Design Techniques and Architectures for Low-Leakage SRAMs

Andrea Calimera, *Student Member, IEEE*, Alberto Macii, *Senior Member, IEEE*, Enrico Macii, *Fellow, IEEE*, and Massimo Poncino, *Member, IEEE*

***Abstract*—In high performance Systems-on-Chip, leakage power consumption has become comparable to the dynamic component, and its relevance increases as technology scales. These trends are even more evident for memory devices, for two main reasons. First, memories have historically been designed with performance as the primary figure of merit; therefore, they are intrinsically non power-efficient structures. Second, memories are accessed in small chunks, thus leaving the vast majority of the memory cells unaccessed for a large fraction of the time. In this paper, we present an overview of the techniques proposed both in the academic and in the industrial domain for minimizing leakage power, and in particular, the subthreshold component, in SRAMs. The surveyed solutions range from cell-level techniques to architectural solutions suitable to system-level design.**

***Index Terms*—Caches, leakage, memories, power, power management, standby, sub-threshold.**

I. Introduction

THE emergence of static power consumption in CMOS devices has been one of the first adverse effects of technology scaling. Roughly, when feature size broke the 100 nm barrier, the CMOS transistor ceased to be a virtually ideal switch consuming power only when changing state. While static (or leakage) power affects all kinds of CMOS circuits, it is particularly critical for SRAMs, for two main reasons.

First, leakage power is proportional to the total number of transistors on chip. As reported in the ITRS Roadmap, transistors devoted to memory structures in a typical microprocessor-based system is about 70% today and it is expected to rise to 80% in the near future [1].

Another reason is related to the temperature dependence of some sources of leakage power. SRAMs are highly optimized structures resulting in very high density: Typical SRAM cells have areas in the order of 0.1 μ m^2. Such a high density, coupled with large power consumption, translates into an increase of temperature, which in turn affects leakage current (and, in particular, the subthreshold component) exponentially.

For these reasons, there has been a wide spectrum of research on the reduction of leakage power for SRAMs, at various abstraction levels, from optimized cells structures to alternative memory architectures. The purpose of this survey is to present an exhaustive review of such methods, and to provide a systematic classification and qualitative assessment of the various solutions proposed in the literature.

Manuscript received December 19, 2010; revised June 22, 2011; accepted December 01, 2011. Date of publication February 03, 2012; date of current version August 24, 2012.

The authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (e-mail: enrico.macii@polito.it).

Digital Object Identifier 10.1109/TCSI.2012.2185303

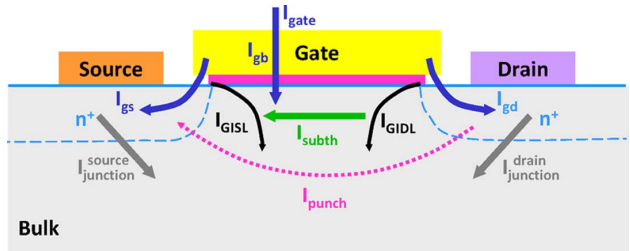


Fig. 1. Leakage Currents in a CMOS Transistor.

The paper is organized as follows. Section II describes the relevant sources of leakage currents in SRAM cells, by characterizing the functional conditions under which the most important sources of leakage, namely, subthreshold currents, manifest themselves. Then, it provides a classified review of the approaches for reducing subthreshold leakage in SRAMs. In particular, Section III addresses methods for bitline leakage minimization, while Section IV surveys the much richer landscape of techniques for cell leakage reduction. Section V highlights some technological perspectives and presents a qualitative comparative analysis of the various classes of solutions that have been considered. Finally, Section VI closes the paper with some universal guidelines to memory designers interested in exploiting the techniques described in this paper.

II. Overview

A. Sources of Leakage Consumption in SRAMs

Leakage power in a CMOS transistor originates from several sources, corresponding to various leakage currents flowing in the device [2]. The list mostly comprises currents that are present when the channel is non-conducting (off state): Subthreshold leakage $I_{\rm subth}$, gate-induced drain leakage $I_{\rm GIDL}$, and depletion punch-through leakage $I_{\rm punch}$. Two other relevant sources of leakage exist independent of the conduction state of the channel: Gate tunneling leakage through bulk, source and drain $I_{\rm gb}, I_{\rm gs}, I_{\rm gd}$ (usually regarded as a single current $I_{\rm gate}$), and p-n junction leakage $I_{\rm junction}$ (from both source and drain). The latter has various sub-sources, and it is dominated by the band-to-band (BTBT) tunneling effect ($I_{\rm BTBT}$). Fig. 1 summarizes the flow of such currents in the transistor's schematic.

Most of the leakage sources can be considered as parasitic effects and are indeed negligible. In [3], the authors show that three are the currents that should be considered for power anal-

1549-8328/$31.00 © 2012 IEEE

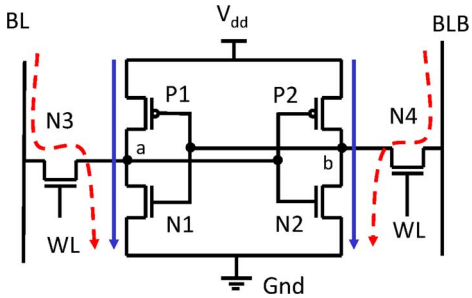


Fig. 2. Reference 6T SRAM Cell.

ysis: Sub-threshold, gate and junction leakage. Their projections on predictive technologies indicate that the three sources will have comparable magnitude already in the 32 nm node. However, technologies that are currently in use do not seem to support this projection. Device materials technology has managed to keep gate leakage under control thanks to the adoption of high-k dielectrics, which helped to move forward in time the point in which gate leakage will become a limiting issue. Band-to-band tunneling currents, conversely, seem to be critical only for strongly reverse-biased devices; although reverse-biasing is a common strategy in low-power design, it is used selectively and with moderate amount of bias for performance reasons. Based on these considerations, our survey addresses only the reduction of the most relevant component of the various leakage currents, that is, subthreshold leakage.

B. Sub-Threshold Leakage in a SRAM Cell

In the following, for ease of reference we will use the structure of a conventional 6T SRAM cell depicted in Fig. 2, and in particular the labeling of the transistors shown in the figure: P1/N1 and P2/N2 denote p and n transistors of the bitcell, and N3/N4 the two nMOS access transistors.

Sub-threshold leakage occurs whenever a transistor is off and $V_{\rm ds}$ is non-zero. Therefore, leakage can occur either inside the bitcell or on the access transistors paths, as shown in Fig. 2: Solid arrows denote *internal* or *cell* leakage, whereas dashed arrows denote *bitline* leakage. Which transistor is actually leaking in a cell depends on (i) the value stored, (ii) the logic level of the wordline, and (iii) the type of operation (i.e., the value of the bitlines). Because of the symmetric structure of the cell, whatever the value stored, the wordline value and the operation, there will be some leaking transistors in the cell.

C. Taxonomy of Approaches

Fig. 3 shows a taxonomy of the leakage reduction techniques for SRAMs. A first level of classification distinguishes between approaches that target bitline leakage versus those that reduce leakage in the cell array. The former give fewer degrees of freedom (leakage is due to the access transistors only), and the possible solutions are based on the design of the bitlines and/or the wordlines. Reduction of leakage in the cell array offers more alternatives: We categorized the methods based on whether they target *active* or *standby* leakage. It is essential to underline here that our distinction between active and standby leakage is somehow different from the one often used in the literature. By *active* leakage we literally mean "leakage in the active state", that is, when the cell is used (read or written); therefore, these techniques aim at reducing *intrinsic* leakage of the bitcell. Conversely, by *standby* leakage we mean "leakage in the standby state"; thus, reduction techniques addressing standby leakage include all solutions in which the memory features some low-leakage state into which some portions (of variable size) of the memory block can be put during inactive phases. In the case of active leakage, solutions are limited to schemes that propose customized design of the memory cell. In the case of standby leakage, methods under this category are assimilated to different implementations of *dynamic power management*. In other words, a given *unit of power management* is identified, and based on some criterion related to the access pattern to the unit, the latter is selectively put into a low-leakage state. The granularity of the unit is quite variable, and it may range from a single cell (cell-based power management) to more or less large portions of the memory array (coarse-grain power management). In the latter category, the vast majority of the approaches refer to caches, which represent the typical embodiment of SRAMs at the architectural level. For this reason, we will term these category *cache dynamic power management so as to characterize it more precisely. For those solutions,* as the granularity of the unit increases, the policy used to drive the transition between the two states and the architectural implications become more important. Given the wide scope of these approaches, Section IV.B presents a further and more specific sub-classification.

It is worth emphasizing that, for approaches based on power management, an important feature of the methods is the *persistence* of the memory values. Depending on how the low-leakage state is implemented, the unit of power management may or may not preserve the stored value. In the sequel, we consider data persistence as a property of each leakage optimization method rather than a dimension of the space of possible solutions.

III. Bitline Leakage Reduction Techniques

Although bitline leakage is less relevant in magnitude compared to cell leakage, it is nevertheless very important because it also affects the reliability of the memory device.

Fig. 4 shows the worst-case scenario for a single-column portion of a SRAM. The accessed cell (WL = "1") stores a "1" while all the others store a "0". When the cell is read, the bitline leakage current (i.e., the current absorbed by the cells whose WL is at "0") becomes the noise against the cell current (i.e., the current injected by the accessed cell to sustain the bitline pre-charge), thus inducing a sensible voltage drop on the bitline. Under this condition, the sense amplifier needs more time to detect the input variation causing more read latency, or, if the leakage approaches the current of the accessed cell, it may provide a faulty output. While in old CMOS technologies the empirical design rule of using an *on-off* current ratio larger than 10 (e.g., by limiting the height of the SRAM column) was conservative enough to guarantee functionality, with the advent of leakage-dominated technologies new dedicated design techniques have become essential.

In the sequel, we review the most effective design solutions to reduce bitline leakage current. They can be broadly classified into approaches that re-design the bitline and techniques

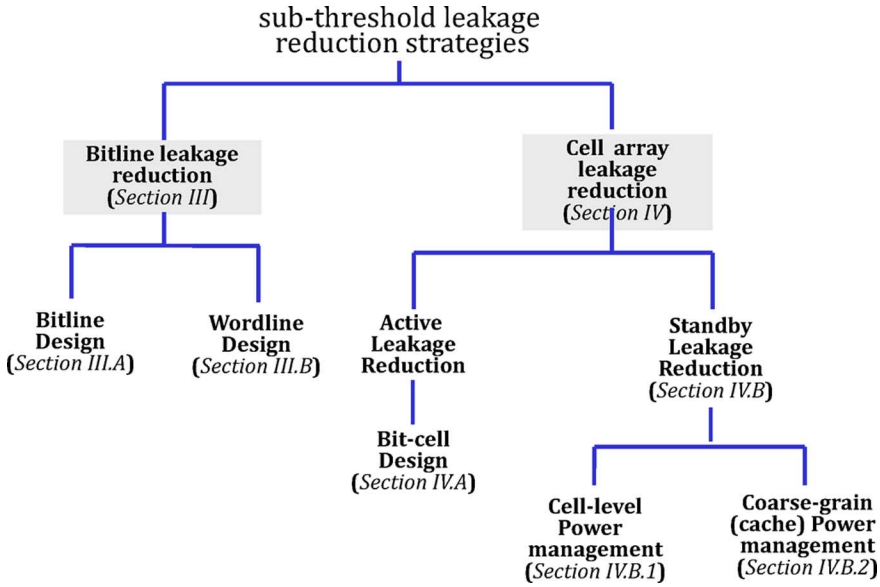


Fig. 3. Classification of SubThreshold Leakage Reduction Techniques for SRAMs.

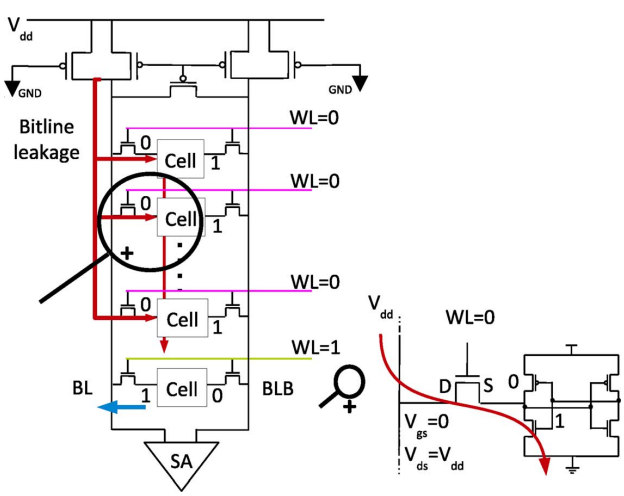


Fig. 4. Worst-Case Bitline Leakage.

that re-design the wordline. Methods in the first category rely on direct *leakage mitigation* or on the assignment of appropriate *pre-charging values*. By leakage mitigation methods we refer to techniques that try to shield the sense amplifier from the leakage current. Two main strategies can be grouped under this definition: Bitline leakage *compensation* (BLC) and bitline voltage *calibration* (X-calibration). The pre-charging value assignment techniques, conversely, include those methods which operate on the pre-charging phase and assign to bitlines special, leakage-aware values. We will present two techniques: the *self-reversed bias* (SRB), and the *floating bitline*. Within the second category (wordline design), we discuss two types of techniques: Those based on *wordline value assignment*, (Negative Wordline) and those which use leakage-aware *modified access structures* (8-Transistor cell). Table I summarizes the bitline leakage reduction techniques that are surveyed next.

TABLE I
SRAM DESIGN TECHNIQUES FOR BITLINE LEAKAGE REDUCTION

	Bitline (BL) Design		Wordline (WL) Design	
	Leakage Mitigation	*Bitline Pre-charge*	*Value Assignment*	*Access Transistors*
Compensation	[4]			
Calibration	[5]			
BL Biasing		[6]		
Floating BL		[7]		
Negative WL			[8] [9]	
8-T Cell				[10]

A. Bitline Design

1) Leakage Compensation: The compensation method was first proposed in [4]. As shown in Fig. 5, it is based on a pre-charge circuit which includes a Bit-Line Compensation (BLC) scheme. The idea is to inject into the bitline an amount of current equal to the leakage current, so that the leakage is compensated and then not sensed by the SA. Although intuitive, this approach requires the implementation of two tricky functions: *i*) Leakage detection and *ii*) current injection. According to the BLC structure described in [4], at the end of the pre-charge cycle the bitline leakage is measured by means of an additional capacitor, whose function is that of storing the amount of bitline leakage. The charge accumulated in the capacitor provides a potential, which is proportional to the detected leakage current. Such a potential is applied to the gate terminal of a dedicated pMOS transistor, which serves as a voltage-to-current converter. The resulting current is then injected into the bitline, thus balancing the bitline leakage and guaranteeing stable memory access.

Although effective, this scheme presents two severe limitations. First, the detection/injection structure, which is based on a dynamic current mirror, is susceptible to threshold voltage variations induced by the fabrication process. Second, the gate voltage of the pMOS transistor that decides the injected amount

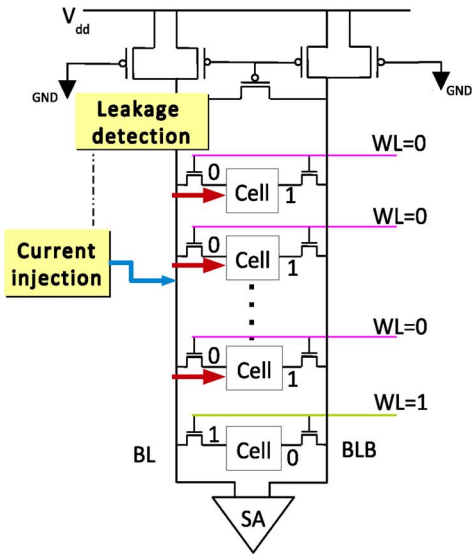


Fig. 5. BLC Scheme.

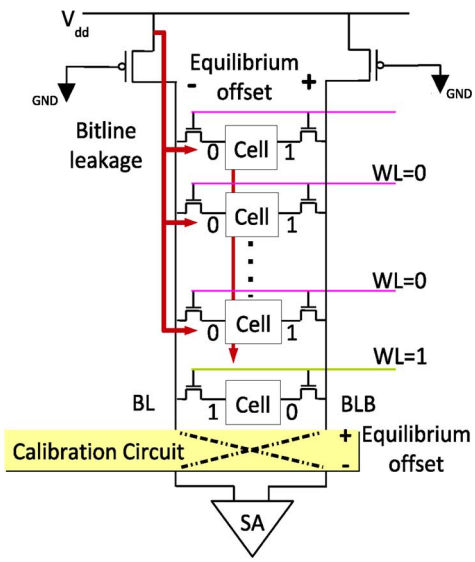


Fig. 6. Calibration Scheme.

of the compensation current is extremely sensitive to the coupling noise generated on the bitline during the pre-charge phase.

2) Sensing Calibration: In [5], the authors propose an orthogonal approach which is based on the concept of bitline voltage calibration, that is, where the voltage difference sensed by the SA is calibrated in order to take into account the effect of the bitline leakage current. The calibration process, performed by a dedicated circuitry placed at the input of the SA (see Fig. 6), consists of a two-step process: *i*) Generation of the equilibrium offset and *ii*) cancellation of the offset.

In the first phase, which starts after the pre-charge cycle, the pull-up transistors are turned-off and the bitlines are left floating. This triggers a leakage-induced discharge transient, at the end of which the voltage on the bitline BL reaches an equilibrium level which defines the equilibrium offset across the bitline pair. This offset reflects the amount of bitline leakage current and it represents the actual voltage noise sensed by the SA. Bringing the same voltage offset on the complement bitline

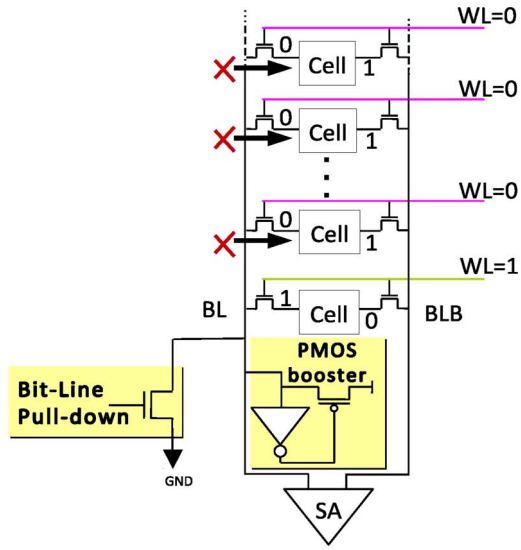


Fig. 7. SRB Local Bitline Scheme.

BLB would make the reading operation insensitive to leakage. That is the goal of the second phase, during which the generated offset is flipped over the BLB line thanks to a crossing structure. The reversed offset is then stored in a coupling capacitor and made available at the input of the SA. The offset voltage is now present on both the SA's inputs; thus the differential sensing results leakage-free.

Compared to the BLC scheme, the calibration technique can handle a higher bitline leakage current at the cost of an extra area overhead and a bitline loading capacitance, resulting in higher power consumption and longer access times.

3) Self-Reverse Bias Bitline: The idea behind this technique is to invert the pre-charging value of the bitline so as to force a low-leakage state on the unaccessed cells [6], as shown in Fig. 7.

Unlike conventional SRAM, characterized by a dynamic pull-up, the bitline is now *pre-discharged* to GND through a nMOS pull-down transistor. This arrangement imposes a different role for the bit-cell: Oppositely to the normal operations, here the bitcell drives the pull-up of the bitline when a logic "1" is stored and sustains the bitline *pre-discharge* in the case of a logic "0". The polarity of the access transistors is thus inverted (i.e., source terminal connected to the bitline). However, since n-transistors exhibit a self shut-off when used as pull-up devices, during the evaluation phase the charging process stops when the bitline saturates at $V_{\rm DD} - V_t$, where V_t is the threshold voltage of the nMOS access transistor. Adding a pMOS buster stage to the bitline allows complete full-swing transitions. This helps the SRB memory to maintain constant robustness and read/write stability as conventional SRAMs.

The main advantage of the SRB structure is that, after the pre-discharging phase, the access transistors' leakage causes the bitline to charge up towards $V_{\rm DD}$. This induces a negative $V_{\rm gs}$ reverse-bias under-drive voltage on the access transistors that, in turn, lowers the bitline leakage. Fig. 8 compares the access transistor operation of a conventional SRAM to the SRB scheme. Moreover, the body effect on the access transistors increases due to the under-drive, which further elevates their V_t. Finally,

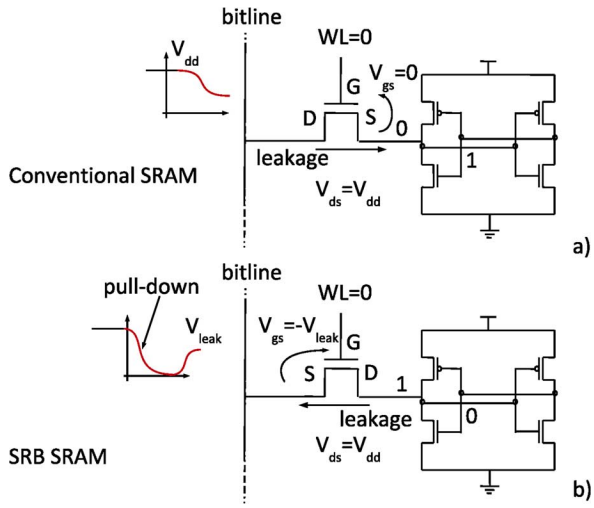


Fig. 8. Conventional versus SRB Access Transistor Operation.

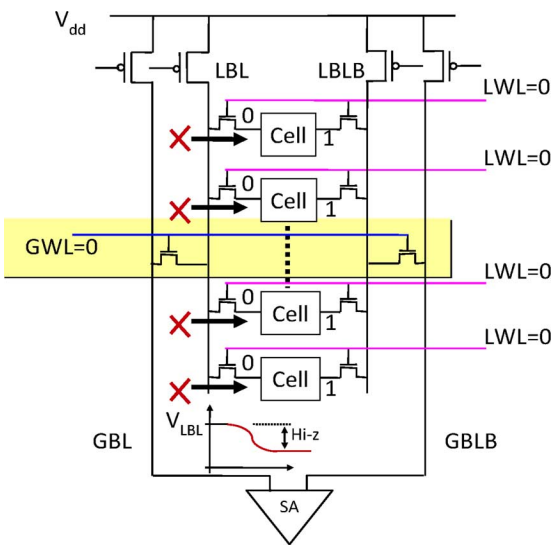


Fig. 9. Floating Bitline Scheme.

the drain-to-source voltage for the SRB access transistors is reduced due to the elevated source voltage, which further lowers the drain-to-source leakage due to mitigated drain-induced barrier lowering (DIBL).

4) Floating Bitline: In [7], the authors propose a design technique that allows bitline floating by turning off the pre-charging transistors. The leakage current from the bit cells automatically biases the bitline to a midrail voltage ($<V_{\rm dd}$) that reduces the bitline leakage current. This current depends on the data pattern stored in the SRAM column. As a result, the amount of leakage current may change at each writing operation. This also impacts the effectiveness of the scheme: If all the cells store a "0", the leakage currents will fully discharge the bitline BL (Fig. 9), while the BLB will be held high. If all the cells store a one, the BL will be held high and the BLB is discharged.

Typically, a mix of ones and zeros biases the bitline to a midrail voltage. Since bitline voltage floats to an undefined level, the SA must be disconnected from the local bitline to avoid other sources of power consumption. In Fig. 9, this is done through additional switches that isolate the global bitline

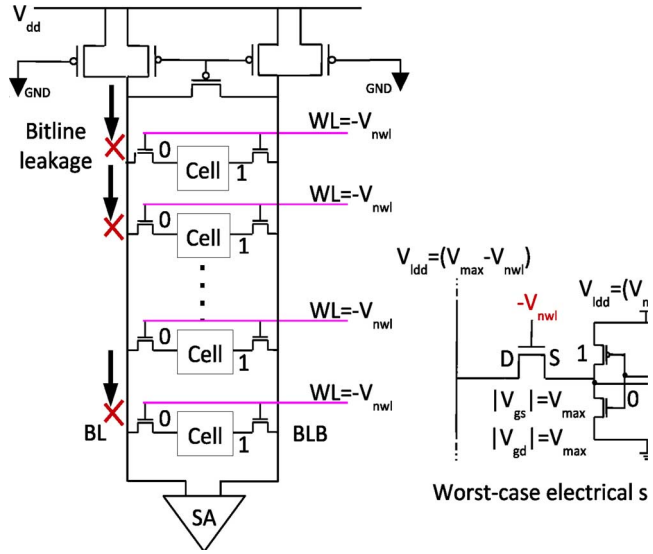


Fig. 10. Wordline Under-Drive.

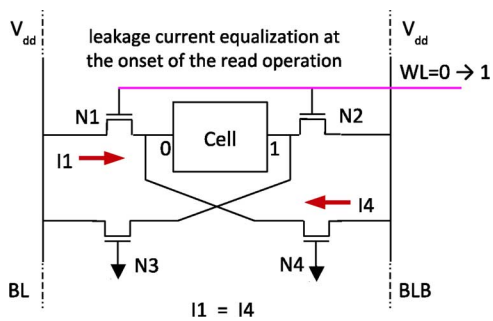


Fig. 11. Leakage Equalization in a 8T Cell.

(at which the SA is connected) from the local bitline. This technique has a low transition energy overhead, because the pre-charge transistor switches as many times as in a conventional SRAM. However, since the bitlines are floating, they are not immediately available for a read operation and an extra pre-charge is required when a new column is accessed.

5) Negative Wordline: It has been shown in [8] that applying negative voltage to inactive wordlines successfully cuts off the bitline leakage. A wordline under-drive, in fact, imposes a negative gate-to-source voltage on the access transistors, which in turn show a reduced subthreshold conductance. This method, however, has never been used alone on real SRAMs, because it suffers from degradation of device reliability since the oxide of the pass-gate is over-stressed.

To solve these issues, the authors of [9] propose the use of statically lower voltages for both storage cells and bitlines. As illustrated in Fig. 10, the wordline drivers are supplied with a global supply voltage $V_{\rm dd} = V_{\rm max}$ and a negative $V_{\rm ss} = -V_{\rm nwl}$ (hundreds of mV). The access transistors of the selected cell are then driven by a gate voltage equal to $V_{\rm max}$, while all the other unselected transistors are under-driven at $-V_{\rm nwl}$. For the storage cells and the bitlines, instead, a supply voltage lower than the global $V_{\rm dd}$ is used, i.e., $V_{\rm ldd} = V_{\rm max} - V_{\rm nwl}$. This ensures that gate-to-source $V_{\rm gs}$ and gate-to-drain $V_{\rm gd}$ voltages of any transistor do not exceed the voltage limit of $|V_{\rm max}|$, thus preserving

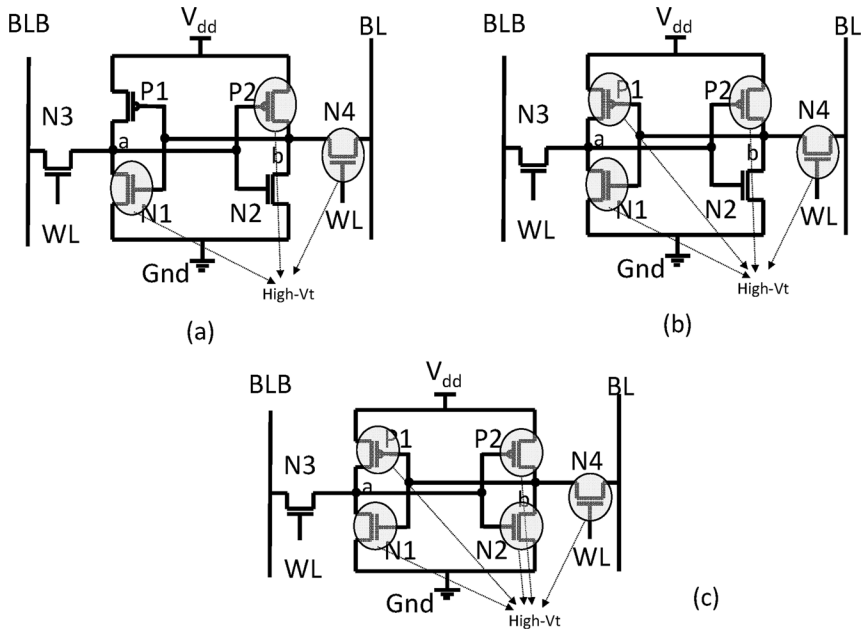


Fig. 12. Basic Asymmetric SRAM Cell and its Improvements.

the stability of the SRAM while minimizing the leakage power significantly. Needless to say, there is a dynamic power overhead for generating and differentiating the supply voltages in the SRAM layout.

6) 8T Cell: In [10], the authors propose the use of a leakage-equalized 8-transistor storage cell. As shown in Fig. 11, the new cell uses a modified access structure made of two additional access transistors, N3 and N4. The two transistors, which are permanently turned off ($V_{\rm gs} = 0$), are sized so as to match the two standard access transistors N1 and N2, and serve as leakage compensation devices. Thanks to this symmetric structure, the 8T memory cell injects identical leakage currents into the two bitline rails BL and BLB. In fact, after the pre-charge cycle, when both bitlines are charged up to $V_{\rm dd}$, I_4 equals I_1, thus eliminating the differential offset voltage of the contending bitlines.

Clearly, this structure assures a fully balanced bitline voltage offset only at the beginning of the read operation, i.e., immediately after the pre-charge cycle. As soon as the differential voltage on the bitline pair increases (due to the evaluation phase), the leakage equalization effect degrades. However, the SA operates in a small sensing windows at the beginning of the evaluation phase, when the leakage is fully balanced. Although no special timing and/or additional control structure in needed, the structure has about 40% area overhead, if compared to a conventional 6-transistor cell.

IV. Cell Array Leakage Reduction Techniques

A. Active Leakage Reduction

Traditional SRAM cells use the six-transistor (6T) symmetric configuration of Fig. 2. The sizing of the transistors is usually driven by performance constraints, with limited concern about the static and/or dynamic power consumption, and they typically have the same threshold voltage. By playing with threshold voltages, it is then possible to obtain a low-leakage cell with reduced impact on read/write performance. A careful selection of which transistors can be made low-leakage is mandatory: In fact, a straightforward replacement of all the transistors with high-Vt ones will unacceptably degrade performance. A possible solution for having low-leakage SRAM cells that guarantee high performance and stability is to adopt asymmetric cells. Asymmetry can be used in two ways. The first option consists in keeping the traditional 6T cell and changing the threshold voltages or the doping profiles of selected transistors. The second strategy is based on the idea of adding more transistors to the original 6T cell, giving more flexibility and design choices for reducing leakage.

6T Asymmetric Cells: Asymmetric cells are based on the following principle: Select a "preferential" state ("0" or "1") and replace with high-Vt transistors only those transistors necessary to reduce leakage when the cell is in that state. These cells show asymmetric leakage currents and access behavior. In [11], the authors propose several architectures of asymmetric cells. They rely on "0" as preferential state, which is the typical situation of most memory arrays, especially when storing data. Therefore, their schemes sensibly reduce the leakage in the "0" state, and possibly do not increase the one in the "1" state. The first architecture, named Basic Asymmetric (BA), is depicted in Fig. 12(a). Transistors N1, P2 and N4 are replaced with three high-V_t ones (indicated in the figure by the shadowed circles). This cell exhibits the same leakage power as traditional cells when a "1" is stored, but it reduces leakage by 70X when a "0" is stored. Unfortunately, since transistors N1 and N4 have now a high threshold voltage, the read access time is degraded (i.e., the bitline discharge is longer w.r.t. the regular cell).

pMOS transistors have no effect on the read access time, since the bitlines are pulled down by the two nMOS transistors on the side of the cell storing the "0". For this reason, an asymmetric cell with reduced leakage is obtained from the BA cell by setting

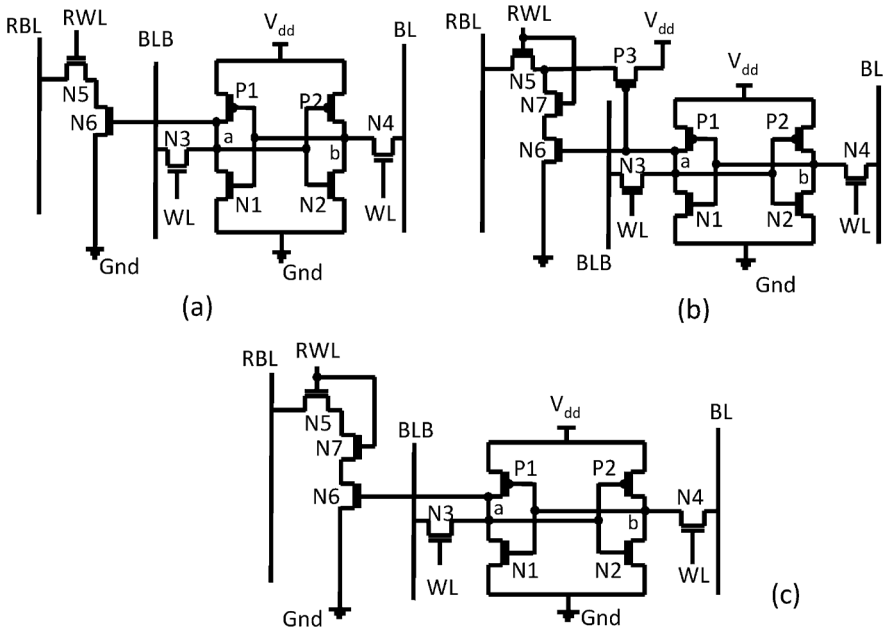


Fig. 13. 8T, 10T and 9T SRAM Cells.

P1 to high-V_t. This cell (Fig. 12(b)) can reduce leakage also when a "1" is stored (1.6X better than the traditional and the BA cells), while it maintains a leakage reduction of 70X when a "0" is stored.

In asymmetric cells, the discharge times for BLB and BL are different. Hence, a particular sense amplifier (SA) that matches the read time on the slow side of the cell to the fast side is needed. If such an SA is available, also N2 can be set to high-V_t. This leads to the cell of Fig. 12(c). This cell further reduces leakage in the high leakage state (i.e., 7X in the "1" state).

More-Than-6T Asymmetric Cells: Conventional 6T SRAM cells may encounter some stability problems, and in particular low read Static Noise Margin (SNM) at very small feature sizes. To address this problem, the read and write operations can be separated by adding read access structures to the original 6T cell, thus increasing the transistor count to 8. As the read current does not significantly affect the cell value, then the read stability of the 8T cell is dramatically increased compared with the original 6T SRAM cell [13].

The problem for the 8T cell (Fig. 13(a)) is that the read bitline leakage is significant: When the column is not accessed, the leakage current flowing through the extra transistor N5 may cause a severe voltage drop at the read bitline (RBL), thus errors may appear at the output. Since it may not be possible to design a high-density SRAM using 8T cells, this conclusion leads to an investigation of other cells, such as 10T and 9T structures ([14], [15]). In [14], in order to prevent the leakage current flowing through N5, a pMOS transistor P3 is added to the read access circuit (See Fig. 13(b)). Transistors N5, N6, N7 and P3 implement a buffer used for reading that eliminates the problem of read SNM by buffering the stored data during a read access. Here the problem is that when the bitcell stores a "1", the pMOS P3 is turned on and the power consumption in a standby mode is large. When the bitcell stores a "0", the subthreshold leakage through the pMOS P3 is also high due to the voltage difference between its drain and source. Therefore, in [15] the pMOS is removed for low-power operation, thus reducing the transistor count of this scheme to 9, as sketched in Fig. 13(c).

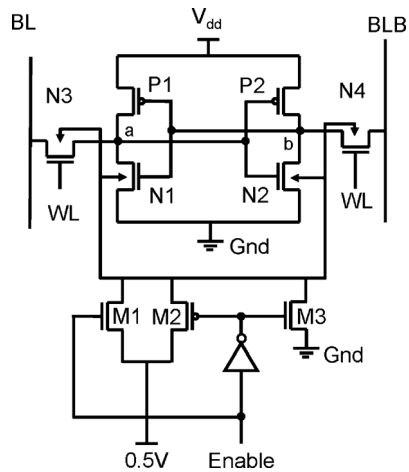


Fig. 14. FBSRAM Cell with Body Bias Driver.

Another cell architecture is proposed in [16], where dynamic *forward* body bias (FBB) is used in active mode. Here, super high-V_t transistors [16] are used together with FBB to dynamically reduce the active leakage in SRAM cells. Fig. 14 shows a forward body bias SRAM cell (FBSRAM).

Transistors M1–M3 are the body bias drivers. When the cell is accessed the *Enable* signal is raised, M1 and M2 are turned on and the body voltage is switched to 0.5 V. The drive current is then increased and a fast read/write operation is achieved. Notice that the cell also saves leakage when the cell is not accessed: with the *Enable* signal low and a zero body bias applied, through M3, to the super high-V_t devices.

B. Standby Leakage Reduction

The richest class of approaches consists of the implementation of some form of *dynamic power management* (DPM) scheme to different portions of the memory array. This paradigm relies on the observation that the memory is accessed through small "units" (e.g., a word); therefore, given that only one unit at a time can be accessed, all the memory array but the accessed unit sits idle and dissipates static power; it is therefore essential to put the unused portion into a low-power state whenever possible. This scenario corresponds to a classical instance of the DPM problem, in which the state of a resource having multiple *power states* is dynamically adapted to the workload in order to minimize the average power consumption under given performance constraints [17].

In the context of memories, the resource is the portion of the memory that can be put into a low power state (called hereafter the *unit of power management*); the latter is not necessarily identical to the *unit of access* to the memory. The power states represent the different power modes of the unit of power management. The workload is the pattern of accesses to the memory. An additional dimension is hidden behind the above problem statement, that is, the criterion used to decide for making transitions between power states (*the power management policy*).

In our analysis, we distinguish between the case in which power management is applied to a single cell and the case of larger units, typically applied to caches. In the former case, power management is achieved thanks to the circuit-level implementation of the low-leakage, standby state; therefore, it can be treated as a customized implementation of a bitcell with a normal (active) and low-leakage (standby) mode. Conversely, for coarse-grain approaches, other issues are relevant, such as the granularity and the power management policy. The implementation of the low-leakage state is just the extension of the cell-level solution to a larger scale. Cell-level power management schemes can then be viewed as enablers for the coarse-grain solutions.

1) Cell-Level Power Management: In a 6T SRAM cell, the low-leakage state can be implemented in three ways:

- *Power-gating*: Leakage power is almost entirely nullified by introducing an extra pMOS header transistor on the supply path and/or a nMOS footer transistor in the ground path (*sleep transistors*). These transistors cut the supply-to-ground path when the cell is unused (standby).
- *Body-biasing*: The body terminals of the four bitcell transistors are connected to a voltage source in order to control (i.e., increase) their threshold voltage. To this purpose, *reverse* body bias (body voltage of pMOS larger than $V_{\rm dd}$, body voltage of nMOS smaller than $V_{\rm gnd}$) can be applied in the standby state.
- *Voltage scaling*: Leakage can also be reduced by using a lower supply voltage, since static (leakage) power scales linearly with supply voltage. Therefore, in the standby state the supply voltage pin of the bitcell transistors is connected to a supply voltage smaller than $V_{\rm dd}$.

It is worth emphasizing that these strategies have different effects on the persistence of memory values: Power gating is a *non-state-preserving* implementation of a low-power state, whereas voltage and body bias voltage scaling are *state-preserving* ones. Table II summarizes the standby cell leakage reduction techniques that we illustrate in the sequel.

TABLE II
SRAM CELL TECHNIQUES

Power Gating	[18] [19] [20]
Body-Biasing	[21] [22]
Voltage Scaling	[23] [20]

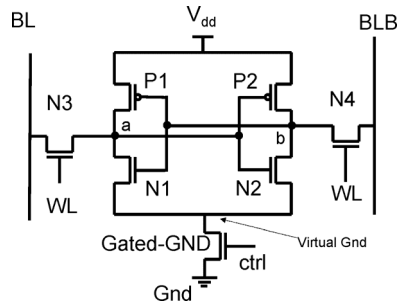


Fig. 15. Gated-GND SRAM Cell.

In [18], the authors propose a cell-level implementation of power gating in which a sleep transistor is added to the supply (gated-$V_{\rm dd}$) or the ground path (gated-GND) of a SRAM cell (Fig. 15). Gated-$V_{\rm dd}$ (gated-GND) sensibly reduces leakage thanks to the stacking effect of self reverse-biasing series-connected transistors. The extra transistor produces the stacking effect in conjunction with the SRAM cell transistors when the gated-$V_{\rm dd}$ transistor is turned off.

Gated-$V_{\rm dd}$ (gated-GND) can be coupled with a dual-threshold voltage (dual-V_t) process technology to achieve even larger reductions in leakage. SRAM cells use low-V_t transistors to keep high performance and the sleep transistors use high-V_t to achieve additional leakage reduction.

The main limitation of this basic cell-level power gating is the persistence of the stored data. Moreover, the gated-GND transistor increases the resistance of the pull-down path; hence, when it is turned off in the standby mode, the node storing "0" and the virtual ground are floating and set to a positive voltage by the weak leakage currents. This makes the cell more sensitive to noise. To solve this problem, a diode can be put in parallel with the gated-GND transistor.

For addressing the problem related to the data retention, possible solutions are presented in [19] and [23]. In [19], the authors use the concept of gated-GND described above and propose to modify the basic power-gated structure as shown in Fig. 16. The signal *ctrl* that drives the gated-GND transistor is externally generated (e.g., by the row decoder) and connected to the WL.

When the gated-GND transistor is on, the cell behaves exactly as a conventional cell in terms of storing data. When the gated-GND is turned off, the leakage path from the cell node that is at "1" to ground is cut-off. In this way, however, the chance of firmly hooking the cell node at "0" to ground is lost. This makes it easier for a noise source to write a "1" into that node. The absence of a conducting path from the node storing "0" to ground results in the voltage at node a to be decided by the leakage current of the transistors connected to this node.

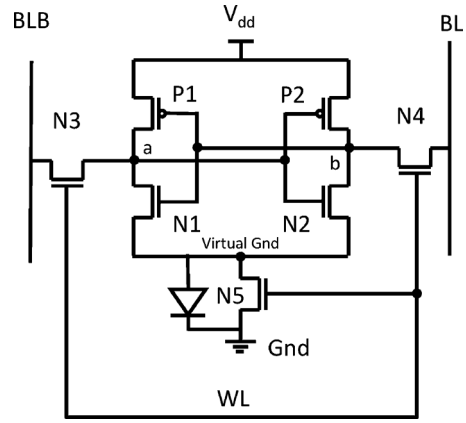


Fig. 16. DRG SRAM Cell.

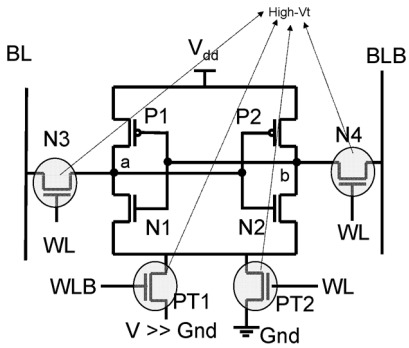


Fig. 17. NC-SRAM Cell.

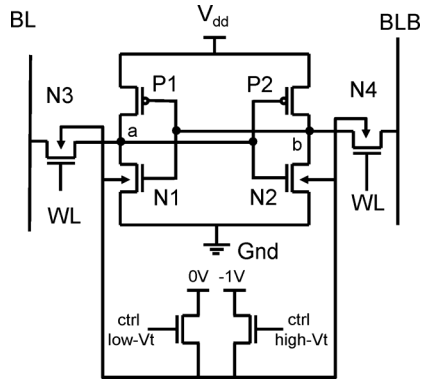


Fig. 18. DTSRAM Cell.

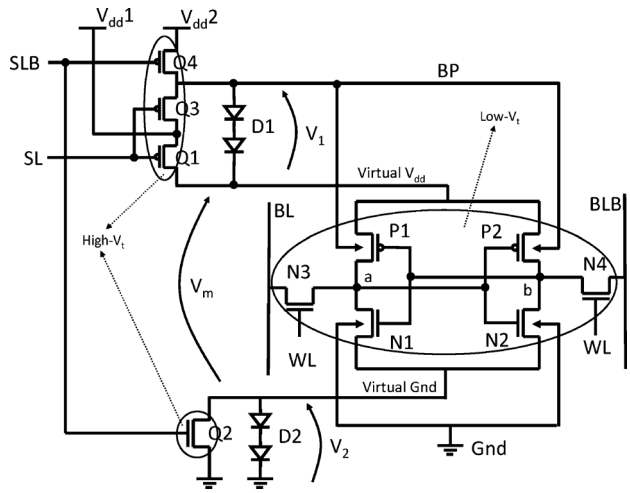


Fig. 19. ABC-MT-CMOS SRAM Cell.

Three are the leakage paths associated with a: Leakage currents through pull-up pFET P1, pass transistor N4, and pull-down nFETs N1 and N5. The first two currents try to charge a, whereas the latter tries to discharge it. These three forces charge a to some intermediate voltage (≈ 0.4 V in the analysis of [19]) at which all the three currents are equal. This voltage is small enough to turn on the pull-up transistor P2, so the net result is that node b is strapped to $V_{\rm dd}$ and it always remains at "1". Upon accessing the cell (i.e., turning on N5), data are restored by pulling a to "0" through N2 and N5. In [23], the authors present a new solution, called NC-SRAM, which exploits Dynamic Voltage Scaling (DVS) to reduce the leakage power of the memory cell and which also retains the data stored during the inactive state. The key idea is to use two pass transistors providing a positive ground voltage when the cell is in standby and connect the cross-coupled inverters to the usual 0 V supply voltage during normal operation to work as a conventional 6T-cell (Fig. 17). Therefore, the operating voltage is scaled by raising the ground voltage (and not lowering the $V_{\rm dd}$ as usual).

In order to reduce the bitline leakage (N3 and N4) and the leakage through PT1 and PT2, transistors N3, N4, PT1, and PT2 are high-V_t devices. In this scheme, none of the internal nodes is left floating in standby, and data retention and stability are achieved without additional complexity or circuitry.

The cell proposed in [21] can be seen as a straight-forward implementation of body bias control in a cell. This scheme assigns low-V_t to the cells that are actively used (body voltage at 0 V), while high-V_t is assigned to cells that are inactive (body voltage at -1 V). Fig. 18 reports the schematic of a dynamic V_t SRAM (DTSRAM) and shows that only the V_t of the nMOS is controlled for ease of design.

A more complicated arrangement for threshold voltage control is the Auto-Backgate-Controlled Multi-Threshold CMOS (ABC-MT-CMOS), proposed in [20]. The schematic of the ABC-MT-CMOS SRAM cell is shown in Fig. 19. In this architecture two different supply voltages $V_{\rm dd1}$ (i.e., 1.0 V) and $V_{\rm dd2}$ (i.e., 3.3 V) are present. Transistors Q1, Q2, Q3 and Q4 are high-V_t transistors. Q1 and Q2 are used as switches to cut off the leakage current. Diodes D1 and D2 (consisting of two diodes each) and the higher voltage $V_{\rm dd2}$ are used to minimize the leakage current and retain the data stored in the SRAM cell. Transistors P1, P2, N1, N2, N3 and N4 are low-V_t.

In active mode, the standby signal SL is low and SLB is high, and Q1, Q2 and Q3 are turned on, Q4 is turned off. The substrate bias BP and the *Virtual Vdd* (through Q1) become $V_{\rm dd1}$. The *Virtual Gnd* is forced to Gnd through Q2. In standby mode, (SL high, SLB low), Q1, Q2 and Q3 are turned off, and Q4 is turned on. The substrate bias BP and the *Virtual Vdd* (through D1) become $V_{\rm dd2}$. The *Virtual Gnd* is connected to Gnd through D2.

The static leakage current, which flows from $V_{\rm dd2}$ to Gnd through D1 and D2, determines the voltages V_1, V_2 and V_m.

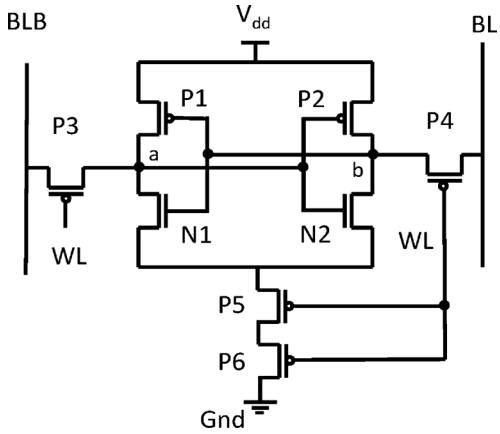


Fig. 20. P4-SRAM Cell.

Here, V_1 represents the bias between the source and substrate of the p-MOS transistors, V_2 denotes that of the n-MOS transistors, and V_m represents the voltage between *Virtual* $V_{\rm dd}$ and *Virtual Gnd*.

If the forward bias of one diode is assumed to be 0.5 V, the forward voltage of D1 and D2 is 1.0 V. *Virtual* $V_{\rm dd}$ becomes about 2.3 V and *Virtual Gnd* about 1.0 V. The static leakage current decreases significantly with respect to that of the active mode, since the threshold voltage of the internal transistors increases by its back-gate bias effect. In standby mode, *Virtual* $V_{\rm dd}$ and *Virtual Gnd* maintain their voltage levels owing to the weak leakage current, so that the data stored in the memory cell is retained. This method does not require a triple-well structure and complicated circuits, such as charge pumps and balloon circuits.

Another approach that exploits stacking effects and body biasing is proposed in [22]. This technique is based on the cell architecture of Fig. 20. When WL $=$ ``0", the cell behaves as a conventional 6T cell. When the WL $=$ ``1", the two access transistors (that are pMOS in this architecture) are OFF and so are both P5 and P6. In the proposed design, in order to reduce the negative impact of the threshold voltage on the speed of the cell and to reduce the active power consumption, a forward, full-supply (i.e., at $V_{\rm dd}$) body-biasing is used in the pMOS transistors. The use of the pMOS transistors increases the dynamic power of the cell, during the read/write operations. On the other hand, the leakage power consumption is reduced by 50% when a "1" is stored and by 46% when a "0" is stored, at a small area penalty of two pMOS stacked transistors.

2) Cache Power Management: Since we focus on hardware solutions, we assume that the memory access pattern cannot be changed. Conversely, the other three parameters (unit of power management, choice of power states, and policy) represent possible implementation choices that characterize the various approaches. In practice, however, the policy dimension is seldom exploited by cache DPM strategies proposed in the literature. This is mostly due to hardware complexity of more sophisticated policies, which are feasible only when implemented in software [17]. The vast majority of the schemes implement the simplest possible policy, i.e., the *time-out* policy: After the unit of power management has been idle for a time longer than some time-out value, it is moved to the off state. Therefore, existing approaches differ for the other two parameters: Granularity of the power management unit and implementation of the low-power state. Table III summarizes the various options for the two dimensions. Table entries report the reference to the solutions reviewed in the sequel.

TABLE III
CLASSIFICATION OF CACHE MEMORY DPM SOLUTIONS

Low-power Implementation	**Unit of Power Management**			
	Subset of a Line	*Line*	*Set of Lines*	*Region*
Voltage Scaling		[24]	[25], [26]	
Body Bias		[27] [28]	[29]	[16]
Power Gating	[30] [31]	[32] [33]	[18]	[34] [35]

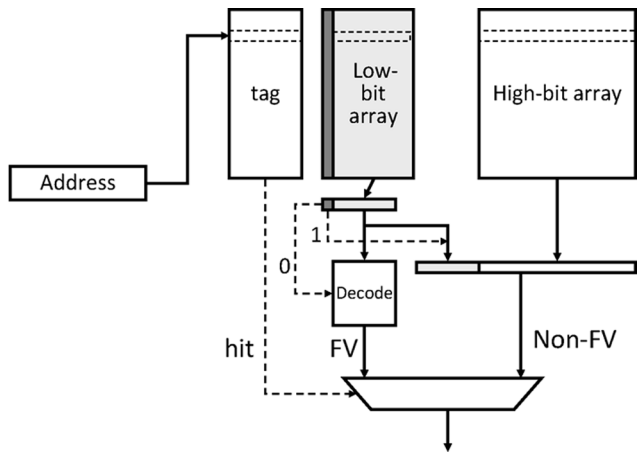


Fig. 21. Frequent Value Cache.

Options for implementing the low-power state include voltage scaling, power gating, and control of the threshold voltage via body-bias, as mentioned in Section IV.B.1. Options for the granularity of the power-managed unit comprise a subset of a memory cache line (e.g., a byte or a word), an entire cache line, or a set of cache lines. All these variants are uni-dimensional and do not include a "vertical" partitioning. Therefore, they are consistent with the semantics of the memory, in which data are stored "by rows". In some particular cases, a bidimensional unit is considered, (last column in Table III): The most typical implementation is the power management of one *way* of a set-associative cache.

a) Line-Subset Granularity: The methods of [30] and [31] rely on the property that also a "value" locality does exist. In [30], it was observed that frequently occurring values often occupy a large portion of a data cache. Therefore, it is possible to choose a small set of these frequent values (FVs) and encode it using a number n of bits smaller than the full data width. The data array is thus partitioned into two arrays, the first containing the n LSBs of the data, the other containing the rest of the bits (Fig. 21).

When accessing a value, the low-bit array is accessed first. Each line has an extra "flag" bit (dark boxes) signaling whether a FV is stored or not. If it is a FV, only the reduced size array can be used, and the remaining bits are turned off (specifically, in a non-state-preserving mode). Clearly, a decoding operation is needed to restore the FV on the full bit-width. Conversely,

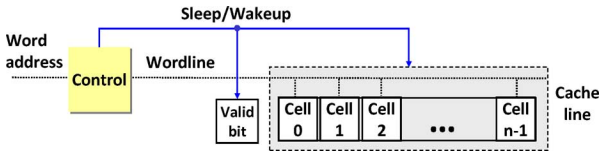


Fig. 22. Generic Line-Based Power Managed Architecture.

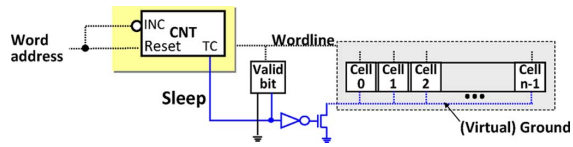


Fig. 23. Cache Decay Architecture.

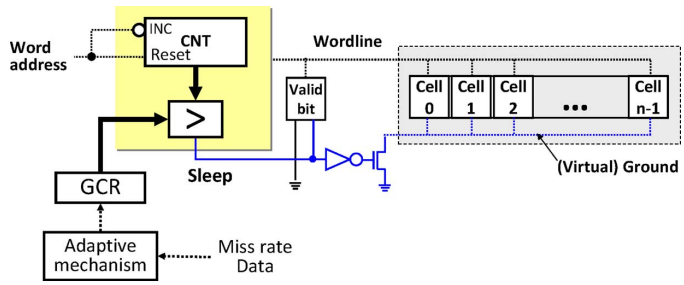


Fig. 24. Adaptive Mode Control (AMC) Cache.

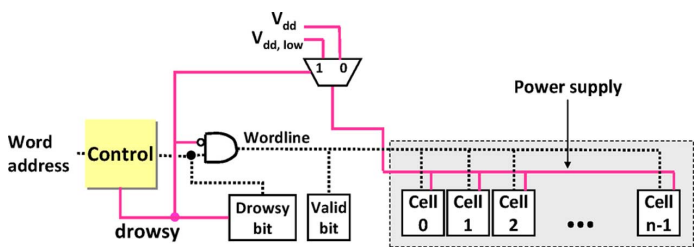


Fig. 25. Drowsy Cache.

if a non-FV is accessed, the remaining bits are then read from the high-bit array and padded with the n LSBs already read. Although apparently this approach defines a "vertical" unit of power management, in practice a fraction of a cache line is frozen. The approach of [31] follows a similar concept, but with a different architectural pattern. First, the unit of power management is always a word (i.e., the unit of access); frequent values are defined by profiling the executed workload; once identified, they are stored in a separate small (typically, no more than 4 values) fully associative buffer. Such values are always read out from the buffer and are always turned off (using power gating) inside the memory array; this implies no overhead in waking-up a unit. Management of FVs inside the cache is done by keeping a set of extra bits per each cache line (1 bit for FV/non-FV, plus other bits corresponding to their address in the FV buffer).

b) Line Granularity: This class of approaches fits into the generic architecture shown in Fig. 22. ewThe *Control* block implements the power management policy. Whatever the policy, when a decision to turn off the cache line is taken (signal *Sleep/Wakeup*), the corresponding mechanism is activated on the cache line. As shown in Table III, all the three low-level mechanisms (voltage scaling, body bias, and power gating) are possible and have been proposed in the literature.

Fig. 22 also shows another important functional issue: The *Sleep/Wakeup* signal must invalidate the cache line, which, on a subsequent access, cannot be read normally. Both in a state-preserving or state-destroying condition, the line cannot be read safely until it is driven into the active state. The *valid* bit, normally present in each cache line, is used to this purpose. The work of [32] uses a non-state preserving policy, and implements a pure time-out-based mechanism, called *decay interval* (the architecture is known as *Decay Cache*). The *Control* block of Fig. 22 consists of a plain binary counter. The decay interval is calculated from the break-even time obtained from the characteristics of the power state machine.

Conceptually, the counter is incremented at each cycle in which the line is not accessed; if no accesses to the cache line occurs, the counter will saturate to its maximum count (i.e., the decay interval has elapsed), and a *Sleep* command is issued to the cache line. Each time the cache line is accessed, however, the counter is reset to its initial value (Fig. 23).

The *Adaptive Mode Control (AMC)* cache proposed in [33] and shown in Fig. 24, overcomes the *static* nature of the Decay Cache (the decay interval is, in fact, pre-calculated and hard-wired into the cache circuitry). The AMC cache monitors in hardware the hypothetical cache miss rate (i.e., as if all lines were always active). This is made possible by keeping the tag portion of a cache line always active, and by adding proper hardware that selectively translates the miss rate into power management commands. As in the Decay Cache, this architecture is non state-preserving.

Similarly to the Decay Cache, a counter is kept for each line, and whenever its count exceeds a threshold, the line is turned off. The main difference is that this threshold value is not fixed, and it is updated according to the cache performance (in terms of monitored miss rate). Block *GCR (Global Control Register)* in Fig. 24 stores this threshold. The decision about the shutdown of a line is taken by comparing (Block ">") the idleness counter with the threshold.

The *Drowsy Cache* [24] was the first work proposing the use of a state-preserving mechanism based on voltage scaling. Conceptually, lines that are turned off are supplied with a voltage that is much lower than the nominal $V_{\rm dd}$, but higher enough to keep the stored value. This $V_{\rm dd,low}$ was shown to be a few mV higher than the threshold voltage of the cell transistors. Fig. 25 shows the abstract structure of a drowsy cache line.

This scheme requires an extra bit per line (the drowsy bit), a mechanism for controlling the voltage of the memory cells, and a wordline gating circuit. When a line has to be turned off, the *drowsy* signal is set. This in turn sets the drowsy bit, blocks access to the wordline, and drives the $V_{\rm dd,low}$ voltage to the cache line. Gating of the wordline is required to prevent accesses when in drowsy mode. Whenever a cache line is accessed, the state of the cache line is given by the drowsy bit. When in normal mode, line contents are read as usual and with no performance penalty. If the line is in drowsy mode, the drowsy bit is cleared and the supply voltage is switched to regular $V_{\rm dd}$, with a one-cycle penalty. The *Control* block in Fig. 25 implements the policy; Paper [24] suggests various policies such as timeout-based ones

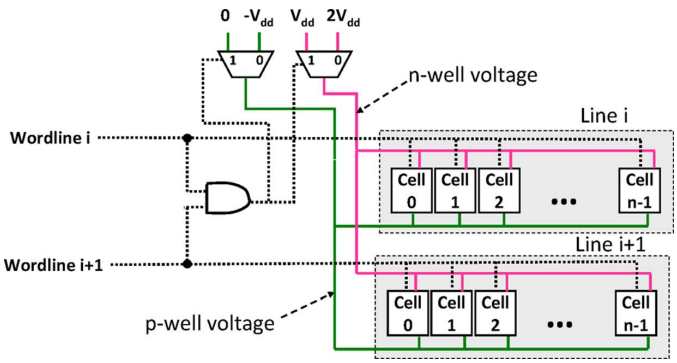


Fig. 26. Dynamic FBB Cache.

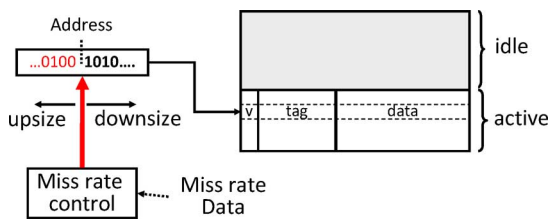


Fig. 27. Dynamically Resizable Cache.

or even simpler ones where a line is put into drowsy mode periodically.

Some solutions use body-bias control for the implementation of the low-power state. The work in [27] does not strictly refer to caches; in the paper, the extension from a single cell to a set of cells is mentioned in terms of an entire memory "row". The internal structure relies on the so-called Auto-Backgate-Controlled MTCMOS described in Section IV.B.1 [20]. In [27], the sleep signal is considered as externally provided, thus there is no indication of a feasible policy.

The solution of [28] differs from that of [27] in the way the body-bias control is achieved. The n- and p-well bias voltage are dynamically changed to $V_{\rm dd}$ and $V_{\rm ss}$ for active memory lines, while the well bias of the inactive ones memory cells are kept at $\approx 2V_{\rm dd}$ and $V_{\rm dd}$ (Fig. 26). In this way, selected memory cells allow fast operation, whereas dormant lines have very little leakage. This scheme requires a triple-well technology for n-well processes to isolate the nMOS body of a line from the others. Two are the peculiar architectural features of this solution. First, the well bias signals are synchronized with the word line signal and *not with a standby signal*; this implies that a non-accessed line is immediately biased properly, so as to reduce leakage. Second, well bias signals are shared between two adjacent lines to simplify the routing.

c) Set of Lines Granularity: This class of solutions is quite heterogeneous according to how the "sets" are defined. We first review the non-state preserving scheme of [18], called Dynamically Resizable Cache. The "set", in this method, is a variable-size group of contiguous cache lines. Fig. 27 shows the concept of the Dynamically Resizable Cache.

The decision about shutting down a set of lines is dictated by the analysis of the miss rate, similarly to the AMC cache. Activation/de-activation of a set of lines is done by *masking bits of the cache address*. If the miss rate is higher than a target bound, we could effectively reduce the working set and thus operate with a smaller cache; this is achieved by masking one bit of the address, or equivalently, disabling one half of the cache, which can then be turned off. If the miss rate keeps on staying beyond the threshold, additional bits can be masked, thus disabling another 1/4, 1/8, etc. of the cache. Whenever the miss rate falls below the threshold, we start re-activating idle blocks by unmasking index bits. This simple scheme is interesting because it provides a variable-size granularity of the power management unit and allows for a fine grain control of the performance/leakage trade-off.

The approaches of [25], [26] represent two variants of the same concept, that is, *partitioning* memories in coarse-grain chunks, yet implemented in two quite different ways. Both works are originally introduced for generic, software-managed scratch-pad memories rather than caches, but the paradigm can be applied with minor variations to caches as well; both methods use a state-preserving mechanism based on voltage scaling for power-management.

In [25] it is assumed that the memory is split into *banks* of uniform size, each of which can be individually power-managed. Their scheme exploits *bank locality*, that is, the property for which successive memory accesses tend to use the same bank. A conventional time-out-based policy is used: A counter is associated to each bank; whenever a bank is accessed, its decay counter is cleared, and decay counters of other non accessed banks are incremented. When a bank counter saturates, a drowsy signal is sent to its bank to move it to the drowsy state.

The partitioning scheme proposed in [26] relies on a totally different architectural concept. It assumes a memory with *non-uniform banks*, whose sizes are computed based on the analysis of the memory access patterns. Using non-uniform sizes instead of uniform ones allows to optimally match the size of the sub-banks to the distribution of memory access. The basic principle is that of fitting the address ranges with the least idleness to the smallest possible bank (that is, the one with the lowest power consumption). In Fig. 28 a simple partitioning case with two banks (B_1 and B_2) is shown. Each bank has its own idleness counter, which, as in previous schemes, is incremented during non-accesses, and reset upon accesses. When the terminal count for a bank is reached, that bank can be put into a low-leakage state. Block C activates the power management "commands" to the voltage selector, which will apply the low voltage to that bank. Moreover, in order to provide the correct address to each bank, the original address must be relocated. This is done by Block *Decode* that, for a given address, calculates (i) to which bank it belongs (1 or 2 in the figure), and (ii) the correct address for that bank.

Relevant is also the so-called Selectively Activated Cache (SAC), proposed in [29]. The paper presents a complex cache organization that allows to dynamically activate/de-activate a group of consecutive cache lines (called a *block*).

This method is one of the few that does not use a time-out-based policy, but rather a history-based *predictive* mechanism, in order to reduce the chance of accessing a block that is in sleep mode. SAC relies on the Auto-Back gate-Controlled MT-CMOS technique [20] as implementation of the sleep mode of the blocks and belongs therefore to the

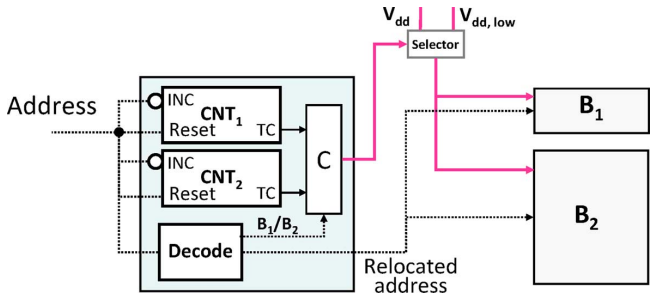


Fig. 28. Multi-Bank Non-Uniform Partitioning.

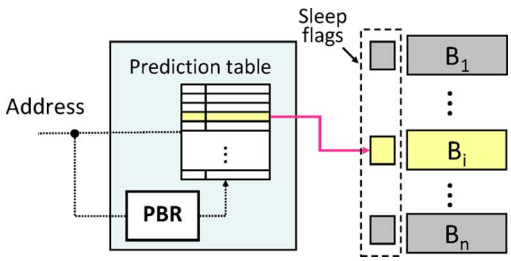


Fig. 29. Selectively-Activated Cache.

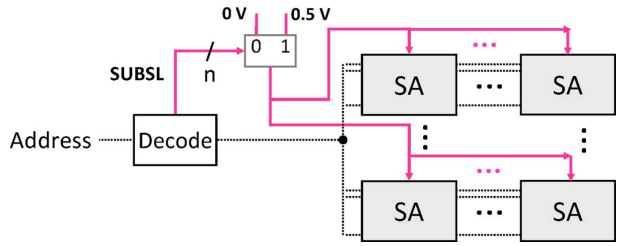


Fig. 30. Dynamic FBB Cache.

class of state-preserving schemes. As shown in Fig. 29, the implementation of this architecture requires the addition of a few extra blocks in the cache control mechanism.

The *Prediction Table* implements the history-based prediction. It contains as many entries as the number of blocks, and each entry i has a list of the last m blocks accessed prior to Block i. Whenever block i is accessed, all the blocks in the list are kept active (using low-V_t), whereas all other blocks are kept inactive (using high-V_t). An extra bit for each block *Sleep flag* indicates the state of each block $(1 = \text{active}, 0 = \text{inactive})$.

The prediction table uses an address register, (previous block address register—PBR), which keeps the previously accessed block address. If the current memory address differs from the value in the PBR (another block is accessed), the sleep flag of the currently accessed block is activated, and the PBR stores the new block value. In the next cycle, blocks in the prediction list of the current block are also activated.

d) Region Granularity: A few approaches have been proposed that use an even larger portion of a cache as the unit of power-management. The distinctive feature of these schemes is that this portion is bidimensional. However, since some sort of access semantics needs to be preserved (i.e., the "meaning" of what is power managed), these approaches typically address way-associative cache memories and the power-managed portion is a *cache way*.

The *Way Decay Cache (WDC)* [34] implements a non-state preserving architecture that is reminiscent of the DRI cache [18]. The shutdown policy is based on the monitoring of miss rate, as follows: A *Way Counter* (one per way) counts way hits, and a global *Miss Counter* counts total misses. At regular intervals, the total miss count is compared to a specified miss bound. If lower, performance is considered as acceptable, and the cache can be "resized" by disabling (through power gating) one way. Conversely, ways are re-activated.

Similarly, the work in [35] uses a dual-V_t approach where threshold selection is done at design time. There are two key points in this solution: First, to allow the ways within a cache to be accessed at different speeds (with different leakage); second, to place infrequently accessed data into the slow ways. The placement is done by using dynamic information regarding data *criticality*, measured in terms of frequency of access. Only critical data will be placed into the fast ways.

The solution proposed in [16] is meant for generic SRAMs; thus, it uses a generic bidimensional array region as unit of power management. Super-high V_t devices are used inside the cell to suppress the leakage, and forward body-bias (FBB) is applied only to the selected SRAM region for fast operation. The design of the specialized cell using super-high V_t is the key enabling technology of the architectural design. The authors demonstrate a 32-by-32 cell region as power-managed unit (called sub-array hereafter). Fig. 30 shows a conceptual block diagram of this scheme. *SA*'s denote the sub-arrays.

The sub-array selection signal (*SUBSL*) goes high when a sub-array is selected. For a given size of the sub-array, this signal can be generated inside the row decoder and be activated even before the word line signal arrives at the cells, thus hiding part of the latency. Dotted lines in the figure denote address and the wordlines of each row of a sub-array.

When a sub-array is selected for access, it is forward body-biased by applying a 0.5 V body voltage of all the nMOS transistors of the cells. If the width of sub-array matches the read/write size (e.g., one 32-bit word) only one sub-array at a time is forward biased; therefore, all other sub-arrays stay in a slow, 0-bias, high-V_t state, thus dissipating very little leakage. In this architecture, there is no need of time-outs or even more sophisticated policies. Entering the low-power state occurs within the cycle.

V. Discussion

Since comparing the various solutions in terms of absolute leakage savings is clearly unfeasible, this section presents a qualitative comparison among the various solutions. Although serving as a summary of this work, such analysis provides useful design insights for SRAM architects and designers.

The analysis ranks the various categories of approaches according to three abstract metrics, namely, *Efficiency* (i.e., tradeoff between leakage reduction and implementation overhead), *Tolerance to Variability* (i.e., robustness to process variability), and *Scalability* (i.e., how efficiency scales with technology scaling). For each class of solutions, a qualitative score (Poor, Moderate, Good) has been assigned (see Table IV).

TABLE IV
QUALITATIVE COMPARISON OF LEAKAGE REDUCTION STRATEGIES

Strategy	*Efficiency*	*Tolerance to Variability*	*Scalability*
Bitline Design	Poor	Poor	Good
Wordline Design	Poor	Moderate	Good
Bitcell Design	Good	Poor	Poor
DVS-based DPM	Moderate	Poor	Poor
PG-based DPM	Moderate	Good	Good
RBB-based DPM	Good	Moderate	Poor

Efficiency: Bitline and wordline design strategies have limited effectiveness at a relatively high implementation cost and performance overhead. This is because the share of bitline leakage with respect to the entire budget is small. Concerning overheads, bitline techniques, and in particular those that are based on pre-charging, may induce sensible read/write latency degradation due to extra pre-charge cycles. Similarly, wordline design schemes, and in particular those based on special value assignment, are characterized by large dynamic power overhead for generating the special voltage levels. Conversely, bitcell design strategies are more effective, but they must carefully weight side effects such as area and cell stability. Solutions based on DPM can be considered, overall, the most efficient. All of them are very flexible since they can be applied at different levels of granularity Power gating (PG) is the most effective of all (it almost nullifies standby leakage power), but it requires a sleep transistor that may induce delay penalty in active mode; moreover, data are lost when the memory moves to the standby mode. Voltage scaling (DVS) has no overhead during the active periods, but it impairs stability during idle periods. Body voltage modulation (RBB) has the least overhead (access time and cell stability are unaffected since the bias is zero during active periods), with benefits comparable to those of DVS.

Tolerance to Variability: Sensitivity to parameter variations is another important metric. Bitline and wordline design schemes tend to be very susceptible to parametric variations: They rely on the compensation of bitline currents achieved by carefully-sized devices, which, however, due to variability, cannot be precisely designed as intended. Similar considerations apply to bitcell design strategies, which play with the threshold voltage of the internal transistors (known for being the most variable parameter). More tolerant are those approaches that assign special voltage values to the wordlines, even if parametric variations may reduce the resulting leakage savings. Concerning DPM, DVS is the most sensitive to process variations; due to variability of the threshold voltage, the margin between $V_{\rm dd}$ and the minimum retention voltage can become excessively small, with a negative impact on the stability of the data. Similar arguments apply to RBB. PG is the most robust architecture: Variations in the sleep transistor may have some performance impact, but its effectiveness in leakage reduction is only marginally sensitive to variability.

Scalability w.r.t. Technology: The main effect of technology scaling that impacts the proposed techniques is related to the reduction of the overdrive voltage, which has a significant impact on most of the leakage reduction techniques. A reduced gate overdrive, for instance, limits the voltage scaling that can be applied during data retention. This has negative impacts on DVS-based solutions. Similarly, having nominal $V_{\rm th}$ closer to $V_{\rm dd}$ makes the control of intermediate $V_{\rm th}$ implants quite difficult, thus making the use of asymmetric bitcells more challenging. The effects of CMOS scaling on RBB-based approaches is even more critical. These schemes become less effective in nanoscale dimensions due to worsening of the body effect caused by shorter channel length. The efficacy of power-gating, on the other hand, does not show any evident limitation with CMOS scaling. The same applies to bitline and wordline design strategies, which however may require special design efforts due to difficulties in controlling the sensing of bitline leakage.

VI. CONCLUSIONS AND FINAL REMARKS

In this paper, we have provided an exhaustive overview of solutions for reducing leakage power in SRAMs. While all the proposed strategies share the same objective of leakage power reduction, they are quite diverse in many aspects.

Drawing a possible set of universal guidelines for SRAM designers and architects is not immediate. However, we can observe a couple of facts that allow a few suggestions that can have general value.

a) Exploit Orthogonality of Strategies: Although some techniques (e.g., bitline and wordline design) have a moderate impact in absolute terms, they are orthogonal to techniques that are based on DPM. The same consideration applies to the customized design of the bitcell. Therefore, whenever the re-design of the internals of the SRAM architecture is allowed, designer should try to apply such techniques to decrease the leakage cost of basic memory operations (bitline/wordline access and reads/writes).

b) Technology Matching: Although many techniques do not scale nicely with technology and/or scaling, designers should try to match the various techniques with the target technology. A technique may become less relevant in future technologies but might be the most suitable for the current ones.

A good example is body-biasing. Although it is expected to become less efficient for nodes beyond the 32 nm, it represents a good solution for 65 nm or 90 nm nodes. Therefore, especially in the embedded domain where technology scaling is limited by the integration of other types of technology (e.g., embedded FLASH memories) and mixed technologies on the same chip are not uncommon [36], body-bias still remains an efficient knob to control leakage.

REFERENCES

[1] "International Technology Roadmap for Semiconductors,"2009 ed. [Online]. Available: http://www.itrs.net/Links/2009ITRS/Home2009.htm

[2] K. Roy, S. Mukhopadhyay, and H. Mahmoodi, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Sep. 2003.

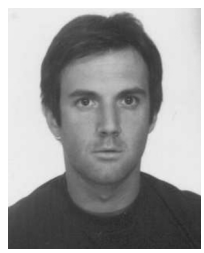
[3] S. Mukhopadhyay, A. Raychowdhury, and K. Roy, "Accurate estimation of total leakage in nanometer-scale bulk CMOS circuits based on device geometry and doping profile," *IEEE Trans. Computer-Aided Design*, vol. 24, no. 3, pp. 363–381, Mar. 2005.

[4] K. Agawa, H. Hara, T. Takayanagi, and T. Kuroda, "A bitline leakage compensation scheme for low-voltage SRAMs," *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 726–734, May 2001.

[5] Y.-C. Lai and S.-Y. Huang, "X-calibration: A technique for combating excessive bitline leakage current in nanometer SRAM designs," *IEEE J. Solid-State Circuits*, vol. 43, no. 9, pp. 1964–1971, Sep. 2008.
[6] S. Hsu, A. Alvandpour, S. Mathew, S.-L. Lu, R. K. Krishnamurthy, and S. Borkar, "A 4.5-GHz 130-nm 32-KB L0 cache with a leakage-tolerant self reverse-bias bitline scheme," *IEEE J. Solid-State Circuits*, vol. 38, no. 5, pp. 755–761, May 2003.
[7] H. Seongmoo, K. Barr, M. Hampton, and K. Asanovic, "Dynamic fine-grain leakage reduction using leakage-biased bitlines," in *Proc. ISCA*, May 2002, pp. 137–147.
[8] K. Itoh, A. R. Fridi, A. Bellaouar, and M. I. Elmasry, "A deep sub-V, single power-supply SRAM cell with multi-V_T, boosted storage node and dynamic load," in *Proc. Symp. VLSI Circuits*, Jun. 1996, pp. 132–133.
[9] Y. Ye, M. Khellah, D. Somasekhar, A. Farhang, and V. De, "A 6-GHz 16-kB L1 cache in a 100-nm dual-VT technology using a bitline leakage reduction (BLR) technique," *IEEE J. Solid-State Circuits*, vol. 38, no. 5, pp. 839–842, May 2003.
[10] A. Alvandpour, D. Somasekhar, R. Krishnamurthy, V. De, S. Borkar, and C. Svensson, "Bitline leakage equalization for sub-100 nm caches," in *Proc. ESSCIRC*, Sep. 2003, pp. 401–404.
[11] N. Azizi, F. N. Najm, and A. Moshovos, "Low-leakage asymmetric-cell SRAM," *IEEE Trans. VLSI Syst.*, vol. 11, no. 4, pp. 701–715, Aug. 2003.
[12] J.-J. Kim *et al.*, "Relaxing conflict between read stability and writability in 6T SRAM cell using asymmetric transistors," *IEEE Electron Device Lett.*, vol. 30, no. 8, pp. 852–854, Aug. 2009.
[13] L. Chang *et al.*, "Stable SRAM cell design for the 32 nm node and beyond," in *Proc. Symp. VLSI Technol.*, Jun. 2005, pp. 128–129.
[14] B. H. Calhoun and A. P. Chandrakasan, "A 256-kb 65-nm subthreshold SRAM design for ultra-low-voltage operation," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 680–688, Mar. 2007.
[15] S. Lin, Y. B. Kim, and F. Lombardi, "A low leakage 9T SRAM cell for ultra-low power operation," in *Proc. GLSVLSI*, May 2008, pp. 123–126.
[16] 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. 3, pp. 349–357, Mar. 2005.
[17] A. Bogliolo, L. Benini, E. Lattanzi, and G. De Micheli, "Specification and analysis of power-managed systems," *Proc. IEEE*, vol. 92, no. 8, pp. 1308–1346, Aug. 2004.
[18] M. Powell, S. H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Reducing leakage in a high-performance deep-submicron instruction cache," *IEEE Trans. VLSI Syst.*, vol. 9, no. 1, pp. 77–89, Feb. 2001.
[19] A. Agarwal, H. Li, and K. Roy, "DRG-cache: A data retention gated-ground cache for low power," in *Proc. ACM/IEEE DAC*, Jun. 2002, pp. 473–478.
[20] H. Makino, T. Shimizu, and T. Arakawa, "An auto-backgate-controlled MT-CMOS circuit," in *Proc. Symp. VLSI Circuits*, Jun. 1998, pp. 41–42.
[21] C. H. Kim and K. Roy, "Dynamic Vt SRAM: A leakage tolerant cache memory for low voltage microprocessors," in *Proc. ISLPED*, Aug. 2002, pp. 251–254.
[22] N. K. Shukla, R. K. Singh, and M. Pattanaik, "Design and analysis of a novel low-power SRAM bit-cell structure at deep-sub-micron CMOS technology for mobile multimedia applications," *Int. J. Adv. Comp. Sci. Appl.*, vol. 2, no. 5, pp. 43–49, 2011.
[23] P. Elakkumanan, A. Narasimhan, and R. Sridhar, "NC-SRAM—A low-leakage memory circuit for ultra deep submicron designs," in *Proc.IEEE Int. SOC Conf.*, Sep. 2003, pp. 3–6.
[24] K. Flautner, N. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: Simple techniques for reducing leakage power," in *Proc. ISCA*, May 2002, pp. 148–157.
[25] M. Kandemir, M. J. Irwin, G. Chen, and I. Kolcu, "Compiler-guided leakage optimization for banked scratch-pad memories," *IEEE Trans. VLSI Syst.*, vol. 13, no. 10, pp. 1136–1146, Oct. 2005.
[26] O. Golubeva, M. Loghi, E. Macii, and M. Poncino, "Architectural leakage power minimization of scratchpad memories by application-driven sub-banking," *IEEE Trans. Comput.*, vol. 59, no. 7, pp. 891–904, Jul. 2010.
[27] K. Nii, H. Makino, Y. Tujihashi, C. Morishima, Y. Hayakawa, H. Nunogami, T. Arakawa, and H. Hamano, "A low-power SRAM using auto-backgate-controlled MT-CMOS," in *Proc. ISLPED*, Aug. 1998, pp. 293–298.
[28] H. Kawaguchi, Y. Itaka, and T. Sakurai, "Dynamic leakage cut-off scheme for low-voltage SRAMs," in *Proc. Symp. VLSI Circuits*, Jun. 1998, pp. 140–141.
[29] T. Ishihara and K. Asada, "An architectural level energy reduction technique for deep-submicron cache memories," in *Proc. ASPDAC*, Jan. 2002, pp. 282–287.
[30] C. Zhang, J. Yang, and F. Vahid, "Low static-power frequent-value data caches," in *Proc. DATE*, Feb. 2004, pp. 214–219.
[31] K. Patel, L. Benini, E. Macii, and M. Poncino, "STV-cache: A leakage energy-efficient architecture for data caches," in *Proc. GLSVLSI*, Mar. 2006, pp. 404–409.
[32] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in *Proc. ISCA*, Jun. 2001, pp. 240–251.
[33] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte, "Adaptive mode control: A static-power-efficient cache design," *ACM Trans. Embedded Comput. Syst.*, vol. 2, no. 3, pp. 347–372, Aug. 2003.
[34] X. Lu and Y. Fu, "Reducing leakage power in instruction cache using WDC for embedded processors," in *Proc. ASPDAC*, Jan. 2005, pp. 1292–1295.
[35] A. Sakanaka, S. Fujii, and T. Sato, "A leakage-energy-reduction technique for highly-associative caches in embedded systems," *Proc. ACM SIGARCH Comput. Architecture News*, vol. 32, no. 3, pp. 50–54, Jun. 2004.
[36] E. Guidetti, G. Notarangelo, and E. Salurso, "Ultra low power REISC SoC design synthesis flow using DC and UPF-based methodology," in *Synopsys User Group Conference: SNUG*, France, 2010 [Online]. Available: http://www.synopsys.com/solutions/endsolutions/eclypses-olutions/pages/TechnicalPapers.aspx

Andrea Calimera (S'08–M'11) received the B.S. and the M.S. (*summa cum laude*) degree in electronics engineering from Politecnico di Torino, Torino, Italy, in 2004 and 2007, respectively.

He joined the Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy, first as Research Assistant, from March 2007 to December 2007, and then as a Ph.D. student. He is currently an Assistant Professor of Computer Engineering at Politecnico di Torino. His research interests focus on physical design challenges, with particular emphasis on several aspects of the development of algorithms, methods and CAD solutions for low-power and design for variability.



Alberto Macii (M'98–SM'07) received the Laurea degree in computer engineering and the Ph.D. degree in computer engineering from Politecnico di Torino, Torino, Italy.

Currently, he is an Associate Professor with the Dipartimento di Automatica e Informatica, Politecnico di Torino. His research interests include several aspects of the computer-aided design of nanoelectronic circuits and systems, with particular emphasis on methodologies, algorithms and tools for power estimation and optimization of systems described at various levels of the design hierarchy. He has authored or coauthored over 130 scientific publications in the areas above, including a scientific book on memory design techniques.

Enrico Macii (M'92–SM'01–F'06) received the Dr.Eng. degree in electrical engineering from Politecnico di Torino, Torina, Italy, the Dr.Sc. degree in computer science from Universita' di Torino, Torino, Italy, and the Ph.D. degree in computer engineering from Politecnico di Torino, Torino, Italy.

He is a Full Professor of Computer Engineering at Politecnico di Torino, Torino, Italy. Prior to that, he was an Associate Professor (from 1998 to 2001) and an Assistant Professor (from 1993 to 1998) at the same institution. From 1991 to 1995 he was also an Adjunct Faculty at the University of Colorado at Boulder Since year 2007, he

is the Vice Rector for Research, Technology Transfer and EU Affairs at Politecnico di Torino, and a Member of the Rectors Advisory Board. His research interests are in the design automation of digital circuits and systems, with particular emphasis on low-power design aspects. In the fields above, he has authored over 350 scientific publications, including the book: "Ultra Low-Power Electronics and Design" by Kluwer. He received the Best Paper Award for articles presented at IEEE EURODAC-96 and at ACM/IEEE GLS-VLSI-08. He was the Editor-in-Chief of the IEEE Trans. CAD/ICAS for the term 2006–2009. Prior to that, he was an Associate Editor for the same journal (1997–2005) and an Associate Editor for the ACM Trans. Design Automation of Electronic Systems (2000–2005). Currently, he is an Associate Editor for the IEEE Trans. Computers.

Dr. Macii is a Fellow of the IEEE. He was a Member of the Board of Governors of the IEEE Circuits and Systems Society for two consecutive terms of duty (2002–2004 and 2005–2007), and a Member of the Board of Governors of the IEEE Council on EDA for the period 2006–2009. Currently, he is the Vice President for Publications of the IEEE Circuits and Systems Society for the term 2011–2012. He has already served in this position during the previous term (2009–2010).

Massimo Poncino (M'97) received the Ph.D. degree in computer engineering and the Dr.Eng. degree in electrical engineering from the Politecnico di Torino, Torino, Italy.

He is currently a Full Professor of computer science at Politecnico di Torino. His research interests include several aspects of design automation of digital systems, with particular emphasis on the modeling and optimization of low-power systems. He is the author or coauthor of more than 200 journal and conference papers, as well as a book on low-power memory design.

Prof. Poncino is also a Member of the Association for Computing Machinery's Special Interest Group on Design. Automation Low-Power Technical Committee. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and a member of the technical program committee of several technical conferences, including the International Symposium on Low Power Electronics and Design, the Design, Automation, and Test in Europe, and IEEE Great Lakes Symposium on VLSI.