Appears in: The 29th Annual Int'l Symposium for Computer Architecture (ISCA-29), Anchorage, AK, May 2002

Dynamic Fine-Grain Leakage Reduction Using Leakage-Biased Bitlines

Seongmoo Heo, Kenneth Barr, Mark Hampton, and Krste Asanović

MIT Laboratory for Computer Science
200 Technology Square, Cambridge, MA 02139
E-mail: {heomoo,kbarr,mhampton,krste}@lcs.mit.edu

Abstract

Leakage power is dominated by critical paths, and hence dynamic deactivation of fast transistors can yield large savings. We introduce metrics for comparing fine-grain dynamic deactivation techniques that include the effects of deactivation energy and startup latencies, as well as long-term leakage current. We present a new circuit-level technique for leakage current reduction, leakage-biased bitlines*, that has low deactivation energy and fast wakeup times. We show how this technique can be applied at a fine grain within an active microprocessor, and how microarchitectural scheduling policies can improve its performance. Using leakage-biased bitlines to deactivate SRAM read paths within I-cache memories saves over 24% of leakage energy and 22% of total I-cache energy when using a 70 nm process. In the register file, fine-grained read port deactivation saves nearly 50% of leakage energy and 22% of total energy. Independently, turning off idle register file subbanks saves over 67% of leakage energy (57% total register file energy) with no loss in performance.*

1. Introduction

Energy dissipation has emerged as a primary design constraint for all microprocessors, from those in portable devices to those in high-performance servers and mainframes. Until recently, the primary source of energy dissipation in digital CMOS circuits has been the dynamic switching of load capacitances. The continuing reduction in feature size reduces capacitance and the accompanying reductions in supply voltage help to further reduce the dynamic switching energy per operation. To maintain performance scaling, threshold voltages must also be scaled along with supply voltage. But lowering threshold voltage increases leakage current exponentially, and within a few process generations it is predicted energy dissipation from static leakage current could be comparable to dynamic switching energy [4, 5]. The trend toward ever more complex microprocessors further exacerbates the situation, as large numbers of transistors are added for relatively small improvements in performance. These additional transistors may dissipate considerable leakage power even when not actively switching.

We can divide previous approaches to reducing leakage power into two categories. Techniques that trade increased circuit delay for reduced leakage current include: conventional transistor sizing, lower Vdd [32, 30], stacked gates [25, 35, 9], longer channels [23], higher threshold voltages [19, 34, 21, 13, 1], and thicker T_{ox}; we collectively refer to these as statically-selected slow transistors (SSSTs). Techniques for dynamic run-time deactivation of fast transistors include body biasing [24, 17, 18, 20, 15], sleep transistors [24, 29, 13, 11, 16], and sleep vectors [35, 9]; we collectively refer to these as dynamically-deactivated fast transistors (DDFTs). SSSTs and DDFTs are complementary approaches: SSSTs reduce leakage on non-critical paths and DDFTs reduce leakage on critical paths. Both can be simultaneously applied to yield larger overall savings [12].

Although many leakage-reduction techniques are implemented at the circuit or device level, architects have considerable scope to influence processor leakage power [3]. One approach is to increase the use of SSSTs by finding additional parallelism, so that a given throughput can be achieved with a larger parallel array of units built with slower, less-leaky transistors, rather than with a smaller number of lower-latency units built with faster but leaky transistors. Unfortunately, available parallelism is limited in single-threaded general-purpose applications, and much of the complexity of modern microprocessors is due to the difficulties of finding such parallelism.

Alternatively, architects can focus on finding opportunities to exploit DDFTs, whereby fast, leaky circuits are deactivated when not required. This approach can potentially maintain the lowest latency for applications with little parallelism, while reducing leakage power to acceptable levels. The difficulty with this approach is that most existing circuit techniques for DDFTs are only effective at reducing leakage energy if a circuit block will be inactive for a long time. This limits the scope for applying DDFTs within an

active processor, where some blocks may only be inactive for a small number of cycles.

In this paper, we introduce a new DDFT circuit technique for reducing leakage power in memory arrays, *leakage-biased bitlines* (LBB). LBB uses leakage currents themselves to bias the bitlines of unused memory subbanks into a low-leakage state. LBB need have no performance impact and has very low transition energy overheads, and can convert even short idle times into leakage energy savings.

We apply LBB to the instruction cache and register file of an out-of-order superscalar microprocessor using predicted process parameters from 180 nm to 70 nm technology generations. For the instruction cache in 70 nm technology, we save 24% of leakage energy, or 22% of total energy, with a maximum performance penalty of 2.5%. For register files, we exploit idleness in two spatial dimensions: we deactivate subbanks when their registers are unused saving over 57% of register file energy. We also deactivate unused read ports when there is not enough parallelism to keep them busy. We save nearly 50% of leakage energy, or around 20% of total register file energy in this fashion, with no loss in performance and minimal area overhead.

This paper is organized as follows. Sections 2 and 3 review previous work in SSFT and DDFT techniques respectively. Section 4 presents the metrics we use to compare DDFT techniques. We show how some existing DDFT techniques require long idle times to be effective because of the large energy overhead of transitioning into a low-leakage state. Section 5 describes how we estimated the process parameters for future process technologies. Section 6 introduces leakage-biased bitlines and describes how we apply it to an instruction cache. Section 7 describes how we apply LBB to multiport register files. Section 8 discusses the results from our evaluation, and Section 9 concludes the paper.

2. Statically-Selected Slow Transistors

SSST techniques replace fast transistors with slow transistors on non-critical paths. This has been common design practice for many decades, where traditional transistor sizing reduces transistor gate width on non-critical paths to reduce parasitic load on critical nodes and to save switching power. Leakage is proportional to gate width, and so these narrower transistors also have lower leakage. Non-critical paths also use slower, more complex gate topologies to reduce area. These more complex gates have deeper transistor stacks, which also reduces leakage.

As leakage power increases, further techniques are being considered for non-critical paths. Leakage decreases superlinearly with gate length and a small increase in transistor length away from minimum can give a significant reduction in leakage current with a small impact on delay. Accordingly, the designers of the StrongARM-1 slightly lengthened cache and pad transistors to reduce leakage in standby mode, yielding a five-fold reduction in leakage with only a small performance penalty [23]. The Alpha 21164 used this approach to control the effects of leakage on dynamic gates [8]. Lengthening gates is only effective for a small increment in channel length, and has the disadvantage of increasing active power because of increased gate capacitance.

At the expense of additional mask processing steps, it is possible to manufacture transistors with several different threshold voltages on the same die. By using slower, high-threshold transistors on non-critical paths it is possible to reduce leakage current without impacting performance [34]. Even though most transistors are non-critical, the achievable leakage reduction is limited, because the non-critical transistors have already been reduced in width and stacked into complex gates and hence have low leakage.

3. Dynamically-Deactivated Fast Transistors

After application of SSST techniques to non-critical paths, leakage is even more highly concentrated in the critical path transistors. One example is a recent embedded PowerPC 750, which employs three threshold voltages: high, standard, and low. The low threshold transistors account for only 5% of the total transistor width, but around 50% of the total leakage [7]. Several techniques have been developed to reduce leakage current from transistors on the critical path. Unlike SSST techniques, where non-critical path transistors are made permanently slower to reduce leakage, DDFT techniques attempt to dynamically switch critical path transistors between fast, leaky, active operation and inactive low-leakage states.

One DDFT technique, popular in low-power processors for portable devices, is a dynamically varying body bias to modulate transistor threshold voltages [24, 29, 13, 11, 16]. Reverse body biasing, by setting the p-well voltage higher than Vdd and the n-well voltage lower than GND, increases V_T because of the body effect, thereby reducing leakage current. This technique requires twin or triple well processes and therefore increases manufacturing costs. A variation on the body biasing approach is to fabricate high-V_T transistors then actively *forward* bias the wells during normal operation to lower V_T [22]. In the idle state, the forward bias is removed returning the transistors to their natural high-V_T state. Other advantages of this technique are that it has less threshold variation than using low-V_T devices directly, and hence can allow higher speed operation for a given leakage current budget [22]. Because of the large capacitance and distributed resistance of the wells, charging or discharging the well has a relatively high time constant and dissipates considerable energy. To allow the latency and

energy costs of transitioning into the low leakage state to be amortized, these schemes are used when the processor enters a sleep state where it will be idle for at least 0.1–100 μs [28, 17, 30].

An alternative DDFT approach is power gating [24, 29, 13, 11, 16]. The power supply to circuits can be cut off by inserting a high V_T *sleep transistor* between Vdd and virtual Vdd (or GND and virtual GND). When turned off, the sleep transistor adds an extra high-V_T transistor in series with the logic transistors, dramatically reducing leakage current. Some of the disadvantages of sleep transistors are that they add additional impedance in the power supply network which reduces circuit speed, they require additional area and routing resources for the virtual power supply nets, and they may consume considerable deactivation energy to switch between active and inactive states. By sizing the sleep transistor [13], boosting the gate voltage for the sleep transistor [11], or forward-biasing the sleep transistor [16], the delay penalty can be reduced in exchange for greater sleep leakage currents and increased deactivation energy.

Another interesting DDFT technique exploits the fact that the leakage current of a block depends on the input pattern and internal state [35, 9]. A *sleep vector* is a combination of input patterns and internal state which minimizes the leakage current, and is applied by forcing internal latches into the correct state and forcing inputs to the correct polarity. However, the application of the sleep vector can require additional circuitry, which reduces performance, and can cause spurious circuit switching, which results in significant deactivation energy.

All DDFT circuits require a policy to decide when to switch to a low-leakage mode. Current microprocessors use a simple policy, usually implemented by the operating system, whereby the entire processor is deactivated when it enters a sleep mode. This coarse-grain policy cannot reduce active mode leakage power.

A few researchers have proposed more fine-grained deactivation techniques that place portions of an active processor into low-leakage states. The dynamically-resized instruction cache [26] uses a virtual-GND power gate to supply power to just enough RAM subbanks to hold the active working set of the current application. An adaptive hardware algorithm is used to determine an adequate cache capacity by monitoring miss rates as the active partition size is varied. This scheme is more complex than using leakage-biased bitlines, and is limited to a direct-mapped instruction cache, but reduces leakage further as both storage cell and access port leakage is cut off. Cache decay [14] dynamically predicts which cache blocks are unlikely to be accessed in the near future, marks them invalid, then powers them down using a power gate. Both of these techniques have long deactivation times of thousands of cycles. Hamzaoglu et al. briefly describe a "precharge-as-needed" scheme [10], apparently similar to our leakage-biased bitlines, but do not describe the dynamic transient effects of the leakage reduction or the use of this technique within a microprocessor. Zhang et al. [36] explored the use of compiler-controlled dynamic leakage reduction mechanisms in a VLIW processor using sleep vectors and sleep transistors.

For more general application of DDFT techniques within an active microprocessor, it is necessary to have circuit techniques that make it worthwhile to deactivate a circuit block for short periods of time, and microarchitectural mechanisms that can detect, or force, a block into an idle state.

4. Comparing DDFT Techniques

The goal of applying a fine-grain DDFT technique is to reduce total processor energy. When attempting to deactivate a block for a short period of time, the performance and energy impacts of entering and leaving the low-leakage state must be considered. Figure 1 introduces the different parameters we use to compare DDFT techniques.

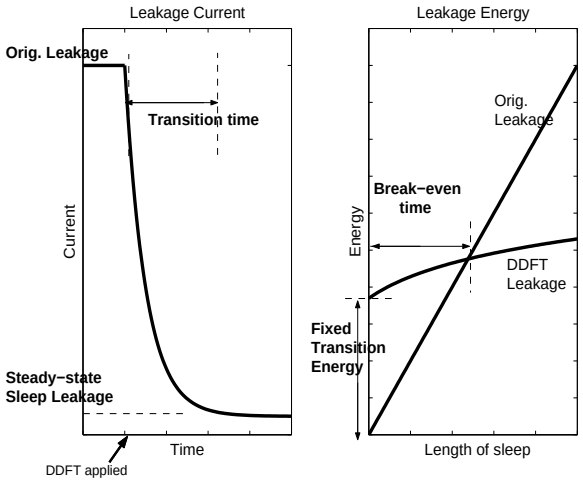

Figure 1. Transition time, steady-state leakage current, and break-even time of DDFT leakage-reduction techniques.

The left-hand side of Figure 1 shows the evolution of leakage current over time on entering the deactivated state. Once deactivated, a block requires some time to reach the lowest leakage state. For example, a substrate biasing scheme will require time to bias the wells, and a virtual-GND scheme requires time for leakage currents to charge up the virtual-GND node. During the transition, leakage current can be substantially higher than in the steady state. For clarity, this graph only shows the leakage current. When

switching into and out of the low-leakage state there can be substantial switching current spikes.

Switching between active and deactivated modes requires additional transition energy, for example, to switch the gates of power-gating transistors or to charge and discharge well capacitances. The right-hand side of Figure 1 illustrates how we compare the overall energy consumed over time when idling in a normal, high-leakage state versus transitioning into a low-leakage state. The original idle leakage energy is shown by the straight line which rises at a constant rate dependent on the leakage current. On the same graph, we show an example curve for a DDFT technique. The fixed energy costs of first moving to the low-leakage state, then moving back to the active state, are summed to give the fixed transition energy cost. In addition to the fixed transition energy, there may be additional variable transition energy costs proportional to the time that the block is deactivated. For example, in a virtual GND scheme, the virtual GND node is slowly charged over the transition time. The amount of energy dissipated when the block wakes up and discharges the virtual GND depends on the idle time. These variable transition energy costs are factored into the energy curve. The curve rises more steeply initially during the transition time, where variable transition energy costs are being incurred and as leakage current drops to its steady state value. After the transition time, the energy curve rises more slowly, as only the steady-state leakage current is being dissipated.

We define the break-even time as the time at which the two curves cross, i.e., when the leakage energy of remaining in an active idle state matches the energy consumed when switching to the DDFT low-leakage state. The circuit must be idle for considerably longer than the break-even time to save significant energy.

Another important factor in comparing DDFT techniques is the wakeup latency (not shown). The wakeup latency is the time for a block to become usable after being in an inactive state. Faster wakeup time is usually preferable to faster transition time because it reduces any performance penalty. Wakeup latency can sometimes be traded for transition energy, for example, using a wider transistor to accelerate discharge of a biased well increases the transition energy to switch the transistor.

Although DDFT techniques do not use slower transistors to reduce leakage power, some DDFT techniques affect the delay and power of the active state. For example, the NMOS sleep transistor technique causes virtual GND to be a slightly higher potential than GND and so the circuit is somewhat slower.

5. Process Technologies

To evaluate our DDFT techniques, we used models of four dual-V_T processes, including 180 nm, 130 nm, 100 nm, and 70 nm process generations. The 180 nm high-V_T and low-V_T transistors were modeled after 0.18 μm TSMC low-leakage and medium-V_T processes respectively. The parameters of the 180 nm process were scaled to future technologies using the SIA roadmap [6]. For example, the SIA roadmap predicts that I_{on} remains the same, but I_{off} jumps twice for each technology generation. Because of the difficulty in predicting future leakage numbers, we bracket our results using our own pessimistic and optimistic estimates of how leakage currents will scale. The pessimistic estimates assume $4\times$ leakage increase per generation while the optimistic estimates assume $2\times$ leakage increase per generation. Important parameters of the processes are summarized in Table 1. We only considered subthreshold leakage in our estimates; although gate leakage might become significant at some point in these technology generations, it is also likely that new gate dielectrics will make gate leakage insignificant again. We believe future leakage currents might be considerably higher that even our pessimistic numbers indicate, as these are based on a low-leakage, moderate-performance base case.

Table 1. Process parameters.

Parameter (nm)	180	130	100	70
Vdd (V)	1.8	1.5	1.2	0.9
Temp (Celsius)	100	100	100	100
FO4 delay (ps)	61.1	47.4	36.7	24.0
16 FO4 freq. (GHz)	1.0	1.3	1.7	2.6
LVT I_{on} (μA/μm)	732	732	732	732
LVT I_{off} (nA/μm) (optimistic)	21.8	43.6	87.2	174
LVT I_{off}(nA/μm) (pessimistic)	21.8	87.2	349	1395
HVT I_{on} (μA/μm)	554	554	554	554
HVT I_{off} (nA/μm) (optimistic)	0.35	0.71	1.42	2.83
HVT I_{off} (nA/μm) (pessimistic)	0.35	1.42	5.68	22.6

Based on the table, we estimated the scaling of active and leakage power for circuits. The results are shown in Figure 2, where numbers are normalized to the 180 nm process. It is important to note that leakage power per transistor increases significantly although Vdd and the total area of the circuit decreases. The active power is decreasing quadratically as expected from constant field scaling. If the leakage power was 10% of the total power at the 180 nm process, it will increase to 47-87% for the 70 nm process if the circuit is scaled unchanged. In practice, devices, circuits, and microarchitectures will be redesigned to limit leakage to a manageable fraction of total power. The techniques in this paper can be used to help keep leakage current within this budget without sacrificing performance.

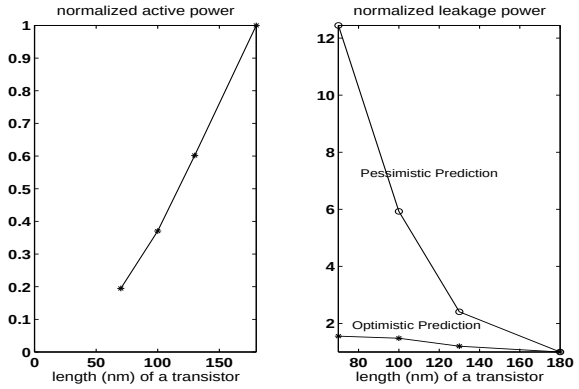

Figure 2. Normalized active and leakage power for different processes.

6. Leakage-Biased Bitlines for Caches

The L1 caches of microprocessor can cause significant leakage current, as they contain a large number of transistors which must be high-speed to avoid impacting processor performance. Figure 3 shows the structure of an L1 cache SRAM cell together with the two primary leakage current paths when the word line is off. One leakage path, L_1, is from the precharged bit-line, through the access transistor, and across the turned-on n-type pull-down. The other leakage path, L_2, is from the enabled p-type pullup to the turned-off n-type in the cross-coupled inverters. The pullup transistors have been made high-V_T so that there is negligible leakage current across the turned-off p-type ($L_3 \simeq 0$), and the access and pulldown transistors have been made low-V_T to maintain circuit speed. The current through the leakage path, L_4, is insignificant since the path has two turned-off transistors and the V_{DS} of the access transistor is zero.

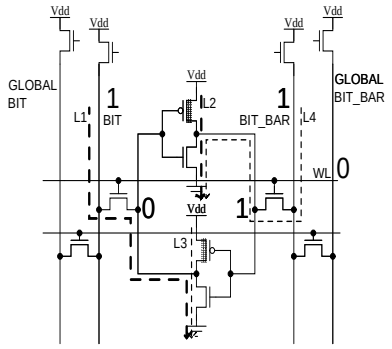

Figure 3. A dual-V_T SRAM cell. High V_T transistors are shaded.

With technology scaling, the leakage currents from non-accessed bits will reduce the effective signal from the accessed bit, requiring that SRAMs have fewer cells connected to each bitline segment to obtain sufficient noise margin. We assume that only 32 bit cells are attached to each local bitline within a subbank, and that these local bitlines are connected through pass-transistor switches to a global bitline which in turn connects to the senseamp.

A key observation is that the leakage current, L_1, from each bitline into the cell depends on the stored value on that side of the cell; there is effectively no leakage if the bitline is at the same value as that stored in the cell (L_4). We might consider using a sleep vector on the bitlines to force the SRAM subbank into a low leakage state. For example, it is known that there are usually more zeros than ones stored in a cache [33], so if we force the true bitline to a zero value while keeping the complement bitline precharged, we could statistically reduce the bitline leakage of an inactive cache subbank. There are two disadvantages to this approach. First, if the percentage of zero bits is under 50%, the sleep vector technique *increases* leakage energy. Second, this technique requires additional transition energy to force the bitlines into and out of the sleep vector state.

We have developed a simple circuit technique, *leakage-biased bitlines* (LBB), that reduces bitline leakage current due to the access transistors of these structures with minimal transition energy and wakeup time. Rather than force zero sleep values onto the read bit lines of inactive subbanks, this technique just lets the bitlines float by turning off the high-V_T NMOS precharging transistors. The leakage currents from the bit cells *automatically* bias the bitline to a mid-rail voltage that minimizes the bitline leakage current. If all the cells store a zero, the leakage currents will fully discharge the non-inverted bitline ("BIT" in Figure 3) while the inverted bitline ("BITBAR") will be held high. If all the cells store a one, the non-inverted bitline will be held high and the inverted bitline will discharge. For a mix of ones and zeros, the leakage currents bias the bitline at an appropriate midrail voltage to minimize leakage. Although the bitline floats to mid-rail, it is disconnected from the senseamp by the local-global bitline switch, so there is no static current draw. This technique has little additional transition energy because the precharge transistor switches exactly the same number of times as in a conventional SRAM—we only delay the precharge until the subbank needs to be accessed. The wakeup latency is just that of the precharge phase.

Figure 4 compares the steady-state leakage power of the leakage-biased bitline and the forced-zero/forced-one sleep vector techniques with the original leakage power for a 32-row×16B SRAM subarray with varying numbers of stored ones and zeros. It is clear that the leakage-biased bitline technique has the lowest leakage power independent of stored bit values.

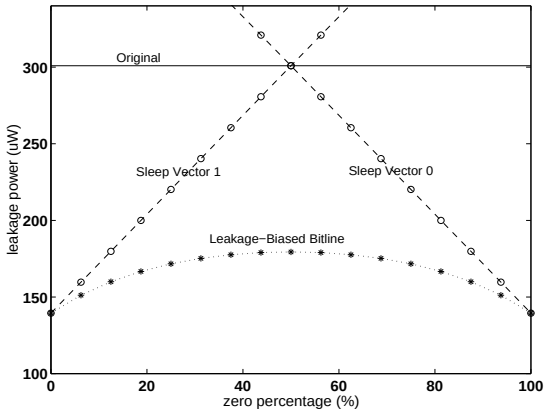

Figure 4. The leakage power of 32-row×16B SRAM subbank for forced-zero and forced-one sleep vectors and leakage-biased bitlines versus percentage of stored zero bits.

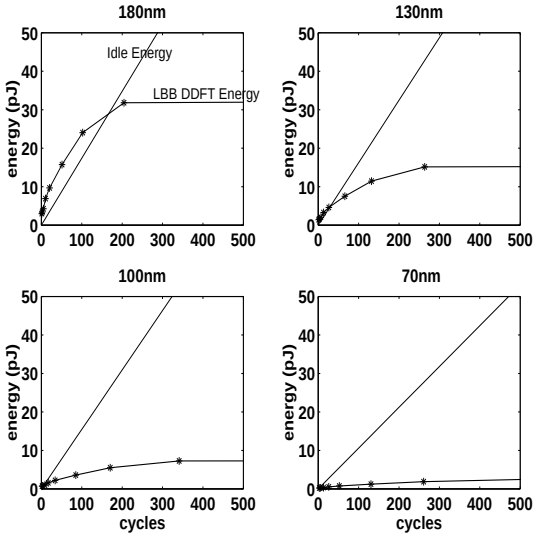

Figure 5. Idle energy and LBB DDFT energy of 32-row×32B SRAM subbank for different processes (optimistic leakage current was used).

Figure 5 compares the cumulative idle energy and the LBB DDFT energy consumption for different processes. The LBB DDFT technique must replace the lost charge on the bitline before the attached memory cells can be used. The break-even time, is around 200 cycles in a 180 nm process. However, since active energy scales down faster than leakage energy, the break-even time decreases with feature size. In a 70 nm process, the break-even time is less than one cycle.

Each subbank must be precharged before use, which will add latency to the cache access if the subbank is not known in time. We focus in this paper on the application of LBB DFFT technique to the processor instruction cache, because of its predictable access pattern. For an N-way set-associative cache structure, each way consists of some number of subbanks and we access N subbanks in parallel, where each subbank returns a fetch group of instructions. In the most optimistic case, we can assume that the simple subbank decode happens sufficiently before the more complex word-line decode to allow precharge to complete before word-line drive; in this case, there would be no performance penalty. In the most pessimistic case, we can assume that the additional precharge latency adds an additional cycle to the fetch pipeline, and hence increases the branch misprediction penalty by one cycle.

7. LBB for Multiport Register Files

Multiport register files can also consume considerable leakage power. For example, in the proposed Alpha 21464 design, the multiported register file was several times larger than the 64 KB primary caches [27]. Figure 6 shows an 8-read port, 4-write port, register file cell. Because there are many leakage paths in a multiport register file cell, we chose a baseline design that was already optimized for leakage power. The cell has a high-V_T storage cell connected to multiple low-V_T single-ended read ports. The write ports are not as latency critical and so these access transistors are high-V_T. To reduce active and leakage energy further, we make the cell asymmetrical, with all read ports arranged so that if the cell stores a zero, the single-ended bitline is not discharged [31]. Our experiments showed that around 75% of the bits read from the register file are zero.

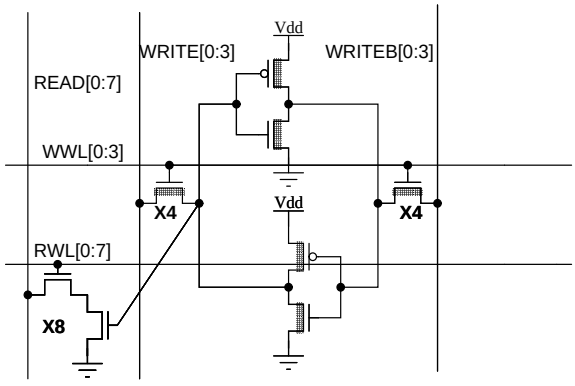

Figure 6. An embedded dual V_T unbalanced 8-read, 4-write register file cell. High V_T transistors are shaded.

As with the cache SRAM, the register file array is divided into subbanks with local bitlines connected to global bitlines to save switching energy and to increase speed and noise margin. The LBB technique can be applied to the single-ended read port bitlines. By turning off the precharger on an idle subbank read port, leakage currents will discharge the bitlines towards ground if any bits are holding a one, reducing bit line leakage current significantly. If the dead time is long enough, the energy overhead to precharge the bitline before an access becomes relatively small compared to the leakage energy saved. Note that this technique does not corrupt the state stored in the register file. Figure 7 shows the hierarchical bitlines and a modified column cell for the LBB multiported register file.

We deactivate read ports using two orthogonal techniques. The first deactivates dead registers whose contents are not needed. We exploit the fact that in a superscalar machine with register renaming, the contents of a physical register are not needed from the time the register enters the free list until the time it is next written. If all the registers in a subbank are dead, then all subbank read ports can be turned off. Because the register is allocated in the decode stage of the pipeline and written to several cycles later (and before any read access), there is ample time to precharge the floating bitline with no performance impact.

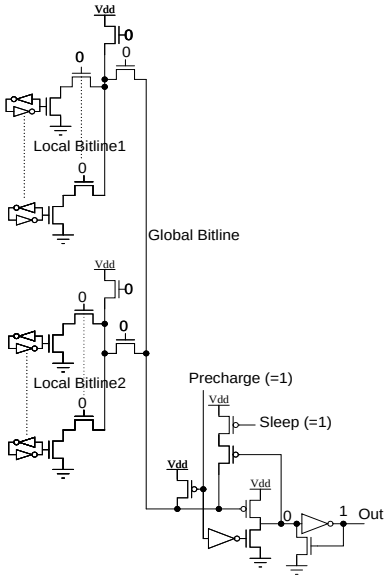

Figure 7. Leakage-biased bitline scheme for multiported register file. Each local bitline can be left unprecharged, biased by local leakage currents.

As a comparison, we considered an alternative DDFT approach to turn off dead registers using a virtual-GND sleep transistor (Figure 8). This approach has the advantage that registers can be turned off individually, rather than a subbank at a time, but has the disadvantage that the read access time increases due to the sleep transistor in the pull-down path. The delay penalty can be reduced by increasing the size of the sleep transistor, but this also increases the steady-state leakage current and the transition energy. We sized the sleep transistor to give an overall 5% slowdown.

The second technique deactivates idle read ports. In a superscalar machine, when fewer than the maximum number of instructions issue, some register file read ports will be idle. There is no performance impact when the port is reactivated because it is known whether a read port is needed before it is known which register will be accessed in the pipeline. The port precharge time can be overlapped with register file address decode.

Table 2 shows the energy consumption when reading/writing 32-bit zeros or ones from the 32×32-bit register file with the unbalanced embedded dual V_T cells. All read/write energy numbers are per single read/write port. The energy consumption for 180 nm was measured using Hspice simulation and those for other processes were scaled using Figure 2. The average read and write energy numbers were calculated assuming 75% of values stored in the register files and write data are zero and that the values are statistically independent. The total active energy consumption is simply the sum of total read energy and total write energy.

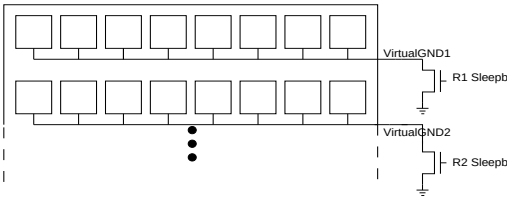

Figure 8. Idle register deactivation scheme using NMOS high V_T sleep transistors.

Table 3 shows the steady-state leakage power when different leakage techniques are applied to the register file and the idle leakage power of the original circuit for different processes. We again assumed 75% of the bits in the register file are zeros when measuring the leakage power. We also include numbers for the sleep vector fixed-zeros technique. All three techniques, sleep vector (SV), leakage-biased bitline (LBB), and NMOS sleep transistor (NST) reduce the leakage power to less than 1.5% of the original idle power when in the steady state.

Figure 9 shows the sleep-time dependent energy consumption of the register file DDFT techniques across the set of process technologies. We can see that all of the DDFT techniques become applicable at shorter time scales as transistors scale down. This is partly because leakage current

Table 2. The active read and write energy consumption of 32×32b multiported register file subbank for different processes.

tr. length(nm)	180	130	100	70
zero read E(pJ)	6.0	2.9	1.4	0.5
one read E(pJ)	17.3	8.2	4.0	1.4
avg. read E(pJ) (E_r)	8.8	4.2	2.0	0.7
0-to-0 write E(pJ)	0.7	0.4	0.2	0.1
0-to-1 write E(pJ)	16.5	7.9	3.8	1.3
1-to-0 write E(pJ)	2.2	1.0	0.5	0.2
1-to-1 write E(pJ)	13.0	6.2	3.0	1.0
avg. write E(pJ) (E_w)	4.7	2.3	1.1	0.4

Table 3. The leakage power of 32×32-b multiported register file subbank (optimistic leakage current was used).

Process Tech. (nm)	180	130	100	70
Original (uW)	177.9	214.1	263.6	276.7
SV steady-state (uW)	2.0	2.4	3.0	3.1
LBB steady-state (uW)	2.0	2.4	3.0	3.1
NST steady-state (uW)	1.8	2.2	2.7	2.9

grows as a fraction of active power, but also partly because most of the transition energy cost scales with active power and so the relative overhead of switching is reduced. Figure 10 is an expanded view of the graph for the 70 nm process technology.

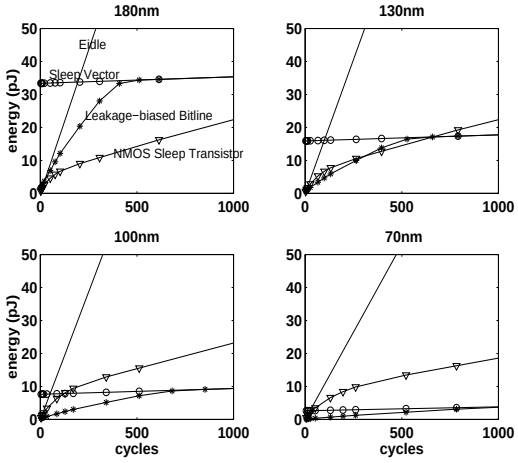

Figure 9. Sleep-time-dependent cumulative leakage energy of different register file DDFT techniques for different processes (optimistic leakage current was used).

We see that for the sleep vector technique, the break-even time is around 200 cycles at the 180 nm process, but shrinks to only 24 cycles in the 70 nm process. The sleep vector technique has high fixed transition energy costs, and so below the break-even time, the energy consumption is much higher than the original leakage energy.

For the leakage-biased bitline, the break-even time in the 180 nm process is only around 10 cycles. Moreover, the cumulative energy rises slowly from the initial deactivation time, and is not much larger than the original leakage before the break-even time. With technology scaling, the break-even time becomes less than a cycle and this technique can therefore give useful leakage energy savings even for a few cycles of dead time.

The NMOS sleep-transistor performs better than the leakage-biased bitlines in the coarser feature sizes, but suffers from a long transition time in the finer-pitch process technologies. The time taken to charge the virtual GND node leaves this scheme with higher cumulative leakage energy for small numbers of cycles in the 70 nm technology, though at large numbers of cycles the cumulative energy drops below that of the leakage-biased bitline scheme.

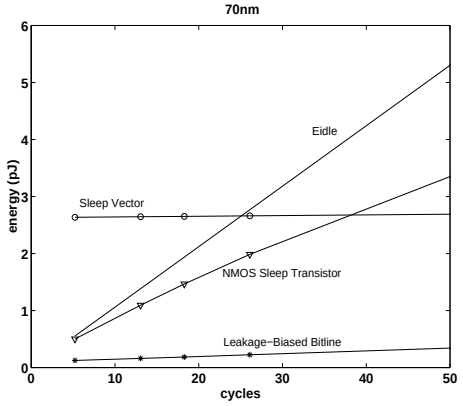

Figure 10. Expanded view of cumulative leakage energy in 70 nm process technology (optimistic leakage current was used).

8. Evaluation

In this section, we use detailed simulation of an out-of-order processor to estimate the energy savings that can be achieved by using DDFT techniques on instruction cache subbanks and a multiported register file.

8.1. Simulation Methodology

We instrumented SimpleScalar 3.0b [2], an out-of-order, superscalar processor simulator, to track the activity of the

Table 4. Simulated Processor Configuration

Issue Width	4
RUUs	64
Integer Physical Registers	100
Integer ALUs (Mult/Div)	4 (1)
FP ALUs (Mult/Div)	1 (1)
Load/Store Units	2
Load/Store Queue Depth	32
Instruction length	4 Bytes
I-Cache/D-Cache	16KB/4-Way/32B Block
Unified L2-Cache	256KB/4-Way/64B Block 6 cycle latency
Memory Latency	First access: 50 cycs. Subsequently: 2 cycs.

instruction cache and the physical register file. We obtained results for a four-wide issue machine with the configuration shown in Table 4. We also simulated a four-wide issue machine with 128 RUUs and performed cache simulation on an eight-wide issue machine with 256 RUUs, but the results were similar and thus we omit them for brevity. We used the SPECint95 benchmark suite and the benchmarks were run on their reference data sets until 100 million instructions had committed. In the figures that follow, black bars denote the optimistic assumptions of future leakage as described in Section 5. White bars denote the pessimistic view of the future (greater leakage).

8.2. Cache Subbank Deactivation Results

Figure 11 shows the energy savings achieved for the instruction cache subbank deactivation scheme. For the 180 nm generation, there is a net energy increase, but for all other process technologies there is a net energy savings. In the 70 nm generation, over 20% of total instruction cache energy is saved.

As discussed in Section 6 there can be a performance penalty from the additional precharge latency if the subbank precharge cannot be overlapped with the rest of the bank address decode. We modeled the effect of lengthening the fetch pipeline by one cycle to allow for subbank precharge, which increases branch misprediction latency from 3 to 4 cycles. Our results show that this decreases IPC by around 2.5% on average across all benchmarks. We note that this estimate of performance impact is highly pessimistic, as the precharge latency is much less than one cycle and the extended pipeline could be used to support a much larger instruction cache.

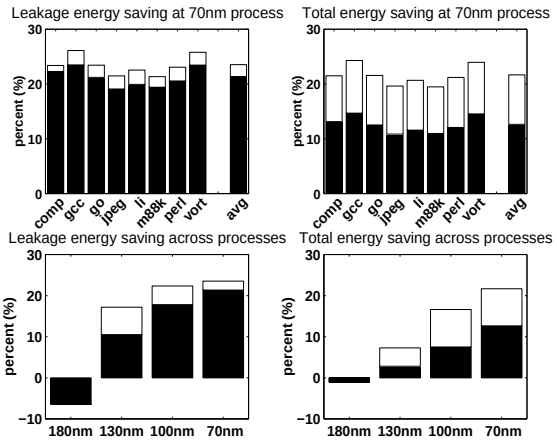

Figure 11. I-cache energy savings for subbank deactivation.

8.3. Dead Register Deactivation Results

To quantify energy saved by deactivating dead registers, we modified SimpleScalar to model a machine with a separate unified physical register file pool holding both committed architectural registers and renamed registers. We maintain a set of physical register tags which move between a free list and the register update units. We restricted our study to the integer register file. The number of physical registers is determined by the number of writable architected registers (fixed by the ISA at 33) plus the number of values that can be produced by in-flight instructions.

Figure 12 presents results for the dead register deactivation techniques. For reference, we present two variants of the NMOS sleep transistor (NST) technique. The two variants of NST are FIFO and LIFO free list policies. A FIFO policy (or circular queue) is the conventional free list policy, but a LIFO policy (stack) has the advantage of keeping some registers dead for very long times. Experiments with the 70 nm process reveal that LIFO gives an additional 2.4 to 10.0% savings over FIFO in terms of total register file energy saved. The figure also shows the benefits of LIFO increasing as feature size decreases.

We also show results for LBBs used in a subbanked register file, where a subbank's read ports are deactivated when all registers in the subbank are dead. The allocation policy is a stack of subbanks, where registers are allocated from a new bank only when the previous bank is empty. As shown in Figure 12, despite the increased granularity of deactivation, LBB is competitive with NST in terms of energy savings. Because the cumulative sleep energy of LBB circuits is less than that of NST circuits for the majority of sleep times encountered in practice, LBB outperforms NST by 31.5% for the worst case leakage. In addition, LBB has no

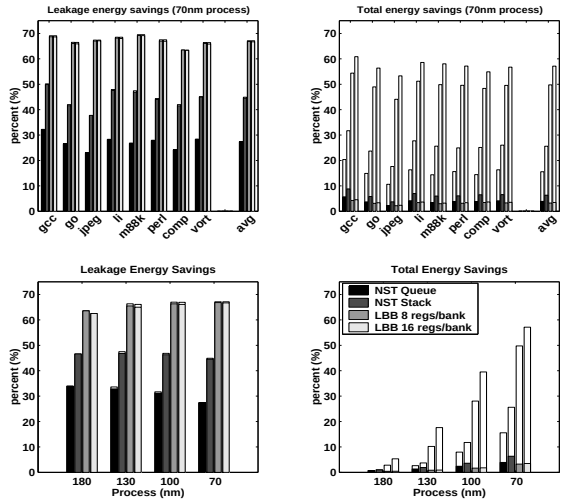

Figure 12. Register file energy savings by dead register deactivation.

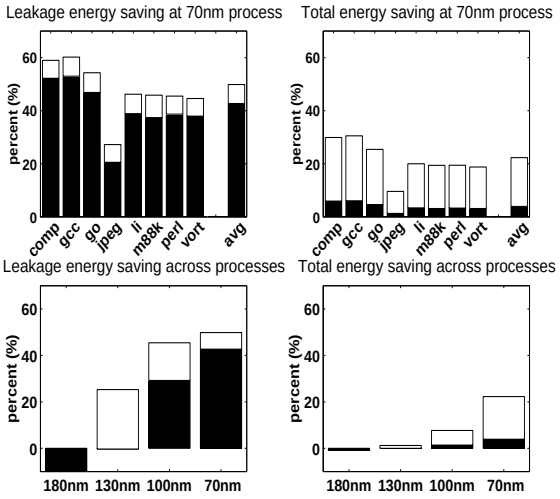

Figure 13. Register file energy savings by global read port deactivation.

performance penalty, whereas NST has a 5% performance slowdown. The figure also shows that having fewer registers per bank (eight rather than sixteen) allows deactivation at a finer granularity which translates to greater savings.

8.4. Regfile Global Read Port Deactivation Results

Figure 13 shows the energy savings achieved by deactivating the read ports. As with cache subbank deactivation, there is a net energy increase for the 180 nm generation, but for the remaining process technologies, there is a net energy savings. In the 70 nm generation, nearly half of the leakage energy is removed, resulting in a total register file energy savings of over 20% with no performance penalty.

As the processor's issue width increases, the peak number of read ports increases. However, IPC does not scale linearly with issue width, so in general a greater percentage of read ports will be idle. Thus, we expect the energy savings to be greater for wider-issue processors.

The savings from global read port deactivation can be combined with those from dead subbank deactivation, giving greater total energy savings while still avoiding any performance penalty.

9. Conclusion

Most leakage current is dissipated on critical paths, especially after slower, low-leakage transistors are used on non-critical paths. To reduce leakage energy further without impacting performance, it is desirable to dynamically deactivate the fast transistors on the critical path. This paper has shown that fine-grain leakage reduction techniques, whereby a small piece of a processor is placed in a low-leakage state for a short amount of time, can yield significant energy savings in future process technologies. To attain savings, the circuit-level leakage reduction technique must have low transition energy and rapid wakeup times. We present leakage-biased bitlines, a circuit technique that has these properties. To exploit a DDFT technique, the microarchitecture must be designed to force blocks to be idle for multiple cycles and preferably to give early notice when the blocks are to be reawakened.

We have presented three applications of leakage-biased bitlines that apply these principles and have shown how they enable leakage current reductions in the context of a wide superscalar processor. SRAM read path deactivation saves over 22% of leakage energy and nearly 24% of total I-cache energy when using a 70 nm process. Dynamically deactivating idle registers reduces register file leakage energy by up to 67.1% and total register file energy by 57.1%. Dynamically deactivating read ports within a multiported register file saves 42.7-49.8% of leakage energy and 3.9-22.3% of total energy depending on the prediction of the future process. We are investigating further circuit techniques of this type for other components of a superscalar microprocessor.

10. Acknowledgments

We'd like to thank Christopher Batten, Ronny Krashinsky, Rajesh Kumar, and the anonymous reviewers for comments on earlier drafts. This work was funded by DARPA PAC/C award F30602-00-2-0562, NSF CAREER award CCR-0093354, and a donation from Infineon Technologies.

References

[1] M. Allarm, M. H. Anis, and M. I. Elmasry. High-speed dynamic logic styles for scaled-down CMOS and MTCMOS technologies. In *ISLPED*, pages 155–160, 2000.

[2] D. Burger and T. Austin. The SimpleScalar tool set, version 2.0. Technical Report CS-TR-97-1342, University of Wisconsin, Madison, June 1997.

[3] J. A. Butts and G. S. Sohi. A static power model for architects. In *MICRO-33*, pages 191–201, December 2000.

[4] A. Chandrakasan, W. J. Bowhill, and F. Fox. *Design of High Performance Microprocessor Circuits*. IEEE Press, 2000.

[5] V. De and S. Borkar. Technology and design challenges for low power and high performance. In *ISLPED*, pages 163–168, 1999.

[6] I. T. R. for Semiconductors. 2000 update, process integration, devices, and structures. Technical report, ITRS, 2000.

[7] S. Geissler et al. A low-power RISC microprocessor using dual PLLs in a 0.13 μm SOI technology with copper interconnect and low-k BEOL dielectric. In *ISSCC Digest*, February 2002.

[8] P. E. Gronowski et al. A 433-MHz 64-b quad-issue RISC microprocessor. *IEEE JSSC*, 31(11):1687–1696, November 1996.

[9] J. P. Halter and F. Najm. A gate-level leakage power reduction method for ultra-low-power CMOS circuis. In *Custom Integrated Circuits Conf.*, pages 457–478, 1997.

[10] F. Hamzaoglu et al. Dual-vt SRAM cells with full-swing single-ended bit line sensing for high-performance on-chip cache in 0.13 μm technology generation. In *ISLPED*, pages 15 – 19, 2000.

[11] T. Inukai. Boosted-gate MOS (BGMOS): Device/circuit cooperation scheme to achieve leakage-free giga-scale integration. In *Custom Integrated Circuits Conf.*, pages 409–412, 2000.

[12] M. C. Johnson, D. Somasekhar, L.-Y. Chiou, and K. Roy. Leakage control with efficient use of transistor stacks in single threshold cmos. In *IEEE Transactions on VLSI Systems*, February 2002.

[13] J. T. Kao and A. P. Chandrakasan. Dual-threshold voltage techniques for low-power digital circuits. *IEEE JSSC*, 35(7):1009–1018, July 2000.

[14] S. Kaxiras, Z. Hu, and M. Martonosi. Cache decay: exploiting generational behavior to reduce cache leakage power. In *ISCA 28*, pages 240–251, May 2001.

[15] A. Keshavarzi et al. Effectiveness of reverse body bias for leakage control in scaled dual Vt CMOS ICs. In *ISLPED*, pages 207–212, 2001.

[16] S. V. Kosonocky et al. Enhanced multi-threshold (MTCMOS) circuits using variable well bias. In *ISLPED*, pages 165–169, 2001.

[17] T. Kuroda et al. A 0.9-V, 150-MHz, 10-mW, 4 mm^2, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme. *IEEE JSSC*, 31(11):1770–1779, November 1996.

[18] T. Kuroda et al. Variable supply-voltage scheme for low-power high-speed CMOS digital design. *IEEE JSSC*, 33(3):454–462, March 1998.

[19] W. Lee et al. A 1-V programmable DSP for wireless communications. *IEEE JSSC*, 32(11):1766–1776, November 1997.

[20] H. Makino et al. An auto-backgate-controlled MT-CMOS circuit. In *Symp. on VLSI Circuits*, pages 42–43, 1998.

[21] T. McPherson et al. 760MHz G6 S/390 microprocessor exploiting multiple Vt and copper interconnects. In *ISSCC Digest*, pages 96–97, 2000.

[22] M. Miyazaki et al. A 1000-MIPS/W microprocessor using speed-adaptive threshold-voltage CMOS with forward bias. In *ISSCC Digest*, pages 420–421, 2000.

[23] J. Montanaro et al. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE JSSC*, 31(11):1703–1714, November 1996.

[24] S. Mutoh et al. 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS. *IEEE JSSC*, 30(8):847–854, August 1995.

[25] S. Narendra et al. Scaling of stack effect and its application for leakage reduction. In *ISLPED*, pages 195–200, 2001.

[26] M. Powell et al. Gated Vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *ISLPED*, 2000.

[27] R. P. Preston et al. Design of an 8-wide superscalar RISC microprocessor with simultaneous multithreading. In *ISSCC Digest and Visuals Supplement*, February 2002.

[28] K. Seta et al. 50% active-power saving without speed degradation using standby power reduction (SPR) circuit. In *ISSCC Digest*, pages 318–319, 1995.

[29] S. Shigemasu et al. A 1-V high-speed MTCMOS circuit scheme for power-down application circuits. *IEEE JSSC*, 32(6):861–869, June 1997.

[30] M. Takahasi et al. A 60-mw MPEG4 video codec using clustered voltage scaling with variable supply-voltage scheme. *IEEE JSSC*, 33(11):1772–1778, November 1998.

[31] J. Tseng and K. Asanović. Energy-efficient register access. In *Proc. of the 13th Symposium on Integrated Circuits and Systems Design*, Manaus, Brazil, September 2000.

[32] K. Usami et al. Automated low-power technique exploting multiple supply voltages applied to a media processor. *IEEE JSSC*, 33(3):463–471, March 1998.

[33] L. Villa, M. Zhang, and K. Asanović. Dynamic zero compression for cache energy reduction. In *MICRO-33*, 2000.

[34] L. Wei et al. Design and optimization of low voltage high performance dual threshold CMOS circuits. In *DAC*, pages 489–494, 1998.

[35] Y. Ye, S. Borkar, and V. De. A technique for standby leakage reduction in high-performance circuits. In *Symp. on VLSI Circuits*, pages 40–41, 1998.

[36] W. Zhang et al. Exploiting VLIW schedule slacks for dynamic and leakage energy reduction. In *MICRO-34*, December 2001.