Boomerang: Reducing Power Consumption of Response Packets in NoCs with Minimal Performance Impact

Zhen Fang, Erik G. Hallnor, Bin Li, Michael Leddige, Donglai Dai, Seung Eun Lee, Srihari Makineni, Ravi Iyer

Abstract— Most power reduction mechanisms for NoC channel buffers rely on on-demand wakeup to transition from a low-power state to the active state. Two drawbacks of on-demand wakeup limit its effectiveness: 1) performance impact caused by wakeup delays, and 2) energy and area cost of sleep circuitry itself. What makes the problem harder to solve is that solutions to either problem tend to exacerbate the other. For example, faster wakeup from a power-gated state requires greater charge/discharge current for the sleep transistors while using nimbler sleep transistors implies long wakeup delays. As a result, powerdowns have to be conservatively prescribed, missing many power-saving opportunities.

We propose Boomerang, a novel power-saving method that overcomes the above drawbacks. Specifically, based on the observation that a response is always preceded by a request, we let the *request* trigger wakeup of the buffer that is to be used by its *response* in the (near) future, instead of using on-demand wakeups. Hiding the wakeup delay completely, Boomerang allows us to employ aggressive sleep policies and use low-cost power gating circuits on response buffers.

Index Terms— Interconnection networks, low-power design, packet-switching networks.

1 Introduction

Channel buffer leakage has been identified by a number of independent studies as the single most significant source of energy dissipation on network-on-chips (NoCs) after clock gating is used [3, 5]. The total buffer capacity and consequently total buffer leakage increase proportionally with the network diameter. In this paper we propose a novel hardware solution for this problem.

There are two fundamental issues for any hardware power consumption reduction mechanism: 1) performance impact as a result of low-power state exit delays, and 2) energy and area cost of activation/deactivation circuitry itself. The most widely used hardware method to reduce flit buffer power uses idle counter-based deactivation and on-demand activation. When a counter reaches a threshold value (e.g., 1000 successive clock cycles of idleness), the controlled structure is deactivated by techniques like clock gating and power gating[Soteriou04]. The rationale for counter-based deactivation policy is that future traffic patterns are same as the past, an assumption that may or may not hold in real-life applications. Clock gating is straightforward to implement, but its benefits are diminishing too, as leakage power starts to dominate for deep-submicron processes. In the rest of the paper, we limit our discussions to power gating.
A packet that needs to use a deactivated buffer causes the control circuit to wake it up. Such on-demand wakeup incurs a performance penalty. Usually, the deeper the sleep mode, the bigger the wakeup latency is required, and the greater transition power is dissipated. For example, as the most widely used power gating technique to cut leakage, sleep transistors are used to set the supply voltage of a block of logic [8]. For a sleep transistor, the switching speed $\propto$ channel width $\propto$ charge and discharge energy dissipation. This poses a challenge in power-gating implementations: using a bigger sleep transistor minimizes performance penalty but it incurs big instantaneous power dissipation surge and large real estate area cost, and vice versa. To avoid excessive toggles of the power states, power gatings have to be conservatively prescribed, missing many power-saving opportunities. Thus, knowing when a packet is to arrive is crucial to the effectiveness of hardware power saving solutions. Some designs use special packets on an always-on sideband network to pass this information, e.g., from the source to routers that are hops away downstream [4]. Such a global sideband network introduces high hardware cost and tends to break design modularity.

2 Boomerang: Requests As Harbingers of the Responses

A boomerang is a throwing stick that returns to its point of origin when thrown correctly. It is actually what most of the on-chip transactions resemble, the easiest example being a processor module sends out a read request and the data reply comes back to the requesting processor from the memory module. Indeed, every response is preceded by a request. Here we use the terms request/response in the sense of proactive/reactive. For example, a write followed by a write completion is also a request/response pair, in which the write command and data payload are the request and the write completion is the response.

The key observation we make is that a request provides "oracle" information that a response will soon come back to the same router.

All the authors except Lee are with Intel Corporation. Seung Eun Lee is with Seoul National University of Science and Technology
Contact author: Zhen Fang, Email: *zhen.fang@intel.com*

Manuscript submitted 20-Aug-2010 Manuscript accepted 09-Sep-2010. Final manuscript received 10-Sep-2010.

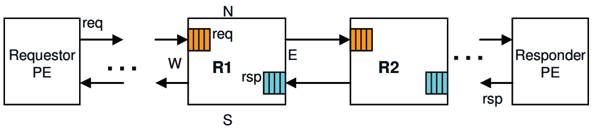

Fig. 1. Overview of Boomerang.

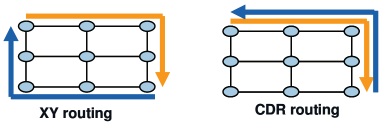

Fig. 2. CDR Routing[Abts09]: same route for a request and its response

2.1 Overview

The boomerang analogy leads us to a solution in which an arrival of a request turns on local resources used by the responses ahead of their actual usage. Specifically, we propose a NoC router buffering mechanism that consists of the following three ingredients.

1) Message class based VCs: Buffers at each port are partitioned into two groups: request virtual channels and response virtual channels.

2) Power-off condition: Deactivate a response buffer as soon as it gets empty.

3) Power-on event: Arrival of a request activates the response buffer in the same router that its response will soon use.

Boomerang wakes up the response buffers ahead of use. Thus the latency of exit from a low-power state is hidden. Compared with on-demand wakeup, performance penalty is minimized.

With stringent latency requirement lifted, there are two implications that lead to maximized power savings. First, we can go to low leakage states more aggressively without suffering from performance penalty. Second, slower and less expensive power control circuitry can be used. E.g., sleep transistors are huge transistors. In Intel's Polaris 80-core chip [4], single-cycle-delay sleep transistors increase the whole chip area by 5.4%. Boomerang allows us to use smaller sleep transistors, which not only takes less area, but also consumes less charge/discharge energy.

This work does not increase or decrease the power consumption or performance of request transfers. Since request payloads (which are usually commands) are typically much smaller than responses (which are usually data replies), response VC buffers are much larger than request buffers, giving us ample opportunities to optimize the chip power.

Fig.1 shows an example in which each router has four ports (E, W, S, N). Each input port has a request VC and a response VC. Let's assume that the response buffer in router R1's East port is sleeping (turned off earlier when they had become empty) when a request arrives. Suppose the request is routed to leave R1 from the East *egress* port, then Boomerang starts the wakeup process of the response buffer in R1's East *ingress* port. By the time that the response packet arrives at R1 from R2, the response queue in R1 will have already been in the active state, ready to take incoming flits – the wakeup latency is completely hidden. In the whole process, only local information is used by Boomerang to power-off and power-on the response VC buffer. We discuss more implementation details in Section 2.2.

2.2 Design Details

In this section we present more details on the Boomerang mechanism.

1) Virtual Channels

In some existing router designs (e.g., Intel 48-core SCC NoC[6]), one VC is reserved for requests and one is for responses, for deadlock avoidance purposes, while the rest of the VCs (8 in the case of Intel SCC) are shared by both types. In our design, no VC is shared by both requests and responses; half of the VCs are for requests only and the other half are for responses only. Most of the requests are small packets and most responses are large packets. For example, a data reply is often more than 6 times larger than a memory read request. Since the total number of requests is about equal to that of responses, total SRAM capacity of the response VCs should be several times larger than that of the request VCs.

The easiest way to differentiate between request and response is to encode an extra request/response bit for each flit. A side effect of the req/response bit is that it can reduce the size of the VC allocation arbiter by half, since a request flit only uses the request VC buffers and a response flit only uses the response VC buffers.

2) Routing

Localized request-triggered response buffer wakeup requires that a response packet always travel through the same routers as what its request packet went through. In XY routing, the request and response routes are guaranteed to be NOT the same. In Boomerang, we use Class-based Deterministic Routing (CDR)[1], in which requests are routed using XY turns and responses using YX turns. CDR routing guarantees that a pair of request and response take the same path if one packet's destination is the other's source. This routing method has also been shown to be one of the best algorithms for NoCs [1].

3) Power-Gating

The power-off condition of the proposed scheme is straightforward; as soon as a response buffer is empty, we cut off its Vcc. Boomerang guarantees that such an aggressive power-off policy does not impact applications' performance or offset the power saving benefits for sleep transistors' charge/discharge current. This guarantee is provided by our novel power-on mechanism: since response buffer activation can start as early as its request appears in the same router, the Vcc fire up delay can be completely hidden. This also allows us to use much narrower sleeping transistors which use less energy on state transitions but take longer time to fire up Vcc. For example, our experience showed that a 50% reduction in sleep transistor size can be achieved by increasing the latency from 1-cycle wakeup to a 5-cycle wakeup with minimal (<5%) frequency impact to the buffers.

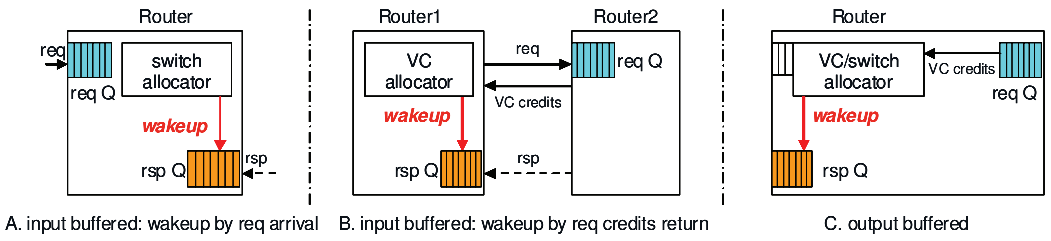

Fig. 3 Options for the Response Buffer Wakeup

4) Wakeup Implementations

Depending on whether input or output buffers are used in the router, the wakeup process can be triggered by different events. If the router uses input buffering, we can either start to wakeup the response buffer when the request arrives (Fig. 3A), or when the credits used by the request return (Fig. 3B). The Fig. 3A implementation turns on the sleeping buffer earlier while Fig. 3B allows more power savings. If the router uses output buffering, request arrival and request credit return occur at about the same time, shown in Fig. 3C.

5) Distance-aware Delayed Wakeup

Immediately starting response buffer wakeup upon arrival of a request packet is usually not necessary, since the response packet may need to travel multiple hops and go through processing element delays. In Fig. 4, suppose a

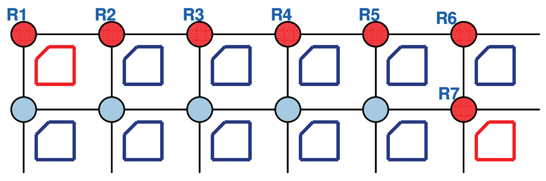

Fig.4 Motivation for Distance-Aware Delayed Wakeup.

read request going towards R7 arrives at R1. If the request is sunk at the PE connected to R7 then the response packet (which is the fetched memory block) will not arrive at R1 until after

$T_{memory} + 12 * T_{hop}$, where T_{memory} is memory delay and T_{hop} is per-hop router delay. A natural extension to the basic Boomerang mechanism is to introduce a distance-aware delay for each port. Ideally, response buffer wakeup in router R_m should be delayed by

$T_{processing} + \texttt{distance} * 2 * T_{hop} - T_{wakeup}$

where `distance` is the number of hops between R_m and destination of the request, $T_{processing}$ is the processing time at the destination node, and T_{wakeup} is the latency of low power state exit. Examples of $T_{processing}$ include 100 ns for DDR memory read, and 8 ns for a last-level cache access. Hardware logic for obtaining accurate $T_{processing}$ estimates may offset the benefits of cutting response buffer leakage power. In practice, we only need to map destination node IDs to a few pre-defined fixed values for the delay timer. The cost for the delay assignment and decrementing counter is just a small fraction of the response buffer.

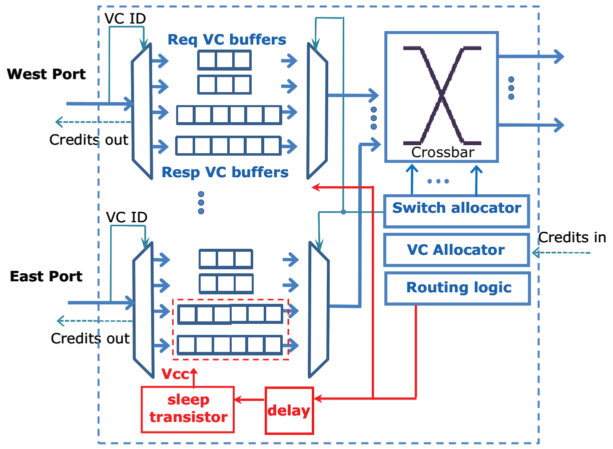

Fig.5 Router with Response Buffers Controlled by Delayed Wakeup

Fig.5 shows a router microarchitecture with the proposed new logic in red color. In this example, sleeping response buffers are woke up by arrivals of requests instead of by arrival of responses. Specifically, the routing logic extracts delay amount based on destination node ID in the request header and sets up the delay timer. When the delay timer expires, the sleep transistor is flipped to enable normal supply voltage to the response buffer that the corresponding reply will soon use.

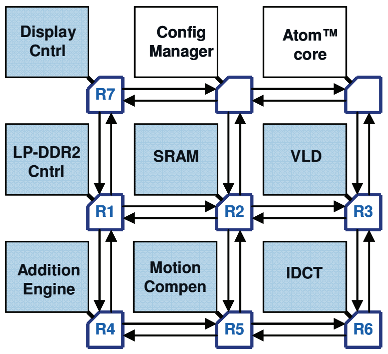

Fig. 6 NoC in a Mobile SoC Platform (MPEG2 Video Decoder)

Table 1. Evaluation of Boomerang on a Media SoC

		Baseline (clock-gated)		**Leakage Optimized with Boomerang**		**Power Savings, *Boomerang***		**Power Savings, *Thresh = 81 ns***	
Router	**Router Utilization**	**buffer leakage**	**router total**	**buffer leakage**	**router total**	**buffer leakage**	**router total**	**buffer leakage**	**router total**
R1	26%	0.61 mW	1.21 mW	0.34 mW	0.94 mW	44%	22%	25%	13%
R2	21%	0.61 mW	1.10 mW	0.32 mW	0.81 mW	47%	26%	27%	15%
R3	6%	0.61 mW	0.75 mW	0.27 mW	0.41 mW	56%	46%	8%	7%
R4	10%	0.61 mW	0.84 mW	0.28 mW	0.51 mW	54%	39%	31%	22%
R5	40%	0.61 mW	1.54 mW	0.39 mW	1.32 mW	36%	14%	21%	8%
R6	25%	0.61 mW	1.19 mW	0.34 mW	0.91 mW	45%	23%	6%	3%
R7	10%	0.61 mW	0.84 mW	0.28 mW	0.51 mW	54%	39%	31%	22%
Total or Mean	**16.5%**	**4.29 mW**	**7.47 mW**	**2.22 mW**	**5.41 mW**	**47.7%**	**28.1%**	**18.3%**	**10.8%**

2.4 Evaluation

In a reconfigurable low-power SoC with strong media processing capabilities, we use a 2D mesh NoC as the system interconnect. In an MPEG2 playback configuration of this SoC shown in Fig.6, majority of the traffic travels through 7 tiles, shown in shaded boxes.

We use an in-house network-on-chip performance simulator to gather performance statistics. Then we dial in power parameters (obtained through MOSIS) to calculate power consumption. In our experiment setup, clock frequency = 300 MHz; flit width = 64b; 4 flits per request VC, 24 flits per response VC; each port has 2 request VCs and 2 response VCs. We assume 45nm low-leakage process with a supply voltage of 1.0V. The baseline flit buffers use aggressive clock-gating to control dynamic power. We wake up a sleeping response buffer as the corresponding request arrives (Fig. 3A). Exit latency from the low-leakage state is 6.6ns.

For the workload, we use synthetic memory traces that mimic playback of 1080p, 25fps MPEG2 video (4:2:0, MP@HL). Table 1 shows that the seven routers have rather different traffic. For example, R5 is the busiest router, with an average memory bandwidth of over 8X of R3. The two un-numbered routers are not actively used in this workload so we omit them from our evaluation.

Based on our experiments, Boomerang was able to reduce buffer leakage power by 47.7% (geomean), which corresponds to total router power savings of 28.1%. As a comparison, we also experimented with a history-based opportunistic powerdown with idle threshold = 81ns. In this scheme, a response buffer enters low-leakage state after it has been idle for 81ns. It uses on-demand wakeup. On-demand wakeup incurs performance penalty (not shown in the table), and history-based opportunistic powerdown often makes incorrect decisions on when to put a buffer to sleep mode. Boomerang literally eliminates both problems, and has a significant advantage over the traditional history-based opportunistic powerdown.

3 Summary and Future Work

The Boomerang mechanism avoids performance impact typically associated with on-demand wakeup by waking up response buffers before their service is needed. This allows us to use aggressive sleep policies and smaller sleep transistors. Using a NoC in a media SoC, our simulation shows that Boomerang is significantly more effective than the traditional scheme.

As a part of the future work, we plan to evaluate Boomerang's efficacy on a cache-coherent CMP (e.g., using a directory or a snoopy protocol [2]) using standard benchmarks. Depending on the implementation of the protocol, the percentage of messages that can be request-response paired varies. For example, a 4-hop implementation of the directory-based protocol and a home-snooping protocol will see a high percentage (e.g., > 95%) of the response messages optimizable while a 3-hop directory protocol or a source-snooping protocol may not be able to benefit from Boomerang as much.

References

[1] D. Abts, et al., "Achieving Predictable Performance through Better Memory Controller Placement in Many-core CMPs," ISCA 2009, pp.451-461

[2] N. Agarwal, L.Peh, and N.Jha, "In-Network Snoop Ordering (INSO): Snoopy Coherence on Unordered Networks", HPCA-15, 2009, pp.67-78

[3] X. Chen and L. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks", ISLPED 2003, pp.90-95

[4] Y. Hoskote, et al., "A 5Ghz Mesh Interconnect for a Teraflops Processor", IEEE Micro 27(5), 2007, pp.51-61

[5] H. Matsutani, et al., "Adding Slow-Silent Virtual Channels for Low-Power On-Chip Networks", NOCS 2008, pp.23-32

[6] P. Salihundam, et al., "A 2Tb/s 6x4 Mesh Network with DVFS and 2.3Tb/s/W router in 45nm CMOS", VLSI Symposium 2010

[7] V. Soteriou and L. Peh, "Design-Space Exploration of Power-Aware On/Off Interconnection Network", ICCD 2004, pp.510-517

[8] J. Tschanz, S. Narendra, Y. Ye, B. Bloechel, S. Borkar and V. De, "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors", IEEE JSSC, 38(11), Nov. 2003. pp 1838-1843