Importance of Single-Core Performance in the Multicore Era

Toshinori Sato **Hideki Mori** **Rikiya Yano** **Takanori Hayashida**
Department of Electronics Engineering and Computer Science
Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, Japan

toshinori.sato@computer.org

Abstract

This paper first investigates what the best multicore configuration will be in the future, when the number of usable transistors further increases. Comparing five multicore models: single-core, many-core, heterogeneous multicore, scalable homogeneous multicore, and dynamically configurable multicore, surprisingly unveils that single-core performance is a key to improve multicore performance. Based on the findings, this paper secondly proposes a technique to improve single-core performance. It is based on Intel's Turbo Boost technology. From the detailed simulations, it is found that the technique achieves single-core performance improvement.

Keywords: Multicore, Amdahl's law, Pollack's rule, Turbo Boost technology.

1 Introduction

Multicore processors have already been popular to improve the total performance (Howard, et al. 2010). Considering the power and temperature constraints, they might be the sole practical solution. A lot of studies to determine the best multicore configuration is conducted (Annavaram, et al. 2005, Balakrishnan, et al. 2005, Ekman and Stenstrom 2003, Hill and Marty 2008, Kumar, et al. 2005 and Morad, et al. 2006) and it is believed that the heterogeneous multicore is the best in power and performance trade-off. However, it is not clear whether this answer is still correct in the future. As Amdahl's law states, performance of parallel computing is limited by that of its serial computing portion inside. Hence, to utilize the increasing number of transistors for increasing the number of cores on a chip might not be the best choice.

This paper has two contributions. The one is that it investigates which configuration is the best multicore processor and unveils that single-core processor performance should be still improved. The other is that it proposes a technique that improves single-core performance, which we name Cool Turbo Boost technique.

The rest of the paper is organized as follows. The next section summarizes the related works. Section 3 investigates the best multicore configuration. Section 4 proposes Cool Turbo Boost technique. Section 5 concludes.

2 Related Works

There are a lot of studies investigating configurations of multicore processors (Annavaram, et al. 2005, Balakrishnan, et al. 2005, Ekman and Stenstrom 2003, Hill and Marty 2008, Kumar, et al. 2005 and Morad, et al. 2006). Most of them assume the number of usable transistors is fixed and then search the best processor configuration. Early studies mostly conclude that integrating a lot of simple cores is better in the power-performance trade-off than integrating a single complex core (Ekman and Stenstrom 2003). Later, heterogeneity and dynamic configurability are also considered. They might be keys to overcome Amdahl's law (Hill and Marty 2008).

On the top of the above studies, this paper further investigates what the best multicore configuration is. We also consider the advance in semiconductor technologies. We guess the best choice is different when the number of transistors increases.

3 Searching for Best Multicore

As the number of transistors on a chip increases, the flexibility to determine a processor configuration also increases. The current trend is to use them to integrate multiple cores on a chip and we have almost 50 cores (Howard, et al. 2010). With such a large flexibility, we are confused what the best configuration is. How many cores should be integrated on a chip? Should each core have simple in-order pipelines or complex out-of-order ones? Should all cores are the same? These questions have to be answered.

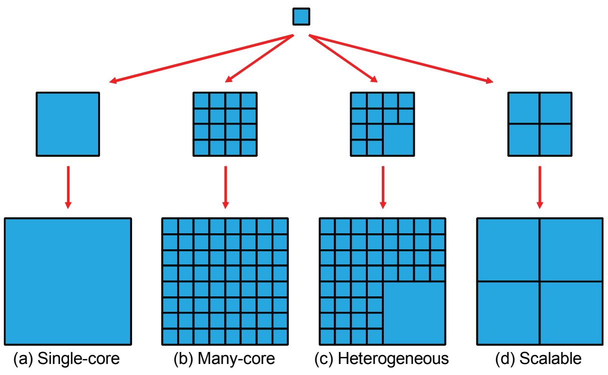

Figure 1: Variations of Multicore Processors

Figure 1 shows some variations of multicore processors. When chip integration is advanced, there are two choices. One is to increase the size of a core, and the other is to increase the number of cores. Figure 1a explains the former choice. All transistors on a chip are utilized by a single core. Figure 1b explains the latter choice. The core microarchitecture is fixed and multiple copies of the core are integrated on the chip. Figures 1c and 1d consider hybrids of the two choices. In Figure 1c, only one core becomes large and the other cores remain small, and hence the multicore is heterogeneous. In Figure 1d, all cores become large and thus the multicore is homogeneous.

The best configuration must be different according to applications executed on the processor. Hence, we analyse theoretically and use simple model to compare these variations in the following subsections.

3.1 Single-core vs. Many-core

First, a single-core (Figure 1a) and a many-core (Figure 1b) processors are compared. As the single-core processor becomes larger and larger, its area-performance ratio meets a diminishing return as explained by Pollack's rule (Borkar 2007). It says processor performance is proportional to the square-root of its area. Performance improvement rate $I_{Pollack}$ is expressed as:

$$I_{Pollack} = \sqrt{N}$$

where N explains that the processor is N times larger in area than the baseline one. On the other hand, multiprocessor performance is dominated by Amdahl's law. It says if a portion p of a program is executed in parallel by N baseline processors, the speedup I_{Amdahl} is:

$$I_{Amdahl} = \frac{1}{\frac{p}{N} + (1-p)}$$

Hence, p is an important factor that determines how efficient in performance the many-core processor is. Note that we use N both for area of the single-core processor and for the number of cores in the many-core processor. It is not confusing because the area of N cores equals that of N-times larger core. N is interchangingly used in this paper as the size of core, the number of cores, and the chip area.

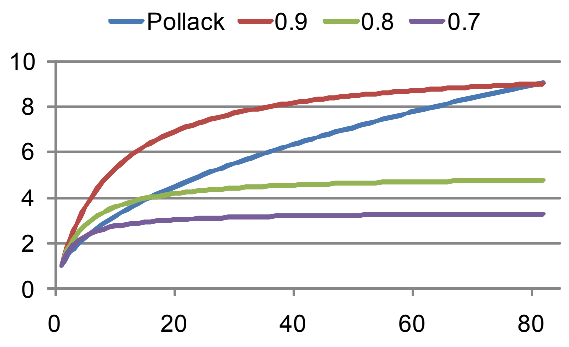

Figure 2: Pollack's Rule vs. Amdahl's Law

Figure 2 compares the single-core processors and the many-core processors. The horizontal axis indicates N and the vertical one indicates performance improvement rate. There are four lines. The blue line labelled with Pollack presents the performance improvement rate of the singe-core, $I_{Pollack}$. The other three lines present the rate of the many-core, I_{Amdahl}. The labels indicate the parallelized portion, p.

When p is as large as 0.9, the many-core processor is always better in performance until N reaches 81. This matches with the investigation in (Ekman and Stenstrom 2003): multiple simple cores are better in power-performance ratio than a big core. Please note that power consumption is proportional to the total area, N, in the model of Pollack's rule. Unfortunately, only large scale scientific computing enjoys such a large p. The conventional computing such as desktop and mobile cannot be easily parallelized. p is small (Wang, et al. 2009). As p becomes smaller and smaller, many-core performance is seriously limited. When p equals 0.7, the 8-core processor is poorer in performance than the 8-times larger single-core processor. We already have commercial 8-core processors such as Intel's Xeon 7500 series and AMD's FX series. Now is the time when single-core processors would be more beneficial for desktop and mobile applications than many-core processors, if single-core processor performance were ideally scalable to Pollack's rule.

3.2 Single-core vs. Heterogeneous Multicore

Next, the winner single-core processor (Figure 1a) is compared with a heterogeneous multicore processor (Figure 1c) (Kumar, et al. 2005). The heterogeneous or asymmetric multicore processors are widely studied for improving energy efficiency (Annavaram et al. 2005, Balakrishnan et al. 2005 and Morad et al. 2006). They can be utilized to attack Amdahl's law (Hill and Marty 2008). Parallelized portions are executed by multiple small cores and hard-to-parallelize portions are executed by a big and strong core.

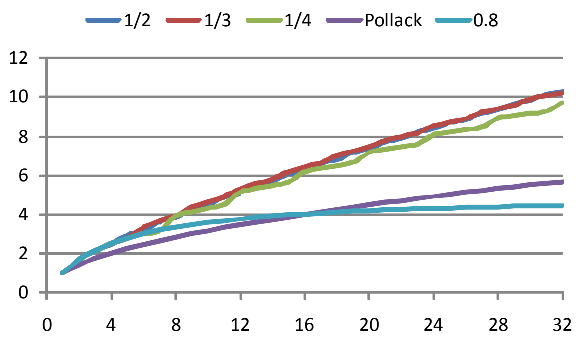

Figure 3: Single-core vs. Heterogeneous Multicore

Figure 3 compares the heterogeneous multicore processors with the single-core processors when p equals 0.8. Each heterogeneous multicore has only one big core. The figure has the same layout to Figure 2. The vertical axis additionally includes performance improvement rate of the heterogeneous multicore processors, I_{Hetero}. The lines labelled with Pollack and 0.8 are for $I_{Pollack}$ and I_{Amdahl}. The other three lines are for I_{Hetero}. The labels 1/2, 1/3, and 1/4 mean the big core occupies half, one third, and one fourth of the chip area, respectively. Pollack's rule models the big core's performance. The rest of chip area is used by the baseline cores. Figure 1c presents the case labelled with 1/4.

Interestingly, the heterogeneous multicore processors have equivalent performance regardless of the big core's size and their performance is much scalable to the chip area. This confirms that hard-to-parallelize portion dominates the speedup. If that portion can be executed by the big core, the speedup is significantly improved.

3.3 Heterogeneous vs. Scalable Homogeneous

Next, the winner heterogeneous (Figure 1c) and a scalable homogeneous multicore (Figure 1d) are compared with each other. The scalable multicore is different from the many-core investigated in Section 3.1 in that the former has the smaller number of large cores. The number of large cores is determined as follows. Guess when utilizing the half number of double-size cores become desirable in terms of performance. That is expressed as:

$$\frac{1}{(1-p)+\frac{p}{N}} < \frac{1}{\frac{1-p}{\sqrt{2}}+\frac{p}{\sqrt{2}\times\frac{N}{2}}}$$

$$N > \frac{\sqrt{2}\times p}{1-p}$$

Hence, when p equals 0.8 for example, 3 double-size cores have better performance than 6 small cores do.

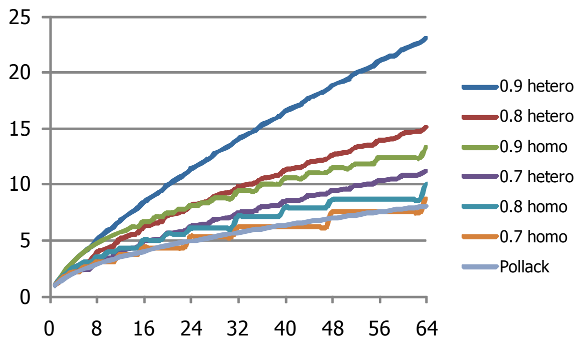

Figure 4: Heterogeneous vs. Scalable Homogeneous

Figure 4 compares two kinds of multicore processors: heterogeneous and scalable homogeneous. The figure has the same layout to Figures 2 and 3. The vertical axis additionally includes performance improvement rate of the scalable homogeneous multicore processors, I_{Homo}. The grey line labelled with Pollack presents $I_{Pollack}$. The other six lines consist of three for the heterogeneous and three for the scalable homogeneous. Each of their labels is the combination of p and the multicore type. For example, "0.9 hetero" and "0.7 homo" present I_{Hetero} when p equals 0.9 and I_{Homo} when p equals 0.7, respectively. The heterogeneous multicore utilizes one fourth of its chip area as its big core.

As for the heterogeneous multicore, performance is still improved when p is increased. When p is increased from 0.8 to 0.9, performance is improved by approximately 50%. In addition, different from the case of many-core, which we have already seen in Figure 2, its scalability is not diminished regardless of p. In contrast, the scalable homogeneous one shows poor performance. Even though p equals 0.9, its performance improvement rate is smaller than that of the heterogeneous multicore when p equals 0.8. This means that the heterogeneous multicore exploit performance even when hard-to-parallelize portions are large.

3.4 Hetero vs. Dynamically configurable

Up to now, the heterogeneous multicore processor is the best choice. However, we only investigated the statically configured multicores. We have not yet considered dynamically configurable multicores such as Core-fusion (Ipek, et al. 2007) and CoreSymphony (Wakasugi, et al. 2010). They dynamically configure the number of cores and the size of each core, as shown in Figure 5. When the currently executing portion of a program is easy to parallelize, the dynamically configurable multicore processor increases the number of cores. In contrast, otherwise, it combines some cores to a large core. The adaptability will improve the performance.

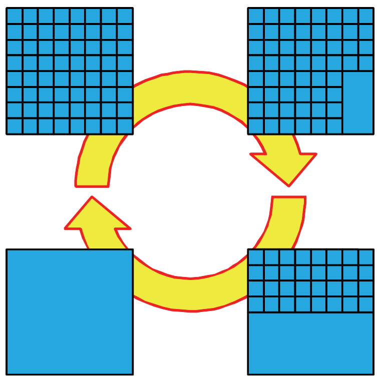

Figure 5: Dynamically configurable Multicore

Figure 6 compares the dynamically configurable multicore processor (Figure 5) with the current winner heterogeneous one (Figure 1c) when p is 0.8. Since we do not perform a simulation of an application, the configuration does not change dynamically but can be determined statically. The parallelizable part is executed by all small cores and the not-parallelizable part is executed by a single core, which is a combination of all small cores. The figure has the same layout to Figures 2-4. There are five lines. The green one labelled with "0.8 hetero" presents I_{Hetero}. The other four lines presents performance improvement rate of the configurable multicore, I_{DC}. We consider four models of the reconfigurable multicore. The dynamic reconfigurability suffers a penalty in performance. It is approximately 25% of performance loss in comparison with a monolithic core (Ipek, et al. 2007 and Wakasugi, et al. 2010). The models labelled with "0.8 DC-n" and "0.8 DC-8" considers the penalty. The other models labelled with "0.8 DC-in" and "0.8 DC-i8" do not consider it and thus has an ideal single-core performance. As the number of cores increases, it becomes difficult to combine all cores due to the increasing complexity of interconnects. Hence, we consider a limit in the number of combinable cores. In this investigation, we assume the number is 8. The models labelled with "0.8 DC-i8" and "0.8 DC-8" consider the limit. The other models labelled with "0.8 DC-in" and "0.8 DC-n" do not consider it and thus are ideally scalable. The model "0.8 DC-8" is the most practical one.

When there are not any limits in the number of combinable cores, the performance improvement is very significant. I_{DC} is almost twice better than I_{Hetero}. The penalty slightly diminishes performance, but the improvement rate is still fascinating. Comparing with I_{Amdahl}, I_{DC} is increased by approximately 400% even if the penalty is considered. However, if the number of combinable cores is limited to 8, I_{DC} is seriously degraded. When N is around 30, the heterogeneous multicore becomes better in performance than the dynamically configurable one. As mentioned earlier, the model "0.8 DC-8" is the most practical one. Comparing it with the heterogeneous multicore unveils that the latter is better when N is larger than 27. The red dashed line in Figure 6 presents the current technology, where four 6-instruction-issue cores or sixteen 2-instruction-issue cores can be integrated on a chip. Remember that the integration is doubled generation by generation. N will be 32 very soon. Considering the above, the heterogeneous multicore processor is the best choice in the near future.

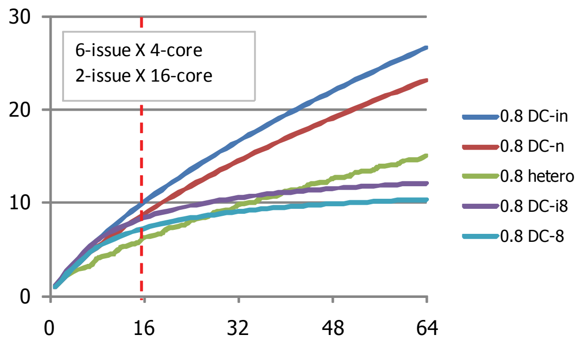

Figure 6: Hetero vs. Dynamically configurable

In order to enjoy the continuously improving performance of the heterogeneous multicore, one serious problem should be solved. It consists of a single big core and a lot of small cores and its configuration is statically determined on the design phase. Both size and performance of the big core have to be increased with the same pace of N. This means that single-core performance is still important.

4 Single-Core Performance Improvement

This section presents a preliminary study that aims to improve single-core performance. Increasing clock frequency is the easiest way to improve single-core performance. However, as widely known, it also increases the power supply voltage, resulting in serious power and temperature problems. This section proposes a technique that increases clock frequency without the increase in the supply voltage.

4.1 Cool Turbo Boost Technique

Intel's Turbo Boost technology (Intel Corporation 2008) has a unique feature that increases the supply voltage and thus the clock frequency when the number of active cores is small. This is possible because TPD (Thermal Design Power) is determined by considering the case when all cores are active and thus it has a large margin in that case. We extend it and further increase the core clock frequency. Different from the baseline Turbo Boost technology, our technique will not require the increase in the supply voltage, and hence we name it Cool Turbo Boost technology.

Cool Turbo Boost exploits the small critical path delay of a small core. If the hardware size and complexity become small, its critical path delay is reduced. Hence, there is an opportunity to increase its clock frequency. Intel's ATOM processor (Thakkar 2008) is a good example that shows a simple and small core improves energy efficiency. In Cool Turbo Boost, processors datapath, where data flow and are processed, dynamically becomes small to boost clock frequency.

When the datapath becomes small, its computing performance is degraded. If the performance loss is not compensated by the clock frequency boost, the total processor performance is diminished. This is not our goal. Hence, we consider the following observation. When instruction level parallelism (ILP) is small, the small datapath is enough. Otherwise, the datapath should not become small. Hence, the datapath is dynamically configured according to ILP in each program phases. In order to realize the idea, we utilize Multiple Clustered-Core Processor (MCCP) (Sato and Funaki 2008). It is shown in Figure 7. MCCP configures its datapath according to ILP and thread level parallelism (TLP) in the program. We extend MCCP so that its clock frequency is increased when it configures its datapath small.

The amount of ILP varies between application programs and by more than a factor of two even within a single application program (Bahar and Manne 2001). Figure 8 shows an example of the issue rate for SPECint2000 benchmark gcc. The horizontal axis indicates the execution cycles and the vertical one represents the average number of instructions issued per cycles (issue IPC) over a window of 10,000 execution cycles. The issue IPC varies by more than a factor of two over a million cycles of execution. These variations can be exploited to determine when the datapath should become small. We manage MCCP to utilize wide datapath only when issue IPC is high and similarly to utilize narrow datapath only when issue IPC is low. When issue IPC is low, there are idle execution resources and thus the narrow datapath provides dependability without serious performance loss. We assume that past program behaviour indicates future behaviour. Hence, based on past issue IPC, future issue IPC could be predicted. We measure the number of instructions issued over a fixed sampling window. We predict future issue IPC based on the past number of issued instructions rather than on past issue IPC. We use predicted issue IPC to determine when the datapath should become small. If it is smaller than a predetermined threshold value, MCCP switches to use the narrow datapath. Similarly, if predicted issue IPC is larger than another predefined threshold value, MCCP switches to use the wide datapath.

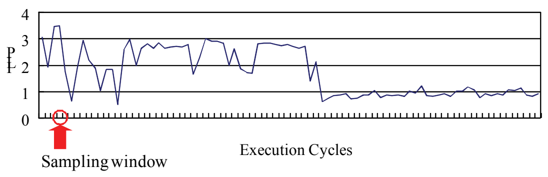

Figure 8: Issue IPC Variation (gcc)

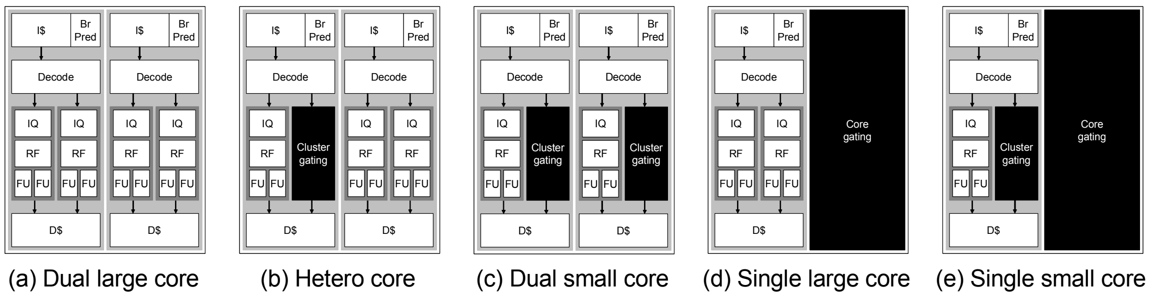

Figure 7: Multiple Clustered-Core Processor

4.2 Evaluation Methodology

SimpleScalar tool set (Austin, et al. 2002) is used for evaluation. Table 1 summarizes the processor configuration. The frontend and the L2 cache do not change. When issue IPC is larger than 2.0, the wide datapath is used. On contrary, when issue IPC is smaller than 1.6, the narrow datapath is used. These threshold values are not optimally determined and thus further study to determine the optimal values is required. Six programs: gzip, vpr, gcc, parser, vortex and bzip2 from SPECint2000 are used. 1 billion instructions are skipped before actual simulation begins. After that each program is executed for 2 billion instructions.

	Narrow	Wide
Fetch width	16 instructions	
L1 I cache	16KB,2-way	
Branch predictor	1K-gshare,512-BTB	
Dispatch width	4 instructions	
Scheduling queue	64 instructions	128 instructions
Issue width	2 instructions	4 instructions
Integer ALUs	2	4
Integer MULs	2	4
Floating ALUs	2	4
Floating MULs	2	4
L1 D cache	16KB,2-way,1-port	16KB,2-way,2-port
L2 cache	512KB,2-way	

Table 1: Processor Configurations

Boosting ratio is defined as the clock frequency in the narrow datapath mode divided by that in the wide mode. We vary the boosting ratio between 1.0 and 2.0 and evaluate how processor performance is improved.

4.3 Results

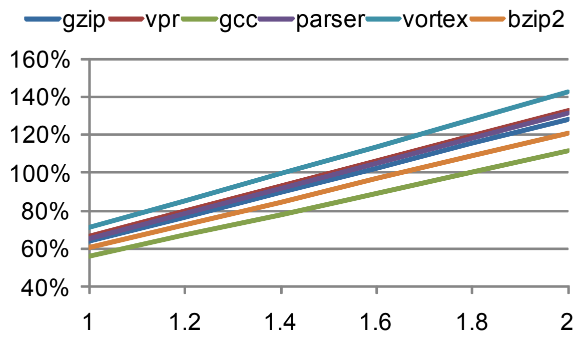

Figure 9: Narrow Datapath Results

Figure 9 presents the normalized performance when the narrow datapath is always used. The horizontal axis indicates the boosting ratio and the vertical one indicates the single-core performance normalized by the baseline performance. When the vertical value is less the 100%, processor performance is degraded. When the boosting rate is 1.0, performance is seriously diminished. It is not improved until the boosting ratio reaches 1.6. Hence, it is very difficult to improve single-core performance only by combining the narrow datapath with high clock frequency.

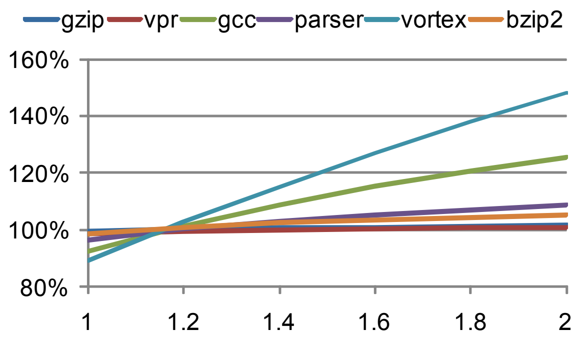

Figure 10: Cool Turbo Boosting Results

Figure 10 presents how Cool Turbo Boosting technique improves single-core performance. The figure has the same layout to Figure 9. When the boosting ratio equals 1.0, performance is degraded in all programs. However, the average performance loss is only 4.2% and is much smaller than that seen in Figure 9, which is 36.1% loss. When the boosting rate reaches 1.4 and 1.6, performance is improved by 5.0% and 8.7% on average, respectively. gzip and gcc achieve significant improvements, which are 26.7% and 15.2%, respectively.

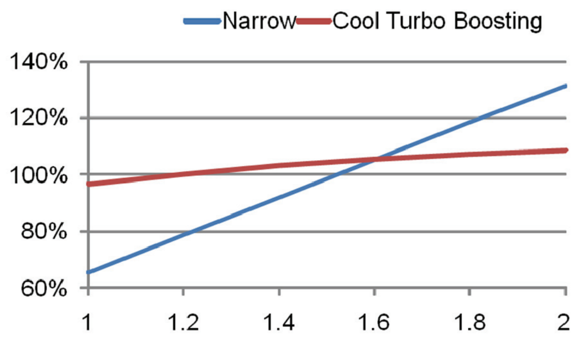

Figure 11: Comparison of 2 Techniques (parser)

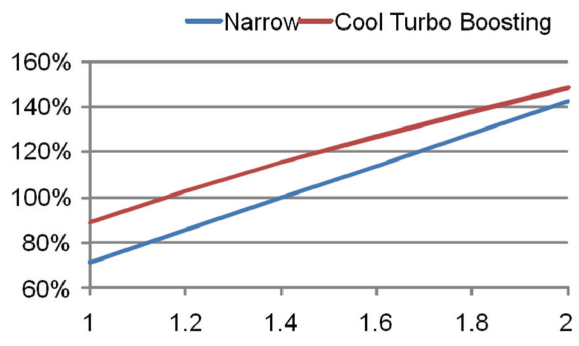

Figure 12: Comparison of 2 Techniques (vortex)

Figures 11 and 12 compare 2 techniques. In Figure 11, parser represents the group of gzip, vpr, parser and bzip2. When the boosting ratio is large, Cool Turbo Boosting technique does not work well. In Figure 12, vortex represents the group of gcc and vortex. Cool Turbo Boosting achieves better performance regardless of the boosting ratio. While determining the boosting ratio requires future studies, the boosting ratio larger than 1.5 will be impractical. Because Cool Turbo boosting achieves single-core performance improvement in the small boosting ratio, it has the potential to further improve performance.

Achieving much single-core performance improvement requires further investigations.

5 Conclusions

This paper investigates what the best multicore configuration is in the future. Five models of single-core, many-core, heterogeneous multicore, scalable homogeneous multicore, and dynamically configurable multicore are compared with each other. From the investigations, it is unveiled that single-core performance is still important. Without the achievement, the heterogeneous multicore processor cannot continue to improve performance in the near future. This is the major contribution of this paper.

In the latter half of the paper, we present the preliminary case study that aims to improve single-core performance. We named it Cool Turbo Boost technique. When ILP in the program is small, the execution resources in the processor are dynamically configured to be narrow and thus its clock frequency is increased. From the detailed simulations, we found the average performance improvement of 5.0% is achieved. Unfortunately, this achievement is not enough to continue multicore performance improvement and the future studies are strongly required.

The future studies regarding the heterogeneous multicore processors include investigating the heterogeneity to enhance dependability. Hardware defects also cause heterogeneity. We are studying to utilize the cores with defects to improve dependability. For example, high performance is not always required for checking correctness. Combining the idea with the findings in this paper will explore a new horizon for dependable multicore processors.

6 Acknowledgments

This work was supported in part by JSPS Grant-in-Aid for Scientific Research (B) #20300019, and is supported in part by JST CREST program and by the fund from Central Research Institute of Fukuoka University.

7 References

Annavaram, M., Grochowski, E. and Shen, J. (2005): Mitigating Amdahl's law through EPI throttling. *Proc. International Symposium on Computer Architecture*, Madison, WI, USA:298-309, IEEE Computer Society Press.

Austin, T., Larson, E. and Ernst, D. (2002): SimpleScalar: an infrastructure for computer system modeling. *IEEE Computer*, **35**(2):59-67.

Bahar, R.I. and Manne, S. (2001): Power and energy reduction via pipeline balancing. *Proc. International Symposium on Computer Architecture*, Goteborg, Sweden:218-229, ACM Press.

Balakrishnan, S., Rajwar, R. Upton, M. and Lai, K. (2005): The impact of performance asymmetry in emerging multicore architectures. *Proc. International Symposium on Computer Architecture*, Madison, WI, USA:506-517, IEEE Computer Society Press.

Borkar, S. (2007): Thousand core chips: a technology perspective. *Proc. Design Automation Conference*, San Diego, CA, USA:746-749, ACM press.

Ekman, M. and Stenstrom, P. (2003): Performance and power impact of issue-width in chip-multiprocessor cores. *Proc. International Conference on Parallel Processing*, Kaohsiung, Taiwan: 359-368, IEEE Computer Society Press.

Hill, M.D. and Marty, M.R. (2008): Amdahl's law in the multicore era. *IEEE Computer* **41**(7): 33-28.

Howard, J., et al. (2010): A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS. *Digest of Technical Papers International Solid-State Circuit Conference*, San Francisco, CA, USA:19-21, IEEE Press.

Intel Corporation (2008): Intel^R Turbo Boost technology in Intel^R Core^{TM} microarchitecture (Nehalem) based processors. *White Paper*.

Ipek, E., Kirman, M., Kirman, N. and Martinez, J.F. (2007): Core fusion: accommodating software diversity in chip multiprocessors. *Proc. International Conference on Computer Architecture*, San Diego, CA, USA:186-197, ACM Press.

Kumar, R., Tullsen, D.M., Jouppi, N.P. and Ranganathan, P. (2005): Heterogeneous chip multiprocessors. *IEEE Computer* 38(11): 32-38.

Morad, T.Y., Weiser, U.C., Kolodny, A., Valero, M. and Ayguade, E. (2006): Performance, power efficiency and scalability of asymmetric cluster chip multiprocessor. *IEEE Computer Architecture Letters* **5**(1): 14-17.

Sato, T. and Funaki, T. (2008): Dependability, power, and performance trade-off on a multicore processor. *Proc. Asia and South Pacific Design Automation Conference*, Seoul, Korea:714-719, IEEE Computer Society Press.

Thakkar, T. (2008): Intel Centrino Atom processor technology-enabling the best internet experience in your pocket. *Proc. Symposium on Low-Power and High-Speed Chips*, Yokohama, Japan:329-337.

Wakasugi, Y., Sakaguchi, Y., Miyoshi, T. and Kise, K. (2010): An efficient physical register management scheme for CoreSymphony architecture. *IPSJ SIG Technical Report*, **2010-ARC-188**(3): 1-10 (in Japanese).

Wang, Y., An, H., Yan, J., Li, Q., Han, W., Wang, L. and Liu, G. (2009): Investigation of factor impacting thread-level parallelism from desktop, multimedia and HPC applications. *Proc. International Conference on Frontier of Computer Science and Technology*, Shanghai, China:27-32.