

Designing Message-Dependent Deadlock Free Networks on Chips for Application-Specific Systems on Chips

Srinivasan Murali^{*}, Paolo Meloni[§], Federico Angiolini[‡], David Atienza^{†+}, Salvatore Carta[¶],
Luca Benini[‡], Giovanni De Micheli[†], Luigi Raffo[§]

^{*}CSL, Stanford University, Stanford, USA, smurali@stanford.edu

[§]DIEE, University of Cagliari, Cagliari, Italy, {paolo.meloni@diee.unica.it, luigi@diee.unica.it}

[‡]DEIS, University of Bologna, Bologna, Italy, {fangiolini@deis.unibo.it, lbenini@deis.unibo.it}

[¶]DMI, University of Cagliari, Cagliari, Italy, salvatore@unica.it

[†]LSI, EPFL, Lausanne, Switzerland, {david.atienza, giovanni.demicheli@epfl.ch}

⁺DACYA, Complutense University of Madrid (UCM), Madrid, Spain.

Abstract

Networks on Chip (NoC) has emerged as the paradigm for designing scalable communication architecture for Systems on Chips (SoCs). Avoiding the conditions that can lead to deadlocks in the network is critical for using NoCs in real designs. Methods that can lead to deadlock-free operation with minimum power and area overhead are important for designing application-specific NoCs. A major class of deadlocks that occur in NoCs are due to the dependencies among the resources shared by different message types. In this work, we consider the problem of avoiding message-dependent deadlocks during the NoC topology synthesis phase. We show that by considering this issue during topology synthesis, we can obtain a significantly better NoC design than traditional methods, where the deadlock avoidance issue is dealt with separately. Our experiments on several SoC benchmarks show that our proposed scheme provides large reduction in NoC power consumption (an average of 38.5%) and NoC area (an average of 30.7%) when compared to traditional approaches.

Keywords: Networks on Chips, Systems on Chips, Message-dependent deadlocks, routing-dependent deadlocks, topology, synthesis.

1 Introduction

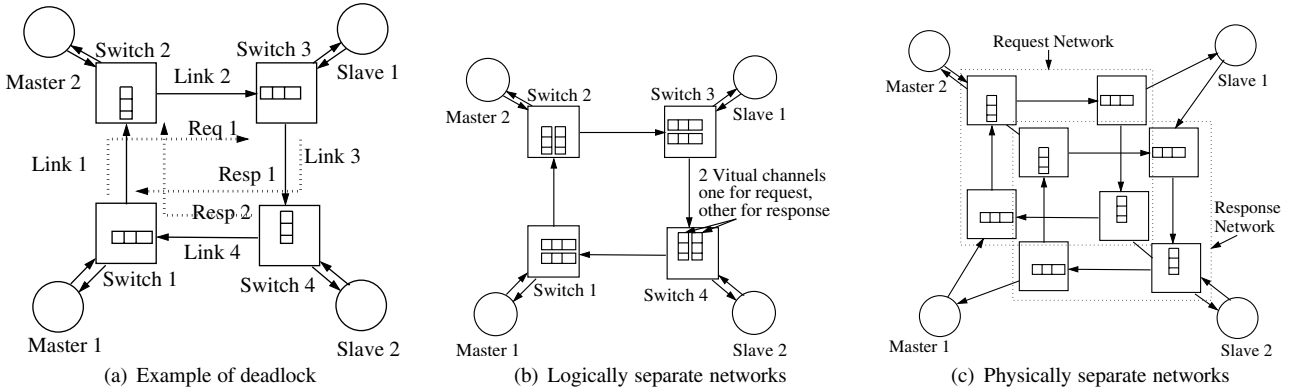
Today's *Systems on Chips (SoCs)* consist of a large number of computing and storage cores that are interconnected by means of single or multiple layers of buses. In order to cope with the large communication demands of such SoCs, a modular, scalable interconnect based on *Networks on Chips (NoCs)* is needed [1]-[6].

Designing a custom-tailored interconnect that satisfies the performance and design constraints of the SoC is important to achieve efficient NoC designs [27]-[32]. A critical, but often neglected issue when designing NoCs is that they have to guarantee deadlock-free operation. If the NoC has no support to either avoid or recover from deadlocks, then

correct functionality of the system cannot be guaranteed. This can lead to system crashes and unexpected system behavior, which is clearly unacceptable for SoCs. Designing efficient methods that avoid such a situation with minimum power and area overhead is an important research area in the NoC domain.

The deadlocks that can occur in NoCs can be broadly categorized into two classes: *routing-dependent* deadlocks and *message-dependent* deadlocks [33], [7]-[12]. Routing-dependent deadlocks occur when there is a cyclic dependency of resources created by the packets on the various paths in the network. For regular topologies (such as the mesh, torus), the use of restricted routing functions based on turn models is an effective way to avoid routing-dependent deadlocks [9], [10]. For custom application-specific NoCs, obtaining deadlock-free paths is a bigger challenge. In fact, recently there have been several works that have addressed deadlock-free path selection mechanisms for custom NoC designs [12], [22], [32]. While there are several works that target freedom from routing-dependent deadlocks in NoCs, relatively few works exist on obtaining freedom from message-dependent deadlocks. The major focus of this paper is to address this important issue of obtaining message-dependent deadlock-free network operation.

Message-dependent deadlocks occur when interactions and dependencies are created between different message types at network endpoints, when they share resources in the network. Even when the underlying network is designed to be free from routing-dependent deadlocks, the message-level deadlocks can block the network indefinitely, thereby affecting the proper system operation. An example situation where a message-dependent deadlock occurs is presented in Figure 1(a). In this example, two of the cores are masters and two other cores are slaves. In this system, we assume two kinds of messages: *request* and *response*. Consider the following situation: Master 1 sends a request to Slave 1 (*Req 1*), Slave 1 is replying to a previously issued request to Master 1 (*Resp 1*) and at the same time, Slave 2 sends a response to Master 2 (*Resp 2*). When requests and responses share the same links, *Resp 2* is waiting for link 1 which is used by *Req 1* and *Resp 1* waits for link 4 used by



Resp 2. Meanwhile, *Req 1* is waiting for Slave 1, the operation of which has been stalled as *Resp 1* could not complete. Thus, none of the messages can move ahead, leading to a deadlock situation. An interesting point to note here is that message-level deadlocks can be avoided if the receivers have infinitely large buffering or if they have perfectly ideal operation (consuming all received data instantly), which would avoid queuing of the packets in the network. Obviously, such a solution is not feasible to obtain in practice.

In traditional multi-processor interconnection networks, the most common ways to avoid message-dependent deadlocks are the use of separate logical or physical networks for the different message types [13]-[21]. This would ensure that the different message types do not share the network components, thereby guaranteeing freedom from message-dependent deadlocks. The most common method to achieve separate logical networks is the use of separate virtual channels for the different message types [13]. For the example design presented in Figure 1(a), each router input will need two virtual channels: one for the request messages and the other for the response messages (refer Figure 1(b)). This separation of message types is maintained at all the switches in the network. In the case of separate physical networks, the request network is built separately from the response network, an example of which is shown in Figure 1(c). This is the most commonly used solution in complex bus designs such as the STBus from STMicroelectronics [19] and several multi-processor designs [20], [21].

In this work, we show that by mapping the different message types onto different network resources during the topology mapping and synthesis phase, we can achieve much better NoC designs (in terms of power consumption and network area) than traditional approaches. We present a topology synthesis algorithm that specifically considers the message types and ensures the creation of a network that is free from message-dependent deadlocks. We also implement the common methods of deadlock avoidance: having separate virtual channels and having physically separate networks for the message types. For all the schemes, we make the underlying network operation free from routing-dependent deadlocks by applying existing methods from [12]. We perform experiments on several SoC designs, which show that our proposed scheme provides large reduction in the NoC power consumption (an average of 38.5%) and area (an average of 30.7%) when compared to the traditional approaches.

2 Previous Work

The motivation for the use of NoCs has been established in several works [1]-[6]. The use of turn models to avoid deadlocks in mesh and torus networks has been presented in [10]. There has been a large body of work that have focused on developing routing-dependent deadlock-free operation for interconnection networks [9]-[12]. Several other works exist in the area of recovering from deadlocks in networks [7], [8].

The design of application specific NoCs has been explored in several works [24]-[30]. None of these works address the issue of message-level deadlock avoidance, which is critical for proper system operation. Avoiding routing-dependent deadlocks for mesh topologies has been considered in [24]. Avoiding routing deadlocks for custom NoC topologies have been presented in [22], [31], [32].

The use of logically separated networks to avoid message-dependent deadlocks has been utilized in several industrial multi-processors, such as [14]-[17]. The use of physically separated networks to remove message-dependent deadlocks is used in many designs, such as [20], [21]. In [5], message-level deadlock freedom is achieved by a different mechanism than using logically or physically separated networks. It utilizes an end-to-end flow control scheme, which ensures that messages are sent from the sender only when the receiver has enough buffering resources to store them. This is coupled together with a network design that uses time division multiplexing to divide the network resources among the various communicating elements, providing guaranteed throughput to connections. This leads to buffering free network for such connections and removal from message-level deadlocks. The deadlock avoidance mechanism using their protocol is presented in [23]. As we target general NoC designs that need not support such end-to-end flow control mechanisms, we do not compare such a scheme with our method presented here.

3 Topology Synthesis with Message-Level Deadlock Freedom

We implement our message-dependent deadlock-free path selection routine as a plug-in to an established NoC topology synthesis flow. We assume that the application kernels are parallelized and mapped onto different processors and hardware cores using existing tools, as done in

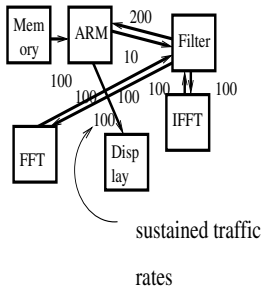


Figure 1. Example filter application

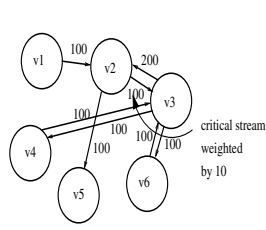


Figure 2. Core graph with sustained rates and critical streams

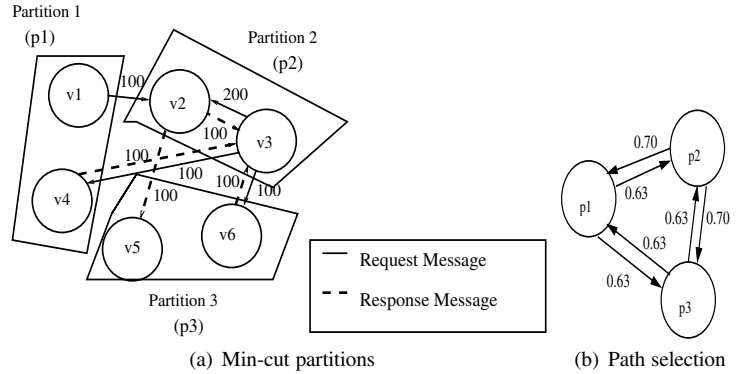


Figure 3. Algorithm Examples

earlier works [24]-[32]. The communication traffic flow between the various cores is represented by a core graph, which is taken as the input to the topology synthesis flow. The core graph for a small filter example (Figure 1) is shown in Figure 2. The edges of the core graph are annotated with the sustained rate of traffic flow, multiplied by the criticality level of the flow, as done in [26].

Before presenting the message-dependent deadlock-free path selection routine, we first present the basic topology synthesis flow. In the topology synthesis procedure (Algorithm 1), we synthesize several topologies: starting from a topology where all the cores are connected to a single switch to a topology where each core is connected to a separate switch. For the chosen switch count, the input core graph is partitioned into those many min-cut partitions (refer to step 2 of Algorithm 1). At this point, the communication traffic flows within a partition has been resolved.

Now, we integrate in the main flow the core contribution of this work (in step 4 of Algorithm 1), i.e. an algorithm (*PATH_COMPUTE*) that maps the communication flows to physical paths while guaranteeing deadlock freedom. This algorithm is explained in detail in the following paragraphs. Once the paths for a topology are selected, Algorithm 1 resumes, where the design area, power consumption and wire-length for the topologies are obtained. Then, the topology that best optimizes the user objectives and satisfies all the design constraints is chosen. The topology synthesis flow, without considering freedom from message level deadlocks, has been presented by us in detail in [32].

Now, we explain the path selection mechanism (Algorithm 2) that guarantees message-dependent deadlock-free operation of the NoC. In the first step of the *PATH_COMPUTE* algorithm, the flows are ordered in decreasing rate requirements, such that bigger flows are assigned first. The heuristic of assigning bigger flows first has been shown to provide better results (such as lower power consumption and more easily satisfying bandwidth constraints) in several earlier works [25], [31]. Then, for each flow in order, we first evaluate the message type of the flow (step 2 of Algorithm 2). The message types can either be fed explicitly by the user, or can be implicitly considered by the tool. As an example for implicitly considering the type, in shared memory systems, all the traffic flows that originate from processors and terminate into memory devices are of request type. While those that originate from the memories and terminate in the processors are

Algorithm 1 Topology Design Algorithm

- 1: Vary the number of switches in the design from 1 to the total number of cores in the design. Repeat steps 2 to 7 for each switch count.
 - 2: For the chosen switch count, find that many min-cut partitions of the communication graph. Cores in each partition are attached to the same switch.
 - 3: Check for bandwidth constraint violations when establishing the switches. The bandwidth of each link is the product of the NoC operating frequency and link width, which are inputs to the flow.
 - 4: Find the connectivity between the switches using the function *PATH_COMPUTE* (presented in Algorithm 2).
 - 5: Evaluate the switch power consumption and average hop-delay based on the selected paths.
 - 6: Perform floorplan of the design. Obtain design area, wire-lengths. Check for timing violations on the wires and evaluate the power consumption on wires.
 - 7: If solution minimizes objective, satisfies all constraints, note the design point and the topology.
 - 8: Choose the best topology and design point based on the user objectives.
-

of response type. Note that in shared memory systems, all inter-processor communication occur through the memory devices. Note that, if the connection between any pair of cores constitutes multiple message types, then each message type needs to be treated as a separate traffic flow.

Next, we evaluate the amount of power that will be dissipated across each of the switches, if the traffic for the chosen flow uses that switch (steps 3-5 of Algorithm 2). This power dissipation value on each switch depends on the size of the switch, the amount of traffic already routed on the switch and the frequency of operation. It also depends on how the switch is reached (from which other switch) and whether an already existing physical channel will be used to reach the switch or a new physical channel will have to be opened. The last information is needed, because opening a new physical channel increases the switch sizes and hence the power consumption of this flow and others that are routed through the switch.

In our NoC architecture, we permit the instantiation of multiple physical links between any two switches. When finding whether a switch is reachable from another switch for the current traffic flow, we evaluate whether any physical

links between the switches have already been established. If so, we see the message type of the traffic flows that have already been routed on the links. From the set of established links, we choose a link that supports the same message type as the current traffic flow and has enough bandwidth available to support the current flow. If no such link is available between the switches, we evaluate the cost of opening up a physical link for the current traffic flow.

The process of evaluating the power consumption for the current traffic flow is repeated for all pairs of switches. Finally (in step 6 of Algorithm 2), the set of links from the source to destination of the flow that has the least power consumption is chosen. When choosing the paths, routing-dependent deadlocks are also avoided by applying an existing method from [12]. Now physical connections are actually established on the chosen path and the message type of the current flow is assigned to the links that have been used for the flow.

Algorithm 2 PATH_COMPUTE

- 1: For each traffic flow in decreasing order of the bandwidth requirements, perform steps 2 to 6.
 - 2: Find the message type supported by the chosen traffic flow.
 - 3: For $i1$ from 1 to number of switches in the current design and $j1$ from 1 to number of switches in the current design, repeat steps 4 and 5.
 - 4: If one or more physical links exists between the switches $i1$ and $j1$, evaluate whether any link exists that has already been supporting the current message type & has bandwidth to support the current flow. If so, find the marginal power consumption to re-use this existing link.
 - 5: Else find the marginal power consumption for opening and using the link for this traffic flow.
 - 6: Find the least cost path (path with least power consumption) across the switches. For any links that were newly established for this traffic flow, associate the message type of this flow to the links. When selecting paths, choose only those paths that have turns not prohibited for removing routing-dependent deadlocks (based on the method from [12]).
 - 7: Return the chosen paths, new switch sizes, connectivity between switches and the type of message supported by each of the links.
-

Example 1 Let us consider the example from Figure 3(a). The input core graph has been partitioned into 4 partitions. We assume 2 different message types: request and response for the various traffic flows. Each partition p_i corresponds to the cores attached to the same switch. Let us consider routing the flow with a bandwidth value of 100 MB/S between the vertices $v1$ and $v2$, across the partitions $p1$ and $p2$. The traffic flow is of the message type request. Initially no physical paths have been established across any of the switches. If we have to route the flow across a link between any two switches, we have to first establish the link. The cost of routing the flow across any pair of switches is obtained. We annotate the edges between the switches by the cost (marginal increase in power consumption) of sending the traffic flow through the switches (Figure 3(b)). The cost on the edges from $p2$ are different from the others due to the difference in initial traffic rates within $p2$ when compared to the other switches. This is because, the switch $p2$ has to support flows between the vertices $v2$ and $v3$ within the partition. The least cost path for the flow, which is

across switches $p1$ and $p2$ is chosen. Now we have actually established a physical path and a link between these switches. We associate the message type request for this particular link. This is considered when routing the other flows and only those traffic flows that are of request type can use this particular physical link. We also note the size and switching activity of these switches that have changed due to the routing of the current flow.

4 Experimental Results

In this section, we present detailed experimental studies of our approach (which we further refer to as *INT-TOP* meaning message-dependent deadlock avoidance integrated with topology synthesis process) and compare it with traditional approaches:

(1) Using logically separate networks (*L-SEP*): In this scheme, we use separate buffers at each input, with as many buffers as the different message types, modeling the virtual channel based approach to remove message-dependent deadlocks.

(2) Using physically separate networks (*P-SEP*): In this scheme, we design physically different networks for each message type. For both these schemes we apply our topology synthesis procedure to obtain the network topologies.

(3) With a design that has no support to avoid message-dependent deadlocks (*ORIG*). Note that this base system cannot be employed in SoCs, as it cannot guarantee proper system operation. We present the experimental results for this scheme to only evaluate the overhead incurred in the other schemes to support deadlock-free operation.

4.1 Experimental Platform and Models

We utilize the \times pipes [36] NoC architecture to implement the synthesized topologies. We built accurate analytical models for the power consumption and area of the network components. To get the power estimates, the placement and routing of the components is performed using Cadence SoC Encounter [35] and accurate wire capacitances and resistances are obtained, as back-annotated information from the layout, with a 0.13μ technology library. The switching activity in the network components is varied by injecting functional traffic. The capacitance, resistance and the switching activity report are combined to estimate power consumption using Synposys PrimePower [34].

4.2 Comparison on SoC designs

We apply the deadlock prevention methods to five different SoC designs: *Multi-media system (MULT 30 cores)*, *Image Processing application (IMP-27 cores)*, *Video Processor (VPROC-42 cores)*, *MPEG4 decoder (12 cores)* and *Video Object Plane Decoder (VOPD-12 cores)*. The communication characteristics of some of these benchmarks is presented in [37]. There are two types of messages that are supported in each design: *request* and *response*. Each design consists of almost equal number of request and response traffic flows. This is because, every processor core communicates through the memory core, necessitating two-way communication (hence a request and response traffic flow) between the processors and memories. To make a fair comparison of the different schemes, we use the same synthesis approach and design constraints for synthesizing the topologies.

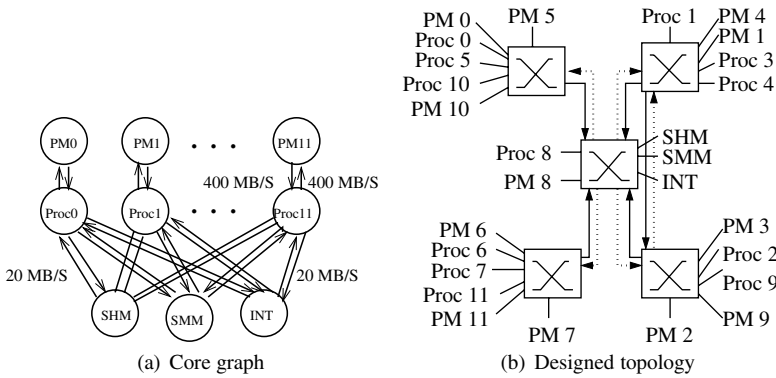


Figure 4. Core graph and designed topology for IMP

The communication pattern (core graph) for one of the applications (*IMP*) and the best synthesized topology for our proposed scheme (*INT-TOP*) are presented in Figures 4(a) and 4(b). The design consists of 12 processors (Proc 0 to Proc 11), a private memory for each processor (PM 0 to PM 11), a shared memory (SHM), a semaphore memory (SMM) and an interrupt device (INT). In the application, all communication from the processors are of request message type and communication to the processors are of response message type. In Figure 4(b), those links that support request message type are in bold and those links that support response message type are dashed.

The network power consumption, based on the functional traffic for the various designs using the different schemes is presented in Figure 5. As seen from this figure, the *INT-TOP* scheme presented in this work, outperforms the two conventional message-dependent deadlock avoidance schemes: *L-SEP* and *P-SEP*. Our proposed scheme leads to an average of 38.5% reduction in NoC power consumption when compared to the state-of-the-art deadlock avoidance schemes. When compared to our *INT-TOP* scheme, the *L-SEP* scheme requires large buffering requirements, as each virtual channel needs separate buffering resources. The *P-SEP* scheme requires more switches than the *INT-TOP* scheme, as the request and response messages utilize different networks. Interestingly, our proposed scheme incurs only a 2.5% increase in power consumption when compared to the *ORIG* scheme, where no message-dependent deadlock avoidance support is provided. This is mostly due to the efficient allocation of links to the different message types by our topology synthesis procedure. The switch area for the different schemes for the SoC designs, normalized with respect to the area of the base system (*ORIG*) is presented in Figure 6. The proposed method results in an average of 30.68% reduction in area when compared to the state-of-the-art schemes.

4.3 Effect of Different Number of Message Types

In this sub-section, we examine the power consumption of the proposed scheme, when the number of different message types is varied. The number of message types in a system depends on the underlying computation architecture. Cache coherent systems typically support several different

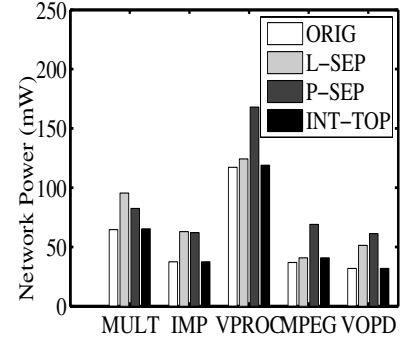


Figure 5. Power consumption of different schemes

message types. As an example, the S-1 multi-processor supports 4 different message types [18] and each type must be mapped onto different resources in the network. In [17], a more sophisticated protocol is used, which leads to seven different message types. To see the impact on the number of different message types, we created a synthetic benchmark having the traffic characteristics of the *VPROC* design. In this benchmark, around 80 different traffic flows exist, each one representing a message. We fixed the number of messages and varied the number of message types in the design from 1 to 7. The network power consumption for our proposed scheme, for the different number of message types is presented in Figure 7. This figure shows that our proposed scheme results in efficient designs, even for a large number of message types. Moreover, the rise in power consumption with an increasing number of message types saturates (designs with 6 and 7 message types have nearly the same power consumption), as most messages are already mapped onto unique links in the network.

4.4 Frequency Trade-offs

The algorithm presented here can also be used to perform frequency selection for a certain design. In this case, the frequency of operation of the NoC can be varied and the best topology can be synthesized for each frequency point. A higher operating frequency results in links having more bandwidth. Thus a smaller NoC can satisfy the design constraints. A trade-off curve for frequency vs power consumption of the network for the *VPROC* is presented in Figure 8. From such a curve, the most power-efficient operating frequency can be chosen for the design.

5 Conclusions

For *Networks on Chips (NoCs)* to be used in industrial designs, NoCs should guarantee proper system operation under all conditions. Achieving deadlock-free operation of the network with minimum power consumption and area overhead is critical for application-specific NoCs. In this work, we have focused on addressing the major issue of avoiding message-dependent deadlocks during the network operation. We have shown that by mapping the

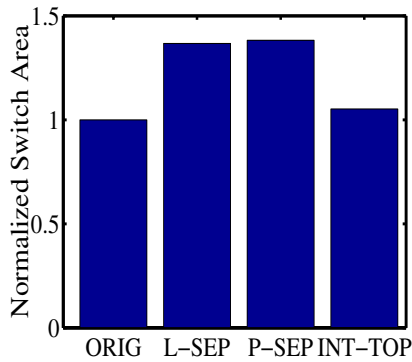


Figure 6. Normalized switch area for the different schemes

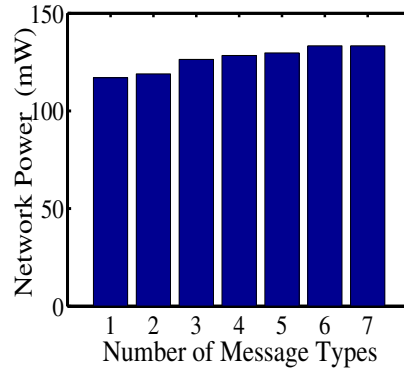


Figure 7. Effect of number of message types

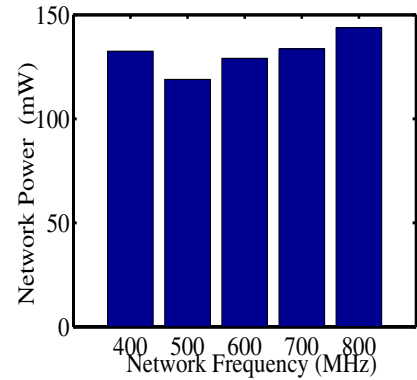


Figure 8. Power consumption variation with frequency

different message types onto different network resources during the topology mapping and synthesis phase, we can achieve large reductions in network power consumption and network area when compared to the state-of-the-art approaches. In future work, we plan to compare deadlock recovery schemes with the proposed scheme for NoCs.

6 Acknowledgments

This work is supported by the US National Science Foundation (NSF, contract CCR-0305718) for Stanford University. It is also supported by the Swiss National Science Foundation (FNS, Grant 20021-109450/1) and the Spanish Government Research Grant TIN2005-5619. The work is also supported by a grant from Semiconductor Research Corporation (SRC project number 1188) and a grant by STMicroelectronics for DEIS.

References

- [1] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm", *IEEE Computers*, pp. 70-78, Jan. 2002.
- [2] M. Sgroi et al., "Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design", *Proc. DAC*, pp. 667-672, June 2001.
- [3] S. Kumar et al., "A Network on Chip Architecture and Design Methodology", *Proc. ISVLSI*, pp. 117-122, April 2002.
- [4] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections", *Proc. DATE*, pp. 250-256, March 2000.
- [5] K. Goossens et al., "The Aethereal network on chip: Concepts, architectures, and implementations", *IEEE Design and Test of Computers*, Vol. 22(5), pp. 21-31, Sept-Oct 2005.
- [6] W. Dally, B. Towles, "Route Packets, not Wires: On-Chip Interconnection Networks", *Proc. DAC*, pp. 684-689, June 2001.
- [7] Y. H. Song, T. M. Pinkston, "A Progressive Approach to Handling Message-Dependent Deadlock in Parallel Computer Systems", *IEEE TPDS*, Vol. 14(3), pp. 259-275, March 2003.
- [8] Y. Choi, "Deadlock Recovery Based Router Architectures for High Performance Networks", PhD Dissertation, University of Southern California, June 2001.
- [9] G. Chiu, "The Odd-Even Turn Model for Adaptive Routing", *IEEE TPDS*, Vol. 11(7), pp. 729-738, July 2000.
- [10] C. Glass, L. Ni, "The turn model for adaptive routing", *Proc. ISCA*, pp. 278-287, 1992.
- [11] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks", *IEEE TPDS*, Vol. 8(8), pp. 790-802, Aug 1997.
- [12] D. Starobinski et al., "Application of network calculus to general topologies using turn-prohibition", *IEEE/ACM Transactions on Networking*, Vol. 11, Issue 3, pp. 411-421, June 2003.
- [13] W. J. Dally, H. Aoki, "Deadlock-Free Adaptive Routing in Multi-computer Networks Using Virtual Channels", *IEEE TPDS*, Vol. 4(4), pp. 466-475, April 1993.

- [14] S. Scott, G. Thorson, "Optimized Routing in the Cray T3D", *Proc. Workshop Parallel Computer Routing and Comm.*, pp. 281-294, May 1994.
- [15] S. Scott, G. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus", *Proc. Symp. Hot Interconnects IV*, pp. 147-156, Aug. 1996.
- [16] J. Carbonaro, Cavallino, "The Teraflops Router and NIC", *Proc. Symp. Hot Interconnects IV*, pp. 157-160, Aug. 1996.
- [17] S.S. Mukherjee et al., "The Alpha 21364 Network Architecture", *Proc. Symp. HOT Interconnects 9*, pp. 113-117, Aug. 2001.
- [18] L. Widdoes, S. Correll, "The S-1 Project: Developing High Performance Computers", *Proc. COMPCON*, pp. 282-291, Spring 1980.
- [19] "<http://www.st.com/stonline/prodpres/dedicate/soc/cores/stbus.htm>".
- [20] J. Laudon, D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server", *Proc. ISCA*, pp. 241-251, June 1997.
- [21] D. Lenoski et al., "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor", *Proc. ISCA*, pp. 148-159, 1990.
- [22] A. Hansson, K. Goossens, A. Radulescu, "UMARS: A Unified Approach to Mapping and Routing on a Combined Guaranteed Service and Best-Effort Network-on-Chip Architecture", Technical Report 2005/00340, Philips Research, April 2005.
- [23] B. Gebremichael et al., "Deadlock Prevention in the Aethereal Protocol", *Proc. Working Conference on Correct Hardware Design and Verification Methods (CHARME)*, Oct 2005.
- [24] J. Hu, R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures", *Proc. DATE*, March 2003.
- [25] S. Murali, G. De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs", *Proc. DAC* 2004.
- [26] S. Murali et al., "Mapping and Physical Planning of Networks on Chip Architectures with Quality-of-Service Guarantees", *Proc. ASPDAC* 2005.
- [27] A. Pinto et al., "Efficient Synthesis of Networks on Chip", *ICCD* 2003, pp. 146-150, Oct 2003.
- [28] W.H. Ho, T.M. Pinkston, "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns", *HPCA* 2003, pp. 377-388, Feb 2003.
- [29] T. Ahonen et al., "Topology Optimization for Application Specific Networks on Chip", *Proc. SLIP* 04.
- [30] K. Srinivasan et al., "An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks", *Proc. ICCAD '05*.
- [31] A. Hansson et al., "A unified approach to constrained mapping and routing on network-on-chip architectures", pp. 75-80, *Proc. ISSS* 2005.
- [32] S. Murali et al., "Designing Application-Specific Networks on Chips using Floorplan Information", to appear in *ICCAD* 2006.
- [33] W. J. Dally, B. Towles, "Principles and Practices of Interconnection Networks", Morgan Kaufmann, Dec 2003.
- [34] "http://www.synopsys.com/products/power/primepower_ds.pdf".
- [35] "http://www.cadence.com/products/digital_lic/soc_encounter/index.aspx".
- [36] S. Stergiou et al., "xpipeLite: a Synthesis Oriented Design Library for Networks on Chips", pp. 1188-1193, *Proc. DATE* 2005.
- [37] D. Bertozzi et al., "NoC Synthesis Flow for Customized Domain Specific Multi-Processor Systems-on-Chip", *IEEE Transactions on Parallel and Distributed Systems*, Feb 2005.