NoC Design and Performance Optimization

Pingqiang Zhou*, Jieming Yin†, Antonia Zhai † and Sachin S. Sapatnekar*
* Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, Minnesota 55455
Email: {pingqiang, sachin}@umn.edu
† Department of Computer Science and Engineering, University of Minnesota, Minneapolis, Minnesota 55455
Email: {jyin, zhai}@cs.umn.edu

***Abstract*—Systems-on-Chip (SoCs) and Chip Multiprocessors (CMPs) have strong global communication requirements, and networks-on-chip (NoCs) have been proposed as a scalable solution to overcome these communication challenges. This work explores the application of NoCs in both SoCs and CMPs. For SoCs, our work considers the application-specific NoC architecture design problem in a 3D environment. We present an efficient 3D NoC synthesis algorithm, based on simulated allocation (a stochastic method for traffic flow routing), interleaving between floorplanning and NoC synthesis, and accurate power and delay models for NoC components. For CMPs, we observe that intermittent traffic patterns imply that voltage and frequency scaling can be used effectively to reduce NoC energy consumption, but the associated increase in latency and degradation in throughput must be managed. We propose flexible pipeline routers where pipeline stages are reconfigured upon frequency scaling. By reducing the number of pipeline stages, the proposed router enables us to scale down the network frequency without increasing router latency.**

I. Application-Specific 3D NoC Design

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], [2]. 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 [3], [4]. 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.

Our work [5] addresses the problem of designing application-specific 3D NoC architectures for custom SoC designs, in conjunction with floorplanning. Our approach has three significant features that together make it uniquely different from competing approaches: first, we use improved traffic flow routing using a stochastic flow allocation method Simulated Allocation (SAL) that accommodates a realistic objective function with 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.

A. 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.

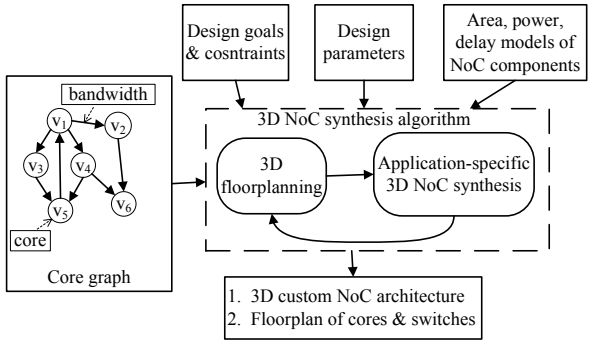


Fig. 1. Application-specific 3D NoC synthesis flow.

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:

$$\text{Capacity constraints:} \quad \sum_{i=1}^{k} f_i(u,v) \le c(u,v)$$

$$\text{Flow conservation:} \quad \sum_{\omega \in V', u \neq s_i, t_i} f_i(u,\omega) = 0$$

$$\text{where} \quad \forall v, u \; f_i(u,v) = -f_i(v,u)$$

$$\text{Demand satisfaction:} \quad \sum_{\omega \in V'} f_i(s_i,\omega) = \sum_{\omega \in V'} f_i(\omega,t_i) = d_i$$

Superficially, this idea seems similar to [6], 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

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

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 solving the MCF problems where the objective function is in such a form. SAL [7] 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 SAL procedure yields the NoC topology and the paths for all the traffic flows in the core graph.

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.

B. Experimental Results

In our work, we use Orion [8] to estimate the power dissipation of the switches. The link power and delay are modeled based on the equations from Pavlidis *et al.* [9], and the delay of switches are estimated using the analytical delay model presented in [10], which includes the effects of queueing delay and network contention.

Our algorithm *3D-SAL-FP* improves upon the previous algorithms in [11], [12] 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.* [11], 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 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 [12] to generate the large synthetic 3D benchmarks. 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.

II. NoC Frequency Scaling with Flexible-Pipeline Routers

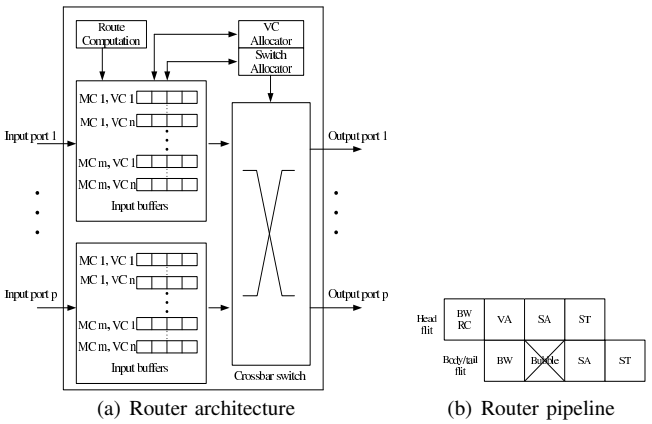
Over the last decade, chip multiprocessors (CMPs) have emerged as a potential solution to maximize computation while remaining with a stringent power envelope, by integrating multiple smaller and more energy efficient cores in a single chip. These cores must communicate through an efficient on-chip interconnection network (NoC), and NoC design is vital to both performance and power.

A critical design parameter that directly affects both performance and power of NoC is the network frequency. Techniques such as VFS [13] have been widely investigated to allow the network to operate at a lower frequency to reduce energy consumption, when possible. However, reducing NoC frequency increases latency and reduces network throughput, which in turn, degrades overall system performance. As a result, in prior work, frequency is only moderately scaled, by up to 20%, with the network and the cores operating asynchronously at different frequencies. To fully realize the potential of VFS, it is desirable to be able to scale down the network frequency significantly (our work [14] examines changes of 2×-4×) up to the point where the performance penalty remains small. Moreover, since our frequency scaling uses integer multiples of the clock frequency, we can continue to operate within a fully synchronous paradigm.

For some applications, a reduction in NoC throughput has relatively little impact on performance, but increasing NoC latency can cause significant performance degradation. For these types of workloads, we propose to reconfigure the router pipeline when scaling down the network frequency. With this technique, the impact on the average time to traverse the NoC for a single message is minimal, allowing us to reduce NoC energy consumption, without significant performance degradation. We refer to the proposed routers as *flexible-pipeline* routers, since the router pipeline stages are adaptively configured based on the workload.

A. Flexible-pipeline Router

We use a classic four-stage-pipelined virtual-channel (VC) router [15], shown in Figure 2(a), as an example to show how our strategy works. However, our approach can definitely work on other enhanced router designs so long as they have multiple pipeline stages. The router supports multiple message classes (MCs), and VCs from all MCs are multiplexed across the input port. Figure 2(b) shows the corresponding fixed four-stage router pipeline. When a head flit arrives at an input port, it is first decoded and buffered in the buffer write (BW) pipeline stage In the same cycle, a request is sent to the route computation unit (RC) to calculate the output port for this packet. The header then arbitrates for a free VC of its output port in the VC allocation (VA) stage. Upon successful allocation of an output VC, it proceeds to the switch allocation (SA) stage where it arbitrates for the switch input and output ports. On winning the switch, the flit moves to the switch traversal (ST) stage, where it traverses the crossbar and is placed on the output link connected to the next node. The body and tail flits follow a similar pipeline except that they do not go through RC and VA stages, instead inheriting the VC allocated by the head flit. The tail flit deallocates the VC reserved by the packet when it leaves the router.



(a) Router architecture (b) Router pipeline

Fig. 2. Classic four-stage virtual-channel router.

Although such fine-granularity router pipeline design can run at high frequency, and therefore support high throughput, it may degrade the system performance significantly when the network is slowed down to save power consumption. Therefore, we propose a flexible-pipeline reconfiguration approach to adapt to a change in the network speed, based on accurate delay models for the components in the baseline router.

TABLE III
Delay values (in units of τ) of each router component

Router	BW+RC(BR)		VA		SA		ST	
	t_{BR}	h_{BR}	t_{VA}	h_{VA}	t_{SA}	h_{SA}	t_{ST}	h_{XB}
5-port	100	0	56.5	9	68.7	9	45.0	0
6-port	100	0	58.7	9	70.2	9	46.8	0

We model the delay of each router component shown in Figure 2(a) by the technology-independent parametric equations presented in [16]. For each component i, we have two delay estimates: *latency* (t_i) and *overhead* (h_i). Table III lists the t_i and h_i numbers of each component for the cases of a 5-port router and a 6-port router, where τ be the delay of an inverter with identical input capacitance. From this data we can clearly see that the pipeline stages are imbalanced, so we apply the time-borrowing techniques to boost the pipeline frequency using clock skew optimization [17]: Let T_i be the delay for pipeline stage i and let n be the total number of stages, then the clock time for the n-stage router after time-borrowing is $T_{clk} = \frac{\sum_{i=1}^{n} T_i}{n}$.

Furthermore, this four-stage linear router can be reconfigured as either a three-stage, or a two-stage, or even a one-stage pipeline. For example, a three-stage pipeline can be obtained by combined any of the two successive components such as 1) BR and VA, or 2) VA and SA, or 3) SA and ST. Of these, we choose the one with the minimum optimized clock period T that maximizes the router frequency that the pipeline can support: note that since the h_i values for various stages are different, the optimal frequency varies with our choice. The best choices are found to be: 1) for the three-stage pipeline, we combine SA and ST into one single stage and 2) for the two-stage pipeline, we combine VA, SA and ST into one single stage.

TABLE IV
Optimal clock periods/frequencies for various pipeline configurations

Router	4-stage		3-stage		2-stage		1-stage	
	T_{clk}	Max. F	T_{clk}	Max. F	T_{clk}	Max. F	T_{clk}	Max. F
5-port	72.05	1.78	93.07	1.32	135.10	0.87	270.2	0.39
6-port	73.43	1.75	94.90	1.30	137.85	0.85	275.7	0.38

Table IV summarize the results for two different router sizes after applying time-borrowing techniques. For each pipeline design case, we list the optimal clock time T_{clk} in units of τ, and the maximum frequency F that the pipeline can support in the GHz range. Given the maximum frequency numbers for

each pipeline design, it is straightforward to select the optimal pipeline design at a given network speed. The basic principle behind this idea is that at the same network speed, a shorter pipeline with fewer stages leads to lower router latency, and therefore, better network performance. Table V presents the results of optimal pipeline stage number $N = 1, 2, 3, 4$ for different routers with different slow down parameter S – the router to processor clock ratio. For example, $S = 2$ implies that one network clock cycle is equivalent to two processor cycles. The processor frequency is 1.5 GHz.

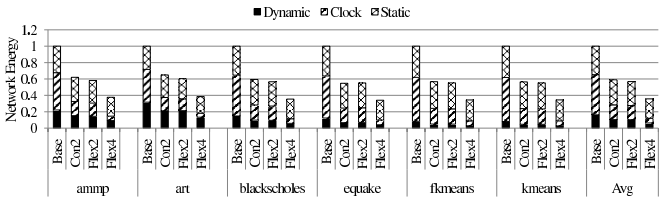
TABLE V
THE OPTIMAL N WITH DIFFERENT S

Router	S = 1	S = 2	S = 3	S = 4	S = 5
5-port	4	2	2	1	1
6-port	4	2	2	1	1

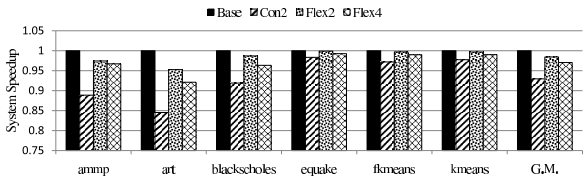
B. Experimental Results

In our work, we simulate an 8-core CMP system with 65nm technology nodes, where the cores/L2 cache banks and memory controllers are connected using a mesh network. We choose applications from the SPEC OMP2001 [18], NU-Mine [19] and PARSEC [20] benchmark suites as our workload input. In our baseline, the processors and NoC operate at the same frequency ($S = 1$). We found that, the link delays are small enough that for any of the S values used here, a flit can traverse a link within one network cycle.

Figure 3 shows the performance and energy results, with routers scaled with different slow down factors ($S = 2, 4$). The results are normalized to the case when $S = 1$. The first set of bars in Figure 3 corresponds to the unscaled NoC (bars *Base*), the next two sets of bars correspond to CMP systems with fixed-pipeline routers (bars *Con2*) and flexible-pipeline routers (bars *Flex2*) when the network frequency is scaled by 50% and voltage is scaled from 1.2V to 1.0V.



(a) Network energy. *Dynamic* corresponds to the total dynamic energy consumed by both routers and network links; segment *Clock* corresponds to dynamic energy for distributing the clock; *Static* corresponds to the total static (leakage) energy consumed.



(b) System performance

Fig. 3. Comparison of fixed-pipeline and flexible-pipeline routers. All results are normalized to *Base*.

In general, reducing the network frequency can reduce energy consumption (Figure 3(a)), but it can also increase network latency and degrade network throughput. Such impacts directly translate into system performance degradation, as shown in Figure 3(b). For fixed-pipeline routers (bars *Con2*), on average, energy consumption is reduced by 41%, but system performance is degraded by 7%. The reduction in energy consumption is mainly due to reduction in clock energy and dynamic energy. When the flexible-pipeline router is deployed, with the same voltage/frequency scaling, the average system performance degradation is only 1.6%, and the average network energy consumption is reduced by 43% (bars *Flex2*). This corresponds to a 2% additional reduction from the fixed pipeline case. This small reduction in energy is a side effect of performance improvement: by completing the application faster, we are able to reduce static energy. Overall, NoCs with flexible-pipeline routers are more energy efficient.

Bars 3 and 4 of Figures 3(a) and 3(b) show the performance and network energy consumption as the network is scaled by a factor of two and four ($S = 2, 4$), with flexible-pipeline routers. It is clear that scaling reduces both the performance and the energy. And all workloads suffer noticeable performance degradation. However, the energy savings in going from a scaling factor of two to four shows diminishing returns as compared to the case where we go from scaling factor one to two. This is primarily related to the fact that applications take longer to complete, and the NoC hardware is activated for a longer duration, during which static (leakage) energy is expended. This increase in static energy offsets the gains made in reducing the dynamic network energy.

ACKNOWLEDGMENT

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

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*, vol. 89, no. 5, 2001.
[2] W. R. Davis *et al.*, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 498–510, Nov.-Dec. 2005.
[3] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.
[4] W. J. Dally and B. Towles, "Route packets, not wires: On-chip inteconnection networks," in *Proc. DAC*, 2001, pp. 684–689.
[5] P. Zhou, P.-H. Yuh, and S. S. Sapatnekar, "Application-specific 3D network-on-chip design using simulated allocation," in *Proc. ASPDAC*, 2010, pp. 517–522.
[6] Y. Hu, Y. Zhu, H. Chen, R. Graham, and C.-K. Cheng, "Communication latency aware low power NoC synthesis," in *Proc. DAC*, 2006, pp. 574–579.
[7] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.
[8] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *MICRO 35*, 2002, pp. 294–305.
[9] V. F. Pavlidis and E. G. Friedman, "3-D topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 15, no. 10, pp. 1081–1090, 2007.
[10] U. Y. Ogras and R. Marculescu, "Analytical router modeling for networks-on-chip performance analysis," in *Proc. DATE*, 2007, pp. 1096 – 1101.
[11] S. Murali, C. Seiculescu, L. Benini, and G. D. Micheli, "Synthesis of networks on chips for 3D systems on chips," in *Proc. ASPDAC*, 2009, pp. 242–247.
[12] S. Yan and B. Lin, "Design of application-specific 3D networks-on-chip architectures," in *Proc. ICCD*, 2008, pp. 142–149.
[13] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A Case for Dynamic Frequency Tuning in On-chip Networks," in *Proc. Int. Symp. Microarchitecture*, 2009, pp. 292–303.
[14] P. Zhou, J. Yin, A. Zhai, and S. S. Sapatnekar, "Noc frequency scaling with flexible-pipeline routers," in *Proc. Int. Symp. Low Power Electronics & Design*, 2011.
[15] N. Agarwal, L.-S. Peh, and N. Jha, "Garnet: A Detailed Interconnection Network Model inside a Full-system Simulation Framework," Princeton University, Tech. Rep. CE-P08-001, 2008. [Online]. Available: http://www.princeton.edu/ niketa/garnet
[16] L.-S. Peh and W. J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2001, pp. 255–266.
[17] S. Sapatnekar, *Timing*. Boston, MA: Kluwer Academic Publishers, 2004.
[18] "SPEC OMP2001," Available at http://www.spec.org/omp/.
[19] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, J. Pisharath, G. Memik, and A. Choudhary, "MineBench: A Benchmark Suite for Data Mining Workloads," in *IEEE Int. Symp. on Workload Characterization*, 2006, pp. 182–188.
[20] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Proc. Int. Conf. Parallel Architectures and Compilation Techniques*, 2008, pp. 72–81.