On-Chip ECC for Low-Power SRAM Design
Hsin-I Liu (hsil@eecs.berkeley.edu)

Abstract — As the standby supply voltage for the static RAM (SRAM) design scales down for the low-power purpose, the static noise margin of SRAM also decreases. If the supply voltage is below the data retention voltage (DRV), the data need to be checked and corrected before they are sent out of the memory block. In this project, we study the effect of implemented error correction code (ECC) technique to the SRAM design. The performance is evaluated in both model simulation and circuit performance perspective.

I. INTRODUCTION

In order to reduce the leakage current of the SRAM design, the most aggressive way is to reduce the standby supply voltage or to increase the threshold voltage $V_T$. As the supply voltage scales down, the static noise margin (SNM) of SRAM also decreases. Once the supply voltage is below the data retention voltage (DRV), the SNM will go to zero, and then the data stored in the memory cell cannot be retained by the cross-coupled inverters [1]. This defines the lower bound of voltage reduction in the low-power SRAM design.

However, due to the process variation, DRV is different from cell to cell. As a result, it is hard to define a low supply voltage such that all memory cells can hold the data simultaneously. Therefore, applying error correction code (ECC) technique inside of memory circuitry becomes essential. The similar technique has been applied to the memory design for many years to address other issues [2][3], e.g. cosmic-ray injection [4].

In this project, we propose to use the (255, 223) Reed-Solomon code structure to implement the ECC technique in the low-power SRAM design. The design combines the ECC method with conventional redundancy strategy to reduce both the systematic error and random error in the SRAM design. The evaluation of the design will be in both model simulation and circuit performance aspects, comparing the bit error rate, area overhead, and the operating frequency of the ECC circuit.

II. MATHEMATICAL DRV MODEL FOR LOW-POWER SRAM DESIGN

The measurement of DRV variation for the 90nm SRAM design is shown in figure 1. It is easy to observe that there are two major factors causing the distribution of DRV. One is the systematic variation due to the boundary effect, and the other is a random distributed variation. As studied in [5], the random variation of DRV can be modeled as a gamma process, as shown in figure 2. Although the physical explanation of this distribution model is not very clear, we can still apply this model for the simulation, since it is based on real measurement.

According to the analysis from [5], the DRV of SRAM cell $i$ can be modeled as

$$\text{DRV}_i = 60\text{mV} + 10\text{mV} G_i (5, 1)$$

(1)

where $G_i$ is a gamma variable with the parameter of $\Gamma(5, 1)$ scaled by 10mV. The probability density function of $G_i$ can be expressed as

$$g(x) = \frac{1}{5}x^4 e^{-x}$$

(2)

In order to minimize the energy spending on the valid bits, the result in [5] has shown that the optimum operating point is around 155mV, where the fraction of error $\beta$ is about 0.09. At this point, the energy is three to five times reduced. And this point is also the suggested operating point for this project.

Figure 1 The DRV measurement of a 90nm technology 128x256 SRAM [1].
III. ERROR CORRECTION TECHNIQUE

There are many ECC techniques in literature [6]. In the previous researches regarding to SRAM ECC circuit design, Hamming code was always applied [2][3][4], since it introduces least hardware overhead. However, Hamming code is only good for sparse error detection. When it comes to correcting multiple errors inside a single block, multiple Hamming encoders/decoders are applied to the system. Due to the block effect, this duplicated structure cannot correct successive errors inside one sub-block. In addition, according to the analysis in [5], the energy efficiency reaches the asymptotic result with block size around 1000. In order to reach this block size, we choose to implement Reed-Solomon code for the ECC technique.

Reed-Solomon code is a linear block code [6], and is designed for correcting multiple errors within a block of symbols, especially for a large block size. For m-bit sequences, we can assign a Reed-Solomon code with two parameters: the total symbol length n and length of information symbols k, expressed as RS(n,k). As shown in figure 3, there will be extra n-k symbols in a block as the parity check symbols. Reed-Solomon code in sure that each codeword will have the maximum distance, in this case, is equal to (n-k)/2. In the worst case, it can correct t bits in a block of total nm bits; while in the best case, it corrects tm bits within nm bits.

For determining the parameters n and k, there are a few constraints. The n is usually assigned to be $2^m-1$, where m is the length of each symbol. The k is flexible within the range between 0 and n, only the difference between n and k should be an even number to insure the maximum error correction. In our model, since the desired block size is around 1000, which is defined in the cell basis, we can reform the block into a block of 255 symbols, where each symbol contains 8 bits. This will provide a block of 2040 cells. In addition, the error fraction is around 0.09, which implies the minimum distance is greater than 30. Therefore, a RS(255,223) Reed-Solomon code is proposed.

On the other hand, if we consider the complexity of implementing non-binary arithmetic operations, the binary Reed-Solomon codes should be applied. In the binary case, the block length is the length of a row; in this model, the value is around 1000. The main drawback of binary Reed-Solomon codes is the inevitable large latency in the hardware implementation of decoder, which may cause the ECC low-power SRAM not appropriate for existing applications.

IV. PROJECT APPROACHES

In this project, we want to find a reasonable operating point of the ECC technique. Originally we want to design the corresponding Reed-Solomon codec and evaluate its performance. However, as we went through the literature, we discover that the recursive operation in the Reed-Solomon decoding process will introduce a huge latency in the existing Reed-Solomon IP cores such that Reed-Solomon code is not feasible for existing SRAM applications [7][8]. Therefore, the first step to address the ECC implementation problem will be discovering the complexity of Galois Field Multiplier. This will provide the background information of the possibility to apply Reed-Solomon Code as the ECC method in the low-power SRAM. If the latency and complexity are beyond the scale, then maybe the simple ECC method should be the appropriate approach.

As for the hardware implementation, the object will be estimating the hardware consumption of the Galois Field Multiplier with a fixed block size 127×7 bits. After the designing of multiplier in the gate level, the total area and timing of it will be estimated using 90nm STMicroelectronics database and the Synopsis design complier from BWRC.

Nevertheless, the whole low-power SRAM design is an optimization problem, and the ECC circuit overhead is also part of the cost function.
Therefore, the optimal operating point may not be as described in [5]. However, this project can provide a preliminary estimation of the ECC performance.

REFERENCE


