System-Level SRAM Yield Enhancement

Fadi J. Kurdahi, Ahmed M. Eltawil, Young-Hwan Park
EECS Department, University of California
Irvine, CA 92697-2625
{kurdahi,aeltawil,younghwp}@uci.edu

Rouwaida N. Kanj, Sani R. Nassif
IBM Austin Research Labs
11501 Burnet Road, Austin, TX 78758
{rouwaida,nassif}@us.ibm.com

Abstract

It is well known that SRAM constitutes a large portion of modern integrated circuits, with 80% or more of the total transistors being dedicated to SRAM in a typical processor or SOC. Thus yield management of these SRAMs plays a crucial role in insuring design success. This paper demonstrates analysis techniques to model and improve the yield of SRAMs at the system level by proper accounting for the coupling between the algorithms targeted for an SOC and the performance, power, and yield of SRAMs used in implementing the algorithms. It is shown that coupling the algorithm and SRAM design phases provides significant advantages over independent optimization.

1. Introduction

With continued technology scaling, an increasing proportion of integrated circuits implements SRAM in the form of caches, register files, or other structures. With that same scaling, however, comes an increasing susceptibility of SRAM to manufacturing variations, the impact of which results in SRAM failures at the bit, row, or column levels. Such failures have existed for many years, but have been predominantly caused by defects causing topological changes (short and/or opens) in the SRAM circuit, thus rendering certain parts inoperable. More recently, parametric fluctuations in quantities such MOS device threshold voltages have resulted in SRAM circuits which are topologically correct, but that fail to meet performance or reliability metrics. In [4] for example, the authors illustrate an analysis methodology that can be used for the independent yield enhancement of an SRAM. While clearly an important and relevant problem for general design, creating a "perfect" SRAM is very difficult and expensive. Thus we examine the relationship between (a) the yield specifications placed on an SRAM, which are typically stated as the probability of a bit, row, or column not operating correctly, and (b) the needs, assumptions and requirements of the algorithms executing on the design. Clearly, if these needs are ignored, then the SRAM is required to provide the correct result 100% of the time, and the SRAM designer will then have to use any of several techniques (the most popular of which is redundancy [1]) in order to insure that the SRAM will meet these requirements. If the algorithm implemented has the ability to accept and possibly even correct memory errors, however, then it becomes possible to co-design the algorithm and the memory simultaneously and thereby reduce the burden the SRAM designer faces with technology scaling and the increasing level of ensuing manufacturing variability.

The remainder of this paper is structured as follows. Section 2 introduces a parametric yield model of SRAMs and addresses the need for circuits/systems-level co-design. Sections 3 and 4 introduce two system-level case studies. Section 5 highlights the advantages of exploiting the system-level design space. Finally, conclusions are drawn in Section 6.

2. Problem Formulation

One of the most prominent challenging by-products of feature scaling that is proving extremely difficult to manage is random dopant fluctuations (RDF), which results in variations of transistor threshold voltages that are in close proximity of each other (intra-die variation). In advanced technologies, the concentration of dopants is down from several thousands per transistor to literally a few hundred. These atomic-level intrinsic fluctuations cannot be eliminated by external control of the manufacturing process and are expected to severely affect minimum geometry transistors (which are used extensively in SRAM cells) leading to cell failures and yield degradation. These failures are manifested as either an increase in the cell access time or unstable read and write operations. In this paper, we will focus on parametric failures, which are failures caused by process variations that can be lumped into an effective change in the individual transistor threshold voltage. Since for a single cell the transistors are in close proximity, RDF will be the primary cause of mismatches between the transistors. For a standard 6T SRAM cell, the V_t fluctuations (δV_t) are considered as six independent Gaussian random variables with mean=0 [4](one for each transistor). The assumption of independency is justified since the placement and the number of dopants in each transistor is independent and is a function of that specific transistor geometry. The standard deviation of V_t due to RDF depends on the specific manufacturing process, doping profile and transistor geometry. Equation (1) relates the δV_t of a generally sized transistor as referred to a minimum sized transistor with $\delta V_t = \delta V_{t0}$

$$\delta V_t = \delta V_{t0} \sqrt{\frac{L_{\min}}{L} \frac{W_{\min}}{W}} \qquad (1)$$

Given equation (1), we can consider a function $y = f(x_1 ... x_n)$ where $x_1 .., x_n$ are independent Gaussian random variables with mean $1_1 .., 1_n$ and STD $c_1 .., c_n$. It was shown in [4], that the mean and STD of y can be estimated using multi-variable Taylor-series expansion to be

$$\mu_y = f(\eta_1 .., \eta_n) + \frac{1}{2}\sum_{i=1}^{n} \left. \frac{\partial^2 f(x_1 .., x_n)}{\partial (x_i)^2} \right|_{\eta_i} \sigma_i^2$$

$$\sigma_y = \frac{1}{2}\sum_{i=1}^{n} \left(\left. \frac{\partial^2 f(x_1 .., x_n)}{\partial (x_i)^2} \right|_{\eta_i} \right)^2 \sigma_i^2 \qquad (2)$$

The probability distribution function (PDF) of y will also be Gaussian with the calculated mean and STD. To illustrate this concept, a Monte Carlo simulation was performed on a standard 6T SRAM cell that is shown in Figure 1. The parameters of the circuit are detailed in Table 1. Note that the choice of the control circuitry and device dimensions is for purposes of illustration only, and the conclusions should be generalizable to any other memory circuit. The simulation uses the Berkeley 70nm predictive transistor models [3] as a basis, where we assumed that the V_t of the SRAM cell transistors follow a Gaussian distribution, with the same distribution function parameters as those proposed in [2]. Figure 2 depicts the results of the simulation for the both read and write access time assuming a nominal voltage of 1.0 volts and a nominal read(write) access time of 45ps(36.5ps). The effect of variations is obvious in the Gaussian spread of the access time.

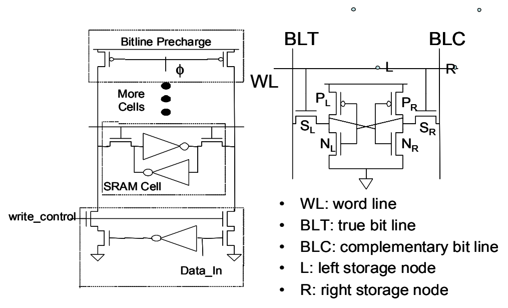

Figure 1 6T SRAM Cell

Table 1 Device dimensions for the circuit presented in Fig. 1

Precharge	$(W/L)_P = 1/0.07$
Write_control	$(W/L)_N = 2/0.07$
SRAM cell	$(W/L)_P = 0.258/0.07$
	$(W/L)_N = 0.15/0.07$

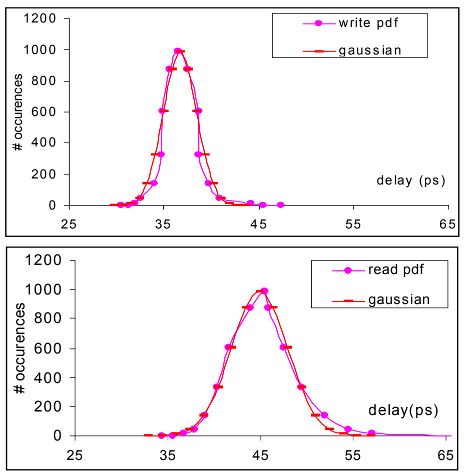

Figure 2 Read and write PDFs obtained by smoothing the Monte-Carlo simulations.[1]

The question at hand now, is, given the fact that variations will cause mismatches between transistors that lead to memory faults; can these issue be alleviated by exploiting the system-level design space to allow enough redundancy in the system for it to tolerate a bounded amount of variation-induced errors?

To illustrate this new design space, Monte Carlo simulations were performed where the cell access time for both read and write is tested under realistic manufacturing variations. For example in read mode, a specific access time is required and the cell is considered to fail, if that access time is not honored. The same methodology applies for write operations. Figure 3 illustrates the simulation setup where the output of the simulation is (a) a matrix of cell failure rate as a function of delay and V_{DD}, and (b) a vector of power as a function of V_{DD}.

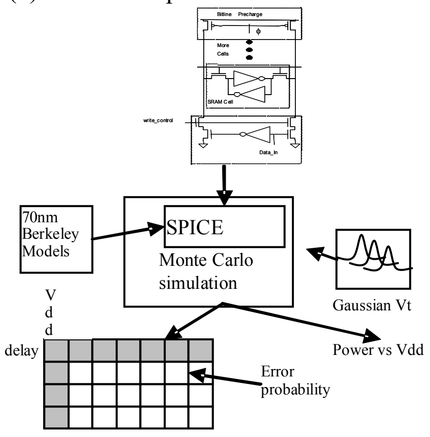

Figure 3. Simulation setup

[1] The corresponding Gaussian approximations rely on mean and STD values derived from the same set of Monte-Carlo simulations. The results shown are for VDD=1.0V.

The simulation was performed for a range of supply voltages and a range of access times as shown in Figure 4 and Figure 5 which respectively depict the contour lines of equal failure probability for a write and read operation. The contour lines depict the following cell failure probabilities: 10^{-4}, 10^{-3}, 1, 5, and 10%. From the graphs, it becomes clear that read operations are more susceptible to failures under the assumptions made in this work.

The interesting observation to make based on these graphs is the large design space that is available between the contour lines representing 10^{-4}% error (the current target for most applications) and 1% error. For example, to operate at 10^{-4}%, the only possible operating point is 1.1 volts at a delay of 65 ps. While if we relax the error requirements to 1% the entire design space bounded by the 1% contour is now available allowing operation –for example– at 0.95 volt with the same 65 ps delay. This expanded design space allows for many tradeoffs such as decreasing power consumption via voltage scaling, or a reduction in the memory correction circuits (such as ECC and BIST/BISR) necessary to protect against memory faults. In the following sections we will explore the feasibility of designing defect tolerant systems that can tolerate up to 1% errors in memory. Specifically we will present two cases studies, one based on wireless communications and the other based on multimedia applications with the common theme that both applications are by nature redundant and that the system can make use of this redundancy to co-design the algorithms and the memory structure.

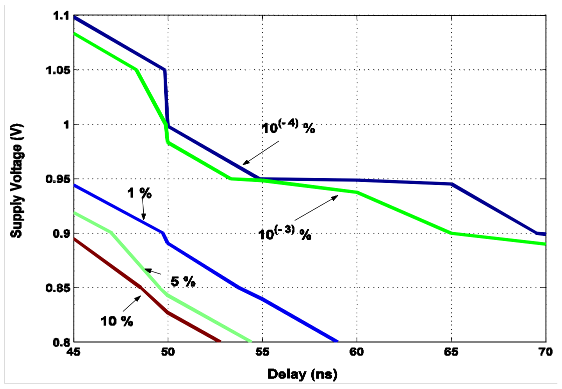

Figure 4 Equi-probable Contours for Write Operation

3. Case Study I: A Wireless System

Wireless systems are a perfect example to showcase the concepts of system redundancy because by nature the data stream is protected against the harsh wireless channel through channel coding. Furthermore, a large portion of on-chip memories are typically dedicated to buffering data. By understanding the error statistics of integrated memories, we can tolerate using less than perfect memories, by inserting or exploiting the controlled redundancy available in the data stream.

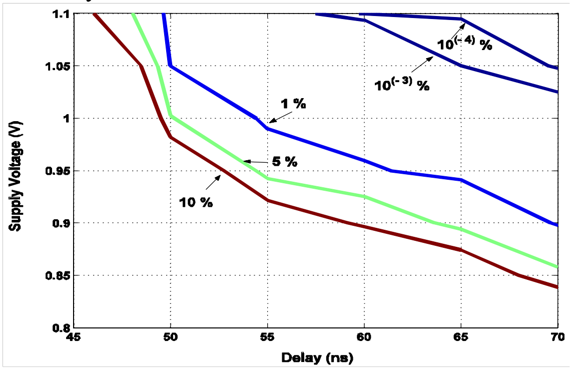

Figure 5 Equi-probable Contours for Read Operation

Figure 6 illustrates a channel coding configuration that is typically encountered in wireless communication applications. For most wireless application this encoding process is performed by either a Viterbi or a Turbo encoder, however a multitude of channel codes exist that address this issue. By coding, it is implied that redundancy is added to the signal to increase it's robustness against impairments in the transmission medium. Turbo codes are currently used in many systems due to their robust performance and the ability to come to within a fraction of a dB of the theoretical ideal solution.

Performance of channel decoders is specified by curves that indicate how much energy should be transmitted to guarantee a specific quality of service (QoS). Coding gain refers to the savings in average bit energy required to achieve a specific BER. For example, it is possible to achieve a coding gain of up to 20 dB utilizing Turbo decoders. This coding gain comes at an expense of an increase in the redundancy inserted in the original data (the code rate) and an increase in the logic complexity of the circuits required to decode the received bit stream.

When discussing memory faults with this background of wireless communication one can immediately extract a synergy in approach. In memory architectures, redundancy, either data redundancy (ECC) or hardware redundancy (BIST/BISR) are used to protect against manufacturing faults, whereas in wireless communication this redundancy is inherent to the method the data stream is communicated. This inherent redundancy can be exploited to allow key memories within an SoC to be defect tolerant. Obviously, not all memories can be defect tolerant, for example, control memories that supervise the operation of the SoC have to be error free, similarly program memory for the processor have to be error free. This is often achieved at the extra cost of increasing the supply voltage (hence power) or memory correction circuitry. However, data buffering memories can be defect tolerant, since the data stored within these memories will be processed eventually by the channel decoder.

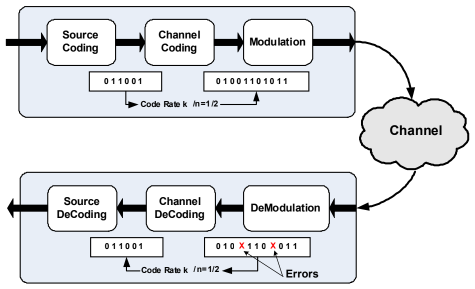

Figure 6 Channel Coding

A. System Setup

To quantify the promise of using system redundancy to compensate for circuit faults a simulation was setup to test different system coding schemes with various error configurations in the memory used to store data samples. The system setup is illustrated in Figure 7. A random bit generator is used to generate data bits that are fed to a channel encoder, the output is then fed into an AWGN channel to simulate a channel. The resulting samples are then scaled and passed to a quantizer. This process mimics the performance of an automatic gain control circuit followed by an analog to digital converter. The quantized samples are then stored into a memory where contents of bit locations can be flipped according to any required statistics (e.g. uniform or Poisson etc.). This part of the simulation is intended to simulate data memory faults. Finally the output is fed into the appropriate decoding scheme and the generated bit stream is compared to the original transmitted stream to generate a final bit error rate measurement for different signal to noise (SNR) ratios. The decoder under test is a rate ½ Turbo decoder with a programmable interleaver memory depth. Note that a larger interleaver depth implies that the decoding algorithm handles a larger data-block at a time. The constraint length of the encoders is 4.

B. Simulation Results

Three simulation sets were performed with 256, 512 and 1024 bits constituting the interleaver memory which is assumed to be error free. The data buffering memory on the other hand is the device under test and is corrupted uniformly with 1% error. The bit width is assumed to be 5 bits. In all cases, random information bits are generated and passed through the system described in Figure 7.

Figure 8 summarizes the simulation results with the following simulation parameters:

a) P=0% memory errors for N=256, 512 and 1024,

b) P=1% errors for N=512 and 1024.

In this simulation, we are comparing the performance of the baseline system with N=256 with other systems where the interleaver depth is increased as a means of protection against memory faults.

It is interesting to note that the configuration with (P=0, N=256) offers the worst performance. In fact by increasing the system redundancy to N=512 and introducing 1 % errors the faulty system performs better in terms of BER. Finally, it is also noteworthy to see that by further increasing the system redundancy to N=1024, The faulty system (P=1%, N=1024) performs almost identical to the ideal system (P=0, N=512).

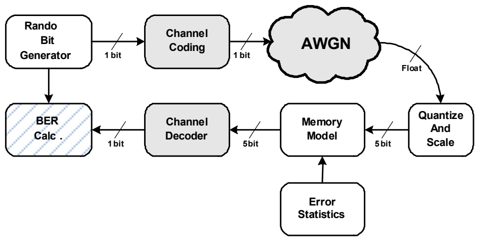

Figure 7 Simulator Setup

These results stress the fact that rather than striving to achieve error free operation, co-operative redundancy utilizes redundancy that is inherently available in the system to mask or minimize the effects of memory errors for specific structures. Co-operative redundancy for wireless communication applications is particularly suited for large data storage memories since eventually all data samples are processed by the channel decoder which utilizes this redundancy to meet a specific SNR or BER metric. This approach leads to significantly larger error tolerance than any technique currently used. This is evident in the fact that the Turbo decoder based system discussed in this paper can easily tolerate 1 % errors and in fact can meet all system specifications (at 10^{-5} BER).

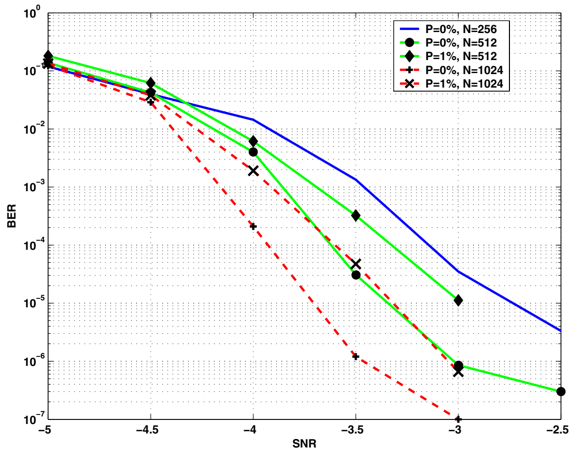

Figure 8 Performance Comparison

4. Case Study II: An H.264 Video Decoder

Multimedia is another perfect example that presents a set of challenging applications targeting SoCs, and happens to be naturally error tolerant. H.264 [6] is emerging as the most promising video standard with higher quality, better compression and many features. Figure 9 illustrates a typical configuration for an H.264 Video codec.

Similar to communication systems, multimedia systems also have inherent error resilience. In such systems, the quality of an output image or sequence of video frames is measured in terms of Peak Signal-to-Noise Ratio (PSNR) which compares the output image(s) to a reference set and computes the PSNR function as:

$$PSNR\,(db) = 10\log_{10}\frac{255^2}{\left\langle (x_i - y_i)^2 \right\rangle}$$

Where x_i and y_i are the pixels at location i of the output and reference images, respectively and $\langle \dots \rangle$ denote the summation over all the image pixel locations. Even in the case of ideal transmission conditions, residual errors occur due to quantization and/or filtering so systems typically have less-than-perfect PSNR values. As a result, standard compliance for key multimedia kernels such as DCT does not require perfect signal recovery as compliance criteria. In order to cope with imperfect signal recovery, some multimedia systems have explicit error resiliency built into the algorithms.

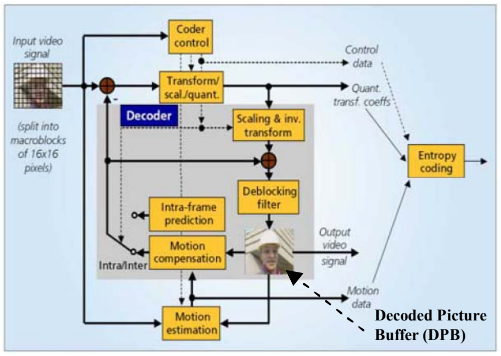

Figure 9 H.264 System

Video compression algorithms utilize two characteristics of image sequences, spatial correlation and temporal correlation. Motion compensation (inter prediction) is the standard way to extract temporal correlation from video and intra prediction uses spatial correlation. The underlying assumption of motion compensation is that scene change is due mainly to object and camera motion and the difference between temporally adjacent pictures is so small that many parts of the current frame can be borrowed from previously decoded frames which are stored in a memory called the decoded picture buffer, or DPB. Inter prediction is the way to create a prediction model from one or more previously encoded video frames or fields. Since DPB stores decoded images, it requires large memory space and can easily be the dominant memory in SoC design of H.264 or other similar standard such as MPEG-2 or MPEG-4 [5] which also require reference picture buffer as large as 16Mb. Therefore, our study is focused on the DPB.

In our experiments [7] we simulated both permanent and transient errors. In this paper, we concentrate on permanent errors only. More specifically, we concentrate on parametric failures described in Section 2 since the impact of those variations can be further amplified by power management. In this case, it is possible to utilize BIST engines to scan the memory upon power-up and build a defect map for target memories [8] (the DPB in this case). This map can be stored either locally or in external memory (and cached in on demand). This defect map permits the identification of locations where errors exist and targets only those pixels stored in these locations for repair.

In DPB, decoded YUV components which represent Luma (brightness) and Chroma (color) of the image are stored. In the 4:2:0 sampling format ('YV12'), U and V each have half the horizontal and vertical resolution of Y. Each YUV component has 8 bit depth (in the simulated H.264 decoder) and a total of 12 bits are used to make one pixel because only one U and V components are necessary for every four Y components.

While we recognize that many error concealment schemes that attempt to overcome transmission errors rely on both temporal and spatial redundancies, the temporal components are more complicated to implement and would require more elaborate schemes but at the cost of more hardware and power consumption. This is certainly an area of future research. Therefore, we limited our scheme to use just adjacent pixels to rebuild defective pixels and several schemes are applied to find best performance in terms of PSNR and compared visually. Those techniques are described in details in [7]. The highest performance techniques were:

(1) *Protecting the 4 Most Significant Bits (MSBs) of memory*: The MSBs of pixel values typically contain the most useful information. Thus, it would make sense to protect the MSB's of each word in the memory. From an implementation point of view protecting selected bits in a word can be accomplished by column-wise banking.

(2) *8-pixel median image filtering:* 8 neighboring pixels are sorted by its values, two median pixels are chosen, then mean of those two median is used to replace the defective pixel. While this technique would increase the hardware and/or performance requirements it would result in only small increase in power consumption since only the defective pixels will be processed (about 0.1mW of power in 0.18µ [7]).

Figure 10 and Figure 11 show the impact of those techniques on the "Foreman" and "News" clips for various bitrates when 3% errors are injected into the DPB to model parametric variations. We can derive two major conclusions from the results shown in Figure 10 and Figure 11. First, that either techniques can significantly

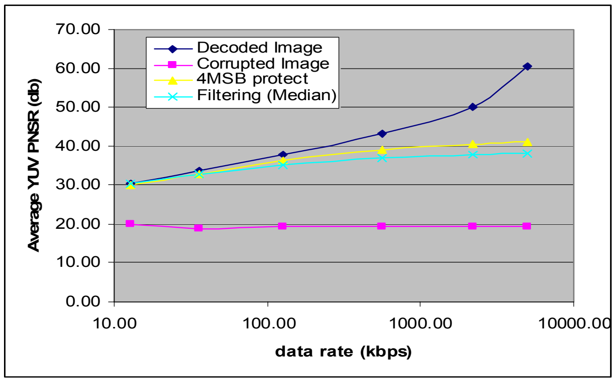

Figure 10 PSNR values for the "News" video clip

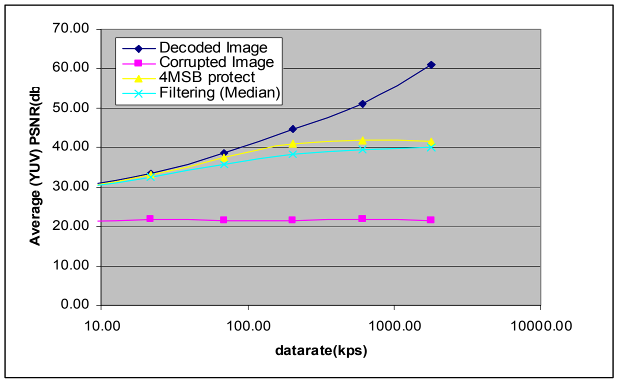

Figure 11 PSNR values for the "Foreman" video clip

improve the PSNR quality of the corrupted output, achieving what is generally accepted as good quality output with PSNR values greater than 30db. Secondly, that the absolute PSNR difference between the different techniques and the uncorrupted image decreases as data rates decrease and more quantization is applied in order to meet those data rates. At 159kbps, for example, the difference between the PSNRs of the uncorrupted video and that of the filtered one is only a few dbs and a largely imperceptible drop in quality is observed.

5. Power savings potential

These results obtained for both case studies (wireless receiver and multimedia decoder) motivate us to consider the expansion of the design space of both systems to encompass more aggressive power/performance points. Referring back to Figure 4 and Figure 5, we note that depending on the application, and target performance, the memory error rate floor can be raised to around 1-10%. In this case, the aggressive design points allow the further reduction of the supply voltage while maintaining the same delay level, or alternatively, achieve lower delays at the same voltage than before, or a combination of both.

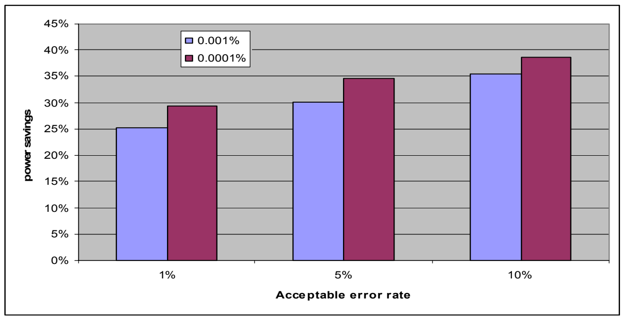

Figure 12. Power savings for different error floor assumptions

Figure 12 shows the expected power savings between non-aggressive and aggressive error floors assuming the same delay targets. These savings can be over 38% when comparing the 10% and 10^{-4}% error floors.

6. Conclusion

This paper proposed a paradigm shift in system level design. The inherent error resilience of many applications and the implementation of system level, manufacturability-aware techniques allows the "absorption" and concealment of relatively large amounts of system memory errors. Experimental results indicate that this error tolerance can be exploited in order to improve the power efficiency of the overall system.

7. References

[1] W. Maly, "Design Methodology for Defect Tolerant Integrated Circuits," Proc. of CICC 1988.
[2] A.J. Bhavnagarwala, X. Tang, J.D. Meindl, " The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability", IEEE JSSC, Vol. 36, #4, April 2001, pp. 658-665
[3] "Berkeley Predictive Technology Model (BPTM)", http://www-device.eecs.berkeley.edu/~ptm/
[4] "Statistical Design and Optimization of SRAM Cell for Yield Enhancement", S. Mukhopadhyay, H. Mahmoodi, and K. Roy, ICCAD 2004
[5] H. Arakida et. al. "A 160mW, 80nA Standby, MPEG-4 Audiovisual LSI with 16Mb Embedded DRAM and a 5GOPS Adaptive Post Filter". Proc. ISSCC 2003. Paper 2.3.
[6] H.264 white paper, http://www.vcodex.com/h264.html
[7] Omitted for blind review
[8] Shoukourian, S.; Vardanian, V.; Zorian, Y.; SoC yield optimization via an embedded-memory test and repair infrastructure. Design & Test of Computers, IEEE Volume 21, Issue 3, May-June 2004 Page(s):200 – 207