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

Ciprian Seiculescu*, Srinivasan Murali§ *, Luca Benini‡, Giovanni De Micheli*
* LSI, EPFL, Lausanne, Switzerland,{ciprian.seiculescu, giovanni.demicheli}@epfl.ch
§ iNoCs, Lausanne, Switzerland, murali@inocs.com
‡ DEIS, Univerity of Bologna, Bologna, Italy, lbenini@deis.unibo.it

ABSTRACT

Three-dimensional integrated circuits 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 3D *SoC* that not only meets the application performance constraints, but also the constraints imposed by the 3D technology, is a significant challenge. In this work we present a design tool, *SunFloor 3D*, to synthesize application-specific 3D *NoCs*. The proposed tool determines the best NoC topology for the application, finds paths for the communication flows, assigns the network components on to the 3D layers and performs a placement of them in each layer. We perform experiments on several *SoC* benchmarks and present a comparative study between 3D and 2D NoC designs. Our studies show large improvements in interconnect power consumption (average of 38%) and delay (average of 13%) for the 3D NoC when compared to the corresponding 2D 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.

Keywords

3D ICs, Networks on chip (NoC), synthesis, topology, placement

1. INTRODUCTION

To continue the growth of the number of transistors on a chip, the 3D IC technology, where multiple silicon layers are stacked vertically, is emerging as a promising solution. The 3D ICs have a smaller footprint than a comparative 2D implementation. The long horizontal wires in a 2D design can be replaced by shorter and more efficient vertical wires, leading to lower interconnect delay and power consumption. Wafer-to-wafer bonding technology, where the vertical interconnects are implemented using *Through Silicon Vias (TSVs)*, is one of the popular choices for 3D integration [36].

The interconnects for 3D have evolved from simple vertical links connecting buses in different 3D layers to a more scalable *Network on Chip (NoC)* solution [1]-[3]. NoCs consists of switches and links and use circuit or packet switching to transfer data through the system. NoCs are a necessity for 3D chips, as they are modular, provide configurable parallelism and can control the number of TSVs required across layers.

Most *Systems on Chips (SoCs)* are composed of heterogeneous cores and target a specific application domain. To obtain a power-performance efficient implementation, it is important to build a NoC topology that is tailored to match the application communication characteristics, such as bandwidth and latency constraints of data flows. Recently, researchers have addressed the issue of NoC topology synthesis for 2D SoCs [11]-[18], including our earlier work on developing a synthesis flow for 2D SoCs [18]. However, topology synthesis for a 3D SoC introduces several new and significant challenges: (i) the number of TSVs that can be allowed across any two layers strongly depends on the underlying 3D fabrication technology. For technologies where a large number of TSVs are available, cores across different 3D layers can share the same switches, as more vertical links can be established. On the other hand, with a tight TSV constraint, a core on a layer may need to connect to a switch in the same layer. Thus, depending on the TSV constraint, the topology synthesized can be very different and the synthesis procedure should be able to handle this, (ii) the assignment of switches to different layers needs to be performed, and (iii) TSV macros need to be introduced in each layer for a vertical link and placement of switches and TSV macros need to be performed.

Building a custom application-specific NoC topology will be instrumental in pushing 3D NoC technology in industrial designs. In this paper, we make two major contributions: first, we present, *SunFloor 3D*, a tool for synthesizing application specific NoCs for 3D SoCs. Second, we consider several realistic SoC benchmarks and perform a comparative study of the NoCs designed for a 3D and a 2D implementation of the benchmarks. The study shows the advantages of using 3D technology for reducing interconnect power consumption and zero-lead latency (in the rest of the paper, this referred to as just latency) on realistic SoC designs.

The application communication characteristics and, optionally, a floorplan of the SoC (with the positions of each core) before instantiating the NoC are taken as inputs to the design process. The tool synthesizes the NoC topology that optimizes the design objectives (such as minimizing power consumption or latency), performs layer assignment of switches and placement of switches and TSV macros. The placement is performed such that the resulting floorplan is minimally perturbed from the input floorplan and the NoC wire lengths are minimized. From the floorplan, the tool can obtain the length of the NoC links, and hence account for wire delay and power consumption accurately during the synthesis process.

Several works have addressed the problems of assigning cores to the different layers and performing 3D floorplanning, considering thermal issues [22]-[25]. Also, in many designs, the layer assignment of cores is dictated by technology constraints (such as having logic and memory dies on different layers). Thus, in this work, we take the floorplan of the cores as an input to our flow. Our work is complementary to the works on 3D floorplanning [22]-[25], as we only place the network components on the given input floorplan, minimally perturbing it.

We perform experiments on several SoC benchmarks that show large power (54% on average) and latency (21% on average) im-

978-3-9810801-5-5/DATE09 © 2009 EDAA

provement for the synthesized topologies when compared to standard topologies. For comparative purposes, we also apply a 2D synthesis flow developed earlier by us [18] for a corresponding 2D implementation of the benchmarks. Our results show that a 3D design can significantly reduce the interconnect power consumption (38% on average) and latency (13% on average).

2. 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 2D SoCs has been presented in [11]-[18]. In [18], we presented a method to synthesize the most power-performance efficient NoC topologies for 2D SoCs.

Several works have been investigating the 3D manufacturing processes [20], [21], [36]. Methods for 3D floorplanning and placement of cores, taking into account the thermal issues has been presented in [22]-[25]. Manufacturing of 3D interconnects has been addressed by [26] and [27]. Multi-dimensional regular topologies (like k-ary n-cubes, hypercubes) have been explored by researchers as viable interconnect solutions for chip-to-chip networks [19]. However, such standard topologies are not suitable for application specific SoCs, which are heterogeneous in nature.

Synthesis of NoCs for 3D SoCs is a relatively new topic. New switch architectures for 3D have been presented in [33] and [35]. In [34], the authors present the use of NoCs as interconnects for 3D multi-processors. The electrical characteristics of vertical interconnects are analyzed in [36] and the authors also present a back-end design flow to implement 3D NoCs. An analytical cost metric for 3D NoCs is presented in [30]. Design of standard topologies for 3D is analyzed in [31] and mapping of cores on to NoC topologies is presented in [32]. Power-delay analysis of 3D interconnects is presented in [28]. However, none of these works address the issue of synthesizing custom NoCs topologies for 3D SoCs. Moreover, the works do not present a comparison of the NoC power and latency for 2D and 3D NoC implementations.

In [38], we presented a synthesis algorithm, which is based on a direct extension of the 2D NoC synthesis procedure. In the work, the NoC was designed for each layer separately, and the connectivity of the switches across the layers was then determined. The method forces cores in a layer to be connected to switches in the same layer and allows vertical interconnects only across adjacent layers. Thus, the inter-layer flows incur large power and latency penalty (shown in Sub-section 4.2). In this paper, we propose a more general approach for topology synthesis, where the cores across layers can share switches, depending on the TSV constraints. Also, the work in [38] does not address additional issues, such as assigning switches to layers, placement of TSV macros and placement of network components with minimum perturbation of the input floorplan.

3. TOPOLOGY SYNTHESIS

The core names, their communication requirements and the floorplan (core sizes, locations) are obtained as inputs to the tool. The maximum number of TSVs across any two layers is also obtained as an input. The NoC architectural parameters (such as NoC operating frequency and data width) can either be given as inputs, or can be varied in steps in a user-defined range. For a given link width, the constraint on the maximum TSVs can be translated to a constraint on the maximum number of inter-layer links (referred as max_ill) allowed. To estimate the power, timing and area for the generated topology, models of the NoC components (switches and links) are obtained as inputs.

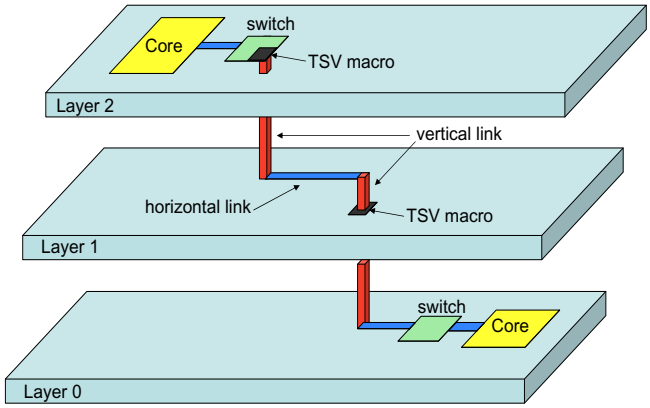


Figure 1: Example vertical link

In Figure 1, we show an example vertical link connecting two switches that are in the bottom-most and top-most layers. 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.

As topology synthesis is NP-Hard [10], we present efficient heuristics for solving the problem. Varying the number of switches in a design has a great impact on the power consumption and latency. Increasing the number of switches may lead to smaller switch sizes, but also an increase in power consumption, as the packets have to traverse more switches. On the other hand, with more switches, the link power consumption may be reduced, as the links get shorter. The total power consumption is given by the combined effect of both these factors and is very hard to predict beforehand. In our method, topologies with different switch counts are synthesized and the most power-performance efficient design is chosen.

In the synthesis procedure, we perform the following steps: first, we establish core to switch connectivity (Sub-section 3.1). Then, we obtain deadlock-free paths for traffic flows (Sub-section 3.2). Then, placement of switches and TSVs on the input floorplan are obtained (Sub-section 3.3).

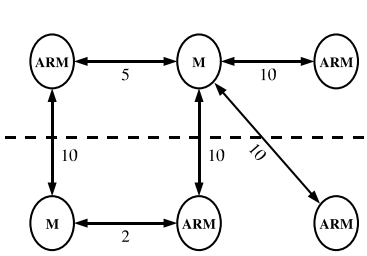
3.1 Establishing Core to Switch Connectivity

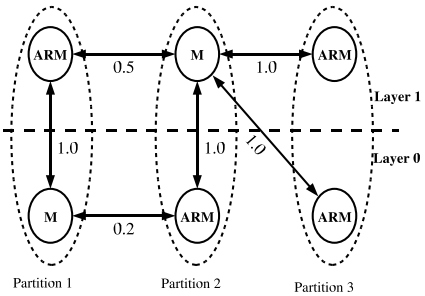
In this sub-section, we present methods for establishing connectivity between the cores and switches.

DEFINITION 1. *Let n be the number of cores in the design. The 3D layer to which a core i is assigned is represented by $layer_i$.*

The communication characteristics of the application are obtained and represented by a graph [7], 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*





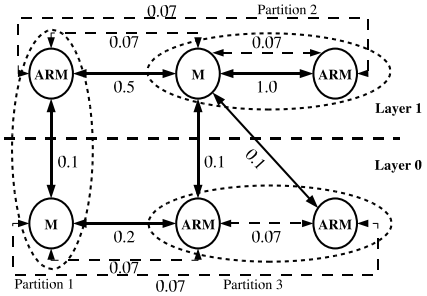


Figure 2: LPG and the min-cut partitions **Figure 3: PG and the min-cut partitions** **Figure 4: SPG and the min-cut partitions**

cores v_i and v_j. The bandwidth of traffic flow from cores 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 partitioning graph is a directed graph, $PG(U, H, \alpha)$, that has the same set of vertices and edges as the communication graph. The weight of the edge (u_i, u_j), defined by $h_{i,j}$.*

In this paper, $h_{i,j}$ 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. 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.

Algorithm 1 Core-to- switch connectivity

```
1: Build partitioning graph PG(U,H,α)
2: Unmet = φ.
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 j th partition, ∀j ∈ 1 ··· i.
6:    {Compute layer assignment for each switch:}
7:    layer_sw_j = (Σ_{∀k∈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: θ = θ_min
12: while ((Unmet! = φ) & (θ ≤ θ_max)) do
13:    for Each i ∈ 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:    θ = θ + θ_scale
20: end while
```

In the first step of Algorithm 1, the partitioning graph is built. 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, that many min-cut partitions of PG are obtained. All the cores in a partition are connected to the same switch and the partitioning is done such that each partition has about equal number of cores. Thus, those traffic flows with large bandwidth requirements or tight latency constraints are assigned to the same partition and traverse a single hop in the network.

EXAMPLE 1. *For the communication graph from Figure 2, an example partitioning graph is shown in Figure 3. 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 $\alpha = 1$ and the bandwidth of the traffic flowing between cores with in a layer is lower than the traffic between the cores across the layers. In the figure, we also show an example of 3 min-cut partitions of the graph. The partitioning leads to cores in different layers being assigned to the same partition.*

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 *Scaled Partitioning Graph*, defined as follows:

DEFINITION 4. *A scaled partitioning graph 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 $layer_i = layer_j$.*

That is, in the SPG, along with the edges in PG, we define new edges between all cores in the same layer of 3D. 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 to 19), until the $\max\_ill$ constraint is met. After several experimental runs, we determined that varying θ from 1 to 15 in steps of 3 gives good results.

In order to cluster cores in a layer that actually communicate, 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:

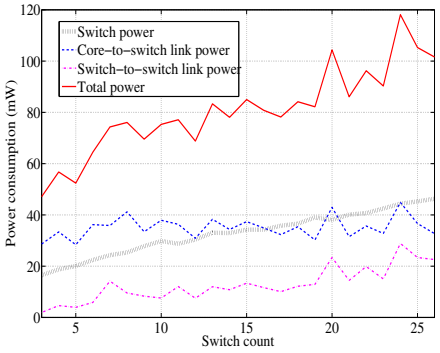


Figure 5: Power consumption in 2D

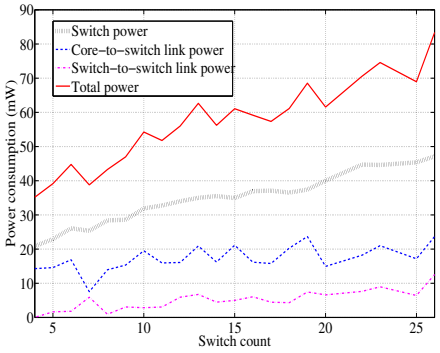


Figure 6: Power consumption in 3D

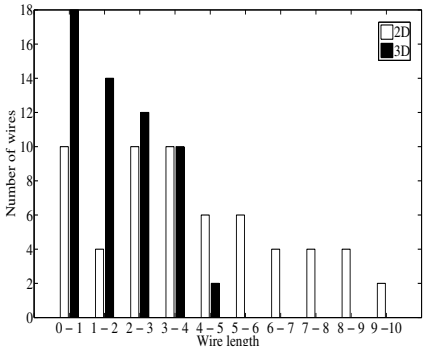


Figure 7: Wire length distributions

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

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 Figure 4. In the SPG, the inter-layer links have lower weights and new edges are added between cores with in the same layer. The 3 min-cut partitions are now different, with more cores in the same layer belonging to the same partition.*

3.2 Path Computation

When establishing links across switches, we need to consider the max_ill constraint. In this paper, we do not show the entire path computation algorithm (including the removal of deadlocks), as it is similar to the 2D case presented in earlier works, such as [15] and [18]. We only show how the max_ill constraint is handled.

During path computation, a high cost ($SOFT\_INF$) is assigned for establishing links between switches that would lead to increasing the number of vertical links across any layer above $soft\_max\_ill$ value. The $soft\_max\_ill$ is set to be few links less than the max_ill value. The use of this softer constraint first, facilitates the procedure to obtain more valid paths, when compared to directly using the hard max_ill constraint. From experiments, we set the cost $SOFT\_INF$ to be ten times the maximum cost of any flow and the value of $soft\_max\_ill$ to be 2 or 3 links less than max_ill. We show the use of this softer constraint in Algorithm 2, for evaluating the cost of establishing a physical link across two switches i and j.

Algorithm 2 Check Constraints (i,j)

1: Define $soft\_max\_ill$ and $SOFT\_INF$
2: Estimate power consumption increase in opening and using link from i to j.
3: If establishing link increases vertical link count across any layer above $soft\_max\_ill$, assign cost of SOFT_INF for using switch i to j
4: If establishing link increases vertical link count across any layer above max_ill, assign cost of INF for using switch i to j

3.3 Placement of NoC Components

The ideal position of the switches and TSVs can be computed based on the positions of the cores and the connectivity between the switches and cores. For example, a switch position can be computed to be equidistant from all the cores to which it is connected. 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. 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. As more components are placed, they can re-use the gap created by the earlier components. For lack of space, we do not present the detailed algorithm for placement in this paper.

4. EXPERIMENTS AND CASE STUDIES

For the experiments, the NoC component library from [37] is used. The power and latency values of the switches and links of the library are determined from post-layout simulations, based on 65nm low power libraries. The vertical interconnects using TSVs are implemented based on the models from [36]. In [36], the authors show that the vertical links have much lower resistance and capacitance (an order of magnitude reduction) when compared to horizontal links.

4.1 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 multi-media processing. 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 3D. For comparisons, we also consider a 2D implementation of the benchmark. The floorplans of the cores in each layer of the 3D and for the 2D design are obtained using existing tools [39]. 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 2D case, We use our synthesis flow developed earlier [38].

In Figures 5 and 6, 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 and the NoC operating frequency to the minimum point at which valid topologies are obtained (computed to be 400 MHz for this benchmark). We use a *max_ill* constraint of 25 links for this and the experiments in the next sub-section. In Sub-section 4.3, we study the impact of varying this constraint.

When very few switches are used in the design, they need to have more input/output ports, as they need to connect to more cores. A large switch can only support a low operating frequency, as the crit-

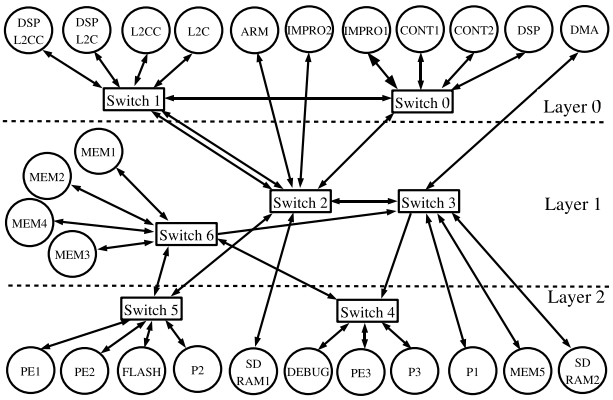


Figure 8: Most power-efficient topology

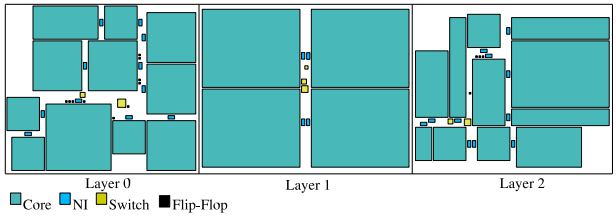


Figure 9: Resulting 3D floorplan with switches

Table 1: 2D vs 3D NoC Comparison

Benchmark	Power (mW)						Latency (cyc)	
	Link power		Switch power		Total power			
	2D	3D	2D	3D	2D	3D	2D	3D
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

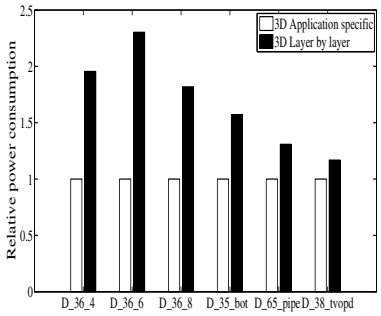


Figure 10: Comparison with layer-by-layer

ical path inside the switch increases with its size. In order to meet the 400 MHz requirement, we could only obtain valid topologies with 4 or more switches, thus the plots starts at 4 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 savings of 25% for the 3D relative to the 2D case. This is due to the fact that the long horizontal wires in a 2D design are replaced by shorter vertical wires. In Figure 7, we show the wire-length distribution of the links in 2D and 3D cases. From the figure, as expected, the 2D design has many long wires. In Figures 8 and 9, we present an example topology synthesized by our tool and the floorplan of the cores and network components for the 3D case.

4.2 2D vs. 3D Comparison

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 4, 6 and 8 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 3 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_tvopd). In the last two benchmarks, each core communicates only to one or few other cores.

The power consumption for the least power design points for 2D and 3D, as well as the average latency are presented in Table 1. Most of the power savings obtained in 3D 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 3D case when compared to a 2D implementation.

In Figure 10, we show the power consumption of the topologies synthesized by the approach presented in [38], with respect to the proposed approach for the different benchmarks. The method in [38] connects cores in a layer to switches in the same layer. Thus, the inter-layer traffic needs to traverse more switches to reach the destination, leading to an increase in power consumption and latency. As seen from Figure 10, the method presented in this work results in 40% reduction in NoC power consumption, when compared to the earlier work.

4.3 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, there by leading to higher power consumption and latency. We perform topology synthesis for the D_26_media design with different *max_ill* constraint values, and the power, latency values for the different points are presented in Figures 11 and 12. With a tighter TSV constraint, the power consumption and latency increases significantly, as more switches are needed in the design.

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 Figure 13. Our experiments also showed that we obtain 21% reduction in latency when compared to the optimized mesh.

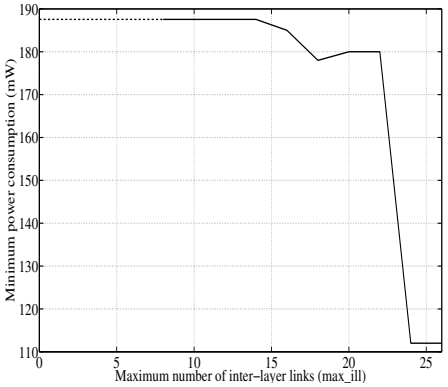


Figure 11: Impact of max_ill on power

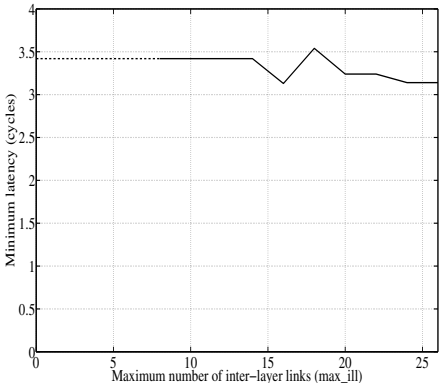


Figure 12: Impact of max_ill on latency

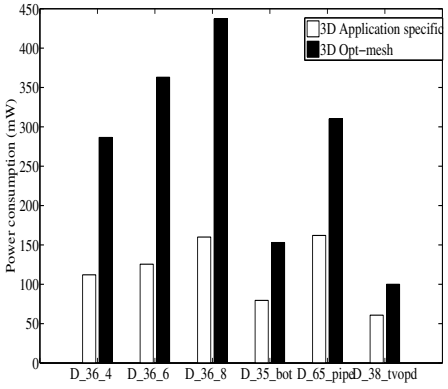


Figure 13: Comparisons with mesh

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). 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.

5. CONCLUSIONS

Networks on Chips (NoCs) are necessary to achieve a scalable communication infrastructure in 3D 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 3D technological constraints, is a critical problem that needs to be addressed. In this work, 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 3D layers. Our experiments on several realistic benchmarks show 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 2D and 3D, which shows that 3D integration can produce large interconnect power and latency reduction (38% and 13%, respectively). In future, we plan to address the design of NoCs for *Globally Asynchronous Locally Synchronous (GALS)* paradigm for 3D ICs.

6. ACKNOWLEDGMENT

We would like to acknowledge the financial contribution of the European Union under Project ICT-ARTIST-DESIGN.

7. REFERENCES

[1] L.Benini and G.De Micheli, “Networks on Chips: A New SoC Paradigm”, IEEE Computers, pp. 70-78, Jan. 2002.
[2] P.Guerrier, A.Greiner,”A generic architecture for on-chip packet switched interconnections”, Proc. DATE, pp. 250-256, March 2000.
[3] G. De Micheli, L. Benini, “Networks on Chips: Technology and Tools”, Morgan Kaufmann, First Edition, July, 2006.
[4] J. Hu et al., “System-Level Point-to-Point Communication Synthesis Using Floorplanning Information”, Proc. ASPDAC ’02.
[5] S. Murali, G. De Micheli, “An Application-Specific Design Methodology for STbus Crossbar Generation”, pp. 1176-1181, Proc. DATE ’05.
[6] S. Pasricha et al., “Floorplan-aware automated synthesis of bus-based communication architectures”, Proc. DAC ’05.
[7] J. Hu, R. Marculescu, ’Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures’, Proc. DATE, March 2003.
[8] S. Murali, G. De Micheli, “SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs”, Proc. DAC 2004.
[9] S. Murali, G. De Micheli, “Bandwidth Constrained Mapping of Cores on to NoC Architectures”, Proc. DATE 2004.
[10] S. Murali et al., “Mapping and Physical Planning of Networks on Chip Architectures with Quality-of-Service Guarantees”, Proc. ASPDAC 2005.
[11] A.Pinto et al., “Efficient Synthesis of Networks on Chip”, ICCD 2003, pp. 146-150, Oct 2003.
[12] W.H.Ho, T.M.Pinkston, “A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns”, HPCA, 2003.
[13] T. Ahonen et al. ”Topology Optimization for Application Specific Networks on Chip”, Proc. SLIP 04.
[14] K. Srinivasan et al., “An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks”, Proc. ICCAD ’05.
[15] A. Hansson et al., “A Unified Approach to Mapping and Routing on a Combined Guaranteed Service and Best-Effort Network-on-Chip Architectures”, Technical Report No: 2005/00340, Philips Research, April 2005.
[16] X.Zhu, S.Malik, “A Hierarchical Modeling Framework for On-Chip Communication Architectures”, ICCD 2002, pp. 663-671, Nov 2002.
[17] J. Xu et al., “A design methodology for application-specific networks-on-chip”, ACM TECS, 2006.
[18] S. Murali et al., “Designing Application-Specific Networks on Chips with Floorplan Information”, pp. 355-362, ICCAD 2006.
[19] W. J. Dally, “Performance Analysis of k-ary n-cube Interconnection Networks”, IEEE Transactions on Computers, Vol. 39, No. 6, pp. 775-785, 1990.
[20] K. Banerjee et al., “3-D ICs: A Novel Chip Design for Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration”, Proc. of the IEEE, vol. 89(5), pp. 602, 2001.
[21] B. Goplen and S. Sapatnekar, “Thermal Via Placement in 3D ICs”, Proc. Intl. Symposium on Physical Design, pp. 167, 2005.
[22] J. Cong et al., “A thermal-driven floorplanning algorithm for 3D ICs”, ICCAD, Nov. 2004.
[23] W.-L. Hung et al., “Interconnect and thermal-aware floorplanning for 3D microprocessors”, Proc. ISQED, March 2006.
[24] S. K. Lim, “Physical Design for 3D System on Package”, IEEE Design & Test of Computers, vol. 22(6), pp. 532539, 2005.
[25] P. Zhou et al., ”3D-STAF: Scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits”, ICCAD, Nov. 2007.
[26] C. Guedj et al., “Evidence for 3D/2D transition in advanced interconnects”, Proc. IRPS, 2006.
[27] IMEC, http://www2.imec.be/imec_com/3d-integration.php
[28] V. F. Pavlidis and E. G. Friedman, ” 3-D Topologies for Networks-on-Chip,” IEEE TVLSI, October 2007
[29] R. Weerasekara et al., “Extending Systems-on-Chip to the Third Dimension: Performance, Cost and Technological Tradeoffs”, Proc. ICCAD, 2007.
[30] V. F. Pavlidis and E. G. Friedman, “Topologies for networks-onchip”, Proc. SOCC, 2006.
[31] B. Feero and P. P. Pande, “Performance evaluation for three-dimensional networks-on-chip”, Proc. ISVLSI, 2007.
[32] C. Addo-Quaye, “Thermal-Aware Mapping and Placement for 3-D NoC Designs”, Proc. SOCC, 2005.
[33] J. Kim et al., “A novel dimensionally-decomposed router for on-chip communication in 3d architectures”, ISCA, 2007.
[34] F. Li et al., “Design and Management of 3D Chip Multiprocessors Using Network-in-Memory”, ISCA, pp. 130-141, 2006.
[35] D. Park et al., “MIRA: A Multi-Layered On-Chip Interconnect Router Architecture”, Proc. ISCA, 2008.
[36] I. Loi, F. Angiolini, L. Benini, Supporting vertical links for 3D networks on chip: toward an automated design and analysis flow, Proc. Nanonets, 2007.
[37] S. Stergiou et al., “×pipesLite: a Synthesis Oriented Design Library for Networks on Chips”, pp. 1188-1193, Proc. DATE 2005.
[38] S. Murali et al., “Synthesis of Networks on Chips for 3D Systems on Chips”, ASPDAC 2009.
[39] S. N. Adya, I. L. Markov, ”Fixed-outline Floorplanning : Enabling Hierarchical Design”, IEEE TVLSI, Dec 2003.