wide.

5.2.2 M=64 Modules (2-hop inter-module network)

To achieve a bisection width of 8192, the bundles of the 2-hop network have to be 64 wires wide. Since each bundle corresponds to 4 channels the inter-module channels will be 16 bits wide.

Basic 2-hop network

We derive the basic 2-hop network by partitioning a butterfly as described earlier. Since we need to construct a $K_8 \times K_8$ inter-module network, we start with a butterfly network with $8 * 8 * 8 = 512$ inputs. We connect two processors to each input of the butterfly network to get a 1024 processor machine. Since the inter-module channels are 16 bits wide, we get a uniform network if we make the intra-module channels also 16 bits wide. We call this network *Butterfly(2-hop).16.16* and is shown in figure 5.5.

Widening channels

We consider using wider intra-module channels of 32 bits instead of 16 and we call this network *Butterfly(2-hop).32.16*.

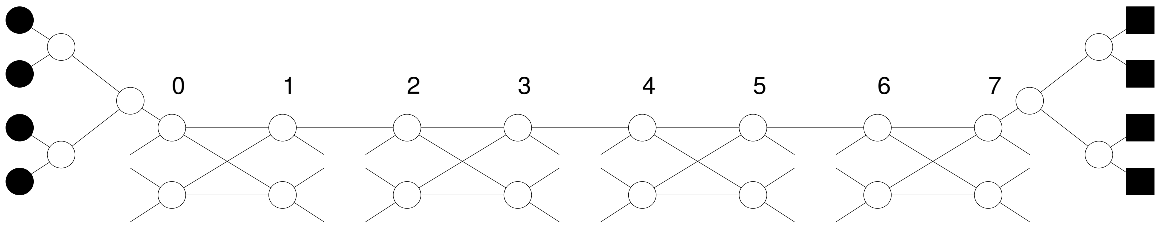

Figure 5.6: Intra-Module Network: 3-hop network, M=64 Modules

5.2.3 M=64 Modules (3-hop inter-module network)

To achieve a bisection width of 8192, the bundles of the 3-hop network have to be 128 wires wide. Since each bundle corresponds to 4 channels the inter-module channels will be 32 bits wide.

Basic 3-hop network

Since we need a 3-hop network on 64 modules, we start with a butterfly network on $4*4*4*4 = 256$ inputs and partition it to get a $K_4 \times K_4 \times K_4$. We connect 4 processors to each network input. This network is shown in figure 5.6. The inter-module channels are 32 bits wide. We get the network *Butterfly(3-hop).32.32* when the intra-module channels are also 32 bits wide.

Widening channels

We consider using wider intra-module channels of 64 bits instead of 32 and we call this network *Butterfly(3-hop).64.16*.

5.3 Simulation Results

The results corresponding to the open-network work-load model are presented in figures 5.7, 5.8 and 5.9. Figure 5.7 compares the different networks which use a 2-hop inter-module network, with M=32. The data corresponding to *Butterfly.8.8* and *Butterfly.32.8*, from the previous chapter are included for comparison. In the latter two networks the inter-module network is a complete graph. Although all the networks have the same bisection width, the latter two networks use fewer pins per module, and more bundles. Therefore, their costs are not directly comparable. Figures 5.8 and 5.9 compare networks

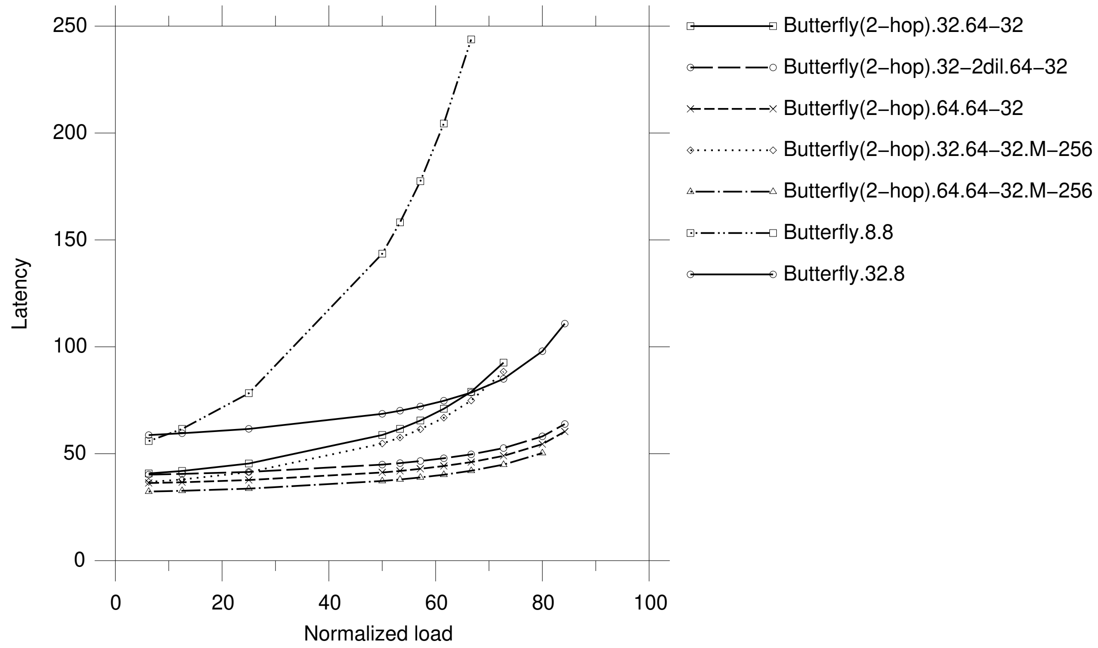

Figure 5.7: Open-Network Model; M=32 Modules, 2-hop intermodule network

where M=64. The data corresponding to *Butterfly.8.2* and *Butterfly.32.2* (inter-module network = complete graph) are included for comparison.

Figures 5.10 and 5.11 show the results for the multithreading work-load when the mean access interval is 16 cycles, and figures 5.12 and 5.13 correspond to a mean access interval of 32 cycles.

5.3.1 Discussion of results

The results indicate the same general trends as seen in the previous chapter. We can improve performance by making the intra-module networks richer either using dilation or wider channels. We do not see much difference between dilation and widening channels since the channels are already quite wide to begin with. Increasing the channel widths further does not make much of an impact with respect memory service rate. Dilation and widening channels have similar effects with respect to reducing contention. As before, we see that reducing the number of memory units does not degrade performance. Actually it improves the performance by a small amount since we reduce the number of switching stages that the messages have to pass through, by eliminating the fan-out trees at the memory end.

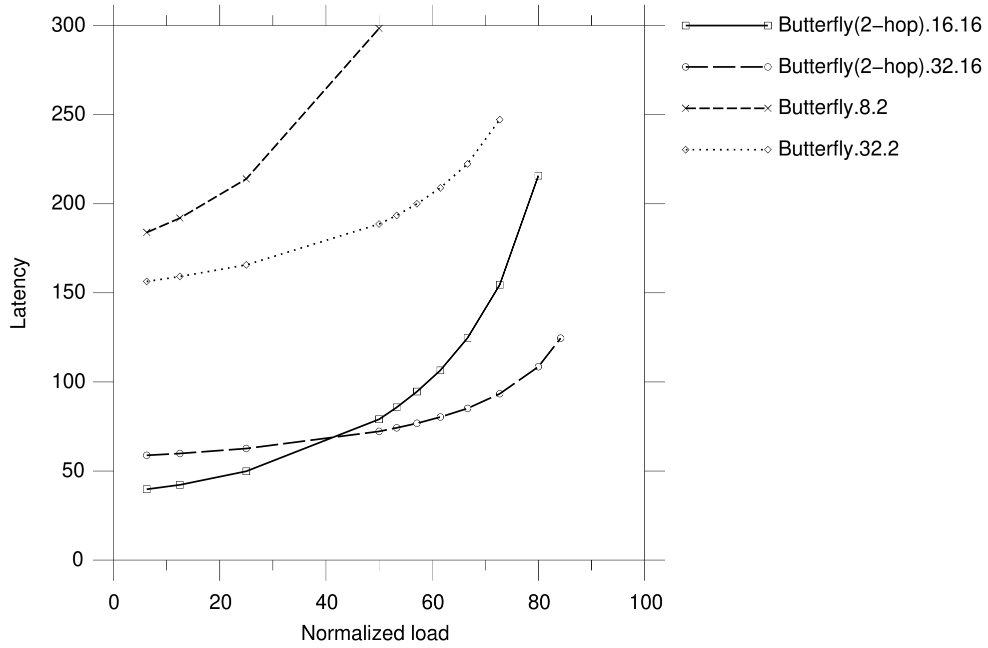

Figure 5.8: Open-Network Model; M=64 Modules, 2-hop intermodule network

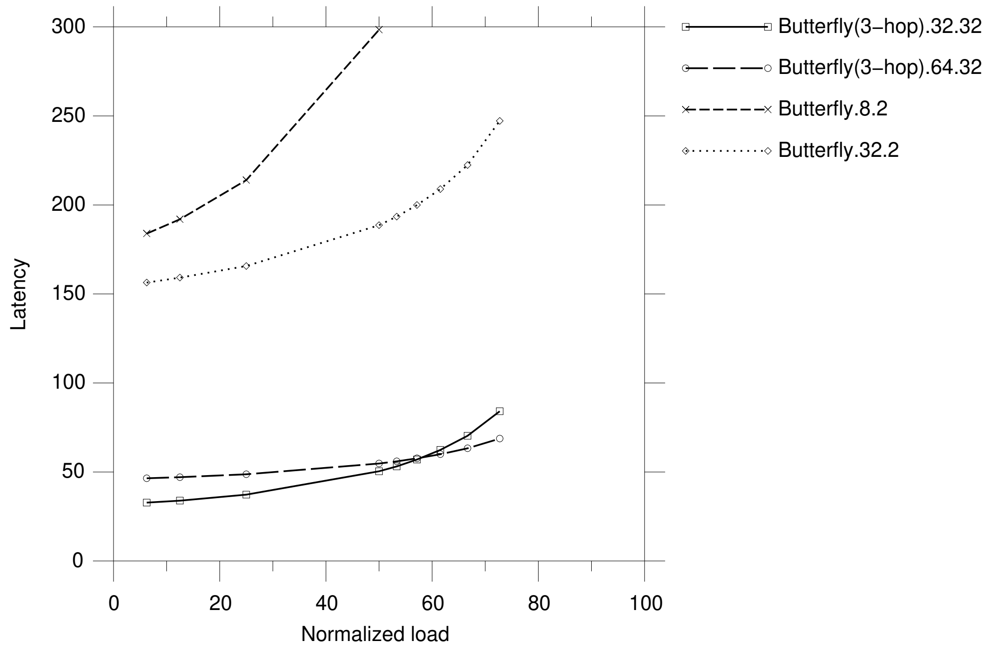

Figure 5.9: Open-Network Model; M=64 Modules, 3-hop intermodule network

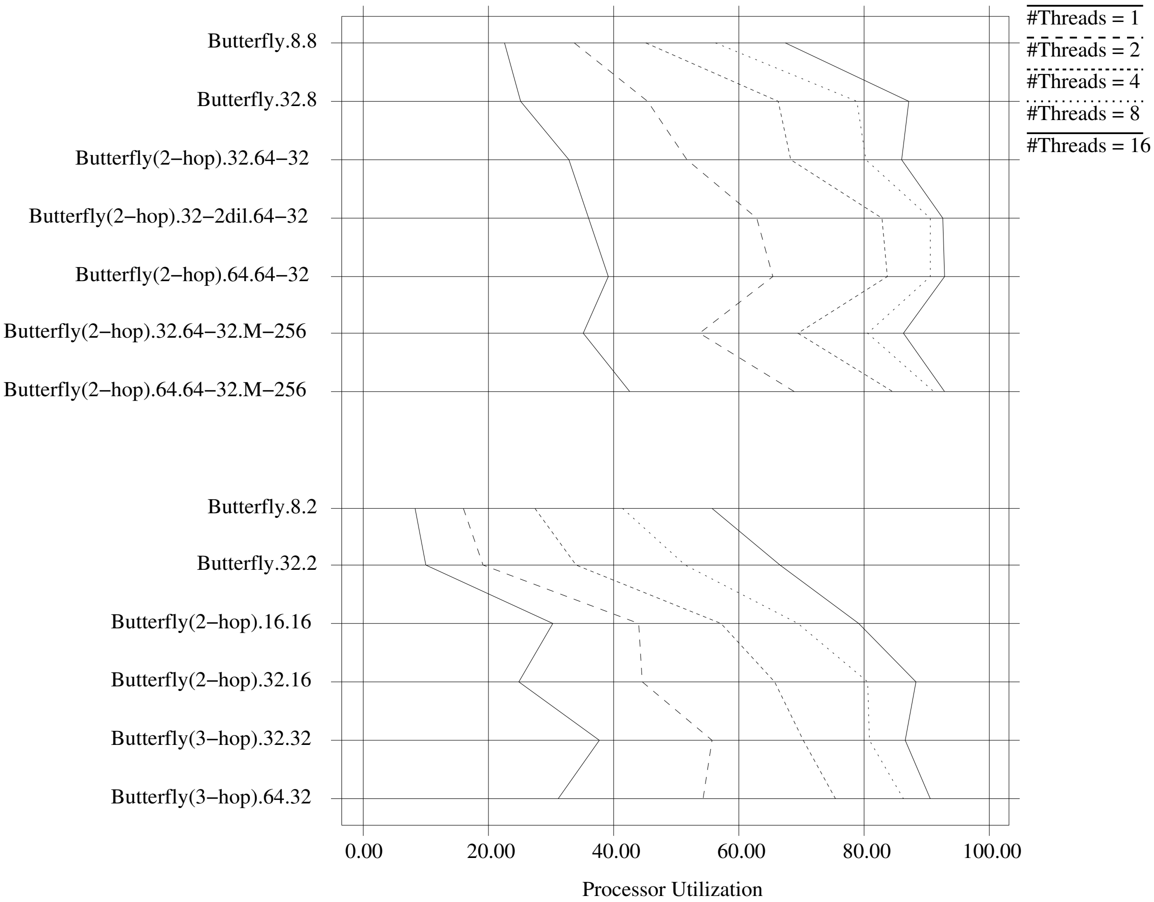

Figure 5.10: Periodic Access Model, access interval = 16 cycles

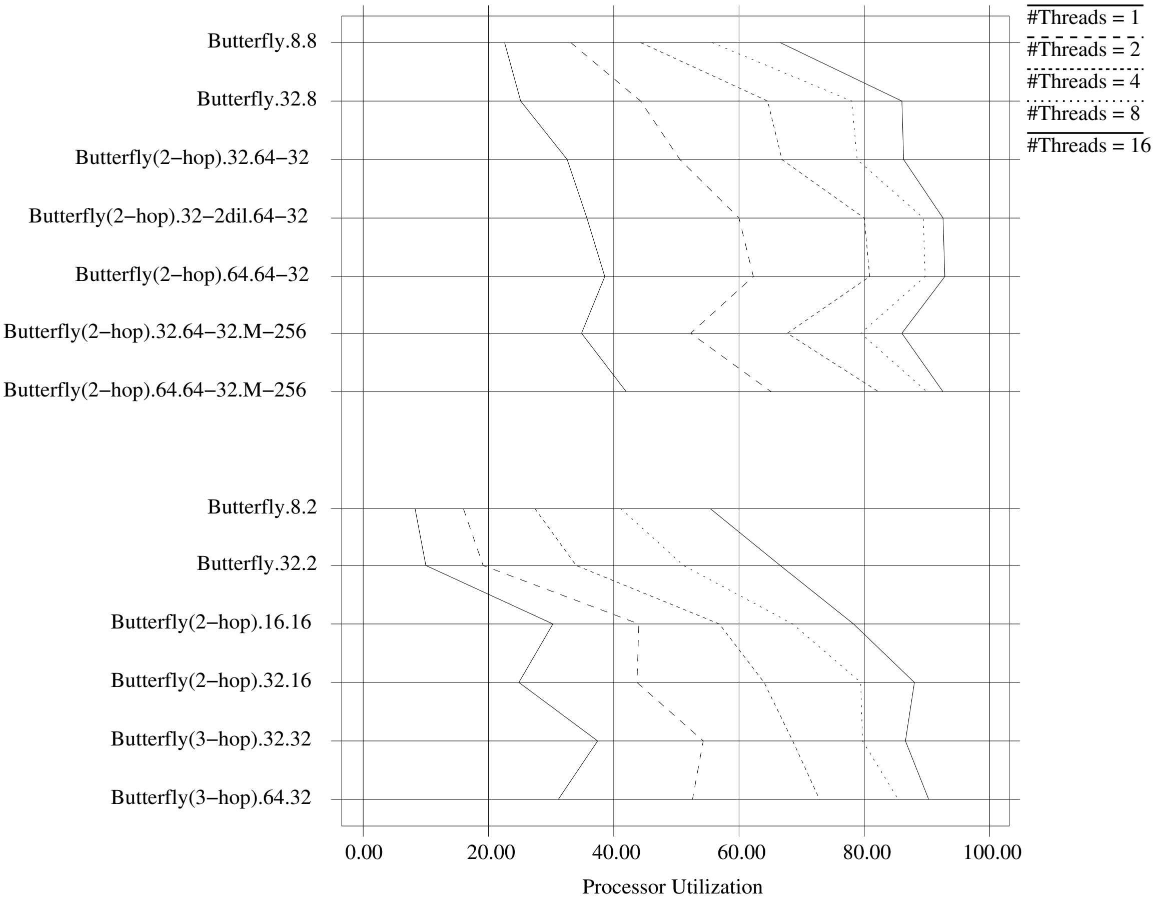

Figure 5.11: Geometric Access Model, access interval = 16 cycles

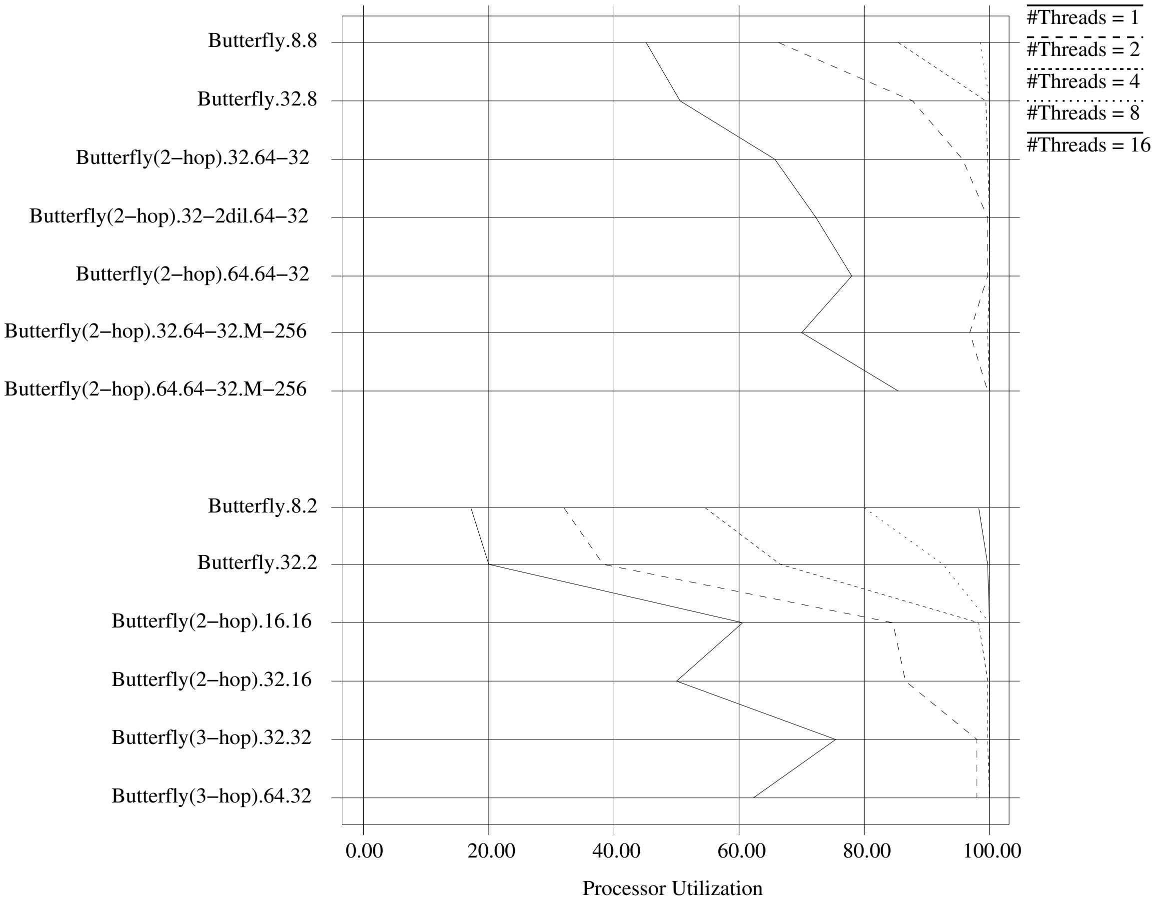

Figure 5.12: Periodic Access Model, access interval = 32 cycles

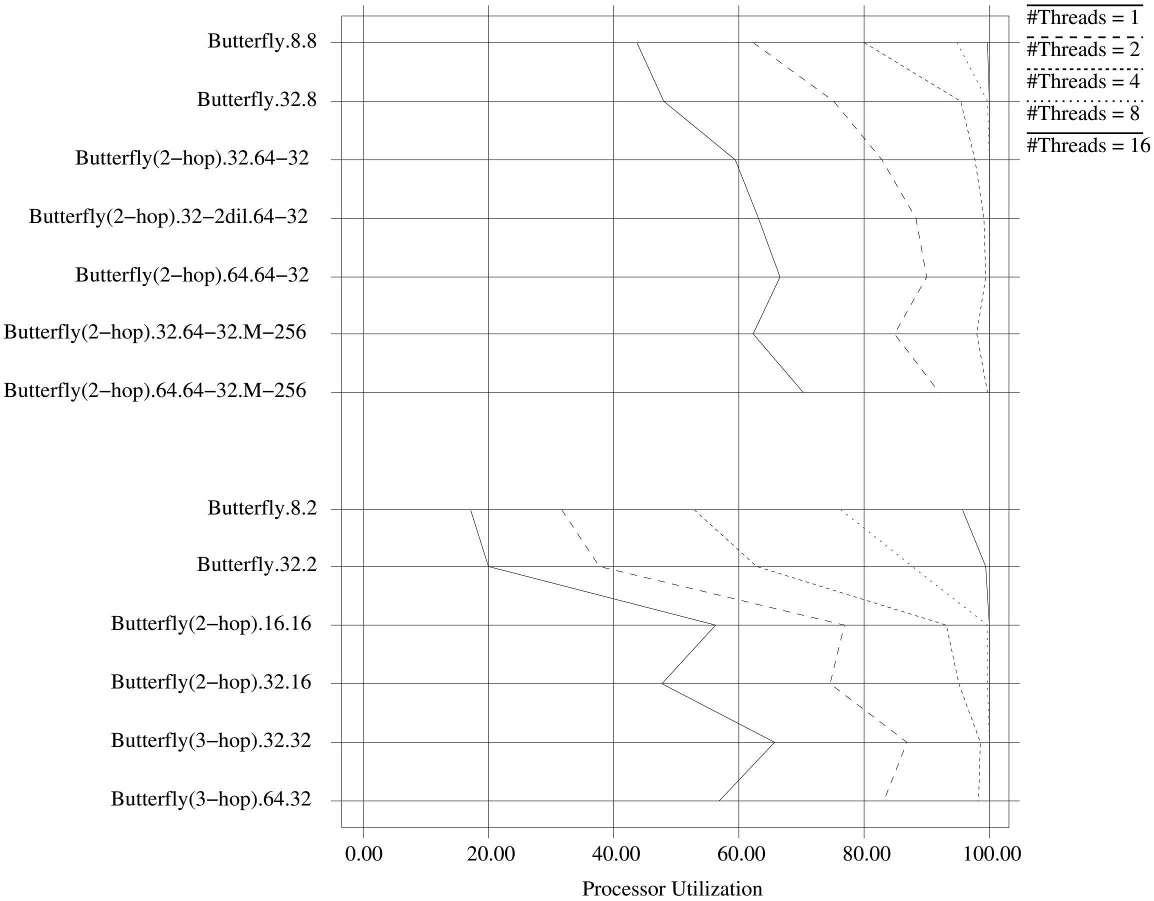

Figure 5.13: Geometric Access Model, access interval = 32 cycles

While comparing 1-hop, 2-hop, and 3-hop networks, we must keep in mind that their cost vectors are different, although they have the same peak throughput. However, if we ignore the differences in the cost vector, and just compare networks with the same peak throughput, we see that the 1-hop networks are worse than 2-hop and 3-hop networks. The reason for this difference in performance is the wider channels in the latter networks. When the channels are wider, it is easier to achieve higher channel utilizations.

5.4 Summary

In this chapter, we considered a cost vector that was slightly more complex than that of the previous chapter. We included the number of bundles in the list of cost parameters. When we expanded our list of cost parameters to include the number of bundles, we saw that our set of interesting top-level was no longer limited to a complete graph network, as in the previous model, where we ignored the number of bundles.

In this chapter, we first compared the performance of networks of fixed cost, where we fixed the cost by fixing all the three cost parameters; namely the number of modules, the pin-count per module and the number of bundles per module. We saw that it was difficult to find the network with maximum performance for any arbitrary set of parameters. Instead, we were able to find networks that had performance close to the maximum possible performance. We showed that the n-hop networks and de Bruijn networks were close to optimal.

When we examined the network design problem from the point of view of fixing performance and comparing costs, we discovered a trade-off between pin-count and number of bundles. In going from a complete graph to a 2-hop network we reduced the number of bundles per module from $O(M)$ to $O(\sqrt{M})$, but doubled the pin-count. In general, reducing the number of bundles implied that the pin-count had to increase to maintain the same level of performance. We also showed that the trade-offs offered by n-hop and de Bruijn networks were close to optimal, in the sense it was not possible to reduce the pin-count and number of bundles by more than small constant factors and still achieve the same performance.

A question that we might ask is, "Which of the n-hop or de Bruijn networks should we use?". In order answer this question, we need to compare the costs of the different n-hop and de Bruijn networks. To compare their costs, we need to know the relative weights of the cost dimensions, but we chose not to specify these weights because they were technology dependent. Therefore, the answer to the above question is technology dependent, and we can answer it by first determining the relative weights of the cost dimensions based on the technology of interest. Then, we would use these weights to compute the actual monetary costs of the different networks and select the network that has the lowest cost.

Similar to the previous chapter, we evaluated several alternative network organizations for the lower level of the hierarchy, and found that the general results were similar.

The results in this chapter also reinforce a result we obtained in the previous chapter. In the previous chapter, we did not include the number of bundles in the cost vector, and under the previous model we saw that the complete graph was the best inter-module network. Comparing networks evaluated under model 2 on the basis of equal pin-counts illustrates this result. If we permit the complete graph network to use the same number of pins as a 2-hop network, the bisection width of the complete graph becomes twice that of the 2-hop network. Although we did not explicitly simulate a complete graph with twice the bisection width, we can get a good approximation of the performance of such a network by looking at the performance of the complete graph network under 50% load, because doubling the bisection width also doubles the peak throughput. When we compare the performance of the complete graph with a mean access of 32 cycles with the performance of the 2-hop network with a mean access interval of 16 cycles, we see that the complete graph network achieves significantly greater processor utilizations. In fact, most of the networks in which the inter-module network is a complete graph, achieve almost 100% processor utilization. Therefore, we can clearly see that comparing networks based on equal pin-counts favors the complete graph.

Chapter 6

Model 3: Three level hierarchy

This model is motivated by the fact that the cost of the different wires that go between modules may not all be identical. Wires that connect modules which are physically close to each other will be short and therefore, likely to be less expensive than wires that connect long distances. For instance, in a packaging hierarchy where modules correspond to card-cages, wires that go between card-cages within a cabinet are likely to be cheaper than wires that go between cabinets. In this model, we use an additional level of packaging to characterize this difference in costs. In general, we can increase the number of levels in the packaging model, to characterize the technology to a greater level of detail. In this chapter, we assume that the machine is packaged using a 3 level hierarchy. At the top level the machine consists of level-2 modules which we call *clusters*. Each level-2 module consists of level-1 modules which we simply call *modules* as before. The connections between modules in the same cluster correspond to the short connections and the connections that go between modules in different clusters correspond to the long distance connections.

As before we have intra-module and inter-module connections, but now, the inter-module connections are divided into two categories: intra-cluster connections and inter-cluster connections. Each module has two kinds of pin-count: intra-cluster pin-count and inter-cluster pin-count. The intra-cluster pins are used to make connections between modules within the same cluster and the inter-cluster pins are used to make connections between modules in distinct clusters.

The cost vector under this model is more complicated than the ones used in the previous chapters. In this model, the cost vector has five dimensions: the number of clusters, number of modules per cluster, the level-2 pin-count, number of level-2 bundles, and the level-1 pin-count. The level 2 pin-count is the number of long-distance wires connecting to each cluster or the total number of inter-cluster pins used by the modules within a cluster. As before, we shall assume that there is a limit on the maximum number of processors that we can put in a module. We shall also assume that a similar limit exists on the number of modules in a cluster.

Since the cost vector has many more dimensions than the earlier models, an approach we took in the earlier chapters, namely one of fixing the cost by fixing all the parameters, is overly restrictive. Instead, in this chapter, we shall fix the performance and explore the trade-offs between the dimensions of the cost vector. Similar to the previous models, we shall fix some of the dimensions of the cost vector and examine the trade-offs between the other dimensions. In this chapter, we fix the number of clusters, the number of modules per cluster and the level-2 pin-count. In fixing the first two parameters, we minimize both the number of clusters and total number of modules based on the limits on the maximum number of modules per cluster and maximum number of processors per module respectively. Since the previous models have already analyzed the relationship between the top-level pin-count and performance, in this chapter we also fix the top-level (or level-2) pin-count. We are now left with two dimensions of the cost vector: the number of level-2 bundles and the level-1 pin-count. We explore the trade-off between these two parameters by considering different values for the level-1 pin-count. We shall see that we if we have more intra-cluster pins, we can reduce the number of bundles, while maintaining the same level of performance. As before, we use fix the performance of the network by fixing its bisection width.

6.1 Network choices

Let the total number of modules be M, and the number of modules per cluster be C, for a total of M/C clusters. We have already seen that the best top-level network when the number of top-level pins is fixed, is a complete graph. Since there are (M/C) clusters, fixing the bisection width at B, implies that each edge in the complete graph topology must correspond to $\frac{4B}{(M/C)^2}$ wires. Since each cluster is connected to $(M/C)-1$ other clusters, the number of level-2 pins required per cluster is $\frac{4B(M/C-1)}{(M/C)^2} = \frac{4B(M-C)C}{M^2}$. In this chapter we consider the case M=32, and C=4 and as before, we fix B=8192. For this particular case, each pair of clusters is connected using 512 wires, and the number of level-2 pins required per cluster is 3584.

However, we still consider the connections to be between modules, since clusters are conceptual units of the hierarchy that mainly help us distinguish between short and long-distance connections. We observe that we can have many different inter-module topologies that lead to a complete graph between clusters.

For example, if we use a complete graph between modules, the interconnection between the clusters is also a complete graph. Each inter-cluster edge corresponds to $C^2 = 16$ inter-module bundles, since each module within a cluster has a connection to each module in the other cluster. Since the inter-cluster edges correspond to $\frac{4B}{(M/C)^2} = 512$ wires, each inter-module bundle consists of $\frac{4B}{M^2} = 32$ wires. There are a total of $\frac{M(M-1)}{2} = 496$ inter-module bundles, of which $(M/C) * \frac{C(C-1)}{2} = 48$ are short-distance bundles and $\frac{1}{2}(\frac{M}{C})(\frac{M}{C} - 1) * C^2 = 448$ are long-distance bundles. Each module requires $\frac{4B}{M^2}(C-1) = 96$ intra-cluster pins and $\frac{4B}{M^2}(M-C) = 896$ inter-cluster pins. Figure 6.1 shows this network.

Another inter-module topology that leads to a complete graph between clusters is the 2-hop network $K_C \times K_{M/C}$. In this network, messages make 2 hops, one hop over the intra-cluster connections and the other hop over the inter-cluster connections. Between any pair of modules, there are $C = 4$ inter-module bundles, because each module is only connected to its corresponding module in the other cluster. Therefore, the long-distance bundles consist of $\frac{1}{C}\frac{4B}{(M/C)^2} = 128$ wires. This network has a total of $\frac{1}{2}(\frac{M}{C}) * (\frac{M}{C} - 1) * C = 112$ long-distance bundles. The number of short-distance bundles is the same as before: $(M/C) * \frac{C(C-1)}{2} = 48$, but each bundle is wider, in order to support the larger bandwidth that is required by the 2-hop network. Each of these short-distance bundles is 256 wires wide and the number of intra-cluster pins per module is be 768. This topology trades-off a larger number of intra-cluster pins for a reduction in the number of bundles. Figure 6.2 shows this network.

The complete graph and the 2-hop network topologies discussed above are the fundamentally the same as the inter-module topologies that we used in Chapters 4 and 5. The only difference between them is the way the pins and bundles are accounted for. For the complete graph inter-module network, all the networks with $M = 32$ considered in Chapter 4, are valid candidates under this packaging model. We only consider the networks *Butterfly.8.8* and *Butterfly.32.8* for the purposes of comparison in this chapter. Similarly all the networks for M=32, considered in Chapter 5, are valid candidates, and we consider the networks *Butterfly(2-hop).32.64-32* and *Butterfly(2-hop).64.64-32* in this chapter.

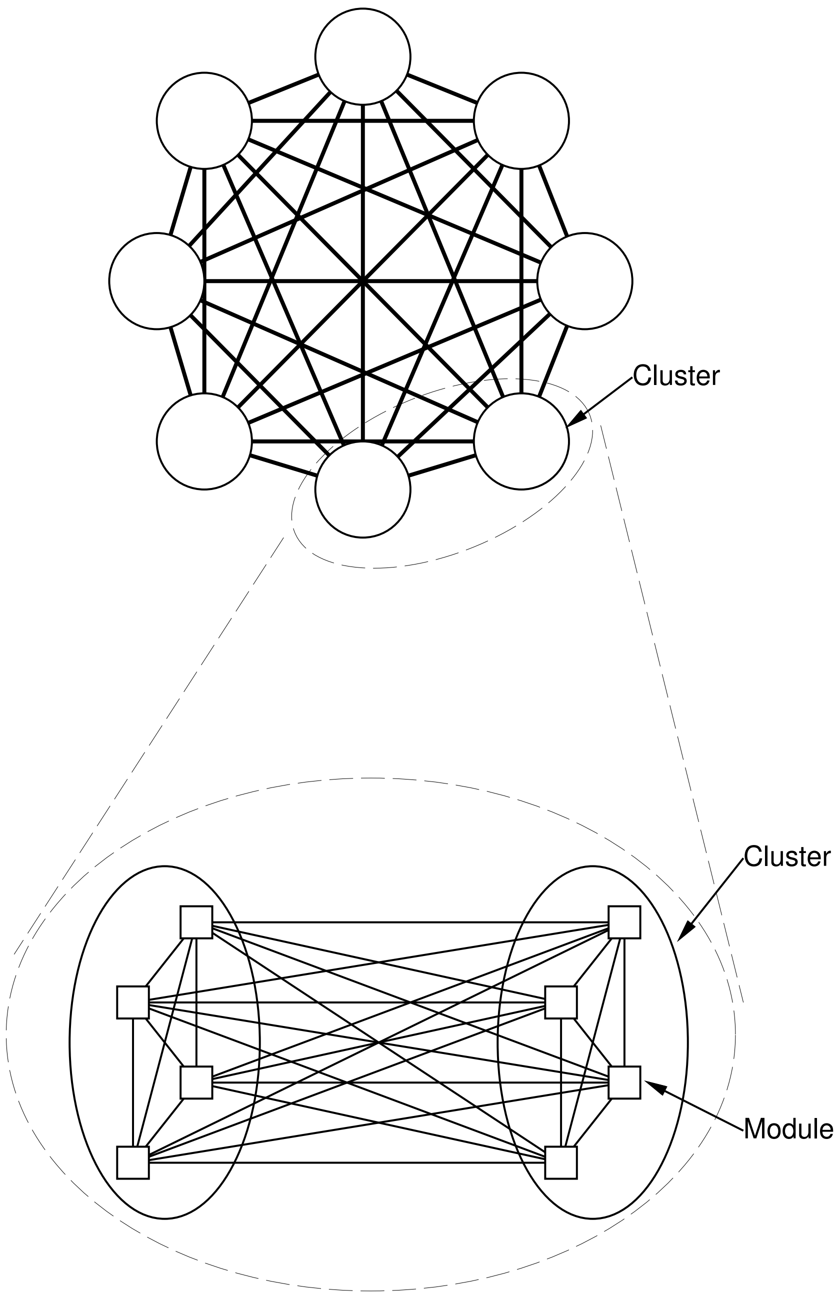

Figure 6.1: Complete graph between modules

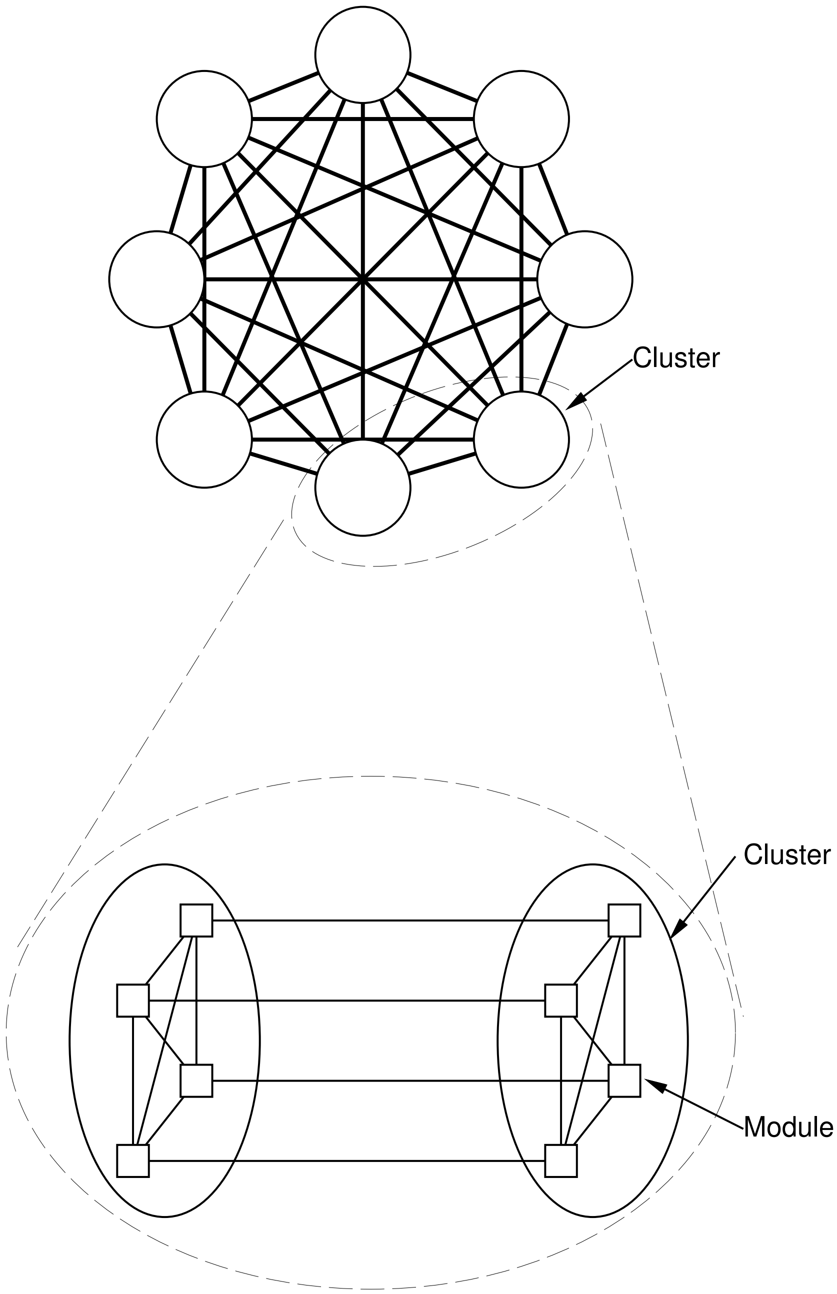

Figure 6.2: 2-hop network $K_4 \times K_8$ between modules

The 2-hop network described above requires 112 long-distance bundles and 768 intra-cluster pins per module. If we have an even larger number of intra-cluster pins, we can reduce this number of bundles further. In going from the complete graph to the 2-hop network we reduced the number of long-distance bundles by a factor of C. The number of bundles between any pair of clusters was C^2 in the case of the complete graph, and C in the case of the 2-hop network. We can build a network which we call a *Clustered network*, in which we reduce the number of inter-cluster bundles by another factor of C, down to exactly one bundle between any pair of clusters. Figure 6.3 shows this network. The total number of inter-cluster bundles required by this network is $\frac{1}{2}(\frac{M}{C}) * (\frac{M}{C} - 1) = 28$ and the number of intra-cluster bundles remains $(M/C) * \frac{C(C-1)}{2} = 48$. In this network, all the wires that go between any pair of clusters are connected between some pair of modules, one in each cluster. Messages in this network make a maximum of 3 hops to get from one module to another. First the message goes from the source module to the appropriate module within its cluster. The second hop is made between clusters, taking the message to the correct destination cluster. The third hop then moves the message within the destination cluster to the correct destination module. Of the 3 hops, two hops are within clusters and one hop is between clusters. Therefore, the short-distance bundles have to be twice as wide as the bundles in the 2-hop network, namely 512 wires each. Each module requires 1536 intra-cluster pins. As in the two earlier cases, the total number of inter-cluster pins per cluster is 3584, but since there are 7 inter-cluster channels and 4 modules per cluster, three of the modules have 2 channels each, and one module only has a single channel. Since each channel is 512 bits wide, three of the 4 modules require 1024 inter-cluster pins while one module requires 512 inter-cluster pins.

We can use a strategy similar to the one used in Chapter 5 to reduce the number of intra-cluster bundles, by using a multi-hop network that requires more intra-cluster pins. But for the case that we consider in this chapter, $C = 4$ modules per cluster, it does not make much sense to make the intra-cluster network sparser since the complete graph interconnect is only on 4 modules. For larger cluster sizes, we could consider using a 2-hop or a 3-hop network between the modules within a cluster.

We can generate a clustered network by suitably partitioning a butterfly network as shown in figure 6.4. The figure shows the connections within a single cluster. Based on the bisection width and the total number of pins used for inter-cluster, and intra-cluster connections, the inter-cluster channels are 128 bits wide and the intra-cluster channels are

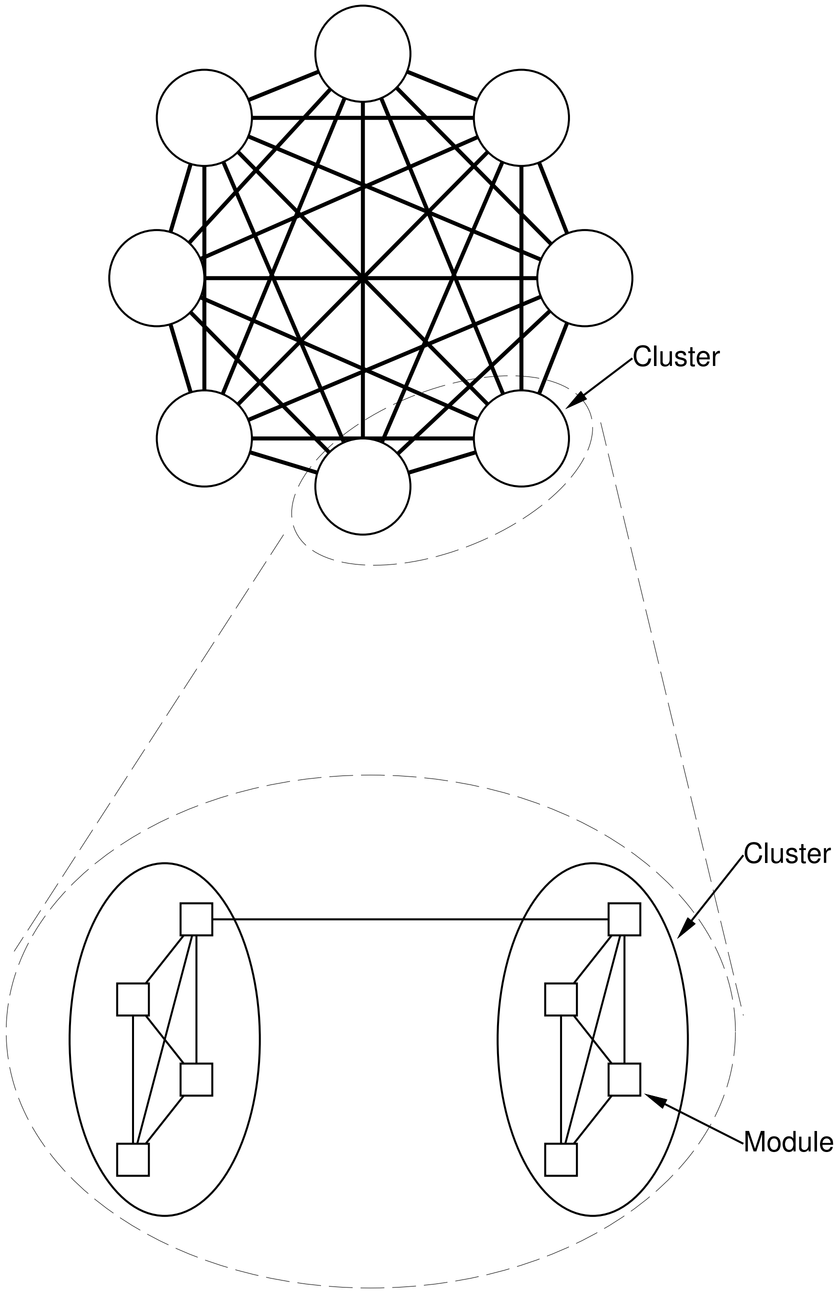

Figure 6.3: Clustered Network

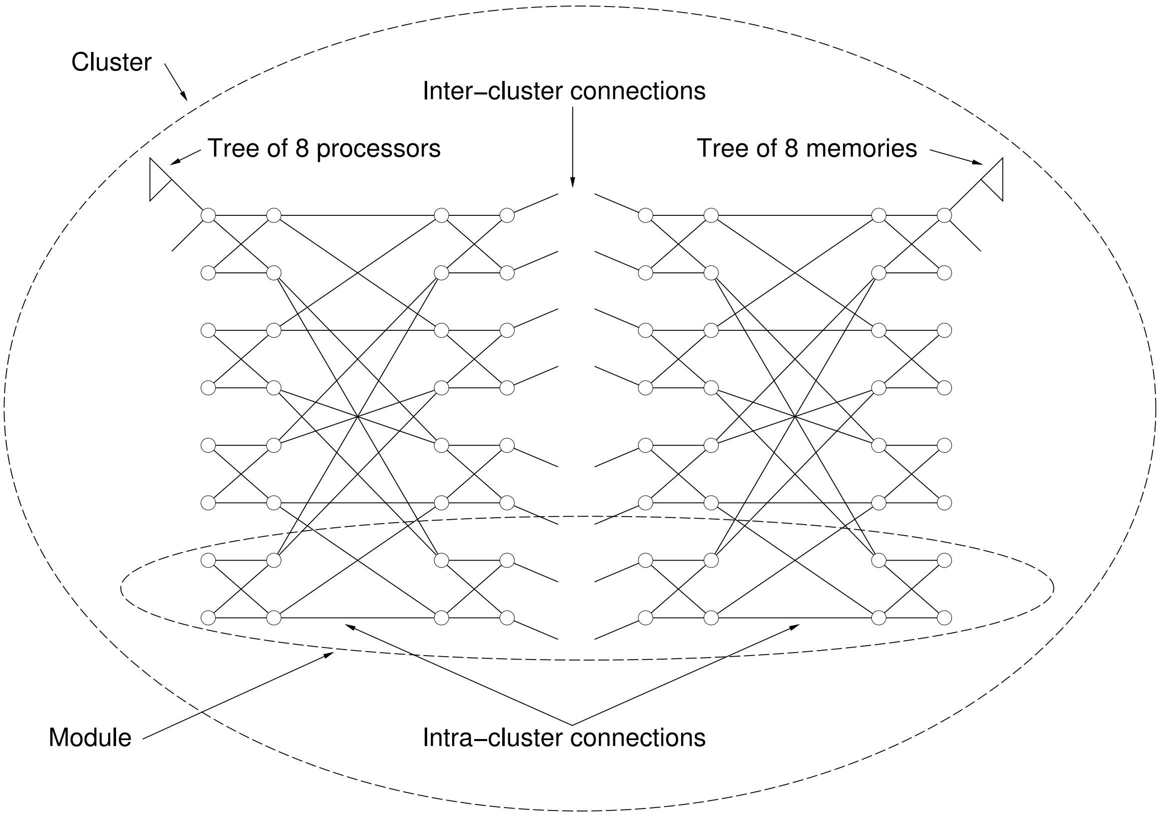

Figure 6.4: Partitioning a butterfly to yield a Clustered network

Network	Inter-cluster Pins/module	Inter-cluster Bundles	Intra-cluster Pins/module	Intra-cluster Bundles
Butterfly.8.8	896	448	96	48
Butterfly.32.8	896	448	96	48
Butterfly(2-hop).32.64-32	896	112	768	48
Butterfly(2-hop).64.64-32	896	112	768	48
Butterfly(Clus).64.64.128	512, 1024	28	1536	48
Butterfly(Clus).128.128.128	512, 1024	28	3072	48

Table 6.1: Pin and bundle requirements

64 bits wide. To balance the bandwidths, intra-module channels of 64 bits are sufficient. As per our convention, we call this network, with the above channel width specifications, *Butterfly(Clus).64.64.128*. If we have more intra-cluster pins than the number required by *Butterfly(Clus).64.64.128*, we can make the intra-cluster connections wider. We also consider the network *Butterfly(Clus).128.128.128* in which all the channels are 128 bits wide. This network uses twice the number of intra-cluster pins per module as *Butterfly(Clus).64.64.128*. Table 6.1 summarizes the characteristics of the different networks.

6.2 Simulation Results

Figure 6.5 shows the results of the open-network workload simulations. Figures 6.6, 6.7, 6.8 and 6.9 show the results for the multithreading workload model.

6.2.1 Discussion of results

Again, the results this model follow the same general trends as seen in the previous chapters, i.e., network performance can be improved by improving the local communication bandwidth (even when the top level bandwidth is the same), and it is better to organize a fixed amount of communication bandwidth as fewer wider channels. This is evident from the improvement in performance resulting from using denser networks at the intra-cluster and intra-module levels. All the networks shown in the graphs have the same number long-distance pins per cluster, but they use different numbers of bundles and intra-cluster pins. The 2-hop and Clustered networks have fewer bundles that are wider, and they achieve better network performance than the networks based on a complete graph between modules.