Reduce Register Files Leakage Through Discharging Cells

Lingling Jin, Wei Wu, Jun Yang
Dept. of Computer Science
Univ. of California, Riverside, 92521
Email: ljin,wwu,junyang@cs.ucr.edu

Chuanjun Zhang
Dept. of Computer Science
and Electrical Engineering
Univ. of Missouri-Kansas City, 64110
Email: zhangchu@umkc.edu

Youtao Zhang
Dept. of Computer Science
Univ. of Pittsburgh, 15260
Email: zhangyt@cs.pitt.edu

***Abstract*— We propose a low-leakage register file cell design based on the observation that the physical registers in a super-scalar processor have very short life cycles. When a register is dead, we discharge its cells to '0' to greatly reduce the leakage current from the read bitlines to the ground. Our design has no impact to critical register read access path. Projected to future 45nm technology, our design yields additional 38% and 47% leakage power savings on top of the existing low-leakage cell designs for 64-bit and 32-bit datapath, respectively. Taking into the account of dynamic energy savings due to the elimination of write '0' operations, our design saves nearly 20% of total energy.**

I. Introduction

Recent trend in continuous reduction in CMOS technology size has introduced new challenges to microprocessor designs. Aggressive V_t scaling results in exponential growth in leakage current and within a few process generations, it is predicted that the leakage energy could be comparable to the dynamic energy [2]. Such a situation is particularly critical for register files. Recent study has shown that the integer register file is typically the hottest spot of a microprocessor [7], and leakage power is an exponential function of the temperature. Therefore, reducing the leakage power of the register file is a critical problem for efficient thermal management and high-performance processor designs.

There have been various techniques to reduce the leakage power of a register file. Most leakage optimized design adopts low leakage, but also slow transistors, e.g. with high-V_t, on non-critical paths. However, this method cannot be applied to critical path, such as register read path. R. Krishnamurthy *et. al.* reported 13%[5] slowdown by using high-V_t gate on read bitline. A more substantial set of leakage reduction techniques called "LBB" were presented by S. Heo *et. al.* [3]. They segment read bitline, along with the register file into subbanks. Once an entire subbank is dead, the subbank read bitline is turned off, saving the leakage on the bitline. As we can see, this scheme is only effective when all of the registers in a bank are dead. Also, when the register bank turns to be alive, pre-charging the read bitline to '1' is required, which causes energy overhead and impacts register read access time.

In this paper, we propose a new technique that can reduce the leakage current on the read bitlines without sacrificing the CPU performance. Our design is based on the findings that the leakage power dissipated when a cell stores '0' is only 1/8 of the leakage power when a cell stores '1'. The basic idea of our scheme is to turn bit '1' into '0' whenever the register is dead. We discharge the '1' using a small high-V_t gate so that the energy overhead by discharging is minimum. Our scheme begins to save leakage immediately after the register is dead, rather than waiting for the entire bank of registers to become free as in [3]. Also, nullifying a dead register will not change the correctness of the program execution. In our design, no performance overhead is imposed to read operations.

We evaluated our design using both circuit simulation and architecture simulation. Both 64-bit and 32-bit datapath are studied. Projected to 45nm technology, we saved 38% and 47% of total leakage power for 64-bit and 32-bit datapath respectively, or 11% and 14% of total register power. The discharge circuit only imposes 2.5% dynamic energy overhead on write operations but no overhead on read operations. This overhead can be reduced partially, because write '0' instruction does not need to be performed since after discharging, the register's default value is '0'. With this optimization, we can save additional 6% and 7% dynamic power, and a total of 16% and 20% of register power for 64-bit and 32-bit processor respectively.

The remainder of the paper is organized as follows. Section 2 reviews the low-leakage register cell structure and the motivation of our design. Section 3 elaborates the new cell structure. Section 4 and 5 presents the circuit and simulation experiment results. Section 6 concludes this paper.

II. Background and Motivation

In this section, we first give a brief review of the conventional low-leakage register file cell structure which is optimized for stored value of '0'. Then, we analyze its leakage features and our observations.

A. Register cell analysis

Due to the large number of read and write ports, standard register's cell is different from cache cell. First, the read bitline is in single-ended favor to reduce the routing and area cost. Second, in order to increase access speed and protect the bit value in cell, register file uses a two-transistor read port. Third, to optimize register access energy, all the read ports are arranged to one side of the storage cell. Based on the fact that

more zeros are stored in register than ones[12], designer placed read ports to the side so that when '0' is stored, the single-ended read bitline is not discharged during evaluation[9].

In Figure 1, we illustrate such an 8-read and 4-write ports, dual-V_t cell structure in which all transistors except those (e.g., $\mathtt{m}_3$ and $\mathtt{m}_2$) on read ports have high-V_t transistors.

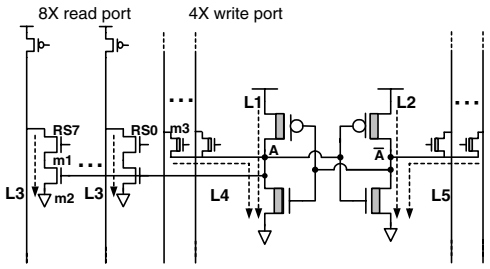

Fig. 1. An 8-read 4-write low-leakage register file cell. High-V_t transistors are shaded.

The cell in Figure 1 has five main leakage paths, labeled from L1 to L5 in the graph. We used HSPICE TSMC 0.18 μm technology to measure the leakage power at 100°C on all five types of paths when the cell is storing '1' and '0' respectively. The results are summarized in Table I.

	Storing '1'		Storing '0'	
	(nW)	%	(nW)	%
L1	1.013	0.28	0.606	1.24
L2	0.618	0.17	0.968	1.98
L3(×8)	363.22	99.23	46.171	94.41
L4+L5(×4)	1.192	0.33	1.159	2.37
Total	366.042	100	48.905	100

TABLE I

LEAKAGE DISTRIBUTION OF THE CELL WHEN STORING A '1' AND A '0'.

As we can see that L1, L2 and L4, L5 are all low-leakage paths for both bit '1' and '0' since their leakage path contains high-V_t transistors. L3 is critical read path, and uses low-V_t transistors. So its leakage dominates the total leakage expense. What is interesting here is that the leakage power on L3 when the bit is '1' versus '0' differs greatly so that total leakage current for bit '1' is approximately 7.5 times the current of bit '0'.Therefore, suppressing the leakage from the bitline to a cell storing a '1' becomes very important on this already low-leakage cell structure.

B. Register live range analysis

In a typical high-performance superscalar processor, an architecture register A that appears as the destination of some instruction is first mapped to a physical register P. Some cycles later, P will be written with the result of the instruction, and the result is also passed to pending reads. When the instruction is retired, P will be returned to the free list maintained by the renaming logic. At this point, physical register P is considered *dead* since its stored value will be considered as invalid by the processor. Therefore, we define the life cycle of a register to be from the time when it is written till the time it becomes dead.

Our design is based on the fact that registers usually have very short life cycles in a superscalar processor. Here we refer to *physical* registers as opposed to *architecture* registers. When the register is not alive, its value needs not be maintained and we can discharge all the '1's into '0' so that the leakage on L3 in Figure 1 is greatly reduced.

To see the opportunities of reducing the leakage on L3 by discharging a register once it is dead, we ran experiments on SPEC2000 to measure the percentages of the time the registers being dead vs. live. We found that on average, a register is alive for only 16% of the time. This feature shows great opportunity for saving the leakage power when a register is dead. If we assumed the same ratio of number of '1' versus number of '0' as before, i.e., 25% vs. 75%, final leakage that exist is

$$\frac{25\% \times (7.5 \times 16\% + 1 \times 84\%) + 75\% \times 1 \times 100\%}{75\% \times 1 + 25\% \times 7.5} = 48\%$$

of the current total leakage if all the cells holding a '1' is discharged once a register is dead. Next, we will describe how our new cell is designed.

III. THE DESIGN OF THE CELL

The primary modification to the register circuitry is to add a discharge gate to the storage cell of each bit, as shown in Figure 2. When a register is dead, we use this discharge gate to force the bit value to become zero. As we have shown earlier, a cell storing a '1' has 7.5× the leakage than storing a '0'. Turning a '1' to '0' will greatly reduce the leakage current on read bitline.

A. The new storage cell design

We add a transistor with a width of two feature size to point A. We term this transistor a "discharge" gate since its function is to discharge the cell storing a '1' when the entire register is dead. The discharge gate is turned of during the normal read/write operations. This transistor itself is of high-V_t so that it consumes little leakage power when it is off. The control signal is generated from the renaming logic, indicating when the register is dead.

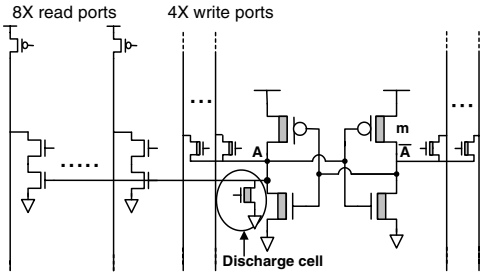

Fig. 2. The design of new cell with discharge gate.

The discharging procedure works as the following. When the register is dead and the discharge gate is turned on, if a cell already contains a '0', no effect is taken; if it contains a '1', the voltage at A is pulled down to '0' gradually, causing the PMOS m to open, which pulls up $\overline{\mathtt{A}}$ to '1'. Therefore, the cell is flipped from storing a '1' to storing a '0'.

This procedures resembles a write access that flips the cell's content. A normal write is performed by the two write bitlines that push the value into the cell simultaneously. Whereas in our case, the cell is flipped by a single transistor. Note that all

the bits in a register share the same discharge signal, similar to an extremely simplified write port with no decoder, no bitline, but a single write select with a fixed bit value of zero. As a result, it is slower than the normal write operation. This can be seen through the voltage variation at A when the cell is discharging versus when the cell is being written with a '0'. We also measured the total current in the cell for the two different operations (see Figure 3).

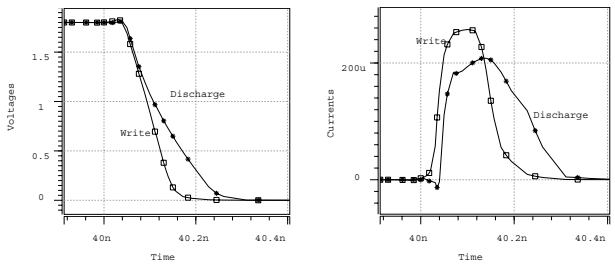

Fig. 3. Voltage at A and current in the cell.

From Figure 3, it is easy to see that the voltage of the discharging process drops slower than that of a write access. This has made the current in the cell persist longer, introducing a slightly larger energy than that in a normal write operation. An implication here is that if a cell originally stored a '1' and was discharged to '0', but then will be written with '1' again, there will be extra dynamic power burned to change the state of this cell twice. However, our discharging method is very lightweight. We have measured that the energy dissipated in the storage cell is at most 4% of the total energy on the entire write path including the decoder and write bitline, the detailed number will be show in Section 4 TableIV.

Another overhead of our design comes from the discharge signal wordline that drives the discharge cells. It resembles a normal wordline. Adding such a line to the discharge cell introduces some area overhead. In the original register file design, there are 12 wordlines in total: eight for reads and 4 for writes. Adding extra wordline imposes about 8% (1/12) of area overhead to register data array. We used Cacti [8] to analyze the area of other components such as the sense amplifiers and data output drivers. We found that the overall area overhead is below 5% for a 64-bit register file. We will give detailed power analysis for the discharge cell and the signal line in the section 4.

B. Setting the discharge signal

This discharge signal is set by the renaming logic because it keeps the information about the liveness of a physical register. In a conventional superscalar, the renaming logic has one entry for each physical register to record its status [14]. Typical renaming logic contains an "unmapped" bit and a "completed" bit[13]. The "unmapped" bit indicates whether or not a physical register is mapped to an architecture register; and the "completed" bit denotes if the physical register has been written. These two bits can be utilized to switch the discharge gate. Most renaming logic is placed close to register file such as Alpha 21264, because each read/write register operation will trigger the updating of the status bits. The energy dissipated to pass the signal from renaming logic to register file should be negligible

C. Zero detection for saving partial write energy

Since the register values are discharged to '0' as soon as they are not live anymore, we can safely claim that a register contains a '0' before it is written with a new value. Moreover, there is no need to re-write a '0' into a register since it already contains one. Such an optimization can save the dynamic power dissipated due to the write of '0'.We ran SPEC2000 benchmarks, and found that on average, above 25% of the values being written are '0'. There are substantial opportunities for saving the dynamic power.

To avoid a redundant write of '0', it is sufficient to detect if the value is a '0' before it is written into the register file. Such a detection is typically present in modern processors for various optimizations such as early zero detection [6], early detection of divide-by-zero [1], or partitioned register file design [4]. Adding such a detection logic does not prolong the datapath but often reduces the latency in the execution stage. Therefore, we leverage the existing techniques for the zero-detection before the register writes and throttle the access if a zero is detected. The overhead of the detection circuitry is negligible as it can be combined with the ALU without any additional latency [6].

IV. Circuit Evaluation

In this section, we present both timing and power simulation results for base case and our design. We built the register file circuit using TSMC low-V_t and high-V_t processes, and simulated read and write operations in HSPICE. As described earlier, we modeled two register files of 128 physical registers, each of which is 64-bit or 32-bit. The entire register file is divided into 8 sub-banks with each sub-bank containing 16 registers similar to the design in [5].

A. Process technologies

We used five generations of dual-Vt processes: 180nm, 130nm, 90nm, 65nm and 45nm process, to evaluate our design. We used the TSMC 0.18μ library from MOSIS [11] to simulate the 180nm high-V_t and low-V_t transistors. Other technologies' parameters are scaled based on ITRS's recent consensus prediction[10]. The parameters of 180nm technology is measured at 100°C.

Based on [10],we estimated the percentage of leakage power within the total power, and ploted it in Figure 4. We can see that for 180nm technology, the leakage power is only about 2% of the total register file power, while in 45nm technology size, it increases to 32%. The role of leakage power becomes evident as the technology size shrinks. Hence, our technique will be of more significance to future technologies.

B. Leakage power reduction with the discharge gate

The purpose of putting a discharge gate in a cell is to pull the bit value from '1' to '0' so as to close the leakage path

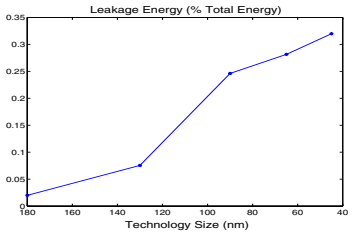

Fig. 4. The trend in leakage and dynamic power with future technologies.

from the bitline due to a '1'. Therefore, the more the number of '1's in the register array, the more leakage power it can save. Another factor that affects the amount of total leakage savings is how long a register stays "discharged", i.e., being dead. We will report such results in the Experiment section as it requires architecture simulations. To see the leakage power for a single register line before and after the discharge, which depends on the 0-1 distribution in the cells, we first collected the data from 21 SPEC2K benchmarks for both the 32-bit and 64-bit register files. On average, there are 24% and 14% of cells that contain a '1' for 32-bit and 64-bit register arrays respectively. Recall that the leakage due to a '1' is nearly 7.5 times the leakage of a '0' (Table I). Such a difference makes the potential savings for flipping '1' significant.

Combining the distributions and the leakage power of a cell containing a '0' or a '1' in Table I (the leakage due to the added high-V_t gate is negligible), the leakage power of a 64-bit or a 32-bit register can be derived. We show such leakage for a *dead* register line with five technology sizes in Table II. As we can see that with the discharge cell, 47.6% and 59.9% of savings can be achieved for a 64-bit and a 32-bit register respectively.

64-bit (μW)					
Tech Size(nm)	180nm	130nm	90nm	65nm	45nm
Base Case	6.17	18.58	51.45	51.10	48.24
w/discharge cell	3.13	9.42	26.10	25.90	24.45
32-bit (μW)					
Base Case	3.90	11.73	32.49	32.27	30.46
w/discharge cell	1.56	4.71	13.04	12.95	12.22

TABLE II

LEAKAGE POWER OF A DEAD REGISTER.

C. Dynamic energy overhead with the discharge gate

Discharging storage cell's content has impact on pending write operations, and driving the discharge gate causes extra dynamic energy. In this section, we give quantitative analysis on dynamic energy overheads and compare them to normal write energy.

Our new cell structure introduces a small amount of dynamic power to the write accesses. The energy of read accesses is intact since reads happen only when the register is live. The writes consume more power because the cells may have changed states when they were dead. Specifically, the dynamic power overhead in write accesses mainly comes from the following three aspects:

1) Discharging a cell containing a '1' — a '1' becomes '0' in the cell;
2) Writing a '1' into a cell following action (1) — a '0' becomes '1' again;
3) Charging the extra gate during the writing of a '1' — an 'x' becomes '1';

The first three aspects affect all state changes in a storage cell except for the case of '0' to '0'. To understand the power during the state changes quantitatively, we simulated those cases in HSPICE and list the results in TableIII. The numbers presented here only includes energy dissipation on storage cells.

before → after	Base (pJ)	Our design (pJ)		
		discharge	write	total
0→0	0.00005	0.00001	0.00005	0.00006
0→1	0.04811	0.00001	0.04865	0.04866
1→0	0.05158	0.05997	0.00005	0.06002
1→1	0.00005	0.05997	0.04865	0.10916
Weigh Avg(64b)	0.01204	0.01526		
Weigh Avg(32b)	0.01769	0.02503		

TABLE III

DYNAMIC ENERGY OF WRITING ONE BIT REGISTER CELL (180NM TECHNOLOGY).

In the first column, 'x'→'y' means writing a value 'y' into the cell holding an 'x'. Second column shows the write energy in the base case. In our design, a write operation undergoes two steps: discharge, and then write, which are shown in column three and four respectively. The last column summarizes the previous two columns to obtain the total dynamic power. That is, the total energy of 'x'→'y' is computed as the energy in discharging 'x' plus the energy of '0'→'y'.

We can see that in case of '0→0', little energy is consumed in either base case or our new design because the state of the cell does not change. In case of '0→1', the discharging energy is still very little, but the write energy in our design is larger than the base case (0.04865 vs. 0.04811) because the discharge gate itself is effectively a small capacitor added to A in Figure 1. It would be charged slightly when '1' is written. In case of '1→0', the discharging energy becomes dominant and is also larger than writing the cell with '0' ($0.05997 > 0.05158$) for the reasons explained in section 3. However, this is only for the energy in the cell. If we use a write access to flip the bit, it would cost a lot more energy since other logic such as the decoder would be involved. For the above three states, the energy increases in our design are all quite modest since the cell state changes at most once.

Lastly, in case of '1→1', the discharging and the writing energy share equal role since the cell state changes twice. The energy for one cell in this case increases significantly. However, the probability of '1→1' is quite low. we have found that across different benchmarks, the write accesses are roughly 39% of the total register accesses while the 1→1 happens only 3.4% (6.1%) of the total writes for 64-bit (32-bit) registers. We will see later that the overall dynamic energy increase is very minimum

Another major overhead comes from the extra wordline that

drives to the discharge gate. The energy of this wordline is around half of the the normal register wordline energy because the discharge signal only drives one gate while the wordline drives two gates per cell. For example, the energy for driving the signal of a 64-bit register file is 0.315pJ.

Based on above detail energy analysis, we list the dynamic energy for base case and overhead in our design in Table IV. Since only write access is affected by our scheme, only dynamic write energy is listed. The overhead is measured for one discharge operation. Values for different process technologies are also listed.

64-bit (1E-3 pJ)					
Tech Size(nm)	180nm	130nm	90nm	65nm	45nm
BaseCase	20392	11800	2547	947	459
cell overhead	206	119	26	10	5
discharge WL	315	182	39	15	7
Total overhead	521	301	65	25	12
32-bit (1E-3 pJ)					
BaseCase	16335	9453	2040	769	367
cell overhead	235	140	29	11	5
discharge WL	182	105	23	8	4
Total overhead	417	245	52	19	9

TABLE IV

DYNAMIC WRITE ENERGY INCREASE.

It can be seen that, our design only increases around 2.5% of total write energy. As technology size scaled down, dynamic overhead reduce quadratically. When it gets to 45nm technology, the overhead would be as low as 0.012pJ. Also, from our experimental experience, the writes account for roughly 39% of the total register accesses. So the dynamic energy overhead is only a small portion of the total energy consumption.

D. Discharge cell analysis

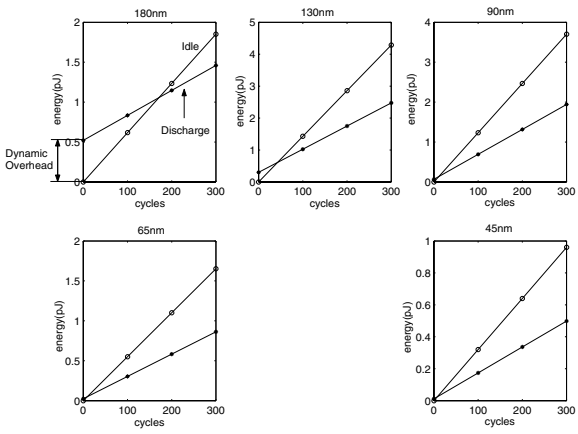

Fig. 5. Cumulative Base case energy and with discharge cell energy comparison of one dead register line for 64-bit datapath.

From above discussion, we can see that after a register line is dead, using the discharge gate can achieve 40%-60% of leakage power savings. However, the discharge gate itself dissipates extra dynamic energy due to the pull-down activity. If the leakage energy saved did not exceed the dynamic energy the cell spends, there would be no net energy savings. To achieve overall energy savings, the register dead cycles have to be long enough. We now discuss the opportunities of energy savings as register dead cycle increases. We illustrate the relation between cumulatedenergy consumption and dead cycles for different technologies in Figure 5.

The line labeled 'Idle' represents the energy consumption as the register dead cycle increases in the base case. The line labeled 'Discharge' represents our new register design. After the register is dead, the cumulative energy for the base case is increasing at a constant rate from the origin due to its constant leakage consumption after register is dead. With the discharge gate, cumulated energy starts from 0.512pJ (from TableIV) because the dynamic energy in the discharging process, and increases at almost half the speed of base case(due to reduced leakage, refer to TableII). From cycle '0' to the point where these two lines meet, our new design consumes more energy. However, as long as the dead cycles surpass the point, our design starts to gain energy savings. We term the point where two lines meet as 'breakeven' point.

As we can see, when the technology size decreases the breakeven point also moves backward. The breakeven point is 160 cycles for the 180nm technology, and reduced to 8 cycles for the 45nm technology. As a result, in the near future technologies, using our scheme can achieve energy savings in only few cycles after the register is dead.

V. EXPERIMENTS

A. Architecture parameters and benchmarks

To measure the collective leakage energy and dynamic energy changes, we simulated our design on a modified Simplescalar [15]. Detailed parameters of the simulator is listed in TableV.

Issue width	4
Reorder buffer	80
Int physical register	128
Ld/St queue	64/64
I-cache/D-cache	64K/2-Way/64B Block
Unified L2 cache	4M/8-Way/128B Block
IntALUs/IntMult/FPAlUs/FPMult	8/4/4/2

TABLE V

SIMULATION PARAMETERS.

We simulated 21 benchmarks from SPEC2K. Each benchmark is fast forwarded for one billion instructions and then executed for five hundred million instructions.

B. Overall leakage reduction and the impact on dynamic power

The overall leakage energy and total energy savings for 45nm technology normalized to the base case is shown in Figure 6. The leakage saving result is close to what we have estimated in the previous sections since .the lifetimes of the registers across different benchmarks do not vary too much. Also, as technology scales, the dynamic overhead is almost negligible. The register file with 32-bit values have some more savings than the 64-bit registers since the percentage of bit '1' is higher. On average, there are up to 38% and

47% leakage energy savings for 64-bit and 32-bit register files respectively. The overall energy shows about 11% and 14% savings. However, the savings will grow as technology size keeps on decrease.

Figure 7 reports the leakage energy and total energy savings for different processes. Around 30% leakage savings is observed in 130nm process. After 90nm, the leakage saving reaches 37% and 45%, and grows slowly afterwards. For 180nm technology, since its dynamic power is much larger than leakage power, the dynamic overhead of our scheme is an overkill, resulting negative leakage savings. However the dynamic overhead is quite small compared with register read/write energy, the total energy overhead is only 0.17% and 0.36% of the total energy for 180nm technology.

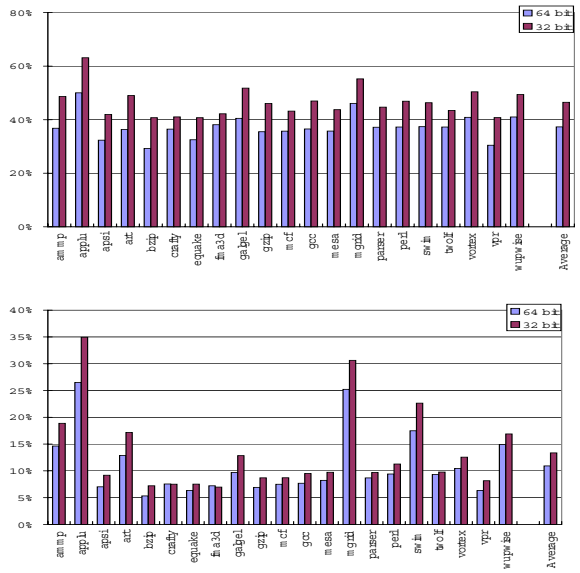

Fig. 6. Leakage energy and total energy savings for 45nm process

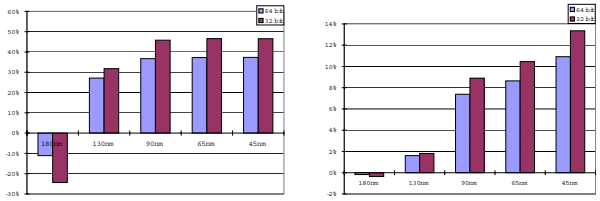

Fig. 7. Leakage energy and total energy savings across different processes

C. Saving dynamic energy by eliminating writing '0's

Though our design has introduced a small amount of dynamic power overhead, it also creates opportunities for saving dynamic power from other sources. We have explained earlier that writing a '0' into a register is unnecessary since by default, any register contains a '0'. Eliminating those writes saves power in decoder, wordline driving, and cell energy. On average 26% of writes can be eliminated. Considering the dynamic power overhead from the discharge gate, the net savings are 6% and 7% on average for the benchmarks we tested.

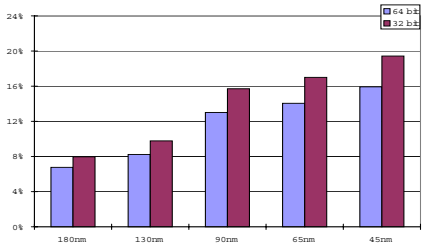

Fig. 8. Overall energy savings.

With the savings in dynamic energy and leakage energy, we re-evaluated the total energy savings across all technology sizes for our design. From Figure 8, we can see that even with 180nm technology, there are 7% and 8% of total energy savings, mostly from eliminating writing '0's. For 45nm, our total energy savings are 16% and 19.5%, showing the effectiveness of our design using a very simple technique.

VI. Conclusion

We proposed a technique to save leakage and dynamic power of a register file by observing that most physical registers have short life cycle time. We can discharge the cells in the dead registers to save the leakage on the bitlines. Experiments have shown significant savings (16-19.5%) and great potential using future technology sizes.

References

[1] D. Brooks and M. Martonosi, "Value-based clock gating and operation packing: dynamic strategies for improving processor power and performance," *ACM Transaction on Computer Systems,* Vol. 18, No. 2, pp. 89-126, May 2000.

[2] V. De and S. Borkar, "Technology and design challenges for low power and high performance," *ISLPED,* pp. 163-168, 1999.

[3] S. Heo, K. Barr, M. Hampton, and K. Asanovic, "Dynamic fine-grain leakage reduction using leakage-biased bitlines," *the 30th International Symposium on Computer Architecture,* pp. 137-147, 2003.

[4] M. Kondo, H. A. Nakamura, "Small, Fast and Low-Power Register File by Bit-Partitioning," *the 11th High-performance Computer Architecture,*, pp.40-49, Feb 2005.

[5] R. Krishnamurthy, A. Alvandpour, G. Balamurugan, N. Shanbhag, K. Soumyanath, and S. Borkar, "A 130-nm 6-GHz 256×32 bit leakage-tolerant register file," *IEEE Journal of Solid-State Circuits,* Vol. 37, No. 5, May 2002.

[6] D. R. Lutz, D. N. Jayasimha, "Early zero detection," *International onference on Computer Design,* pp. 545-551, 1996

[7] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. "Temperature-Aware Microarchitecture," *the 30th International Symposium on Computer Architecture,* pp. 2-13, 2003.

[8] P. Shivakumar, N. Jouppi, "Cacti 3.0: An integrated cache timing, power and area model," *Western Research Lab (ERL) Research Report,* 2001/2.

[9] J. Tseng and K. Asanovic, "Energy-efficient register access," *the 13th Symposium on Integrated Circuits and Systems Design,* Sept. 2000.

[10] ITRS 2003 Edition, http://public.itrs.net/Files/2003ITRS /Home2003.htm.

[11] MOSIS, https://www.mosis.org

[12] L. Villa, M. Zhang, K. Asanovic "Dynamic Zero Compression for Cache Energy Reduction," *the 33rd International Symposium on Microarchitecture,* Dec. 2000.

[13] M. Moudgill, K. Pingali, S. Vassiliadis "Register Renaming and Dynamic Speculation: an Alternative Approach," *In Proceedings of the 26th annual international symposium on Microarchitecture (Micro26),* 1993

[14] J.S. Liptay, "Design of the IBM Enterprise System/9000 high-end processor," *IBM J. Res. Develop. Vol.36 No.4,* July 1992

[15] D. Burger and T. Austin, "The SimpleScalar Tool Set, Version 2.0," *Technical Report 1342, Univ. of Wisconsin-Madison, Comp. Sci. Dept.,* 1997.