*IRWIN AND JOAN JACOBS*
*CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES*

# The Design of a Latency Constrained, Power Optimized NoC for a 4G SoC

**Isask'har Walter, Israel Cidon, Avinoam Kolodny, Rudy Beraha**

**CCIT Report #724**
**March 2009**


*DEPARTMENT OF ELECTRICAL ENGINEERING*
*TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL*

CCIT REPORT #724 March 2009

# The Design of a Latency Constrained, Power Optimized NoC for a 4G SoC

Isask'har Walter[1], Israel Cidon[2], Avinoam Kolodny[2]
Electrical Engineering Department, Technion – Israel Institute of Technology, Haifa 32000, Israel
Rudy Beraha[3]
Qualcomm Corp. Research and Development, San Diego, California 92121, USA
[1]zigi@tx.technion.ac.il , [2]{cidon, kolodny}@ee.technion.ac.il [3]rberaha@qualcomm.com

## Abstract

*In this paper, we examine the process of porting a high-end commercial SoC application from a segmented bus implementation to a NoC-based one. We present several design choices and focus on the power optimization of the NoC while achieving the required performance. Our design steps include module placement optimization using simulated annealing and allocation of minimal and different capacities to links. Unlike previous studies, in which point-to-point, per-flow timing constraints were used, we introduce and demonstrate the importance of using the application's end-to-end traversal latency requirements during the optimization process. In order to quantify and evaluate the different alternatives, we report the actual throughput and timing requirements of the commercial SoC as well as the synthesis results. According to our findings, the proposed technique offers up to 40% savings in the total router area and a reduction of up to 49% in the inter-router wiring area.*

## 1. Introduction

Application-specific systems on-chip (SoC) make extensive use of busses as the interconnect infrastructure. These busses are typically enhanced along product generations to match the increasing needs of the application. Such enhancements include increasing the bus frequency and width as well as enriching the bus semantics and transfer modes. By avoiding fundamental changes, the SoC architects can leverage their past experience in designing shared busses and successfully overcome the growing complexity of the design. However, in recent years research has shown that Network on-Chip (NoC) is likely to replace busses in future SoCs, due to superior performance, power and area tradeoffs it offers as the number of modules increases [1][2][3][4]. This is mainly attributed to the spatial parallelism and statistical multiplexing of networks, to their short, unidirectional point-to-point wires and to their scalable architecture [5].

NoCs are being adopted by companies as a means to improve design productivity. As the number of modules connected to a bus increase, the physical implementation of the bus becomes very complex, and achieving the desired throughput and latency requires time consuming custom modifications. Conversely, NoCs are designed separately from the functional units of the system to handle all foreseen inter-module communication needs. Their inherent scalable architecture facilitates the integration of the system and shortens the time-to-market of complex products.

In this work, we discuss and evaluate the design process of a NoC for a state-of-the-art SoC. More specifically, we describe our experience in porting a high-performance, power constrained 4G wireless modem application from a segmented bus based architecture to a cost optimized NoC architecture. As the design process includes many "degrees of freedom" creating a very large design space, finding the absolute optimal solution is an extremely difficult problem. Instead, we focus on some of the important choices that should be made by the system architect while selecting some well-accepted, practical solutions to other questions.

Previous work that has dealt with the design process of the NoC frequently attempted to minimize power consumption and/or maximize network performance as measured by the network's throughput and latency. When real applications are considered, simply minimizing the power consumption alone (e.g., by module placement) is impossible, as performance constraints for each given application are to be met. Similarly, maximizing performance alone is inefficient, since excessive power might be used for improving performance beyond the needs of the application. Therefore, in this paper we look for a tradeoff between the power and performance of the NoC that is characterized by a minimal power consumption that still meets the demands of all targeted applications.

Moreover, in many of the studies where network latency was used as a performance goal (either as the optimized cost function or as a constraint), the average delay of all packets over all communicating pairs was considered. However, in a practical SoC, different streams of communication may require different delays and therefore the overall average latency is an inappropriate measure. Consequently, the individual per-flow, point-to-point (source-destination) latencies should be accounted for to get better results. In this paper, we go further to suggest a third, improved approach: knowing the application that is to be used in the SoC, we utilize its functional timing requirements, which are defined by the application end-to-end latency constraints. Each of those *end-to-end traversal delay* requirements is composed of the cumulative requirement of a sequence (or a "chain") of flows. For example, the application may require that a block of data which is generated by module A is sent to module B in order to be processed. Then, the processed data is to be sent by module B to module C for some additional processing. By observing that the performance of the application is subject to the total time it would take the data to get from module A to module C, we can use this delay as the targeted performance measure, rather than specifying the two separate latency constraints (for the flow from module A to module B and from module B to module C). Since pair-wise delays may be traded, the timing constraints are relaxed and the optimization program can use more freedom in its operation. To the best of our knowledge, this paper is the first to discuss and quantify the benefits of specifying the end-to-end traversal requirements during the design process.

The design process itself has several steps. First, using simulated annealing optimization, we search for a module's placement consuming minimal power, taking into account application latency and throughput constraints. Then uniform link capacities among routers are defined to meet these performance constraints. Finally, the resulting uniform NoC is tuned by reducing the capacity of selected links. The design process case study becomes more complex as our SoC has several modes of operation, each with a different set of traffic and latency requirements. The designed NoC needs to meet the requirements of all these modes while reducing power as much as possible.

As an important part of this work, we present the bandwidth and timing requirement of the high-performance, state-of-the-art 4G application we examine. This information can be used by the NoC community as a benchmark for future research.

The rest of this paper is organized as follows: in Section 2 related work is discussed. In Section 3, we describe the characteristics of the application of the designed SoC. In Section 4, we discuss the design and optimization process of the NoC, and analyze its cost and performance. In Section 5 we summarize the paper. Appendix A describes the architecture of the NoC components.

## 2. Related Work

NoC design was the subject of many papers in recent years. In particular, the problem of mapping the communicating cores onto the die has received considerable attention, due to the power and performance implications it has. In [6], the authors propose a branch-and-bound mapping algorithm to minimize the communication energy in the system, but the resulting communication delay is not considered. In [7], a heuristic algorithm is used to minimize the average delay experienced by packets traversing the network. By allowing the splitting of traffic, an efficient implementation is found. In [8][9], the authors use the message dependencies of the application in addition to its bandwidth requirements to find a mapping that reduces the power consumption and the application execution time. The authors of [10] use a multi-objective genetic algorithm to explore the mapping space so that a good tradeoff between power consumption and application execution time is found. While these papers use unique mapping schemes, they all use packet delay or application execution time as a quality measure rather than as an input to the mapping phase. Moreover, the metrics used does not consider the individual requirements of each pair of communicating cores, only reflecting the overall average delay or performance.

The earliest published work to consider energy efficient mapping of a bandwidth and latency constrained NoC is [11], in which the authors specify a an automated design process providing quality-of-service guarantees. Another mapping scheme that uses delay constraints as an input is described in [12]. There, a low complexity heuristic algorithm is used to map the cores onto the chip and then routing is determined so that all constraints are met. Similarly, the mapping schemes used in [13][14][15] all use the per-flow, source-destination latency requirements of the application as input to the design process.

In this work, we motivate a third approach: rather than optimizing the NoC for power only and evaluating the resulting delays; or using the per-flow delay requirements as constraints during the design process,

we suggest using the end-to-end traversal delay latencies dictated by the needs of the application. Wherever applicable, we replace "a chain" of point-to-point delay constraints with a single, unified constraint, describing the end-to-end latency requirement of the application, measured from the time the first module in the chain generates the data until the last module receives the data, as explained above.

## 3. The Application

The design chosen for evaluating the conversion into the NoC architecture is an ASIC that supports all major 2G, 3G and 4G wireless standards for use in base stations and femto-cells (Cell Site Modem – CSM), depicted in Figure 1. The CSM is designed to support any of the CDMA or UMTS standards, because different markets around the world are at different points in their adoption of wireless standards.

This CSM is comprised of several subsystems. These fall into three basic categories: (1) Generic Element. These are the processor and DSP modules on chip. They are programmable and can be used for a variety of different functions; (2) Dedicated hardware. These blocks are designed to optimize the operations/milliwatt metric. They perform a single (or a small set) of operations extremely efficiently and off-load the work from the generic elements (which typically could perform the same operation but with a significant power penalty); and (3) Memory/IO. As with most SoCs, there are memory elements and I/O modules used for information storage and communication with the outside. For the purposes of this paper, these elements are grouped together.

The SoC consists of several subsystems tied together with a 64-bit wide, 166MHz AXI bus at the top-level. There are a total of 34 nodes on the bus. Due to design considerations such as P&R and timing closure, the interconnect fabric is segmented into two separate busses with approximately half the nodes on each bus and a bridge between the busses.

The CSM chosen for this study supports multiple modes of operation, each identified by its own BW and latency requirements. In particular, it can operate in a 2G mode, in a 3G mode, or in a 4G mode. There is also a combination of modes for simultaneous voice and data transmissions. To find the low-cost 2D mesh topology for the NoC, an artificial set of bandwidth requirements is generated: for each pair of nodes, the maximum BW requirement it has in any of the modes of operations is selected. Similarly, we combined all the latency requirements in one table. This scenario, which we call "MAX", represents the worst-case requirements in any of the modes ("synthetic worst-case" in [16], "design envelope" in [17]). Designing the NoC according to the “MAX” scenario is likely to make it easier to meet requirements of all modes of operation in the following phases of the design.

If we expect to design a NoC to replace the top-level AXI busses, it must be flexible enough to meet the BW and latency requirements for each mode. However, we also do not want to overdesign the network because this will waste area and power. A summary of the bandwidth and latency requirements is given in the tables 1-3: Table 1 describes the bandwidth requirement between master and slave modules in the system. Table 2 describes the timing requirements of point-to-point communication in the system while Table 3 specifies the application's end-to-end traversal delay requirements, derived from the application characteristics. Additional characteristics, that are omitted here due to space limitations, are

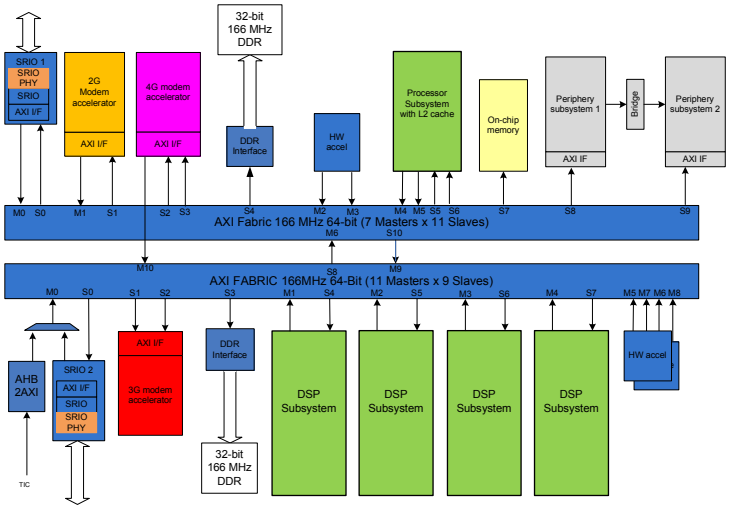


**Figure 1: Bus-based system architecture**

**Table 1**

| | | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 | S16 | S17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M0 | R | 0 | 0 | 492 | 0 | 3 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 492 | 0 | 53 | 0.6 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 38 | 0 | 0 | 0 | 0 |
| M1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 20 | 201 | 0.5 | 1 | 0.5 |
| | W | 0 | 0 | 0 | 0 | 53 | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 | 35 | 2 | 2 | 1 | 202 | 1 |
| M2 | R | 0.3 | 2 | 0 | 0.4 | 125 | 0 | 0.2 | 250 | 4 | 1 | 0 | 0.4 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0.4 | 2 | 0 | 5 | 0.1 | 0 | 0.2 | 125 | 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M3 | R | 0 | 2 | 0 | 1.5 | 0.1 | 0 | 1 | 5.6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 2 | 0 | 61 | 0.1 | 0 | 1 | 4.4 | 0 | 0 | 0 | 41 | 0 | 1.9 | 0 | 0 | 0 | 0 |
| M4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.3 | 0 | 1 | 0.5 | 1 | 0 | 10 | 0 | 10 | 10 | 10 |
| | W | 0 | 0.3 | 0 | 0.3 | 0 | 2.5 | 0 | 1.3 | 0 | 1 | 2 | 1 | 0 | 10 | 0 | 10 | 10 | 10 |
| M5 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.3 | 0 | 1 | 0.2 | 1 | 0 | 10 | 10 | 0 | 10 | 10 |
| | W | 0 | 0.3 | 0 | 0.3 | 0 | 2.5 | 0 | 1.3 | 0 | 1 | 1 | 1 | 0 | 10 | 10 | 0 | 10 | 10 |
| M6 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.3 | 0 | 1 | 0.5 | 1 | 0 | 6 | 10 | 10 | 0 | 10 |
| | W | 0 | 0.3 | 0 | 0.3 | 0 | 2.5 | 0 | 1.3 | 0 | 1 | 2 | 1 | 0 | 10 | 10 | 10 | 0 | 10 |
| M7 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0.2 | 1 | 0 | 6 | 10 | 10 | 10 | 0 |
| | W | 0 | 0.3 | 0 | 0.3 | 0 | 2.5 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 10 | 10 | 10 | 10 | 0 |
| M8 | R | 13 | 0 | 0 | 0 | 0 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M9 | R | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 29 | 0 | 0 | 0 | 1 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M10 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 415 | 0 | 1 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 215 | 400 | 0 | 1 | 0 | 0 | 0 | 0 |
| M11 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 400 | 400 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 400 | 400 | 0 | 0 |
| M12 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 400 | 0 | 1 | 3.1 | 39 | 3.6 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 400 | 0 | 1 | 0 | 0 | 200 | 200 |
| M13 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 400 | 400 |
| | W | 9.4 | 0 | 0 | 36 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 400 | 400 |
| M14 | R | 0 | 0 | 0 | 0 | 15 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 23 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M15 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 153 | 160 | 0 | 1.2 | 0 |
| | W | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 54 | 0 |

Bandwidth demands [Mb/s].
'R' is for read operations, 'W' is for write operations.

**Table 2**

| | | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | S11 | S12 | S13 | S14 | S15 | S16 | S17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M0 | R | 0 | 0 | 5000 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 5000 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M1 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M2 | R | 0 | 0 | 0 | 500 | 300 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 500 | 300 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M3 | R | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M4 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 200 | 200 | 200 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 200 | 200 | 200 |
| M5 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 200 | 0 | 200 | 200 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 200 | 0 | 200 | 200 |
| M6 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 200 | 200 | 0 | 200 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 200 | 200 | 0 | 200 |
| M7 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 200 | 200 | 200 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 200 | 200 | 200 | 0 |
| M8 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M9 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M10 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 150 | 150 | 0 | 0 | 0 | 0 | 0 | 0 |
| M11 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 100 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 100 | 0 | 0 | 0 |
| M12 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 100 | 100 | |
| M13 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 100 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 100 |
| M14 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| M15 | R | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 |
| | W | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 300 | 0 | 0 | 0 | 0 |

Point-to-point timing requirements [nSec].
'R' is for read operations, 'W' is for write operations.

available at [18]. As is seen in the tables, there is a wide variability in the requirements, at both the bandwidths and the delays. For example, M0 sends S2 492Mb per second, with a latency constraint of 5000nsec, while M4 sends S16 only 10 Mb per second, but with a much tighter delay requirement of 200nsec.

## 4. NoC Design and Cost Optimization

The design process of the NoC is composed of five phases: placing of the communicating modules (e.g. [6]-[16]); trimming and adjusting the network resources to meet the application requirements [19]; synthesizing the network; placing and routing of the NoC and extracting area and power numbers. The initial topology chosen for the NoC is a 2-dimension mesh grid that mitigates the concern of deadlocks and also simplifies the routing algorithm. In order to minimize the buffering cost and allow fast delivery of data, we use wormhole switching.

### 4.1 Cost Optimized Placement

In order to come up with the best 2D mesh topology, we explore three possible optimization goals: (1) Power-only: In this placement, only the bandwidth requirements of the application are considered, while meeting the timing requirements is left for the following stages of the design process; (2) (Power+P2P)-based placement: Here, point-to-point (P2P) latency requirements are introduced as constraints in the placement phase; (3) (Power+E2E)-based placement: instead of specifying latency requirements for each source-destination pair, the end-to-end (E2E) traversal latency requirement of the stream of information in the application is used. The difference is that with E2E latency, the particular operation may be composed of several flows of data, each traversing several nodes. For example, if data is sent from node-X to node-Y and from node-Y to node-Z, the E2E latency is measured between node-X and node-Z. A point-to-point requirement that is not a part of a larger chain is considered as an ETE traversal latency constraint. The E2E constraints are extracted from the application's characteristics and may replace some of the P2P requirements, creating a more relaxed set of constraints.

### 4.2 Placement Tool

In order to search for an optimal placement for a set of modules, a topology optimization tool that uses a simulated annealing (SA) algorithm was developed. The tool, which is capable of evaluating different MxN configurations for the 2D mesh, takes as input a spreadsheet listing connectivity and bandwidth requirements between nodes. In addition, it can read a spreadsheet with latency requirements. There are two options for listing latency: one table will list the maximum latency allowed between any two nodes on the network. The second option is to list the allowed end-to-end traversal latency requirements in the

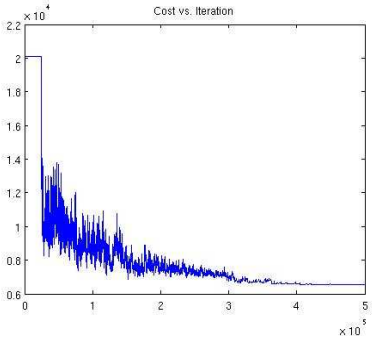


**Figure 2: Cost behavior along time**

An example of the reduction of the network cost using simulated annealing. The X-axis represents the iteration number; the Y-axis represents the cost of the network.

**Table 3**

| # | Mod#1 | Mod#2 | Mod#3 | Mod#4 | Req. |
|---|---|---|---|---|---|
| 1 | M0 | S7 | M3 | S4 | 770 |
| 2 | S10 | M10 | S11 | | 315 |
| 3 | M12 | S11 | | | 150 |
| 4 | S14 | M11 | S15 | | 215 |
| 5 | S16 | M13 | S17 | | 215 |
| 6 | S16 | M12 | S17 | | 215 |
| 7 | M2 | S4 | | | 310 |
| 8 | M2 | S7 | | | 310 |
| 9 | M4 | S13 | | | 310 |
| 10 | M5 | S13 | | | 310 |
| 11 | M6 | S13 | | | 310 |
| 12 | M7 | S13 | | | 310 |
| 13 | M0 | S2 | | | 5000 |
| 14 | S7 | M0 | | | 300 |
| 15 | S13 | M15 | | | 300 |
| 16 | M2 | S3 | | | 510 |
| 17 | M4 | S15 | | | 210 |
| 18 | M4 | S16 | | | 210 |
| 19 | M4 | S17 | | | 210 |
| 20 | M5 | S16 | | | 210 |
| 21 | M5 | S17 | | | 210 |
| 22 | M6 | S17 | | | 210 |

End-to-end traversal timing requirements [nSec].

network: rather than specify point-to-point latency between two nodes, the user can list all the nodes a particular operation must traverse and specify the total maximum latency allowed for that operation.

In order to find an optimal placement for the SoC, we also define a cost function and calculate the cost every time the SA algorithm swaps nodes. The cost function is defined as:

$$Cost = \alpha AREA_{router} + \beta \sum_{l \in links} BW_l \qquad (1)$$

where $AREA_{router}$, is an estimate for the total resources required to implement the router logic (accounting for the number of ports and their link speeds), and $BW_l$ is the bandwidth delivered over a link *l*. While $AREA_{router}$ models the area and static power used by the NoC resources, the second term is used to capture the dynamic power consumed by the communication.

The SA algorithm starts with a random placement of all the nodes on a 2D mesh and calculates the cost for this initial condition. It then proceeds to try and swap nodes in order to find a lower cost solution. The BW spreadsheet will drive the selection of a topology as this is directly included in the cost function above. However, for each solution that the SA algorithm generates, the tool will use the latency spreadsheet to check if the latency requirements are met. The latency check at this stage of the design reflects the length of the path traversed by the packet and the pipeline delay of the routers along that path. When the requirements are not met, the solution is rejected regardless of its cost. Figure 2 depicts a typical example of the cost reduction behavior along the run time of the SA optimization.

As different placements lead to NoCs with different costs, we use the SA tool to generate placements using the Power-only, Power+P2P, and Power+E2E schemes (Section 4.1), resulting in three topologies to compare and analyze. For the purpose of this paper, we used $\alpha$=10, $\beta$=1 and relative empirical weights for routers with different numbers of ports, as generated by synthesis tools. Figure 3 illustrates the generated placements using the three schemes.

### 4.3 Setting the network speed

As a significant portion of the NoC area and power consumption is due to the network links, minimizing the resources used by the links may have a considerable impact on the total cost of the NoC. Consequently, in the second phase of NoC design process, the required capacity of the network links is determined. More specifically, we attempt to find the minimal capacity of the links that would still meet all the latency constraints, as the placement tool didn't consider the dynamic contention within the network. In this paper, we consider two possible schemes: a uniform allocation, in which all links have the same capacity, and a heterogeneous allocation where different links may have different capacities.

Uniform link capacity is commonly used in wormhole networks. In such cases, the process of finding the minimal capacity that meets the latency requirements using simulations is rather simple, as a single parameter (the identical speed of all network links) is optimized. However, due to the variety of timing requirements presented by the application, this allocation causes some links to be over-provisioned. In order to reduce the cost of the links, we differentiate between two types of links: the first type of links is links that are used to route at least one flow which has timing requirement. The second type of links is those that deliver flow with no such requirements. Intuitively, it is possible to scale down links of the latter type more aggressively than those of the former type. However, it is important to note that scaling down the capacity of links that have no flows with timing requirement may hinder the delivery of flows that have latency constraints but do not traverse these links. This is due

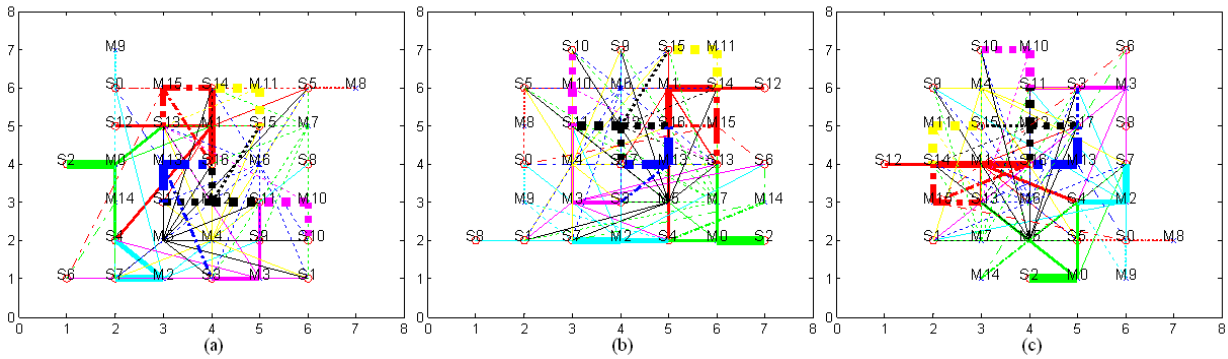


**Figure 3: Placement results**

(a) Power optimized; (b) power optimized+P2P timing constraints; (c) power optimized+E2E traversal timing constraints. Line widths represent the relative volume of traffic.

to the backpressure mechanism of wormhole switching: when a flow is slowed down in a certain router on its path, it occupies resources in other routers on its path for a longer time. Consequently, the delay of flows that share these other routers and which may have latency constraints increases. In this work we generate the custom, tuned allocation by scaling down the capacity found in the uniform assignment scheme: the capacity of links that are used only by flows with no timing requirements is re-assigned according to a selected utilization factor. The capacity of links which have at least one flow with latency constraint is reduced proportionally to the slack time of the flow with the lowest slack. Simulation is then used to verify that all latency constraints are met. In both the uniform and custom tuning schemes, links that are not used by any flow in any of the modes are completely removed.

Using an OPNET-based simulator [20] that models a detailed wormhole network (accounting for the finite router queues, backpressure mechanism, virtual channel assignment, link capacities, network contention, etc.), the basic three topologies (generated by the Power-only, Power+P2P and Power+E2E optimizations) were simulated, using one and two virtual channels (VCs). For each case, we went through the process of finding the optimal network speed for both the uniform and the tuned links capacity cases. Figure 4 illustrates the per-link capacity allocated for the placement created by the Power-only optimization. This phase results in 12 generated networks (3 basic placements * 2 VC configurations * 2 capacity schemes). At the end of this phase, all timing requirements are met (P2P constraints in the Power-only and Power+P2P placements, and E2E-traversal constraints in the Power+ETE generated placements). Figure 5 summarizes the results, presenting the total capacity required in each of the 12 configurations.

## 4.4 Synthesis

The optimization of the 1 VC network versus the 2 VC network results in different speed requirements for both the uniform and tuned cases. For some links, the two VC approach resulted in a lower link speed because of the improved link utilization offered by the additional VCs. However, the area impact of a two VC router must also be taken into account when choosing the best topology. Another factor to consider in the design of the network is the flit width we support. The current NoC architecture supports flit widths of 32, 64, and 128 bits. While the network bandwidth allocation algorithm allowed for any speed, the implementation of the NoC on the ASIC is limited to the clock frequencies and flit widths available in the design. For this reason, we bin the resulting router configurations into discrete categories supported on chip. We applied this binning strategy to all topologies and synthesized the network for each (the implementation is discussed in Appendix A). Figure 6 and Figure 7 show the results reported by the synthesis tool, separately listing the cell area and routing area. The cell area includes the area taken up by the rate matching blocks when we transition from one flit width to another in the network (Appendix A). It also accounts for the trimming of the routers, achieved by the removal of unused ports.

As expected, by applying a network capacity allocation scheme we are able to reduce the speed of the over-provisioned links thus saving area and power. The results also indicate that the Power+E2E latency approach provides a considerable better solution. To understand this, we must go back to our SA algorithm in the topology planning phase. For each case, the topology tool will use the connectivity and BW requirements to calculate the cost of the network (Eq. 1). However, in the Power-only case, the latency requirements are completely ignored. The Power-only

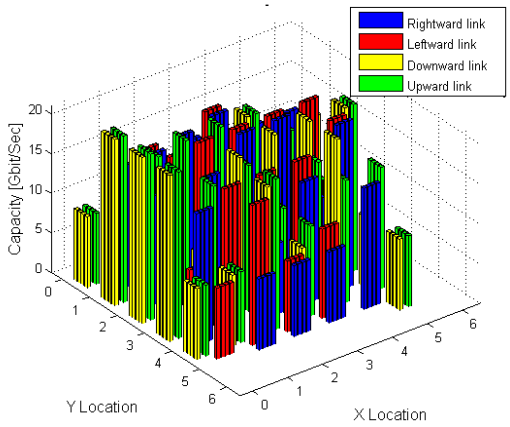


**Figure 4: An example of the tuning of link capacities**

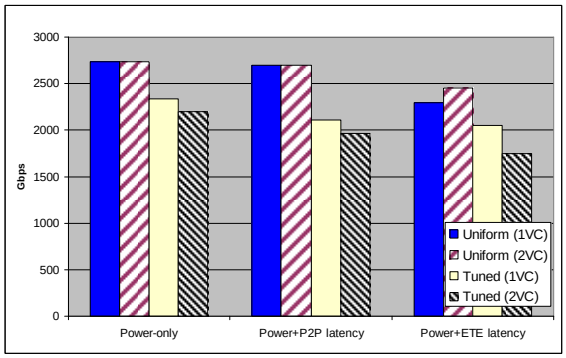


**Figure 5: Capacity requirements**

The total capacity needed to match the requirements of the examined configuration, using one and two virtual channels.

approach gives the SA algorithm the most flexibility in placing the nodes on the network. When latency is included in the topology planning, the tool will reject any solution that does not meet the latency requirements. This effectively reduces the solution space for the SA algorithm. Because of this, the Power+P2P latency is the most restrictive. In the case of the Power+E2E latency, we allow the tool some more flexibility in moving the nodes around as long as the latency is met for the full E2E traversal path.

The above explanation taken alone would imply that the Power-only case should produce the best results because the topology tool has the highest flexibility to reduce the cost. However, Figure 6 and Figure 7 show that the Power-only implementation has the largest area, in each VC/allocation scheme. To understand this, one must examine the bandwidth and latency requirements: there are some communication streams that have relatively low bandwidth but still have strict latency requirements. The nature of the topology cost function will place high bandwidth nodes close together in order to minimize the cost. When high bandwidth nodes are put close together, the other nodes get pushed further apart. As a result, some low latency nodes will be separated by many hops. This is exactly what the Power-only approach is doing. During the link capacity tuning phase, this causes a need to increase the link data rate for the latency path just to meet the requirements. The further apart these latency-critical nodes are from each other, the higher the link speed along the path will need to be. In addition, due to the nature of wormhole routing, the increased link speed will affect many other links in the network. This is why the Power-only case has a very high network capacity. In order to meet these high bandwidth requirements, we are forced to either increase the clock frequency significantly or we need to choose a larger flit width to support the high throughput. This leads to an overall larger area.

The Power-E2E latency had the most flexibility to place nodes on the topology while at the same time making sure that the latency critical signals were relatively close together. Hence, during the network capacity allocation phase, we could use a lower link speed as compared to the Power-only case. The proximity of the nodes also limited the impact of the link speed on the rest of the network. This all translates into the use of smaller flit widths and/or slower clock frequencies for the network. Consequently, the ETE-traversal approach reduces the cell area by 25%-40% and wiring resources by 13%-49% compared to the traditional power+P2P placement scheme.

When we consider the number of VCs, we see that one VC is preferable from an area perspective. In the case of 2 VCs, we can reduce some of the link capacities more than in the 1 VC case. However, the savings provided by this reduced capacity are more than offset by the increased router sizes. Therefore, the 2 VC approach does not benefit the application.

Finally, we see that the Uniform and Tuned Power+E2E topologies have the same area. The reason for this is our binning strategy. Because we are limited to certain flit widths, our link and router selection is limited to certain, discrete choices. While it is true that the tuned Power+E2E topology can theoretically run some links at a slower speed, the difference from the uniform topology is not significant enough in this case. For example, the tuned topology can reduce the speed of some links down from 15Gbps to 14Gbps. However, given the supported flit widths, this does not change the size of the link or router we can choose. Hence the two topologies come out equal.

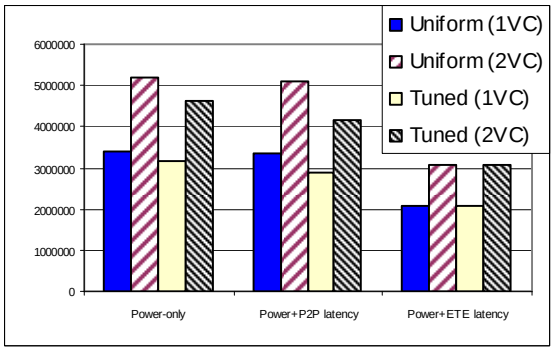


**Figure 6: Total router logic area**

The total area consumed by routers in each placement scheme.

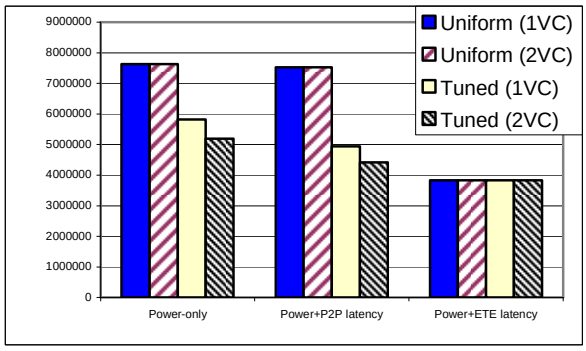


**Figure 7: Total wiring area**

The total area consumed by inter-router wires

## 5. Summary and Conclusions

The increasing communication requirements in system on-chip (SoC) implementations created a need for a new interconnection paradigm. For a few years now, academic research is suggesting network on-chip (NoC) as a means for providing efficient inter-modules communication within chips. Recently, a few companies have reported the usage of NoC in some prototypes and in commercial products.

In this paper, we describe our efforts to design a complex SoC around a NoC-based interconnect. In the first phase of the design, we explore three schemes to perform the placing of cores onto the chip: the first scheme only considers the power consumed by the transmission of packets while the second scheme uses the application's source-destination latency constraints during the optimization of the power consumption. A third placement technique replaces the pair-wise requirements with application-level end-to-end constraints, allowing more freedom in the process of seeking a solution that minimizes power consumption.

Next, we trim redundant network resources (links, ports) and tune the bandwidth of network links so that the requirements of the application are met. Finally, we synthesize the resulting networks to estimate their cost.

The main contribution of this work is the introduction of the end-to-end traversal delay constraints during the NoC design process. By replacing the source-destination requirements with end-to-end requirements wherever possible, we reduce the total area needed for the implementation of routers by 25% to 40% and the link wiring resources by 13%-49%. In addition, we demonstrated the potential benefit that lies in the implementation of links with individually assigned capacities. Future work includes placing and routing the NoC and evaluating it against a bus-based system that delivers the same performance.

## 6. References

[1] P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections", Proc. Design, Automation and Test in Europe (DATE) 2000, 250-256

[2] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal Network on Chip: Concepts, Architectures, and Implementations", IEEE Design and Test of Computers, 2005, 414-421

[3] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS Architecture and Design Process for Network on Chip", Journal of Systems Architecture, Vol. 50, February 2004, 105-128

[4] D. Bertozzi and L. Benini, "Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip", Circuits and Systems Magazine, IEEE Volume 4, Issue 2, 2004, 18-31

[5] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost Considerations in Network on Chip", Integration - the VLSI Journal, Volume 38, pp. 19-42, 2004

[6] J. Hu and R. Marculescu, "Energy-Aware Mapping for Tile-Based NoC Architectures Under Performance Constraints", Proc. Asia South Pacific design automation (ASP-DAC) 2003, pp. 233–239

[7] S. Murali and G. De Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures", Proc. Design, Automation and Test in Europe Conference (DATE), 2004, pp. 896–901

[8] C. Marcon, A. Borin, A. Susin, L. Carro, and F. Wagner, "Time and Energy Efficient Mapping of Embedded Applications onto NoCs", Proc. Asia South Pacific design automation, 2005, pp. 33–38

[9] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, and F. Hessel, "Exploring NoC Mapping Strategies: an Energy and Timing Aware

Technique", Proc. Design, Automation and Test in Europe Conference (DATE), 2005, pp. 502–507
[10] G. Ascia, V. Catania, and M. Palesi, "Multi-Objective Mapping for Mesh-Based NoC Architectures", Proc. International conference on hardware/software co-design and system synthesis (CODES ISSS), 2004, pp. 182–187
[11] S. Murali, L. Benini, and G. De Micheli, "Mapping and Physical Planning of Networks-on-Chip Architectures with Quality-of-Service Guarantees", Proc. Asia South Pacific design automation (ASP-DAC), 2005, pp. 27–32
[12] K. Srinivasan, and K.S. Chatha, "A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures", Proc. Low Power Electronics and Design 2005, pp. 387–392
[13] K. Goossens, A. Radulescu, and A. Hansson, "A Unified Approach to Constrained Mapping and Routing on Network-on-Chip Architectures", Proc. International conference on Hardware/software co-design and system synthesis (CODES ISSS), 2005, pp. 75–80
[14] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu, and E. Rijpkema, "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification", Proc. Design, Automation and Test in Europe Conference (DATE), 2005, 1182- 1187
[15] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "Mapping and Configuration Methods for Multi-Use-Case Networks on Chips", Proc. Asia South Pacific design automation, 2006, pp. 146–151
[16] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A Methodology for Mapping Multiple use-cases onto Networks on Chips", Proc. Design, Automation and Test in Europe Conference (DATE) 2006, pp. 118-123
[17] R. Gindin, I. Cidon and I. Keidar, "NoC-Based FPGA: Architecture and Routing," First International Symposium on Networks-on-Chip (NOCS), 2007, pp. 253-264
[18] Will be provided upon demand
[19] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network Delays and Link Capacities in Application-Specific Wormhole NoCs," VLSI Design, vol. 2007, Article ID 90941, 15 pages, 2007
[20] OPNET modeler (www.opnet.com)

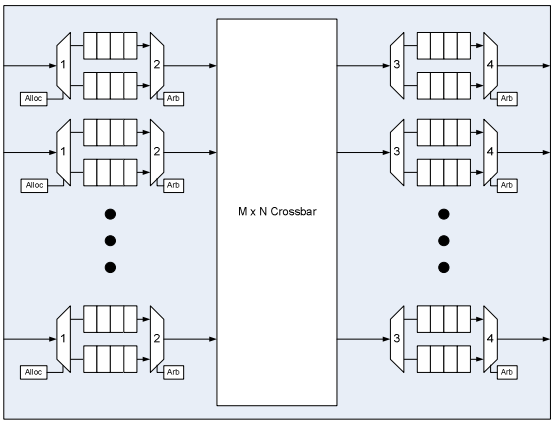


**Figure 8: Router architecture**

## Appendix A: Router and Rate Matcher Architectures

The basic router used for our analysis consists of an M x N switch, with M inputs and N outputs. A variable number of VCs can be implemented on each input and output. Figure 8 shows an implementation with two VCs per port.

Each input or output port arrow actually represents multiple signals. Packet flits are routed with two control bits, allowing flow control at each stage in the router. The VALID flag indicates that the word is valid and the WAIT flag indicates whether the receiver will accept the flit. When multiple VCs are present, there is one VALID/WAIT pair per VC.

The full data path for an Input VC consists of a demultiplexor ("1" in Figure 8), a VC allocator ("Alloc"), some number of buffer/FIFO stages (rectangles between "1" and "2"), a multiplexor ("2"), and a VC arbiter ("Arb"). The Input VC blocks are shown in Figure 9.

The demultiplexor ("1") diverts incoming packets to a particular input VC queue based upon an assignment made by the allocator block. The buffer stages allow separate packets to queue in parallel, thus allowing one packet to bypass another blocked packet.

The allocation block maintains a record of each incoming packet to determine if it was previously assigned a VC queue or it is a new packet that needs assignment. Each flit that comes into an input port comes from a VC in the sender. There is a VALID for each of the sender output VC. Let us call this VC-sender. When a header flit is received at an input port, the VC allocator must map the VC-sender to a VC queue that is free. To do this, the allocator maintains a list of which queues are free. It will read an entry from the "free list" and store the mapping from VC-sender to

internal VC queue. From that point on, any flits that come in on VC-sender are mapped to VC queue. When the last flit of the packet is sent out of the VC queue, the allocator will clear the mapping and make it available in the "free list". Finally, the arbiter decides which flit accesses the crossbar switch based upon the queue status at the output of the router, and controls mux "2" appropriately.

Packet flits sitting in the VC queue must be routed to the appropriate output port. This is accomplished with two levels of arbitration to decide which flits pass from input queues to output queues. The first arbitration occurs on each input port individually, to determine which input VC queue passes a flit. The arbiter must tell the multiplexor ("2") which input VC queue will access the switch each clock cycle. The decision is a function of which of the input queues are not empty and whether the destination switch port (assigned output VC queue) can accept another flit (one of the WAIT_OPORTi_VCi bits). Without this first level of arbitration, the switch would grow to be 2M x 2N, assuming two VCs on each input and output.

The second level of arbitration is done in the crossbar switch. It arbitrates between multiple inputs accessing the same output port (despite possibly different output VCs). The mux should ensure that if a destination output queue is not ready for the next flit it will not block flits from other input VCs going to other output queues.

The data path structure of the Output VCs is very similar to the Input VCs. It consists of a demultiplexor ("3"), buffers (rectangles), multiplexor ("4"), and arbiter ("Arb"). Each component operates similar to the corresponding component for input VCs.

In the case of our tuned networks, a set of *rate matching* blocks that will allow the transition from one packet width to another are needed. Two designs for the rate matching blocks are shown below.

The design in Figure 10(a) converts a 128-bit packet width to a 32-bit packet width. The incoming packet is first stored in a queue. The control logic will then read the queue in groups of 32-bits and multiplex them onto the output. Because the output rate is much slower then the input rate, the control logic must also manage the WAIT signals back to the sender. This will throttle the rate of the incoming packets. We provide a mechanism to bypass the queue and send 32-bits straight to the output. This is done to minimize the latency of the rate matching block. In this case, the control logic will select the upper 32-bits of the incoming packets and mux them directly on the output.

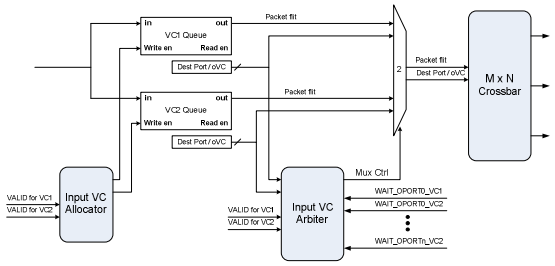


**Figure 9: Router Input VCs**

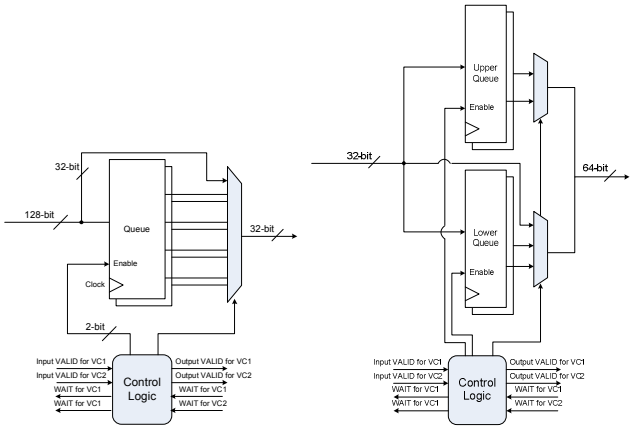


**Figure 10: Rate matching blocks**
(a) 128-bit to 32-bit; and (b) 32-bit to 64-bit

Conversely when going from a low rate to high rate we use a design as shown in Figure 10b. In this case we store the first valid word of the packet in the upper set of holding registers and wait for the second valid word to come in. When the second word appears it is directed to the lower holding registers. When both upper and lower queues have data, the control logic will combine these onto one output bus and set the appropriate enable high. Here too we provide a mechanism to reduce the latency through the rate matching block. This is done by providing the input directly to the lower mux. Similar to the 128-to-32 rate matching blocks, the control logic must take into account the WAIT signals coming from the downstream receiver and must appropriately throttle the incoming packets.

It is worth noting that the number of queues must match the number of VCs. In our examples above, there are 2 sets of queues, one for each VC. The reason for this is that the rate matching block should not block any packet from moving forward if the VC it is on is open downstream.