Run-Time Power Gating of On-Chip Routers Using Look-Ahead Routing *

Hiroki Matsutani† Michihiro Koibuchi‡ Daihan Wang†
Hideharu Amano†

†Keio University, 3-14-1, Hiyoshi, Kohoku-ku, Yokohama, 223-8522, JAPAN
‡National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, JAPAN
E-mail: matutani@am.ics.keio.ac.jp

Abstract— Since on-chip routers in Network-on-Chips play a key role in on-chip communication between cores, they should be always preparing for packet injections even if a part of cores are in standby mode, resulting in a larger standby power of routers compared with cores. The run-time power gating of individual channels in a router is one of attractive solutions to reduce the standby power of chip without affecting the on-chip communication. However, a state transition between sleep and active mode incurs the performance penalty, and turning a power switch on or off dissipates the overhead energy, which means a short-term sleep adversely increases the power consumption. In this paper, we propose a sleep control method based on look-ahead routing that detects the arrival of packets two hops ahead, so as to hide the wake-up delay and reduce the short-term sleeps of channels. Simulation results using real application traces show that the proposed method conceals the wake-up delay of less than five cycles, and more leakage power can be saved compared with the original naive method.

I. Introduction

Network-on-Chips (NoCs) have emerged as a promising approach to connect a number of processing cores on a single chip, by introducing a network structure similar to that in parallel computers[4]. They have been utilized not only for high-performance microarchitectures but also for cost-effective embedded devices mostly used in consumer equipment such as set-top boxes or mobile wireless devices.

Power consumption is one of the major concerns in such applications, since it affects their battery life or packaging costs for heat dissipation. It is divided into dynamic switching power and static leakage power, and various saving techniques have been used, such as clock gating and operand isolation for switching power, and multi-threshold voltages and power gating for leakage reduction. Leakage power has already become a major component of power consumption in recent process technologies, and it will further increase while switching power becomes smaller when the technology is scaled down.

Leakage reduction techniques have been widely researched for application processors used in mobile phones, and their standby power is very small (e.g., the leakage current is only 11μA in ultra standby mode [8]). In other words, the more leakage reduction such processor cores achieve, the more impact the leakage power of on-chip routers will give; therefore leakage reduction techniques are essential for on-chip routers as well as processing cores in order to reduce the standby power of the chip.

The run-time power gating of individual channels in a router is one of attractive solutions to reduce the standby power of chip if it does not affect the on-chip communication. However, a state transition between sleep and active mode incurs the performance penalty, and turning a power switch on or off dissipates the overhead energy, which means a short-term sleep adversely increases the power consumption. In this paper, especially for simple on-chip wormhole routers, we propose a sleep control method based on the look-ahead routing that detects the arrival of packets two hops ahead, so as to hide the wake-up delay and reduce the occasion of short-term sleeps which adversely increase the power consumption.

*A part of this work was supported by Toshiba Semiconductor Company, NII, and JST CREST. The authors thank to VLSI Design and Education Center (VDEC) and Kyoto University for the design flow of ASPLA/STARC 90nm CMOS process.

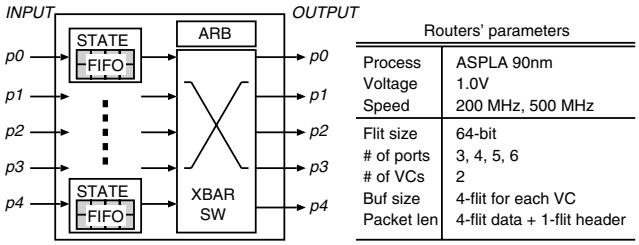


Fig. 1. Wormhole router architecture used

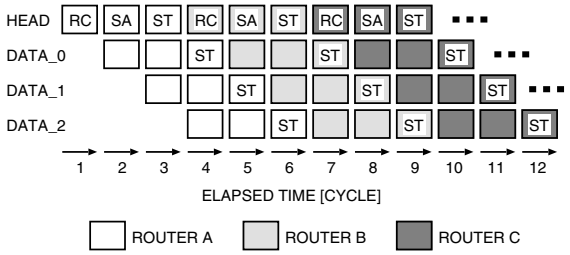


Fig. 2. Router pipeline (a packet is transferred from router A to C)

The rest of this paper is organized as follows. Section II analyzes power consumption of an on-chip router, and discusses the need for run-time power gating of individual channels in the router. Section III surveys run-time power gating techniques on microprocessors. The sleep control method based on look-ahead routing is proposed in Section IV and is evaluated in Section V. Conclusions are derived in Section VII.

II. Power Analysis of On-Chip Router

A. On-Chip Router Architecture

Prior to discussing low power techniques for on-chip routers, we analyzed power consumption of a typical on-chip router used in [9]. Fig.1 shows the wormhole router architecture used in this analysis. This router consists of a crossbar switch (XBAR), an arbitration unit (ARB), and input physical channels (or ports). Each physical channel has two virtual channels[6], each of which has a FIFO buffer for storing four 64-bit flits. This is a typical input buffered router, which has buffers only at its input channels, not output channels. In this design, the FIFO buffers were implemented with small registers, and up to 75% of the total router area was used for them.

The router architecture was fully pipelined, and it transferred a header flit through three pipeline stages that consisted of routing computation (RC), virtual channel and switch allocation (SA), and switch traversal (ST). Fig.2 shows the router pipeline, in which a packet consisting of a single header flit and three data flits is transferred from router A to router C.

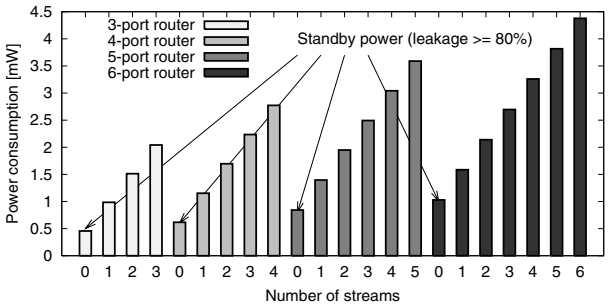


Fig. 3. Router power at fixed throughputs (200MHz)

Table I
Breakdown of standby power on the 5-port router (200MHz, 500MHz)

	200MHz		500MHz	
	dynamic	leakage	dynamic	leakage
Crossbar & others	13μW	207μW	33μW	207μW
Channels (FIFOs)	146μW	476μW	365μW	476μW
Total	842μW		1,081μW	

B. Power Analysis

To estimate the power consumption of the router mentioned previously, the following steps were performed: 1) synthesis by Synopsys Design Compiler, 2) place and route with buffer insertions at CTS using Synopsys Astro, 3) post place-and-route simulation by Cadence Verilog-XL to obtain switching activity information of the router, 4) power analysis based on the switching activity using Synopsys Power Compiler. A 90nm CMOS process with a core voltage of 1.0V was selected in this analysis. Clock gating and operand isolation were fully applied to the router to minimize its switching activity.

In the step 3), the router was simulated at 200MHz and 500MHz with various fixed workloads (throughputs), in the same manner as [3]. A packet stream is defined as intermittent injections of packets, which utilize about 30% of the maximum link bandwidth of a single router link. Each header flit contains a fixed destination address, while data flits contain a random payload. The number of packet streams injecting into the router was changed so as to generate various workloads. In this experiment, up to n streams were applied for a router that has n physical channels (i.e., an n-port router), and power consumption at each workload level was analyzed, where $3 \leq n \leq 6$.

Fig.3 shows the results at 200MHz. As we expected, the router consumes more power as it processes more packet streams. Notice the router consumes a certain standby power even with no traffic (i.e., 0-stream). Table I shows the breakdown of standby power on the 5-port router. Leakage power consumes a substantial portion of the standby power (e.g., 81.1% of the standby power at 200MHz). Particularly, the leakage power consumed by input channels is the largest. The remaining power is dynamic power consumed by latches inserted for clock gating; so further reduction of dynamic power would be difficult.

Processor cores can be stopped if they are not used, though it takes millisecond order time to wake-up. However, on-chip routers must be always preparing for packet injections even if a part of cores are in standby mode, since they play a key role for enabling on-chip communication between cores; thus the run-time power gating of individual channels in a router is one of attractive solutions to reduce the standby power of chip without affecting the on-chip communication. Run-time power gating techniques are surveyed in the next section.

III. Run-Time Power Gating

Power gating technique shuts off the power supply of idle blocks by turning off (or on) power switches inserted between V_{dd} line and the blocks or between GND line and the blocks. This concept has been applied to circuit blocks with various granularities, such as processor cores[8], execution units in a processor[7][1], and primitive gates[11]. In this paper, we focus on the execution unit level, since its granularity is close to our router channels.

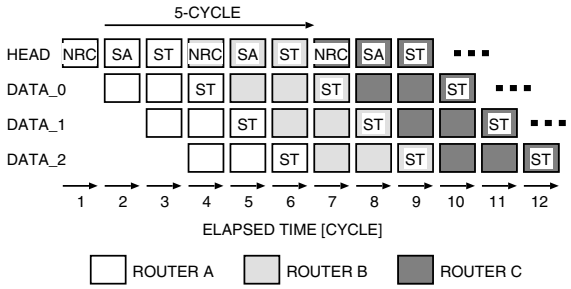


Fig. 4. Pipelined router with look-ahead routing

We need to understand both negative and positive impacts of power gating when we use it. Actually, a state transition between sleep and active mode incurs the performance penalty, and turning a power switch on or off dissipates the overhead energy, which means a short-term sleep increases the power consumption. Reference [7] provides an analytical model of run-time power gating of execution units in a microprocessor. The following three parameters quoted from [7] affect performance and energy.

- T_{wakeup}: Number of cycles required to charge up the local voltage of a sleeping block. Delay for turning on its power switch is also lumped into T_{wakeup} value.
- $T_{idledetect}$: Number of cycles required to detect an idle duration at an active block and decide to shut off the block. Delay for turning off its power switch is also lumped into $T_{idledetect}$ value.
- $T_{breakeven}$: Number of sleep cycles at least required to compensate the overhead energy for turning the power switch on and off.

T_{wakeup} value affects the performance (e.g., traffic throughput on routers), since a pipeline stall will be caused if a new request suddenly comes to a sleeping block. Also, $T_{idledetect}$ shortens the sleep duration of blocks, since an idle block must stay in the active state for $T_{idledetect}$ cycles before it decides to go to the sleep mode.

A short-term sleep of less than $T_{breakeven}$ cycles cannot compensate the energy overhead of driving a power switch, and power consumption will be increased; thus we need to avoid such short-term sleeps.

Although $T_{breakeven}$ value depends on various parameters, such as sizes of a power switch and a decoupling capacitance, Reference [7] provides $T_{breakeven} \approx 10$ based on typical parameters on a typical microprocessor. In the same manner as [7], we also estimate $T_{breakeven}$ value on our on-chip router mentioned previously (see Section V.D).

IV. Sleep Control Using Look-Ahead Routing

In this section, we propose a sleep control method based on look-ahead routing that detects the arrival of packets two hops ahead, so as to mitigate the negative impacts of T_{wakeup} and short-term sleeps of less than $T_{breakeven}$ cycles.

We first explain the general look-ahead routing, and then we propose our method for deterministic routing. Extensions for adaptive routing are also shown in this section.

A. Look-Ahead Routing

Look-ahead routing has been used for several purposes, such as reducing pipeline stages for low latency communication[6]. Here we just explain the differences between a look-ahead router and a normal router without look-ahead from the viewpoint of their router pipeline structure.

In the case of normal routers, an output channel at i-th hop router is selected by the RC stage of the same router (i.e., i-th hop router), as shown in Fig.2.

In the case of one-hop look-ahead routers, on the other hand, an output channel at i-th hop router is selected by $(i-1)$-th hop router. That is, each router performs a routing computation for the next hop (denoted as NRC), as shown in Fig.4.

B. Basic Sleep Control

We apply the look-ahead routing for early detections of both idle and busy periods on each router channel. Basic sleep control without look-ahead routing is preliminarily shown here.

Each input channel monitors a "request" signal, which indicates that new packets (or requests) are approaching the input channel, and it is used for making a decision to go to the active mode. That is, if the request signal for a sleeping channel is asserted, then the channel starts the wake-up procedure to receive new packets. The request signal is also used for making a decision to go to the sleep mode. Each input channel checks the request signal after forwarding a packet in the channel, and if the request signal is de-asserted, the channel will go to the sleep mode.

Obviously, performance penalty due to the wake-up delay depends on how early the new requests can be detected.

C. Look-Ahead Sleep Control on Deterministic Routing

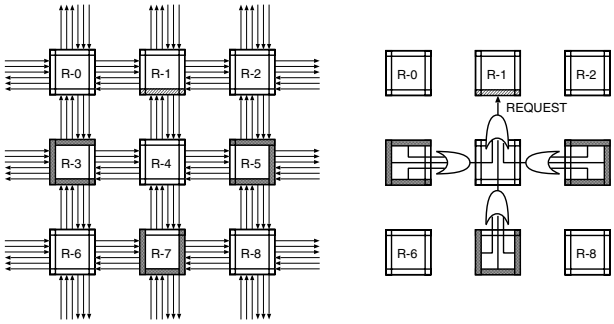
Routing algorithms are categorized as deterministic and adaptive. All routing paths are statically fixed in deterministic routing, while they are dynamically changed in adaptive routing in response to network conditions. Since a simple deterministic routing such as dimension-order routing (DOR) [6] has been widely used in NoCs, we mainly explain our sleep control method focusing on deterministic routing. Extensions for adaptive routing are summarized in Section IV.D.

As we stated previously, the output channel at i-th hop router is determined at the NRC stage of $(i-1)$-th hop router, in the case of look-ahead routers with deterministic routing. This means that, at the same time, we can also find which input channel at $(i+1)$-th hop router is going to be used, because the output channel of i-th hop router is directly connected to the input channel of $(i+1)$-th hop router. For example, during cycle 1 in Fig.4, the NRC stage of router A detects which input channel of router C should be power-gated or waked-up from now. There is a five-cycle margin between the NRC stage at router A and the packet arrival at router C; so our proposed sleep control method can reduce the performance penalty, or remove that when $T_{wakeup} \leq 5$.

However, the proposed method requires the following two modifications on look-ahead routers; 1) every input channel needs to assert (or de-assert) the request signals to all input channels which are located in two hops far from the channel; 2) every input channel needs to receive all request signals from input channels which are located in two hops far from the channel. Additional hardware amount required for these modifications is modest, since they can be implemented with simple combinational logic, which is much smaller than the buffers. [1]

Fig.5 illustrates the sleep control signals for the South channel of router 1. Let us call the South channel of router 1 a "target". The target channel monitors the request signals from input channels in router 3, 5, and 7 to make a decision to sleep or wake-up. Notice that the target channel does not need to monitor several input channels, since "U-turn" paths that introduce livelocks are usually prohibited even with heavy traffic. That is, the target channel has only to monitor the input channels colored in Fig.5(a).

[1] Although we omitted such considerations from the evaluation section due to the page limitations, we implemented the look-ahead based sleep controller on the router, and estimated the hardware overhead (in gates) is only 1.08% of original. Notice this estimation does not include power switches, which will further increase the total area.



(a) Input channels in the NoC (b) Request network using OR-gates

Fig. 5. Request signal for the South channel of router 1 (R denotes a router)

The colored input channels assert their request signal for the target channel when they detect a packet destined for router 1, in order to notify that the packet will be arriving at the target at least five cycles later (Fig.5(b)). As shown, these request signals cover the twice longer length than normal links between routers cover. Although the critical path of the target router is not related to the sleep controllers (it is related to the arbitration logic of output ports), the wire delay of these request signals will be increased as the distance between routers increases. On the other hand, these request signals can be pipelined and be transferred in two cycles in order to mitigate the wire delay. In such cases, each channel can still detect the packet arrivals four cycles ahead.

D. Extensions for Adaptive Routing

In the case of adaptive routing, the RC stage provides several candidates of output channels, and one of them is selected based on an output selection function (OSF); thus look-ahead mechanism cannot detect the arrival of packets two hops ahead.

Here, in order to save leakage power of routers' channels even with adaptive routing, we proposed to change the OSF dynamically in response to the network load, as follows:

- At heavy load: Randomly selects a channel every time
- At light load: Continuously selects the same channel

Clearly, the former OSF improves the network throughput compared with the latter one, but the arrival of packets can be correctly detected in the case of the latter OSF.

At light network load, the network throughput is limited by the amount of packet injections, not OSFs; thus performance penalty of the latter OSF is negligible, and look-ahead sleep control with the latter OSF can save leakage power.

At heavy load, on the other hand, there are few opportunities for power gating in most cases due to the frequent packet arrivals, so the former OSF without power gating would be reasonable.

V. Evaluations

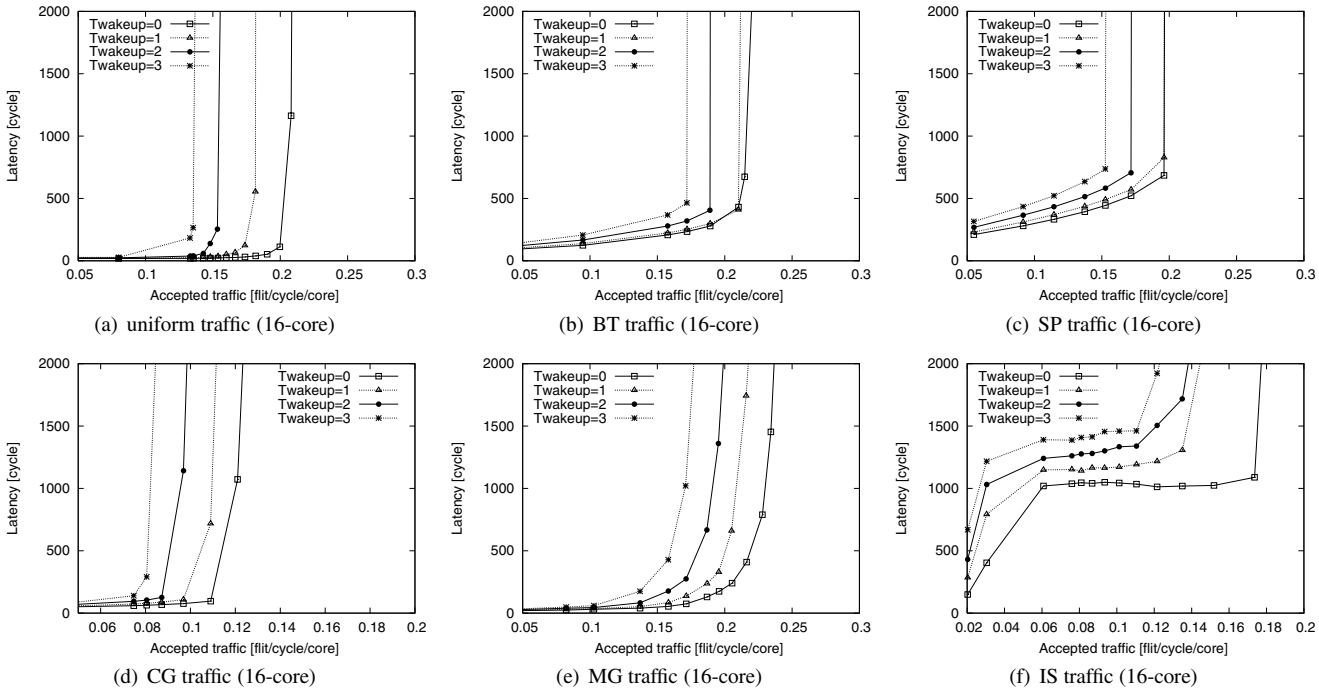
The following three sleep control methods (including the look-ahead based one) were evaluated in terms of performance impact of wake-up delay and leakage power saving.

- **ideal:** Power gating with no overhead (equivalent to the ideal case where $T_{wakeup} = 0$ and $T_{idledetect} = 0$)
- **lookahead:** Power gating with look-ahead based sleep control, which detects the packet arrivals five cycles ahead
- **naive (original):** Power gating with naive sleep control, which detects the packet arrivals zero cycle ahead

A. Performance Impact of Wake-Up Delay

To show the impact of the wake-up delay on the performance, we evaluated the network throughput when power gating with *naive* method was applied for various T_{wakeup} values.

A flit-level network simulator written in C++ was used for this experiment. A simple model corresponding to the wormhole router mentioned in Section II.A was used as a switching



(a) uniform traffic (16-core) (b) BT traffic (16-core) (c) SP traffic (16-core)

(d) CG traffic (16-core) (e) MG traffic (16-core) (f) IS traffic (16-core)

Fig. 6. Performance of naive method ($T_{wakeup} = 0, 1, 2, 3$)

fabric in the simulator. That is, each router had three pipeline stages, and each packet consisted of five flits including one flit header (see Fig.1). Network topology and routing algorithm used in this simulation were 4×4 2-D Mesh and DOR[6].

As for the traffic patterns, we used five application traces and uniform random traffic. These application traces were captured from NAS Parallel Benchmark (NPB)[2] programs, because they would enable us to evaluate with various patterns of real application traffic. We selected five matrix computation programs from NPB: Block Tridiagonal solver (BT), Scalar Pentadiagonal solver (SP), Conjugate Gradient (CG), Multi-Grid solver (MG), and large Integer Sort (IS). The class of problem was set to "W", and the numbers of tasks was 16. In addition to these real applications, we used uniform traffic as a baseline for comparison.

Fig.6 shows the throughput (accepted traffic) versus the latency with *naive* method in the cases where $T_{wakeup} = 0, 1, 2, 3$. In the case of *naive* method, a pipeline stall of T_{wakeup} cycles always occurs when a new packet arrives at a sleeping channel; thus its throughput is reduced as T_{wakeup} value is increased.

Notice that the throughput of *ideal* method is equivalent to the case where $T_{wakeup} = 0$. As for the *lookahead* method, since it can detect the arrivals of packets five cycles ahead as we stated in Section IV, its throughput is also equivalent to the *ideal* if $T_{wakeup} \leq 5$. As a result, *lookahead* method successfully mitigates the performance penalty of wake-up delay.

B. Break-Even Point of Power Gating

$T_{breakeven}$ value determines the benefits of power gating. In this section, we first estimated energy parameters of a channel in our router. Then we calculated $T_{breakeven}$ value of the router channel based on these energy parameters in the same manner proposed in [7], because the circuit block size of the channel would be close to that of an execution unit in a microprocessor.

The router was simulated at 200MHz and 500MHz. Table II shows the energy parameters of the router channel, such as dynamic power, leakage power, and switching activity.

As reported in [7], the total energy saved over N cycles (denoted as E^N_{saved}) is calculated as follows:

$$E^N_{saved} = E_{leak} \frac{\mathrm{DIBL}}{mV_t} \frac{N^2}{2} \frac{\alpha L V_{dd}}{2(\frac{1}{2} + \frac{C_D}{C_S})}, \quad (1)$$

where DIBL is the drain-induced barrier lowering, which is typically close to the value of 0.1, $V_t = kT/q \approx 25$mV is the thermal voltage, and $m \approx 1.3$ [7]. Also, W_H is the ratio of the total area of the power switch to the area of the target block (i.e., the router channel), C_S is the total switching capacitance of the block, and C_D is the total capacitance at the local power supply including the power switch. Typically, W_H is 0.1 and C_D/C_S is 0.5 as reported in [7].

Table II
Parameters of the router channel for energy calculation († Energy values are expressed in Watts to show the differences between 200MHz and 500MHz)

		200MHz	500MHz
V_{dd}	supply voltage	1.0V	1.0V
E_{leak}	static energy †	0.095mW	0.095mW
E_{sw}	dynamic energy †	0.105mW	0.261mW
L	leakage factor ($= E_{leak}/E_{sw}$)	0.91	0.36
α	switching factor	0.10	0.10

As reported in [7], the dynamic energy overhead to turn the power switch (denoted as $E_{overhead}$) is calculated as follows:

$$E_{overhead} \approx 2 \frac{W_H}{\alpha} E_{leak}. \quad (2)$$

Based on Equation 1 and 2, we calculated the average leakage power consumed in an N-cycle sleep, in the cases of 200MHz and 500MHz. Note that the average leakage power consumption in this experiment includes the dynamic energy overhead to turn the power switch. Since the impact of the overhead energy is increased as the sleep duration is shortened, the average leakage power of a power-gated channel would be larger than that without power gating if the sleep duration is very short.

The results are shown in Fig.7. "NoPG" stands for the leakage power of the router channel without power gating, while "PG(200MHz)" and "PG(500MHz)" are the results of the power-gated channel operating at 200MHz and 500MHz, respectively. As shown in the graph, the average leakage

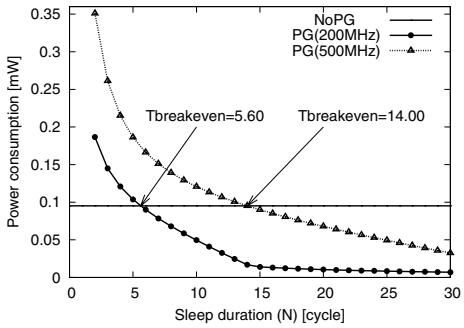


Fig. 7. Leakage power consumption vs. sleep duration

power of PG(200MHz) and PG(500MHz) are decreased as the sleep duration N becomes long. Since the leakage power of NoPG is 0.095mW, we can estimate that $T_{breakeven} \approx 6$ for PG(200MHz) and $T_{breakeven} = 14$ for PG(500MHz).

C. Compensated Sleep Cycles

Although a long-term compensated sleep ($\geq T_{breakeven}$) can save the leakage power consumption, a short-term uncompensated sleep ($< T_{breakeven}$) adversely increases the power. In this section, we show ratio of the compensated sleep cycles.

The total execution time of each input channel in the NoC can be divided into the following three states:

- N_{csc}: Compensated sleep state (duration $\geq T_{breakeven}$)
- N_{usc}: Uncompensated sleep (duration $< T_{breakeven}$)
- N_{active}: Active state (neither N_{csc} nor N_{usc})

To calculate the ratio of N_{csc} over the total execution time, we performed the network simulations in the same manner as Section V.A. We assumed 200MHz and 500MHz cases again. $T_{breakeven}$ was set to 6 for the 200MHz simulation, while it was set to 14 for the 500MHz, as estimated in the previous section. To see the differences of these $T_{breakeven}$ values, T_{wakeup} was fixed to 2 and $T_{idledetect}$ was fixed to 4, in both cases.

Fig.8 shows the results in the case where $T_{breakeven} = 6$. As we expected, the ratio of N_{csc} is reduced as the network load (accepted traffic) increases. Especially *naive* method cannot conceal the wake-up delay, and pipeline stalls are caused more frequently than the other methods. The stalled packets stay long at channel buffers, and the sleep opportunities are reduced. Thus, the ratio of N_{csc} in *lookahead* method is higher than that in *naive* method, and it is comparable to *ideal* method.

Fig.9 shows the results in the case where $T_{breakeven} = 14$. Compared with Fig.8, the ratio of N_{csc} is decreased in all methods. Particularly the sleep opportunities are much reduced in uniform traffic, which is the worst case for the run-time power gating because all channels are evenly used at short intervals.

D. Leakage Power Saving

In the previous section, we focused on the ratio of the compensated sleep cycles, but we did not take account of the energy overhead induced by the uncompensated sleeps. In this section, we estimated the leakage power saving which includes the overhead energy for driving the power switches.

To estimate the leakage power consumption, we obtained the length (i.e., number of cycles) of every sleep in every channel during the total execution time. Then we calculated the average power consumption for each sleep according to its length, based on the "power consumption vs. sleep duration" graph shown in Fig.7. That is, a sleep longer than $T_{breakeven}$ contributes to reduce the average power consumption, while a sleep shorter than $T_{breakeven}$ adversely eats additional power.

Fig.8 and 9 also show the average leakage power consumption over all channels in the NoC. As shown in these graphs, more leakage power can be saved as N_{csc} increases. In other words, the leakage power saving depends on the network load. Since the router channel consumes 0.095mW when power gating is not applied, our proposed *lookahead* based power gating can save 13.1% to 71.1% of leakage power, even at the maximum (peak) throughput. Furthermore, we can see the leakage power saving of *lookahead* method is larger than that in *naive* method. Actually, the results of *lookahead* are close to *ideal*.

In this paper, we studied the run-time power gating of router channels to reduce the standby power of on-chip routers, because the leakage power of channels consumes more than half of the standby power (Table I). Although the leakage power is still small compared with the switching power at the peak throughput (Fig. 3), the impact of leakage power will further increase while switching power becomes smaller as the technology is scaled down.

VI. Related Works

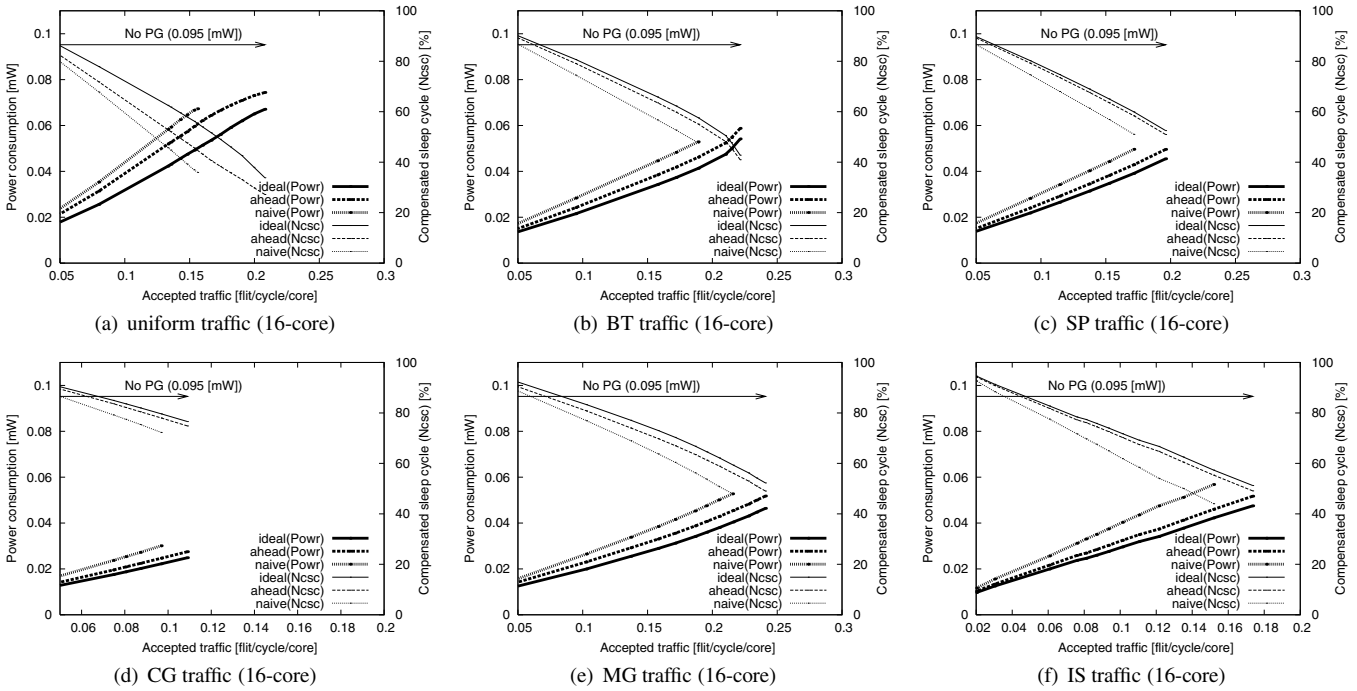
Various low power techniques have been studied for on-chip and off-chip routers. Reference [5] proposes power-aware router buffers based on Drowsy and Gated V_{dd} SRAMs to regulate their leakage power. Reference [10] provides a thorough discussion about power-aware networks whose links can be turned on and off, in terms of connectivity, routing, wake-up and sleep decisions, and router pipeline architecture. These related works are intended not only for off-chip interconnects but also for on-chip interconnects, and they assume to use relatively large buffers in their routers, compared with simple on-chip wormhole routers used in [3] and [9]. In [5], router buffers are constructed with SRAMs and each input port has a 256-flit buffer in the experiment. As a sleep control policy of the buffer, a certain portion (i.e., window size) of the buffer is waked up before it is accessed. By tuning the window size, input ports can always provide enough buffer space when packets arrive, and network performance will never be affected[5].

On the other hand, our light-weight wormhole router uses a small 4-flit buffer for each channel[9], unlike routers that use SRAMs for their input and output buffers. Since the buffer depth is shorter than the window size in the low-cost router, the wake-up delay of buffers directly affects the network performance if links or channels are dynamically turned on and off. Our look-ahead based sleep control method can remove the negative impact of wake up delay on the light-weight on-chip routers by detecting the arrival of packets two hops ahead.

VII. Conclusions

The run-time power gating of individual channels in a router is an attractive solution to reduce the standby power of chip without affecting the performance of the on-chip communication. In order to mitigate the performance penalty of T_{wakeup} and increase the compensable sleep opportunities, we proposed a sleep control method based on the look-ahead routing. Simulation results using real application traces showed that *lookahead* method conceals T_{wakeup} of less than five cycles, and more leakage power can be saved compared with the original *naive* method. Although the technique proposed here is simple, it is essential to mitigate the performance and energy overheads of the run-time power gating on simple on-chip routers.

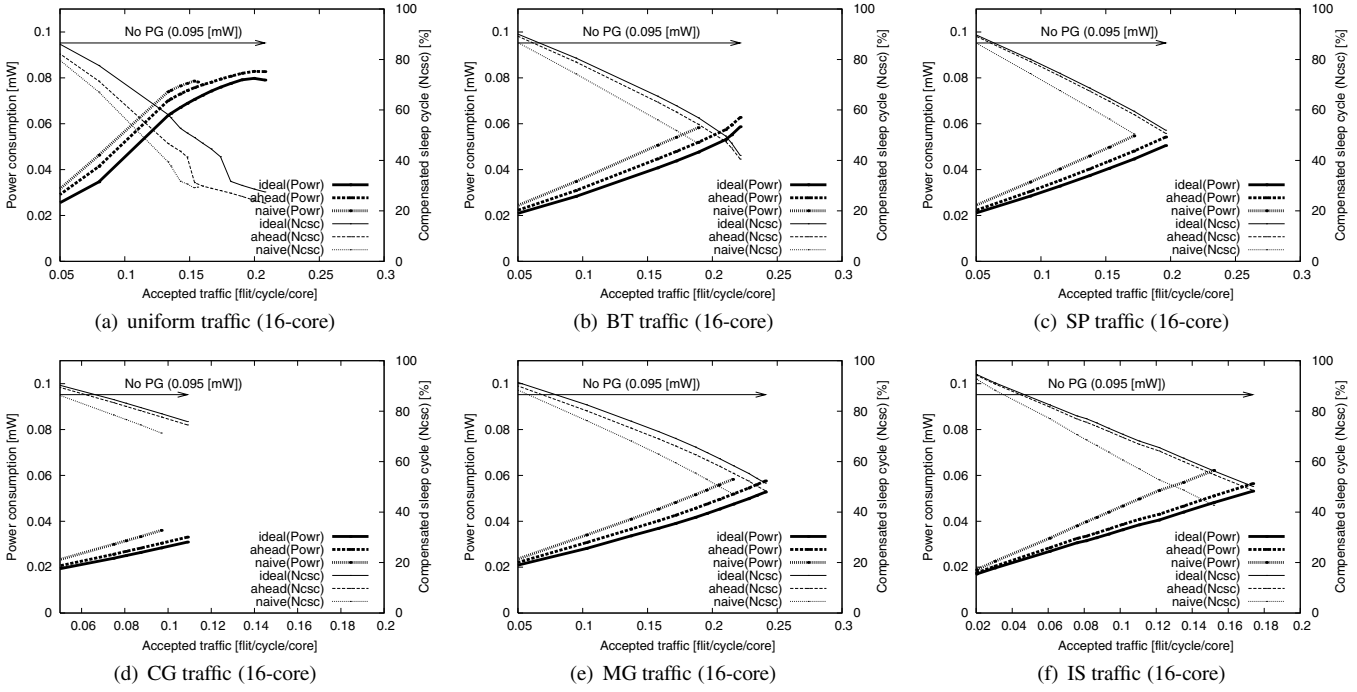
Based on the results provided here, we are planning to explore the power-gating aware router architecture and network topology. We will apply the proposed method to various network topologies such as fat trees that have multiple trees, each of which can be individually power-gated according to the traffic load. In addition, we are developing an on-chip router that has five virtual channels, each of which can be individually power-gated. The virtual-channel mechanism has been considered to increase the leakage power due to its large buffer size while it can provide good performance. The virtual-channel level power gating would overcome such negative impacts. As a result, we can benefit from the high peak performance of virtual channels without increasing the standby power of the chip.



(a) uniform traffic (16-core) (b) BT traffic (16-core) (c) SP traffic (16-core)

(d) CG traffic (16-core) (e) MG traffic (16-core) (f) IS traffic (16-core)

Fig. 8. N_{csc} and leakage power consumption with ideal, lookahead, and naive methods ($T_{wakeup} = 2, T_{idledetect} = 4, T_{breakeven} = 6$)



(a) uniform traffic (16-core) (b) BT traffic (16-core) (c) SP traffic (16-core)

(d) CG traffic (16-core) (e) MG traffic (16-core) (f) IS traffic (16-core)

Fig. 9. N_{csc} and leakage power consumption with ideal, lookahead, and naive methods ($T_{wakeup} = 2, T_{idledetect} = 4, T_{breakeven} = 14$)

References

[1] N. Agarwal and N. J. Dimopoulos. Automated Power Gating of Registers using CoDeL and FSM Branch Prediction. In *Proceedings of the Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 294–303, July 2007.

[2] D. Bailey et al. The NAS Parallel Benchmarks 2.0. *NAS Technical Report, NAS-95-020*, Dec. 1995.

[3] A. Banerjee, R. Mullins, and S. Moore. A Power and Energy Exploration of Network-on-Chip Architectures. In *Proceedings of the International Symposium on Networks-on-Chip*, pages 163–172, May 2007.

[4] L. Benini and G. D. Micheli. *Networks on Chips: Technology And Tools*. Morgan Kaufmann, 2006.

[5] X. Chen and L.-S. Peh. Leakage Power Modeling and Optimization in Interconnection Networks. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 90–95, Aug. 2003.

[6] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[7] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose. Microarchitectural Techniques for Power Gating of Execution Units. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 32–37, Aug. 2004.

[8] M. Ishikawa et al. A 4500 MIPS/W, 86μA Resume-Standby, 11μA Ultra-Standby Application Processor for 3G Cellular Phones. *IEICE Transactions on Electronics*, E88-C(4):528–535, Apr. 2005.

[9] H. Matsutani, M. Koibuchi, and H. Amano. Performance, Cost, and Energy Evaluation of Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network. In *Proceedings of the International Parallel and Distributed Processing Symposium*, Mar. 2007.

[10] V. Soteriou and L.-S. Peh. Exploring the Design Space of Self-Regulating Power-Aware On/Off Interconnection Networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(3):393–408, Mar. 2007.

[11] K. Usami and N. Ohkubo. A Design Approach for Fine-grained Run-Time Power Gating using Locally Extracted Sleep Signals. In *Proceedings of the International Conference on Computer Design*, Oct. 2006.