Synthesis of Networks on Chips for 3D Systems on Chips

Srinivasan Murali*, Ciprian Seiculescu*, Luca Benini‡, Giovanni De Micheli*

* LSI, EPFL, Lausanne, Switzerland,
{srinivasan.murali, ciprian.seiculescu, giovanni.demicheli}@epfl.ch
‡ DEIS, Univerity of Bologna, Bologna, Italy, lbenini@deis.unibo.it

ABSTRACT

Three-dimensional stacking of silicon layers is emerging as a promising solution to handle the design complexity and heterogeneity of *Systems on Chips (SoCs)*. *Networks on Chips (NoCs)* are necessary to efficiently handle the 3D interconnect complexity. Designing power efficient NoCs for 3D SoCs that satisfy the application performance requirements, while satisfying the 3D technology constraints is a big challenge. In this work, we address this problem and present a synthesis approach for designing power-performance efficient 3D NoCs. We present methods to determine the best topology, compute paths and perform placement of the NoC components in each 3D layer. We perform experiments on varied, realistic SoC benchmarks to validate the methods and also perform a comparative study of the resulting 3D NoC designs with 3D optimized mesh topologies. The NoCs designed by our synthesis method results in large interconnect power reduction (average of 38%) and latency reduction (average of 25%) when compared to traditional NoC designs.

Keywords

3D, networks on chip, topology synthesis, application-specific

1. INTRODUCTION

The 2D chip fabrication technology is facing lot of challenges in utilizing the exponentially growing number of transistors on a chip. The wire delay and power consumption is increasing dramatically and achieving interconnect design closure is becoming a challenge. Designing the clock-tree network for a large chip is becoming very challenging and its power consumption is a significant fraction of total chip power consumption. Moreover, diverse components that are digital, analog, MEMS and RF are being integrated on the same chip, resulting in large complexity for the 2D manufacturing process [19].

Vertical stacking of multiple silicon layers, referred to as 3D stacking, is emerging as an attractive solution to continue the pace of growth of Systems on Chips (SoCs) [19]-[24]. The 3D technology results in smaller footprint in each layer and shorter vertical wires that are implemented using *Through Silicon Vias (TSVs)* across the layers. Heterogeneous systems can be built easily, with each layer supporting a diverse technology [19]. The 3D technology has been maturing over the years in addressing thermal issues and achieving high yield [20].

To tackle the on-chip communication problem, a scalable communication paradigm, *Networks on Chips (NoCs)* has recently evolved [1]-[3]. NoCs are composed 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 3D 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 3D integration technologies and NoCs introduces new opportunities and challenges for designers. Building power-efficient NoCs for 3D 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 3D has received some attention [30]-[33], there has been little work on design methods for 3D NoCs. The design methods for 2D NoCs do not consider important 3D information, such as the technology 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 3D.

In this work, we address this important problem and present a synthesis approach for designing the most power efficient 3D NoC that meets application performance and technology constraints. We present 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 3D layers and the floorplan of the cores in each layer are taken as inputs to 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. Please note that the assignment of cores to the different layers and the floorplan of each layer needs to consider several performance and technological constraints, such as thermal issues. There are several works that address these issues [21]-[24] and our work is complementary to them. Here, we only address the issue of designing the NoC topology and determining the placement of the NoC switches. As in our output floorplan (after placing the switches), the core positions are almost the same as the input floorplan, we minimally affect these (such as the thermal) issues.

We perform experiments on varied, realistic SoC benchmarks to validate the methods. Our results show that the topologies synthesized by our method results in large interconnect power reduction (an average of 38%) and latency reduction (25% on average), when compared to optimized standard NoC topologies.

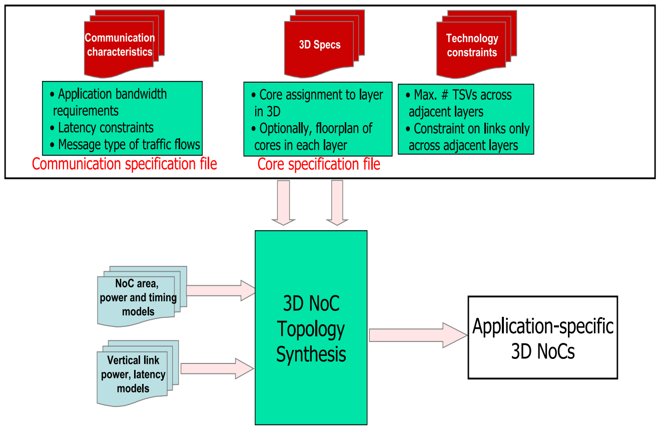
Communication characteristics
3D Specs
Technology constraints
• Application bandwidth requirements
• Latency constraints
• Message type of traffic flows
Communication specification file
• Core assignment to layer in 3D
• Optionally, floorplan of cores in each layer
Core specification file
• Max. # TSVs across adjacent layers
• Constraint on links only across adjacent layers
NoC area, power and timing models
Vertical link power, latency models
3D NoC Topology Synthesis
Application-specific 3D NoCs

Figure 1: Proposed 3D NoC design approach

2. RELATED WORK

The use of NoCs to replace bus-based designs has been presented in [1]-[2]. Several different NoC architectures and design methods [4]-[5] have been developed over the past few years. A detailed description of the important design issues and the current state-of-the-art in NoC architectures and design methods is presented in [3].

Synthesis of bus and NoC architectures has been addressed by several researchers for 2D systems. Mapping and placement of cores onto standard NoC topologies has been explored in [6]-[9]. Synthesis of application specific NoC topologies has been addressed in [10]-[17]. In [17], we presented a NoC synthesis method for 2D SoCs and performed detailed comparisons with standard topologies and other mapping tools. In this paper, we use the basic principles from the 2D method to address the important issues in 3D NoC design.

Several works have been presented on the 3D manufacturing processes and interconnects [19], [20], [33]. A performance and cost trade-off analysis of 3D integration is presented in [26]. Several works have explored 3D floorplanning, placement and temperature issues of cores [21]-[24]. These works do not consider the interconnect synthesis problem. Multi-dimensional topologies (such as k-ary n-cubes, hypercubes) have been extensively explored in the chip-to-chip interconnection field [18]. However, such works only consider standard topologies suitable for homogeneous designs. Most SoCs, especially in 3D, are heterogeneous in nature and require application-specific interconnect architecture to optimize power and performance. Moreover, such works do not address the optimization of topologies based on traffic patterns.

Analysis and synthesis of NoCs for 3D technology is a relatively new topic. Novel NoC switch architectures for 3D are presented in [30] and [32]. In [31], the authors present the use of NoCs in 3D multi-processors. In [33], the authors analyze the electrical characteristics of vertical interconnects and show a back-end design flow to implement 3D NoCs. In [27], the authors present an analytical model for cost metrics of 3D NoCs and compare them with 2D NoCs. In [28], design of standard NoC topologies (such as mesh) for 3D is analyzed. Mapping and placement of cores with thermal constraints on to NoC topologies is presented in [29]. However, none of these works address the issue of synthesizing application-specific 3D NoC topologies.

3. DESIGN APPROACH

The approach used for topology synthesis is presented in Figure 1. In the *core specification* file, the name of the different cores, the sizes and positions are obtained as inputs. The assignment of the cores to the different layers in 3D is also obtained as an input. 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.

To achieve high yield, the number of TSVs that can be established across two layers may need to be restricted below a threshold [25]. In the rest of the paper, we model the maximum TSV 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.

For the synthesis procedure, the power, area and timing models of the NoC switches and links are also taken as inputs. We also take the power consumption and latency values of the vertical interconnects as inputs. The output of the topology synthesis procedure is a set of Pareto design 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 3D layers and the positions of the switches. The TSV macros needed for establishing vertical links are directly integrated in the switch input/output ports, as done in [33].

As the topology synthesis and mapping problem is NP-Hard [10], we present efficient heuristics to synthesize the best topology for the design. For achieving high yield, it is important to restrict the number of vertical links used and to allow vertical connections only across adjacent layers on the 3D chip [25]. Thus, in our procedure, we connect cores in a layer only to switches in the same layer, and ensure that switches of a layer are directly connected only to switches in adjacent layers.

A NoC having fewer switches leads to longer core to switch links and hence, higher link power consumption. On the other hand, when many smaller switches are used, the flows have to traverse more switches, leading to larger switch power consumption. Thus, we need to explore designs with several different switches to obtain the best solution, starting from one where all the cores are connected to a single switch in a layer to a design point where each core is connected to a separate switch. For each switch count, we determine the core to switch connectivity, as explained in Section 4. Then, we determine connectivity across the different switches (Section 5). Then, we determine the optimal positions of the switches on the floorplan (Section 6) and determine the wire lengths and link power consumption.

4. ESTABLISHING NUMBER OF SWITCHES

In this section, we present methods for establishing connectivity between the cores and switches. From the *core specification* file, we obtain the core specifications:

DEFINITION 1. *Let n be the number of cores in the design. The x and y co-ordinate positions of a core i are represented by xc_i and yc_i respectively, $\forall i \in 1 \cdots n$. The 3D 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 [6], [9], 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 cores v_i to v_j*

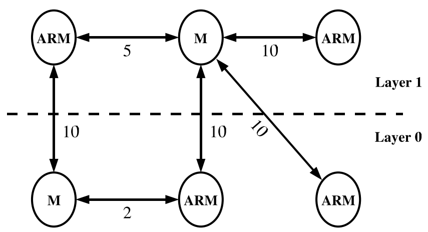

Figure 2: Communication graph example

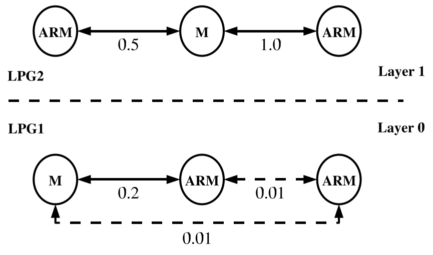

Figure 3: LPGs for the two layers

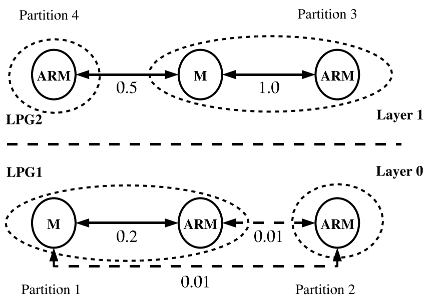

Figure 4: Two min-cut partitions of LPGs

is represented by $bw_{i,j}$ and the latency constraint for the flow is represented by $lat_{i,j}$.

We define the *Local Partitioning Graph* for each layer:

DEFINITION 3. *A local partitioning graph, LPG(Z, M, ly), is a directed graph, with the set of vertices represented by Z and edges 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.*

The LPGs for the two layers of the *communication graph* from Figure 2 are shown in Figure 3. Since the LPGs are built layer by layer, the graphs for the two layers are independent of one another. 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 1. As the number of input/output 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 local partitioning graph for each layer is built.

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, that 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. Two min-cut partitions of the LPGs of Figure 3 are shown in Figure 4. Once the partitions for all the layers are obtained, the cores in a partition are attached to the same switch and hence the core to switch connectivity is obtained. The next step is to determine switch to switch connectivity, by finding paths for the inter switch traffic flows. This is explained further in the next section.

Algorithm 1 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(Z,M,j)
14:     end for
15:     Compute paths for inter-switch flows (Section 5).
16:     If valid paths found, save the current design point
17: end for
```

5. 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 3D case is similar to the 2D case, such as those presented in [14], [17], 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], [17], 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.

DEFINITION 4. *Let nsw be the total number of switches used across all the layers and let $layer_i$ 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 2, 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, 4, we assign a cost of INF 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, 6). From experiments, we found that a reasonable value for $SOFT\_INF$ to be 10 times the maximum cost of any flow and $soft\_max\_switch\_ill$ to be few (2 to 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 only using the hard constraints.

Algorithm 2 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
```

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. Due to space limitations, in this paper, we do not explain the details of how the indirect switches are established.

6. 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) [34].

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 by:

$$coredist_{i,k} = \begin{cases} |xs_i - xc_k| + |ys_i - yc_k| & \text{, if } switch_i \text{ connected to } core_k \\ 0 & \text{, otherwise} \end{cases} \quad (1)$$

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

$$swdist_{i,j} = \begin{cases} |xs_i - xs_j| + |ys_i - ys_j| & \text{, if } switch_i \text{ connected to } switch_j \\ 0 & \text{, otherwise} \end{cases} \quad (2)$$

The above equations can be easily represented as a set of linear equations [34]. 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:

$$\begin{aligned} obj \ = \ & \textstyle\sum_{\forall i} \sum_{\forall k} coredist_{i,k} * bw\_sw2core_{i,k} \\ & + \textstyle\sum_{\forall i} \sum_{\forall j} swdist_{i,j} * bw\_sw2sw_{i,j} \end{aligned} \quad (3)$$

The LP for optimization is written as follows:

$$\begin{array}{ll} \text{minimize} & obj \\ \text{subject to} & Equations\ 1-3 \\ & xs_i, ys_i \ \geq \ 0, \ \forall i \in 1 \cdots nsw \end{array} \quad (4)$$

We use the *lp_solve* package [35] 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. However, the optimal positions can result in overlap of switches among themselves or with the cores. To remove the overlaps, we use the floorplanner, Parquet [36], layer by layer. 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. We also pipeline long links to support full throughput on the NoC and add Network Interfaces (NIs) to connect the cores to the network. The resulting design is a valid floorplan of the NoC.

7. EXPERIMENTS AND CASE STUDIES

For our experiments, we use the switch and link libraries from [5]. The power consumption and latency numbers of the components of the library are obtained after post-layout analysis. We use 65nm low power technology libraries for the layout studies. For the electrical characteristics of vertical interconnects, we use the models from [33]. To obtain the electrical characteristics, a wafer-to-wafer bonding technique is used as the underlying 3D integration technology. The vertical links are shown to have an order of magnitude lower resistance and capacitance than a horizontal link of the same dimension. This translates to a traversal delay of less than 10% of clock cycle for 1 GHz operation and negligible power consumption on the vertical links.

7.1 Multimedia SoC case study

For experimental case study, we consider a multi-media SoC, *Triple Video Object Plane Decoder*, that has 38 cores (D_38_tvopd). The *communication graph* of the benchmark is presented in Figure 5, where each vertex represents a core and the weight on the edge represents the bandwidth between the cores expressed in MB/s. The application is highly heterogeneous in nature, having three independent decoders working in parallel to improve performance. Each decoder has 12 cores organized in a pipeline fashion. There are two extra memories that are shared between the pipelines that serve as input and output buffers. We consider the design implemented on to 3 layers in 3D. The assignment of cores to the different layers and the floorplan of each layer were done manually, such that the performance and manufacturing constraints (such as thermal issues) are met. The processing cores are placed on the top and bottom layers, so that they are close to the heat sink. The large memory cores are all placed on the middle layer because they produce less heat and because this allows the manufacturer to use an efficient integration process for implementing the memories. The floorplan of the design (along with the network components synthesized by our procedure) is presented in Figure 7.

The data width of the NoC links is fixed to 32 bits, to match the data width of the cores in the design. We allowed the synthesis method to sweep the NoC frequency and obtain NoC design points

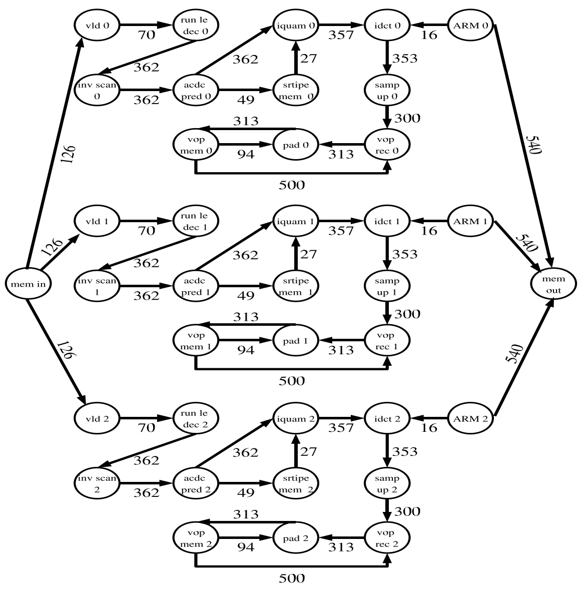

Figure 5: Communication graph for the D_38_tvopd benchmark

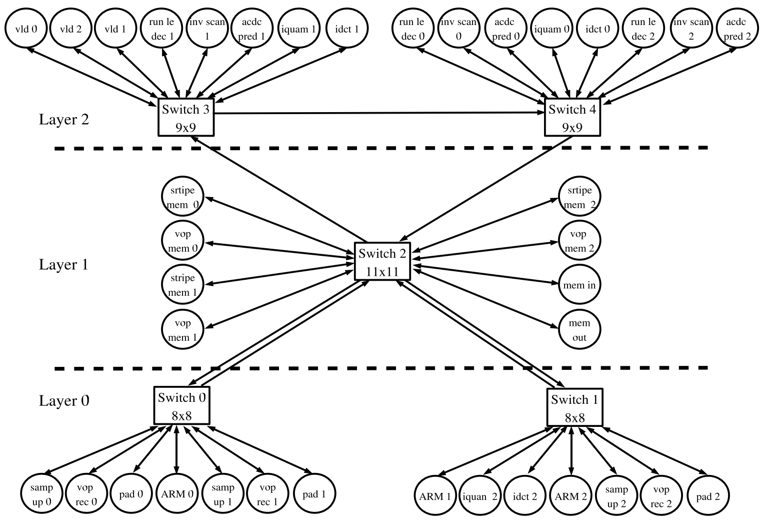

Figure 6: Most power-efficient topology

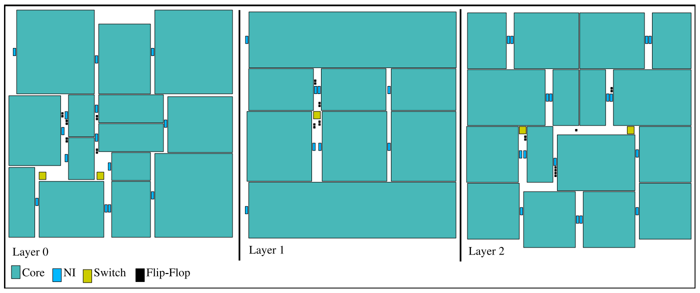

Figure 7: Resulting 3D floorplan with switches

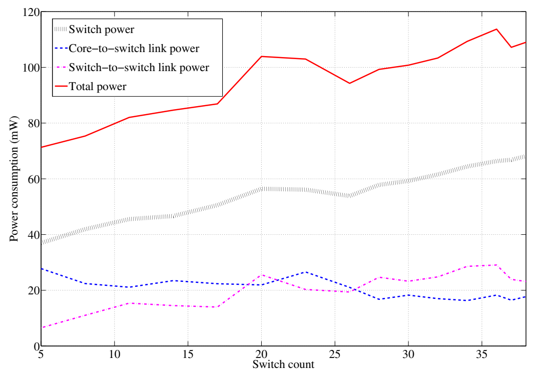

Figure 8: Power consumption

for different frequencies. From the resulting design points, we found that the lowest operating frequency (of 500 MHz) resulted in least power consumption for this design. The power consumption of NoC designs synthesized by our procedure for different switch counts, at 500 MHz operation, is presented in Figure 8. In the figure, we show the core-to-switch link power, the switch-to-switch link power, the switch power and the total power consumption. The plot starts with 5 switches (on x-axis), as the maximum size of a switch to support 500 MHz operation was 11x11 and the top and bottom layers needed 2 switches each (topology shown in Figure 6), as they have more than 10 cores each. Because the number of cores and the communication demand on each layer is different, we obtain different number of switches on each layer.

Since the area of each 3D layer is small (approximately 20 mm^2), the links are short and switch power has higher impact on the total power consumption. With increasing switch count, the switch power increases significantly, leading to higher power consumption. For this design, the NoC with 5 switches is most power optimal and the resulting floorplan is shown in Figure 7.

7.2 Comparisons with mesh

Custom topologies that match the application characteristics can result in large power-performance improvement when compared to the standard topologies, such as mesh and torus [17]. For this comparison we used the D_38_tvopd benchmark presented in Section 7.1 and five other benchmarks that model different traffic scenarios. We consider 3 benchmarks: D_36_4, D_36_6 and D_36_8 with 36 cores, each core communicating to 4, 6 and 8 other cores, respectively, modeling designs with multiple local memories. We also consider a benchmark with shared memory bottleneck communication (D_35_bot). For a larger design, we performed tests on the D_65_pipe which has 65 processing elements distributed on three layers and organized in a pipeline fashion. All of the benchmarks are mapped on to 3 layers in 3D.

We compared the custom topologies generated for the bench marks against an optimized mesh topology. For the optimized mesh each core is connected to a switch and only the necessary links among switches are opened. The results of the comparison between the best custom topology and the optimized mesh are presented in Figure 9. As can be seen from results, the topology synthesized by our method results in large power savings (38% on average) when compared to the optimized mesh topologies. The synthesized topologies also resulted in 24.5% reduction in average zero-load latency, when compared the optimized mesh based NoC.

7.3 Impact on inter-layer link constraint

Limiting the number of inter-layer links has a great impact on power consumption and average latency. Reducing the number of TSVs is desirable for improving the yield of a 3D design. However, a very tight constraint on the number of inter-layer links can

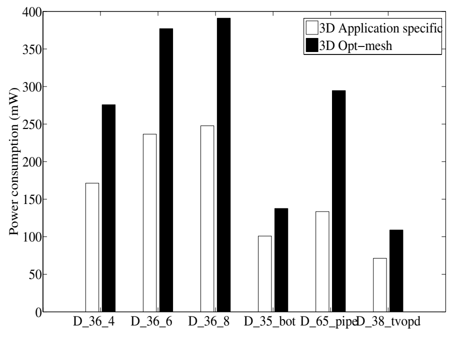

Figure 9: Comparisons with mesh

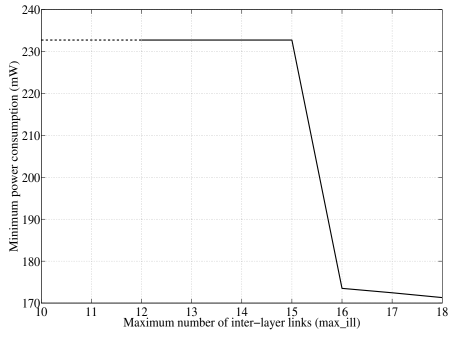

Figure 10: Impact of max_ill on power

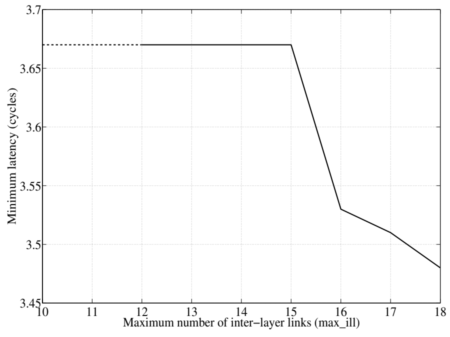

Figure 11: Impact of max_ill on latency

lead to a significant increase in power consumption. To see the impact of the constraint, we varied the value of max_ill constraint and performed topology synthesis for each value, for one of the benchmarks (D_36_4). The power and latency values for the different max_ill design points are shown in Figures 10 and 11. The dotted line in the figures represent points where the max_ill constraint was too tight to produce any feasible topologies. When there is a tight constraint on the inter-layer links, more flows are routed through existing inter-layer links instead of opening new ones. This leads to traversing more intermediate switches and higher switch activities, leading to higher latency and power consumption. Please note that our synthesis algorithm also allows the designers to perform such power, latency trade-offs for yield, early in the design cycle.

The synthesis algorithm explores a large solution space. However, thanks to the efficient heuristic methods presented, the entire topology design process completed in *few hours* for all the experiments, when run on a 2 GHz Linux workstation. Please note that the synthesis process is performed once at design time and this computational time incurred is negligible.

8. ACKNOWLEDGEMENTS

This work is supported by the Swiss National Science Foundation (FNS, Grant 20021-109450/1).

9. CONCLUSIONS

The use of *Networks on Chips (NoCs)* for communication in 3D chips has posed new opportunities and challenges for designers. One of the most important problems is to design the most power-performance efficient NoC topology that satisfies the application characteristics and 3D technology requirements. In this work, we presented a synthesis approach to solve this problem. We also presented methods to place switches optimally on the 3D floorplan, so that accurate power and delay numbers are obtained for the wires. Our detailed comparisons with regular 3D optimized mesh show that the custom 3D topologies lead to a large reduction in interconnect power consumption. In future, we plan to explore tuning the link data widths to meet the TSV constraints and to improve the yield of the 3D NoCs.

10. 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] K. Goossens et al., ”A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification”, DATE 2005.
[5] S. Stergiou et al., “×pipesLite: a Synthesis Oriented Design Library for Networks on Chips”, pp. 1188-1193, Proc. DATE 2005.
[6] J. Hu, R. Marculescu, ’Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures’, Proc. DATE, March 2003.
[7] S. Murali, G. De Micheli, “SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs”, Proc. DAC 2004.
[8] S. Murali, G. De Micheli, “Bandwidth Constrained Mapping of Cores onto NoC Architectures”, Proc. DATE 2004.
[9] D. Bertozzi et al., “NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip”, IEEE TPDS, Feb 2005.
[10] A.Pinto et al., “Efficient Synthesis of Networks on Chip”, ICCD 2003, pp. 146-150, Oct 2003.
[11] W.H.Ho, T.M.Pinkston, “A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns”, HPCA, 2003.
[12] T. Ahonen et al. ”Topology Optimization for Application Specific Networks on Chip”, Proc. SLIP 04.
[13] K. Srinivasan et al., “An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks”, Proc. ICCAD ’05.
[14] A. Hansson et al., “A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures”, Proc. CODES-ISSS, 2005.
[15] X.Zhu, S.Malik, “A Hierarchical Modeling Framework for On-Chip Communication Architectures”, ICCD 2002, pp. 663-671, Nov 2002.
[16] J. Xu et al., “A design methodology for application-specific networks-on-chip”, ACM TECS, May 2006.
[17] S. Murali et al., “Designing Application-Specific Networks on Chips with Floorplan Information”, pp. 355-362, ICCAD 2006.
[18] W. J. Dally, “Performance Analysis of k-ary n-cube Interconnection Networks”, IEEE Transactions on Computers, Vol. 39, No. 6, pp. 775-785, 1990.
[19] K. Banerjee et al., “3-D ICs: ANovel Chip Design for Deep-Submicrometer Interconnect Performance & SoC Integration”, Proc. of IEEE, 2001.
[20] B. Goplen and S. Sapatnekar, “Thermal Via Placement in 3D ICs”, Proc. Intl. Symposium on Physical Design, pp. 167, 2005.
[21] J. Cong et al., “A thermal-driven floorplanning algorithm for 3D ICs”, ICCAD 2004.
[22] W.-L. Hung et al., “Interconnect and thermal-aware floorplanning for 3D microprocessors”, Proc. ISQED, March 2006.
[23] S. K. Lim, “Physical Design for 3D System on Package”, IEEE Design & Test of Computers, vol. 22(6), pp. 532539, 2005.
[24] P. Zhou et al., ”3D-STAF: Scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits”, ICCAD 2007.
[25] N. Miyakawa et al., “ New Multi-Layer Stacking Technology and Trial Manufacture”, 3-D Architectures for Semiconductor Integration and Packaging, Oct 2007.
[26] R. Weerasekara et al., “Extending Systems-on-Chip to the Third Dimension: Performance, Cost and Technological Tradeoffs”, Proc. ICCAD, 2007.
[27] V. F. Pavlidis and E. G. Friedman, “Topologies for networks-onchip”, Proc. SOCC, 2006.
[28] B. Feero and P. P. Pande, “Performance evaluation for three-dimensional networks-on-chip”, Proc. ISVLSI, 2007.
[29] C. Addo-Quaye, “Thermal-Aware Mapping and Placement for 3-D NoC Designs”, Proc. SOCC, 2005.
[30] J. Kim et al., “A novel dimensionally-decomposed router for on-chip communication in 3d architectures”, ISCA, 2007.
[31] F. Li et al., “Design and Management of 3D Chip Multiprocessors Using Network-in-Memory”, ISCA, pp. 130-141, 2006.
[32] D. Park et al., “MIRA: A Multi-Layered On-Chip Interconnect Router Architecture”, Proc. ISCA, 2008.
[33] I. Loi, F. Angiolini, L. Benini, Supporting vertical links for 3D networks on chip: toward an automated design and analysis flow, Proc. Nano-Nets, 2007.
[34] S. Boyd and L. Vandenberghe, “Convex Optimization”, Cambridge University Press, 2004.
[35] Package available at: http://sourceforge.net/projects/lpsolve
[36] S. N. Adya, I. L. Markov, ”Fixed-outline Floorplanning : Enabling Hierarchical Design”, IEEE TVLSI, Dec 2003.