Application-Specific 3D Network-on-Chip Design Using Simulated Allocation *

Pingqiang Zhou
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455
pingqiang@umn.edu

Ping-Hung Yuh
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan
r91089@csie.ntu.edu.tw

Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455
sachin@umn.edu

Abstract— Three-dimensional (3D) silicon integration technologies have provided new opportunities for Network-on-Chip (NoC) architecture design in Systems-on-Chip (SoCs). In this paper, we consider the application-specific NoC architecture design problem in a 3D environment. We present an efficient floorplan-aware 3D NoC synthesis algorithm, based on simulated allocation, a stochastic method for traffic flow routing, and accurate power and delay models for NoC components. We demonstrate that this method finds greatly improved topologies for various design objectives such as NoC power (average savings of 34%), network latency (average reduction of 35%) and chip temperature (average reduction of 20%).

I. Introduction

Three dimensional (3D) integrated circuits, in which multiple tiers are stacked above each other and vertically interconnected using through-silicon vias (TSVs), are emerging as a promising technology for SoCs [1–4]. As compared to 2D designs, 3D circuits permit reduced latencies for critical interconnect structures, resulting in higher system throughput, performance, and power, and allow other benefits such as heterogeneous integration. All of these flexibilities enable the design of new high-performance System-on-Chip (SoC) structures that were previously thought to have prohibitive overheads. In spite of well-known challenges such as thermal bottlenecks (to which several solutions have been proposed), the benefits of 3D integration are considerable. In the context of intrachip communication, 3D technologies have created significant opportunities and challenges in the design of low latency, low power and high bandwidth interconnection networks.

In 2D SoCs choked by interconnect limitations, networks-on-chip (NoCs), composed of switches and links, have been proposed as a scalable solution to the global communication challenges: compared to previous architectures for on-chip communication such as bus-based and point-to-point networks, NoCs have been shown to provide better predictability, lower power consumption and greater scalability [5, 6]. 3D circuits enable the design of more complex and more highly interconnected systems: in this context, NoCs promise major benefits, but impose new constraints and limitations. 3D NoC design introduces new issues, such as the technology constraints on the number of TSVs that can be supported, problems related to optimally determining tier assignments and the placement of switches in 3D circuits, and accurate power and delay modeling issues for 3D interconnects.

This work addresses the problem of designing application-specific 3D NoC architectures for custom SoC designs, in conjunction with floorplanning. Specifically, our work determines both the NoC topology and the floorplan of the NoC switches and cores. We propose a synthesis method to find the best topology for the application, under different optimization objectives such as power and network latency, and determine the paths for traffic flows. We use a 3D thermally-aware floorplanner to assign the cores to different 3D tiers, while optimizing chip temperature, and find an initial floorplan for the cores on each tier. Given the positions of cores, we use a stochastic flow allocation method, Simulated Allocation (SAL), to route the traffic flows and build the topology for the application, initially using a simple strategy for determining the approximate locations of the switches. When the best topology is found, a fast floorplanner is applied to further optimize the positions of the added switches. Accurate power and delay models for switches and links are integrated into our algorithm.

Our approach has three significant features that together make it uniquely different from competing approaches: first, we use improved traffic flow routing using SAL that accommodates a realistic objective function that has components that are nonlinear and/or unavailable in closed form; second, we interleave floorplanning with NoC synthesis, using specific measures that encourage convergence by discouraging blocks from moving from their locations in each iteration; and third, we use an accurate NoC delay model that incorporates the effects of queueing delays and network contention.

Our algorithm is extremely flexible and is applicable both to 2D and 3D layouts, but we demonstrate that the use of 3D designs results in significantly reduced NoC power and latency, when compared to optimal 2D implementations.

II. Contributions of our Work

There has been a great deal of prior work on NoCs alone and on 2D and 3D layout alone, but less on integrating the two. In the area of designing NoC architectures for 3D ICs, most of the literature has focussed on regular 3D NoC topologies such as meshes [7–11], which are appropriate for regular 3D designs [12, 13]. However, most modern SoC architectures consist of heterogenous cores such as CPU or DSP modules, video processors, and embedded memory blocks, and the traffic requirements among the cores can vary widely. Therefore, regular topologies such as meshes may have significant area and power overhead [14, 15], and tuning the topology for application-specific solutions can provide immense benefits.

The synthesis of an application-specific NoC topology includes finding the optimal number and size of switches, establishing the connectivity between the switches and with the cores, and finding deadlock-free routing paths for all the traffic flows. For 2D systems, the problem of designing application-specific NoC topologies has been explored by several researchers [16–20]. Srinivasan *et al.* [17] present a three-phase NoC synthesis technique consisting of sequential steps that

*This work was supported in part by the SRC under contract 2008-TJ-1819.

floorplan the cores, next perform core-to-router mapping, and then generate the network topology. In [19], Murali *et al.* present an NoC synthesis method that incorporates the floorplanning process to estimate link power consumption and detect timing violations. Several topologies, each with a different number of switches, are explored, starting from one where all the cores are connected to one switch, to one where each core is connected to a separate switch. The traffic flows are ordered so that larger flows are routed first.

In the 3D domain, Yan *et al.* [14] present an application-specific 3D NoC synthesis algorithm that is based on a rip-up-and-reroute procedure for routing flows, where the traffic flows are ordered in the order of increasing rate requirements so that smaller flows are routed first, followed by a router merging procedure. Murali *et al.* [15] propose a 3D NoC topology synthesis algorithm, which is an extension to their previous 2D work [19], described above. The 3D NoC synthesis problem has been shown to be NP-hard in [21].

Our work is motivated by the following observations:

- The final results of application-specific NoC topology synthesis depend on the order in which the traffic flows are routed. In some cases, routing larger flows first provides better results [18, 19], while in others, routing the smaller flows first may yield better results [14]. A strategy is required to reduce the dependency of the results on flow ordering.

- In all of the works mentioned previously, the average hop count is used to approximate the average packet latency in NoCs. This ignores the queueing delays in switch ports and the contention among different packets for network resources such as switch ports and physical links, and cannot reflect the impact of physical core-to-switch or switch-to-switch distances on network latency. More accurate delay models that include the effects of queueing delay and network contention, and better delay metrics, should be applied for NoC performance analysis.

- The delays and power dissipation for physical links in NoCs are closely linked to the physical floorplan and topology of cores and switches. We show in Section VI that interleaving floorplanning and NoC topology synthesis process leads to superior results.

We address these important problems in application-specific NoC topology synthesis. Our solution to overcoming the ordering problem is based on the use of a multicommodity flow network formulation for the NoC synthesis problem: the advantage of such an approach is that it takes a global view of the problem and eliminates the problem, described above, of finding the best order in which to route the traffic flows. The multicommodity flow problem is a well-known approach for solving such problems, but has seen little use in NoC design, with a few exceptions. In [22, 23], Hu *et al.* propose a scheme to optimize NoC power consumption through topology exploration and wire style optimization, subject to the average communication latency constraints, but do not handle layout synthesis issues, and assume simple linear objective functions.

Our work utilizes a stochastic SAL approach to efficiently solve the multicommodity flow problem under a nonlinear objective function that can be evaluated by an oracle, but is hard to express in closed form. The SAL framework has previously been used to solve multicommodity flow problems in computer network design. We also use an accurate delay model for switches in NoCs which consider the queueing delay and network contention. Finally, our algorithm performs the floorplanning of cores/switches and NoC topology synthesis in an integrated iterative loop, attempting to find the optimal solution for the problem of application-specific NoC design.

III. Problem Inputs, Objectives, and Constraints

The input to our 3D NoC synthesis problem is a directed graph, called the core graph, $G(V, E, \lambda)$. Each node $v_i \in V$ represents a core (either a processing element or a memory unit) and each directed edge $e_{v_i,v_j} \in E$ denotes a traffic flow from source v_i to destination v_j. The bandwidth of traffic flow from core v_i to v_j is given by $\lambda(e_{v_i,v_j})$ in MB/s. In addition, NoC architectural parameters such as the NoC operating frequency, f , and the data link width, W, are also assumed to be provided as inputs. The operating frequency is usually specified by the design and data link width is dictated by the IP interface standards.

Our 3D NoC synthesis framework permits a variety of objectives and constraints, including considerations that are particularly important in 3D, such as power dissipation, temperature, and the number of TSVs, and NoC-specific issues such as minimizing the average/maximum network latency, limitations on the maximum bandwidth, as well as general factors such as the design area. In addition, the solution must be free of deadlocks, which can occur during routing flows due to cyclic dependencies of resources such as buffers. We use the turn prohibition algorithm presented in [24] to ensure that our topology is deadlock-free. The specific optimization objectives in each step of our approach are described in Section IV.

The output of our 3D NoC synthesis solution is an optimized custom deadlock-free network topology with pre-determined paths on the network to route the traffic flows in the core graph and the floorplan of the cores and switches in the NoC such that the constraints are satisfied.

IV. The Overall Design Flow

The design flow of our NoC synthesis algorithm is presented in Fig. 1. Given a given a core graph, we first obtain an initial floorplan of the cores using a thermally-aware floorplanner. This precedes the 3D NoC synthesis step, and is important because the core locations significantly influence the NoC architecture. Associating concrete core positions with the NoC synthesis step better enables it to account for link delays and power dissipation.

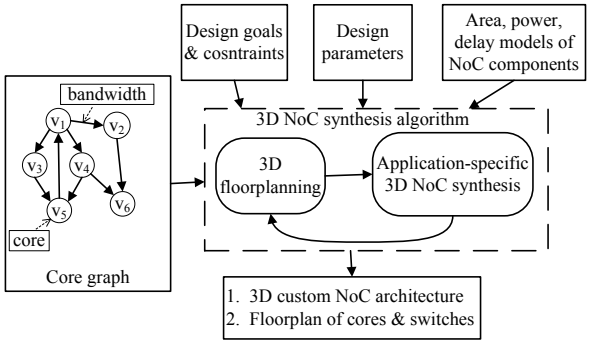

Fig. 1. Application-specific 3D NoC synthesis flow.

Our 3D NoC synthesis algorithm is performed on a directed routing graph $G'(V', E')$: V' is the vertex set, which is the union of core set V in the input core graph $G(V, E, \lambda)$ and the set of added switches, V_s. We assume that the maximum number of switches that can be used in each 3D tier l equals

to the number of cores in that tier, although it is easy to relax this restriction. The edge set E' is constructed as follows: we connect cores in a tier l only to the switches in the same tier l and adjacent tiers $l-1, l+1$ and the switches from all the 3D tiers form a complete graph. A custom NoC topology is a subgraph of the routing graph, G'.

The 3D NoC synthesis problem can be viewed as a *multi-commodity flow* (MCF) problem. For a core graph $G(V, E, \lambda)$ and a corresponding routing graph $G'(V', E')$ (corresponding to a flow network), let $c(u, v)$ be the capacity of edge $(u, v) \in E'$. The capacity $c(u, v)$ equals to the product of the operating frequency f and data link width W. Each commodity $K_i = (s_i, t_i, d_i), i = 1, \cdots, k$ corresponds to the weight (traffic flow) along edge e_{s_i,t_i} in the core graph from source s_i to destination t_i, and $d_i = \lambda(e_{s_i,t_i})$ is the demand for commodity i. Therefore, there are $k = |E|$ commodities in the core graph. Let the flow of commodity i along edge (u, v) be $f_i(u, v)$. Then the MCF problem is to find the optimal assignment of flow which satisfies the constraints:

$$\begin{aligned}
&\text{Capacity constraints:} && \textstyle\sum_{i=1}^{k} f_i(u,v) \le c(u,v)\\
&\text{Flow conservation:} && \textstyle\sum_{\omega \in V', u \neq s_i, t_i} f_i(u,\omega) = 0\\
& && \text{where} \quad \forall v, u \; f_i(u,v) = -f_i(v,u)\\
&\text{Demand satisfaction:} && \textstyle\sum_{\omega \in V'} f_i(s_i,\omega) = \sum_{\omega \in V'} f_i(\omega,t_i) = d_i
\end{aligned}$$

Superficially, this idea seems similar to [23], where an MCF formulation is proposed. However, that work is directed to 2D NoC synthesis with a single objective of minimizing NoC power, modeled as a linear function of the flow variables $f_i(u, v)$. The corresponding Linear Programming (LP) problem is solved using an approximation algorithm. Our more general formulation integrates more objectives and more accurate modeling for NoC components. In fact, most components of our objective function are nonlinear or, as in case of network latency, unavailable in closed form, rendering an LP-based approach impossible.

We choose to apply an SAL-based flow allocation approach that is particularly suitable for (see Section V-A for details) solving the MCF problems where the objective function is in such a form. The SAL procedure yields the NoC topology and the paths for all the traffic flows in the core graph. In this optimization, we assume single-path routing for each traffic flow in the core graph, but conceptually, the SAL method can easily be extended to deal with the multipath routing problem.

After the 3D NoC synthesis step, the actual switches and links in the synthesized 3D NoC architecture are fed back to the floorplanner to update the floorplan of the cores and used switches, and the refined floorplan information is used to obtain more accurate power and delay estimates. The process continues iteratively: with the refined floorplan, a new SAL based 3D NoC synthesis procedure is invoked to find a better synthesis solution, and so on.

The specific optimization objectives used in various steps of our approach are as follows:

- For NoC topology construction, we optimize a linear combination of the network power, average network latency and TSV count, with constraints on bandwidth.
- For the initial floorplanning step (Section V-C), we optimize a linear combination of chip temperature and weighted inter-core distance.
- For subsequent steps that floorplan the cores and switches, we optimize a linear combination of design area, link power, link delay and chip temperature.

V. Technical details

In this section, we present the major elements in our 3D NoC synthesis algorithm.

A. Simulated Allocation Algorithm

Simulation Allocation (SAL) [25, 26] is a stochastic framework for finding near-optimal solutions for the multicommodity traffic flow problems. It has been shown to be simpler, but often faster and more efficient, than other stochastic algorithms such as simulated annealing and evolutionary algorithms. The details of the SAL framework used in our work are described in Algorithm 1.

In the core graph $G(V, E, \lambda)$, let

- P_i be the number of available paths for traffic demand $K_i = (s_i, t_i, d_i)$,
- x_{ip} be the amount of traffic flow realizing the traffic $K_i = (s_i, t_i, d_i)$ allocated to path p in routing graph G',
- $x = \{x_{ip} : i = 1, 2, \cdots, k, p = 1, 2, \cdots, P_i\}$ be the allocation state,
- $|x| = \sum_i \sum_p x_{ip}$ be the total allocated traffic flow, and
- $H = \sum_i d_i$ be the total amount of traffic flow.

Note that since we consider single-path routing for each commodity, at most k paths, one per commodity, will have nonzero flows. Therefore, even though the number of paths can be exponentially large, it is never necessary to enumerate P_i; storing the allocation state x does not impose a significant memory overhead.

Algorithm 1 Simulated Allocation (SAL) (adapted from [26])

```
1:  n = 0; counter = 0; x = 0; F^best = +∞;
2:  repeat
3:    if random(0,1) < q(|x|) then
4:      allocate(x);
5:    else
6:      disconnect(x);
7:    end if
8:    if |x| = H then
9:      n = n + 1;
10:     counter = counter + 1;
11:     if F(x) < F^best then
12:       F^best = F(x);
13:       x^best = x;
14:       counter = 0;
15:     end if
16:   end if
17: until n = N or counter = M
```

The SAL algorithm may start with a given partial allocation state x_0 or the with the zero state ($x_{ip} \equiv 0$). In each step, it chooses, with state-dependent probability $q(|x|)$, between *allocation*(x), i.e., adding the traffic flow for one non-allocated commodity to the current state x, and *disconnect*(x), i.e., removing the traffic flow for one allocated commodity from current state x. After a sequence of such moves, from time to time, the algorithm will reach a full allocation state, yielding a feasible solution for the considered problem. The procedure terminates when the number of visited full allocation states reaches a user-specified limit N or no better solution is found within M visited full allocation states.

Procedure *allocation*(x) selects one currently non-allocated commodity, $K_i = (s_i, t_i, d_i)$, at random and allocates it to one

of the allowable paths that have enough residual capacity to support K_i. The path for allocating K_i is chosen to be the minimum cost path p with respect to the cost function for the NoC topology construction step. Then we add flow $x_{ip} = d_i$ to the current state x and reduce the capacities of the links on the selected path p in the routing graph by d_i. When routing commodity K_i, several new links and switches from the routing graph may be added to the NoC topology and the sizes of the switches on the path p may need to be adjusted accordingly.

Procedure *disconnect*(x) selects an allocated commodity $K_i = (s_i, t_i, d_i)$ at random and removes the corresponding flow x_{ip} from current state x. We then increase the capacities of the links on the path p by d_i. If some links/switches become unused in the resulting solution, such links/switches are also removed from the NoC topology. The sizes of the switches on the path p may need to be adjusted accordingly.

Function $q(\gamma)$, defined for $0 \leq \gamma \leq H$, has the properties:

$$\begin{cases} q(0) = 1 \\ q(H) = 0 \\ \frac{1}{2} < q(\gamma) \leq 1, \quad 0 < \gamma < H \end{cases}$$

According to [26], if

$$q(|x|) = q_0 > \frac{1}{2} \quad \texttt{for} \quad 0 < \gamma < H$$

then the expected average number of steps (allocations and disconnections) required to reach a full allocation state starting from state x is no greater than

$$(H - |x|)/(2q_0 - 1)$$

For instance, if $q_0 = \frac{2}{3}$ then a full allocation state will be reached from the zero allocation state in only $3H$ steps.

B. Analytical Switch Modeling for NoCs

Accurate delay models for switches are required as an input to our 3D NoC synthesis problem. We utilize the analytical delay model presented in [27], which includes the effects of queueing delay and network contention. The model considers first-come-first-serve input buffered switches and targets wormhole flow control under deterministic routing algorithms.

Let S be the packet size and H the service time for a header flit passing through switch i. The service time of a packet passing through switch i, excluding the queueing delay, is

$$T = H + \frac{S - W}{f \cdot W} \tag{1}$$

where W is the data link width and f is the operating frequency. For switch i, let

- p be the total number of ports.
- λ_j be the traffic arrival rate at port j.
- N_j be the average number of packets in the buffers of input port j, and $N = [N_1, N_2, \ldots, N_p]^T$.
- c_{jk} be the probability that packets of input ports j and k compete for the same output port, and C_j be the row vector $C_j = [c_{j1}, c_{j2}, \ldots, c_{jp}]$.
- R be the residual service time seen by the incoming packets, defined as follows: if another packet n is being served when packet m arrives, then R is the remaining time before packet n leaves the switch.

Then we can write the *equilibrium condition* for the switch as:

$$(I - T\Lambda C)N = \Lambda \bar{R} \tag{2}$$

where $\Lambda = diag\{\lambda_1, \lambda_2, \ldots, \lambda_p\}$, $C = [C_1, C_2, \ldots, C_p]^T$, $\bar{R} = ([R, R, \ldots, R]_{1\times p})^T$.

The switch model described by Equation (2) provides a closed form expression for the average number of packets at each input port of the switch i, given the traffic arrival rate (Λ), the packet contention probabilities (C), switch design specifications (H, W) and packet size S. For further details, the reader is referred to [27].

C. 3D Floorplanning

An initial step of thermally-aware floorplanning is applied to assign the cores into 3D tiers under thermal considerations, and to optimize the positions of the cores so that highly communicating cores are placed close to each other. In our implementation, we use the 3D thermally-aware floorplanner tool in [28] based on B*-tree floorplan model. The floorplanner uses a built-in thermal analysis technique based on the HS3D [28] tool. Of course, any other similar tools can also be integrated into our program.

For each edge e_{v_i,v_j} which connects two cores, v_i and v_j, the edge weight of e_{v_i,v_j} is set to be the product of edge bandwidth $\lambda(e_{v_i,v_j})$ and the distance d_{ij} between v_i and v_j. Our cost function is a weighted sum of the chip temperature and the sum of these edge weights. Therefore, we use the floorplanner to find a good initial floorplan of cores that favors our next step of 3D NoC synthesis.

During initial floorplanning, we only consider the communicating cores, since no switches have been introduced at this time. Once a full allocation of traffic flows is found, the topology of the NoC is determined, including the switches that are used to route traffic. We then invoke the floorplanner to find a refined floorplan of cores and NoC switches, under an objective function that is a linear combination of design area, link power, link delay and chip temperature.

D. Switch Location Estimation and Path Cost Estimation

When routing a flow from source s to destination d, our objective is to find a minimum cost path in the routing graph. While the initial solution considers the physical locations of only the cores, as flow allocation proceeds, new switches will be included in the NoC topology and their physical positions must be estimated to compute the link power and delay.

We estimate the switch locations in the following way: for a newly added switch i, the switch is initially placed at the centroid of the source and destination nodes of switch i in the routing graph. Given these initial estimates of the positions of the newly added switches, we apply Dijkstra's shortest path algorithm on the routing graph to find the minimum cost path for the traffic flow, which is required by *allocation*(x). When the 3D NoC synthesis step is complete, we feed the actual switches and links in the synthesized architecture to the floorplanner to update the switch locations, for more accurate power and delay estimation. Since the floorplanner is stochastic, it is possible for the new floorplan to be vastly different from the one that was used to generate the NoC topology, negating the assumptions used to build the topology. To avoid this, we add a penalty to the objective function of the floorplanner to ensure that the blocks do not move far away from their initial locations, and optimize the precise locations of the switches, which were initially placed in (possibly illegal) centroid locations.

VI. Experimental Results

A. Experimental Setup

We have implemented *3D-SAL-FP*, our SAL-based 3D NoC synthesis algorithm with floorplan feedback, in C++. All experiments were conducted on an Intel Pentium 4 CPU 3.20GHz machine with 2G memory running Linux.

The design parameters are set as: 900MHz clock frequency, 512-bit packets, 4-flit buffers and 32-bit flits. We use Orion [29] to estimate the power dissipation of the switches. The link power and delay are modeled based on the equations from Pavlidis *et al.* [8], and the delay of switches are estimated using the model described in Section V-B. All switches and links are evaluated under a 45nm technology.

Several parameters affect the efficiency and performance of the SAL algorithm (Section V-A). In our implementation, we found that a constant function $q(\gamma) = q0$, where $q0 \in [0.75, 0.90]$, can produce good solutions. The user-specified iteration limit N is empirically set to be three times of k, the number of commodities in the core graph, and M is set to be 50. We find that the best solutions are often obtained within k full visited allocation states for all the benchmarks.

B. Impact of each strategy applied in our algorithm

Our algorithm *3D-SAL-FP* improves upon the previous algorithms in [14, 15] by: 1) using a more sophisticated traffic flow routing algorithm (SAL), 2) adding a feedback loop of floorplanning and NoC synthesis to refine the NoC architecture, 3) using a more accurate switch delay model including the effects of queueing delay and network contention. To show the separate impact of these techniques on the NoC design, we have implemented three other 3D NoC synthesis algorithms.

The first algorithm, based on the work by Murali *et al.* [15], has two stages: 3D NoC synthesis and floorplanning of the synthesized NoC architecture. At the 3D NoC synthesis stage, simple delay model (average hop count) is used to approximate the average network latency and the traffic flows are routed in fixed order (in the order of decreasing flow rate). In the next stage, we move on to find the floorplan of cores and used switches in the NoC architecture. We refer to this algorithm as the *Baseline1* algorithm. The second algorithm differs from *Baseline1* in that it applies an improved traffic flow routing strategy (SAL) in the 3D NoC synthesis stage. We refer to this algorithm as the *Baseline2* algorithm. The third algorithm improves upon *Baseline2* by feeding back the results of floorplanning stage to refine the NoC synthesis. The process continues iteratively: after the 3D NoC synthesis step, the actual switches and links in the synthesized solution is fed back to the floorplanner to refine the floorplan of the cores and used switches; with the refined floorplan, a new NoC synthesis procedure is invoked to find a better synthesis solution, and so on. We refer to this algorithm as the *Baseline3* algorithm. Our *3D-SAL-FP* differs from *Baseline3* in that it use the accurate switch delay model (described in Section V-B) to incorporate the queueing delay and network contention issues.

We then applied these four algorithms to design 3D application specific NoC topologies. We compared these algorithms on both a set of existing published benchmarks and several large synthetic 3D benchmarks. Since large standard benchmarks are not available, we use the method proposed in [14] to generate the large synthetic 3D benchmarks. This method is based on the NoC-centric bandwidth version of Rent's rule proposed by Greenfield *et al.* [30]. For the small published benchmarks, two 3D tiers are used, where each tier contains one layer of devices and multiple layers of interconnect. For all of the large synthetic benchmarks, four 3D tiers are used.

The corresponding results are shown in Tables I and II. For each algorithm, we report the following: the network power (in mW, including switch power and link power), the average network latency (in ns, evaluated by the accurate delay model), the number of TSVs and the maximum chip temperature (in $^{\circ}C$).

We can observe that using the improved traffic flow routing algorithm, the *Baseline2* algorithm outperforms *Baseline1*, achieving 23% power saving for the published benchmarks, 10% reduction in chip temperature and better network performance. The corresponding numbers for synthetic benchmarks is 21% in power saving and 9% in chip temperature reduction. Furthermore, *Baseline3* uses the feedback from the floorplanning step to improve upon *Baseline2*, and shows 34% reduction in the power dissipation for both published and synthetic benchmarks, about 20% reduction in chip temperature and 10% reduction in average network latency. Finally, with more accurate delay model, *3D-SAL-FP* improves upon *Baseline3*, with 26% reduction in average network latency for published benchmarks and 44% for the synthetic benchmarks. Since the objective function for these algorithms is a linear combination of several metrics, the use of different sets of weighting factors can result in different Pareto-optimal solutions. For a fair comparison, we have used identical weighting factors for all four algorithms discussed here. In the solutions shown here, *3D-SAL-FP* performs significantly better than *Baseline3* in reducing the delay, and is slightly better on average (and sometimes worse on specific examples) in terms of power and temperature. By altering the weights, other tradeoff points may be identified.

C. Delay and Power Reduction Potential in 3D NoCs

In this section, we further investigate the benefits that 3D circuits can bring to the NoC architecture design. The benchmark B3, with 69 cores and 136 flows, was selected and our *3D-SAL-FP* algorithm was applied to synthesize this benchmark with different numbers of 3D tiers, from 1 to 4. *The 1-tier case is the design that uses conventional 2D technology.* The results are shown in Table III. For each case, we list the following results: the design footprint, the network power, the maximum path length, the maximum total link delay, the maximum network latency, the average network latency, the total number of TSVs, the maximum chip temperature and the CPU time.

As we can see from Table III, as the number of 3D tiers increases, the footprint size continues to decrease, together with the maximum length of the path to route the packets. The reduced path length further brings down the maximum link delay and the total link power at the cost of increased number of TSVs and higher chip temperature. The switch power depends on the specific network traffic and the sizes of the switches determined by the customized NoC architectures and does not change much as the number of 3D tiers increases.

VII. Conclusion

We have proposed an efficient algorithm, *3D-SAL-FP*, to synthesize application-specific 3D NoC architectures. Our algorithm utilizes a stochastic approach called simulated allocation to reduce the dependency of NoC design results on flow ordering. We also use accurate delay model for switches in NoCs which consider the queueing delay and network contention. Finally, our algorithm performs the floorplanning of cores/switches and NoC topology synthesis in an integrated iterative loop, attempting to find the optimal solution for the problem of application-specific NoC design. Experimental re-

TABLE I

COMPARISON OF THREE ALGORITHMS ON SEVERAL SMALL PUBLISHED BENCHMARKS

Ben	Cores	Flows	*Baseline1*						*Baseline2*						*Baseline3*						*3D-SAL-FP*					
			Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}
			Switch	Link	Total				Switch	Link	Total				Switch	Link	Total				Switch	Link	Total			
PIP	8	8	54	5	59	3.8	8	66.4	44	4	48	3.7	6	60.6	39	4	43	3.6	7	58.2	38	4	42	3.2	6	55.1
MWD	12	13	94	8	102	4.1	10	72.8	74	7	81	4.0	9	66.5	65	6	71	3.8	12	62.5	65	6	71	3.5	9	62.3
VOPD	12	15	99	11	110	7.3	14	67.8	82	10	92	7.2	7	64.5	73	10	83	6.9	7	59.4	72	9	81	5.1	9	50.9
MEPG4	12	26	165	15	180	10.3	14	70.8	108	15	123	10.1	13	64.7	88	12	100	9.0	14	58.2	90	13	103	6.3	14	59.6
IMP	27	96	612	90	702	9.4	42	78.8	413	99	512	8.0	44	65.2	335	87	422	7.8	42	55.7	346	79	425	6.4	40	56.9
					1	1		1			0.77	0.95		0.90			0.66	0.91		0.82			0.66	0.74		0.80

TABLE II

COMPARISON OF THREE ALGORITHMS ON LARGE SYNTHETIC BENCHMARKS

Ben	Cores	Flows	*Baseline1*						*Baseline2*						*Baseline3*						*3D-SAL-FP*					
			Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}	Power			Delay	# of TSVs	T_{max}
			Switch	Link	Total				Switch	Link	Total				Switch	Link	Total				Switch	Link	Total			
B1	56	196	1033	291	1324	16.3	119	157.8	956	302	1258	16.0	132	145.4	808	209	1017	15.0	139	128.3	785	214	999	6.7	132	133.2
B2	80	96	783	128	911	7.9	117	133.5	561	118	689	7.9	116	119.6	490	99	589	7.6	124	107.1	494	96	590	4.6	126	107.5
B3	69	136	866	210	1076	13.1	122	150.6	494	243	737	12.0	95	134.4	509	165	674	11.5	105	118.2	504	141	645	9.4	116	118.0
B4	114	396	3128	827	3955	15.9	196	166.4	2230	888	3118	15.5	214	151.6	1826	643	2469	13.9	192	128.6	1721	632	2353	7.3	208	137.0
B5	124	266	1827	848	2675	13.9	254	135.9	1517	686	2203	11.8	264	125.2	1352	432	1784	11.4	256	104.4	1338	468	1806	9.1	241	102.7
					1	1		1			0.79	0.94		0.91			0.66	0.89		0.79			0.65	0.56		0.80

TABLE III

COMPARISON OF THE IMPACT OF DIFFERENT NUMBERS OF 3D TIERS ON NOC ARCHITECTURE DESIGN FOR BENCHMARK B3

Layers	Footprint	Network Power			Maximum Path Length	Maximum Link Delay	Maximum Network Latency	Average Network Latency	# of TSVs	T_{max}	Time
		Switch	Link	Total							
	(mm^2)	(mW)	(mW)	(mW)	(mm)	(ns)	(ns)	(ns)		$(^\circ C)$	(s)
1	216.8	510.5	288.4	798.9	22.1	6.45	14.40	12.42	0	43.8	85.8
2	110.3	505.8	189.2	695.0	17.0	4.95	12.28	9.56	86	63.7	83.9
3	72.0	510.7	164.8	675.5	11.9	3.50	11.51	9.49	94	96.2	87.3
4	56.1	504.8	141.0	645.8	9.2	2.68	11.32	9.44	116	118.0	87.4

sults on a set of benchmarks show that our algorithm can produce greatly improved solutions.

ACKNOWLEDGMENTS

The authors would like to thank professor Yuan Xie in Pennsylvania State University for providing the 3D floorplanner.

REFERENCES

[1] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat. 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems. *Proc. IEEE*, 89(5), 2001.

[2] L. Xue, C. C. Liu, H.-S. Kim, S. K. Kim, and S. Tiwari. Three-dimensional integration: technology, use, and issues for mixed-signal applications. *IEEE Transactions on Electron Devices*, 50(3):601–609, Mar. 2003.

[3] W. R. Davis et al. Demystifying 3D ICs: The pros and cons of going vertical. *IEEE Design & Test of Computers*, 22(6):498–510, Nov.-Dec. 2005.

[4] S. K. Lim. Physical design for 3D system on package. *IEEE Design & Test of Computers*, 22(6):532–539, Nov.-Dec. 2005.

[5] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35(1):70–78, 2002.

[6] W. J. Dally and B. Towles. Route packets, not wires: On-chip inteconnection networks. In *Proc. DAC*, pages 684–689, 2001.

[7] C. Addo-Quaye. Thermal-aware mapping and placement for 3-D NoC designs. In *IEEE International SOC Conference*, pages 25–28, 2005.

[8] V. F. Pavlidis and E. G. Friedman. 3-D topologies for networks-on-chip. *IEEE Transactions on Very Large Scale Integration Systems*, 15(10):1081–1090, 2007.

[9] B. Feero and P. P. Pande. Performance evaluation for three-dimensional networks-on-chip. In *Proc. ISVLSI*, pages 305–310, 2007.

[10] J. Kim et al. A novel dimensionally-decomposed router for on-chip communication in 3D architectures. In *Proc. ISCA*, pages 138–149, 2007.

[11] H. Matsutani, M. Koibuchi, and H. Amano. Tightly-coupled multi-layer topologies for 3-D NoCs. In *ICPP*, pages 75–75, 2007.

[12] F. Li et al. Design and management of 3D chip multiprocessors using network-in-memory. In *Proc. ISCA*, pages 130–141, 2006.

[13] P. Morrow et al. Design and fabrication of 3D microprocessors. In *Materials Research Society Symposium*, 2007.

[14] S. Yan and B. Lin. Design of application-specific 3D networks-on-chip architectures. In *Proc. ICCD*, pages 142–149, 2008.

[15] S. Murali, C. Seiculescu, L. Benini, and G. D. Micheli. Synthesis of networks on chips for 3D systems on chips. In *Proc. ASPDAC*, pages 242–247, 2009.

[16] T. Ahonen, H. Bin, and J. Nurmi. Topology optimization for application-specific networks-on-chip. In *Proc. International Workshop on System Level Interconnect Prediction*, pages 53–60, 2004.

[17] 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*, pages 231–237, 2005.

[18] A. Hansson, K. Goossens, and A. Rădulescu. A unified approach to constrained mapping and routing on network-on-chip architectures. In *Proc. CODES-ISSS*, pages 75–80, 2005.

[19] S. Murali et al. Designing application-specific networks on chips with floorplan information. In *Proc. ICCAD*, pages 355–362, 2006.

[20] S. Yan and B. Lin. Application-specific network-on-chip architecture synthesis based on set partitions and Steiner trees. In *Proc. ASPDAC*, pages 277–282, 2008.

[21] C. Seiculescu, S. Murali, L. Benini, and G. D. Micheli. SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chip. In *Proc. DATE*, 2009.

[22] Y. Hu, H. Chen, Y. Zhu, A. A. Chien, and C.-K. Cheng. Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimizations. In *Proc. ICCD*, pages 111–118, 2005.

[23] Y. Hu, Y. Zhu, H. Chen, R. Graham, and C.-K. Cheng. Communication latency aware low power NoC synthesis. In *Proc. DAC*, pages 574–579, 2006.

[24] D. S. Mark, M. Karpovsky, and L. Zakrevski. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transactions on Networking*, 11(3):411–421, 2003.

[25] M. Pióro and P. Gajowniczek. Solving multi commodity integral flow problems by simulated allocation. *Telecommunication Systems*, 7(1-3):17–28, 1997.

[26] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[27] U. Y. Ogras and R. Marculescu. Analytical router modeling for networks-on-chip performance analysis. In *Proc. DATE*, pages 1096 – 1101, 2007.

[28] W.-L. Hung, G. M. Link, Yuan Xie, N. Vijaykrishnan, and M. J. Irwin. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *Proc. ISQED*, pages 98–104, 2006.

[29] H. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A power-performance simulator for interconnection networks. In *MICRO 35*, pages 294–305, 2002.

[30] D. Greenfield, A. Banerjee, J.-G. Lee, and S. Moore. Implications of Rent's Rule for NoC design and its fault-tolerance. In *Proc. NOCS*, pages 283–294, 2007.