Optimized 3D Network-on-Chip Design Using Simulated Allocation

PINGQIANG ZHOU, University of Minnesota
PING-HUNG YUH, National Taiwan University
SACHIN S. SAPATNEKAR, University of Minnesota

12

Three-dimensional (3D) silicon integration technologies have provided new opportunities for Network-on-Chip (NoC) architecture design in Systems-on-Chip (SoCs). In this article, 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 (SAL), a stochastic method for traffic flow routing, and accurate power and delay models for NoC components. We demonstrate that this method finds greatly improved solutions compared to a baseline algorithm reflecting prior work. To evaluate the SAL method, we compare its performance with the widely used simulated annealing (SA) method and show that SAL is much faster than SA for this application, while providing solutions of very similar quality. We then extend the approach from a single-path routing to a multipath routing scheme and explore the trade-off between power consumption and runtime for these two schemes. Finally, we study the impact of various factors on the network performance in 3D NoCs, including the TSV count and the number of 3D tiers. Our studies show that link power and delay can be significantly improved when moving from a 2D to a 3D implementation, but the improvement flattens out as the number of 3D tiers goes beyond a certain point.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: 3D, NoC, application-specific, simulated allocation

ACM Reference Format:
Zhou, P., Yuh, P.-H., and Sapatnekar, S. S. 2012. Optimized 3D network-on-chip design using simulated allocation. ACM Trans. Des. Autom. Electron. Syst. 17, 2, Article 12 (April 2012), 19 pages.
DOI = 10.1145/2159542.2159544 http://doi.acm.org/10.1145/2159542.2159544

1. 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 [Banerjee et al. 2001; Davis et al. 2005; Lim 2005; Xue et al. 2003]. As compared to 2D designs, 3D circuits permit reduced latencies for critical interconnect structures, resulting in higher system throughput, performance, and power, and allowing 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-

This work was supported in part by the SRC under contract 2008-TJ-1819.
Some initial results of this article are published in *Proceedings of the Annual Asia and South Pacific Design Automation Conference* [Zhou et al. 2010].
Authors' addresses: P. Zhou and S. S. Sapatnekar, Electrical and Computer Engineering Department, University of Minnesota; email: pingqiang@umn.edu; P.-H. Yuh, Computer Science and Information Engineering Department, National Taiwan University.

DOI 10.1145/2159542.2159544 http://doi.acm.org/10.1145/2159542.2159544

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 [Benini and Micheli 2002; Dally and Towles 2001].

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. Compared to wire interconnects, NoCs not only enable scalable and parallel communication within and across 3D tiers but also reduce the number of TSVs for vertical interconnects. However, 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. Third, we use an accurate NoC delay model that incorporates the effects of queueing delays and network contention.

NoC synthesis can be based on either single-path or multipath routing: single-path routing can guarantee in-order delivery of packets and is much simpler to implement; multipath routing can exploit path diversity to evenly distribute the traffic across the network and to relieve traffic congestion, but the packets are sent in out-of-order fashion and re-ordering mechanisms are needed at the reconvergent nodes [Murali et al. 2006]. We demonstrate that our SAL approach can work with either single-path or multipath routing schemes.

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

2. 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 focused on regular 3D NoC topologies, such as meshes [Addo-Quaye 2005; Feero and Pande 2007; Kim et al. 2007; Matsutani et al. 2007; Pavlidis and Friedman 2007], which are appropriate for regular 3D designs [Li et al. 2006; Morrow et al. 2007]. 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 [Murali et al. 2009; Yan and Lin 2008b], 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 [Ahonen et al. 2004; Hansson et al. 2005; Murali et al. 2006; Srinivasan et al. 2005; Yan and Lin 2008a]. Srinivasan et al. [2005] present a three-phase NoC synthesis technique consisting of sequential steps that floorplan the cores, then perform core-to-router mapping, and finally generate the network topology. Murali et al. [2006] 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 and Lin [2008b] present an application-specific 3D NoC synthesis algorithm that is based on a rip-up-and-reroute procedure for routing flows in which 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. [2009] propose a 3D NoC topology synthesis algorithm, which is an extension to their previous 2D work [Murali et al. 2006], previously described. The 3D NoC synthesis problem has been shown to be NP-hard in Seiculescu et al. [2009].

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 [Hansson et al. 2005; Murali et al. 2006], while in others, routing the smaller flows first may yield better results [Yan and Lin 2008b]. 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 6 that interleaving floorplanning and the NoC topology synthesis process lead 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, previously described, 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. Hu et al. [2005, 2006] propose a scheme to optimize NoC power consumption through topology exploration and wire-style optimization, subject to the average communication latency constraints. Their scheme, however, does not handle layout synthesis issues and assumes 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 considers 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.

Our work targets SoC applications, which have static or semi-static traffic characteristics in the network. Dynamic traffic behaviors are observed in the many-core processors, but such applications are beyond the scope of this article. In the context of synthesizing application-specific 3D NoC architectures for custom SoC designs, this article makes the following contributions.

— 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. The effects of these strategies have been verified by the experiment results.
— We perform a comparative study between single-path and multipath routing schemes in the SAL framework. Simulation results show that a trade-off exists between single-path and multipath routing systems in terms of network power consumption and the efficiency of solving multicommodity flow problems.
— We also compare our stochastic SAL approach with simulated annealing (SA). Our results show that SAL is much faster than SA in finding approximately the same-quality solutions.
— After that, we present the impact of TSV count on the network performance in 3D NoCs. Our results show that within a certain extent, TSV count can be effectively reduced with a mild penalty on the network performance.
— Finally, we investigate the impact of 3D integration on the NoC architecture design. Our studies show that link power and delay can be largely improved when moving to 3D implementation, at the cost of the TSV area and chip temperature. We also observe that the improvement on link delay and power flattens out as the number of 3D tiers goes beyond a certain point.

3. PROBLEM INPUTS, OBJECTIVES, AND CONSTRAINTS

The input to our application-specific 3D NoC synthesis problem is a directed graph $G(V, E, \lambda)$, called the core graph. 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 the 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

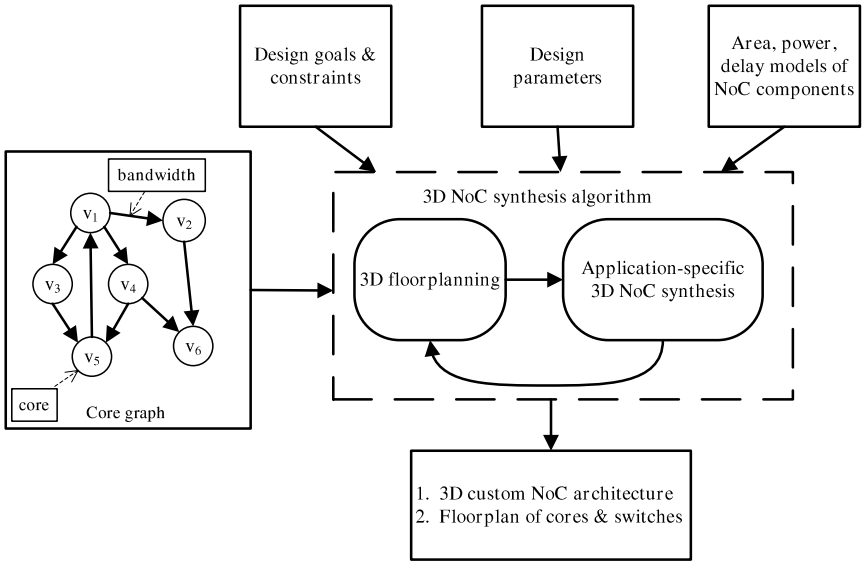


Fig. 1. Application-specific 3D NoC synthesis flow.

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 Mark et al. [2003] to ensure that our topology is deadlock-free. The specific optimization objectives in each step of our approach are described in Section 4.

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

4. THE OVERALL DESIGN FLOW

The design flow of our NoC synthesis algorithm is presented in Figure 1. 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.

Our 3D NoC synthesis algorithm is performed on a directed routing graph $G'(V', E')$, where 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 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 *multicommodity 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 following constraints.

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

Superficially, this idea seems similar to that of Hu et al. [2006], 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 the 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 solving the MCF problems where the objective function is in such a form (see Section 5.1 for details). The SAL procedure yields the NoC topology and the paths for all the traffic flows in the core graph. In our article, we first present the SAL approach using single-path routing and then show how to extend it to deal with the multipath routing problem in the experimental section.

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 the initial floorplanning step, we optimize a linear combination of chip temperature and weighted intercore distance (Section 5.4).

$$\texttt{Objective cost} = w_1 * \texttt{temperature} + w_2 * \texttt{inter-core distance}, \tag{1}$$

where $w_1 = 1$ and $w_2 = 5$ are default weights.

— For NoC topology construction, we optimize a linear combination of the network power, average network latency, and TSV count, with constraints on link bandwidth.

$$\texttt{Objective cost} = w_1 * \texttt{power} + w_2 * \texttt{latency} + w_3 * \texttt{TSV count}, \tag{2}$$

where $w_1 = 10$, $w_2 = 5$, and $w_3 = 3$ are default weights.

— For subsequent steps that floorplan the cores and switches, we optimize a linear combination of design area, link power, link delay, and chip temperature.

$$\texttt{Objective cost} = w_1 * \texttt{area} + w_2 * \texttt{power} + w_3 * \texttt{delay} + w_4 * \texttt{temperature}, \tag{3}$$

where $w_1 = 10$, $w_2 = 5$, $w_3 = 3$, and $w_4 = 1$ are default weights.

In Equations (1)–(3), we normalize the metrics, such as power and latency, using their initial numbers from a preliminary solution of the NoC synthesis. In a practical

setting, the weights of these metrics in each cost function are user-specified and can be chosen depending on the emphasis that the user wishes to place on each of these metrics.

5. TECHNICAL DETAILS

In this section, we present the major elements in our 3D NoC synthesis algorithm. We first introduce the SAL algorithm—the approach to synthesizing the NoC topology—in Section 5.1. In Sections 5.2 and 5.3, we present the delay model and the method for estimating the path cost used in our SAL algorithm. Finally, we introduce the 3D floorplanner for the initial floorplanning step and subsequent floorplan refinement of cores and NoC switches.

5.1. Simulated Allocation Algorithm

Simulation Allocation (SAL) [Pióro and Gajowniczek 1997; Pióro and Medhi 2004] is a stochastic approach for finding near-optimal solutions for the multicommodity traffic flow problems in computer network design. It has been shown to be simpler, but often faster and more efficient, than other stochastic algorithms, such as simulated annealing and evolutionary algorithms. We adopt the SAL framework from Pióro and Medhi [2004] but adapt it to solve the 3D NoC synthesis problem in our work. The details of the SAL algorithm 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;
— $H = \sum_i d_i$ be the total amount of traffic flow.

Note that in this section, we use single-path routing to introduce how the SAL method works. In Section 6.3, we extend SAL to deal with multipath routing problems. For single-path routing, we assume that each commodity is nonbifurcated, and in the routing graph, 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.

The SAL algorithm may start with a given partial allocation state x_0 or with the zero state ($x_{ip} \equiv 0$). In each step, it chooses (with state-dependent probability $q(|x|)$) between *allocation*(x), that is, adding the traffic flow for one non-allocated commodity to the current state x, and *disconnect*(x), that is, 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 in the routing graph. 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

ALGORITHM 1: Simulated Allocation (SAL)

```
n = 0; counter = 0; x = 0; F^best = +∞ ;
repeat
    if random(0, 1) < q(|x|) then
        allocation(x);
    end
    disconnect(x);
    if |x| = H then
        n = n + 1;
        counter = counter + 1;
        if F(x) < F^best then
            F^best = F(x);
            x^best = x;
            counter = 0;
        end
    end
until n = N or counter = M;
```

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 following properties.

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

According to Pióro and Medhi [2004], 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.

5.2. Analytical Switch Delay Modeling for NoCs

Accurate delay models for switches are required as an input to our 3D NoC synthesis problem, since we need the models to 1) estimate the switch delay when routing a traffic flow in the *allocation(x)* step in Section 5.1, and 2) evaluate the final 3D NoC synthesis solutions. In our work, we utilize the analytical delay model presented in Ogras and Marculescu [2007], 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_i 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

$$T_i = H_i + \frac{S - W}{f \cdot W}, \tag{4}$$

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_{ij} 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

$$(I - T\Lambda C)N = \Lambda \bar{R}, \tag{5}$$

where $\Lambda = diag\{\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{ip}\}$, $C = [C_1, C_2, \ldots, C_p]^T$, and $\bar{R} = ([R, R, \ldots, R]_{1\times p})^T$.

The switch model described by Equation (5) 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_i, W), and packet size S.

We further use this switch model to compute the average packet latency from source core s to destination core d (used in Equation (2)) as

$$L_{sd} = \sum_{i \in \prod_{sd}} (H_i + \tau_i) \; + \; D_{sd} \; + \; \frac{S - W}{f \cdot W}, \tag{6}$$

where

— $\prod_{sd}$ is the set of switches along the path of the packets sent from source s to the destination d;
— τ_i is the average waiting time of the incoming packets at switch i, which can be estimated as $\tau_i = N_j/\lambda_{ij}$ by Little's theorem [Little 1961]; and
— D_{sd} is the total link delay from s to d.

For further details, the reader is referred to Ogras and Marculescu [2007].

5.3. Switch Location Estimation and Path Cost Estimation

When routing a flow from source s to destination d in the *allocation(x)* step (refer to Section 5.1), 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 the 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).

Here, the path cost is the cost for NoC topology synthesis, as shown in Equation (2). 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.

5.4. 3D Floorplanning

As described in Section 4, 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 Hung et al. [2006] based on a B*-tree floorplan model. The floorplanner uses a built in thermal analysis technique based on the HS3D [Hung et al. 2006] 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.

6. EXPERIMENTAL RESULTS

6.1. 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 [Kahng et al. 2009; Wang et al. 2002] to estimate the power dissipation of the switches. The link power and delay are modeled based on the equations from Pavlidis and Friedman [2007]. Considering that the delay and power of TSV (interdie interconnect) in 3D circuits is much smaller (at least one order in magnitude) than that of the intradie interconnect wires [Loi et al. 2007; Yan and Lin 2008b], we ignore the delay and power of TSVs in this work. The delays of switches are estimated using the model described in Section 5.2. All switches and links are evaluated under a 45nm technology.

Several parameters affect the efficiency and performance of the SAL algorithm (Section 5.1). In choosing the $q(\gamma)$ function for SAL, we found that for $0 < \gamma < 1$, a constant function $q(\gamma) = q_0 = 0.9$ can produce good solutions. The user-specified iteration limit N is empirically set to three times that of k (the number of commodities in the core graph), and M is set to be 50 or 100, depending on the size of the MCF problem. We find that the best solutions are often obtained within k-visited full allocation states for all the benchmarks.

6.2. Impact of Each Strategy Applied in Our Algorithm 3D-SAL-FP

Our algorithm *3D-SAL-FP* (based on single-path routing) improves upon the previous algorithms of Murali et al. [2009] and Yan and Lin [2008b] 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; and 3) using a more accurate switch delay model that includes 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 four algorithms that we will compare in our results are as follows.

— *Baseline1*, based on the work by Murali et al. [2009], has two stages: 3D NoC synthesis and floorplanning of the synthesized NoC architecture. At the 3D NoC synthesis stage, a 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 find the floorplan of cores and used switches in the NoC architecture.
— *Baseline2* differs from *Baseline1* in that it applies an improved traffic flow routing strategy (SAL) in the 3D NoC synthesis stage.
— *Baseline3* improves upon *Baseline2* by feeding back the results of the 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 are 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.
— *3D-SAL-FP* is our proposed approach and differs from *Baseline3* in that it uses the accurate switch delay model (described in Section 5.2) 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 to 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 Yan and Lin [2008b] to build large, synthetic 3D benchmarks, which can be viewed as the "many-core" version of the small published ones. This method is based on the NoC-centric bandwidth version of Rent's rule proposed by Greenfield et al. [2007]. 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 $^{\circ}C$). Considering that SAL is a stochastic approach, we run each algorithm ten times and present data in the tables showing the best results among all the runs. The same strategy is applied to the experiments in the subsequent sections.

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 our

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

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. 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 trade-off points may be identified.

6.3. 3D-SAL-FP Based on Multipath Routing

In Section 6.2, our *3D-SAL-FP* algorithm is based on single-path routing, which means that each commodity (traffic flow in the given core graph) is nonbifurcated, and we choose one single path in the routing graph for one commodity. In this section, we extend *3D-SAL-FP* to work with multipath routing, where each commodity can be split into several subflows, and each subflow can be routed independently in the routing graph.

Let L be the capacity of each subflow, then a commodity with traffic demand d_i can be split into $\lceil d_i/L \rceil$ subflows. Here we use the capacity L to control the granularity of the subflow, so that the size of the MCF problem in multipath routing can be controlled. In our experiments, L is set individually for each benchmark because the values of the traffic demand d_i vary greatly from benchmark to benchmark. After splitting the commodities in the core graph, we treat each of the resulting subflows as a single routing unit and select one minimum cost path to route it. Since the subflows constituting the same commodity can be routed on different paths on the routing graph, we refer to the new routing problem supporting subflows as *multipath routing*. In fact, the costs for the switches and links in the routing graph are state-dependent: when the subflows are routed one after another, it is highly possible for the subflows of one commodity to be routed on different paths.

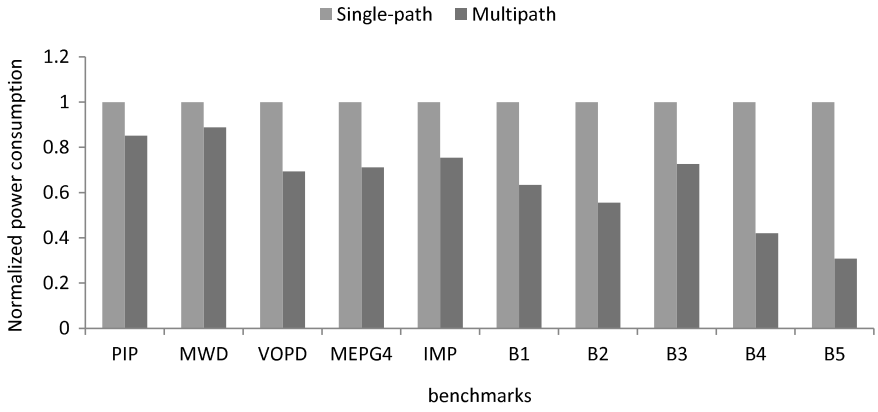
Figure 2 presents the comparison results of single-path and multipath routings. The results are normalized to the single-path case. Considering that multipath routing can reduce the peak link bandwidth needs and therefore lower network operating frequency [Murali et al. 2006], we evaluate the network power consumption using the optimized frequency number corresponding to the peak link bandwidth in the NoC synthesis solution. Given the peak link bandwidth, we can obtain the optimized NoC frequency as optimized frequency = peak link bandwidth/link width.

From Figure 2, we can see that, on average, multipath routing can obtain 35% power savings compared to single-path routing. The overhead in runtime for multipath routing is more than 3 times that of most of the benchmarks. This is because 1) it takes longer time for SAL to find a full allocation solution in multipath routing, and 2) SAL needs to explore an expanded solution space for the multipath case. However, this overhead is related to the increased search space and will affect any other algorithm (e.g., SA) that solves this problem formulation.

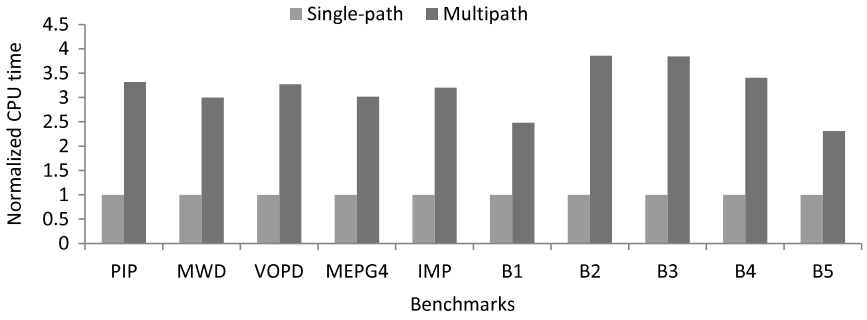
6.4. Comparison of SAL and Simulated Annealing

In this section, we compare the performance of our single-path-based SAL algorithm with another widely used stochastic approach, Simulated Annealing (SA), by replacing SAL with SA in the *3D-SAL-FP* implementation. We implement two kinds of SA moves in our work.

(1) Consider that in Algorithm 1, the SAL approach applies two basic moves, *allocation*(x) and *disconnect*(x). In order to perform a fair comparison, we



(a) Network power.



(b) CPU time.

Fig. 2. Comparisons of single-path and multipath routing schemes.

integrate these two basic moves into the SA engine: in SA, given one full allocation state x, a move to a neighbor full allocation state *neighbor(x)* is implemented as a series of single-flow moves (*allocation*(x) and *disconnect*(x)), as introduced in Section 5.1. We refer to this SA implementation as *Single-flow SA*.

(2) Given one full allocation state x, a move to a neighbor full allocation state *neighbor(x)* can be obtained in one of two ways.
   — Disable one of the used switches and reroute all the traffic flows passing through that switch.
   — Disable one of the used links and reroute all the traffic flows passing through that link.

   For this SA implementation, several flows may be rerouted in one single move, so we refer to it as *Multi-flow SA*.

The performance and runtime of the SA algorithm is affected by several parameters, such as the initial and end temperatures T_i and T_e, the inner loop number N_{inner} at each temperature, and the temperature reduction parameter τ. We investigate the impact of runtime on SA's performance with one randomly selected benchmark *IMP*. Figure 3 shows the simulation results when *Single-flow SA* is applied. We use the following cost function to evaluate the final 3D NoC solutions.

$$cost = w_1 * \texttt{chip area} + w_2 * \texttt{network power} + w_3 * \texttt{network latency} + w_4 * \texttt{TSV count}, \quad (7)$$

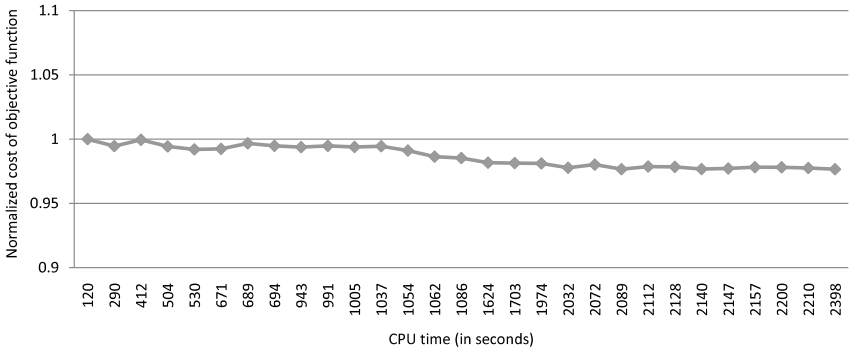


Fig. 3. The impact of runtime on the performance of SA for benchmark IMP.

where the default weights are $w_1 = 1$, $w_2 = 10$, $w_3 = 5$, and $w_4 = 3$. We normalize all the costs to the baseline case with a runtime of 120 seconds. We tune the parameters for an appropriate runtime/quality trade-off from SA. For example, for benchmark *IMP*, Figure 3 shows that as we increase the runtime of SA from 120 seconds to 2,398 seconds, the improvement to the objective function shows diminishing returns: the improvement is very small, but the increase in the runtime is about 20 times. We factor this into our experiments, and the runtime of SA for the benchmarks ranges from several minutes to several hours.

Figure 4 presents the results of SAL and two SA implementations on both the published and synthetic benchmarks. We use the cost function shown in Equation (7), where all results are normalized to the SAL case. Figure 4 shows that in terms of the quality of the solutions, SAL performs approximately as well as SA, but that the execution times are much smaller than those of SA. For example, for the large benchmark B5 with 124 cores and 266 flows, the cost reported by SAL is about 3% less than that of *Single-flow SA*, while the speedup is about 18 times. Compared to *Multi-flow SA*, *Single-flow SA* has longer execution time for most of the benchmarks because it needs more moves to find a neighbor full allocation state, but it can find slightly better solutions in most cases.

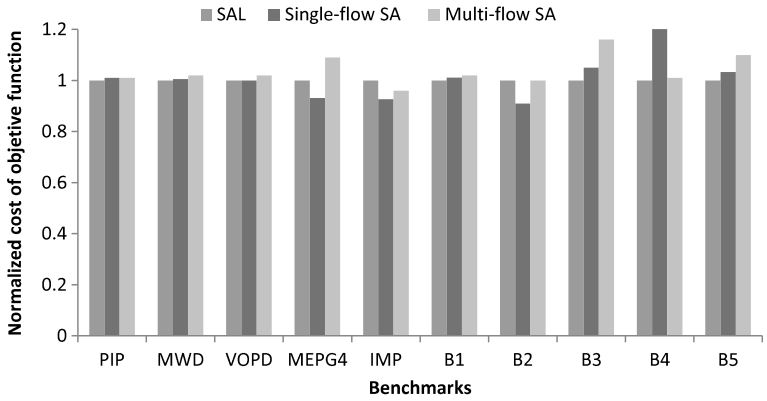
6.5. Exploration of TSV Count

Next, we explore the trade-offs associated with using more or fewer TSVs in the design. In 3D circuits, more TSVs imply more vertical interconnects, which mean that the latency can be reduced in the resulting NoC topology. However, the corresponding overhead includes increased design area and excessive utilization of a valuable vertical resource (note that TSVs are also required for routing supply nets, clock nets, thermal vias, etc.). In this section, we explore the trade-off between TSV count and network latency. The single-path-based *3D-SAL-FP* algorithm is applied for this experiment.

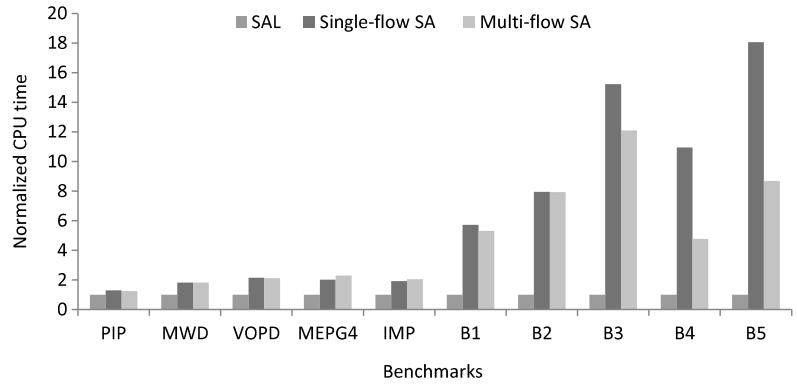
Figure 5 shows the trade-off curve when we gradually increase the weight of TSV w_4 (Equation (2)) from 1 to 18. As we can see from this figure, the number of TSVs can be largely optimized when we increase w_4 from 1 to 6, and the increase of the network latency is less than 2%. After that point, the TSV count gradually levels off, and the network latency increases much faster. The minor nonmonotonicities in this figure can be attributed to the nature of the stochastic approach.

6.6. Delay and Power Reduction Potential in 3D NoCs

In this section, we further investigate the impact of 3D integration on the NoC architecture design. The benchmark B3 (with 69 cores and 136 flows) was selected, and



(a) Cost of objective function.



(b) Runtime.

Fig. 4. Comparisons with SA.

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.

Our results show a clear trade-off when implementing NoC architecture using 3D circuits: 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 an increased number of TSVs and higher chip temperature. In addition, we can observe that although 3D circuits have the potential to reduce the link delay and power, the improvement flattens out as the number of 3D tiers goes beyond a certain point. For example, as shown in Table III, the network latency does not decrease much as we go from three layers to four.

7. 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

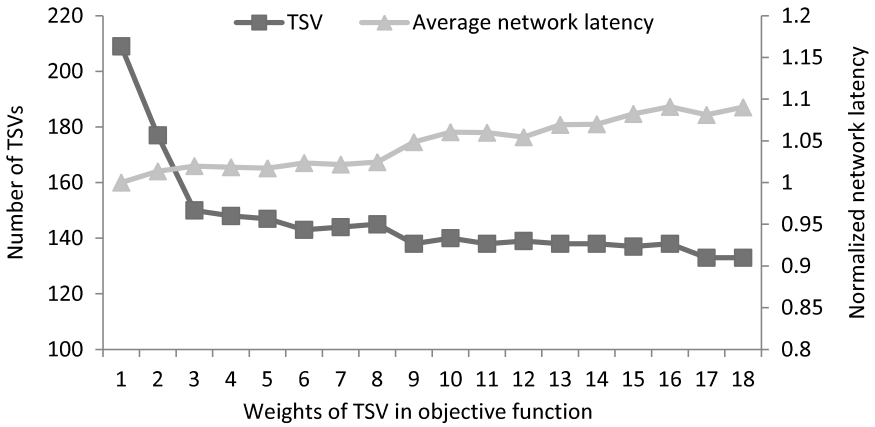


Fig. 5. The trade-off between number of TSVs and average network latency for benchmark B1.

Table III. Comparison of the Impact of Different Numbers of 3D Tiers on NoC Architecture Design for Benchmark B3

Layers	Footprint (mm^2)	Network Power			Max Path Length (mm)	Max Link Delay (ns)	Max Network Latency (ns)	Avg Network Latency (ns)	# of TSVs	T_{max} ($^{\circ}C$)	Time (s)
		Switch (mW)	Link (mW)	Total (mW)							
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

allocation (SAL) to reduce the dependency of NoC design results on flow ordering. We also use an accurate delay model for switches in NoCs, which considers 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 results on a set of benchmarks show that our algorithm can produce greatly improved solutions, compared to the baseline algorithm with fixed-order flow-routing, simple delay model and without feedback from floorplanning step, reflecting prior work. In comparison with SA, we show that SAL can find approximately the same quality solutions but with better computational efficiency.

We have also investigated several degrees of freedom in this space. First, our comparative study between single-path and multipath routing schemes in the SAL framework shows that multipath routing can achieve large power savings with slightly larger computation times. Second, when we study the impact of TSV count on the network performance in 3D NoCs, we find that there is a "sweet spot" where the TSV count is effectively controlled without much penalty on the network performance. Third, we investigate the benefits that 3D circuits can bring to the NoC architecture design and show that link power and delay can be largely improved when moving to 3D implementation, at the cost of TSV and chip temperature.

ACKNOWLEDGMENTS

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

REFERENCES

ADDO-QUAYE, C. 2005. Thermal-aware mapping and placement for 3-D NoC designs. In *Proceedings of the IEEE International System-on-Chip Conference*. 25–28.

AHONEN, T., BIN, H., AND NURMI, J. 2004. Topology optimization for application-specific networks-on-chip. In *Proceedings of the International Workshop on System Level Interconnect Prediction*. 53–60.

BANERJEE, K., SOURI, S. J., KAPUR, P., AND SARASWAT, K. C. 2001. 3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems. *Proc. IEEE 89,* 5.

BENINI, L. AND MICHELI, G. D. 2002. Networks on chips: A new SoC paradigm. *Computer 35,* 1, 70–78.

DALLY, W. J. AND TOWLES, B. 2001. Route packets, not wires: On-chip inteconnection networks. In *Proceedings of the ACM/IEEE Design Automation Conference*. 684–689.

DAVIS, W. R., WILSON, J., MICK, S., XU, J., HUA, H., MINCO, C., SULE, A. M., STEER, M., AND FRANZON, P. D. 2005. Demystifying 3D ICs: The pros and cons of going vertical. *IEEE Des. Test Comput. 22,* 6, 498–510.

FEERO, B. AND PANDE, P. P. 2007. Performance evaluation for three-dimensional networks-on-chip. In *Proceedings of the IEEE Annual Symposium on VLSI*. 305–310.

GREENFIELD, D., BANERJEE, A., LEE, J.-G., AND MOORE, S. 2007. Implications of Rent's rule for NoC design and its fault-tolerance. In *Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip*. 283–294.

HANSSON, A., GOOSSENS, K., AND RĂDULESCU, A. 2005. A unified approach to constrained mapping and routing on network-on-chip architectures. In *Proceedings of the International Conference on Hardware-Software Codesign and System Synthesis*. 75–80.

HU, Y., CHEN, H., ZHU, Y., CHIEN, A. A., AND CHENG, C.-K. 2005. Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimizations. In *Proceedings of the International Conference on Computer Design*. 111–118.

HU, Y., ZHU, Y., CHEN, H., GRAHAM, R., AND CHENG, C.-K. 2006. Communication latency aware low power NoC synthesis. In *Proceedings of the ACM/IEEE Design Automation Conference*. 574–579.

HUNG, W.-L., LINK, G. M., XIE, Y., VIJAYKRISHNAN, N., AND IRWIN, M. J. 2006. Interconnect and thermal-aware floorplanning for 3D microprocessors. In *Proceedings of the International Symposium on Quality Electronic Design*. 98–104.

KAHNG, A., SAMADI, K., LI, B., AND PEH, L.-S. 2009. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of the Design, Automation and Test in Europe*.

KIM, J., NICOPOULOS, C., PARK, D., DAS, R., XIE, Y., NARAYANAN, V., YONSIF, M. S., AND DAS, C. R. 2007. A novel dimensionally-decomposed router for on-chip communication in 3D architectures. In *Proceedings of the International Symposium on Computer Architecture*. 138–149.

LI, F., NICOPOULOS, C., RICHARDSON, T., XIE, Y., NARAYANAN, V., AND KANDEMIR, M. 2006. Design and management of 3D chip multiprocessors using network-in-memory. In *Proceedings of the International Symposium on Computer Architecture*. 130–141.

LIM, S. K. 2005. Physical design for 3D system on package. *IEEE Des. Test Comput. 22,* 6, 532–539.

LITTLE, J. D. C. 1961. A proof for the queuing formula: $L = \lambda W$. *Oper. Res. 9,* 3, 383–387.

LOI, I., ANGIOLINI, F., AND BENINI, L. 2007. Supporting vertical links for 3D networks-on-chip: Toward an automated design and analysis flow. In *Proceedings of the International Conference on Nano-Networks*. 1–5.

MARK, D. S., KARPOVSKY, M., AND ZAKREVSKI, L. 2003. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Trans. Netw. 11,* 3, 411–421.

MATSUTANI, H., KOIBUCHI, M., AND AMANO, H. 2007. Tightly-coupled multi-layer topologies for 3-D NoCs. In *Proceedings of the International Conference on Parallel Processing*. 75–75.

MORROW, P., BLACK, B., KOBRINSKY, M. J., MUTHUKUMAR, S., NELSON, D., PARK, C.-H., AND WEBB, C. 2007. Design and fabrication of 3D microprocessors. In *Proceedings of the Materials Research Society Symposium*.

MURALI, S., MELONI, P., ANGIOLINI, F., ATIENZA, D., CARTA, S., BENINI, L., DE MICHELI, G., AND RAFF, L. 2006. Designing application-specific networks on chips with floorplan information. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 355–362.

MURALI, S., SEICULESCU, C., BENINI, L., AND MICHELI, G. D. 2009. Synthesis of networks on chips for 3D systems on chips. In *Proceedings of the Asia-South Pacific Design Automation Conference*. 242–247.

OGRAS, U. Y. AND MARCULESCU, R. 2007. Analytical router modeling for networks-on-chip performance analysis. In *Proceedings of the Design, Automation and Test in Europe*. 1096–1101.

PAVLIDIS, V. F. AND FRIEDMAN, E. G. 2007. 3-D topologies for networks-on-chip. *IEEE Trans. VLSI Syst. 15,* 10, 1081–1090.

PIÓRO, M. AND GAJOWNICZEK, P. 1997. Solving multi commodity integral flow problems by simulated allocation. *Telecomm. Syst. 7,* 1–3, 17–28.

PIÓRO, M. AND MEDHI, D. 2004. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

SEICULESCU, C., MURALI, S., BENINI, L., AND MICHELI, G. D. 2009. SunFloor 3D: A tool for networks on chip topology synthesis for 3D systems on chip. In *Proceedings of the Design, Automation and Test in Europe*. 9–14.

SRINIVASAN, K., CHATHA, K. S., AND KONJEVOD, G. 2005. An automated technique for topology and route generation of application specific on-chip interconnection networks. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 231–237.

WANG, H., ZHU, X., PEH, L.-S., AND MALIK, S. 2002. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture*. 294–305.

XUE, L., LIU, C. C., KIM, H.-S., KIM, S. K., AND TIWARI, S. 2003. Three-dimensional integration: Technology, use, and issues for mixed-signal applications. *IEEE Trans. Electron Devices 50,* 3, 601–609.

YAN, S. AND LIN, B. 2008a. Application-specific network-on-chip architecture synthesis based on set partitions and Steiner trees. In *Proceedings of the Asia-South Pacific Design Automation Conference*. 277–282.

YAN, S. AND LIN, B. 2008b. Design of application-specific 3D networks-on-chip architectures. In *Proceedings of the International Conference on Computer Design*. 142–149.

ZHOU, P., YUH, P.-H., AND SAPATNEKAR, S. 2010. Application-specific 3D network-on-chip design using simulated allocation. In *Proceedings of the Asia-South Pacific Design Automation Conference*. 517–522.

Received March 2011; revised June 2011, September 2011; accepted November 2011