Application-Specific Network-on-Chip Architecture Customization via Long-Range Link Insertion

Umit Y. Ogras
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
e-mail: uogras@ece.cmu.edu

Radu Marculescu
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
e-mail: radum@ece.cmu.edu

Abstract — **Networks-on-Chip (NoCs) represent a promising solution to complex on-chip communication problems. The NoC communication architectures considered so far are based on either completely regular or fully customized topologies. In this paper, we present a methodology to automatically synthesize an architecture where a few application-specific long-range links are inserted on top of a regular mesh network. This way, we can better exploit the benefits of both complete regularity and partial customization. Indeed, our experimental results show that inserting application-specific long-range links significantly increases the critical traffic workload at which the network state transits from a free to a congested regime. This, in turn, results in a significant reduction in the average packet latency and a major improvement in the achievable network throughput.**

I. INTRODUCTION

Continuous scaling of CMOS technology makes it possible to put many heterogeneous devices on a single chip. Large-scale integration of these blocks onto a single chip calls for truly scalable Networks-on-Chip (NoC) communication architectures [1,2,3]. Regular NoC architectures based on grid-like (or 2D lattice) topologies provide well-controlled electrical parameters and reduced power consumption across the links. However, due to the lack of fast paths between remotely situated nodes, such architectures may suffer from long packet latencies. Indeed, having many hops between different communicating nodes, not only increases the message latency, but also increases the message blocking probability thus making the end-to-end packet latency more unpredictable. Consequently, such generic platforms may become easily less attractive for application-specific designs that need to guarantee a given level of performance.

On the other hand, fully customized topologies [8-11] can improve the overall network performance, but they distort the regularity of the grid structure. This results in links with widely varying lengths, performance and power consumption. Consequently, better *logical* connectivity comes at the expense of a penalty in the structured nature of the wiring which is anyway one of the main advantages offered by the regular on-chip networks [1]. In the extreme case, fully customized solutions may end up resembling ASIC-style designs where individual modules communicate by packet switching. Hence, the usual problems of cross-talk, timing closure, global wires *etc.* may undermine the overall gain obtainable through customization.

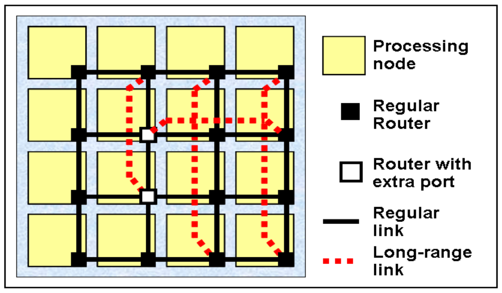


Figure 1. Illustration of adding long-range links to a 4x4 standard mesh network.

Fortunately, these two extreme points in the design space (*i.e.* designs based on purely regular or completely customized topologies) are *not* the only solutions possible for NoC architectures. In fact, it is interesting to note that many technological, biological, and social networks are neither completely regular, nor irregular [14,15]. One can view these networks as a superposition of clustered nodes with short links and a collection of random long-range links that produce "shortcuts" among different regions of the network. Regular lattice networks with a number of additional random long-range links, similar to the one shown in Figure 1, can be used to model such networks [15]. This paper explores precisely the potential of using standard mesh networks in conjunction with a few additional long-range links to improve the performance of regular NoCs.

Inserting long-range links to regular architectures clearly reduces the average distance between remote nodes. However, inserting long-range links cannot be done randomly for NoCs because adding extra links has a more pronounced, yet barely studied, impact on the *dynamic properties* of the network, which are characterized by traffic congestion. At low traffic loads, the average packet latency exhibits a weak dependence on the traffic injection rate. However, when the traffic injection rate exceeds a critical value, the packet delivery times rise abruptly and the network throughput starts collapsing (Figure 2). The state of the network before congestion (*i.e.* the region on left hand side of the critical value) is the *free state*, while the state beyond the critical value (right hand side) is said to be the *congested state*. Finally, the transition from the free state to the congested one is known as *phase transition* region.

As it turns out, the phase transition in regular networks can be significantly delayed by introducing additional long-range links (see Figure 2) [16]. Due to the exponential increase in the latency beyond the critical point, even a small right hand shift

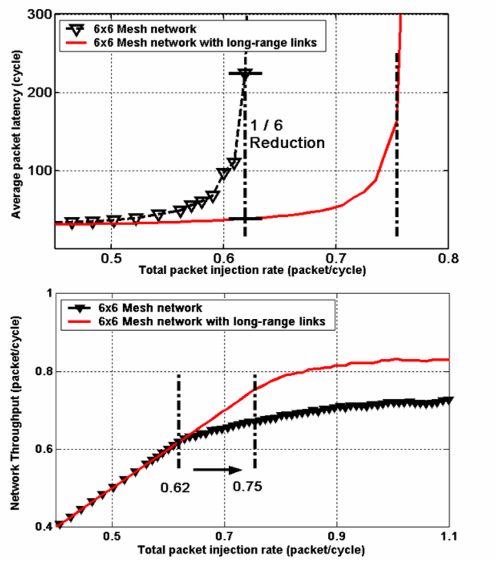


Figure 2. Shift in the critical traffic load after the insertion of long-range links to a 6x6 mesh network under hotspot traffic (Section V).

in the critical traffic value may result in orders of magnitude reduction for latency. Similarly, the achievable throughput grows with the right shift of the critical traffic value. This phenomenon is at the very heart of our optimization technique.

The main objective of our optimization technique is to boost the network performance (*i.e.* reduce the average packet latency and increase the network throughput) by maximizing the value of the critical traffic load via *smart insertion of application-specific long-range links*. To this end, our contribution is twofold:

• First, for a given application, we propose an algorithm that determines the most beneficial long-range links to be inserted in a mesh network.

• Second, we present a deadlock-free decentralized routing algorithm that exploits the long-range links to achieve the desired performance level.

The paper is organized as follows. Section II reviews related work. The proposed approach for smart link insertion is explained in Section III. Practical considerations in implementing long-range links are discussed in Section IV, while the experimental results appear in Section V. Finally, Section VI concludes the paper by summarizing our main contribution.

II. RELATED WORK

The use of NoCs as a scalable communication architecture is discussed in [1,2,3]. Design methodologies for application-specific NoCs are discussed in [4-11]. Studies in [6,7] consider regular network topologies and present algorithms for application mapping under different routing strategies. On the other hand, fully customized communication architecture synthesis for a given application is addressed in [8,9,10].

To the best of our knowledge, the idea of optimizing a generic grid-like network with application-specific long-range links is first addressed in this paper. Previous work on a similar idea comes from the networks theory side and uses very idealistic assumptions. For instance, the authors of [16] investigate the effect of adding random links to mesh and torus networks under uniform traffic. The packets in the network consist of a single atomic entity containing address information only. Also, due to the infinite buffer assumption, the authors do not deal with deadlock states explicitly.

In contrast to this prior work, we do consider wormhole routing with *arbitrary* flit sizes and network routers with *bounded* input buffers. Most importantly, instead of uniform traffic, we assume *application-specific* traffic patterns and present an algorithm which inserts the long-range links in a smart manner rather than randomly. Due to the bounded input buffers, additional long-range links may cause deadlock states. For this reason, we also present a deadlock-free routing algorithm that exploits the long-range links to achieve the desired performance boost.

III. LONG-RANGE LINK INSERTION ALGORITHM

We start by formulating the long-range link inserting problem. After that, we present the details of the solution following a top-down approach.

A. System model and basic assumptions

The system of interest consists of a set T of $m \times n$ tiles interconnected by a 2D mesh network[1]. The tiles of the network (referred to as *PE*s) are populated with processing and/or storage elements that communicate with each other via the network. We do not make any assumption about the distribution of the packet injection rate into the network, but only consider the frequencies at which the PEs communicate with each other.

Due to limited on-chip buffer resources and low latency requirements, it makes sense to assume wormhole switching for the network; the results derived here, however, are also applicable to packet- and virtual cut-through switching networks. Further, we do *not* assume any particular routing algorithm for the mesh network; the only requirement is that the underlying routing algorithm has to be deadlock-free and minimal. Deadlock-free property is desirable for on-chip networks for two reasons: First, implementing deadlock detection and recovery mechanisms is expensive in terms of silicon resources. Second, such mechanisms can cause unpredictable delays, which need to be avoided for most embedded applications.

After inserting the long-range links as explained in Section C, the routers without extra links simply use the default routing strategy. Since this default strategy cannot route the packets across the newly added links, we define a deadlock-free routing strategy which enables the use of the newly added long-range links (Section E).

B. Problem formulation

The communication volume between the PE located at tile $i \in T$ and the PE located at tile $j \in T$ is denoted by V_{ij}. We compute the frequency of communication, f_{ij}, between PEs i and j by normalizing the inter-tile communication volume as follows:

1. The following discussion assumes a 2D mesh but the proposed technique is applicable to other topologies for which a distance definition as in equations 6 and 7 (in Section III) exists.

$$\forall i, j, p, q \in T \quad f_{ij} = \frac{V_{ij}}{\sum_{p} \sum_{q \neq p} V_{pq}} \qquad \textbf{(1)}$$

The addition of long-range links introduces an overhead due to the additional wires and repeaters[1] connecting the wire segments to ensure the latency-insensitive operation. Hence, we need to model the cost of long-range links to have a measure of this overhead.

Without losing the generality, we measure the size of long-range links, $s(l)$, in multiples of basic link units which are identical to the regular links used in the mesh network. This is reasonable, since the long-range links consist of a number of standard links connected by repeaters, as shown in Figure 8. The number of repeaters required by a long-range link is given by $s(l) - 1$. Consequently, the maximum amount of permissible overhead can be expressed as a multiple of the standard link segments that make up the long-range link. For example, a resource constraint of *S* means that only long-range links consisting of at most *S* units of standard links, total, can be added.

We can now state the application-specific long-range link insertion problem as follows:

Given

• $f_{ij}\ \forall i, j \in T$

• Maximum number of links that can be added, *S*

• Routing strategy for the mesh network, *R*

Determine

• The set of long-range links to be added on top of the mesh network, L_S

• A deadlock-free routing strategy that governs the use of the newly added long-range links, R_L

such that

$$max(\lambda_c) \quad subject\ to \quad \sum_{l \in L_S} s(l) < S \qquad \textbf{(2)}$$

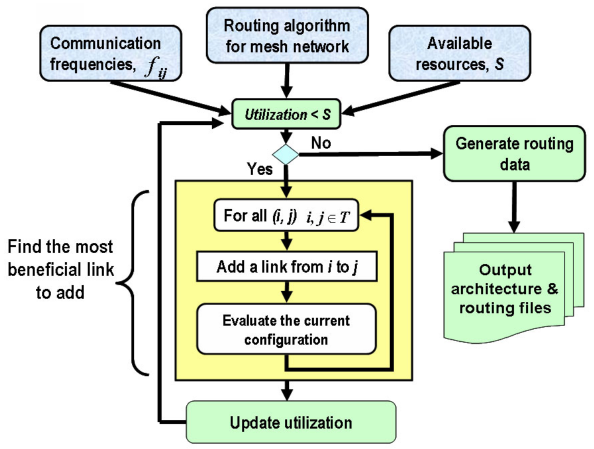
where λ_c is the *critical load* at which the network enters the congested phase. To give some intuition, the long-range links are added to maximize the critical traffic, λ_c, subject to the total amount of available resources; that is, the phase transition region is delayed beyond the value a standard mesh network (of exactly same size) can offer. Maximizing λ_c increases the achievable throughput and reduces the latency compared to the original critical load, as shown later in the experiments.

Note that, the objective of inserting long-range links is by no means limited to maximizing λ_c. Indeed, other objective functions (*e.g.* increased fault-tolerance, guaranteed service, *etc.*), can replace or augment the objective of maximizing λ_c in order to take the full advantage of the inserted links.

C. *Iterative link addition algorithm*

We propose an efficient iterative algorithm that inserts the most beneficial long-range links to the current configuration of the network, provided that the available resources are not used up yet. The link insertion algorithm is summarized in Figure 3.

1. In our terminology, the repeaters act primarily as storage elements (like FIFO buffers), as explained in Section IV in more detail.



Communication frequencies, f_{ij}
Routing algorithm for mesh network
Available resources, S
Utilization < S
No
Yes
Generate routing data
For all (i, j) $i, j \in T$
Add a link from i to j
Evaluate the current configuration
Find the most beneficial link to add
Output architecture & routing files
Update utilization

Figure 3. The flow of the long-range link insertion algorithm. The evaluation step is detailed in Section D.

The algorithm starts with a standard mesh network and takes the communication frequencies between the network tiles, the default routing algorithm and the amount of resources allowed to use as inputs. Then, the algorithm selects all possible pairs of tiles (*i.e.* $C(\|T\|, 2)$ pairs where $\|T\|$ is the number of nodes in the network) and inserts links between them. After inserting each long-range link, the resulting network is evaluated to find out the gain obtained over the previous configuration. Since our goal is to maximize λ_c, we compare different configurations in terms of critical traffic load as detailed in Section D. After the most beneficial long-range link is found, the information about this link is stored and the amount of utilized resources updated. For example, if a long-range link consisting of four equivalent segments of standard links is added, the utilization is incremented by four.

The procedure described above repeats until all available resources are used up. Once this happens, an architecture configuration file is generated. Then, the routing strategy governing the use of long-range links is produced and written to a routing configuration file, as described in Section E.

D. *Evaluation of the critical traffic value*

While the impact of routing strategy, switching techniques and network topology on the critical point have been studied via simulation [18], no work has been done to maximize the traffic critical value subject to resource constraints. The major obstacle in optimizing the critical load comes from the difficulty in modelling the variation of critical point as a function of the design decisions. Several theoreticians [16,17] estimate the criticality point using mean field models. However, unlike our work, these studies assume uniform traffic, infinite buffers, and the estimates are valid only for regular grids without long-range connections.

The key idea of our contribution is to reduce the estimation of critical point of the network to just *one* parameter that can be computed analytically, much faster than simulation. This is important since using very accurate estimates obtained through simulation would be simply too costly to use within an optimization loop. The optimization goal can still be achieved using the simple parameter, as long as the comparison between two network configurations matches the one with the critical load.

In the following, we relate λ_c to the *free packet delay* (τ_0) in the network, which can be efficiently computed. Let the number of messages in the network (at time t) be $N(t)$ and the aggregated packet injection rate be λ, *i.e.*

$$\lambda = \sum_{i \in T} \lambda_i \text{ , } \lambda_i\text{:}\textit{the injection rate of tile } i \in T$$

In the *free state* (*i.e.* when $\lambda < \lambda_c$), the network is in the steady-state, so the packet *injection* rate equals the packet *ejection* rate. As a result, we can equate the injection and ejection rates to obtain the following approximation:

$$\lambda \approx \frac{N_{ave}}{\tau_{ave}} \qquad (3)$$

where τ_{ave} is the *average time* each packet spends in the network, and $N_{ave} = \langle N(t) \rangle$ is the average number of packets in the network. The exact value of τ_{ave} is a function of the traffic injection rate, as well as topology, routing strategy, *etc.* While no exact analytical model is available in the literature, we observe that τ_{ave} shows a weak dependence on the traffic injection rate when the network is in the free state. Hence, τ_0 can be used to approximate τ_{ave}. If we denote the average number of packets in the network, at the onset of the criticality, by N^c_{ave}, we can write the following relation:

$$\lambda_c \approx \frac{N^c_{ave}}{\tau_0} \qquad (4)$$

This approximation is also an *upper bound* for the critical load λ_c, since $\tau_0 \le \tau_{ave}(\lambda_c)$. We note that $\lambda = N_{ave}/\tau_{ave}$ follows also Little's law. Indeed, other theoretical studies proposed to approximate $\lambda_c = N^c_{ave}/\tau_0$ using *mean field* [16] and *distance* models [17] under uniform traffic.

Given that the number of messages in the network, at the onset of the criticality, is bounded by its capacity, N^c_{ave}, the critical traffic load λ_c and the average packet latency are inversely proportional to each other. Indeed, if the average packet latency decreases, the phase transition point is delayed, as demonstrated in Figure 2, where the reduction in the latency is obtained due to the long-range links. Our optimization technique uses the relationship between λ_c and τ_{ave} to maximize the critical load. More specifically, we minimize τ_0 which can be efficiently computed in the optimization loop, as opposed to λ_c for which there is no known analytical result to date.

Experimental Verification of the Equations 3 and 4

For completeness, we verified experimentally both Equation 3 and Equation 4, as shown in Figure 4. The dotted line shows the actual packet injection rate (λ), for reference. The solid line with the square marker is obtained for a 8×8 network under the *hotspot* traffic, as the ratio between the average number of packets in the network and the average packet delay at that particular injection rate. From these plots, it can be clearly seen that there is a good agreement between the actual value obtained through simulation and the one predicted by Equation 3 *before* entering the criticality. Since the network is not in steady-state beyond the critical traffic value, the Equation 3 does not hold for higher injection rates. As mentioned before, the exact value of the average packet delay at a given load, $\tau(\lambda)$, can only be found by simulation. The dashed

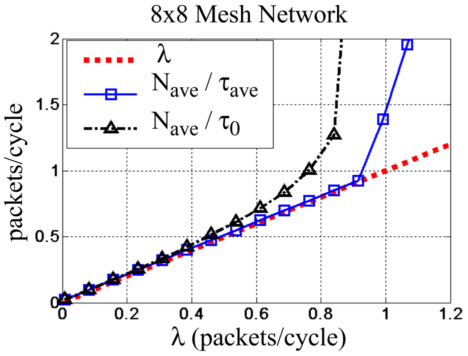


Figure 4. Experimental verification of Equation 3 and Equation 4 for a 8x8 mesh network.

line with triangular markers in Figure 4 illustrates the upper bound given in Equation 4. We observe that this expression provides a good approximation at lower data rates and holds the upper bound property.

Computation of τ_0

For arbitrary traffic patterns characterized by the communication frequencies f_{ij} $\forall i, j \in T$, τ_0 can be written as

$$\tau_0 = \sum_i \sum_{j \ne i} f_{ij} \left[d(i,j)(t_r + t_s + t_w) + max(t_s + t_w) \left\lceil \frac{L}{W} \right\rceil \right] \qquad (5)$$

where $d(i,j)$ is the *distance* from routers i to router j, and t_r, t_s, t_w are the is architectural parameters representing time to make the routing decision, traverse the switch and the link, respectively. Finally, L is the length of the packet, while W is the width of the network channel.

For the standard mesh network, the *Manhattan distance* (d_M) is used to compute $d(i,j)$, *i.e.*

$$d_M(i,j) = |i_x - j_x| + |i_y - j_y| \qquad (6)$$

where subscripts x and y denote the x-y coordinates, respectively. For the routers with long-range links, an *extended distance* definition is needed in order to take the long-range connections into account. Hence, we use the following generalized definition:

$$d(i,j) = \begin{cases} d_M(i,j) & \text{if no long-range link is attached to } i \\ min(d_M(i,j),\ 1 + d_M(k,j)) & \text{if } l(i,k) \text{ exists} \end{cases} \qquad (7)$$

In this equation, $l(i,k)$ means that node i is connected to node k via a long-range link. The applicability of the distance definition is illustrated in Figure 5. Note that the distance computation does *not* require any global knowledge thus making the routing decision algorithm decentralized.

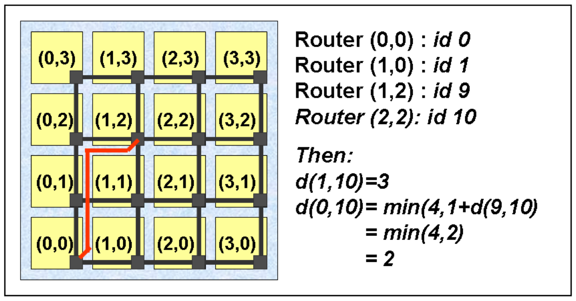


Figure 5. Illustration of distance definition (see Eqn. 7).

Interestingly enough, Equation 4 also confirms that the average number of hops between the nodes, a common performance metric, has indeed a positive impact on the dynamic behavior of the network.

E. Routing strategy for long-range links

The routers without any extra link use the default routing strategy. Defining a strategy for the routers with long-range links is a necessity dictated by two factors:

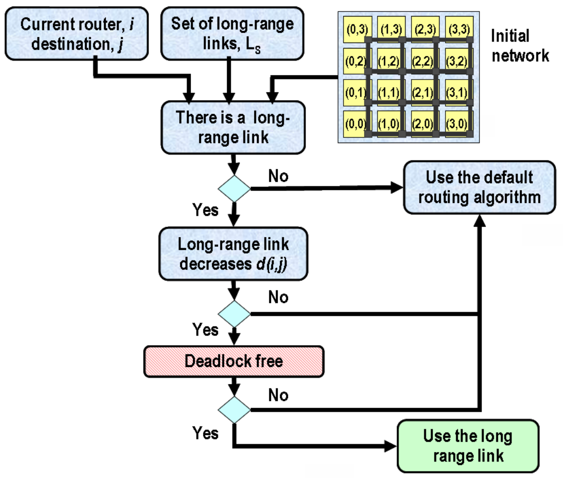
• Without a customized mechanism in place, the newly added long-range links cannot be utilized at all by the default routing strategy;

• Introducing long-range links may result in cycling dependencies. Therefore, arbitrary use of these links may result in deadlock states.

The routing strategy proposed in this section tries to produce minimal paths towards the destination by utilizing the long-range links effectively, as shown in Figure 6. To this end, we first check whether or not there exists a long-range connection to the current router. If there is one, the distance to the destination with and without the long-range link is computed using Equation 7. It is interesting to note that we can obtain global improvements in the network dynamics by using local information only.

If a newly added long-range link produces a shorter distance to the destination, we check whether or not using this link may cause deadlock before accepting it as a viable route. In order to guarantee that using the newly added link does not cause a deadlock state, some limitations on the use of long-range links are introduced.

We achieve deadlock-free operation by extending the turn-model [13] to long-range links. More precisely, in the original turn model, one out of four possible turns is prohibited to avoid cyclic dependencies (Figure 7). However, unlike standard links, the long-range links can extend in two directions, such as NE, NW, *etc.* For this reason, one has to consider the rotation from middle directions, NE, NW, SE, SW to the main directions N, S, E and W. In the extended model, we arbitrarily chose the S-to-E, S-to-W, SE-to-E, SE-to-W, SW-to-E, and SW-to-W turns



Current router, *i* destination, *j*
Set of long-range links, L_S
(0,3) (1,3) (2,3) (3,3)
(0,2) (1,2) (2,2) (3,2)
(0,1) (1,1) (2,1) (3,1)
(0,0) (1,0) (2,0) (3,0)
Initial network
There is a long-range link
No
Use the default routing algorithm
Yes
Long-range link decreases *d(i,j)*
No
Yes
Deadlock free
No
Yes
Use the long range link

Figure 6. The description of the routing strategy.

as being illegal, as in Figure 7. Therefore, we check whether or not the long-range links cause any of these prohibited turns. If it is legal to use a long-range link, then the packet is forwarded to this link. Otherwise, the default routing algorithm is employed.

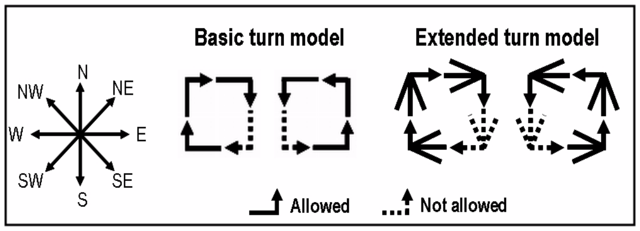


Figure 7. Possible routing directions, basic and extended turn models.

A long-range link may become a traffic attractor and jam the network earlier than the regular short links. For this reason, if the routing algorithm is not adaptive (as it is our case), one more check point is needed before assigning a traffic stream to a new link. For instance, by assessing the amount of traffic assigned to a long-range link, further traffic can be routed over the link only if it is not likely to become a bottleneck.

IV. PRACTICAL CONSIDERATIONS

We analyze next the implications of customizing the regular network architecture with long-range links on the actual design implementation.

A. Implementation of long-range links

In order to preserve the advantages of structured wiring, the long-range links are segmented into regular, fixed-length, network links connected by repeaters. The use of repeaters with buffering capabilities guarantees latency-insensitive operation, as discussed in [19]. Repeaters can be thought as simplified routers consisting of only two ports that accept an incoming flit, stores it in a FIFO buffer, and finally forwards it to the output port, as illustrated in Figure 8. If the depth of buffers in the repeaters is at least 2 flits, then the packets can be effectively pipelined to take the full advantage of the long-range links [20].

The final consideration in terms of implementation is the increased size of the routers with extra links due to increased number of ports (Figure 8). To measure the area overhead, we implemented routers with 5 and 6 ports using Verilog and synthesized them for a 1M gate Xilinx Virtex2 FPGA. The router with 5 ports utilizes 387 slices (about 7% of total resources), while the one with 6 ports utilizes 471 slices of the target device. We also synthesized a pure 4×4 mesh network, and a 4×4 mesh network with 4 long-range links and observed that the extra links induce about 10% area overhead. This overhead has to be taken into account, while computing the maximum number of long-range links that can be added to the regular mesh network. While there is no theoretical limitation imposed by our approach on the number of additional links a router can have, a maximum of one long-range link per router is used in our experiments. This way the regularity is minimally altered, while still providing significant improvements over the standard mesh networks, as explained in Section V.

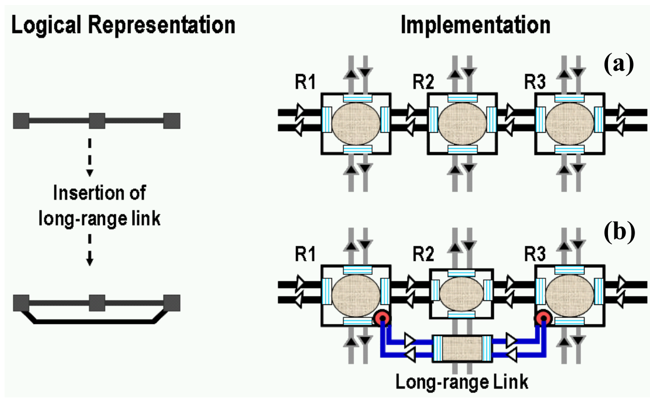


Figure 8. Implementation of the repeaters. Routers 1 and 3 are both connected by Router 2 (a), the underlying mesh network, and the inserted long-range link (b).

B. Energy-related considerations

One can measure the energy consumption using the E_{bit} metric [12], defined as the energy required to transmit one bit of information from the source to the destination. E_{bit} is given by

$$E_{bit} = E_{L_{bit}} + E_{B_{bit}} + E_{S_{bit}} \quad \textbf{(8)}$$

where $E_{L_{bit}}$, $E_{B_{bit}}$ and $E_{S_{bit}}$ represent the energy consumed by the interconnect, buffering and switching in the router, respectively. Analyzing the energy consumption before and after the insertion of long-range links shows that the proposed approach does *not* induce a significant penalty in the total communication energy consumption of the network. Indeed, since the long-range links consist of several regular links with repeaters between them (instead of routers), the link energy consumption stays approximately the same whether the traffic flows over the long-range link or over the original path provided by the mesh network. On the other hand, the energy consumption due to the switch and routing logic is greatly simplified in the repeater design compared to the original routers. This results in a reduction in the switching energy consumption. Finally, the routers with extra links will have slightly increased energy consumption due to the larger crossbar switch.

We compared the energy consumption obtained by simulation before and after the insertion of the long-range links, for the traffic patterns reported in Section 6. We observed that the link and buffer energy consumptions increase about 2% after the insertion of long-range links, while the switch energy consumption drops about 7%, on average. The results show that the overall energy consumption increases by only about 1%.

V. EXPERIMENTAL RESULTS

We present next an extensive experimental study involving a set of benchmarks with synthetic and real traffic patterns. The NoCs under study are simulated using an in-house cycle accurate C++-based NoC simulator developed specifically for this project. The simulator models the long-range links precisely as explained in Section IV. The configuration files describing the additional long-range links and the routing strategy for a given traffic pattern are generated using the proposed technique and supplied to the simulator as an input.

The worst-case complexity of the technique, that is link insertion and routing table generation, is $O(SN^{\alpha})$ where $2<\alpha<3$. The run-time of the algorithm for the examples analyzed ranges from 0.14sec for a 4×4 network to less than half hour for a 8×8 network on a Pentium III machine with 768MB memory under Linux OS.

A. Experiments with synthetic traffic workloads

We first demonstrate the effectiveness of adding long-range links to standard mesh networks by using synthetic traffic inputs. Table 1 compares the critical traffic load, average packet latency and throughput at the edge of criticality under hotspot traffic pattern for 4×4 and 6×6 networks. For the hotspot traffic three nodes are selected arbitrarily to act as hotspot nodes. Each node in the network sends packets to these hotspot nodes with a higher probability compared to the remaining nodes.

	Critical load (*packet/cycle*)		Latency at the critical load (*cycles*)	
	λ_{Mc}	λ_{Lc}	$L_M(\lambda_{Mc})$	$L_L(\lambda_{Mc})$
hotspot *4x4*	0.41	0.50	196.9	34.4
hotspot *6x6*	0.62	0.75	224.5	38.2

Table 1: Critical load (*packet/cycle*) and latency comparison (*cycles*) for regular mesh (M) and mesh with long links (L).

As shown in Table 1, inserting 4 long-range links (consisting of 10 short link segments) to a 4×4 network makes the phase transition region shift from 0.41 *packet/cycle* to 0.50 *packet/cycle* (the resulting network appears in Figure 1). Similarly, the average packet latency at 0.41 *packet/cycle* injection rate drops from 196.9 to 34.4 *cycles*. We also show the variation of the network throughput and average packet latency as a function of traffic injection rate using a much denser scale in Figure 9.

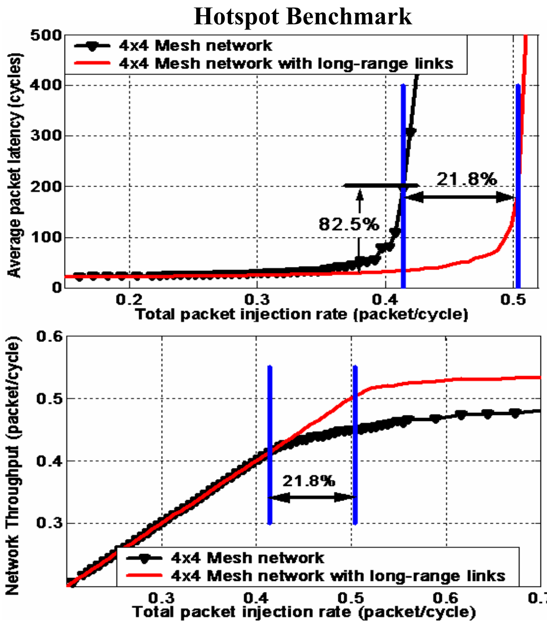


Figure 9. Traffic injection rate vs. average packet latency and network throughput for hotspot traffic is shown. The improvement in terms of critical point and latency values at criticality are indicated on the plots.

Similar results have been obtained for a 6×6 grid, as shown in Table 1. In this case, the phase transition region shifts from 0.62 *packet/cycle* to 0.75 *packet/cycle*. Likewise, with the addition of long-range links, the average packet latency at 0.62 *packet/cycle* injection rate drops from 224.5 to 38.2 *cycles*.

Comparison with the torus network

We also compared the performance of the proposed approach against that achievable with a torus network, which provides wrap around links added in a systematic manner. Our simulations show that application-specific insertion of only 4 long-range links, with an overhead of 12 extra standard link segments, provides 4% improvement in the critical traffic load compared to a 4×4 torus under hotspot traffic. Furthermore, the average packet latency at the critical point of the torus network, 0.48 *packet/cycle*, drops from 77.0 to 34.4 *cycles*. This significant gain is obtained over the standard torus network by utilizing only *half* of the additional links, since the extra links are inserted in a smart way considering the underlying application rather than blindly adding wrap-around channels.

Scalability Analysis

To evaluate the scalability of the proposed technique, we also performed experiments involving networks of sizes ranging from 4×4 to 10×10. Figure 10 shows that the proposed technique results in consistent improvements when the network size scales up. For example, by inserting only 6 long-range links, consisting of 32 regular links in total, the critical load of a 10×10 network under hotspot traffic shifts from 1.18 *packet/cycle* to 1.40 *packet/cycle* giving a 18.7% improvement. This result is similar to the gain obtained for smaller networks. Figure 10(a) also reveals that the critical traffic load grows with the network size due to the increase in the total bandwidth. Likewise, we observe consistent reduction in the average packet latency across different network sizes, as shown in Figure 10(b).

B. *Experiments involving real applications*

In this section, we evaluate the performance of the link insertion algorithm using two applications with realistic traffic: A 4x4 *auto industry* benchmark and a 5x5 *telecom* benchmark retrieved from E3S benchmark suite [21].

The variation of average packet latency and network throughput as a function of traffic injection rates for *auto industry* benchmark is given in Figure 11. These plots show that the insertion of long-range links shifts the critical traffic load from 0.29 *packet/cycle* to 0.33 *packet/cycle* resulting in a 13.6% improvement. Similarly, as shown in Table 2, we observe that the average packet latency for the network with long-range links is consistently lower compared to that of a pure mesh network. For instance, at 0.29 *packet/cycle* injection rate, the latency drops from 98.0 *cycles* to 30.3 *cycles* giving about 69.0% reduction.

Similar improvements have been observed for the *telecom* benchmark as shown in Figure 12. Specifically, the critical traffic load is delayed from 0.44 *packet/cycle* to 0.60 *packet/cycle* showing a 36.3% improvement due to the addition of long-range links. Likewise, the latency at 0.44 *packet/cycle* traffic injection rate drops from 73.1 *cycles* to 28.2 *cycles* (Table 2).

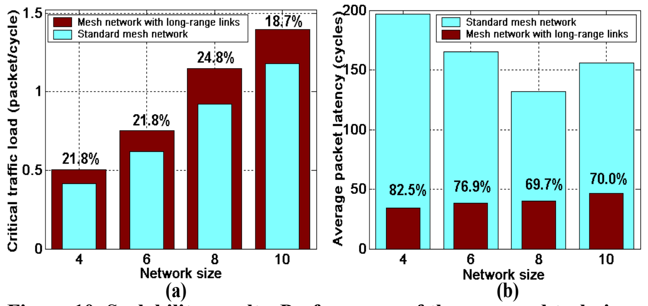


Figure 10. Scalability results. Performance of the proposed technique for larger network sizes.

Comparison with the mesh network with extra buffers

Implementation of long-range links requires buffers in the repeaters. To demonstrate that the savings are the result of using the long-range links, we also added extra amount of buffers to the corresponding channels of the pure mesh network, equal to the amount of buffers utilized for the long-range links.

Table 2 summarizes the results for standard mesh network (*M*), standard mesh network with extra buffers (*MB*) and the network with long-range links (*L*). We observe that insertion of buffers improves the critical load by 3.5% for the *auto industry* benchmark. On the other hand, the corresponding improvement due to long-range links is 13.6% over initial mesh network and 10% over the mesh network with additional buffers. Likewise, we note that with the insertion of long-range links, the average packet latency reduces by 69% compared to the original value and 57.0% compared to the mesh network with extra buffers.

Consistent results have been obtained for the synthetic traffic workloads mentioned in the previous section and for the *telecom* benchmark. Due to the limited space, we report only the results for *telecom* benchmark (Table 2). The results show that, with the addition of extra buffers, the critical traffic point shifts only from 0.44 *packet/cycle* to 0.46 *packet/cycle*. Inserting long-range links, on the other hand, shifts the critical point to 0.60 *packet/cycle* which is a huge improvement in the network capability. Similarly, the average packet latency obtained by the proposed technique is almost 1/3 of the latency provided by standard mesh and about 1/2 of the latency provided by mesh with extra buffers.

	Critical load (*packet/cycle*)	**Latency at critical load (*cycles*)**
	λ_c	$L(\lambda_{Mc})$
auto-indust M	0.29	98.0
auto-indust MB	0.30	70.5
auto-indust L	0.33	30.3
telecom M	0.44	73.1
telecom MB	0.46	56.0
telecom L	0.60	28.2

Table 2: Critical load (*packet/cycle*) and latency (*cycles*) comparison for pure mesh (M), mesh with extra buffers (MB) and mesh with long links (L).

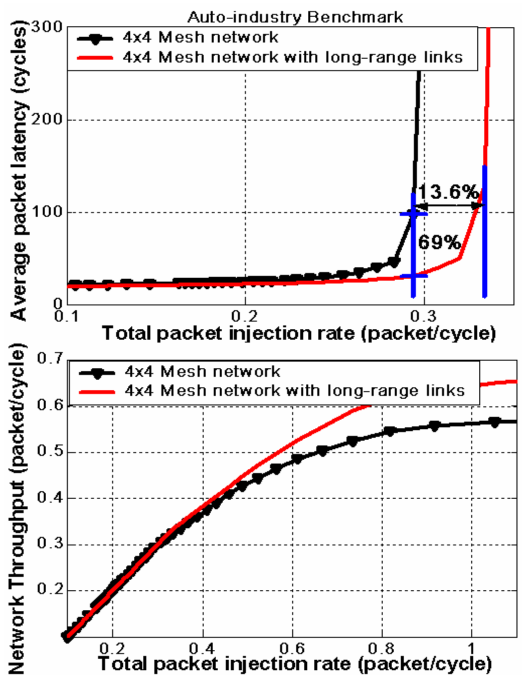


Figure 11. Traffic rate vs. packet latency and network throughput for *auto industry* benchmark.

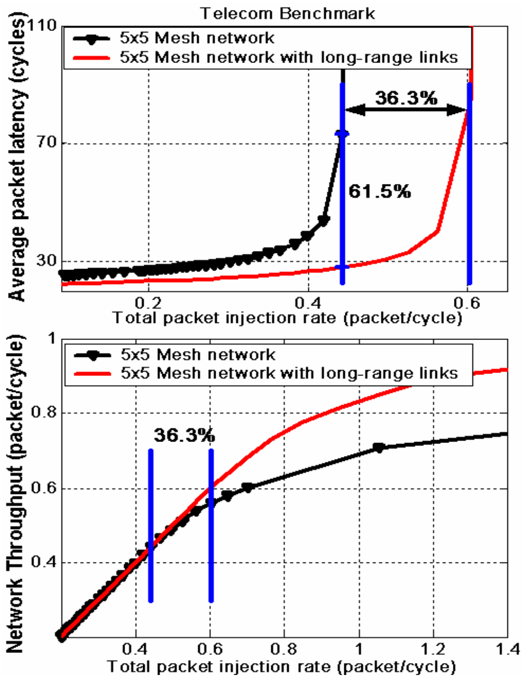


Figure 12. Traffic injection rate vs. average packet latency and network throughput for *telecom* benchmark.

VI. CONCLUSION AND FUTURE WORK

We have presented a design methodology to insert application-specific long-range links to standard grid-like networks. It is analytically and experimentally demonstrated that insertion of long-range links has an important impact on the dynamics, as well as static properties of the network. Specifically, additional long-range links increase the critical traffic workload. We have also demonstrated that this increase means significant reduction in the average packet latency in the network, as well as improvement in the achievable throughput.

Our current work employs oblivious routing to utilize the long-range links. We plan to extend this work to employ adaptive routing instead. Other possible extensions include inserting long-range links for different objective functions such as fault-tolerance and QoS operation.

Acknowledgements: This research is supported by Marco GSRC, NSF CCR-00-93104, and SRC 2004-HJ-1189.

VII. REFERENCES

[1] W. Dally, B. Towles, “Route packets, not wires: On-chip interconnection networks,” In Proc. DAC, June 2001.

[2] L. Benini, G. De Micheli. “Networks on chips: A new SoC paradigm,” IEEE Computer. 35(1), 2002.

[3] A. Jantsch, H. Tenhunen (Eds.). Networks on Chip. Kluwer, 2003.

[4] M. Millberg, E. Nilsson, R. Thid, S. Kumar, A. Jantsch, “The Nostrum backbone - a communication protocol stack for networks on chip,” In Proc. VLSI Design, Jan. 2004.

[5] K. Goossens, *et. al.* “A Design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification,” In Proc. DATE, March 2005.

[6] J. Hu, R. Marculescu, “Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures,” In Proc. DATE, March 2003.

[7] S. Murali, G. De Micheli, “SUNMAP: A tool for automatic topology selection and generation for NoCs,” In Proc. DAC, June 2004.

[8] A. Pinto, L. P. Carloni, A. L. Sangiovanni-Vincentelli, “Efficient synthesis of networks on chip,” In Proc. ICCD, Oct. 2003.

[9] K. Srinivasan, K. S. Chatha, G. Konjevod “Linear programming based techniques for synthesis of Network-on-Chip architectures,” In Proc. ICCD, Oct. 2004.

[10] U. Y. Ogras, R. Marculescu, “Energy- and performance- driven customized architecture synthesis using a decomposition approach,” In Proc. DATE, March 2005.

[11] A. Jalabert, S. Murali, L. Benini, G. De Micheli, “xpipesCompiler: A tool for instantiating application specific networks on chip,” In Proc. DATE, March 2004.

[12] T. T. Ye, L. Benini, G. De Micheli, “Analysis of power consumption on switch fabrics in network routers,” In Proc. DAC, June 2003.

[13] C. J. Glass, L. M. Ni, “The turn model for adaptive routing,” In Proc. ISCA, May 1992.

[14] D. J. Watts, S. H. Strogatz, “Collective dynamics of 'small-world' networks,” Nature, 393:440-442, 1998.

[15] J. Kleinberg, “The Small-World phenomenon and decentralized search,” SIAM News 37(3), April 2004.

[16] H. Fuks, A. Lawniczak, “Performance of data networks with random links,” Mathematics and Computers in Simulation, vol 51, 1999.

[17] M. Woolf, D. Arrowsmith, R. J. Mondragon, J. M. Pitts, “Optimization and phase transitions in a chaotic model of data traffic,” Physical Review E, vol. 66 (2002).

[18] J. Duato, S. Yalamanchili, N. Lionel, Interconnection Networks: an Engineering Approach. Morgan Kaufmann, 2002.

[19] L. P. Carloni, K. L. McMillan, A. L. Sangiovanni-Vincentelli, “Theory of latency-insensitive design,” IEEE Trans. on CAD of Integrated Circuits and Systems. 20(9), 2001.

[20] V. Chandra, A. Xu, H. Schmit, L. Pileggi, “An interconnect channel design methodology for high performance integrated circuits,” In Proc. DATE, Feb. 2004.

[21] R. Dick, “Embedded system synthesis benchmarks suites (E3S),” http://helsinki.ee.princeton.edu/dickrp/e3s/.