PAPER Special Section on Parallel and Distributed Computing and Networking

# A Runtime Optimization Selection Framework to Realize Energy Efficient Networks-on-Chip*

**Yuan HE**[†a)], ***Nonmember***, **Masaaki KONDO**[†], **Takashi NAKADA**[†], ***Members***, **Hiroshi SASAKI**[††], ***Nonmember***, **Shinobu MIWA**[†††], ***and*** **Hiroshi NAKAMURA**[†], ***Members***

**SUMMARY** Networks-on-Chip (or NoCs, for short) play important roles in modern and future multi-core processors as they are highly related to both performance and power consumption of the entire chip. Up to date, many optimization techniques have been developed to improve NoC's bandwidth, latency and power consumption. But a clear answer to how energy efficiency is affected with these optimization techniques is yet to be found since each of these optimization techniques comes with its own benefits and overheads while there are also too many of them. Thus, here comes the problem of when and how such optimization techniques should be applied. In order to solve this problem, we build a runtime framework to throttle these optimization techniques based on concise performance and energy models. With the help of this framework, we can successfully establish adaptive selections over multiple optimization techniques to further improve performance or energy efficiency of the network at runtime.

***key words:*** *Networks-on-Chip, performance, energy efficiency, optimization, selection*

## 1. Introduction

Energy efficiency is one of the most critical metric for modern and future computer systems. On one hand, it is highly related to the operating cost (power bill) of high performance computer systems that have large power consumption while on the other hand, it also determines the lifespan of battery-powered hand-held devices per charging. More importantly, the dark silicon phenomenon [1] simply puts energy efficiency top on the priority list for the industry.

Meanwhile, with rapidly increasing number of cores, the demand for scalable and efficient on-chip interconnections grows significantly. As a consequence, both performance and power of NoCs should be considered carefully as their size and complexity scale rapidly with larger number of cores for different forms of computer systems. This has no doubt made NoCs one of the main performance and power contributors and thus their energy efficiency an important concern.

With such a background, there are many existing studies on optimizing performance and power consumption of NoCs. Low latency routers are used to reduce transmission latency [2]–[6]. Hybrid router buffer designs attempt to improve the network throughput [7]. Advanced allocators try to increase the matching efficiency within a router [8]. Traffic compression aims to conserve network bandwidth and shorten propagation latency [9], [10]. Power gating is implemented for the routers to reduce their static power [11]. This wide variety of optimization techniques helps improve both performance and power of NoCs but they also bring several concerns. First, with too many optimization techniques, as the complexity of evaluating and implementing them in design process exponentially increases, a better approach to assess and utilize them is indispensable. Second, when optimization techniques are applied to the network, how the network behaves according to different workloads at runtime is also important. For example, from time to time, some optimization technique may not provide much benefit in power reduction and they may cause performance overhead which even worsens the energy efficiency.

Conventionally, to find out the impacts of NoC optimization techniques, we may need to evaluate them with methods such as cycle-accurate architectural simulations. However, the problem of such simulations is their time. When there are too many optimization techniques and configurations to consider, we cannot afford the time to evaluate all of them with simulations. Additionally, the network behaves differently in different phases at runtime. Therefore, an implemented optimization technique may or may not be useful at different time. To summarize, further ideas are needed to control such optimization techniques in the network at runtime.

To address the above mentioned issue, we start with building and also verifying key performance and energy models for various NoC optimization techniques. With the help of these models, we can more easily carry out design explorations to find the optimal design before production. More importantly, we propose and implement a runtime framework to adaptively control NoC optimization techniques. The objective is that this framework can help switch any implemented optimization techniques on/off (thus selecting them) at the right moment for the best performance or energy efficiency.

Manuscript received January 8, 2016.
Manuscript revised May 18, 2016.
Manuscript publicized August 24, 2016.

[†]The authors are with the Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, 113–8656 Japan.
[††]The author is with the Department of Computer Science, Columbia University, New York, USA.
[†††]The author is with the Graduate School of Information Systems, The University of Electro-Communications, Chofu-shi, 182–8585 Japan.

*This paper is an extended version of the following work: Y. He *et al.*, "Runtime Multi-Optimizations for Energy Efficient On-chip Interconnections," Proc. ICCD '15, pp.484–487, Oct. 2015.

a) E-mail: he@hal.ipc.i.u-tokyo.ac.jp

DOI: 10.1587/transinf.2016PAP0026

The main contributions of this work are summarized as follows.

- First, to the best of our knowledge, this is the first attempt which comprehensively characterizes multiple NoC optimization techniques for the purpose of performance and energy efficiency.
- Second, we have built and verified concise but powerful performance and energy models for representative NoC optimization techniques.
- Third, with the help of the above-mentioned models, we propose and implement a runtime framework to adaptively control NoC optimization techniques and we have proved its effectiveness with very comprehensive evaluations.

The rest of this paper is organized as follows. Section 2 briefly introduces related work. Section 3 presents the modeling of a baseline NoC and representative NoC optimization techniques. Furthermore, we discuss why and how a runtime framework with adaptive control on NoC optimization techniques can be used to further improve the energy efficiency in Sect. 4. In Sect. 5, we provide details on how we carry out model validations and evaluations in this paper. Section 6 then presents the evaluation results and discussions for our runtime framework. Finally, Sect. 7 concludes this paper.

## 2. Related Work

Not surprisingly, many existing NoC optimization techniques have already been studied in order to improve the performance or power of NoCs. Performance-wise, there are many existing router designs focusing on shortening the latency of a router [2]–[6], [12], [13]. There are also studies which focus on improving the bandwidth of NoCs through different approaches, such as compression [9], [10], allocators with higher matching efficiency [8], and different buffer designs allowing higher capacity [7]. Power-wise, there are many existing studies on saving the static power of routers through power management techniques such as PG [11] or proportionally supplying power to the network based on traffic demand [14]. Each one of these optimization techniques can be very useful towards their purposes but up to this point, there is no such work which tries to look at multiple optimization techniques for NoCs. Our work is the first attempt.

Modeling has also been a useful tool to assess and predict NoC's performance and power impacts rapidly. For example, there are studies [15]–[18] targeted at modeling the performance of NoCs. Of these studies, [15] used their results on design explorations while [16] applied their insights gained from traffic patterns to assist synthetic traffic generations. There are also NoC power modeling and estimation work. [19] is an attempt on deriving the energy model of NoC based multicore chips and made comparison to a traditional bus. [20] presents a high level power estimation methodology for a NoC router to enable power exploration at system level. There are also two papers from the same research group which focused on very detailed power estimation of NoC components [21], [22]. Some work targeted at both performance and power modeling at the same time. [23] presented their performance and power analyses on a parameterized RTL level design of the NoC architecture elements, while [24] evaluated the performance, power and area trade-offs between various network topologies. Simply speaking, all these studies lack the support of NoC optimization techniques so we try to fill this gap. Our work also differs from these modeling studies as we use modeling to assist the process of finding the best NoC optimization techniques at runtime.

## 3. Modeling the NoC Optimization Techniques and Their Impacts

In this section, we attempt to model the performance and energy of NoCs and we extend the model by including performance and energy impacts from 3 key NoC optimization techniques. All these models will be validated later in Sect. 6.1.

### 3.1 Performance Modeling

To understand the performance of a NoC, we first focus on the average network latency per flit as in Eq. (1). It can be divided into the zero load latency ($\overline{L_{ZeroLoad}}$), which is the latency from the source to the destination without contention, and queuing latency ($\overline{L_{Queue}}$). Zero load latency can be further divided into several parts, for network interface ($L_{NI}$), routing ($L_{Route}$) and link traversal ($L_{Link}$). It can be written as Eq. (2). $\overline{H}$ is the average number of hops a flit travels in the network.

$$\overline{L_{Net}} = \overline{L_{ZeroLoad}} + \overline{L_{Queue}} \tag{1}$$

$$\overline{L_{ZeroLoad}} = 2L_{NI} + L_{Route} \times \overline{H} + L_{Link} \times (\overline{H} + 1) \tag{2}$$

The average queuing latency per flit, $\overline{L_{Queue}}$, can be approximated with the $M/D/1$ queue model [17], [18] as in Eq. (3). $N_{Packet}$ is the number of injected packets to the network and $N_{NI}$ is the number of network interfaces that inject traffic to the network. $\overline{L_{Prop}}$ is the average propagation delay for the packet containing the modeled flit. It is determined by the number of flits in the packet so we can model it by dividing the amount of flits ($N_{Flit}$) by the amount of packets ($N_{Packet}$) traveled through the network.

$$\overline{L_{Queue}} = \frac{\frac{N_{Packet}}{N_{NI}} \times (\overline{L_{ZeroLoad}} + \overline{L_{Prop}})^2}{2 \times \left(1 - \frac{N_{Packet}}{N_{NI}} \times (\overline{L_{ZeroLoad}} + \overline{L_{Prop}})\right)} \tag{3}$$

$$\overline{L_{Prop}} = \frac{N_{Flit}}{N_{Packet}} \tag{4}$$

### 3.2 Energy Modeling

To model the energy, we start with the average energy consumption per flit as written in Eq. (5). It is further modeled

as static energy in Eq. (6) and dynamic energy in Eq. (7). In Eq. (7), the dynamic router and link energy per flit is modeled with the number of accesses to routers and links ($N_{RA}$ and $N_{LA}$). Since the static power ($P_{S_{Net}}$) and the clock power ($P_{D_{Clk}}$) are always consumed, they are multiplied with the runtime ($T(\overline{L_{Net}})$), as in both Eqs. (6) and (7). It is obvious that $T(\overline{L_{Net}})$ is subject to the network performance (in our case, $\overline{L_{Net}}$) and when we validate our models with simulations, we simply take the application runtime for $T(\overline{L_{Net}})$. However, when we evaluate our runtime framework, we take the runtime of the shutter period for $T(\overline{L_{Net}})$. The power and energy parameters of these models ($P_{S_{Net}}$, $P_{D_{Clk}}$, $E_{D_{Router}}$ and $E_{D_{Link}}$) are simply obtained from the Orion 2 simulator [22].

$$\overline{E_{Net}} = \overline{E_{D_{Net}}} + \overline{E_{S_{Net}}} \tag{5}$$

$$\overline{E_{S_{Net}}} = \frac{P_{S_{Net}} \times T(\overline{L_{Net}})}{N_{Flit}} \tag{6}$$

$$\overline{E_{D_{Net}}} = \frac{E_{D_{Router}} \times N_{RA} + E_{D_{Link}} \times N_{LA} + P_{D_{Clk}} \times T(\overline{L_{Net}})}{N_{Flit}} \tag{7}$$

### 3.3 The NoC Optimization Techniques and Their Impacts

In this subsection, we introduce a few important optimization techniques on NoCs. They are power gating (PG), prediction router (PR) and traffic compression (TC). These optimization techniques are chosen since they are targeting at different problems and each of them is a representative of its kind. For simplicity, we summarize their performance and energy models in Table 1. Subject to the space limitation, we only consider 3 optimization techniques in this paper. However, it is not difficult to include more optimization techniques by adding new models.

- *Power Gating*: PG is a representative static power reduction technique, which helps cutting off the power supply to idle circuit blocks by turning off (or on) the power switches which are inserted between the GND/VDD lines and blocks [25]. PG has been applied to different types of circuit blocks with various granularities. Applying it to routers in NoCs can help the routers to save their static power when they are not actively used [11]. As stated in Eq. (8), PG affects the network latency since gated router needs a few cycles ($L_{Wakeup}$) to wake up. Another overhead of PG comes as dynamic energy as in Eq. (9), the power switch consumes energy ($E_{D_{PS}}$ is the energy to use the power switch, coefficient "2" reflects the process of controlling the power switch twice to turn the circuit off and on while $N_{PG}$ is the number of power gating taken) to control the gated circuit block. Meanwhile, the static power of a router is reduced with PG and it is proportional to the amount of time when the router is switched on ($T_{RouterOn_i}$) as in Eq. (10). $N_{Router}$ is the number of routers in a network.
- *Prediction Router*: The latency of a router determines the transmission delay of a packet. PR [4] is an advanced router design whose speculative switch traversal is enabled with predictions of the output ports before a packet actually comes to a router. If a predictive routing succeeds, the latency of the router pipeline is hidden since they are already carried out with a predicted route computation result. With prediction router, the router latency is the same as the switch traversal delay ($L_{ST}$) when prediction succeeds, as in Eq. (11). But prediction units which are implemented in the input circuitry of routers also consume dynamic and static energy ($E_{D_{PR}}$ and $E_{S_{PR}}$), as modeled in both Eqs. (12) and (13). $R_{Pred}$ is the prediction rate.
- *Traffic Compression*: TC is a popular optimization technique which affects both performance and power. It has been applied in many fields to conserve on-chip/off-chip bandwidth, to enlarge cache/memory capacity, to reduce communication latency or to cut dynamic energy. On NoCs, traffic compression is used to conserve bandwidth and to reduce packet latency [9], [10]. Additionally, it may help saving power since a successfully compressed packet may have its size shrunk, which can help saving dynamic power while the shrunk packet traverses the network. In Eq. (14), TC's performance impact is modeled as it may change the amount of flits which is in turn affecting the propagation delay for network packets. For dynamic energy, it has two impacts as in Eq. (15). First, smaller packets means less flits to traverse the network. This will help reduce the amount of dynamic energy. Second, compressor and de-compressor circuits consumes dynamic energy ($E_{D_{CU}}$). These circuits also consume static power ($E_{S_{CU}}$) as modeled in Eq. (16). $R_{Comp}$ is the compression rate while $N_{TC}$ is the number of compressions.

**Table 1** Performance and energy models of NoC optimization techniques.

| NoC optimization techniques | Performance impacts | Dynamic energy impacts | Static energy impacts |
|---|---|---|---|
| **PG** | $L'_{Route} = L_{Route} + L_{Wakeup}$ (8) | $E'_{D_{Net}} = E_{D_{Net}} + E_{D_{PS}} \times 2N_{PG}$ (9) | $E'_{S_{Net}} = P_{S_{Net}} \times \frac{\sum_i^{N_{Router}} T_{RouterOn_i}}{N_{Router}}$ (10) |
| **PR** | $L'_{Route} = L_{ST} \times R_{Pred} + L_{Route} \times (1 - R_{Pred})$ (11) | $E'_{D_{Net}} = E_{D_{Net}} + E_{D_{PR}} \times N_{PR}$ (12) | $E'_{S_{Net}} = E_{S_{Net}} + E_{S_{PR}} \times N_{Router}$ (13) |
| **TC** | $N'_{Flit} = \frac{N_{Flit}}{R_{Comp}}$ (14) | $E'_{D_{Net}} = \frac{E_{D_{Net}}}{R_{Comp}} + E_{D_{CU}} \times N_{TC}$ (15) | $E'_{S_{Net}} = E_{S_{Net}} + E_{S_{CU}} \times N_{NI}$ (16) |

**Table 2** Possible consequences after having multiple NoC optimization techniques.

| Optimization techniques | Impacts |
|---|---|
| **PG&PR** | PR brings more activities within a router so opportunities for power gating may be fewer. This means, PR may be negative for PG in energy saving. |
| **PG&TC** | TC reduces activities in the network which may result in more idle periods to switch on PG. So TC may be positive for PG in energy saving. |
| **PR&TC** | TC reduces activities on the crossbar switches which can help more predictions to be taken. So TC seems positive for PR in performance. |
| **PG&PR&TC** | TC reduces activities in the network so it may help PG and PR in improving the energy and performance, respectively. But PR may still degrade PG in energy saving. |

### 3.4 Applying Multiple NoC Optimization Techniques

In practice, an optimization technique may help in some degree but it may not be sufficient to satisfy either the performance or power requirement all by itself; this leads us to consider multiple optimization techniques. But using these optimization techniques together is not simple since the impact of optimization techniques varies for different workloads and they may also interact with each other. To help clarify such interactions, we have their qualitative analyses presented in Table 2. TC may have positive impacts on PG while TC may also help PR. On the other hand, PR seems to be negative for PG. Therefore, for the purpose of using multiple optimization techniques, TC seems to be a preferable choice to combine with others while PG and PR are better not used together. We will further discuss these in Sect. 6.

## 4. A Runtime Selection Framework through Adaptive Control on Multiple NoC Optimization Techniques

In this section, we introduce our proposal of adaptively controlling NoC optimization techniques at runtime. Though picking up multiple optimization techniques may help improve both performance and power, it is not good enough since applications have phases among which they may behave very differently. Statically implementing one or several optimization techniques can be inefficient for an application. This leads us to think about a runtime framework which can adaptively throttle these optimization techniques from time to time.

To establish adaptive control for these optimization techniques with our runtime framework, we need to rely on the models introduced in Sect. 3 to determine if an optimization technique should be switched on or off. Our philosophy is that optimization techniques are better switched on for epochs in which they can productively help performance or energy efficiency. To carry out the procedures in Fig. 1, an application will start with a shutter period where all available optimization techniques are turned on to collect the values of related performance counters. Using collected

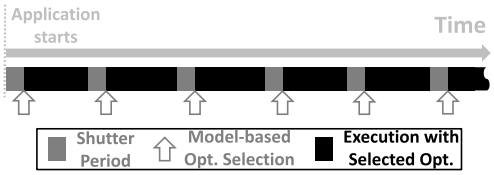



**Fig. 1** Procedures for selecting optimization techniques adaptively in the runtime framework.

performance counter values, these techniques will be turned on or off for the following epoch according to their impacts predicted through the models presented in Sect. 3 and Table 1. For example, if the purpose is to minimize EDP and PG&PR gives the smallest EDP value according to the performance counters and the models in a shutter period, then PG&PR will be enabled in the epoch after this shutter. This process will be repeated until the application ends.

Performance counter values we need to collect are *instructions executed*, *number of flits injected*, *number of power gating* (PG-specific), *router power-on cycles* (PG-specific), *number of predictions* (PR-specific), *number of miss-predictions* (PR-specific), *number of compressions* (TC-specific) and *number of flits before/after compressions* (TC-specific). They can be calculated with very simple counters and they are supposed to be collected and stored locally at each tile.

This proposal can be implemented in hardware and a centralized control is needed since these optimization techniques are implemented on each network component. We assume that the centralized control unit is physically placed at the center of the network and connected to the network through one of the routers. This centralized control will need to communicate with the network components to retrieve performance statistics and send control messages. In the worst case, the communication overhead is equal to the latency traversing from center of the network to the corners. This communication overhead can hardly degrade NoC performance since it is carried out only twice in one period (having one shutter and one epoch) and its latency is negligible compared to the length of the shutter plus the epoch.

## 5. Methodology for Model Validations and Evaluations

The evaluation of this work has two parts. First, we validate and analyze our models for the network and the optimization techniques with full system simulations. Second, we also test the runtime framework with execution traces of various workloads.

We carry out full system simulations on both performance and energy by using GEMS/Simics [26], [27] extended with the network model from GARNET [28] and the power model from Orion 2.0 [22]. To evaluate performance, we have modified the source code of GARNET to provide cycle-accurate timing models of selected optimization techniques and their combinations. In the energy evaluation, we slightly modified Orion to include power models for power gating, prediction router and traffic compression. The eval-

**Table 3** Simulation parameters.

| Simulation Parameters | Value |
|---|---|
| Number of cores | 16 |
| Topology | $4 \times 4$ mesh |
| Processor | 4 GHz, In-order |
| L1 I/D cache | 32 KB per Processor, 4-way set associative |
| L2 cache | 256 KB per Bank, 16-way set associative |
| Cache line | 64 Bytes |
| Main memory | 4 GB |
| Main memory latency | 160 cycles |
| Coherence protocol | MOESI, Directory |
| Link | 128-bit, 1 cycle traversal |
| Packet | 128-bit control, 640-bit data |
| Router | 1 GHz, Virtual channel router |
| Virtual channel | 2 per Virtual network |
| Virtual network | 3 per Physical link |
| Routing algorithm | X-Y routing |

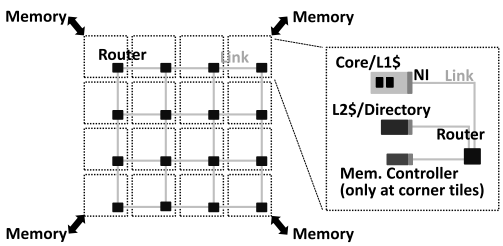



**Fig. 2** Evaluated platform.

uation conditions are summarized in Table 3.

In all full system simulations, we assume a 16-tile mesh network with 128-bit links. Each tile has an in-order processor core and a bank of L2 cache/directory. Each corner tile also has a memory controller so in total there are four of them. These components are connected to a router individually. The schematic view of this simulated platform and what a tile is composed of are illustrated in Fig. 2. The entire network is set to have three virtual networks to support the MOESI directory coherence protocol which has three classes of packets. Each router has a maximum of six ports and each port has two virtual channels while each virtual channel has four 128-bit buffers. Our simulations are carried out with application traffic. The application traffic is based on workloads from NPB 3.3 [29] and SPLASH-2 [30] benchmark suites.

For our runtime framework with adaptive selections among multiple optimization techniques, we collect periodic execution traces with necessary performance counters which are listed in Sect. 4 to help our models to determine which combinations of optimization techniques are the best for a coming epoch. Collecting such periodic execution traces and evaluating our proposal with them have two advantages. First, this method is as accurate as actually implementing our proposal in a simulator but it requires much less effort. Second, we can easily obtain the oracle case (results following ideal selections on the optimization techniques) with such execution trace based evaluation and this helps us show how close our proposal to the ideal case. To broaden

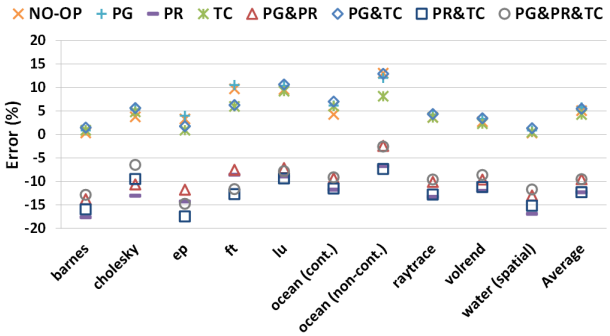



**Fig. 3** Model validation for latency per flit.

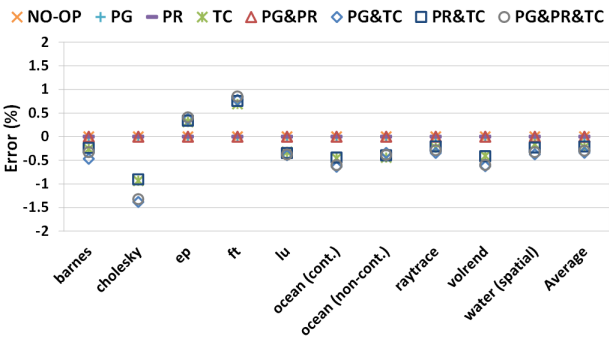



**Fig. 4** Model validation for energy per flit.

our discussions, we also set up two further evaluations with different epoch sizes and varying ratio between dynamic and static power (we do this by changing the static power). The first helps us understand the effectiveness of our framework with respect to epoch sizes while the second helps us carry out discussions on our framework with different application areas or process technologies.

In the evaluation of our framework, the size of measurement period and epoch are determined by the amount of instructions. The metrics we use to compare our results are latency per flit, energy per flit and energy delay product (EDP) per flit. The first metric is for network performance, the second is for energy efficiency while the third is a metric for both performance and energy efficiency. The reason for picking the third metric is, when an application is applied with a NoC optimization technique, both its number of instructions and its amount of network traffic change with respect to its execution time since a thread waiting for other threads in a multi-threaded application is still executing instruction and making synchronizations. Our work only focuses on the network but it can be easily extended to include cores and caches since there are numbers of existing studies for performance and energy modeling of both cores and cache. Our models can be easily extended to include these components.

## 6. Results

### 6.1 Model Validations

Figures 3 and 4 present the validation results for our models

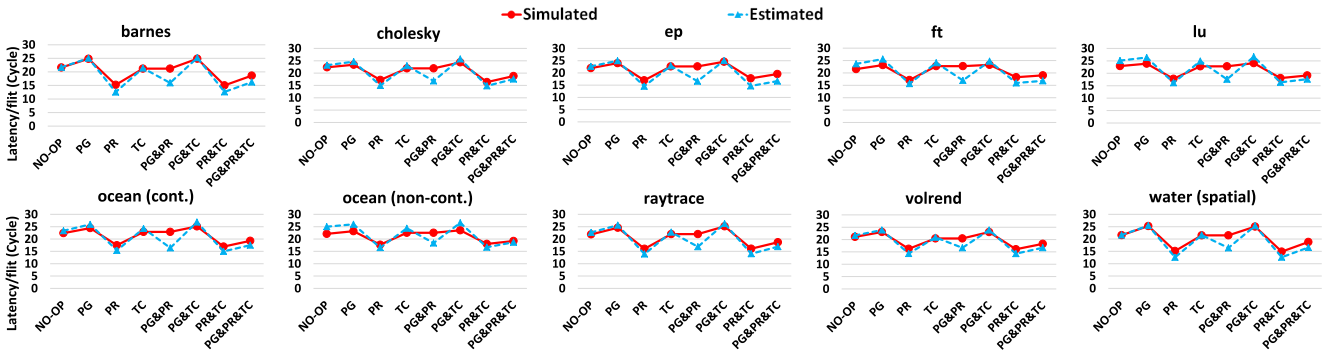



**Fig. 5** Simulated and estimated latency per flit with different NoC optimization techniques.

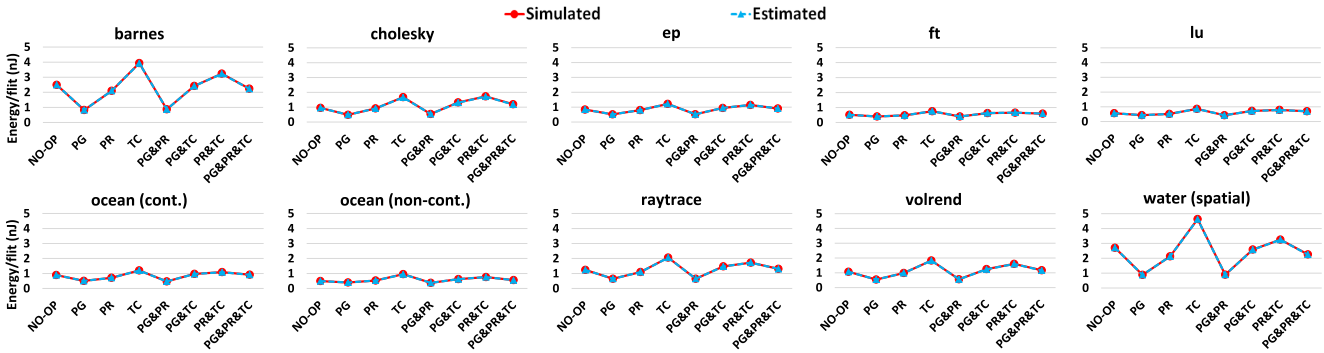



**Fig. 6** Simulated and estimated energy consumption per flit with different NoC optimization techniques.

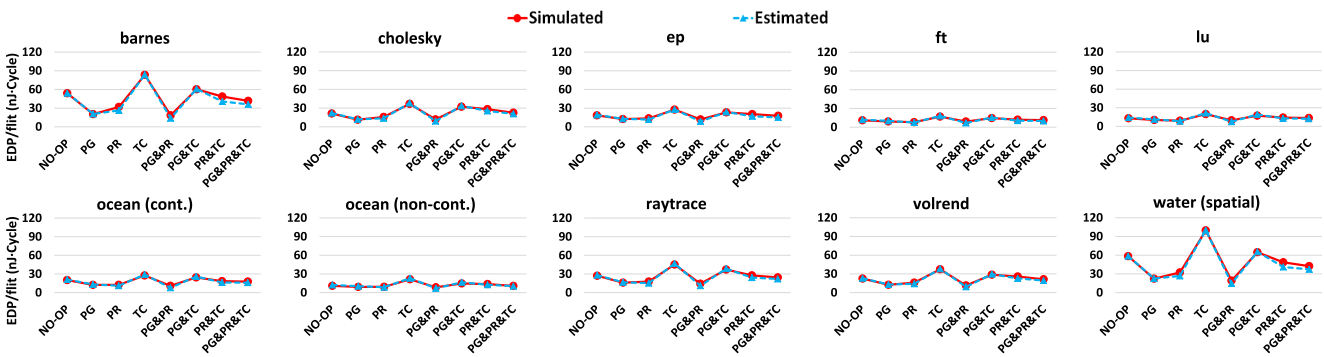



**Fig. 7** Simulated and estimated EDP per flit with different NoC optimization techniques.

(on latency and energy per flit, respectively) against simulations (the baseline). They are drawn with results obtained from both simulations and models we created. It can be seen that both our latency and energy models are accurate. The error for the latency model is mostly within 15% and this error comes from the approximation we have on queuing latency. The energy model is more accurate with an error of less than 2% since the way we model energy is the same as the power simulator we use (Orion 2). Both our model and Orion estimates the energy consumption by counting the events taken at network components.

Similarly, Figs. 5, 6 and 7 are also drawn with results collected from simulations (red lines) and our models (blue lines). We can see that only the performance model has some mismatching between the simulated and estimated results. For example, latency when having PG&PR is a bit under-estimated for all workloads. But such mismatching is not a concern since the trend with different optimization techniques is not heavily affected. This means that our models are good enough for design explorations.

In Fig. 5, we can see that PR results in the best network performance for nearly all the workloads. In a few cases, PG&PR gets close to it when being estimated, but it is because of the error of our model. This problem is not as serious as it looks since the estimated latency under PG&PR is still longer than PR. With this figure, we can conclude that both PR is very positive in network performance and it is the best optimization technique when network perfor-

mance is favored. Latency does not look good for TC since latency overhead from compression is mandatory regardless of the compression rate and in some cases this compression latency is not well compensated with network latency cut by the compression under low compression rate.

In Fig. 6, we can see that the results for different workloads diverge. For workloads *barnes*, *cholesky*, *raytrace*, *volrend* and *water (spatial)*, PG and PG&PR simply bring the least energy consumption. On the other hand, these optimization techniques do not bring much difference to the remaining five workloads. This means, for these workloads, even PG won't help much in reducing their energy consumption. One thing to note is that our energy model is more accurate.

In Fig. 7, some more interesting results can be observed. First, like Fig. 6, *ep*, *ft*, *lu*, *ocean (cont.)* and *ocean (non-cont.)* are not very sensitive to these optimization techniques. For the remaining workloads, PG, PR and PG&PR seem to be very close while PG&PR is sometimes the best, but with a very small margin. This means, to improve EDP, one should either improve the network performance or reduce the power consumption but even if having both PG and PR, the result may not dramatically better since the room to improve is small with one optimization technique already implemented. Another common observation in Figs. 6 and 7 is that TC is better left out when designing a network if energy or EDP is the target.

As a summary, based on these models, we can very easily predict the performance and the energy impacts of different NoC optimization techniques without incurring any time-consuming simulation. This is very important for the runtime framework we build.

### 6.2 The Runtime Framework

Figures 8, 9 and 10 present the per-flit latency, energy and EDP when epoch size varies from 50K to 12.8M instructions with our runtime framework. In these three figures, we normalize all results to the case of 50K instructions as the epoch. For most of the workloads, variations start to surface after the epoch is as large as 400K instructions. This simply tells us that 200K instructions for an epoch is good enough so we use this as the epoch size in later evaluations.

Figure 11 depicts the latency per flit when single and multiple NoC optimization techniques are applied and they are compared to our runtime framework and the oracle case. It can be seen that our proposal performs as good as PR&TC while oracle is marginally better. So the effectiveness of our runtime framework is proved since it selects the best combinations of optimization techniques for each epoch and most of these choices must be PR&TC.

Figure 12 shows energy per flit under different combinations of optimization techniques and they are compared to our proposal and oracle. It can be found that our proposal is not far from oracle while PG and PG&PR are the best for energy per flit.

Figure 13 presents the EDP results with static optimization techniques and these results are also compared to our proposal and oracle. It can be noted that our proposal is again not far from oracle while PG&PR is the best performing combinations of optimization techniques in terms

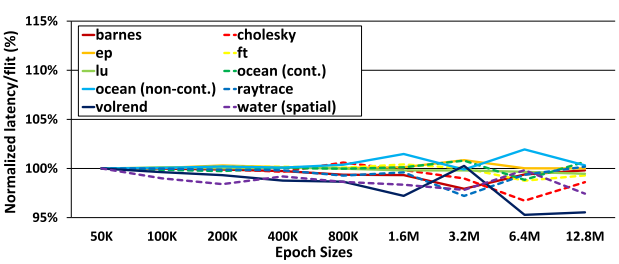



**Fig. 8** Normalized latency per flit under the runtime framework with different epoch sizes.

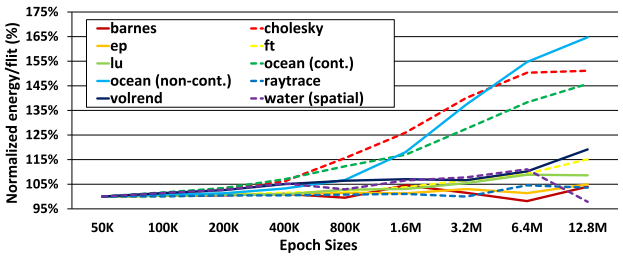



**Fig. 9** Normalized energy per flit under the runtime framework with different epoch sizes.

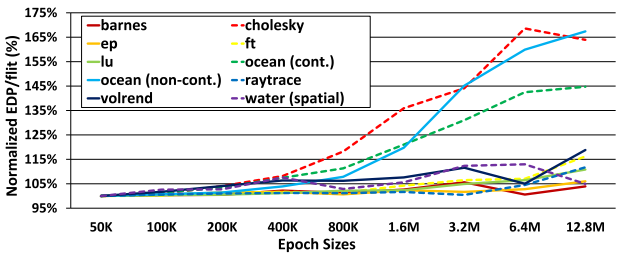



**Fig. 10** Normalized EDP per flit under the runtime framework with different epoch sizes.

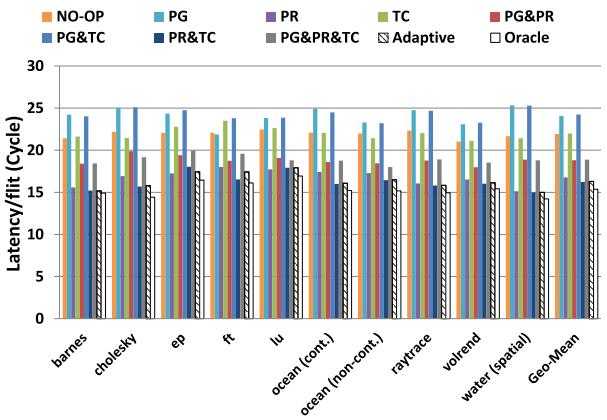



**Fig. 11** Latency per flit under different optimization techniques and our framework.

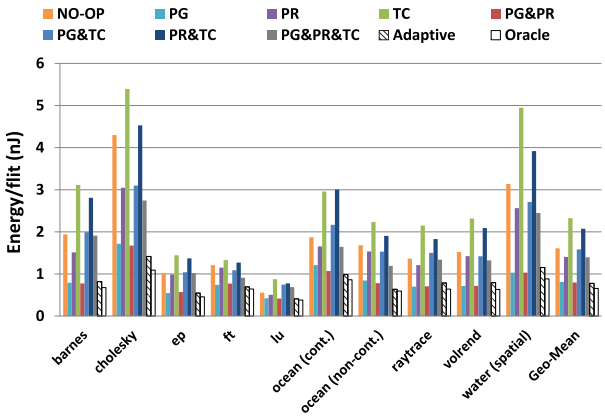



**Fig. 12** Energy per flit under different optimization techniques and our framework.

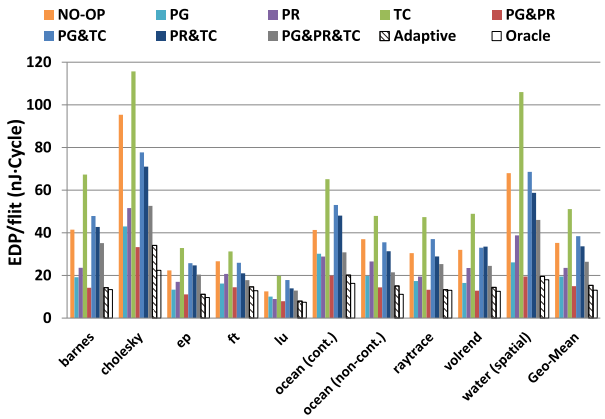



**Fig. 13** EDP per flit under different optimization techniques and our framework.

of EDP per flit.

In Sect. 3.4, we argued that PR may have negative effect on PG for the purpose of energy saving. In general, our results do not support this (see *Geo-Mean* in Fig. 12) but for workloads *ep* and *ft* in Fig. 12, PR can be confirmed for having negative impacts on energy. In Fig. 10, PG&PR is always better than PG since EDP per flit is a metric which favors performance more than energy.

Another interesting finding is that TC seems to be a positive optimization when used with others in our qualitative analysis in either performance or energy. But the result tells a different story. For energy, when either PG or PR is used with TC, their gain is too small to compensate TC's energy overhead, which is quite large. Thus, within all these optimization techniques and their combinations, it can be seen that TC and combinations with TC perform not well for energy delay product. However, TC is more performance beneficial as can be seen from Fig. 11. This supports our qualitative discussions at the end of Sect. 3.4.

As a summary, if the target is performance, PR&TC is preferred while PG&PR is more suitable for improving the energy efficiency. In either cases, our runtime framework performs very well and it produces outcomes very similar to oracle.

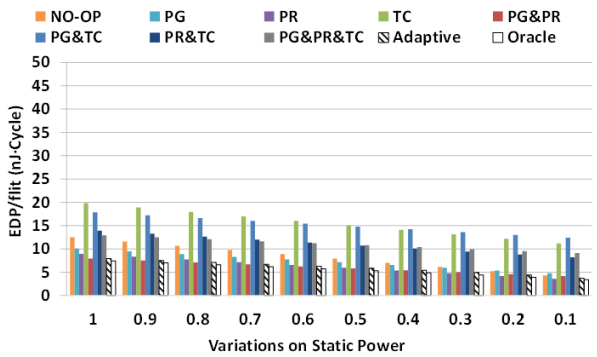



**Fig. 14** EDP per flit under different amount of static power for "lu".

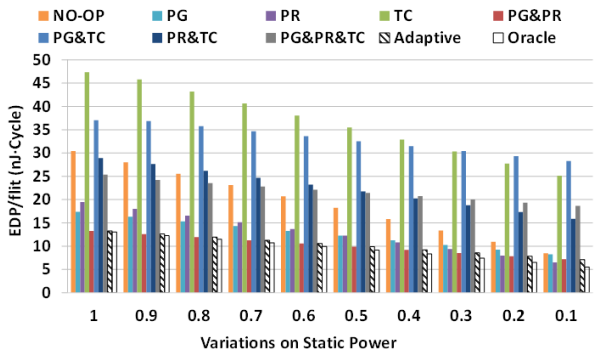



**Fig. 15** EDP per flit under different amount of static power for "raytrace".

Figures 14 and 15 present the evaluation results on how our proposal performs when the ratio between dynamic and static power is different for two workloads. We change this ratio by varying the amount of static power. In this evaluation, we vary the static power from 10% of its original value to 90%. The results show us that as static power gets smaller, the advantage of PG are degrading.

In Fig. 15, we can see PR surpasses PG when the static power is set to 0.5 of its original size. With 10% of the original static power, PR becomes the best static optimization technique while using PG and PR together no longer produces the best result for workload *raytrace*. For workload *lu*, this observation can be seen when the static power is started to be set to 0.4 of its original value.

As for TC, when static power gets smaller, it is also getting better in terms of EDP since dynamic power is more dominant and TC improves dynamic power through reducing the amount of traffic. However, it is still not efficient for energy purpose since its energy overhead is too large with current platform and parameters.

Another observation is, regardless of the amount of static power, our runtime framework still works pretty well and it is similar to oracle. This again proves its effectiveness on improving both performance and energy efficiency.

## 7. Conclusions

In this work, we have presented a study on multiple NoC op-

timization techniques. Through modeling, we have shown that how we are able to capture the impacts of selected NoC optimization techniques on both performance and energy in a much simpler way. Then, with the help of modeling, we try to establish adaptive control on multiple NoC optimization techniques at runtime. Our work is the first to have in-depth analyses on the usage of multiple NoC optimization techniques and our framework is the first attempt to throttle these techniques at runtime for better performance and energy efficiency. Through evaluations, we have proved that the models we created are very accurate and adaptively throttling the NoC optimization techniques with our runtime framework is very promising for both network performance and energy efficiency.

## Acknowledgments

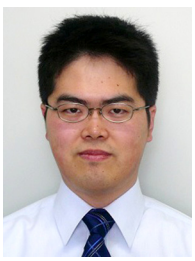
This work was, in part, supported by the Japan Science and Technology Agency CREST program, "A Power Management Framework for Post Peta-Scale Supercomputers".

### References

[1] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," Proc. 38th ISCA, pp.365–376, 2011.
[2] L.-S. Peh and W.J. Dally, "A delay model and speculative architecture for pipelined routers," Proc. 7th HPCA, pp.255–266, 2001.
[3] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," Proc. 31st ISCA, pp.188–197, June 2004.
[4] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, "Prediction router: A low-latency on-chip router architecture with multiple predictors," IEEE Trans. Comput., vol.60, no.6, pp.783–799, 2011.
[5] Y. He, H. Sasaki, S. Miwa, and H. Nakamura, "Predict-more router: A low latency NoC router with more route predictions," Proc. 27th IPDPSW, pp.842–850, May 2013.
[6] Y. He, H. Sasaki, S. Miwa, and H. Hiroshi, "McRouter: Multicast within a router for high performance Network-on-Chips," Proc. 22nd PACT, pp.319–330, 2013.
[7] H. Jang, B.S. An, N. Kulkarni, K.H. Yum, and E.J. Kim, "A hybrid buffer design with STT-MRAM for on-chip interconnects," Proc. 6th NoCS, pp.193–200, 2012.
[8] G. Michelogiannakis, N. Jiang, D. Becker, and W.J. Dally, "Packet chaining: Efficient single-cycle allocation for on-chip networks," Proc. 44th MICRO, pp.83–94, 2011.
[9] R. Das, A.K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M.S. Yousif, and C.R. Das, "Performance and power optimization through data compression in Network-on-Chip architectures," Proc. 14th HPCA, pp.215–225, Feb. 2008.
[10] Y. Jin, K.H. Yum, and E.J. Kim, "Adaptive data compression for high-performance low-power on-chip networks," Proc. 41st MICRO, pp.354–363, 2008.
[11] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Ultra fine-grained run-time power gating of on-chip routers for CMPs," Proc. 4th NoCS, pp.61–68, 2010.
[12] A. Kumar, L.-S. Peh, P. Kundu, and N.K. Jha, "Express virtual channels: Towards the ideal interconnection fabric," Proc. 34th ISCA, pp.150–161, June 2007.
[13] M. Hayenga and M. Lipasti, "The NoX router," Proc. 44th MICRO, pp.36–46, Dec. 2011.
[14] R. Das, S. Narayanasamy, S.K. Satpathy, and R.G. Dreslinski, "Catnap: Energy proportional multiple Network-on-Chip," Proc. 40th ISCA, pp.320–331, 2013.
[15] T. Nakada, S. Miwa, K. Yano, and H. Nakamura, "Performance modeling for designing NoC-based multiprocessors," Proc. RSP'13, pp.30–36, Oct 2013.
[16] V. Soteriou, H. Wang, and L. Peh, "A statistical traffic model for on-chip interconnection networks," Proc. MASCOTS'06, pp.104–116, Sept. 2006.
[17] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network delays and link capacities in application-specific wormhole NoCs," VLSI Design, vol.2007, Article ID 90941, 2007.
[18] Y. Ben-Itzhak, I. Cidon, and A. Kolodny, "Delay analysis of wormhole based heterogeneous NoC," Proc. 5th NoCS, pp.161–168, May 2011.
[19] P.T. Wolkotte, G.J.M. Smit, N. Kavaldjiev, J.E. Becker, and J. Becker, "Energy model of Networks-on-Chip and a bus," Proc. SoC'05, pp.82–85, Nov, 2005.
[20] S.E. Lee and N. Bagherzadeh, "A high level power model for Network-on-Chip (NoC) router," Comput. Electr. Eng., vol.35, no.6, pp.837–845, Nov. 2009.
[21] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," Proc. 35th MICRO, pp.294–305, 2002.
[22] A.B. 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," Proc. DATE'09, pp.423–428, April 2009.
[23] N. Banerjee, P. Vellanki, and K.S. Chatha, "A power and performance model for Network-on-Chip architectures," Proc. DATE'04, pp.1250–1255, Feb. 2004.
[24] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for Network-on-Chip interconnect architectures," IEEE Trans. Comput., vol.54, no.8, pp.1025–1040, Aug. 2005.
[25] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," Proc. ISLPED'04, pp.32–37, 2004.
[26] M.M.K. Martin, D.J. Sorin, B.M. Beckmann, M.R. Marty, M. Xu, A.R. Alameldeen, K.E. Moore, M.D. Hill, and D.A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," SIGARCH CAN, vol.33, no.4, pp.92–99, Nov. 2005.
[27] P.P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," Computer, vol.35, no.2, pp.50–58, 2002.
[28] N. Agarwal, T. Krishna, L.-S. Peh, and N.K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," Proc. ISPASS'09, pp.33–42, 2009.
[29] "NAS parallel benchmarks 3.3," http://www.nas.nasa.gov/Resources/Software/npb.html
[30] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," Proc. 22nd ISCA, pp.24–36, 1995.

**Yuan He** is a postdoctoral researcher at the Graduate School of Information Science and Technology, the University of Tokyo. He received his Ph.D. from the University of Tokyo in 2014, M.E. (Hons) and B.Sc. from the University of Auckland in 2009 and 2005, respectively. His research interests include computer architecture, energy efficient computing at all scales and post-Moore performance scaling. He is a member of the ACM and IEEE.

**Masaaki Kondo** received the B.E. degree in College of Information Sciences and the M.E. degree in Doctoral Program in Engineering from University of Tsukuba in 1998 and 2000 respectively, and the Ph.D. degree in Graduate School of Engineering from the University of Tokyo in 2003. He is currently an associate professor in the Graduate School of Information Science and Technology at the University of Tokyo. His research interests include computer architectures, high-performance computing, and dependable computing. He is a member of the IEEE, the ACM, and the IPSJ.



**Takashi Nakada** received his M.E. and Ph.D. degrees from Toyohashi University of Technology in 2004 and 2007 respectively. He has been an Assistant Professor at the University of Tokyo since 2012. His research interests includes Normally-Off Computing, processor architecture and related simulation technologies. He is a member of IEEE, ACM and IEICE.

**Hiroshi Sasaki** received the B.E., M.E., and Ph.D. degrees from The University of Tokyo in 2003, 2005, and 2008, respectively. Currently, he is an associate research scientist at Columbia University. His work focuses on improving energy-efficiency of emerging power-constrained heterogeneous and/or many-core processors through improvements to the processor architectures, system softwares and memory systems. He is a member of ACM and IEEE.

**Shinobu Miwa** was born in 1977 and received the Doctor of Informatics degree from Kyoto University in 2007. He is now an associate professor at the University of Electro-Communications. His research interests are computer architecture, high performance computing and embedded systems. He received the best paper award in 2010 Embedded System Symposium. He is a member of ACM, IEEE, IEICE and IPSJ.

**Hiroshi Nakamura** is a Professor in the Department of Information Physics and Computing at The University of Tokyo. He is also the director of Information Technology Center at The University of Tokyo. He received the Ph.D. degree in Electrical Engineering from The University of Tokyo in 1990. His research interests include power-efficient computer architecture and VLSI design for high-performance and embedded systems. He is a senior member of IEEE and ACM.