A Hybrid Ring/Mesh Interconnect for Network-on-Chip Using Hierarchical Rings for Global Routing

S. Bourduas, Z. Zilic
Department of Electrical and Computer Engineering
McGill University
Montreal, Quebec, Canada
{stephan,zeljko}@macs.ece.mcgill.ca

***Abstract*— A popular network topology for Network-on-Chip (NoC) implementations is the two-dimensional mesh. A disadvantage of the mesh topology is in its large communication radius. By partitioning a two-dimensional mesh into several sub-meshes and connecting them using a global interconnect, we can reduce the average number of hops for global traffic. This paper presents a hybrid architecture that partitions a large 2D-mesh into several smaller sub-meshes which are globally connected using a hierarchical ring interconnect. Hierarchical rings have been selected for study because of their simplicity, speed and efficiency in embedding onto a circuit layout, as well as for their suitability for efficient cache coherent protocols. An original SystemC modeling platform was implemented in order to compare the traditional 2D-mesh with the hybrid ring/mesh architectures and the simulation results will show that our hybrid architecture does indeed have a positive effect on the average hop count.**

I. Introduction

The first system-on-chip (SoC) implementations used buses to connect components together. As technology scaling enabled more cores to be integrated onto a single chip, it became apparent that bus-based approaches could not adequately handle the communication demands of multi-processor SoC (MPSoC) implementations. The main problem with shared medium approaches is that as the number of connected components increases, so does the parasitic capacitance, propagation delay, power consumption and arbitration times. There is therefore a practical limit to the number of cores which can be connected using shared medium approaches.

Systems consisting of *tens* or even *hundreds* of cores are not feasible using shared medium architectures. A scalable alternative is presented by a network-centric approach, whereby packets are routed through an interconnect network [1], [2]. A *network-on-chip* (NoC) [1], [3], [4] which routes packets in a way similar to traditional networks would replace the bus and allow for systems with many more components to be efficiently connected together. A NoC interconnect consist of multiple switches connected together to form a suitable network topology [1]. In contrast to bus-based approaches, NoCs are more energy-efficient, can support higher aggregated bandwidth, and most importantly offer greater scalability.

The 2D-*mesh* [2] topology is a popular topology used for NoC interconnects. It consists of $M \times N$ number of tiles arranged in a grid where each tile is connected to its four neighbors (with the exception of the edge tiles). Because only neighboring nodes are connected, packets that need to travel long distances suffer from large hop counts.

In this paper, we propose a hybrid topology that breaks a large mesh into smaller meshes connected by a hierarchical ring interconnect for routing global traffic. We have selected hierarchical rings for study because of their simplicity, speed and the ease with which such structures can be laid out in 2D IC layouts. Recent industrial designs, such as the Cell processor and some high-end graphical processor from ATI employ multiple rings as their interconnect networks.

To study the proposed topology in detail, we have developed and implemented a modular simulation platform in SystemC which enables us to compare the performance characteristics of our hybrid interconnect with that of the traditional 2D-mesh.

II. Related Work

There are several topologies that have been proposed for use in NoCs which can be classified as either flat or hierarchical. A detailed comparison of several architectures is presented in [5], [6]. As previously discussed, the 2D-mesh proposed in [2] is the most popular. The *torus* and *folded torus* are similar to the mesh but the edge switches are connected to the opposite switches in order to form a ring. The torus interconnect has a higher bisection bandwidth than the mesh, but also exhibits higher energy consumption [2].

An example of a hierarchical topology is the *fat tree* architecture presented in [7] which can support low latencies and high bandwidths depending on the chosen configuration. The architecture in [7] uses dedicated feedback wires between pairs of receivers and senders for flow control which we feel is not suitable for large scale NoCs with a large number of nodes. A second example of a hierarchical topology is the *butterfly fat-tree* (BFT) [8] where the number of switches converges to a constant depending on the number of levels. Unlike the mesh architecture proposed in [2], [9] where each cell is composed of a PE and a switch, the fat-tree and butterfly fat-tree place their processing elements at the leaves and the switches at the vertices of the tree [7], [8].

The *Proteo* [10] NoC is a hierarchical network topology which uses a global bidirectional ring to connect several subnets together. The topology of each subnet is chosen to suit local traffic requirements. The architecture presented in citesiguenca-tortosa:csn02 has been built using the *Scalable Coherent Interface* (SCI) standardized by IEEE [11]. This standard has garnered some industrial acceptance as a ring-based network topology with its own distributed directory cache coherency protocol that is cache based and requires a linked list of cache locations to be maintained to keep the shared memory data coherent. Practical problems with SCI arose exactly because of the need to traverse the network following the linked list for each coherence operation. Somewhat similar ideas in the topology were also reused with the emergence of the Infiniband storage area network, which to our knowledge has not been used in any SoC implementations.

Similarly, the *Ring Road* [12] topology was proposed with the idea of using ring switching elements in a manner that provides more bisectional bandwith and eliminates hotspots in the center. There are two types of ring interfaces, just like with the hierarchical rings: those that are on local rings, and those on intersection of a pair of rings. The motivation for this kind of interconnect is in avoiding congestion in the center of the area, similar to the rings of roads outside large cities.

In [13], a reconfigurable system which uses a hierarchical mesh interconnection network consisting of *nearest neighbor* connectivity at the lowest level of the hierarchy and horizontal and vertical buses for global connectivity. The architecture presented in [14] takes the opposite approach and uses a hierarchical interconnect to link multiple bus-based SoCs together.

A parameterizable library of components called *Xpipes* which can be used to generate domain-specific heterogeneous architectures is described in [15]. The architectures discussed in [15] are not hierarchical in nature and the problems of growing hop-counts and latencies associated with increasing network size are not addressed, however the authors state that arbitrary topologies can be achieved by their tool.

In [16], the problem of large hop counts associated with routing packets over long distances in 2D-meshes is addressed through the use of *express channels* which span multiple hops. The drawback of using express channels is that they require long wires in each dimension and they also increase the router complexity.

In [17], a hierarchical ring topology was introduced which made use of a two-tier hierarchical configuration of unidirectional rings. The use of this topology was inspired by a NUMAchine multiprocessor designed at University in Toronto. As shown in [18], good speedups were observed for virtually all multiprocessor benchmarks, in spite of its apparent bisectional bandwidth limitations. The architecture was shown amenable to efficient implementations, and the cache coherence protocol incorporated in NUMAchine exploited well the given topology resulting in a feasible and correct implementation.

III. Motivation

A. Hybrid Topology

It was shown in [16] that express channels can help reduce the latencies associated with routing global packets. The major disadvantage of this approach is that express channels are needed in both the x and y directions and so the routing complexity of this topology increases with the number of express channels; a problem which is further exacerbated as the mesh size gets larger and longer express channels are needed.

In Section II, several tree-like hierarchies were briefly discussed. When comparing hierarchical architectures to a 2D-mesh architecture, we can see that on average, the hop count for global packets is smaller, but the aggregate bandwidth is also reduced because of the bottlenecks associated with routing global packets through a small number of switches. On the other hand, less resources are required and thus the energy requirement for global interconnect will be reduced which is of critical importance as the interconnect has already been reported to account for a large portion of the total system energy requirement [19]. The challenge then becomes finding a balance between resource requirements and system performance.

Since both the mesh and hierarchical topologies exhibit desirable characteristics with respect to embedding them onto the 2D layout of SoCs, combining them would enable us to draw on the strengths of each and result in reduced average hop-counts and latencies for global traffic while still maintaining the high throughput that meshes exhibit for local traffic.

B. Suitability of Hierarchical Rings

Our topology of unidirectional rings connected in a hierarchical manner exhibits several characteristics which are of importance to NoC implementations. The simplicity of the rings reduces the complexity at each node which results in reduced buffer, area and energy requirements. Furthermore, the topology discussed in this paper has no global routing so place-and-route will also be more efficient than using global channels as was shown in [20]. The unidirectional nature of the rings reduces the overhead associated with routing and thus results in low latencies and high throughput.

One limitation of the NUMAchine architecture that we try to address here is in the scalability of the hierarchical ring. Because the buses are used at the lowest level of the hierarchy in [18], the total number of nodes will be kept modest if the number of hierarchy levels is kept low. By replacing local buses with meshes, we can accommodate more processors for the same hierarchical ring. Similarly to [18], we aim to use the concept of the network cache placed at the gateways to rings, to facilitate efficient cache coherency primitives, including the use of low-cost multicast/broadcast invalidations with guaranteed serialization property, which is in turn critical for both cache coherence and the useful consistency models.

Also of interest is the fact that the hierarchical ring interconnect can be easily partitioned into multiple clock domains, giving designers increased flexibility when tuning design parameters for individual applications. As discussed in [17], distinct clock domains enable the application of dynamic frequency and/or voltage scaling (DVS) techniques for energy optimization. Taking the ideas from [17] one step further, we can also envision the application of clock throttling to an entire sub-mesh which can be easily achieved since each mesh can exist in a separate clock domain. In the case of a heterogeneous architecture where certain types of computational units are assigned to a specific mesh, the energy savings has the potential to be significant if entire sub-meshes could be powered down during idle periods.

IV. Architecture

Our hybrid architecture a combination of a hierarchical ring interconnect with meshes, used in conjunction with a specific addressing and routing scheme. The hierarchical ring is used for routing global traffic between several meshes.

A. Hybrid Interconnect

In contrast to other approaches which add complexity to the interconnect through the use of global global wiring [7], [16], we leave the 2D-mesh architecture unmodified and simply replace a processing element in one of the cells by a bridge component which enables sending global traffic through the hierarchical interconnect transparently. Figure 1 shows the hybrid-mesh architecture where a 8×8 mesh has been split into 16 4×4 sub-meshes which are globally connected using a 2-level hierarchical ring interconnect. From Figure 1, we can deduce that if the width of the mesh being partitioned is of width N, then the width of a sub-mesh and the width of a local mesh is described by Eqs. 1, 2, where a sub-mesh is the smallest mesh in the system a local mesh is defined as the theoretical mesh obtained by combining the 4 meshes connected to a local ring.

$$W_{sub} = \frac{N}{4} \tag{1}$$

$$W_{local} = \frac{N}{2} \tag{2}$$

In a normal mesh network, the maximum number of hops that a packet will travel is when a packet is sent from a corner node to its diagonal opposite. For example, if node (0,0) send a packet to node (M-1, N-1), then the total number of hops will be $M + N = 2N$ for when $M = N$. For some large enough value of N, the latencies incurred by the network will be too large for application software to support.

The worst-case hop count for our hybrid interconnect depends on which tile in the mesh is used to connect to the hierarchical ring interconnect. If a corner tile is used, the worst case hop count can be described by Eq. 3 whereas if a tile in the middle of the sub-mesh is used, the worst case hop count can be described by Eq. 4. Note that the maximum number of hops a packet can take to travel through the hierarchical ring interconnect is 12.

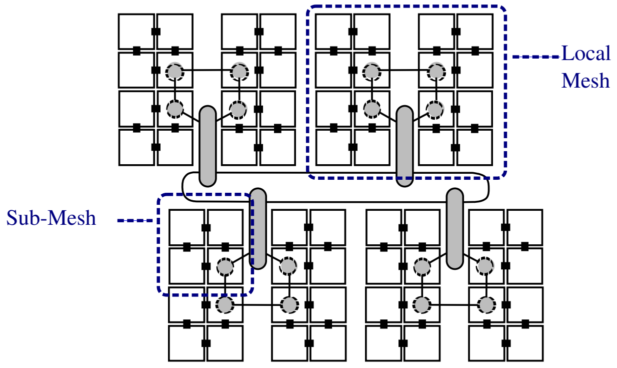

Fig. 1. Hybrid mesh architecture using hierarchical rings for global interconnect ($N = 8$)

$$H_{worst} = 2\left[\frac{M}{4} + \frac{N}{4}\right] + 12 \tag{3}$$

$$H_{best} = 2\left[\frac{M}{8} + \frac{N}{8}\right] + 12 \tag{4}$$

Figure 2 shows how the hop counts increase as N increases for the mesh and hybrid architecture, but it does not necessarily imply that the latencies will increase proportionately. There are many parameters that can affect the latency resulting in a large design space. A property of the hybrid interconnect that we have implemented is that the xy-routing algorithm tends to route global traffic away from the center of sub-mesh towards the edges when the bridge tile is located in a corner. As will be seen in Section VI-A, the overall effect is that the resources in the center of the mesh end up processing less global traffic and local traffic is handled more efficiently.

When global traffic is routed through a bridge component, that component can quickly become flooded causing the flow control mechanisms to assert themselves having a negative impact on latency. Increasing the input buffer sizes of the bridge tile solved the problem and enabled traffic to flow much more smoothly. The required buffer size is very sensitive to the ratio of global to local traffic as well as the size of the sub-mesh since the amount of traffic increases quadratically with N.

B. Hierarchical Rings

The hierarchical-ring architecture used for the global interconnect in our hybrid-architecture is an adaptation of the design presented in [21] which was used in shared memory multiprocessor systems. As shown in Figure 3, our two-level hierarchy consists of four local rings connected to a global ring.

In order to keep buffering and latencies to a minimum, each packet is actually a *flit/phit* and can be forwarded in 1 clock cycle. As shown in Figure 3, packets are routed onto a local ring via a ring-interface (RI). Once on the local ring, a packet can be forwarded to another RI on the same local ring or it can be routed upwards to the global ring via the inter-ring interface (IRI).

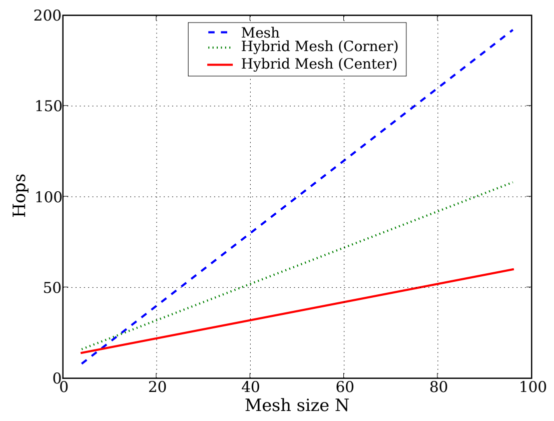

Fig. 2. Worst case hop counts for the mesh and hybrid-mesh topologies

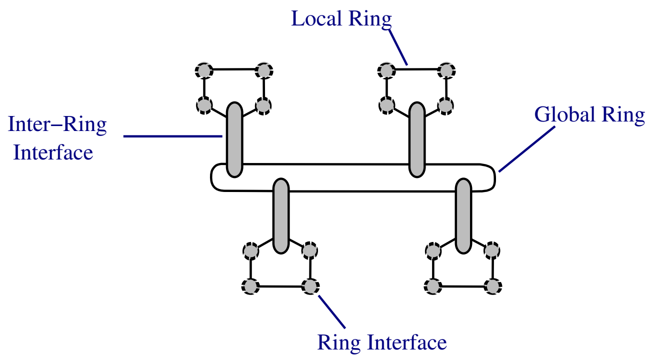

Fig. 3. Hierarchical Ring Interconnect.

1) Routing and Flow Control: The hierarchical rings provide lossless communications through the the use of a backpressure mechanism for handling network congestion which prevents packets from being dropped. At each hierarchical level, a backpressure signal can be propagated in order to prevent injection of new packets until outstanding packets have been drained from the interconnect and the backpressure signal has been de-asserted. A *Stop Up* signal is asserted by the global ring in order to stop the local rings from injecting new packets because of congestion on the global ring. A *Stop Down* signal is asserted by a local ring in order to stop the global ring from putting new packets on a local ring.

The hierarchical ring interconnect was originally designed to be realistically used on an FPGA [20]. The interconnect had to be area efficient yet support both point-to-point addressing, multicasting and broadcasting. This was solved by virtue of the hierarchical configuration of the rings coupled with a one-hot encoding of the addresses. The destination address in each packet header is of the form {Global Routing Mask, Local Routing Mask}. For example, if a PE has to send data to all stations on the local ring, the destination address would look like {G=0001,L=1110}. If the same data had to be broadcast to all stations on Local Ring 3, the destination address would be {G=0100, L=1111}.

C. Enhanced Hybrid Interconnect

The hierarchical ring interconnect described in [17], [20] is prone to congestion when backpressure signals are asserted resulting in the network being underutilized. The hierarchical nature of the interconnect and its backpressure signals can be exploited in order to boost performance. It can be seen from Figure 3 that the system bottleneck is the global ring. If the global ring asserts a backpressure signal, all 4 local rings must stop sending data, which can result in the local rings being under-utilized. In the work presented in [17], [20], the hierarchical ring interconnect was used to connect PEs together, so the situation was acceptable, especially considering that the resources usage of the interconnect is efficient [20]. In this paper, we propose to use the same structure to connect multiple NoCs together instead of PEs, so the bandwidth requirement on the hierarchical ring interconnect will be much greater.

Traffic on the hierarchical ring interconnect can be classified as *local* and *global*. Local traffic does not need to travel to the global ring, which is a fact that can be exploited and enable the implementation of an *enhanced* version of our hybrid interconnect. The *Stop Up* and *Stop Down* signals described in Section IV-B.1 can be used to determine if data can still be injected into the interconnect. If the global ring asserts a *Stop Down* signal because of congestion, this does not necessarily mean that the local ring is also congested. We can thus decouple the sending of local and global data in order to reduce the overall latencies of traffic in our hybrid architecture.

By modifying the implementation of the ring-interface component of the hierarchical rings, we can achieve two virtual channels over the same hardware for every local ring:

- The *inner* channel is used for sending data locally on a ring.
- The *outer* channel is used for sending global data through the global ring.

The splitting of the network into two logical channels requires that we sacrifice a second tile on the mesh in order to keep from having to multiplex the data at the level of the mesh. Figure 4 shows how the enhanced architecture uses two tiles to connect the mesh and hierarchical rings together. Since a corner tile has only two input and output ports, we prefer to route traffic to the inner ring through the interface that is not on an edge.

D. Mesh

The switch used in the mesh portion of our NoC simulation model has 5 input ports which have input FIFOs to store incoming packets. The output ports are unbuffered and are connected directly to their neighbors. Flow control between nodes on the mesh is achieved through the use of 'on-off' flow control [3]. To facilitate the interfacing of the mesh interconnect with the hierarchical ring interconnect, we chose to route flits in the mesh instead of a more complicated scheme such as *wormhole routing*. We use simple *xy-routing* to route packets through the mesh.

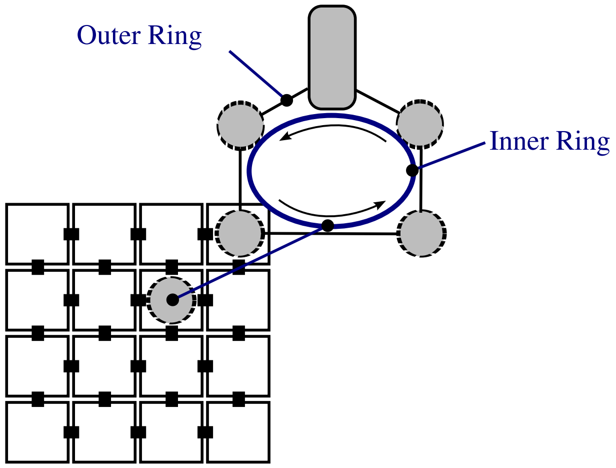

Fig. 4. Enhanced architecture which uses two tiles to send data through the hierarchical ring interconnect.

Some mesh implementations use more complicated flow control schemes such as *stop-and-wait* or *go-back-n* in order to recover from bit errors. We decided to make the network interconnect as simple as possible so that the resource usage and latencies were reduced. We felt that if needed, error detection and resolution can be done at the next level up in the processing stage. Since our interconnect routes flits instead of multi-flit packets, we can easily implement a variety of protocols on top of the interconnect depending on QoS constraints.

E. Globally Asynchronous, Locally Synchronous

As discussed in [17], [20], the hierarchical-ring interconnect can be partitioned into separate clock domains. The fact that the clock rate of the different rings can be independent allows for increased flexibility when tuning the interconnect for specific applications. For example, the clock rate of the global ring can be higher than that of the local rings in order to reduce the latency of global traffic [21]. Furthermore, multiple clock domains provide the facility for the eventual introduction of dynamic clock throttling [17] which can allow rings to be slowed down or sped up as needed to accommodate changing bandwidth requirements while reducing energy consumption.

V. Simulation Platform

As mentioned previously, we implemented our model in *SystemC* using an object-oriented style which enabled the modeling of different architectures by plugging modules from a library of components together. We incorporated into the model a module which tracks packet latencies and hop-counts during simulation and and calculates the average of each at the end of a run.

A. Traffic Types

As we are primarily interested in discovering the effects of using the hierarchical ring model in order to route global traffic, we have partitioned our traffic into three possible categories that correspond to the layers in the hierarchy as illustrated in Figure 1:

1) *level 0* (L0): local traffic that is sent between stations at the lowest level of the hierarchy.
2) *level 1* (L1): traffic that goes through a single local ring to reach a station on another sub-mesh (e.g. L0 up to L1 and back down to L0).
3) *level 2* (L2): traffic that is routed between local rings needs to pass through the global ring.

TABLE I
Probabilities used to generate traffic during simulations.

Traffic Type	Probability
L0	0.7
L1	0.2
L2	0.1

The packet types relate to the levels that need to be traversed in the hierarchy. In order to compare the mesh and hybrid architectures, a similar classification of traffic types was devised for the mesh architecture. From the description of the hybrid architecture in IV-A, we see that the maximum distance a packet can travel locally is equal to W_{sub} in either direction, so the maximum hop count for local packets (L0) can be expressed as $2 \times W_{sub}$. Similarly, we we can define the maximum hop count for *L1* traffic to be $2 \times W_{local}$. Lastly, traffic of type *L2* has a maximum hop count equal to N.

For the hybrid interconnect, a local packet is physically constrained to be destined to a station belonging to the same sub-mesh as the sender. In the normal mesh topology, there is no such physical constraint, so we define a local packet on the normal mesh to be any packet that does not need to travel farther than the maximum allowed by L0. Types L1 and L2 are similarly constrained.

B. Traffic Generation

Traffic was generated using two methods. The first method consisted of using random variables to generate traffic and the second consisted of mapping a task graph to the interconnect.

1) Pseudo-random Traffic Generation: Packets were injected into the interconnect using random variables which were generated using the *Boost* random number library [22]. Several random generators were used to generate traffic of the three types described in Section V-A. The first 3 variables, *P0*, *P1*, *P2* relate to the probability that a packet will be of type *L0*, *L1* or *L2* respectively. The probabilities use to generate each type of traffic are shown in Table I.

2) Task Graphs: A separate application called *gengraph* was implemented in C++ which randomly generates tasks graphs and maps them to a mesh. Each vertex in the graph corresponds to a task and each *directed edge* between two vertices indicates that the *source* vertex will send data to the *sink* vertex during simulation. The complexity of the task graph is controlled by several input parameters. The process of generating a task graph and mapping consists of the following steps:

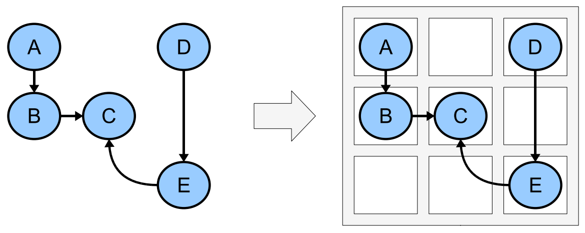

Fig. 5. Example of a task graph being mapped onto a 3×3 mesh.

1) A task graph is generated based on input parameters.
2) The task graph is mapped to a mesh.
3) The final mapping is written to a configuration file.
4) The configuration file is read by the simulator.

As we are only interested in generating application-like traffic on the interconnect, we have made some simplifying assumptions in our models:

- Only a single task is mapped to a node on the interconnect.
- Each task can send to (*out-degree*) and receive from (*in-degree*) zero or more tasks.
- The maximum *in/out-degrees* of a task can be constrained.

The simulator constructs a *send schedule* for each node based on the *out-degree* of the task that has been mapped to it. An simple example of a task graph and subsequent mapping is shown in Figure 5. When the configuration file is read by the simulator, the node which has been assigned task *A* will construct a send schedule consisting of a single entry, namely *B*. During the simulation, task *A* will send data to task *B* at randomly spaced intervals.

VI. Simulation Results

A. Comparison of the Hybrid and Mesh Architectures

In order to affirm the suitability of the hierarchical ring interconnect for global routing, we ran our simulator for increasing sizes of N and collected the results for both the hybrid and mesh architectures. Table II shows how the latencies varied for both architectures. What is quite interesting is the fact that the contrary to expectation, the latencies for local traffic (type L0) passing through the hybrid interconnect is actually less than that observed in the normal mesh topology. This phenomenon can be explained by the fact that for the case where the mesh-ring bridge component is located on a corner tile, the xy-routing algorithm causes global traffic (types L1 and L2) to be routed away from the center of the sub-mesh and towards the edges. The net effect of the xy-routing of global traffic towards the edges is that there is less congestion in the center of the sub-mesh thereby resulting in lower latencies for local traffic. Since local traffic will make up the majority of traffic in the system (assuming locality has been exploited), the net effect on the average latency on all traffic is positive, and in fact we can see from Table II that the average latency for hybrid architecture is less than that of the mesh.

TABLE II
Latencies (cycles) for the Hybrid and Mesh architectures for different values of N

	Hybrid				Mesh			
N	L0	L1	L2	Avg.	L0	L1	L2	Avg.
36	150	458	517	215	210	359	512	248
40	162	483	532	229	228	392	567	270
44	176	526	576	249	247	428	619	293
48	190	572	627	269	266	464	673	316

TABLE III
Hop Counts for the Hybrid and Mesh architectures for different values of N

	Hybrid				Mesh			
N	L0	L1	L2	Avg.	L0	L1	L2	Avg.
36	5	18	23	8	8	16	23	10
40	6	20	25	9	9	18	26	11
44	7	22	27	10	10	19	29	13
48	7	24	29	11	11	21	32	14

B. Exploiting Locality

One of the most important things to consider when measuring system performance is the fact that traffic patterns are not completely random. If systems designers simply assigned communicating tasks to nodes on the network randomly, the resulting system performance would be far from optimal. Exploiting *locality* has a significant impact on system performance. In our hybrid interconnect, there are two major ways to exploit locality. The first way is try to assign communicating tasks to the same sub-mesh so that the amount of local traffic (L0) dominates the amount of global traffic (L1, L2). This was modeled in VI-A by setting the probability of sending global traffic to be much smaller than that of sending local traffic. While effective at reducing global traffic, we still have not taken advantage of the placement of the bridge to the hierarchical ring interconnect in relation to stations which need to send/receive global traffic.

If the bridge component is placed at the corner of a sub-mesh, the system performance is improved by placing components that need to send global traffic closer to the bridge tile, thereby reducing the average number of hops required for global traffic to reach the bridge component. Conversely, components which send predominantly local traffic are placed as far as possible from the bridge tile. This will have the effect of decoupling global from local traffic and will have an overall positive effect on traffic latency. In order to model this behavior, we modified the probability that a station would send global traffic based on the radial distance from the bridge tile. Table IV shows the simulation results for a 36×36 size mesh where the latencies and hop counts decreased when stations closer to the bridge tile had a higher probability of sending global traffic as compared to stations farther away. The net effect is that the total amount of traffic being routed through the center of the mesh is reduced, leaving more resources free

TABLE IV

LATENCY (CYCLES) AND HOP COUNTS FOR UNIFORM AND NON-UNIFORM DISTRIBUTION OF TRAFFIC TYPE PROBABILITIES AT THE PES FOR THE HYBRID ARCHITECTURE OF SIZE $N = 36$

	Uniform		Non-Uniform	
	Latency. (cycles)	Hops	Latency (cycles)	Hops
L0	150	5	148	5
L1	523	18	410	16
L2	674	23	449	20
Avg.	278	10	211	8

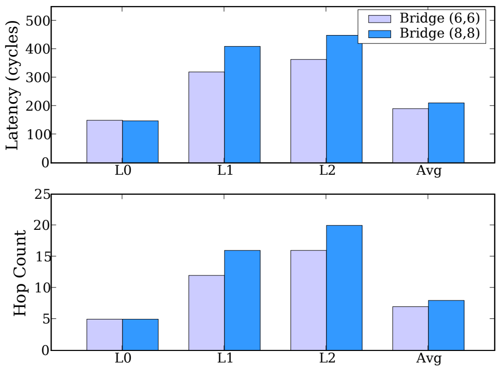

Fig. 6. Performance improvement of the hybrid architecture when the bridge component is moved away from the absolute corner location of the sub-mesh for local traffic.

In our experiments, we have chosen to place the bridge to the hierarchical ring interconnect at a corner tile of the mesh. While this has the effect that global traffic gets routed away from the center of the mesh, the number of input ports for a corner tile is limited to 2. It is obvious that a bridge component will have to handle more packets than a normal PE. If we move the bridge component away from the absolute edge of the sub-mesh while still keeping the bridge in the relative corner of the mesh, we would expect to see an improvement in the performance of the system. Figure 6 shows the improvement in performance for a $36x36$ mesh when the bridge is moved from the absolute corner position of $(8, 8)$ to $(6, 6)$. The latencies for local traffic stay relatively unchanged while the L1 and L2 latencies have decreased. These results make intuitive sense since stations which are more likely to send global data have been placed near the bridge location, which is located in the upper-right quadrant of the sub-mesh.

C. Enhanced Architecture

We implemented the enhanced architecture described in Section IV-C using our SystemC library of components. We mentioned previously that it is possible to implement the hybrid interconnect by making some modifications to the architecture from [20], but for simulation purposes, it was simpler to simply instantiate an extra ring-interface component for every sub-mesh and to connect them together to form the inner ring as shown in Figure 4.

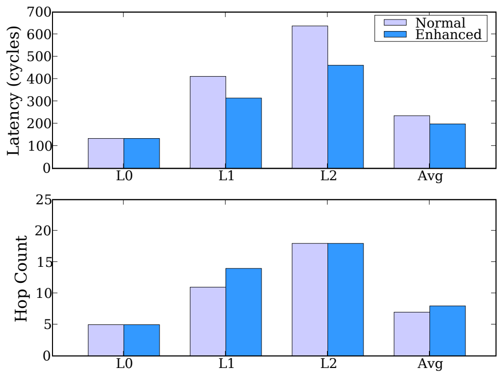

Fig. 7. Performance improvement of the enhanced hybrid over the normal architecture for $N = 32$.

We performed simulations for a mesh size of 32 and the results obtained are shown in Figure 7. It can be seen that the latency for the hybrid interconnect has improved for types L1 and L2 traffic. What is interesting is that hybrid interconnect has had a significant impact on type L2 traffic. Since traffic types L1 and L2 are unaffected by each other's stop signals, we see an overall decrease in their latencies.

D. Task Graphs

The *gengraph* program described in Section V-B.2 was used to generate a task graph with 1000 vertices and 700 edges. Figure 8 shows the results for the generated task graph when simulating the normal mesh and hybrid architectures.

The latencies and hop counts observed for the hybrid topology are lower than that for the mesh. Furthermore, type *L2* traffic is the most improved traffic type, which is consistent with the trend shown in Figure 2. Interestingly, Figure 8 shows that approximately 70% of traffic for the hybrid mesh was of type *L2*, indicating that the mapping did not exploit locality and resulted in a large amount of global traffic relative to the amount of local traffic. Irrespective of the poor mapping, the hybrid architecture still performed well.

VII. CONCLUSION

We have presented a hybrid architecture which uses a mesh topology for local routing and a hierarchical ring interconnect for global routing. A SystemC simulation model was used to simulate our hybrid topology and to compare the performance to the mesh architecture.

The partitioning of a large mesh into smaller sub-meshes leads to some interesting results. A well known problem with the mesh topology is the fact that traffic hotspots develop in the center of the mesh. In the hybrid topology, the global traffic is routed towards the edges of the sub-mesh thereby reducing the congestion in the center of the mesh and resulting

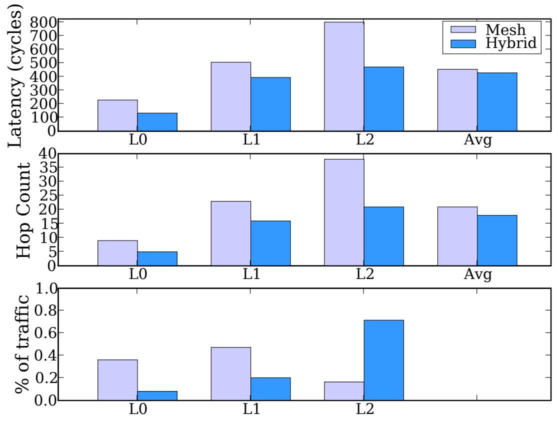

Fig. 8. Performance characteristics of a task graph mapped to the mesh and hybrid architectures for $N = 32$.

in reduced latencies for local traffic. By placing resources which need to send global traffic closer to the bridge station and taking advantage of locality, latencies and hop counts can be decreased. Furthermore, we showed that placing the bridge location away from the absolute corner of a sub-mesh, a further gain in performance could be achieved for global traffic without negatively affecting the latencies of local traffic. Lastly, we presented an enhanced version of our architecture which decreased latencies by providing separate virtual paths through the hierarchical-ring interconnect.

As the size of the mesh is increased, the sub-meshes also get larger, which results in quadratic increase in traffic that can go through the hierarchical ring interconnect. We can therefore conclude that there is some optimal size for the sub-meshes before performance will degrade to unacceptable levels due to congestion. The problem could be alleviated by simply scaling up the hierarchical ring, but we suggest that since a mesh size of about 36 resulted in similar performance characteristics for both the mesh and hybrid networks, it would be better to keep the sub-mesh sizes fixed while increasing the size of the hierarchical ring interconnect by adding an extra level to the hierarchy.

In future, we plan to investigate further the use of routing protocols that deal with path congestion for the mesh part of the topology.

REFERENCES

[1] L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm. *IEEE Computer*, 35, January 2002.

[2] W.J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the Design Automation Conference*, 2001.

[3] L. Benini and D. Bertozzi. Network-on-chip architectures and design methods. In Bashir Al-Hashimi, editor, *System On Chip: Next Generation Electronics*, pages 3–28. IEE Press, january 2006.

[4] D. Bertozzi. Network interface architecture and design issues. In G. De Micheli and L. Benini, editors, *Networks on Chips: Technology and Tools*, The Morgan Kaufmann Series in Systems on Silicon, chapter 5, pages 147–202. Morgan Kaufmann, July 2006.

[5] T. Ye, L. Benini, and G. Micheli. Packetization and routing analysis of on-chip multiprocessor networks. *Journal of Systems Architecture*, 50(2-3):81–104, 2004.

[6] C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance evaluation and design trade-offs for Network-on-Chip interconnect architectures. *IEEE Trans. Comput.*, 54(8):1025–1040, 2005. Student Member-Partha Pratim Pande and Senior Member-Andre Ivanov and Senior Member-Resve Saleh.

[7] P. Guerrier and A. Greiner. A generic architecture for on-chip packet switched interconnections. 2000.

[8] P. Pratim et al. Design of a switch for network on chip applications. In *ISCAS (5)*, pages 217–220, 2003.

[9] S. Kumar et al. A Network on Chip architecture and design methodology. *isvlsi*, 00:0117, 2002.

[10] D. Sigüenca-Tortosa and J. Nurmi. Proteo: A new approach to network-on-chip. In *Proceedings of IASTED International Conference of Communication Systems and Networks, CSN'02*, 2002.

[11] Scalable Coherent Interfafce. IEEE Standard 1596-1992, 1992.

[12] H. Samuelsson and S. Kumar. Ring road network on chip architecture. In *Proceedings of the IEEE Norchip Conference*, 2004.

[13] H. Singh, M.-H. Lee, G. Lu, N. Bagherzadeh, F.-J. Kurdahi, and E.-M. Chaves Filho. MorphoSys: An integrated reconfigurable system for data-parallel and computation-intensive applications. *IEEE Trans. Comput.*, 49(5):465–481, 2000.

[14] A. Brinkmann, J-C. Niemann, I. Hehemann, D. Langen, M. Porrmann, and U. Ruckert. On-chip interconnects for next generation System-on-Chips. In *ASIC/SOC Conference*, pages 211–215, 2002.

[15] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini. xpipes: a latency insensitive parameterized Network-on-Chip architecture for multi-processor SoCs. In *ICCD '03: Proceedings of the 21st International Conference on Computer Design*, page 536, Washington, DC, USA, 2003. IEEE Computer Society.

[16] W.J. Dally. Express cubes: Improving the performance of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 40(9):1016–1023, 1991.

[17] S. Bourduas, B. Kuo, Z. Zilic, and N. Manjikian. Modeling and evaluation of an energy-efficient hierarchical ring interconnect for System-on-Chip multiprocessors. In *NEWCAS*, 2006.

[18] R. Grindley et al. The NUMAchine multiprocessor. In *Proc. 29th Int'l Conf. on Parallel Processing*, pages 487–496, Toronto, Ontario, 2000.

[19] C.A. Zeferino, M.E. Kreutz, L.C., and A. Susin. A study on communication issues for Systems-on-Chip. In *Proceedings of the 15 th Symposium on Integrated Circuits and Systems Design (SBCCI 02)*, 2002.

[20] S. Bourduas J-S. Chenard and Z. Zilic. A RTL-level analysis of a hierarchical ring interconnect for Network-on-Chip multi-processors. In *Proceedings of International System-on-a-Chip Design Conference (ISOCC)*, October 2006.

[21] G. Ravindran and M. Stumm. A performance comparison of hierarchical ring and mesh-connected multiprocessor networks. In *3rd IEEE Symposium on High-Performance Computer Architecture*, pages 58–69, 1997.

[22] Boost random number library. Available at: http://www.boost.org/libs/random/index.html.