Abstract—A fundamental difference between ASICs and FPGAs is that the wires in ASICs are designed to match the requirements of a particular design. Conversely, in an FPGA, area is fixed and routing resources exist whether or not they are used. Modern FPGAs have the logic and routing resources to implement networks of multiprocessor systems, and system-level interconnection becomes a key element of the design process.

In this paper we investigate how the resource usage affects the mapping of various network topologies to a modern FPGA routing structure. By exploring the routability of different multiprocessor network topologies between 8 and 64 nodes on a single FPGA, we illustrate that the resource utilization increases linearly for different topologies. Furthermore, the difference in logic resources and routing overhead for these topologies is not significant up to 60 nodes. We also show that ring topologies do not incur in congestion and they are limited in size by the available area of the die, whereas the size of fully-connected networks is limited by the available routing resources. Finally, we identify factors that impact the routability of multiprocessor systems on FPGAs and predict whether a particular multiprocessor system is routable and to approximate the operating frequency.

I. INTRODUCTION

With the growing complexity of System-on-Chip (SoC) circuits, more sophisticated communication schemes are required to connect the increasing number and variety of intellectual property (IP) blocks. Approaches like AMBA [1], CoreConnect [2], WISHBONE [3] and SiliconBackplane [4] follow a shared bus scheme that works well for Master-Slave communication patterns, where there are peripherals (slaves) that wait for data to be received or requested from a more complex processing IP (master). When there are several masters (e.g., processors) in the system, synchronization, data interchange and I/O may saturate the bus, and contention will slow down data transfers.

A Network-on-Chip (NoC) [5, 6] provides a possible solution for this problem by creating a scalable interconnection scheme. The concept uses a set of buses connected to routers or switches that interchange packets, much in the same way as traditional computer networks or multiprocessor machines do. Consequently, NoC approaches have design parameters and properties similar to traditional networks. One of these parameters is the topology, which defines the interconnection pattern between the routers and switches.

Multiple topologies have been studied for NoCs on ASICs [7, 8]. Topologies like the hypercube, ring, star, torus, and trees have simple packet routing algorithms. However, a popular choice is the mesh [6, 9] because it provides structure, better control over electrical characteristics, and has an easy packet routing algorithm. These advantages are clear for ASICs, but not necessarily for FPGAs [10]. The electrical characteristics of the FPGA are solved by the chip vendor, not by the user. Therefore, the FPGA’s programmable fabric can be used to implement an application-specific topology based on the particular communication patterns and needs of the application without worrying about the physical implementation. Instead, the concern becomes whether or not a particular topology is too complex to be routable given the fixed set of logic and routing resources available on the chip. This question should be answered during the early stages of the design process of a multiprocessor system to prevent a costly redesign of the system.

It is perhaps intuitive to use a mesh topology for multiprocessor systems implemented on FPGAs since the reconfigurable fabric layout is in the form of a mesh. However, the placement and routing of components on an FPGA will not typically result in a symmetric, well-organized structured layout that resembles a mesh. Furthermore, manually restricting the placement of components or routing of nets may lead to inefficient resource utilization for the logic that is not part of the network.

In this paper, we investigate the routability of point-to-point network topologies on FPGAs, measuring the impact of soft multiprocessor system topologies and their characteristics when implemented on modern commercial FPGAs. We do this by measuring the logic utilization, logic distribution (area), maximum clock frequency, number of nets, and the place and route time for five well-known network topologies and a number of random generated topologies. From this, we identify factors that impact the routability of multiprocessor systems on FPGAs. Based on these factors, we demonstrate that it is possible to predict whether a particular multiprocessor network topology is routable and to approximate the probable operating frequency.

The rest of this paper is organized as follows. Section II provides some background about research on NoCs. Section III describes the topologies implemented and gives a brief description of the block used as the network nodes, which we call the computing node. Section IV describes the implementation platform, and how the systems are generated. Section V presents the results obtained for some well-known topologies. Section VI explores the congestion around a particular node of the star topology. Section VII shows the estimation of the placement, routability and expected frequency for any topology. Finally Section VIII provides some conclusions.
II. RELATED WORK

In this section we present examples of typical research on NoCs, and how it relates to this study. In Brebner and Levi [10] discuss NoC implementations on FPGAs, but their focus is on the issues of using packet switching on a mesh topology in the FPGA and on implementing crossbar switches in the routing structure of the FPGA. Most NoC work assumes ASIC implementations and there are numerous studies including work on mesh topologies [6] [9] and fat trees [7]. Other studies on NoCs are done using register-transfer-level simulations [7] and simulation models [11], but they do not show the implementation side of the NoC. Instead, we focus on real implementations of network topologies and how well they can be mapped to a real FPGA routing fabric using commercial tools.

In Bertozzi et al [12], a scheme to synthesize application-specific network topologies is presented. The authors mean to provide MPSoC designers with a tool to map a particular application to a particular topology and generate an ad-hoc NoC, which can be simulated. In this work, we focus on the next level in the design flow after mapping an application to a particular topology: we map particular topologies to a particular chip and investigate the current limits of commercial technology.

Based on the philosophy of routing packets, not wires [9], [13], NoC architectures have been proposed as packet-switching networks, with the network interface itself being the focus of much of the research. In this paper, we use a simple network interface, similar to a network hub, which does not provide packet forwarding or packet routing. Packets can only be sent to, and received from nearest-neighbor nodes using point-to-point links. This makes the network interface extremely simple, but it is sufficient for our purposes as the focus of this work is on the routability of various topologies, not on the switching element architecture or the network interface.

III. EXPERIMENTAL ENVIRONMENT

The actual processor and network interface used are not the critical elements in this study. What is required is to create circuits that form particular communication patterns between the computing nodes to see how the implementation resources of these circuits on the FPGAs varies as the patterns (i.e. topologies) are changed. In this section we describe the Network-on-Chip topologies and computing node architecture used to perform the experiments in this paper.

A. Network Topologies

Networks can be classified into two categories. Static networks consist of point-to-point, fixed connections between processors, and dynamic networks, which have active elements, such as switches, that can change the connectivity pattern in the system according to a protocol. In an FPGA, the network can also be dynamically reconfigured to adapt to communication patterns by utilizing the reconfigurability [14] of the FPGA.

Fig. 1 shows examples of systems with different numbers of nodes and topologies that are implemented to carry out our experiments. Every topology can be seen as a graph that is made of edges (links) and vertices (computing nodes). In our implementations, the links are 70 bits wide (i.e. Channel_width of 70 bits) with 32 bits used for transmission and 32 bits used for reception to provide a full-duplex communication system. The links also include six control lines used by the network interface.

In this paper, we focus on static message passing networks. The ring, star, mesh, hypercube, 2D-torus, and fully-connected topologies are selected as a representative sample of static networks, ranging from the simplest ring topology to the routing-intensive fully-connected system.

Network topologies can be characterized by a number of properties: node degree, diameter, link complexity, bisection width and regularity [15]. Node degree is the number of links from a node to its nearest neighbors. Diameter is the maximum distance between two nodes. Link complexity is the number of links the topology requires. A network is deemed to be regular when all the nodes have the same degree. Bisection width is the number of links that must be cut when the network is divided into two equal set of nodes. Table I shows a summary of these characteristics for each of the well-known topologies used in this paper.

The characteristics of the network topology define the network interface of each node in the system. For example, the four-dimensional hypercube is a regular topology consisting of 16 nodes, with all nodes having a degree of four. This means that this topology requires a single network interface type, each with four ports, i.e. four communication links. The network interface is used to communicate with other nodes in the network. The maximum distance from one in the system to another (diameter) is four, which means that data going through the network may require redirection or routing at intermediate nodes and travel on up to four links to reach its destination. The link complexity is 32, which is the total number of point-to-point links that the overall system will have. In contrast, a 16-node mesh has a total of 24 links in the system, but it is not a regular topology, requiring three different versions of the network interface. Inner nodes require an interface with four ports, perimeter nodes require one with three ports and corner nodes use a two-port interface.

Table I shows the characteristics of some well-known topologies.
B. Computing Node

The computing nodes in Figure 1 consist of a computing element and a network interface module. Figure 2 shows the structure of a computing node. Commercial FPGA CAD tools can only implement designs that produce at least one output signal that must be transmitted off-chip. Otherwise, the entire system is optimized away as unused logic. Thus for each of our systems, there is a master computing node configured to communicate with the external world using a UART. The UART is attached to the peripheral bus as shown inside the dashed box of Figure 2. The rest of the nodes have no peripheral bus.

IV. IMPLEMENTATION PLATFORM

This section describes the experimental framework employed to conduct the experiments. First, the tools used and the implementation platform are explained, followed by a description of how the multiprocessor systems are generated. We use the Xilinx [16] EDK tools version 7.1i in combination with the Xilinx XST synthesis tool to build the net lists, map, place, and route the multiprocessor designs. The network interface is developed in VHDL and simulated using ModelSim version 6.0b [17]. The routed nets are counted with the help of Xilinx XDL tool, which provides a direct access to Xilinx proprietary native circuit description files (NCD) to generate reports of the physical resource utilization for any design. All the systems are generated on IBM workstations with Pentium Xeon processors running at 3.0 GHz and Hyperthreading enabled and 3 GB of memory.

The Xilinx MicroBlaze soft-processor core [16] is used as the computing element. The hard multiplier option for the MicroBlaze is disabled to minimize the impact of hard core blocks that may influence or limit the placement and routing. The internal Blocks of RAM memory (BRAMs) in Xilinx FPGAs are hard core blocks that also affects the placement and routing, but they are essential for the MicroBlaze core to synthesize so they cannot be removed. The MicroBlaze is connected to the network interface module using two Fast Simplex Links (FSLs), which is a Xilinx FIFO core that provides a unidirectional communication channel.

We use the Xilinx XC4VLX160 FPGA because it is the largest LX version Virtex4 chip sold at the highest speed grade. The LX version of the Virtex4 family has the highest logic density and fewer DSP and BRAM cores than the SX version. It also does not have the PowerPC processors.
A. System Generation

Manually describing large multiprocessor systems is time consuming and error prone. For example, a 32-node, fully-connected system requires 1056 links to be specified. Instead, an automated process for generating the systems used in our experiments is developed and the flow is shown in Figure 3. The flow starts by using the topology generator, which is a Perl script that takes a high-level description of the system, such as the topology name, and the number of nodes to generate a well-known topology, such as those mentioned in Section III-A. Additionally, the script can generate random topologies based on the number of nodes and the average node degree (i.e., average number of links per node).

The topologies are described as graphs in a Graph Description File that is input to the System Generator [18]. The System Generator translates the graph into IP cores and buses, and generates the files used by the EDK. These files are the Microprocessor Hardware Specification file (.mhs), the Microprocessor Software Specification file (.mss), the Xilinx Project file (.xmp), the User Constraint file (.ucf), and a template C source file for each processor. In turn, EDK uses these files to synthesize, place and route the designs.

The logic resource usage is measured in terms of the total number of LUTs required for the entire design and the number of LUTs related to only the interconnection network, i.e. those used to implement the network interface modules and the topology links. The logic resource utilization for the entire design is extracted from the main EDK log. The logic resources needed to implement only the interconnection network is estimated by first synthesizing the network interface modules as stand-alone blocks to determine the number of LUTs required. These numbers are then used to estimate the usage of the entire network. For example, the 8-node star topology requires one 7-port network interface, which uses 345 LUTs, and seven 2-port network interfaces, each requiring 111 LUTs. The total number of LUTs required by the network is \(345 + (7 \times 111) = 1122\) LUTs. Note that this is only an estimate as the values reported from synthesizing the stand-alone block may change from the actual resources used in the system’s implementation due to optimizations that may occur.

The routing resource utilization is measured in terms of the number of nets. Similarly to the logic utilization, we distinguish between the total number of nets in the system and the nets related only to the network. The total number of nets in the system is extracted from the main EDK log file. The number of nets attributed only to the network is found by counting the number of nets related to all the network interface modules in the design. This includes all nets that are used in the network interface modules as well as the nets in the network topology itself. Including the nets in the network interface modules is reasonable because more complex topologies use more complex network interface modules that also consume FPGA routing resources. The counting of nets is done by using the Xilinx XDL tool, which allows the user to filter out net names from the report generated by the tool.

A. Results

Figure 4 shows a plot of the number of LUTs needed to implement the complete systems, including the MicroBlaze, FSLs, memory interface controllers, switches, UART, and OPB bus. As expected, the system with the fully-connected network has the highest logic utilization, and as the system size increases, the difference with respect to the other topologies becomes more pronounced because of the \(O(n^2)\) growth in size. The difference in logic utilization between topologies is attributed to the larger network interface modules required for more complex topologies.

The routing resource utilization for these systems exhibits a similar trend to the logic resource utilization shown in Figure 4. The fully-connected system requires the most nets, as expected, while the difference amongst the remaining topologies is not significant. Moreover, the 24-node fully-connected topology can be mapped and placed but not routed. In other words, there are enough logic and memory resources, but the design fails to route because there are insufficient routing resources for this particular topology implemented on the XC4VLX160 FPGA.

An interesting observation from the experimental data is that the logic and routing resource utilization exhibits a linear trend...
for all the well-known topologies used in this experiment, except for the fully-connected, which has a square trend. Table II illustrates how consistent the resource usage of a topology is with the topology’s link complexity. Column 1 lists all of the different topologies we investigated and Column 2 details which type of curve fit is applied to the resource utilization data. Column 3 shows the determination coefficient ($R^2$) for the curve fit of the routing resource utilization data.

The determination coefficient is a measure of the fit of a set of data points to an interpolated equation. For example, both the ring and 2D-torus have a determination coefficient of 1.0000 for the measured data. This means that the measured routing utilization data is a perfect fit to the interpolated linear equation. As the value of $R^2$ decreases, so does the correlation of the data to the applied curve fit. Theoretically, the routing resource utilization will be proportional to the link complexity of the system. Column 4 of Table II provides the theoretical $R^2$ value for each of these topologies. By comparing the theoretical $R^2$ values and measured $R^2$ values, we can see that they are consistent.

Figure 5 shows the routing overhead of the network interface modules and topology as a percentage of the entire design. The routing overhead is calculated as the number of nets that are related to the topology links and all the nets from network interface modules divided by the total number of nets in the entire design. As expected, the ring topology has the lowest routing overhead with about 6% of the total number of nets.

For those topologies that have a linear increase in routing resources, the routing overhead is practically independent of the system size because most of the total routing comes form the processors and other non-related logic to the network. The fully-connected topology overhead increases rapidly as the number of nodes increases. With 22 nodes, the fully-connected interconnection network accounts for 42% of the total routing in the system, compared to only 13% routing overhead for the 32-node hypercube, which is the next most complex topology.

The logic overhead is computed as the number of LUTs used for the network interfaces as a fraction of the total LUTs required for the entire system. The logic overhead has a similar pattern as the routing overhead. For 8 nodes, the logic overhead of the fully-connected topology is roughly 22%, and with 22 nodes the overhead is 40%. As expected, the lowest logic overhead of all the topologies is the ring, at only 8% independent of the system size.

### Table II

<table>
<thead>
<tr>
<th>Topology</th>
<th>Curve Fit</th>
<th>Theoretical Routing $R^2$</th>
<th>Measured Routing $R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring</td>
<td>Linear</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Star</td>
<td>Linear</td>
<td>1.0000</td>
<td>0.9996</td>
</tr>
<tr>
<td>Mesh</td>
<td>Linear</td>
<td>0.9997</td>
<td>0.9998</td>
</tr>
<tr>
<td>Hypercube</td>
<td>Linear</td>
<td>0.9981</td>
<td>0.9987</td>
</tr>
<tr>
<td>Fully-Connected</td>
<td>Square</td>
<td>1.0000</td>
<td>0.9993</td>
</tr>
</tbody>
</table>

The place and route time data varies considerably because of the random nature of the place and route algorithm. However the ring, star, mesh and hypercube topologies have average times to place and route that are approximately the same for a given number of nodes. For example, on our IBM workstations, the 8, 16 and 32-node systems the average times are 15 min., 35 min., and 1 hours 58 min., respectively. In general, the ring and mesh are requiring slightly less than this time to place and route, the hypercube requires very close to the average time, and the star requires increasingly more time than the average due to the congestion around the central node.

The torus topology requires considerably more time to place and route designs with 8, 16, and 32 nodes – specifically 37 min., 72 min., and 3 hours 15 min. Finally, the fully-connected topology exhibits an exponentially growing time of 16 min. for the 8-node system, 4 hours 5 min. for the 16-node system, and remained unroutable after three days for the 32-node system.

### VI. CONGESTION EXPERIMENT

The goal of this experiment is to investigate the impact of congestion on the routability of a multiprocessor design. The intention is to stress local routing resources in a particular region of the FPGA fabric. We do this by gradually increasing the number of nodes of the star topology, which is a special case because the central node will experience severe congestion without increasing the demands on the global resources.
by the other nodes in the system. A fully-connected topology would also stress the local congestion around the system’s nodes, but the total resources required to implement a fully-connected topology would also quickly increase as discussed in the previous section. This would limit the system to only 22 nodes as stated in Section V. By using the star topology we are able to extend the system size up to 63 nodes.

In our multiprocessor systems, the slowest block is the MicroBlaze, which is synthesized to a maximum frequency of 150 MHz. Therefore, the timing constraint for the entire multiprocessor system is also set to 150MHz. The placement and routing effort levels are set to high and with emphasis on timing by enabling the timing driven option for the mapping stage. However, no time is spent to try and push the tools to improve the results that did not meet the target frequency. We attempt to generate a variety of systems ranging from 8 to 64 nodes in size at four node increments. The smallest star topology is fixed at eight nodes for this experiment because star systems with less than eight nodes are not interesting as they will all place and route and meet timing constraints.

A. Results

Figure 6 shows the results of the experiment. We can see that the congestion only caused degradation in the design’s performance by reducing the maximum frequency from 150MHz with 28 nodes or less, down to 109 MHz with 63 nodes. The 64-node star system did not place because of a lack of space in the FPGA fabric. However, with 63 nodes, the star topology uses only 62% of the available slices in the FPGA. We estimate that there are enough logic and routing resources available to place another 30 MicroBlazes in the remaining 38% of the fabric.

Figure 6 shows the results of the experiment. We can see that the congestion only caused degradation in the design’s performance by reducing the maximum frequency from 150MHz with 28 nodes or less, down to 109 MHz with 63 nodes. The 64-node star system did not place because of a lack of space in the FPGA fabric. However, with 63 nodes, the star topology uses only 62% of the available slices in the FPGA. We estimate that there are enough logic and routing resources available to place another 30 MicroBlazes in the remaining 38% of the fabric.

Figure 7 illustrates the placement of the 62 node star topology using Xilinx’s FPGA Floorplanner. A visual inspection reveals that the design has a porous placement. Several empty spaces are scattered all over the FPGA fabric, particularly along the right side, illustrating that the logic could be more tightly packed. This is caused by the Relative Placement Macro (RPM) constraint on the MicroBlaze core, which forces the core’s logic into a fixed relative placement on the FPGA to achieve higher operating frequencies. Removing the constraint would allow better packing of the design’s logic, but, depending on the placement, could seriously degrade the core’s operating frequency. The RPM produces a brick effect that results in an inefficient placement as the shape and size of the brick does not allow the processor cores to be tightly packed on the fabric.

Figure 7 also indicates the placement of the Central MicroBlaze, which is the processor in the node at the centre of the star, and its network interface, the Central Network Interface. While it is interesting to note that the Central MicroBlaze is actually located in the top left corner of the die, the placement of its network interface is of even greater interest. The Central Network Interface is located near the centre of the die, but it is also distributed over an area larger than four MicroBlazes. The placer is able to do this because the network interface module is essentially a large multiplexer and the FPGA’s architecture has multiplexers distributed throughout the fabric. This distributed placement allows the placer to reduce the congestion around the central node’s network interface, which provides it with increased routing resources.

The consequence is a longer path delay between the internal nets in the network interface with a corresponding decrease in the maximum frequency for the system. An examination of the longest critical path, which is a critical net within the central network interface, shows that 58% of the delay is attributed to routing delay and 42% is attributed to logic delay. However, this routing delay for the star system is not as critical as it is for the 22-node fully-connected where the lack of available routing resources causes 80% of the longest critical path delay to be attributed to the routing and only 20% to the logic.

VII. ESTIMATION OF ROUTABILITY AND PERFORMANCE

The objective of this experiment is to identify the factors and evaluate their impact on the routability of our multiprocessor systems. These factors are then used to predict whether a particular topology is placeable and routable, and to estimate the expected performance. For the purposes of this investigation, we have limited the topologies to those that only allow those where the degree of each node in the system is less than that of the total number of nodes in the system. This means that given any two nodes in a topology, there is a maximum of one link between them. Therefore, we exclude fat-tree topologies.
from our consideration, where the number of redundant links increases for those nodes closer to the root node. We do not include these topologies in our study because the current version of the network interface and the design flow tools do not support that type of connectivity pattern.

The results from our investigation show interesting trends, despite the implicit randomness of the implementation tools. However, these trends are applicable to FPGAs with similar routing structures and logic to routing resource ratios. Further study of how these trends change with different FPGA fabrics, for example with different FPGA architectures or families of FPGAs, or with embedded hard blocks into the FPGA fabric, such as the PowerPC processor in Xilinx Virtex2Pro FPGAs is an area for future work.

The experimental data for this analysis was obtained from the placement and routing of 112 different multiprocessor systems with varying number of nodes and different topologies. From this data, we have generated plots that illustrate an empirical predictor of the routability and performance of our multiprocessor systems. These graphs represent an approximation of what the outcome could be if similar systems are being generated.

In this experiment, we wanted to establish boundaries on the complexity of the topologies that can be successfully implemented on an FPGA. For a given number of nodes, the fully-connected topology has the largest number of links and is used as an upper bound to the implementation complexity because no other topology will require more resources. The ring topology has the lowest link complexity and is used as a lower bound to the implementation complexity because it is the least resource demanding topology. Theoretically, it should have equal or better timing performance than any other topology. The remainder of the well-known topologies – the star, mesh, 2D-torus, and hypercube – have intermediate implementation complexities relative to the ring and fully-connected topologies. Thus they are located in the region bounded by these curves.

To better stress the upper and lower bounds of this region, a number of random topologies are generated using the topology generator as explained in Section IV-A. The input parameters for the topology generator are the number of nodes and the average node degree, which is used as an indicator of the link complexity for the random topologies. A total of 80 random topologies are generated, with average node degrees ranging from 3 to 12, and the number of nodes ranging from 8 to 60. Not all the possible combinations of these input parameters for the topology generator are used due to the considerable time required to synthesize, and place and route our multiprocessor systems, particularly the larger ones. Fortunately, it is not necessary to generate some of these topologies as they are too complex and will not place and route successfully. Our initial experiments determined the upper bounds for both input parameters that could successfully generate systems, thus reducing the number of experiments we needed to run. Similarly, some topologies are so simple that they will place, route, and meet timing requirements without testing the capabilities of the tools or fabric, allowing us to neglect them for the purpose of this study.

A. Results

Figure 8 graphically illustrates our empirically derived routability predictor using a plot of the bounded region in which topologies should successfully be able to place and route. The x-axis denotes the total number of nodes in the system and the y-axis provides denotes average node degree for the system. The average node degree and the number of nodes for a topology define a point in the plot. If that point falls within the region below the boundary line, it is highly probable that system will place and route. Each point in Figure 8 corresponds to a successfully implemented topology, including both the well-known topologies and the randomly generated ones. The probability of a successful place and route for an n-node system increases as the coordinates that represent the topology are further away from the boundary line, i.e. such that the average node degree decreases. This means that it is possible for a system to fail to place or route if a particular topology maps exactly onto the boundary line. However, by changing the seed or applying some optimizations to the system, a successful implementation of the design is still possible. We ran the place and route process multiple times for those points located on the boundary line in Figure 8, changing the seed in every iteration. We found that, on average, for these topologies on the boundary line, more than half of the attempts to generate the system resulted in a successful implementation.

Figure 8 is divided into five regions based on the routability of the design. From left to right, the first is the invalid region. The first segment of the boundary line on the right side of the invalid region represents all the fully-connected systems that can be successfully routed on the FPGA. For the XC4VLX160 used in this investigation, the largest fully connected topology that can be implemented is 22 nodes. To the left of this line, all of the topologies have average node degrees that are greater than or equal to the number of nodes in the system. Based on our previously described parameters for this study, this region is invalid as the topologies will include redundant links between nodes.

The second region is below the “fully-connected” portion boundary line and represents the fully routable region. In this region, any user defined topology should place and route. The third region is located below the remainder of the disjoint boundary line and is called the routing resource limited region. Initially, the maximum average node degree for which a topology will successfully route drops significantly for this region. This indicates that average node degree is the predominant factor that limits the routability of a system for this portion of the region as the demand on routing resources is more significant than the demand on the logic resources. Eventually the boundary line terminates at 64 nodes, where the ring system (the lower bound of the implementation complexity) becomes unplaceable caused by a lack of area on the fabric to place another node. Above the disjoint portion of the boundary line is located the fourth region, the unrouteable region. As in the routing resource limited region, there are sufficient logic resources to place designs with this many nodes. However, there are insufficient routing resources to route these topologies due to their average node degree and
link complexity. The final region to the right of the disjoint portion of the boundary line, which terminates at 64 nodes is the unplaceable region. In this region, the lack of logic resources makes the routing resource requirements irrelevant.

Figure 8 is a binary predictor, it predicts whether a system will place and route, but not the expected frequency. As a complement to the previous predictor, Figure 9 shows the expected frequency based on the number of nodes. The points in the plot are the same as in Figure 8. The fully-connected topology again has the lowest maximum frequencies due to its high resource utilization. Conversely, the ring topology meets the required timing constraint of 150MHz up to 63 nodes. Since the MicroBlaze is in the critical path of our multiprocessor systems, no other topology will fall into the region above 150 MHz. Initially, some of the random topologies between 8 and 22 nodes were constrained to lower frequencies than the fully-connected topology, even though they are not as complex and require less resources. This situation was caused by the implicit randomness of the tools. By rerunning the place and route process for these same systems with different seeds, their operating frequency increased above that of the fully connected network.

Another observation from Figure 9 is that topologies with less than eight nodes will place and route, and will also meet timing requirements, regardless of the link complexity. The topologies between 8 and 22 nodes will place and route, but the timing performance will depend on the complexity of the topology. The placement, routing and the timing performance for systems with more than 22 nodes and less than 63 depend on the number of nodes, the average node degree and the regularity of the topology. Systems with more than 63 nodes will not place or route on the FPGA used in this experiment if the MicroBlaze’s RPM constraint is used.

B. The Topology Regularity Factor

An interesting observation from the experimental data shown in the previous section is that for topologies that exhibit greater regularity, it is possible to obtain a better maximum operating. To explain this, let us define a couple of concepts. The first concept is how we measure the regularity of a topology. We do this by calculating the standard deviation of the node degree, where a lower standard deviation indicates a more regular topology. For example, from Table I we know that the 2D-torus topology has a node degree of four and it is a regular topology because all the nodes have the same number of links. The standard deviation in this case is zero. In contrast, a randomly generated topology, with the same number of nodes and an average node degree of four, has a standard deviation different than zero because the number of links per node is based on a normal distribution. This means that such random topology likely have nodes with more than four links and nodes with less than four links, but on average all the nodes will have four links.

The second concept is the notion of equivalent complexity for topologies. We say that a random topology is equivalent in complexity to a well-known topology when they have the same number of nodes in their topology and the same average node degree. Random topologies with equivalent complexity to a well-known topology should provide similar stress to the place and route tools their average resource requirement is approximately the same.

The maximum operating frequency of the 2D-torus, the mesh and their equivalent random topologies are shown in Figure 10. We can see that the 2D-torus and the mesh both perform consistently better than their equivalent complexity random topologies. The standard deviation of the mesh varies very little and actually decreases as the number of nodes in the system increases. For these reasons, the mesh performs similar to the 2D-torus. For example, at 48 nodes (Case B in Figure 10), the standard deviations of the node degrees are 0, 0.6 and 1.7 for the 2D-torus, the mesh and the equivalent complexity random topology, respectively. The maximum frequencies (in MHz) are 150.5, 150.1 and 134.3 for the 2D-torus, the mesh and the equivalent complexity random topology, respectively.

A greater standard deviation may cause a wider difference between the maximum and minimum number of links per node in a topology. For example, three randomly generated topologies with the same average node degree (<Degree> = 4) and the same number of nodes (n = 36) are shown in Figure 10 (Case A). The maximum number of links per node for each system is 7, 9, and 11 with standard deviations of 1.9, 1.8 and 2.2, respectively. The maximum frequencies
achieved for these random topologies in the same order are 150.6 MHz, 137.6 MHz and 129.8 MHz. Systems with larger standard deviations tend to have more congested nodes, and as consequence can reduce the maximum operating frequency.

![Fig. 10. Topology impact on global routing](image)

**VIII. CONCLUSIONS**

When implementing an NoC using an ASIC, only the required routing is included. Thus, it is important to make the correct tradeoffs between the communication requirements for the application and the routing requirements of the interconnect topology. FPGAs have a rich routing fabric and the wires exist independent of their usage by the design. This work has shown that the ring, star, mesh, 2D-torus and hypercube topologies can all be routed on a modern FPGA and are able to scale well for systems with a large number of nodes. Even a fully-connected topology can work for small numbers of nodes, which suggests that for small networks, it is less important to worry about routing considerations when picking the network topology for an NoC on an FPGA. We show that the routing overhead of a ring, star, mesh, 2D-torus and hypercube is less than 15% of the total number of routing resources used by a particular multiprocessor design, independent of the number of nodes, and the difference in routing overhead among these topologies is not significant.

We found that there is a linear trend in logic and routing resource utilization for our MPSoCs implemented using the ring, star, mesh, 2D-torus and hypercube, whereas the fully-connected topology’s resource usage follows a square trend. A system with a fully-connected topology can be implemented up to 22-nodes. After this point, there is lack of routing resources and the designs are unrouteable. Conversely, the systems implemented using the simplest topology (the ring) reached 63 nodes, after which there is insufficient area to place more processors. However, the chip utilization is only 60%, which means there are sufficient logic resources to implement more nodes, but the placement of large Relational Placement Macros, such as the MicroBlaze processor, inhibit an optimal placement for denser packing.

We successfully implemented 112 MPSoCs on a single FPGA, with the number of processors ranging from 8 to 63, using different well-known interconnection topologies as well as a number of randomly generated ones. From this data, we empirically generated plots of predictors that can indicate whether a MPSoC with a particular topology can place and route successfully along with an approximation of the maximum frequency, based on the size of the chip, they should remain consistent for FPGAs with a similar architectures. The specific boundary conditions for our experimental platform are now given.

From the experimental data, we can conclude that any MPSoC with less than or equal to eight nodes and no redundant links, should successfully place and route and will likely meet the timing constraints, regardless of the selected interconnection topology. For systems between 8 and 22 nodes, these topologies should still place and route, but the maximum frequency depends on the link complexity, the regularity of the topology, and the number of nodes. For systems with more than 22 nodes and less than 63, a successful placement and routing, as well as meeting timing requirements are not guaranteed. The average number of links per node becomes the predominant factor that limits the routability of our multiprocessor systems. Finally all desired topologies of more than 64 nodes will fail to place on the current FPGA platform due to insufficient logic resources.

**ACKNOWLEDGMENTS**

We acknowledge Xilinx and CMC Microsystems/SOCRN for providing the tools, development hardware and computing equipment. Research funding for Manuel Saldaña is provided by CONACYT in Mexico. Lesley Shannon’s research is funded by the O’Brien Foundation. Additional funding is also provided by the National Science and Engineering Research Council.

**REFERENCES**


