Fine-Grain Dynamic Leakage Reduction*

Seongmoo Heo, Ken Barr, Mark Hampton, and Krste Asanović
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
{heomoo,kbarr,mhampton,krste}@lcs.mit.edu

ABSTRACT

Previous work in leakage current reduction for digital circuits can be divided into two main categories: static design-time selection of slow, low-leakage transistors for non-critical paths and dynamic deactivation of fast leaky transistors on critical paths. Leakage power is dominated by critical paths, and hence dynamic deactivation of fast transistors could potentially yield large savings. We introduce methodologies for comparing fine-grain dynamic deactivation techniques that include the effects of deactivation energy and startup latencies, as well as long-term leakage current. Existing dynamic leakage reduction techniques, although they have low leakage current also have large deactivation energies and significant startup latencies. The large deactivation energies require long idle times to amortize their overhead, and large startup latencies impact performance, limiting the applicability of these techniques within an active microprocessor. We introduce new circuit techniques that have a low deactivation energy when transitioning a circuit block into a low leakage state from which it can be woken quickly. We show how these techniques can be applied at a fine grain within an active microprocessor, and how microarchitectural scheduling policies can improve their performance. The first technique deactivates SRAM read paths within I-cache memories saving over 40% of idle circuit leakage energy and over 20% of total I-cache energy when using a 70 nm process. The second technique dynamically deactivates idle registers reducing idle circuit leakage energy by 41.1% and up to 12.4% of total regfile energy. The third technique dynamically deactivates read ports within a multiported register file. Independent of the second technique, read port deactivation saves up to 98.5% of idle circuit leakage energy and 47.8% of total energy.

1. INTRODUCTION

Energy dissipation has emerged as the primary design constraint for all microprocessors, from those used in portable devices to those used in high-performance servers and mainframes. Until recently, the primary source of energy dissipation in digital CMOS circuits has been the dynamic charging and discharging 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 down along with supply voltage. But lowering the threshold voltage increases static 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 [1, 4]. The trend towards 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.

A number of techniques have been proposed to combat this increase in leakage power. We divide these approaches into two categories depending on whether they focus on the static design-time selection of slow transistors, or the dynamic run-time deactivation of fast transistors. Techniques that trade increased circuit delay for reduced leakage current include: conventional transistor sizing, lower Vdd [10, 14], stacked gates [18, 32, 26], longer channels [8], higher threshold voltages [24, 11, 23, 28, 30], 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 [17, 21, 22, 7, 5], sleep transistors [17, 19, 28, 27, 20], and sleep vectors [32, 26]; we collectively refer to these as dynamically-deactivated fast transistors (DDFTs). SSSTs and DDFTs are complementary leakage reduction techniques where SSSTs reduce leakage on non-critical paths and DDFTs reduce leakage on critical paths, and both can be simultaneously applied to yield larger overall savings.

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. Moreover, if any additional parallelism is found, this could instead have been used to attain the same throughput at a reduced clock rate and supply voltage, saving both active and leakage power.

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 challenges in this approach are 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.

*This work was partly funded by DARPA PAC/C award F30602-00-2-0562 and by NSF CAREER award CCR-0093354.

In this paper, we introduce a new methodology for comparing DDFT techniques and show how some existing techniques require long idle times to be effective at reducing energy because of the large energy overhead of transitioning into a low-leakage state. We then present new circuit-level techniques for caches and register files that have very low energy overhead to transition into a low-leakage state thereby enabling even short idle times to translate into energy savings. We use predicted process parameters from 180 nm to 70 nm technology generations to estimate energy savings that can be achieved by applying these techniques to an out-of-order superscalar microprocessor architecture.

For caches, we deactivate the access ports on unused cache sub-banks to save bitline leakage energy. For the instruction cache, we save 40% of leakage energy, or 20% of total energy, with only a 3% performance penalty. For register files, we exploit idleness in two spatial dimensions: we deactivate individual dead registers, and we also deactivate unused access ports. The physical registers inside a superscalar microprocessor are dead from the time they enter the free list until the time they are written with a value. Register file access ports are unused when fewer than the maximum number of instructions begin execution on a given cycle. For the register file in 70 nm technology, we save up to 90% of leakage energy, or around 20% of total register file energy, with no loss in performance and minimal area overhead.

This paper is organized as follows. Section 2 reviews previous work in statically-selected slow transistor techniques. Section 3 describes previous work in dynamically-deactivated fast transistor techniques. Section 4 introduces metrics for comparing DDFT techniques. Section 5 describes how we estimated the process parameters for future process technologies. Section 6 describes the DDFT technique we use on caches. Section 7 describes the DDFT techniques we use on multiport regfiles. Section 8 details how we evaluated our DDFT techniques and discusses the results. Section 9 looks at related work in leakage reduction and modeling. Section 10 concludes the paper.

2. STATICALLY-SELECTED SLOW TRANSISTOR TECHNIQUES

The usual application of SSSTs is to 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 save switching power and to reduce parasitic load on critical nodes. Leakage is proportional to gate width, and so these narrower transistors also have lower leakage. Traditional design methods also use slower, more complex gate topologies on non-critical paths to reduce area. These more complex gates have deeper transistor stacks, which as a side effect also reduce leakage.

As power dissipation from leakage current increases, further techniques are being considered to reduce leakage current on 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 [8]. This approach was also used in the Alpha 21164 processor to control the effects of leakage on dynamic gates [15]. Lengthening transistor gates to control leakage has the disadvantage that active power can increase because of the increased gate capacitance.

Although it is desirable to scale threshold voltage along with supply voltage to maintain performance scaling, thresholds need only be lowered on critical path transistors. 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. It has been shown that the leakage of random static logic can be reduced by more than 50% [11] in this way. 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 TRANSISTOR TECHNIQUES

After application of SSST techniques to non-critical path transistors, 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 [16]. 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, these techniques attempt to dynamically switch critical path transistors between fast, leaky, active operation and inactive low-leakage states. We refer to this general category of techniques as dynamically deactivated fast transistors (DDFTs).

One DDFT technique that has become popular in low-power processors for portable devices is the application of a dynamically varying body bias to modulate transistor threshold voltages [17, 19, 28, 27, 20]. 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, but then to actively *forward* bias the wells during normal operation to lower V_T [12]. 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 specification [12]. 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. These schemes are mainly used to reduce leakage when the processor enters a sleep state, where the processor is expected to be idle for at least 0.1–100 μs [9, 21, 14] allowing the latency and energy costs of transitioning into the low leakage state to be amortized over the long sleep time.

An alternative DDFT approach is power gating [17, 19, 28, 27, 20]. Power supply to circuits can be cut off from Vdd (or GND) by inserting a high V_T *sleep transistor* between Vdd and virtual Vdd (or GND and virtual GND). When switched off, the sleep transistor adds an extra high-V_T transistor in series with the logic transistors, which dramatically reduces 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 [28], boosting the gate voltage for the sleep transistor [27], or forward-biasing the sleep transistor [20], 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 [32, 26]. If a combination of input patterns and internal state can be found which minimizes the leakage current, then this *sleep vector* can be applied to place the circuit into a low-leakage state. The sleep vector can be applied by forcing internal latches into a known state, and by forcing inputs to the correct polarity. However, it is sometimes difficult to find the optimal sleep vector and also the application of the sleep vector can cause spurious toggling in the circuits, which results in significant deactivation energy.

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

A few researchers have proposed more fine-grained deactivation techniques that place portions of the processor into low-leakage states during active operation. The dynamically-resized instruction cache [13] 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. Cache decay [29] 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.

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. The main contribution of this paper is to consider the use of DDFT techniques applied over a much shorter time scale than in previous work. In the next section we present a methodology for comparing DDFT techniques that factors in the energy costs of transitioning into a low-leakage state. In the following sections, we present circuit-level DDFT techniques that can profitably reduce leakage energy over short idle times and show how these can be used within a superscalar microprocessor.

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.

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 depending on the time constants of internal nodes. 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 time, leakage current is substantially higher than in the steady-state.

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

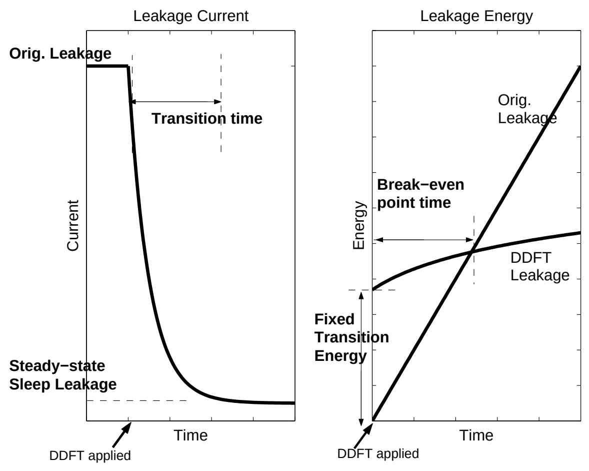

Figure 1: Transition time, steady-state leakage current, and break-even point time of DDFT leakage-reduction techniques.

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 over time dependent on the leakage current. On the same graph, we show an example curve for a DDFT technique. There is an initial fixed transition energy cost, which is given by summing fixed energy costs to move to the low-leakage state and to move back into the active state. In addition to the fixed transition energy costs, there may be additional variable transition energy costs proportional to the time that the block is deactivated. For example, in a virtual GND technique, the GND node is slowly charged over the transition time, and the amount of energy dissipated when this node is discharged to wakeup the block 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 point time, $t_{\text{breakeven}}$, as the time when the two curves cross, i.e., when the leakage energy of remaining in an active idle state matches the energy consumed after switching to idle in the low-leakage state. The circuit must be idle for considerably longer than the break-even point to save significant energy.

Another important factor in comparing DDFT techniques is the wakeup time. The wakeup time 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 time can sometimes be traded for transition energy, for example, using a wider transistor to discharge a biased well increases the transition energy to switch the transistor.

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

5. PROCESS TECHNOLOGIES

To evaluate our DDFT techniques, we use models of four dual-V_T processes, including 180 nm, 130 nm, 100 nm, and 70 nm pro-

Table 1: 180nm, 130nm, 100nm, and 70nm processes.

Parameter	180nm	130nm	100nm	70nm
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 Ion (uA/um)	732	732	732	732
LVT Ioff (nA/um) (optimistic)	21.8	43.6	87.2	174
LVT Ioff (nA/um) (pessimistic)	21.8	87.2	349	1395
HVT Ion (uA/um)	554	554	554	554
HVT Ioff (nA/um) (optimistic)	0.35	0.71	1.42	2.83
HVT Ioff (nA/um) (pessimistic)	0.35	1.42	5.68	22.6

cess 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 the future technologies, using the SIA roadmap [25]. For example, the SIA roadmap predicts that Ion remains the same, but Ioff 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 4X leakage increase per generation while the optimistic estimates assume 2X 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.

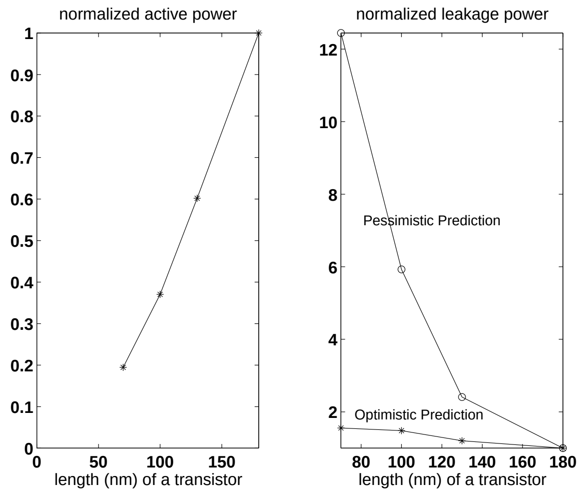

Figure 2: Normalized active and leakage power for different processes.

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 the leakage power 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.

6. DDFT TECHNIQUES FOR CACHES

The L1 caches of high-performance processors can cause significant leakage current, as they contain a large number of transistors which must be high-speed to avoid impacting processor cycle time. In this section, we present circuit-level techniques that support fine-grain DDFT schemes for reducing leakage.

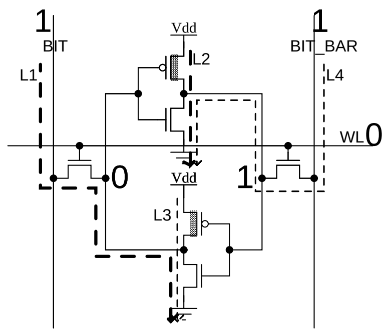

Figure 3: A dual-V_T SRAM cell. High V_T transistors are shaded.

Figure 3 shows the structure of an L1 cache SRAM cell together with the two primary leakage current paths when the word line is disabled. One leakage path, L_1, is from the precharged bit-line, through the access transistor, and across the turned-on n-type pulldown. 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.

With future technology scaling, noise margin concerns will require that SRAMs have fewer cells connected to each bitline segment, as the leakage currents from non-accessed bits reduce the effective signal from the accessed bit. 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 attached 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 [31], 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 by using this sleep vector. However, this has some disadvantages. First, if the percentage of zero bits is under 50%, the sleep vector technique *increases* leakage energy. Second, this technique requires additional bitline charging energy to transition 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. Instead of forcing 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 on one side, the leakage currents will fully discharge the bitline on that side. If all the cells store a one, the bitline will be held high. 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.

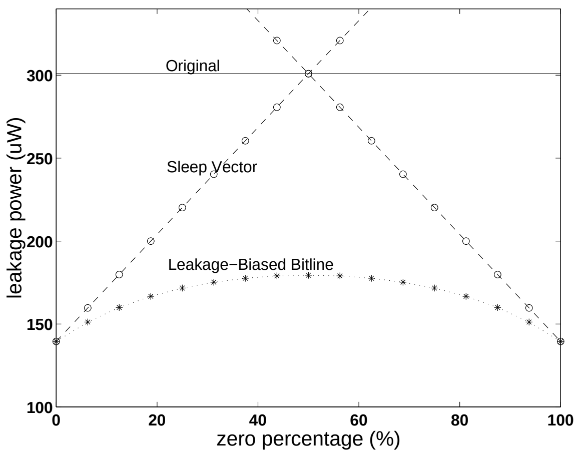

Figure 4: The leakage power of 32-row×16B SRAM subbank for forced-zero and forced-one sleep vectors and leakage-biased bitlines versus percentage of stored zero bits.

Figure 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 5 compares the idle energy and the LBB DDFT energy consumption for different processes. The LBB DDFT technique should charge the lost charge on bitlines back before use and the energy cost saturates at the bitline full swing energy. The break-even point time, $t_{breakeven}$, is around 200 cycles at 180 nm process. However, since active energy scales down faster than leakage energy, the break-even point time gets smaller reaching below one cycle in the 70 nm process.

Each subbank must be precharged before use, which can add latency to the cache access if the subbank is not known in time. In this paper, we focus on the application of the LBB DFFT technique to the processor instruction cache, because the instruction cache has a predictable access pattern. In the most optimistic case, we can assume that sufficient address bits are available to allow the required subbank to be precharged while the remainder of the address decode and word line drive completes, and therefore there is 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

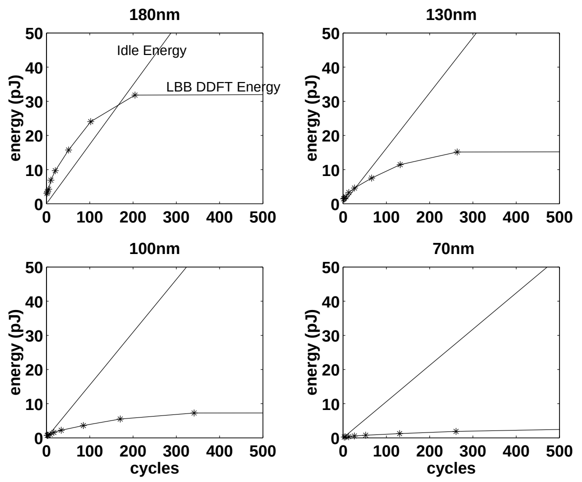

Figure 5: Idle energy and LBB DDFT energy of 32-row×32B SRAM subbank for different processes. (optimistic leakage current was used.)

penalty by one cycle. We also investigate alternative schemes that do not lengthen the pipeline. By mapping instruction addresses to cache subbanks such that sequential instructions are next to each other within a cache subbank, we can predict that the next access will be to the current subbank, adding a stall cycle when instruction fetch moves to a different subbank. A small variant of this scheme is to predict that we will move on to the next subbank when we access the last line in the current subbank. For an N-way set-associative cache structure, we access N subbanks in parallel, where each subbank returns a fetch group of instructions.

7. DDFT TECHNIQUES FOR MULTIPORT REGFILES

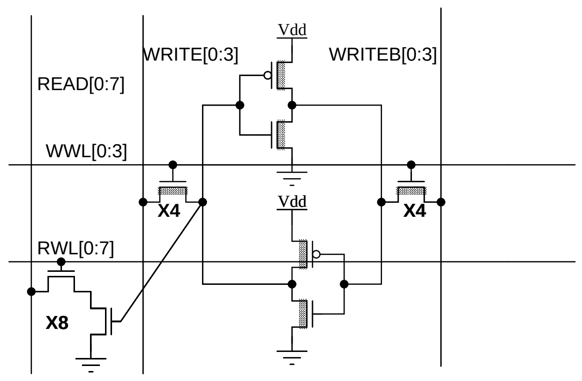

Figure 6: An embedded dual V_T unbalanced 8-read, 4-write register file cell. High V_T transistors are shaded.

Multiport regfiles are another component that can consume considerable leakage power. Figure 6 shows an 8-read port, 4-write port, regfile cell. Because there are many leakage paths in a multiport regfile 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. Our experiments showed that around 75% of the bits read from the register file are zero. (The alternative balanced cell would have half the read ports with true polarity and half with inverted polarity).

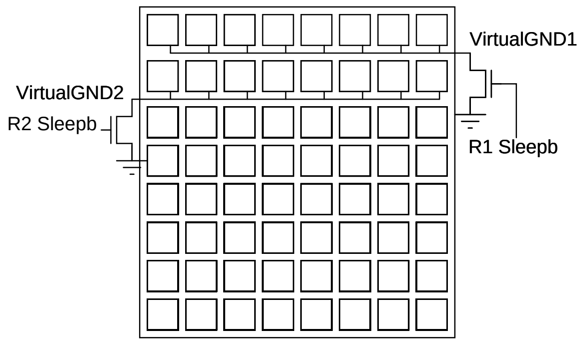

Figure 7: Idle register deactivation scheme using NMOS high V_T sleep transistors.

We applied two DDFT techniques to the multiported register file. The first technique deactivates individual registers whose contents are not needed. We exploit the fact that in a superscalar machine with register renaming, the contents of a physical register is not needed from the time it enters the free list until the time it is next written to. During this dead time, the physical register can be turned off. By using a virtual-GND line and an NMOS high-V_T sleep transistor between the virtual GND and GND, the leakage current can be reduced dynamically (Figure 7). The disadvantage of this scheme is that the read access time increases because the sleep transistor resides in the critical read path, and the virtual GND rises slightly above zero on an access. 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 DDFT technique acts in the orthogonal direction to deactivate read ports that are not in use, again using a leakage-biased bitline to reduce bitline leakage. We can then exploit the fact that in a superscalar machine, when fewer than the maximum number of instructions can issue, some of the regfile read ports will be idle. The LBB technique can be implemented for the single-ended read bitlines, in which case the bitlines simply discharge towards ground if any bits are holding a one. By turning off the precharger when idle, the leakage current through the read bit lines is reduced significantly. If the dead time is long enough, the energy overhead to precharge the bitline back up before an access becomes relatively small compared to the leakage current reduction. Note that unlike the virtual-GND technique, this technique does not corrupt the state stored in the register file. Also, there is no performance loss, because it is known whether a read port is needed before it is known which register will be accessed in the pipeline, allowing the precharge time to be overlapped with register file address decode.

As with the SRAM, the register file array is divided into subbanks with local bitlines connected to the global bitline, to save energy and to increase speed and noise margin. We place around 32 register bits on each local bitline. Figure 8 shows the hierarchical bitlines and a modified column cell for the LBB scheme for our multiported regfile.

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

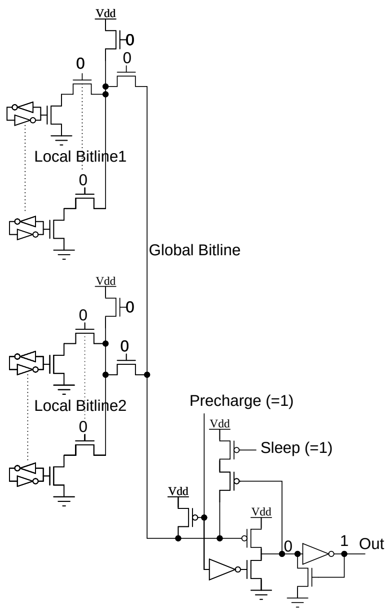

Figure 8: Leakage-biased bitline scheme for multiported register file. Each local bitline can be left unprecharged, biased by local leakage currents.

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)	6.02	2.87	1.38	0.47
one read E(pJ)	17.27	8.23	3.95	1.36
avg. read E(pJ) (E_r)	8.83	4.21	2.02	0.70
0-to-0 write E(pJ)	0.73	0.35	0.17	0.06
0-to-1 write E(pJ)	16.50	7.86	3.77	1.30
1-to-0 write E(pJ)	2.17	1.03	0.50	0.17
1-to-1 write E(pJ)	13.04	6.22	2.98	1.03
avg. write E(pJ) (E_w)	4.72	2.25	1.08	0.37

Table 3: The leakage power of 32×32-b multiported regfile subbank for different DDFT techniques.

Process Tech. (nm)	180	130	100	70
Original (uW)	177.9	214.1	263.6	276.7
SV steady-state (uW)	2.02	2.43	2.99	3.14
LBB steady-state (uW)	2.02	2.43	2.99	3.14
NST steady-state (uW)	1.84	2.21	2.73	2.86

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.

Table 3 shows the steady-state leakage power when different leakage techniques are applied 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. Both the leakage-biased bitline (LBB), and NMOS sleep transistor (NST) techniques reduce the leakage power to less than 1.5% of the original idle power when in the steady state.

Figure 9 shows the dynamic energy consumption of the DDFT techniques for 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 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.

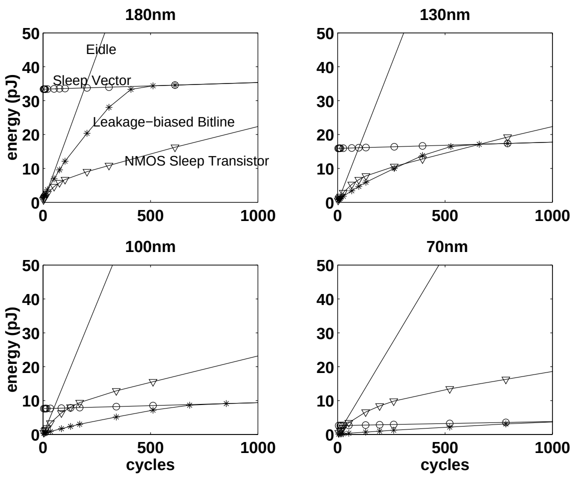

Figure 9: Idle leakage and deactivated leakage energy of different DDFT techniques for different processes (optimistic leakage current was used).

We see that for the sleep vector technique, the break-even point time is around 200 cycles at the 180 nm process, but shrinks to only 40 cycles in the 70 nm process. The sleep vector technique has high fixed transition energy costs, and so below the breakeven point, the energy consumption is much higher than the original leakage energy.

For the leakage-biased bitline, the break-even point time in the 180 nm process is only around 10 cycles. Moreover, the cumulative

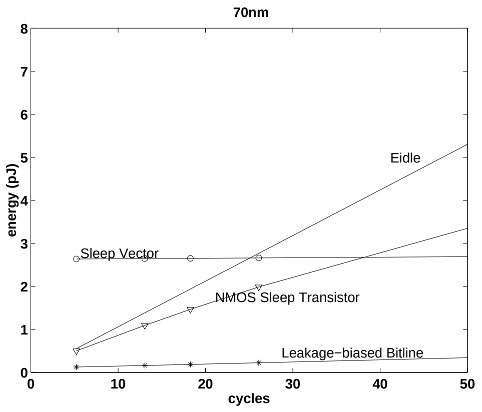

Figure 10: Expanded view of dynamic leakage energy consumption in 70 nm process technology (optimistic leakage current was used).

energy rises slowly from the initial deactivation time, and is not much larger than the original leakage before the breakeven point. With technology scaling, the break-even point 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.

8. ARCHITECTURAL 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 [2] 3.0b, an out-of-order, superscalar processor simulator, to track the activity of a physical register file and instruction cache. We consider both four-wide and eight-wide issue machines with the configurations shown in Table 4. We modified the SimpleScalar simulator to model a machine with a separate unified physical register file pool holding both committed architectural registers and renamed registers. The number of physical registers in the simulated architecture is determined by the number of writeable architected registers fixed by the ISA (33) plus the number of values that can be produced by in-flight instructions. On SimpleScalar, these in-flight instructions are stored in register update units (RUUs) which unify the reservation stations and reorder buffer. Thus, to examine our techniques on various sized register files, it was necessary to increase the number of RUUs in the simulated machine. While it may be overly optimistic to assume such a machine could be implemented, increasing the RUUs increases activity in the machine and would lead to more conservative estimates of the energy saved by fine-grain dynamic leakage reduction techniques. We restricted our study to the integer register file and used the SPECint95 benchmark suite. Seven [1] benchmarks were

[1]The compress benchmark contained floating point operations that could not be tracked by our simulation model

Table 4: Simulated Configurations

	4-Wide (64 RUU)	8-Wide (256 RUU)
Physical Registers (Integer)	100	292
Integer ALUs	4	8
Integer Mult/Div	1	2
FP ALUs	1	2
FP Mult/Div	1	2
Load/Store Units	2	4
Load/Store Queue Depth	32	64
Instruction length	4 Bytes	
I-Cache	16KB/4-Way/32B Block	
D-Cache	16KB/4-Way/32B Block	
Unified L2-Cache	256KB/4-Way/64B Block 6 cycle latency	
Memory Latency	First access: 50 cycs. Subsequent accesses: 2 cycs.	

Table 5: Cache subbank deactivation impact on average IPC

	Configuration		
Mispredict penalty (cycles)	4-Wide (64 RUU)	4-Wide (128 RUU)	8-Wide (256 RUU)
3	1.8852 IPC	1.8872 IPC	2.4030 IPC
4	1.8419 IPC	1.8433 IPC	2.3330 IPC

run on their reference data sets until 100 million instructions had committed.

8.2 Cache Deactivation Results

Figures 11–12 show the energy savings achieved for the instruction cache subbank deactivation scheme. In these figures and those that follow, black bars denote the optimistic assumptions of future process generations as described in Section 5. White bars denote the pessimistic view of the future (greater leakage). For the 180 nm generation, there is a net energy increase, but for all other process technologies there is a net energy savings that reaches over 20% of total instruction cache energy in the 70 nm generation. The leakage energy graphs show the energy saved in idle circuits, i.e., using our DDFT technique we save over 40% of the idle leakage energy for the I-cache subbanks that are not in use (we do not include the leakage energy expended in an active circuit).

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. Table 5 presents the change in IPC when misprediction latency increases from 3 to 4 cycles to allow for a pipelined precharge access cycle. Figure 13 compares the performance penalty observed when we increase the branch misprediction latency; when we predict all accesses will be in the same subbank; and when we predict accesses will be in the current subbank except when they are to the last line of the subbank (in which case they are in the next subbank).

Overall, increasing the branch misprediction latency results in the least performance degradation, of around 3%. This can be attributed to a reasonably accurate branch prediction strategy combined with the frequency that the code switches between subbanks. Assuming the subbank stays the same actually performs slightly better than incorporating a simple predictor when the last line in a subbank has been reached. Since a branch occurs roughly every 6 or 7 instructions, and each cache line contains 8 instructions,

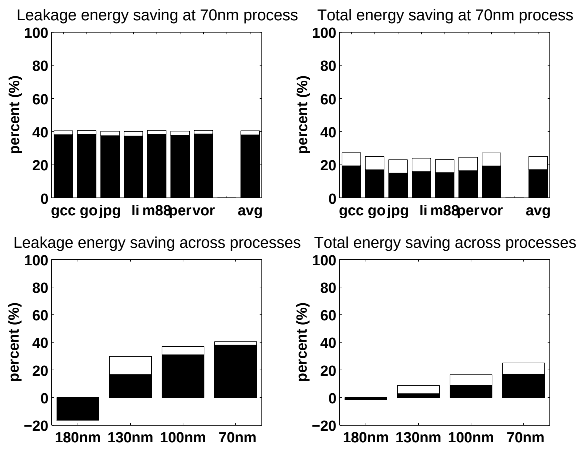

Figure 11: I-cache energy saving : 4-Wide 64 RUU 32-rows/subbank I-cache configuration.

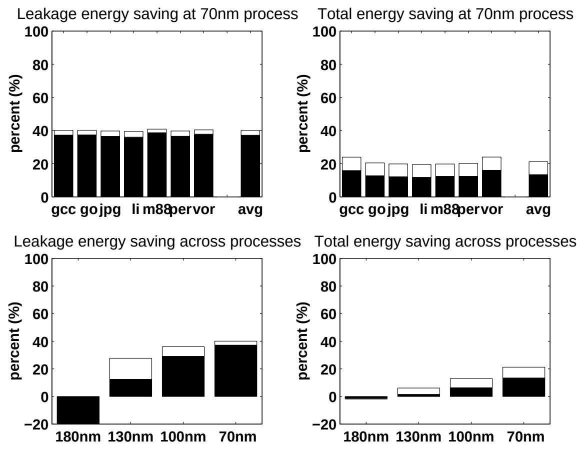

Figure 12: I-cache energy saving : 8-Wide 256 RUU 32-rows/subbank I-cache configuration.

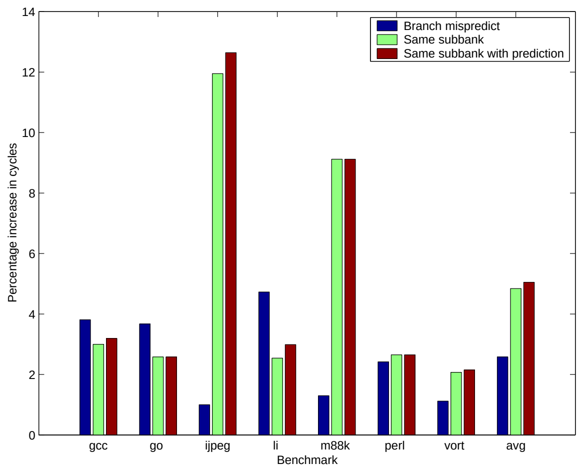

Figure 13: Performance penalty incurred by deactivating the I-Cache subbanks.

then typically a branch will be found in each line. Because most branches are taken, and most branch backwards, it follows that when the last line in a cache subbank is reached, it is more probable that the next line will be in a same or previous subbank. Still, both strategies decrease performance by about 5%, which is worse than simply using the branch predictor. To reduce the misprediction penalty further, it might be possible to extend the branch predictor to predict both the next branch target, and the subbank for following fetch following the branch target.

8.3 Regfile Dynamic Deactivation Results

To quantify energy saved by deactivating idle registers with sleep transistors, we maintain a set of physical register tags which move between a freelist, the register update units, and the rename table. To ensure simulation accuracy, no micro-architectural changes are made to the modeled machine. We note how many times every physical register is read and written; how many cycles it spends allocated but inactive; and how many cycles it is dead. When a register is read more than once in a particular cycle, we approximate the additional read energy by assuming it increases linearly. These counts, together with the the energy, power, and process speed data presented above, enable us to compute total register file energy via its active and leakage components. Figure 14 shows an average leakage energy savings of 41.1% in the 70 nm process (as with results in the previous section, we only measure savings in the leakage energy of idle circuits). This corresponds to a 3.0-12.4% savings in overall register file energy. The figure also shows total energy savings for both optimistic and pessimistic assumptions of future processes. Even in the most optimistic case, we see potential for energy savings. While the overall savings is not as great as was noted in the I-Cache, note that there is no performance penalty for deactivating registers. In addition, register file sleep transistors save energy in all process generations whereas the cache-specific DDFT optimizations result in an energy penalty in the current 180 nm process.

To further improve these savings, we are experimenting with various physical register allocation policies. Both LIFO (stack) and FIFO (circular queue) may easily be applied to allocate a single bank of registers as well as a banked register file. In the multi-banked design, registers are assigned from a new bank only when the previous bank is empty, allowing an entire bank to be deactivated at times. Stack-based allocation has the advantage of keeping *some* registers completely unused and others dead for very long times. Due to long transition times, such registers see the biggest savings from DDFT techniques. However, a stack limits the dead time of frequently used registers while a circular queue distributes non-negligible deadtime evenly across all registers which may lead to a greater number of dead cycles overall.

Figure 15 shows the energy savings achieved by deactivating the read ports. As with the cache subbank deactivation scheme, 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, 87.8-98.5% of the possible leakage energy is removed on average, resulting in a total register file energy savings of 9.3-47.8%.

As the processor's issue width increases, a higher number of read ports is needed. 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. Due to time and space constraints, results are presented for the four-wide processor only.

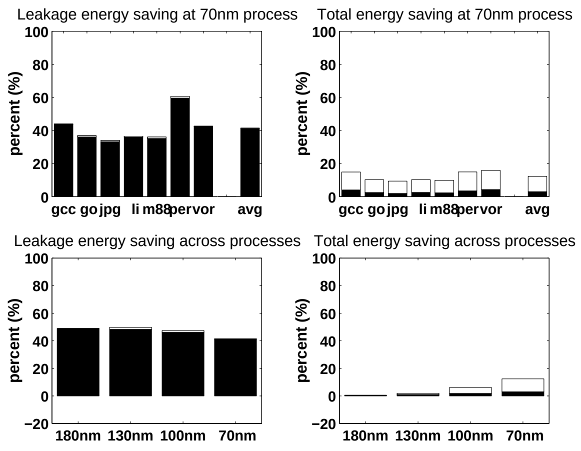

Figure 14: Register file energy saving by idle register deactivation : 4-Wide 64 RUU configuration.

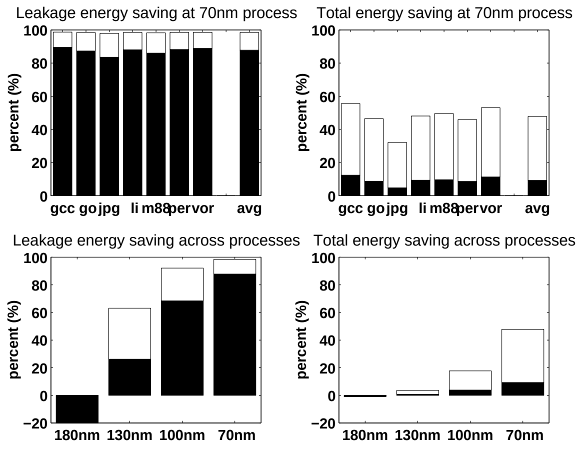

Figure 15: Register file energy saving by read port deactivation : 4-Wide 64 RUU configuration.

9. RELATED WORK

Butts and Sohi [3] presented a survey of static leakage reduction techniques and a simple leakage power model, but this model does not account for state-dependent leakage or dynamic deactivation techniques. Hamzaoglu et al. [6] briefly describe a "precharge-as-needed" scheme, 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.

The dynamically-resized instruction cache using gated-Vdd [13] also reduces instruction cache leakage energy with dynamic deactivation. 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. The cache decay approach [29] is another dynamic deactivation technique, that is complementary to the techniques we present here.

10. 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, and 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 techniques that apply this principle and have shown how they enable leakage current reductions in the context of a wide superscalar processor. SRAM read path deactivation saves over 40% of idle circuit leakage energy and over 20% of total I-cache energy when using a 70 nm process. Dynamically deactivating idle registers reduces regfile idle-circuit leakage energy by 41.1% and total regfile energy by 12.4%. Dynamically deactivating read ports within a multiported register file saves 87.8-98.5% of idle circuit leakage energy and 9.3-47.8% 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.

11. REFERENCES

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

[2] D.C. Burger and T.M. 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] V. De and S. Borkar. Technology and design challenges for low power and high performance. In *ISLPED*, pages 163–168, 1999.

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

[6] 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.

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

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

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

[10] 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.

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

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

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

[14] 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.

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

[16] 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*, February 2002. To appear.

[17] 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.

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

[19] 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.

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

[21] 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.

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

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

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

[25] International Technology Roadmap for Semiconductors. 2000 update, process integration, devices, and structures. Technical report, ITRS, 2000.

[26] 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.

[27] 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.

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

[29] 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.

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

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

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