NBTI-aware design of NoC buffers

Davide Zoni
Politecnico di Milano – DEI
Via Ponzio 34/5, 20133 Milano, Italy
zoni@elet.polimi.it

William Fornaciari
Politecnico di Milano – DEI
Via Ponzio 34/5, 20133 Milano, Italy
fornacia@elet.polimi.it

ABSTRACT
Network-on-Chips (NoC) play a central role in determining performance and reliability in current and future multi-core architectures. Continuous scaling of CMOS technology enable widespread adoption of multi-core architectures but, unfortunately, poses severe concerns regarding failures. Process variation (PV) is worsening the scenario, decreasing device lifetime and performance predictability during chip fabrication. This paper proposes two solutions exploiting power-gating to cope with NBTI effects in NoC buffers. The techniques are evaluated with respect to a variable number of virtual channels (VCs), in the presence of process variation. Moreover, power gating delay overhead is accounted. Experiments reveal a net NBTI V_{th} saving up to 54.2% against the baseline NoC, with an area overhead below 5%.

Categories and Subject Descriptors
B.8 [**Performance and Reliability**]: General

1. INTRODUCTION
The presence of multiple cores makes Network-on-Chip (NoC) [3] a crucial component to fully exploit the available computing power. Beside, the actual integration density is limited by the reliability of the device: rising of operating temperatures and physical failure mechanisms (e.g., Negative Biased Temperature Instability (NBTI) or stress-migration) can seriously limit the device lifetime. At the same time, *Process Variability* (PV), produces unexpected power/performance fluctuations, which are becoming a major fabrication challenge for the upcoming technology scales [12]. This paper aims at mitigating NBTI degradation in the NoC virtual channel (VC) buffers, considering PV issues as well as different microarchitectural parameters and the possible presence/absence of NBTI sensors. The paper encompasses two contributions. *Reliability enhancement* – two different techniques are presented with emphasis on the NBTI mitigation; each technique considers process variation and architectural parameters, i.e. the number of virtual channels. *Cooperative and modular approach* – each router can improve NBTI mitigation on its buffers, exploiting information from neighbor routers. The paper is organized as follows. Section 2 reports an overview of the state-of-the-art on NBTI related methodologies. Section 3 describes the proposed NBTI-aware mitigation strategies. Results are then reported and discussed in Section 4. Conclusions are drawn in Section 5.

2. RELATED WORKS
This section provides an overview of the state-of-the-art related to NBTI mitigation design methodologies considering also PV. A linear programming based approach has been proposed in [5] to optimally drive the inputs to individual gates in order to prevent static NBTI fatigue. The work addresses the NBTI stress due to long stand-by periods. Li et al. exploits idle time in functional unit inside a processor core to recover NBTI degradation with a negligible performance loss and area overhead [11]. [13] proposed a multi-level technique to reduce the impact of NBTI degradation on the functional units of a high performance processor core. Li et al. observes how process variation can significantly influence on-chip network design choices [10]. Xin Fu et al. present a comprehensive approach to mitigate NBTI degradation considering NoC architectures [7]. This work discusses several approaches to effectively manage NBTI impact, focusing on different micro-architectural components, namely buffers and arbiters. However, [7] does not exploit the information on actual NoC traffic between each router pair, hence it does not catch the possibility to aggressively reduce NBTI degradation as well as to perform a NBTI/performance trade-off. Finally, [6] reviewed the main NBTI mitigation mechanisms providing also an accurate NBTI model that we adopted to extract absolute NBTI values from our experimental data.

3. PROPOSED ESTIMATION FLOW
Our work seats on the baseline 3-stage virtual channel pipelined router implemented in the Garnet model available in GEM5 [2]. Section 3.1 provides the proper background and rationale, while the proposed methodologies are discussed in Sections 3.2 and 3.3.

3.1 Methodology Background
Considering CMOS circuits, NBTI occurs in PMOS transistors when $V_{gs} = -V_{dd}$, namely when the PMOS is active. Such *stress condition* increases the threshold voltage (V_{th}) producing, as a side-effect, a degradation of the driving current and then performance degradation. When the NBTI

stress is removed, i.e. a logic "1" is applied to the gate the transistor, it enter in a *recovery state*, that induces a progressive, yet partial, recovery of the device threshold voltage. NBTI is a time dependent mechanism and it is known to be exponentially dependent on temperature, supply voltage and *duty-cycle*, as explained by the long-term model of Equation 1 [4], where K_v is a parameter depending on supply voltage and operating temperature, T_{clk} is the clock period, α is the stress probability of PMOS devices, i.e. the *NBTI-duty-cycle*, β_t is influenced by the temperature, while n is generally set to 1/6.

$$|\Delta V_{th}| \approx \left(\frac{\sqrt{K_v^2 \cdot T_{clk} \cdot \alpha}}{1 - \beta_t^{1/2n}} \right)^{2n} \tag{1}$$

We recover NBTI degradation on buffers by reducing the *stress period* exploiting *power gating*, which has been demonstrated to be a viable solution to address such class of problems [6]. We consider each buffer enhanced with a header PMOS transistor that is responsible to cut the supply voltage when required. These *sleep* transistors act as switches that toggle supply voltage when required (from V_{DD} to *virtual* V_{DD}). The sleep transistor design is out of the scope of this paper, even if we account for power gating overhead that reduces NBTI mitigation effectiveness. Details on header PMOS transistors design, as well as on the influence of aging on sleep transistor can be found in [9]. Starting from the baseline NoC we made two useful observations. For simplicity, let us consider a single upstream and downstream routers pair, where the upstream router sends flits to the input buffers of the corresponding downstream one. First, a single flit can flow from each upstream downstream routers pair at each cycle, thus at most an idle virtual channel is needed to store such flit if it belongs to a new packet. Note that we do not allow packet mixing in a single VC buffer. Last, the single idle VC can be switched-off based on traffic information, since the VC allocation in the downstream router happens in the upstream one. Upstream router holds all the information on new packets that want to go to the specific downstream router and have not a VC allocated yet on the downstream router. These new packets are eventually stored in the input ports of the upstream router. In such a way the upstream router needs an additional link to send the VC identifier to be left idle by the downstream router if new packets are waiting for the downstream router. Note that we refer to new packets as these waiting in an upstream router input port, while they have no VC assigned, yet. Otherwise the downstream router can switch off all its idle VCs, since no new flit are expected. This way all the incoming flits to the downstream router input port will be stored in one of its already active VC.

3.2 Sensor-less Round-Robin Approach

This section details the sensor-less round-robin policy (*rr-no-sensor*) to minimize the NBTI stress period on the most degraded virtual channel buffer. No information on the most degraded virtual channel buffer is used, thus it recovers all the virtual channel buffers on a specific input port cyclically (see Algorithm 1). Algorithm 1 is implemented in the output port on the upstream router to select the buffers to be recovered in the corresponding downstream input port router, since the Virtual channel Allocation (VA) takes place in the upstream router. Note that during VA a virtual channel (from the selected output port) is reserved for the new packet. It considers the state of each buffer in the downstream router input port through the corresponding output virtual channel state. Moreover, it cyclically compile an ordered list of all VC buffers, which represent the visiting order to allocate new flits from new packets. In particular the list resemble a FIFO buffer to cyclically stress different buffers in a balanced way. Since the actual recovery on buffers is done in the downstream router, that owns buffers, the upstream router sends to the downstream one the VC id that must be left idle, while all the other idle VC can be switched off to recovery NBTI. Thus, an additional upstream downstream control link that is used by the upstream router to signal to the downstream one the VC buffer identifier to be left idle. This additional control flow channel drives the *enable* signal also to invalidate the VC buffer identifier in case of aggressive recovery, since no new virtual channel is needed for the current VA stage. As a result, the additional control flow link has $1 + log_2 v$ lines only, where v is the number of virtual channels in the input port of the downstream router.

Algorithm 1: The *rr-no-sensor* pre-VA stage for each upstream router.

```
Input: out_vc_state, active_candidate_vc, is_new_traffic_outport_x()
Output: enable, active_vc
enable ← 0;
active_candidate = get_vc_candidate();
boolTraffic = is_new_traffic_outport_x();
if not boolTraffic then
    enable ← 0;
    active_vc ← active_candidate;
    return;
offset_vc ← active_candidate;
foreach iter ∈ (1..num_vcs) do
    if is_idle(offset_vc) or is_recovery(offset_vc) then
        set_idle(offset_vc);
        enable ← 1;
        active_vc ← offset_vc;
        return;
    offset_vc++;
    if offset_vc ≥ num_vc then
        offset_vc ← 0;
```

3.3 Sensor-wise Approach

This section details the logic modifications implemented in the baseline on-chip network to obtain a second methodology we called *sensor-wise*, that uses NBTI sensors to detect the most degraded buffer in an input port. It is presented using a couple of routers, i.e. upstream and downstream, only for the sake of clarity and conciseness. The main idea of this proposal is the decoupling of the measurement and the selection steps. In particular, for each upstream downstream routers pair, the NBTI degradation is evaluated in the downstream router by means of sensors, since it physically hosts the VC buffers. On the other hand, the selection of the downstream router VCs that can be recovered is in charge to the upstream one, since it has all the required traffic information and performs the VA stage for the VCs in the downstream router. Algorithm 2 details the recovery policy the upstream router uses to select a single virtual channel to be idle in the downstream router, if at least a new packet can eventually traverse the link to the downstream router. Otherwise it signals the downstream router to recover all its idle VC buffers. The algorithm knows the number of incoming packets that wants to go to the East output port of the router itself as well as the most degraded VC in the corresponding input port of the downstream router.

Algorithm 2: The *sensor-wise* pre-VA stage for Router A East output port.

```
Input: out_vc_state, most_degraded_vc, is_new_traffic_East_outport()
Output: enable, idle_vc
enable ← 0;
idle_vc ← -1;
count_idle ← 0;
boolTraffic = is_new_traffic_East_outport();
foreach iter ∈ (1..num_vcs) do
    if is_idle(offset_vc) or is_recovery(offset_vc) then
        set_idle(offset_vc);
        count_idle++;
if is_idle(most_degraded_vc) and count_idle> boolTraffic then
    set_recovery(most_degraded_vc);
    count_idle −−;
foreach iter_vc ∈ (1..num_vcs) do
    idle_vc ← iter_vc;
    if is_idle(iter_vc) and count_idle> boolTraffic then
        set_recovery(iter_vc);
        count_idle−−;
if boolTraffic then
    enable ← 1;
```

Table 1: Experimental setup: processor and router micro-architectures and technology parameters.

Processor core	1GHz, out-of-order Alpha core
func units	4 iALU, 4 iMUL/DIV units and 4 FP units
L1 cache	64kB 2-way set assoc. split I/D, 2 cycles latency
L2 cache	512KB per bank, 8-way associative
Coherence Prot.	MOESI token (for real traffic)
Router	3-stage wormhole switched with 32b link width
Topology	2D-mesh, based on Garnet [2]
	for link width and NoC frequency (@1GHz)
Technology	$V_{th} = 0.160$ at 32nm and $V_{th} = 0.180$ 45nm, Vdd=1.2V

Last, it knows the actual state of the input port VCs in the downstream router through its output port state information. An NBTI sensor [14] is require for each VC buffer to monitor the V_{th} degradation, so to select the most degraded VC buffer. Moreover, two additional control flow links are inserted between each couple upstream downstream router. One link allows to send a single VC identifier (VC-ID) from the upstream to the downstream router as in the methodology proposed in Section 3.2, thus requiring $1+log_2(num\_vc)$ lines. The other link allows to signal the upstream router the most degraded VC in the downstream router input port. The link requires $log_2(num\_vc)$ lines and there is no need for an additional enable line, since a most degraded VC can be supposed to be always present.

4. RESULTS

This section adresses the reduction of the NBTI stress period considering also process variation. Results for synthetic traffic are discussed in Section 4.1, while Section 4.2 discusses the power gating overhead. Cycle-accurate simulations have been performed using `GEM5` simulator and `Garnet` network model [2] considering 3-stage pipelined routers and 2D-mesh multi-core architectures integrated in HANDS [15]. We simulated $30*10^6$ cycles for each scenario, being the NoC traffic heavier at warmup. The multi-core architecture is composed of tiles; each tile in the 2D-mesh is composed of an out-of-order processor based on Alpha-21264 ISA, private L1 cache and shared distributed L2 cache banks, and a memory controller. Table 1 summarizes the setup and the parameters of the simulated multi-core architecture. We also consider within-die process variation on the PMOS threshold voltage V_{th}, assuming the impact of die-to-die variation to be constant in the same chip [7]. Moreover, we assume no variation in the same VC buffer [7], hence a PMOS transistor is associated to each VC buffer of each router; each modeled PMOS transistor has its own starting V_{th}, that has been

Table 2: stress (%) for all the VCs using *rr-no-sensor* and *sensor-wise* policies. Results for 4-cores and 16-cores with variable injection rate and 4 VCs. (Gap = rr-no-sensor−sensor-wise)

Scenario (4 VCs)	MD VC	rr-no-sensor				sensor-wise				Gap
		VC0	VC1	VC2	VC3	VC0	VC1	VC2	VC3	
4core-inj0.10	1	11.9%	12.1%	12.5%	11.7%	1.5%	0.1%	9.9%	36.8%	11.7 − 0.1 = 11.6%
4core-inj0.20	0	19.8%	19.9%	20.2%	20.0%	1.2%	6.6%	22.6%	49.3%	20.2 − 1.2 = 19.0%
4core-inj0.30	3	27.7%	27.8%	27.8%	27.8%	14.5%	34.5%	58.1%	4.0%	27.8 − 4.0 = 23.8%
16core-inj0.10	1	17.6%	17.3%	17.5%	17.4%	5.2%	0.9%	18.9%	44.8	17.3 − 0.9 = 16.4%
16core-inj0.20	1	31.6%	31.5%	31.6%	31.4%	19.6%	7.9%	39.3%	60.5%	31.5 − 7.9 = 23.6%
16core-inj0.30	2	46.2	46.3%	46.2%	46.3%	37.6%	56.2%	19.6%	71.4%	46.2 − 19.6 = 26.6%

extracted from a Gaussian distribution with absolute average value of 0.180 Volt for 45nm technology, and a standard deviation equal to 0.005 [1]. Moreover, the two proposed techniques have an area overhead lower then 5%. In particular we obtained the link area for a 45nm technology node using *Orion2.0* [8], achieving an increase in area overhead for a single additional link of 1.9%. Then, we integrated the two techniques in NetMaker and synthesized using the Cadence toolchain coupled with an opensource standard cell 45nm library, obtaining an area overhead of less then 1% and 3.20% for the *sensor-wise* and *sensor-less* techniques, respectively.

4.1 Synthetic Results

This section reports the results considering uniform traffic patterns on both 4-core and 16-core 2D-mesh architectures. Each result is sampled from the upper left-most router on its east input port. We report results for 4- and 16-cores scenarios, varying the injection rates: Table 3 considering 2 virtual channels per input port, while the results for the same 4- and 16- architectures using 4 virtual channels per input port are reported in Table 2. Table 3 and Table 2 share the same format, where the first and the second column report the identifier of the simulated scenario and the most degraded virtual channel buffer identifier for the simulated architectures (i.e., *Scenario* and *MD* columns). Note that the most degraded VC changes through different simulated scenarios due to the random sampling process that mimics process variation impact. Both Table 3 and 2 report two multi-columns, one for each evaluated policy, i.e. *rr-no-sensor* and *sensor-wise*. For each policy, i.e. multi-column, we report the *stress* obtained at the end of the simulation for each VC. The last column of each table (*Gap*) reports the *stress* difference between the two policies on the most degraded VC buffer. Positive *Gaps* means that *sensor-wise* behaves better than *rr-no-sensor* policy, i.e. the second obtains lower *stress*. *Sensor-wise* policy always obtains lower *stress period* values than the *rr-no-sensor* policy as expected. However, it is interesting to note the *stress* gap reduction between *sensor-wise* and *rr-no-sensor* with the traffic load, while increases with the number of virtual channels. The increase in the traffic load prevents the *sensor-wise* policy to employ VCs different from the most degraded one to steer new packets, since we have an high probability that all the VCs are busy at the same time. On the contrary, the *rr-no-sensor* approach is independent of this aspect, since new packets are assigned to VCs in a round-robin fashion. Thus, if multiple virtual channels are present the *sensor-wise* maintains the most degraded virtual channel buffer still recovered, by moving the traffic on the other available buffers. This aspect is verified considering the first three rows in Table 3, where the injection rate is 0.1 0.2 and 0.3, respectively; the

Table 3: stress (%) for all the VCs using *rr-no-sensor* and *sensor-wise* policies. Results for 4-core and 16-core with variable injection rate and 2 VCs. (Gap = rr-no-sensor−sensor-wise)

Scenario (2 VCs)	MD VC	rr-no-sensor VC0	rr-no-sensor VC1	sensor-wise VC0	sensor-wise VC1	Gap
4core-inj0.10	1	23.8%	23.8%	37.3%	10.4%	$23.8 - 10.4 = 13.4\%$
4core-inj0.20	1	39.3%	39.3%	52.1%	26.5%	$39.3 - 26.5 = 12.8\%$
4core-inj0.30	0	56.2%	56.5%	46.7%	65.9%	$56.2 - 46.7 = 9.5\%$
16core-inj0.10	0	33.4%	33.6%	20.1%	46.9%	$33.4 - 20.1 = 13.3\%$
16core-inj0.20	0	61.8%	61.6%	51.7%	71.8%	$61.8 - 51.7 = 10.1\%$
16core-inj0.30	1	72.9%	73.0%	80.7%	65.1%	$73.0 - 65.1 = 7.9\%$

Table 4: Percentage of idle cycles grouped per bursts from 1 to 10 considering *rr-no-sensor* and the *sensor-wise* policies, 4-core, 2 VCs. Data are reported for both synthetic and real traffic patterns and for the most degraded VC only.

Scenario	duty-cycle	1	2	3	4	5	6	7	8	9	≥10
inj0.10-rr-no-sensor	23.80	2.99	2.60	2.85	2.54	2.39	2.35	2.32	2.21	2.39	77.36
inj0.30-rr-no-sensor	56.50	8.33	7.32	6.35	5.87	5.86	4.77	4.03	3.65	5.52	48.31
inj0.10-sensor-wise	10.40	3.82	3.42	3.66	3.37	3.17	3.09	3.03	2.86	2.91	70.67
inj0.30-sensor-wise	46.7	5.96	5.54	4.80	4.51	4.57	3.87	3.29	2.84	3.73	60.90

corresponding *Gap* is 13.4%, 12.8% and 9.5%.

4.2 Possible Overhead of Power Gating

This section quantifies the power gating possible overhead since the buffer switch-on and off is not instantaneous. To this purpose, we assume the availability of a predictor providing the idle intervals of each buffer with no delay. We define $idle - burst_i$ as the number of consecutive idle cycles on buffers, where i is the length of the burst, i.e. the consecutive idle cycles that are used to recovery the buffer NBTI. Last, we define T_{wakeup} as the number of cycles required to switch-on a module from the off state. In particular we want to measure the portion of the idle cycles that can be used for NBTI recovery, i.e. the cumulative count of *idle-burst*$_i$ where $i \geq T_{wakeup}$. Table 4 reports the percentage of *idle-burst*$_i$ for $i \in (1..10)$ considering uniform random traffic on a quad-core only due to space limit. The first column identifies the scenario, i.e. simulated architecture, injection rate and the NBTI mitigation technique used (sensor-less, sensor-wise). The second column reports the duty-cycle. The 10 following columns report the collected *idle-burst*. The last column groups all the *idle-bursts* that are longer or equal to 10 cycles (*idle-burst*$_{10}$). Three main observations can be derived from data. First, the length of the *idle-bursts* decreased as traffic increases, as expected. For example, the first two lines in Table 4 show that the idle-burst percentage is shifted towards lower values when injection rate switches from 0.1 to 0.3 flits/cycles/port. Moreover, the *sensor-wise* policy allows for longer idle-bursts on the most degraded VC, being the focus on recovery. For example comparing rows 2 and 4 we observe a greater percentage of *idle-burst*$_{10}$ for the sensor-wise policy than the sensor-less. However, comparing line 1 and 3 *sensor-less* has a greater *idle-burst*$_{10}$ than *sensor-wise*. Although the higher recovery percentage underlines that the *idle-burst*$_{10}$ must be longer for the *sensor-wise*. Last, all the idle-bursts are completely spent in recovery up to $T_{wakeup} \leq 5$, since the upstream router sends a signal to the downstream one to switch on the buffer when it receives the head of a new packet. Then, the head of the new packet reaches the downstream router in 4 cycles considering the optimistic case with zero-contention.

5. CONCLUSIONS

Two NBTI-aware techniques have been described to minimize the impact of NBTI degradation in NoC buffers by reducing the NBTI stress period. Moreover, we analyzed the impact of power gating latency and of process variation using HANDS [15]. The validation has been carried out on both synthetic and real traffic (not shown due to space limitation) patterns and presents a net NBTI mitigation of up to 54.2% with respect to the baseline NoC that does not account for NBTI. The net NBTI mitigation has been computed starting from the collected duty-cycle values on buffers used as input in the NBTI model presented in [6]. From experimental results, *sensor-wise* emerges as the most suitable technique, when heavy process variation effects are expected or when several VC buffers are present, since it can focus the recovery on the most degraded one.

Acknowledgments

Work partially supported by EC Seventh Framework Programme (FP7/2007-2013), under agreements no. 248716 (`2PARMA` project `www.2parma.eu`).

6. REFERENCES

[1] K. Agarwal and S. Nassif. Characterizing process variation in nanometer cmos. In *44th ACM/IEEE DAC.*, june 2007.

[2] N. Agarwal, T. Krishna, L.-S. Peh, and N. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *ISPASS*, 2009.

[3] A. Banerjee, R. Mullins, and S. Moore. A Power and Energy Exploration of Network-on-Chip Architectures. In *NOCS '07*, pages 163–172. IEEE Computer Society, 2007.

[4] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula. Predictive modeling of the nbti effect for reliable design. In *Custom Integrated Circuits Conference. CICC '06.*, pages 189 –192, sept. 2006.

[5] D. Bild, G. Bok, and R. Dick. Minimization of nbti performance degradation using internal node control. In *Design, Automation Test in Europe Conference Exhibition, DATE*, pages 148 –153, 2009.

[6] T.-B. Chan, J. Sartori, P. Gupta, and R. Kumar. On the efficacy of nbti mitigation techniques. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011.*

[7] X. Fu, T. Li, and J. Fortes. Architecting reliable multi-core network-on-chip for small scale processing technology. In *Dependable Systems and Networks (DSN), IEEE/IFIP Int. Conf. on*, 2010.

[8] A. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *DATE '09.*, pages 423 –428, 2009.

[9] M.-C. Lee, Y.-G. Chen, D.-K. Huang, and S.-C. Chang. Nbti-aware power gating design. In *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific.*

[10] B. Li, L.-S. Peh, and P. Patra. Impact of process and temperature variations on network-on-chip design exploration. In *NoCS 2008 ACM/IEEE International Symposium on.*

[11] L. Li, Y. Zhang, J. Yang, and J. Zhao. Proactive nbti mitigation for busy functional units in out-of-order microprocessors. In *DATE*, 2010.

[12] C. Nicopoulos, S. Srinivasan, A. Yanamandra, D. Park, V. Narayanan, C. Das, and M. Irwin. On the effects of process variation in network-on-chip architectures. *Dependable and Secure Computing, IEEE Transactions on*, 2010.

[13] T. Siddiqua and S. Gurumurthi. A multi-level approach to reduce the impact of nbti on processor functional units. In *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, GLSVLSI 2010, New York, NY, USA. ACM.

[14] P. Singh, E. Karl, D. Sylvester, and D. Blaauw. Dynamic nbti management using a 45 nm multi-degradation sensor. *Circuits and Systems I, IEEE Transactions on*, 58(9):2026 –2037, 2011.

[15] D. Zoni, S. Corbetta, and W. Fornaciari. Hands: Heterogeneous architectures and networks-on-chip design and simulation. In *IEEE ISLPED'12*, aug. 2012.