Synthesis of Predictable Networks-on-Chip-Based Interconnect Architectures for Chip Multiprocessors

Srinivasan Murali, *Student Member, IEEE*, David Atienza, *Member, IEEE*, Paolo Meloni, Salvatore Carta, Luca Benini, *Fellow, IEEE*, Giovanni De Micheli, *Fellow, IEEE*, and Luigi Raffo

***Abstract*—Today, chip multiprocessors (CMPs) that accommodate multiple processor cores on the same chip have become a reality. As the communication complexity of such multicore systems is rapidly increasing, designing an interconnect architecture with predictable behavior is essential for proper system operation. In CMPs, general-purpose processor cores are used to run software tasks of different applications and the communication between the cores cannot be precharacterized. Designing an efficient network-on-chip (NoC)-based interconnect with predictable performance is thus a challenging task. In this paper, we address the 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. In our synthesis approach, we use accurate delay and power models for the network components (switches and links) that are obtained from layouts of the components using industry standard tools. The synthesis approach utilizes the floorplan knowledge of the NoC to detect timing violations on the NoC links early in the design cycle. This leads to a faster design cycle and quicker design convergence across the high-level synthesis approach and the physical implementation of the design. We validate the design flow predictability of our proposed approach by performing a layout of the NoC synthesized for a 25-core CMP. Our approach maintains the regular and predictable structure of the NoC and is applicable in practice to existing NoC architectures.**

***Index Terms*—Bandwidth, chip multiprocessors (CMPs), networks-on-chip (NoCs), power consumption, predictability, synthesis, throughput.**

I. Introduction

WITH scaling of process technology, the operating frequency of processors has been rapidly increasing [1]. Due to the increase in clock speed, the fraction of the total chip area that is reachable within a single clock cycle is reducing. It has been estimated that in the future, it will take up to ten clock cycles for the signals to traverse across a processor die [1]. Moreover, the cost involved in designing and verifying such a complex system is so high that the price-to-performance ratio of the design increases significantly.

Manuscript received July 7, 2006; revised January 30, 2007. This work was supported in part by the U.S. National Science Foundation (NSF) under Contract CCR-0305718 for Stanford University, by the Swiss National Science Foundation (FNS) under Grant 20021-109450/1, and by the Spanish Government Research under Grant TIN2005-5619.

S. Murali is with the Computer Science Laboratory, Stanford University, Stanford, CA 94305 USA (e-mail: smurali@stanford.edu).

D. Atienza is with the Integrated Systems Laboratory, EPFL, Lausanne CH-1015, Switzerland and also with the Computer Architecture and Automation Department, Complutense University of Madrid (UCM), 28040 Madrid, Spain (e-mail: david.atienza@epfl.ch).

P. Meloni and L. Raffo are with the Dipartimento di Ingegneria Elettrica ed Elettronica (DIEE), University of Cagliari, 09123 Cagliari, Italy (e-mail: paolo.meloni@diee.unica.it; luigi@diee.unica.it).

S. Carta is with the Computer Science Department, University of Cagliari, 09123 Cagliari, Italy (e-mail: salvatore@unica.it).

L. Benini is with the Dipartimento Elettronica Informatica E Sistemistica (DEIS), University of Bologna, 40136 Bologna, Italy (e-mail: lbenini@deis.unibo.it).

G. De Micheli is with the Integrated Systems Centre, EPFL, Lausanne CH-1015, Switzerland (e-mail: giovanni.demicheli@epfl.ch).

Digital Object Identifier 10.1109/TVLSI.2007.900742

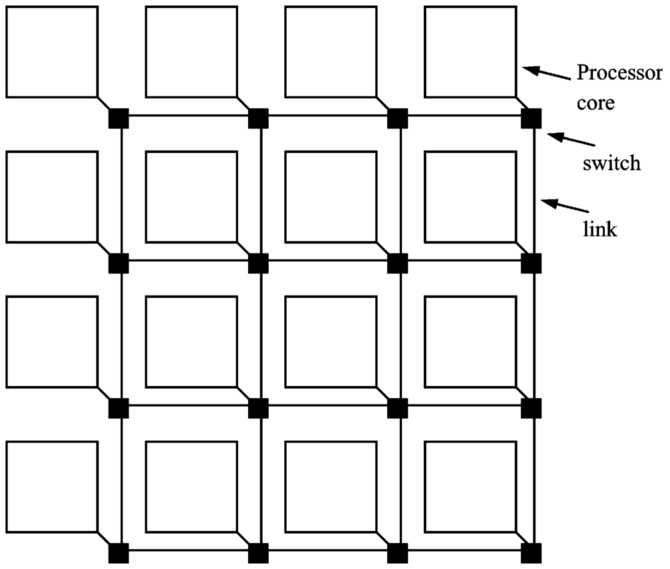


Fig. 1. Example tile-based CMP architecture.

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 [1]–[4]. 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.

The different tiles on the chip are interconnected by means of an on-chip micronetwork or networks-on-chip (NoCs). The use of NoCs to replace the global wires enables better structure, performance, and modularity [5]–[8]. An example of a mesh-based NoC is shown in Fig. 1. The NoC links between adjacent tiles in such an architecture are small and can be traversed in a single clock cycle. Moreover, the layout of the design is predictable and the wires have uniform length and electrical properties.

A. 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, a 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. On the other hand, in CMPs, general-purpose processor cores are used to run software tasks of different applications. 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 [1]. While a large body of research works exists ([21]–[28]) on designing the NoC architecture for ASSoCs, relatively few works have considered the issue of NoC design for CMPs. As the total system performance of CMPs is increasingly dominated by the interconnect performance [1], 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 [35]. 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 [35].

On the other hand, CMPs have enormous wiring resources at their disposal [7]. 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, the following several challenges need 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.

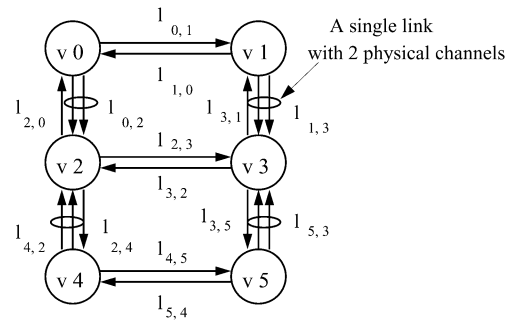


Fig. 2. Example 2×3 mesh topology.

In this paper, we address the important problem of synthesizing the most power efficient NoC for CMPs, providing theoretically guaranteed optimum throughput and predictable performance for any application to be executed on the CMP. To the best of our knowledge, this is the first paper that presents an NoC synthesis method for CMPs where the application traffic cannot be precharacterized.

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.

B. 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 Fig. 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 our synthesis process, we size the different links with a 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 [21]. 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 our synthesis procedure.

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 respins and faster time-to-market, which are critical for today's complex chips. We validate the design flow predictability of our proposed approach by performing a layout of the NoC synthesized for a 25-core CMP. Our approach maintains the regular and predictable structure of the NoC and is applicable in practice to existing NoC architectures.

II. Previous Work

Today, several academic as well as commercial CMP architectures exist [1]–[4]. The use of scalable micronetworks to replace the global wiring of CMPs and ASSoCs has been presented in [5]–[8]. The advantages of NoCs when compared to the state-of-the-art bus-based communication architectures have been presented in [7] and [41]. Several NoC architectures have been developed in the recent years [9]–[14]. The *Æthereal* [9] architecture addresses the issue of building an on-chip network with predictable performance by providing quality-of-service (QoS) guarantees to traffic streams. The architecture supports time-division multiple access (TDMA) of the network components across the different traffic streams, with the TDMA time-slots allocated to match the application characteristics. In [15] and [16], the use of virtual channels and tokens to achieve QoS guarantees have been presented.

Several works have addressed different aspects of the NoC design for ASSoCs. Methods to collect and analyze application traffic information for ASSoCs that can be fed as input to the NoC design process have been presented in [17] and [18]. In [21]–[23], methods to map cores onto standard NoC topologies and for computing paths for the different traffic streams, while satisfying the design constraints of the application have been presented. The mapping approaches in [22] and [23] also consider the floorplan information, so that the wire delay and power consumption values can be accurately estimated during the mapping process itself. Mapping, routing, and resource reservation for predictable NoC designs have been presented in [24]. In [19] and [20], the authors consider the problem of mapping multiple applications onto a single NoC architecture. The synthesis of application-specific irregular topologies for ASSoCs have been presented in [25]–[28]. In these works, an NoC topology that is tailor-made to suit the application-traffic characteristics is automatically synthesized. Application-specific insertion of long links in regular NoCs for known communication traffic pattern has been presented in [31]. However, all these earlier works address NoC synthesis where the application traffic is statically known and cannot be applied to synthesize the NoC for CMPs.

The design of traditional multiprocessor interconnection networks has been addressed by several researchers [34], [35]. In [35], the effect of bisection bandwidth on interconnect performance has been studied in detail. An algorithm to find the worst case throughput for interconnection networks and a method to find a routing function for achieving near optimal throughput have been presented in [38] and [39]. We show that, unlike traditional multiprocessor networks, the links of the NoCs for CMPs can be sized in a heterogeneous fashion.

Topology exploration of on-chip networks for different process technologies has been presented in [32]. The use of accurate floorplan information for estimating interconnect delay and power consumption have been used in many early works [22], [26], [27], [29], [30]. In this paper, while considering the interconnect delay in the synthesis process, we also consider accurate information from layout level on the delay values of the network components. As most CMPs use a regular NoC topology (such as a mesh topology), the wire lengths and delays are more predictable than the general bus/NoC designs considered in these earlier works.

III. Design Flow

In this section, we present the synthesis flow used to design the NoC (see Fig. 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 outer loop of the synthesis process, the operating frequency of the NoC is varied in a user-defined range. The designer usually has knowledge of the range of the frequencies that need to be explored. As an example, the designer can choose the set of possible frequencies from minimum to a maximum value, with allowed frequency step sizes. 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, which is explained in detail in Sections IV and V.

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) [37]. 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 [40], 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

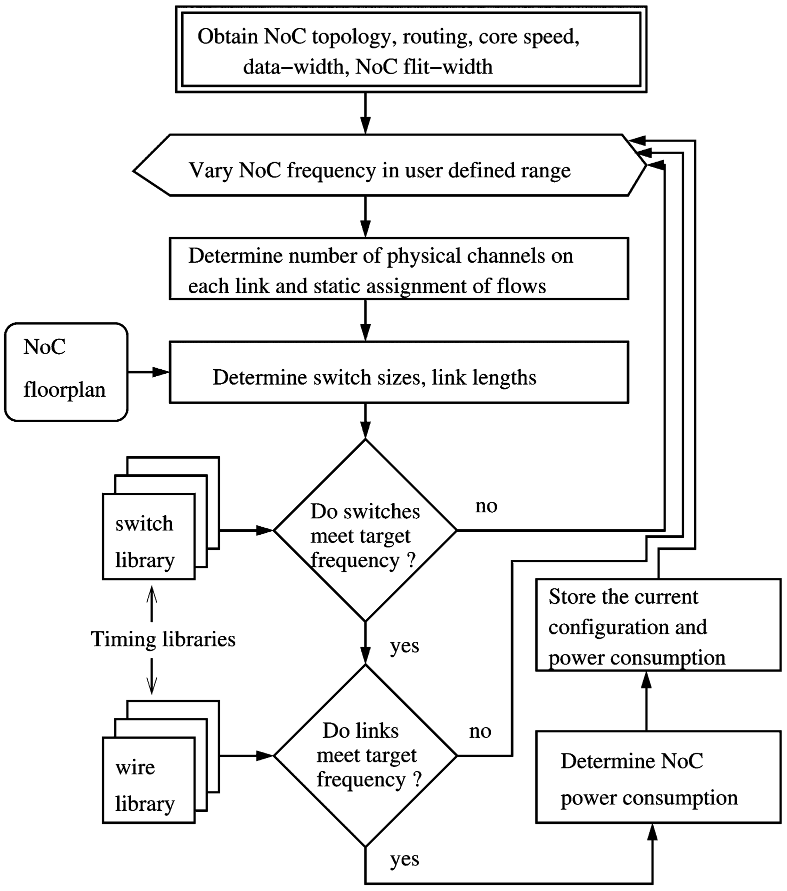

Obtain NoC topology, routing, core speed, data–width, NoC flit–width
Vary NoC frequency in user defined range
Determine number of physical channels on each link and static assignment of flows
NoC floorplan
Determine switch sizes, link lengths
switch library
Do switches meet target frequency ?
no
yes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

Fig. 3. NoC synthesis design flow.

of all feasible NoC designs, the design with the least power consumption is finally chosen by the synthesis process.

IV. 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 1: 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 Fig. 2, which represents a 2 × 3 mesh network. The graph has 6 vertices (v_0–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 two physical channels. Please note that this number is an output of the synthesis process. To begin with, all the links are initialized to have no physical channels. Then, the communication among NoC nodes can be defined as follows.

Definition 2: The communication between each pair of cores is treated as a flow of single commodity, represented as d_k, $k = 0, 1, \ldots, |V| \times |V|$, with the source of the commodity represented as $\mathrm{source}(d_k)$ and the destination represented as $\mathrm{dest}(d_k)$.[1]

We assume that a deterministic routing function is utilized for routing packets, as most existing NoC architectures support only a deterministic routing function [9], [12], [13]. 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 in the following.

Definition 3: 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 Fig. 2, links $l_{1,0}$ and $l_{0,2}$ are used by the traffic flow that has vertex v_1 as 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 4: 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

[1]In the rest of this paper, we follow the convention that variables i and j are defined for $0, \ldots, |V|-1$ and the variable k is defined for $0, \ldots, |V| \times |V| - 1$.

as $\mathrm{rate}(d_k)$, is equal to the rate of traffic injection of the source core of that commodity, i.e., $r_{\mathrm{source}(d_k)}$.

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., 4B $\times$ 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 as follows.

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.

In order to solve this problem, we need to determine the following two things: the number of physical channels needed between the switches and a mapping of the traffic flows onto the channels. Please note that the complexity of the problem is due to the fact that the worst case traffic pattern that can load each link has to be determined and this process is not trivial. Moreover, the number of physical channels needed between the switches cannot be obtained by merely dividing the worst case traffic rate on the link by the bandwidth available on a physical channel. This is because, in this case, the worst case traffic may not be evenly spread across the different physical channels. Thus, while some of the channels may be under utilized, some other channels will be overloaded. So, a procedure that evenly spreads the traffic on the different channels and guarantees that the worst case traffic rate can be supported by each of the channels is needed.

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 [9], [15], and [16].

V. Synthesis Algorithm

The detailed synthesis algorithm to solve the defined problem is presented in Algorithm 1. In step 1, the NoC frequency of operation is varied in user-defined steps. Then, in step 2, we consider each link individually to size the different links with different channels.

Algorithm 1 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 $bwsat$ to true
 9:     for each max-cut partition do
10:       Build Partition Loading Graph (PLG)
11:       $\mathrm{max\_load} = \mathrm{maximum\_weight\_matching}$ (PLG)
12:       If $(\mathrm{max\_load} > \mathrm{freq} \times \mathrm{width})$, $bwsat = \mathrm{false}$
13:     end for
14:     if $bwsat$ then
15:       Assign those commodities $k$ such that $\mathrm{source}(d_k)$
          is in partition $m1$ to channel $m1, \forall\, m1 \in 1 \ldots m$
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 frequency (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 consumption
```

A. NoC Link Sizing

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

Definition 5: The $\mathrm{LLG}_{i,j}(LV, LL)$ 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, $\forall\, x \in 0 \ldots |V| - 1$, $\forall\, y \in |V| \ldots 2 \times |V| - 1$, if $\exists\, k$ such that $\mathrm{source}(d_k) = lv_x$ and $\mathrm{dest}(d_k) = lv_{y-|V|}$ 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 $\mathrm{rate}(d_k)$.

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.

1) Example 1: The LLG for the link $l_{0,2}$ (i.e., $\mathrm{LLG}_{0,2}$) of the 2×3 mesh example is presented in Fig. 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

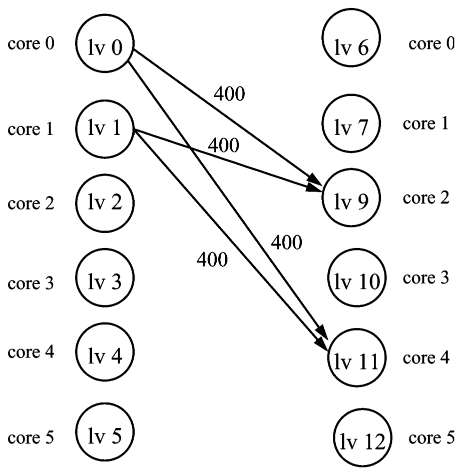


Fig. 4. Link loading graph. The edges are annotated with weights in megabytes per second.

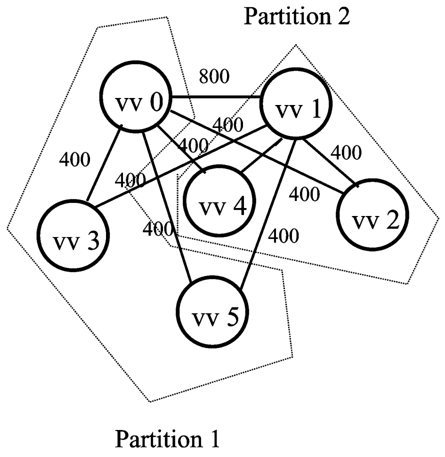


Fig. 5. VCG and example partitions.

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.

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 [38], 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 step 4 of the algorithm, we build the vertex conflict graph (VCG), defined as follows.

Definition 6: The $\mathrm{VCG}_{i,j}(VV, VL)$ 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 $\mathrm{degree}(lv_i) + \mathrm{degree}(lv_j) > 0$. 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.

2) Example 2: The VCG for link $l_{0,2}$ is presented in Fig. 5. In $\mathrm{LLG}_{0,2}$, 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 shown in Fig. 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 one 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 $\mathrm{VCG}_{i,j}$ 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 [36]. 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.

3) Example 3: The two max-cut partitions of the VCG graph for link $l_{0,2}$ are shown in Fig. 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 1) is taken as an user input.

4) Example 4: The two PLG graphs for the two partitions for the mesh example are shown in Fig. 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 (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.

B. 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

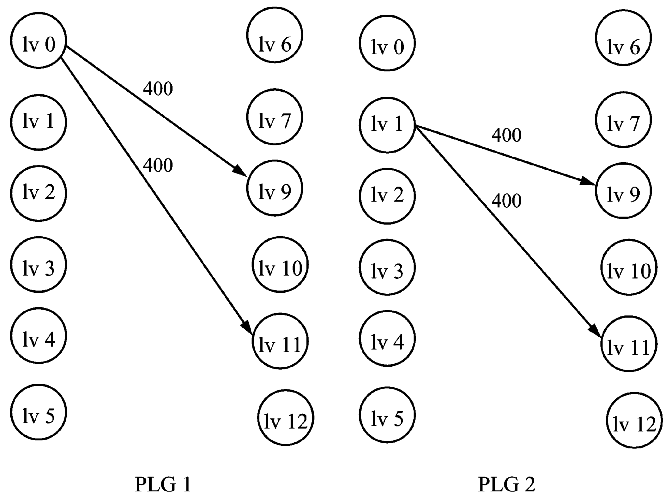


Fig. 6. Example physical channel loading graphs for the two partitions.

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) [37]. 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 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 [40], for a 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.

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 [38] 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 (see Section VII for more details).

VI. NoC Power and Delay Models

We built accurate analytical models for the power consumption of the network components based on the $\times$*pipes* architecture [37]. To get the power estimates, the place&route of the components is performed using Cadence SoC Encounter[2] and accurate wire capacitances and resistances are obtained, as back-annotated information from the layout, with 0.13-μm technology library. The switching activity in the network components is varied by injecting functional traffic. The capacitance, resistance, and the switching activity report are combined to estimate power consumption, using Synopsys PrimePower.[3]

TABLE I
Component Area-Power

Component	Parameter	Analytical	Experimental
4x4 switch	power(mW)	22.16	22.54
5x5 switch	power(mW)	28.38	28.70
link (1 mm)	power(mW)	8.7	8.7

Several implementation runs were performed, varying parameters such as the number of input, output ports, and the amount of switching activity at the layout level. These runs were performed by individually varying each one of these parameteres, while fixing the others. Linear regression was used to build analytical models for the power consumption of the components as a function of these parameters. Due to the intrinsic modularity and symmetry of NoC components, the models built are very accurate (with maximum and mean error of less than 7% and 5%, respectively) when compared to the actual values. Power consumption on the wires are obtained from the models presented in [40]. The analytical and experimental power consumption values for some of the NoC components (with 900-MHz frequency, link width of 32 bits, buffer depth of 3 in switches, and 50% switching activity at all components) are presented in Table I. For the total NoC power consumption, we also include the power consumption of the clock-tree for the clock supplied to the NoC.

When the size of a NoC switch increases, its maximum supported frequency of operation decreases, as the critical path inside the switch increases. From the layouts, we also obtain the maximum frequency of operation of the switches for different sizes, which is used in the synthesis process to prune infeasible designs. Similarly, the maximum delay of the wires for different lengths are obtained from the models presented in [40] and utilized in the synthesis process.

VII. 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.

[2][Online]. Available: www.cadence.com

[3][Online]. Available: www.synopsys.com

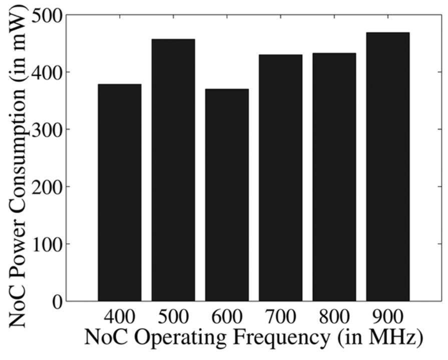


Fig. 7. Power consumption of 5 × 5 mesh topology.

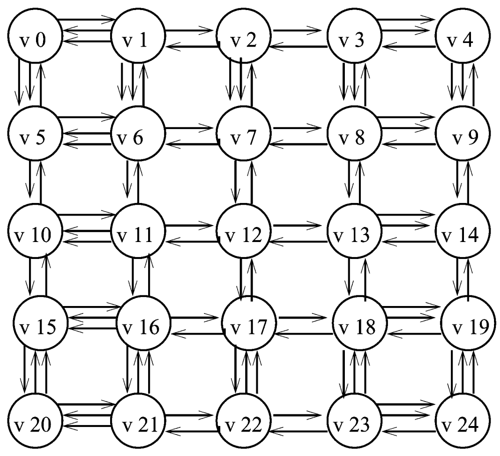


Fig. 8. Synthesized 5 × 5 mesh.

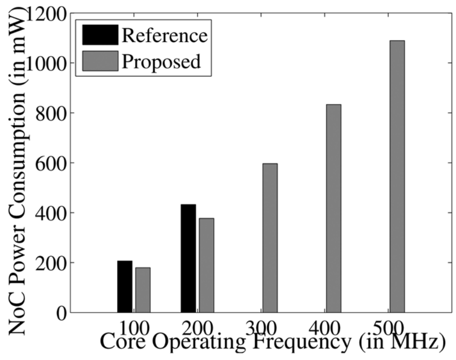


Fig. 9. Effect of increasing injection rates.

A. 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 Fig. 7. In Fig. 7, no result is shown below 400 MHz and at 1 GHz operating frequency. This is because, at operating frequencies lower than 400 MHz, a large number of physical channels were needed for each link. This resulted in switches with a large number of inputs and outputs, and the designed switches could not support the required NoC operating frequencies. Similarly, at the 1-GHz frequency point, the designed switches could not support the frequency point.

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 Fig. 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 [38] would perform (we call this the *Reference* approach). When the procedure from [38] 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 Fig. 7, the NoC designed using the *Reference* approach would consume 1.17× more power than the optimal NoC designed using our proposed approach.

B. 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 cores varies widely across the different CMP architectures proposed in the literature. As an example, the RAW architecture has cores operating around few hundred megahertz [43], while some of the commercial CMPs operate at much higher operating frequencies [4].

The power consumption requirements for different operating frequencies of the cores for the *Reference* and *proposed* approaches are depicted in Fig. 9. Fig. 9 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 frequencies, which could not be supported by the switches. 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× 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, our approach supports a larger range

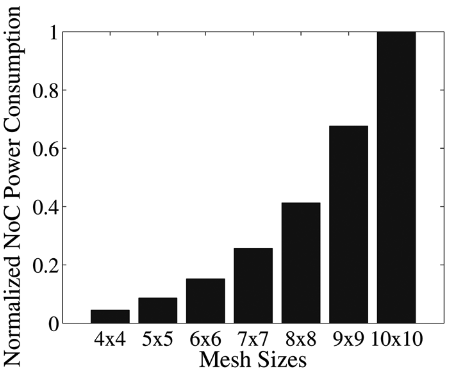


Fig. 10. Effect of mesh sizes.

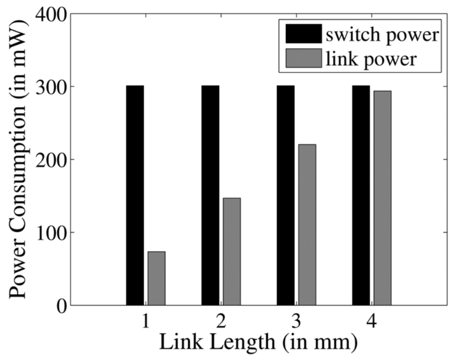


Fig. 11. Effect of different link lengths.

of core operating speeds and produces more power-efficient designs as well.

C. 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 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 [3]. In this experiment, we study the impact of different mesh sizes on the quality of the synthesized NoCs.

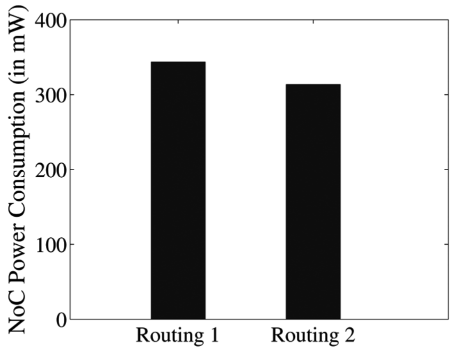
The NoC power consumption for different mesh sizes for the proposed synthesis approach is presented in Fig. 10. The power numbers are normalized with respect to the power consumed by the 10×10 mesh design. As expected, when the mesh size increases, NoC power consumption rapidly grows as well. Even for the largest 10×10 mesh design, our 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.

D. 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 mm $\times$ 5 mm, but when the link length is 4 mm, the dimensions are 20 mm $\times$ 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 [2], the link lengths of the global network are around 4 mm [44], while the link lengths in a smaller NoC design are from 1 to 2 mm [41].

The NoC switch and link power consumption values for different link lengths are presented in Fig. 11. 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 our approach.



3.2in

Fig. 12. Results for torus topology.

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 [32]. Thus, the exploration of such technology dependent effects is a necessary direction for future work in the design of efficient on-chip interconnects.

E. Application to Torus Topology

Our 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 two 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 Fig. 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 our synthesis approach, this type of architectural test can be easily performed, showing its effectiveness for NoC design exploration purposes. Usually, standard NoC topologies, such as mesh and torus, are used for CMPs, as the NoC floorplan for such topologies is predictable [1], [2]. This is the reason for choosing these topologies for our experiments. However, our synthesis approach is general and applicable to any NoC topology.

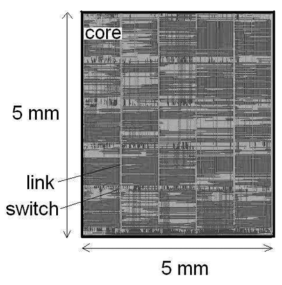


Fig. 13. Layout of a 5 × 5 mesh topology.

F. 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 our paper, 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 [41]. To validate the predictability of the design flow, we implemented the layout of the optimal 5 × 5 mesh topology synthesized by our procedure at 600 MHz (refer to Section VII-A). The CMP consists of 25 cores, and the area of each core is 1 mm × 1 mm.

To obtain the layout, we have first generated the RTL code of the designed NoC components using a custom built tool, × ipesCompiler [42]. Then, we have synthesized the RTL design using Synopsys Design Compiler.[3] After this, we have performed the place&route phase of the synthesized design using Cadence SoC Encounter.[2] The resulting layout of the design is presented in Fig. 13. For the layout, a 0.13-μm process technology with eight metal layers are used for wire routing. Among these, five metal layers are used for intra-cell routing inside the cores and the remaining three 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 to the 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 our 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 [7], [41], which have shown that sufficient routing area is available between the switches of regular topologies to route a large number of wires.

VIII. Conclusions and Future Work

Having a predictable interconnect architecture is critical to manage the increasing interconnect complexity of current CMPs. Interconnect predictability needs to be tackled 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. The use of NoCs to replace the point-to-point or bus-based global wiring aims at helping with both aspects, by leading to modular interconnect architectures with predictable layouts. However, designing an efficient NoC architecture that provides predictable performance for any application running on a CMP is a challenging task.

In this paper, 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 our proposed approach by performing a layout of the NoC synthesized for a 25-core CMP. Our 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, our approach maintains the predictable layout of regular NoC architectures; Thus, it can be applied to existing NoC architectures.

There are several interesting directions for future research using our approach. First, the impact of different NoC topologies and routing functions for CMPs of different sizes can be explored. Second, the impact of different technology generations on the choice of the topologies and routing functions can be analyzed. An earlier study (see [32]) has tried to address some of these topics, but without considering the requirements for predictable performance. Finally, the design flow predictability of nonregular topologies can be investigated since it would be interesting to see how well nonregular topologies perform when compared to regular topologies for CMPs.

References

[1] M. Taylor *et al.*, "The raw microprocessor: A computational fabric for software circuits and general purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Apr. 2002.
[2] K. Mai *et al.*, "Smart memories: A modular reconfigurable architecture," in *Proc. ISCA*, 2000, pp. 161–171.

[3] K. Sankaralingam *et al.*, "Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture," *IEEE Micro*, vol. 23, no. 6, pp. 46–51, Nov./Dec. 2003.

[4] R. Kalla *et al.*, "IBM Power5 chip: A dual-core multithreaded processor," *IEEE Micro*, vol. 24, no. 2, pp. 40–47, Mar./Apr. 2004.

[5] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computers*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[6] D. Wingard, "Micronetwork-based integration for SoCs," *Proc. DAC*, pp. 673–677, 2001.

[7] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," *Proc. DAC*, pp. 684–689, 2001.

[8] M. Sgroi *et al.*, "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proc. DAC*, 2001, pp. 667–672.

[9] E. Rijpkema *et al.*, "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip," *Proc. DATE*, pp. 350–355, 2003.

[10] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet switched interconnections," *Proc. DATE*, pp. 250–256, 2000.

[11] S. Kumar *et al.*, "A network on chip architecture and design methodology," in *Proc. ISVLSI*, 2002, pp. 105–112.

[12] D. Bertozzi *et al.*, "NoC synthesis flow for customized domain specific multi-processor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, Feb. 2005.

[13] D. Siguenza-Tortosa *et al.*, "Issues in the development of a practical NoC: The Proteo concept," *VLSI Integr. J.*, vol. 38, pp. 95–105, 2004.

[14] E. Bolotin *et al.*, "QNOC: QoS architecture and design process for network on chip," *J. Syst. Arch.*, vol. 50, no. 2–3, pp. 105–128, Feb. 2004.

[15] T. Bjerregaard and J. Sparso, "Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip," in *Proc. NORCHIP*, 2004, pp. 269–272.

[16] M. Millberg *et al.*, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Proc. DATE*, 2004, pp. 20890–20895.

[17] K. Lahiri *et al.*, "Design space exploration for optimizing on-chip communication architectures," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 6, pp. 952–961, Jun. 2004.

[18] S. Murali and G. D. Micheli, "An application-specific design methodology for STBUS crossbar generation," in *Proc. DATE*, 2005, pp. 1176–1181.

[19] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. D. Micheli, "Mapping and configuration methods for multi-use-case networks on chips," in *Proc. ASPDAC*, 2006, pp. 146–151.

[20] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. D. Micheli, "A methodology for mapping multiple use-cases onto networks on chips," *Proc. DATE*, pp. 118–123, 2006.

[21] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," *Proc. DATE*, pp. 10688–106993, 2003.

[22] S. Murali and G. D. Micheli, "Sunmap: A tool for automatic topology selection and generation for NoCs," in *Proc. DAC*, 2004, pp. 914–919.

[23] S. Murali *et al.*, "Mapping and physical planning of networks-on-chip with quality-of-service guarantees," in *Proc. ASPDAC*, 2005, pp. 27–32.

[24] A. Hansson *et al.*, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proc. ISSS*, 2005, pp. 75–80.

[25] A. Pinto *et al.*, "Efficient synthesis of networks on chip," in *Proc. ICCD*, 2003, pp. 146–150.

[26] T. Ahonen *et al.*, "Topology optimization for application specific networks on chip," in *Proc. SLIP*, 2004, pp. 53–60.

[27] K. Srinivasan *et al.*, "An automated technique for topology and route generation of application specific on-chip interconnection networks," in *Proc. ICCAD*, 2005, pp. 231–237.

[28] W. H. Ho and T. M. Pinkston, "A methodology for designing efficient on-chip interconnects on well-behaved communication patterns," in *Proc. HPCA*, 2003, pp. 377–388.

[29] J. Hu *et al.*, "System-level point-to-point communication synthesis using floorplanning information," in *Proc. ASPDAC*, 2002, pp. 573–579.

[30] S. Pasricha *et al.*, "Floorplan-aware automated synthesis of bus-based communication architectures," in *Proc. DAC*, 2005, pp. 565–570.

[31] U. Y. Ogras and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion," in *Proc ICCAD*, 2005, pp. 246–253.

[32] H. Wang *et al.*, "A technology-aware and energy-oriented topology exploration for on-chip networks," in *Proc. DATE*, 2005, pp. 1238–1243.

[33] N. McKeown *et al.*, "Achieving 100% throughput in an input-queued switch," in *Proc. INFOCOM*, 1996, pp. 296–302.

[34] V. K. Adve and M. K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing," *IEEE TPDS*, pp. 225–246, 1994.

[35] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Parrellel Distrib. Syst.*, pp. 775–785, 1990.

[36] B. Hendrickson and R. Leland, "The Chaco user's guide: Version 2.0," (1994). [Online]. Available: //www.cs.sandia.gov/~bahendr/chaco.html

[37] S. Stergiou *et al.*, "× pipesLite: A Synthesis oriented design library for networks on chips," *Proc. DATE*, pp. 1188–1193, 2005.

[38] B. Towles and W. J. Dally, "Worst-case traffic for oblivious routing functions," *Proc. SPAA*, pp. 1–8, 2002.

[39] B. Towles *et al.*, "Throughput-centric routing algorithm design," *Proc. SPAA*, pp. 200–209, 2003.

[40] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[41] F. Angiolini *et al.*, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," *Proc. DATE*, pp. 124–129, 2006.

[42] A. Jalabert *et al.*, "× pipesCompiler: A tool for instantiating application specific networks on chip," in *Proc. DATE*, 2004, pp. 20884–20889.

[43] M. B. Taylor, "The RAW processor specification," (1997). [Online]. Available: http://cagwww.lcs.mit.edu/raw/documents/index.html

[44] R. Ho, K. Mai, and M. Horowitz, "Efficient on-chip global interconnects," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2003, pp. 271–274.

Srinivasan Murali (S'02) received the B.S. degree (with a gold medal) in computer science and engineering from the University of Madras, Madras, India, in 2002. He is currently pursuing the Ph.D. degree in electrical engineering at Stanford University, Stanford, CA.

His research interests include reliable and efficient design methods for networks-on-chips and systems-on-chips.

Mr. Murali was a recipient of a Best Paper Award in the DATE Conference in 2005.

David Atienza (M'05) received the M.Sc. and Ph.D. degrees in computer science from Complutense University of Madrid (UCM), Madrid, Spain, in 2001 and 2005, respectively.

Currently he is an Associated Postdoctoral Researcher with the Integrated Systems Laboratory (LSI), EPFL, Lausanne, Switzerland. He also holds the position of an Invited Associate Professor at the Computer Architecture and Automation Department (DACYA), UCM. His research interests include several aspects of design technologies for integrated circuits and systems, with a particular emphasis on dynamic memory management on embedded systems, flexible networks-on-chip (NoC) interconnection paradigms for multiprocessors system-on-chip, design automation, and low-power design. In these fields, he is reviewer and coauthor of more than 70 publications in prestigious journals and international conferences, such as ACM TODAES, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, *VLSI Journal*, *Journal of Embedded Systems*, DATE, DAC, etc. He is also a part of the Technical Program Committee of the IEEE/ACM DATE and ICCAD conferences.

Paolo Meloni received the M.S. degree in electrical engineering from the University of Cagliari, Cagliari, Italy, in 2004, where he is currently pursuing the Ph.D. degree in the electrical and electronics engineering.

His research interests are mostly focused upon advanced digital designs and development of network-on-chip architectures.

Salvatore Carta received the B.S. degree (*summa cum laude*) in electronic engineering and the Ph.D. degree in electronics and computer science from the University of Cagliari, Cagliari, Italy, in 1997 and 2003, respectively.

In 2005, he became an Assistant Professor with the Computer Science Department, University of Cagliari. His research interests mainly include architectures, software and tools for embedded and portable computing, with particular emphasis on operating systems, middleware and applications modeling for multiprocessor-systems-on-chips, networks-on-chip, and reconfigurable computing. He is the author of several papers in these fields.

Luca Benini (S'94–M'97–SM'04–F'06) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1997.

He is currently a Professor with the University of Bologna, Bologna, Italy. He also holds a Visiting Faculty Position with the Ecole Polytecnique Federale de Lausanne (EPFL), Lausanne, Switzerland. His research interests include the design of systems for ambient intelligence, from multiprocessor systems-on-chip/networks on chip to energy-efficient smart sensors and sensor networks. From there, his research interests have spread into the field of biochips for the recognition of biological molecules, into bioinformatics for the elaboration of the resulting information, and further into more advanced algorithms for in silicobiology. He has published more than 300 papers in peer-reviewed international journals and conferences, three books, several book chapters, and two U.S. patents.

Dr. Benini has been Program Chair and Vice-Chair of Design Automation and Test in Europe Conference. He has been a member of the 2003 MEDEA and EDA Roadmap Committee 2003. He is a member of the IST Embedded System Technology Platform Initiative (ARTEMIS), a working group on Design Methodologies, a member of the Strategic Management Board of the ARTIST2 Network of Excellence on Embedded System, and a member of the Advisory Group on Computing Systems of the IST Embedded Systems Unit. He has been member of the Technical Program Committee and organizing committee of several technical conferences, including the Design Automation Conference, International Symposium on Low Power Design, and the Symposium on Hardware-Software Codesign. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF CIRCUITS AND SYSTEMS and of the *ACM Journal on Emerging Technologies in Computing Systems*.

Giovanni De Micheli (S'79–M'79–SM'80–F'94) is a Professor and Director of the Integrated Systems Centre, EPF Lausanne, Lausanne, Switzerland, and the President of the Scientific Committee of CSEM, Neuchatel, Switzerland. Previously, he was a Professor with the Electrical Engineering Department, Stanford University, Stanford, CA. His research interests include several aspects of design technologies for integrated circuits and systems, with a particular emphasis on synthesis, system-level design, hardware/software co-design, and low-power design. He is the author of *Synthesis and Optimization of Digital Circuits* (McGraw-Hill, 1994), coauthor and/or co-editor of six other books, and of over 300 technical articles. He is, or has been, a member of the Technical Advisory Board of several companies, including Magma Design Automation, Coware, Aplus Design Technologies, Ambit Design Systems, and STMicroelectronics.

Dr. De Micheli was a recipient of the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems, the Golden Jubilee Medal for outstanding contributions to the IEEE Circuits and Systems (CAS) Society in 2000, the D. Pederson Award for the Best Paper on the IEEE Transactions on CAD/ICAS in 1987, two Best Paper Awards at the Design Automation Conference, in 1983 and in 1993, and a Best Paper Award at the DATE Conference in 2005. He was the President of the IEEE CAS Society in 2003, and he is currently the President Elect of the IEEE Council on EDA and chairing the IEEE Product Package Committee. He was the Program Chair of the pHealth and VLSI SOC Conferences in 2006. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 1997 to 2001, and he was the Program Chair and General Chair of the Design Automation Conference (DAC) in 1996–1997 and 2000, respectively. He is a Fellow of ACM.

Luigi Raffo received the M.S. and Ph.D. degrees in electronics and computer science from University of Cagliari, Cagliari, Italy, in 1989 and 1994, respectively.

In 1994, he was an Assistant Professor with the Department of Electrical and Electronics Engineering, University of Cagliari, Cagliari, Italy, where, in 1998, he was a Professor with the Digital System Design, Integrated Systems Architectures and Microelectronics. His research interests lie mainly in the design of low-power analog and digital architectures/chips. He has been a Project Manager of many local and international projects. He is an author of more than 50 international papers in the field.