1st Intl. Workshop on Emerging Memory Solutions, DATE Conference 2016, Dresden, Germany

Enabling Low Leakage SRAM Memories at system level: A case study

Ajay Kapoor
NXP Semiconductors
Eindhoven, The Netherlands
Ajay.Kapoor@nxp.com

Nur Engin
NXP Semiconductors
Eindhoven, The Netherlands
Nur.Engin@nxp.com

***Abstract*— In this paper, we show the feasibility of low supply voltage for SRAM (Static Random Access Memory) by adding error correction coding (ECC). In SRAM, the memory matrix needs to be powered for data retentive standby operation, resulting in standby leakage current. Particularly for low duty-cycle systems, the energy consumed due to standby leakage current can become significant. Lowering the supply voltage (VDD) during standby mode to below the specified data retention voltage (DRV) helps decrease the leakage current. At these VDD levels errors start to appear, which we can remedy by adding ECC. We show in this paper that addition of a simple single error correcting (SEC) ECC enables us to decrease the leakage current by 45% and leakage power by 72%. We verify this on a large set of commercially available standard 40nm SRAMs.**

***Keywords—* Low leakage; SRAM; Data retention voltage (DRV); Error correcting coding (ECC); Hamming code; SECDED**

I. Introduction

A new class of smaller, cheaper computers is developed approximately every decade. This trend was identified by Gordon Bell, and it is referred to as Bell's Law [1]. This trend is continued with current millimeter-scale computer systems for ubiquitous applications in domains such as Internet-of-things (IoT), wearable computing, smart home and office environments, smart grid solutions, and personal identification products. Low power design solutions are at the heart of this evolution to improve form-factor, design costs and battery life. It is important to realize that for these (low duty-cycle) applications, energy consumption in static modes (leakage) with data retention is often equally or more important than dynamic energy consumption [2].

A common method of decreasing (static and dynamic) power consumption is supply voltage (VDD) scaling [3]. Although VDD scaling is an effective method to decrease power consumption in digital logic, possibility for VDD scaling on memory (SRAM) is limited. With CMOS technology scaling (especially 90nm and beyond), the problem gets worse due to increased leakages and limited voltage reduction in SRAM compared to logic [4][5]. For example, moving from 90nm to 45nm technology, SRAM supply voltage is scaled from 1.2V to 1.1V only. It is also observed that the relative contribution of leakage energy due to on-chip memory is increasing due to increased amount of memory content [6]. This causes the standby leakage of SRAM memory to become a major bottleneck for ultra-low power data retentive standby operation.

In standby mode, the bit cell matrix needs to remain powered for data retention and peripheral circuitry can be powered-off [7]. Consequently, the bit cell matrix is responsible for leakage during the standby operation. During standby operation, the supply voltage of SRAM memory cells can be decreased while ensuring data-retention. This critical voltage level for reliable data retention is called data-retention voltage (DRV). Typically, leakage current value grows exponentially with the VDD value during retention [8]. There has been extensive research on design techniques that attempt to decrease standby leakage for SRAM memory. In general, the approaches can be categorized into three groups:

A) *Bit cell design optimizations* such as design of different 6T cells or cells with more transistors [7], selective power gating[9], simultaneous dynamic voltage scaling and adaptive body biasing[10]. These approaches require a redesign of the memory cell. Observe that SRAM bit cell design cycle is quite long due to relatively large amounts of testing and validation data assessment. Therefore, many SOC design companies are relying on the bit cells supplied by the memory vendors (especially for advanced technology nodes) and SOC designers cannot do any more optimization on bit cell themselves. Furthermore, in general approaches with more transistors are costlier in silicon area.

B) *Memory architecture optimization* e.g. in [11], SRAM capacity is reduced at lower voltage by masking the error bits. However, memory requirements in applications are fixed and cannot be reduced generally. In [12], error bits are mapped to additional redundant bits. This makes the address decoding logic exceedingly complex. Overall, such techniques are very intrusive to overall system design which make them difficult to deploy.

C) *Error correction coding (ECC) for SRAM* has been proposed in recent years for decreasing SRAM power [13][14]. This approach is based on observing that as VDD gets lower, leakage power decreases and number of error bits gradually start increasing during retention. The proposed method uses error correction coding (ECC) to eliminate these error bits. Mostly single error correcting codes like hamming codes have been used for this aim. In general, deploying ECC is relatively straightforward and can be invisible to the rest of the system, provided that the system requirements (e.g., speed, area etc.) can be met. Moreover, ECC techniques are already industrially accepted to improve the reliability [15].

Various published approaches in B) and C) rely on adding limited redundant bits in the SRAMs to overcome the bit-errors due to standby mode voltage reductions. However, theoretical analysis to determine the relationship between the amounts of redundancy needed for a corresponding supply voltage scaling is missing. It must also be noted that adding redundancy comes at a cost (area, dynamic power, speed), which should be evaluated at system level. To the best of our knowledge, system level evaluation of incorporating ECC based SRAM is not covered in the literature. Moreover, the conclusions in past publications have not been based on statistically significant sets of measurements to reach the bit error probability (BER) levels targeted for reliable memory operation. These limitations restrict the adoption of above techniques for industrial purposes. In this paper, we aim to overcome the above limitations. The theoretical analysis is demonstrated for approach C), but can easily be extended for approach B).

In section II, we introduce the architectural constraints, tradeoffs with respect to the choice of ECC and the conditions for applicability of ECC to a particular SRAM memory which should be considered before deploying ECC. In section III, we derive the empirical relationship between SRAM BER with and without ECC. Here, we will also discuss how to correlate the above with the voltage scaling limit for SRAM. The above theoretical analysis is used and verified on 40nm testchip in section IV. Here, statistically sufficient measurements results for SRAM BER at low voltages are shown. These measurements verify that the application of simple single error correcting (SEC) ECC can result in significant power savings in standby mode. In section V, we present synthesis and power analysis results on an ECC design and analyze the added overhead in power and area due to addition of ECC and discuss the tradeoffs for low duty-cycle systems, and finally in section VI, we present the conclusions of our work.

II. Architectural constraints

In this section, we begin by introducing the system level constraints and voltage scaling constraints for using ECC based SRAM in the design. These constraints are applied to evaluate the system performance using ECC in the following sections.

A. *System Design Limitations*

ECC is widely applied to semiconductor memories to prevent errors in memory [16][17]. Although the most common use is in DRAM and flash memories, some SRAM memories also include an ECC option, to prevent soft or lifetime errors on the operation [15][18]. However, system level design trade-offs should be accounted for the feasibility of ECC. Clearly, adding ECC increases the amount of memory (coding adds redundancy bits) and logic (for encoding and decoding). This results in additioal area. Furthermore, such a system requires that during normal operation, data is to be ECC-encoded before writing to SRAM and ECC-decoded after reading from SRAM. This will increase the data access time to SRAM, i.e. speed and latency will be negatively impacted. As ECC will become part of an embedded memory subsystem, the system clock and memory access time will limit the latency the encoding and decoding circuit is allowed to have. To give an example, a system with 50 MHz (20ns clock period) is quite common for low-duty cycle applications. If we assume that the data access time of SRAM accounts for about 40% of timing critical path, i.e. 8ns, the ECC encoder/decoder needs to have a latency that is relatively shorter, if it is to be hidden in the access cycle. For this reason, we try to limit our analysis to very simple ECC that has 1-2ns of latency. Moreover, due to wider SRAM wordsize and ECC (Encoder, Decoder) requires additional dynamic power. These costs must be evaluated with the benefit of static power decrease that will be achieved by adding ECC. This trade-off is especially relevant for low duty-cycle systems that are limited by static energy use. These are the systems we are targeting in this work.

B. *ECC Trade-offs and Voltage Scaling Constraints*

In general, area and latency overhead of different ECC types have been investigated by others, e.g. in [19]. In our work, we do not wish to repeat this analysis but instead choose directly for SEC Hamming codes because of their simplicity and low costs. It is clear that our stringent area and latency overhead requirements (as explained previously) will not be possible to satisfy with a higher complexity code [19]. We will elaborate further on the overhead of ECC in section V. Before moving further let us quickly revise the basics of Hamming code.

Hamming codes are linear block codes denoted by *(n, k)* where k is the number of data bits and n is the total code word length [20]. For each integer *r*, an *(n, k)* Hamming code word can be constructed where $n=2^r-1$ and $k=n-r$. The code rate becomes k/n. Such a code corrects one error per code word, i.e. 1 error per *n* (encoded) bits, also called a SEC code. It follows that in the original Hamming code word, data size is always $n-k = 2^r-1-r$ bits, e.g. the codes become (15, 11), (31, 26), (63, 57), (127,120), etc. In practice, the codes are often shortened according to the data size required, such as (38, 32) SEC code for a 32-bit word size. Furthermore, an additional parity bit is often added to enable the detection (but not correction) of double errors, called single error correcting double error detecting (SECDED) code.

In order to use Hamming codes we need to first establish the conditions for this to be feasible. For this, we can translate number of error bits during retention into BER for SRAM. First condition is that the BER at retention changes gradually with the standby VDD. This is illustrated by hypothetical BER vs VDD graphs as shown in Figure 1. Full lines represent raw BER and dashed lines represent BER when coding is applied. It can be seen that the slope of the graph determines how effective ECC can be applied to lower the VDD. Note that in both cases the BER ‘drop’ the ECC enables is roughly the same, although the VDD advantage is significantly higher in the gradually decreasing case at the right.

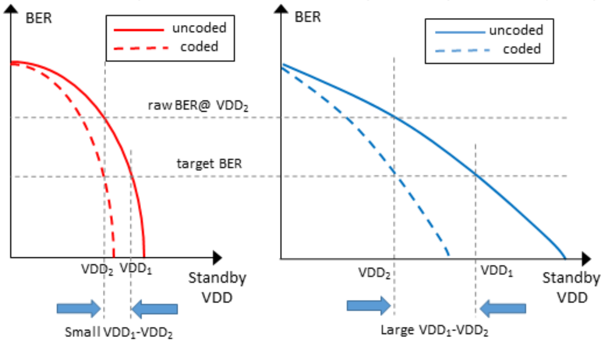

Figure 1 Sharp (left) Vs gradual (right) decrease of raw BER with increased VDD.

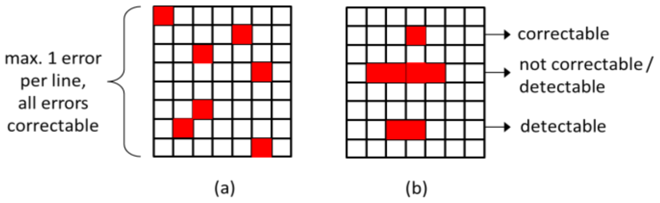
max. 1 error per line, all errors correctable
correctable
not correctable / detectable
detectable
(a)
(b)

Figure 2 SECDED example for error distribution impact: (a) possible to correct errors (b) not possible to correct

Another important condition is the distribution of errors in the memory. This is shown in the example in Figure 2.When multiple errors occur in one code word, a SEC code cannot correct them. Having a SECDED code makes possible to detect the two error case, and the (rare) case of triple error will result in mis-correction. So even if the number of occurring errors is similar, depending on the distribution it can be possible or not to correct them with a SEC/SECDED code.

III. Theoretical Analysis: ECC for Low voltage SRAM

In this section, we present the theoretical analysis needed to derive the impact of ECC on SRAM BER. This helps us in determining the voltage scaling limits to enable low voltage standby mode for SRAM.

A. ECC impact on SRAM BER

We begin by assuming the SRAM bit cell error likelihood to be equal at all cell locations and independent of neighboring errors (in the coming sections, we shall show by means of measurement results that these assumptions hold for the SRAM memory we target). Let the memory in question have M bits in total, whereby each bit has an error probability, p, i.e., uncoded (raw) *BER*=p. Yield is defined by the probability of having no errors in the complete memory:

$$Yield = (1-p)^M \qquad (1)$$

Figure 3 illustrates the relationship given in (1). In this work, we are targeting a 16kB SRAM memory design with 32-bit words and 4096 lines. Accordingly, we need a *BER* of 10^{-9} assuming we need to reach a 99.99% yield. Addition of ECC to correct the errors in the memory matrix changes the *BER*. In this case we talk about coded *BER* for a given p.

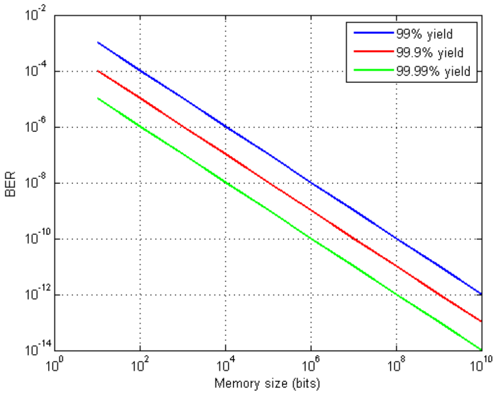

Figure 3 Memory cell bit error rate (BER) for different memory sizes and memory yield.

Let’s derive analytically the relationship between the two for a single error correction per word code. For this, we define *WER* (word error rate), which is the probability of having one or more error bits per memory word. Assuming each memory word is *m* bits, the raw WER can be obtained as

$$WER = 1-(1-p)^m \qquad (2)$$

If we introduce redundancy of *n* bits per word, then the probability of having exactly *f* errors in an *m*-bits word (with *n* bits parity)

$$P_{f\_errors\_out\_of\_m\_bits} = \binom{m+n}{f} p^f (1-p)^{(m+n-f)} \qquad (3)$$

Therefore, for a single error correcting code (*f=1)* for each memory word of size *m*, the coded *WER* and *BER* can by calculated as

$$WER_{coded} = 1-\left((1-p)^{m+n} + \left((m+n)\cdot p\cdot(1-p)^{m+n-1}\right)\right) \qquad (4)$$

$$BER_{coded} = 1-10^{\left(\frac{\log(1-WER_{coded})}{m+n}\right)} \qquad (5)$$

It should be noted that when the independence assumption holds, the likelihood of multiple errors per memory word decreases fast with decreasing p. For example, for *m=32, n=0* and $p=10^{-6}$, the probability of one error per word becomes 3.2 x 10^{-5} while having two errors per word is as low as 5 x 10^{-10}. In other words, if the raw BER is moderately low and the equal likelihood and independence assumptions hold, then the likelihood of burst errors is very low. From the above analysis, we can determine the error correction requirements to fulfill the desired yield. In Figure 4, the tradeoff between the error correction performance versus code word size and memory overhead (*(n-r)/r*) for a SEC code is plotted for several raw BER values. The memory overhead and the corresponding code word size are shown on the left and right axes. By using above, we can link between the raw and coded BER for a given word size requirement. It can be seen here that the target BER of <10^{-9} can be reached from a raw BER of 10^{-5} for codeword size of 63 bits. This corresponds to the shortened code (38,32), which we can use for our system of interest.

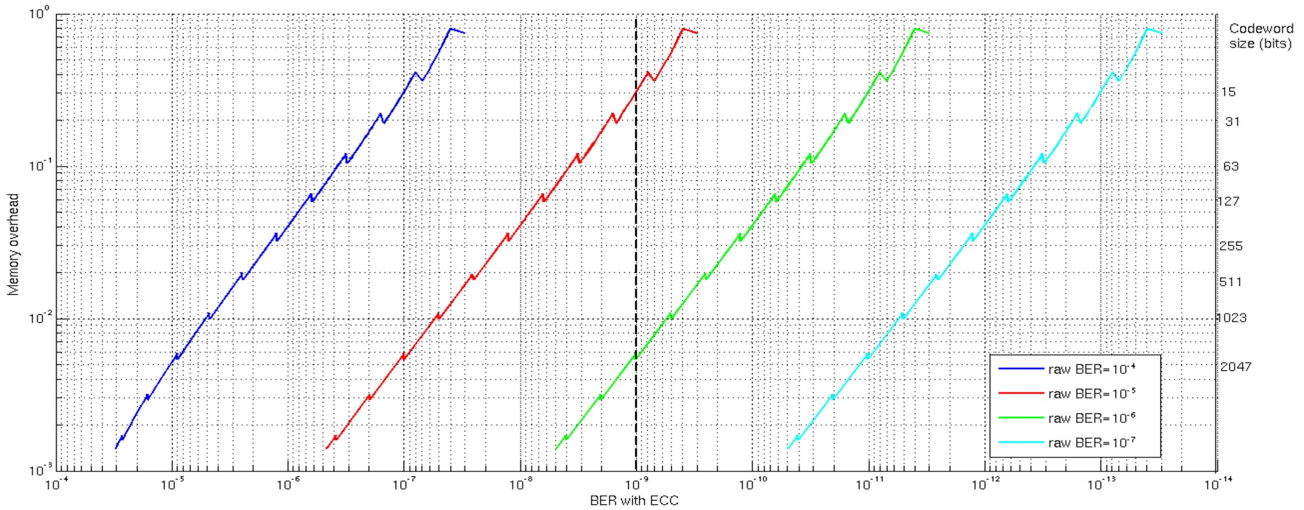

Figure 4 Memory overhead vs BER correction for single error correcting (SEC) Hamming codes.

B. *Voltage scaling limits for ECC Deployment*

Next step is to determine the voltage scaling limits for ECC deployment. This can be done by the mapping of raw BER into the voltage scaling curves to determine the allowed VDD in the system during standby. Our calculations from previous subsection lead us to expect that starting from a moderately low raw BER, a single error correcting code can help significantly in lowering the BER to the target value. The corresponding VDD will be the voltage scaling limit for SRAM. In section IV, we will show with measurement results that this indeed is the case for the SRAM standby errors.

IV. Measurement Results

In this section, we present the measurement results for 40nm SRAM bit cell data retention performance as a function of VDD. Our test chip is realized in commercially available standard 40nm CMOS process. It contains multiple instants of 16kB SRAM arranged in 4096 words of 32 bits each and measurement board is shown in Figure 5. The standard specifications for such memories suggest 0.8V minimum voltage during standby. To perform measurements, we first load the desired data into the SRAM at nominal voltage. After that, the clock to the memory is stopped (i.e. standby mode), then we lower the SRAM supply voltage and wait for 1s. Next, we raise the voltage again to nominal value and read the contents of the memory. Comparing the readout data and original data provides us an error map of the SRAM for data retention at low voltage. After this, SRAM data is refreshed. The same procedure is carried out repeatedly for various voltage settings. Moreover, this testing is carried out on multiple datasets: all 0s, all 1s, alternate 1010s, alternate 0101s and all random bits. The tests are carried out on 21 different samples. In total, 14-Mbit of SRAM was measured at room temperature and SRAM supply was lowered in steps down to 275mV. On the measured data, the two main conditions from the previous section are verified next.

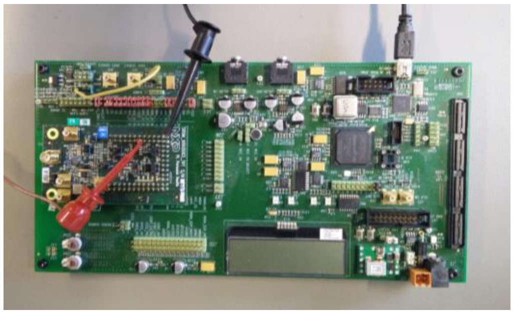

Figure 5 40nm test chip and board setup used for measurements to determine BER vs standby VDD.

Figure 6 shows the measured BER as a function of supply voltage during data retention. Note that there is no sudden drop in the BER curve, which is, as discussed earlier and depicted in Figure 1, encouraging for the potential drop in VDD using ECC. Furthermore, as shown in Figure 4, a BER of 10^{-5}-10^{-6} is sufficient to reach the target BER of 10^{-9} using ECC assuming equally likely and independent errors. This means that ECC can allow us keeping the SRAM at 400mV during standby mode for data retention. Also, observe that the total amount of tested bits provide sufficient confidence for this conclusion.

We still need to confirm the equal likelihood and independence assumptions we made in section II. To this end, we analyzed the cumulative bit error map plotted in Figure 7. Here the 32x4096 bit memory is shown wrapped into a matrix of size 256x512. In this matrix, the cumulative number of errors (from 0.8V to 0.4V) from different memory instants and samples are noted. An extensive analysis of the bit errors has shown that no burst errors are visible above a supply voltage of 375mV. Below 325mV, we start seeing a few double errors. This is however expected as likelihood of two errors per word becomes high (around 10^{-4}) at the BER (10^{-3}) at this voltage, as can be calculated from equation (3). Above 400mV this probability becomes much lower (~ 10^{-8} due to BER=10^{-5}). To sum up, it can be seen that burst errors are not likely in the measured memories and the error likelihood seems in accordance with the equal likelihood and independence

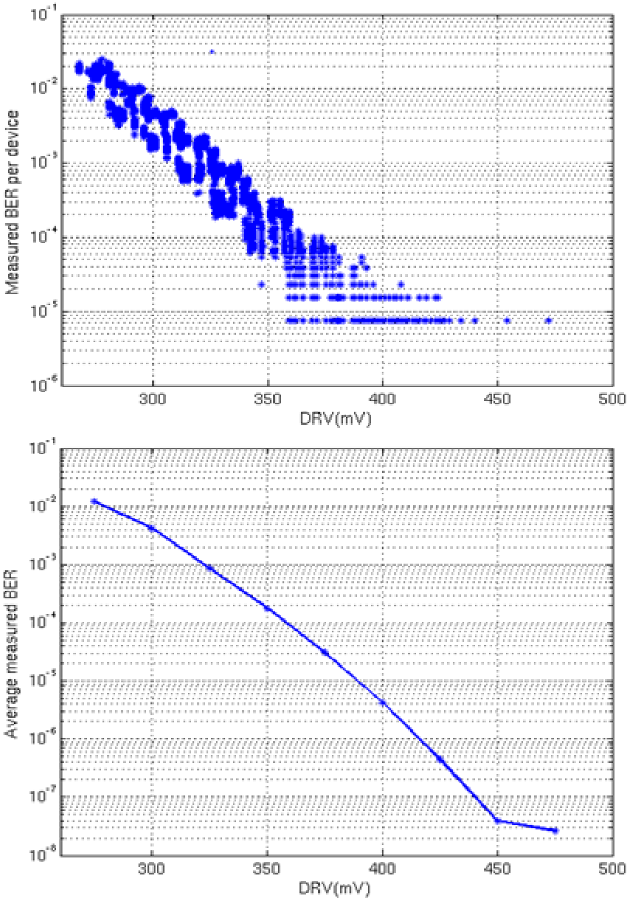

Figure 6 Bit error rate (BER) vs standby VDD (measured data). Graph above shows sample points as measured and the graph below depicts the average BER.

conditions of section II. By zooming into different areas, examples of the random behavior discussed above can be seen. From these measurements, we conclude that data retention requirements during standby mode can be met for SRAM supply voltage of 0.4V and above in these SRAM memories. Similar results have been measured for different temperatures from -40C to 85C. This concludes that a SEC scheme can be deployed to facilitate the SRAM standby mode voltages from 0.8V to 0.4V.

V. System view for SRAM with ECC

After noting the feasibility of ECC during standby mode, let us evaluate the ECC deployment cost and benefits in the overall system (as described in qualitatively in section II). To quantify all this, we look into the hardware cost of ECC for an SRAM module of 4k words of 32 bit each in commercially available 40nm CMOS process. Such an SRAM has a speed of approximately 130 MHz in slow corner, 0.99V VDD, -40C. It requires an area of 64000µm^2 and consumes 11µA/MHz for its read/write operations in dynamic mode. We use for our analysis here a (39, 32) SECDED design consisting of encoder and decoder units. Our synthesis results indicate that SECDED design (39, 32) requires 580 µm^2 logic cell area with delay-time of 1.6ns at optimal performance-per-area (PPA) [2]. For encoding and decoding, power consumption is 0.3µA/MHz and 2.2µA/MHz respectively. Assuming an equal amount of read and write to the SRAM, the ECC block will consume 1.25µA/MHz Additionally, we will now require 39 bits 4k word SRAM in order to accommodate the 7 parity bits (note that the use of SEC will decrease this overhead. However, at low voltages double error detection starts to be required. In this case, the SRAM area becomes 74000µm^2 and dynamic power consumptions increases to 13µA/MHz power at 130MHz throughput. This means that our example SRAM memory will have 20% speed, 15% area and 29% dynamic power overhead. By reducing the SRAM voltage from 0.8V to 0.4V during standby mode, a leakage current drop of 45% and a power consumption drop of 72% is achieved under typical process and temperature conditions. These figures include the overhead of increased memory due to additional parity bits in SRAM due to ECC unit. In Table 1, these numbers are summarized in normalized form. Note that in the standby mode no data access to SRAM is allowed (hence, retention only), while in dynamic mode data access is allowed. For overall system energy budgeting, one must carefully analyze the impact of ECC in standby and dynamic operating mode.

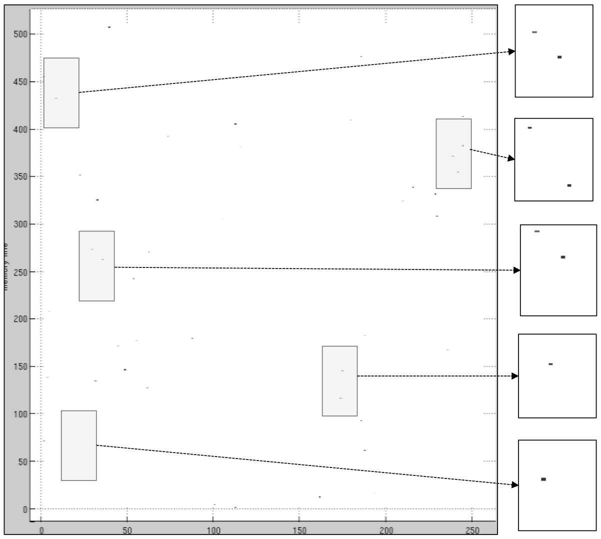

Figure 7 Cumulative bit error map for 0.4V to 0.8V VDD illustrating the absence of burst errors

Table 1 : Comparison overview (normalized)

Mode	Parameter	Existing approach	Proposed approach		
		SRAM	SRAM	ECC Encoder & Decoder	Overall impact
	Area	1	1.15x	~0	1.15x
Standby	Min. VDD	1	0.5x	0	0.5x
	Current	1	0.55x	0	0.55x
	Power	1	0.28x	0	0.28x
Dynamic (Read /Write)	VDD	1	1x	1x	1x
	Current	1	1.18x	0.114x	1.29x
	Power	1	1.18x	0.114x	1.29x
	Latency	1	1x	0.2x	1.2x

Typically, in an overall microcontroller system where SRAM consumes about 20-30% of total power [21], one will have about 5-9% dynamic power overhead due to ECC in SRAM (refer to Table 1). However, during standby mode operation a large portion of energy consumption comes from the SRAM leakage. Standard microcontroller datasheet indicates 15-99% standby mode leakage contribution from SRAMs depending on the data retention requirements [22]. For various low duty cycle applications [23][24] where standby mode leakage power is more dominant for overall energy budgeting and battery lifetime, the overhead in the dynamic mode can be accommodated due to significantly lower energy during standby mode. On the other hand, if energy costs in dynamic mode dominates, the ECC deployment is not advisable. This correlation can be used to quantify the duty-cycle beyond which ECC should be activated in a system. It must also be noted that the area overhead due to ECC encoder and decoder logic is not of primary interest, as the simplicity of Hamming codes and the shrinking CMOS feature sizes enable the ECC area overhead to become insignificant.

VI. Conclusions

We have demonstrated the feasibility of ECC for reducing the standby leakage current of SRAM memories. Using both theoretical analysis and large set of memory error test measurements, we have shown that ECC based SRAM can be used to retain data until significantly lower voltages than standard SRAM specifications (shown here for 0.8V to 0.4V standby VDD drop for 4kx32 SRAM). This can provide substantial (up to 45%) decrease in standby leakage current for SRAM memories in commercially available standard 40nm CMOS process. The scaling down of voltage results in 72% decrease in leakage power for SRAM. This has shown to be practically feasible with a single error correcting code. There is no change to the memory instance interface to the system, which enables straightforward integration to a SoC. The additional processing due to ECC encoding and decoding result in power consumption overhead in dynamic mode. We have shown that the power overhead for one ECC unit per memory instance of 4kx32 bits can result in 27% increase in dynamic power consumption for SRAM. However, the overhead is 5-8% for the overall system. In terms of latency, using a simple SEC encoder/decoder enables relatively low latency. For this design, it is around 1.6ns, however, with appropriate optimization it can be decreased further.

To sum up, we have shown that addition of simple SEC ECC can help decrease the standby power in SRAM memories significantly, e.g. 70% in the presented system. The addition of ECC has minimum impact in the memory access latency and the integration is straightforward. Increase in system area and dynamic power are relatively small

Acknowledgments

The authors would like to thank Tom Huybrechts, Patrick van de Steeg, Steven Thoen, Jose Pineda de Gyvez, Clemens Wouters, Leo Sevat, Semih Serbetli, Rabindra Rijal and other NXP colleagues for their cooperation.

References

[1] G. Bell, *“Bell’s Law for the birth and death of computer classes,”* Commun. ACM, vol. 51, no. 1, pp. 86–94, Jan. 2008.

[2] A. Kapoor. et. al, *“Digital Systems Power Management for High Performance Mixed Signal Platforms”,* IEEE Trans. on Circuits and Systems–I : Regular Paapers, Vol. 61, No. 4, April 2014

[3] M. Meijer, J. Pineda de Gyvez, and A. Kapoor, *“Ultra-low-power digital design with body biasing for low area and performance-efficient operation,”* J. Low Power Electronics, vol. 6, pp. 521–532, Dec. 2010.

[4] System Drivers, *“International Technology Roadmap for Semiconductors,”* http://www.itrs.net, pp. 1–25, 2005

[5] S. Borkar, *"Design challenges of technology scaling,"* IEEE Micro, vol. 19, issue 4, pp. 23-29, Jul-Aug 1999.

[6] F. Ricci et. al, *"A 1.5 GHz 90 nm embedded microprocessor core,"* Digest of Technical Papers. Symposium on VLSI Circuits, pp. 12-15, June 2005.

[7] Huifang Qin, “Deep Sub-Micron SRAM Design for Ultra-Low Leakage Standby Operation”, Ph.d thesis, Univ of California Berkeley, May 2007

[8] H. Qin et. al *,“SRAM leakage suppression by minimizing standby supply voltage,”* in ISQED’04: Proc. of Fifth Intl. Symposium on Quality Electronic Design, 2004, pp. 55–60

[9] S. Kaxiras et. Al, *“Cache decay: exploiting generational behavior to reduce cache leakage power*”, Proc. of the 28th Annual InternationalSymposium on Computer Architecture, pages 240–251, 2001

[10] S. Martin, et. al, *“Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads”,* In Proc. of the 2002 International Conference on Computer Aided Design, pages 721-725, New York, NY, USA, 2002. ACM.

[11] C. Wilkerson, et. al, *“Trading off cache capacity for reliability to enable low voltage operation*”, Proc. Of the 35th Annual International Symposium on Computer Architecture, 0:203-214, 2008,

[12] A. Ansari et al, *“Enabling Ultra Low Voltage System Operation by Tolerating On-Chip Cache Failures”*, ISLPED, 2009.

[13] H. Qin, et. al, *“Error-Tolerant SRAM Design for Ultra-Low Power Standby Operation”*, 9th International Symposium on Quality Electronic Design, 2008 (ISQED 2008), Publication Year: 2008 , Page(s): 30 – 34.

[14] Brian Zimmer, *“Resilient Design Methodology for Energy-Efficient SRAM”*, M.S. Thesis, University of California, Berkeley, May 2012.

[15] K. Osada et. al, *“16.7-fA/cell tunnel-leakage-suppressed 16-Mb SRAM for handling cosmic-ray-induced multierrors”*, IEEE Journal of Solid-State Circuits, Volume: 38 , Issue: 11, Publication Year: 2003 , Page(s): 1952 – 1957

[16] T. Tanzawa et. al, *“A compact on-chip ECC for low cost flash memories”*, IEEE Journal of Solid-State Circuits, Volume: 32 , Issue: 5, Publication Year: 1997, Page(s): 662 - 669.

[17] H.L. Kalteret al, *“A 50-ns 16-Mb DRAM with a 10-ns data rate and on-chip ECC”*, IEEE Journal of Solid-State Circuits, Volume: 25 , Issue: 5, Publication Year: 1990 , Page(s): 1118 – 1128.

[18] S.M. Jahinuzzaman et. al, *“Design and Analysis of A 5.3-pJ 64-kb Gated Ground SRAM With Multiword ECC”,* IEEE Journal of Solid-State Circuits, Volume: 44 , Issue: 9, Publication Year: 2009 , pp 2543 – 2553.

[19] R. Naseer and J. Draper, *‘DEC ECC design to improve memory reliability in Sub-100nm technologies’,* 15th IEEE International Conference on Electronics, Circuits and Systems, 2008, Page(s): 586 – 589.

[20] S. Lin and D.J. Costello, *‘Error Control Coding: Fundamentals and Applications’*, Prentice-Hall, 1983.

[21] Borisav Jovanović et. al, “*Low power digital design in Integrated Power Meter IC”* Proceedings of Small Systems Simulation Symposium, 2010

[22] *http://www.nxp.com/documents/data_sheet/LPC5410X.pdf*

[23] Shih-Lun Chen et al, “*Wireless Body Sensor Network With Adaptive Low-Power Design for Biometrics and Healthcare Applications”,* IEEE SYSTEMS JOURNAL, VOL. 3, NO. 4, DECEMBER 2009

[24] *http://www.ti.com/lit/sg/slab034z/slab034z.pdf*