Predicting the Performance of Application-Specific NoCs Implemented on FPGAs

Jason Lee and Lesley Shannon
Simon Fraser University
Burnaby, BC, Canada
{leeaj,lshannon}@sfu.ca

ABSTRACT

Modern FPGAs are able to implement complex systems such as Systems-on-Chips (SoCs) and Networks-on-Chips (NoCs). Appropriate NoC topology choices for ASICs have been investigated and typically topologies that can be easily mapped to a two-dimensional fabric are used to reduce chip area and ensure electrical characteristics. However, for FPGAs, each device's size and routing fabric are fixed. Since these resources exist independent of use, the choice of topology is only limited by the performance of the NoC itself. In this work, we investigate how topology characteristics impact a NoC's performance on an FPGA. From this analysis, we have created an analytical model that describes the maximum operating frequency of a NoC as a function of the topology's network parameters. This model is in the form of a simple equation that is accurate to within 4.68% across a range of topologies, chip sizes, and device families. It demonstrates how an FPGA's prefabricated routing interconnect provides increased freedom in the selection of application-specific topologies. Furthermore, it can also be used by designers for topology design space exploration before implementation.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; C.2.1 [**Computer-Communication Networks**] Network Architecture and Design – *Network Topology*.

General Terms

Measurement, Performance, Design, Experimentation, Theory, Verification.

Keywords

NoCs, FPGAs, Performance, Topologies, Routability, Application-Specific, Homogeneous, Heterogeneous.

1. INTRODUCTION

As the logic capacity of modern Field-Programmable Gate Arrays (FPGA) increases, they are used to implement much more complex systems than ever before. These complex systems often take the form of a System-on-Chip (SoC) in which individual blocks, or *nodes*, are connected using a Network-on-Chip (NoC). Although a shared bus may be appropriate for simple systems [1,2], this communication infrastructure does not scale well for very complex systems. As the number of potential bus masters and slaves increases, the shared bus may suffer from increased bus contention, slowing down throughput and limiting bandwidth. Thus, more complex communication networks are becoming increasingly attractive for many-node systems.

A NoC is characterized by its *topology*, which defines the connectivity pattern between nodes [3]. There are many possible choices of topologies for NoCs; however, Application-Specific Integrated Circuits (ASICs) typically employ a mesh-style topology. A mesh topology maps well to an ASIC's two-dimensional implementation platform, providing control over the network's electrical characteristics [4, 5]. More complex networks, including star and hypercube topologies, implemented on an ASIC lead to increased chip area and an increasingly difficult routing task as the number of nodes grow.

Previous work has shown, however, that when implemented on an FPGA, these more complex networks are feasible [6]. The primary reason is that modern FPGAs are over-provisioned with routing; that is, FPGA architects provide significantly more routing than is needed for the "common case" to ensure a high fitting rate by the Computer Aided Design (CAD) tools for customer designs. The work in [6] showed that due to this over-provisioning, more complex topologies such as the star and hypercube are possible. In fact, it suggested that in some cases these types of networks may be preferable to a mesh, since they have better network latency and bandwidth characteristics, yet can still be implemented easily on a modern FPGA.

This result implies that designers have the increased freedom to select more complex topologies when implementing NoCs on FPGAs as opposed to ASICs. To leverage these findings, however, a more concrete understanding of the performance of various network topologies on FPGAs is required. This paper provides a step in this direction. In particular, this paper provides *an analytical model in the form of a simple equation that describes the maximum operating frequency (performance) of a NoC as a function of various network parameters related to the overall network topology*. Such a model is important for two reasons. Firstly, it quantifies the effects of specific network parameters on performance, and thereby the suitability of network topologies for implementation on an FPGA. This is an important first step to understanding the flexibility and limitations of mapping application-specific network topologies to an FPGA's prefabricated routing interconnect using its commercial tool flow. Secondly, it provides guidance to a designer during early design space exploration, when a suitable network topology is being chosen. To provide concrete results, we calibrated our model for the Xilinx Virtex 2 Pro, Virtex 4, and Virtex 5 FPGA device families. We then verified it with over 2000 different sets of experiments for these device families, along with Xilinx's newest Virtex 6 FPGA device family. Across the range of topologies represented by these experiments, we found that the accuracy of the operating frequency predicted using the model has a geometric mean error of 4.68%.

The remainder of the paper is structured as follows. Section 2 provides background on previous NoC work, Section 3 describes our proposed analytical framework, and Section 4 explains our experimental environment. Section 5 illustrates our experimental results, Section 6 describes the performance predictor we developed, and Section 7 verifies this predictor. Finally, Section 8 concludes the paper and comments on future work.

2. BACKGROUND

While little has been done to characterize the routability and performance of NoC architectures on FPGAs, there has been research investigating the use of FPGAs for NoCs [7] and appropriate switch architectures. Kapre investigated a high-performance packet-switched on-chip network on FPGAs [8]; Mehta explored well engineered, highly scalable time-multiplexed FPGA networks [9]. These studies provided a measure of performance for two types of network interfaces measured on the Xilinx XC2V4000 and focused on selecting appropriate switch architectures. When used in conjunction with our work, selecting appropriate switch architectures could improve the overall throughput of a NoC topology.

In addition, Mak et al. performed a survey of on-chip architectures including different NoC topologies. The paper identified that due to the unique requirements of different applications, there exists a problem of searching for optimal communication architectures from a huge design space since choices are often performed ad-hoc. Our work takes a step in this direction providing a method of predicting the performance of NoCs to enable easier design space exploration.

Possible architectural changes to an FPGA's routing fabric have also been investigated to better support NoCs. Francis et al. demonstrated that fine-grain, time-division-multiplexed wiring outperforms conventional wiring for networks on FPGAs [10]. Goossens et al. proposed a dedicated NoC interconnect fabric [11]. These hardwired NoC fabrics show improved performance, however, reduce the amount of configurable logic. Whereas these investigations propose possible changes to FPGA architecture to support NoCs, our objective is to understand how the existing interconnect fabric and CAD tools constrain NoC performance on commercial FPGAs.

To fully exploit FPGA resources, complex CAD algorithms are used to place and route NoCs on FPGAs. Research has been done to create automated design flows that generate NoCs for FPGAs. Kumar et al. developed an automated design flow to instantiate Multi-Processor Systems on Chip (MPSoC), with a NoC communication scheme [12] similar to the work done in [6]. The design flow provides a high level of abstraction, reducing the design time needed for these types of systems. A framework based on Xilinx Embedded Development Kit (EDK) is also presented by Lukovix et al. [12]. In our research, we employed an updated verson of the tool flow in [6] to automatically generate the wide variety of NoC systems required for this investigation. Our experimental setup and CAD flow are described in the following section.

The most closely related work to our current investigation includes a preliminary study that suggested the routability and performance of homogeneous multiprocessor NoCs displayed different characteristics on FPGAs than ASICs [6]. A study determining that the heterogeneity of network nodes did not impact NoC performance followed [14]. The remainder of this paper investigates the effect of a topology's routing demand on NoC performance.

3. PROPOSED ANALYTICAL FRAMEWORK

The objective of this work is to create an analytical model that describes the maximum operating frequency of a NoC. Our overall approach is as follows. We arbitrarily chose a 8-node ring topology with a 32-bit link-width as a *baseline architecture*, and denote the maximum frequency of this baseline architecture implemented on a given FPGA as F_{base}. For different NoC architectures, we then scale F_{base} using two factors:

$$F_{pred} = k_{GRD} k_{LRD} F_{base} \quad [1]$$

where the terms k_{LRD} and k_{GRD} are functions of the NoC topology, the link width, and the number of nodes in the NoC.

The first scaling factor in the above equation, k_{LRD}, models the impact of local routing demand. Local routing demand is defined by the routing requirements of a single network node. Topologies with a high average node degree and larger link widths will have more congestion around their network interfaces. The CAD tools resolve this congestion by either using non-direct routes, or by spreading out the logic related to those nodes with high node degree. The impact is a decrease in the maximum frequency of the network; the magnitude of this decrease will be encapsulated in k_{LRD}.

The second scaling factor in the above equation, k_{GRD}, models the impact of global routing demand. Global routing demand is characterized by the routing requirements of the entire NoC. Topologies that have a high average node degree will have more links in the overall topology. This is more pronounced as the number of nodes in the network increases. As the number of links increases, the difficulty in routing these connections increases, again leading to a reduction in the maximum frequency of the network. The magnitude of this decrease will be encapsulated in k_{GRD}.

The remainder of this paper aims to derive closed-form expressions for k_{LRD} and k_{GRD} using analytical methods, and empirical curve-fitting for the coefficients. As explained, both global and local routing demand can be encapsulated using three properties: average node degree, link width, and number of nodes. The following sections analyze these properties to derive our analytical model. In Section 5, we describe a number of experimental investigations that justify our analytical approach. In Section 6, we then derive the equations, and use an additional set of experiments to calibrate the model to Xilinx Virtex 2 Pro, Virtex 4, Virtex 5, and Virtex 6 devices using a curve fitting approach. The accuracy of the tuned model for these devices is then measured in Section 7.

4. EXPERIMENTAL ENVIRONMENT

As explained in the previous section, the form of the equations will be derived analytically. We will use experimentation, however, for three purposes. First, in Section 5, we will use experiments to justify our overall approach in deriving our analytical model. Second, in Section 6, we will use experimental results to calibrate

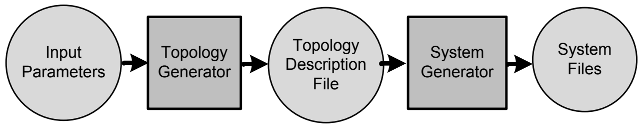

Figure 1. System generations tool flow

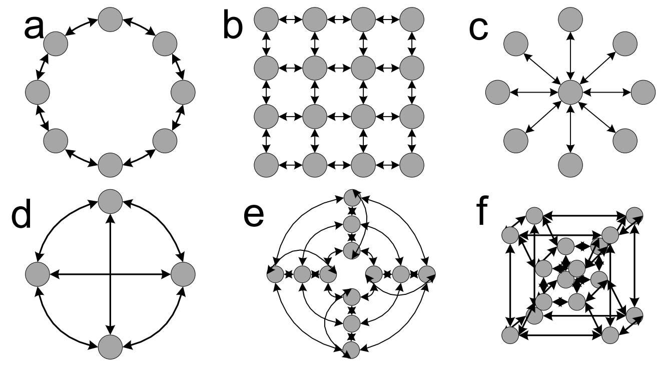

Figure 2. NoC topologies: (a) ring, (b) mesh, (c) star, (d) fully, (e) torus, and (f) hypercube

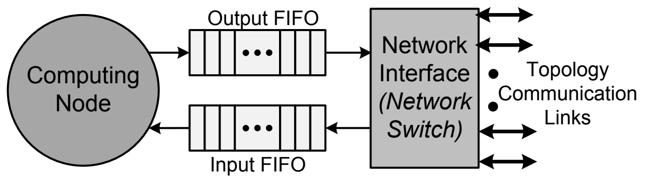
Output FIFO
Computing Node
Network Interface (Network Switch)
Topology Communication Links
Input FIFO

Figure 3. Network node

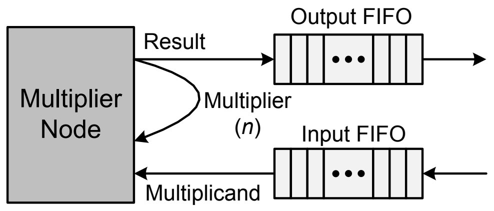

Figure 4. Scalable multiplier computing node

the coefficients for our model. Finally, in Section 7, we use a separate set of experiments to evaluate the accuracy of the model. In this section, we describe the methodology used in these three sets of experiments.

4.1 Benchmark Circuits

In our experiments, we utilize a large set of benchmark circuits using an automated generator. The generator supports six different network topologies and several different node sizes. The use of such a generator allows us to run many more experiments than would be possible using "real" benchmark circuits; this, in turn, allows us to isolate the impact of each NoC parameter on the overall performance of the system.

Figure 1 shows the tool flow used to generate the benchmark circuits[1]. The topology *generator* outputs a description file that defines the connectivity between each network node based on a given set of *input parameters*. The *system generator* parses the *topology description file* and produces the necessary *system files*, which include the Verilog files that describe the NoC and the CAD flow constraints. These *system files* are then used by the FPGA CAD flow to generate the final NoC implementation on the FPGA. The system and topology generators are capable of producing NoCs based on a set of configurable parameters, including:

- *Number of Nodes*: the number of nodes in the overall system
- *NoC Type*: heterogeneous or homogeneous
- *Node Degree:* the number of links to a given node
- *Topology type*: ring, mesh, star, fully-connected, torus, hypercube (shown in Figure 2), or application-specific topologies (represented by randomly generated topologies).
- *Average Node Degree*: the average number of links to each node in a topology
- *Link-width*: the width of the point-to-point links in terms of # of bits

While it is possible to optimize our NoC benchmark circuits for performance (e.g. pipelining the topology communication links), our analysis was solely focused on looking at wire length delay. We do not make any attempts at improving NoC performance through architectural or topological changes.

4.2 Network Nodes

Figure 3 illustrates the structure of the network node used in our experiments. Each node consists of a computing element, two synchronous 16-word deep FIFOs, and a network interface. The network interface is connected to other nodes through the *topology communication links*. Since we are interested in the performance achieved by the CAD flow's ability to leverage the over-provisioned routing resources to implement the links that define the NoC topology, and not the NoC's performance in terms of bandwidth and throughput, we use a lightweight network switch. Our network switch is a simple packet switch that broadcasts an address control packet to all its linked neighbors. A receiving switch only reads the control packet and subsequent packets if the address matches its own and otherwise ignores them. Due to their simplicity, these switches are not capable of multi-hop communication and are only capable of sending to and receiving from their directly connected neighbors. In order to completely isolate the performance of the network node from the *topology communication links*, the output from the network interface is latched.

Each computing node consists of a multiplier, as shown in Figure 4. The multiplier consists of two-bit partial-products pipelined into stages. Actual implementations would contain more complex computing nodes, such as the previously used MicroBlaze soft processor [7]. However, using such a computing node in this study would complicate our characterization of performance degradation due to the CAD flow's mapping of the topology links to the routing fabric. If a MicroBlaze were used as the compute node, then in small NoC systems, the critical path would be in the node itself, and not in the network links. Since we wish to focus specifically on the network connectivity in this study, this is not desirable. In addition, designs employing a MicroBlaze would not be portable to non-Xilinx FPGAs.

Another advantage of using a multiplier is that the size of the computing node can be adjusted, allowing us to experiment with different link widths, and topologies with varied network node sizes. As shown in Figure 4, the multiplicand of the multiplier node is equal to the link's width, and the multiplier is equal the

[1] The topology generator and system generator tools are available for download from: http://www.ensc.sfu.ca/~lshannon/students.html

lower *n* bits of the result. We consider both *homogeneous* and *heterogeneous* NoC types; in a homogenous network, all nodes are of the same size, while in a heterogeneous network, different node sizes exist. As the multiplicand always remains fixed to the link's width, the multiplier (*n*) is varied to scale the resource usage. Table 1 lists the multiplier sizes used in our homogeneous experiments. *Mult* is defined as the width (*n*-bits) of the multiplier, and *Link Width* is the multiplicand width. We used five different link widths, with three different multiplier widths for each link-width. The baseline multiplier *Mult_Base* was chosen to have approximately the same resource usage as a MicroBlaze on a Virtex 5. *Mult_Half* is approximately half the size, and *Mult_Double* approximately twice the size. As the link width changes, the multiplier width also needs to be adjusted to maintain approximately constant resource usage for the computing node. The percent variation in resource usage for each node type is 7.12% with a standard deviation of 3.2%.

To generate heterogeneous NoCs, we kept the link width (multiplicand) fixed for the design and varied the network node size by scaling the multiplier width. We use three types of heterogeneous NoCs (*mult_small*, *mult_full*, and *mult_large*) generated using a range of multiplier node sizes defined by a minimum and maximum multiplier width. The size of each multiplier node in a heterogeneous topology is chosen at random, and uniformly distributed across the range of multiplier widths defined by the heterogeneous NoC type and fixed link width. Table 2 lists the range of sizes used in our heterogeneous experiments for varied link widths.

4.3 Experimental Methodology

To tune our model parameters, we first ran training experiments using Xilinx EDK 10.1.02 with the Virtex 2 Pro, Virtex 4, and Virtex 5 FPGAs listed in Table 3. Using these experiments, we extrapolated coefficients for the equations in our analytical model explained in Section 6. In order to evaluate the accuracy of our model, we used the Virtex 4, Virtex 5, and Virtex 6 FPGAs listed in Table 3, with EDK 11.2. Although we ran ~2400 training experiments for each device, the results shown in this paper will focus primarily on the Virtex 5 LX330 due to space limitations. However, when normalized to account for the performance improvements due to new device family technology, we previously found that the results among different families only varied by 3.5% and among devices within a family varied by 4.2% [6]. In order to obtain accurate results, each design is synthesized multiple times. Initially, we ran experiments using the Xilinx Xplorer utility, which synthesizes designs using known place and route parameters to provide the best results. However, we found that the utility resulted in extremely long run times (on the order of 5-7 days) for each experiment. Therefore, we only used the utility to form a baseline of comparison for the remaining experiments.

Rather than using the Xplorer utility, we are able to approximate Xplorer's process by synthesizing designs multiple times using different seeds, with the maximum operating frequency averaged over each run. The number of iterations is determined by repeating iterations until at least five iterations are run and the change in the average result over all runs is less than 5%. When the results did not converge (which occurred in less than 8% of our experiments), we set an upper bound on the experiments to ten iterations. This method had an average variation of 2.1% to the Xilinx Xplorer utility. A design is deemed *unroutable* if the design would not route for at least eight out of the ten iterations.

Table 1: Homogeneous multiplier sizes

Link Width	**uBlaze**	**Mult_Half**		**Mult_Base**		**Mult_Double**	
	LUTs	*Mult*	*LUTs*	*Mult*	*LUTs*	*Mult*	*LUTs*
48		2	347	4	658	6	1312
40		2	396	4	703	6	1487
32	629	4	371	6	694	8	1429
24		6	328	9	652	12	1393
16		8	311	12	614	16	1374

Table 2: Heterogeneous multiplier size ranges

Width	**Mult_Small**	**Mult_Full**	**Mult_Large**
	Range of Mults	*Range of Mults*	*Range of Mults*
48	2,4	2,4,6,8	4,6,8
40	2,4	2,4,6,8	4,6,8
32	2,4,6	2,4,6,8,10	6,8,10
24	4,6,8	4,6,8,10,12,14	10,12,14
16	6,8,10	8,10,12,14,16	12,14,16

Table 3: FPGA devices

Family	**Devices**
Xilinx – Virtex 2 Pro	XC2VP100-6
Xilinx – Virtex 4	XC4VLX200-11 XC4VLX160-11 XC4VLX100-11
Xilinx – Virtex 5	XC5VLX330-2 XC5VLX220-2 XC5VLX155-2 XC5VLX110-2
Xilinx – Virtex 6	XC6LX240T-2 XC6LX350T-2

5. GENERAL NOC PERFORMANCE TRENDS ON FPGAS

Before experimentally deriving the *specific* constants to calibrate our model to the Xilinx device families, we present the general performance trends exhibited by specific NoC topologies that strain local and global routing demand on FPGAs. As previously stated, we use the results from the Virtex 5 LX330 to highlight these key trends (unless otherwise specified), as it has similar trends to all the devices listed in Table 3.

5.1 Previous Research

Previous work showed that homogeneous multiprocessor networks-on-chip exhibited different characteristics on FPGAs when compared to ASICs [6]. This was extended to show that: 1) the number of nodes is more important than node size or node heterogeneity; 2) varying the node size does not impact performance as long as the node is not the critical path; 3) topology has a greater effect on performance than resource usage; and 4) resource usage only becomes significant as it approaches 80% where larger NoCs consistently fail to route [7]. In this paper, these results are incorporated into the expressions of k_{LRD} and k_{GRD} in terms of the NoC's number of nodes (*N*) and topology.

5.2 Local Routing Demand

As described in Section 3, the local routing demand factor (k_{LRD}) characterizes the impact of the routing requirements of a single network node. These routing requirements are related to the total number of wires connecting the node's network interface to the communication network. The total number of wires is dependent on both the node degree (*ND*) and link width (LW) of the network. In this section, we illustrate the effects of local routing demand on performance by isolating the effects due to node degree and link width. We use the star topology with a fixed 32 bit link width to demonstrate the effect of node degree on a single node; a star topology's central node has $ND = N - 1$, while for all other nodes $ND = 1$. Figure 5 shows the performance of the star topology using our three homogeneous and three heterogeneous NoCs. As previously stated, the node size for our heterogeneous star topologies are generated at random. Therefore, we do not look at specific heterogeneous cases such as only increasing the size of the central node.

As the graph shows, although the central node in a star has an extremely high node degree, very large star topologies are routable as long as there are sufficient resources for the overall system. As the number of nodes increases, more links are added to the central node. This increases the network interface's connectivity, causing the CAD tools to distribute the network interface across the FPGA fabric to enable multiple links to be routed to the central node (this was verified by analysing various star topology sizes using FPGA Editor). As the network interface spreads out, longer wires are used to successfully route the design causing a severe degradation in performance as shown in Figure 5.

The numeric labels in Figure 5 indicate the logic resource usage of the FPGA for 80, 96, and 112-node systems. The labels shown above the performance line are for *mult_small* NoCs and those below the line represent *mult_large* NoCs. Although there is a large difference in resource usage, the overall performance is roughly the same for all heterogeneous systems considered. This suggests that the resource usage does not have a significant impact on performance until routability fails at >80% resource usage, which is consistent with the results from our previous work [7]. Since resource usage does not impact performance, the loss in performance is primarily a function of how well the tools manage the wiring demand of the central node. The tools attempt to limit congestion by distributing the network interface of the central node. As a result, the node degree has a significant impact on the topology's performance. However, due to the availability of global routing resources, this degradation eventually flattens out when the network interface is spread over almost the entire FPGA (~64 nodes on the Virtex 5 LX330) and larger star topologies will continue to be routable with constant performance, as long as there are sufficient logic resources.

To demonstrate the effect of varying link width on local routing demand for a fixed node degree over an increasing number of nodes, we use the ring and torus topologies. As seen in Figure 6, increasing the link width has a linear impact on performance due to the increase in the number of wires required to connect two network nodes to each other. Each line in Figure 6 illustrates the average performance of our three heterogeneous and three homogeneous ring and torus topologies from 8 to 128-nodes for a fixed link width of 16, 24, or 32 bits. The dotted lines represent the torus topologies, and the solid lines are the ring topologies. As can be seen in Figure 6, generally each line is monotonically decreasing except for a few cases. This can be attributed to the CAD flow, as the CAD algorithms are random, and there still exists a certain amount of unpredictability in performance. This can also be seen in the following sections, but with our exhaustive experiments, small variations can be affectively "averaged" out over a large exploration space.

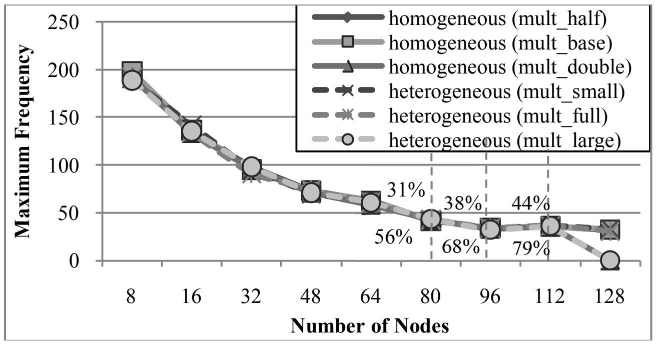

Figure 5: Star topology performance on Virtex 5 LX330

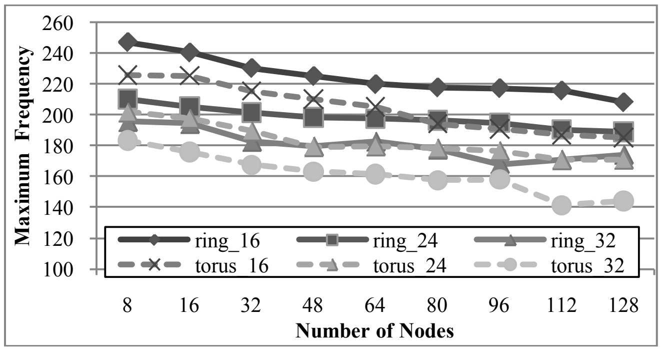

Figure 6: Varying link widths on Virtex 5 LX330

As seen in Figure 6, as link width decreases, performance increases for both the ring and torus topology. For the ring topology, decreasing the link width from 32 to 24 bits resulted in an 8.1% performance increase, while a change from 32 to 16 bits resulted in a 14.2% increase. The torus topology exhibited slightly larger increases; a change in link width from 32 to 24 bits increased performance by 9.3%, and a change from 32 to 16 bits increased performance by 16.8%. As seen in the previous analysis, the node degree has a significant impact on performance. Therefore, since each node of the torus topology has twice the node degree of a ring, decreasing the link width by 8 bits for the torus topology, results in 2x the reduction in wires (*ND**LW) compared to that same change for the ring topology. This results in link width having a greater impact on performance for higher node degrees.

Another interesting observation from Figure 6 is that a ring topology with a 32-bit link width has a lower maximum frequency than a torus topology with 16-bit link width. Both networks have the same number of total bits incident to each node, and hence the local routing demand should be the same. Using FPGA Editor, we have observed that the tools tend to route the wires in a single link using the same global route (along the same set of channels). Thus, networks with larger link widths create a harder routing problem.

5.3 Global Routing Demand

As described in Section 3, the global routing demand factor (k_{GRD}) characterizes the impact of the routing requirements of the entire topology. These routing requirements are related to the total number of network links in the system. The total number of network links is equal to the number of nodes in the system (*N*) multiplied by the average node degree (*AND*). To show the effects of changing the total number of links on performance, we vary *AND* and *N* using the fully connected topology (a regular topology where $AND = N - 1$).

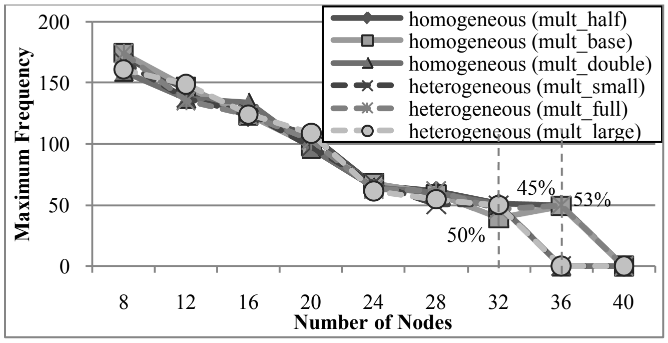

Figure 7: Fully connected topologies on Virtex 5 LX330

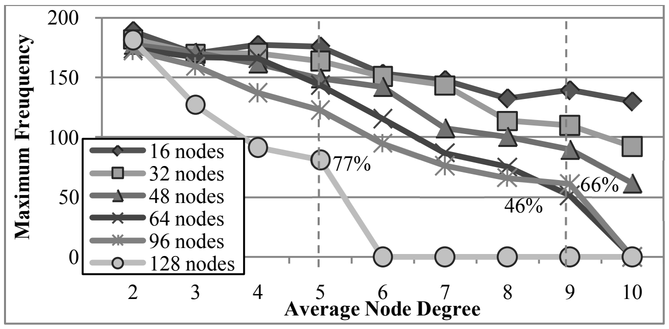

Figure 8: Heterogeneous random topologies on Virtex 5 LX330

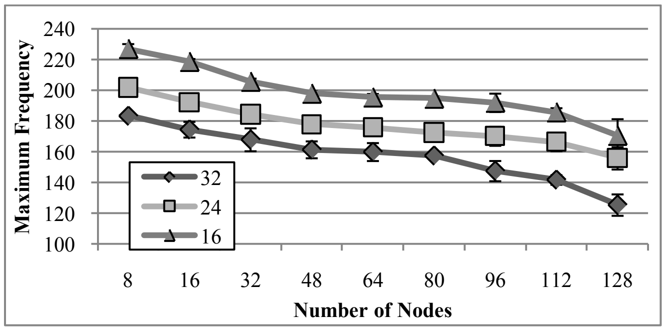

Figure 9: Performance of torus topology and random topologies (*AND* = 4) on Virtex 5 LX330

Figure 7 shows the performance on a Virtex 5 LX 330 for fully connected topologies with 32-bit link widths as a function of the number of nodes. We chose the 32-bit link width systems to allow comparisons with Figure 5. As a fully connected topology grows in size, the impact on performance is severe as the total number of links increase quadratically. As seen in Figure 7, for up to 20 nodes, performance has a rapid linear degradation in performance. Performance then drops dramatically and flattens out at 24 nodes before routing eventually fails. As the average node degree increases up to 24 nodes, the network interface for each node is distributed to allow the CAD tools to use the available routing resources (much like the star topology). However, this is a much more difficult problem than that for a star topology since there are many more links in a fully connected network. Therefore, the designs become unroutable *before* running out of logic resources, as shown in Figure 7 where the largest fully connected topologies capable of routing used only ~50% of the logic. Fully connected topologies with 24 and 16-bit link widths exhibited the same trends, with a significant performance drop at ~28-30 nodes and routing failure at 36-40 nodes.

Comparing Figure 5 and Figure 7 reinforces the greater impact of global routing demand than local routing demand; we see that the performance of the two topologies varies by only 8.2% up to 20 nodes and then diverges as the star topology's performance flattens out and the fully connected topologies performance continues to decline. This suggests that up to 20 nodes, the CAD tools are able to leverage the global routing resources to facilitate fully connected topologies. However, above this point the impact due to the global routing demand of all nodes outweighs that of the local routing demand of each node, resulting in rapid performance decline and routing failure.

Fully connected topologies show that the total number of links has a significant impact on performance by varying the number of nodes (*N*) and average node degree (*AND*), without isolating the two variables as they both increase at the same rate. In order to isolate the effects of average node degree from the number of nodes, we create and map benchmarks containing random topologies ranging from 16 to 128 nodes with average node degrees of 2 to 10. Figure 8 shows the performance results for heterogeneous NoCs utilizing the *mult_full* range of node sizes, where each line represents random topologies with a fixed number of nodes. The topologies with 16 nodes have the highest performance, which degrades as the number of nodes increases to 128. For a fixed number of nodes, the performance decreases almost linearly as the average node degree increases, until routing fails. The rate of degradation increases as the number of nodes increases, as shown by the increase in the slope's magnitude for each line in Figure 8. This is because for a fixed number of nodes, *0.5N* links are added as the average node degree increases by one. Thus, we expect a greater drop in performance for systems with more nodes.

Much like the fully connected topology, systems with high average node degrees failed routing before logic resource usage exceeded 80% total resources. For example, for a 64-node random topology, the highest average node degree that consistently routed is nine, using 46% of the logic. 96 and 128-node random topologies are routable with an average node degree of nine and five respectively, but only use 66% and 77% of the total resources, respectively. This is because for high average node degree, the stresses of global routing demand requirements on the CAD tools for the FPGA fabric cause routing to fail well before logic utilization approaches 80%.

5.4 Regularity

The previous subsections focused on regular topologies such as the torus, ring, and fully connected topologies. Many applications may benefit from an irregular topology which is optimized specifically for that application. In this section, we demonstrate these random topologies can also be efficiently implemented on an FPGA.

Figure 9 illustrates the average performance of a *mult_full* heterogeneous torus topology in comparison to six random topologies with the same number of nodes (*N*), the same average node degree (*AND*), and the same link width (*LW*). Each line corresponds to the performance of the torus topology for a fixed link width and the error bars represent the performance variation exhibited by the corresponding random topologies. The error bars show an average variation of 2.3% and a maximum variation of 4.8%. These results suggest that an expression written in terms of only the number of nodes (*N*), average node degree (*AND*), and link width (LW) can apply to irregular application-specific networks as well as regular networks. The derivation and calibration of the factors k_{GRD} & k_{LRD}, in terms of these variables, are described in the following section.

6. PERFORMANCE MODEL

In this section, we derive the equations for k_{GRD} & k_{LRD} and tune the coefficients using experimental curve fitting to predict the maximum operating frequency of a NoC implemented on Xilinx FPGAs. We use experiments run on three different Xilinx FPGA families. Each device family has a unique frequency, F_{base}, for the 8-node ring topology with 32-bit link widths that we use as our baseline architecture. For our training data, the baseline frequencies used to generate our equation are 190MHz for Virtex 5, 145MHz for Virtex 4, and 110MHz for Virtex 2 Pro. Recalling from Equation [1] that we chose to express the model in terms of k_{GRD} & k_{LRD} multiplied by the F_{base}, the predicted frequency represents the remaining percentage of F_{base} after accounting for changes in global and local routing demand with respect to our baseline architecture. Therefore, if we wish to "predict" the performance of our baseline 8-node ring topology with a 32-bit link width, we expect $k_{GRD} = k_{LRD} = 1$ and $F_{pred} = F_{base}$.

6.1 Local Routing Demand

Local routing demand (k_{LRD}) describes the performance impact from the perspective of a single network node. As discussed in Section 5.2, local routing demand is directly correlated with the node degree (*ND*) and link width (LW) of the network node. Since we want to model both regular and irregular topologies, we approximate the node degree of individual nodes in the topology as being equal to the average node degree ($ND \approx AND$). However, as discussed previously in Section 5.3, the average node degree also affects the global routing demand of the entire system. Therefore, we chose to only encapsulate the effects due to a change in link width and how it is magnified by average node degree in k_{LRD}. The overall effect of average node degree will be encapsulated in k_{GRD} and discussed in the following section.

Increasing the link width increases the number of wires to a node, causing the CAD tools to distribute the network interface, creating longer wire lengths and impacting performance. As average node degree increases, a fixed change in link width will result in more wires being added for higher average node degrees, thus further impacting performance. Therefore, k_{LRD} is expressed as a change in link width amplified by the average node degree. The expression for local routing demand is shown below:

$$k_{LRD} = LRD_{SL}\Delta LW + 1, \qquad [2]$$

where ΔLW represents the change in link width given by ($\Delta LW = LW_{F_{pred}} - LW_{F_{base}}$ where $LW_{F_{base}} = 32$). If $\Delta LW = 0$ then there is no change in link width and ΔLW reduces to zero resulting in $k_{LRD} = 1$. The slope (LRD_{SL}) describes the rate of change of performance due to link width for a *fixed* node degree. In order to calculate LRD_{SL}, Figure 10 shows a family of lines representing the performance of all topologies as a percentage of F_{base} with respect to changing the link width, where each line in the figure represents topologies with a constant average node degree. The horizontal axis indicates the change in link width from the 32-bit link width in the baseline degree, a constant change in link width results in a constant change in the number of wires added to a network node. Since the number of wires is directly proportional to the link width

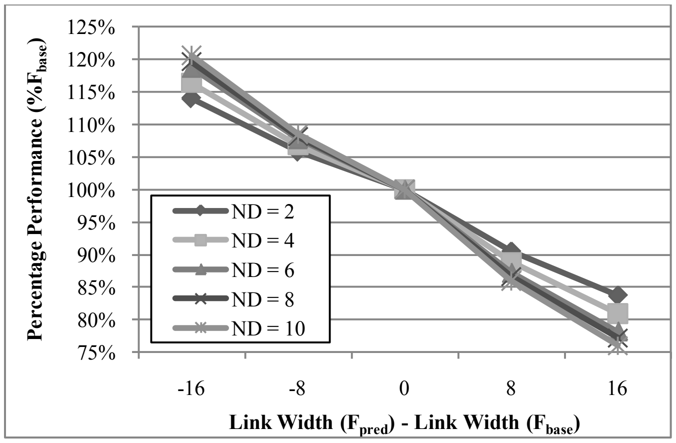

Figure 10: Performance variation due to a change in link width

(# of wires = $ND*LW \approx AND*LW$), the impact on performance due to link width has the linear relationship given in Equation [2] for a constant value of *AND*.

As shown in Figure 10, changing the average node degree changes the *impact* of changing the link width on performance, corresponding to the slope of each line (i.e. LRD_{SL}). For example, if a node has $AND = 2$, then increasing the link width by 8 bits requires 16 additional wires to be routed to that node. However, if a node has $AND = 3$, then 24 more wires must be routed to that node. Therefore a topology with an *AND* of three would result in more wires being routed, and thus an increased local routing demand (on average), than the same change in link width for a topology with an average node degree of two. Thus, the higher the average node degree, the greater the effect link width has on performance. Since the value of LRD_{SL} is directly proportional to the average node degree, the slope remains constant for a fixed node degree, and decreases linearly as the node degree increases. We isolated the slope (LRD_{SL}) of each line in Figure 10 and mapped the change in slope as a linear relationship, deriving the slope and intercept values through curve fitting to be:

$$LRD_{SL} = (-0.0012AND - 0.0046) \qquad [3]$$

6.2 Global Routing Demand

Global routing demand (k_{GRD}) is characterized by the routing requirements of the entire system. Independent of network node, type, and size, global routing demand is directly affected by the total number of links in the system (*N***AND*). A link is defined as a single unidirectional link between nodes, thus a channel between two nodes has two links. In order to model these effects, we first consider how average node degree impacts performance and how the number of nodes magnifies this effect. An increase in average node degree results in more links being added to the topology since $\Delta Links = N * AND$. As the number of nodes in a system increases, a change in average node degree results in even more links being added. The number of links affects performance by increasing the number of wires in the system, thus expanding the network over the fabric and effectively decreasing performance. Since k_{GRD} is directly correlated with the # of links, the effect due to k_{GRD} is shown below:

$$k_{GRD} = GRD_{SL}(AND - 2) + GRD_{INT} \qquad [4]$$

The linear equation is characterized by a slope, GRD_{SL}, and an intercept, GRD_{INT}, which vary when the number of nodes change. Since k_{GRD} should be equal to one for an eight node ring topology (N=8, AND=2), the $(AND - 2)$ terms reduces the slope to zero and the intercept must be equal to one when N=*1*. In order to determine GRD_{SL} and GRD_{INT}, we analyse the effect of varying average node degree for sets of NoCs with fixed numbers of nodes. Figure 11 shows the remaining percentage of F_{base} as a function of average node degree for a given number of nodes. Each line corresponds to a set of topologies with the same number of nodes and link width at varying average node degrees as expressed in Equation [4]. For a fixed number of nodes, as the average node degree increases, a constant number of wires is added to the network. This number of wires is directly proportional to the average node degree, thus a change in average node degree from 2 to 3, or 4 to 5 will always result in the same number of additional wires being added to the system.

When moving between the different lines in Figure 11, the slope (GRD_{SL}) and intercept (GRD_{INT}) change to reflect the rate of performance loss (slope) and maximum possible performance (intercept) for that number of nodes. Since the slope (GRD_{SL}) describes the rate of performance decrease when changing the average node degree for a *fixed* N, it varies when N changes. For example, changing the average node degree of a system with 128 nodes should have a higher impact on performance than changing the average node degree of a 16-node system as there are a greater number of links added to the system. For a change in average node degree of one, adding 128 links results in significantly more network connectivity than 16 links. Analytically, this corresponds to the slope having a polynomial relationship to account for the increased connectivity in very large systems. In order to find the different values of GRD_{SL} for each number of nodes, we found the slope of each line in Figure 11. We found that the slope had a square polynomial relation to the number of nodes and increased in magnitude when the number of nodes increased.

$$GRD_{SL} = -0.0000025N^2 - 0.00026N - 0.0336 \qquad [5]$$

The slope (GRD_{SL}) takes the form of a square polynomial to account for the increased complexity of routing very large systems. For small systems (<64 nodes), the N^2 term becomes negligible and a linear relationship approximately models the effect of the CAD tools distribution of the topology over the FPGA fabric. However, as systems and their corresponding network become very large, the tools are challenged to find available global resources, which cause a significant decrease in performance resulting in the N^2 term.

The intercept (GRD_{INT}) defines the fixed performance loss when N≠8 and ND = 2 and changes as the number of nodes is changed. Increasing the number of nodes should result in a linear decrease in performance as adding one node to a toplogy results in a fixed gain in routing demand. In other words, going from 10 to 11 nodes or 100 to 101 nodes would result in the same increase in routing demand and consequently the same decrease in maximum performance. Therefore, GRD_{INT} can be expressed as a linear relationship. In order to find the coefficients describing this linear line, we isolated the intercept for each line in Figure 11 and used curve fitting to find that the values of GRD_{INT}. The intercept

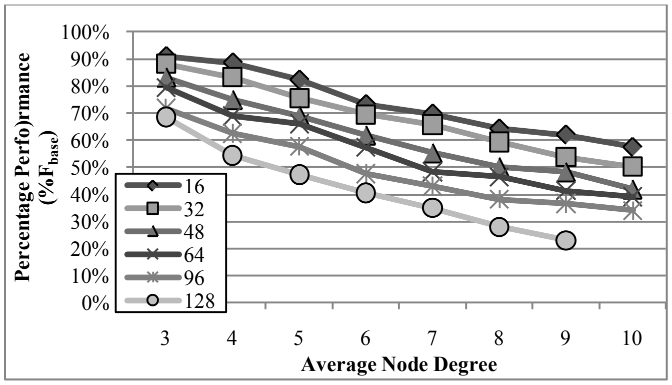

Figure 11: Performance loss due to node degree

decreased linearly as the number of nodes in the system increased according to the following relation:

$$GRD_{INT} = -0.0015N + 1.012 \qquad [6]$$

The intercept for Equation [6] results in GRD_{INT} = 1, when N = 8 to ensure that k_{GC} = 1 for our baseline 8-node ring topology. By substituting the relations given in Equations [2]-[6] back into our original framework given in Equation [1], we obtain our final model for predicting the frequency of any regular or irregular topology. The accuracy of our model will be verified in the following section.

7. VERIFICATION

To measure the accuracy of our model, we created 750 *new* benchmark circuits heterogeneous and homogeneous systems with 8 to 128 nodes with a random topology with node degrees from 2-10 and link widths of 16, 24, 32, 40 or 48 bits. The circuits are mapped to Virtex 4, Virtex 5, and Virtex 6 FPGAs, resulting in 2250 data points. Since the Virtex 6 FPGA's recent release is not supported in Xilinx EDK 10.1.02, we used Xilinx EDK 11.2 to run all our verification experiments. In order to ensure that the trends we saw in EDK 10.1 still existed in 11.2, we ran several preliminary experiments on the Virtex 5 LX330 in EDK 11.2. The results showed that the actual performance increased on average by 8.4% with a standard deviation of 3.2%. However, the normalized performance to the new F_{base} values obtained using the new CAD flow only varied by ~1.8% with a standard deviation of 1.4%.

Table 4 shows a sample set of the operating frequencies predicted by our model, the actual frequency obtained from the CAD tools, and the geometric mean error for selected NoC topologies characterized by the number of nodes, node degree and link width. We chose the geometric mean error as it weights the error's magnitude depending on the maximum operating frequencies. The new F_{base} for each device generated with EDK 11.2 is also listed in Table 4. The overall error was found to be 4.68%.

As seen in Table 4, our equation resulted in large errors for some topologies (see highlighted entries). To determine the relationship between error and NoC characteristics, we analyzed the relationship between error and the values of N, LW, and AND. In Figure 12, we plotted the geometric mean error as a function of average node degree, maximum node degree, and (max node degree – average node degree). We chose these parameters to see how our model encapsulates application specific topologies. If a

Table 4: Predicted Operating Frequencies

# of Nodes	*Node Degree*	*Width*	*Predicted (MHz)*	*Actual (MHz)*	*Error*
Virtex 4		Base Frequency = 150MHz			
16	5	48	105.1	110.1	4.58%
16	8	24	116.8	113.6	2.73%
32	3	16	154.5	150.4	2.68%
32	4	32	128.6	122.9	4.53%
32	6	32	112.5	114.7	1.94%
32	8	24	107.5	111.4	3.61%
64	3	16	144.2	144.0	0.10%
64	5	32	107.5	103.5	3.86%
64	6	24	106.8	97.5	9.48%
64	7	40	78.5	80.4	2.37%
96	4	24	113.4	108.7	4.28%
96	6	32	80.8	74.1	8.88%
96	8	48	43.3	47.1	8.01%
96	9	16	54.4	52.8	18.8%
128	3	32	107.8	92.9	15.9%
128	4	16	106.5	102.9	3.44%
128	5	24	83.9	84.9	1.21%
128	6	40	56.3	53.1	5.91%
Virtex 5		Base Frequency = 200MHz			
16	7	32	149.4	147.9	1.08%
16	9	24	146.2	150.7	2.97%
32	2	24	203.6	201.1	1.23%
32	5	16	187.9	178.3	5.44%
32	7	32	139.3	136.4	2.10%
32	9	40	103.4	99.4	4.03%
64	3	48	203.5	201.1	1.23%
64	5	16	167.7	166.5	0.73%
64	7	32	116.8	115.4	1.20%
64	8	40	91.2	93.1	1.46%
96	2	24	183.3	190.1	3.58%
96	3	32	157.1	148.9	5.52%
96	4	32	140.6	136.3	3.20%
96	5	40	113.6	110.4	2.93%
128	4	48	104.9	110.1	4.77%
128	5	24	111.9	122.4	8.60%
128	6	32	82.8	73.2	13.1%
128	8	16	51.9	45.7	13.5%
Virtex 6		Base Frequency = 240MHz			
16	3	32	225.6	232.2	2.85%
16	7	24	197.9	196.5	0.76%
32	3	16	247.2	256.9	3.78%
32	4	24	221.2	223.3	0.98%
32	6	40	163.1	162.2	0.58%
32	8	32	154.4	170.1	0.49%
64	2	32	219.8	218.1	0.82%
64	3	48	177.2	180.4	1.80%
64	5	16	201.2	191.2	5.25%
64	7	24	154.8	160.9	3.80%
96	3	16	213.3	215.3	0.95%
96	4	40	156.1	153.1	1.94%
96	6	32	129.2	133.6	3.27%
96	9	24	78.5	72.5	8.19%
128	2	24	207.8	211.1	1.58%
128	4	32	148.1	125.8	17.8%
128	5	40	113.3	115.1	1.61%
128	6	16	118.2	92.5	27.9%
Geometric Mean					**4.68%**

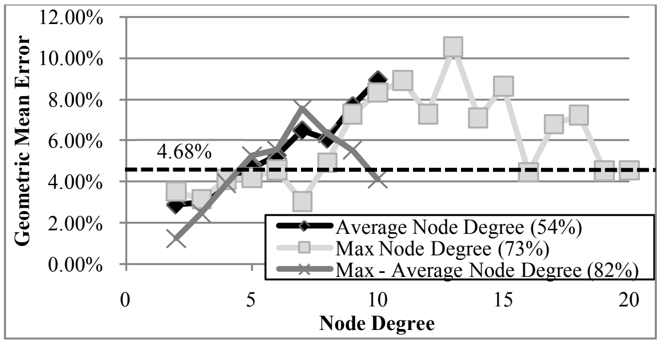

Figure 12: Geometric mean error as a function of node degree

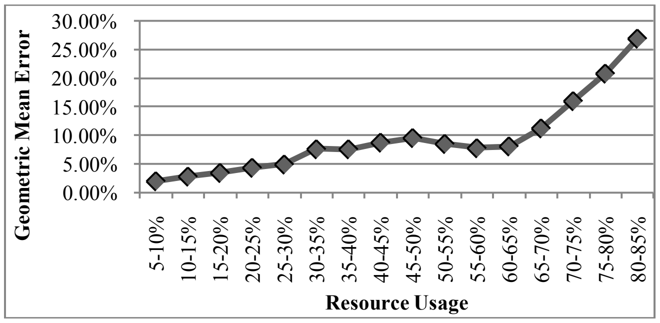

Figure 13: Geometric mean error as a function of resource usage

single node has much higher node degree, max node degree >> average node degree. Thus, local routing demand should have a large effect on performance as seen for the star topology. We also analyzed the variation in error as *N* and *LW*, error remained small (1.3%), and never exceeded 7%, thus we do not show them here due to space limitations. The dotted line in Figure 12 shows the geometric mean error of all our benchmarks. While numerous data points lie above this line, a majority of our benchmarks resulted in points below this line. The percentage of systems below this line is given in the legend.

From Figure 12, we can see that the error increases minimally and remains less than 11% for all three cases, indicating that error is not directly correlated with the node degree as there are no significant trends. Thus our analytical model is still capable of accurately predicting random topologies where max node degree may be greater than average node degree. However, not captured in this analysis is the star topology, which represents an extreme case when the maximum node degree is much larger than the average node degree. For the star topology, our predictor exceeds 25% error when the number of nodes is larger than 16. Addressing this is a topic of future work.

For all benchmarks in which the error was above 10% had a resource usage above 70%. Our current model does not include the resource usage as an input parameter. We plotted our set of benchmark circuits by resource usage, and calculated the average error for each bin. The results are shown in Figure 13. For benchmarks with less than 65% resource usage, the geometric mean error is less than 10%; however above this point, the geometric mean error increases significantly.

These results suggest that to improve the accuracy of our equation, resource usage needs to be considered. Our objective was to provide designers with a means of early design space exploration and if resource usage was included as a parameter in the performance equation, it would be necessary for the designer to fully map the NoC. Therefore, provided the NoC has sufficient routing resources on a respective FPGA, our equation provides an accurate method of predicting the maximum frequency with minimal design time.

8. SUMMARY OF ANALYTICAL MODEL

Below is a summary of our analytical model. Our analytical framework is:

$$F_{pred} = k_{GRD} k_{LRD} F_{base} \qquad [7]$$

where global routing demand (k_{GRD}) and local routing demand (k_{LRD}) are:

$$k_{LRD} = LRD_{SL} \Delta LW + 1, \qquad [8]$$

$$k_{GRD} = GRD_{SL}(AND - 2) + GRD_{INT} \qquad [9]$$

Global routing demand and local routing demand are both defined as linear equations with variable slope and intercept, which are affected by the number of nodes and average node degree. These expressions are tuned for Xilinx FPGAs and shown below:

$$LRD_{SL} = (-0.0012AND - 0.0046) \qquad [10]$$

$$GRD_{SL} = -0.0000025N^2 - 0.00026N - 0.0336 \qquad [11]$$

$$GRD_{INT} = -0.0015N + 1.012 \qquad [12]$$

As previously stated, the expressions of our analytical model shown in Equations [7]-[12] are analytically derived. In order to tune our model to Xilinx FPGAs, we used empirical curve-fitting to determine the coefficients given in Equations [10]-[12]. Therefore, using both analytical and empirical analysis, we were able to develop our analytical model that is capable of predicting performance of a NoC on a Xilinx FPGA with 4.68% error.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an analytical model in the form of a simple equation that describes the maximum operating frequency (performance) of a NoC as a function of various network parameters related to the overall network topology. The equation was shown to be accurate to within 4.68%, even for random topologies. This model provides a measure of the effect of varying different topology parameters on NoC performance on FPGAs. Furthermore, it provides guidance to a designer during early design space exploration when a suitable network topology is being selected.

A key observation from this work is that modern FPGAs contain enough routing to implement fairly complex NoCs. This opens the door to new system architectures based on application-specific NoC's rather than the more restricted mesh topologies that are typically used in ASIC SoC implementations. These application-specific NoC's can be tailored to the problem at hand, leading to an overall improvement in system-level performance measures.

We are currently calibrating our model for Altera devices. Preliminary work suggests that the model has a predictive error of 7.98%, utilizing different coefficients tuned to Altera FPGAs. We are also investigating the possibility of manually placing nodes using relatively placed modules (RPMs).

10. ACKNOWLEDGMENTS

We would like to thank NSERC for funding this project and Xilinx and CMC for their equipment and software donations. We would also like to thank the reviewers and Dr. Steve Wilton for their recommendations.

11. REFERENCES

[1] IBM Corporation, NY, "The Coreconnect Bus Architecture," (1999). [Online]. Available: http://chips.ibm.com Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] OpenCores.org, "The WISHBONE system architecture," (2002). [Online]. Available: http://opencores.org/projects.cgi/web/wishbone

[3] G. de Micheli and L. Benini, "Networks on Chip: A new paradigm for systems on chip design," in *Proc. Conf. Des., Autom. Test Eur. (DATE)*, 2002, pp. 418-418

[4] N. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2002, pp. 105–112.

[5] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Conf. Des. Autom. (DAC)*, 2001, pp. 684–689.

[6] M. Saldana, L. Shannon, and P. Chow, "The Routability of Multiprocessor Network Topologies in FPGAs," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines,* pages 63-72, April 2005.

[7] G. Brebner and D. Levi, "Networking on chip with platform FPGAs," in *Proc. IEEE Int. Conf. Field-Program. Technol. (FPT)*, 2003, pp. 13–20.

[8] N. Kapre, "Packet-switched on-chip FPGA overlay networks," 2006, California Institute of Technology.

[9] N, Mehta, "Time-multiplexed FPGA overlay networks on chip," 2006, California Institute of Technology

[10] R. Francis, S. Moore, and R. Mullins, "A Network of Time-Division Multiplexed Wiring for FPGAs," in *Proc. Second ACM/IEEE Int. Symp. On Networks on Chip. (NOC)*, 2008, pp. 35-44.

[11] K. Goossens, M. Bennebroek, J.Y. Hur, and M.A. Wahlah, "Hardwired Networks on Chips in FPGAs to Unify Functional and Configuration Interconnects," in *Proc. Second ACM/IEEE Int. Symp. On Networks on Chip. (NOC)*, 2008, pp. 45-54

[12] A. Kumar, A. Hansson, J. Huisken, H. Corporaal, "An FPGA Design Flow for Reconfigurable Network-Based Multi-Processor Systems on Chip," in *Proc. Design, Automation & Test in Europe Conference & Exhibition. (DATE)*. 2007, pp. 1-6.

[13] S. Lukovic, and L. Fiorin, "An Automated Design Flow for NoC based MPSoCs on FPGA," in *Proceedings of the 19th IEEE/IFIP Int. Symp. On Rapid System Prototyping*. 2008, pp. 58-64

[14] J. Lee and L. Shannon, "The Effect of Node Size, Heterogeneity, and Network Size on FPGA based NoCs," in *Proc IEEE Symposium on Field-Programmable Technologies*, December 2009