# Route Packets, Not Wires: On-Chip Interconnection Networks

William J. Dally and Brian Towles
Computer Systems Laboratory
Stanford University
Stanford, CA 94305
{billd,btowles}@cva.stanford.edu

## Abstract

Using on-chip interconnection networks in place of ad-hoc global wiring structures the top level wires on a chip and facilitates modular design. With this approach, system modules (processors, memories, peripherals, etc...) communicate by sending packets to one another over the network. The structured network wiring gives well-controlled electrical parameters that eliminate timing iterations and enable the use of high-performance circuits to reduce latency and increase bandwidth. The area overhead required to implement an on-chip network is modest, we estimate 6.6%. This paper introduces the concept of on-chip networks, sketches a simple network, and discusses some challenges in the architecture and design of these networks.

## 1 Introduction

We propose replacing design-specific global on-chip wiring with a general-purpose on-chip interconnection network. As shown in Figure 1, a chip employing an on-chip network is composed of a number of network clients: processors, DSPs, memories, peripheral controllers, gateways to networks on other chips, and custom logic. Instead of connecting these top-level modules by routing dedicated wires, they are connected to a network that routes packets between them. Each client is placed in a rectangular tile on the chip and communicates with all other clients, not just its neighbors, via the network. The network logic occupies a small amount of area (we estimate 6.6%) in each tile and makes use of a portion of the upper two wiring layers.

Using a network to replace global wiring has advantages of structure, performance, and modularity. The on-chip network structures the global wires so that their electrical properties are optimized and well controlled. These controlled electrical parameters, in particular low and predictable cross-talk, enable the use of aggressive signaling circuits that can reduce power dissipation by a factor of ten and increase propagation velocity by three times [3]. Sharing the wiring resources between many communication flows makes more efficient use of the wires: when one client is idle, other clients continue to make use of the network resources.

An on-chip interconnection network facilitates modularity by defining a standard interface in much the same manner as a backplane bus. For the past three decades systems have been constructed by plugging modules into standard backplane buses such as VME or PCI. The definition of a standard interface facilitates reusability and interoperability of the modules. Also, standard interfaces allow shared interconnect to be highly optimized since its development cost can be amortized across many systems.

Of course, these modularity advantages are also realized by on-chip buses [1][5][8], a degenerate form of a network. Networks are generally preferable to such buses because they have higher bandwidth and support multiple concurrent communications. Some of our motivation for intra-chip networks stems from the use of inter-chip networks to provide general system-level interconnect [7].

The remainder of this paper describes our initial thoughts on the design of on-chip interconnection networks. To provide a baseline, we start in Section 2 by sketching the design of a simple on-chip network. Section 3 revisits the design choices made in this simple network and discusses the challenges and open research issues in the design of such networks. Section 4 discusses the advantages and disadvantages of on-chip networks.

## 2 Example on-chip interconnection network

To give a flavor for on-chip interconnection networks this section sketches the design of a simple network. Consider a 12mm x 12mm chip in 0.1µm CMOS technology with an 0.5µm minimum wire pitch. As shown in Figure 1, we divide this chip into 16 3mm x 3mm tiles. A system is composed by placing client logic (e.g., processors, DSPs, peripheral controllers, memory subsystems, etc.) into the tiles. The client logic blocks communicate with one another only over the network. There are no top-level connections other than the network wires.

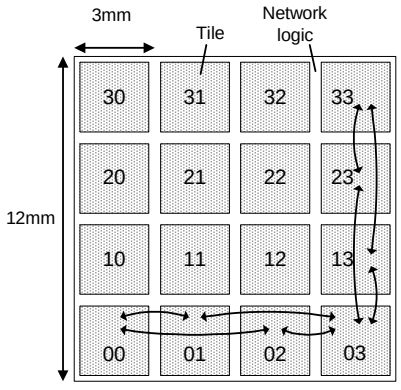


**Figure 1 Partitioning the die into module tiles and network logic**


*DAC 2001*, June 18-22, 2001, Las Vegas, Nevada, USA.

The network logic occupies a small amount of area between the tiles and consumes a portion of the top two metal layers for network interconnect. This baseline network uses a 2-dimensional folded torus topology with the nodes 0-3 in each row cyclically connected in the order 0,2,3,1. I/O pads may connect directly to adjacent tiles or may be addressed as special clients of the network.

## 2.1 The network presents a simple reliable datagram interface to each tile

Each tile interfaces with the network over an input port that is used to insert packets into the network and an output port over which the network delivers packets to the tile[1]. The input port consists of a 256-bit data field and a control field with the following subfields:

- *Type* (2 bits): encodes whether the flit (flow control digit) on the data port is the start of a new packet (head), the continuation of a packet (body), the end of a packet (tail), or an idle cycle (idle). Note that a flit may be both a head and a tail. A multi-flit packet is inserted by inserting a head flit, zero or more body flits, and a tail flit.
- *Size* (4 bits): logarithmically encodes the size of the data in the data field from 0 (1 bit) to 8 (256 bits). When a short data field is sent the size field prevents the unused bits from dissipating power.
- *Virtual Channel* (8 bits): bit mask that specifies which of eight virtual channels this packet may be routed on. The virtual channel mask identifies a class of service. Packets from different classes may be in progress simultaneously. Thus, the injection of a long, low priority packet may be interrupted to inject a short, high-priority packet and then resumed.
- *Route* (16 bits): A source route that specifies two bits for each hop (left, right, straight, or extract). The destination of the route may be one of the sixteen tiles or special network clients including I/O pads and internal network registers. This field is only used on a head flit and can be used to carry data on non-head flits.
- *Ready* (8 bits): a signal from the network back to the client indicating that the network is ready to accept the next flit on each virtual channel.

The output port consists of a 256-bit data field and a control field consisting of type, size, virtual channel, and ready signals with meanings identical to those on the input port. On the output port the ready signal is from the client to the network.

In addition to this simple port interface, the network also presents a number of registers that can be used to reserve resources for particular virtual channels. For example, one use of these registers is to provide time-slot reservations for certain classes of traffic, such as data streams and logical wires. This pre-scheduling is required to provide guaranteed, predictable performance for latency-critical or jitter-critical applications. The details of these registers are beyond the scope of this paper.

1. Input and output here refer to packets going in and out of the network, not the tile.

## 2.2 Higher level protocols can be layered on top of the simple interface.

This port interface presents a simple, low-level reliable datagram service to the network clients. Logic local to the network clients can layer higher level services on top of this interface. For example, this local logic could present a memory read/write service, a flow-controlled data stream, or a logical wire to the client. Local logic can also provide a translation from a destination node to a route.

As an example of layering, we will examine how a logical wire is layered on top of the interface described above. Suppose tile *i* has a bundle of *N*=8 wires that should be logically connected to tile *j*. The local logic monitors these wires for changes in their state. Whenever the state changes, the logic arbitrates for access to the network input port, possibly interrupting a lower priority packet injection, and injects a single flit packet with data size 16, an appropriate virtual channel mask, and destination of tile *j*. Eight of the 16 data bits hold the state of the lines while the remaining data bits identify this flit as a containing logical wires. Upon arrival at tile *j*, the flit is decoded and the output of the logical wires are updated with the new state. As discussed below, the latency of transporting the state of wires in this manner can be made competitive with dedicated wires. We expect the logic to implement many higher-level services on top of the simple network will be made readily available so it won't have to be independently redesigned with each module.

## 2.3 Router architecture

The router needed at each tile consists of five input controllers (one for each direction and one for input from the tile) and five output controllers. The router for each tile is distributed across the four edges of a tile with the input and output controller for a given direction (e.g., west) located on that edge of the tile. Figure 2 shows the west input controller and its connection to four output controllers (one for each of the other directions and one for the tile). By convention, the tile input and output controllers are located on the west edge of the tile. Controllers handling traffic in opposite directions have their MSBs at opposite ends to keep all wires that turn a corner approximately the same length, 3mm.

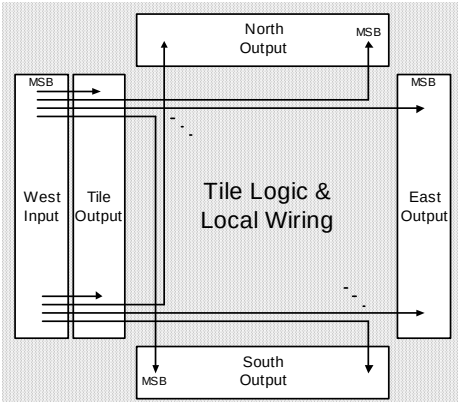


**Figure 2 West input connections to output controllers**

The simple network employs virtual-channel flow control [2][6]. Details of an input controller and output controller are shown in Figure 3. Each input controller has an input buffer and input state logic for each virtual channel. When a head flit arrives at each tile, the input controller strips the next entry off the route field and uses these two bits to select one of four output ports. The flit then arbitrates with the other virtual channels on that input port and, if it wins the arbitration, is forwarded to the output port. This arbitration and forwarding takes place in parallel with allocating a virtual channel and checking available buffer space to reduce latency. The output port provides a single stage of buffering for each input port connection. The flits in these buffers arbitrate for the link to the input controller on the next tile. Credits for buffer allocation are piggybacked on flits travelling in the reverse direction.

In our example network, the wires are driven at the same frequency as the input and output controllers. This choice allows for the simplest possible controller logic, at the expensive of more global wiring resources. An alternative architecture could easily drive the network wires at several times the router frequency, taking advantage of their available bandwidth. This approach would reduce global wiring usage with the addition of slightly more controller logic.

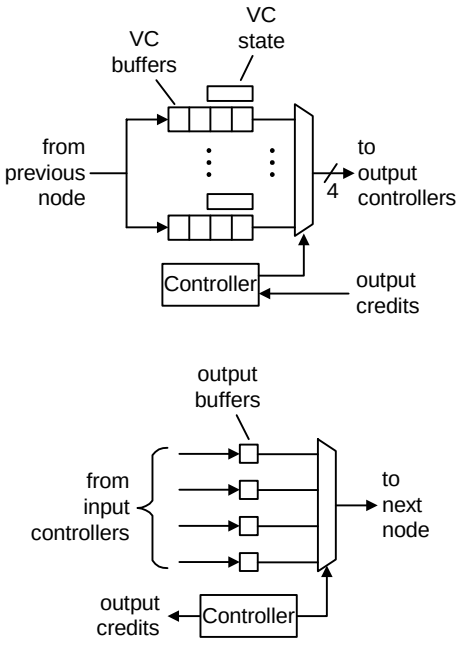


**Figure 3 Input (top) and output controllers**

## 2.4 The router uses only a small fraction (6.6%) of the tile area and some upper-level wires.

The logic of the virtual channel router is very simple, a few thousand gates along each edge of the tile. Hence the area of the router is dominated by buffer space. If we provide four flits of buffering for each of the eight virtual channels with about 300b per flit (with overhead), the total buffer requirement is about $10^4$ bits along each edge of the tile. We estimate the logic, driver and receiver circuits, buffer storage, and routing will occupy an area less than 50µm wide by 3mm long along each edge of the tile for a total overhead of 0.59mm$^2$ or 6.6% of the tile area. In addition to this area, the router also uses about 3000 of the 6000 available wiring tracks on the top two metal layers for routing differential signals and shields[2]. This 'overhead' replaces the drivers, receivers, repeaters, and wires for global signals. Thus, the net effect on a design will be substantially less than this and may even be positive.

## 2.5 Networks enable the use of fault-tolerant wiring and protocols

To prevent a single fault in a network wire or buffer from killing the chip, a spare bit can be provided on each network link and in each network buffer[3]. After test, laser fuses are blown (or registers are set at boot time) to identify any faulty bits. Bit steering logic then shifts all bits starting at this location up one position to route around the faulty bit. Similar logic at the endpoint of the link restores the original positioning of the bits. Although not employed in our design, the use of link-level error correction reduces the possibility of a transient fault, with the cost of additional delay. Alternatively, modules that required transient fault tolerance could employ end-to-end checking with retry by layering the checking protocol on top of the network interfaces.

## 2.6 The network handles both pre-scheduled and dynamic traffic

On-chip networks must efficiently support both pre-scheduled and dynamic traffic. For many applications, large data flows are relatively static and demand high-bandwidth with low latency and low jitter. For example, a flow of video data from a camera input to an MPEG encoder is entirely static and requires high-bandwidth with predictable delay. Such static traffic must share the network with dynamic traffic, such as processor memory references, that cannot be predicted before run-time and hence cannot be statically scheduled. In contrast, most inter-chip networks are used in applications such as processor-memory interconnect where only dynamic traffic need be accommodated.

Static and dynamic traffic can be handled together using virtual channels and a cyclic reservation registers. When the system is configured, routes are laid out for all of the static traffic and reservations are made for each link of each route by setting entries in the appropriate reservation register. At run time, a pre-scheduled packet is sent on a special virtual channel. At each hop, the packet moves from one link to another without arbitration or delay using the pre-scheduled reservations. Dynamic traffic arbitrates for the cycles on each link that are not pre-reserved.

---

2. Depending on process parameters, one or more levels of repeaters may be required along the 3mm length of each of these wires for optimum signal velocity. These repeaters add a small amount to the overhead.
3. If yield analysis indicates that more than one spare bit is required, multiple spare bits can be provided using the same method.

## 3 Challenges in architecture and design

While the same principles apply to interconnection networks at all scales, on-chip networks have a number of characteristics that make their design quite different than the inter-chip (and inter-board) networks that have been designed for years. In particular, wires and pins are more abundant than in inter-chip networks and buffers space is less abundant. These differences enable a number of new network topologies, flow control methods, and other techniques. In particular, we identify three areas that are ripe for future research:

### 3.1 What topologies are best matched to the abundant wiring resources available on chip?

On chip networks have enormous wiring resources at their disposal. In the example network described above, there can be up to 6,000 wires on each metal layer crossing each edge of a tile. It is quite easy to achieve over 24,000 'pins' crossing the four edges of a tile. In contrast, inter-chip networks have historically been pin limited, required to limit the connections of one router chip to far less than 1,000 total pins. This large, 24:1, difference between router pin limitations allows the designer to trade wiring resources for network performance, making a qualitative difference in network architecture.

The simple network described above uses two methods to exploit the large available pin count. First, a wide (almost 300-bit) flit is sent broadside across router channels to use the maximum possible number of pins. In contrast, most inter-chip routers use relatively narrow channels (8-16 bits) to fit into the tight pin constraint of a chip package.

Second, a folded torus topology is employed. This topology has twice the wire demand and twice the bisection bandwidth of a mesh network. Hence, it effectively converts some of the plentiful wires into bandwidth. In general, the folded torus topology trades a longer average flit transmission distance for fewer routing hops.

While these choices of wide, broadside flits and a folded topology increase the wire utilization, much room for improvement remains. There are many alternative topologies and the choice of a topology depends on many factors.

For example, if power dissipation is critical, a mesh topology may be preferable to a torus. Simple expressions illustrate the trade-off of power between the folded torus and a traditional mesh, where $k$ is the radix of the network ($k = 4$ in our 16 tile example):

$$P_{total} = hops \cdot P_{hop} + dist \cdot P_{wire}$$

$$P_{mesh} \approx k \cdot \left(\frac{2}{3} \cdot P_{hop} + \frac{2}{3} \cdot P_{wire}\right)$$

$$P_{torus} \approx k \cdot \left(\frac{1}{2} \cdot P_{hop} + \frac{k-1}{k} P_{wire}\right)$$

As shown, the total power required to send a flit through the network can be decomposed into the power per hop (traversal of input and output controllers) and power per wire distance traveled. By substituting approximate equations for the average number of hops and average transmission distance, average power per flit is expressed for the mesh and torus. From these expressions, if wire transmission power dominates per hop power, the mesh is more power efficient.

Our estimates do show that wire transmission power is significantly greater than per hop power for our 16 tile network. However, in our example, the power overhead of the torus is small, less than 15%, and is outweighed by the benefit of the larger effective bandwidth of the torus.

### 3.2 What flow control methods reduce buffer count and hence router overhead?

Buffer space in an on-chip router directly impacts the area overhead of the network and thus must be kept to a minimum. In contrast, most inter-chip network routers are pin limited and thus have ample room for very large buffers.

Our example network uses conventional virtual channel flow control and uses a large amount of buffer space: 10K bits in each input controller. Alternative flow control methods can substantially reduce the buffer storage requirements at the expense of reduced performance. For example, if packets are dropped or misrouted when they encounter contention very little buffering is required. However, dropping and misrouting protocols reduce performance and increase wire loading and hence power dissipation. Research is required to invent and explore flow-control methods that strike the right balance between buffer requirements, performance, control complexity, and power.

### 3.3 What circuits best exploit the structured wiring of on-chip networks?

Much of the advantage of on-chip networks derives from the regular, structured nature of their wiring. As described below, the well controlled electrical parameters of this wiring enable the use of high-performance circuits such as pulsed low-swing drivers and receivers to reduce power dissipation, reduce latency, and increase repeater spacing. While these transceivers yield big performance dividends, they only scratch the surface of what is possible. Additional circuit innovations can improve the performance of on-chip networks substantially.

Circuits can be used to boost the bandwidth of individual wires by sending several bits each clock cycle. In 0.1μm technology, it is feasible to transmit 4Gb/s per wire. This translates to 2-20 bits per clock cycle depending on whether the chip uses an aggressive (2GHz) or slow (200MHz) clock. Circuits here are needed both to drive and receive the line at these high rates, to provide timing for the link, and to interface the high-frequency timing domain of the link to the low-frequency timing domains of the tiles. For generality it is necessary to support tiles operating at different frequencies.

Circuits can also ease the area overhead of buffering by integrating buffer storage into the drivers, receivers, and repeaters. Such integrated buffers directly replace area-consuming buffers in the input and output controllers. They also have the potential of reducing the overall need for buffers by closing flow control loops locally so credits can be quickly recycled [4].

## 4 Discussion

The on-chip network sketched above can provide very general communication services for its clients. Compared to dedicated wires, however, it incurs some overhead. Area (6.6%) is expended for the routers and additional data is transported in the packet control fields. Why then should one use an on-chip net-work and

incur these overheads rather than use dedicated wiring? There are several compelling reasons to use these networks that more than justify this small overhead.

## 4.1 Predictable electrical parameters enable high-performance circuits

While dedicated, per-design global wiring provides ultimate flexibility, the practical problems associated with electrically characterizing this wiring and it's potential late-stage impact make the cost associated with this flexibility far outweigh any benefits. Top-level interconnect on conventional integrated circuits consists of a set of dedicated global wires that span large distances across the chip. These wires are typically laid out by an auto-router late in the design. The continued viability of this methodology is challenged by several difficult electrical problems.

First, unstructured wires have parasitic capacitance and crosstalk to adjacent wires, which is difficult to predict early in the design process and may differ significantly from one run of the router to the next. Because their electrical properties are poorly characterized, very conservative circuits must be used to drive and receive these wires. Typically, full-swing static CMOS gates (or inverters) are employed to achieve good noise immunity at the expense of increased delay and high power dissipation. Compounding this inefficiency, synthesis tools size drivers according to a statistical wire model that oversizes most of the drivers but undersizes enough of the drivers to make timing closure a difficult problem. Second, long wires require repeaters at periodic intervals to keep their delay linear (rather than quadratic) with length. Properly placing these repeaters is difficult and places additional constraints the auto-router. Finally, all of these problems only become more complex with each successive technology scaling.

In contrast, an on-chip interconnection network structures the top-level wiring of an integrated circuit. The paths between tiles are precisely defined at the beginning of the design process. Moreover, these paths can be optimized for their signal integrity characteristics. In particular, the structured wiring can be isolated from intra-tile wiring to make both noise and delay variation due to crosstalk negligible. The result is that the L, R, and C parameters of all top-level wires can be controlled to provide low-noise and excellent signal propagation properties. Also, knowing these parameters at the beginning of the design process allows architects to account for global wiring delays, minimizing late-stage design iterations due to timing "surprises".

Predictable wires with low noise also enable the use of aggressive circuit techniques to reduce the power and improve the performance of global interconnects. For example, pulsed low-swing drivers and receivers [3] (Chapter 8) can be used to drive each of the 3mm wires across tiles (input to output controller) and between tiles (output to input controller). These low-swing circuits provide three significant advantages. First, by using 100mV or less of signal swing, they reduce power by an order of magnitude compared to 1.0V full swing signaling in our 0.1µm process. Second, by overdriving the transmit end of the wire, they produce about three times the signal velocity of a full-swing driver. This greatly increased signal velocity compensates for the overhead of the routing logic. In fact, with efficient pre-scheduled flow control, the latency of a signal transported over an on-chip network could be lower than a signal transported over a dedicated full-swing wire with optimum repeatering. Finally, the overdrive also increases the optimum repeater spacing by about 3x. This simplifies layout and reduces the area overhead of global signals. For many 0.1µm processes, this will make it possible to traverse a 3mm tile without the need for an intermediate repeater. As described above, we also expect that the use of aggressive circuits will enable higher bandwidth to be achieved per wire and will reduce the area burden of buffer storage.

## 4.2 Facilitating reuse with a universal interface

By defining a standard interface, an on-chip network facilitates the development and employment of reusable system components. All components, such as microprocessors or peripheral controllers, which are designed to be compatible with a single network interface, are interoperable. If these modules were instead directly connected over a non-standard interface, a different peripheral controller would need to be designed for each such interface. The interoperability advantage of standard hardware interfaces has been used for many years in board-level systems where standards like S-100, VME, SCSI, and PCI have allowed single modules to be reusable across a wide variety of systems. Similar ideas are beginning to gain acceptance with the recent introduction of standard on-chip bus interfaces [1][5][8]. A standard network offers the same features, but with greater scalability and performance than a bus.

A potential drawback of using a standard interface is the inefficiency introduced if clients are transmitting flits with small data payloads compared to the size of the flit. For example, our network has 256-bit wide flits, but it is reasonable to assume not all client transfers will be this wide. A simple solution is to partition the width of the interface into several separate physical networks. Each partition of the interface will require its own control signals, so some additional signal overhead is introduced over the original, wider interface. So, for example, we could split our 256-bit flit into eight, 32-bit flits and duplicate the control signals eight times. Wide flits could still be transferred by using several of the 32-bit interfaces in parallel, but smaller flits would now only use a fraction of the total interface bandwidth.

## 4.3 Extending reuse to the network

In addition to simplifying the reuse of design modules, the on-chip network itself, including the routers and top-level metal layout, is a reusable component. Because the network can be reused across a wide range of designs, the designer can afford to dedicate more resources to its design, validation, and tuning than is practical to apply to the top-level wiring of any single design. This additional optimization combined with the regular structure of the wiring further allows the designer to extract more bandwidth and lower latencies with lower power than is possible with dedicated wiring.

However, fixing the size of a tile can potentially waste die area if client modules only occupy a fraction of their tile's area. Unless the design is pin-limited, unused die area would result in a larger die, increasing per-chip cost and reducing gross margin. This increase in chip area affects the number die per wafer, but does not impact yield since empty silicon is not vulnerable to defects.

For a low-volume part, or even the first spin of a high-volume part, design time is almost always more important than chip cost.

In this case, if the network reduces design time by eliminating the need to do dedicated top-level wiring, then the reduction in die per wafer is acceptable. For a high-volume part, die area can be reduced by compacting the tiles. An optimal compaction may require moving client modules so that all of the big (small) clients are in the same row or column.

### 4.4 On-chip networks improve the duty factor of wires

Often key portions of chips are limited by wire density. Yet, the average wire on a typical chip is used (toggles) less than 10% of the time. Thus, this critical resource is being underutilized. Each dedicated wire or bus is needed to operate a full speed some of the time, but most of the time they may be idle.

A network solves this problem by sharing the wires across many signals. This averages out the demand and results in a much higher duty factor for this scarce resource. Pre-scheduling of critical communications and priority scheduling of other traffic prevents this resource sharing from increasing the latency of the most critical signals. The use of aggressive circuit design allows us to operate on-chip networks with very high duty factors - over 100% if we transmit several bits per cycle - without collapsing the power supply. If full-swing global wires were operated at this rate, careful attention would be required to prevent the resulting large current draw from collapsing the power supply voltage [3] (Chapter 5).

## 5 Conclusion

Using an on-chip interconnection network to replace top-level wiring has advantages of structure, performance, and modularity. A network structures the top-level wires simplifying their layout and giving them well-controlled electrical parameters. These well controlled electrical parameters in turn 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. In many applications the improvements in performance due to these aggressive circuits more than offset the performance lost to network overhead. Bandwidth is also enhanced by sharing network channels across all clients. In contrast, dedicated wiring often leaves many wiring resources idle at times of peak load. Also, an on-chip network, like popular system-level buses, enhances modularity by providing a standard interface.

While on-chip networks leverage many years of research in inter-chip networks, many challenges remain in adapting these networks to the unique opportunities and constraints of intra-chip communication. Topologies must balance power efficiency with wire utilization. New flow control methods are required to reduce the buffer sizes, and hence the area overhead of these networks while maintaining high wire- and power-efficiency. These flow control methods must also seamlessly handle a combination of statically scheduled and dynamic traffic. The network interface must be simple yet able to support a wide variety of higher-level protocols and a wide variety of data widths with low overhead.

## References


[1] "The CoreConnect™ Bus Architecture" IBM, 1999, http://www.chips.ibm.com/products/coreconnect/docs/crcon_wp.pdf.

[2] DALLY, WILLIAM J., "Virtual Channel Flow Control." *IEEE Transactions on Parallel and Distributed Systems*, March 1992, pp. 194-205.

[3] DALLY, WILLIAM J., AND POULTON, JOHN W., *Digital Systems Engineering*, Cambridge University Press, 1998.

[4] MIZUNO, MASAYUKI AND DALLY, WILLIAM J., "Elastic Interconnects: Repeater-Inserted Long Wiring Capable of Compressing and Decompressing Data," *2001 ISSCC*, February, 2001, pp. 346-347.

[5] "Open Core Protocol™ Data Sheet" Sonics, Inc., http://www.sonicsinc.com/Documents/OpenCoreProtocol_DS.pdf.

[6] PEH, LI-SHIUAN AND DALLY, WILLIAM J., "A Delay Model and Speculative Architecture for Pipelined Routers." *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, Jan. 2001, pp. 255-266.

[7] SEITZ, CHARLES, "Let's Route Packets Instead of Wires." *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, 1990, pp. 133-138.

[8] VSI Alliance, http://www.vsi.org.