# POWER AND AREA BENEFITS OF MODULAR INTERCONNECT DESIGN

Jonah Probell

Arteris, Inc. Sunnyvale, CA, USA
jonah@arteris.com

**ABSTRACT**

The SoC interconnect spans the entire floorplan of the chip and consumes a significant portion of the power. The interconnects of today's SoCs are a distributed architecture of switches, buffers, firewalls, register slices, and clock and power domain crossings. One approach is to implement these units modularly with a simple, universal transport protocol between all units. This approach enables unit level clock gating, eliminating clock tree switching power when no traffic is present. Modularity localizes logic, which minimizes long wires and further limits power consumption by keeping capacitance low. The simplicity of the protocol also allows each function to be performed with minimal logic overhead, minimizing area and leakage power consumption. This design approach is worth consideration for power sensitive SoCs.

## INTRODUCTION

Low power is among the greatest considerations of chip designs today. The top level interconnect fabric is particularly important because its long wires consume disproportionate power for the amount of logic. The clock tree is the greatest power sink, and clock gating is the solution of greatest benefit. After that, leakage power is most significant and is proportional to the logic area of the fabric.

This whitepaper discusses the power and area benefits of designing a modular network-on-chip. These benefits results from greater localization of clock tree management, data path serialization, and fine grained pipe stage insertion. The following contains a history of interconnect fabrics, the philosophy of modular network-on-chip (NoC) design, and experimental data.

## INTERCONNECT TRENDS

A system-on-chip (SoC) is a chip with a CPU and peripherals. Almost as soon as CPUs and peripherals were put together within chips interface protocol standards were developed. With the advent of additional bus masters, connections to the peripherals were shared. Like board-level protocols, a central arbiter with a request-grant handshake was used to control access to the bus.

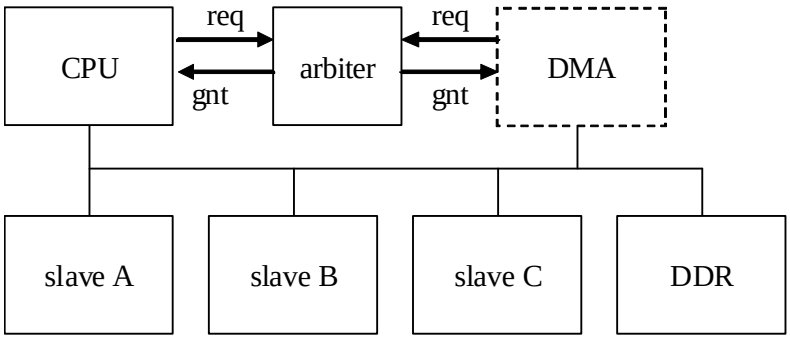


Figure 1: A shared bus with an arbiter

As SoCs became more complex they included more IP cores, each with one or more bus interfaces. Under busy operating conditions a significant fraction of masters' clock cycles were spent waiting for access to the bus, even when different masters were requesting transactions to different slaves. Crossbar switches were used to allow concurrent accesses between masters and different slaves within on-chip interconnects. Figure 2 is a logical diagram showing four masters performing four simultaneous transactions to four different slaves.

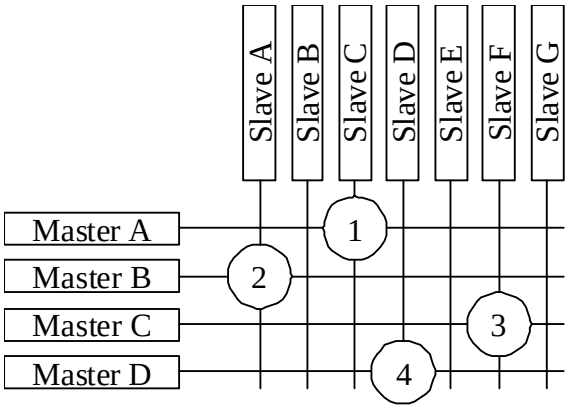


Figure 2: The logical view of a crossbar switch

Physically, a crossbar switch is implemented with a mux at each slave. Each mux is coupled with an arbiter in a distributed arbitration scheme.

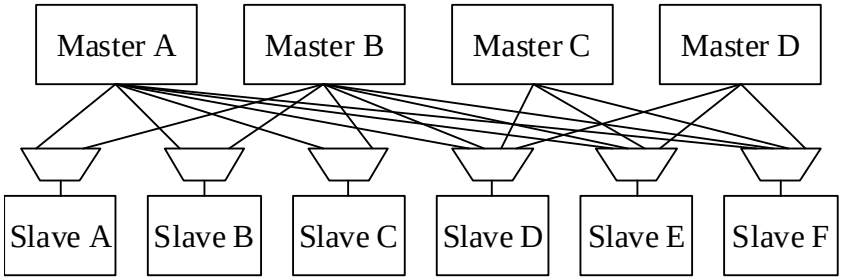


Figure 3: The implementation of a 4 master, 6 slave crossbar

This approach scales up to several master and slave interfaces. Beyond that the size of routing of full data paths around the SoC becomes impractical for place and route.

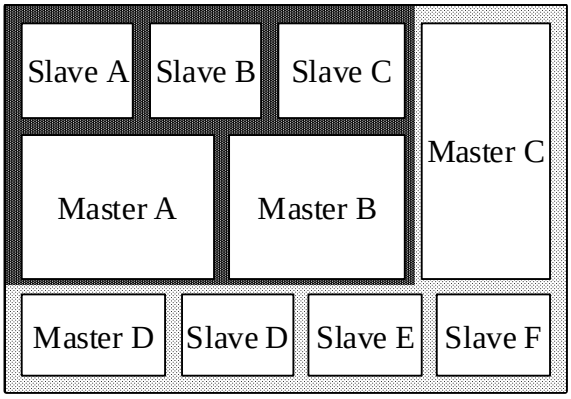


Figure 4: An SoC floor plan

For chips with more than several master/slave interfaces it is necessary to design the interconnect regionally, according to the physical placement of groups of IP cores. Bridges are designed between regions to give necessary connectivity between masters.

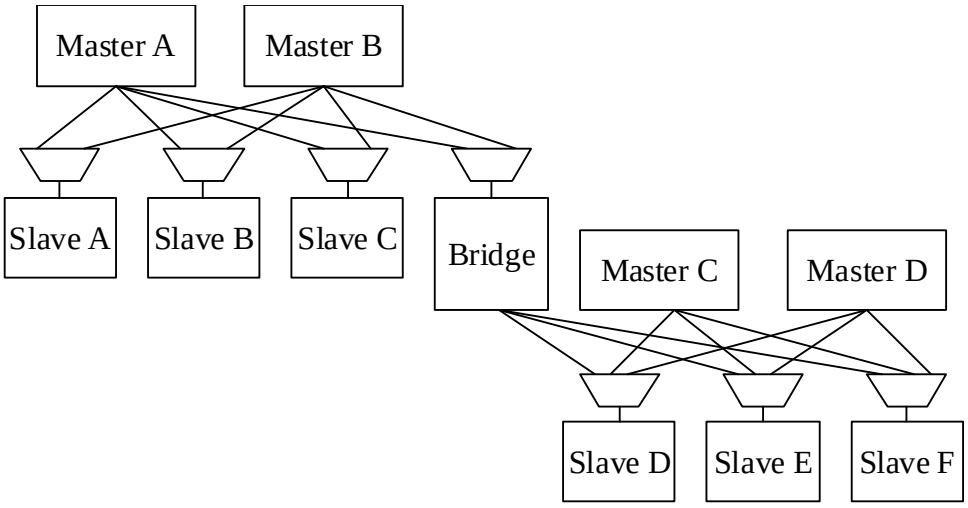


Figure 5: Interconnect of 4 masters by 6 slaves with a bridge

Bridges carry a die size overhead and add cycles of latency to data transactions through the interconnect. Since there is only one route that transactions can travel between each initiator and every connected slave, addresses can be decoded at the master interface and converted to a simple route ID. The route ID can be used in an on-chip network of muxes and demuxes. Spreading the distribution of routing by linking simple pseudo-switch muxes around the chip allows more area efficient placement of the interconnect logic. This gives easier place and route, which is increasingly valuable for the growing numbers of wires in chips.

Crossbar interconnects fix a system architecture problem of concurrent access but create a physical implementation problem in high-convergence chips. NoC solves both problems. Because of its physical advantages, the use of Network-on-Chip (NoC) has become the standard methodology for advanced chips.

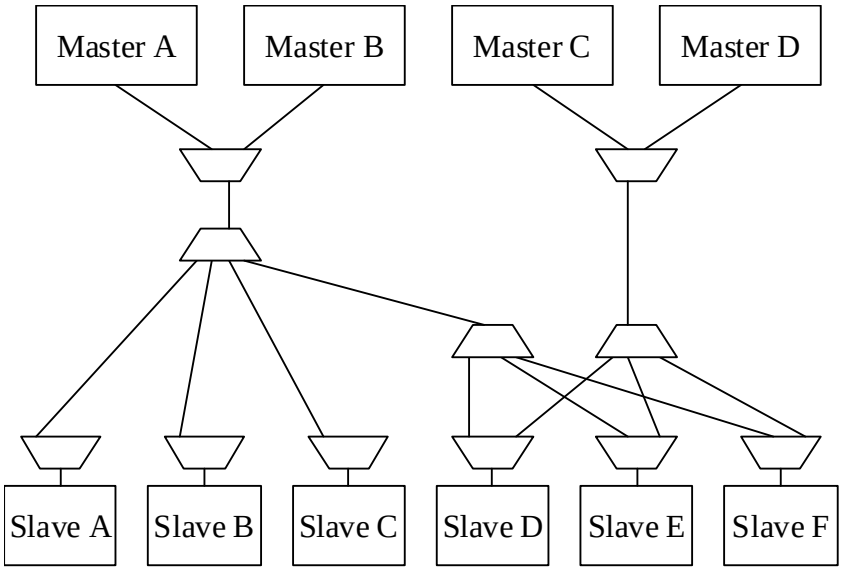


Figure 6: Interconnect of 4 masters by 6 slaves with a NoC

## MODULAR DESIGN

As SoCs have grown more feature rich so have the requirements demanded of the interconnect. Today's interconnects include:

- Interfaces to different transaction protocols
- Switches (arbiter-muxes)
- QoS (priority)
- Buffers
- Data path serialization
- Statistics probes
- Debug tracing
- Firewalls
- Register slices
- Clock domain crossings
- Voltage domains
- Power domains

These have caused new challenges to interconnect design.

Naturally, it is desirable to design an interconnect technology in a way that is reusable. That requires making it configurable. Supporting the feature requirements of interconnects entirely in the logic of crossbars creates a lot of complexity and potentially slow critical paths. Furthermore, a lot of wires are toggled even for a small volume of traffic, which consumes more power than necessary. The following proposes a reusable modular interconnect design that has superior results in simplicity, speed, area, and power efficiency.

We employ a 3 layer protocol. A transaction layer performs the reads and writes requested using AMBA, PIF, OCP, BVCI, or other industry standard protocols. This is the interface visible to the designers of the IP blocks connected through the interconnect.

A network interface unit (NIU) manages a transport layer protocol. It creates one or more packets for each transaction. All packets have a header. A read request packet or write response packet is just a header. Read data and write data packets include data after the header. The packet header encodes addresses, transaction parameters, and sideband signals as fields. The NIU controls outstanding transactions and tagged sequences. The header format is minimal, and optimized differently for each chip design. The header is used at each pseudo-switch within the interconnect to route requests from initiators to targets and responses from targets to initiators. The request and response paths are independent. This has the distinct advantage of eliminating logic and architectural dependencies.

The transported packets are transferred on the physical layer using a very simple protocol. This is the key enabler of modular design. The protocol consists of the following signals.

- Data [N bits] (driven by the sender)

- Valid [1 bit] (driven by the sender)
- Ready [1 bit] (driven by the receiver)

Valid and Ready implement flow control, which enable back-pressure feedback. This simple handshake protocol exists between all units of the NoC. Standardizing on a simple interface allows units to be connected interchangeably, in the style of LEGO® blocks.

**CLOCK GATING**

With well known chip design methodologies it is possible to gate the clock at each flip-flop in cycles that they are unchanged. This is applicable to the flops in all interconnect technologies, however it does not address clock tree power consumption.

The clock tree is a single signal and therefore much narrower than data paths. However, to reach all physically distributed flops the clock tree has a lot more metal than each data path bit. Since clocks, by definition, toggle twice per clock cycle, the clock tree typically consumes significantly more power than data paths.

In a crossbar, every clock net toggles twice per clock cycle, even when and where data is not flowing. While it is theoretically possible to achieve some clock gating to all crossbar logic in cycles when no data is transferred anywhere in the crossbar, that is largely impractical. It would require a large clock gating mux of many distant signals to generate enable signals back to many distant flops.

Building the interconnect from atomic modules of combinatorial logic allows unit level clock gating with much finer granularity than is possible within a monolithic crossbar.

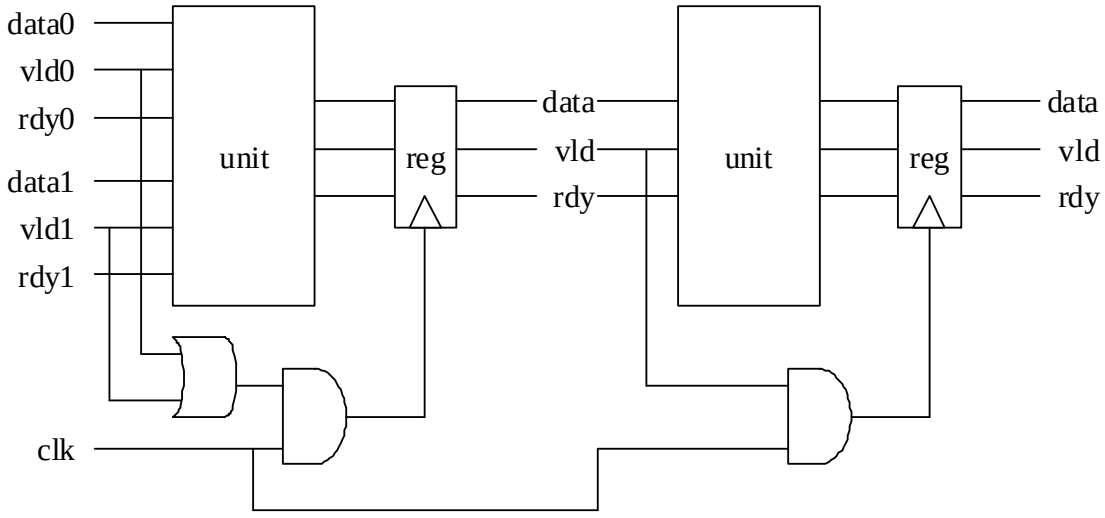


Figure 7: Unit level clock gating

Registers within and between units only toggle when the valid handshake signal is asserted, indicating that data traffic is present. Gating logic is local to each unit, so paths are short and little muxing is required to generate the enable signal. Clock gating is

distributed and each module of the modular interconnect is gated off for idle clock cycle, regardless of the rest of the system. This gives nearly ideal minimum switching power consumption.

## OTHER BENEFITS

Aside from the clock gating benefits of modular interconnect design, other benefits include improved use of mixed $V_t$ synthesis, reduced leakage power consumption, improved logic simplicity, and localization.

### Mixed $V_t$ synthesis

The ability to insert pipe stages anywhere between small modules to meet timing requirements with minimum latency improves the ability of synthesis tools to close timing. With margin they promote fewer paths from default high $V_t$ cells to faster low $V_t$ cells. In this way, pipelining between the elements of a modular design reduces leakage.

### Leakage

Furthermore, easier timing closure also improves EDA tools' ability to optimize for minimum area. A smaller die area reduces leakage power.

### Logic simplicity

A 64 bit AXI transaction interface protocol (or a feature-wise equivalent OCP interface protocol) has 272 wires. For the NoC a 64 bit packet interface requires 148 wires (64 bits data + 8 byte enables + ready + valid = 74 in each of the request and response networks). As a result, packetizing transactions to transport them between initiators and targets reduces wire count within the chip floorplan by a factor of 272/148 = 1.8.

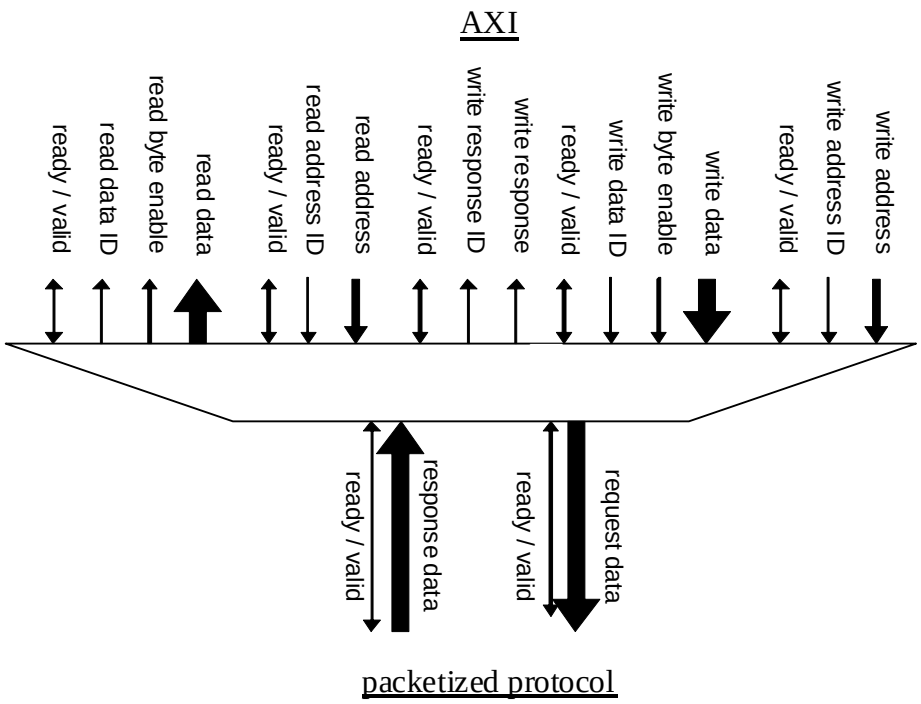


Figure 10: AXI to packetized protocol conversion

Because of the simple physical layer protocol for interfaces between units it is easy to change the serialization of packet data. All that is required is a simple mux and register half the data path width.

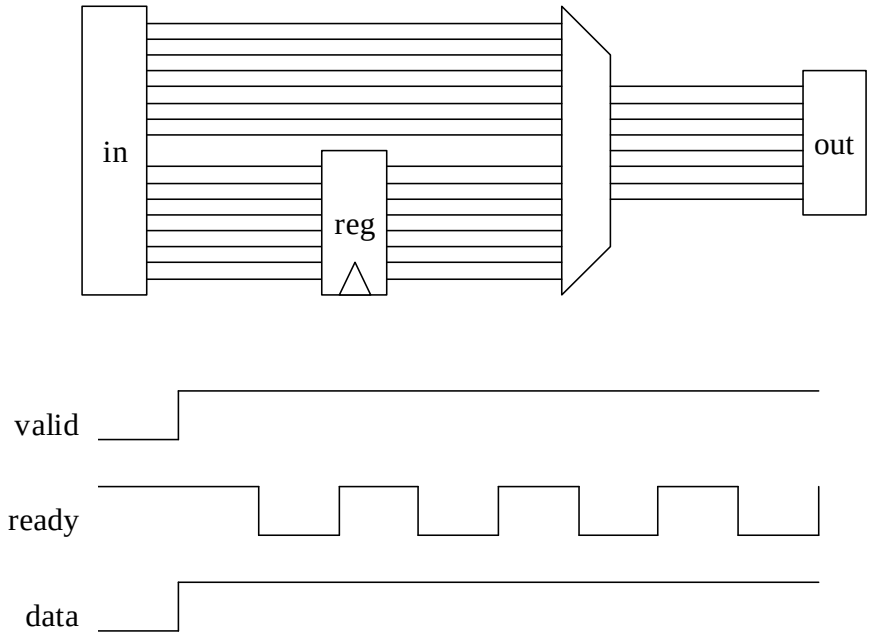


Figure 8: Data path of packet-based serialization from high to low bandwidth

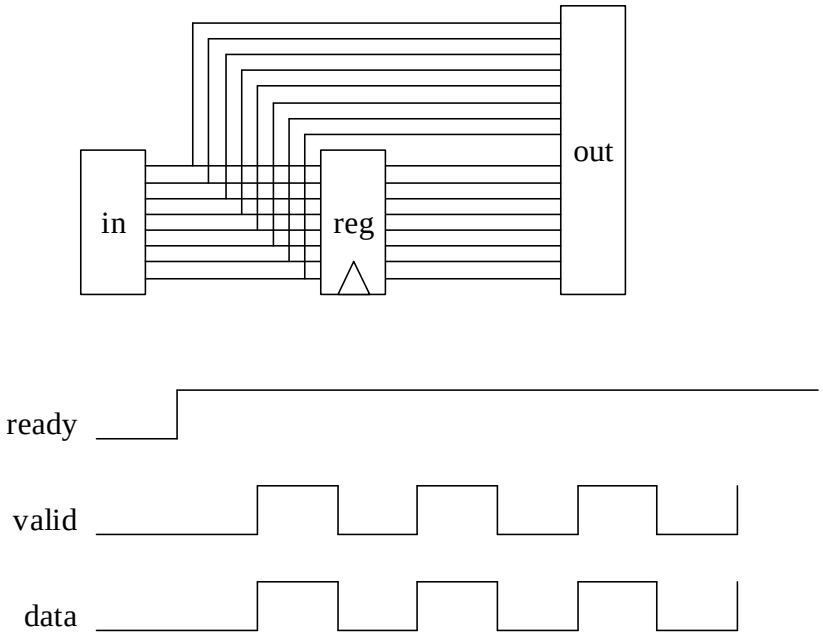


Figure 9: Data path of packet-based serialization from low to high bandwidth

Changing the serialization of data paths to be no larger than needed to meet bandwidth requirements in different parts of the chip reduces the interconnect logic area for all parts of the chip with less than the maximum bandwidth requirement. This comprises the large majority of the top level interconnect in most chips.

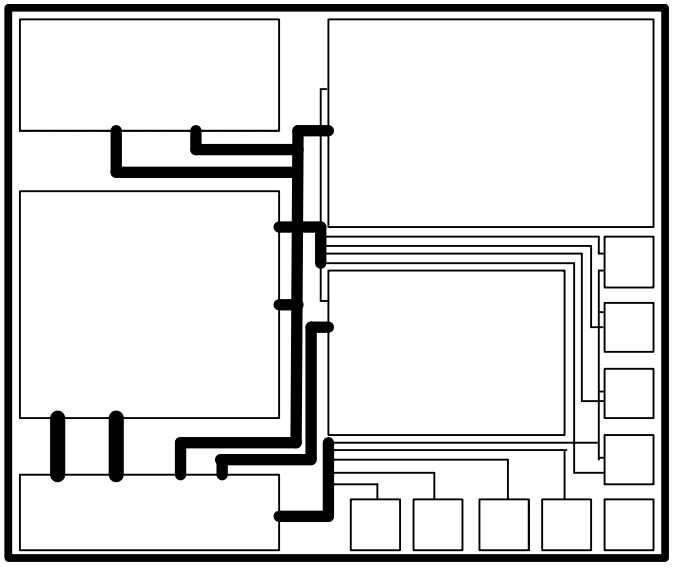

Figure 11: An example floorplan with localized serialization

**Localization**

By localizing units, such as the muxes between interfaces shown in Figure 6, the average length of wires between units is shorter. That means that less charge is consumed due to the capacitance of wires. It also simplifies the back end layout process by reducing connectivity dependencies between logic that is necessarily placed at great distances.

**RESULTS**

An analysis was performed on a hypothetical modular NoC interconnect for a mid-range set-top box SoC supporting 1080p120 video display. An interconnect of 11 master and 6 slave NIUs were modeled. The logic area is estimated at 183k gates, including buffering to meet performance requirements.

The analysis measured clock gated switching activity for three scenarios. The first is a worst-case video processing scenario. The video decoder, 120 Hz display output, and CPU together heavily load the system and use nearly all available DDR bandwidth. The second scenario is for average case video decode complexity. The third scenario is for web browsing with no video decode and just a 30 fps display rate.

| | Worst case | Typical video | Web browsing |
|---|---|---|---|
| DDR activity | 98% | 86% | 26% |
| NoC flop toggling | 42% | 35% | 7.6% |
| Toggle savings | 2.3x | 2.5x | 3.4x |

Given that a crossbar would be enabled for at least as many cycles as DDR activity, the toggle savings represents the factor of reduced power consumption from using modular design.

The analysis did not consider a standby scenario, which is expected to show an even greater toggle savings for NoC over crossbar. Furthermore, larger chips have more master NIU logic that accesses the same limited, shared resources. Such chips have a larger number of flops gated for a larger percentage of time. Therefore toggle savings for a modular NoC design improve with increased chip size.

## CONCLUSION

A modular NoC addresses the low power requirements of the top level interconnect fabric in highly integrated chips. By localizing clock gating, clock tree power is consumed only along the routes and for the cycles in which data is transferred. This gives minimal clock tree power consumption. Localized serialization enabled by a NoC uses no more data path logic than needed to support the bandwidth requirements of each link. This minimizes leakage area. Furthermore, modularity allows fine granularity of pipelining in order to close timing without wasted margin. This in turn allows the synthesis using smaller and less leaky gates.

Chip fabrication trends favor modular NoC design as a future technology. It is front end design approach that avoids back end physical design risks.

## REFERENCES

[1] Lecler, J., and Baillieu, G., Application driven network-on-chip architecture exploration & refinement for a complex SoC. Springer Design Automation for Embedded Systems 2011, vol. 15, issue 2, 133-158.

[2] Sundararajan, V., and Parhi, K. K., Low power synthesis of dual threshold voltage CMOS VLSI circuits. IEEE ISLPED 1999, 139-144.

[3] ARM, AMBA® AXI™ and ACE™ Protocol Specification, 2011.

[4] Dono, M., Macii, E., and Mazzoni, L., Power-Aware Clock Tree Planning. Proceedings of the 2004 international symposium on Physical design, April 18–21 2004.