Chapter 7
Supporting Dynamic Application Patterns

To efficiently utilize the large number of transistors that are available on the chip with manageable design complexity and wiring requirements, *Chip Multiprocessors* (*CMPs*) have been recently proposed [100–103]. In CMPs, the chip area is divided into a number of regular and identical tiles, where each tile represents a processor/memory core. The use of a simpler architecture for the processor in a single tile, coupled together with the reuse of the tile across the chip, results in a reduced design complexity, when compared to conventional single-core processor systems.

7.1 NoC Design Challenges for CMPs

The systems that utilize NoCs can be broadly classified into two types: *Application-Specific Systems-on-Chip* (*ASSoCs*) and CMPs. In ASSoCs, single or a fixed set of applications are statically mapped onto the different processor and hardware cores in the design. The communication between the various cores is known and the interconnect architecture can be tailor-made to suit the application traffic characteristics. In all the preceding chapters, we targeted the design of such ASSoCs. On the other hand, in CMPs, general-purpose processor cores are used to run software tasks of different applications (an example shown in Figure 7.1). In such systems, the communication between the cores cannot be precharacterized, as the different application processes can be mapped differently to the cores, typically with the support of the compiler [100]. As the total system performance of CMPs is increasingly dominated by the interconnect performance [100], designing an interconnect architecture with predictable performance is critical.

In NoC-based systems for CMPs, to provide predictable performance and optimal network throughput, the bandwidth capacity of the different links of the NoC should be sufficient to support the peak rate of traffic on the links. If the network links cannot support the peak traffic that can be routed on them, then the network might experience traffic congestion. In a congested network, the latency for the traffic streams, and hence the interconnect performance will become unpredictable, which needs to be avoided for dependable system operation.

In traditional multiprocessor interconnection networks (the chip-to-chip networks), the bandwidth on the network links is limited by the number of pins that are available on the chip and all the links of the network have the same bandwidth capacity [107]. For most interconnect topologies and routing patterns, the load on the different links of the network is nonuniform. Thus, in traditional multiprocessor networks, the interconnect throughput is limited by the bottleneck links of the network [107].

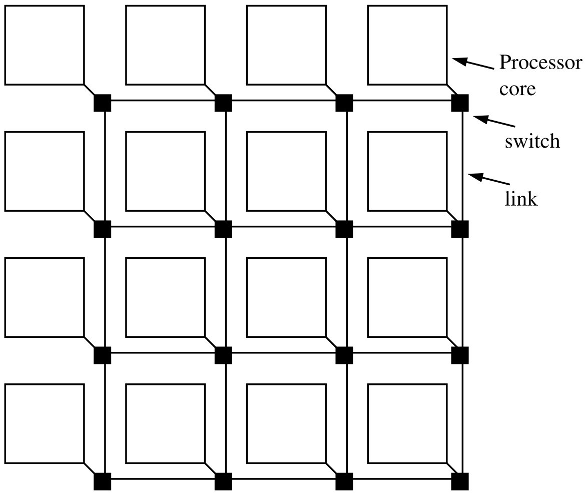


Fig. 7.1 Example tile-based CMP architecture

On the other hand, CMPs have enormous wiring resources at their disposal [22]. The links in different parts of the network can be sized differently, so that the network throughput is no longer limited by few bottleneck links. As chip designs are increasingly power consumption limited, when sizing the links, it is important to achieve a NoC design with the least power consumption.

However, in order to design such a network, there are several challenges that have to be addressed:

- The first challenge is that the exact traffic pattern of the CMP cannot be precharacterized. Usually, to evaluate the quality of the interconnection network in multiprocessors, the network is simulated with different traffic patterns, such as uniform, nearest neighbor, hot-spot, etc. If such a template of traffic patterns is used to size the links of the NoC, there is a huge drawback that the methodology is ad hoc and does not guarantee network throughput for other traffic patterns that can occur when real applications are executed.
- The second challenge is to efficiently utilize the link bandwidth (which is a product of link width and frequency) available. Traditionally, links with different bandwidth capacities are obtained by varying either their frequency of operation or their width. However, both schemes require complex frequency and width converters for potentially every input of every switch in the design. This drastically increases the design complexity and NoC area. Moreover, such designs incur significant serialization and parallelization delay at every switch, which results in high packet latencies. Thus, a way to efficiently utilize the link bandwidth is needed.
- Finally, the interconnect has to maintain a regular structure, so that a predictable and modular architecture is obtained.

In this chapter, we address the important problem of synthesizing the most power efficient NoC for CMPs that have dynamic traffic patterns, providing theoretically guaranteed optimum throughput and predictable performance for any application to be executed on the CMP.

We achieve a predictable interconnect design in two ways: First, the architecture is designed to provide predictable performance under all application traffic conditions. Second, the synthesis approach considers accurate information of the physical layer measures (such as wire-lengths, wire delays, network component delays), thereby bridging the gap between the synthesis models and the actual physical layout implementation. Thus, the design process becomes more predictable, leading to quicker design convergence.

7.2 Basics of the Synthesis Approach

To efficiently utilize the large on-chip wiring resources that are available, we use multiple physical channels for each link, namely, a link is segmented into different physical channels that can be utilized by different traffic flows in parallel. As an example, a 2×3 mesh topology is presented in Figure 7.2. Each vertex in the figure represents a switch (and the core that is connected to the switch) and a link between two vertices has one or more physical channels. For example, the link from vertex v_1 to vertex v_3 has two physical channels, while the link from vertex v_0 to vertex v_1 has one physical channel. In the synthesis process, we size the different links with different number of physical channels, such that each channel supports the load due to any traffic pattern of the NoC.

When multiple physical channels are used between two switches, if different channels are dynamically assigned to incoming packets, it may lead to out-of-delivery of packets. In this case, reorder buffers are required for ordering the packets at each receiver. Such buffers have large power and area overhead and deterministically sizing them is infeasible in practice [43]. To avoid such out-of-order delivery, for the traffic flow from each source to destination, we statically assign a single channel in every link that is used by the flow. We integrate this mapping of traffic flows to the different channels in the synthesis procedure.

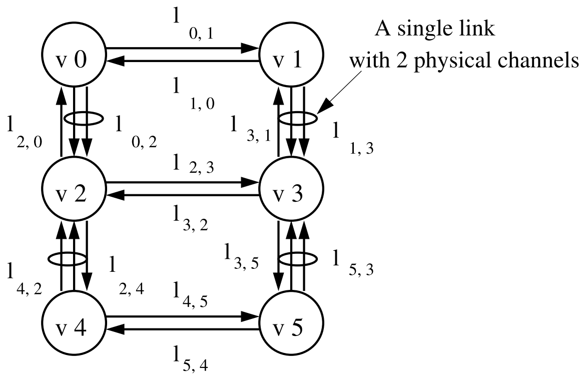


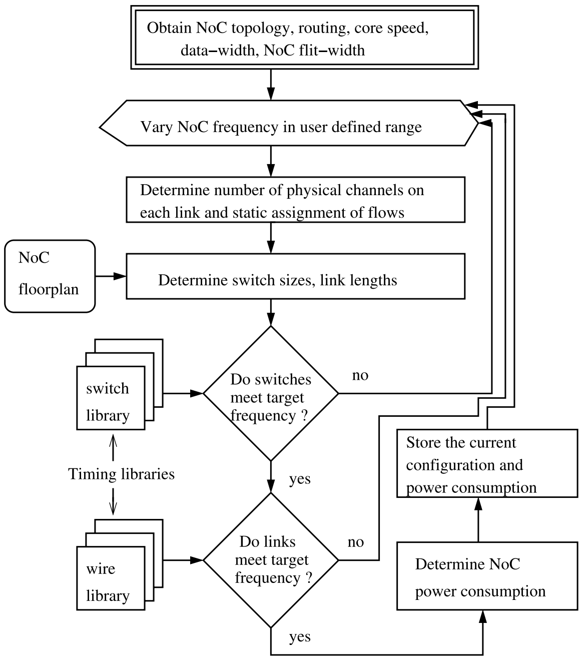
Fig. 7.2 Example 2×3 mesh topology

We also tune the setting up of NoC operating frequency during the synthesis process. To evaluate the quality of the different NoC designs, we use accurate analytical models for power consumption of the network components. The power consumption values are obtained from layouts with back-annotated resistance and capacitance information at 0.13 μm technology using standard industrial tools.

During the synthesis of the NoC, we consider the physical layer measures as well: the delay encountered on the wires in the NoC and the target frequency that can be supported by the designed network components. The synthesis approach utilizes the floorplan knowledge of the NoC to detect timing violations on the NoC links early in the design cycle. This results in a faster design cycle that leads to a reduction in the number of design re-spins and faster time-to-market, which are critical for today's complex chips. We validate the design flow predictability of the proposed approach by performing a layout of the NoC synthesized for a 25-core CMP. The approach maintains the regular and predictable structure of the NoC and is applicable in practice to existing NoC architectures.

7.3 Design Flow

In this section, we present the synthesis flow used to design the NoC (see Figure 7.3). The network topology, utilized routing function, operating frequency of the core, core data width and network link width are inputs from the user. In the



Obtain NoC topology, routing, core speed, data−width, NoC flit−width
Vary NoC frequency in user defined range
Determine number of physical channels on each link and static assignment of flows
NoC floorplan
Determine switch sizes, link lengths
switch library
Timing libraries
wire library
Do switches meet target frequency ?
no
yes
Do links meet target frequency ?
no
yes
Store the current configuration and power consumption
Determine NoC power consumption

Fig. 7.3 NoC synthesis design flow

outer loop of the synthesis process, the operating frequency of the NoC is varied in a user-defined range. For each frequency point, the number of physical channels on each link and an assignment of traffic flows to the different physical channels are computed by the synthesis process.

From the number of physical channels instantiated between the switches, the different switch sizes are obtained. Then we evaluate whether every switch of the NoC can support the corresponding frequency point (chosen in the outer loop). As the switch size increases, the maximum frequency of operation it can support reduces (as the critical path inside the switch gets longer) [34]. This information is obtained from the layout of the switches for different sizes, which is taken as an input library for the synthesis method. Then all the links in the NoC are checked for timing delay violations. For evaluating the wiring delays, we include the floorplan of the NoC as an input to the synthesis flow. Usually, standard topologies, such as mesh, are used for CMPs because the floorplan of the NoC is regular and known at design time. Based on the link lengths and wire models from [58], the delay values on the NoC links are calculated. Any timing violations on the NoC links are then evaluated by the method. If the design satisfies the timing constraints on NoC switches and links, then the power consumption of the NoC is computed, based on the layout-level power models. From the set of all feasible NoC designs, the design with the least power consumption is finally chosen by the synthesis process.

7.4 Problem Formulation

The topology of the network that defines the connectivity between the switches and the cores is taken as input. The number of physical channels used for each link is to be determined by the synthesis procedure. Formally, the NoC topology is defined by the *topology graph*:

Definition 17 The NoC topology graph is a directed graph $P(V, L)$, with each vertex $v_i \in V$ representing a core (and the switch to which it is connected) and the directed edge (v_i, v_j), denoted as $l_{i,j} \in L$, representing a link between vertices v_i and v_j. The set of physical channels that are instantiated for each link $l_{i,j}$, is represented by the set $CH_{i,j}$.

An example topology graph was presented earlier in Figure 7.2, which represents a 2×3 mesh network. The graph has 6 vertices (v_0 through v_5) and 14 links ($l_{0,1}, \ldots, l_{5,4}$). The number of physical channels used in each link varies. For example, link $l_{0,2}$ has 2 physical channels. Please note that this number is an output of the synthesis process. To begin with (when the inputs are fed), all the links are initialized to have no physical channels. Then the communication among NoC nodes can be defined as follows.

Definition 18 The communication between each pair of cores is treated as a flow of single commodity, represented as d_k, $k = 1, 2, \ldots, |V| \times |V|$, with the source of the commodity represented as $source(d_k)$ and the destination represented as $dest(d_k)$.[1]

We assume that a deterministic routing function is utilized for routing packets, as most existing NoC architectures support only a deterministic routing function [30, 33]. This is because the area-power overhead involved in adaptive routing is quite high. Moreover, adaptive routing presents several problems such as out-of-order packet delivery, which are hard to tackle in on-chip networks that need to have low power overhead. The routing function defines the set of links used by each commodity as follows.

Definition 19 The routing function R maps the traffic flows of commodities onto the links of the network, i.e., $R : d_k \to L, \forall k$. The set of links utilized by the commodity k for the routing function is represented by the set L_k.

In Figure 7.2, links $l_{1,0}$ and $l_{0,2}$ are used by the traffic flow that has vertex v_1 and source and vertex v_2 as destination, for the dimension-ordered (with x first, y next) routing scheme.

The maximum rate at which each core injects traffic into the network is also taken as an input to the synthesis engine. It is defined formally as follows.

Definition 20 The rate of traffic injection of each core, v_i, $\forall i$, is represented by r_i. The rate of each commodity d_k, represented as $rate(d_k)$, is equal to the rate of traffic injection of the source core of that commodity, i.e., $r_{source(d_k)}$.

Practically, for most CMPs, each core can inject one data word into the network every clock cycle. Thus, the injection rate is the product of the operating frequency of the core and its data width. For instance, if a core has a data width of 32 bits and operates at 100 MHz, its injection rate is 400 MB/s (i.e., 4 B $\times$ 100 MHz).

We also obtain as inputs the set of interesting operating frequencies to explore for the NoC design, and the data width of the channels (which is usually set to match the data width of the cores).

Then the *Problem Statement* is the following:

The synthesis procedure has to determine the number of channels ($|CH_{i,j}|$) required for each link ($l_{i,j}$) and a static mapping of each commodity (d_k) onto a single channel ($ch \in CH_{i,j}$) of each link $l_{i,j} \in L_k$. The mapping has to satisfy the constraint that every channel should support the traffic rates of all the commodities mapped onto that channel for any traffic pattern. The synthesis process should also determine the NoC operating frequency that results in the most power efficient NoC design.

[1] In the rest of this chapter, we follow the convention that variables i, j are defined for $1, \ldots, |V|$ and the variable k is defined for $1, \ldots, |V| \times |V|$.

An optimum (100%) throughput can be achieved if each channel supports its worst-case load, i.e., the channel bandwidth matches or exceeds the channel load. Here, we would like to point out that to practically achieve the full throughput value, the NoC architecture should have a predictable communication behavior, as in [30, 104, 105].

7.5 Synthesis Algorithm

The detailed synthesis algorithm to solve the defined problem is presented in Algorithm 6. In step 1, the NoC frequency of operation is varied in user-defined steps.

Algorithm 6 Synthesis Algorithm

```
 1: for Each NoC frequency (freq) design point in user defined range do
 2:     for each link $l_{i,j} \in L$ do
 3:         Build the Link Loading Graph ($LLG_{i,j}$) for the link $l_{i,j}$
 4:         Build the Vertex Conflict Graph ($VCG_{i,j}$) for the link $l_{i,j}$
 5:         Initialize number of channels to zero, $m = 0$
 6:         Increment $m$ by 1 and instantiate new physical channel $ch_m$
 7:         Find $m$ max-cut partitions of $VCG$
 8:         Assign bw_satisfied to true
 9:         for each max-cut partition do
10:             Build Partition Loading Graph ($PLG$)
11:             $max\_load$ = maximum_weight_matching($PLG$)
12:             If ($max\_load > freq \times width$), $bw\_satisfied$ = false
13:         end for
14:         if bw_satisfied then
15:             Assign those commodities $k$ such that $source(d_k)$ is in partition $m1$ to
                channel $m1$, $\forall m1 \in 1,\dots,m$
16:             Set $CH_{i,j} = \bigcup_{\forall m1 \in m} ch_{m1}$
17:         else
18:             Go to step 6
19:         end if
20:     end for
21:     From computed $CH_{i,j}$, $\forall i$, and $j$, compute the switch sizes
22:     Evaluate whether the switch size implementations can match the target fre-
        quency (freq)
23:     Evaluate whether all the links in the NoC can meet the target frequency (freq).
        Utilize the NoC floorplan information to estimate the link lengths
24:     If target frequency met, obtain the power consumption for the synthesized
        NoC
25: end for
26: From the set of synthesized NoCs, choose the design with least power consump-
    tion
```

Then in step 2, we consider each link individually to size the different links with different channels.

7.5.1 NoC Link Sizing

For each link $l_{i,j}$, we first build a *Link Loading Graph* (*LLG*) (in step 3), defined as follows.

Definition 21 The $LLG_{i,j}(LV, LL)$ is a bipartite graph with $|LV| = 2 \times |V|$ (i.e. with $2 \times |V|$ vertices). An edge exists between vertices lv_x and lv_y, $\forall x \in 1, \ldots, |V|$, $\forall y \in |V| + 1 \cdots 2 \times |V|$, if $\exists k$ such that $source(d_k) = lv_x$ and $dest(d_k) = lv_{y-|V|}$ and $l_{i,j} \in L_k$. The weight of the edge is the rate of traffic flow of the commodity, i.e., equal to $rate(d_k)$.

The edges of the *LLG* represent the set of all traffic flows that utilize the link, depending on whether the link is part of the route for the different traffic flows. The weights of the edges represent the rate of the traffic flows.

Example 6 The *LLG* for the link $l_{0,2}$ (i.e., $LLG_{0,2}$) of the 2×3 mesh example is presented in Figure 7.4. With x–y routing, the link $l_{0,2}$ is used by the traffic flows that originate from vertex v_0 to v_2 and v_4, and by the traffic flows that originate from vertex v_1 to v_2 and v_4. The maximum rate of all these traffic flows is 400 MB/s. In the *LLG* bipartite-graph, each vertex on both the left and the right columns represents a single core. Those vertices with traffic flows that utilize this particular link have edges between them. In this example, there are edges between those vertices that represent *core*_0 and *core*_1 with *core*_2 and *core*_4, with the edge weights being the rate of the flows.

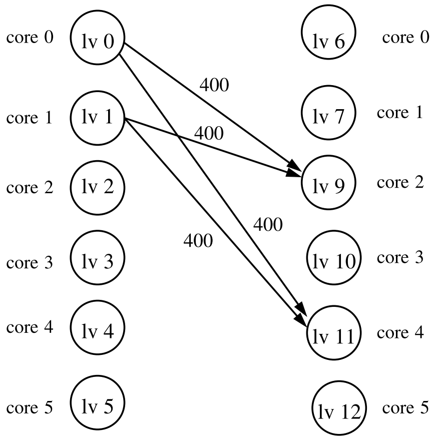


Fig. 7.4 Link loading graph. The edges are annotated with weights in MB/s

The load on a link is equal to the sum of the loads caused by each source-destination pair using that link. The worst-case link load can be obtained by considering all possible permutation traffic patterns. In [108] and [109], the authors show that the worst-case load can be obtained by representing all permutations as matchings within the *LLG* bipartite graph. A maximum-weight matching on the graph yields the exact worst-case permutation for a particular link and the worst-case (maximum) load on that link. We utilize this basic approach to evaluate the worst-case load on the different channels.

In the next step of the algorithm (step 4), we build the Vertex Conflict Graph (*VCG*), defined as follows.

Definition 22 The $VCG_{i,j}(VV, VL))$ is an undirected graph with $|VV| = |V|$ (i.e. with $|V|$ vertices). An edge $vl_{i,j}$ exists between two vertices vv_i and vv_j if $degree(lv_i) + degree(lv_j) > 0$. The weight of the edge is the value of the maximum weight bipartite matching of modified *LLG*, where the edges from all vertices other than lv_i and lv_j are removed.

The edge-weight assignment in *VCG* is such that if the traffic flows from a pair of cores (representing two vertices connected by an edge in *VCG*) are mapped onto the same physical channel, then they would together cause a maximum load on the channel that is given by the edge-weight.

Example 7 The *VCG* for link $l_{0,2}$ is presented in Figure 7.5. In $LLG_{0,2}$, as the two cores *core_0* and *core_1* have traffic flows originating from them, the edges from vertices vv_0 and vv_1 to all other vertices exist. Let us consider the edge between vv_0 and vv_5. The value of the maximum weight matching obtained on the modified *LLG* when only edges of vertex lv_0 and lv_5 are maintained is 400. Thus, the weight of the edge between vertices vv_0 and vv_5 is 400, as seen in Figure 7.5.

Then in steps 5–20, physical channels are instantiated for the link and the commodities are mapped onto the channels. The number of channels is increased from

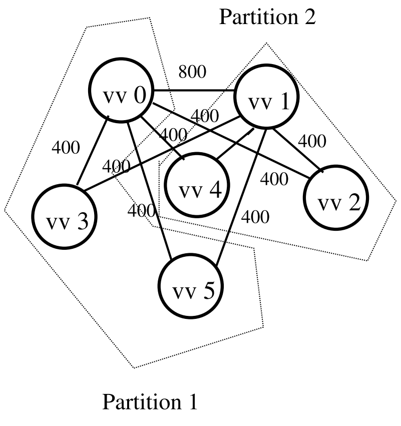


Fig. 7.5 Vertex Conflict Graph (*VCG*) and example partitions

1 until the load on each channel can be satisfied by the channel. Note that the maximum number of instantiated physical channels would be $|V|$. Thus, the traffic flows from every source that utilizes the link would be assigned to a separate channel.

For a certain number of physical channels, the $VCG_{i,j}$ is divided into that many number of partitions (step 7 of the algorithm). The partitioning is such that the sum of the edge weights cut across the partitions is maximized and the total number of vertices within each partition is almost the same. For partitioning, we use *Chaco*, an efficient hierarchical graph partitioning tool [93]. The intuition behind such partitioning is that the traffic flows that would cause higher channel loads are assigned to different channels, and channels are loaded uniformly.

Example 8 The 2 max-cut partitions of the *VCG* graph for link $l_{0,2}$ are shown in Figure 7.5. Note that vertices vv_0 and vv_1 are in different partitions.

To evaluate the load on each physical channel, we build the *Partition Loading Graph* (*PLG*) for each partition. This bipartite graph is obtained from a modified *LLG*, where the edges from all vertices other than those of the partition are removed. By finding the maximum weight matching of the *PLG*, the load caused by the partition on a channel is obtained. Then (in step 12), we check whether the load on each channel is less than or equal to the bandwidth capacity of the channel. For the channel bandwidth calculation, the data width of all the channels (*width* in Algorithm 6) is taken as an user input.

Example 9 The two *PLG* graphs for the two partitions for the mesh example are shown in Figure 7.6. The load on the two physical channels, onto which the flows from the vertices of the two partitions are mapped is 400 MB/s (obtained from the value of the maximum weight matching of each of the *PLG* graphs). If the vertices vv_0 and vv_1 had been assigned to the same partition, then the load on the channel supporting the traffic flows from the vertices of the partition would be 800 MB/s

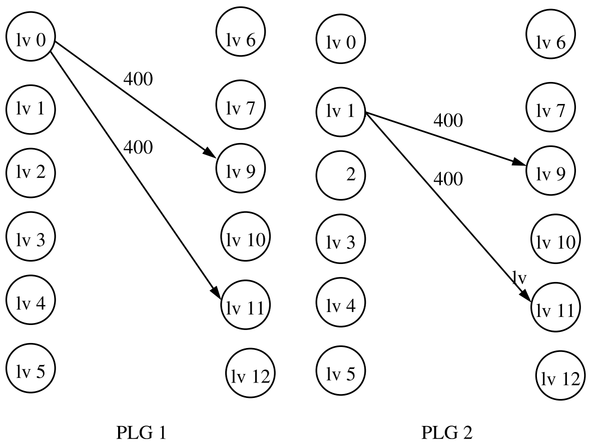


Fig. 7.6 Example physical channel loading graphs for the two partitions

(with the load on the other physical channel being 0). Thus, the partitioning process is steered to uniformly load the different channels of the link.

7.5.2 Timing Feasibility Check

In step 21 of the algorithm, the sizes of the different switches are obtained, which are based on the number of physical channels instantiated for each link. In the next step, we evaluate whether all the switches can meet the particular NoC operating frequency design point. This check is needed because, when switch size increases, the maximum supported frequency of operation reduces (as the critical path inside the switch gets longer) [34]. This information is obtained from the Place&Route of the switches, which is an input to the synthesis algorithm. Based on the frequency design point and the size of the switches, the power consumption values of the switches are obtained. For power consumption estimations, the switching activities of NoC components are obtained from several functional traffic traces. In the next step (step 26), the different links of the NoC are checked for timing violations. The length of the links are obtained from the NoC floorplan, which is taken as an input to the synthesis engine. The timing models for the interconnect wires are obtained from [58], for 0.13 μm technology.

For each frequency design point (steps 1–25, outer loop), the best NoC topology is synthesized. Finally, the most power efficient design across all these points is chosen in step 26.

7.5.3 Algorithm Run-Time

The run-time complexity of the algorithm is dominated by the maximum weight matching calculations carried out for each channel (as fast heuristics are used for partitioning, it has a low impact on the algorithm run-time). The maximum weight matching for a *PLG* graph can be computed in $O(|V|^3)$ time complexity [108] and the total number of times the matchings are performed (for each frequency design point) is at most $O(|L||V|^2)$. This is because each link can have at most $|V|$ channels and we need to perform at most $O(|V|^2)$ matchings for each link. Overall, the algorithm finds the best solution for even large CMP designs in few tens of minutes, running on a 3.2 GHz workstation.

7.6 Experimental Results

In this section, we present the experimental results obtained after applying the proposed synthesis algorithm on NoC designs with different parameter values. First, we present the application of the method to a 5×5 mesh topology. Then we study the impact of varying the data injection rates and the number of processing cores in the design. Then we perform experiments to show the effect of link lengths on the solutions produced. The generality of the method (applicability to any CMP NoC topology and deterministic routing function) is shown next, by applying it on a torus topology with two different routing functions. Finally, the design flow predictability is validated by performing a complete layout of the synthesized NoC architecture.

7.6.1 Experiments on a Mesh Topology

In this experiment, we consider a 5×5 mesh topology. We assume the operating frequency of each core is 200 MHz, the data width of the cores and NoC channels are 32-bits, and dimension-ordered (x-first, y-next) routing is utilized. We assume that the length of each NoC link to be 1 mm. We assume these as the default values and in the subsequent subsections we study the impact of varying some of these parameters.

We vary the NoC operating frequency from 200 MHz to 1 GHz and synthesize the efficient NoC for each frequency point using the proposed synthesis procedure. The total power consumption values for the synthesized NoCs (sum of switch and link power consumption) for the different frequency points are plotted in Figure 7.7. At operating frequencies lower than 400 MHz, a large number of physical channels were needed for each link, which resulted in switches with a large number of inputs and outputs. Hence, the designed switches could not support the required NoC operating frequencies. Similarly, at the 1 GHz frequency point, the designed switches

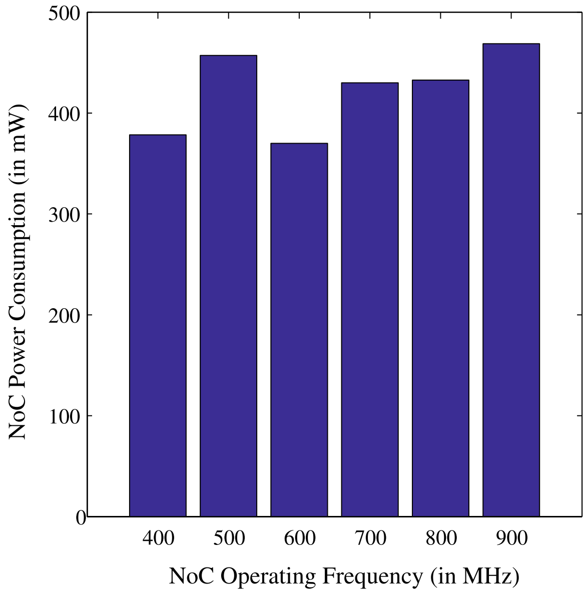


Fig. 7.7 Power consumption of 5×5 mesh topology

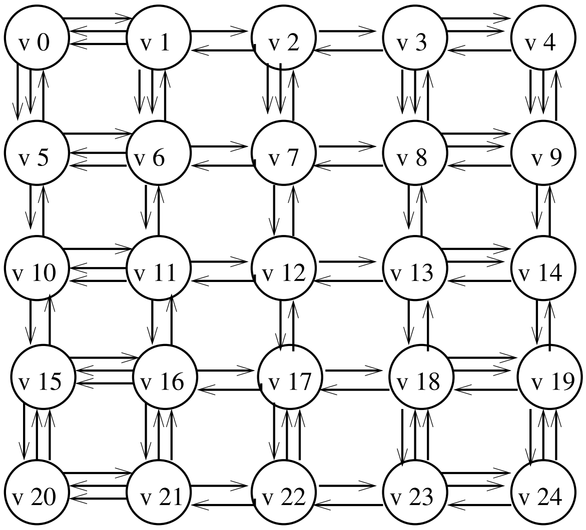


Fig. 7.8 Synthesized 5 × 5 mesh

could not support the frequency point. Thus, no feasible NoC design is obtained below 400 MHz and at 1 GHz. NoCs synthesized at lower operating frequencies (e.g., 400 MHz) require larger switches, which leads to higher power consumption. At higher operating frequencies, such as 900 MHz, the switch hardware complexity is higher (as more logic is needed to achieve faster clock speeds during physical design) and the clock-net power consumption is also higher. In fact, clock nets account for approximately 15% of NoC power consumption. Note that for the power consumption estimations of the NoC components, we run several functional traffic traces and obtain the average values. Thus, we do account for the fact that the switch input/output ports and the links of a NoC running at a higher frequency have lower switching activities than for a NoC design operating at a lower frequency. The most power optimal frequency point for the 5 × 5 mesh is 600 MHz, and the synthesized NoC at this frequency is presented in Figure 7.8.

As no previous work has directly addressed NoC design for CMPs, for comparison purposes, we evaluate how a direct extension of the approach from [108] would perform (we call this the *Reference* approach). When the procedure from [108] is applied to the 5 × 5 mesh topology, the maximum load on a link is computed to be 4× the traffic rate of each core. Thus, the NoC operating frequency required would be 800 MHz. As seen from Figure 7.7, the NoC designed using the *Reference* approach would consume 1.17× more power than the optimal NoC designed using the proposed approach.

7.6.2 Effect of Core Injection Rates

When the processor operating frequency increases, the rate of traffic injected on the NoC links also increases significantly. The actual operating frequency of the

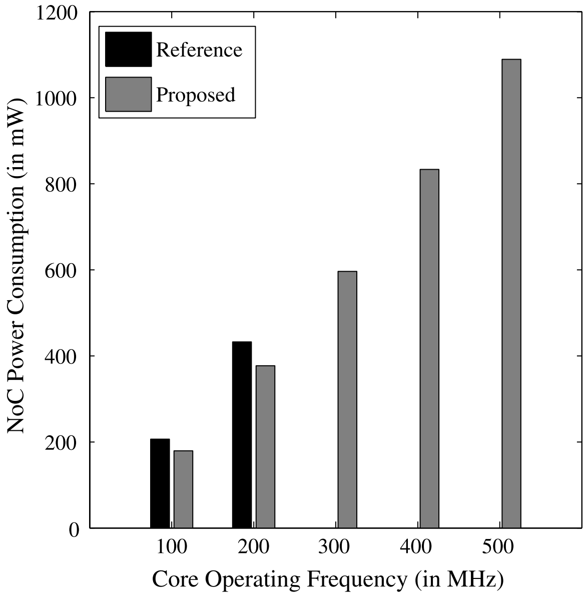


Fig. 7.9 Effect of increasing injection rates

cores varies widely across the different CMP architectures proposed in the literature. As an example, the RAW architecture has cores operating around few hundred MHz [110], while some of the commercial CMPs operate at much higher operating frequencies [103].

The power consumption requirements for different operating frequencies of the cores for the *Reference* and *proposed* approaches are depicted in Figure 7.9. This figure shows that the *Reference* approach does not produce valid NoC designs when the operating speed of the cores exceeds 200 MHz. This is because the designed NoCs needed very high operating frequency, which could not be supported by the switches. As an example, a 4×4 switch of the $\times$pipes architecture can only operate at a maximum frequency of 1 GHz approximately. While these values strongly depend on the underlying NoC architecture, the basic fact is that the *Reference* approach typically requires the NoC to be several times faster than the cores (4 times for the 5×5 mesh and higher for larger topologies). In systems where the cores themselves operate at high frequencies, it would not be feasible in practice to clock the network at such excessively high frequencies. Thus, the *Reference* approach cannot produce a valid design. On the contrary, the proposed approach supports a larger range of core operating speeds and produces more power-efficient designs as well.

7.6.3 *Effect of Different NoC Sizes*

The different CMP architectures available today have different number of tiles on the chip, and thus require NoCs of different sizes. As an example, for exploiting fine

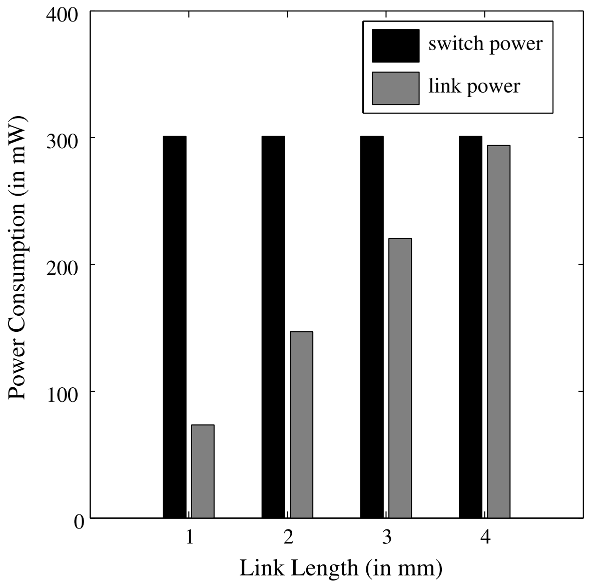


Fig. 7.10 Effect of different link lengths

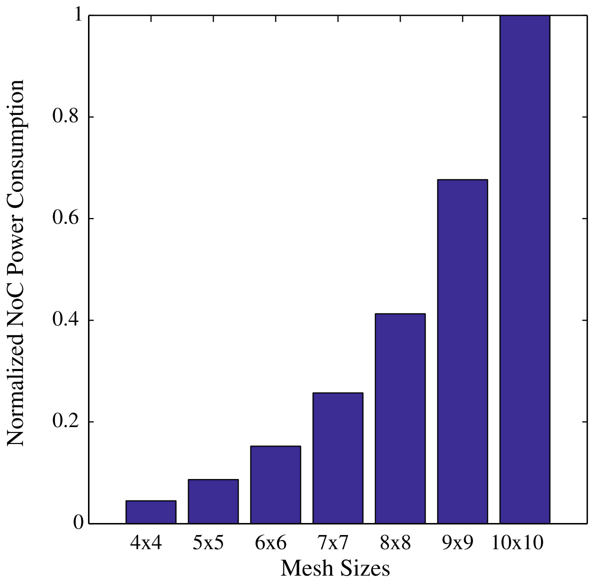


Fig. 7.11 Effect of mesh sizes

grained parallelism, CMP architectures with 50–100 tiles can be utilized, while to exploit coarse-grained parallelism, architectures with few tens of tiles are utilized [102]. In this experiment, we study the impact of different mesh sizes on the quality of the synthesized NoCs.

The NoC power consumption for different mesh sizes for the proposed synthesis approach is presented in Figure 7.11. The power numbers are normalized with respect to the power consumed by the 10 × 10 mesh design. As expected, when the mesh size increases, NoC power consumption rapidly grows as well. Even for the largest 10 × 10 mesh design, the method completed in few tens of minutes on a 3.2 GHz workstation. This shows that due to the use of fast heuristics and exact polynomial algorithms, the proposed synthesis method is highly scalable to large problem instances.

7.6.4 Effect of Link Length

To see the importance of considering wire power consumption during the synthesis process, we have varied the length of the NoC links in the design. For this experiment, we fixed the NoC topology to be a 5 × 5 mesh. Thus, the designs with different link lengths represent designs with different total chip area. For example, when the link length is 1 mm, the dimensions of the mesh NoC are 5 × 5 mm, but when the link length is 4 mm, the dimensions are 2 × 20 mm.

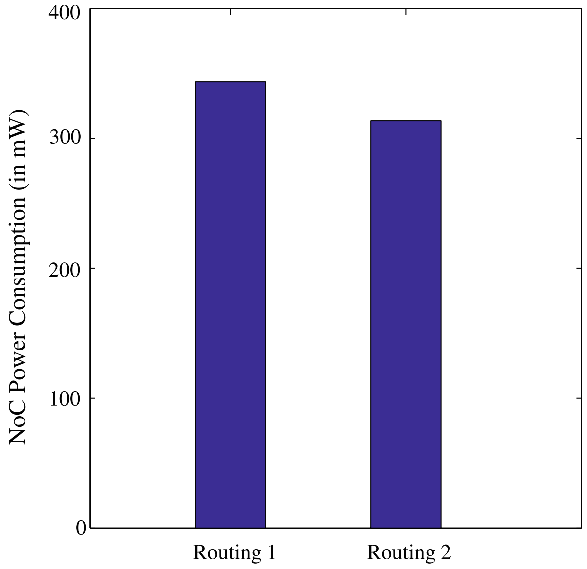
The motivation for considering different link lengths is that different CMP architectures have wires of different lengths. As an example, in the *Smart Memories* architecture [101], the link lengths of the global network are around 4 mm [111], while the link lengths in a smaller NoC design are from 1 mm to 2 mm [88].

The NoC switch and link power consumption values for different link lengths are presented in Figure 7.10. As the link length starts to increase, the link power consumption largely augments. This shows that the wire power consumption must be considered during the NoC synthesis phase, as it is done in this approach. Note that the power numbers are for 130 nm technology. With more advanced process technologies (especially at 90 nm and below), the impact of wire power consumption on the total NoC power consumption is expected to increase considerably [106]. Thus, the exploration of such technology dependent effects is a necessary direction for future work in the design of efficient on-chip interconnects.

7.6.5 Application to Torus Topology

The proposed approach is applicable to any NoC topology and deterministic routing function. We have applied it to a 5 × 5 torus topology and studied the impact of 2 different routing functions: One routing function in which the wrap-around links of the torus are not used (*Routing 1*), and another one where the wrap-around channels are utilized (*Routing 2*). The NoC power consumed by the synthesized designs for the two routing functions are shown in Figure 7.12. It shows that the use of the wrap-around links in the torus topology is beneficial, not only as generally believed for latency, but also for power. This is because when the wrap-around links are utilized, the traffic is spread more evenly in the network. Thanks to the proposed synthesis approach, this type of architectural test can be easily performed, showing its effectiveness for NoC design exploration purposes. Usually, standard NoC topologies,

Fig. 7.12 Results for torus topology



such as mesh and torus, are used for CMPs, as the NoC floorplan for such topologies is predictable [100, 101]. This is the reason for choosing these topologies for the experiments. However, the synthesis approach is general and applicable to any NoC topology.

7.6.6 Validating Design Flow Predictability

Usually, a design gap exists between the architectural level model and the actual physical layout implementation. Bridging this design gap is key to decrease the number of design iterations and to achieve quicker design convergence and faster time-to-market. In this work, we achieve a predictable design flow by bridging this design gap between the architectural and physical models. This is achieved due to two factors. First, we consider the physical layer measures, such as wire delays and accurate NoC component delays, during the synthesis process. Second, the use of regular NoC topologies results in easily predictable NoC floorplan and link lengths, which help us to accurately model the wire delays. In fact, achieving a predictable design flow is one of the most important reasons for utilizing NoC-based interconnects [88]. To validate the predictability of the design flow, we implemented the layout of the optimal 5×5 mesh topology synthesized by the procedure at 600 MHz. The CMP consists of 25 cores, and the area of each core is 1×1 mm.

To obtain the layout, we have first generated the RTL code of the designed NoC components using a custom built tool, ×ipesCompiler [62]. Then we have synthesized the RTL design using Synopsys Design Compiler [98]. After this, we have performed the place&route phase of the synthesized design using Cadence SoC Encounter [99]. The resulting layout of the design is presented in Figure 7.13. For the

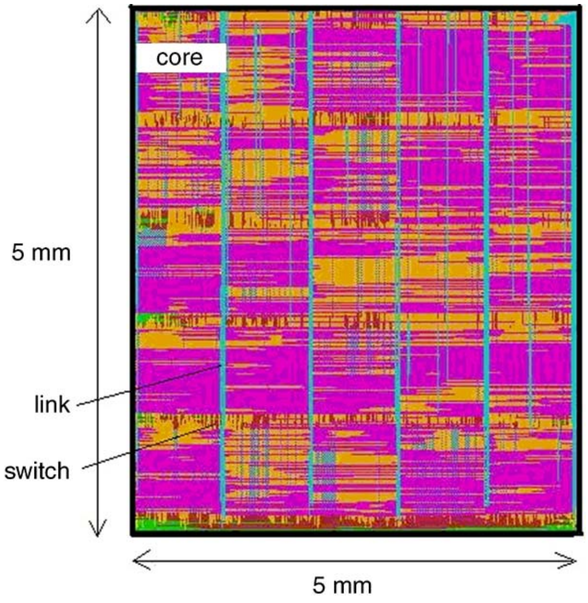


Fig. 7.13 Layout of a 5 × 5 mesh topology

layout, a 0.13 μm process technology with 8 metal layers are used for wire routing. Among these, 5 metal layers are used for intracell routing inside the cores and the remaining 3 metal layers are used for over-the-cell routing of NoC links.

We have performed post-layout timing checks on the different switches and links of the NoC. We could achieve a fully functional design at the target frequency of 600 MHz, without any timing violations. We could design the NoC till layout level quickly, thanks to the predictability of the design flow.

Finally, we studied the impact of adding multiple physical channels on NoC area. For the 5 × 5 mesh topology, the use of multiple physical channels increased the total switch area from 0.94 mm^2 (when only a single physical channel is used for all the links) to 1.18 mm^2, which is negligible when compared to the total chip area of the CMPs. From the layouts, we also found that sufficient routing area was available for the multiple physical channels that were instantiated. This is in accordance with several earlier studies [22, 88], which have shown that sufficient routing area is available between the switches of regular topologies to route a large number of wires.

7.7 Summary

Having a predictable interconnect architecture is critical to manage the increasing interconnect complexity of current *Chip Multiprocessors* (*CMPs*). The CMPs differ from *Application Specific SoCs* (*ASSOCs*) in the fact that their traffic characteristics cannot be predetermined. Thus, the NoC predictability for CMPs needs to be tack-led at several design levels. On the one hand, from the architectural viewpoint, the interconnect has to provide predictable performance under different operating conditions. On the other hand, from the design flow viewpoint, the design gap between the architectural model and the physical implementation should be minimized, so that a quicker design convergence is obtained. Designing an efficient NoC architecture that provides predictable performance for any application running on a CMP is a challenging task.

In this chapter, we have presented a synthesis method that addresses this important design issue of synthesizing the most power efficient NoC interconnect for CMPs, providing guaranteed optimum throughput and predictable performance for any application to be executed on the CMP. We achieve a predictable interconnect design in two ways: first, the architecture is designed to provide predictable performance under all application traffic conditions. Second, the synthesis approach considers accurate information of the physical layer measures, such as wire-lengths, wire delays, and network component delays, thereby bridging the gap between the synthesis models and the actual physical layout implementation. This leads to a faster design cycle and quicker design convergence across the high level synthesis approach and physical implementation of the design. We have validated the design flow predictability of the proposed approach by performing a layout of the NoC synthesized for a 25-core CMP. The proposed synthesis approach can also be used as a design space exploration tool to evaluate the efficiency of different NoC topologies and routing functions. Finally, the presented approach maintains the predictable layout of regular NoC architectures; thus, it can be applied to existing NoC architectures.

In the preceding chapters, we have seen methods to design NoC architectures under various operating conditions. Now, it is time to proceed to make their operation reliable. This will be focus of the next part of the book.