Designing Efficient Irregular Networks for Heterogeneous Systems-on-Chip

Christian Neeb, Norbert Wehn
University of Kaiserslautern, Erwin-Schrödinger-Straße, 67663 Kaiserslautern, Germany
Phone: (+49) 631 205 3553, Fax: (+49) 631 205 4437 E-mail: {neeb, wehn}@eit.uni-kl.de

***Abstract*— Networks-on-Chip will serve as the central integration platform in future complex SoC designs, composed of a large number of heterogeneous processing resources. Most researchers advocate the use of traditional regular networks like meshes, tori or trees as architectural templates which gained a high popularity in general-purpose parallel computing.**

However, most SoC platforms are special-purpose tailored to the domain-specific requirements of their application. They are usually built from a large diversity of heterogeneous components which communicate in a very specific, mostly irregular way. In this work, we propose a methodology for the design of customized irregular networks-on-chip, called *INoC*. We take advantage of a priori knowledge of the applications communication characteristic to generate an optimized network topology and routing algorithm. We show that customized irregular networks are clearly superior to traditional regular architectures in terms of performance at comparable implementation costs for irregular workloads. Even more, they inherently offer true scalability and expansibility which can normally not be accomplished by traditional approaches.

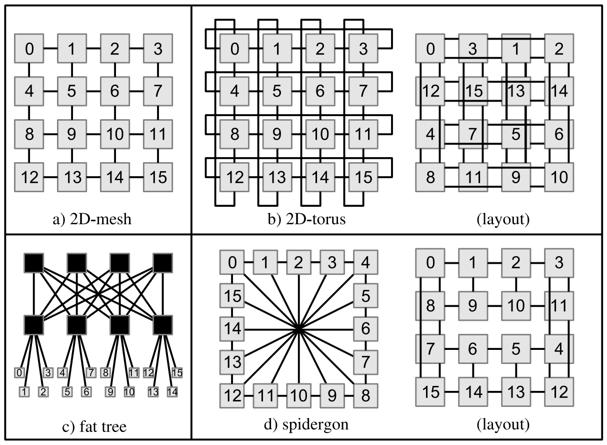


Fig. 1. Network-on-Chip topologies

I. Introduction

Due to the continued shrinking of feature sizes in today's silicon technologies, the integration of complex systems-on-chip (SoC) has become feasible today, offering a tremendous amount of computational power. In order to tackle design complexity and facilitate reuse, these systems are typically built from predesigned and preverified heterogeneous building blocks like programmable RISC cores, DSPs, dedicated hardware accelerators, memory blocks etc. which are plugged together in a domain specific integration platform. These designs typically show a high degree of modularity and inherent computational parallelism. This trend is accompanied by revolutionary changes of the employed design methodology where a paradigm shift from a computation-centric view to a communication-centric becomes evident [1]. Functionality of such systems is often captured by a set of communicating tasks at a high level of abstraction. These are mapped to computational resources which are then interconnected by an central communication backbone. The efficiency of the overall design is governed by this communication architecture which plays a key role in modern SoC platforms. It impacts both performance and implementation costs in terms of silicon area and energy consumption to a substantial extent.

There has been a lively discussion among system engineers and researches how to find an adequate network-on-chip (NoC) architecture that efficiently accommodates the applications communication needs. A vast range of NoC architectures has been proposed based on regular building patterns like meshes, tori, k-ary n-cubes or fat trees (Figure 1) for the implementation of on-chip networks to overcome traditional bus-based designs with all their known limitations [12]. The network topology determines how the different computational blocks are physically interconnected and thus has a great impact on performance and implementation costs. The routing function is tightly coupled to the underlying topology, defining the set of allowed paths on which packets may travel from a sender to the target node. The proper selection of the adequate topology and routing function form key decisions in the design of a network architecture.

The efficiency of the selected topology and routing function directly depends on the ability to handle the specific communication traffic which usually varies considerably between different pairs of resources. Many well-known parallel implementations of mathematical algorithms like the FFT or matrix multiplications exhibit a very regular communication scheme. Hypercubes or systolic arrays proved to be the best choice for highly efficient implementations in these cases. However, the aforementioned diversity of processing elements employed in most SoC architectures lead to communication patterns which are expected to be highly irregular with varying temporal and spatial intensity. If the network has insufficient bandwidth, which can be the case for only a small local region, the overall application might suffer poor performance. On the other hand, if the network offers much more bandwidth than that which is required, chip area and energy is wasted.

We therefore expect networks with irregular topology which are tailored to the applications requirements to be superior compared to the proposals promoting regular structures. As irregular networks are not built from fixed construction patterns like meshes they enable much more flexible interconnection schemes. Further, they offer true scalability and incremental expansion capabilities for practically any number of processing nodes.

However, a key problem is to ensure that no deadlock-situations can block the whole network by its routing algorithm. This phenomenon has extensively been analyzed in the past years primarily for regular topologies [5], [15], [7]. Such situations arise from cyclic wait dependencies caused by typical flow-control schemes in order to prevent buffer-overflow. Deadlock-avoidance based approaches have traditionally been preferred over deadlock-recovery schemes [21], [27]. As former approaches guarantee that no deadlocked networks states can be reached by severe restrictions of the routing function, the latter must handle these situations during runtime and are usually based on heuristic methods. We expect avoidance-based approaches to be the better choice for SoC architectures as they lead to more predictable solutions compared to deadlock-recovery schemes. Traditionally, the proof of deadlock-freedom has mostly been carried out on the assumption of the regular construction pattern and is thus far more complicated in irregular networks with arbitrary structure.

In this paper, we promote the design of application-specific irregular networks, called *INoCs*, based on a joint construction of both topology and deadlock-free routing algorithm in unity. We consider both aspects to be tightly coupled that lead to suboptimal solutions when optimized in isolation. Section II briefly summarizes the related work. We utilize a *resource communication graph* in section III to capture the communication pattern of an application that has already been mapped to dedicated hardware resources. Section IV gives basic definitions and assumption on our INoC-architecture. Based on this, section V defines a deadlock-free routing scheme applicable for irregular topologies which are generated in section VI using a greedy growing algorithm. Section VI summarizes some experimental results followed by a brief conclusion of this work.

II. Related Work

Kumar et al. [13] proposed a 2D-mesh based NoC architecture called *CLICHE* (Chip-Level Integration of Communicating Heterogeneous Elements). Based on this architecture, the same authors presented the Nostrum NoC [17] supporting multiple communication services modeled by a protocol stack. Due to the small router nodes and the straight forward VLSI layout of the network, a mesh topology was chosen as the candidate communication backbone.

In [6], Dally and Towles advocate a NoC architecture based on a folded torus. This topology has twice the wire demand and twice the bisection bandwidth of a mesh network. Further, wide channels of almost 300-bit are used taking into account the high wiring resources which are available on-Chip.

Guerrier and Greiner presented the SPIN architecture (Scalable, Programmable, Integrated Network) in [8] as a generic interconnection template. SPIN is based on a fat-tree topology where every node has four children and the parent is duplicated four times at any level of the tree. The resource nodes form the leaves of the tree which are interconnected by parent switch nodes.

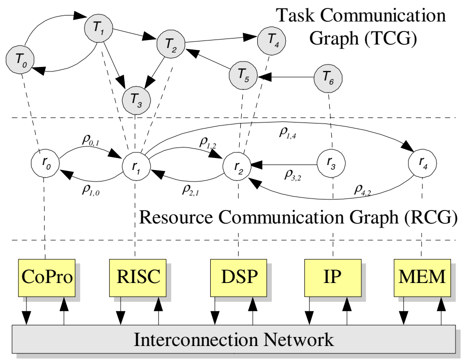
Spidergon was proposed by Coppola et al. [4] as an innovative interconnection topology, which claims to deliver significant cost/performance advantages compared to other NoC topologies. Here, the IP blocks are organized in a bidirectional ring where additional channels connect the direct diagonal counterparts in the network to decrease the length of the routing paths. Spidergon is a node-symmetric topology where all nodes are of degree three, and thus only offers a limited amount of scalability.

Opposed to these architectures, Murali and DeMicheli present an algorithm that maps processing cores onto a mesh NoC architecture under bandwidth constraints [18]. Starting from a weighted communication graph, they heuristically derive a near optimal placement of the individual cores inside the mesh aiming to minimize traffic congestion on its channels. This problem is known as the graph embedding problem which is NP-hard and, hence, does not readily yield itself to theoretical solutions [24].

In [19] the same authors extend their work by the introduction of the SUNMAP-tool where a larger set of candidate topologies like torus, hypercube, clos and butterfly are evaluated under delay, area and energy constraints. The tool automatically selects the best topology for a certain application by means of exhaustive search.

In contrast to the former approaches, where a rather small set of regular network topologies is considered, [14] investigates a filtering approach of randomly generated topologies. A large set of random graphs is generated and passed to a bank of filters like diameter-, embeddability- or fault tolerance filter. Depending on the filtering thresholds, a subset of graphs is identified which has the desired characteristics with respect to the specified requirements. The authors employed a simple mapping heuristic based on simulated annealing to perform the actual embedding in a second step.

However, all approaches emphasize the topological characteristics and the choice of the routing function only plays a subordinate role. Many efficient routing algorithms have been developed for meshes, tori or hypercubes in the last two decades. In contrast to that, only a small number of efficient deadlock-free routing algorithms are known for irregular networks, where nodes can be interconnected almost arbitrarily. The most promising schemes have been employed in networks-of-workstations [3], [11] where the *up*/down** [25] and *adaptive-trail* routing [22] are the most common examples. Inspired by these approaches, we present a flexible, deadlock-free routing scheme that enables the joint design of topology and routing for an optimized application specific network architecture.



Task Communication Graph (TCG)
T_0 T_1 T_2 T_3 T_4 T_5 T_6
r_0 r_1 r_2 r_3 r_4
$\rho_{0,1}$ $\rho_{1,0}$ $\rho_{1,2}$ $\rho_{2,1}$ $\rho_{1,4}$ $\rho_{3,2}$ $\rho_{4,2}$
Resource Communication Graph (RCG)
CoPro
RISC
DSP
IP
MEM
Interconnection Network

Fig. 2. Modeling application-specific communication

III. Application Communication Model

Many researches promote the use of task graphs to model the behavior of complex SoC applications on an abstract level [2], [10]. The tasks T are mapped to a set of hardware resources R which communicate through unidirectional point-to-point abstract channels (Figure 2). In this work, we assume that this mapping process has already been carried out in a sensible way. We refer to the *resource communication graph (RCG)* as the primary input of our NoC generation approach to capture the communication of the allocated components in a SoC platform.

Definition 1 *The resource communication graph RCG(R,E) is a directed graph where a vertex r_i represents a hardware component and a directed weighted edge $e_{i,j}$ an abstract communication channel between the resources r_i and r_j. An associated edge-weight $\rho(e_{i,j}) = \rho_{i,j}$ represents the average communication rate measured in data units per clock cycle.*

We further say that the communication rate of a single resource is given as

$$\rho(r_i) = \frac{1}{\Delta_{out}(r_i)} \sum_{e \in E_{out}(r_i)} \rho(e) \leq 1,$$

where $\Delta_{out}(r_i) = |E_{out}(r_i)|$ being the number of edges leaving node r_i and the average communication rate as

$$\rho = \frac{1}{|R|} \sum_{r \in R} \rho(r).$$

To facilitate comparison of different *RCGs* and network performance measures, we normalize all communication rates $\rho_{i,j}$ to a reference bandwidth ρ_0, such that $0 \leq \rho \leq 1$, which usually equals the physical bandwidth of a single point-to-point network channel.

These communication graphs are easily obtained by means of static code analysis or simulation. They can be derived in an early design stage where no detailed knowledge of the target platform is available yet.

IV. The Basic INoC Architecture

We consider a packet-switched network architecture called *INoC*, where a resource is directly connected to a dedicated router for the use in on-chip networks. In general, the topology represents the most important characteristic of NoC architectures. It defines how router nodes are physically interconnected and thus has a predominant influence on network performance and implementation costs.

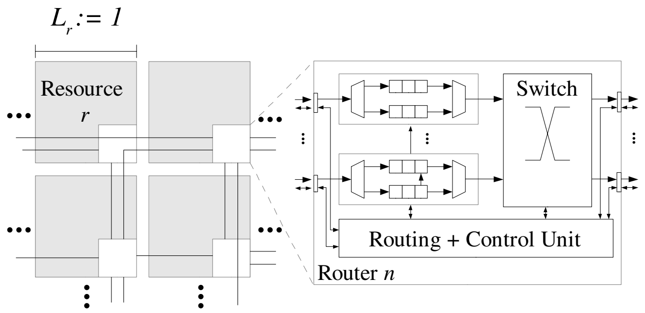


Fig. 3. Tile-based INoC with irregular topology and input-queued routers

Definition 2 *The interconnection topology is a directed graph $I(N,C)$, where a vertex $n_i \in N$ represents a router and a directed weighted edge $c_{i,j}$ a (physical) communication channel between its incident routers n_i and n_j. An associated edge-weight $\tau(c_{i,j}) = \tau_{i,j}$ represents the average communication traffic measured in data units per clock cycle.*

Given the network topology, the routing function defines the set of allowed paths on which a packet may travel from a sender towards its destination.

Definition 3 *A routing function $F_R : C \times N \rightarrow P(C)$ provides set of candidate out-channels in the power set of C, $C_{out} \in P(C)$, for a packet located in $c_{in} \in C$ destined to its destination node $n_d \in N$, such that $F_R(c_{in}, n_d) = C_{out}$ with $c_{in} \notin C_{out}$.*

If the set of out-going channels contains at most one channel, such that $|C_{out}| \leq 1 \forall (c_{in}, n_d)$, the routing function is denoted *deterministic*. If multiple alternative channels are provided, the routing is *adaptive*.

We deploy input-queued network routers, implementing a wormhole switching flow-control scheme [20], [23]. Here, packets are split into an arbitrary number of flits (flow control digit) and forwarded through the network in a pipelined fashion. If the header of the packet blocks somewhere on its way to the target, all the following flits block in place if the buffering space of the routers is exhausted. It enables the use of small buffers compared to other switching techniques like store-and-forward or virtual-cut-through. To prevent degradation of performance by head-of-line blocking, we support multiple virtual channels for the use with an extended routing scheme presented in section V.

We adopt a round-robin based scheme for switch arbitration in the router nodes called SLIP-scheduling [16], [9]. It provides fair bandwidth allocation while effectively preventing scheduling anomalies like starvation.

For chip layout, we assume a tile based floorplan in which each resource is mapped to one of the equally sized rectangular tiles of unit size length $L_R = 1$. Each tile is organized in a regular grid structure [13] which is independent of the chosen interconnect topology. It is worth mentioning that this assumption depicts an idealized picture in the context of heterogeneous SoCs and which is not a requirement of our approach. As we do not focus on floorplanning as a prime concern, it essentially simplifies the following ideas about the proposed network generation technique.

V. Routing in Irregular Topologies

Most traditional routing algorithms rely on the specific structure of the underlying topology and usually cannot be employed when the topology is changed. As already mentioned, one of the major restrictions in the design of efficient routing algorithms is the need to avoid network-deadlock. In such a state, no packet can make progress, because buffering resources in the adjacent routers are exhausted. These situations originate from cyclic wait dependencies that last forever as long as no further means are taken to break them. The notion of channel dependency, incorporated by a channel-dependency graph, and a necessary and sufficient condition for deadlock-freedom is given as [5], [7]:

Definition 4 *The channel dependency graph is a directed graph CDG(C,D), where a vertex c_i represents a channel of I and a directed edge a pair of channels (c_{in}, c_{out}) with a direct dependency from c_{in} to c_{out} if $c_{out} \in F_R(c_{in}, n)$ for any $c_{in} \in C$ and $n \in N$.*

Theorem 1 *A routing function F_R is deadlock-free if its related channel-dependency graph has no cycles.*

We will refer to this theorem to derive a set of routing constraints for a basic routing scheme employed in our *INoC* approach. The basic scheme is suitable for deterministic routing, but also offers limited degree of adaptivity. To increase routing freedom and adaptivity, we extend the basic scheme for the use of virtual channels.

A. Basic Scheme

To track channel dependencies caused by the applied routing function, we define an ordering among the nodes of routers N. Each node is given an index beginning from 0 up to $|N|-1$ and thus defining a permutation of N.

$$order(N) = (n_0, n_1, n_2, ..., n_{|N|-2}, n_{|N|-1})$$

To ensure that a routing path exists between arbitrary pairs of routers in the following, we assume that all nodes are bidirectionally connected following the established order above (see figure 4). This significantly simplifies the maintenance of the coherence of the routing function while the topology is modified later on. We will refer to this linearly ordered chain topology in the next section as the starting point of our network generation.

In a next step, we partition the channel set C into two disjoint subsets C_{inc} and C_{dec}: a channel is denoted an *increasing* channel $c \in C_{inc}$ if the incident source node is lower than its target node with respect to the assigned node index, and *decreasing*, $c \in C_{dec}$, in the opposite case.

$$C_{inc} = \{c_{i,j} | index(n_i) < index(n_j), \quad c_{i,j} \in C\}$$

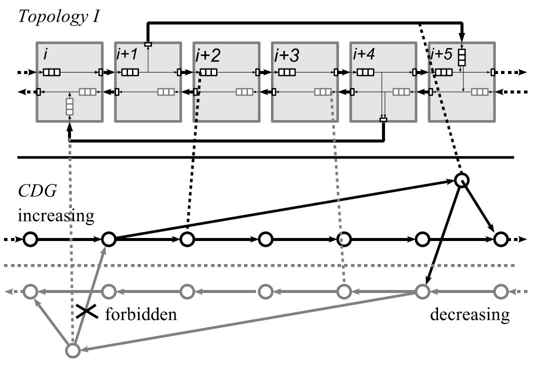


Fig. 4. Linear order of routers in the topology *I* and the related channel dependency graph *CDG* with forbidden circular dependency

$$C_{dec} = \{c_{i,j} | index(n_i) > index(n_j), \quad c_{i,j} \in C\}$$
$$C = C_{inc} \cup C_{dec}; \quad C_{inc} \cap C_{dec} = \{\}$$

As channels in the topology *I* are represented by nodes in the channel dependency graph *CDG* (definition 4), the same partitioning scheme remains valid for nodes in the *CDG*. We therefore can state the following theorem:

Theorem 2 *Each of the two node partitions comprising the node sets C_{inc} and C_{dec} in the CDG, always induces an acyclic subgraph CDG_{inc} and CDG_{dec}, respectively, for any given routing function F_R.*

This fact becomes obviously clear while trying to construct a cycle in one of the two subgraphs. Let's consider CDG_{inc} as an example. A dependency edge $d = (c_{in}, c_{out})$ caused by a routing function only belongs to it if its incident source node c_{in} and the target node c_{out} are both increasing. To form a cycle, at least one dependency is needed which targets a decreasing channel being incident to a router node with a lower order which is contradicting the given assumptions.

In order to make the *CDG* acyclic, all channel dependency edges connecting two nodes in opposite partitions must be disallowed in one of the two possible directions. Without loss of generality, we will restrict the routing in a way that no packet will transition from a decreasing to an increasing channel. In other words, a packet may travel on any path which consists of zero or more increasing channels followed by any number of decreasing channels afterwards. Once the packet is transferred on a decreasing channel, it is not allowed to change (back) to an increasing one.

Theorem 3 *A routing function F_R is deadlock-free if the related CDG contains no dependency edges $d = (c_{in}, c_{out})$ where $c_{in} \in C_{dec}$ and $c_{out} \in C_{inc}$.*

Referring to the example given in Figure 4, suppose router $i+4$ intends to forward a packet to $i+1$ via the shortcut channel to i and then to its target $i+1$. This would introduce a cyclic dependency from a decreasing channel to an increasing which, according to theorem 3, might lead to deadlock. In this case, the longer path via $i+3, i+2, i+1$ must be provided by the routing function.

It is worth mentioning that these rules correspond to those enforced by the *turn-model* for regular networks presented in

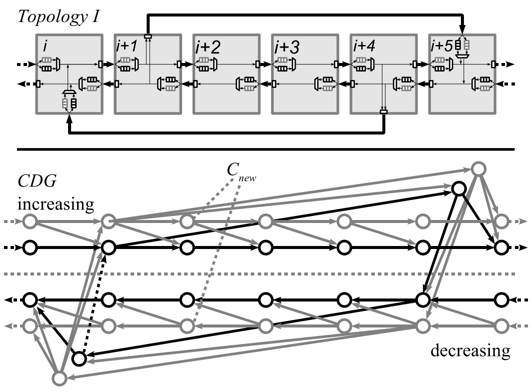


Fig. 5. Channel dependency graph *CDG* for the extended routing scheme using two virtual channels

[7]. But as we do not assume any specific topology here, the application of theorem 3 is more general.

B. Extended Scheme

It has been shown in [5], [15] that the constraints imposed by theorem 1 are overly restrictive for adaptive routing, forbidding more paths than necessary to avoid deadlock. As a consequence, a routing function F_R will most probably not always be able to supply paths of minimal length as shown in the previous section. This leads to a waste of physical bandwidth and transmission energy as more hops are required for a packet to reach its destination. Cyclic dependencies can explicitly be allowed as long as at least one routing path is provided to *escape* from them in any case. We extend our basic scheme by using virtual channels that can be used in an unrestricted way to increase adaptivity and to provide minimal paths inspired by [26]. Note that there is no principal difference between physical and virtual channels from the routing functions point of view. Every virtual channel might be realized by a physical channel as well.

We introduce a set of new channels C_{new} by splitting every physical channel into two or more virtual channels (gray nodes in Figure 5). Let C_{new} consist of all newly added channels, such that

$$C = C_{new} \cup C_{orig}; \quad C_{new} \cap C_{orig} = \{\}$$

$$with \quad C_{orig} = C_{inc} \cup C_{dec}$$

We maintain the basic routing function F_R^{orig} from section V-A defined on the original channel set C_{orig}. To exploit the increased routing flexibility offered by the new channels, we extend the F_R as follows: a packet arriving on a new channel may be routed to any channel without restrictions. If no new channels are available due to congestion one of the original channels must be provided. However, once a packet acquires an original channel it is not allowed to transition to a new channel anymore. This is summarized in the next theorem.

Theorem 4 *The extended routing function $F_R^{ext} : C \times N \rightarrow P(C)$ is deadlock-free if following conditions are fulfilled:*

- $|F_R^{ext}(c_{in}, n_d) \cap C_{orig}| \geq 1, \forall c_{in} \in C_{new}, n_d \in N$
- $F_R^{ext}(c_{in}, n_d) = F_R^{orig}(c_{in}, n_d), \forall c_{in} \in C_{orig}, n_d \in N$

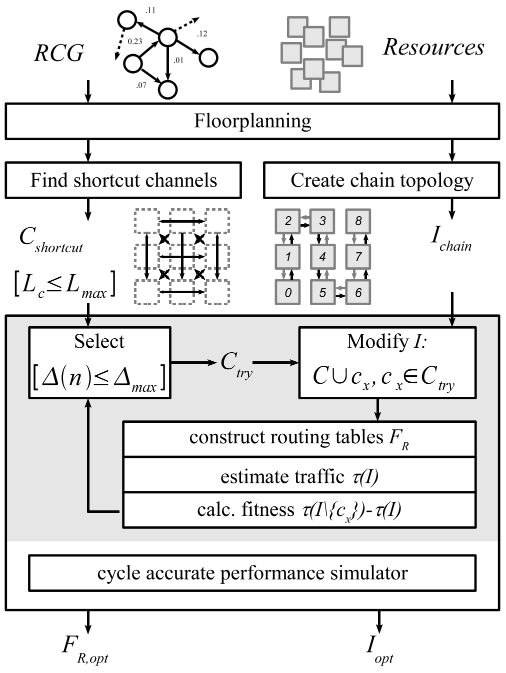


Fig. 6. Network construction flow using a greedy growing algorithm

Because the original routing function F_R^{orig} is proven to be deadlock-free, each packet has at least one safe path that is not involved in a cyclic dependency to reach its destination.

C. Constructing the Routing Tables

A restricted *Dijkstra shortest-path* algorithm is used to fill the routing tables regarding the rules above. The algorithm starts the traversal from every source node looking for all destination nodes. Every time the transition to the next channel is forbidden, the algorithms skips the traversal in that direction and so, guarantees that only allowed paths are found.

VI. Network Generation

The generic routing scheme presented so far allows us to incorporate a deadlock-free routing during the generation of a custom topology. This is crucial for estimating the traffic which is heavily impacted by the way packets are routed in the network.

A. Basic Idea

Referring to figure 6, the resource communication graph *RCG* is utilized to perform a floorplanning as a first step where positions of the hardware resources are determined. This is done by a simulated annealing heuristic placing frequently communicating resources next to each other. In the following, we exploit this layout information to constrain the length of channels that can be inserted in subsequent steps.

As already claimed in Section V-A, the network construction is started with a *bidirectional chain topology*, denoted as I_{chain}, in a way that all channels C_{chain} connect neighboring nodes with minimal length. The initial topology is strongly connected, and thus provides a path between every pair of nodes. This property will be retained throughout the generation process. We identify the set of candidate shortcut channels

$C_{shortcut}$ that might be added to the topology to increase its physical bandwidth. This set is constrained by a maximum channel length $L_{c,max}$ due to physical signaling delay and so prevents the algorithm from inserting wires that span long distances across the chip.

Starting from the initial topology, a greedy algorithm grows the topology by iteratively adding shortcuts. In the first iteration, all shortcut channels are considered in the set $C_{try} = C_{shortcut}$ by the growing algorithm. Each shortcut in C_{try} is temporarily added to the topology. If it exceeds a given maximum node-degree $\Delta_{max}(n)$ of either its source or target node, it is discarded. This constraint prevents the algorithm from instantiating slow routers with a large number of I/O-channels which would decrease the achievable clock frequency due to internal routing and scheduling delay of the router. In the presence of a new channel, the routing tables implementing F_R are constructed by following our basic INoC-routing scheme presented in section V.

To get an adequate fitness measure of the added channel, we evaluate the decrease of the resulting average network traffic $\tau(I)$ in the context of the given *RCG* communication pattern (see section VI-B). The higher the fitness, the better it supports the embedding of the *RCG* into the network. We store these results of all candidate shortcuts in a list which is sorted in decreasing order of fitness afterwards.

For the next iteration, we reuse this information to avoid the reevaluation of all remaining shortcuts. It is sufficient to restrict $C_{try} \subseteq C_{shortcut}$ to a subset which temporarily increases the locality of the search, and so, the performance of the greedy algorithm. When the number of elements in C_{try} becomes lower than a given threshold, it is extended again to include all remaining shortcuts. This is mandatory to get an (almost) optimal solution with minimum network traffic and so to impede the algorithm to get stuck in a local optimum.

To control the state of the network growth, we employ fast cycle accurate simulations to measure the performance in terms of network throughput and transmission latency. So we can stop growing the network at any point where the required bandwidth is provided to carry the application traffic. This offers a high degree of flexibility to trade-off cost and performance to find the most efficient solution.

B. Network Traffic Evaluation

To get fast performance estimates inside the main loop of the greedy algorithm, a multi commodity traffic flow analysis is carried out (algorithm 1). Each abstract communication, modeled as a *RCG* edge, is treated as an unique flow between any pair of resources. The routing algorithm maps all flows to routing paths in the network, increasing traffic $\tau(c_i)$ on the used channels by its rate ρ. This is illustrated in Figure 7. As the average rate ρ is fixed for a given *RCG*, lower traffic values indicate smaller routing distances. We found a good correlation of traffic estimates and the resulting network performance in general, proved by many simulations (see section VII-B).

Fig. 7. Evaluation of network traffic: long routing paths cause high channel traffic

Algorithm 1 Fast Traffic Estimation

```
input : $I$, $F_R$, $RCG$
output: $\tau(I); \tau(c_i)\,\forall c_i \in C$
$\tau(c) \leftarrow 0 \quad \forall c \in C$
**for all** $e \in edges(RCG)$ **do**
   $s \leftarrow resource(source(e));\ t \leftarrow resource(target(e))$
   $paths_{s\to t} \leftarrow route(s,t,F_R)$
  **for all** $path \in paths_{s\to t}$ **do**
    **for all** $c \in path$ **do**
      $\tau(c) = \tau(c) + \frac{\rho(e)}{|paths_{s\to t}|}$
    **end for**
  **end for**
**end for**
$\tau(I) \leftarrow \frac{1}{|C|}\sum_{c\in C}\tau(c)$
```

VII. Experimental Results

To evaluate the suitability of our INoC approach for the use in heterogeneous SoC platforms, we carried out various performance comparisons. We consider communication to be highly irregular caused by the diversity of hardware components in such platforms. In order to obtain a broad range of different irregular traffic scenarios, we randomly generated multiple *RCG* benchmark patterns for different network sizes N: we assumed that each resource communicates with $\sqrt{N}..2\sqrt{N}$ other components in average to provide a varying degree of communication locality. Further, each node shows different probabilities to select a target node. In this way the heterogeneity of the resources is taken into account where communication intensity varies between different pairs of nodes. Each resource injects one flit every four clock cycles into the network in average which yields a rate $\rho = \rho(r_i) = 0.25[flits/cycle], \forall r_i \in R$. We measured the accepted traffic in flits/cycle normalized to N which is the fraction of injected traffic, the network is able to handle. The maximum value is denoted as the network throughput. The flit latency determines the number of clock cycles it takes from entering the network until the reception at the target node. It is worth mentioning that these latency values do not include the source queuing time which is the time span, a flit waits at the source node in the case of a congested out-channel. Therefore, latency is expected to saturate towards high injection loads.

All data queues in the network routers are sized to buffer six flits per channel.

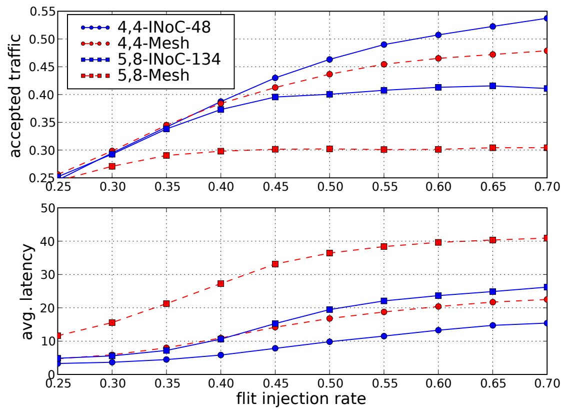


Fig. 8. Average performance of INoC vs. 2D-mesh networks with sizes $N = 16, 40$ over 100 irregular random traffic patterns

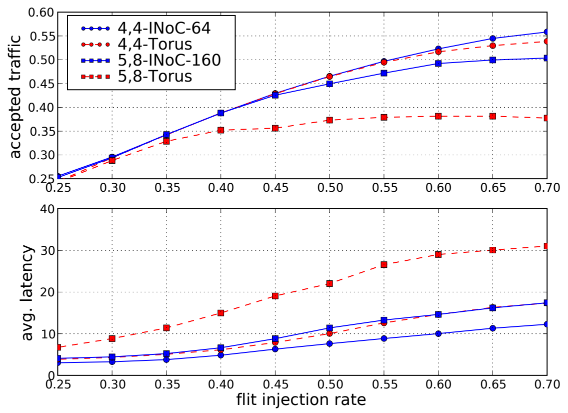


Fig. 9. Average performance of INoC vs. 2D-torus networks with sizes $N = 16, 40$ over 100 irregular random traffic patterns

A. Performance of INoC, Mesh, Torus and Spidergon

Figure 8-10 summarize the performance results averaged over 100 traffic patterns and as many generated INoCs compared to 2D-mesh, 2D-torus and spidergon networks. To allow a fair comparison, we employed a very similar mapping heuristic as presented in [18], [19] based on simulated annealing to map resources in the *RCG* onto the network topology *I*. A deterministic dimension-order routing algorithm was chosen for meshes and a scheme presented by Dally in [5] using two virtual channels for torus networks. These algorithms are the most common NoC implementations. For routing in the spidergon network, we chose the scheme proposed by [4]. Each generated INoC was optimized with respect to the applied traffic pattern, where the maximum channel length $L_{c,max} = 2L_r$ was set to be twice the size of a tile. We always inserted the same number of channels in the INoC as found in the mesh, torus and spidergon, respectively, i.e. 134 for comparison with a 5x8 mesh network comprising $N = 5 \cdot 8 = 40$ nodes. Thus, all networks offer the same physical bandwidth and so are comparable in terms of implementation costs. Here, the basic scheme from section V-A was employed for routing in INoCs. To get performance results in the vicinity of the communication requirements modeled in the *RCG*, we scale the injection rate ρ.

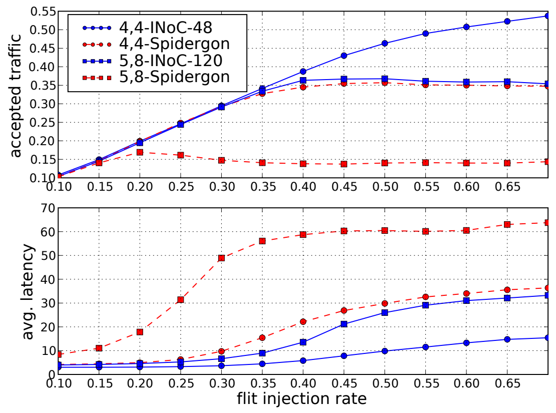


Fig. 10. Average performance of INoC vs. spidergon networks with sizes $N = 16, 40$ over 100 irregular random traffic patterns

Figure 8 to 10 show that optimized INoCs sustain a higher throughput and lower transmission latency in all cases. For small networks ($N = 16$), the performance difference is quite small and compared to a torus almost negligible. With increasing network size this gap grows significantly. A 40-node mesh needs almost twice the time to deliver a single flit at about 25% lower throughput compared to the INoC-134. This difference becomes even more distinct when comparing spidergon networks and INoCs in Figure 10. Our approach clearly outperforms the spidergon which saturates early at an injection rate of about 0.20 for $N = 40$ in average. In this case a spidergon network would most probably not deliver enough bandwidth to handle the application specific traffic.

B. INoC Scaling

To illustrate the advantage of the inherent true scalability and expansion capabilities of the INoCs, Figure 11 shows several stages of the generation process of a single INoC and its resulting performance. Figure 11a) depicts a typical run of the greedy algorithm which continuously lowers the traffic on the network channels as its cost function. Doing so, the algorithm minimizes the routing distances between any source/target pair of nodes given in the *RCG* by inserting the appropriate shortcut channels.

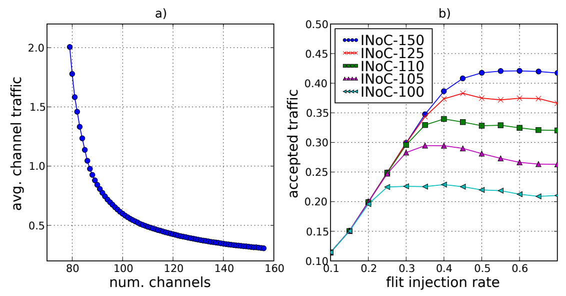


Fig. 11. Growing a 5x8 INoC: a) channel traffic, b) normalized throughput

This flexibility allows a fine-grained trade-off of performance and implementation costs and can not be achieved by most regular architectures in general.

C. Basic vs. Extended Routing Scheme

A comparison of the basic and extended routing scheme is shown in figure 12. Packet length is set to four flits and 32, respectively, for a 5x8 INoC. Two virtual channels are required for the extended scheme. For that, we split the buffering space of six flits in the basic scheme and distribute it equally among the two virtual channel queues. The extended routing scheme clearly outperforms the basic scheme with its limited degree of adaptivity, and so, enables higher throughput at substantially lower latency. Here, packets can bypass blocked packets located in different virtual channel queues and thus reduce head-of-line blocking.

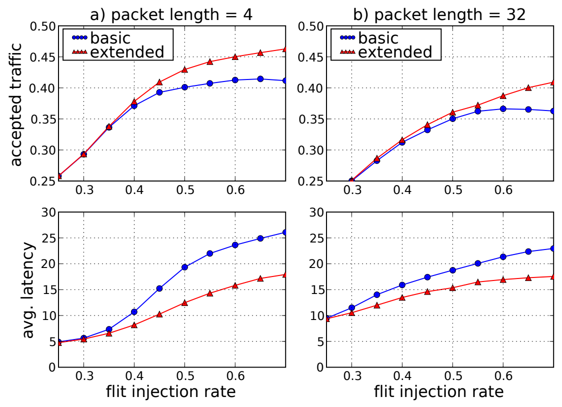


Fig. 12. Basic and extended routing scheme for a) 4 flit packets, b) 32 flit packets in a 5x8 INoC

VIII. Conclusion

We presented a generative approach for the construction of customized, irregular networks-on-chip (INoC). A simple greedy algorithm was implemented to tailor the network topology to the requirements of the application captured in a resource communication graph. We derived a set of constraints for routing in irregular networks to ensure that no deadlock can occur. This is an essential requirement which otherwise might lead to an overall system crash.

INoCs proved to be superior in terms of performance but even more due to their true scalability compared to traditional approaches. Motivated by the obtained results, we expect that irregular NoCs will deliver the required performance and flexibility in modern SoC design methodologies. INoCs are a very good match for heterogeneous architectures where the traffic workload is irregular and usually can be characterized to a large extent. We believe that the combined treatment of the routing algorithm and topology generation offers a huge potential of optimization for future application-specific NoC architectures.

References

[1] L. Benini and G. DeMicheli. Networks on chips: A new SoC paradigm. *IEEE Computer Vol. 1*, pages 70–78, 2002.

[2] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. DeMicheli. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. *IEEE Transactions on Parallel and Distributed Systems*, pages 229–244, Dec. 1997.

[3] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W.-K. Su. Myrinet - A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15:29–36, Feb. 1995.

[4] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra. Spidergon: a novel on-chip communication network. In *Proc. 2004 International Symposium on System-on-Chip (ISSOC 2004)*, page 15, Nov. 2004.

[5] W. Dally and C. Seitz. Deadlock-free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, pages 547–553, 1987.

[6] W. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proc. 2001 Design Automation Conference (DAC 2001)*, pages 683–689, 2001.

[7] C. Glass and L. Ni. The Turn Model for Adaptive Routing. In *Proc. 19th International Symposium on Computer Architecture*, pages 278–287, May 1992.

[8] P. Guerrier and A. Greiner. A Generic Architecture for On-Chip Packet-Switched Interconnections. In *Proc. 2000 Design, Automation and Test in Europe (DATE '00)*, pages 250–256, Mar. 2000.

[9] P. Gupta and N. McKeown. Designing and Implementing a Fast Crossbar Scheduler. *IEEE Micro*, 1:20–28, Jan. 1999.

[10] W. H. Ho and T. M. Pinkston. A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns. In *Proc of the 9th International Symposium on High-Performance Computer Architecture*, 2003.

[11] R. Horst. ServerNet Deadlock Avoidance and Fractahefral Topologies. In *Proc. 1996 International Parallel Processing Symp.*, pages 274–280, Apr. 1996.

[12] A. Ivanov and G. DeMicheli. The Network-on-Chip Paradigm in Practice and Research. *IEEE Design and Test of Computers*, pages 399–403, 2005.

[13] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani. A Network on Chip Architecture and Design Methodology. In *IEEE Computer Society Annual Symposium on VLSI*, pages 105–112, Apr. 2002.

[14] V. Lakamraju, I. Koren, and C. Krishna. Filtering Random Graphs to Synthesize Interconnection Networks with Multiple Objectives. *IEEE Transactions on Parallel and Distributed Systems*, 2002.

[15] X. Lin, P. McKinley, and L. Ni. The Message Flow Model for Routing in Wormhole-Routed Networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 755–760, 1995.

[16] N. McKeown. *Scheduling Algorithms for Input-queued Cell Switches*. PhD thesis, University of California, Berkeley, 1995.

[17] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed Bandwidth using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip. In *Proc. 2004 Design, Automation and Test in Europe (DATE '04)*, Mar. 2004.

[18] S. Murali and G. DeMicheli. Bandwidth-Constrained Mappings of Cores onto NoC Architectures. In *Proc. 2004 Design, Automation and Test in Europe (DATE '04)*, 2004.

[19] S. Murali and G. DeMicheli. SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs. In *Proc. 2004 Design Automation Conference (DAC 2004)*, pages 914–919, 2004.

[20] L. Ni and P. McKinley. A survey of Wormhole Routing Techniques in Direct Networks. *Computer*, 26:62–76, Feb. 1993.

[21] T. M. Pinkston and S. Warnakulasuriya. On Deadlocks in Interconnection Networks. In *Proc. of the 24th annual international symposium on Computer architecture*, 1997.

[22] W. Qiao, L. M. Ni, and T. Rokicki. Adaptive-Trail Routing and Performance Evaluation in Irregular Networks Using Cut-Through Switches. *IEEE Transactions on Parallel and Distributed Systems*, pages 1138–1158, Nov. 1999.

[23] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proc. 2003 Design Automation Conference (DAC 2003)*, 2003.

[24] J. Saxe. Embeddability of Graphs in K-Space Is Strongly NP-Hard. In *Proc. 17th Allerton Conf. Comm., Control and Computing*, pages 480–489, 1979.

[25] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterhwaite, and C. Thacker. Autonet: A high-speed, self-configuring local area network using point-to-point links. Technical report, Apr. 1990.

[26] F. Silla. *Routing and Flow Control in Networks of Workstations*. PhD thesis, University of Politécnia de Valencia, Spain, 1998.

[27] S. Warnakulasuriya and T. M. Pinkston. Characterization of Deadlocks in Irregular Networks. In *Proceedings of the 1999 International Conference on Parallel Processing*, 1999.