Stretching the Limit of Microarchitectural Level Leakage Control With Adaptive Light-Weight Vth Hopping

Hao Xu, Wen-Ben Jone and Ranga Vemuri
Dept. of ECE, University of Cincinnati
814 Rhodes Hall, Cincinnati, OH 45221-0030, USA
{xuho, ranga, wjone}@ececs.uc.edu

Abstract

Power gating (PG) and body biasing (BB) are popular leakage control techniques at microarchitectural level. However, their large overhead prevents them from being applied for active leakage reduction. The overhead problem is further magnified by temperature and process variation, leading to the "corner case leakage control" problem. This paper presents an Adaptive Light-Weight Vth Hopping technique. This technique dramatically reduces the overhead for mode transition, addresses the corner case leakage control problem, and thus enables active leakage control.

1. INTRODUCTION

MOSFET scaling into sub-100nm region has resulted in significant increase in leakage power consumption. Many leakage reduction techniques have been introduced and studied so far. They can be characterized into two classes: runtime techniques and design-time techniques. Runtime techniques, such as power gating, body biasing and input vector control [1], tune the circuit into a lower-leakage state during runtime, based on the circuit workload variation.

Runtime techniques turn out to be very effective, since most circuits have significant amount of idleness during runtime. There are two overheads associated with runtime techniques, namely energy overhead and delay overhead. We use two design parameters, energy breakeven time (EBT) and wakeup time (WUT), to quantify them. EBT is defined as the minimal time for a circuit to stay in low-leakage mode, such that energy saving catches up with the energy overhead for performing mode transition. WUT is defined as the minimal time required for a circuit to fully recover from the low-leakage mode to normal mode. Due to the EBT and WUT, leakage control techniques can only be applied when circuit idleness (T_{idle}) satisfies:

$$T_{idle} \geq EBT + WUT \quad (1)$$

We call this time as minimum idle time (MIT) [2]. As we will show in Section 2, the MIT of two popular leakage control techniques, power gating and body biasing, are very high. Despite of their large overhead, PG and BB are widely adopted because of their high leakage reduction ratio.

Modern CPUs have a large number of advanced FUs to accelerate performance, for example multiple ALUs, FPUs (floating point unit), LSU (load-store unit), BTB (branch target buffer). When different types of instructions are executed, some FUs are used, and some are not. This mutual exclusion creates idleness at MA level and thus opportunities to reduce FUs' leakage. Figure 1 demonstrates the ALU activities when running benchmark program of 64 points FFT [3] on a Alpha CPU. The dynamic energy consumption (red) of one calculation is averaged and normalized upon the leakage energy consumption (blue) of one clock cycle. It can be observed that even when the CPU is in active mode, significant idle leakage exists, demanding better solutions for more aggressive leakage control.

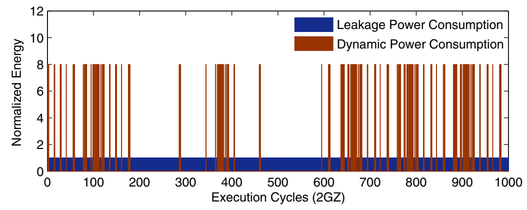


Figure 1. Dynamic Power VS. Idle Leakage on ALU 32nm 110°C

Research has been conducted to study leakage control at MA level for an active CPU [4, 5, 6, 7, 8]. Hu et al. [4] discussed using PG to turn off function units (FUs) when they are idle or branch mis-prediction occurs. [7] pointed out that idleness prediction is highly dependent on program regularity and large mis-prediction can lead to large energy penalty instead of saving. He et al. [6] emphasized the importance of designing leakage control system considering the interdependency of temperature and leakage. Kannan et al. [8] proposed the scheme of using leakage sensor to determine the runtime leakage of FUs, and turn off the FU with the highest leakage. Roy et al. [5] studied power gating control by inserting control instructions during compilation.

These studies have three common problems. First, they did not perform quantitative study. They were not able to determine the optimum design point, such as when to apply leakage control and what the granularity of control should be. Second, PG and BB are actually inefficient for MA level leakage control due to their large overhead, as will be shown in Section 2. Third, most previous studies did not consider the temperature and process variation (TV/PV) impact on leakage control. Kannan et al. [8] was the first one to introduce the consideration of TV/PV. However, their scheme simply chooses the FU with the highest leakage and applies PG on it. This scheme cannot guarantee the net energy saving by applying PG to be positive.

In this paper, we introduce an adaptive Light-Weight Vth Hopping (LW-VH) scheme to stretch the limit of MA level leakage control. Our study has the following contributions. First, we derive a systematic and quantitative approach for leakage control at MA level leakage using LW-VH. Second, we propose the usage of leakage sensor to calculate the runtime MIT, and a technique to dynamically adjust leakage control policy at runtime for TV/PV compensation. Finally, we introduce the scheme of instruction-queue monitoring to generate the leakage control signals.

2. LEAKAGE CONTROL TECHNIQUES

2.1 Power Gating, Body Biasing and Light-Weight V_{th} Hopping

The basic PG diagram (ground gating) is shown in Figure 2.a. Since footers are responsible for providing active currents for the target circuit, they should be designed in large size to minimize performance penalty. Buffers are needed to drive these large-scale footers. Denote the sum of both capacitance as C_F for convenience. For each mode transition of PG, large energy overhead (O_{PG}) is incurred for switching the footers and buffer capacitance:

$$O_{PG} = C_F V_{DD}^2 \tag{2}$$

EBT of PG can then be calculated by its definition:

$$B_{PG} = \frac{O_{PG}}{S_{PG/time}} \tag{3}$$

where $S_{PG/time}$ is the leakage energy saving per unit time by applying PG. For PG, although its $S_{PG/time}$ is high, its large C_F leads to large overhead (O_{PG}). Consequently, long EBT time (B_{PG}) is required to compensate the overhead. The typical value of C_F and B_{PG} is shown in Table 1.

The basic BB design is shown in Figure 2.b. For each mode transition of BB, energy overhead (O_{BB}) is incurred for charging the body capacitance (C_B), and for switching the control transistors (C_S):

$$O_{BB} = (C_B + C_S)\Delta V_{bias}^2 \tag{4}$$

where ΔV_{bias} is the offset of body voltage when BB is applied. EBT of BB can then be calculated by:

$$B_{BB} = \frac{O_{BB}}{S_{BB/time}} \tag{5}$$

where $S_{BB/time}$ is the leakage energy saving per unit time by applying BB. For BB, its large C_B also leads to large overhead (O_{BB}), and thus long EBT (B_{BB}) for BB. The typical value of C_B and B_{BB} is shown in Row 2 of Table 1.

Conventionally, the optimum body biasing voltage of V_{th} Hopping is determined when the sum of subthreshold and BTBT leakage is minimized. This voltage achieves high leakage reduction ratio, but incurs large energy overhead due to the quadratic dependency of of O_{BB} on ΔV_{bias} (Equation 4). Figure 3 demonstrates the energy overhead versus leakage saving of different biasing voltages on an inverter chain (our studies [9]).

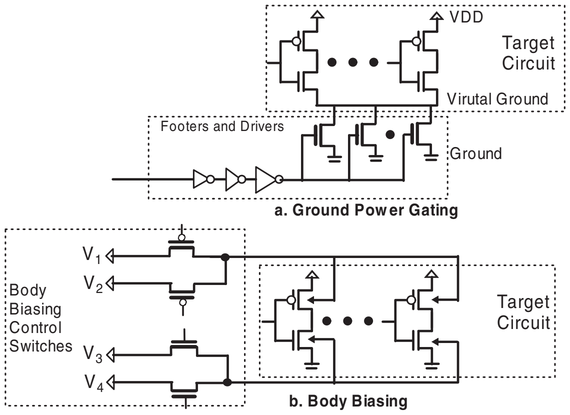


Figure 2. Power Gating and Body Biasing

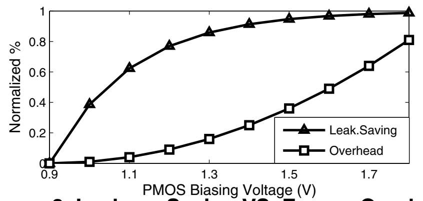


Figure 3. Leakage Saving VS. Energy Overhead

We further conducted experiments on an ALU. As shown in the Row 2 and 3 of Table 1, instead of using large ΔV_{bias} (0.7V), we can choose small ΔV_{bias} (0.15V) to reduce EBT for $8\times$, while maintaining 51% leakage reduction (Z_{min}). 51% might not be high enough for standby leakage reduction, but is sufficient for short idleness exploitation. This is the basic idea of LW-VH.

Table 1. Typical Values of PG, BB and LW-VH on ALU 32nm

	C_F	C_B	ΔV_{bias}	$overhead$	I_{Leak}	Z_{min}	EBT	MIT	$RMIT$
PG	77	-	-	62	23	99%	15	17	89
BB	-	401	0.7	199	23	95%	23	25	130
LW-VH	-	401	0.15	9.1	23	51%	3	4	18

2.2 Leakage Control with TV/PV Impact

Leakage of the target circuit varies significantly with temperature and process in scaled technologies. When designing leakage control policy, worst case scenario should be considered. Assume that leakage current of the target circuit varies from I_{min} to I_{max} (3σ values). According to its definition, EBT (B_{norm}) of the circuit varies:

$$\frac{E_{overhead}}{Z \cdot I_{max} V_{DD}} = B_{min} \leq B_{norm} \leq B_{max} = \frac{E_{overhead}}{Z \cdot I_{min} V_{DD}} \tag{6}$$

where Z is the leakage reduction ratio. A robust leakage control policy should guarantee positive energy saving in all dies with all possible temperatures. Thus, the worst case EBT (B_{max} in Equation 6) must be chosen to represent EBT of the target circuit. We call this phenomenon as "corner case leakage control". We conducted Monte Carlo simulation on an ALU with 32nm technology to demonstrate this phenomenon. The 3σ variations of gate length and T_{ox} are 12% and 4%, respectively. The temperature variation is set to be from 70°C to 110°C, with normal temperature as 90°C. Figure 4 shows the leakage variation induced by TV and PV. The corner case leakage can be as low as 1/6 of the nominal value. According to Equation 6, runtime EBT of the target circuit can be magnified *6 times*, as shown in the last column of Table 1. This dramatically weakens the effectiveness of any leakage control technique.

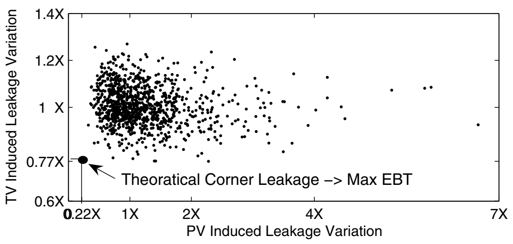


Figure 4. TV/PV Induced Leakage Variation

3. LEAKAGE CONTROL POTENTIALS AT MICROARCHITECTURAL LEVEL

The potential of MA level leakage control is determined by three factors: 1) How much idleness there exists at MA level. 2) How much idleness can be detected. 3) How much idleness can be exploited by leakage control techniques to yield leakage saving. We will answer the first question in Section 3.1, the second question in Section 4, and the third question in Section 3.2.

3.1 Mutual Exclusion, Locality and Exceptions

The idleness at MA level is caused by three mechanisms: mutual Exclusion, exceptions and locality. Mutual exclusion refers to the case that when CPU is executing instructions in sequence, some FUs are activated by certain types of instructions, while others are not. Here we classify all instructions into four basic types: control, memory manipulation (load/store), integer arithmetic, floating point arithmetic. Table 2 summarizes the busy/idle FUs when different instructions are executed. The leakage of each FU is also shown in the last column [2]. We can see that ALU, FPU, and branch target buffer (BTB) are the three major sources of CPU leakage.

Table 2. Leakage Power and Idleness of FUs (32nm)

FU	Idleness				Leakage [2]
	Control	Load/Store	Int. Arithmetic	F.P. Arithmetic	
ALU *3	Y	Y	N	Y	34.4
FPU	Y	Y	Y	N	21.57
BTB	N	Y	Y	Y	87.39
Integer Reg.	Y	N	N	Y	1.57
FP. Reg.	Y	N	Y	N	1.57
RUU	N	N	N	N	3.48
LSQ	N	N	N	N	3.14
Decode	N	N	N	N	1.60

The second type of idleness is created by the locality in programs. A typical example is the memory access locality. At runtime, it is common that a subset of cache system will be accessed frequently during a short period. So the rest of cache system exhibits very long idleness ($>$ 100 cycles), and creates perfect applications for leakage reduction. Cache leakage reduction is well studied, and thus will be ignored here. The third type of idleness at MA level is caused by the exceptions during CPU runtime. A typical example is branch mis-prediction. When branch mis-prediction occurs, the current pipeline execution will be flushed. This event usually creates 10 to 20 cycles of idleness for ALU, FPU and BTB, and can be exploited for leakage reduction.

3.2 Leakage Saving Potentials

A suitable leakage control technique is needed to efficiently exploit these idleness, and maximize leakage saving. Denote the EBT, WUT, leakage reduction ratio and energy overhead of a certain leakage control technique as B, W, Z and O respectively. Also denote the length of idle period as T_i, and the occurrence of T_i as U_i. The net energy saving (E_i) for idleness of length T_i is:

$$E_i = (T_i - W)V_{DD}I_{leak}Z - O \tag{7}$$

The first term represents the leakage saving. By subtracting the second term (energy overhead), we have the net energy saving. By mutating Equation 3 or 5, the second term can be further expressed as:

$$O = V_{DD}I_{leak}ZB \tag{8}$$

Thus, Equation 7 can transform into:

$$E_i = (T_i - B - W)V_{DD}I_{leak}Z = (T_i - M)V_{DD}I_{leak}Z \tag{9}$$

where M denotes MIT ($B + W$). Finally, we can determine the overall efficacy of leakage control by:

$$P = \frac{\sum_i U_i E_i}{V_{DD}I_{leak}T_{runtime}} = \frac{\sum_i U_i((T_i - M))Z}{T_{runtime}} \tag{10}$$

where the numerator represents the total net energy saving for all idle events, the denominator represents the total leakage energy consumption during program runtime. P is the percentage of energy saving by leakage control. It can be observed that the MIT (M) and leakage reduction ratio (Z) determine the efficacy of leakage control. For PG and BB, their M and Z values are fixed. M and Z of LW-VH are tunable, depending on the biasing voltages. Figure 5 shows the experiments on the net leakage saving of an ALU with different leakage control techniques. Although the Z value of PG and BB is high, their high M value leads to large overhead and thus low leakage saving. LW-VH can squeeze its M to a very small value, and achieve better leakage saving. (Maximum is 19% when $M = 5$.)

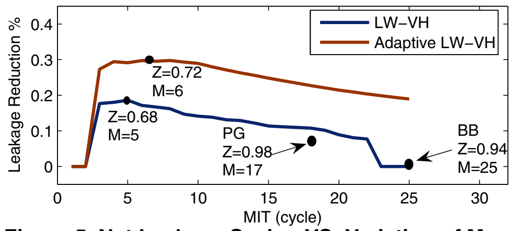


Figure 5. Net Leakage Saving VS. Variation of M and Z

Now assume that we implement a mechanism for TV/PV compensation, such that LW-VH can be applied according to the actual runtime MIT of the target circuit, instead of the "corner case" MIT. As such, leakage saving can be further improved. We call this technique as "adaptive LW-VH". The effectiveness of adaptive LW-VH on ALU is shown in Figure 5, where it achieves 30% maximum leakage saving when $M = 6$. The implementation of it will be introduced next.

4. STRETCHING THE LIMIT OF LEAKAGE CONTROL WITH ADAPTIVE LW-VH

4.1 The Overall Scheme

In many CPU architectures, an instruction queue (IQ) exists between the fetch unit and decode unit. This IQ provides us the ability to monitor future workloads, and apply leakage control accordingly. A counter can be used to monitor instruction types in IQ, and decides which FUs to be turned off. For example, if there is no FP instructions in the next N cycles, and MIT of FPU is larger than N, FPU can be turned off. Similarly, ALU and BTB can be turned off based on monitoring integer arithmetic and control instructions, respectively. The whole scheme is illustrated in Figure 6.a, where each major FU (ALU, FPU and BTB) has a dedicated counter to monitor the consecutive idleness of its corresponding instructions. When the counter value exceeds MIT of the FU that it is monitoring, LW-VH will be triggered and applied to that particular FU.

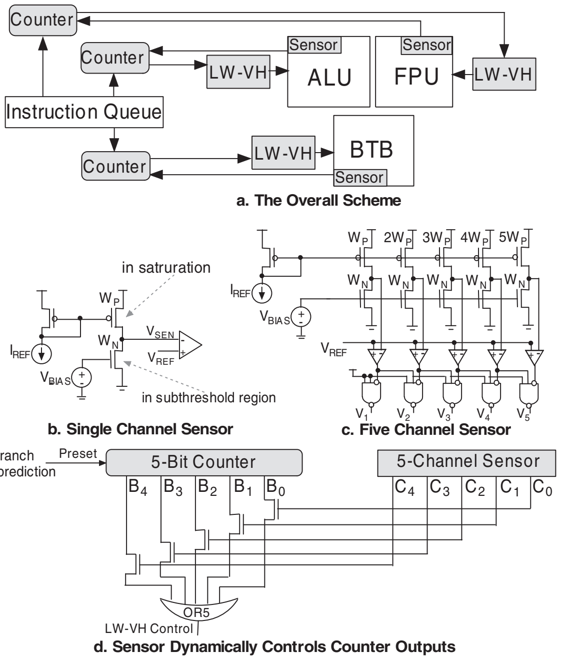


Figure 6. Microarchitectural Level Adaptive LW-VH

To address the "corner case leakage control" problem, leakage sensor [10] is used. The diagram of this sensor is shown in Figure 6.b. The output of the single-channel sensor is a one-bit digital signal, indicating whether the actual leakage is larger than the reference. Multiple-channel sensor can be used to improve the resolution, as shown in Figure 6.c. The sensed leakage value will be used to determine the runtime MIT (RMIT) of the FU, and change the control policy dynamically. Take ALU for example. The HSPICE simulation on 1024 die samples (Table 1) suggested that the RMIT of ALU spans from 1 to 18 cycles. So the counter for ALU should have at least 5 bits ($2^5 > 18$). Next, we use a sensor with 5 channels (C_0 to C_4 in Figure 6.d) to represent 5 different leakage levels, which yield 1, 2, 4, 8 and 16 cycle RMIT respectively. Whenever the actual leakage value is larger than the reference value, the corresponding channel will give output '1'. The output of the sensor, together with the counter value, are used to generate the control signal for adaptive LW-VH, as shown in Figure 6.d. For example, when runtime leakage of ALU yields 6 cycle RMIT, the output of sensor channel C_3 (corresponding to 8-cycle RMIT) and C_4 (corresponding to 16-cycle RMIT) will be '1', so the B_3 and B_4 of the counter will be passed to the OR5 gate. So if the counter detects more than 8 (binary '01000') cycles of idleness, the output of OR gates will be logic '1', and LW-VH will be triggered.

To exploit the idleness created by exceptions, we program the idle length created by the exception into counters. Assume that branch mis-prediction creates 10 cycles of idleness for each FU. Then we can program the preset value of all counters as 10 (binary '01010'). Whenever branch mis-prediction occurs, each counter will be preset to 10, as shown in Figure 6.d. If the sensed RMIT of any FU is larger than 10 cycles, LW-VH will be applied to that FU.

4.2 Leakage Control Resolution and Granularity

In Section 4.1, the resolution of the sensor is determined arbitrarily, leading to sub-optimal leakage control. Increasing counter and sensor resolution can address this problem, but incurs more area for leakage sensor. The granularity of leakage control is another open question. In the previous study, we arbitrarily decided that the granularity is at the FU level. An FU is considered as a whole for sleep control. Leakage control with finer granularity is possible, if we consider the sub-circuits inside each FU. As shown in Figure 7, an ALU is separated into three sub-circuits: adder, multiplier and logic units. Each sub-circuits has a dedicated counter, a leakage sensor and LW-VH control switches. Each counter needs to counter the corresponding instruction types (add instruction for adder; multiply instruction for multiplier; other ALU instructions for logic operations), and decides the application of adaptive LW-VH on its sub-circuit. Even finer granularity is also possible. For example, we can divide logic operations into or/shift/and, and separately control the circuit for each operation. Another example is that the adder can be divided into LSB and MSB. By monitoring the operands, we can decide whether to turn off the MSB or not. However, finer granularity also demands for more control and monitoring circuits, and thus consumes more area and power.

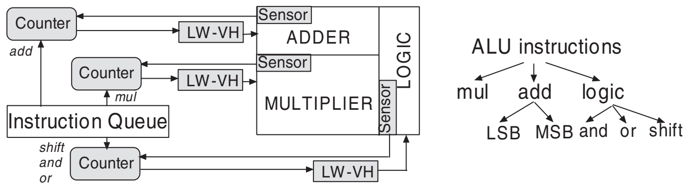
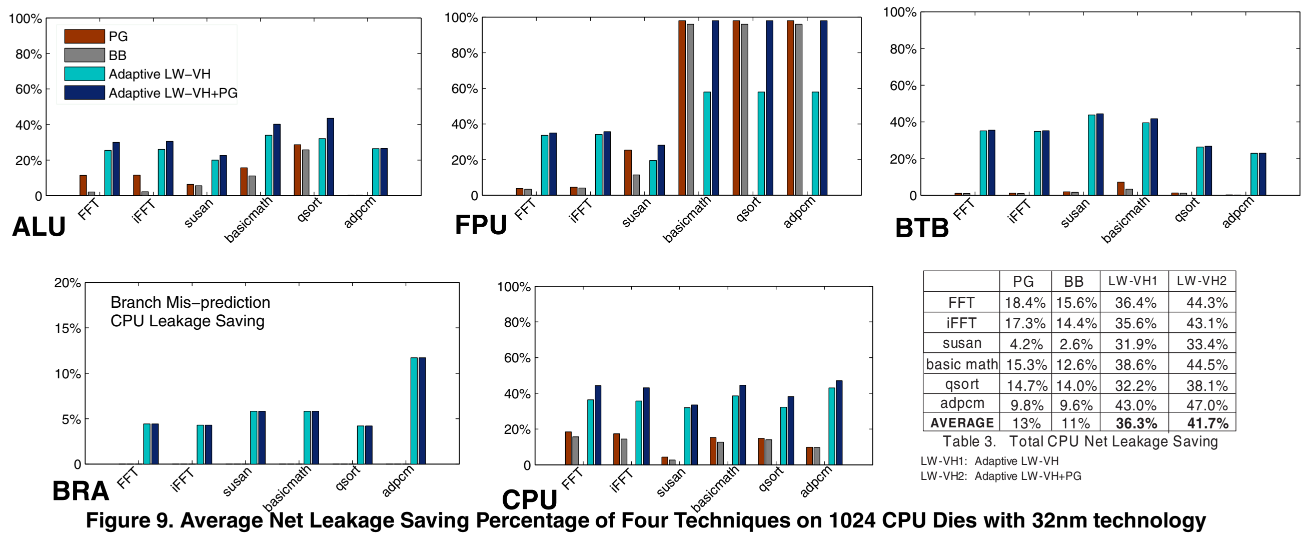


Figure 7. Leakage Control With Finer Granularity

5. EXPERIMENTAL RESULTS

We conducted experiments to verify the effectiveness of LW-VH with adaptive control for MA level leakage control. The CPU layout is illustrated in Figure 8.a. 1024 CPU die samples with TV/PV were generated with 32nm predictive technology. On each die, there are three high-leakage FUs: ALU, FPU and BTB. For each FU, LW-VH was designed and optimized in nominal conditions. HSPICE simulations were conducted to determine the RMIT span of each FU for all dies. Then the RMIT span is used to design the adaptive control scheme. Finally, we modified SimpleScalar to estimate the net leakage saving of



	PG	BB	LW-VH1	LW-VH2
FFT	18.4%	15.6%	36.4%	44.3%
iFFT	17.3%	14.4%	35.6%	43.1%
susan	4.2%	2.6%	31.9%	33.4%
basic math	15.3%	12.6%	38.6%	44.5%
qsort	14.7%	14.0%	32.2%	38.1%
adpcm	9.8%	9.6%	43.0%	47.0%
AVERAGE	13%	11%	**36.3%**	**41.7%**

Table 3. Total CPU Net Leakage Saving

LW-VH1: Adaptive LW-VH
LW-VH2: Adaptive LW-VH+PG

Figure 9. Average Net Leakage Saving Percentage of Four Techniques on 1024 CPU Dies with 32nm technology

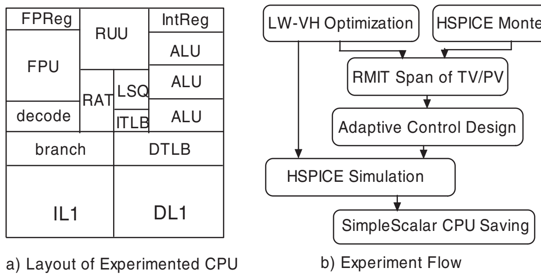


Figure 8. Experiment Settings

the whole CPU. The experiment flow is shown in Figure 8.b.

Figure 9 and Table 3 show experimental results on the overall leakage reduction for different benchmark programs. We compare the results of four techniques: PG, BB, Adaptive LW-VH and Adaptive LW-VH with PG. In Figure 9, PG and BB have minor savings on most benchmark programs for each FU (ALU, FPU, BTB) and for branch mis-prediction (BRA), while adaptive LW-VH achieve 36.3% total CPU leakage reduction. That is 23.3% improvement over PG, and 25.3% over BB.

Figure 10 demonstrates the improvement of ALU leakage saving by using finer resolution and granularity. The improvement is significant when resolution is low, and is minor when resolution is larger than 5. Leakage control of finer granularity yielded 26% average improvement, since most benchmark programs have little to no multiplication operations (except 'susan'). Hence putting Multiplier into sleep separately can yield substantial leakage saving.

6. CONCLUSION

This paper targets active leakage control at MA level. The obstacle is the energy overhead problem of PG and BB for mode transition. When TV/PV are considered, PG and BB can hardly achieve leakage saving for an active CPU. Adaptive Light-Weighted Vth Hopping (LW-VH) technique has been introduced for overhead reduction and TV/PV compensation. We systematically analyze the leakage control potentials at MA level, and propose a comprehensive scheme to maximize leakage control at MA level.

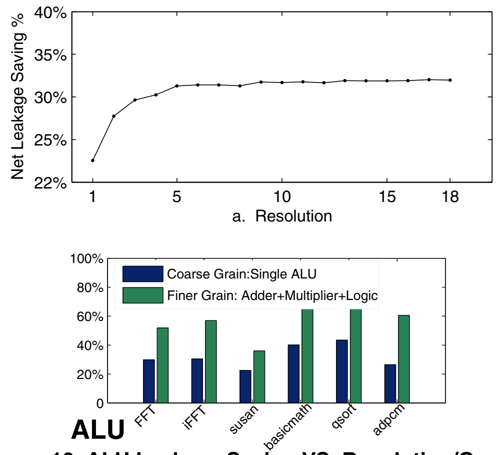


Figure 10. ALU Leakage Saving VS. Resolution/Granularity

References

[1] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," in *IEEE*, Feb. 2003, pp. 305–327.

[2] W. Liao, L. He, and K. Lepak, "Temperature and supply voltage aware performance and power modeling at microarchitecture level," *IEEE J. Technol. Computer Aided Design*, vol. 24, pp. 1042–1053, July 2005.

[3] U. of Michigan at Ann Arbor. Mibench. [Online]. Available: www.eecs.umich.edu/mibench/

[4] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *ISLPED*, Aug. 2004, pp. 32–37.

[5] S. Roy, N. Ranganathan, and S. Katkoori, "A framework for power-gating functional units in embedded microprocessors," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 1–1, Jan. 2009.

[6] L. He, W. Liao, and M. R. Stan, "System level leakage reduction considering the interdependence of temperature and leakage," in *DAC*, 2004, pp. 12–17.

[7] A. L. et al., "Dynamic power gating with quality guarantees," in *ISLPED*, Aug. 2009, p. to appear.

[8] D. Kannan, A. Shrivastava, V. Mohan, S. Bhardwaj, and S. Vrudhula, "Temperature and process variations aware power gating of functional units," in *VLSID*, Jan. 2008, pp. 515–520.

[9] H. Xu, R. Vemuri, and W.-B. Jone, "Run-time active leakage reduction by power gating and reverse body biasing: An energy view," in *ICCD*, Oct. 2008, pp. 618–625.

[10] C. Kim, K. Roy, S. Hsu, R. Krishnamurthy, and S. Borkar, "A process variation compensating technique with an on-die leakage current sensor for nanometer scale dynamic circuits," *IEEE Trans. VLSI Syst.*, vol. 14, pp. 646–649, Nov. 2006.