NoCGEN: A Template Based Reuse Methodology for Networks on Chip Architecture

Jeremy Chan and Sri Parameswaran
School of Computer Science and Engineering
The University of New South Wales, Australia
{jeremyc, sridevan}@cse.unsw.edu.au

ABSTRACT

In this paper, we describe NoCGEN, a Network On Chip (NoC) generator, which is used to create a simulatable and synthesizable NoC description. NoCGEN uses a set of modularised router components that can be used to form different routers with a varying number of ports, routing algorithms, data widths and buffer depths. A graph description representing the interconnection between these routers is used to generate a top-level VHDL description.

A wormhole output-queued 2-D mesh router was created to verify the capability of NoCGEN. Various parameterized designs were synthesized to provide estimated gate counts of 129K to 695K for a number of topologies varying from a 2 x 2 mesh to a 4x4 mesh, with constant data bus size width of 32. The NoC was simulated with random traffic using a mixed SystemC / VHDL environment to ensure correctness of operation and to obtain performance and average latency. The results show an accepted load of 53% to 55.6% with an increase in buffer depth from 8 to 32 flits for the 4x4 mesh router.

1. INTRODUCTION

The design of multi-million gate Systems on Chip (SoCs) introduces many difficult problems, such as poor scaling of interconnect [6], increasing design complexity, and tightening power constraints under the pressure of a decreasing time-to-market. Intellectual property (IP) reuse is a method to reduce the productivity gap in SoC design [8]. Platform-based design methods [17] accelerate time-to-market through extensive reuse of an architectural platform. Such design methods decouple computation from communication concerns, simplifying problems, and enabling orthogonal problems to be solved independently.

Recently, the Network on Chip (NoC) design methodology has been proposed as a scalable alternative to the adhoc shared bus structure. The benefits of this methodology include: (a) increased bandwidth via high performance switching networks; (b) regular wiring allowing planning; (c) regularity allowing layering and reuse of routers; and (d) lower energy dissipation through shorter wire segments. NoCs need to be customized for a specific application; i.e. buffers need to be small and a suitable topology needs to be selected. For that reason, it is beneficial to be able to rapidly parameterize and change the topology of a network for any given application.

In this paper, we present, NoCGEN, an extensible template description that allow rapid design-time customizations of NoC circuits. We exploit the regularity and similarities that exist between different router components, to create varied NoC routers that can be customized as well as allowing the topology to be diverse. These customizations are defined in an annotated graph, which is used as an input to NoCGEN. The output of the tool is a synthesizable and simulatable HDL description of the NoC circuit. NoCGEN utilises a set of libraries (consisting of routing algorithm, arbitration component, and flow control component libraries), along with the input from an annotated graph input to produce the HDL description. A traffic generator library is then used to instantiate components allowing the complete simulation of the NoC. SystemC is used to create producers and consumers of traffic. This tool makes system-level changes in the interconnection possible with minimal effort. Previous efforts have not allowed such a flexible approach to the creation of NoC circuits.

The paper is organized as follows: Section 2 briefly discusses the related work in the area of NoC architecture and design space exploration. The generic router architecture used to describe interconnection networks is discussed in Section 3. Section 4 presents the simulation platform. A case study of the mesh router is shown in Section 5. Lastly, we present our conclusions and future work.

1.1 Related work

On-chip interconnection networks have been proposed to replace global bus wires in large SoC designs. Many of the proposed networks [3, 18, 16, 9, 5] are an adaptation of previous work in direct networks for parallel and distributed systems. However, the routers used in NoCs have tighter resource constraints than traditional networks such as area, layout and power. These routers are typically limited to smaller buffering requirements, deterministic routing and hardware efficient flow control mechanisms. These routers may also need to provide services such as bandwidth reservation[16].

Recently, several NoC architectures have been proposed each with varying topology, flow control and routing. Some of the popular topologies include: fat-tree [5], tile-based mesh [18, 16, 9] and folded torus [3]. NoCs are usually packet-switched and support a single flow control mechanism such as wormhole [5] and virtual cut-through switching [3]. Virtual channels are also added to increase throughput. NoC architectures can also be constructed from hybrid of networks and buses [20] to support legacy components.

The two main analysis techniques used for analyzing communication architectures are: (i) simulation; and (ii) static analysis techniques [7, 10]. Simulation of routers has been performed using discrete event hardware description languages (HDL) such as Verilog [15], VHDL [5]and SystemC [14] or high level models created in C/C++. HDL simulators are usually more detailed than C models but are slower and less flexible due to limited support for high-level modelling. Our NoC routers are highly modular like [19], but our topologies are not limited to rings, stars and buses.

1.2 Contributions of this Paper

Previous prototype architectures [3, 18, 16, 9, 12] do not support the mixture of different architectures. The NoCGEN methodology proposed in this paper increases flexibility by allowing different interconnection networks to be created from the same template of the router (Figure 1). The tools provide similar benefit to those provided by the Sonics Backplane [21] but applied to NoC and switched networks.

The specific contributions of this paper are: a) an implementation of differing NoC routers by utilizing a pre-designed library of sub-components; b) a synthesizable HDL generation methodology to rapidly configure the above routers into a NoC circuit, given a high level graph description; and c) a mixed SystemC VHDL simulation environment to simulate the generated NoC circuit with a realistic workload. The created NoC circuit can be synthesized to give accurate estimates of power,

Proceedings of the 17th International Conference on VLSI Design (VLSID'04)
1063-9667/04 $ 20.00 © 2004 IEEE

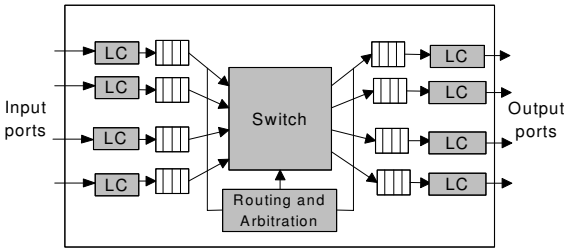

Figure 1: Canonical router architecture

area and latency.

2. BACKGROUND

In this section, we describe the properties of an interconnection network in the context of the network domain. There are two properties that make the interconnection network suitable for reuse: (i) all interconnection networks can be described in general terms by their topology, routing and flow control [4] and (ii) routers can be described using a common model or template.

2.1 Topology, Flow control and Routing

The network topology describes the interconnection between the components in the network. The topology can be described using a graph where the vertices are the nodes, and the edges are the links. The nodes are elements that can consume, produce or forward a message on the network such as routers, hosts and bridges. A host is a network element that can produce and / or consume messages, such as masters and slaves. A router switches messages among several nodes, while a bridge translates messages between two or more nodes with different message formats.

The interconnections between the nodes or edges in the graph are links. A link is a bidirectional communication medium that represents a set of control and data signals used to transfer messages between a pair of nodes. These links can have many properties including the ability to handle pipelining, different link widths and packet formats.

The flow control deals with how and when packets are forwarded. Flow control is strongly related to the buffering management that determines when messages are blocked in the network. Some of the common flow control mechanisms include store and forward, virtual cut-through and wormhole switching. The main difference between these three flow control mechanisms is how the messages are blocked. In store and forward, packets are blocked at the router until the whole packet has been received. Virtual cut-through only forwards the packet when the whole message can be buffered at the next node. In contrast, wormhole routing breaks up a packet into flow control digits called flits. This allows lower buffering requirement to be needed and each of the the flits are forwarded as soon as they can be received and buffered.

The routing algorithm determines the path that a message travels along from its source to its destination. Packets can either be routed at the source or at each hop, called source routing and distributed routing respectively. The choice of routing algorithm can be based on a number of factors including the efficiency of hardware realization and the effect of path diversity on performance.

2.2 Canonical Router Design

Previous studies [2, 15] have shown that routers can be described using a canonical router structure. A router consists of the following elements: input / output ports, link control units, buffers, a switch, and a routing and arbitration unit. Figure 1 shows a generic router with four input ports and four output ports. Input and output ports contain links that enter and exit the router. Two of these ports (one input and one output) will be reserved for connection to the processing element or host port. These routers are buffered (usually implemented using a FIFO) to store messages that cannot be immediately forwarded. Different routers may be input buffered (at the input only), output buffered (output only) or both. Between two

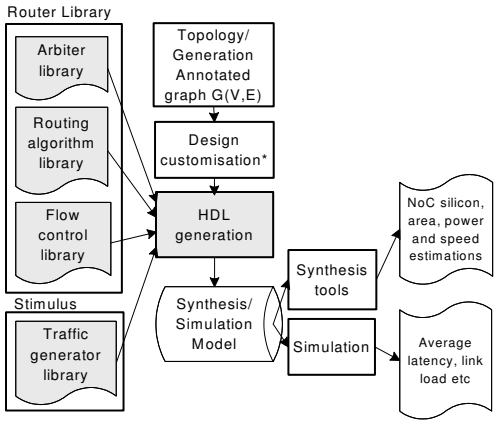

Figure 2: NoCGEN Design Flow

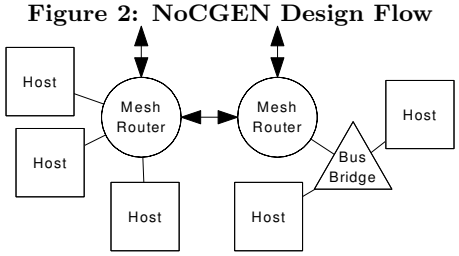

Figure 3: Partial Graph input into NoCGEN

routers the links communicate using some common link protocol. The link controller (LC) controls the flow of messages between the input ports, buffers and output ports. The routing and arbitration unit selects the destination port (described in greater detain in section 3.2). Multiple requests may arrive at the same output port and an arbiter is required to serialize the messages.

3. DESIGN METHODOLOGY

NoCGEN consists of two major parts: a library of components that make up the router and an HDL generation program. The shaded boxes in Figure 2 indicate which tools or libraries that have been built to assist the creation of the HDL output. The HDL generator takes the graph description and instantiates and port maps every router and host. Each router has a set of parameters describing it in the annotated graph (such as type, routing algorithm and bit widths). The HDL generator uses these parameters and the libraries in NoCGEN to create each router. Each link is generated as a set of signals. The HDL generation step finally produces a VHDL file describing the desired NoC.

The libraries in NoCGEN help create each router or produce the stimulus to enable simulation. These routers are made up of IP blocks consisting of routing and arbitration, flow control and buffers (see Figure 1). To enable simulation, different traffic generators serve as inputs to the interconnection network. These generators are described in the input graph and connected in the same way as the routers.

Figure 3 shows a partial graph of a hybrid bus and mesh architecture. The buses are connected to the mesh via bridges (triangles). Each router is connected via either directional or bidirectional point-to-point links. From a template topology such as a mesh, it is possible to add and remove links by editing the graph. The HDL generator visits each node and generates an input and output port for each in and out edges. Each node is instantiated and connected to signals representing the links. Parameters such as the number of ports and bus widths are passed to the HDL entities through the generics feature. The VHDL generic allows a single router block in the library to be used in designs with varied number of ports and allows internal blocks to be parameterized at design time.

3.1 Creation of Router Components

To enable reuse through modularization, the interfaces be-

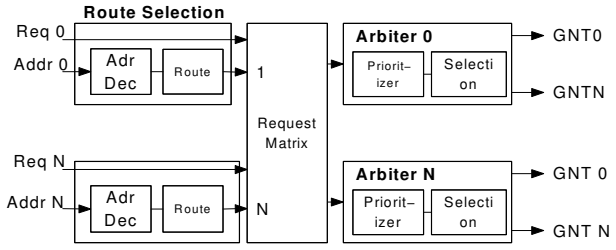

Figure 4: Arbitration block

tween components have to be compatible with similar naming conventions etc. In NoCGEN we have fixed the link control mechanism to allow router sub-module interfaces to be standardized. Variable parameters are used to ensure that the number of input ports and output ports are scalable, allowing routers to be reused for different topologies. Other examples of variable parameters are buffer sizes and the physical data width. The sub module for the flow control needs to support the use of these parameters.

Link Control Mechanism

NoCGEN uses a request, grant and ready handshake protocol to enable flow control on point-to-point links. This is similar to the interface that is used in the AMBA[1] high-speed bus (AHB). Like AMBA, it supports pipelined operation (overlapped data and address phases). It provides in-order delivery, and packets are formed into frames to ensure that they are not interrupted while traversing the network. This is achieved by locking the arbiter once a packet has started.

Switch Circuitry

The switching circuitry signals data from the inputs to the outputs. Switches are implemented as crossbars. In this paper, we have only implemented the multiplexor tree crossbar because it can be easily described and generated using synthesizable HDL. Crossbars grow by $O(MN)$ where M is number of input ports and N is the number of output ports. For a small number of ports, the size of these crossbars is still reasonable as the number of ports is small (for a 2-D mesh $N \leq 5$)

Arbitration

There are many types of arbitration schemes (e.g. matrix, round robin, lottery-based and static priority) [11] used for output scheduling that can possibly be used to replace the arbitration block (Figure 4). Arbiters select the line with the highest priority from a set of request lines. The route selection block determines the destination port of each request by decoding the address and selecting an output port. The request lines and the selected ports combine to form the request matrix that determines which input ports are requesting a certain output port. Then, the arbiter selects the port with the highest priority and asserts the grant signal.

A complex arbiter can be created by adding inputs such as buffering, source or target address into the selection decision to allow fairer distribution of traffic. Currently, our arbiters do not support the request-grant-accept interface used by more complex schemes such as iSLIP[13].

Route Selection and Packet Headers

In our implementation we separated the address signals from the data signals. Separation of address wires used for routing simplifies our route selection interface logic because the router does not have to extract the header from the data packet. This separation allows some flexibility in the size and types of packets because data can be forwarded without knowing what is contained in the packet. The downside of having this separation is that it adds log N wires (where N is number of hosts) to represent the address.

The route selection can be either adaptive or deterministic. Deterministic routing can be implemented easily using combinational logic based on the addresses or using a ROM. The routing algorithm must prevent deadlocks. In [4], a summary of some of the prevention, avoidance and recovery schemes used in traditional networking is presented. In the NoCGEN router library, we presently only have dimension order routing, which is known to be deterministic and deadlock-free. Routing, arbitration and switching are performed in one clock cycle in our implementation. It is possible to pipeline these operations to improve throughput.

Buffers

Increasing the buffer size in routers can increase accepted load until a network saturates. These buffers are implemented as a first in first out buffer (FIFO). NoCs have tight area and power constraints and therefore the buffer size has to be small. In the NoCGEN routers, buffer size is parameterizable and it is possible to change both depth and width.

Routing in Hybrid networks

Hybrid networks can be made of differing topologies connected together. These may consist of both a regular packet-switched architecture and a shared medium architecture [20] or other combinations of topology. The benefits of supporting a mixture of buses and packet switching are: (i) it is possible to aggregate bandwidth between several processing elements close together; (ii) fewer routers are required resulting in lower area; and (iii) there is support for legacy components that require a bus based interface rather than a packet switch (e.g., connecting an AMBA component to a NoC). Hybrid networks increase latency. A simple method for supporting both buses and packet switching is to encapsulate the address and control information inside the packet. This packet will be routed as data by every router until it reaches its destination bridge. The packet-to-bus bridge will de-frame the packet and convert it to the relevant bus signalling convention. Similarly, a bus based device needs to communicate through the bridge to convert bus signals into packets. Bridges between protocols are generally hand-made by inserting glue logic between the incompatible interfaces.

4. EVALUATION OF THE ENVIRONMENT

To evaluate the simulation and synthesis environment, a number of 2D meshes with differing bus widths, FIFO depths and routing were created using NoCGEN. These meshes were synthesized and simulated to obtain useful information about the interconnection network for different combinations of parameters. Xilinx XST 5.2 FPGA Synthesis tools were used to obtain an estimate of the equivalent number of gates. Modelsim and SystemC simulation system were used via the foreign language interface to simulate the VHDL interconnect network and the SystemC traffic generators together.

4.1 Simulation Environment

We used two different discrete event simulators, Modelsim and SystemC for simulation. The two simulators run interchangeably though Modelsim's foreign language interface with each reading the input and outputs of the interfaces between them before advancing the simulation. The mixed simulation speeds obtained are between 500 to 2000 clock cycles per second when simulating a fully loaded 4 x 4, 2-D mesh. The simulation speed is relatively slow due to the synthesizable VHDL model. The trade-off to this lack of speed is the increased accuracy is the ability to accurately estimate in performance and size.

4.2 VHDL to SystemC wrappers

SystemC provides high level simulation tools to create complex functional verification test benches and stimuli. A bit-level to transaction level interface was constructed in SystemC, to allow high-level traffic generators to be created. These traffic generators can be classified as consumers and producers. The consumer component converts events into calls to an interface method, while the producer part of the wrapper converts calls to events. From this, a test bench can be created by a mixture of blocking and non-blocking reads and writes.

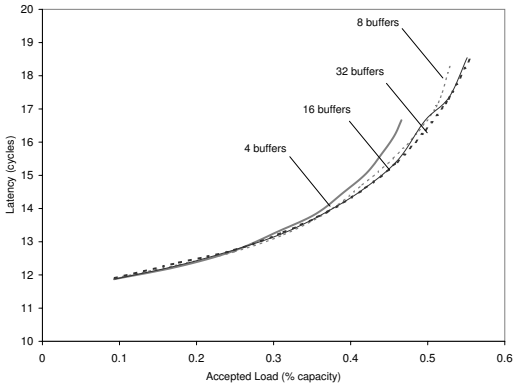

Figure 5: Accepted Load vs. Latency for different buffering sizes

Producer Interface

To send a packet through the mesh using a blocking request, the process must call the write method as follows:

```
delay = write(address, data, length);
```

To implement a simple random waiting traffic generator, we have added an infinite loop with a random waiting time between each write operation. The use of SystemC wrappers augments and abstracts the behavior of the host elements in the network. These can be as complex as an instruction set simulator. More complex stimuli such as instruction set simulators can be attached in the same fashion by intercepting their read and writes.

Consumer Interface

The consumer interface provides a mechanism for responding to events on the host ports of the router. Whenever, a packet is received at the destination, the host invokes a callback interface. This interface is implemented as a blocking read. A typical host would wait for an event and respond to it by sending a response. In the case of our traffic generators, the consumer waits until a packet is received and the relevant received event is logged for further analysis

5. CASE STUDY

We simulate and synthesize a wormhole routed mesh router with the following configurations: physical data width size of 4, 8, 16 and 32 bits; varying buffer depth from 4 to 32 flits; and varying mesh sizes from 2 x 2 to 4 x 4. We use dimension-ordered routing, which is known to be deterministic and deadlock-free. A simple Poisson distribution with randomly distributed destinations and load increasing from 10% to 100% was used to verify the effect of different routers under different loads. The traffic generators were connected to a "packet to bus" bridge that framed and de-framed packets onto a bus. This bridge interface incurs an extra clock cycle latency.

Figure 5 shows the effect of increasing the buffer size in a 4 x 4 wormhole router. We send packets from a source node to a destination node and log the arrival time. As expected, increasing the buffer size generally increases the accepted load. The wormhole routed mesh saturated at about 55.8%. It must be noted that this traffic pattern is only for used to validate NoCGEN, as a mesh topology it is not ideal for random traffic. It is likely that a fat-tree would perform better in this case as the average number of hops is lower for random traffic.

For a 5 x 5 mesh router, the VHDL description was 1706 lines, 130 sets of directional links that need to be connected, each with 6 signals to connect to 25 routers and 25 hosts. The area results are shown in Figure 6. It was found that for the mesh configuration, the increase in area is linear with the increase in the bus width. This result is fairly intuitive because the crossbar switches are likely to dominate the area in the FPGA configuration. By increasing the data bus width, we increase the number of multiplexors needed.

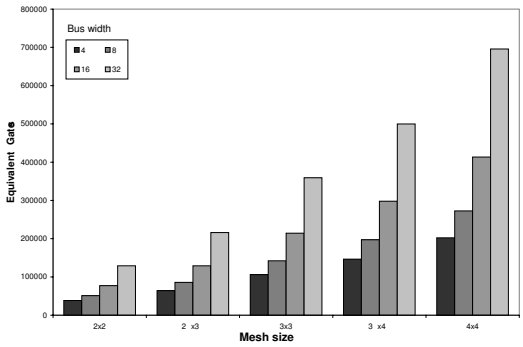

Figure 6: Effect of different mesh sizes on area

6. CONCLUSIONS AND FUTURE WORK

We have proposed a rapid Networks on Chip generation methodology, based on an input graph and a set of libraries to allow both synthesis and simulation. We have shown that routers can be modularized to reuse components. Furthermore, we created several modules that can be used to make NoC routers. The modules are parameterizable to allow rapid application specific customization. Future development will continue to develop new router models, and have a greater number of options in the libraries. We also aim to use these models to rapidly estimate power. In addition, we plan to use this tool as a mechanism to gain insights into different hybrid NoC architectures.

References

[1] ARM. Amba 2.0 specification, 2002.
[2] A. Chien. A cost and speed model for k-ary n-cube wormhole routers. In *Hot Interconnects*, Stanford, 1993.
[3] W. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *DAC*, pages 684–689, 2001.
[4] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An engineering approach*. Morgan Kaufmann, 2003.
[5] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *DATE*, pages 250–256, 2000.
[6] R. Ho, K. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, 2001.
[7] J. Hu and R. Marculescu. Energy-aware mapping for tile-based noc architectures under performance constraints. In *ASPDAC*, 2003.
[8] M. Keating. *Reuse Methedology Manual*. Kluwer Academic Publishers, 1998.
[9] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *ISVLSI*, 2002.
[10] K. Lahiri, A. Raghunathan, and S. Dey. Fast performance analysis of bus-based system-on-chip communication architectures. *Computer-Aided Design*, pages 566 –572, 1999.
[11] K. Lahiri, A. Raghunathan, and G. Lakshminarayana. Lotterybus: A new high-performance communication architecture for system-on-chip designs. In *DAC*, Las Vegas, 2001.
[12] J. Liang, S. Swaminathan, and R. Tessier. Asoc: a scalable, single-chip communications architecture. In *Parallel Architectures and Compilation Techniques*, 2000.
[13] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, pages 188–201, 1999.
[14] P. G. Paulin, C. Pilkington, and E. Bensoudane. Stepnp: A system-level exploration platform for network processors. *IEEE Design and Test of Computers*, pages 17–26, 2002.
[15] L.-S. Peh. *Flow Control and Micro-Architectural Mechanisms for Extending the Performance of Interconnection Networks*. Phd, Stanford University, 2001.
[16] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *DATE*, 2003.
[17] A. Sangiovanni-Vincentelli and G. Martin. Platform-based design and software design methodology for embedded systems. *IEEE Design and Test of Computers*, pages 23–33, November 2001 2001.
[18] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, R. Jan, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC*, 2001.
[19] D. Siguenza Tortosa and J. Nurmi. Vhdl-based simulation environment for proteo noc. In *HLDVT02*, Cannes, France, 2001.
[20] P. Wielage and K. Goossens. Networks on silicon: blessing or nightmare? In *Euromicro Symposium on Digital System Design*, pages 196–200, 2002.
[21] D. Wingard. Micronetwork-based integration for socs. In *DAC*, Las Vegas, 2001.