An Energy-Efficient Speculative Chip-Multiprocessor Utilizing Trace-level Value Prediction

Takenori Koushiro

Department of Artificial Intelligence
Kyushu Institute of Technology
zin@mickey.ai.kyutech.ac.jp

Toshinori Sato

Precursory Research for Embryonic
Science and Technology
Japan Science and Technology Corporation
toshinori.sato@computer.org

Abstract

While lower supply voltage is effective for reducing energy consumption, it suffers the problem of performance loss. In order to mitigate the performance loss, we propose to execute only the part, which does not have any influence on execution speed, with low-speed and low-voltage. We are investigating a multithreaded execution, named Contrail Architecture*, which divides an instruction stream into two streams using trace-level value prediction. One is the speculation stream, which is the main part of an application program and is applied value predictions, and the other is the verification stream, which verifies the value predictions. The energy consumption is reduced by the decrease in the number of instructions in the speculation stream and by the low-speed execution in the verification stream. In this paper, we evaluate the energy consumption and performance of a Contrail processor.*

Keywords: trace-level value prediction, speculative chip-multiprocessors, energy efficiency

1 Introduction

The increasing popularity of portable and mobile computer platforms such as laptop PCs and smart cell phones is a driving force in the investigation of high-performance and power-efficient microprocessors. For example, modern embedded microprocessors support out-of-order execution. As the computing power of microprocessors for mobile devices increases, however, their power consumption also increases. In addition, while power is already a major design constraint in the area of mobile and embedded computer platforms, it has also become a limiting factor in general-purpose microprocessors.

The energy consumed in a microprocessor is the product of its active power and execution time. Thus, to reduce energy consumption, we should decrease either or both of them. The active power P_{active} and gate delay t_{pd} of a CMOS circuit are given by

$$P_{active} \propto f C_{load} V_{dd}^2 \quad (1)$$

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd} - V_{th})^{\alpha}} \quad (2)$$

where f is clock frequency, C_{load} is load capacitance, V_{dd} is supply voltage, and V_{th} is the threshold voltage of the device. α is a factor depending upon the carrier velocity saturation and is about 1.3–1.5 in advanced MOSFETs [8]. Based on Eq.(1), it is easily found that power supply reduction is the most effective way to lower power consumption. However, Eq.(2) tells us that supply voltage reduction increases gate delay, resulting in a slower clock frequency. Thus, the computing performance of the microprocessor is diminished.

In order to mitigate the performance loss, we can exploit parallelism [5]. Two identical circuits are used in order to make each unit to work at half the original frequency while the original throughput is maintained. Since the speed requirement for the circuit becomes half, the supply voltage can be decreased. In this case, the amount of parallelism can be increased to further reduce the total power consumption. Another kind of parallelism, which is thread level parallelism, is utilized for energy reduction with maintaining processor performance [12]. In this paper, we propose an energy-efficient speculative chip-multiprocessor based on trace-level value prediction.

The reminder of this paper is organized as follows: Section 2 describes the Contrail processor architecture. Section 3 proposes a hardware-cost effective trace-level value predictor. Section 4 presents our preliminary evaluation results. Section 5 surveys related works and Section 6 concludes the paper.

2 Contrail Processor Architecture

To reduce the energy consumption, we divide an execution of an application into two streams. One is called the *speculation stream* and consists of the main part of the execution. However, it exploits trace-level value prediction [15, 20], and thus several regions of the execution are skipped. In other words, the number of instructions in the speculation stream is smaller than that in the original execution, resulting in energy reduction. In contrast, the other stream is called the *verification stream* and supports the speculation stream by verifying each value prediction. The key idea is that the trace-level value prediction translates each critical path into a non-critical one and we move it from the speculation stream into the verification stream. Hence, the verification stream can execute slowly if the value prediction accuracy is considerably high. We can reduce the clock frequency of the components for the verification stream. Furthermore, the supply voltage is also reduced. From these considerations, its energy consumption is significantly reduced. This technique is called Contrail architecture [21, 22], because the speculation stream runs ahead just like a jet plane, and the verification stream is left behind by the speculation stream and fades away in the manner of a contrail.

Each stream executes as a thread on a chip multiprocessor (CMP) [14, 27], each processing element (PE) of which independently works at the variable clock frequency and supply voltage [9, 26]. The speculation stream is executed on a PE in high-speed mode and the verification stream is executed on another PE in low-speed and low-supply-voltage mode. In the ideal case, that means there are no mispredictions; the speculation stream finishes silently and waits for the verification process. In the case in which a misprediction is detected in a verification stream, all threads from the misprediction point to tail, including the speculation stream and any verification streams, are squashed, and processor state is recovered by the verification stream that detect the misprediction. And then, the verification stream becomes the speculation stream.

The cost of spawning a new thread might be larger than the cost of verifying a value prediction on a single-threaded processor. However, in single-threaded processor model, only datapath alternates between high-speed and low-speed modes. The other blocks, especially instruction-supply front-end, should be always in high-speed mode. This reduces the effect of the variable frequency and voltage technique. On the other hand, every component of each processor core can alternate two modes in CMP model, and thus improving energy efficiency is expected. From these observations, we determined to adopt multi-threaded model rather than single-threaded model.

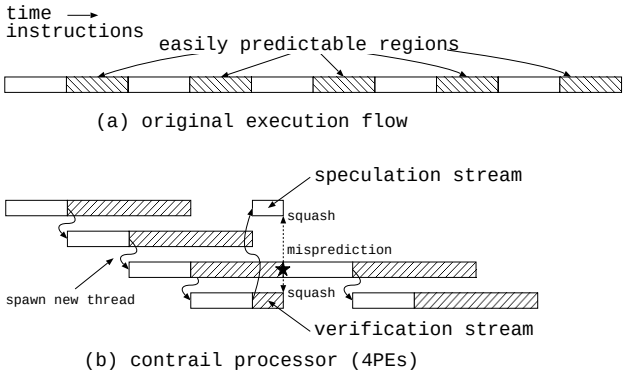

Figure 1: Execution on a Contrail processor

We explain how the program is logically executed on a Contrail processor, using Figure 1. We assume that half of the original execution of an application is ideally predicted and is distributed uniformly as explained in Figure 1(a). This is a reasonable assumption, since it has been reported that 59% of dynamic traces can be reused with the help of the value prediction [15]. We also assume the clock frequency for the verification PEs at half that for the speculation PE. Under these assumptions, the execution is divided into speculation and verification streams in a Contrail processor in a distributed manner as depicted in Figure 1(b). The predicted regions are skipped in the speculation stream and execute in the verification streams while enlarging their execution time. Determining trigger points is based on the confidence information obtained from the trace-level value predictor. When an easily predictable region is detected, the head thread spawns a new speculation stream on the next PE and it turns into a verification stream. Thus, the Contrail processor can be build as a ring-connected chip-multiprocessor such as MultiScalar architecture [7], as shown in Figure 2. Each verification stream stays alive until all instructions in the corresponding trace removed from the speculation stream are executed. After that, the verification PE is released for a future speculation stream.

Each PE has its dedicated voltage/frequency controller. It also has a trace cache [18] and a trace-level value predictor. One of the differences from these processors is that the Contrail processor does not require any mechanism to detect memory dependence violations. Since the Contrail processor strongly relies on

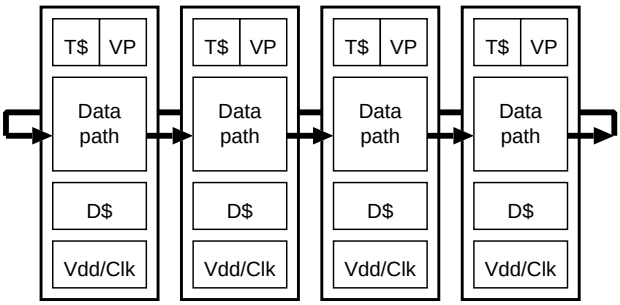

Figure 2: Contrail processor

trace-level value prediction, any memory dependence violations cannot occur. Instead, it suffers from value mispredictions. This simplifies hardware complexity. This is because value mispredictions can be detected locally in an PE, while detecting memory dependence violations requires a complex mechanism such as ARB or Versioning Cache [7].

The potential effect of the Contrail processor architecture on energy-efficiency is estimated as follows: We determine the clock frequency and supply voltage for the verification PEs at half that for the speculation PE. Energy consumption is calculated as follows: For the speculation stream, energy consumption becomes half that of the original execution since the number of instructions is reduced by half. In contrast, for the verification streams, the sum of every execution time remains unchanged since the execution time of each instruction increases doubly while the total number of instructions is reduced by half. Its energy consumption is decreased by the reduction of the clock frequency and the supply voltage. Based on Eq.(1), it is reduced to $\frac{1}{8}$. Thus, the total energy savings is 37.5%. It is true that the effectiveness of Contrail processors (energy savings) depends on the value prediction accuracy and the size of each predicted region. However, we have confirmed that the potential effect of Contrail processors on energy savings is substantial.

One of the differences between this architecture and previously proposed pre-computing architectures [19,25] is that the Contrail processor architecture does not rely on redundant execution. In the ideal case, the number of executed instructions is unchanged. Another difference is that its target is the improvement of energy efficiency instead of the improvement of performance.

3 Trace-Level Value Prediction

As described above, the Contrail processors strongly rely on the trace-level prediction. In order to minimize the overhead in energy consumption due to value prediction, any low-cost trace-level value predictor is required. In this section, we will propose a low-cost trace-level predictor, which we call the decoupled trace-level value predictor [10].

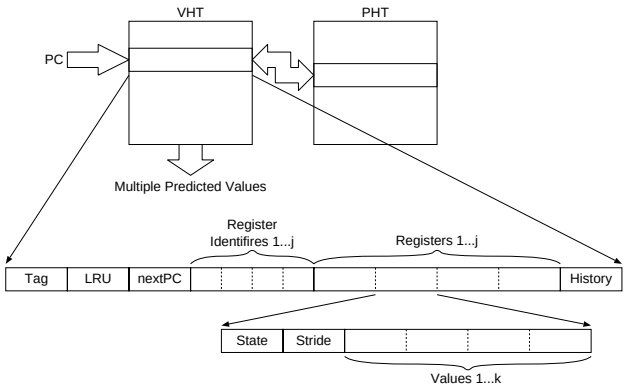

Figure 3: Conventional trace-level value predictor

Figure 3 shows a conventional trace-level value predictor [20], which holds up to four unique values for every register value. The first table, VHT, consists of `Tag`, `LRU Info`, `next PC`, `Register Identifiers`, `Register Values`, and `Value History Pattern` fields. The `Tag` field is used for distinguishing individual among traces. The `LRU Info` field keeps track of the order in which the four data values appear. The four values are identified using binary encoding, {`00, 01, 10, 11`}. The `Register Identifiers` field tells which registers are included in the associated trace. Each `Register Values` holds `State`, `Stride`, and four `Data Values` fields. The `State` field decides whether the prediction should be initiated. The `Stride` field holds the difference between the last two values generated by the instruction. The `Value History Pattern` field saves the history of the last p outcomes for an instruction. The history is maintained as a $2p$-bit pattern consisting of the p binary encoding (2-bit) codes explained above. Every $2p$-bit pattern kept in the `Value History Pattern` field is used for indexing the second table, the Pattern History Table (PHT). An entry in the PHT has four saturating up-down counters, each of which is associated with a corresponding value held in the VHT `Data Values` field. The counter values are used for selecting one of the four values as the predicted value. When an actual value is obtained, the counter corresponding to the correct outcome is incremented and the others are decremented.

As can be easily observed, the hardware cost of the conventional trace-level predictor increases signif-

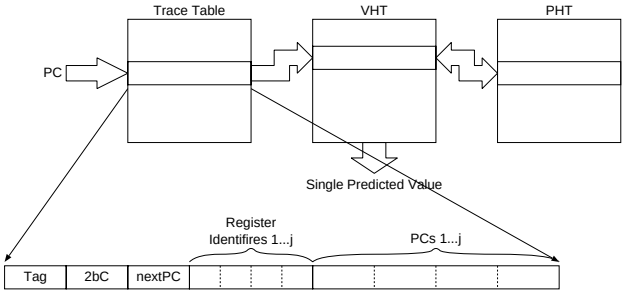

Figure 4: Decoupled trace-level value predictor

icantly as the number of registers included in a trace is increased. In order to solve the hardware cost explosion, we propose to decouple the trace information from the value prediction information as shown in Figure 4. We call this predictor the decoupled trace-level value predictor. It consists of a table keeping trace information, which we call the trace table (TT), and an instruction-level value predictor. The TT consists of `Tag`, 2-bit saturating up-down counter (`2bC`), `next PC`, `Register Identifiers`, and `PCs` fields. The `Tag` field is used for distinguishing individual among traces. The `2bC` field decides whether the prediction should be initiated. The `Register Identifiers` field tells which registers are included in the associated trace, and each instruction address in the `PCs` field is associated with the register identified in the `Register Identifiers` field. Based on the trace information provided by the TT, the instruction-level value predictor is referred multiple times. That is, multiple register values are predicted sequentially. This reduces the hardware cost of the value predictor considerably, while it requires several clock cycles to predict the whole register values in a trace.

4 Evaluation

4.1 Methodology

We implemented a timing simulator using the SimpleScalar tool set (ver.3.0) [3]. The SimpleScalar/PISA instruction set architecture (ISA) is based on MIPS ISA. The SPEC2000 CINT benchmark suite is used for this study. Table 1 lists the benchmarks and the input sets. We use the object files provided by the University of Michigan. For each program, 1 billion instructions are skipped before the actual simulation begins, and the next 100 instructions are executed. We do not count nop instructions.

Because our chip-multiprocessor simulator is under construction, we use an out-of-order superscalar processor simulator in this evaluation. The evaluated processor models are listed in Table 2. While the model,

Table 1: Benchmark programs

program	input set
164.gzip	input.compressed
175.vpr	net.in arch.in
176.gcc	cccp.i
197.parser	2.1.dict
255.vortex	lendian.raw

`Contrail`, benefits from value prediction, the remaining models, `Fast` and `Slow`, do not. Each model has 6 integer ALUs. The fast ALUs can execute integer operations in one cycle, and the slow ALUs execute operations in two cycles.

Table 2: Evaluated configurations

Model	# fast ALU	# slow ALU
Contrail	3	3
Fast	6	0
Slow	0	6

As explained, the key idea of energy reduction via value prediction is the verification should not be fast if it is expected that its prediction is highly accurate. In other words, predicted instructions are not on critical paths and thus they can be executed on the slow ALU. Such units can be operate by low supply voltage, resulting in significant power reduction according to Eq(1). If a prediction is correct, there are no penalty on execution cycles and thus energy consumption is reduced. Otherwise, there are considerable penalty due to long latency of each mispredicted instruction and re-execution of each misspeculated instruction, resulting in degrading energy efficiency. In summary, in this evaluation, only unpredictable instructions are executed on fast and power-hungry ALUs, and predictable instructions can be executed on the slow and power-efficient ALUs, as shown in Figure 5.

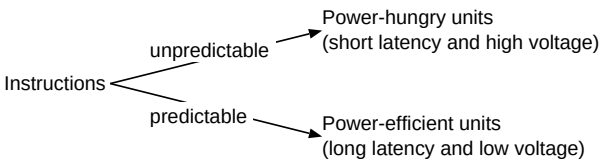

Figure 5: Instruction execution policy

In this evaluation, we use an instruction-level hy-

brid value predictor [20] instead of the trace-level predictor described in Section 3. The trace-level predictor has been evaluated using a functional simulator and the results can be found in [10]. Currently, we are integrating the trace-level predictor into the timing simulator. The instruction-level hybrid predictor consists of a 4096-entry VHT and a 4096-entry PHT, and each entry keeps 4 data values. When a misspeculation occurs, it is necessary to revert the processor state to a safe point where the speculation is initiated. We use an instruction reissue mechanism which selectively flushes and reissues misspeculated instructions [7].

Table 3: Supply voltage vs. frequency

Frequency	Supply voltage
400 MHz	1.0V
800 MHz	1.3V

We will evaluate a scaling for supply voltage and clock frequency based on Intel XScale [9]. The scaling is summarized in Table 3 [11]. The higher frequency and voltage are applied to the fast ALUs and the lower ones are applied to the slow ALUs. Note that in this evaluation clock frequency and supply voltage are not dynamically adaptable but are unchanged. The energy consumed in ALUs is calculated as the product of its active power and execution time. Because power is proportional with clock frequency and quadratically with supply voltage, we can estimate the energy from the active rate of each unit and the execution cycles, using the supply voltage and frequency scaling shown in Table 3. While using Wattch [2] or SimplePower [29] will give us more detailed results, it is remained as a future study.

4.2 Results

Figure 6 explains where and how instructions are executed. Each bar is divided into three part. The bottom part indicates the percentage of instructions that are executed in the speculation stream. The middle part indicates the percentage of instructions that are executed in the verification stream and their corresponding value predictions are correct. And the top part indicates the percentage of instructions that are executed in the verification stream and their corresponding value predictions are mispredicted. On average, 32% of instructions are executed in the verification stream and found to be correctly predicted, contributing energy reduction.

Figure 7 presents execution cycles. For each group of three bars, the left one indicates execution cycle of

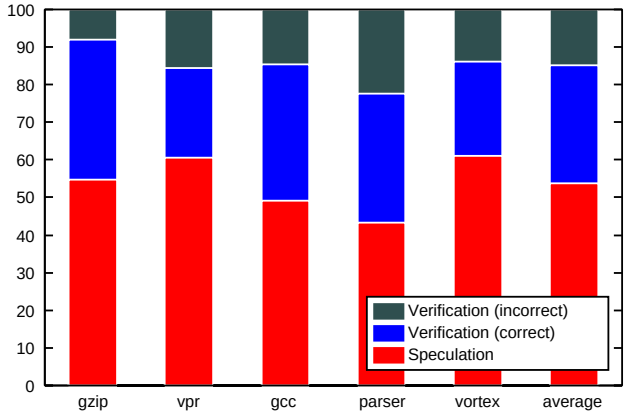

Figure 6: %Classification of instructions

`Contrail` model, and the middle and the right ones indicate those for `Fast` and `Slow` models, respectively. Each execution cycle is normalized by that of the `Fast` model. As can be easily seen, while `Slow` model degrades its performance significantly, `Contrail` model mitigates the performance loss.

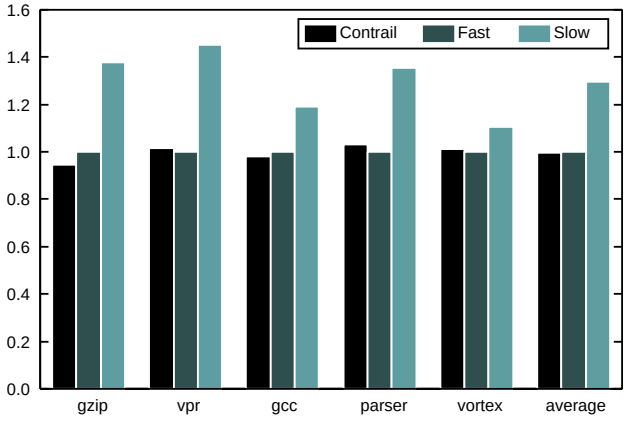

Figure 7: Relative execution cycles

Figure 8 shows energy reduction of `Contrail` model from `Fast` model. We can find an average of 28% energy reduction in ALUs is achieved.

5 Related Work

This section surveys techniques exploiting instruction criticality for power reduction.

The critical path is a chain of dependent instructions, which determines the number of cycles executing the program. And thus, the performance of the processor is limited by the speed at which it executes the instructions along the critical path. If we can identify which instructions are critical, we can accelerate

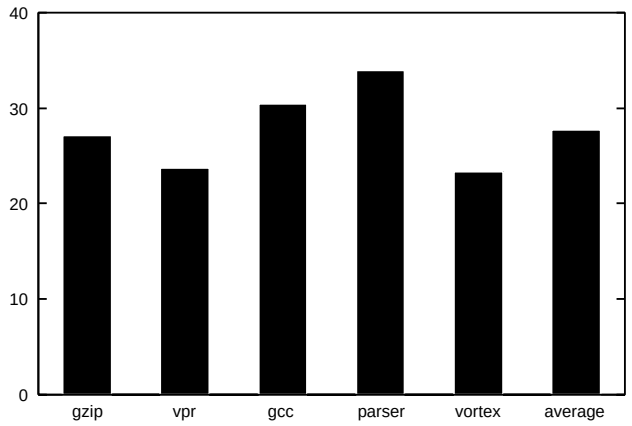

Figure 8: %Energy reduction

their execution by any means. Critical path prediction [6,28] is such a technique for identifying critical instructions dynamically.

Exploiting information regarding instruction criticality is effective not only for improving processor performance but also for reducing energy consumption [4,16,22,24]. Casmira et al. [4] studied the potential benefit of exploiting instruction criticality, which they call slack, for power reduction. They found there is significant availability of non-critical instructions, but did not mention any practical mechanism that utilize the results for power reduction. Pyreddy et al. [16] use profiled-based heuristics proposed by Tune et al. [28] for identifying critical instructions. From a profile run, each instruction is marked as critical or non-critical. When the program is executed, the critical instructions are executed on fast and power-hungry functional units and the non-critical ones are executed on slow and power-efficient units. In contrast, Sato et al. [22] and Seng et al. [24] utilized a dynamic mechanism. They proposed to use the critical path predictor to identify non-critical instructions. Seng et al. [24] focused on eliminating hot spots of power density in general-purpose high-performance microprocessors. On the other hand, Sato et al. [22] evaluated the potential of energy reduction via this criticality-based instruction scheduling.

6 Summary

Currently, it is expected that multi-threading and dual-power functional units are key techniques for energy reduction [17]. We have proposed such an energy-efficient speculative chip-multiprocessor based on value prediction. Our proposed architecture exploits thread level parallelism, resulting in mitigating performance loss caused by the supply voltage reduction. In this paper, we show preliminary evaluation results of Contrail processor, which is unfortunately based on the single processor model. We found that the contrail processor has a potential of approximately 30% energy savings in ALUs with maintaining processor performance.

At the moment, we do not consider energy consumed in the value predictor. Recently, some studies regarding power consumption of value predictors are reported [1,13] and it is found that complex value predictors are power-hungry. One of the solutions for reducing power consumed in value predictors is using simple value predictors such as zero-value predictor [23]. Thus, future study regarding the Contrail processor includes investigating energy-efficient value predictors.

Acknowledgments

The authors thank the anonymous reviewers whose comments and suggestions helped to improve the quality of this paper.

References

[1] R.Bhargava, L.John, "Latency and energy aware value prediction for high-frequency processors," 16th International Conference on Supercomputing, June 2002.

[2] D.Brooks, V.Tiwari, M.Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," 27th International Symposium on Computer Architecture, June 2000.

[3] D.Burger, T.M.Austin, "The SimpleScalar Tool Set, Version 2.0," ACM SIGARCH Computer Architecture News, vol.25, no.3, June 1997.

[4] J.Casmira, D.Grunwald, "Dynamic instruction scheduling slack," Kool Chips Workshop, December 2000.

[5] A.P.Chandrakasan and R.W.Brodersen, "Minimizing power consumption in digital CMOS circuits," Proceedings of IEEE, vol.83, no.4, April 1995.

[6] B.A.Fields, S.Rubin, R.Bodik, "Focusing processor policies via critical-path prediction," 28th International Symposium on Computer Architecture, June 2001.

[7] M.Franklin, "Multiscalar processors," Kluwer Academic Publishers, 2003.

[8] T.Hiramoto, M.Takamiya, "Low power and low voltage MOSFETs with variable threshold voltage controlled by back-bias," IEICE Transactions on Electronics, vol.E83-C, no.2, February 2000.

[9] Intel Corporation, "Intel$^{(R)}$ XScale^{TM} technology," `http://developer.intel.com/design/intelxscale/`, 2002.

[10] T.Koushiro, T.Sato, I.Arita, "Decoupled trace-level value predictor for hardware cost reduction," 14th Joint Symposium on Parallel Processing, May 2002 (In Japanese).

[11] M.Levy, "Samsung twists ARM past 1GHz," Microprocessor report, vol.16, arc.10, October 2002.

[12] S. Matsushita, S. Torii, M. Nomura, T.Inoue, A.Shibayama, S.Shimada,T.Osawa,H.Inoue, K.Minami, J.Sakai, Y.Ito, Y.Nakamura, M.Edahiro, N.Nishi, M.Yamashina, "Merlot: a single-chip tightly coupled four-way multi-thread processor," COOL Chips III, April 2000.

[13] R.Moreno, L.Pinuel, S.Del-Pino, F.Tirado, "A power perspective of value speculation for superscalar microprocessors," International Conference on Computer Design, September 2000.

[14] K. Olukotun, B.A. Nayfeh, L. Hammond, K. Wilson, K. Chang, "The case for a single-chip multiprocessor," 7th International Conference on Architectural Support for Programming Languages and Operating Systems, October 1996.

[15] M. L. Pilla, A. T. da Costa, F. M. G. Franca, P. O. A. Navaux, "Predicting trace inputs with dynamic trace memoization: determining speedup upper bounds," 10th International Conference on Parallel Architectures and Compilation Techniques, WiP session, September 2001.

[16] R.Pyreddy, G.Tyson, "Evaluating design tradeoffs in dual pipelines," Workshop on Complexity-Effective Design, June 2001.

[17] J.Rattner, "Electronics in the Internet age," 10th International Conference on Parallel Architectures and Compilation Techniques, Keynote Address, September 2001.

[18] E.Rotenberg, S.Bennet, J.Smith, "Trace cache: a low latency approach to high bandwidth instruction fetching," 29th International Symposium on Microarchitecture, December 1996.

[19] A.Roth, G.Sohi, "Speculative data-driven multithreading," 7th International Symposium on High-Performance Computer Architecture, January 2001.

[20] R.Sathe, K.Wang, M.Franklin, "Techniques for performing highly accurate data value prediction," Microprocessors and Microsystems, vol.22, no.6, November 1998.

[21] T.Sato, I.Arita, "Contrail processors for converting high-performance into energy-efficiency," 10th International Conference on Parallel Architectures and Compilation Techniques, WiP session, September 2001.

[22] T.Sato, T.Koushiro, A.Chiyonobu, I.Arita, "Power and performance fitting in nanometer design," 5th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, January 2002.

[23] T.Sato, I.Arita, "Low-cost value predictors using frequent value locality," 4th International Symposium on High Performance Computing, May 2002.

[24] J.S.Seng, E.S.Tune, D.M.Tullsen, "Reducing power with dynamic critical path information.," 34th International Symposium on Microarchitecture, December 2001.

[25] K. Sundaramoorthy, Z. Purser, E. Rotenberg, "Slipstream processors: improving both performance and fault tolerance" 9th International Conference on Architectural Support for Programming Languages and Operating Systems, November 2000.

[26] Transmeta Corporation, "Crusoe^{TM} processor model TM5800," Product Brief, May 2001.

[27] M.Tremblay, "An architecture for the new millennium," Hot Chips 11, August 1999.

[28] E.Tune, D.Liang, D.M.Tullsen, B.Calder, "Dynamic prediction of critical path instructions," 7th International Symposium on High Performance Computer Architecture, January 2001.

[29] N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, W. Ye, "Energy-driven integrated hardware-software optimizations using SimplePower," 27th International Symposium on Computer Architecture, June 2000.