Next Entries Pre-activation for Low Power Drowsy BTB

Wei-Hau Chiao, Han-Lun Pan, Jean Jyh-Jiun Shann, and Chung-Ping Chung
Department of Computer Science
National Chiao Tung University
Taiwan, R.O.C
{whchiao, hlpan, jjshann, cpchung}@cs.nctu.edu.tw

ABSTRACT

Recently, low-power design becomes popular research topics. This paper addresses power mode management for drowsy BTB. We propose an accurate next entry pre-activation method cooperated with the decay policy to reduce BTB leakage energy consumption. In the decay policy, a BTB entry which has not been accessed for a period of time is put into drowsy mode. Moreover, the BTB locations of all branch instructions existing in BTB are cached for next BTB entry pre-activation. Simulation results show that our method reduces the leakage consumption in BTB by an average of 61.9% with only paying 0.5% performance cost.

1: Introduction

Recently, most portable devices are battery-powered, such as MP3 player, mobile phone, and personal digital assistant (PDA). Low power design helps to increase the battery life time of these devices. On the other hands, in high-end machines, high working voltage produces enormous heat, and high temperature which causes the system unreliable. Low power design helps to promote the system reliability. Therefore, low-power issues become popular research topics.

The power consumption of CMOS circuits consists of dynamic power and leakage power. While dynamic power is mainly dissipated by charging and discharging the gate output capacitance, the leakage power is induced by the current that leaks through transistors even when they are turned-off. The International Technology Roadmap for Semiconductors projects that leakage power consumptions will exceed dynamic ones as technology drops below the 65-nm feature size [1]. The leakage power is posing new low-power design challenges.

Almost all processors are deeply pipelined today. In order to reduce pipeline stall cycles due to branch instructions, most processors adopt dynamic branch prediction techniques. BTB, which is on-chip memory with the tag and branch target address RAMs, is used to support dynamic branch prediction. It dissipates about 7% of the total processor power [2]. Besides caches, BTB is the largest on-chip memory in processor. Moreover, BTB is a thermal hotspot [3]. The leakage energy of BTB may be more than its size would suggest since the leakage energy increases exponentially with temperatures increment.

Drowsy cache [4] is a well-known circuit technique for SRAM leakage power reduction. It provides two power modes for each row of SRAM arrays. While the normal mode is fully functional, the drowsy mode is data preserving only. A drowsy row needs to be waked up with about one cycle latency before accessed. Since only a small part of BTB entries would be accessed in a period of time due to the program locality reasons, if the other unused entries can be managed into drowsy mode, the BTB leakage energy would be reduced significantly.

In this paper, we propose a next entry pre-activation method cooperated with the decay policy [5] to manage power modes of each BTB entries. In the decay policy, a BTB entry which has not been accessed for a period of time (decay interval) is put into drowsy mode. In next entry pre-activation, the possible BTB next entries are cached for each branch instruction existing in BTB and used to pre-activate the BTB entries that will be used in the near feature. Unlike the decay management, our method may help to shorten the decay interval and hide wake-up latency, and both of these benefit leakage energy reduction.

The rest of this paper is organized as follows. Section two presents the related work of leakage reduction techniques. Section three describes our power mode management policies for BTB. Section four gives the simulation results, and the last section is our conclusion.

2: Related Work

2.1: Circuit techniques to reduce SRAM leakage

The circuit techniques to reduce SRAM leakage power are classified into state-destroying and state-preserving techniques. State-destroying techniques such as gated-Vdd [6] that gate the supply voltage of the SRAM cells in its leakage saving mode. Therefore, the stored data is lost in this mode.

Unlike the state-destroying methods, state-preserving techniques preserve the data stored in SRAM cells in its leakage saving mode. One popular method of state-preserving techniques is scaling the supply voltage of SRAM cells dynamically [7]. It provides two power modes with different supply voltages for each row of SRAM arrays. In normal mode, the data stored in

SRAM cells can be accessed arbitrarily. In drowsy mode, the data in SRAM cells is preserved only. If an access requirement is comes for the drowsy row of the SRAM, it needs to be waked up with about one cycle latency before accessed.

2.2: Techniques for BTB leakage reduction

In [5], it adopts gated-Vdd circuit in BTB and use a decay approach to manage the power modes. If a BTB entry has not been accessed for a fixed period of time (decay interval), this entry is expected as useless and is gated its supply voltage immediately (destroyed mode). If an access requirement comes to this destroyed BTB entry, a branch misprediction may happened since the branch target address of this branch instruction is lost. The simulation results show that the decay interval of this method is very long (about 64k cycles) to ensure a BTB entry is useless.

In [8], it adopts drowsy circuit and uses a compiler-directed approach to manage the BTB power modes in a loop. During the first iteration of a loop, the accessed BTB entries would be marked. During the other iterations of the loop, the BTB entries without marking are placed into drowsy mode. All BTB entries are switched to normal mode after exiting the loop.

2.3: Observations

In [5], thousands cycles are required as the decay interval to ensure a BTB entry is useless. This limits the low-power opportunity. The compiler-directed approach only covers the branch instructions in loop, not the whole program. Moreover, it needs to insert extended instructions to support compiler-directed power mode managements. It causes the increase of both code size and instruction decoder complexity.

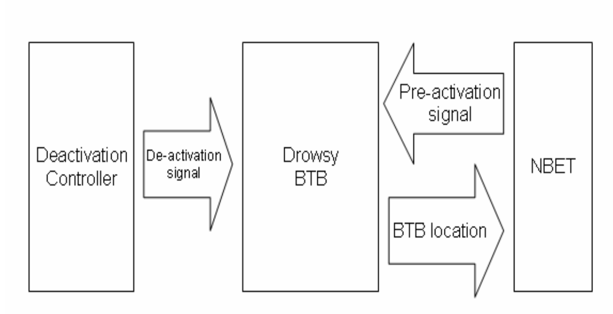


Figure 1: architecture overview of our design

3: Pre-activation policy

3.1: Overview of pre-activation policy

Generally speaking, for each branch instruction, there are at most two possible directions, taken or non-taken. Moreover, most branch instructions have fixed target address. Therefore, for most branch instructions, the possible next branch instructions on execution path are fixed.

In a drowsy BTB, if the next BTB location is revealed to processor early, the pre-activation operation becomes trivial. Therefore, our research focuses on how to record the next possible BTB locations.

Figure 1 shows the architecture overview of our design. The deactivation controller puts a BTB entry into drowsy mode if this entry has not been accessed for a period of time (the decay approach proposed in [5]). Next BTB Entry Table (NBET) collects BTB location for next BTB entry pre-activation.

3.1: Two-direction pre-activation policy

3.1.1: Two-direction based NBET. In order to record the two possible next BTB locations (which set and which way in BTB) of all branch instructions existing in BTB, we configure NBET as two-direction based NBET which has the same number of entries as BTB. Each entry has two fields: one for next BTB entry of taken path and another one for not-taken path.

3.1.2: Operation of two-direction based NBET

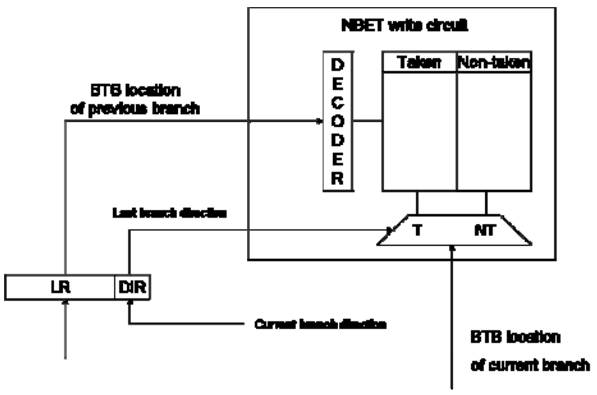


Figure 2: Next BTB location collection circuits

Initially, All NBET fields are invalid. Figure 2 shows the next BTB location collection circuit. The branch target of a branch instruction is resolved after the branch instruction is executed; however, the BTB location of the next branch instruction is resolved while the next branch instruction is executed. Therefore, in next BTB entry collection circuit, while the branch target address of the current branch instruction is updated into BTB, the current BTB location is updated into the NBET entry of previous branch instruction. Note that the write-port of NBET is independent of BTB. The BTB location and branch direction of one branch instruction is reserved in the Location Register (LR) and direction bit (DIR) until the next branch instruction is executed. A row decoder and a selection circuit are required to enable the write-port of the corresponding NBET field. After the collection operation, the NBET field becomes valid until the corresponding BTB entry is replaced.

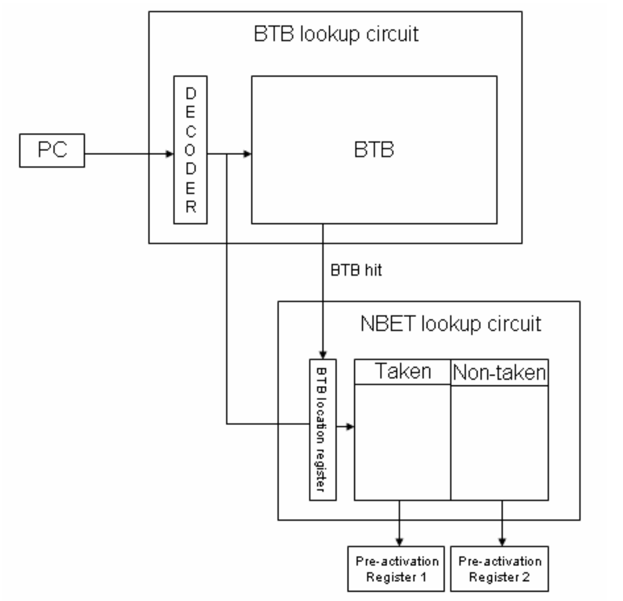


Figure 3: Lookup-circuit of two-direction based NBET

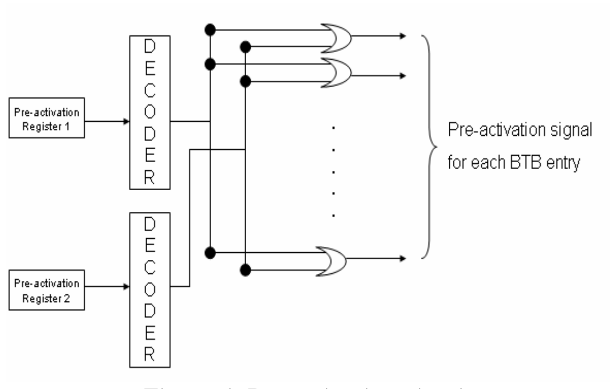

Pre-activation Register 1
Pre-activation Register 2
DECODER
DECODER
Pre-activation signal for each BTB entry
Figure 4: Pre-activation circuit

Figure 3 displays lookup-circuit of two-direction based NBET. BTB is looked up every cycle for target address prediction. If BTB hits, it means that current instruction is a branch instruction and the outputs of BTB row decoder are latched in BTB location register. Then, in the next cycle, the NBET lookup operation is performed to get the next BTB location. Note that this operation is performed only while the BTB lookup is hit in the previous cycle. After NBET lookup operation, the valid entries of the two corresponding NBET entries are latched in pre-activation registers. Figure 4 presents the pre-activation circuit. It decodes the contents of the two pre-activation registers to generate the pre-activation signals to power mode controller of drowsy BTB.

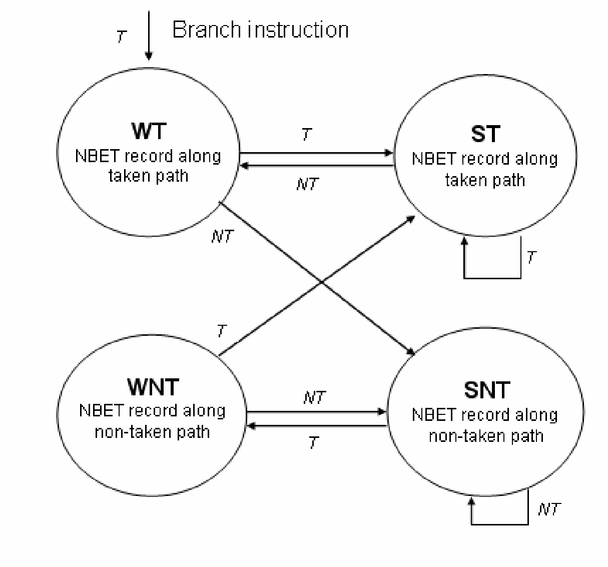
3.2: One-direction pre-activation

3.2.1: Why one-direction pre-activation. There are two drawbacks for two-direction pre-activation. The first one is that there is a wasted filed at sometimes. For example, unconditional branch instruction is always taken and highly biased branch instruction is frequently taken or not-taken. Another drawback is that if we always pre-activate the next possibly accessed BTB locations along taken and not-taken paths of a branch, at least one pre-activated BTB entry is unnecessary.

If each NBET entry has only one filed to store the BTB location of next branch instruction, half of NBET size and pre-activation circuits can be saved and the above problems can be ignored. Therefore, the problem becomes how to record the most possible next BTB location in one-field NBET.

3.2.2: Design of one-direction pre-activation. Our design idea is let the field of NBET consists with next predicted branch direction. In other word, if the next predicted direction is taken, the BTB location of next branch instruction along taken path is recorded. Otherwise, that of not-taken path is recorded.

Figure 5 shows the state transition diagram of a typical 2-bit branch predictor. Initially, a taken branch is placed in BTB with weakly-taken (WT) state. The BTB location of next branch instruction along taken path is recorded in NBET. Then, in the following executions of the branch instruction, the NBET update is contents only when the next predicted direction is changed. In 2-bit branch predictor, only while the predictor state changes from WT to SNT or WNT to ST, the NBET is needed to be update.


T
Branch instruction
WT
NBET record along taken path
ST
NBET record along taken path
WNT
NBET record along non-taken path
SNT
NBET record along non-taken path
T
NT
NT
T
T
NT
T
NT
Figure 5: NBET recording with predictor state

3.2.3: Circuit modifications of one-direction based NBET

Figure 6 displays next BTB entry collection circuits for one-field NBET. A little modification is required in the control circuit of NBET write enable signal. The DIR is now different form that of two-direction based NBET. It records if the previous branch instruction changed its next predicted direction here. The write enable signal of NBET is set only while the previous branch instruction changed its next predicted direction.

The modifications for NBET lookup and pre-activation circuit are trivial. Therefore, we ignore the description here.

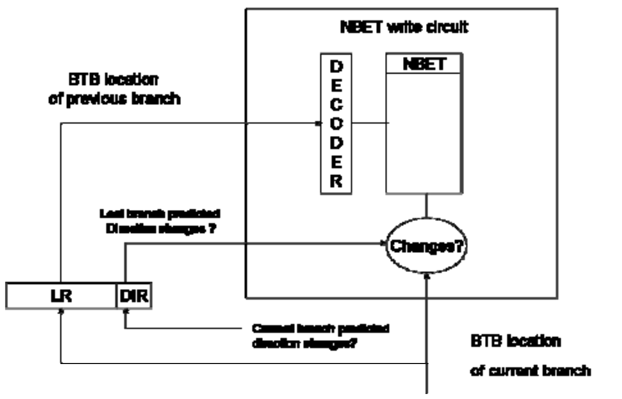

NBET write circuit
DECODER
NBET
BTB location of previous branch
Changes?
LR
DIR
BTB location of current branch

Figure 6: Next BTB location collection circuits for one-field NBET

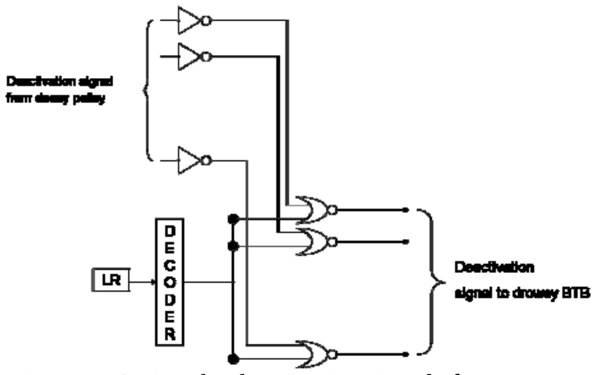

Deactivation signal from drowsy policy
LR
DECODER
Deactivation signal to drowsy BTB

Figure 7: Gating the deactivation signals for most recently accessed BTB and NBET entry

3.3: BTB entry deactivation

We adopt decay strategy proposed in [5] to deactivate BTB entries. In this method, a BTB entry is put into drowsy mode if this entry has not been accessed for a period of time (decay interval). For the implementation of decay idea, a global counter and a set of local counters are required. The global counter reset itself after a period of time (global interval). The local counter adopted for each BTB entries resets itself while the corresponding BTB entry is accessed and increments itself at each time that the global interval is reached. If any local counter reaches its maximum value, the corresponding BTB and NBET entry is put into drowsy mode. Note that the power modes of NBET are managed together to further save the NBET power overhead.

With accurate pre-activation, the decay interval becomes only about hundreds of cycles, since the energy overhead due to power mode changes is very small. Unfortunately, while program execution flow enters into a large basic block, the previous accessed NBET entry may be deactivated before the next BTB location updating. Therefore, the previous accessed NBET entry should prevent to be deactivated. Figure 7 shows its implementation circuits. This circuit the gates deactivation signals for most recently accessed BTB and NBET entry.

4: Experiments

4.1: Simulator and Benchmarks

The architectural simulator used in this research is Simplescalar/Alpha 3.0 [9]. It is a commonly used execution-driven simulator in architecture design domain. Table 1 is the default processor configuration in our experiment. We use SPEC2000 as our evaluation benchmark.

Parameter	Value
Inst. Window	16-RUU, 8-LSQ
Issue Width	4 instructions per cycle
Function Units	4 IntALUs 4 FPALUs
L1 I-cache	16KB, 2-way, 32B block
L1 D-cache	16KB, 2-way, 32B block
I-TLB	32 entries, fully assoc
D-TLB	32 entries, fully assoc
BTB	512-entries, 4-way
Direction Predictor	Bimodal predictor build in BTB

Table 1: Processor parameters

4.2: Evaluation Metrics

The following metrics are used to evaluate our design:

- BTB leakage energy consumption
- Performance loss

The performance loss is defined as the extra execution cycles of the system that adopts drowsy BTB over the execution cycles of the system that adopt regular BTB. The BTB leakage energy consumptions may include the extra energy caused by additional hardware and performance loss. Therefore, it composed of the following terms:

- BTB leakage energy
- NBET energy
- System leakage energy duo to performance loss
- Energy of extra control logics

Parameter	Value
Active leakage energy per bit	0.001289pJ/cycle
Drowsy leakage energy per bit	0.0001934pJ/cycle
Transition energy per bit	0.043pJ

Table 2: Leakage energy parameters

Most of the energy numbers are obtained from the power libraries in XTREM [10] tool set. The SRAM energy parameters of different power modes and the mode transition are listed in table 2 [11]. The number of execution cycles is obtained from Simplescalar/Alpha 3.0.

4.3: Experiment results

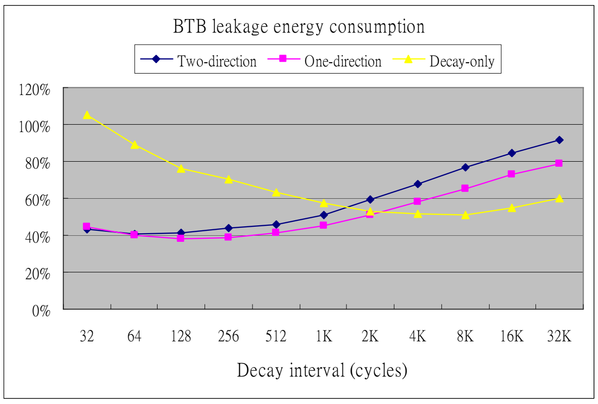


Figure 8: BTB leakage energy consumption with different decay interval

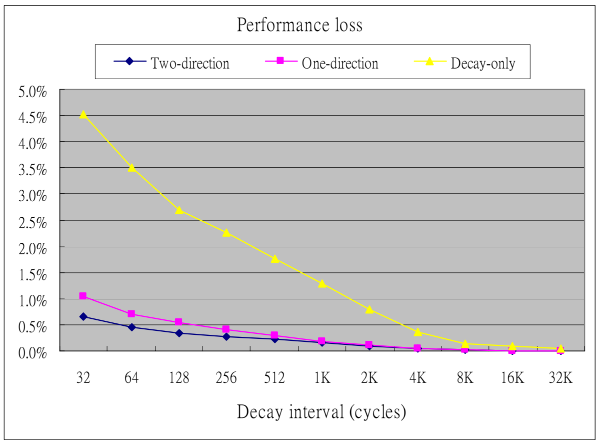


Figure 9: Performance loss with different decay interval

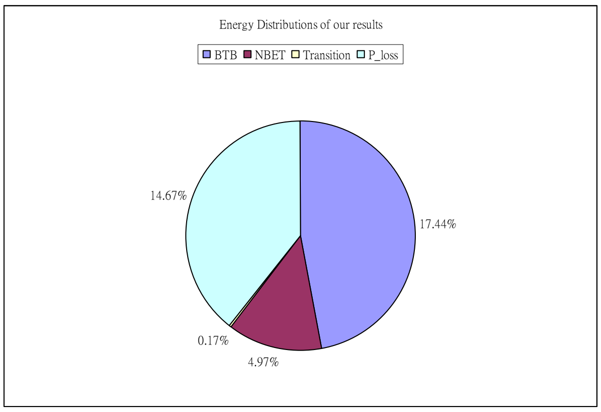


Figure 10: Energy distributions of the proposed design

Figure 8 and Figure 9 respectively present drowsy BTB leakage energy consumption and performance loss of various power mode managements. The leakage energy consumption of drowsy BTB is normalized the energy values to that of regular BTB. From these results, we can summarize the following findings:

- The most energy efficient decay intervals of various power mode managements are 64 for two-direction pre-activation, 128 for one-direction pre-activation, and 8K for decay only.
- Our pre-activation methods benefits for BTB leakage energy reduction and performance loss, since accurate pre-activation helps shorten the long decay intervals and hiding the wake-up latency.
- The BTB energy consumptions of one-direction pre-activation (38.1%) are less then that of two-direction pre-activation (40.7%) due to the reasons of half energy overhead.
- We suggest one-direction pre-activation, since it yields lowest BTB leakage consumptions (38.1%) with only 0.5% performance loss.

Figure 10 displays the energy distributions of the proposed design (one-direction pre-activation with 128-cycles decay interval). We further partitioned the energy numbers according to the energy sources including BTB leakage energy (BTB), NBET energy (NBET), transition energy due to power mode changes (Transition), and chip-wide energy due to performance loss (P_loss). From these results, we found that the main energy overhead of this design comes from additional chip-wide energy due to performance loss. This implies that the performance loss due to power mode management of BTB has large impact of energy, since BTB dissipates about 10% of total chip energy.

5: Conclusion

In this paper, we propose a next entry pre-activation method cooperated with the decay policy to manage power modes of each BTB entries. For the decay policy, a BTB entry which has not been accessed for a period of time is put into drowsy mode. For next entry pre-activation, the possible BTB next entries are cached for next BTB entry pre-activation. Unlike the decay management, our method helps to shorten the decay interval and hide wake-up latency, and both of these benefit leakage reduction. Simulation result shows that our method reduces the leakage consumption in BTB by an average of 61.9% with only paying 0.5% performance cost.

Several related research directions worth further studying. For example, power mode managements for instruction caches, data caches, and L2 caches. The management policies are designed according to the different access patterns, and we have already going on these works now.

REFERENCES

[1] NS Kim, T. Austin, D. Blaauw, T. Mudge, K.Flautner, JS Hu, Mj Irwin, M. Kandemir, and V. Narayanan, “Leakage Current: Moore’s Law Meets Static Power”, IEEE Computer Society, Volume 36, Issue 12, pages 68-75, December 2003.

[2] D. Parikh, K. Skadron, Y. Zhang, and M. Stan. “Power-aware Branch Prediction: Characterization and Design”, IEEE Transactions on Computers, Volume 53, Issue 2, pages 168-186, February 2004.

[3] K. Skadron, T. Abdelzaher, and M. R. Stan. “Control-theoretic Technique and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management”, In

Proceedings of the 8th International Symposium on High-Performance Computer Architecture, pages 17-28, February 2002.

[4] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge. “Drowsy caches: Simple techniques for Reducing Leakage Power”, In Proceeding of the 29th International Symposium on Computer Architecture, pages 148-157, May 2002.

[5] Z. Hu, P. Juang, K. Skadron, D. Clark and M. Nartonosi “Applying Decay Strategies to Branch Predictors for Leakage Energy Savings”, In Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors, pages 442-445, September 2002.

[6] M. D. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, “Gated Vdd: A Circuit Technique to Reduce Leakage in Cache Memories”, In Proceedings of the International Symposium on Low Power Electronics and Design, pages 90-95, July 2000.

[7] N. Kim, K. Flautner, D. Blaauw, and T. Mudge, “Circuit and Microarchitectural Techniques for Reducing Cache Leakage Power”, IEEE Transactions on Very Large Scale Integration Systems, Volume 12, Issue 2, pages 167-184, February 2004.

[8] Wei Zhang, Bramha Allu “Loop-based Leakage Control for Branch Predictors”, In Proceedings of the 2004 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, pages 149-155, September 2004.

[9] D.C. Burger and T. M. Austin. “The Simplescalar Tool Set, Version 2.0”, ACM SIGARCH Computer Architecture News, Volume 25, Issue 3, pages 13-25, June 1997.

[10] Gilberto Contreras, Margaret Martonosi, Jinzhan Peng, Roy Ju, and Guei-Yuan Lueh “XTREM: A Power Similator for the Intel XScale ® Core”, In Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems, pages 115-125, June 2004.

[11] W. Zhang, M. Karakoy, M. Kandemir, G. Chen “Reducing Data Cache Leakage Energy Using a Compiler-based Approach”, ACM Transactions on Embedded Computing Systems, Volume 4, Issue 3, pages 652-678, August 2005.