Timing-Resilient Network-on-Chip Architectures

Alexandros Panteloukas*, Anastasios Psarras*, Chrysostomos Nicopoulos† and Giorgos Dimitrakopoulos*
*Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece
†Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus

Abstract—**Networks-on-Chip (NoC) have been established as the de facto standard for on-chip communication in multi-/many-core systems, due to their innate scalability properties pertaining to performance and physical implementation. Spanning the entire chip, the NoC suffers from both inter-die and intra-die variations. In addition to static variability, the NoC is also afflected by dynamic variations, such as fast VDD droops, temperature, and aging effects. Such variations cause unpredictable behavior in the timing characteristics of the NoC components, which need significant timing margins to ensure always-correct operation. Consequently, the potential for high-frequency operation is impeded. In this paper, we propose a timing-error-resilient mechanism called TRNoC, which allows the NoC to operate at higher frequencies, at the expense of sporadically experiencing run-time timing errors, which are handled by an error recovery strategy. TRNoC includes a lightweight timing-error detection mechanism, together with a distributed error recovery mechanism, which allow for lossless operation, while leaving the error-free parts of the network unaffected. Hardware implementation results demonstrate the efficiency of TRNoC and its potential as a scalable timing-error-resilient NoC architecture.**

I. Introduction

Process variability in device and circuit parameters is one of the primary challenges currently faced by the semiconductor industry [1]. Besides static variations that occur during chip fabrication, dynamic parameter variation – resulting from environmental and workload changes – is also possible during the chip's operation. Examples of dynamic variations include supply voltage droops, temperature changes, and transistor aging degradation. Variations change the circuit's characteristics in terms of timing and power consumption, and, thus, if not appropriately handled, they may adversely impact performance, power, and the system's overall reliability.

At the same time, technology scaling and power constraints have led to a fundamental paradigm shift in digital system design: the transition to the multi-core archetype, whereby several processing elements are employed to enhance performance. Consequently, the role of the on-chip communication fabric has become pivotal to the system's operation. Packet-based Networks-on-Chip (NoC) have been established as the dominant communication backbone in multi-core environments, primarily due to their innate scalability attributes with respect to performance and physical implementation traits [2]. NoC components are distributed throughout the chip, thereby experiencing both inter- and intra-die variations, while also being sensitive to any local variation effects arising from VDD droops and temperature increases.

Conventional synchronous digital systems require substantial timing guard-bands to ensure proper operation across manufacturing and environmental variations. Circuits usually run at a clock frequency that covers all potential delay variations under any possible operating conditions. However, a significant part of the cycle period is often wasted, due to the conservative margins used to cover these variations.

To mitigate this limitation, previous work proposed to reduce these margins by employing in-situ timing-error detection and recovery, where occasional timing failures are detected and corrected by re-playing/re-executing the failed operation [3], [4]. With this approach, the operating clock frequency can be kept higher, with significant overall performance benefits. In fact, even power savings are possible, when timing resiliency is combined with dynamic voltage scaling, which allows scaling the voltage to the minimum point of correct operation [4].

In this paper, motivated by the same design philosophy, we derive a low-cost timing resiliency mechanism tailored to NoC architectures. The proposed approach, named *TRNoC*, is characterized by the following salient attributes:

- It allows for independent per-input and per-output error handling, which is confined only to the *control path* of the NoC routers.
- It guarantees no packet/flit loss and provides full protection against single and/or bursty (back-to-back) timing errors in a unified manner.
- It offers error isolation. The components of the NoC that are unaffected from timing errors continue to work correctly, with no performance impact.

TRNoC carefully identifies the specific timing paths that should be protected, and the appropriate timing-error-detection mechanism that best fits each path. Without due diligence, a naive application of state-of-the-art timing-error tolerant mechanisms – such as Razor [4], TIMBER [5], or time dilation [6] – to the NoC will incur significant clock-frequency overhead, due to the necessary propagation of error information for eliminating the switching of invalid (erroneous) data.

The experimental results presented in Section V – using standard-cell-based hardware synthesis implementations – validate the efficiency of TRNoC. The proposed new features work in unison to facilitate the design of scalable NoC architectures that can operate at high clock frequencies. More importantly, with minimum area and delay overhead, TRNoC ensures timing-error-resilient operation within the NoC, even when timing guard-bands are reduced to a minimum.

Timing-error tolerant NoCs have been explored in the past, covering either the avoidance of timing errors, or detection and recovery, but only on specific parts of the NoC (e.g., the links). For example, SturdiSwitch [7] employs an array of micro-architectural tweaks to increase the resilience of the NoC router to process-variation-induced timing violations, but remains merely proactive; once a timing error actually occurs, SturdiSwitch cannot detect it. On the contrary, TRNoC takes a reactive approach, which guarantees complete protection from timing errors. In [8], the internal paths of the NoC routers are exercised on purpose, in order to keep them "safe" from aging effects that may appear due to low utilization of the NoC. Focusing only on the links of the NoC, the work in [9] derives elastic flow-controlled buffers that detect and recover from link-timing errors. Also, the work in [10] discards the slow wires due to variations, and uses only the remaining wires, which can work at the nominal clock frequency. In [11], timing self-calibration is allowed on the links. Other closely related fault-tolerant architectures enhance the flow control mechanism to guarantee lossless operation in the presence of faults [12] [13], while in [14], self-checking NoC components have been designed. TRNoC focuses on holistic protection against variation-induced timing errors, which are handled in a fine-grained manner. More importantly, the error-free parts of the NoC remain completely unaffected.

II. NoC Landscape & Basic Concepts

A typical NoC consists of a connected set of network interfaces and network routers, which allow for packet-based communication across the Intellectual Property (IP) cores of the system; an example is shown in Figure 1. Network Interfaces (NI) communicate with the attached IP cores and convert IP transactions into transport-layer packets. The latter are subsequently delivered to their destination using the network routers, which handle switching and routing, as well as link-level lossless flow control.

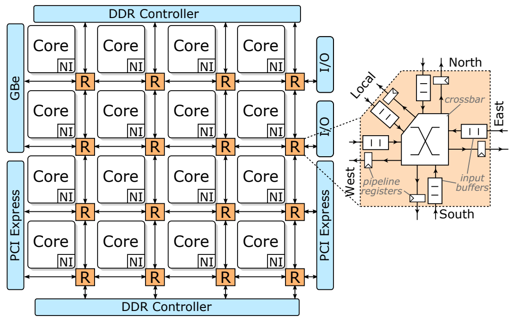

Fig. 1. A typical multi-core system connected with a Network-on-Chip.

Data travels within the network as packets comprising one head flit, a number of body flits, and a tail flit that marks the end of the packet. Flits flow on the links between connected routers using credit-based flow control. The sender (upstream router) keeps track of the available buffer slots at the receiver's side of the link (downstream router) using a credit counter. When the number of credits is larger than zero, the sender is allowed to send a new flit after consuming one available credit (credit counter is decremented). When a buffer slot becomes empty at the receiver, the sender is notified via a credit-update signal to increase the available credit count. No flit can be dropped, since each flit reaches the receiver after having consumed the credit associated with a free buffer slot.

Flit (packet) buffering and switching is performed in NoC routers. The block diagram of a wormhole NoC router is shown in Figure 2. Each packet first completes Routing Computation (RC) (using its head flit), which identifies the router's output port that the packet must exit from, in order to get closer to its destination. Then, each packet should arbitrate for access to the specified output port via Switch Allocation (SA), after generating an eligible request in the Request Generation (RG) stage, which checks the output port's availability and the existence of credits in the downstream router. Once a packet wins SA, its flits move (one per cycle) to the appropriate output port via the multiplexers of the crossbar (Switch Traversal, ST), after consuming one credit per flit (Credit Calculation, CC). Eventually, each flit will reach the next router, after leaving the output port and crossing the link (Link Traversal, LT). Since, the flits of one packet should traverse each link as an atomic entity (packet mixing is not permitted). the link is allocated to the packet as a whole: after the head flit of the packet wins SA, it locks access to the output port (State Update, SU), which is later released by the tail flit of the packet. Both SU and CC are executed independently per output port, and in parallel to SA, by checking if at least one eligible request is asserted for the specific output port.

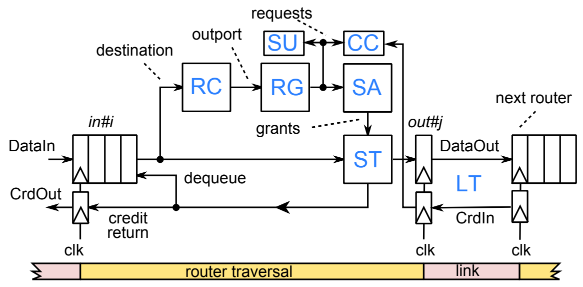

Fig. 2. The micro-architecture of a typical wormhole NoC router. Only one input-output port pair is shown here for clarity.

The timing paths of the router start from the head-of-line position of the input buffer and fork to different directions. The first one ends early at the CC and SU logic, while the remaining three paths pass through SA and follow the distribution of the grant signals. The grant signals produced by SA play a triple role: (a) they drive the select-signals of ST, which switch the flits of the selected input ports to the appropriate output ports; (b) they are used per-input-port to dequeue the winning flits out of their input buffers; (c) they are used in the same cycle to prepare a credit-update signal, which is sent upstream in the next cycle to inform the previous router that a buffer slot has emptied. The paths that pass through SA are the most critical ones and are most vulnerable to any form of variations. The other paths are shorter and have a lot of available slack before becoming critical.

Clocking constraints should also be respected on the inter-router paths, i.e., the links. In most cases, the NoC links are routed in certain intermediate layers with 2× and 4× lower resistance than the resistance of local routing layers. This occurs since the majority of local-routing metal layers are used by the processing cores and their caches, while the top metal layers (with almost 8× lower resistance) are primarily occupied by power and clock routing [15], [16]. As measured in [15], and used in a real prototype at 32 nm, the wire delay of the intermediate group of metal layers available to the NoC links ranges from 55-350 ps/mm, depending on repeater placement and wire spacing. Recent studies show the same trend, even in scaled technologies [17]. As it will be shown in the experimental results, this link wire delay corresponds to only a small part of the router delay, assuming medium wire lengths on the order of 2-4 mm. Note that the link length scales down proportionally to the cores and their caches in low-radix networks, such as 2D meshes. For instance, at the 32 nm technology node, the longest side of a typical Chip Multi-Processor tile (i.e., CPU core + 32 kB L1 instruction and data caches + 512 kB L2 cache slice) is 3.27 mm – as demonstrated in [15].

This delay asymmetry between link and router delays has already been identified and utilized to either (1) allow for the traversal of multiple network hops in a single clock cycle [18], or (2) to enable single-cycle crossing of multi-millimeter links employed by high-radix routers in low-diameter network topologies [19]. In this paper, we plan to exploit the short delay of link traversal to simplify timing-error detection within the router. For longer links, pipelining could be employed to increase the available slack, since, in such cases, only part of the link traversal delay should fit in the original clock period.

III. Timing-Error Detection

Timing-error detection/handling in TRNoC takes different forms, depending on the examined delay path. Possible timing errors are detected and handled on the three most critical paths, which pass through the SA logic of NoC routers and stop either at (1) the downstream data output, or (2) the upstream credit-update output of the router, and the input buffers.

A. Input-to-Data-Output Errors: Switching

In a baseline router, both the launch and the capture registers of an input to data-output path are triggered by the same clock edge (see Figure 2). Thus, any additional variation-induced delay to this path should be covered by an additional timing margin that would increase the clock period, thus making the routers slower. The required timing margin can be alternatively added by skewing the clock edge of the capture register by $+d$ ps (Figure 3). In this way, the signals of this path have $+d$ ps more time to settle to their final value, while still operating at the original fast clock frequency. After suitable calibration, an appropriate value of d that is sufficient for covering the cumulative delay effect of process variations can be selected.

Skewing the edge seen by the capture flip-flops at the output ports of the router makes the hold-time constraint of these paths worse by d ps. However, the hold constraints can be handled at design-time during logic synthesis, by introducing additional delay to the switching part of the router, in order to increase the delay of the possible fast paths.

At the same time, the extra margin offered to the outputs of the router makes the link observe d ps less time for flit traversal from the output of the router to the input buffers of the next router. This increases the possibility of link-timing errors. However, as analyzed in Section II, in the majority of cases, link traversal experiences significantly lower delay than the delay of the routers (the delay of the router typically determines the clock frequency of the NoC). Thus, simply "borrowing" d ps from link traversal will not create any timing problems for reasonable wire lengths between 2–4 mm.

Even in cases where borrowing time from the link may be risky (i.e., when dealing with very long inter-router links), *link timing can always be solved by trading off packet latency and area with slack availability*. For example, an optional pipeline register acting as a timing-slack donor stage [20] to LT can be placed at the router's output (see Figure 3), or even at the middle of the link. In this way, the original clock period, or even the large time portion left after lending d ps to the router, is allocated for traversing *only part of the link*, which is more than sufficient for prohibiting timing errors. In extreme cases, even more pipeline (donor) registers can be added to adjust the available timing slack to the desired level. Every pipeline register added increases the round-trip credit notification loop, thus additional buffers are required for achieving 100% throughput on the links in the error-free case. Optionally, flow-controlled pipeline registers can be added to the links, as proposed in [9], after being adapted to credit-based flow control.

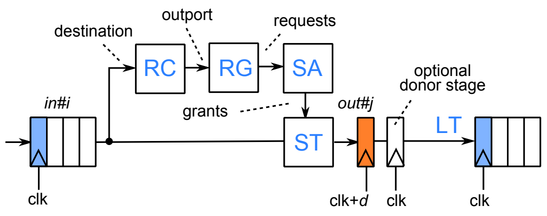

Fig. 3. In TRNoC, the router's output ports sample data using a delayed clock, by "borrowing" time from the LT stage (which is significantly faster than the intra-router delay).

State-of-the-art timing error detection techniques [3], [4], [5], [6], employ a pair of flip-flops (or latches) to capture the logic signals at the end of the critical paths twice: first by the main flip-flop after a clock period, and next by the second flip-flop after some delay (to account for worst-case operation). If the captured values are identical, no error has occurred and the computation proceeds using the speculated value sampled in the main flip-flop. If a mismatch is detected, a recovery process is engaged.

TRNoC avoids double sampling at the output of the routers, since the latter is inherently problematic within a NoC context. Double-sampling detection and recovery stategies would allow erroneous data to leave the router before the error flag is asserted. Therefore, the error signal should also propagate to the next router to enable checking of the validity of the arriving data. Since the error signal is triggered after the delayed edge of the clock, it inevitably steals some portion of the clock cycle from the link, in order to allow its safe propagation under the original clock frequency. Also, the speculative nature of forward data propagation does not allow for distributed per-input and per-output error recovery. Once an error is detected at one output, all inputs should be frozen in the next cycle.

Double-sampling techniques rely on the fundamental assumption that the second sample taken using the delayed clock edge is always correct. Therefore, the correctness feature is always preserved in TRNoC, which uses only one delayed sample of the router's outgoing data, while still allowing inputs unaffected by errors to continue working unimpeded.

B. Input-to-Input Errors: Dequeue and Credit Update

Even if double-sampling is not useful at the outputs of the router, it is the preferred approach for detecting errors on the timing paths that stop at the input buffers of the router (either for a buffer dequeue, or for sending the necessary credit updates upstream to the previous router). As shown in

Figure 4, a late-arriving dequeue signal to each input buffer may cause two different types of error scenarios, which should both be detected and distinguished:

- In the first scenario, shown in Figure 4(a), the dequeue signal is asserted correctly, but belatedly. Consequently, the head-of-line flit (that has won in arbitration) is correctly transferred to the output port via the multiplexing logic of the router, but it still occupies the head-of-line position in the input FIFO buffer (we call this error "ERR-NO-DQ"). Also, a credit referring to the input buffer of the next router has already been consumed, and, possibly, the status of the input port and the selected output port have changed (if the head-of-line flit that generated the request was a head/tail flit).
- The second scenario, depicted in Figure 4(b), is the opposite of the first scenario: the input buffer receives a delayed active dequeue signal, although it has never generated an active request (either the input buffer was empty, or the head-of-line flit wanted to leave from an occupied output port, or from an output port without available credits), or it has received a grant from the wrong output port. As a result, the head-of-line flit inadvertently leaves the input buffer and should somehow be returned to its position (this error is termed "ERR-DQ"). Note that even if the flit departs from the input buffer, it is not captured by any output port of the router (thus not causing any additional error). Recall that the output ports of the routers take a delayed sample (Figure 3), which is *always* correct when the added timing margin is enough to cover worst-case variations.

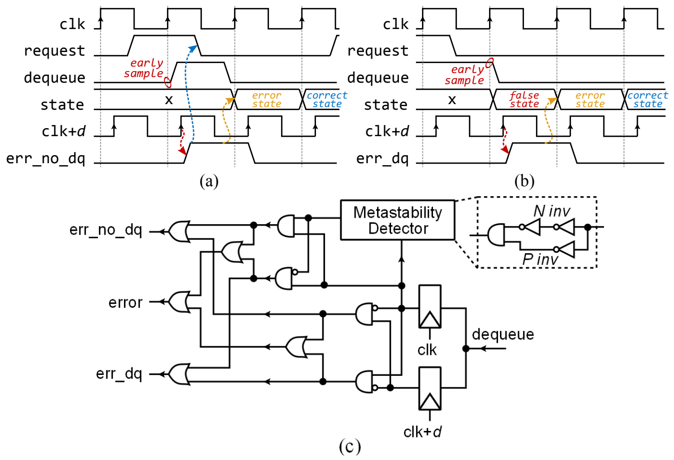

Fig. 4. A late-arriving dequeue signal to each router input buffer may cause two different types of error scenarios: (a) ERR-NO-DQ, and (b) ERR-DQ. Errors are detected and distinguished by TRNoC's timing-error detector (c) added to each input port of the router.

To detect these two error cases, a timing-error detector is added to each input port of the router. The detector's structure is illustrated in Figure 4(c). The per-input dequeue signal is sampled twice using the normal and a skewed (delayed by d ps) version of the clock edge. Depending on the value of the two samples, the two error scenarios are distinguished from one another. If at least one of the two error scenarios is detected, an error signal is asserted per input port. The error signal is also asserted when the main flip-flop enters meta-stability, due to the transition of the dequeue signal close to the clock edge. Then, the output of the main flip-flop cannot be used for error detection. This situation is detected by the meta-stability detector within the TRNoC detector. As clarified in [21], the meta-stability phenomenon cannot be eliminated in any double-sampling-and-comparing strategy. It can only be confined to the control path, as done in TRNoC, thus not allowing any meta-stable state to propagate in the router's datapath and to corrupt the packet contents.

The closely-related timing path that starts from the head-of-line position of the input buffer and ends at the credit-update output line – as shown in Figure 2 – is also handled via the same error-detection logic of Figure 4(c). When such an error is detected, a credit is not returned to the upstream (previous) router, as will be described in Section IV.

Just as double-sampling is not well-suited for detecting timing errors at the outputs of the router, the equivalent use of skewed (delayed) sampling – as used at the output ports – is not applicable to the timing paths ending at the NoC input ports. These timing paths, which involve the per-input registers and buffers, start from and end at the same register (forming a loop). Thus, delaying the sampling of these flip-flops will not solve timing errors within the looped timing paths. On the contrary, this is possible in the output ports, since the timing slack is either borrowed from the link, or provided by the extra donor stages. Adding donor stages inside the router would be possible, in order to break the self loops. However, this addition will significantly change the semantics of the request/grant protocol governing the switching operation.

IV. Error Handling and Recovery

Recovery from errors in NoC routers is more involved than error recovery in processor pipelines. Processor pipelines are structurally linear, and, once an error is detected, instruction fetch can be stalled and in-flight instructions can be replayed [4], [22]. On the contrary, in NoC routers, both failed and correct connections between multiple input and output ports may concurrently exist at any given time, while (simultaneously) upstream connections from neighboring routers continue to dispatch new data to the affected router. Hence, any error recovery mechanism derived for NoC routers should be distributed in nature, and it should only affect the failed input-output connections; the remaining connections should be able to continue working uninterrupted.

As described in Section III, the grant signals that are distributed *in parallel* to the crossbar and back to the input buffers are given a sufficient timing margin to settle to their final values. At the output ports, this margin guarantees that data always arrives safely, while, for the inputs, the TRNoC error detector differentiates between correct and erroneous (ERR-NO-DQ or ERR-DQ) operation. Once an error is detected in an input port, a single-cycle error-handling and recovery process begins for that particular input port, *independently* from the rest. Multiple input ports may be in recovery mode, but only if they are simultaneously affected by a timing error.

Each input port can be in any one of three states, as depicted by the Finite State Machine (FSM) of Figure 5. The transitions between states are determined solely by the output signals of the per-input timing-error detector. The states also guide the immediate actions that should be taken for error-handling and recovery, which occur in the next cycle after error detection. Depending on the error scenario, different actions should take place. Nevertheless, two error-handling actions are common to all error situations, i.e., prohibiting next-cycle request generation, and stopping the return of credits.

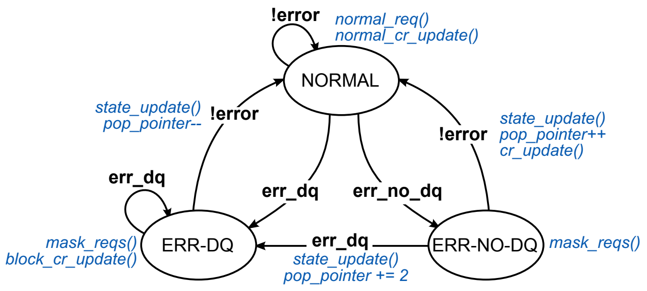

Fig. 5. The FSM describing the three possible states of a TRNoC router input port. The transitions between states are determined solely by the output signals of the per-input timing-error detector (shown in Figure 4(c)).

Once an error is detected, the requests from the affected input port should be masked. During the next cycle (which is spent for error recovery), no request should be sent to the output ports, until the state of the affected input port returns to NORMAL. Request masking also ensures that no redundant credit will be consumed from the input buffer of the next (downstream) router.

The requests are masked after the RG logic (see Figure 2). In this way, the delay of (1) the error-detection logic, and (2) the error path that launches from the flip-flop triggered by the skewed edge of the clock, are partially hidden by the RG delay. If the d ps delay is stretched (increased) to a large portion on the clock period, then, inevitably, the critical path will migrate to the paths launched by the error-detection logic. This situation is not desirable, and it is a sign that the maximum allowed value of d has been reached.

Besides request-masking, credit updates should also be prohibited when an error is detected. In TRNoC, the credit update is delayed until the input returns to its NORMAL state. Therefore, even if a flit has been instructed to depart (dequeue) from the input buffer by mistake, the upstream (previous) router on the other side of the link is not informed about this action and assumes that the flit's position is still occupied. Once the state of the input buffer has returned to NORMAL and the head-of-line flit is appropriately dequeued, then a credit update is allowed to be sent upstream.

An input port transitions into the ERR-NO-DQ state once it receives a delayed dequeue signal, due the increased delay of the path starting from the head-of-line flit of the input buffer and passing through RG and SA (see Figure 2). Therefore, the flit is not dequeued on time and still occupies the head-of-line position. However, the flit has been correctly transferred to the output of the router through the multiplexing logic and the provided delay margin at the output register of the router. Therefore, in the following cycle (dedicated to recovery), the flit should be removed from the input buffer, by increasing the index of the pop (head) pointer of the FIFO buffer. If the flit was a head or a tail flit, the input and output port states should also be correctly updated to reflect that the flit has, indeed, left the router. These actions occur during the transition from the ERR-NO-DQ state to the NORMAL state, assuming that no other consecutive error occurs.

On the contrary, an input port transitions to the ERR-DQ state if a flit was dequeued by mistake, and – under normal operation – it should never have received a grant in this cycle. Although the delayed sampling at the output ports guarantees that this (erroneously departed) flit will never be captured by an output port, the erroneous dequeuing action increases the pop pointer of the input buffer. Note that the data of the flit is not lost; it still resides in the input buffer (only the buffer pop pointer has moved). Thus, in order to bring the head-of-line position of the input buffer to its previous state, we must merely decrease the pop pointer. This rewind of the state of the input buffer can safely occur, since the requests are masked during the recovery cycle, and credit updates are prohibited.

Most importantly, it is guaranteed – by construction – that the position that the flit resides in the input buffer cannot be over-written by a new incoming flit from the upstream (previous) router during the recovery cycle. Data over-writing is only possible if the upstream router can "observe" more credits than the ones really available. Since credit updates are blocked until the input port returns to NORMAL (error recovery has been completed), this scenario is impossible. Every incoming flit allocates an empty buffer slot that corresponds to a previously consumed credit (in the upstream router).

The FSM governing the recovery operations of TRNoC (Figure 5) also covers the possibility of back-to-back burst errors. When an input receives consecutive delayed dequeue signals, it stays in the ERR-DQ state, and neither the pop pointer, nor the input/output port state variables are updated. They temporarily keep their old values, which are corrected on the transition to the NORMAL state. Nevertheless, back-to-back delayed dequeues are rare (or even impossible, depending on the logic-level implementation of the arbiters), since, during recovery, the requests are masked, and, thus, the arbiters are not expected to produce any new grant.

When an ERR-DQ error occurs right after (next cycle) an ERR-NO-DQ error, the effect of ERR-NO-DQ has not been corrected yet; the corrections are scheduled to occur during the transition to NORMAL. The ERR-NO-DQ situation requires an increase of the pop pointer, while the ERR-DQ error (that has occurred in the meantime) requires a decrease of the pop pointer. Thus, to guarantee that both errors will be corrected when the affected input port of the router eventually moves from ERR-DQ to NORMAL, the pop pointer is increased by two during the transition from ERR-NO-DQ to ERR-DQ.

The transition from ERR-DQ to ERR-NO-DQ is not possible. An input port that has entered the ERR-DQ state does not expect any dequeue signal to arrive, since it is not trying to send a flit to an output port. Therefore, a de-asserted dequeue signal (either late, or on time) does not make any difference, and it is simply treated as a return to the NORMAL state. Only the occurrence of another back-to-back erroneous asserted dequeue signal will keep the input port in the ERR-DQ state.

V. Experimental Evaluation

In this section, we examine the area/delay overhead of the proposed timing-error detection and recovery mechanisms of

TRNoC. The operational behavior of TRNoC was thoroughly tested in simulation and on an FPGA-based prototype, where signals were deliberately delayed with extra circuitry, in order to cause timing errors and triggering of the recovery process. In all cases, the traffic injected in the NoC was delivered correctly to its destination, albeit with possible interruptions to recover from the injected timing errors.

All routers under comparison were implemented in VHDL, mapped to a commercial low-power 45 nm standard-cell library under worst-case conditions (0.8 V, 125 °C), and placed-and-routed using the Cadence digital implementation flow. The generic router models were configured to 5 input-output ports, as needed by a 2D mesh network (see Figure 1). The flit-width was set to 64 bits and each input buffer holds 4 flits. Results were obtained for all designs, after constraining the tools using the same parameters for all designs, and assuming that each output is loaded with a 2 mm wire.

In the first set of experiments, we verify the minimum area overhead imposed by TRNoC. Figure 6 depicts the area of a baseline router without any timing-error detection mechanism and two variants of TRNoC, when optimized for three different clock frequencies. The two TRNoC variants assume a timing margin of d ps, corresponding to 10% of the NoC's clock period in each case. The "TRNoC" variant assumes that the timing margin can be safely borrowed from the output links, while "TRNoC-D" also includes additional donor stages at the outputs of the router. The derived results show that the two TRNoC variants are minimally intrusive, requiring on average an area increase of 7% over a baseline NoC router.

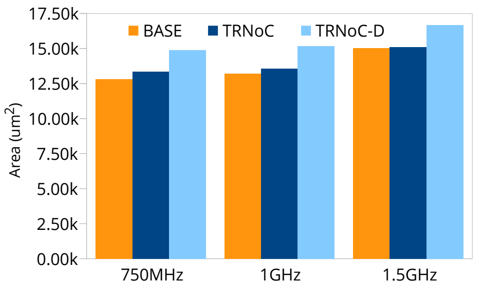

Fig. 6. The area of a baseline NoC router and two variants of TRNoC for 5 input-/output-port routers, under three frequency targets.

As pointed out in Section III, the maximum timing margin facilitated by TRNoC depends on the delay of the error-detection logic and the delay of the RC and RG router modules, which operate in parallel. Figure 7 highlights the maximum achievable timing margin, as a percentage of the clock period, under various clock frequency targets at 0.8 V. As expected, the margin is around 20% at 1 GHz and diminishes to a sufficient 12% at higher speeds. Lower frequency cases make sense under scaled voltage (low-power mode), following the reported results for fast industrial NoC designs [23].

VI. Conclusions

Process variability has emerged as a major challenge for the semiconductor industry. This work tackles timing-error violations due to variations within the NoC of multi-core systems. The TRNoC architecture minimizes the increasingly unwieldy timing guard-bands required to ensure proper operation in the presence of variations, while providing full

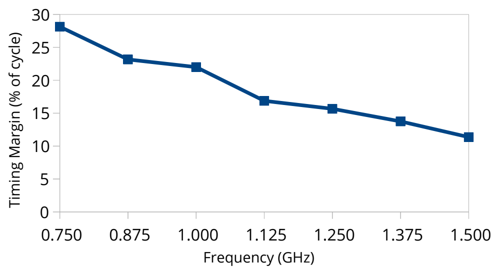

Fig. 7. The maximum achievable timing margin, as a percentage of the targeted clock frequency, at a supply voltage of 0.8 V.

timing-error resilience. TRNoC employs a combination of delayed-sampling techniques and distributed low-cost timing-error detector modules, which cover all salient timing paths of the router. Timing errors trigger a distributed single-cycle recovery mechanism, which guarantees lossless operation and ensures that the error-free components remain unaffected.

References

[1] S. Bhunia and S. Mukhopadhyay, *Low-Power Variation-Tolerant Design in Nanometer Silicon*. Springer, 2011.
[2] Arteris, "A comparison of network-on-chip and buses," Tech. Rep., 2005.
[3] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *Proc. of IEEE VTS*, 1999, pp. 86–94.
[4] D. Ernst and et al., "Razor: a low-power pipeline based on circuit-level timing speculation," in *Proc. of Intern. Symp. on Microarchitecture*, Dec 2003, pp. 7–18.
[5] M. R. Choudhury and et al., "Time-borrowing circuit designs and hardware prototyping for timing error resilience," *IEEE Trans. on Computers*, vol. 63, no. 2, Feb. 2014.
[6] S. Valadimas and et al., "The time dilation technique for timing error tolerance," *IEEE Trans. on Computers*, vol. 63, no. 5, pp. 1277–1286, May 2014.
[7] C. Nicopoulos and et al., "On the effects of process variation in network-on-chip architectures," *IEEE Trans. on Dependable and Secure Computing*, vol. 7, no. 3, pp. 240–254, Jul.-Sept. 2010.
[8] H. Kim, A. Vitkovskiy, P. V. Gratz, and V. Soteriou, "Use it or lose it: Wear-out and lifetime in future chip multiprocessors," in *Proc. of Intern. Symposium on Microarchitecture*, 2013, pp. 136–147.
[9] R. Tamhankar and et al., "Timing-error-tolerant network-on-chip design methodology," *IEEE Trans. on CAD*, vol. 26, no. 7, July 2007.
[10] C. Hernandez and et al., "A new mechanism to deal with process variability in noc links," in *Proc. of IEEE IPDPS*, May 2009.
[11] S. Medardoni, M. Lajolo, and D. Bertozzi, "Variation tolerant noc design by means of self-calibrating links," in *DATE*, 2008, pp. 1402–1407.
[12] Y. Kang, T. Kwon, and J. Draper, "Fault-tolerant flow control in on-chip networks," in *Proc. of Intern. Symp. on Networks-on-Chip*, 2010.
[13] M. H. Neishaburi and Z. Zilic, "A fault tolerant hierarchical network on chip router architecture," *JETTA*, 2013.
[14] A. Dalirsani, M. Kochte, and H.-J. Wunderlich, "Area-Efficient Synthesis of Fault-Secure NoC Switches," in *Proc. of IEEE International On-Line Testing Symposium*, 2014, pp. 13–18.
[15] K. Sewell and et al., "Swizzle-switch networks for many-core systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 278–294, 2012 2012.
[16] W. J. Dally, C. Malachowsky, and S. W. Keckler, "21st century digital design tools," in *Proc. of Design Automation Conference*, 2013.
[17] R. Manevich and et al., "Designing single-cycle long links in hierarchical nocs," *Microprocessors and Microsystems*, pp. 814 – 825, 2014.
[18] T. Krishna and et al., "Smart: Single-cycle multi-hop traversals over a shared network-on-chip," *IEEE Micro*, May/June 2014.
[19] N. Abeyratne and et al., "Scaling toward kilo-core processors with asymmetric high-radix topologies," in *IEEE HPCA*, 2013.
[20] A. Tiwari, S. R. Sarangi, and J. Torrellas, "Recycle:: Pipeline adaptation to tolerate process variation," in *Proc. of ISCA*, 2007, pp. 323–334.
[21] S. Beer and et al., "Metastability in better-than-worst-case designs," in *Proc. of ASYNC*, May 2014.
[22] K. Bowman and et al., "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 194–208, Jan 2011.
[23] P. Salihundam et al., "A 2Tb/s 6x4 Mesh Network with DVFS and 2.3Tb/s/W router in 45nm CMOS," in *VLSI Circuits*, 2010.