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

Houman Homayoun[‡], Mohammad Makhzan[†], Alex Veidenbaum[‡]
[†]*Department of Electrical and Computer Engineering, UC Irvine*
[‡]*Department of Computer Science, UC Irvine*
*{hhomayou,alexv}@ics.uci.edu {mmakhzan}@uci.edu*

***Abstract- Based on*** **Recent studies peripheral circuit (including decoders, wordline drivers, input and output drivers) constitutes a large portion of the cache leakage. In addition as technology migrate to smaller geometries, leakage contribution to total power consumption increases faster than dynamic power, promoting leakage as the largest power consumption factor.**

**This paper proposes zig-zag share, a circuit technique to reduce leakage in SRAM peripheral. Using architectural control of zig-zag share, an integrated technique called Sleep-Share is proposed and applied in L1 and L2 caches. The results show leakage reduction by up to 40X in deeply pipelined SRAM peripheral circuits, with only a 4% area overhead and small additional delay.**

## I. INTRODUCTION

CMOS technology scaling has been a primary factor utilized to increase the processor performance. A draw-back of this trend is increase of 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].

To overcome the growing leakage problem in SRAM memories, a number of technology, circuit, 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, retaining data but not allowing access. These techniques include usage of body bias control, reduced $V_{dd}$, Gated-$V_{dd}$ multiple $V_{th}$, 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].

Many recent studies [8][12][17] have shown that peripheral circuit, including decoders, wordline drivers input drivers, output drivers, and column decoders is responsible for considerable portion of leakage in SRAM memories. Figure 1 shows the leakage power breakdown for a 2MB L2 cache in a 64-bit, 3.4 GHz processor. These results were obtained using CACTI-5 [22] for 65nm 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. The reason is the use of larger, faster and leakier transistors in peripheral circuits to satisfy timing requirements, while smaller and less leaky transistors are used in memory cells. In fact, SRAM memory cells can be optimized for low leakage currents without a significant impact on the cell area or performance [8][25][26] (CACTI5 assumes that SRAM cell leakage reduction techniques have been applied). Thus approaches that concentrate on cell leakage power alone are insufficient and it is very important to address leakage in the peripheral circuits.

Unlike memory cells, peripheral circuits use very large transistors to drive high loads to meet the memory timing constraints. Thus applying traditional leakage reduction techniques in these circuits could introduce significant area and delays overhead. This is mainly due to the impact of leakage reduction techniques on peripheral circuits rise time, fall time, propagation delay and area overhead and will be discuss later in this work. 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 we will discuss in this work. In addition, some of the peripheral circuits are driving a value and cannot be simply disabled. For instance, a word-line driver has to drive a $V_{low}$ 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. 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 when 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 Spice in TSMC 65nm technology. The results show that the zig-zag share scheme reduces leakage by more than 95% while increasing peripheral circuits delay and area by 5% and 4%, respectively.

This work was supported in part by the NSF Award CCF-0811882

978-1-4244-2658-4/08/$25.00 ©2008 IEEE

Micro-architectural control of this circuit technique, i.e. when to assert and de-assert 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 Sleep-Share, adds architectural control of zig-zag share for L1 and L2 caches. Sleep- 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.

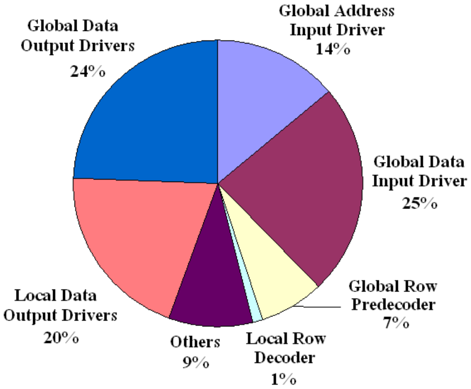

Figure 1: Leakage power components of an L2 cache access

The rest of this paper is organized as follows: Related work is described in Section 2. Section 3 provides some preliminaries. Section 4 presents the leakage control circuit scheme. Section 5 presents the experimental analysis of the zig-zag share circuit technique. Section 6 describes the architectural sleep-share technique and finally Section 7 concludes this work.

## II. Related Work

In this section we briefly review past work on reducing leakage power at technology, circuit, architecture and compiler/OS 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 in L1 cache memory cells. Voltage scaling can be combined with Frequency Scaling (DVFS) to reduce both dynamic and leakage power more effectively [10].

Another technique is Gated-$V_{dd}$ which turns off the supply voltage of memory cells by using a sleep transistor [25]. The advantage of this technique is in reducing the leakage power virtually completely. However, it doesn't retain the state of the memory cells. Similarly, Vss can also be gated (gated Vss).

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, [8] proposed a Replica Cell Biasing scheme in which the cell bias is not affected by $V_{dd}$ and $V_{th}$ of peripheral transistors.

[4][14] proposed a forward body biasing scheme (FBB) 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-$V_{dd}$ 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-$V_{dd}$ 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.

### B. *Architectural Techniques*

A number of architecturally driven cache leakage reduction techniques have been proposed. Powell et al proposed applying gated- $V_{dd}$ 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 $V_{dd}$ and $V_{ss}$ 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-$V_{dd}$ 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 Figure 1, peripheral circuits are equally if not more important to address in 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 ($I_{Dsub}$) flows from drain of a transistor to its source when the transistor is off (gate voltage is below the threshold voltage).

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

$$V_T = V_{T0} + \gamma(\sqrt{|(-2)\phi_F + V_{SB}|} - \sqrt{|2\phi_F|}) \qquad \text{Equation 1}$$

The parameter $\gamma$ is called the body effect coefficient. An effective way to reduce leakage of an off transistor is thus increasing its source voltage (for an NMOS increasing VSB). This can be done by stacking it with a sleep transistor as shown in Figure 2. By stacking transistor N with slpN the source to body voltage (VM ) of transistor N increases and thus reduces its subthreshold leakage current, when both transistors are off [9]. Such leakage reduction is related to the virtual ground voltage (VM) during sleep mode which in turn is determined by the size of the sleep transistor and its bias voltage (Vgslpn) [9][15]. Reducing the sleep transistor bias reduces the leakage while increasing the circuit transition wakeup period (waking up transistor N requires pulling down the voltage VM to ground). Thus there is a trade-off 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, mainly the circuit rise time, fall time and propagation delay. Also, stacking sleep transistor would make all high voltage nodes discharge during the stand-by mode due to the large leakage current of the circuit [30]. Such instability has to be addressed for circuits which have to keep a specific state during stand-by mode, e.g. wordline drivers of SRAM memories.

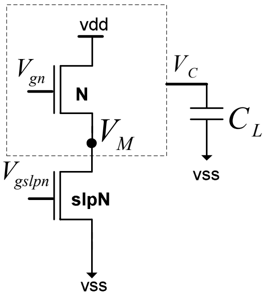

Figure 2. Stacking sleep transistor to reduce leakage

## IV. Sleepy peripherals

Now Let us 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 can be treated similarly.

An analysis of sub-threshold leakage sources in a wordline driver implemented as a four-stage inverter (buffer) chain (shown in Figure 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. But rise and fall time and propagation delay of the circuit are affected. Next we discuss several approaches to minimize this effect while maximizing leakage reduction.

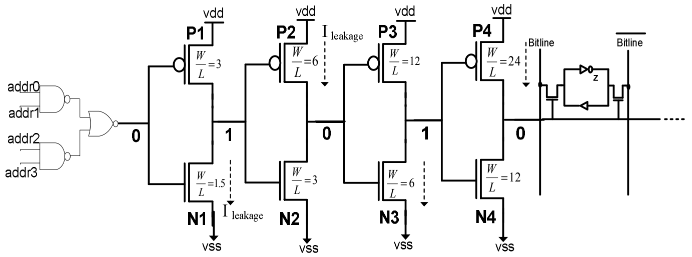

Figure 3. Leakage in the wordline driver

### A. A Redundant Circuit Approach

In this technique sleep transistors are stacked with all PMOS and NMOS transistors in the inverter chain to reduce their leakage while they turned off (Figure 4(a)). We refer to this approach as a redundant circuit scheme. As discussed above, inserting the sleep transistor introduces instability 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 such instability when the circuit is in sleep mode, we propose to insert a minimum size NMOS transistor in the last stage of the inverter chain, as shown in Figure 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.

The drawback of applying 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 Rpeq * CL and Rneq * CL respectively, where Rpeq is the equivalent resistance of the PMOS transistor, Rneq is the equivalent resistance of the NMOS transistor, and CL is the equivalent wordline output capacitive load [7]. Inserting sleep transistors increases R neq , R peq and thus the rise time and fall time of the wordline driver as well as its propagation delay.

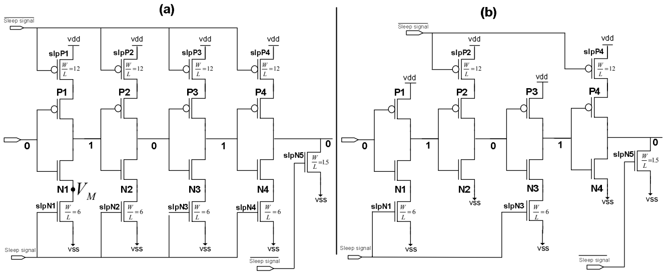

Figure 4. (a) redundant circuit (b) zig-zag circuit

While increasing the rise time and propagation delay is not desirable due to the impact on access time, increasing the fall time is not tolerable since it can affect memory functionality [18][20]. Any increase in the fall time of wordline driver results in an increase in pass transistor active period during a read operation. This results in the bitline over-discharge and the memory content over-charge during the read operation. Such over-discharge not only increases the dynamic power dissipation of bitlines but more importantly can cause a memory cell content to flip if the over-discharge period is large [7][20]. 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 bitline 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]. 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 and moreover not to increase bitline dynamic power dissipation we propose a different circuit 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 [31], as shown in Figure 4(b). By inserting the sleep transistors in a zig-zag fashion the Rpeq for the first and third inverters and Rneq 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 increased 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 Figure 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

Figure 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 RNeq. 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 Neq-zz-hs is thus equal to the R N1 + R 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 Neq-zz-hs is equal to: $R_{N1} + (R_{nlp-zz} / 2)$ and is smaller than the R Neq-zz which is $R_{N1} + R_{nlp-zz}$.

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.

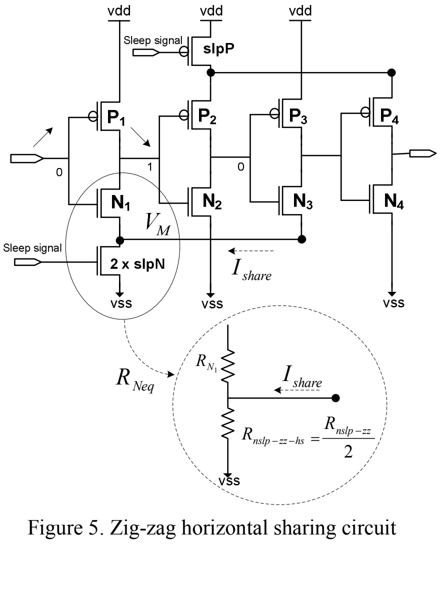

Figure 5. Zig-zag horizontal sharing circuit

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

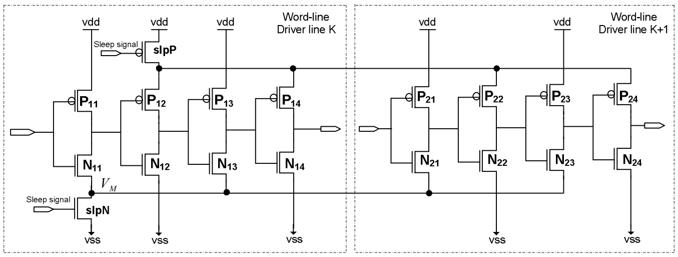

Figure 6. Zig-zag horizontal and vertical sharing circuit

To better explain this, Figure 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 of the leakage current of one of N11 or N21 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:

$$V_{M1} = \frac{n.\log_{10}^{\frac{W_{N11}}{W_{slpN}}} + \phi V_{dd} - V_{g0}}{2\phi}$$ where $\phi$ is the DIBL coefficient and n is the subthreshold slope. 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:

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

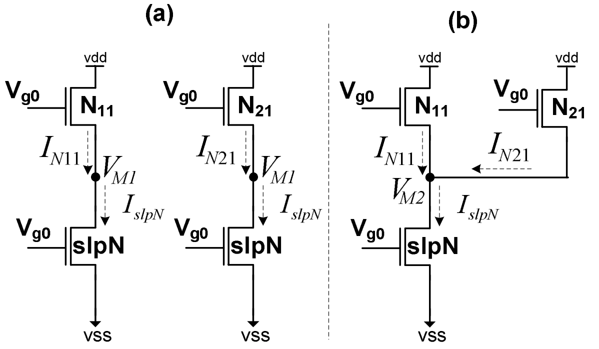

Figure 7. (a) zz-hs and (b) zz-hvs quivalent during sleep mode

## V. Circuit Evaluation

In order to evaluate the proposed zig-zag share approaches on memory peripherals, 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 65nm technology. The empirical results presented are for the leakage current, rise time and fall time, propagation 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 1.08V. The results for proposed circuits (except for zz-hvs) in comparison with baseline, redundant and zig-zag circuits are shown in Figure 8. To have a fair comparison between zig-zag and zig-zag-share scheme we also 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 compare to the zig-zag scheme. As results show, the dynamic power of all scheme with sleep transistors increased slightly compared to the baseline, from 1.5% to 3.5%. 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%. The rise time, fall time and propagation delay of the circuits are shown in Figure 8(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, the result show that in both zig-zag and zig-zag share circuits the wordline driver fall time is not affected compared to the baseline circuit. Also as expected, zz-hs-2W has the least impact on rise time and propagation delay.

The area of all schemes is reported in Figure 8(c). The increase in the area varies significantly from 25% for zz-hs-1W circuit to 115% for the redundant scheme.

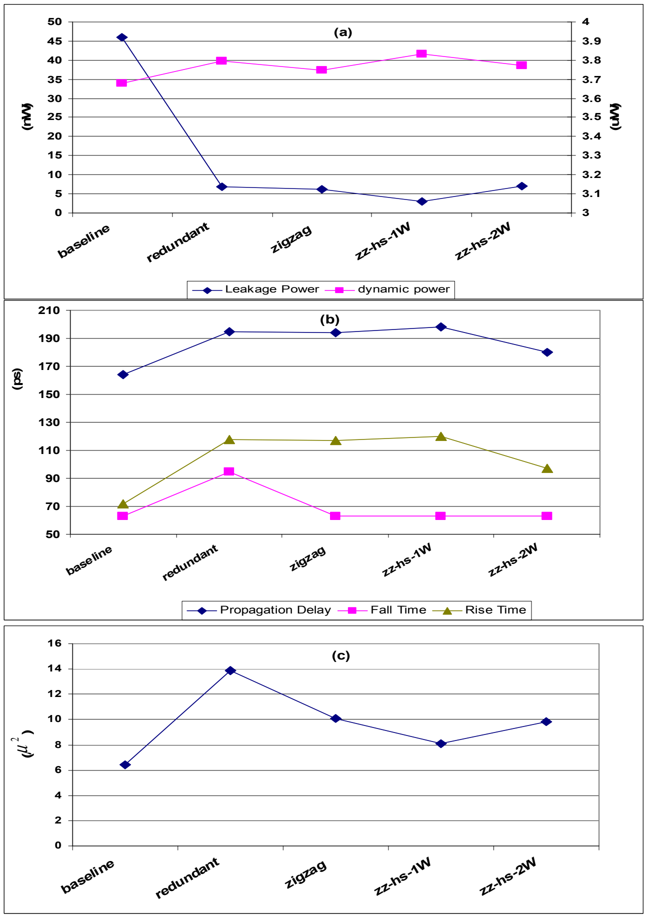

Figure 8. (a) Leakage and dynamic power (b) propagation delay, rise time and fall time (c) area for various leakage control circuits

Figure 9 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 results show, the more wordline rows share sleep transistors the higher reduction in leakage is achieved and less impact is incurred on the area. This in fact validates the analysis in section 2. Overall, the leakage power is reduced significantly, form a 10X to a 100X reduction when 1 to 10 wordline shares the same sleep transistors. This is 2~10X more leakage reduction, compare 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 Figure 9(b). As shown zz-hvs scheme has the least impact on area, 4~25% depends on how many wordline row are shared.

Finally, Table 1 shows a trade-off between the leakage savings and impact on the wordline driver propagation delay for different width of sleep transistor when zz-hvs is shared by 10 rows of wordline drivers. As expected, by increasing the width of sleep transistor we can minimize the impact of zz-hvs on wordline propagation delay but with a reduction in leakage savings.

Note that as explained, the impact of zz-hvs circuits on wordline driver rise time and propagation delay is similar to the zz-hs scheme; Results for zz-hvs-3W (3X) show an optimal trade-off; almost 40X (98%) reduction in leakage power while increasing the wordline driver propagation delay by only 5%.

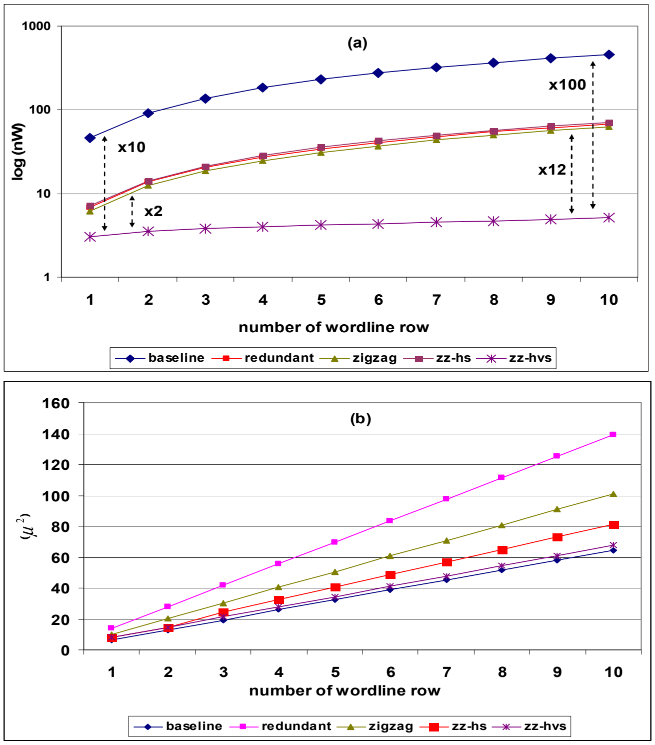

Figure 9. (a) Leakage power (b) area of proposed zig-zag-share circuits compared to zig-zag, redundant and baseline.,

### A. *Wakeup Latency*

As explained in section III, to benefit the most from the leakage savings of stacking sleep transistors we need to keep the bias voltage of NMOS sleep transistor as low as possible (and for PMOS as high as possible). In all simulation results presented in the previous section we assume that the bias voltage of the sleep NMOS transistor during sleep mode is kept at the ground voltage level (PMOS at source voltage). The drawback of such biasing is the impact on the wakeup latency of wordline drivers which occurs when the circuit is transitioning from sleep mode to active mode and requires the voltage of virtual ground to reach to the ground voltage. Our Spice simulation results indicate that the wakeup latency associated with the zz-hvs-3W circuit (when shared with 10 rows of wordline driver) is almost 1.3ns. This translate to almost 4 cycles in our architecture assuming the operating clock frequency of 3.3 GHz. Increasing the depth of wordline driver and reducing the size of sleep transistors increases the wake up latency further. In addition, driving the wake up signal to the sleep transistors increases the overall wakeup latency further. For large memory, such as 2MB L2 cache the overall wake up latency can be as high 6 to 10 cycles. Due to space limitation we do not present the wake up delay variation with circuit parameters discussed.

Furthermore, the zz-hvs increases the propagation delay of the peripheral circuit by 5%, when applied to wordline drivers, input/output drivers, etc. Such an increase can translate to 5% reduction in maximum operating clock frequency of the memory in a single pipeline memory. Unlike single pipelined memories, deep pipelined memories such as L1 and L2 cache can hide this negligible increase in peripheral circuit latency.

Table 1. Leakage power vs. propagation delay

| | baseline | W(1X) | 2W(2X) | 3W(3X) | 4W(4X) |
|---|---|---|---|---|---|
| **Leakage power (nW)** | 460 | 5.11 | 9.13 | 12.63 | 15.7 |
| **Propagation delay (ps)** | 164 | 198 | 180 | 174 | 169 |

## VI. SLEEP-SHARE: ZZ-HVS CIRCUITS PLUS ARCHITECTURAL CONTROL

Some of the highest leakage units in the processor are the L1 (both data and instruction) and L2 caches. Thus it makes a lot of sense to apply the zz-hvs circuit technique in these units. This section briefly describes an integrated architectural approach called Sleep-Share, to control the zz-hvs sleep transistors in L1 and L2 caches.

As explained above, there is a latency associated with waking up the cache peripheral circuitry. The key issue is thus how to avoid loss of processor performance due to cache sleep mode transitions. One approach is to turn off the peripheral circuitry once the processor is idle. Such idle periods can occur frequently during the run-time of a program.

As shown in recent work, when an L2 cache miss occurs the processor executes a number of miss-independent instructions and then ends up stalling [6,10]. The processor stays idle until the L2 cache miss is serviced. This may take hundreds of cycle (300 cycles for our processor architecture). It should be noted that after an L2 cache miss the 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 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 low-power mode. Sleep-share detects the stall period as following:

The instruction queue and functional units of the processor are monitored after an L2 miss. If the instruction queue has not issued any instructions and functional units have not executed any instructions for K consecutive cycles (K=10) the sleep signal is asserted to the cache peripheral circuits. Since there is a 6 to 10 cycle latency to wake up the L1 and L2 cache peripherals, the sleep signal is de-asserted 10 cycles before the miss service is completed.

It should be noted that the assumption here is that the memory access latency is deterministic. Under this assumption the proposed technique does not impact processor performance.

Table 2. 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 |

The technique is evaluated by simulating a processor described in Table 2. 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.

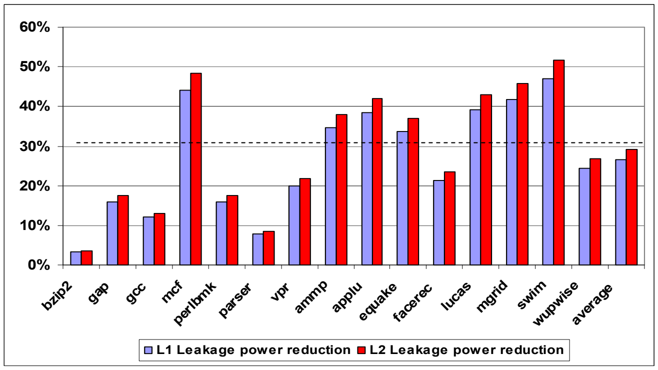

Figure 10. L1 and L2 leakage power reduction

Figure 10 shows the leakage power reduction for L1 and L2 caches. Peripheral circuit leakage under SleepShare is estimated using CACTI5. The peripheral circuit leakage accounts for 90% of total leakage power of the L2 cache (Figure 1). The peripheral circuit leakage is slightly smaller for the L1 cache, 82%. Under SleepShare (e.g. using zz-hvs circuits) the cache leakage power is reduced by at least 90% (see Figure 9 (a)). The leakage in L1 and L2 caches is reduced by up to 47% and 52% respectively (for swim). The average leakage reduction is 30% for the L2 cache and 28% for the L1 cache.

## VII. 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, Sleep-Share, uses architectural control of zig-zag share circuits in L1 and L2 cache peripherals. Our results show that using Sleep-Share the leakage in L1 and L2 caches is reduced, on average, by 30% for the L2 cache and 28% for the L1 cache, respectively across SPEC2K benchmark (and up to 47% and 52% respectively).

## REFERENCES

[1] D. Nicolaescu et al,. Fast Speculative Address Genera-tion and Way Caching for Reducing L1 Data Cache Energy. in. ICCD, 2006.

[2] R. Bai et al,. Total leakage optimization strategies for multi-level caches in Proc. ACM Great Lakes symposium on VLSI, 2005.

[3] T. Skotnicki et al,.The end of CMOS scaling: toward the introduction of new materials and structural changes to improve MOSFET performance. IEEE Circuits and Devices Magazine, Jan.-Feb. 2005,

[4] C. H. Kim et al,. A forward body-biased low-leakage SRAM cache: device, circuit and architecture considerations. IEEE Trans. on VLSI Systems, vol. 13, 2005, pp. 349-357.

[5] S. Borkar et al,. Platform 2015: Intel® Processor and platform evolution for the next decade. Intel Technology Magazine, March 2005.

[6] H. Li, C.-Y. Cher, T. Vijaykumar, and K. Roy. VSV: L2-miss-driven variable supply-voltage scaling for low power. International Symposium on Microarchitecture , December 2003.

[7] J. M. Rabaey et al., Digital integrated circuits: a design perspective, Prentice Hall, Second. Edition, 2003.

[8] Y. Takeyama et al,. A Low Leakage SRAM Macro with Replica Cell Biasing Scheme. IEEE Journal Of Solid- State Circuits, April 2006.

[9] J. Kao, S. Narendra, and A. Chandrakasan, “MTCMOS hierarchical sizing based on mutual exclusive discharge patterns,” DAC 1998.

[10] D. Marculescu. On the use of microarchitecture-driven dynamic voltage scaling. In Workshop on Complexity-Effective Design,2000.

[11] SimpleScalar4 tutorial, http://www.simplescalar.com/tutorial.html.

[12] 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, pp. 684-693, Apr. 2004.

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

[14] A. Agarawal et al., DRG-Cache: A data retention gated-ground cache for low Power, DAC 2002. pp. 473-478.

[15] K. Agarwal, H. Deogun, D. Sylvester, K. Nowka. Power gating with multiple sleep modes. In ISQED 2006.

[16] K. Flautner et al,. Automatic performance setting for dynamic voltage scaling. In Journal of Wireless Networks, pages 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. CODES+ISSS 2004.

[18] B S. Amrutur et al,. Speed and power scaling of SRAMs, IEEE Journal of Solid State Circuits. Feb 2000, vol. 35.

[19] S. Kaxiras et al,. Cache decay: exploiting generational behavior to reduce cache leakage power. IEEE-ISCA, 2001.

[20] B.S. Amrutur, et al., A replica technique for wordline and sense control in low-power SRAM's, IEEE Journal of Solid-State Circuits, vol. 33, No. 8, Aug.2000.

[21] K. Flautner et al,. Drowsy caches: simple techniques for reducing leakage power. IEEE ISCA, 2002.

[22] Cacti5, 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 Journal Of Solid-State Circuits, 2007.

[24] J. C. Park, V. J. Mooney III. Sleepy stack leakage reduction. IEEE Trans. VLSI Syst. 14(11): 1250-1263 (2006).

[25] M. Powell et al,. Gated-Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories. IEEE-ISLPED 2000.

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

[27] S. H. Kim and V. J. Mooney, Sleepy keeper: a new approach to low-leakage power VLSI design. VLSI-SoC 2006.

[28] W. Zhang and J. S. Hu. Compiler-directed instruction cache leakage optimization. In Proc. In MICRO-35, 2002.

[29] Y. Meng, T. Sherwood, and R. Kastner. On the limits of leakage power reduction in caches. In HPCA-11, 2005.

[30] H. Kawaguchi, K. Nose, and T. Sakurai. A super cut-off CMOS (SCCMOS) scheme for 0.5-V supply voltage with picoampere stand-by current. Journal of Solid State Circuit, VOL. 35,NO. 10, 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,” ISSCC 2003.