A Practical Design and Implementation of On-Chip NI for Integrating Bus Based IP Legacies

Jung-Ho, Lee
Information and Communications University
System Integration Technology Institute
119, Munjiro, Yuseong-gu, Daejeon 305-732
Korea, Rep. Of

Sin-Chong, Park
Information and Communications University
System Integration Technology Institute
119, Munjiro, Yuseong-gu, Daejeon 305-732
Korea, Rep. Of

Abstract: Decoupling communication and computation by adapting on-chip network has already become an irreversible trend, as we proceed into deeper sub-micron era. Nevertheless, innumerable number of IP cores is developed for conventional bus protocol. Brute force aggregation of bus based IP cores is not a good idea, because it can cause excessive load on on-chip network, by stalling routers and network interfaces. Therefore, for a bus based IP usually has better performance when it is plugged into a local bus with dedicated components which have high affinity with the core. This paper argues practical design and implementation issues of NI for on-chip network interconnection between bus-connected components. Implemented system gives out waveform with significantly small latency of packetization at the clock boundary.

Key–Words: Network On Chip, IP Legacy Reusing, Efficient Network Interface

1 Introduction

According to a report taken from the International Technology Roadmap for Semiconductors[1], wire delay is going to be an increasingly significant comparing to gate delay as CMOS fabrication technologies scales down into deeper submicron level. This undesirable defect arises from magnified wire resistance per-mm, with unchanging wire capacitance. Greater wire delay induces greater possibility of synchronization error from tight clock skew control which is required to compensate undefined latency from slow wire. This obstacle leads it harder to fix on the global wires among IP blocks sharing common bus.

Furthermore, denser the chip fabric becomes, more amounts of IP cores are getting aggregated into one chip. It has made possible to integrate many PCBs (Printed Circuit Board) on a chip. For ample space enough for many cores has been allocated on a chip, the problem of communication among various cores, which have different protocol and clock frequency, has naturally come into a question. Considering this problem, we can no more maintain globally synchronous bus protocol among IP cores. The request for remedy of this problem has begun from on-board communication designers, because they had confronted to the problem of synchronicity between very high speed device (such as, 3D graphics accelerator, and video composing board) and low speed device (such as, modem, and sound card). Therefore, on-board communication paradigm has already shifted from on-board buses, like a PCI (Peripheral Component Interconnect) to a point-to-point high-speed network, like a PCI-Express. Likewise, on-chip communication scheme is about to move to point-to-point network to make various IP cores working together on a chip.

This new on-chip network communication requires some components, which are analogous to those of macro-network: network interface (NI), and router. NI implements interface among IP cores, and router transports data from source spot to destination spot. Router is a major component, which entangled with various design considerations to discuss, such as scheme for error recovery (e.g. CRC vs. Hamming), switching method (packet vs. circuit), and type of routing method (deterministic vs. adaptive). This paper, however, would mainly focus on implementation issues of NI, which is the first barrier to on-chip network design era. We arrive to a goal meeting NI services that QoS support by offering guaranteed throughput, and decoupling between computation and communication, which ensures orthogonal operations among IP cores. This paper also provides a transparent tutorial on sequence of generating control signals for AXI port.

Topics covered in this paper are as follows. In next section, we briefly state related work on NIs, which is shown in various on-chip network implementations. In Section 3, we briefly present functions of our NI in the context of targeted system. In Section 4, we will clarify the details of AXI ports implementation. In Section 5, we clarify how the main functions of our NI implemented, separated from a specific protocol. In Section 6, we discuss RTL (Register Transfer Level) simulation result, and future improvement of the system, and we conclude in Section 7.

2 Related Works

NI design has already been an issue that attracted many researchers of computer network [2]. Nevertheless, researchers should devise new designs for on-chip interconnection, because NI designs developed for macro-network can not be directly adapted because of high cost. Pioneers of NI have poured much effort on designing NI, which has minimum size and power, guaranteeing QoS (Quality of Service). In this section, we briefly recapitulate a set of NI design, extracted from representative on-chip networks: AETHEREAL[3][9], BONE[4][10], and XPIPES[5].

2.1 AETHEREAL

The AETHEREAL focuses on offering guaranteed services (such as, lower bounds on throughput, and upper bounds on latency). To meet this requirement, they inserted a sequence of local timing table, which allocates timing slot to each packet stream, into routers. But soon they found that the design requires excessive amount of area. After this realization, theyve moved the role of offering guaranteed service from routers to NIs. The information is now provided in packet header which is generated by NI. Their NI can allocate slots either statically or dynamically; static allocation is used when GT (guaranteed throughput) is required, while dynamic allocation pursues BE (best effort) utilization of network bandwidth. Here, GT packet has higher priority than BE packet, because when GT packet reserves a series of slots, no BE packet can use those slots; GT packet preoccupies a deterministic path, and BE packet is dynamically allocated in round-robin manner. And its NI also equipped with end-to-end flow control, threshold mechanism with flush, narrowcast (one master, multiple slaves, a transaction is executed by only one slave), and multicast (one master, multiple slaves, all slaves executing each transaction) capabilities. Their threshold mechanism allows incoming flits (basic flow unit) to bypass a channel as long as the total sum of transmitted flits is smaller than threshold.

2.2 BONE

The design of BONE aims at very practical implementation, without less critical functions, pursuing for reality of VLSI implementation, such as Go-Back-N error recovery capability. With respect to its NI, it used 4:1 serialization which transfers 80-bits packet through 20-bits links to minimize energy utilization per packet transfer. They provided a mathematical justification of [6] to argue that their scheme is optimal structure in perspective of energy. For energy optimization, their packet size is naturally restricted to a fixed size, which becomes drawback when we want to accommodate it real-world point-to-point protocol (e.g. AXI[7], OCP[8]). The reason why variable packet size is essential for those protocols is discussed in next section. The unique characteristic of the design of BONE NI is two flow control units for two different switching modes (circuit-switching and packet-switching). Their NI should have implemented special synchronization unit called programmable delay synchronizer to compensate the delay of packet switch with respect to bypassed packet in circuit switch (nearly no delay). Authors of BONE claim that their design is for using multi-level and mixed packet/circuit switching, combining the benefits of two switching mode; A packet switched network shows efficient usage of channel resource, while a circuit switched network leads low-latency system.

2.3 XPIPES

The Xpipes is also frequently recited on-chip network model proposed by a well-known group. Xpipes is a library of System-C based soft IPs(switches, NI, and link topologies). The significant trait of its design is that Xpipes is powered by a reconfigurable design automation tool, called XpipesCompiler. This compiler is used to automatically generate soft IP components and map them into an application specific topology. As for NI design, we can find some important design criteria that comprise fundamental on-chip network principles. They present a standard packet preparation process that consists of building the packet header, payload and packet tail. The header contains the necessary routing and network information. The packet tail has a CRC code to keep error-resiliency. Alongside that basic packetizing capability, it has a dedicated register to fill some space left in flit with header information; that increases efficiency of network by growing information density. Authors of Xpipes also states about the block that translates response packet to OCP response signal.

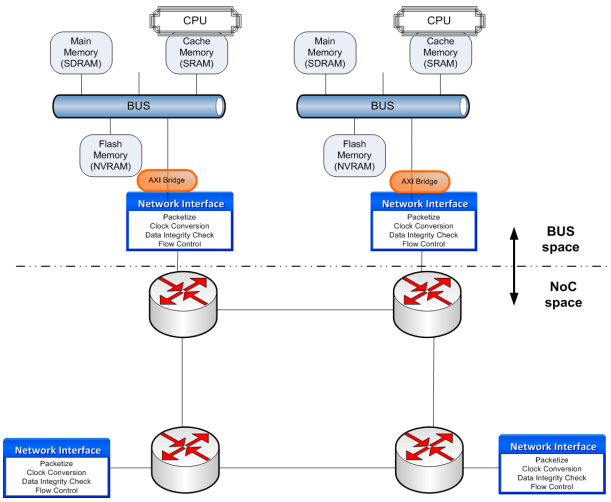

Figure 1: Target System

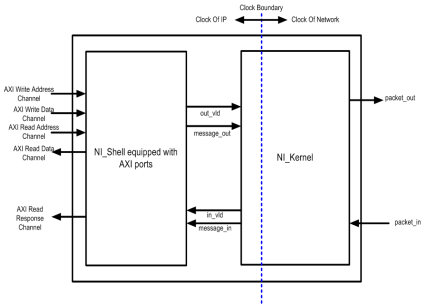

Figure 2: Modular Structure of NI

3 Characteristics of Proposed NI

As mentioned above, a bewildering array of on-chip networks have their own NIs, which seems to provide supple functions. However, they are all lack of something.

In the first place, NI of AETHEREAL has evolved for a course of time, making it more robust; it, however, has some drawback for practical system which has only limited number of nodes as presented in Fig. 1. The system, illustrated in Fig. 1, targets for the earliest on-chip network system that has a few wholesome nodes that share their own local memories through the local bus interface. This configuration can be quickly adapted to the current bus based IP cores, because it supports local bus interconnection among the components that requires high affinity. For the system shown above, 2bit control data associated with a flit for supporting two types of services (BE-best efforts, and GT-guaranteed throughput) and decision of packet size (last bit should be set to mark the tail) is just a redundant load; As for type of service, it almost always prefers GT (Guaranteed Throughput), if the number of nodes is smaller than locally bus-connected components; this is because round-robin arbitration process for BE traffic occurring in all intermediate switches are inefficient and redundant. Not only that, tail marker is also redundant, because burst size and length is predefined in AXI control signal. Therefore, proposed NI fixes its service type to GT scheme, which is implemented by deterministic source routing.

Secondly, NI of BONE also has its drawback that it only supports fixed-length packet, which cannot be applicable to dominant point-to-point protocols like AXI. To put it clearly, AXI has feature, called outstanding address issue, that prefetches all control information including burst size and length, before corresponding data are sent off. This means that data stream of various size and length are intermingled together, and fixed length packet is not viable.

Thirdly, XPipes is a too massive System-C model that focuses on design automation guaranteeing optimal mapping of application specific IP cores under perfect on-chip network environment, and, hence not proper for bus-network mixed architecture.

As shown in Fig. 2, overall structure is equivalent to AETHEREALs, separating NI Shell, that implement the protocol specific operations from NI Kernel, that implements the main function of NI, providing packetization, end-to-end flow controls, and clock domain crossings.

4 Implementation of NI Shell with AXI Ports

AXI protocol has 5 distinct channels, as listed here: read/write address channel, read/write data channel, and response channel. Address channels are composed of 32bit address data and associated control signals, and data channels are composed of 32bit payload and associated control signals. Here, read operation is performed by sending control signals with starting address, and write operation adds data payload which amounts specified from awlen field. The awlen control signal, which is declared from the initiating point of transaction, makes the NI design more efficient, because the system can determine the number of flits to transmit before all data is stacked into the NI. For both read and write operation, slave should receive burst size and length with corresponding operation ID (identifier for data interleaving), and, hence

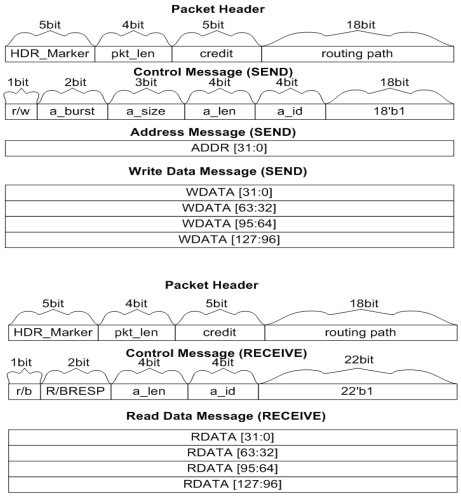

Figure 3: Packet Specification

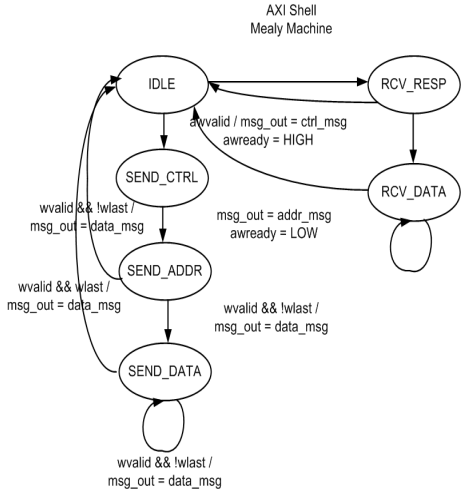

Figure 4: Mealy FSM for NI Shell

these signals ,transported on address channel, are included in control message, as shown in Fig. 3.

For a response message (Fig. 3), we've made a bit(r/b) to indicate if the message corresponds read data or to a write response. AXI response signal indicates whether the read or write transaction was successful or not. According to the response packet from the slave, the master determines whether retransmission is required or not. Note that packet length field is copied to response control packet to take an advantage of predetermined packet length.

Many readers, who are entangled with implementation, may ask for detailed control flow to generate message from control and address signals, and vice versa. Fig. 4 gives you a mealy machine I used for implementation of NI shell presented in this paper. When reset signal is asserted state is set to IDLE. When the state is IDLE, when control and address signals on address channel comes up, it snatches out them and makes messages described in Fig. 3. In SEND_CTRL and SEND_ADDR state, it is worthy to note the generation of awready signal to hold address data for two clock cycle, otherwise master can issue a new address after one cycle of SEND_CTRL phase. After sending all messages on data channel, the state machine comes back, stimulated by wlast signal. As for response phase, it is a completely reverse process of message generation; based on incoming message it makes corresponding response signal or forwards data flits.

5 NI Kernel Architecture

Fig. 5 shows NI kernel architecture targeting for interconnection among a series of bus-based IP cores. To meet the requirements of targeted system, proposed NI operates as described below. Requests are generated by an AXI control message. Request generator extracts packet length from control message flit, and raise pktstart signal when it detects new control message issued by an AXI master. Header builder fabricates a packet header, enlisting packet length, credits to report, and routing path; credit tells NI of the slave side that master side NI can receive at most specified amounts of packet; and deterministic routing table, which keeps a compete path from source to destination, fills the routing path field of packet header. When a packet inflows from network into NI, Packet Analyzer pills of packet header and push forward AXI messages.

When data are passed from one domain into another and the clock of the destination domain is not related to the clock of the source domain, the control signals of both source and destination side needs to be synchronized. Fig. 6 is a control signal timing diagram for design of flit controller.

In our case, sel/enable hdrbuilder signal is initiated by startpkt signal. To circumvent metastable state of startpkt, we inserts some delay component, and make it synchronized to the falling edge of network clock. One of the most significant role of our NI design is to develop a high throughput and low latency synchronizer. After generating header, encnt/readfromstack signals are asserted to enable counter, while push forward packets extracted from FIFO. Then, counter reaches packet length specified in pktlen signal, it resets counter and de-asserts readfromcounter/encnt signal. The most important design criterion of flit control design is reducing time stalling on FIFO as much as possible, but preventing metastable state which occurs between different clock domains.

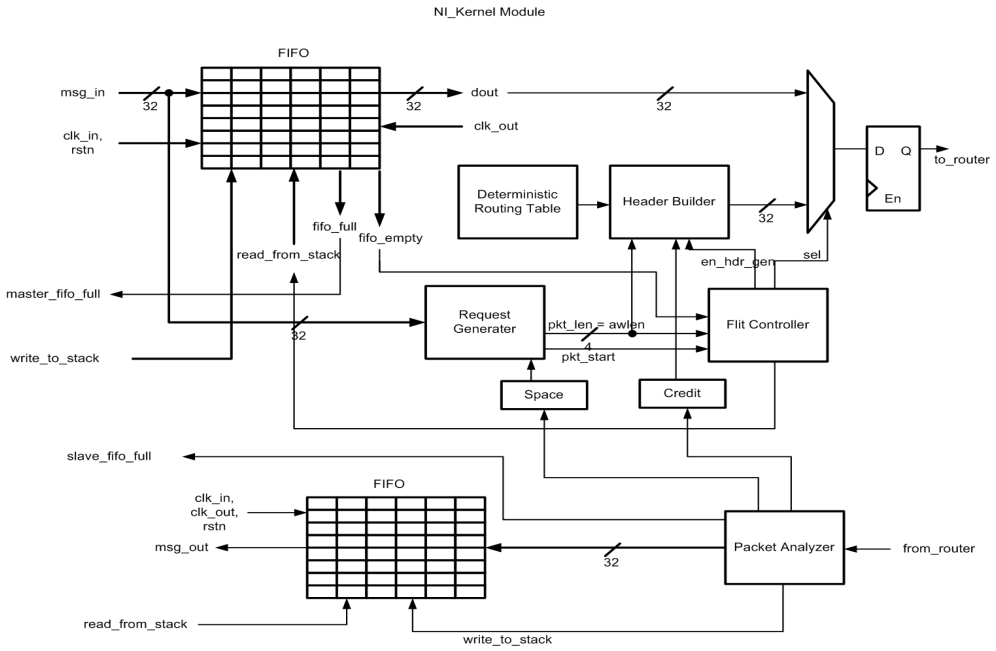

Figure 5: NI Kernel Block Diagram

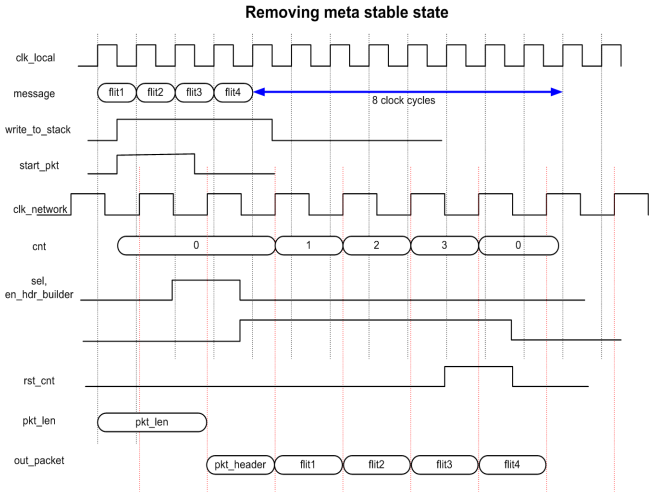

Figure 6: Timing Diagram of Flit Control Signals

6 Simulation Result

Simulation has been carried out based on cycle-accurate Verilog RTL code. Here, the AXI master issues 4-beats (awlen = 3) and 4-bytes (awsize = 2) FIFO type (awburst = 0) write burst command, as example case. Waveform of Fig. 7 is the result of the simulation performed under the environment where input clock is 100MHz, and network operates at 500MHz. And Fig. 8 shows a result that is performed under the condition of 100MHz and 75MHz, of input and output respectively. In both case, we can confirm that packets is generated as soon as possible quickly releasing expensive FIFO space of NI, and, also, the synchronization between clock domain boundary is guaranteed in most efficient and successful way.

7 Conclusion

This paper handles practical issues, arisen from the on-chip network implementation, under the environment that is composed of bus-based nodes interconnected by network. We offer a complete set of design datasheet for rapid NI design which is directly applicable to real-world AXI protocol. Our NI has adapted modular design, and, hence, separating protocol specific NI shell from NI kernel, just like AETHEREAL's NI. And we also make our NI more feasible for aggregation of bus-based IP cores, by removing some sources of overhead that comes because the management of packet based on types of service (GT or BE), and associated control signal for marking start and end of packet. And we also points out a practical problem of NI of BONE and provides a solution. And we show a cycle-accurate implementation of our NI, showing immediate response, through minimized amount of packet remaining in FIFO.

Acknowledgements: This work was partly sponsored by ETRI SoC Industry Promotion Center, Human Resource Development Project for IT SoC Architect.

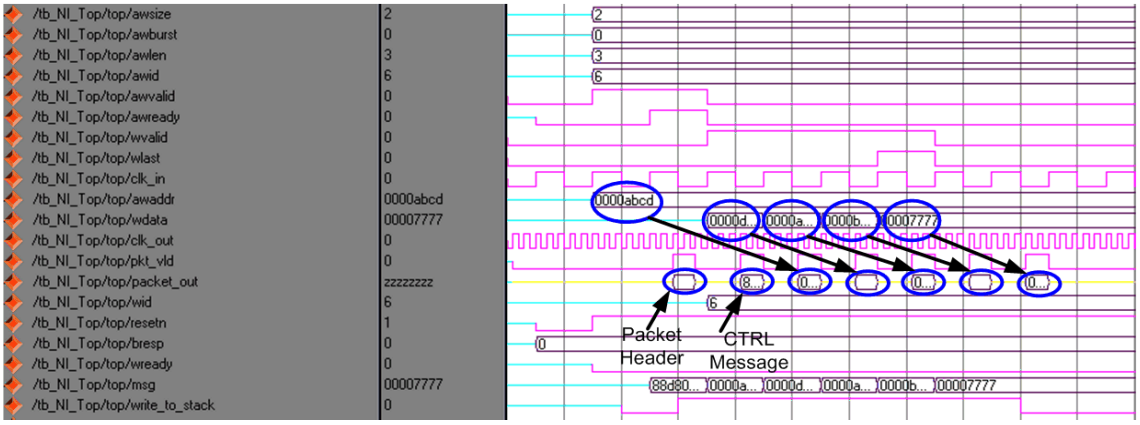

Figure 7: Input Clock: 100Mhz, Output Clock: 500Mhz

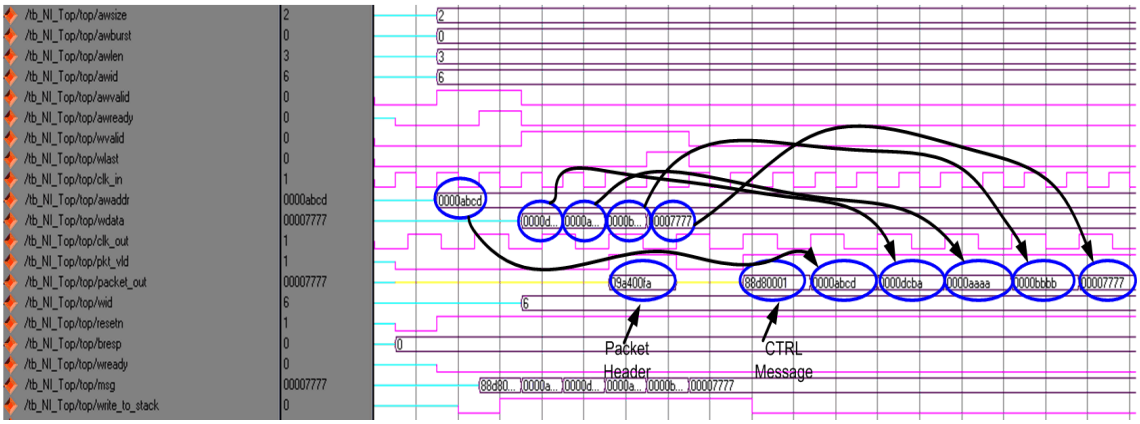

Figure 8: Input Clock: 100Mhz, Output Clock: 75Mhz

References:

[1] ITRS. 2001. *International technology roadmap for semiconductors. Tech. rep.*, International technology Roadmap for Semiconductors.

[2] Andrei Radulescu, John Dielissen, Santiago Gonzalez Pestana, Om Prakash Gangwal, Edwin Rijpkema, Paul Wielage, and Kees Goossens, An Efficient On-Chip NI offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration, *IEEE Trans. On Computer Aided Design Of Itegrated Circuit and Systems*, vol. 24, no. 1, Jan. 2005.

[3] K. Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, Networks on silicon: combining best-effort and gauranteed services, in Proc. *Design Automation Test Europe*, 2002. 2002

[4] S.-J. Lee et al., Packet Switched On-Chip Interconnection Network for System-onChip Applications, *IEEE Trans. Circuits and Systems Part II*, vol. 52, no. 6, June 2005, pp. 308-312.

[5] Davide Bertozzi, Antone Jalabert, and Giovanni De Micheli, NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip, *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 2, Feb 2005

[6] H. -S. Wang et al., A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers, *IEEE Micro*, vol. 23, no. 1, Jan.-Feb 2003, pp. 26–35

[7] ARM, AMBA AXI Protocol Specification, Mar. 2004.

[8] OCP International Partnership, Open Core Protocol Specification. 2.0 Release Candidate, 2003.

[9] K. Goossens, J. Dielissen, and A. Radulescu, AEthereal Network on Chip: Concepts, Architectures, and Implementations, *IEEE Design and Test of Computers*, vol. 22, no. 5, 2005.

[10] S. -J. Lee, K. Lee, and H. Yoo, Analysis and Implementation of Practical, Cost-Effective Networks on Chips, *IEEE Design and Test of Computers*, vol. 22, no. 5, 2005.