MZZ-HVS: Multiple Sleep Modes Zig-Zag Horizontal and Vertical Sleep Transistor Sharing to Reduce Leakage Power in On-Chip SRAM Peripheral Circuits

Houman Homayoun, *Member, IEEE*, Avesta Sasan, *Member, IEEE*, Alexander V. Veidenbaum, *Member, IEEE*, Hsin-Cheng Yao, Shahin Golshan, and Payam Heydari, *Senior Member, IEEE*

***Abstract*—Recent studies show that peripheral circuit (including decoders, wordline drivers, input and output drivers) constitutes a large portion of the cache leakage. In addition, as technology migrates to smaller geometries, leakage contribution to total power consumption increases faster than dynamic power, indicating that leakage will be a major contributor to overall power consumption. This paper presents zig-zag share, a circuit technique to reduce leakage in SRAM peripherals by putting them into low-leakage power sleep mode. The zig-zag share circuit is further extended to enable multiple sleep modes for cache peripherals. Each mode represents a trade-off between leakage reduction and the wakeup delay. Using architectural control of multiple sleep modes, an integrated technique called MSleep-Share is proposed and applied in L1 and L2 caches. MSleep-share relies on cache miss information to guide leakage control mechanism and switch peripheral circuit's power mode. The results show leakage reduction by up to 40× in deeply pipelined SRAM peripheral circuits, with small area overhead and small additional delay. This noticeable leakage reduction translates to up to 85% overall leakage reduction in on-chip memories.**

***Index Terms*—Horizontal and vertical sleep transistor sharing, leakage power, multiple sleep modes, SRAM peripheral.**

I. Introduction

CMOS technology scaling has been a primary driving force to increase the processor performance. A drawback of this trend lies in a continuing increase in leakage power dissipation, which now accounts for an increasingly large share of processor power dissipation. This is especially the case for large on-chip SRAM memories.

A large fraction of processor area is devoted to the SRAM memories, mainly L1 and L2 caches. For instance, 60% of StrongARM and more than 50% of Intel Itanium 2 9010 processor are devoted to on-chip caches. Such a large area makes caches responsible for a large fraction of on-chip leakage power dissipation [3], [5]. For instance, on-chip L1, L2 and L3 caches account for 22% (12 Watts) of the total power dissipation in the Intel Potomac processor chip [23].

Typically, the subthreshold leakage current is the dominant off-state current in nanoscale CMOS technologies. This is mainly due to using low-$\mathrm{V_{th}}$ transistors to maintain the circuit switching speed [39]. To overcome the growing subthreshold leakage problem in SRAM memories, a number of technological, circuit-level, and architectural approaches have been proposed, many of which have focused on reducing the SRAM *memory cell* leakage by keeping them in a low-power state, and retaining data but not allowing access. These techniques include usage of body bias control, reduced $\mathrm{V_{dd}}$, Gated-$\mathrm{V_{dd}}$, multiple $\mathrm{V_{th}}$, etc. A number of architectural techniques were proposed to utilize such circuits by targeting SRAM cells, e.g., cache decay [19] and drowsy cache [21].

Recent results have shown that leakage in SRAM peripheral circuits, such as word-line as well as input and output drivers are now the main sources of leakage [8], [12], [17], [32], [33]. For instance, a wordline driver drives its signal to a large number of memory cells. To drive high capacitive loads, a chain of tapered inverter buffers is used, typically with three to five levels. We compared the leakage power consumption of a 65 nm SRAM6T memory cell[1] with an inverter of different sizes. The results are shown in Fig. 1(a). It shows that the leakage power of a standard memory cell is significantly lower than the leakage power of inverter buffers and that the inverter leakage grows exponentially with its size. For instance, let us assume that a driver has to drive 256 one-bit memory cells. This will require three stages of inverter buffers (of increasing size, by a factor of e). The combined leakage power of these three drivers is 12 times larger than the leakage of the 256 memory cells. In addition to the wordline driver one has to consider leakage in data input and output drivers, which are also high.

In brief, two main reasons explain this difference in leakage.

- Memory cells are designed with minimum-sized transistors mainly for area considerations. Unlike memory cells, peripheral circuits use larger, faster and accordingly, more

Manuscript received February 18, 2010; revised July 18, 2010; accepted August 26, 2010. Date of publication December 03, 2010; date of current version October 28, 2011.

H. Homayoun is with the Department of Computer Science and Engineering, University of California San Diego, CA 92093 USA (e-mail: hhomayou@eng.ucsd.edu).

A. Sasan, H.-C. Yao, and P. Heydari are with the Department of Electrical and Computer Engineering, University of California Irvine, CA 92697 USA (e-mail: mmakhzan@uci.edu; hsinchey@uci.edu; payam@uci.edu)

A. V. Veidenbaum and S. Golshan are with the Department of Computer Science, University of California Irvine, CA 92697 USA (e-mail: alexv@ics.uci.edu; golshans@uci.edu)

Digital Object Identifier 10.1109/TVLSI.2010.2086500

[1] Results were obtained for TSMC, TOSHIBA, IBM, UMC and CHARTERED foundries using their libraries and evaluating leakage with SPICE.

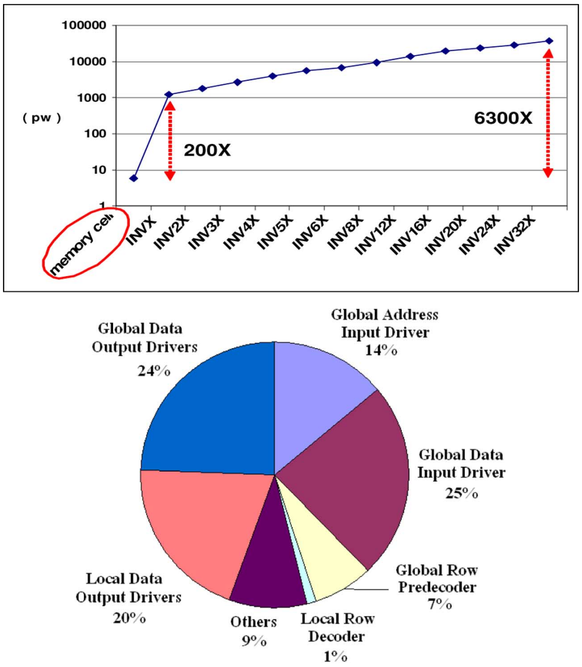


Fig. 1. (a) Leakage power dissipation of one SRAM6T memory cell compared with different size inverter buffers (INVX is the smallest inverter buffer with drive strength of 1). (b) Leakage power components of an L2 cache access.

leaky transistors so as to satisfy timing requirements [8], [17].

- Memory cells use high threshold voltage transistors, which have a significantly lower leakage reduction compared with typical threshold voltage transistors used in peripheral circuits [17], [32].

In summary, SRAM memory cells are optimized for low leakage current and area without a significant impact on the cell performance [8], [32]–[34]. In addition, circuit techniques such as gated-$\mathrm{V_{dd}}$ and drowsy cache can be applied to further reduce the memory cell leakage and widen the gap between cell array and peripheral leakage power dissipation.

A similar result is obtained using CACTI 5.1 [22]. CACTI uses characteristics of transistors modeled by the ITRS [35]. It includes data for two device types that the ITRS defines—High Performance (HP) and Low Standby Power (LSTP). The HP transistors are fast transistors with short gate lengths, thin gate oxides, low $\mathrm{V_{th}}$, and low $\mathrm{V_{dd}}$. The LSTP transistors, on the other hand, have longer gate lengths, thicker gate oxides, higher $\mathrm{V_{th}}$, and higher $\mathrm{V_{dd}}$. As explained in [8], HP transistors are used in the peripheral circuitry while the LSTP transistors are used in the memory cells array. While it is possible to use LSTP transistors in peripheral circuits for reducing leakage, the impact on memory access time would be significant (for instance, an increase from 3.8 ns to 12.5 ns access delay for a 2 MB L2 cache). Fig. 1(b) shows the leakage power breakdown for a 2 MB L2 cache in a 64-bit, 3.0 GHz processor. These results were obtained using CACTI-5 [22] for 65 nm technology. SRAM peripheral circuits, such as global and local input/output drivers, word-line drivers, and row decoders, dissipate more than 90% of the total leakage power. Thus, approaches that concentrate on cell leakage power alone are insufficient and it is very important to address leakage in the peripheral circuits.

Since peripheral circuits use very large transistors to drive high loads to meet the memory timing constraints applying traditional leakage reduction techniques such as "fine grained" sleep transistor insertion [42], [44] in these circuits could introduce significant area and delay overhead. This is mainly due to the impact of leakage reduction techniques on peripheral circuits' rise time, fall time, propagation delay and area overhead, which will be discussed in Section IV. These delays not only can significantly increase the cache access time, but also require significant amount of redesign and verification efforts to avoid impacting memory functionality, as will be discussed in Section IV. In addition, peripheral circuits are driving logic-level values and cannot be simply power-gated. For instance, a word-line driver has to drive a $\mathrm{V_{low}}$ voltage for all wordlines not being accessed. These leakage reduction techniques are thus not applied in high-performance processors. The focus of this paper is, therefore, leakage-related reduction of power dissipation in the on-chip SRAMs, specifically in their peripheral circuits.

This paper explores an integrated circuit and architectural approach to reduce leakage in the cache peripherals. It expands the work published in [48], highlighting the importance of leakage power reduction in on-chip SRAM peripherals.

We propose a circuit technique referred to as *zig-zag horizontal and vertical share* (in brief, *zig-zag share*) to effectively reduce leakage in the cache peripherals. Zig-zag share circuit is based on the well-known sleep transistor approach [25] and requires minimal modification in the peripheral circuitry. While zig-zag share has little impact on cache performance and none on functionality during cache access period, it reduces the peripheral circuit leakage during period in which the cache is not used and the sleep signal is asserted. The delays and area overhead introduced by the zig-zag share scheme and its power savings are evaluated using Cadence Simulation in a CMOS 65 nm process. The results show that the zig-zag share scheme reduces leakage power in peripheral circuits by more than 95% (relative to the peripheral circuits with no leakage control mechanism), while increasing peripheral circuits' delay and area by 1% and 5%, respectively. This noticeable reduction translates to up to 85% leakage power reduction of a entire 2 MB L2 cache.

We further extend the zig-zag share circuit to enable multiple sleep modes for cache peripherals. Each mode represents a trade-off between leakage reduction and the wakeup delay. This is done by controlling the bias voltage of the sleep transistors in the zig-zag share circuit. We propose a low-power design for the bias generator circuit required for the multiple sleep modes operation, shown to be fairly robust against six-sigma process, voltage and temperature variations. In addition, we explore the design space of sleep transistor insertion in SRAM peripheral circuitry and show the effect of sleep transistor size, its gate bias and the number of horizontal and vertical level sharing on the trade off between the leakage power savings and the impact on instability, area, dynamic power, propagation delay, wakeup delay, rise time and fall time of the peripheral circuit of SRAM.

Micro-architectural control of this circuit technique(i.e., timing evaluation for assertion and de-assertion of the sleep signal) is a challenging problem for on-chip caches in high-performance processors, because transitions to and from the low-power mode introduce additional delays. We propose an integrated approach, called multiple modes sleep share (or in brief MSleep-Share), which adds architectural control of zig-zag share for L1 and L2 caches. MSleep-share technique exploits the fact that the processor is idle, waiting for the memory response on an L2 miss for a portion of program executing time. As such, we propose to assert the sleep signals and put peripheral circuits into sleep mode while the processor is idle.

The rest of this paper is organized as follows: related work is described in Section 2. Section 3 provides some background knowledge on sleep transistor stacking effect. Sections 4 and 5 present the leakage control circuit scheme. Section 6 presents the experimental analysis of the zig-zag share circuit technique. Section 7 describes the architectural MSleep-Share technique and finally Section 8 concludes this work.

II. Related Work

In this section, we briefly review past work on reducing leakage power at technology, circuit, architecture and compiler levels.

A. Circuit-Level Leakage Control

Four main circuit schemes have been proposed to reduce the leakage power in SRAM memories. These techniques are mainly applied to the SRAM memory cells. The primary technique is voltage scaling, which reduces the source voltage of the cells. As explained in [16], due to short-channel effects in deep submicron processes, voltage scaling reduces the leakage current significantly. In [21], voltage scaling is shown to be effective in reducing leakage when applied on L1 cache memory cells. Voltage scaling can be combined with Frequency Scaling (DVFS) to reduce both dynamic and leakage power more effectively [37].

Another technique is Gated-$\mathrm{V_{dd}}$, which turns off the supply voltage of memory cells by using a sleep transistor [25]. The advantage of this technique lies in its substantial reduction of the leakage power. However, it does not retain the state of the memory cells. Similarly, $\mathrm{V_{ss}}$ can also be gated (gated $\mathrm{V_{ss}}$).

The third technique, ABB-MTCMOS, increases threshold voltage of a SRAM cell dynamically through controlling its body voltage [26]. The overhead of applying this technique in terms of performance and area makes it inefficient.

Device scaling leads to threshold voltage fluctuation, which makes the cell bias control difficult to achieve. In response, [32] proposed a Replica Cell Biasing scheme in which the cell bias is not affected by $\mathrm{V_{dd}}$ and $\mathrm{V_{th}}$ of peripheral transistors.

The fourth technique is forward body biasing scheme (FBB) [4], [14] in which the leakage power is suppressed in the unselected memory cells of cache by utilizing super Vt devices.

In addition to these four major techniques applied to SRAM memories, there are also some leakage reduction techniques in literature which concentrated on generic logic circuits. Examples are sleepy stack [24], sleepy keeper [27] and zigzag super cut-off CMOS (ZSCCMOS) techniques [31]. ZSCCMOS reduces the wakeup overhead associated with Gated-$\mathrm{V_{dd}}$ technique by inserting the sleep transistors in a zigzag fashion. Sleepy stack proposed to divide the existing transistors into two half size and then insert sleep transistor to further reduce leakage. This approach was shown to be area-inefficient as it comes with 50 to 120% area overhead. Sleepy keeper is a variation of Gated-$\mathrm{V_{dd}}$ approach which can save logic state during sleep mode. The draw back of this approach is significant additional dynamic power overhead compare to the base circuit.

Optimal sizing of sleep transistor to minimize the impact of sleep transistor insertion on the circuit delay has been researched extensively in [40]–[43]. Reference [40] used the average current consumption of logic gates to find the size of sleep transistor for satisfying circuit speed. Their proposed technique is based on the assumption that the circuit speed is weakly depending on the circuit operating pattern for large enough sleep transistor size.

B. Architectural Techniques

A number of architecturally driven cache leakage reduction techniques have been proposed. Powell *et al.* proposed applying gated-$\mathrm{V_{dd}}$ approach to gate the power supply for cache lines that are not likely to be accessed [13]. This technique results in the data loss in the gated cache line. This leads to an increase in the cache miss rate and loss of performance.

Kaxiras *et al.* proposed a cache decay technique which reduces cache leakage by turning off cache lines not likely to be reused [19]. Flautner *et al.* proposed a drowsy cache which reduces the supply voltage of the L1 cache line instead of gating it off completely [21]. The advantage of this technique is that it preserves the cache line information but introduces a delay in accessing drowsy lines. However, the leakage power saving is slightly lower than the gated $\mathrm{V_{dd}}$ and $\mathrm{V_{ss}}$ techniques. Nicolaescu *et al.* [1] proposed a combination of way caching technique and fast speculative address generation to apply the drowsy cache line technique to reduce both the L1 cache dynamic and leakage power. Bai *et al.* optimized several components of on-chip caches to reduce gate leakage power [2]. Zhang *et al.* proposed a compiler approach to turn off the cache lines for the region of the code that would not be accessed for a long period of time [29]. Meng *et al.* presented a prefetching scheme which combines the drowsy caches and the Gated-$\mathrm{V_{dd}}$ techniques to optimize cache leakage reduction [28].

The research mentioned above primarily targeted the leakage in the SRAM cells of a cache. Given the results in Fig. 1, peripheral circuits are equally important in leakage power reduction as a cache.

III. Sleep Transistor Stacking Effect

This section provides some background material on one of the most well-known traditional leakage reduction circuit technique; stacking sleep transistor. To better help understand the remainder of this paper first let us elaborate on the leakage current mechanism.

Subthreshold or weak inversion current ($\mathrm{I_{Dsub}}$) flows from the drain of a transistor to its source during off-state (i.e., gate voltage is below the threshold voltage).

The subthreshold current is an inverse exponential function of threshold voltage and is expressed as following [7]:

$$I_{\text{leakage}} = I_0 \left(\frac{W}{L}\right) 10^{\frac{(V_{\text{GS}}-V_{\text{th}})+\lambda(V_{\text{DS}})}{S_t}} \quad (1)$$

$$V_T = V_{T0} + \gamma(\sqrt{|-2\phi_F + V_{\text{SB}}|} - \sqrt{|2\phi_F|}) \quad (2)$$

where $\mathrm{V_{th}}$ is the threshold voltage, λ is the DIBL coefficient, S_t is the subthreshold slope, and γ is the body effect coefficient. An

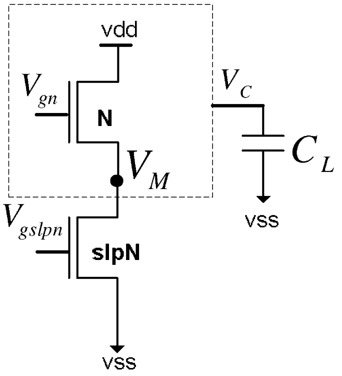


Fig. 2. Stacking sleep transistor to reduce leakage.

effective way to reduce leakage of a transistor is thus increasing its source voltage (for an nMOS increasing V_{SB}). This can be done by stacking it with a sleep transistor as shown in Fig. 2 [47]. Stacking transistor N with slpN increases the source-to-body voltage $(\mathrm{V_M})$ of transistor N, and thus, reduces its subthreshold leakage current when both transistors are off [9]. Such leakage reduction is related to the virtual ground voltage $\mathrm{V_M}$ during sleep mode which in turn is determined by the size of the sleep transistor and its bias voltage (V_{gslpn}) [9], [15]. Reducing the sleep transistor bias reduces the leakage, while increasing the circuit transition wakeup period, because waking up transistor N requires pulling down the voltage $\mathrm{V_M}$ to ground. Thus, a trade-off exists between the amount of leakage saved and the wakeup overhead [15]. In addition to wakeup overhead, a major drawback of stacking sleep transistor with a circuit is its impact on circuit timing and area, and the circuit output rise time, fall time and propagation delay. Also, stacking sleep transistor would make all high voltage nodes to discharge during the stand-by mode due to the large leakage current of the circuit [30]. This discharging induced voltage error (i.e., metastability) has to be addressed for circuits which have to keep a specific state during stand-by mode, e.g., wordline drivers of SRAM memories.

IV. Sleepy Peripherals

We investigate how sleep transistor stacking can be applied to reduce subthreshold leakage in the peripheral circuitry. First, we analyze the source of subthreshold leakage in the peripheral circuitry. Of all SRAM peripheral circuits, the discussion here will be limited to word-line drivers. Other drivers use a similar inverter chain and are treated similarly. An analysis of subthreshold leakage sources in a wordline driver implemented as a four-stage inverter (buffer) chain (shown in Fig. 3) is performed and a solution to reduce its leakage is proposed. The wordline driver controls pass transistors enabling access to a row of SRAM memory cell(s). The number of buffer stages is chosen to meet timing requirements of SRAM access. The inverter chain has to drive a logic value 0 to the pass transistors when a memory row is not selected. Thus, the driver cannot be simply shut down when idle. Transistors N1, N3 and P2, P4 are in the off state and thus they are leaking. To reduce the leakage in these transistors we apply sleep transistor stacking in the inverters chain. Due to induced negative source to gate biasing, subthreshold leakage current in the inverter chain reduces significantly. However, rise and fall times and propagation delay of the circuit are affected. Next, we discuss several approaches to minimize this effect while maximizing leakage reduction.

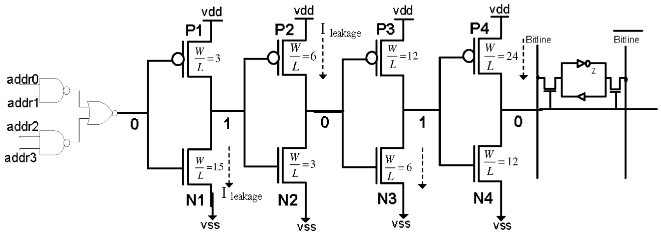


Fig. 3. Leakage in the wordline driver.

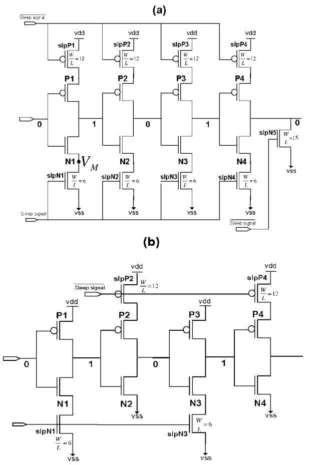


Fig. 4. (a) Redundant circuit. (b) Zig-zag circuit.

A. A Redundant Circuit Approach ("Fine Grained" Sleep Transistor Insertion)

In this technique, pMOS and nMOS sleep transistors are stacked at the finest granularity level in the inverter chain to reduce the chain's leakage during the off-state (Fig. 4(a)). We refer to this approach as the redundant circuit scheme. As discussed above, inserting the sleep transistor introduces metastability in the circuit, since high voltage nodes are discharged slightly during the sleep mode. In addition wordline driver has to drive a zero to the bypass transistors in memory cells to preserve their state during sleep mode.

To eliminate metastability, when the circuit is in sleep mode, we propose to insert a minimum size nMOS transistor in the last stage of the inverter chain [52], as shown in Fig. 4(a) (we refer to this transistor as a state preserving-transistor). During the sleep mode the state-preserving transistor is turned on to maintain the state of the wordline driver output by pulling it down to the ground.

The drawback of the redundant circuit technique is its impact on wordline driver output rise time, fall time, and hence, the propagation delay.

The rise time and fall time of an inverter output is proportional to the $R_{\mathrm{peq}} * C_L$ and $R_{\mathrm{neq}} * C_L$, respectively, where R_{peq} is the equivalent resistance of the pMOS transistor, R_{neq} is the equivalent resistance of the nMOS transistor, and C_L is the equivalent wordline output capacitive load [7]. Inserting sleep transistors increases both R_{neq} and R_{peq}, and thus the rise time and fall times and propagation delay of the wordline driver.

Increase in rise and fall times and propagation delay is not desirable, as the memory functionality and access time are negatively affected [18], [20], [45], [46]. Any increase in the fall time of wordline driver results in an increase in pass transistor active period during a read operation. This result in the bit-line over-discharge and the memory content over-charge during the read operation. Such over-discharge not only increases the dynamic power dissipation of bit-lines, more importantly, it can cause a memory cell content to enter a metastable state if the over-discharge period is large [20], [45], [46]. In addition, such increase in pass transistors active period requires re-timing of the sense amplifier active period. Note that it is typical in low power SRAM memories to use self-timed clocked sense amplifier to limit sense power and track the bit-line delay to setup the sense amplifier in the high gain region for sensing right before the pass transistors are turned off. Such a self-timed clocked sense amplifier has to cope with any increase in the pass transistor M1 active period [20], [46]. In brief, to avoid impacting memory functionality, the sense amplifier timing circuit and the wordline pulse generator circuit need to be redesigned. To avoid revisiting these critical units [45], [46] and moreover not to increase bit-line dynamic power dissipation [20], we propose an alternative circuit solution, described in the next section.

B. A Zig-Zag Circuit

To alleviate the drawback of redundant scheme on peripheral circuit fall time, the sleep transistors are inserted in a zig-zag fashion, as shown in Fig. 4(b). By inserting the sleep transistors in a zig-zag fashion the R_{peq} for the first and third inverters and R_{neq} for the second and fourth inverters doesn't change. Therefore, the fall time of the circuit does not change compared to the baseline circuit with no leakage control, unlike the redundant circuit.

Unlike the fall time, the rise time of the circuit is affected by the zig-zag scheme. This is due to the fact that the rise time of the circuit is determined by the fall time of the output of the first and third stage inverters and the rise time of the output of the second and fourth inverters, which both increase due to insertion of sleep transistors.

To minimize the impact on the rise time, one can resize the sleep transistors. Increasing the size of sleep transistor would reduce its equivalent resistance and thus the inverter chain rise time, while it reduces the leakage savings. This is due to the fact that increasing the size of sleep transistor reduces the virtual ground node voltage (VM in Fig. 4), which in turn reduces the leakage savings. In addition, using one sleep transistor per inverter logic increases the area for the zig-zag scheme.

C. A Zig-Zag Share Circuit

To improve both leakage reduction and area-efficiency of the zig-zag scheme, we propose using one set of sleep transistors shared between multiple stages of inverters which have similar logic behavior, such as stage 1 and 3 in our studied chain of inverters. There are several ways of accomplishing this.

1) Zig-Zag Horizontal Sharing: Fig. 5 shows a zig-zag horizontal sharing circuit (in brief zz-hs) in which one set of sleep transistors is shared across multiple stages of a single row of wordline driver. We compare the zig-zag and zig-zag share schemes with the same area overhead; i.e., the size of the sleep transistor slpN in zig-zag share scheme is two times the size of sleep transistor slpN in the zig-zag scheme. As we will see, zz-hs circuit has less impact on wordline driver output rise time compared to the zig-zag circuit. The wordline driver output rise time is determined by the first and third inverters' output fall time, which in turn is proportional to R_{neq}. Stacking sleep transistors in both zig-zag and zz-hs increases the R_{neq}. The same figure shows the switching model of the first stage inverter during its input rise time (output fall time). During input rise time transistor N1 is in switching transient from the off region to the saturation region and finally resistive region. During this transient period transistor N3 is in off state (assuming the delay between the output fall time of first stage inverter and input rise time of third stage inverter is long enough). As such, the current flowing through the share wire shown in the figure can be said to be negligible. The equivalent resistor $R_{\text{Neq-zz-hs}}$ is thus equal to the $R_{N1}+R_{\text{nlp-zz-hs}}$. Since the size of slpN transistor for the zz-hs scheme is two times of the size of zig-zag scheme, $R_{\text{Neq-zz-hs}}$ is equal to: $R_{N1}+R_{\text{nlp-zz}}/2$ and is smaller than the $R_{\text{Neq-zz}}$ which is $R_{N1}+R_{\text{nlp-zz}}$.

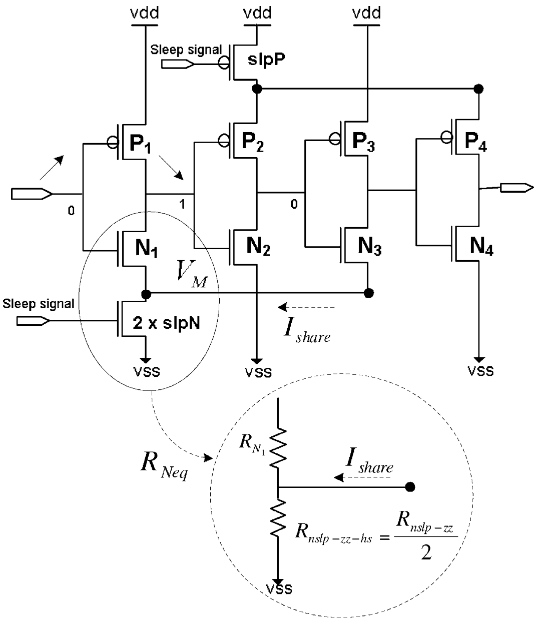


Fig. 5. Zig-zag horizontal sharing circuit.

Both zig-zag scheme and zz-hs reduce the leakage to almost the same level. This is due to the fact that in both schemes the virtual ground voltage (VM) increases to almost the same level.

2) Zig-Zag Horizontal and Vertical Sharing: To improve the leakage reduction of zig-zag horizontal sharing circuit we propose to share one set of sleep transistors (slpN and slpP) not only horizontally for multiple stages of an inverter chain in a wordline driver but also vertically and with other rows of wordline driver. Fig. 6 shows the zig-zag horizontal and vertical sharing circuit (in brief zz-hvs) when two adjacent wordline drivers share one set of sleep transistors. Assuming that one wordline is accessed at a time, both zz-hvs and zz-hs circuits have the same impact on wordline driver rise time and fall time and accordingly propagation delay (with similar-sized sleep transistors). The benefit of sharing sleep transistors further, not only horizontally but also vertically is in reducing leakage current. Intuitively, when we apply vertical sharing (for instance for N11

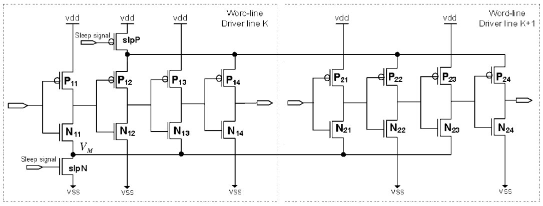


Fig. 6. Zig-zag horizontal and vertical sharing circuit.

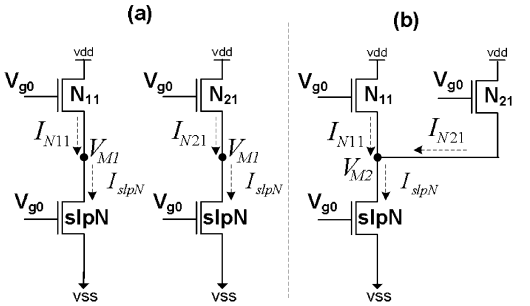


Fig. 7. (a) zz-hs and (b) zz-hvs equivalent during sleep mode.

and N21), the virtual ground voltage (VM in Fig. 6) increases compared to when there is no vertical sharing.

To better explain this, Fig. 7 shows the equivalent zz-hvs circuit when both N11 and N21 are in off region and share one sleep transistor, compare to when there is no vertical sharing (zz-hs). Note that the size of sleep transistor, slpN, for both zz-hs and zz-hvs is the same.

When there is no vertical sharing, the leakage current of the circuit is two times that of one of N11 or N21 transistors and is decided by the virtual ground voltage VM1. The higher VM1 results in lower leakage current. VM1 can be obtained by matching the leakage current of N11 and the leakage of slpN as following:

$$V_{M1} = \frac{n.\log_{10}^{\frac{W_{N11}}{W_{\text{slpN}}}} + \phi V_{\text{dd}} - V_{g0}}{2\phi} \tag{3}$$

where n is the subthreshold slope and V_{g0} is the sleep transistor's gate voltage. To find the virtual ground voltage (VM2) when we share the sleep transistor, we use the same methodology by matching the leakage current flowing through the sleep transistor, IslpN with cumulative leakage current of N11 and N21; IN21 + IN11. Since both transistors are similar and both in the same region (off), their leakage current is almost the same. As a result VM2 can be approximated as follows:

$$V_{M2} = \frac{n.\log_{10}^{\frac{2.W_{N11}}{W_{\text{slpN}}}} + \phi V_{\text{dd}} - V_{g0}}{2\phi} \tag{4}$$

V_{M2} is larger than V_{M1} which explains why zz-hvs is more effective in reducing leakage compare to zz-hs.

It should be noted that the reasoning made for vertical sharing is based on the assumption that the vertically shared inverters have the same size and characteristics.

V. Multiple Sleep Modes Zigzag-Share

As explained in Section 3, to benefit the most from the leakage savings of stacking sleep transistors we need to keep the bias voltage of nMOS footer sleep transistor as low as possible (and for pMOS header transistor as high as possible). The drawback of such biasing is its impact on wakeup latency and wakeup power of the circuit transitioning from sleep mode to active mode, which requires the voltage of virtual ground to reach the true ground. Such wakeup delay and wakeup power overhead would significantly impact performance and can diminish the power savings if incurred frequently. Appropriately sizing the sleep transistor (both footer and header) and controlling its bias voltage are two effective ways to minimize the impact on wakeup delay and wakeup power overhead. For instance, increasing the gate voltage of footer sleep transistor (in Fig. 2) reduces the virtual ground voltage (VM), which subsequently reduces the circuit wakeup delay and wakeup power overhead. The negative impact of such biasing is a reduction in leakage power savings. To better explain this, lets study the wakeup delay and wakeup power overhead as a function of sleep transistor size and its gate bias voltage.

The wakeup delay and power overhead are measured as the time and power required for the virtual ground voltage (VM in Fig. 2) to be discharged through a sleep transistor to reach the ground level [18]. This wakeup delay is expressed as follows:

$$E_{\text{wakeup}} = \frac{1}{2} C_{\text{block}} \times V_M^2 \tag{5}$$

$$T_{\text{wakeup}} = \frac{\int I_{\text{slp}}(t) \cdot dt}{I_{\max}} \cong \frac{V_M \times C_{\text{block}}}{I_{\max}} \tag{6}$$

where $\mathrm{I_{slp}(t)}$ is the current of sleep transistor after it turned on to wake up the block, $\mathrm{I_{max}}$ is the maximum current and C_{block} is the total capacitance of the circuit block [53]. Such wakeup overhead is decided by the equivalent load, as shown in (5).

As (3) indicates virtual ground voltage is linearly dependent on the sleep transistor gate voltage; increasing the gate voltage of the nMOS sleep transistor reduces VM. According to (5) and (6), such reduction lowers the wakeup delay and wakeup power overhead.

Also as discussed in Section III and according to (1), increasing the gate voltage of the sleep transistor results in higher leakage power dissipation. In fact, by controlling the gate voltage of footer and header transistors we can thus define different sleep modes where each mode has a different wakeup delay overhead and a different amount of leakage power reduction. We refer to this scheme as multiple sleep modes zig-zag horizontal and vertical sharing or in brief MZZ-HVS.

A. The Bias Generator

As described in the previous section, the benefit of multiple sleep modes is shown by controlling the gate voltage of the sleep transistor. A stable and robust voltage reference generator is needed to ensure the performance of multiple sleep modes operation over process, voltage, and temperature variations.

Conventional bandgap reference circuit consists of bipolar transistors and resistors that might occupy appreciable amount of area. A CMOS voltage reference in Fig. 8 consisting of transistors in saturation, triode, and subthreshold regions can provide sub-1-V reference required in our design. Transistors M1–M4 in Fig. 8 are always in saturation and generate supply

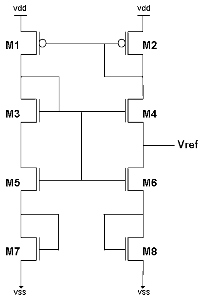


Fig. 8. Robust gate bias generator circuit.

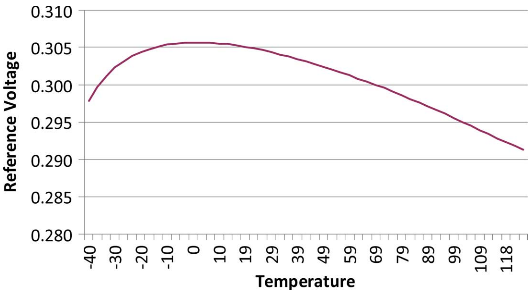


Fig. 9. Reference voltage versus temperature from −40 to 125°C.

independent current. Transistors M5 and M6 operate in the triode region as voltage controlled resistors and produce PTAT (Proportional to Absolute Temperature) voltage. Transistors M7 and M8 operate in the subthreshold region, behaving as bipolar transistors to produce CTAT (complementary-to-absolute temperature) voltage.

The reference voltage is taken from the drain node of M6. Fig. 9 shows the simulation results of reference voltage versus temperature from −40°C to 125°C, where only ±1.6% voltage variation around the nominal value is observed across this wide temperature range. The bias generator uses a compact layout and its overall area is not more than 30 × 30 (nm)2.

VI. Circuit Evaluation

In order to evaluate the proposed zig-zag share and multiple sleep modes approaches on memory peripherals, and specifically the wordline driver, we setup a test experiment assuming that the wordline inverter chain drives 256 one-bit memory cells. We laid out the memory cells and wordline drivers using Mentor Graphic IC-Station in TSMC 65 nm technology. The empirical results presented are for the leakage current, rise and fall times, propagation delay, wakeup delay, dynamic power, and finally area for all circuits discussed above compared to the baseline circuit without leakage control. All simulations use Synopsis HSPICE with extracted netlist and the supply voltage of 1.08 V.

A. Zig-Zag Share

The results for proposed circuits (except for zz-hvs and mzz-hvs)) in comparison with baseline, redundant and zig-zag circuits are shown in Fig. 10.

To have a fair comparison between zig-zag and zig-zag-share scheme, we report the result for zz-hs-2W which has almost the same area overhead to the baseline circuit as zig-zag scheme. Note that in zz-hs-1W the size of sleep transistors does not change compared to the zig-zag scheme. As shown in Fig. 10, the dynamic power of all schemes with sleep transistors increases slightly from 1.5% to 3.5% compared to the baseline. Unlike dynamic power, the leakage power reduction is significant. The maximum reduction is achieved in zz-hs-1W circuit where leakage is reduced by 94% relative to leakage in peripheral circuits prior to using leakage reduction technique.

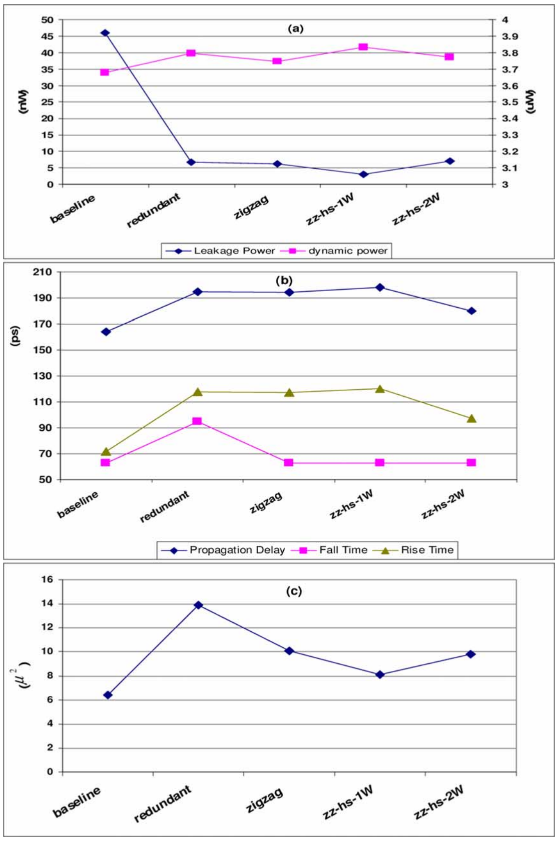


Fig. 10. (a) Leakage and dynamic power (b) propagation delay, rise time and fall time (c) area for various leakage control circuits.

The rise and fall times and propagation delay of the circuits are shown in Fig. 10(b). These results validate the approach described in Sections IV.B and IV.C with respect to the impact of zig-zag and zig-zag share circuits on rise and fall times. As expected, in both zig-zag and zig-zag share circuits the wordline driver, the fall time is not affected compared to that of the baseline circuit.

Also as expected, zz-hs-2W has minimum impact on rise time and propagation delay. Thus, increasing the size of the sleep transistor reduces the impact on circuit propagation delay as well as its rise time (we will discuss more on this later).

The area of all schemes is reported in Fig. 10(c). The increase in the area varies significantly from 25% for zz-hs-1W circuit to 115% for the redundant scheme.

Fig. 11 presents the experimental results for zig-zag horizontal and vertical sharing circuit (zz-hvs) and for different number of wordline rows compare to baseline (no leakage control), redundant, zig-zag and zz-hs schemes.

As depicted in Fig. 11, the more the number of wordline rows sharing sleep transistors, the higher the leakage reduction and less the overhead on the chip area. This in fact validates the analysis in Section 2. Overall, the leakage power is reduced

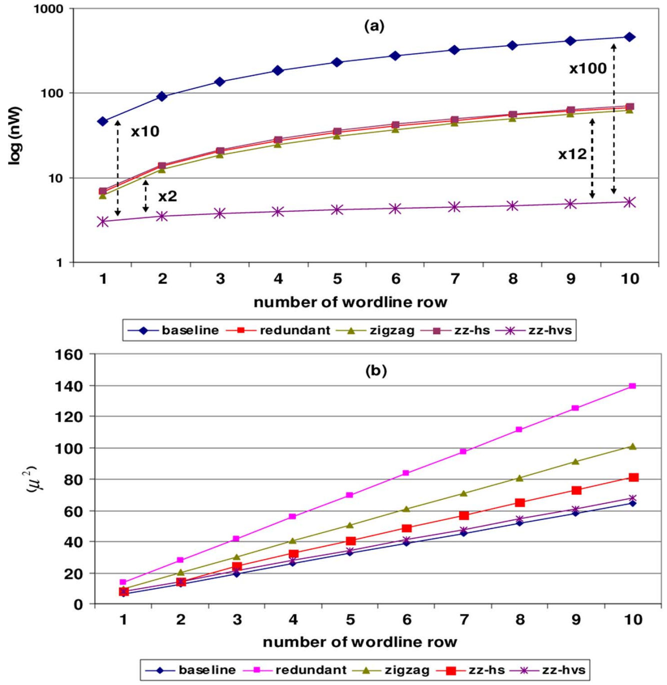


Fig. 11. (a) Leakage power. (b) Area of proposed zig-zag-share circuits compared to zig-zag, redundant and baseline.

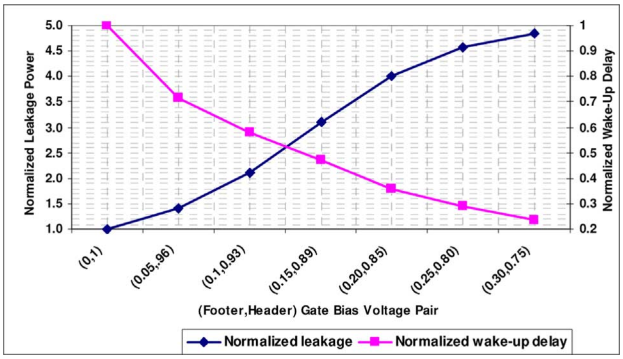


Fig. 12. Normalized wakeup delay and leakage power for different pair of footer and header gate bias voltage.

significantly, a 10× to a 100× reduction when 1 to 10 wordline rows share the same sleep transistors. This indicates 2–10× more leakage reduction, compared to the zig-zag scheme.

Note that by increasing the number of wordline rows sharing one sleep transistor, the load on the sleep transistor increases (mainly due to the extra wiring capacitance) and make its transition slower. Therefore, number of rows sharing one sleep transistor is a limiting factor on sleep transistor switching speed (for instance for wakeup delay).

The area for all schemes is shown in Fig. 11(b). As shown zz-hvs scheme has the least impact on area, 4–25% depends on how many wordline row are shared.

B. Multiple Sleep Modes

Fig. 12 shows normalized wakeup delay and normalized leakage power for different pairs of footer and header gate bias voltage when MZZ-HVS is shared by 10 rows of wordline drivers with each wordline driving 256 one-bit memory cells.

A clear trade-off can be seen between the normalized wakeup delay and leakage power. Increasing the bias voltage of the footer transistor reduces the leakage power savings as well as the wakeup delay.

C. Impact of Sleep Transistor Sizing on Leakage Power Reduction/Wakeup Delay and Propagation Delay

As discussed in Section IV, inserting sleep transistor in a zig-zag share fashion increases the circuit propagation delay as well as its rise time. Appropriately sizing of sleep transistor can minimize the impact on circuit propagation delay as well as its rise time. Assume that a $\beta\%$ increase in the wordline driver propagation delay is tolerable. Now let's find the appropriate size of sleep transistor so that the propagation delay does not increase beyond $\beta\%$. Delay of a gate without sleep transistor is expressed as

$$T_d = \frac{C_L \times V_{\rm DD}}{(V_{\rm DD} - V_{\rm tl})^{\alpha}}. \tag{7}$$

Delay of a gate with sleep transistor is expressed as

$$T_{\rm dsleep} = \frac{C_L \times V_{\rm DD}}{(V_{\rm DD} - V_x - V_{\rm tl})^{\alpha}} \tag{8}$$

where C_L is the load capacitance at the gate's output, $V_{\rm tl}$ is the threshold voltage of the inverter chain.

With $\beta\%$ overhead on delay allowed during active operation of the word line driver

$$\frac{T_d}{T_{\rm dsleep}} = \left(1 - \frac{\beta}{100}\right) \tag{9}$$

$$V_x = \frac{\ln\left(1 - \frac{\beta}{100}\right)}{\alpha}(V_{\rm DD} - V_{\rm tl}) \tag{10}$$

$$I_{\rm sleep} = \mu_n \times C_{\rm ox}\left(\frac{W}{L}\right)_{\rm sleep} \times \left((V_{\rm DD} - V_{\rm th}) \times V_x - \frac{V_x^2}{2}\right). \tag{11}$$

For the word line driver, $V_{\rm tl}$ is the same as $V_{\rm th}$ of sleep transistor

$$I_{\rm sleep} = \mu_n \times C_{\rm ox}\left(\frac{W}{L}\right)_{\rm sleep}\left(\left(\left(\frac{\ln(1-\frac{\beta}{100})}{\alpha}\right) - \frac{\left(\frac{\ln\left(1-\frac{\beta}{100}\right)}{\alpha}\right)^2}{2}\right)(V_{\rm DD} - V_{\rm th})^2\right) \tag{12}$$

$I_{\rm sleep}$ is the maximum current flowing through the ground, which is the total discharge current of the word line driver. Assuming each inverter stage is sized a times larger than the previous stage the total discharge current is expressed as

$$\begin{aligned} I_{\rm discharge} &= \frac{(V_{\rm DD} - V_x))}{R_{{\rm eq},n}} + \frac{(V_{\rm DD} - V_x))}{\frac{R_{{\rm eq},n}}{a^2}} + \frac{(V_{\rm DD} - V_x))}{\frac{R_{{\rm eq},n}}{a^4}} + \cdots \\ &= \frac{(V_{\rm DD} - V_x))}{R_{{\rm eq},n}} \times (1 + a^2 + a^4 + \ldots). \end{aligned} \tag{13}$$

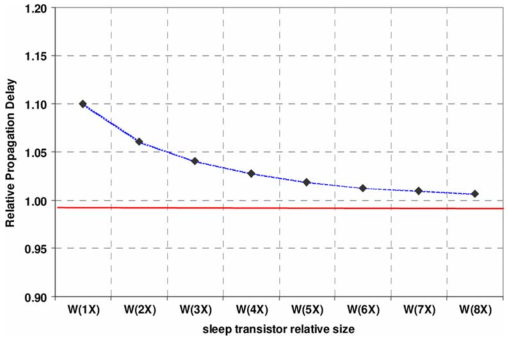


Fig. 13. Impact of sleep transistor sizing on propagation delay.

TABLE I
IMPACT OF SLEEP TRANSISTOR SHARING AND SIZING ON THE WAKEUP DELAY

#shared inverter chains	W(1X) (ns)	W(2X) (ns)	W(3X) (ns)	W(4X) (ns)	W(5X) (ns)	W(6X) (ns)	W(7X) (ns)	W(8X) (ns)
1	0.256	0.137	0.093	0.064	0.045	0.037	0.032	0.029
2	0.620	0.367	0.273	0.205	0.155	0.136	0.124	0.115
3	1.190	0.732	0.583	0.464	0.381	0.345	0.321	0.309
4	1.655	1.072	0.877	0.736	0.637	0.596	0.564	0.556
5	2.130	1.438	1.214	1.065	0.952	0.905	0.884	0.882
6	2.595	1.817	1.609	1.453	1.336	1.298	1.277	1.275
7	3.050	2.196	1.983	1.830	1.739	1.708	1.699	1.696
8	3.525	2.609	2.432	2.291	2.203	2.178	2.171	2.170
9	4.010	3.036	2.887	2.767	2.695	2.675	2.667	2.663
10	4.450	3.471	3.338	3.235	3.182	3.168	3.163	3.160

The size of sleep transistor can be found by substituting the discharge current into (12), shown in

$$\left(\frac{W}{L}\right)_{\text{sleep}} = \frac{I_{\text{discharge}}}{\mu_n \times C_{\text{ox}} \left(\left(\left(\frac{\ln\left(1-\frac{\beta}{100}\right)}{\alpha} \right) - \frac{\left(\frac{\ln\left(1-\frac{\beta}{100}\right)}{\alpha} \right)^2}{2} \right) (V_{\text{DD}} - V_{\text{th}})^2 \right)} \tag{14}$$

In Fig. 13 we report the impact of sleep transistor sizing on propagation delay for zz-hvs circuit when 10 rows of wordline drivers share sleep transistor. As results shown increasing the size of sleep transistor reduces the propagation delay overhead. As shown by increasing the size of sleep transistor (by 8X for instance) we can reduces the impact on propagation delay from 10% down to almost 1%.

In Table I we report the impact of sleep transistor sharing and sizing on wakeup delay when each wordline drives 256 one-bit memory cells. More sharing of sleep transistor results in larger wakeup delay. Increasing the size of sleep transistor reduces the wakeup delay.

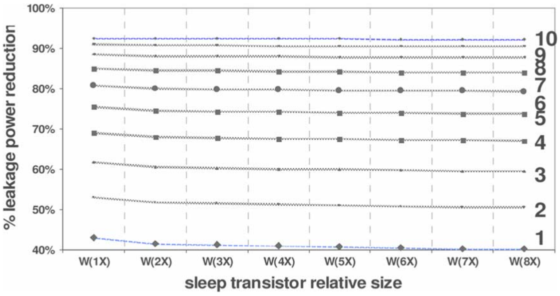


Fig. 14. Leakage power reduction relative to the total leakage power of peripheral circuits.

In Fig. 14 we report the leakage power reduction relative to the total leakage power of peripheral circuits as a function of sleep transistor size, and the number of sharing inverter chains. Results show that the sleep transistor size has a small impact on leakage power savings.

D. Impact of Sleep Transistor Insertion on Peripheral Circuit Layout and on Power Grid Network

Area efficiency and routability of sleep transistor is another critical factor in the implementation of MZZ-HVS circuit technique. To better evaluate and understand the impact of sleep transistor insertion in the peripheral circuit we first provide an overview of the standard cache architecture layout. Fig. 15 shows the high level composition of a cache memory [22]. The cache array is composed of multiple identical banks. Each bank is composed of multiple identical sub banks. Only one sub banks can be active at a time. A sub bank is also composed of multiple identical mats. The mat itself is a memory structure with a memory cell array, a row decoder, a wordline driver, a precharge circuit, local input and output drivers, a sense amplifier and a bitline multiplexer (Fig. 15(c)). The area of the cache memory (data array part) is estimated based on the area of a single bank $(\mathrm{W_{Bank}} \times \mathrm{H_{Bank}})$ and the area occupied in routing the address and the data to/from the banks [22]. Address and data are typically routed in an H-Tree distribution network to the mats [22]. Fig. 16(a) shows the layout configuration of horizontal routing of address and data to each of memory cell array. The drivers are typically placed at each of the nodes in the H-tree (V and H node in the figure). Fig. 16(b) shows the layout configuration of wires for the horizontal H-tree assuming that the wires are laid out using single layer of metal. Fig. 17 shows the vertical h-tree routing layout for address and data. The area of a mat shown in Fig. 15(b) can be calculated as $(2\mathrm{W_{Bank}} + \mathrm{P_{wires}}) \times (2\mathrm{H_{Bank}} + \mathrm{P_{wires}})$. The size of $\mathrm{P_{wire}}$ is decided by the pitch of routed wires [22]. As shown in Fig. 16(b) pitch of all routed wires is equivalent to

$$\begin{aligned} \mathrm{P_{wires}} &= \mathrm{P_{wires}} \times \mathrm{N_{wires-routed}} \\ &= \mathrm{P_{wires}} \times (\mathrm{N_{bank-addr-bits}} + \mathrm{N_{bank-datain-bits}} \\ &\quad + \mathrm{N_{bank-dataout-bits}} + \mathrm{N_{way-select-signals}}) \end{aligned} \tag{15}$$

where $\mathrm{N_{way-select-signals}}$ is the number of way-select bits to a mat, $\mathrm{N_{bank-addr-bits}}$ is the number of address bits to a bank, $\mathrm{N_{bank-datain-bits}}$ is the number of data-in bits to a mat and $\mathrm{N_{bank-dataout-bits}}$ is the number of data-out bits from a mat. Inserting sleep transistor in the input and output drivers comes

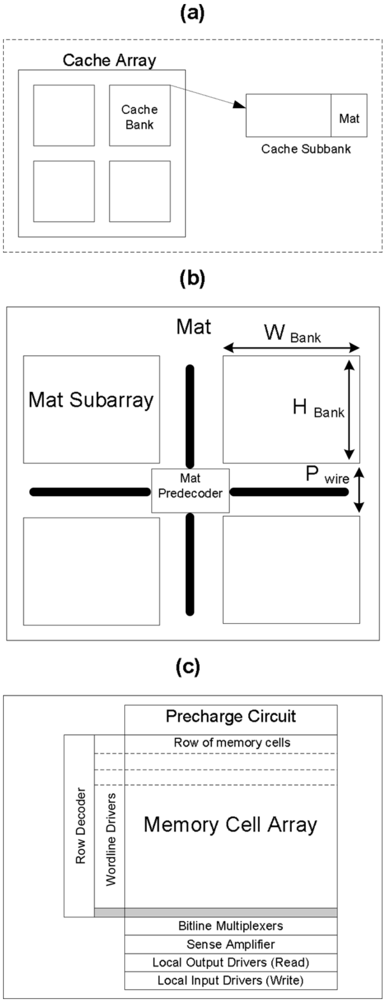


Fig. 15. High Level Composition of a cache memory.

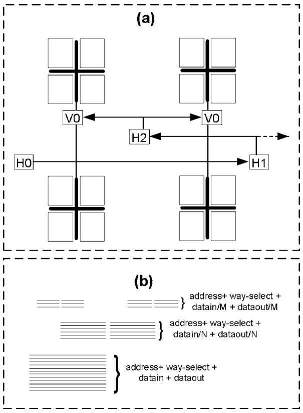

(a)
V0
V0
H2
H0
H1
(b)
address+ way-select + datain/M + dataout/M
address+ way-select + datain/N + dataout/N
address+ way-select + datain + dataout

Fig. 16. (a) Layout of horizontal H-tree address and data routed to the memory cell array. (b) Layout of wires for horizontal H-tree.

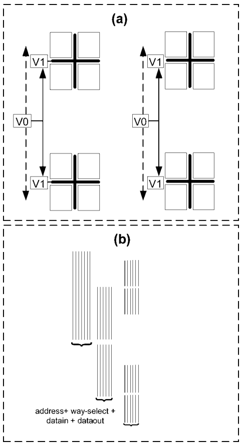


Fig. 17. Layout of vertical H-tree address and data routed to the memory cell array. (b) Layout of wires for vertical H-tree.

with an additional small area overhead in the layout. The effective wire pitch after inserting the sleep transistor in the layout is

$$
\begin{aligned}
&\mathrm{P}_{\mathrm{wires-+sleep-transistor}} \\
&\quad = \mathrm{P}_{\mathrm{wires}} \times \mathrm{N}_{\mathrm{wires-routed}} + \mathrm{H}_{\mathrm{sleep-transistor}} \\
&\quad = \mathrm{P}_{\mathrm{wires}} \times (\mathrm{N}_{\mathrm{bank-addr-bits}} \\
&\qquad + \mathrm{N}_{\mathrm{bank-datain-bits}} + \mathrm{N}_{\mathrm{bank-dataout-bits}} \\
&\qquad + \mathrm{N}_{\mathrm{way-select-signals}}) + \mathrm{H}_{\mathrm{sleep-transistor}}
\end{aligned} \tag{16}
$$

where $\mathrm{H}_{\mathrm{sleep-transistor}}$ is the height of sleep transistor gate. Using CACTI [22] and based on the methodology presented in [49] the effective wire routing pitch after sleep transistor insertion increases by 2% to 4% depending on the number of bits of input address and the number of in/out data bits (we assume that at most 10 line of data output drivers can share one sleep transistor).

To find out the area impact of sleep transistor insertion in the wordline driver we refer to Fig. 15(c). Wordline drivers are typically pitch matched to the memory cell. As a result, the height of the wordline drivers block shown in Fig. 15 is equal to the height of memory cell array. Inserting one sleep transistor and sharing it for the entire local wordline drivers requires an extra area in the layout as highlighted in Fig. 15(c). Assuming that the sleep transistor is shared both horizontally and vertically, it is inserted at the bottom of the wordline drivers block. Note that we could also implement sleep transistor as rows of parallel transistors placed at one side of the memory cell array to pitch match the cell [14]. The latter implementation is for the case where sleep transistor is shared only horizontally and thus incur minimal area overhead.

The area at the bottom of the memory cell block is thus empty. Note that the height of highlighted area is smaller than the height of one row of memory cells. The extra area has a height of a gate (sleep transistor size) which is much smaller in comparison to the height of a memory cell. Using CACTI [22] and based on the methodology explained in [49] the area overhead estimated to vary from 3% to 5% assuming that a mat contains 4 to 10 rows of memory cells (5% corresponds to the case where a mat has 10 rows of memory cells and all word line driver rows are sharing one sleep transistor). Therefore, inserting sleep transistor increases the effective area of a bank by 2% to 5%.

In Tables II and III we report the area for various components of a 128 KB of an L1 and a 2 MB of an L2 cache. The area overhead found to be smaller for L2. This in fact is due to a large wire pitch in an L2 cache which is a result of a large number of address and data in/out signal.

Altogether, inserting sleep transistor in the wordline drivers and input output address/data drivers increases the effective

TABLE II
AREA OF VARIOUS COMPONENTS OF A 128 KB L1 CACHE BEFORE AND AFTER SLEEP TRANSISTOR INSERTION

L1 Data Cache	Pre sleep transistor insertion area	Post sleep transistor insertion area
routing_area_width_ within_bank (micron)	76.32	77.46
routing_area_height_ within_bank (micron)	218.16	222.52
subarray_memory_cell_ area_height (micron)	112.89	114.92
subarray_memory_cell_ area_width (micron)	199.8	199.8
mat_width (micron)	431.04	432.49
mat_height (micron)	258.10	261.21
area (mm2)	1.43	1.45

TABLE III
AREA OF VARIOUS COMPONENTS OF A 2 MB L2 CACHE BEFORE AND AFTER SLEEP TRANSISTOR INSERTION

L2 Cache	Pre sleep transistor insertion area	Post sleep transistor insertion area
routing_area_width_ within_bank (micron)	1635.84	1650.43
routing_area_height_ within_bank (micron)	823.68	837.06
subarray_memory_cell_ area_height (micron)	112.89	114.92
subarray_memory_cell_ area_width (micron)	199.8	199.8
mat_width (micron)	430.87	431.26
mat_height (micron)	270.89	271.16
area (mm2)	28.08	28.52

height and the effective width of a mat. Using CACTI area model we estimated the increase to be 1% to 3% depending on the number of data and address in/out bits.

Finally it should be noticed that routing virtual ground node of sleep transistor is as difficult as routing the VSS and VDD in a design with no sleep transistor. The same metal line used to route the true ground (VSS) and true source voltage (VDD) is being used to route the virtual ground node. The additional routing overhead due to sharing sleep transistor is small as indicated in [50], [51] and can be reduced by as much as 65% by technique such as [50].

VII. MSLEEP-SHARE: ZZ-HVS CIRCUITS PLUS ARCHITECTURAL CONTROL

Some of the highest leakage units in the processor are the on-chip caches (such as L1 and L2). Thus, it makes a lot of sense to apply the MZZ-HVS circuit technique in these units. Note that we assume the small impact of MZZ-VHS on propagation delay (1%) can be tolerated and hidden in deep pipeline memories such as L1 and L2 caches and thus it does not degrade their operating frequency.

This section briefly describes an integrated architectural approach called MSleep-Share, to control the multiple sleep mode zig-zag share circuits in L1 and L2 caches. While it is possible to apply MZZ-HVS to other SRAM units such as the register file, reorder buffer, instruction cache, branch predictor, DTLB and ITLB, this paper studies the application in DL1 and L2 caches only.

TABLE IV
DL1 AND L2 CACHE PERIPHERALS MULTIPLE SLEEP MODES NORMALIZED LEAKAGE POWER SAVINGS AND WAKEUP DELAY OVERHEAD

power mode	**L1 Cache**			**L1 Cache**		
	STL delay (cycle)	leakage reduction relative to peripheral leakage (%)	leakage Reduction relative to total cache leakage (%)	STL delay (cycle)	leakage reduction relative to peripheral leakage (%)	leakage reduction relative to total cache leakage (%)
basic-lp	1	42%	38%	2	46%	42%
aggr-lp	2	58%	52%	3	54%	49%
ultra-lp	7	79%	72%	9	93%	84%

As explained above, there is a latency associated with waking up the SRAM peripheral circuitry. The overall time delay for transition to/from the standby mode, STL, is the sum of sleep transistors wakeup delay and a propagation delay of the sleep signal. Both of these delays increase as the memory area increases, especially for the latter delay, because the sleep signal needs to be transmitted over a greater distance. Accordingly, depending on the memory size and configuration, there is a different wakeup delay overhead for a specific MZZ-HVS bias voltage. To find the STL delay for an SRAM array, SPICE and CACTI were used to measure the wakeup delay of a sleep transistor and the propagation delay of the sleep signal, respectively. To estimate the propagation delay we assumed that the sleep signal has to be transmitted across the SRAM peripherals. Based on these experimental results different sleep modes were defined for DL1 and L2 caches (see Table IV). The first sleep mode is the basic-lp mode in which the STL delay is dominated by the sleep signal propagation delay. Aggr-mode is the second sleep mode with a larger power reduction compared to the basic-lp but at a higher wakeup delay cost; 2 cycles (1 cycles sleep transistor wakeup +1 cycles sleep signal propagation delay) for the DL1 cache and 3 cycles $(1+2)$ for the L2 cache. The highest saving power mode is the ultra-lp, which shuts down the peripheral circuitry completely through $\mathrm{V_{ss}}$ and $\mathrm{V_{dd}}$ bias selection for nMOS and pMOS sleep transistors, respectively. More precisely, the proposed scheme achieves 79% leakage power reduction relative to peripheral leakage (with no leakage mechanism control) in the DL1 cache and 93% power reduction in the L2 cache (with no leakage mechanism control).

Given these wakeup delays, the key issue is thus how to avoid loss of processor performance while benefiting the most from the largest power saving modes.

To maximize the leakage reduction in each of DL1 and L2 caches peripherals one solution would be to always put them into the ultra low power mode. However, this requires a wakeup of their peripheral circuits before accessing them adding at least 7 cycles to access latency and significantly reducing performance. Alternatively, one can put SRAM peripherals into the basic low power mode, which requires 2 cycles of wakeup delay and reduces the performance loss. However, this doesn't significantly reduce leakage power (see Table IV). To achieve the large leakage reduction of ultra and aggressive low power modes with the minimal performance impact of basic-lp mode one has to dynamically change the peripheral circuit sleep modes. During periods of frequent access they need to be kept in the basic-lp mode (to limit performance loss) and when their access frequency is low they can be kept in aggr-lp or ultra-lp modes.

One period of infrequent access to any of the relevant units is when an L2 cache miss occurred which can result in a processor stall. Thus, one approach is to turn off the peripheral circuitry once the processor becomes idle. Such idle periods can occur frequently during program execution. As shown in recent work, when an L2 cache miss occurs the processor executes a number of miss-independent instructions and then stalls [6], [10], [37], [38]. We refer to the interval between an L2 cache miss occurring and the processor stalling as a pre-stall period. The processor stays idle until the L2 cache miss is serviced. This may take hundreds of cycle (300 cycles for our processor model which is similar to the Intel® Core™ 2 Duo processor). It should be noted that during pre-stall period processor still executes some instructions until its resources fill up (such as reorder buffer, instruction queue and load/store queue). As a result, the processor stalls for a large fraction of the L2 cache miss service time. During such a stall period there is no access to L1 and L2 caches and they can be put into ultra low-power mode. Since processor performance decrease noticeably during the pre-stall period, and the number of accesses to DL1 and L2 reduces significantly [6], [10] we thus propose to put DL1 and L2 cache peripheral into aggr-lp mode during such period. MSleep-share detects the stall period as following:

The instruction queue and functional units of the processor are monitored after an L2 miss. The sleep signal is asserted to the cache peripheral circuits if the instruction queue has not issued any instructions and functional units have not executed any instructions for K consecutive cycles ($K = 10$). Given a 7 to 9 cycle wakeup latency for the DL1 and L2 cache peripherals, the sleep signal is de-asserted 7 and 9 cycles before the miss service is completed. Note that the memory access latency is assumed to be known a-priori. In a non-deterministic memory latency model the cache miss return will trigger the wakeup signal. This would contribute a small delay of 7 to 9 cycles to the overall delay of L2 cache miss service time. To further improve the leakage reduction, we propose to always put L2 cache peripherals in the basic-lp low power mode. This mode adds 2 cycles to the L2 cache access latency. Considering the delay of accessing L2 which is already 20 cycles, the additional 2 cycles have a small impact on performance (see results below). The DL1 cache is accessed more frequently than the L2 and thus its peripherals cannot be kept in a low power mode as this may degrade the performance noticeably. However, the DL1 cache read ports are accessed more frequently than its write ports. Results in Table V show that on average a DL1 write port is accessed once every 30 cycles, while a read port is accessed every 8 cycles. Such different access patterns require different control mechanism for reducing leakage. Our evaluation showed that making a write port one cycle slower (the wakeup delay of basic-lp mode) has a very small impact on performance. But a one extra delay cycle on every DL1 read (port) leads to a noticeable performance degradation in some of the benchmarks (e.g., gzip, twolf and vpr). Therefore, the DL1 write port is always kept in the basic-lp mode and is awakened only when it is accessed.

During the pre-stall period the DL1 peripherals are put into aggr-lp mode. Both read and write ports are put into the ultra-lp mode once the processor is stalled. Fig. 18 shows the general state machine control to put DL1 and L2 cache peripherals into different low power modes.

TABLE V
AVERAGE TIME BETWEEN ACCESSES TO DL1 READ AND WRITE PORTS (CYCLES)

	DL1 read	DL1 write		DL1 read	DL1 write
ammp	15.8	73.8	**lucas**	22.7	58.1
applu	10.5	23.6	**mcf**	28.9	203.8
apsi	5.4	11.1	**mesa**	3.7	9.8
art	6.3	29.5	**mgrid**	7.0	52.9
bzip2	4.2	14.3	**parser**	6.4	18.2
crafty	3.2	17.8	**perlbmk**	5.9	11.5
eon	3.9	6.02	**sixtrack**	3.7	10.3
equake	5.7	14.1	**swim**	18.1	44.8
facerec	6.1	10.8	**twolf**	6.6	22.2
galgel	3.5	33.6	**vortex**	4.0	7.6
gap	7.8	15.5	**vpr**	6.6	20.8
gcc	5.8	8.9	**wupwise**	8.9	18.2
gzip	5.3	19.7	**average**	8.2	30.4

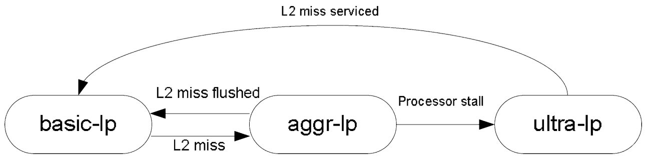
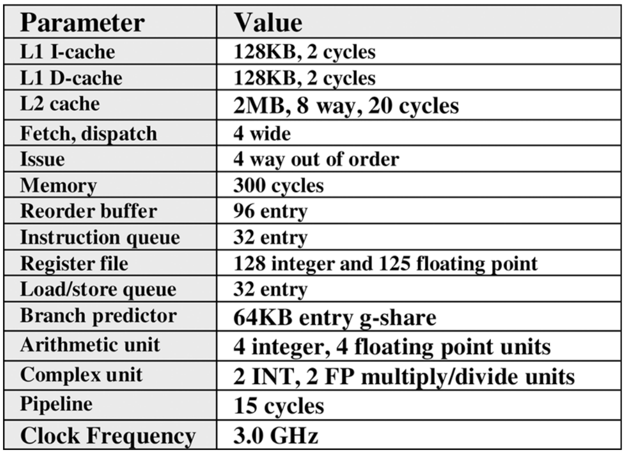


Fig. 18. State machine to control MZZ-HVS for DL1 (write port) and L2 cache.

TABLE VI
PROCESSOR ORGANIZATION



Parameter	Value
L1 I-cache	128KB, 2 cycles
L1 D-cache	128KB, 2 cycles
L2 cache	2MB, 8 way, 20 cycles
Fetch, dispatch	4 wide
Issue	4 way out of order
Memory	300 cycles
Reorder buffer	96 entry
Instruction queue	32 entry
Register file	128 integer and 125 floating point
Load/store queue	32 entry
Branch predictor	64KB entry g-share
Arithmetic unit	4 integer, 4 floating point units
Complex unit	2 INT, 2 FP multiply/divide units
Pipeline	15 cycles
Clock Frequency	3.0 GHz

MSleep-share technique is evaluated by simulating a processor described in Table VI. The architecture was simulated using an extensively modified version of SimpleScalar 4.0 [11] using SPEC2K benchmarks. Benchmarks were compiled with the -O4 flag using the Compaq compiler targeting the Alpha 21264 processor. The benchmarks were fast-forwarded for 3 billion instructions, then fully simulated for 4 billion instructions using the reference data sets.

Fig. 19 shows the leakage power reduction for DL1 and L2 caches. The leakage in DL1 and L2 caches is reduced by up to 62% (applu) and 72% (ammp) respectively. On average, the leakage power is reduced by 40% and 54% for the DL1 and L2 caches, respectively. An interesting observation is that the reduction due to the ultra-lp mode is very significant and is the highest of all low-leakage modes in many cases. This is because all on-chip SRAM units transition to this mode when the processor stalls. The large number of stalls ([6], [10]), combined with the large leakage savings associated with the ultra-lp mode

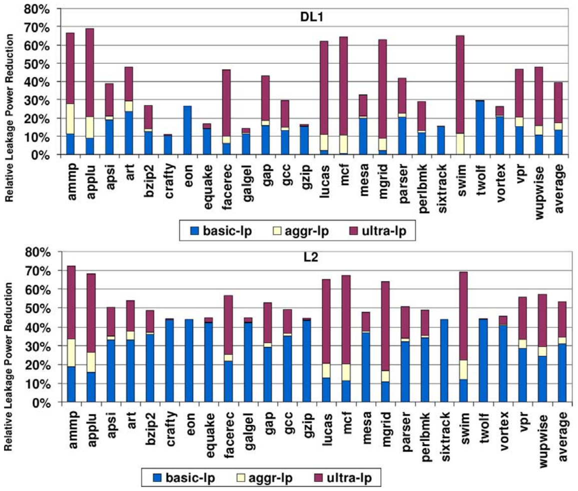


Fig. 19. Leakage power reduction for DL1 and L2 cache.

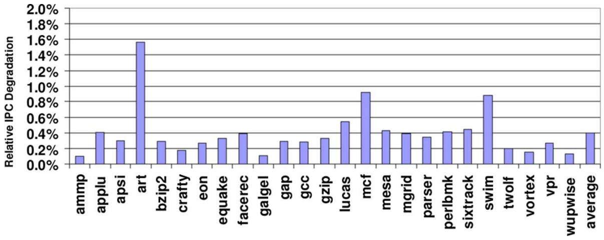


Fig. 20. Performance degradation of applying MSleep-Share in DL1 and L2.

make this mode the major source of leakage reduction for all SRAM units. Exceptions are crafty, eon, gzip and sixtrack with almost no leakage savings of ultra-lp mode. The reason is that in these benchmark the L2 cache miss rate is almost negligible (see [6], [10]).

Fig. 20 shows the performance degradation in terms of IPC (committed instruction per cycles) when applying MZZ-HVS and controlling it using Msleep share in the DL1 and L2 caches. The performance reduction is minimal, far below 1%, across most of the benchmarks. The exceptions are art, mcf and swim with 1.6, 0.92 and 0.88% IPC drop, respectively. In fact art, mcf and swim have large DL1 cache miss rates and hence L2 is accessed very frequently (almost one out of every two DL1 accesses is a miss in art). Recalling that L2 is always kept in a basic-lp mode which incurs a 2 cycle wakeup delay before access, explains the performance degradation in these benchmarks. Adaptive L2 mode control may help alleviate this problem.

VIII. Conclusion

This paper proposed a circuit design, zig-zag share, to reduce leakage in SRAM memory peripheral circuits. zig-zag share reduces peripheral leakage by up to 40X with only a small increase in memory area and delay. A new technique, MSleep-Share, uses architectural control of zig-zag share circuits in the L1 and L2 cache peripherals. Our results show that using MSleep-Share the leakage in L1 and L2 caches is reduced, on average, by 54% for the L2 cache and 40% for the L1 cache, respectively and up to 72% and 62%, respectively, across SPEC2K benchmarks.

References

[1] D. Nicolaescu, B. Salamat, A. Veidenbaum, and M. Valero, "Fast speculative address generation and way caching for reducing L1 data cache energy," in *Proc. ICCD*, 2006, pp. 101–107.

[2] R. Ba, N. S. Kim, D. Sylvester, and T. Mudge, "Total leakage optimization strategies for multi-level caches," in *Proc. ACM Great Lakes Symp. VLSI*, 2005, pp. 129–136.

[3] T. Skotnicki, J. A. Hutchby, K. Tsu-Jae, H.-S. P. Wong, and F. Boeuf, "The end of CMOS scaling: Toward the introduction of new materials and structural changes to improve MOSFET performance," *IEEE Circuits Devices Mag.*, vol. 21, no. 1, pp. 16–26, Jan.–Feb. 2005.

[4] C. H. Kim, J. J. Kim, S. Mukhopadhyay, and K. Roy, "A forward body-biased low-leakage SRAM cache: Device, circuit and architecture considerations," *IEEE Trans. VLSI Syst.*, vol. 13, no. 25, pp. 349–357, Aug. 2005.

[5] S. Borkar, P. Dubey, K. Kahn, D. Kuck, and H. Mulder, "Platform 2015: Intel® processor and platform evolution for the next decade," *Intel Technology Mag.*, Mar. 2005.

[6] H. Homayoun, M. Makhzan, J.-L. Gaudiot, and A. Veidenbaum, "A centralized cache miss driven technique to improve processor power dissipation," in *Proc. Int. Symp. SAMOS VIII*, 2009.

[7] J. M. Rabaey, A. Chandrakasan, and B. Nikolić*, Digital Integrated Circuits: A Design Perspective*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 2003.

[8] Y. Nakagome *et al.*, "Review and future prospects of low-voltage RAM circuits," *IBM J. Res. Develop.*, vol. 47, no. 5, pp. 525–552, Sep. 2003.

[9] J. Kao, S. Narendra, and A. Chandrakasan, "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," in *Proc. DAC*, 1998, pp. 495–500.

[10] A. Kejariwal *et al.*, "Comparative architectural characterization of SPEC CPU2000 and CPU2006 benchmarks on the intel core 2 duo processor," in *Proc. Int. Symp. SAMOS VIII*, 2008, pp. 132–141.

[11] *SimpleScalar4 Tutorial.* [Online]. Available: http://www.simplescalar.com/tutorial.html

[12] G. Gerosa *et al.*, "A sub-1 W to 2 W low-power IA processor for mobile internet devices and ultra-mobile PCs in 45 nm Hi-K metal gate CMOS," in *Proc. ISSCC-2008*, pp. 73–82.

[13] M. D. Powell *et al.*, "Gated V_{dd}: A circuit technique to reduce leakage in deep-submicron cache memories," in *Proc. IEEE ISLPED*, 2000, pp. 90–95.

[14] A. Agarawal *et al.*, "DRG-cache: A data retention gated-ground cache for low power," *Proc. DAC*, pp. 473–478, 2002.

[15] K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka, "Power gating with multiple sleep modes," in *Proc. ISQED*, 2006, pp. 633–637.

[16] K. Flautner *et al.*, "Automatic performance setting for dynamic voltage scaling," *J. Wireless Netw.*, vol. 8, no. 5, pp. 260–271, 2001.

[17] M. Mamidipaka, K. S. Khouri, N. Dutt, and M. S. Abadir, "Analytical models for leakage power estimation of memory array structures," in *Proc. CODES+ISSS*, 2004, pp. 146–151.

[18] B. S. Amrutur *et al.*, "Speed and power scaling of SRAMs," *IEEE J. Solid State Circuits*, vol. 35, no. 2, pp. 175–185, Feb. 2000.

[19] S. Kaxiras *et al.*, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in *Proc. IEEE-ISCA*, 2001, pp. 240–251.

[20] B. S. Amrutur *et al.*, "A replica technique for wordline and sense control in low-power SRAM's," *IEEE J. Solid-State Circuits*, vol. 33, no. 8, pp. 1208–1219, Aug. 2000.

[21] K. Flautner *et al.*, "Drowsy caches: Simple techniques for reducing leakage power," in *Proc. IEEE ISCA*, 2002, pp. 148–157.

[22] *Cacti5*, [Online]. Available: http://quid.hpl.hp.com:9082/cacti/

[23] S. Rusu *et al.*, "A 65-nm dual-core multithreaded Xeon® processor with 16-MB L3 cache," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 17–25, 2007.

[24] J. C. Park and V. J. Mooney III, "Sleepy stack leakage reduction," *IEEE Trans. VLSI Syst.*, vol. 14, no. 11, pp. 1250–1263, Nov. 2006.

[25] M. Powell *et al.*, "Gated-V_{dd} : A circuit technique to reduce leakage in deep-submicron cache memories," in *Proc. IEEE ISLPED*, 2000, pp. 90–95.

[26] K. Nii *et al.*, "A low power SRAM using auto-backgate-controlled MT-CMOS," in *Proc. ISLPED*, 1998, pp. 293–298.

[27] S. H. Kim and V. J. Mooney, "Sleepy keeper: A new approach to low-leakage power VLSI design," in *Proc. VLSI-SoC*, 2006, pp. 367–372.

[28] W. Zhang and J. S. Hu, "Compiler-directed instruction cache leakage optimization," in *Proc. MICRO-35*, 2002, pp. 208–218.

[29] Y. Meng, T. Sherwood, and R. Kastner, "On the limits of leakage power reduction in caches," in *Proc. HPCA-11*, 2005, pp. 154–165.

[30] H. Kawaguchi, K. Nose, and T. Sakurai, "A super cut-off CMOS (SC-CMOS) scheme for 0.5-V supply voltage with picoampere stand-by current," *IEEE J. Solid State Circuit*, vol. 35, no. 10, pp. 1498–1501, Oct. 2000.

[31] K.-S. Min *et al.*, "Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: An alternative to clock-gating scheme in leakage dominant era," in *Proc. ISSCC*, 2003, pp. 400–502.

[32] Y. Takeyama *et al.*, "A low leakage SRAM macro with replica cell biasing scheme," *IEEE J. Solid- State Circuits*, vol. 41, no. 4, p. , Apr. 2006.

[33] K. Nii *et al.*, "A 90-nm low-power 32 KByte embedded SRAM with gate leakage suppression circuit for mobile applications," *IEEE J. Solid-State Circuits*, vol. 39, no. 4, pp. 684–693, Apr. 2004.

[34] E. Grossara *et al.*, "Statistically aware SRAM memory array design," in *Proc. ISQED-2006*, p. 30.

[35] *International Technology Roadmap for Semiconductors*, Semiconductor Industries Association, 2005 [Online]. Available: http://www.itrs.net/

[36] S. Somvanshi and S. Kasavajjala, "A low power sub-1 V CMOS voltage reference," in *Proc. 2008 IEEE Int. SOC Conf.*, pp. 271–276.

[37] D. Marculescu, "On the use of microarchitecture-driven dynamic voltage scaling," in *Proc. Workshop on Complexity-Effective Design*, 2000.

[38] H. Li, C.-Y. Cher, T. Vijaykumar, and K. Roy, "VSV: L2-miss-driven variable supply-voltage scaling for low power," in *Proc. Int. Symp. Microarchitecture*, Dec. 2003, pp. 19–28.

[39] F. Fallah and M. Pedram, "Standby and active leakage current control and minimization in CMOS VLSI circuits," *IEICE Trans. Electron.*, vol. E88-C, no. 4, pp. 509–519, Apr. 2005.

[40] S. Mutoh, S. Shigematsu, Y. Gotoh, and S. Konaka, "Design method of MTCMOS power switch for lowvoltage high-speed LSIs," in *Proc. Asian South Pacific Design Autom. Conf.*, 1999, pp. 113–116.

[41] V. Khandelwal and A. Srivastava, "Leakage control through fine-grained placement and sizing of sleep transistors," in *Proc. ACM/IEEE Int. Conf. Computer Aided Design*, 2004, pp. 533–536.

[42] B. H. Calhoun, F. A. Honore, and A. Chandrakasan, "Design methodology for fine-grained leakage control in MTCMOS," in *Proc. Int. Symp. Low Power Electron. Design*, 2003, pp. 104–109.

[43] A. Ramalingam, B. Zhang, A. Davgan, and D. Pan, "Sleep transistor sizing using timing criticality and temporal currents," in *Proc. ASP-DAC*, 2005.

[44] V. Khandelwal and A. Srivastava, "Leakage control through fine-grained placement and sizing of sleep transistors," in *Proc. ACM/IEEE Int. Conf. Comput. Aided Design*, 2004, pp. 533–536.

[45] M. Margala, "Low power SRAM circuit design," in *Proc. IEEE Int. Workshop Memory Technol., Design, Testing*, 1999, pp. 115–122.

[46] K. W. Mai, T. Mori, B. S. Amrutur, R. Ho, B. Wilburn, M. A. Horowitz, I. Fukushi, T. Izawa, and S. Mitarai, "Low-power SRAM design using half-swing pulse-mode techniques," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, pp. 1659–1671, Nov. 1998.

[47] J. Tschanz, S. Narendra, Y. Ye, B. Bloechel, S. Borkar, and V. De, "Dynamic sleep transistor and body bias for active leakage power control of microprocessors," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 1838–1845, Nov. 2003.

[48] H. Homayoun, M. Makhzan, and A. Veidenbaum, "ZZ-HVS: Zig-zag horizontal and vertical sleep transistor sharing to reduce leakage power in on chip SRAM peripheral circuits," in *Proc. IEEE ICCD 2008*, pp. 699–706.

[49] H. Yoshida, K. De, and V. Boppana, "Accurate pre-layout estimation of standard cell characteristics," in *Proc. 41st Ann. Conf. Design Automation*, New York, 2004, pp. 208–211.

[50] K. Shi, Z. Lin, Y. Jian, and L. Yuan, "Simultaneous sleep transistor insertion and power network synthesis for industrial power gating designs," *J. Comput.*, vol. 3, no. 3, pp. 6–13, Mar. 2008.

[51] C. Long and L. He, "Distributed sleep transistor network for power reduction," in *Proc. IEEE/ACM Design Autom. Conf.*, 2003, pp. 181–186.

[52] K. Shi and D. Howard, "Challenges in sleep transistor design and implementation in low-power designs," in *Proc. IEEE/ACM Design Autom. Conf.*, 2006, pp. 113–116.

[53] A. Calimera *et al.*, "Optimal MTCMOS reactivation under power supply noise and performance constraints," *Proc. DATE-08: Design Automation and Test in Europe*, pp. 973–978, Mar. 2008.

Houman Homayoun (M'04) received the B.S. degree in electrical engineering in 2003 from Sharif University of technology, Tehran, Iran, the M.S. degree in computer engineering in 2005 from University of Victoria, BC, Canada, and the Ph.D. degree from the Department of Computer Science at the University of California Irvine in 2010.

Dr. Homayoun was named a 2010 National Science Foundation Computing Innovation Fellow by the Computing Research Association (CRA) and the Computing Community Consortium (CCC). He was a recipient of the 4-years UC-Irvine computer science department chair fellowship

Avesta Sasan (Mohammad A. Makhzan) (S'06) received the B.S. degree (*summa cum laude*) in computer engineering and the M.S. and Ph.D. degrees in electrical engineering from University of California Irvine in 2005, 2006, and 2010, respectively.

His research interest includes low power design, process variation aware architectures, fault tolerant computing systems, nano-electronic power and device modeling, VLSI signal processing, processor power and reliability optimization and logic-architecture-device co-design.

Alexander V. Veidenbaum (M'08) received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1985.

He is currently a Professor of Computer Science at the University of California, Irvine. His research interests include computer architecture for parallel, high-performance, embedded and low-power systems and compilers.

Dr. Veidenbaum is a member of the IEEE Computer Society and the ACM.

Hsin-Cheng Yao received the B.S. degree in electrical engineering from University of Illinois at Urbana-Champaign, in 2005, the M.S. degree in electrical engineering from University of Southern California, Los Angeles, in 2007, and he is currently working toward the Ph.D. degree from the Department of Electrical Engineering and Computer Science at the University of California Irvine.

Shahin Golshan received the B.S. degree in computer engineering from Sharif University of Technology, Tehran, Iran in 2005, and the M.S. degree in systems from the University of California, Irvine (UCI) in 2009, where he is currently working toward the Ph.D. Degree.

His research interests include VLSI/FPGA computer aided design, design automation for embedded systems (high level synthesis and physical design algorithms) with emphasis on low power and reliability, and reconfigurable computing.

Mr. Golshan is a member of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE).

Payam Heydari (M'00–SM'07) received the B.S. and M.S. degrees (hons) in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1992 and 1995, respectively. He received the Ph.D. degree from the University of Southern California, Los Angeles, in 2001.

He is currently a Full Professor of Electrical Engineering at the University of California, Irvine. During the summer of 1997, he was with Bell-Labs, Lucent Technologies where he worked on noise analysis in high-speed CMOS integrated circuits. He worked at IBM T. J. Watson Research Center on gradient based optimization and sensitivity analysis of custom ICs during the summer of 1998. He is the director of the Nanoscale Communication IC (NCIC) Lab. His research interests include design of ulta-high frequency analog/RF/mixed-signal integrated circuits. Results of the research in the NCIC Lab have appeared in more than 80 peer-reviewed journal and conference papers.