SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3-D Systems on Chips

Ciprian Seiculescu, Srinivasan Murali, Luca Benini, *Fellow, IEEE,* and Giovanni De Micheli, *Fellow, IEEE*

***Abstract*—Three-dimensional integrated circuits (3D-ICs) are a promising approach to address the integration challenges faced by current systems on chips (SoCs). Designing an efficient network on chip (NoC) interconnect for a 3-D SoC that meets not only the application performance constraints but also the constraints imposed by the 3-D technology is a significant challenge. In this paper, we present a design tool, *SunFloor 3D*, to synthesize application-specific 3-D NoCs. The proposed tool determines the best NoC topology for the application, finds paths for the communication flows, assigns the network components to the 3-D layers, and places them in each layer. We perform experiments on several SoC benchmarks and present a comparative study between 3-D and 2-D NoC designs. Our studies show large improvements in interconnect power consumption (average of 38%) and delay (average of 13%) for the 3-D NoC when compared to the corresponding 2-D implementation. Our studies also show that the synthesized topologies result in large power (average of 54%) and delay savings (average of 21%) when compared to standard topologies.**

***Index Terms*—3-D integrated circuits (3D-ICs), networks on chip (NoC), placement, synthesis, topology.**

I. Introduction

2-D CHIP fabrication technology is facing lot of challenges in utilizing the exponentially growing number of transistors on a chip. Wire delay and power consumption is increasing dramatically and achieving interconnect design closure is more and more a challenge. Moreover, diverse components that are digital, analog, microelectromechanical systems, and radio frequency modules are being integrated on the same chip, resulting in large complexity for the 2-D manufacturing process [18].

Vertical stacking of multiple silicon layers, referred to as 3-D stacking, is emerging as an attractive solution to continue the pace of growth of systems on chips (SoCs) [18]–[23]. Several technologies and methods are available in the paper for performing 3-D integration. In the *Die-to-Die* bonding process, individual dies are glued together to form the 3D-IC. In the *Die-to-Wafer* process, individual dies are stacked on top of dies which are still not cut from the wafer. The advantages of these processes are that the wafers on which the different layers of the 3-D stack are produced can be of different sizes. Another advantage is that the individual dies can be tested before the stacking process and only "known-good-dies" can be used, thereby increasing the yield of the 3D-IC. In the *Wafer-to-Wafer* bonding, full wafers are bonded together. The use of through silicon vias (TSVs) to connect the components in the different layers is emerging as a promising technology option. TSV-based 3-D technology has been maturing over the years in addressing thermal issues and achieving high yield [20]. In this paper, we consider 3-D integration using TSVs to electrically connect the different dies in the stack. Heterogeneous systems can be built effectively, with each layer supporting a diverse technology [18].

To tackle the on-chip communication problem, a scalable communication paradigm, networks on chips (NoCs) have recently evolved [1]–[3]. NoCs consist of switches and links and use circuit or packet switching technology to transfer data inside a chip. They provide better structure, modularity, and scalability when compared to traditional interconnect solutions.

NoCs are a necessity for 3-D chips. They provide arbitrary scalability of the interconnects across additional layers, efficiently parallelize communication in each layer, and help controlling the number of vertical wires (and, hence, TSVs) needed for inter-layer communication. The combined use of 3-D integration technologies and NoCs introduces new opportunities and challenges for designers. Building power-efficient NoCs for 3-D systems that satisfy the performance requirements of applications, while satisfying the technology constraints, is an important problem. To address this issue, new architectures and design methods are needed. While the issue of designing NoC architectures for 3-D has received some attention [31]–[34], there has been little work on design methods for 3-D NoCs. Design methods for 2-D NoCs do not consider the unique challenges posed by 3-D integration technologies, such as the constraints on the number of TSVs that can be supported, constraints on communication between adjacent layers, determining layer assignment for switches, and placement of switches in 3-D.

Manuscript received October 12, 2009; revised February 7, 2010; accepted July 12, 2010. Date of current version November 19, 2010. This work was supported in part by the CTI, under Project 10046.2, in part by PFNM-NM, and in part by the ARTIST-DESIGN Network of Excellence. This paper was recommended by Associate Editor N. Chang.

C. Seiculescu is with the Integrated Systems Laboratory, EPFL, Lausanne CH-1015, Switzerland (e-mail: ciprian.seiculescu@epfl.ch).

S. Murali is with the Integrated Systems Laboratory, EPFL, Lausanne CH-1015, Switzerland, and also with the iNoCs, Lausanne CH-1015, Switzerland (e-mail: murali@inocs.com).

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

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

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2061610

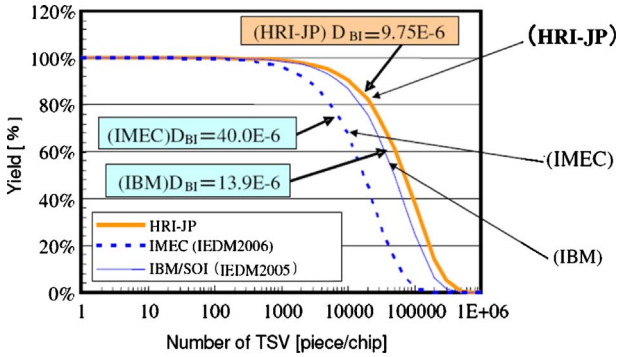

Fig. 1. Yield vs. TSV count [39].

Constraining the number of TSVs is important in order to control the yield of the 3D-IC. In Fig. 1, we show the dependence of yield on the number of TSVs for different manufacturing processes [39]. For all processes, there is a clear upper bound on the number of TSVs after which the yield decreases rapidly. For this reason, it is important for the synthesis process to be aware of this upper bound and be able to ensure that the designed NoC meets this constraint, while at the same time complying with the application requirements. Also, since for the different manufacturing processes the constraint is different, it is necessary for the synthesis process to take this as an input. Another reason to try to reduce the number of TSVs is to save the area. In [19], the pitch of a TSV is reported to be between 3 μm and 5 μm. Reserving area for too many TSVs can cause a considerable reduction in the active silicon area remaining for transistors.

In this paper, we address these important problems and present a synthesis approach for designing power-efficient NoCs for 3-D systems on chips (SoCs) that meet application performance and 3-D technology constraints. Custom topologies that are tailored to meet the application performance constraints can result in large NoC power savings. The need for an application-specific topology has been well studied for 2D-ICs [8], [16]. All these advantages hold in 3-D as well. Our goal in this paper is to explore the design space of custom topologies and to show the comparative advantages in moving to a 3-D technology. The major contribution of this paper is a synthesis approach to determine the most power-efficient topology for the application and for finding paths for the traffic flows that meet the TSV constraints. Our methods account for power and delay of both switches and links. The assignment of cores to different 3-D layers and the floorplan of the cores in each layer are taken as inputs for the synthesis process. To accurately model the link delay and power consumption, for the given core positions, we present a method to determine the optimal positions of switches in the floorplan in each layer. We then place the switches on each layer, removing any overlap with the cores.

The assignment of cores to the different layers and the floorplan of each layer needs to consider several performance and technological constraints. For example, cores that have I/O pins that go off chip have to be placed near the border of the die, cores that operate at the same frequency should be placed close together to share the clock tree, and thermal issues should be considered as well. There are several works that address these issues [21]–[23] and our work is complementary to them. Our experiments show that wires have significant power consumption and delay. Thus, the floorplan of the design should be considered during the topology synthesis process. Here, we only address the issue of designing the NoC topology and determining the placement of the NoC components. We show that using a standard floorplanner to insert the NoC components in an existing floorplan can lead to poor results. We present a simple floorplanning method that shows a significant reduction in area (average 20%) and power consumption (average 7.5%) when compared to a constrained standard floorplanner. Please note that more complex optimization methods, like the ones in [41], could be used to design an even better custom floorplanning routine. As the main objective of this paper is topology synthesis, developing and comparing different custom floorplanning methods is beyond the scope of the paper. Another point to be noted is that a single switch or interface of a NoC has low area (few thousand gates) and power consumption (few megaWatt at 1 GHz) overhead. Thus, the thermal properties of the system are not affected significantly when inserting the NoC components in the floorplan.

Another major contribution of this paper is a comparison between 2-D and 3-D SoCs in terms of the interconnect delay and power consumption. An important advantage in using 3-D technology is that the wires are shorter. However, today, the amount of power and delay gains that is achievable by 3-D integration for custom interconnects for SoCs is still unclear. We compare the same SoC design for the case when all cores are on the same die to the case when the cores are distributed on different dies in a 3-D stack. Therefore, we analyze the power reduction that is due to having shorter wires. We do not consider the case when the initial system was build using different dies in multiple packages and the power reduction was due to removing the I/O pads and drivers. In this paper, for comparative purposes, we also apply a 2-D synthesis flow developed earlier by [16] for a corresponding 2-D implementation of the benchmarks. Our results show that a 3-D design can significantly reduce the interconnect power consumption (38% on average) and latency (13% on average). For completeness, we also show the power consumption reduction in using a custom topology when compared to a regular topology for several benchmarks. However, detailed study of the different advantages of a custom solution is not reported here, as the analysis would be analogous to those done in [8] and [16] for 2D-ICs.

II. Related Work

An introduction to the issues in NoC architecture design and synthesis has been presented in [3]. Methods for synthesizing point-to-point links and bus-based systems are presented in [4]–[6]. In [7]–[9], the authors present approaches to map cores on to regular NoC topologies. Synthesis of custom NoC topologies for 2-D SoCs has been presented in [11]–[16]. In [16], we presented a method to synthesize the most power-performance-efficient NoC topologies for 2-D SoCs.

Several works have been investigating the 3-D manufacturing processes [18], [20], [24], [34]. In [18], the authors make a detailed analysis of performance gains for 3-D technology. The authors analyze how the area, power, and wire length is affected when moving from a 2-D design to 3-D. In [20], a method for placing thermal vias is shown. The method places the thermal vias in order to keep the temperature under a desired maximum value and at the same time tries to minimize the number of thermal vias needed. Models for vertical links containing several parallel wires are presented in [34]. The authors also present methods for using the vertical links within available 2-D design flows. Methods for 3-D floorplanning and placement of cores, taking into account the thermal issues that has been presented in [21]–[23]. In [21], a new data representation is presented that contains the necessary data for performing floorplanning using simulated annealing. The temperature of the chip is used in the objective function that is minimized. A resistive capacitive (RC) model is also presented to calculate the temperature distribution that is used in the objective function. A force-directed 3-D floorplanner is presented in [23]. The floorplanner efficiently takes into account physical information such as temperature gradient, and the authors also present methods to transition from an unconstrained 3-D assignment to a legal layer assignment without overlap. Manufacturing of 3-D interconnects has been addressed by [25] and [26]. The work presented in [28] makes an analysis of the performance and cost tradeoffs of 3-D integration. These works do not address, however, the problem of interconnects in 3-D SoCs. Multi-dimensional regular topologies (such as k-ary n-cubes, hypercubes) have been explored by researchers as viable interconnect solutions for chip-to-chip networks [17]. However, such standard topologies are not suitable for application-specific SoCs, which are heterogeneous in nature.

Synthesis of NoCs for 3-D SoCs is a relatively new topic. New switch architectures for 3-D have been presented in [31] and [33]. In [32], the authors present the use of NoCs as interconnects for 3-D multi-processors. The electrical characteristics of vertical interconnects are analyzed in [34] and the authors also present a back-end design flow to implement 3-D NoCs. Design of standard topologies for 3-D is analyzed in [29] and mapping of cores on to NoC topologies is presented in [30]. Power-delay analysis of 3-D interconnects is presented in [27]. However, none of these works address the issue of synthesizing custom NoCs topologies for 3-D SoCs. Moreover, the works do not present a comparison of the NoC power and latency for 2-D and 3-D NoC implementations.

III. Architecture

We assume a wafer-to-wafer manufacturing process, with TSVs used for the vertical interconnection wires. We present the synthesis methods for the case where each component (core and network component) is designed to span within a single layer. As the network components are small, there will not be a significant performance benefit if they were designed to span on multiple layers. In designs where a single core may span multiple layers, the network interface (NI) would still be assigned to a single layer close to where the communication

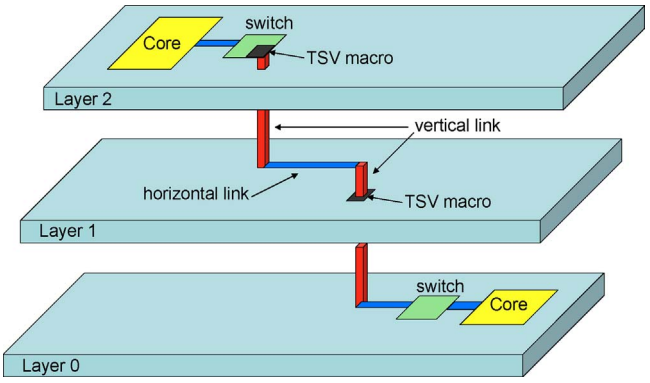

Fig. 2. Example vertical link.

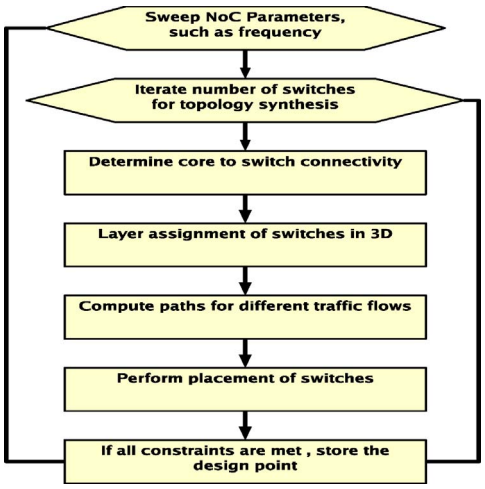

Fig. 3. Algorithm steps.

port of the core is. In this case, our synthesis algorithm is not affected at all and only minor extensions would be needed in the floorplanning routine to account for this fact. As this does not affect the synthesis methods, we do not present such extensions in this paper. In our architecture, the vertical wires using a TSV between two layers uses the global metal routing of the bottom layer, therefore, requiring silicon area to be reserved only on the top layer where the TSVs are drilled. Area reservation is done by placing abstractions in the floorplan called TSV macros. In Fig. 2, we show an example where two switches on two different layers are connected using an inter-layer link.

For the inter-layer links that go through more than one layer, TSV macros are placed in the intermediate layers as well. In Fig. 2, we show an example where there is a TSV macro in the middle layer. From the bottom layer, the link is first routed horizontally on the metal layer and then vertically. In the second layer, an intermediate TSV macro is needed to allow the link to cut through the silicon. Then, the link is routed again on the metal layers in the second layer, and when aligned with the switch on the top layer, the link is fed vertically. The switch in the top layer has a TSV macro embedded for the port that is connected to this link. The area of the TSV macros for a particular link width is taken as input. For the synthesized topologies, our tool automatically places the TSV macros in the intermediate layers and on the corresponding switch ports.

If there is a core that is connected to a switch that is in another layer, then space has to be reserved on the floorplan to place the TSVs. If core and the switch are only one layer apart, then the NI of the core will contain the necessary TSV macro that reserves the space. The NI is responsible to translate the core communication protocol to the network protocol. In the case when the core and the switch are several layers apart, then the TSV macros, to reserve the area to place the TSVs, have to be explicitly placed on the floorplan for the intermediate layers. The explicit TSV macro is the same as in the example of the switch to switch link from Fig. 2 in the middle layer. Active silicon area is lost every time a TSV macro is placed as the area reserved by the macro will be used to construct the TSV. In some designs, redundant TSVs are used to increase reliability [40]. Adding redundant TSVs can be considered by reserving more area with the TSV macros and it is transparent for our tool. The TSV macros are placed automatically by our tool.

IV. Design Approach

In this section, we define the inputs and outputs of our design flow. In the *core specification* file, the name of the different cores, the sizes, and positions are given as inputs. The assignment of the cores to the different layers in 3-D is also specified as input in the file. In the *communication specification* file, the communication characteristics of the application are specified. This includes the bandwidth of communication across different cores, latency constraints and message type (request/response) of the different traffic flows.

The technology used for 3-D integration can result in two main constraints. First, to achieve high yield, the number of TSVs that can be established across two layers may need to be restricted below a threshold. Second, some 3-D technologies can allow TSVs only across adjacent layers. In the rest of the paper, we model the maximum TSV number constraint by using a constraint on the number of NoC links that can cross two adjacent layers, denoted max_ill (for maximum number of inter-layer links). For a particular link width, the maximum number of links can be directly determined from the TSV constraints.

The objective of the topology synthesis procedure can be set by the designer to minimize NoC power consumption or latency or a combination of both.

For the synthesis procedure, the power, area, and timing models of the NoC switches and links are also taken as inputs. For the experimental validation of this paper, without loss of generality, we use the library of network components from [35] and the models are obtained from layout level implementations of the library components. Any other NoC library can also be used with the synthesis process. We also take the power consumption and latency values of the vertical interconnects as inputs. For this, we use the models from [34].

The output of the topology synthesis procedure is a set of tradeoff points of topologies that meet the constraints, with different values of power, latency, and design area. From the resulting points, the designer can choose the optimal point for the application. The synthesis procedure also produces a placement of the switches in the 3-D layers and the positions of the switches.

The different steps of the synthesis algorithm are presented in Fig. 3. As the topology synthesis and mapping problems are NP-Hard [11], we present efficient heuristics to synthesize the best topology for the design. The NoC architectural parameters, such as frequency of operation, are varied and the topology design process is repeated for each architectural point. In the following step, the number of switches needed to connect the cores is varied and different topologies are synthesized. There are some general trends that we observed during the topology design process: 1) when the number of switches is increased, though the switches become smaller, packets need to traverse more hops and the total switch power usually increases, and 2) with more switches, the switch that is connected to a core is closer in the floorplan, thereby leading to lower core-to-switch link power consumption. However, there are also more switch-to-switch links, thereby leading to an increase in the power consumption of switch-to-switch links. In order to choose the most power-efficient topology, the combined effect of all these three trends needs to be considered. Thus, we need to explore designs with different number of switches, starting from one where all the cores are connected to a single switch to a design point where each core is connected to a separate switch.

For a particular switch count, in the next steps, we determine the connectivity between the switches and the cores and the 3-D layer assignment of the switches. During this step, there is a degree of freedom that needs to be explored. A core in a layer of 3-D design that can be restricted to be connected to a switch in the same layer or could be allowed to connect to a switch in any layer. When a core is restricted to be connected to a switch in the same layer, then the traffic flowing from it to a core in another layer needs to traverse at least two switches (one in the current layer and another in the other layer), thereby increasing latency. On the contrary, if a core is allowed to be directly connected to a switch in any layer, then more inter-layer links may be needed. It is important to choose this restriction based on application characteristics. Another degree of freedom that needs to be explored is from a technology standpoint; the technology could allow vertical link across many layers (e.g., a link from first layer to third layer) or could allow connection only across adjacent layers.

To address these two issues, we present a two-phase method to determine the core to switch connectivity. In the first-phase (presented in Section V-A), cores are allowed to be connected directly to switches in any layer. If the resulting designs do not meet the maximum inter-layer link constraints, then in the second phase (presented in Section V-B), cores are restricted to be connected to only switches in the same layer. Also, in the second phase, vertical links are established only across adjacent layers in 3-D. Thus, for systems where the underlying technology supports vertical links only across adjacent layers, the first phase can be skipped and the second phase can be used directly. Please note that Phase 2 can also be used when the objective is to reduce the number of inter-layer vertical links used.

Algorithm 1 Core-to-Switch Connectivity

1: Build partitioning graph, PG(U, H, α)
2: $Unmet = \phi$.
3: {Vary number of direct switches in a range}
4: **for** i = 1 to $|U|$ **do**
5: Perform i min-cut partitions of PG. Let the set $Partition_j$ be set of vertices in jth partition, $\forall j \in 1 \cdots i$.
6: {Compute layer assignment for each switch:}
7: $layer\_sw_j = \frac{\sum_{\forall k \in partition_j} layer\_k}{|partition_j|}$
8: Compute paths for inter-switch flows
9: If path computation failed, add i to set $Unmet$.
10: **end for**
11: $\theta = \theta_{min}$
12: **while** (($Unmet! = \phi$) & ($\theta \leq \theta_{max}$)) **do**
13: **for** Each $i \in Unmet$ **do**
14: Build scaled partitioning graph, SPG(W, L, θ)
15: PG = SPG
16: Repeat Steps 5 to 8
17: If valid paths found, remove i from set $Unmet$.
18: **end for**
19: $\theta = \theta + \theta_{scale}$
20: **end while**

In the next step, the paths for the inter-switch traffic flows is determined, and is explained in Section VI. Then, the optimal positions of the switches are determined and the switches are placed in each layer, minimally changing the input floorplan. In the last step, if the current topology meets the constraints, the design point is saved.

V. Methods to Establish Core to Switch Connectivity

In this section, we present methods for establishing connectivity between the cores and switches.

Definition 1: Let n be the number of cores in the design. The x and y coordinate positions of a core c_i are represented by xc_i and yc_i, respectively, $\forall i \in 1 \cdots n$. The 3-D layer to which the core i is assigned is represented by $layer_i$.

From the *communication specification* file, the communication characteristics of the application are obtained and represented by a graph [7], [8], [10], defined as follows.

Definition 2: The communication graph is a directed graph, $G(V, E)$ with each vertex $v_i \in V$ representing a core and the directed edge (v_i, v_j) representing the communication between the cores v_i and v_j. The bandwidth of traffic flow from vertex v_i to v_j is represented by $bw_{i,j}$ and the latency constraint for the flow is represented by $lat_{i,j}$.

We define a *partitioning graph (PG)* as follows.

Definition 3: The PG is a directed graph, $PG(U, H, \alpha)$, that has same set of vertices and edges as the communication graph. The weight of the edge (u_i, u_j), defined by $h_{i,j}$, is set to a combination of the bandwidth and the latency constraints of the traffic flow from core u_i to u_j: $h_{i,j} = \alpha \times bw_{i,j}/max\_bw + (1 - \alpha) \times min\_lat/lat_{i,j}$, where max_bw is the maximum bandwidth value over all flows, min_lat is the tightest latency constraint over all flows and α is a weight parameter.

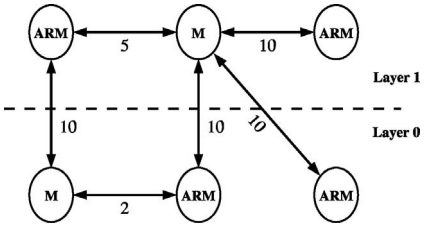

Fig. 4. Communication graph with bandwidth demands on the edges.

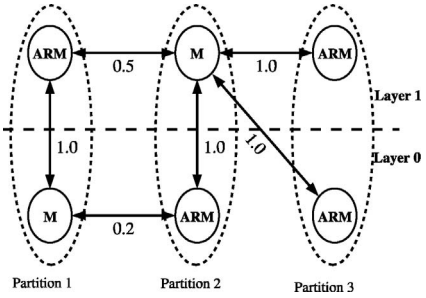

Fig. 5. PG and the min-cut partitions.

The parameter α can be set by the designer based on the application characteristics or swept by the tool over a range of values, in order to meet the latency constraints.

A. Phase 1

As the number of switches is varied in order to explore different design points, most of the times there will be fewer switches in the design than the number of cores. Therefore, multiple cores will be assigned to the same switch in most cases. The cores will be partitioned in as many blocks as there are switches and the cores in the same block will be assigned to the same switch. When using Phase 1 cores that have high communication or tight latency will be assigned to the same switch regardless of the fact that cores may be on different layers. If for a particular core to switch assignment the algorithm using Phase 1 will not be able to meet the inter-layer link constraint, then the influence of bandwidth and latency of the inter-layer flows will be scaled down in steps and new partitions will be generated. The exact steps of Phase 1 are described in the following paragraphs.

In the first step of Algorithm 1, the PG is constructed. Then (in Step 3), the number of switches in the design is varied from 1 to the number of cores in the design. In the next step (Step 5), for the current switch count, many min-cut blocks of PG are obtained. All the cores in a block are connected to the same switch and the partitioning is done such that each block has about equal number of cores. Thus, those traffic flows with large bandwidth requirements or tight latency constraints are assigned to the same block and traverse a single hop in the network.

Example 1: For the communication graph from Fig. 4, an example PG is shown in Fig. 5. The cores are assigned to the two layers such that highly communicating cores are placed one above the other, which is an input to our synthesis algorithm. Here, we assume α = 1 and the bandwidth of the

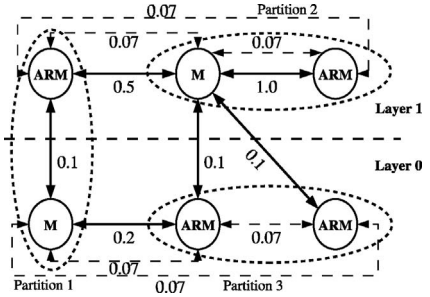

Fig. 6. SPG and the min-cut partitions.

traffic flowing between cores within a layer is lower than the traffic between the cores across the layers as denoted by the values on the edges of the communication graphs. The weights of the PG are calculated with the formula from definition of the PG graph. In the figure, we also show an example of three min-cut partitions of the graph. The partitioning leads to cores in different layers being assigned to the same block.

Then (in Step 7), the layer assignment of each switch is computed as an average of the layers of the cores to which the switch is connected. Alternatively, the switch could also be assigned to the layer containing the most number of cores connected to it. At this point, the intra-partition traffic flows are taken care of and we need to establish connectivity across the switches for the inter-switch traffic flows. This step is explained in the next section. Then (in Step 9), the resulting designs are evaluated to see whether they meet the *max_ill* constraint and the switch counts that do not meet the constraint are stored in the set *unmet*.

In order to facilitate meeting the *max_ill* constraint for the design points in the set *unmet*, we use the *SPG*, defined as follows:

Definition 4: A SPG with a scaling parameter θ, SPG(W, L, θ), is a directed graph that has the same set of vertices as PG. A directed edge $l_{i,j}$ exists between vertices i and j, if $\exists(u_i, u_j) \in P$ or $layer_i = layer_j$.

In the SPG, along with the edges in PG, we define new edges between all cores in the same layer of 3-D. We also reduce the edge weights of inter-layer flows, depending on the scaling parameter θ. If this scaled graph is used for partitioning, then more cores in the same layer will be in a partition, thereby reducing the inter-layer links, at the expense of increasing the power consumption and latency of inter-layer flows. To obtain designs with lower inter-layer links, the parameter θ is varied from θ_{min} to θ_{max} in steps of θ_{scale} in the algorithm (Steps 12–19), until the max_ *ill* constraint is met. After several experimental runs, we determined that varying θ from 1 to 15 in steps of three gives good results.

In order to cluster cores in a layer that actually communicates, we also need to ensure that the newly added edges have a lower edge weight than the original intra-layer edges. Please note that if the new edges are not added, the partitioner may still cluster cores across layers, which will not lead to a reduction in the inter-layer links.

We denote the maximum edge weight in PG by max_wt. We formally define the edge weights in SPG as follows:

$$l_{i,j} = \begin{cases} h_{i,j} & \text{if } (u_i, u_j) \in PG \& layer_i = layer_j \\ \frac{h_{i,j}}{\theta \times |layer_i - layer_j|} & \text{if} (u_i, u_j) \in PG \& layer_i \neq layer_j \\ \frac{\theta \times max\_wt}{10 \times \theta_{max}} & \text{if} (u_i, u_j) \notin PG \ \& \ layer_i = layer_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

from the definition, we can see that the newly added edges have at most one-tenth the maximum edge weight of any edge in PG, which was obtained experimentally after trying several values.

Example 2: The SPG for $\theta = 10$ for the PG from Example 1 is presented in Fig. 6. In the SPG, the inter-layer links have lower weights as they were scaled down by θ and new edges are added between cores within the same layer. The weights of the extra edges are calculated using (1). The three min-cut blocks are now different, with more cores in the same layer belonging to the same block.

The time complexity of the Algorithm 1 is $O(|V|^3|E| ln(|V|))$, where $|V|^2 ln(|V|)$ corresponds to finding paths, E is the maximum number of flows. Also, the number of switches is varied from 1 to the maximum number of cores $|V|$ and that a topology is built for each switch count.

B. Phase 2

In Phase 2, we restrict cores to be connected to switches on the same layer. Also switches can only connect to switches on the same layer or on adjacent layers. For each layer, a certain number of switches will be assigned. Then, on each layer, the cores are partitioned and assigned to the switches on that layer considering the bandwidth and latency of the flows between the cores on that layer. Information of the inter-layer flows is ignored when the cores are assigned to switches when using Phase 2. Because of these restrictions, Phase 2 can be used when a tight inter-layer link restriction is in place or when the 3-D integration technologies forbids links to go across more than two layers. In this section, a detailed description of Phase 2 is given.

We define the *LPG* for each layer as follows.

Definition 5: A LPG(Z, M, ly) is a directed graph with the set of vertices represented by Z and edges represented by M. Each vertex represents a core in the layer ly. An edge connecting two vertices is similar to the edge connecting the corresponding cores in the *communication graph*. The weight of the edge (m_i, m_j), defined by $h_{i,j}$, is set to a combination of the bandwidth and the latency constraints of the traffic flow from core m_i to m_j: $h_{i,j} = \alpha \times bw_{i,j}/max\_bw + (1 - \alpha) \times min\_lat/lat_{i,j}$, where max_bw is the maximum bandwidth value over all flows, min_lat is the tightest latency constraint over all flows, and α is a weight parameter. For cores that do not communicate with any other core in the same layer, edges with low weight (close to 0) are added between the corresponding vertices to all other vertices in the layer. This will allow the partitioning process to still consider such isolated vertices.

Example 3: The LPGs for the two layers of the *communication graph* from Fig. 4 are shown in Fig. 7. Since the

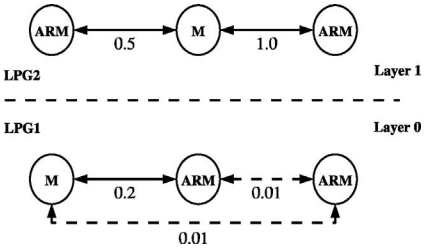

Fig. 7. LPGs for two layers.

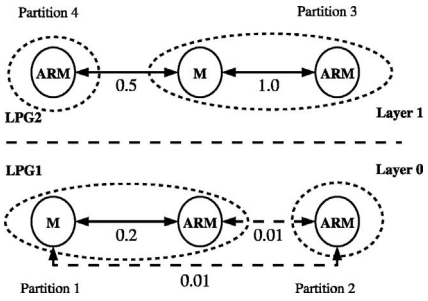

Fig. 8. Two min-cut partitions of LPGs.

LPGs are built layer by layer, the graphs for the two layers are independent of one another. The weights of the remaining edges were calculated with the formula from the definition of the LPG graph for a value of $\alpha = 1$. Extra edges with low weights are added (dotted edges in the figure) from the vertices that have no connections to the other vertices of the LPG.

The algorithm for establishing core to switch connectivity is presented in Algorithm 2. As the number of I/O ports of a switch increases, the maximum frequency of operation that can be supported by it reduces, as the combinational path inside the crossbar and arbiter increases with size. In the first step of the algorithm, for the required operating frequency of the NoC, the maximum size of the switch (denoted by max_sw_size) that can support that frequency is obtained as an input. Based on this and the number of cores in each layer, in the next steps (2–4), we determine the minimum number of switches needed in each layer. Then the LPG for each layer is constructed.

Then, the number of switches in each layer is incremented (starting from the initial count calculated in Steps 2–4) every iteration, until it equals the number of cores in the layer. The term $|LPG(Z, M, j)|$ represents the number of cores in layer j. For each switch count, many min-cut partitions of the LPG of the layer are obtained (Step 13). The cores in the same partition are connected to the same switch.

Example 4: Two min-cut blocks of the LPGs of Fig. 7 are shown in Fig. 8.

The time complexity of the Algorithm 2 is similar to the complexity of the previous algorithm $O(|Z_{max}||V|^2|E|ln(|V|))$, where Z_{max} corresponds to the maximum number of cores in a layer, which decides the number of topologies to be explored.

C. Pruning the Search Space

To reduce the number of explored design points, we use several methods to prune the search space.

Algorithm 2 Core-to-Switch Connectivity

1: Obtain maximum switch size max_sw_size for current frequency
2: **for** each layer $j \in 1 \cdots lr$ **do**
3: $ni_j = \lceil$ number of cores in $layer_j / max\_sw\_size \rceil$
4: **end for**
5: Build $LPG(Z, M, j)$ for each layer j.
6: **for** $i = 0$ to $\max_{\forall j \in 1 \cdots lr}\{|LPG(Z, M, j)| - ni_j\}$ **do**
7: **for** each layer $j \in 1 \cdots lr$ **do**
8: **if** $ni_j + i \leq |LPG(Z, M, j)|$ **then**
9: $np = ni_j + i$
10: **else**
11: $np = |LPG(Z, M, j)|$
12: **end if**
13: Obtain NP min-cut partitions of LPG(Y, M, j)
14: **end for**
15: Compute paths for inter-switch flows (Section VI).
16: If valid paths found, save the current design point
17: **end for**

1) As the number of I/O ports of a switch increases, the maximum frequency of operation that can be supported by it reduces, as the combinational path inside the crossbar and arbiter increases with size. For a required operating frequency of the NoC, we first determine the maximum size of the switch (denoted by max_sw_size) that can support that frequency and determine the minimum number of switches needed.
2) Considering all the possible combinations of the number of switches in each layer would increase the search space too much. Therefore, at each iteration of Algorithm 2, we increment the number of switches in each layer by one. We initialize the number of switches layer by layer as explained in the previous section. Thus, the starting design point can have different number of switches in each layer and the number of switches in each layer is proportional to the number of cores in that layer.
3) For a particular switch count, after partitioning, we evaluate the inter-layer links used to connect the cores to the switches, before finding the paths. If the topology requires more inter-layer links than the threshold, we directly ignore the design point.

VI. Path Computation

The procedure to establish physical links and paths for traffic flows is based on the power consumption increase and latency in using the link. This cost computation in the 3-D case is similar to the 2-D case, such as those presented in [14] and [16], but it needs to account for the max_ill and max_switch_size constraints. Here, we do not show the entire path computation algorithm, but only present the steps needed to meet these constraints. In [14] and [16], the authors present methods to remove both routing and message-dependent deadlocks when computing the paths. We also use the methods to obtain paths that are free of deadlocks.

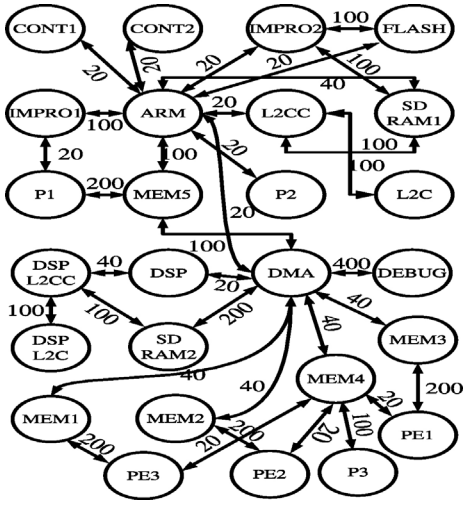

Fig. 9. *D26_media* communication graph.

Definition 6: Let nsw be the total number of switches used across all the layers and $layer_i$ be the layer in which switch i is present. Let $ill(i, j)$ be the number of vertical links established between layers i and j. Let the $switch\_size\_inp_i$ and $switch\_size\_out_i$ be the number of input and output ports of switch i. Let $cost_{i,j}$ be the cost of establishing a physical link between switches i and j.

In Algorithm 3, we show the use of hard and soft thresholds when evaluating the cost of establishing a physical link between switches i and j. In Steps 3 and 4, we assign a cost of INF (infinity) for establishing a link across switches in non-adjacent layers and for switches in layers that have reached the maximum vertical link (max_ill) threshold. To ensure meeting the maximum link constraint, we assign a very high cost (denoted by $SOFT\_INF$) for establishing links between switches that are in layers having vertical links close to the max_ill value, denoted by $soft\_max\_ill$ (Steps 5 and 6). From experiments, we found that a reasonable value for $SOFT\_INF$ to be ten times the maximum cost of any flow and $soft\_max\_ill$ to be few (2–3) links less than max_ill value. We use a similar technique to meet the maximum switch size constraints (Steps 10–12). By using these softer constraints, first, we facilitate the path computation procedure to determine valid paths when compared to only using the hard constraints.

When paths are computed, if it is not feasible to meet the max_switch_size constraints, we introduce new switches in the topology that are used to connect the other switches together. These indirect switches help in reducing the number of ports needed in the direct switches.

Algorithm 3 CHECK_CONSTRAINTS(i, j)

1: **for** $i = 1$ to nsw **do**
2: **for** $j = 1$ to nsw **do**
3: **if** $|layer_i - layer_j| \geq 2$ or $ill(layer_i, layer_j) \geq max\_ill$ **then**
4: $cost_{i,j} = INF$
5: **else if** $|layer_i - layer_j| = 1$ and $ill(layer_i, layer_j) \geq soft\_max\_ill$ **then**
6: $cost_{ij} = SOFT\_INF$
7: **else if** $switch\_size\_inp_i + 1 \geq max\_switch\_size$ or $switch\_size\_out_j + 1 \geq max\_switch\_size$ **then**
8: $cost_{i,j} = INF$
9: **else if** $switch\_size\_inp_i + 1 \geq soft\_max\_switch\_size$ or $switch\_size\_out_j + 1 \geq soft\_max\_switch\_size$ **then**
10: $cost_{i,j} = SOFT\_INF$
11: **end if**
12: **end for**
13: **end for**

VII. Switch Position Computation

Once a topology for a particular switch count is obtained, the next step is to find the latency and power consumption on the wires. In order to do this, based on the input positions of the cores, the optimal position of the switches needs to be determined. For this, we model the problem as a linear program (LP) [36].

Let us consider a topology with nsw switches. We denote the co-ordinates of a switch i by (xs_i, ys_i), $\forall i \in 1 \cdots nsw$. The goal of the LP is to determine the values of xs_i and ys_i, for all switches in the particular topology. The sum of the Manhattan distances between a switch i and a core k is given as follows:

$$coredist_{i,k} = \begin{cases} |xs_i - xc_k| + |ys_i - yc_k| & \text{if } switch_i \text{ connected to } core_k \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The sum of the Manhattan distances between a switch i and switch j to which it is connected to is given as follows:

$$swdist_{i,j} = \begin{cases} |xs_i - xs_j| + |ys_i - ys_j| & \text{if } switch_i \text{ connected to } switch_j \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

The above equations can be easily represented as a set of linear equations [36]. Let $bw\_sw2core_{i,k}$ and $bw\_sw2sw_{i,j}$ be the total bandwidth of traffic flows between switch i and core k and switches i and j, respectively. To minimize the total power consumption of the links, we need to minimize the length of the links weighted by their bandwidth values, so that higher bandwidth links are shorter than lower bandwidth ones. Formulating the objective function mathematically, we get the following:

$$\begin{aligned} obj \;=\; & \textstyle\sum_{\forall i} \sum_{\forall k} coredist_{i,k} * bw\_sw2core_{i,k} \\ & + \textstyle\sum_{\forall i} \sum_{\forall j} swdist_{i,j} * bw\_sw2sw_{i,j}. \end{aligned} \quad (4)$$

The LP for optimization is written as follows:

$$\begin{aligned} &\text{minimize} && obj \\ &\text{subject to} && Equations\ 2-4 \\ & && xs_i, ys_i \;\geq\; 0\ \forall i \in 1 \cdots nsw. \end{aligned} \quad (5)$$

We use the *lp_solve* package [37] to obtain the optimum solution for the switch co-ordinates. Even for big applications (65 cores, tens of switches), the optimal solution is obtained in few seconds. We also pipeline long links to support full throughput on the NoC and add NIs to connect the cores to the network. The resulting design is a valid floorplan of the

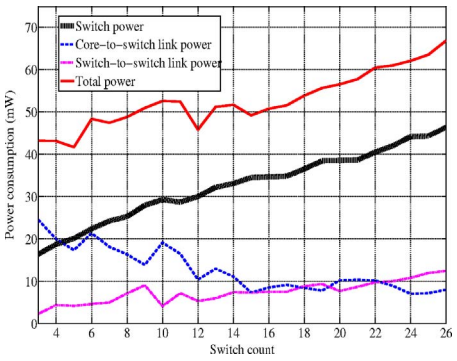

Fig. 10. Power consumption in 2-D.

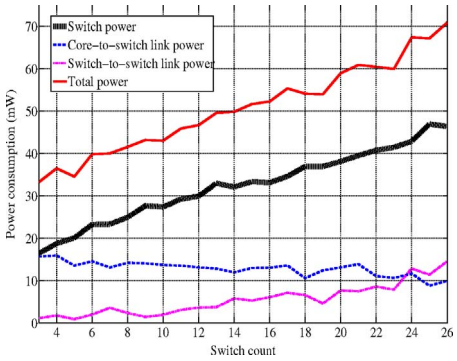

Fig. 11. Power consumption in 3-D.

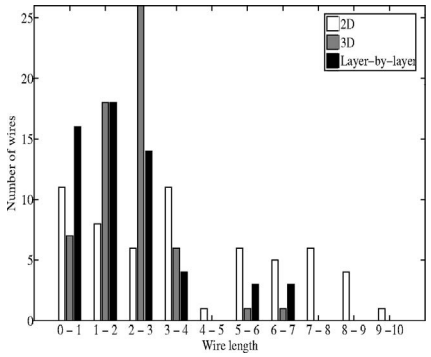

Fig. 12. Wire length distributions.

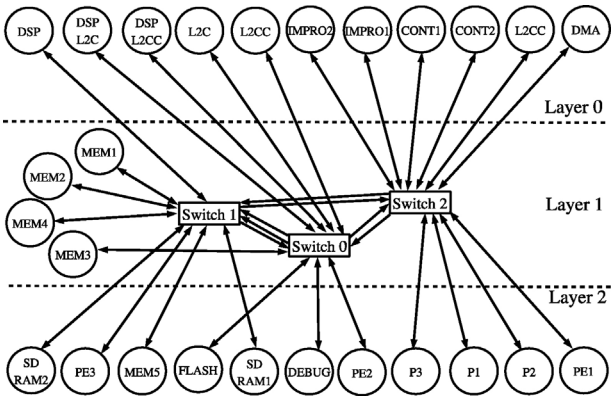

Fig. 13. Most power-efficient topology (*Phase 1*).

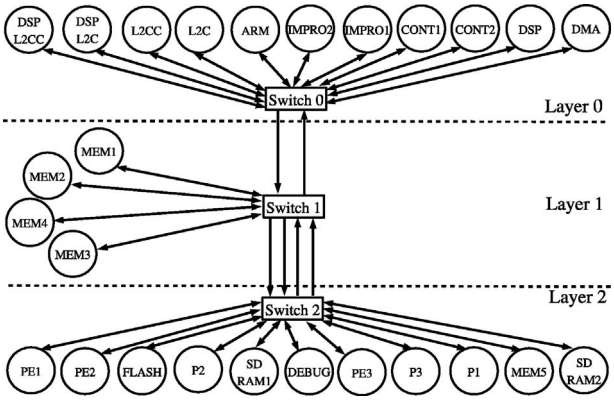

Fig. 14. Most power-efficient topology layer-by-layer (*Phase2*).

NoC. The TSV macros do not need to be included in the LP as TSVs split the wires in two segments, both carrying the same bandwidth. Therefore, the placement of the TSV macro is more relaxed.

However, placing the components at the ideal positions may lead to overlap with the already placed cores. To remove such overlaps, we consider one switch or TSV macro at a time. We try to find a free space near its ideal location to place it. In the case of the switches, the area in which we look for free space is the same for all of the switches, as it is given as a constant to the floorplanning routine. In case of the TSV, the area in which we look depends on the blocks the TSV is connected to as explained before. If no space is available, we displace the already placed blocks from their positions in the x or y direction by the size of the component, creating space. Moving a block to create space for the new component can cause overlap with other already placed blocks. We iteratively move the necessary blocks in the same direction as the first block, until we remove all overlaps. As more components are placed, they can re-use the gap created by the earlier components. As some switches can cause thermal hotspots, a more advanced thermal aware floorplanning routine can try to find position of switches near cooler components, so that the thermal distribution is more uniform. Several existing works, which are complementary to ours, present methods that address this issue [21]–[23]. However, integrating such works with ours is beyond the scope of this paper. In the experimental section, we compare how this custom routine for floorplanning compares to a standard floorplanner that is constrained to only insert the network components without changing the relative placement of the initial cores. The standard floorplanner that we used to compare against is Parquet [38]. The floorplanner was used on each layer separately. We feed the core and switch positions as an input solution to the floorplanner. We allow it to move the switches around the cores, maintaining the relative positions of the cores and minimizing the movement of the switches from the optimal positions computed by the LP.

VIII. Experiments and Case Studies

For the experiments, the NoC component library from [35] is used. The power and latency values of the switches and links

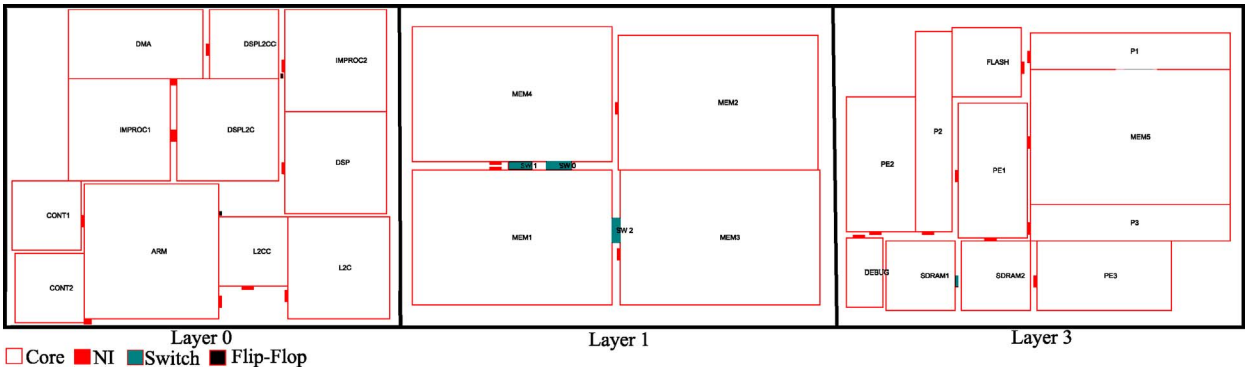

Fig. 15. Resulting 3-D floorplan with switches for the topology from Fig. 13.

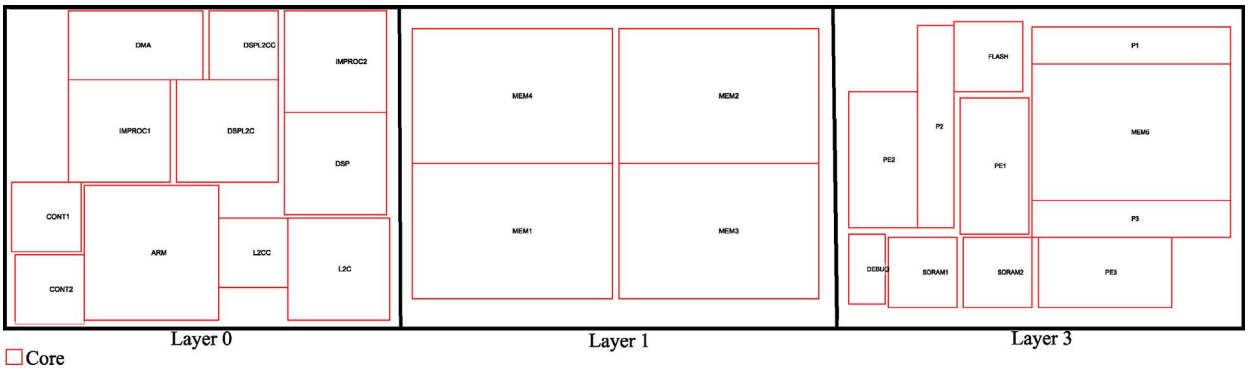

Fig. 16. Initial positions for *D26_media*.

of the library are determined from post-layout simulations, based on 65 nm low power libraries. The vertical interconnects using TSVs are implemented based on the models from [34]. In [34], the reported delay values for TSV placed in a tightly packed TSV bundle are 16 ps and 18.5 ps (for silicon on insulator and bulk silicon, respectively). The considered TSVs have a diameter of 4 μm and a pitch of 8 μm. When compared against the maximum unrepeated planar link length of 1.5 mm in Metal 2 or Metal 3 for the same technology, the authors show that the vertical links have much lower resistance and capacitance (an order of magnitude reduction for the resistance as well as for the capacitance). As a consequence, even tightly packed TSVs are substantially faster and more power efficient than moderate planar links.

A. Multimedia SoC Case Study

We consider a benchmark of a realistic multimedia and wireless communication SoC for case study (referred to as *D_26_media*). The benchmark contains 26 cores with irregular sizes, and performs based-band and multimedia processing. The communication graph of the benchmark is shown in Fig. 9. The system includes ARM, DSP cores, multiple memory banks, DMA engine and several peripheral devices. The cores are manually mapped on to three layers in 3-D. For comparisons, we also consider a 2-D implementation of the benchmark. The initial positions of the cores in each layer of the 3-D and for the 2-D design are obtained using existing tools [38]. For fair comparisons, we use the same objectives of minimizing area and wire-length when obtaining the floorplan for both the cases. To synthesize the topologies for the 2-D case, we use the synthesis flow developed earlier [16].

In Figs. 10 and 11, we present the power consumption of the NoC topologies (power consumption on switches and links) synthesized by our tools for different switch counts for both cases. In all the experiments, we set the data width of the NoC links to 32 bits, to match the core data widths. The frequency for which the topologies are generated has to be given as an input. A range of frequencies can also be swept by the tool to explore more design points. However, for this benchmark, the best power points are obtained for topologies designed at the lowest possible operating frequency, which was found by the tool to be 400 MHz. Higher operating frequency can be used (usually with a higher cost in power consumption). We use a *max_ill* constraint of 25 links for this and the experiments in the next section. In Section VIII-E, we study the impact of varying this constraint.

When very few switches are used in the design, they need to have more I/O ports, as they need to connect to more cores. A large switch can only support a low operating frequency, as the critical path inside the switch increases with its size. In order to meet the 400 MHz requirement, we could only obtain valid topologies with three or more switches, thus the plots starts at three switches. In the plots, we show the switch, switch-to-switch link, and core-to-switch link power consumption values as well. For this benchmark, we can observe a power saving of 24% for the 3-D case with respect to the 2-D case. This is due to the fact that the long horizontal wires in a 2-D design are replaced by shorter vertical wires. In Fig. 12, we

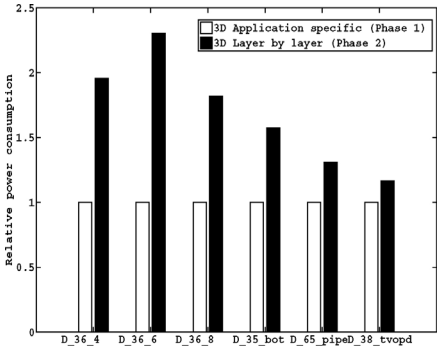

Fig. 17. Comparison with layer-by-layer.

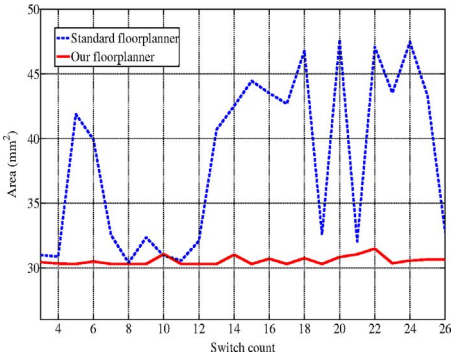

Fig. 18. Area plot for different switch counts.

TABLE I
2-D VS. 3-D NOC COMPARISON

Benchmark	Power (mW)						Latency (cyc)	
	Link Power		Switch Power		Total Power			
	2-D	3-D	2-D	3-D	2-D	3-D	2-D	3-D
D_36_4	150	41.5	65	70.5	215	112	3.28	3.14
D_36_6	154.5	43.5	76.5	82	230	125.5	3.57	3.5
D_36_8	215	55.5	105	104.5	320	160	4.37	3.65
D_35_bot	68	36.2	48	43.3	116	79.5	6.04	4.2
D_65_pipe	106	104	63	58	169	162	2.53	2.57
D_38_tvopd	52.5	22.67	37	38.11	89.5	60.78	4	3.6

show the wire-length distribution of the links in 2-D and 3-D cases. From the figure, as expected, the 2-D design has many long wires. In Figs. 13 and 15, we present the most power-efficient topology synthesized by our tool using Phase 1 of the algorithm and the floorplan of the cores and network components for the 3-D case. The original placement of the cores for this benchmark is shown in Fig. 16.

In order to show how Phase 2 of the algorithm performs, we constrained the tool to use the layer-by-layer approach and we ran it on the same benchmark. The topology for the best power point is presented in Fig. 14. Even though we used the same *max_ill* constraint of 25 links as in the previous case, it can be seen from the figure that the algorithm used a lot less inter-layer links. This is also an intuitive example of why Phase 2 of the algorithm is able to produce valid topologies even for tight *max_ill* constraints where Phase 1 fails. There is also a price to pay for using fewer interlayer-links. In the case of the Phase 2 topology, cores on different layers will have a zero load latency of at least two cycles as they have to go through two switches. For Phase 1, cores on different layers are connected to the same switch. So, even if two cores are on different layer they could still have a zero load latency of just one cycle.

B. Comparison Between Phase 1 and Phase 2

We applied our synthesis procedure on varied set of benchmarks to validate the gains under different application scenarios. We consider three distributed benchmarks with 36 cores (18 processors and 18 memories), *D_36_4*, *D_36_6* and *D_36_8*, where each processor has four, six, and eight traffic flows going to the memories. The total bandwidth is the same in the three benchmarks. We consider a benchmark, *D_35_bot*, that models bottleneck communication, with 16 processors, 16 private memories (one processor is connected to one private memory), and three shared memories to which all the processors communicate. We also consider two benchmarks where all the cores communicate in a pipeline fashion, 65 core (*D_65_pipe*) and 38 core designs (*D_38_tvopd*). In the last two benchmarks, each core communicates only to one or few other cores.

In Fig. 17, we show the power consumption of the topologies synthesized using Phase 2 of the algorithm, with respect to topologies synthesized using Phase 1 for the different benchmarks. Since in Phase 2 cores in a layer are connected to switches in the same layer, the inter-layer traffic needs to traverse more switches to reach the destination. This leads to an increase in power consumption and latency. As seen from Fig. 17, Phase 1 can generate topologies that lead to a 40% reduction in NoC power consumption, when compared to Phase 2. However, Phase 2 can generate topologies with a much tighter inter-layer link constraint.

C. 2-D vs. 3-D Comparison

The power consumption for the least power design points for 2-D and 3-D, as well as the average latency, are presented in Table I. Most of the power savings obtained in 3-D are due to shorter wires. For this reason, we can observe large power savings for the distributed benchmarks, where there are traffic flows to many different cores. We can also notice reasonable power savings for the bottleneck design, because the wires going to shared memories are long, though the traffic to the shared memories is smaller than to the private memories. For the pipelined benchmarks, lower savings are obtained. For the different benchmarks, on average, a 38% power reduction and 13% latency reduction are obtained in the 3-D case when compared to a 2-D implementation.

D. Floorplanning Study

To create a floorplan for a real SoC is quite a complicated process that can require several interactions between the designed and the floorplanning tool. Since the main goal of our tool is to design the NoC and not to create floorplans, we take the initial positions of the cores as input. We then only insert the NoC components as close as possible to their ideal positions in the floorplan in such a way that we minimally affect the initial positions of the cores.

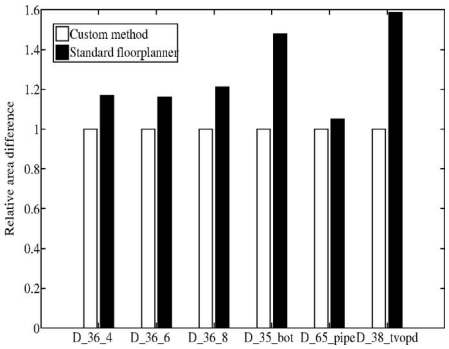

Fig. 19. Area comparison for different benchmarks.

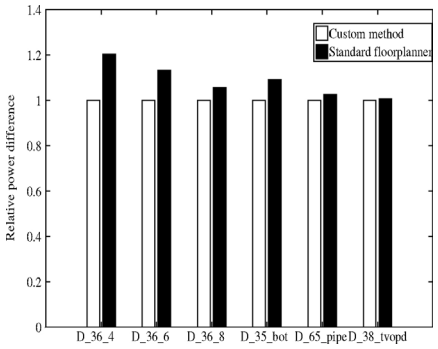

Fig. 20. Power comparison for different benchmarks.

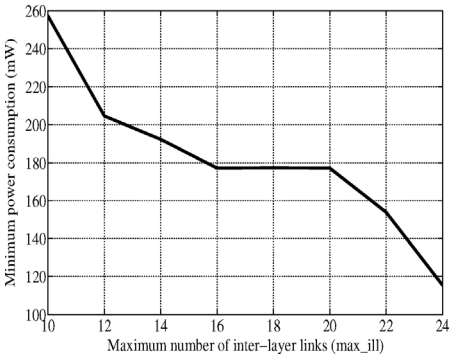

Fig. 21. Impact of max_ill on power.

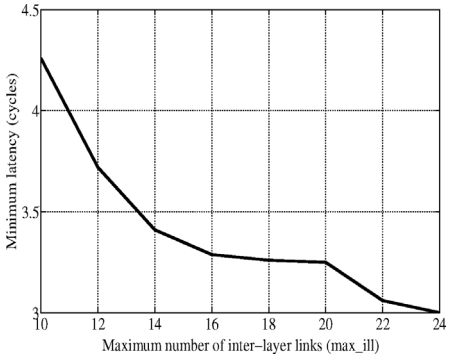

Fig. 22. Impact of max_ill on latency.

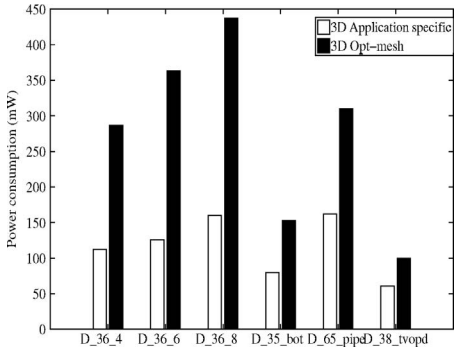

Fig. 23. Comparisons with Mesh.

Most of the time placing the NoC components at their ideal positions will result in overlap with the initially placed cores, especially if the floorplan is tightly packed. Initially, we tried to use a standard floorplanner to remove the overlap. We used the floorplanner from [38], which we have modified in order to constrain it from swapping blocks, so that the relative positions of the input cores remain the same after the NoC insertion. This, however, leads to fairly poor results as the floorplanner has problems to remove the overlap, keep the cores close to their initial placement, and not swap any of them. The floorplanner needs the ability to swap blocks in order to create good floorplans; for this reason, this full floorplanner is ideal for generating the initial placement, but not for the NoC insertion. To achieve better results in inserting the NoC components and removing the overlap, we designed a custom floorplanning routine which we tuned for this specific task alone.

In Fig. 18, we show a comparison between the standard floorplanner and our custom routine for different switch counts using th D_26_media benchmark. From the figure, it can be seen that for some points even the standard floorplanner performs well and that there is a better chance to get a good floorplan when fewer switches are inserted. However, the behavior of the constrained standard floorplanner is unpredictable. A comparison between the best power points for the different benchmarks using the two floorplanning methods is shown in Fig. 19. Since the area has a direct impact on the wire lengths and, consequently, on the power, we also present a comparison of the power consumption for the considered topologies in Fig. 20.

E. Impact of Inter-Layer Link Constraint and Comparisons with Mesh

Imposing a stricter constraint on max_ill results in topologies having more switches. When there are more switches, more cores in a layer are connected to a switch in the same layer, reducing the number of inter-layer links. However, the inter-layer traffic flows would need to traverse more switches, thereby leading to higher power consumption and latency. We perform topology synthesis for the D_36_4 design with different max_ill constraint values, and the power, latency values for the different points are presented in Figs. 21 and 22. With a tighter TSV constraint, the power consumption and latency increases significantly, as more switches are needed in

the design. With less then ten inter-layer links, it is impossible to build any topology and having a *max_ill* constraint larger than 24 does not improve the results anymore. For the *D_26_media* benchmark, we observe a similar trend for the power consumption. However, the zero load latency is not affected by tighter *max_ill* constraints, due to the nature of the traffic of this benchmark.

For completeness, we compare power consumption of the topologies generated by our procedure to a standard topology. We generate best mapping (optimizing for power, meeting the latency constraints) of the cores on to a mesh topology, and remove any unused switch-to-switch links. Compared to this optimized mesh topology, we obtain a large power reduction for the custom topologies (an average of 51%), shown in Fig. 23. Our experiments also showed that we obtain 21% reduction in latency when compared to the optimized mesh.

Even though the algorithm explores a large space of solutions, due to the use of efficient heuristics presented, all the experiments could be performed in few hours (on a system operating at 2 GHz). It takes a few seconds to build a topology with few switches and the run time can go up 2 or 3 minutes for topologies with many switches (50, 60 switches). The total runtime on a benchmark depends on the frequency range and switch count range that are swept. Also, it is important to note that the synthesis algorithm has to be performed only once at design time for a system and the timing overhead is negligible.

IX. Conclusion

NoCs are necessary to achieve a scalable communication infrastructure in 3-D chips. The use of NoCs in 3D-ICs introduces several new and challenging problems. Building a custom NoC topology that meets the application communication requirements, as well as the 3-D technological constraints, is a critical problem that needs to be addressed. In this paper, we presented *SunFloor 3D*, a tool for NoC topology synthesis for 3D-ICs. The tool also performs path computation, assignment, and placement of network components in the 3-D layers. Our experiments on several realistic benchmarks showed that the tool produces topologies that result in large NoC power and latency savings (54% and 21%, respectively) when compared to standard topologies. We also presented a comparative analysis of NoCs in 2-D and 3-D, which showed that 3-D integration can produce large interconnect power and latency reduction (38% and 13%, respectively).

References

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

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

[3] G. De Micheli and L. Benini, *Networks on Chips: Technology and Tools*, 1st ed. San Mateo, CA: Morgan Kaufmann, Jul. 2006.

[4] J. Hu, Y. Deng, and R. Marculescu, "System-level point-to-point communication synthesis using floorplanning information," in *Proc. ASP-DAC*, 2002, p. 573.

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

[6] S. Pasricha, N. Dutt, E. Bozorgzadeh, and M. Ben-Romdhane, "Floorplan-aware automated synthesis of bus-based communication architectures," in *Proc. DAC*, 2005, pp. 565–570.

[7] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Proc. DATE*, Mar. 2003, pp. 688–693.

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

[9] S. Murali and G. De Micheli, "Bandwidth constrained mapping of cores on to NoC architectures," in *Proc. DATE*, Feb. 2004, pp. 896–901.

[10] S. Murali, L. Benini, and G. De Micheli, "Mapping and physical planning of networks on chip architectures with quality-of-service guarantees," in *Proc. ASPDAC*, Jan. 2005, pp. 27–32.

[11] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli, "Efficient synthesis of networks on chip," in *Proc. ICCD*, Oct. 2003, pp. 146–150.

[12] T. Ahonen, D. A. Sigüenza-Tortosa, H. Bin, and J. Nurmi, "Topology optimization for application specific networks on chip," in *Proc. SLIP*, 2004, pp. 53–60.

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

[14] A. Hansson, K. Goossens, and A. Radulescu, "A unified approach to mapping and routing on a combined guaranteed service and best-effort network-on-chip architectures," Philips Research, Eindhoven, The Netherlands, Tech. Rep. 2005/00340, Apr. 2005.

[15] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, "A design methodology for application-specific networks-on-chip," *ACM TECS*, vol. 5, no. 2, pp. 263–280, May 2006.

[16] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. De Micheli, and L. Raffo, "Designing application-specific networks on chips with floorplan information," in *Proc. ICCAD*, 2006, pp. 355–362.

[17] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 775–785, Jun. 1990.

[18] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: A novel chip design for deep-submicrometer interconnect performance and systems-on-chip integration," *Proc. IEEE*, vol. 89, no. 5, pp. 602–633, May 2001.

[19] E. Beyne, "The rise of the 3rd dimension for system integration," in *Proc. Interconnect Technol. Conf.*, Jun. 2006, pp. 1–5.

[20] B. Goplen and S. Sapatnekar, "Thermal via placement in 3-D ICs," in *Proc. Int. Symp. Physical Design*, 2005, pp. 167–174.

[21] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3-D ICs," in *Proc. ICCAD*, Nov. 2004, pp. 306–313.

[22] W.-L. Hung, G. M. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, "Interconnect and thermal-aware floorplanning for 3-D microprocessors," in *Proc. ISQED*, Mar. 2006, pp. 98–104.

[23] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou, "3D-STAF: Scalable temperature and leakage aware floorplanning for 3-D integrated circuits," in *Proc. ICCAD*, Nov. 2007, pp. 590–597.

[24] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A study of through-silicon-via impact on the 3-D stacked IC layout," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, 2009, pp. 674–680.

[25] C. Guedj, N. Claret, V. Arnal, M. Aimadeddine, and J. P. Barnes, "Evidence for 3-D/2-D transition in advanced interconnects," in *Proc. IRPS*, 2006, pp. 64–68.

[26] A. Vignon, S. Cosemans, W. Dehaene, P. Marchal, and M. Facchini, "A novel DRAM architecture as a low leakage alternative for SRAM caches in a 3-D interconnect context," in *Proc. Des. Autom. Test Eur. Conf. Exhibit.*, 2009, pp. 929–933.

[27] V. F. Pavlidis and E. G. Friedman, "3-D topologies for networks-on-chip," *IEEE TVLSI*, vol. 15, no. 10, pp. 1081–1090, Oct. 2007.

[28] R. Weerasekara, L.-R. Zheng, D. Pamunuwa, and H. Tenhunen, "Extending systems-on-chip to the third dimension: Performance, cost and technological tradeoffs," in *Proc. ICCAD*, 2007, pp. 212–219.

[29] B. Feero and P. P. Pande, "Performance evaluation for 3-D networks-on-chip," in *Proc. ISVLSI*, 2007, pp. 305–310.

[30] C. Addo-Quaye, "Thermal-aware mapping and placement for 3-D NoC designs," in *Proc. SOCC*, Sep. 2005, pp. 25–28.

[31] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das, "A novel dimensionally-decomposed router for

on-chip communication in 3-D architectures," *ISCA*, vol. 35, no. 2, pp. 138–149, May 2007.
[32] F. Li, C. Nicopoulos, T. Richardson, X. Yuan, V. Narayanan, and M. Kandemir, "Design and management of 3-D chip multiprocessors using network-in-memory," *ISCA*, vol. 34, no. 2, pp. 130–141, May 2006.
[33] D. Park, S. Eachempati, R. Das, A. K. Mishra, X. Yuan, N. Vijaykrishnan, and C. R. Das, "MIRA: A multilayered on-chip interconnect router architecture," in *Proc. ISCA*, Jun. 2008, pp. 251–261.
[34] I. Loi, F. Angiolini, and L. Benini, "Supporting vertical links for 3-D networks on chip: Toward an automated design and analysis flow," in *Proc. Nanonets*, Sep. 2007.
[35] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. De Micheli, "× pipesLite: A synthesis oriented design library for networks on chips," in *Proc. DATE*, 2005, pp. 1188–1193.
[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[37] S. Skiena. (1997). *The Algorithm Design Manual* [Online]. Berlin, Germany: Springer-Verlag. Available: http://sourceforge.net/projects/lpsolve
[38] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE TVLSI*, vol. 11, no. 6, pp. 1120–1135, Dec. 2003.
[39] N. Miyakawa, "A 3-D prototyping chip based on a wafer-level stacking technology," in *Proc. ASPDAC*, 2009, pp. 416–420.
[40] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for TSV-based 3-D network on chip links," in *Proc. ICCAD*, 2008, pp. 598–602.
[41] C. Lin, H. Zhou, and C. Chu, "A revisit to floorplan optimization by Lagrangian relaxation," in *Proc. ICCAD*, 2006, pp. 164–171.

Ciprian Seiculescu received the Bachelor degree in automation and applied informatics from the University Politehnica of Timisoara, Timisoara, Romania, and the Engineering degree in computer science from the Swiss Federal Institute of Technology, Lausanne, Switzerland. He is currently pursuing the Ph.D. degree from the Swiss Federal Institute of Technology.

Srinivasan Murali received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Palo Alto, CA, in 2007.

He is a Co-Founder and the Chief Technical Officer with iNoCs, Lausanne, Switzerland. He is also currently a Research Scientist with Ecole Polytechnique Fédérale de Lausanne, Lausanne. He has authored a book and presented over 40 publications in leading conferences and journals. His current research interests include interconnect design for systems-on-chips, thermal modeling, and reliability of multi-core systems.

Dr. Murali is a recipient of the European Design and Automation Association Outstanding Dissertation Award in 2007 for his work on interconnect architecture design. He received the Best Paper Award at the Design Automation and Test in Europe Conference in 2005. One of his papers has also been selected as one of "The Most Influential Papers of 10 Years DATE."

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 Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He has published more than 300 papers in peer-reviewed international journals and conferences, three books, several book chapters, and holds two U.S. patents. His current 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, biochips for the recognition of biological molecules, bioinformatics for the elaboration of the resulting information, and more advanced algorithms for in silico biology.

Dr. Benini has been the Program Chair and the Vice Chair of the Design Automation and Test in Europe Conference. He was a member of the 2003 MEDEA and EDA Roadmap Committee. He is a member of the IST Embedded System Technology Platform Initiative (ARTEMIS), a working group on design methodologies, is a member of the Strategic Management Board of the ARTIST2 Network of Excellence on Embedded System, and is a member of the Advisory Group on Computing Systems of the IST Embedded Systems Unit. He has been a member of the technical program committees and organizing committees of several technical conferences, including the Design Automation Conference, the 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 INTEGRATED CIRCUITS AND SYSTEMS and the *ACM Journal on Emerging Technologies in Computing Systems*.

Giovanni De Micheli (S'79–M'83–SM'89–F'94) received the Nuclear Engineer degree from the Politecnico di Milano, Milan, Italy, in 1979, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California, Berkeley, in 1980 and 1983, respectively.

He is currently a Professor and the Director of the Institute of Electrical Engineering and of the Integrated Systems Center, EPFL, Lausanne, Switzerland. He is the Program Leader of the Nano-Tera.ch Program. He was a Professor with the Electrical Engineering Department, Stanford University, Stanford, CA. He is the author of *Synthesis and Optimization of Digital Circuits* (New York: McGraw-Hill, 1994), and a co-author and/or a co-editor of eight other books and over 400 technical articles. His current research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies, networks on chips, and 3-D integration. He is also interested in heterogeneous platform designs including electrical components and biosensors, as well as in data processing of biomedical information.

Prof. Micheli has been serving IEEE in several capacities, including Division 1 Director from 2008 to 2009, Co-Founder and President Elect of the IEEE Council on EDA from 2005 to 2007, President of the IEEE CAS Society in 2003, and the Editor-in-Chief of the IEEE TRANSACTIONS ON CAD/ICAS from 1987 to 2001. He has been the Chair of several conferences, including DATE in 2010, pHealth in 2006, VLSI SOC in 2006, DAC in 2000, and ICCD in 1989. He is the recipient of the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems. He is a Fellow of the ACM. He received the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000. He received 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 1993, and a Best Paper Award at the DATE Conference in 2005.