Network Simplicity for Latency Insensitive Cores

Daniel Gebhardt, JunBok You, W. Scott Lee, Kenneth S. Stevens
University of Utah, Salt Lake City, UT 84112, U.S.A.
gebhardt@cs.utah.edu, {jyou,wlee,kstevens}@ece.utah.edu

Abstract

In this paper we examine a latency insensitive network composed of very fast and simple circuits that connects SoC cores that are also latency insensitive, de-synchronized, or asynchronous. These types of cores provide native flow control that is compatible with this network, thus reducing adapter overhead and buffering needs by applying backpressure directly to the sending core. We show that under realistic traffic patterns our sample network meets performance requirements and uses less power compared to a similar design. This concept of a simplified network, along with latency insensitive cores lends itself well to meeting the needs of low-power interconnect components in future design processes.

Traffic patterns of some system-on-chips (SoCs) are known at design time. In these cases, a custom generated network topology and physical placement of components yields improved performance and power than a regular-pattern network [1]. In many embedded applications, absolute performance is not the most important trait to optimize. Power, energy, or cost are often more important so long as the system meets the minimum performance.

Latency insensitive (LI), or *elastic* design, is an area of research typically orthogonal to NoC protocols. LI protocols combine asynchronous design concepts with a standard clocked environment, and provide tolerance to variation in communication channel latency. When applied to a core's design, LI can yield power savings and other advantages [2], and may soon become a more common design style. We have developed a LI protocol (pSELF) to allow operation with both synchronous and asynchronous circuits [3].

Our network circuits can interface directly with cores that operate with a phased elastic protocol [3]. With this scheme, the core and network interfaces of a network adapter (NA) follow compatible protocols. This simplification eliminates the extra size and power requirements of interfacing with a standardized core protocol. One study indicates an OCP NA implementation can add up to 50% latency over a native interface [4].

The network fabric is implemented using two components: a phase elastic half buffer (pEHB) and a binary-tree router. A pEHB contains a data latch, and associated gates and control latches implementing the phase synchronous elastic flow (pSELF) protocol. pEHBs may be added along long links in order to pipeline them for higher clock rates. A router consists of three bi-directional ports, control logic, and output buffers. It simultaneously routes a packet on each input port to one of its two possible output ports, and arbitrates if there is output contention. Packets contain source-routing bits and are, in this case, a single flit long.

We first evaluated this network topology and protocol using a custom simulator and a 16-core binary tree network to measure throughput under traffic distributions of uniform random and Gaussian. The uniform random traffic simulates highly global communication, and because of this network's low bisection bandwidth, the throughput saturates at only 0.22 packets/cycle per core. The Gaussian traffic simulates more localized communication when a core will more frequently communicate to topologically close cores than to distant ones. We set σ in the distribution such that the nearest one-quarter of the network nodes fall within two σ from a given node. The maximum throughput in this case is much better at 0.45 packets/cycle. With this Gaussian traffic at a 30% offered load, on average one out of five packet send attempts is blocked at the sending core by congestion, compared with two out of 5 attempts for uniform traffic. These results confirm that this network would be a poor choice for connecting many cores needing high bandwidth but communicating in an unknown pattern. However, it also indicates reasonable performance for more localized traffic, while allowing efficient global communication that occurs infrequently.

We also evaluated our network on two realistic traffic models. These have been used in previous work to evaluate application-specific NoC generation techniques [5]. The expected traffic is represented as a *communication trace graph* (CTG) showing the bandwidth requirement between various cores in the SoC. The first, a Multi-Window Displayer (MWD), has mostly one or two connections per core, all with relatively similar (and low) bandwidth requirements. The second model is a MPEG4 decoder (MPEG4). This CTG is quite different, and requires a number of high bandwidth connections to a few shared memories. We construct a tree topology by pairing cores, or groups of cores and their routers, that have the highest edge-weight between them in the core graph [6]. This topology generation is important to minimize network hops on paths needing a high bandwidth (or low latency, as the need may be). The generated topology for the MPEG4 design is shown in Figure 1. We simulated these topologies using the CTG information to determine if our network could support such applications successfully. In both cases, even for the more "difficult" MPEG4 application, the topology and network maintained the average asked bandwidth for various message sizes. Assuming our network is using 4-byte wide flits, and running at 1.3 GHz (in 130nm technology), the link bandwidth is 5.2 GB/s, fast enough to provide headroom for instantaneous bandwidth needs on the combined average 1.8 GB/s path to the sdram in the MPEG4 application.

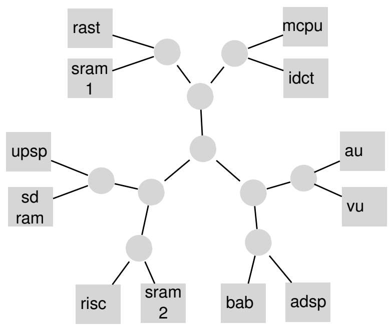

Figure 1. Automatically generated topology for MPEG-4 traffic simulation.

We evaluated the power consumption of our pSELF-based routers against a Xpipes Lite [5] 4x4 router. We used Synopsys Design Compiler and Power Compiler to estimate our router's power at the technology of our fabricated test-chip (0.5 micron), and scaled the values down for a 130 nm process using constant-field scaling theory. The power of a Xpipes 4x4 router with 32-bit flit widths and four output buffer stages is given as 55.5 mW when running at its maximum of 1 GHz. We constructed a functionally similar "4x4" router from two of our binary-tree routers and pEHBs. We estimated the FIFO buffers' power of the Xpipes router, and added that to our 4x4 router construct for fair comparison. The power for this arrangement running at 1.3 GHz is 28.9 mW, 48% less than that of the Xpipes router.

It is likely that additional internal network buffering, virtual channels, and other features could show performance improvements for certain applications. However, a simplified network concept can still offer the performance required for many designs while giving a significant power advantage. A network compatible with elastic protocols yields very efficient circuits and there is no need for extra flow-control logic in the core's network adapter, nor extra *NACK* control messages. If a core is *desynchronized* [7] there is an additional opportunity for greater power savings, and pSELF is directly compatible with the handshaking used in the desynchronization process, requiring only a minimal conversion circuit.

References

[1] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. D. Micheli, and L. Raffo, "Designing application-specific networks on chips with floorplan information," in *Proc. of ICCAD*, 2006, pp. 355–362.

[2] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Synthesis of synchronous elastic architectures," in *Proc. of DAC*, Jul. 2006, pp. 657–662.

[3] J. You, Y. Xu, H. Han, and K. S. Stevens, "Performance Evaluation of Elastic GALS Interfaces and Network Fabric," in *Workshop on FMGALS*. Elsevier ENTCS, May 2007.

[4] L. Ost, A. Mello, J. Palma, F. G. Moraes, and N. Calazans, "Maia: a framework for networks on chip generation and verification," in *ASP-DAC*, 2005, pp. 49–52.

[5] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. De Micheli, "XPipes Lite: a Synthesis Oriented Design Library for Networks on Chips," in *DATE*, vol. 2, 2005, pp. 1188–1193.

[6] D. Gebhardt and K. S. Stevens, "Elastic flow in an application specific network-on-chip," in *Workshop on FMGALS*. Elsevier ENTCS, May 2007.

[7] J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of asynchronous circuits from synchronous specifications," *IEEE Trans. on CAD*, vol. 25, no. 10, pp. 1904–1921, 2006.