Modeling and Analysis of On-Chip Networks

Reinaldo Romero
Electrical Engineering
Penn State Univeristy
rxr246@psu.edu

Graduate Mentor: **Alessandro Pinto**
Research Supervisor: **Dr. Jonathan Sprinkle**
Faculty Mentor: **Prof. Shankar Sastry**

July 29, 2005

Summer Undergraduate Program in
Engineering at Berkeley (SUPERB) 2005

Department of Electrical Engineering and Computer Sciences
College of Engineering
University of California, Berkeley

Modeling and Analysis of On-Chip Networks

Reinaldo Romero

Abstract

The complexity of hardware platforms doubles every eighteen months. The number of processing elements on the same chip is going to be of the order of hundreds in the near future which makes the communication infrastructure very difficult to design. Constraints, especially in terms of power consumption, must be taken into account and the communication infrastructure has to be highly optimized. The optimization of a network gives different results depending on the trade-off between communication and computation. In this work we derive an expression for such trade-off and we predict how future communication topologies will look like by analyzing on-chip networks using simple analytical models.

1 INTRODUCTION

An on-chip network is made up of a number of network modules including the processors, memories, peripheral controllers, gateways to networks on other chips, and network logic. Instead of connecting these modules using dedicated wires, and/or buses, they are connected to a network that routes packets between them. Each module is placed in a square tile on the chip and communicates via the network with every other module, not just its neighbors. The network logic occupies a small amount of area (about 6.6 percent) in each tile and makes use of portion of the upper two wiring layers [1]. On-chip networks are a very popular and favorable interconnect framework for high-performance systems-on-a-chip (SoCs) and they have been implemented on several chip multiprocessors (CMPs). These on-chip networks have had to meet very tight delay requirements, therefore prior designs and microarchitectures have been deeply performance-driven, looking for lower network delays. However, the systems that are trying to be implemented with on-chip networks are becoming more and more power-constrained. One of the primary concerns with today's portable devices is battery life. In PDAs, laptops and other mobile devices their embedded SoCs need to have the best possible battery life. Designers are pressured by systems' power budgets to address this major constraint. In trying to satisfy the increasing demand for interconnect bandwidth, on-chip networks are taking up a sizeable portion of system power budget. For example, the MIT Raw on-chip network connects 16 tiles of processors consumes 36 percent of total chip power, with each router dissipating 40 percent of individual tile power. Thus, although the bandwidth is met, the power dissipation is a big problem. In designing and implementing on-chip networks it is important to understand what underlying factors are responsible for the power consumption problem.

This study is concerned with the analysis of the trade-off between computation and communication. By this, we mean the relative cost of communication of a datum on a link with respect to switching the same datum in a router. Each network component is characterized by a cost. The cost metric that we use is the energy-delay product that summarize both the power consumption and the performance of each component and of the entire network. We model links and routers to derive and expression for the communication/computation trade-off and we plot the result for a $130nm$ technology.

2 Motivation

The most prevalent technologies used in routing signals within chips are buses and global interconnects. These technologies have had difficulties keeping up with the increasing demand for bandwidth and the complexity of multiprocessor chips. Using an on-chip network has many advantages like structure, performance, and modularity.

Since the introduction of on-chip networks, researchers have been focusing their attention on regular interconnection topologies like meshes and tori. Such structures are particularly interesting because are able to well organize the top-level wires simplifying their layout and giving them well-controlled electrical parameters. In turn, these well controlled electrical parameters enable the use of high-performance circuits that result in significantly lower power dissipation, higher propagation velocity, and higher bandwidth that is possible with conventional circuits. The improvements in performance due to these aggressive circuits more than offset the performance lost to network overhead [1]. Moreover, these networks also make more efficient use of wires because there are a lot fewer idle wires. The reduction in the size of wires greatly reduces the capacitance and thus communication latency.

In this work we use an alternative approach to motivate our work. Starting form the point-to-point specification of a communication problem, a network topology can be synthesized. The synthesis algorithm aims at minimizing the sum of the energy delay products of all links and routers used to build a network.

Consider the following experiment. We place 9 processors on a grid and we generate some communication constraints between them. Each constraint is represented by the bandwidth that an end-to-end connection has to satisfy and of course by the distance that has to be spanned from a source to a destination. We then use a synthesis algorithm developed at the University of California at Berkeley and that is part of the COSI (COmmunication Synthesis Infrastructure) project. The synthesis algorithm optimizes the node positions and the routing of data using a cost definition as follows:

$$Cost = \sum_{e \in E} C_e(e) + \lambda \sum_{v \in R} C_v(v)$$

where we have denoted with E the set of all links and with R the set of all routers. The cost of a link is the product of the bandwidth it carries times the distance squared while the cost of a router is the sum of its input bandwidths. λ is a parameter that determines the relative cost of links and routers.

In our experiment we fixed the number of router to 9 and we changed λ to see how the cost of the resulting network changed with it. The result is shown in Figure 2.

The cost of the synthesized network is almost constant for $\lambda > 0.3$ while drops for $\lambda < 0.3$. This effect motivates our study. We want to find an expression for λ to understand what are the factors that contribute to its value in order to understand what can be done to keep it below the threshold.

3 Modeling On-Chip Networks

When analyzing on-chip networks it is important to understand the two network elements contributing to the power consumption: the wires and the routers. While there are a lot of analytical model for on-chip wires that can be used to estimate the cost of a link, we wanted to have a clear idea of the structure of a router in order to have a good model for the energy consumption and the number of clock cycles needed for each switch. The internal scheme of a router is shown in figure 2. Routers were modeled in Verilog HDL, a hardware description language. Verilog HDL allows the designer to model the router hierarchically using modules. In this design

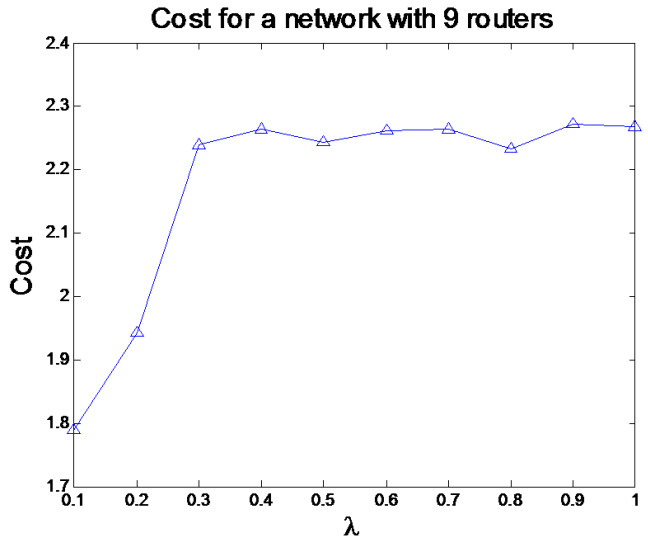


Figure 1. Cost of a synthesized network with respect to λ

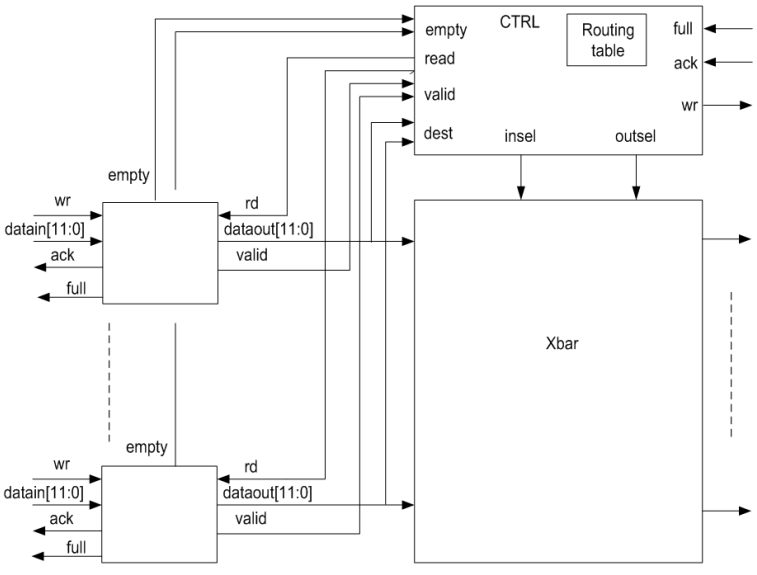


Figure 2. Internal structure of a router

the router consisted of three module types, the FIFOs, the Xbar, and the controller. Moreover, these modules also had sub-modules themselves. Using modules to construct network elements in Verilog HDL makes for a much cleaner looking code that can be modified more easily as well. Also, when you have a module for a network element it can be instantiated as many times as need.

3.1 The FIFO: First-In-First-Out Memory Structures

The FIFOs are first-in-first-out memory structures that were instantiated several times in the router. These FIFO's write to the first available memory space and the first memory space is read after each read request. A value can be shifted in when there is space to do so. If there is a read request then it retrieves the data stored for the longest time. The FIFOs are used to buffer data between the producers of the data and the recievers of the data.

The FIFO is composed of a write FSM module, a read FSM module and a memory module. In making the FIFO, the first module that was modeled was the write FSM. The write FSM has three states that it cycles through, the idle state, the write state, and the acknowledge state. It is a mealy FSM, which means that it shifts between states taking into consideration the input and its previous state. Initially the write FSM is in the idle state, waiting for write requests. When the FIFO receives write request it shifts to the write state and write the input data to the first empty memory slot. Upon writing the input data into the memory slot the FIFO outputs an acknowledge signal telling the requester that the tack has been completed. In addition, a memory address counter is increased each time there is a write request. Once the write address pointer reaches its maximum value it rolls over to the first address again to cycle through the address again. The read FSM is just like the write FSM and it shifts between similar states of behavior except that now it will read info from a memory space, not write. The read FSM's three states are the idle state, the read state, and the valid state. The FIFO does nothing while in the idle state. When a read request is sent to the FIFO, the FIFO shifts to the read state. Once the data has been read a valid output signal is sent from the FIFO to tell the requester that the data has been sent out.

The FIFOs were made with 256 memory slots and the data could be 8 bits wide. When testing the FIFO the test bench was made so that the data written was the address number, so data 0 was written in memory slot 0 and data 255 was written in memory slot 255. Doing this simplified testing the full and empty signals because as soon as 255 was written into the FIFO the full signal would go high. A screen shot of this full signal can be seen below in figure 3.

3.2 The Xbar

The Xbar is a grid of wires and multiplexers (mux) and demultiplexers (demux). The mux and demux route the signals to their appropriate output channels based on selector signals from the controller. The Xbar can be thought of as the roads with street guards directing traffic much like the mux and demux do. Then the controller is like the traffic lights, directing the flow of traffic.

In Verilog the Xbar was designed by instantiating multiple mux and demux modules. The mux and demux are complete inverses of each other. When modeled at the behavioral level these two modules consist of case statements for every selector combination. The mux was designed as a 4-to-1 mux and the demux was designed as a 1-to-4 demux. They were given two bit wide selectors to controller the routing of signals. The Xbar was a 4x4 module which means that it took in 4 input signals and based on the selectors outputted the signals to their corresponding output ports. The Xbar physically is four instantiations of the mux modules and four in-

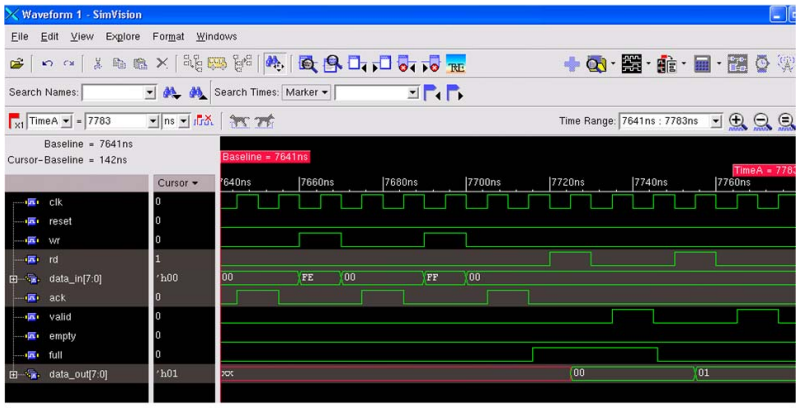


Figure 3. FIFO Verilog simulation result

stantiations of the demux modules.

3.3 The Controller

The controller is a large FSM that generates the selector signals for the Xbar's mux and demux. The controllers in routers are extremely important in getting the data to the proper destinations. It first sends a read request to the FIFO and then read the input data. The input data is twelve bits wide and the last four bits are the destination address bits. The controller has a routing table within it where it is able determine what the output port for this data should be so that it will be routed to the destination port or another router that can route it closer to the destination. The controller also takes into consideration the empty, full, valid, and acknowledge signals. If the FIFOs are empty then there is really nothing that needs routing. The valid signals are important because the controller wants to read a memory slot from each FIFO and until it receives the valid signals from all of the FIFOs confirming that the data has been read it will not proceed to generating selector signals to route the data to the proper output ports of the router. To test the controller the FIFO and the Xbar needed to be wired togther with the controller. Then a test bench is able to be made to determine if the proper selector signals are being generated to route the signals thought the Xbar properly and to the correct output port.

3.4 The Router

Having used Verilog's hierarchy design method, the router was created by instantiating the three major modules, the FIFO, the Xbar, and the controller. The Xbar and the controller were both instantiated once, but the FIFOs were instantiated four times, one for every Xbar input port. The critical part of creating a model through instantiations is to connect the input ports of one module to the output ports of the corresponding module.

The testbench can then show whether or not the data is being properly routed.

4 Analysis of On-Chip Network's Performance

Analytical models were used to assess the performance of the on-chip network. Network synthesis reveals the network cost Equation 1, shown below. Taking a closer look the total cost of the edges and vertices depend on the energy and delay product of both the edges, or wires and the vertices, or nodes. Equation 2. shows that the router's cost is directly proportional to the energy and delay product (EDP). Similarly, Equation 3 says that the cost of the wires is also proportional to the EDP. Therefore, to determine the cost of the network, the energy and logic delay for both the router and the wires needed to be modeled. In modeling these costs some generalization was required.

$$Cost = \sum_{e \in E} C_e(e) + \lambda \sum_{v \in R} C_v(v) \tag{1}$$

$$C_e^{'}(e) = \alpha\omega(e)l^2(e)\epsilon_W d_W \tag{2}$$

$$C_v^{'}(v) = \alpha \left(\sum_{(i,v) \in E} \omega(i,v) \right) \epsilon_R \frac{n}{f_{clk}} \tag{3}$$

4.1 Modeling the Router's EDP

For the router, it was assumed that the Xbar and the controller consumed very little power compared to the FIFO. When the FIFOs write and read the power consumption is substantially larger than the rest of the router components. Therefore, the router's energy consumption is approximately equivalent to the memory's energy which is given as sum of the energy of the address decoder and the energy of the registers, this is seen in Equation 5 Furthermore, router's delay depends on its implementation. This means that the delay is controlled by the number of clock cycles per bit, n, and the clock frequency, fclk. This is seen in Equation 4, and this explains the factors influencing the value of.

$$delay_R = nt_{clk} \tag{4}$$

$$Energy_R \approx Energy_{mem} = E_{addrdec} + E_{reg} \tag{5}$$

4.2 Modeling the Wires' EDP

The edges, or wires contribute significantly to the total cost of the on-chip network. In order to include their cost into the total cost model, a model was needed to approximate their contribution. A model was choosen based on the work in [2] on global interconnect optimization. In their research, Mui et.al., determined a set of parameters that governed the power and delay of interconnects. In calculating these performance factors the wires were generalized to be optimal wires. In their work, Mui, et.al., modeled the power-per-unit-length for interconnects, this power-per-unit-length was modified to give the energy-per-unit-length of interconnect wires.

The interconnect technology 130mm, was chosen to determined the resistance, r and capacitance, c of the wires.

$$delay_W = 2\sqrt{r_s c_o r c}\left(1+\sqrt{\frac{1}{2}\left(1+\frac{c_p}{c_o}\right)}\right) Llog(2) \tag{6}$$

$$Energy_W = \alpha V_{DD}^2\left(\frac{k_{opt}}{h_{opt}}(c_o+c_p)+c\right)+\alpha V_{DD} W_{nm} I_{SC} ln(3) k_{opt}\frac{\tau_{opt}}{h_{opt}} \tag{7}$$

4.3 Modeling Objective

The goal was to evaluate lambda and determine what parameters control it. These parameters could have either been all implementation-based parameters, all technology-based parameters, or a combination of both. Knowing what parameters depends on allows network designers/architects to determine if a network can be optimized and hence be cost efficient. If an interconnect technology is chosen and is studied further, it is seen that at a certain threshold value it reaches a saturation cost. Therefore, designers can know whether or not a network with a certain number of routers can be optimize using a present interconnect technology. Another insight that gives is what implementation setting allow for the best possible. If these implementation setting are reasonable, a network can be implemented for optimal performance and low cost. If it cannot be optimized with present technology, perhaps there exists a promising new technology that can be invested in to improve, and achieve the optimized network.

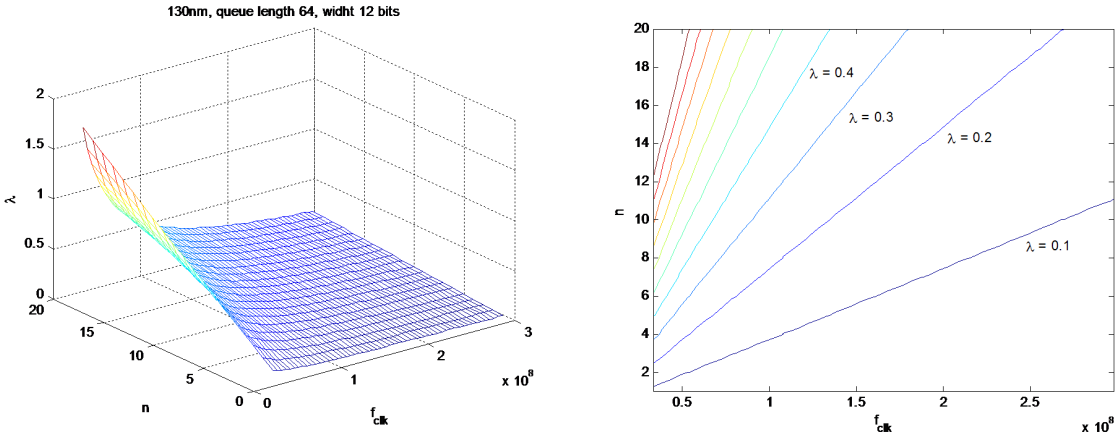
5 Results

An expression for λ can be computed by comparing the general expression of the cost of network given in Equation 1 and the energy-delay product for routers and wires. The expression is given in Equation 8.

$$\lambda = \left[\frac{\epsilon_R}{\epsilon_W d_W}\right]\frac{n}{f_{clk}} \tag{8}$$

The first term $\epsilon_R/(\epsilon_W d_W)$ is technology dependent because the energy of the router, ϵ_R and the energy of the wires, ϵ_W and the delay of the wires, d_W are all technology parameters. The second term n/f_{clk} is the ration of the number of clock cycles needed to process one packet divided by the router clock frequency. n represent a measure of the parallelism of the router: if the router architecture is very parallel then n is small. The parameter f_{clk} represent instead the speed at which the router runs. Both of them have a big impact on the effort that is needed to build the router: n has a impact on the area and f_{clk} on the logic on a combinational path that has to be highly optimized.

Figure 4 shows the results in the case of routers with 4 inputs, 4 outputs and queue length of 64. The contour plot shows the trade-off between n and f_{clk} for the same value of lambda. In our case for instance, the number of clock cycles per packet is 12 that depends on the number of clock cycles for writing and reading into and from the FIFOs, plus the number of clock cycles taken by the controller for address lookup and selectors computation. In order to have $\lambda \leq 0.3$ our router should operate at $f_{clk} \geq 120MHz$ which is an acceptable frequency.



(a) (b)

Figure 4. λ **as a function of** n **and** f_{clk} **for a 130nm technology and queue lenght of 64.**

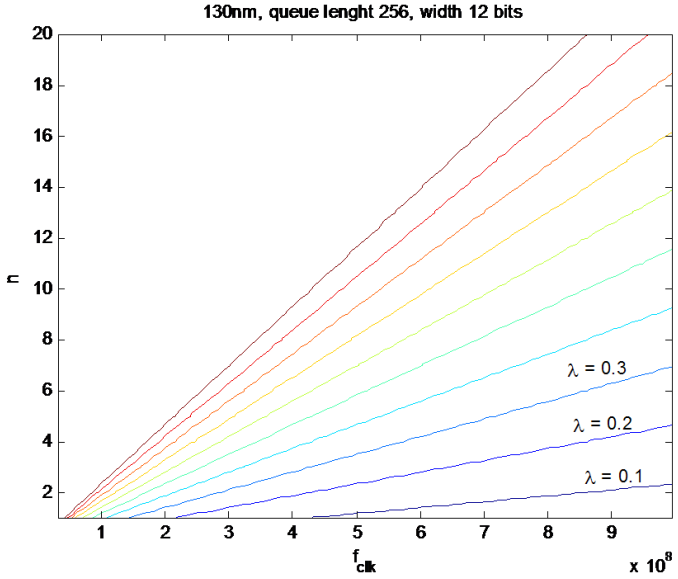


Figure 5. λ **as a funtion of** n **and** f_{clk} **for a 130nm technology and queue lenght of 256**

Figure 5 shows the same contour plot for queue length equal to 256. In this case the router complexity is much higher and in order to have λ small we have to build a very good router in terms of performances. In our case for instance, the operation frequency should be greater than $1GHz$. Alternatively, the router architecture should be changed to have a much more parallel implementation. For instance, if we want to operate at $500MHz$ our router should not take more that 4 clock cycles to process one packet.

6 Conclusion

Modeling on-chip networks provides great insight into the functionality and performance of the interconnect technology. The router model served to illustrate components that are responsible for controlling the flow of traffic. It is important to have a grasp of the functionality of the router before its performance can be modeled and understood. The Verilog HDL model of the router allowed for greater reasoning when it came time to model the performance. Due to time constraints the on-chip network model could not be synthesized. Nevertheless, analytical models were made to create and approximate the performance of on-chip networks. The critical lambda factor was studied and evaluated based on these models. This lambda factor is critical because it reaches a saturation cost where it implies that that technology in use will have a constant cost afte a certain threshold lambda value. Designers that know the threshold lambda value can utilize it to determine if a network can be optimized or not. The lambda factor was revealed to have dependence on both the implementation and the technology of choice. Knowing lambda the possiblity of optimization was considered. The total cost of an on-chip network can be optimized by first determining the threshold lambda and then either using a new technology of implementation.

References

[1] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proceedings of the Design Automation Conference*, pages 684–689, Las Vegas, NV, June 2001.

[2] M. L. Mui, K. Banerjee, and A. Mehrotra. A global interconnect optimization scheme for nanometer scale vlsi with implications for latency, bandwidth, and power dissipation. *IEEE Transaction on Electron Devices*, 51(2):195–203, 2004.