Analysis of Error Recovery Schemes for Networks on Chips

Srinivasan Murali
Stanford University

Theocharis Theocharides, N. Vijaykrishnan, and Mary Jane Irwin
Pennsylvania State University

Luca Benini
University of Bologna

Giovanni De Micheli
École Polytechnique Fédérale de Lausanne

Editor's note:
Error resiliency is a must for NoCs, but it must not incur undue costs—particularly in terms of energy consumption. Here, the authors present an authoritative discussion of the trade-offs involved in various error recovery schemes, enabling designers to make optimal decisions.
—André Ivanov, University of British Columbia

AS DEVICES SHRINK toward the nanometer scale, on-chip interconnects are becoming a critical bottleneck in meeting performance and power consumption requirements of chip designs. Industry and academia recognize the interconnect problem as an important design constraint, and, consequently, researchers have proposed packet-based on-chip communication networks, known as networks on chips (NoCs), to address the challenges of increasing interconnect complexity.[1-5] NoC designs promise to deliver fast, reliable, energy-efficient communication between on-chip components. Because most application traffic is bursty in nature, packet-switched networks are suitable for NoCs.[2,4,5]

Another effect of shrinking feature size is that power supply voltage and device V_t decrease, and wires become unreliable because they are increasingly susceptible to noise sources such as crosstalk, coupling noise, soft errors, and process variation.[6] Using aggressive voltage-scaling techniques to reduce a system's power consumption further increases the system's susceptibility to various noise sources. Providing resilience from such transient delay and logic errors is critical for proper system operation.

Error detection or correction mechanisms can protect the system from transient errors that occur in the communication subsystem. These schemes can use end-to-end flow control (network level) or switch-to-switch flow control (link level). In a simple retransmission scheme, the sender adds error detection codes (parity or cyclic redundancy check codes) to the original data, and the receiver checks the received data for correctness. If it detects an error, it requests the sender to retransmit the data. Alternatively, the sender can add error-correcting codes (such as Hamming codes) to the data, and the receiver can correct errors. Hybrid schemes with combined retransmission and error correction capabilities are also possible. Because the error detection/correction capability, area-power overhead, and performance of the various schemes differ, the choice of error recovery scheme for an application requires exploring multiple power-performance-reliability trade-offs.

In this article, we relate these three major design constraints to characterize efficient error recovery mechanisms for the NoC design environment. We explore error control mechanisms at the data link and network layers and present the schemes' architectural details. We investigate the energy efficiency, error protection efficiency, and performance impact of various error recovery mechanisms.

Our objective is twofold: First, we want to identify the major power overhead issues of various error recovery schemes, so that designers can create efficient mechanisms to address them. Second, we want to provide the

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

Related work

Much research has focused on the quest for reliable, energy-efficient NoC architectures. Error protection is applicable at several levels of a NoC design. For example, Marculescu proposes fault-tolerant routing algorithms.[1] Worm et al. propose dynamically varying the supply voltage according to the error rate on the links.[2] Li et al. propose monitoring the data bus to detect adverse switching patterns (that increase wire delay) and vary the clock cycle to avoid timing errors on the bus.[3] Many researchers, such as Srinivas et al., propose bus-encoding techniques that decrease crosstalk between wires and avoid adversarial switching patterns on the data bus.[4]

Hegde and Shanbhag were first to present a methodology for trading off power and reliability using error control codes for SoC signaling.[5] Bertozzi, Benini, and De Micheli explore the energy behavior of different error detection and correction schemes for on-chip data buses.[6] In the NoC domain, the efficiency of various error detection and correction mechanisms are yet to be studied in detail. Some research refers to incorporating such mechanisms into the network and data link layers of existing architectures; however, these works don't describe the trade-offs involved.[7,8] Zimmer and Jantsch present a fault model notation and explore the use of multiple encoding schemes for different parts of a packet.[9] Vellanki, Banerjee, and Chatha study the use of single-error correction and parity-based error detection schemes for NoCs.[10]

References

1. R. Marculescu, "Networks-On-Chip: The Quest for On-Chip Fault-Tolerant Communication," *Proc. IEEE Computer Soc. Ann. Symp. VLSI*, IEEE Press, 2003, pp. 8-12.
2. F. Worm et al., "An Adaptive Low-Power Transmission Scheme for On-Chip Networks," *Proc. 15th Int'l Symp. System Synthesis* (ISSS 02), ACM Press, 2002, pp. 92-100.
3. L. Li et al., "A Crosstalk Aware Interconnect with Variable Cycle Transmission," *Proc. Design, Automation and Test in Europe Conf.* (DATE 04), vol. 1, IEEE Press, 2004, pp. 102-107.
4. S. Srinivas et al.,"Coding for System-on-Chip Networks: A Unified Framework," *Proc. Design Automation Conf.* (DAC 04), ACM Press, 2004, pp. 103-106.
5. R. Hegde and N.R. Shanbhag, "Towards Achieving Energy Efficiency in Presence of Deep Submicron Noise," *IEEE Trans. VLSI Systems*, vol. 8, no. 4, Aug. 2000, pp. 379-391.
6. D. Bertozzi, L. Benini, and G. De Micheli, "Low-Power Error-Resilient Encoding for On-Chip Data Buses," *Proc. Design, Automation and Test in Europe Conf.* (DATE 02), IEEE Press, 2002, pp. 102-109.
7. M. Dallosso et al., "xpipes: A Latency Insensitive Parameterized Network-on-Chip Architecture for Multi-Processor SoCs," *Proc. Int'l Conf. Computer Design* (ICCD 03), IEEE Press, 2003, pp. 536-539.
8. "The Nostrum Backbone," http://www.imit.kth.se/info/FOFU/Nostrum/.
9. H. Zimmer and A. Jantsch, "A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip," *Proc. 1st Int'l Conf. Hardware/Software Codesign and System Synthesis* (CODES 03), IEEE Press, 2003, pp. 188-193.
10. P. Vellanki, N. Banerjee, and K.S. Chatha, "Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures," *Proc. 14th Great Lakes Symp. VLSI*, ACM Press, 2004, pp. 45-50.

designer with information that will aid in the choice of an appropriate error control mechanism for the targeted application. In practice, different network architectures (with various topologies, switch architectures, routing, and flow control) exist, making generalized quantitative comparisons difficult. Nevertheless, we present a general design methodology and make comparisons of the error recovery schemes based on reasonable assumptions about the network architecture.

Error control mechanisms and on-chip networks

Among the many NoC architectures proposed in the literature (see the "Related work" sidebar), we chose one that incorporates features that have been successful in many NoC designs and represents a reasonable design point. We use it as the basic architecture for incorporating the error recovery schemes used in the experiments.

In the architecture we chose, the processor and memory cores communicate with each other through network components: switches, links, and network interfaces (NIs). NIs packetize data from the cores and build routing information for data communication. Each core has a sender and receiver NI for sending and receiving data from and to the core. We use an input-queued router with credit-based flow control[7] with each packet segmented into multiple flits (flow control units).

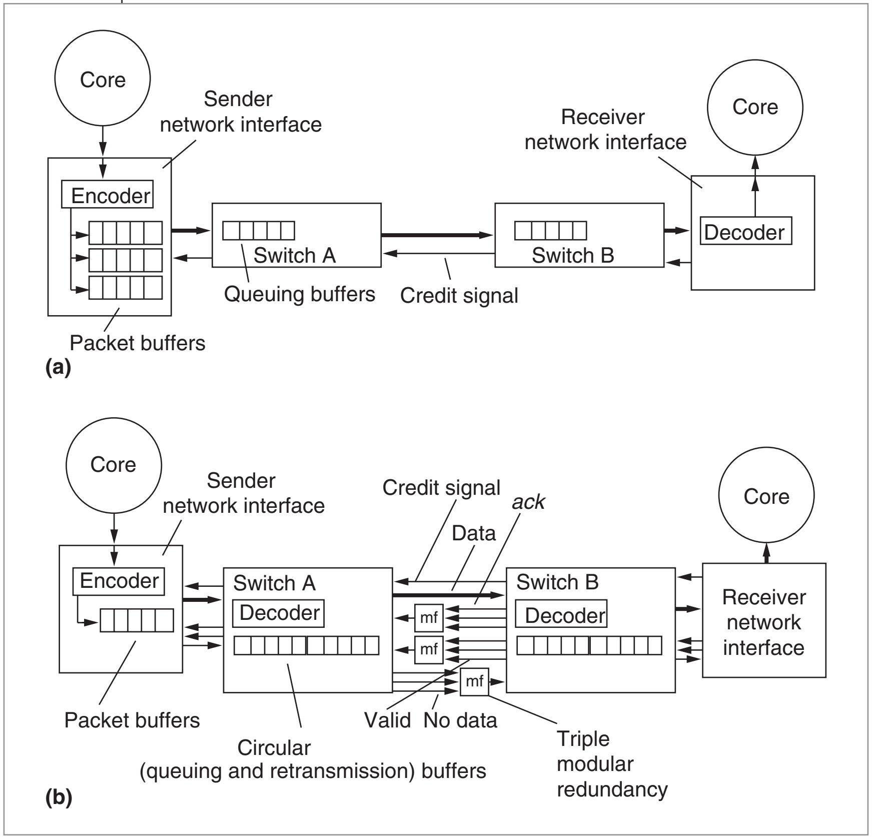

Figure 1. End-to-end (a) and switch-to-switch (b) retransmission architectures.

We assume static routing, with paths set up at the sender NI, and wormhole flow control for data transfer. For maximum network throughput, the number of queuing buffers required at each switch input should be at least $2N_L + 1$ flits, where N_L is the number of cycles required to cross the link between adjacent switches. The reason is that in credit-based flow control, it takes one cycle to generate a credit, N_L cycles for the credit to reach the preceding switch, and N_L cycles for a flit to reach a switch from the preceding switch.

Switch architectures for error detection and correction schemes

We identify three classes of error recovery schemes: end-to-end, switch-to-switch, and hybrid. Figure 1 illustrates the end-to-end and switch-to-switch architectures.

End-to-end error detection

In the end-to-end (ee) error detection scheme, we add parity (ee-par) or cyclic redundancy check (ee-crc) codes to packets. A CRC or parity encoder is added to the sender NI and decoders are added at the receiver NI. The sender NI has one or more packet buffers in which it stores packets that have been transmitted. The receiver NI sends a *nack* or an *ack* signal back to the sender, depending on whether the data contained an error or not. In a request-response transaction (as in the open core protocol, http://www.ocpip.org), the *ack* or *nack* signal can piggyback on the response packet. To account for errors on the *ack* or *nack* packets, we also have a time-out mechanism for retransmission at the sender. To detect reception of duplicate packets, we use packet sequence identifiers. Because the header flit carries critical information (such as routing information), it is protected by parity or CRC codes, which the switch checks at each hop traversal. If a switch detects an error on a packet's header flit, it drops the packet. Also, we use redundancy to protect the flit-type bits (which identify header, body, or tail flits).

Switch-to-switch error detection

Switch-to-switch schemes add the error detection hardware at each switch input and retransmit data between adjacent switches. There are two types of switch-to-switch schemes: parity or CRC at flit level and at packet level. Figure 1b shows how we modify the switch architecture to support these schemes. The additional buffers added at each switch input store packets until an *ack* or *nack* signal comes from the next switch or NI. The number of buffers required to support switch-to-switch retransmission depends on whether error detection occurs at packet or flit level.

In the switch-to-switch flit-level (ssf) error detection scheme, the sender NI adds the parity or CRC bits to each flit. At each switch input, there are two sets of buffers: queuing buffers for credit-based flow control, as in the basic switch architecture, and retransmission buffers to support the switch-to-switch retransmission mechanism. Like queuing buffers, retransmission buffers at each switch input require a capacity of $2N_L + 1$ flits for full-throughput operation. We use redundancy, such as triple modular redundancy (TMR), to handle errors on the control lines (such as the *ack* line).

In the switch-to-switch packet-level (ssp) error detection scheme, we add the parity or CRC bits to the packet's tail flit. Because error checking occurs only when the tail flit reaches the next switch, the number of retransmission buffers required at each switch input is $2N_L + f$, where f is the number of flits in the packet. The ssp scheme also requires header flit protection, like the ee scheme.

Hybrid scheme

In the hybrid single-error-correcting, multiple-error-detecting (ec+ed) scheme, the receiver corrects any single bit error on a flit, but for multiple bit errors, it requests end-to-end retransmission of data from the sender NI. We don't consider pure error-correcting schemes here. In such schemes, when a switch drops a packet (as a result of errors in the header flit), recovering from the situation is difficult because no mechanism exists for retransmitting the packet.

Energy estimation

A generic energy estimation model[8] relates each packet's energy consumption to the number of hop traversals and the energy consumed by the packet at each hop. We expanded this estimation a step further by designing and characterizing the circuit schematics of individual switch components in 70-nm technology using the Berkeley Predictive Technology Model (http://www-device.eecs.berkeley.edu/~ptm/). From this model, we estimated the average dynamic power as well as the leakage power per flit per component.

For correct system functionality, the error detection and correction circuitry and the retransmission buffers must be error free. We used relaxed scaling rules and soft-error-tolerant design methodologies for these components.[9] Our power estimations accounted for the additional overhead incurred in making these components error free (which increases the components' power consumption by about 8% to 10%).

To analyze the error recovery schemes, we fixed a constraint on the residual flit-error rate; that is, we imposed the same probability of one or more undetected errors (per flit) at the decoder side on each scheme. We assumed that an undetected error causes the system to crash.

Experiments and simulation results

We considered two sets of experiments. In one, we assumed the system's operating voltage (with different error recovery schemes) varies to match a certain residual flit-error rate requirement. For this set of experiments, we used previously published error models.[10] In the second set, we assumed the various schemes' voltages to be the same, but we investigated the effects of different error rates on the schemes.

Power consumption with fixed error rates

In these experiments, we assumed that the power supply voltage chosen for each error detection/correction scheme is based on the residual flit-error rate the system must support. We compared the power consumption of systems with parity-based encoding, CRC-based encoding, and hybrid single-error-correcting, multiple-error-detecting encoding with that of the original system (without error protection codes).

Our objective was to compare the error protection efficiency of various coding schemes, so we considered only end-to-end schemes in these experiments. We used a 4×4 mesh network with 16 cores and 16 switches. We assumed four flits in a packet and a 64-bit flit size. Figure 2 shows the network power consumption of the various schemes for an injection rate of 0.2 flits per cycle from each core and a uniform traffic pattern. The residual flit-error rates on the x-axis represent the systems' mean time to failure (MTTF). For example, a residual flit-error rate of 10^{-12} signifies that on average the system operates for 3.13^{11} cycles (assuming 16 cores, with each core injecting 0.2 flits per cycle, so that 10^{12} flits are generated in 3.13^{11} cycles) before an undetected error makes the system crash. For a 200-MHz system, this represents an MTTF of 26 minutes. For most applications, reasonable MTTF values would be on the order of months or years.

As Figure 2 shows, the original and ee-par schemes consume more power than the ee-crc and ec+ed schemes because the original and ee-par schemes have less error detection capability and hence require a higher operating voltage to achieve the same residual flit-error rate. The hybrid ec+ed scheme has lower power consumption at high residual flit-error rates, and the ee-crc has lower power consumption at lower residual error rates. The reason is that at high error rates, the retransmission of packets that had errors results in increased network traffic in the ee-crc scheme, which therefore consumes more power than the ec+ed scheme. At lower error rates, the error correction mechanism in the ec+ed scheme consumes more power than the retransmission mechanism in the ee-crc scheme. Also, the ec+ed scheme requires more bits for error correction and detection codes than the pure detection scheme.

Performance comparison with varying error rates

In the second set of experiments, we investigated the performance of pure end-to-end (ee) and switch-to-

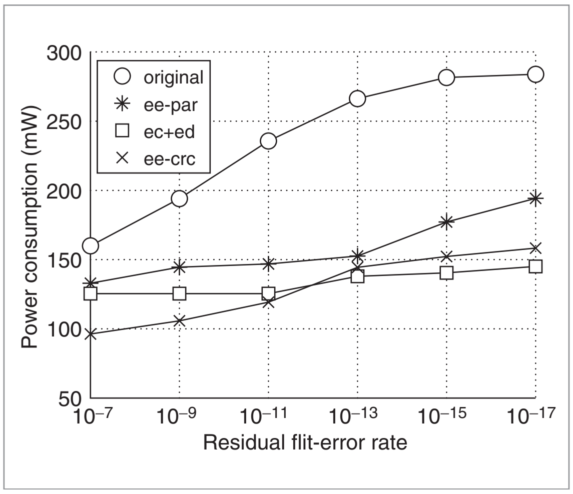

Figure 2. Power consumption of end-to-end error recovery schemes: original; with parity codes (ee-par); hybrid single-error-correcting, multiple-error-detecting (ec+ed); and with cyclic redundancy check codes (ee-crc).

switch (ssf, ssp) error detection schemes and the hybrid error detection/correction (ec+ed) scheme. We experimented on the 16-core mesh with varying injection rates for a uniform traffic pattern. Assuming that the system's operating voltage is fixed at design time (equal to 0.85 V), we investigated the effect of varying system error rates. We use the flit-error rate (the percentage of flits that have one or more errors) metric to define the system error rate. Note the difference between flit-error rate and residual flit-error rate, which we defined earlier. The latter represents the probability of having errors in a flit that remain undetected by the error recovery mechanism.

As Figure 3 shows, with a low flit-error rate and a low injection rate, the various schemes' average packet latencies are almost the same. However, as the error rate and/or the flit injection rate increases, the end-to-end (ee) retransmission scheme incurs a larger latency penalty than the other schemes. The packet-based switch-to-switch (ssp) retransmission scheme has a slightly higher packet latency than the flit-based switch-to-switch (ssf) retransmission scheme because the latter detects errors on packets earlier. As expected, the hybrid (ec+ed) scheme has the lowest average packet latency of the schemes.

Power consumption overhead

Table 1 presents the power consumption of a switch (with five inputs, five outputs, and $N_L = 2$), error detection/correction coders, and retransmission and packet buffers (for 50% switching activity at each component for each cycle). We assume a 200-MHz operating frequency, a 64-bit flit size, and a four-flit packet size. Here, we assume that the base NI power consumption (when there are no packet retransmission buffers) is part of processor and memory core power consumption because it is invariant for all schemes. To facilitate comparison of the various schemes, we analyzed the power overhead associated with error detection and recovery. The following definitions let us formulate analytical expressions for the schemes' power overhead:

- *inj_rate*—rate at which each NI injects traffic;
- N_{pb}—number of packet buffers required at each NI for retransmission in the ee and ec+ed schemes;
- *sw_traf*—rate of traffic injected at each switch;
- *sw_incrtraf*—increase in traffic at each switch resulting from retransmission in the ee scheme;
- $P_{packetsizeinc}$—total power overhead resulting from increase in packet size from added code words and other control bits.

Of these parameters, we obtain the traffic rates from and to the NIs and switches, and the traffic overhead for retransmission in the ee and ec+ed schemes from simulations. The physical implementation of the topology determines the link lengths. We obtain the number of packet buffers required in the ee scheme to support an application performance level from (possibly multiple sets of) simulations.

For simplicity of notation, in formulating power overhead, we represent parameters (such as traffic rate, link length, and buffering) as the same for all NIs and all switches. Also, we represent both dynamic and static power consumption with a single set of variables (see Table 1). We assume that scaling the power numbers on the basis of the traffic through the components causes only the dynamic power consumption part to scale.

The power overhead associated with the ee scheme is

$$P_{\text{overhead\_ee}} = \sum_{\forall \text{NIs}} \left[inj\_rate \times \left(P_{\text{crce}} + P_{\text{crcd}} + N_{\text{pb}} \times P_{\text{pb}} \right) \right] + \sum_{\forall \text{Switches}} \left(sw\_incrtraf \times P_{\text{sw}} \right) + P_{\text{packetsizeinc}}$$

This equation contains two major power overhead components: power overhead associated with the packet buffers at the NIs for retransmission, and power overhead resulting from increased power consumption from increased network traffic. For the ee scheme to work, we need sequence identifiers for packets and mechanisms to detect reception of duplicate packets. We consider the power consumption of look-up tables and the control circuitry associated with these mechanisms as part of the packet buffer power consumption (they typically increase packet buffer power overhead by 10%). Traffic increases in the ee scheme for two reasons:

- When the *ack* or *nack* signals are unable to be piggybacked to the source (for example, writes to memory locations normally don't require a response to the source), they must be sent as separate packets. In this case, an optimization is performable because an *ack* or *nack* packet need be only one flit long. Even with the optimization, we found that this overhead increases the total power consumption by 10% to 15%.
- At higher error rates, packet retransmission increases network traffic. Even at flit-error rates of 1%, however, we found that this increase has far less impact than the preceding case.

$P_{\text{packetsizeinc}}$ affects the schemes in almost the same manner (that is, the ssf scheme requires code bits on each flit, whereas the ee scheme requires additional information for packet identification, header flit protection, and packet code words). Therefore, this parameter has a lesser effect on the choice of scheme.

The power overhead of the ssf scheme is

$$\begin{aligned} P_{\text{overhead\_}ssf} = &\sum_{\forall\ \text{NIs}} \left(inj\_rate \times P_{\text{crce}} \right) \\ &+ \sum_{\forall\ \text{Switches}} \left[sw\_traf\left(P_{\text{crcd}} + P_{\text{srfb}} \left[2N_{\text{L}} + 1 \right] \right) \right] \\ &+ P_{\text{packetsizeinc}} \end{aligned}$$

The power consumption of the switch retransmission buffers is the major component of this overhead, and it depends linearly on the link lengths. We can easily derive the power overhead of the ssp and ec+ed schemes from the overhead equations for the ssf and ee schemes, respectively.

Figure 4 presents the network power consumption for the various error recovery schemes in the 16-core mesh network. We assumed the link length to be two cycles. We performed simulations with a uniform traffic pattern, with each core injecting 0.1 flits per cycle.

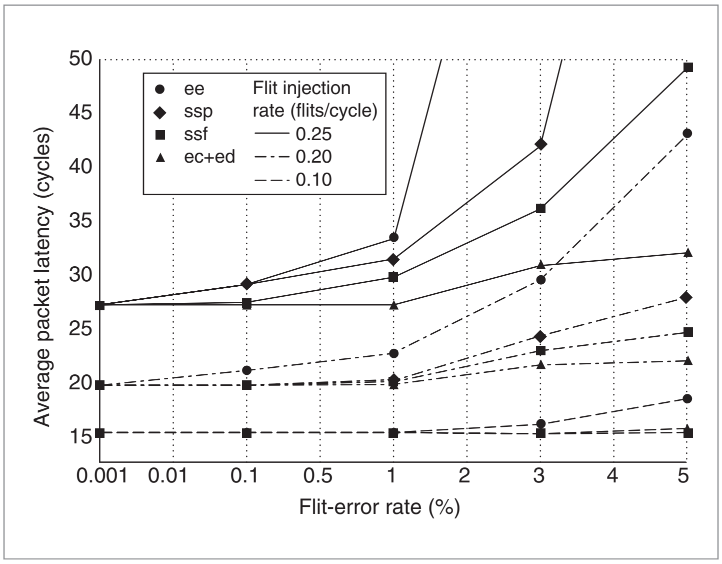

Figure 3. Packet latencies of error detection and correction schemes: pure end-to-end (ee), switch-to-switch packet level (ssp), switch-to-switch flit level (ssf), and hybrid (ec+ed).

Table 1. Component power consumption.

Component	Dynamic power (mW)	Static power (mW)
Switch (5 × 5)		
Buffers	13.10	1.69
Crossbar	4.57	NA
Control	1.87	0.02
Total (P_{sw})	19.54	1.71
Cyclic redundancy check (CRC) encoder (P_{crce})	0.12	NA
CRC decoder (P_{crcd})	0.15	NA
Single-error-correcting (SEC) encoder (P_{sece})	0.15	NA
SEC decoder (P_{secd})	0.22	NA
Switch retransmission flit buffer—1 flit (P_{srfb})	0.52	0.07
Packet buffer—1 packet (P_{pb})	2.29	0.31

For the ee and ec+ed schemes, we obtained the number of packet retransmission buffers required to support the application performance level (two packet buffers per NI) from simulations. This experiment shows that the power consumption of switch-based (ssf and ssp) error detection schemes is higher than that of end-to-end (ee and ec+ed) retransmission schemes. We attribute this to two factors:

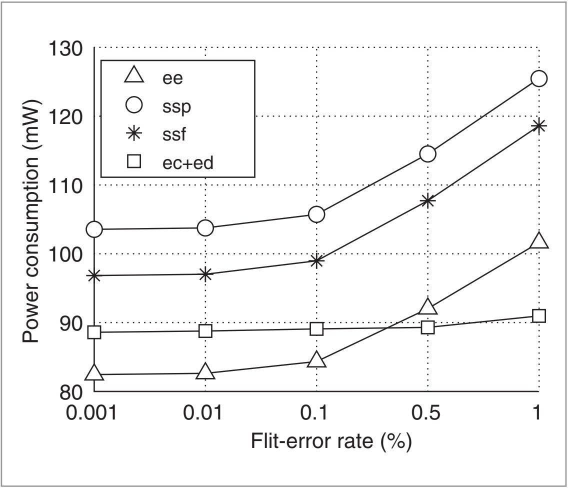

Figure 4. Power consumption of error recovery schemes (0.1 flits per cycle): pure end-to-end (ee), switch-to-switch packet level (ssp), switch-to-switch flit level (ssf), and hybrid (ec+ed).

Table 2. Effect of packet-buffering requirements (N_{pb}) on pure end-to-end error detection scheme ($N_L = 2$).

N_{pb}	Power consumption (mW)
1	75.5
2	84
3	93
4	102
5	111
6	120

Table 3. Effect of link length (N_L) on pure end-to-end (ee) and switch-to-switch flit-level (ssf) schemes. In the ee scheme, $N_{pb} = 2$.

	Power consumption (mW)	
N_L (cycles)	ee	ssf
1	65.12	59.24
2	84	97
3	102.8	134.76
4	121.76	172.52
5	141.22	216.52

- The buffering required for retransmission in the ssf and ssp schemes for this setup is larger than in the ee and ec+ed schemes.
- Because of the uniform traffic pattern, traffic through each switch is greater (since the average number of hops is higher), thus increasing ssf and ssp retransmission overhead.

We examine these points in detail in the following subsection.

Buffering requirements, traffic patterns, and packet size

A major power overhead for the schemes is the amount of packet and switch buffering required for retransmission. To see the impact of buffering requirements, we performed experiments on the mesh network, varying the number of packet buffers and link lengths (and hence the number of retransmission buffers for the ssf scheme). Tables 2 and 3 show the results. For small link lengths and large packet-buffering requirements of the ee scheme, the ssf scheme is more power efficient than the ee scheme. On the other hand, when the link lengths are large, the ee scheme is more power efficient. But when link lengths are short and packet-buffering needs are small, it is difficult to generalize about the schemes' efficiency. However, if the parameters (link length, packet-buffering needs, and so forth) are obtained from user input and simulations, we can feed them into the power consumption equations described earlier to compare the schemes.

Another important parameter that affects the choice of scheme is the application traffic characteristics. To observe the impact of various traffic scenarios, we performed experiments varying the average hop delay for data transfer. Figure 5 shows the power overhead of the ee and ssf schemes (assuming $N_{pb} = 2$ and $N_L = 2$) for the different scenarios. In the figure, an average hop count of 2 corresponds to a neighbor traffic pattern, and other hop delay numbers can be interpreted as representing other traffic patterns. As the average hop count for data transfer increases, the ssf power overhead increases rapidly because more traffic passes through each switch, thereby consuming more power in the switch retransmission buffers. Thus, for traffic flows traversing a larger number of hops or for large networks, switch-to-switch retransmission schemes incur a large power penalty.

Figure 6 compares power consumption of packet-based (ssp) and flit-based (ssf) schemes with a varying number of flits per packet. In this experiment, we assume that the packet size (256 bits) remains constant, and we vary the number of flits per packet. As the number of flits per packet increases, the packet-based scheme's buffering needs increase, so its power consumption increases rapidly. The flit-based scheme also incurs greater power

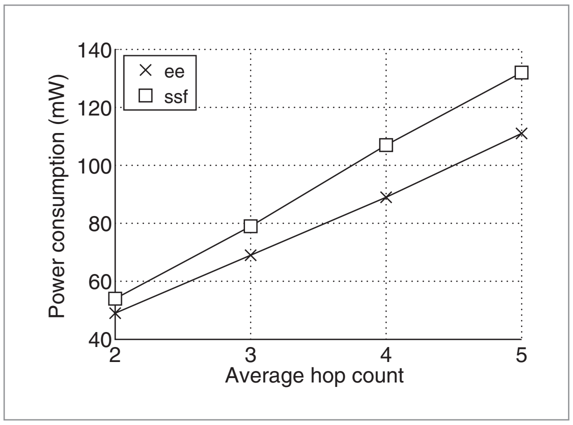

Figure 5. Effect of hop count on pure end-to-end (ee) and switch-to-switch flit-level (ssf) error detection schemes (N_{pb} = 2, N_L = 2).

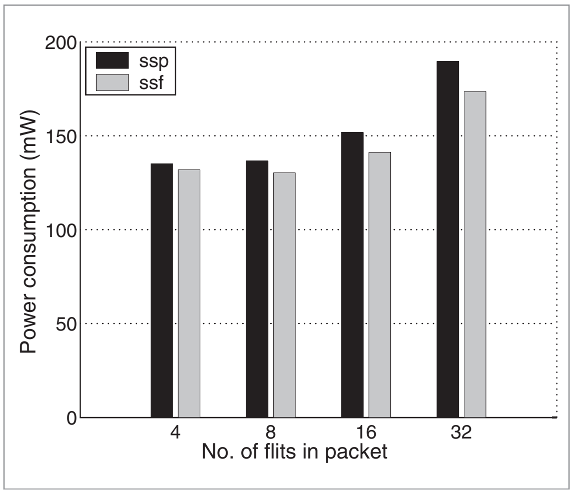

Figure 6. Power consumption of packet-based (ssp) versus flit-based (ssf) schemes.

consumption with increasing flits per packet because the ratio of useful bits to overhead bits (the CRC code bits) decreases as flits per packet increases. However, for reasonable flit sizes, we found that the flit-based scheme is more power efficient than the packet-based scheme.

Table 4 presents the areas of NoC components (switches and additional hardware for error recovery) for the various schemes in the 16-node mesh network (with $N_{pb} = 2$ and $N_L = 2$). The schemes' area overheads are comparable.

Table 4. NoC area for various schemes: original; pure end-to-end (ee); switch-to-switch flit-level (ssf); and hybrid (ec+ed). N_{pb} = 2 and N_L = 2.

Scheme	Area (mm²)
Original	3.36
ee	4.40
ssf	5.76
ec+ed	5.30

For the ee and ec+ed schemes, the major power overhead components are the packet-buffering needs at the NIs and the increase in network traffic caused by *ack*/*nack* packets. For the ssf and ssp schemes, the major power overhead results from the retransmission buffers that are required at the switches. Design methodologies that trade application performance for buffering needs would result in less power overhead. For example, exploring queuing theory methods to design the buffers is a promising research direction. Methods that reduce *ack*/*nack* traffic (such as multiple packets sharing a single *ack*/*nack* signal) are also promising. Another avenue is to explore mechanisms that reduce the control overhead associated with duplicate packet reception in the ee scheme.

Our experiments show that for networks with long link lengths or hop counts, end-to-end detection schemes are power efficient. Switch-level detection mechanisms are power efficient when link lengths are small and when the end-to-end scheme requires large packet buffering at the NIs. At low error rates, all the schemes incur similar latencies. At higher error rates, the hybrid error detection and correction mechanism provides better performance than the other schemes. Because the ee scheme uses a subset of the hardware resources used for the ec+ed scheme, the error correction circuitry can be selectively switched on and off, depending on the system's prevailing error rates. For hierarchical networks, switch-based error control can be implemented for local communication, and end-to-end error control can be implemented for global communication (which traverses longer links and hop counts).

Further work in NoCs should include investigating the effects of application- and software-level reliability schemes and developing online adaptation capabilities, such as reconfigurable designs for error resilience. ■

Acknowledgments

This research is supported by the Marco Gigascale Systems Research Center and the US National Science Foundation under contract CCR-0305718.

References

1. N.R. Shanbhag, "A Mathematical Basis for Power-Reduction in Digital VLSI Systems," *IEEE Trans. Circuits and Systems*, part II, vol. 44, no. 11, Nov. 1997, pp. 935-951.
2. "The Nostrum Backbone," http://www.imit.kth.se/info/FOFU/Nostrum/.
3. E. Rijpkema et al., "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip," *Proc. Design, Automation and Test in Europe Conf.* (DATE 03), IEEE Press, 2003, pp. 10350-10355.
4. P. Guerrier and A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections," *Proc. Design, Automation and Test in Europe Conf.* (DATE 00), IEEE Press, 2000, pp. 250-256.
5. L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, no. 1, Jan. 2002, pp. 70-78.
6. R. Hegde and N.R. Shanbhag, "Towards Achieving Energy Efficiency in Presence of Deep Submicron Noise," *IEEE Trans. VLSI Systems*, vol. 8, no. 4, Aug. 2000, pp. 379-391.
7. W.J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2003.
8. H. Wang, L.-S. Peh, and S. Malik, "Power-Driven Design of Router Microarchitectures in On-Chip Networks," *Proc. 36th Ann. Int'l Symp. Microarchitecture* (Micro 36), IEEE Press, 2003, pp. 105-116.
9. S. Mitra et al., "Robust System Design from Unreliable Components," 2005; http://www.cse.psu.edu/~yuanxie/isca05.pdf.
10. D. Bertozzi, L. Benini, and G. De Micheli, "Low Power Error Resilient Encoding for On-Chip Data Buses," *Proc. Design, Automation and Test in Europe Conf.* (DATE 02), IEEE Press, 2002, pp. 102-109.

Srinivasan Murali is pursuing a PhD at Stanford University. His research interests include mapping of applications onto NoC architectures, multiprocessor SoCs, communication architecture design, and SoC reliability. Murali has a BE in computer science and engineering from the University of Madras, India. He is a student member of the IEEE.

Theocharis Theocharides is a PhD candidate in the Computer Science and Engineering Department of Pennsylvania State University. His research interests include SoC design and interconnects, SoC application mapping, low-power and reliable VLSI design, and application-specific processors. Theocharides has a BSc in computer engineering from Pennsylvania State University. He is a student member of the IEEE and a Fulbright scholar.

N. Vijaykrishnan is an associate professor in the Computer Science and Engineering Department of Pennsylvania State University. His research interests include energy-aware reliable systems, embedded Java, nano/VLSI systems, and computer architecture. Vijaykrishnan has a PhD in computer science and engineering from the University of South Florida. He is a member of the IEEE and the IEEE Computer Society.

Mary Jane Irwin is the A. Robert Noll Chair in Engineering in the Department of Computer Science and Engineering of Pennsylvania State University. Her research and teaching interests include computer architecture, embedded and mobile computing system design, power-aware design, EDA, and emerging technologies in computing systems. Irwin has a PhD in computer science from the University of Illinois. She is a Fellow of the IEEE, a Fellow of the ACM, and a member of the National Academy of Engineering.

Luca Benini is an associate professor in the Department of Electrical Engineering and Computer Science of the University of Bologna. His research interests include all aspects of CAD of digital circuits, with special emphasis on low-power applications and portable-system design. Benini has a PhD in electrical engineering from Stanford University. He is a senior member of the IEEE.

The biography of **Giovanni De Micheli** appears on p. 403 of this issue.

■ Direct questions and comments about this article to Srinivasan Murali, Stanford University, Gates Computer Science Bldg., Room 330, 353 Serra Mall, Stanford, CA 94305; smurali@stanford.edu.