PHASED DROWSY I-CACHE WITH ON-DEMAND WAKEUP PREDICTION POLICY FOR HIGH-PERFORMANCE LOW-ENERGY MICROPROCESSORS

Zhou Hongwei, Zhang Chengyi and Zhang Minxuan
College of Computer Science
National University of Defense Technology
Changsha, China, 410073
E-mail: hongw.zhou@gmail.com, {chengyizhang, mxzhang}@nudt.edu.cn

KEYWORDS
Drowsy cache, Phased cache, Low energy, Instruction cache, On demand, Wakeup prediction.

ABSTRACT

In this paper, we propose a phased drowsy instruction cache with on-demand wakeup prediction policy (called "phased on-demand policy") to reduce the leakage and dynamic energy with less performance overhead. As in prior non-phased on-demand policy, an extra stage for wakeup is inserted before the fetch stage in pipeline. The drowsy cache lines are woken up in wakeup stage and the wakeup latency is overlapped with the fetch latency. Unlike in non-phased on-demand policy, The tag and data array are accessed in two phases. The tag blocks are in active mode all the time and are accessed in wakeup stage. The data blocks are in drowsy mode except when they are accessed in fetch stage. The optimum trade-off point is tried to be reached between the increment of energy caused by always active tag array in wakeup stage and reduction of energy profitted from perfect way prediction. Experiments on 9 SPCE2000 benchmarks show that, compared with prior non-phased on-demand policy, our proposed policy can save 75.4% of energy for I-Cache and improve the EDP of whole processor by 6.9%. The performance overhead is only 0.42% on average.

INTRODUCTION

Energy consumption has become the main restriction on microprocessor design because of the higher density and higher frequency. For modern microprocessors, the large capability caches are integrated in chip to improve the processors' performance. For instance, 60% of the StrongARM and 30% of Alpha 21264 are devoted to cache and memory structures (Gowan et al. 1998; Manne et al. 1998). They comprise a large portion of chip area and produce large energy. Instruction cache (I-Cache) affects total energy consumption particularly due to their high access frequency. For example, ARM920T microprocessor dissipates 25% of its total power in the I-Cache (Segars S. 2001). The energy in I-Cache consists of leakage energy and dynamic energy. According the prediction from the International Technology Roadmap for Semiconductor (SIA, 2004), the leakage energy may constitute as much as 50% of total energy by the 70nm technology. To save the energy of I-Cache at most, the leakage energy and dynamic energy in I-Cache should be reduced at the same time.

To reduce the leakage energy of I-Cache, the cache lines that are not been used recently will be put into a low-energy mode (called "drowsy" mode) and are re-woken up when accessed again. Data is not lost when the cache line is in drowsy mode. This technique is called drowsy cache (Powell et al., 2000; Kaxiras et al. 2001; Flautner et al. 2002; Chengyi Zhang et al. 2006; Kim et al. 2004a; Kim et al. 2004b; Li et al. 2004). In the drowsy mode, the supply voltage is lower than that in normal mode (called "active" mode) and data can be retained. When the drowsy cache line is accessed, the supply voltage must be recovered first. There is a wakeup penalty to restore the voltage level from the drowsy mode into the active mode. A simple policy for drowsy cache is noaccess policy in which the per-line access history is used and all the unused lines are put into drowsy mode periodically (Flautner et al. 2002). The energy saved depends on the ratio of the number of the cache lines in drowsy mode to the number of all cache lines (called "turn-off ratio"). The higher turn-off ratio is, the more leakage energy is saved.

To reduce the performance overhead caused by extra wakeup latency, the next cache line should be woken up before it is accessed. The performance overhead depends on how accurately the next cache line can be predicted and woken up. Zhang Chengyi etc. proposed a PDSR (periodically Drowsy Speculatively Recover) policy based on noaccess policy (Chengyi Zhang et al. 2006). In PDSR, a pre-wakeup mechanism is used for pre-waking up all cache lines in next sequential set when current set is being accessed. Nam Sung Kim proposed a noaccess-JITA policy in which the way predictor is also used for predicting which way may be hit in next sequential set (Kim et al. 2004a). Only the predicted cache line needs to be pre-woken up to increase turn-off ratio. The accuracy of wakeup prediction in these two policies is restricted by taken branch instruction because branch information is not used for wakeup prediction in these policies.

Sung Woo Chung etc. proposed a non-phased drowsy I-Cache with on-demand wakeup prediction policy (called "non-phased on-demand policy") in which an

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

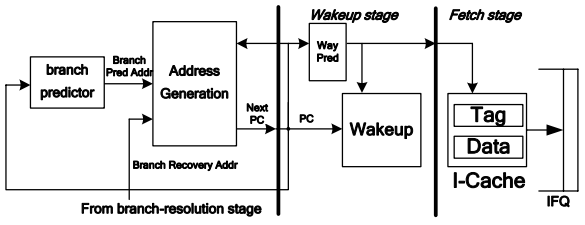
extra wakeup stage is inserted between the branch prediction and the fetch stage in pipeline (Sung Woo Chung and Kevin Skadron. 2006). The branch predictor is also used as a wakeup predictor for more accurate wakeup prediction. Extra stage hides the wakeup penalty, not affecting branch prediction accuracy. Way predictor is used for reducing the lines being woken up. All cache lines except the next expected cache line are in drowsy mode. The tag array and data array are accessed in fetch stage. The non-phased on-demand policy is near optimal policy for I-Cache leakage reduction, but it has some disadvantages yet.

In this paper, a phased drowsy instruction cache with on-demand wakeup prediction policy (called "phased on-demand policy") is proposed. In our proposed policy, the tag array is in active mode all the time and the access to tag array is moved from fetch stage to wakeup stage and the access to data array is in fetch stage as before. The cache line is accessed in two phases. The result of tag comparison is acquired one cycle earlier than non-phased on-demand policy and only the data block in matching way is necessary to be accessed. Way predictor is unnecessary. No latency caused by incorrect way prediction is incurred.

The rest of this paper is organized as follows. We first analyze the disadvantages in prior non-phased on-demand policy. Then we propose an improved phased on-demand policy. Subsequently, we introduce the performance and energy evaluation model, simulation environment and analyze the simulation results. At last, we make the conclusion and give the future work.

DISADVANTAGES IN PRIOR NON-PHASED ON-DEMAND POLICY

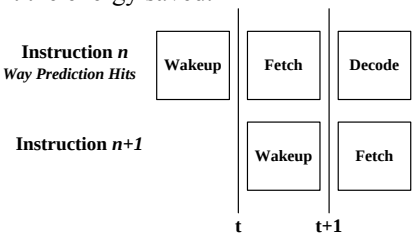
As shown in Figure 1, in non-phased on-demand policy, an extra wakeup stage for wakeup is inserted between the branch prediction and the fetch stage. Branch prediction information is also used for wakeup prediction. All cache lines are in the drowsy mode except the cache line being accessed. After the fetch-address is generated, the cache line indexed by fetch-address is woken up in wakeup stage. In fetch stage, the tag and data array of this woken cache line are both active and can be accessed within one cycle if wakeup prediction is correct.



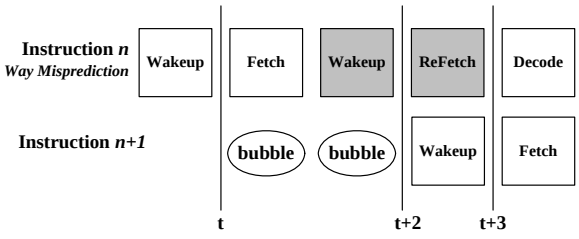
Figures 1: Front-end pipeline in Non-Phased on-demand policy

The non-phased on-demand policy is near optimal policy for reducing leakage energy of I-Cache, but two disadvantages exist and should be improved. One disadvantage is that a two-stage penalty is incurred by incorrect way prediction. This is the main reason for loss of performance.

As shown in Figure 2a, if way prediction hits, the wakeup penalty is overlapped with fetch penalty. No bubble is generated in pipeline. As shown in Figure 2b, if way prediction misses, two bubbles are generated in pipeline. One extra cycle is needed first to wake up all other cache lines in current set being accessed. Another extra cycle is necessary to access these cache lines to acquire the desired data in matching way. The performance is degraded by incorrect way prediction and additional energy for increased execution time may discount the energy saved.



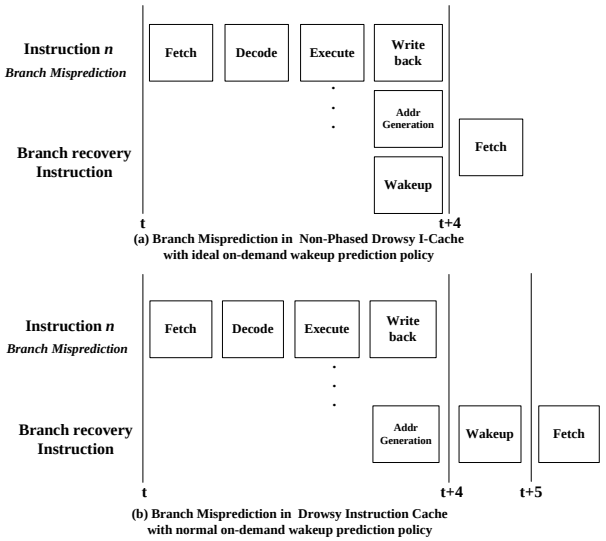
(a) Way Prediction Hits in Non-Phased Drowsy Instruction Cache with on-demand wakeup prediction policy



(b) Way Prediction Misses in Non-Phased Drowsy Instruction Cache with on-demand wakeup prediction policy

Figures 2: Pipeline comparison when Way Predictor hits and misses

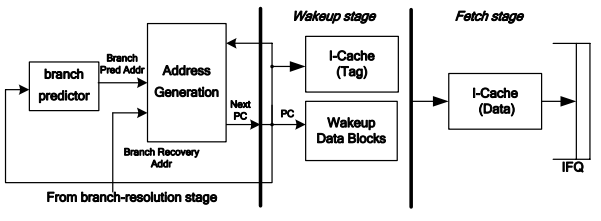
The other disadvantage is that the prior non-phased on-demand policy is ideal for recovery of branch mis-prediction. Assuming the penalty of branch mis-prediction is two cycles in conventional drowsy I-Cache. As shown in Figure 3a, in ideal non-phased on-demand policy, the address generation, branch prediction and wakeup are assumed to operate at the same time after execute/branch-resolution stage of the instruction n. So no extra wakeup penalty is incurred. The branch recovery instruction is fetched at t+4. In this ideal case, more careful adjustments are needed in pipeline design. In fact, one extra wakeup penalty will be incurred in normal non-phased on-demand policy when branch is mis-predicted, as shown in Figure 3b. The wakeup is operated after address generation. The branch recovery instruction is fetched at t+5. In this case, extra wakeup latency is incurred when branch is mis-predicted.



Figures 3: Pipeline comparison when Branch Mis-prediction

IMPROVED PHASED ON-DEMAND POLICY

As shown in Figure 4, we propose a phased drowsy instruction cache with on-demand wakeup prediction policy. The access to tag array of I-Cache is moved from fetch stage to wakeup stage. The tag and data array of I-Cache are accessed in two phases. The tag array is in the active mode all the time and only the data array can be put into drowsy mode. In each cache access, all tag blocks in current set being accessed are accessed at the same time in wakeup stage to ascertain the matching way. Then only one data block in matching way is accessed according to the result of tag comparison in fetch stage. It is the real result of tag comparison not the result of way prediction that is used to tell which way would be accessed in data array, so we can consider that the way prediction is perfect in our policy. In fact the way predictor is not used any more. Since no way prediction information can be used, so all data blocks in the set which will be accessed in fetch stage should be pre-woken up in wakeup stage with the tag comparison at the same time.



Figures 4: Phased Drowsy Instruction Cache with on-demand wakeup prediction policy

For instance, in a 4-way set-associative I-Cache with our improved policy, all four data blocks in current set being accessed are pre-woken up in advance. But in I-Cache with prior on-demand policy, only one data block is necessary to be pre-woken up if way prediction hits. So, a little more leakage energy would be consumed in data array because of high way prediction hit ratio. In addition, all the tag blocks are active at any time in our improved policy. Though the leakage energy in tag array is not saved in our proposed policy, in modern high-performance microprocessor, the number of bits in a data block is far larger than that in a tag block, so the leakage energy of tag array has less influence on the total energy of I-Cache. For example: if the data block size is 16byte and the tag address bits are 20, then the energy consumed by tag array is about 0.16 times of the energy consumed by data array.

In I-Cache with our improved policy, four tag blocks and one data block are accessed in each cache access if cache hits. No data block is accessed if cache misses. In I-Cache with prior policy, the number of tag and data blocks accessed in each cache access is determined by result of way prediction. If way prediction hits, only one tag and one data block are accessed. If way prediction misses, additional three tag and three blocks are accessed, too. Since the way prediction hit ratio is higer in prior on-demand policy, so more dynamic energy is consumed in I-Cache with our improved policy.

Comparing the prior policy, our proposed policy has less performance overhead because no extra wakeup latency is incurred by way mis-prediction. Perfect way prediction can reduce the performance overhead and save the extra energy caused by increasing execution time. That is, we use the reduction of energy profitted from perfect way prediction to balance the increment energy caused by always active tag array in wakeup stage and try to find the optimum trade-off point between them.

EVALUATION METHODOLOGY

For optimum architecture design in high-performance low-energy I-Cache, not only the leakage energy but also the dynamic energy should be considered at the same time. The additional energy in whole processor caused by increased execution time should be considered, too. This section describes the energy and performance evaluation model to evaluate our proposed policy and other policies. The simulation environment is also introduced.

Energy and Performance Evaluation Model

The base model is a conventional I-Cache without any energy controlling policy. IPC' and IPC are the number of instructions committed per-cycle with and without energy controlling policy. The program execution time is T′ or T when energy controlling policy is used or not. The normalized execution time to base model is s and can be calculated as equation (1).

$$s = T'/T = IPC/IPC' \quad (1)$$

Assuming the average proportion between energy of I-Cache (E_{icache}) and energy of whole processor (E) is α. The energy of processor except I-Cache is E_{else}. The ratio of energy in tag array (E_{icache_tag}) to data array (E_{icache_data}) is m and it can be represents proximately as the ratio of the bit width in tag array to data array. So, $E_{icache}/E=\alpha$ and $E_{icache\_tag}/E_{icache\_data}=m$.
E_d and E_s are the baseline dynamic energy and leakage energy respectively, and E'_d and E'_s are the corresponding dynamic and leakage energy with energy controlling policy.

Normalized leakage energy of I-Cache.
The e_{active} and e_{standy} represent the leakage energy of one bit memory unit during one cycle in active mode and in drowsy mode respectively. Assuming e_{standy} is q times of e_{active}, so $e_{standy}=qe_{active}$. The proportion between the number of drowsy cache lines and the number of all cache lines is $R_{turnoff}$. N_{data} and N_{tag} are the number of bits in all tag array and data array in I-Cache. In data array of I-Cache without energy controlling policy, the leakage energy is $e_{active}\ N_{data}T$. In our proposed policy, the leakage energy of drowsy data blocks is $e_{drowsy}R_{turnoff}N_{data}T'$ and the leakage energy of active data blocks is $e_{active}(1-R_{turnoff})N_{data}T'$. Equation (2) shows the normalized leakage energy to base model in data array of I-Cache.

$$E'_{s\_icache\_data}/ E_{s\_icache\_data}= [e_{active} (1-R_{turnoff}) N_{data}+e_{drowsy}R_{turnoff}N_{data}]T'/ (e_{active} N_{data}T) =[(1-R_{turnoff})+ R_{turnoff}q]s \quad (2)$$

Tag array of I-Cache is active at all the time in our proposed policy. The leakage energy of active tag blocks is increased due to longer execution time, thus, $E'_{s\_icache\_tag}=sE_{s\_icache\_tag}$. The energy caused by mode switching between the drowsy and active mode is negligible. The normalized leakage energy to base model in I-Cache is μ and calculated as shown in equation (3).

$$\mu= E'_{s\_icache} / E_{s\_icache} =(E'_{s\_icache\_tag} +E'_{s\_icache\_data})/(E_{s\_icache\_tag} +E_{s\_icache\_data}) = [m+(1-R_{turnoff})+ R_{turnoff}q]s/(m+1) \quad (3)$$

Normalized dynamic energy of I-Cache.
N_{assoc} represents the cache associativity, WPHR represents the way prediction hit ratio and CHR represents the cache hit ratio. In our proposed policy, all tag blocks in current accessed set are accessed at the same time. Only the data block in matching way is accessed when cache hits, no data block is necessary to be accessed when cache misses. The normalized dynamic energy to base model in I-Cache is ν and can be calculated as equation (4) shows.

$$\nu = E'_{d\_icache} / E_{d\_icache} =(E'_{d\_icache\_tag} +E'_{d\_icache\_data})/(E_{d\_icache\_tag} +E_{d\_icache\_data}) = (N_{assoc}\ m+CHR)/ [N_{assoc} (m+1)] \quad (4)$$

Normalized energy of else compenents in processor.
The energy of else compenernts in processor is not optimized by the energy controlling policy. In contrast, it will be increased for longer execution time. We assume that it increases in direct proportion to the execution time approximately. that is $E'_{else}=sE_{else}$.

Normalized energy of whole processor.
Not only the leakage energy but also the dynamic energy in I-Cache is considered at the same time. Assuming the ratio of leakage energy in I-Cache to leakage energy in whole processor is α_1, the ratio of dynamic energy in I-Cache to dynamic energy in whole processor is α_2 and the ratio of dynamic energy of processor to total energy in processor is β. The ratio of dynamic energy to leakage energy in I-Cache is n and can be calculated as equation (5).

$$n =E_{d\_icache}/E_{s\_icache} =(\alpha_2\beta)/[\alpha_1(1-\beta)] \quad (5)$$

So, the normalized energy of I-Cache (γ) can be calculated as equation (6) shows. The normalized energy of whole processor is calculated as equation (7) shows and it is represented as η. Equation (8) shows the normalized of EDP to base model in whole processor.

$$\gamma = (E'_{d\_icache}+E'_{s\_icache})/(E_{d\_icache} +E_{s\_icache}) = (\nu n+\mu)/ (n+1) \quad (6)$$

$$\eta = (E'_{icache}/E_{icache})\ \alpha+(E'_{else}/E_{else})(1-\alpha) =\gamma\alpha+s(1-\alpha) \quad (7)$$

$$EDP'/ EDP = (E'/E)(IPC/IPC') =\eta s \quad (8)$$

Simulation Environment

We use Hotleakage simulator (Zhang et al. 2003) to get the value of the basic parameters (IPC, IPC′, $R_{turnoff}$, and CHR) required by our evaluation modle for evaluating energy and performance. The processor parameters model a high-performance microprocessor similar to Alpha 21264 (Gowan et al. 1998) as shown in Table 1. The energy parameters are based on the 70nm/0.9V technology. Benchmarks are chosen from SPEC CPU2000. Each benchmark is first fast-forwarded a billion instructions and then simulated the 300 millions instructions. We compare our phased on-demand policy with three policies: noaccess-JITA policy, normal non-phased on-demand policy and ideal non-phased on-demand policy. These policies are described in Table2.
Because the average proportion between energy of I-Cache and energy of whole processor is different among different processors, we assume it is a moderate value (α=10%) first. We will also research other cases with different α. The parameter m and q can be calculated according to the processor parameters and energy parameters: m=0.14, q=0.04. According to the prediction from ITRS (SIA. 2004), the leakage energy will be equal to dynamic energy in microprocessors when the process technology is 70nm, so we assumes the β=0.5. According to the estimation from Hotleakage

simulator, the α_1 is 3% and α_2 is 4.3% approximately in our processor parameters model. So, n is 1.43 by equation (5). Because the increased execution time may incur extra energy, so we assume that the decay interval is 32K cycles in noaccess-JITA policy for less performance overhead.

Table 1: Architecture/Energy parameters

Processor parameters	
Instruction Window	8 IFQ, 80 RUU, 40 LSQ
Fetch/Decode/Issue/Commit width	4 instructions/cycle
L1 I-Cache	64KB, 4-way, 32B block, 1 cycle latency
Branch Predictor type	comb
Branch mis-prediction latency	2 cycles
Way prediction policy	MRU
The switch latency between different mode	1 cycle latency
Energy parameters	
Process Technology	70nm
Temperature	353K
Supply Voltage	0.9V in active mode; 0.3V in drowsy mode
Threshold Voltage	NMOS:0.1902V; PMOS: 0.2130V
Leakage energy of I-Cache in drowsy mode	0.019142 J
Leakage energy of I-Cache in active mode	0.441827 J

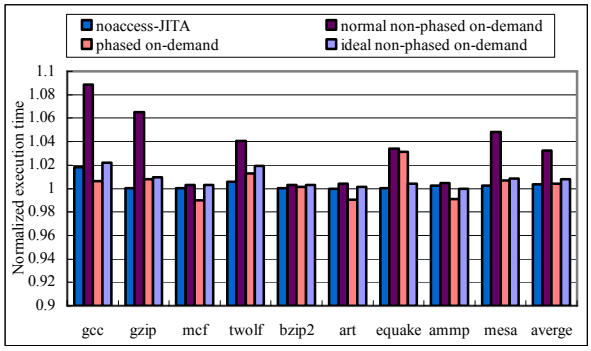
Table 2: Description for different policies

Policy	Description
noaccess-JITA	The decay interval is 32K cycles. Real way predictor is used. Branch prediction information is not used for wakeup prediction.
normal non-phased on-demand	One-cycle extra wakeup latency is incurred when branch is mis-predicted and real way predictor is used. The tag array and data array are accessed in one stage.
Ideal non-phased on-demand	No extra wakeup latency is incurred when branch is mis-predicted and a perfect way predictor is used. The tag array and data array are accessed in one stage.
Phased on-demand	One-cycle extra wakeup latency is incurred when branch is mis-predicted and way predictor is not used. The tag array and data array are accessed in two phases.

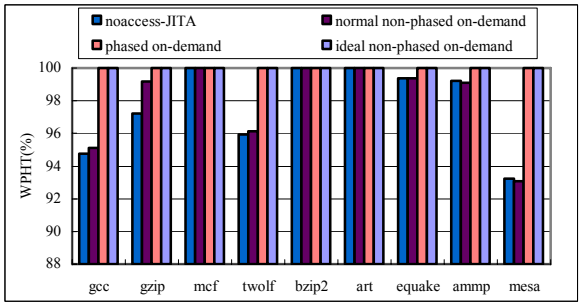
Simulation Results

As shown in Figure 5, our proposed phased on-demand policy has less performance overhead than other on-demand policies and the performance overhead is only 0.42% on average. The normalized execution time in normal non-phased on-demand policy is increased by 3.2% to base model on average. The main reason is that extra two-cycle latency is incurred by incorrect way prediction and extra one-cycle wakeup latency is incurred by incorrect branch prediction. When prefect way predictor is used and extra one-cycle wakeup latency caused by incorrect branch prediction is avoided, the performance overhead can be reduced to 0.79% as in ideal non-phased on-demand policy. The performance overhead in noaccess-JITA policy is 0.33% and it is smallest in all policies. The performance overhead in phased on-demand policy is only a little smaller than ideal non-phased on-demand policy.

The way prediction hit ratio is shown in Figure 6. Since no way predictor is used in our proposed policy, the WPHR is considered as 100% just like in ideal non-phased on-demand policy. The WPHR is 93.1% and 93.3% on average in normal non-phased on-demand policy and in noaccess-JITA policy respectively. The execution time is influenced by WPHR more obviously in normal non-phased on-demand policy. For instance, the WPHR in gcc, gzip, twolf and mesa is far lower than that in other programs, so the normalized execution time in these programs is larger than that in others. The execution time is almost not influenced by WPHR in noaccess-JITA policy.

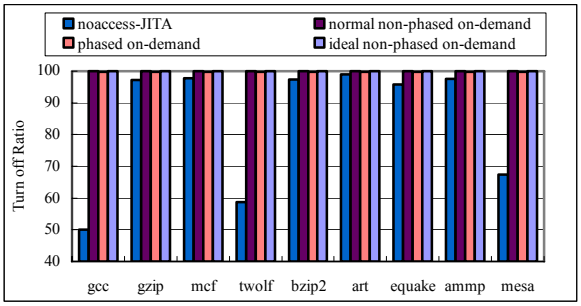


Figures 5: Normalized execution time



Figures 6: Way prediction hit Ratio

As shown in Figure 7, the turn-off ratios in all on-demand policies except noaccess-JITA policy are almost the same. In normal non-phased on-demand policy and ideal non-phased on-demand policy, the turn-off ratio is 99.95% on average and near the optimum turn-off ratio in drowsy cache. In our proposed policy, the turn-off ratio is 99.8% on average and it is only a little smaller than optimum value. In noaccess-JITA policy, the cache lines are turned off only if they are not accessed within a decay interval, so the turn-off ratio in this policy is far smaller than in other policies. It is only 67.35% on average.

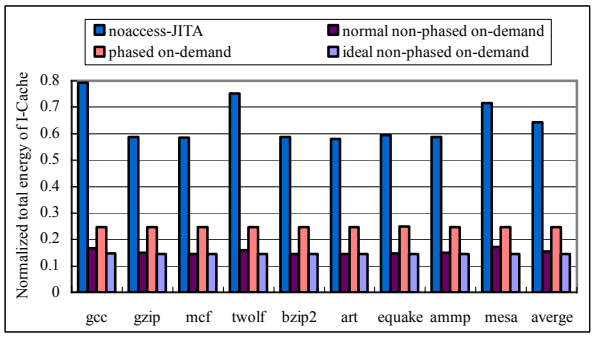


Figures 7: Turn-off ratio in drowsy I-Cache

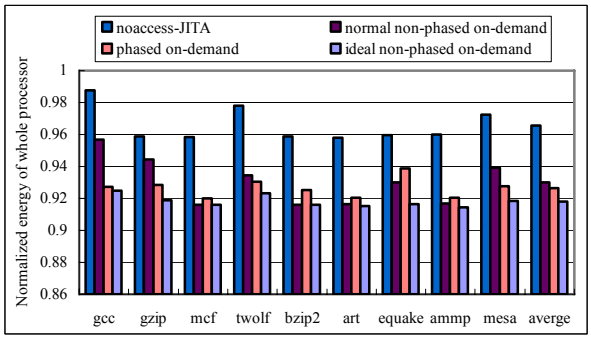
We estimate the normalized total energy of I-Cache according to our proposed energy evaluation model. As shown in Figure 8, in normal non-phased on-demand policy, the normalized total energy of I-Cache is 15.4% on average. It is very near the lowest value 14.6% in ideal non-phased on-demand policy. In our proposed policy, the normalized total energy of I-Cache is 24.6% on average. Compared with other on-demand policies, phased on-demand policy is not very good for reducing the total energy of I-Cache. However, it is better than noaccess-JITA policy. The normalized total energy of I-Cache in noaccess-JITA is 64.3% on average.

The normalized energy of whole processor is also calculated according to our proposed policy and shown in Figure 9. In our proposed policy, the normalized energy of whole processor is 92.7% on average. In ideal non-phased on-demand policy, the normalized energy of whole processor is 91.8% on average. Our proposed policy is very near the ideal non-phased on-demand policy for reducing the energy of whole processor. In normal non-phased on-demand policy, the energy saved in whole processor is less than that in our proposed policy and the normalized energy of whole processor is 93% on average. The normalized energy of whole processor in noaceess-JITA policy is 96.6% on average. It is the worst policy for reducing energy of whole processor.

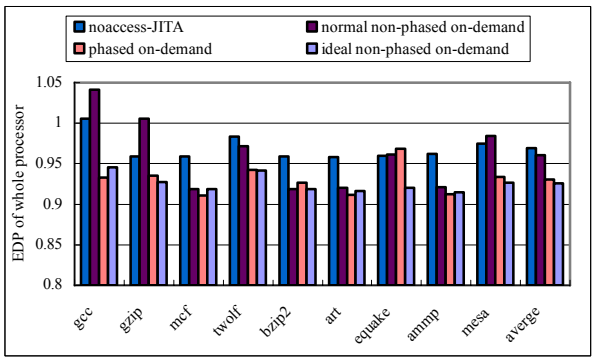
Figure 10 shows the normalized EDP of whole processor. In our proposed policy, the EDP is improved by 6.9% on average and only a little smaller than 7.4% in ideal non-phased on-demand policy. In normal non-phased on-demand policy, the EDP is only improved by 4% due to obvious performance overhead. In noaccess-JITA policy, the EDP can be improved by 3.1% on average. So, except the ideal non-phased on-demand policy, the phased on-demand policy is the best policy for improving the EDP of whole processor.



Figures 8: Normalized total energy of I-Cache



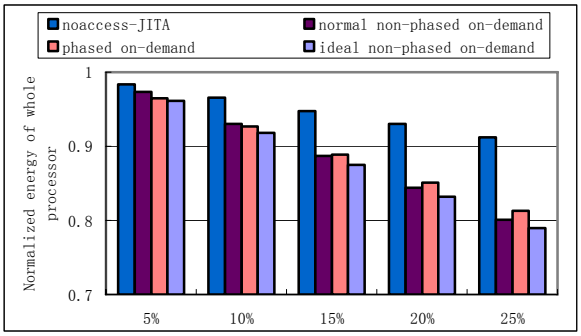
Figures 9: Normalized energy of whole processor



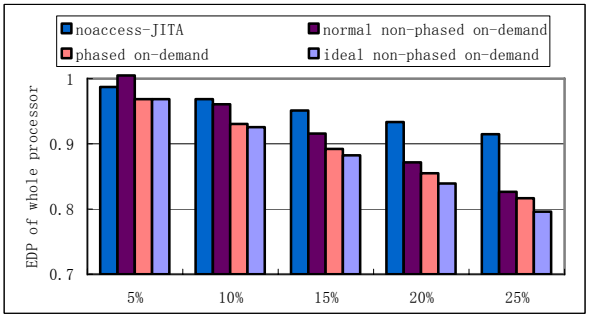
Figures 10: Normalized EDP of whole processor

As shown in Figure 11 and Figure 12, if we change the proportion of I-Cache's energy to whole processor's energy from 5% to 25% (α is increased from 5% to 25%), the energy saving and the EDP in whole processor are both improved in all policies with the increasing α. As shown in Figure 11, except the ideal non-phased on-demand policy, our proposed policy is the best policy for reducing the energy of whole processor when α is smaller than 15%. The normal non-phased on-demand policy is the best policy when α is larger than 15%. That is beause the energy saved in I-

Cache is more pivotal than the energy increased by increased execution time when the proportion of I-Cache's energy to whole processor's energy is larger. Since the performance overhead is not very obvious in I-Cache with on-demand policies, so the EDP is improved in all on-demand policies with different α. As shown in Figure 12, in all cases, the phased on-demand policy is near-ideal policy for improving the EDP in whole processor.



Figures 11: Normalized energy of whole processor



Figures 12: Normalized EDP of whole processor

CONCLUSIONS

This work presents a phased drowsy I-Cache with on-demand wakeup prediction policy. The access to tag array is moved from the fetch stage to the wakeup stage. Way predictor is not used any more. Though energy of I-Cache reduced in our proposed policy is less than that in normal non-phased on-demand policy, the performance overhead in our proposed policy is smaller and it is near the lowest performance overhead in ideal non-phased on-demand policy. In our proposed policy, the optimum trade-off point between the reduction of leakage energy in the tag array and perfect way prediction is reached. When the proportion of I-Cache's energy to whole processor's energy is 10%, with only 0.42% performance overhead, our proposed policy can reduce the energy of whole processor by 7.3% and improves the EDP by 6.9% on average. When the proportion of I-Cache's energy to whole processor's energy is increased from 5% to 25%, our proposed policy is the near-optimum policy in any cases for improving the EDP of whole processor.

REFERENCES

Chengyi Zhang, Hongwei Zhou, Minxuan Zhang, and Zuocheng Xing. 2006. "An architectural leakage power reduction method for instruction cache in ultra deep submicron microprocessors." In *the 11th Asia-Pacific Conference(ACSAC 2006),* 588-594.

Flautner K., N.S.Kim, S.Martin, D.Blaauw, and T.Mudge. 2002. "Drowsy Caches: Simple Techniques for Reducing Leakage Power." In *ISCA2002*, 147-157.

Gowan M.K., Biro L.L. and Jackson D.B. 1998. "Power Considerations in the Design of the Alpha 21264 Microprocessor." In *DAC'98*, Los Alamitos, California, U.S, 26-31.

Inoue, K., Ishihara, T., and Murakami, K. 1999. "Way-predicting Set-Associative Cache for High performance and Low Energy Consumption." In *Proc. Of 1999 International Symposium on low power Electronics and Design (ISLPED1999)*, 273-275.

Kaxiras S., Z. Hu and M. Martonosi. 2001. "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power." In *ISCA2001*, 240-251.

Kim N. S., K. Flautner, D. Blaauw and T. Mudge. 2004a. "Single-Vdd and Single-Vt Super-Drowsy Techniques for Low-Leakage High-Performance Instruction Caches." In *Proc. of Int. Symp. on Low Power Electronics and Design*, 54-57.

Kim N. S., K Flautner, D. Blaauw, and T. Mudge. 2004b. "Circuit and Microarchitectural Techniques for Reducing CacheLeakage Power." *IEEE Transaction on VLSI Systems* 12, No. 2 (Feb). 167-184.

Li Y., D. Parikh, Y. Zhang, K. Sankaranarayanan, M. Stan, and K. Skadron. 2004. "State-Preserving vs. Non-State-Preserving Leakage Control in Caches." In *Proc. of the Design Automation and Test in Europe Conference*. 22-27.

Manne S., A. Klauser, and D. Grunwald. 1998. "Pipeline Gating: Speculation Control for Energy Reduction." In *Proc. Of Int. Symp. on Computer Architecture*, 132-141.

Powell M. D. et al. 2000. "Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories." In *ISLPED2000*, 90-95.

Segars S. 2001. "Low Power Design Techniques for Microprocessors." *ISSCC Tutorial*.

SIA. International Technology Roadmap for Semiconductors, 2004.

Sung Woo Chung and Kevin Skadron. 2006. "Using branch prediction information for near-optimal I-Cache leakage." In *the 11th Asia-Pacific Conference(ACSAC 2006),* 24-37.

Zhang Y., D. Parikh, K. Sankaranarayanan, K. Skadron and M. R. Stan. 2003. "Hotleakage: An Architectural, Temperature-aware Model of Subthreshold and Gate Leakage." Tech. Report CS-2003-05, Department of Computer Sciences, University of Virginia, (Mar).

Zhou Hongwei received the B.S degrees in architecture of computer from National University of Defense Technology, changsha, Hunan, P.R.China, in 2003. Currently he is working toward the Ph.D degree at National University of Defense Technology.His main research interests are architectural level power optimization for Ultra-Deep submicron microprocessors and leakage current reduction in VLSI circuits. His e-mail address is : hongw.zhou@gmail.com