Comparison of a Timing-Error Tolerant Scheme with a Traditional Re-transmission Mechanism for Networks on Chips

Srinivasan Murali[&], Rutuparna Tamhankar[‡], Federico Angiolini[†], Antonio Pullini[†], David Atienza[#], Luca Benini[†], Giovanni De Micheli[#]

[&] CSL, Stanford University, Stanford, USA

[‡] SUN Microsystems, Mountain View, USA

[†] DEIS, University of Bologna, Bologna, Italy

[#] LSI, EPFL, Lausanne, Switzerland

Abstract

On-chip wires are becoming unreliable as the effect of various noise sources increases with technology scaling. This leads to unpredictable timing delay variations on the interconnect wires. There is a significant need to mitigate the effect of parasitics on the interconnects, while keeping performance and area overheads at a minimum. In this work, we present a timing error tolerant design methodology, T-error, that provides dynamic recovery from timing delay variations on the interconnects. We validate the functionality of the T-error methodology using cycle-accurate RTL models of a Network-on-Chip (NoC) design, that are integrated onto a multiprocessor virtual platform. Our comparisons with the state-of-the-art error recovery mechanisms show that the T-error system provides error recovery with higher performance than the existing schemes. We also present the synthesis results for the T-error scheme, which show that the scheme has negligible overhead.

Keywords: Networks on Chips, Systems on Chips, timing errors, retransmission, double sampling

1 Introduction

With technology scaling, the communication complexity of *Systems-on-Chip (SoCs)* is rapidly increasing. To tackle the resulting complexity, *Networks-on-Chip (NoCs)* have emerged as the paradigm for designing scalable communication architecture for SoCs [1]-[5].

Another effect of *Deep Sub-Micron (DSM)* technologies is the appearance of significant delay variations on the wires. Wires are becoming thicker and taller, but their widths are not increasing proportionally, thereby increasing the effect of coupling capacitance on the delay of wires. As an example, the delay of a wire can vary between τ and $(1 + 4\lambda)\tau$ (where τ is the delay of the wire without any capacitive coupling and λ is the ratio of the coupling capacitance to the bulk capacitance) [9]. The wire delay for data transfers on a communication bus depend on the data patterns that have to transferred. As presented in [7], the data-dependent variations in wire delay can be as large as 50% for different switching patterns. With technology scaling, the device characteristics fluctuate to a large extent due to process variations and can cause significant variations in wire delay [8]. Wire delay is also affected by other forms of interference, such as supply bounce, transmission line effects, *etc.* [8].

The major effect of these noise sources is that the delay incurred by the data traversing the interconnect becomes unpredictable, causing timing delay violations and timing errors on the interconnects. There is a significant need to mitigate the effect of parasitics on link performance. In most state-of-the art NoCs, when such errors are detected, the packets that incurred errors are retransmitted [18]-[21]. However, retransmission of data incurs significant performance penalties. Moreover, timing delay variations due to the noise sources can potentially affect multiple data bits in a packet, requiring complex multi-bit error detecting/correcting codes that are of impractical use [21].

In this work, we present a *Timing-error tolerant* design methodology, *T-error*, that makes the NoC design tolerant to timing errors caused by the unpredictability in the environment and wire characteristics. In the *T-error* scheme, a double data sampling technique is used to recover from timing errors in the NoC. The double data sampling technique has been widely used by several researchers for general purpose processor designs [10]-[15]. In this work, we integrate the basic double sampling technique with the network buffers and the link level flow control protocol used in the NoC. The resulting system can detect and correct timing errors dynamically, with minimum performance and area penalty.

In the proposed design methodology, the normal FIFOs used in the network components (links, switches and network interfaces) of the NoC are replaced by the *T-error* FIFOs, which can be designed and used as library elements. Thus, adding support for timing error recovery in existing NoC systems using the proposed approach requires only a marginal amount of design effort. To the best of our knowledge, the *T-error* methodology is the first work that addresses the issue of dynamic timing error resiliency in NoCs using the double data sampling technique. The *T-error* scheme is explained in detail in Section 2.

The basic approach of the *T-error* design methodology

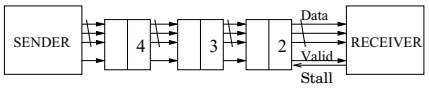

Figure 1. Example NoC system with pipelined links

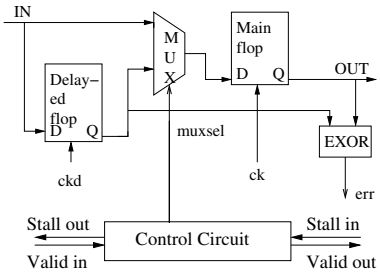

Figure 2. The 2-entry *T-error* FIFO

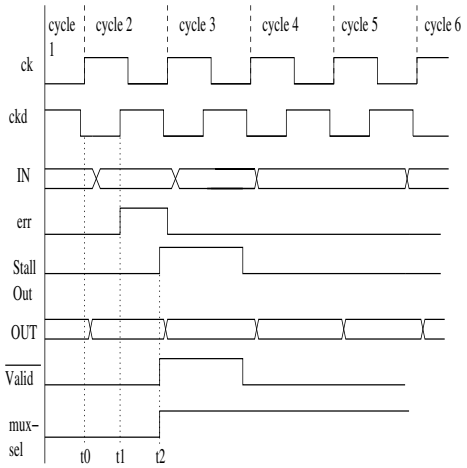

Figure 3. Example waveforms

as applied to network links has been presented by us in [22] and a comparison with other flow control schemes has been presented in [24]. In this work, we validate the performance and area overhead of the methodology by developing cycle-accurate SystemC models of the *T-error* network components, and incorporate them within the $\times$pipes NoC architecture [16]. To obtain results on typical SoC benchmarks, the *T-error* based NoC is integrated into MPARM [23], a general multiprocessor virtual platform.

We perform functional cycle-accurate SystemC simulations on an image processing benchmark application to validate the performance of the scheme. We also run experiments on state-of-the-art NoC error recovery schemes, based on retransmission of data in case of errors, which show that the *T-error* scheme is faster. We also present synthesis results for the *T-error* NoC, which show that the area overhead incurred by the scheme is negligible.

We would like to state here that the *T-error* methodology only targets the recovery from timing errors, which however constitute a major portion of the faults that may be encountered in NoCs. To tolerate other kinds of errors (such as soft errors), mechanisms presented in several other research works (such as [17]-[21]) should be used in conjunction with the *T-error* methodology.

2 T-error Scheme

The basic operation of the *T-error* scheme is explained in this section. In order to obtain high throughput, we realistically assume that the long links in the baseline error-prone NoC are pipelined and have 2-entry FIFOs at each pipeline stage (see Figure 1) [6]. In the *T-error* scheme, the 2-entry FIFOs are modified to support timing error tolerant operation; the FIFOs in the switches and network interfaces of the NoC are also modified in a similar manner.

The modified 2-entry FIFO structure is shown in Figure 2. The second flip-flop of the FIFO (called *delayed flip-flop*) is clocked at a delayed clock (*ckd*) when compared to the clock *ck* of the *main flip-flop*. Both clocks *ck* and *ckd* have the same period, but they are phase shifted. Note that in this configuration, the data has more time to settle if sampled by the *delayed flip-flop* when compared to being sampled by the *main flip-flop*. The phase shift between the clocks is configured after proper delay analysis, as will be discussed later in this section.

There are two modes of operation at each pipeline stage of the link: *main mode* and *delayed mode*. Initially all the pipeline FIFOs are set to the *main mode* and data transmission begins. In every cycle, at the rising edge of clock *ck*, the *main flip-flop* captures and transmits the incoming data (time instant t_0 in Figure 3). At the rising edge of clock *ckd*, the *delayed flip-flop* does a second sampling of the same incoming data (time instant t_1). At this point, a detection circuit checks whether there is any difference between the *main* and the *delayed flip-flop* values, *i.e.* detects a timing error. As shown in Figure 2, an *EXOR* gate is connected to the outputs of the *main flip-flop* and *delayed flip-flop* to detect such occurrences. The *err* signals of all w bits of the link (vertically across the width of the link) at a pipeline stage are *ORed* and fed as an input to the control circuit. A timing error in any bit of the data word triggers the control circuit to toggle the *muxsel* signal; the correct data, as sampled by the *delayed flip-flop*, is therefore fed to the *main flip-flop* on the next rising edge of clock *ck* (time instant t_2), and properly propagated. The correct version of the whole data word is thus sampled at the next pipeline stage.

Whenever a timing error is detected (*i.e.* the *err* signal is raised), a *stall* signal is sent to the previous stage, so that the previous stage suspends data transmission for one cycle while the error is handled. Also, a $\overline{valid}$ signal is sent to the following stage, informing that the data sent in the previous cycle was faulty and that a new, correct copy is now incoming.

A FIFO at a pipeline stage of the link enters the *delayed mode* when a *stall* signal from the next stage causes queuing of data at the FIFO. The *stall* signal can be issued to handle regular congestion, that is as a flow control wire, or to let the downstream stage sort out an error condition. When a FIFO is in *delayed mode*, all timing errors are automatically avoided, as the incoming data is always sampled through the *delayed flip-flop*. Thus, in networks with congestion, most timing errors are automatically avoided. Examples of operation of the FIFOs for a network with no congestion and with congestion are presented in Figures 4, 5. In the network with no congestion, at each pipeline stage, data is always directly sampled by the *main flop-flop* and sent out by it. In the network with congestion, the data from the preced-

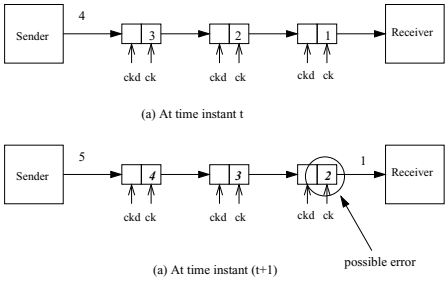

Figure 4. Network operation without congestion

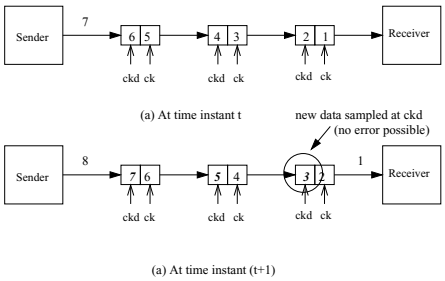

Figure 5. Network operation under congestion

ing pipeline stage is always captured by the *delayed flip-flop* at the current pipeline stage, and later sent out by the *main flop-flop*. Since data is always sent at *ck* from the preceding stage and sampled at *ckd* in the current stage, the wire transitions have more than one clock period to settle and thus timing errors are automatically avoided. In the worst case, if the FIFO always operates in the *main mode*, each timing error occurrence will incur one clock cycle penalty for recovery.

The amount of timing delay that is tolerated by the *T-error* design depends on the phase shift between the clocks of the *main* and the *delayed flip-flops*. This shift should be as large as possible, so that the *delayed flip-flop* is guaranteed to sample the right data and to provide correct system operation. However, the maximum shift is constrained by internal repeater delays (the error detection logic must operate between a *ckd* edge and the following *ck* edge). Detailed timing analysis and SPICE simulations (for a link size of 32 bits) showed that clock *ckd* can be delayed by 53.3% of the clock period with respect to *ck*. In this work, we assume that a maximum delay of 50% of the clock is tolerable with a *T-error* enabled system. Thus, the delayed clock *ckd* is just the inverted value of the main clock, and delay chains are not needed to generate it. At the same time, the maximum delay which is tolerated on a wire is 150% of the clock period, providing ample margin for timing error correction. We refer the interested reader to [22] for transistor-level implementation details, timing analysis and SPICE simulation results of the *T-error* scheme.

3 Experimental Results

3.1 Comparisons with Retransmission Scheme

We now compare the performance of systems that use traditional retransmission mechanisms (we assume *switch-to-switch* retransmission) for handling timing errors against the *T-error* scheme. To evaluate the designs, we define a new metric: *Potential Error-Rate (PER)*. The PER represents the percentage chance that a data word reaching a FIFO incurs one or more timing errors, if the data is sampled directly on the *ck* rising edge. Note that in *T-error*, in most scenarios, the data is sampled first by the *delayed flip-flop* and only later by the *main flip-flop*; for example, this occurs under congestion, or whenever a string of incoming data blocks follows a faulty transmission. This automatically avoids any potential errors. Therefore, even with a PER of 100%, the actual errors happening at the *T-error* FIFO can be few, as most of the faults after the first error are transparently corrected by the design.

We first present an experiment where a serial interconnect (of 1-bit data width) is used for the links. In this experiment, the data bits are sent between two switches and links with two pipeline stages each. This models the nearest neighbor traffic pattern, which is typical for SoCs [21]. In Figure 6, the latency for data transmission for two different PER values (1% and 5%) is analytically plotted for various data sizes for the *T-error* based design and a traditional design where errors are corrected by retransmission. These PER values are reasonable estimates for the error rates in future interconnects, which are obtained from [21].

The *T-error* scheme incurs negligible latency penalties and the plots for both 1% and 5% PER values for the scheme overlap. This is due to the above mentioned fact that most of the faults after the first error are transparently corrected by the design. For a chosen PER value, as the size of data to be transferred increases, there are significant latency savings in the *T-error* system when compared to the traditional scheme of retransmission. Moreover, as the error rate starts to increase, there are much larger savings in latency for the *T-error* based system. For the data size of 1000 bits and a potential error rate of 5%, there is a 35% reduction in latency in the *T-error* based system when compared to the retransmission scheme.

In the second experiment, we leverage upon a full NoC platform. The simulation platform consists of cycle-accurate SystemC models of the *T-error* designs for the switches, links and network interfaces, incorporated within the ×pipes architecture. We use the *MPARM* simulation environment [23], which allows several interconnect structures (such as *AMBA, STBus, ×pipes*) to be utilized to connect processor/memory cores, and has support for a variety of benchmark applications. A functional SystemC simulation is carried out on an image processing benchmark (referred to as *MAT2*). For illustrative purposes, we assume that the retransmission mechanism is capable of detecting all timing errors; in reality, all the data bits can have timing errors and such a scheme may not even be applicable. The percentage increase in application runtime for different PER values for the retransmission scheme when compared to the *T-error* scheme is presented in Figure 7. Please note that the application execution time includes the computational time as well. For a PER value of 5%, the retransmisson scheme when compared to the *T-error* scheme incurs an 8% increase in total application runtime.

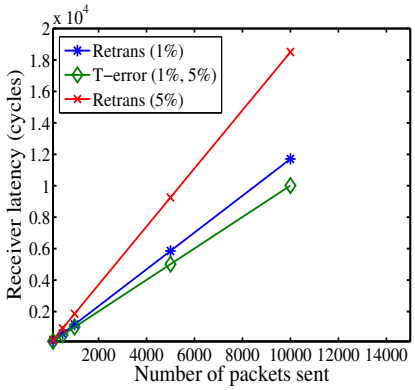

Figure 6. Comparisons for serial links

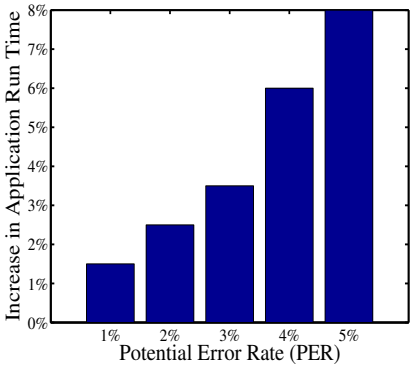

Figure 7. Comparisons for *MAT2* benchmark

Table 1. Area Overhead

Design	Area (mm^2)
Base NoC	4.9
T-error Scheme NoC	5.1

3.2 Synthesis Results

We synthesize with Synopsys Design Compiler the *T-error* based NoC design to get area estimates. For synthesis, we use a UMC 0.13μ technology library, a NoC operating frequency of 1 GHz and an operating voltage of 1.2 V. Table 1 shows the area overhead for the *T-error* scheme, for a 32-bit 5×5 mesh NoC. The base NoC area is the sum of the area of switches, links and network interfaces without the *T-error* design changes. As seen from the table, the *T-error* scheme incurs a negligible increase in area of the NoC.

4 Conclusions

Robust design methods are needed to cope with the increasing timing uncertainty on the interconnects. It is important to achieve a robust design with minimum performance and area overhead. In this work, we present a timing error tolerant design methodology, *T-error*, that provides dynamic recovery from timing errors. We implement the *T-error* scheme using cycle-accurate RTL models and present the simulation and synthesis results for the design. Experimental comparisons with state-of-the-art error recovery mechanisms show that *T-error* provides large performance improvements with minimum area overhead. As future work, we plan to apply the timing error tolerant scheme to the computation architecture.

5 Acknowledgments

This work is supported by the US National Science Foundation (NSF, contract CCR-0305718) for Stanford University. It is also supported by the Swiss National Science Foundation (FNS, Grant 20021-109450/1) and the Spanish Government Research Grant TIN2005-5619. The work is also supported by a grant from Semiconductor Research Corporation (SRC project number 1188) and a grant by STMicroelectronics for DEIS.

References

[1] S. Kumar et al., "A Network on Chip Architecture and Design Methodology", Proc. ISVLSI, pp. 117-122, April 2002

[2] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections", Proc. DATE, pp. 250-256, March 2000.

[3] D. Siguenza-Tortosa, J. Nurmi, "Proteo: A New Approach to Network-on-Chip", in CSN 02, Sep. 2002.

[4] K. Goossens et al., "The Aethereal network on chip: Concepts, architectures, and implementations", IEEE Design and Test of Computers, Vol. 22(5), pp. 21-31, Sept-Oct 2005.

[5] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, Vol. 35(1), pp. 70-78, Jan. 2002.

[6] L. Carloni et al., "Theory of Latency-Insensitive Design", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20(9), pp. 1059-1076, Sept 2001.

[7] Y. Eo et al., "A Decoupling Technique for Efficient Timing Analysis of VLSI Interconnects with Dynamic Current Switching", IEEE Transactions on CAD , Vol. 23(9), pp. 1321-1337, Sept. 2004.

[8] D. Wang, W. McNall,"A statistical model based ASIC skew selection method", IEEE Workshop on Microelectronics and Electron Devices, pp. 64-66, 2004.

[9] P. Sotiriadis, "Interconnect Modeling and Optimization in Deep Sub-Micron Technologies", Ph.D dissertation, Massachusetts Institute of Technology, 2002.

[10] D. Ernst et al., "Razor: A low-power pipeline based on circuit-level timing speculation", Proc. of the International Symposium on Microarchitecture, pp. 7-18, Dec 2003.

[11] A. Uht, "Going Beyond Worst-Case Specs with TEAtime", IEEE Computer, Vol. 37(3), pp. 51-56, March 2004.

[12] M. Favalli, C. Metra, "Low-level error recovery mechanism for self-checking sequential circuit", Proc. DFT, pp. 234-242, Oct 1997.

[13] M.Singh, S. M. Nowick, "MOUSETRAP: Ultra-High-Speed Transition Signaling Asynchronous Pipelines", Proc. ICCD, pp. 9-17, Sept 2001.

[14] E. Dupont et al., "Embedded Robustness IPs for Transient Error Free ICs", EEE Design and Test of Computers, Vol. 19(3), pp. 56-70, May 2002.

[15] W. J. Dally, J. W. Poulton, "Digital Systems Engineering", Cambridge University Press, 1998.

[16] S. Stergiou et al., "xpipes Lite: A Synthesis Oriented Design Library For Networks on Chips", Proc. DATE, pp. 1188-1193, March 2005.

[17] R. Hegde, N. R. Shanbhag, "Toward Achieving Energy Efficiency in Presence of Deep Submicron Noise", IEEE Transactions on VLSI Systems, Vol. 8(4), pp. 379-391, August 2000.

[18] D. Bertozzi et al., "Low Power Error-Resilient Encoding for On-Chip Data Buses", Proc. DATE, pp. 102-109, March 2002.

[19] P. Vellanki et al., "Quality-of-Service and Error Control Techniques for Network on Chip Architectures", Proc. GLSVLSI, pp. 45-50, April 2004.

[20] H. Zimmer, A. Jantsch ,"A Fault Model Notation and Error-Control Scheme for switch-to-Switch Buses in a Network-on-Chip", Proc. ISSS/CODES, pp. 188-193, Sept 2003.

[21] S. Murali et al., "Analysis of Error Recovery Schemes for Networks-on-Chips", IEEE Design and Test of Computers, Vol. 22(5), pp. 434-442, Sep-Oct 2005.

[22] R. Tamhankar et al., "Performance Driven Reliable Link Design for Networks on Chips", Proc. ASPDAC, pp. 749-754, Jan 2005.

[23] M. Loghi et al., "Analyzing On-Chip Communication in a MPSoC Environment", Proc. DATE, pp. 752-757, Feb 2004.

[24] A. Pullini et al., "Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes", Proc. SBCCI, pp. 224-229, Sep 2005.