Unequal-error-protection codes in SRAMs for mobile multimedia applications

Xuebei Yang[†] and Kartik Mohanram[‡]
[†]Department of Electrical and Computer Engineering, Rice University, Houston
[‡]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh
xbyang@rice.edu kartik.mohanram@gmail.com

Abstract

In this paper, we introduce unequal-error-protection error correcting codes (UEPECCs) to improve SRAM reliability at low supply voltages for mobile multimedia applications. The fundamental premise for our work is that in multimedia applications, different bits in the same SRAM word are usually not equally significant, and hence deserve different protection levels. The key innovation in our work includes (i) a novel metric, word mean squared error, to measure the reliability of a SRAM word when different bits are not equally significant and (ii) an optimization algorithm based on dynamic programming to construct the UEPECC that assigns different protection levels to bits according to their significance. The advantage of the UEPECC over the traditional equal-error-protection ECC is demonstrated using two representative multimedia applications. For the same area, power, and encoding/decoding latency, SRAMs with UEPECC increase the peak signal-to-noise ratio by 8 dB in image processing and incur 60% less errors on average in optical flow (motion vector) computation.

1. Introduction

With the rapid growth in the popularity of powerful mobile devices such as smart-phones and handheld game consoles, the demand for mobile multimedia applications is increasing exponentially. As these mobile devices are usually battery-powered, mobile multimedia applications typically have tight power budgets. The static random access memory (SRAM), which is widely used in multimedia systems as cache memories and register files, has been observed to contribute significantly to the total system power consumption [1]. Since the reduction in supply voltage reduces both dynamic and static power consumption of SRAMs [2, 3], it is desirable that SRAMs operate at low supply voltages. Unfortunately, as the supply voltage reduces, the effect of process variations increases in severity, resulting in an exponential increase in the SRAM cell failure probability [3]. This imposes limits on supply voltage scaling and the reduction in overall power consumption.

To improve SRAM reliability at low supply voltages, novel cells and transistors optimized for low-voltage operation have been proposed, including the 8T SRAM cell [4], the 10T SRAM cell [5], and the tunneling transistor [6]. Recently, error correcting codes (ECCs) have received strong interest for SRAM reliability improvement [7–10]. Since ECCs introduce storage and area overhead, while simply scaling up the cell size also reduces the failure probability, the application of stronger ECC in SRAMs was not popular as a means to improve SRAM reliability. However, it was shown in [9] that using stronger ECC can be more effective than scaling up the cell sizes at low supply voltages, because although smaller SRAM cells have higher failure probability, stronger ECCs can correct more cell errors and improve the overall reliability. As a result, the application of stronger ECCs such as double-error or triple-error correcting ECCs for low-power SRAMs is now being actively investigated [8–10]. Furthermore, a reconfigurable ECC architecture was recently proposed in [10], which allows for application-dependent in-field adjustment of the supply voltage, ECC strength, and storage overhead to reduce over-design and/or waste.

In this work, however, we note that all the previously studied ECCs in SRAMs belong to the class of equal-error-protection ECCs (EEPECCs). EEPECCs assume that all the protected cells have equal significance and hence protect them equally. However, in multimedia applications, this assumption is usually not valid. For example, the human eye is far more sensitive to higher order bits of a pixel in image applications [11]. Furthermore, in multimedia applications, zero failure probability is usually not necessary [11–13]. Therefore, in SRAMs reserved for multimedia data storage [11,14], instead of using EEPECC to protect all bits equally, it is intuitive to assign higher/lower protection levels to the more/less significant bits, respectively, because even if the less significant bits fail, they have limited impact on the quality of the multimedia applications. Motivated by this observation, we propose to apply the unequal-error-protection ECC (UEPECC) to SRAMs that allows different protection levels for different bits in multimedia applications. Toward this goal, we make the following two contributions.

First, we propose an alternate metric for SRAM word reliability. Traditionally, the reliability of a SRAM word has been estimated by word error probability [15]. However, in multimedia applications where different bits in a SRAM word are not equally significant, word error probability may be conservative. For such applications, we propose the word mean squared error (WMSE) metric to measure SRAM word reliability. WMSE is motivated by the peak signal-to-noise ratio (PSNR) [16], which is a widely used metric in multimedia applications. However, unlike PSNR that is a general metric and does not take the SRAM implementation into account, WMSE can be analytically expressed as a function of SRAM cell failure probability, thus enabling quantitative measurement of the SRAM word reliability in multimedia applications.

Second, using WMSE as a design metric, we propose an optimization algorithm based on dynamic programming to construct the UEPECC for the SRAM word. Whereas various good UEPECCs have been previously proposed in the coding community [17], the encoding/decoding is usually complicated and is prohibitive for SRAMs. Our proposed UEPECC, however, is constructed using the direct sum method [18] and is based on the simple repetition code and the orthogonal Latin square code (OLSC) [19], both of which have simple encoding/decoding process with minimum impact on SRAM performance. For the same area, power, and encoding/decoding latency, SRAMs with UEPECC achieve 8 dB increase of PSNR in image processing and incur 60% less errors on average in optical flow (motion vector) computation.

This paper is organized as follows. Section 2 presents the background on SRAM reliability and ECCs. Section 3 introduces the WMSE metric for SRAM word reliability. Section 4 presents the optimization algorithm for UEPECC construction. Section 5 presents results and Section 6 is a conclusion.

This research was supported by NSF CAREER Award CCF-0746850 and a gift from Fujitsu Laboratories of America.

978-1-4577-1400-9/11/$26.00 ©2011 IEEE

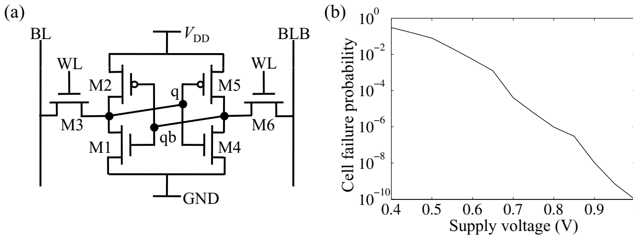


Figure 1: (a) The structure of a 6T SRAM cell. (b) The cell failure probability at different supply voltages.

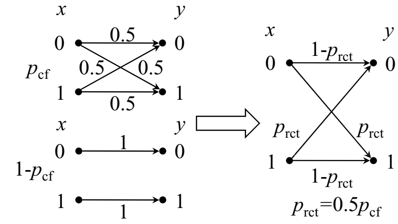


Figure 2: Binary symmetric failure model for the SRAM cell with cell failure probability p_{cf}

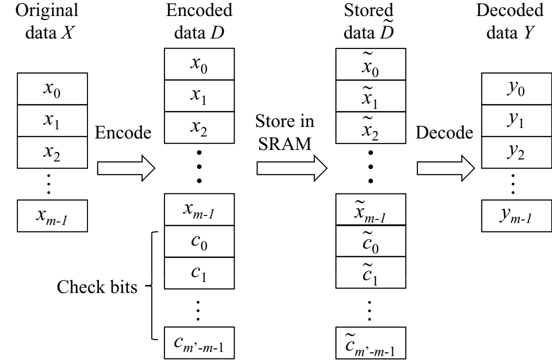


Figure 3: Block diagram for SRAM word with ECC encoding/decoding

2. Background

In this section, we first present the background for SRAM reliability. Although we focus on the traditional 6T SRAM cell in this work, our analysis is not restricted to the 6T structure and can be applied to other SRAM cell structures as well. We then introduce ECC as a widely used technique to improve SRAM reliability. We finally motivate the application of the unequal-error-protection ECC (UEPECC) in multimedia applications.

2.1 SRAM

In Figure 1(a), we present the traditional 6T SRAM cell, which is composed of a pair of cross-coupled inverters (M1, M2, M4, and M5) and two access transistors (M3 and M6). Due to process variations during fabrication and environmental impact during operation, an SRAM cell may fail to function correctly. As extensively studied in previous publications [8, 12, 20], the read upset — defined as the read operation that causes the value stored at q and qb to flip — is the dominant SRAM cell failure mechanism at low supply voltages. Since this work focuses on SRAMs operating at low supply voltages, we approximate the cell failure probability, denoted as p_{cf}, by the probability of read upset in this paper. Note that since one SRAM cell stores one bit of data, we will not distinguish between cell and bit in many cases, for example, bit failure and cell failure will be used interchangeably. In Figure 1(b) we present the simulated p_{cf} using the 45nm PTM model [21] and the technique proposed in [22]. It is observed that as the supply voltage decreases, p_{cf} increases exponentially. As the supply voltage reaches 0.4 V, p_{cf} is on the order of 10^{-1}.

In order to smooth the discussion in this work, we further propose to model the SRAM cell as a binary symmetric channel shown in Figure 2. We denote x and y as the original bit and the corresponding stored bit, respectively. If the cell has a failure probability of p_{cf}, $y = x$ with probability 1-p_{cf}, while $y = 0$ or 1 with equal probability regardless of x with probability p_{cf}. If we define the raw cell transition probability $p_{rct} = 0.5p_{cf}$, the combined effect is that $\mathbf{P}(y = 0|x = 1) = \mathbf{P}(y = 1|x = 0) = p_{rct}$, and $\mathbf{P}(y = 1|x = 1) = \mathbf{P}(y = 0|x = 0) = 1 - p_{rct}$. Hence, it is clear that p_{rct} (instead of p_{cf}) gives the probability that the original bit and the stored bit in an SRAM cell differ. If a cell has a failure probability p_{cf} of 1, the corresponding $p_{rct} = 0.5$ and this SRAM cell will store 0 or 1 with 50% probability regardless of the original bit. This is the worst case for an SRAM cell. Note that in this model, we have assumed a symmetric failure probability. This assumption is valid if the SRAM cell is symmetric, such as the traditional 6T structure considered in this work.

We next consider the organization of an SRAM word. A SRAM word consists of multiple SRAM cells that share the same word line, and the word reliability is traditionally measured using the word error probability p_{wr} [15]. For an SRAM word without any error correction, all the cells in the word store data, so p_{wr} is the probability that the stored bit is different from the original bit in at least one of the cells. If all the SRAM cells in the SRAM word are designed equally and have the same p_{rct}, p_{wr} of an SRAM word consisting of m SRAM cells can be expressed as

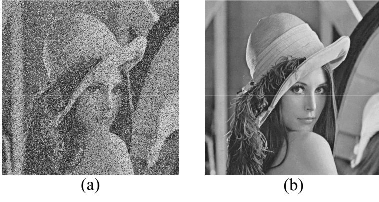
$$p_{wr} = 1 - (1 - p_{rct})^m. \quad (1)$$

2.2 Error correcting codes

Error correcting codes (ECCs) [15] have been widely used in SRAMs for years to improve the SRAM word reliability. In Figure 3, we present the block diagram for an SRAM word equipped with ECC. The original data X is of length m. The encoded codeword D is of length m' with m bits of original data and $m' - m$ check bits. The codeword D is then stored in the SRAM word. In the presence of cell failures, let $\widetilde{D}$ represent the codeword that is actually stored in the SRAM word instead of D. On a read, $\widetilde{D}$ is finally decoded as Y, which is considered the representation of X. When ECC is used, the transition of a cell will be influenced by other cells in the same word. As a result, the probability that cell i has a cell transition, i.e., $\mathbf{P}(x_i \neq y_i)$ is usually different from p_{rct}, which does not take into account the effect of other cells. In this work, the probability for cell transition is denoted as p_{ct}. Furthermore, in this work we will frequently use the term "protection level". Clearly, for the same p_{rct}, p_{ct} of a bit is lower if the bit receives a higher protection level.

Traditionally, all the proposed ECCs for SRAMs fall into the category of equal-error-protection ECCs (EEPECCs), i.e., all the m bits of data are considered equally significant and receive the same protection level. Formally, an (m', m, t) EEPECC maps the data of length m to the codeword of length m', and it can correct up to t bit errors in the m'-bit codeword. For the same number of data bits m, larger codeword length m' (and hence larger area) is required for larger t. The p_{wr} of an SRAM word equipped with an (m', m, t) EEPECC is approximately evaluated as [18]

$$p_{wr} = \mathbf{P}(X \neq Y) \approx 1 - \sum_{i=0}^{t} \binom{m'}{i} p_{rct}^i (1 - p_{rct})^{m'-i}. \quad (2)$$



(a) (b)

Figure 4: (a) The stored image where p_{rct} of the four high-order bits is 0.2 while p_{rct} of the four low-order bits is 0. (b) The stored image where p_{rct} of the four high-order bits is 0 while p_{rct} of the four low-order bits is 0.2.

Traditionally, EEPECC with $t = 1$ was applied to SRAM words to mitigate the effect of soft errors. Since stronger ECCs with larger t require more storage overhead, while simply scaling up the cell size also reduces the cell failure probability, the application of stronger ECCs was not popular as a means to improve SRAM reliability. However, it has been recently reported [9] that as the p_{rct} increases with technology and voltage scaling, using stronger ECC can be more effective than scaling up the cell size at low supply voltages. This observation is based on the fact that whereas smaller SRAM cells have higher failure probability, stronger ECCs can correct more cell errors and improve the overall reliability. As a result, SRAM words equipped with stronger ECCs are now being intensively studied [8–10].

In this work, however, instead of investigating ECCs with larger t, we argue that the traditional EEPECC does not provide the best SRAM word reliability in multimedia applications. The fundamental premise of our argument is that different bits in the same SRAM word can have different significance in multimedia applications, and they deserve different protection levels. We present an illustrative example in Figure 4. Two SRAMs are configured to store a test image consisting of 512×512 pixels, where each pixel is represented by a 8-bit binary number, ranging from 0 to 255. In Figure 4(a), each pixel is stored in a 8-bit SRAM word where the p_{rct} of the four higher-order bits is 0.2, while the p_{rct} of the remaining bits is 0. In Figure 4(b), each pixel is stored in a 8-bit SRAM word where the p_{rct} of the four higher-order bits is 0, while the p_{rct} of the remaining bits is 0.2. Obviously, Figure 4(b) is of higher quality than Figure 4(a). This demonstrates that higher-order bits of a pixel are more significant to image quality in comparison to the lower-order bits. Furthermore, in multimedia applications, zero failure probability is usually not necessary [13]. Therefore, it is intuitive to assign higher/lower protection levels to the higher/lower-order bits, respectively, because even if the lower-order bits fail, they have limited impact on the quality of the multimedia applications.

Motivated by the previous observations, we propose to apply the unequal-error-protection ECC (UEPECC) to SRAMs in multimedia applications in this work. UEPECC allows the protected data to be partitioned into blocks of different significance and assigns different protection levels accordingly, making it an ideal substitute for the EEPECC in SRAMs for multimedia applications. However, while it is clear that higher-order bits should be assigned higher protection levels, the two key questions to be answered are: What is the optimal protection level for each bit and how is the optimal UEPECC constructed?

Since Masnick and Wolf proposed UEPECC in 1967 [23], various studies have tried to answer these questions and to build "good" UEPECCs for different applications [17]. However, these studies were mainly concerned with minimizing the codeword length m' at the expense of encoding/decoding complexity. In SRAM applications, complex encoding and especially complex decoding processes will impose significant performance penalty on the whole system. Therefore, these "good" UEPECCs are of limited use for SRAM applications. In this work, instead of using these existing "good" UEPECCs, we propose to construct an UEPECC in the following manner. The data is first partitioned into b blocks, where each block has m_i bits of data, $i = 0, 1 ... b-1$. For each block i, a corresponding t_i and codeword length m'_i are assigned, and a (m'_i, m_i, t_i) EEPECC is constructed separately for each block. Note that $t_i = t_j$ does not guarantee the same protection level for bits in the ith and jth block, as both m_i and m_j and the specific EEPECC scheme for these two blocks can be different. This UEPECC construction approach is referred to as the direct sum method [18]. Although more check bits may be required in comparison with the "good" UEPECC, this approach always enables simple encoding and decoding.

3. SRAM word reliability measurement

In the previous section, we have motivated the application of the UEPECC in SRAM words for multimedia applications. However, in order to construct a good UEPECC, the first question is how to evaluate the effectiveness of a UEPECC? Specifically, how do we estimate the reliability of the SRAM word equipped with a UEPECC when different bits are not equally significant? Traditionally, the SRAM word reliability is measured by the word error probability p_{wr}, where all bits contribute equally to the SRAM word as shown in Equations 1 and 2. Obviously, p_{wr} is not suitable to measure the reliability of an SRAM word where different cells are not equally significant. For example, although the SRAM configuration in Figures 4(a) and 4(b) have the same p_{wr}, the image quality differs greatly. In this section, we propose an alternative metric to estimate SRAM word reliability when different bits in the SRAM word have different significance. Our proposed metric, termed the word mean squared error (WMSE), can be analytically evaluated using the cell transition probability p_{ct}. WMSE is motivated by the peak signal-to-noise ratio (PSNR) [16], which is a widely used metric for measuring audio, image, and video quality. However, PSNR is a general metric and does not take the SRAM implementation into account. This weakness motivates the proposed WMSE, which can quantitatively estimate the reliability of an SRAM word through an analytical expression.

We first motivate the proposed WMSE by introducing PSNR. Without loss of generality, consider the application of image processing. Assume A is the original image consisting of $M \times N$ pixels and B is another image of equal size that is the approximation of A. PSNR is defined as

$$\text{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (A(i,j) - B(i,j))^2 \quad (3)$$

$$\text{PSNR} = 20 \cdot \log_{10}(\text{MAX}/\sqrt{\text{MSE}}) \quad (4)$$

where $A(i,j)$ and $B(i,j)$ are the values of different pixels of A and B, MSE is the mean squared error between A and B, and MAX is the maximum pixel value. For an 8-bit grayscale image, MAX is 255. Consider the case where n pixels are always grouped together. Without loss of generality, if the grouped n pixels are always in the same row, the MSE can be further expressed as

$$G(i,j) = 1/n \sum_{k=0}^{n-1} (A(i,j+k) - B(i,j+k))^2 \quad (5)$$

$$\text{MSE} = n/MN \sum_{i=0}^{M-1} \sum_{j=0}^{N/n-1} G(i,jn) \quad (6)$$

Assume that the $G(i,j)$ are independent and identically distributed (i.i.d.). Since $M \times N$ is usually large in real applications, for small

n, the law of large numbers gives

$$\text{MSE} \approx \mathbf{E}[G], \text{ where } \mathbf{E}[G] \text{ is the expected value of } G$$
$$\text{PSNR} \approx 20 \cdot \log_{10}(\text{MAX}/\sqrt{\mathbf{E}[G]}) \quad (7)$$

Note that $\mathbf{E}[G]$ can give an approximation value of PSNR by the above expression.

Motivated by the previous analysis, we now formally define the word mean squared error (WMSE) metric for SRAM reliability. We consider an SRAM word consisting of m' cells, where m cells are data bit cells. The SRAM word stores n data sets, where each data set has the data length of $l = m/n$ bits. The stored data set is denoted as $Y_1, \ldots, Y_n$, while the original data set is $X_1, \ldots, X_n$. Note that we do not make the assumption that only one data set is stored in one SRAM word, as this condition is not satisfied in various cases [15]. For example, consider an SRAM word consisting of 22 cells where 16 cells store two 8-bit grayscale pixels and 6 cells store the check bits. In this case, $m' = 22$, $m = 16$, $n = 2$, and $l = 8$. WMSE is defined as

$$\text{WMSE} = \frac{1}{n} \cdot \mathbf{E}\left[\sum_{i=0}^{n-1}(X_i - Y_i)^2\right] = \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{E}\left[(X_i - Y_i)^2\right]$$

Note that the equation above is highly similar to Equation 5. Indeed, consider the case when A is the original image and B is the image stored in the SRAM. Specifically, each pixel of A has l bits and n pixels are stored in one SRAM word. Due to the probability of SRAM cell failure, B is an approximation of A. If each pixel of A is i.i.d. and all the SRAM words are identically constructed, WMSE must be i.i.d. As a result, PSNR between A and B can be approximated through WMSE as in Equation 7, substituting G with WMSE. Note that although we illustrate the physical meaning of WMSE using an image application as example, WMSE can be applied to estimate SRAM word reliability in other SRAM multimedia applications as well.

We next derive the analytical expression for WMSE. Without loss of generality, we assume that X_i is an integer between [0,$2^l - 1$] and follows a uniform distribution. x_i^k and y_i^k ($k = 0, \ldots, l-1$) are the l bits of data X_i and Y_i, respectively. Therefore, X_i and Y_i can be expressed as $X_i = \sum_{k=0}^{l-1} 2^k x_i^k$ and $Y_i = \sum_{k=0}^{l-1} 2^k y_i^k$. We also denote $\epsilon_i^k = x_i^k - y_i^k$ and $p_{\text{ct},i}^k$ as the cell transition probability of the kth bit of the ith data set. As discussed in the previous section, p_{ct} may be different from p_{rct} when the SRAM word is equipped with ECC. However, the model of binary symmetric channel in Figure 2 is still applicable, except that the transition probability is p_{ct} instead of p_{rct}. Under these conditions,

$$\begin{aligned}\text{WMSE} &= \frac{1}{n} \cdot \sum_{i=0}^{n-1}\mathbf{E}\left[\left(\sum_{k=0}^{l-1} 2^k(x_i^k - y_i^k)\right)^2\right] \\ &= \frac{1}{n} \cdot \sum_{i=0}^{n-1}\mathbf{E}\left[\left(\sum_{k=0}^{l-1} 2^k \epsilon_i^k\right)^2\right] \\ &= \frac{1}{n} \cdot \sum_{i=0}^{n-1}\mathbf{E}\left[\sum_{k=0}^{l-1} 4^k(\epsilon_i^k)^2 + \sum_{j=0}^{l-1}\sum_{k=0}^{k\neq j} 2^{k+j}\epsilon_i^k\epsilon_i^j\right] \quad (8)\end{aligned}$$

Note that $\epsilon_i^k = x_i^k - y_i^k$ can be 0, -1, or 1. Since X_i follows a uniform distribution between [0,$2^l - 1$], x_i^k has equal probability to be 0 and 1. Therefore, from the model for the binary symmetric channel in Figure 2, we have

$$\mathbf{P}(\epsilon_i^k = 0) = 1 - p_{\text{ct},i}^k \text{ and } \mathbf{P}(\epsilon_i^k = 1) = \mathbf{P}(\epsilon_i^k = -1) = 0.5p_{\text{ct},i}^k$$

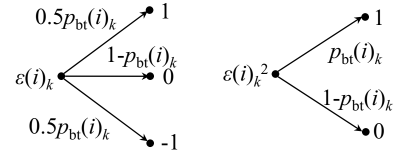


Figure 5: The distribution of ϵ_i^k and $(\epsilon_i^k)^2$

Clearly, $(\epsilon_i^k)^2$ takes the value between 0 and 1, with

$$\mathbf{P}((\epsilon_i^k)^2 = 0) = 1 - p_{\text{ct},i}^k \text{ and } \mathbf{P}((\epsilon_i^k)^2 = 1) = p_{\text{ct},i}^k$$

The distribution of ϵ_i^k and $(\epsilon_i^k)^2$ is presented in Figure 5. From the above observations, we conclude that

$$\mathbf{E}[(\epsilon_i^k)^2] = p_{\text{ct},i}^k.$$

For $\mathbf{E}[\epsilon_i^j\epsilon_i^k]$, since the application of ECC makes ϵ_i^k and ϵ_i^j ($k \neq j$) correlated, we cannot assume $\mathbf{P}(\epsilon_i^j|\epsilon_i^k) = \mathbf{P}(\epsilon_i^j)$. However, from the symmetric modeling of the channel where $\mathbf{P}(\epsilon_i^j = 1) = \mathbf{P}(\epsilon_i^j = -1)$, it is reasonable to assume that $\mathbf{P}(\epsilon_i^j = 1|\epsilon_i^k = 1) = \mathbf{P}(\epsilon_i^j = -1|\epsilon_i^k = 1)$ and $\mathbf{P}(\epsilon_i^j = 1|\epsilon_i^k = -1) = \mathbf{P}(\epsilon_i^j = -1|\epsilon_i^k = -1)$. Therefore, we have

$$\begin{aligned}\mathbf{E}[\epsilon_i^j\epsilon_i^k] =& -1 \cdot \Big(\mathbf{P}(\epsilon_i^j = 1, \epsilon_i^k = -1) + \mathbf{P}(\epsilon_i^j = -1, \epsilon_i^k = 1)\Big) \\ &+ 1 \cdot \Big(\mathbf{P}(\epsilon_i^j = -1, \epsilon_i^k = -1) + \mathbf{P}(\epsilon_i^j = 1, \epsilon_i^k = 1)\Big) \\ =& \Big(\mathbf{P}(\epsilon_i^j = -1|\epsilon_i^k = -1) - \mathbf{P}(\epsilon_i^j = 1|\epsilon_i^k = -1)\Big)\mathbf{P}(\epsilon_i^k = -1) \\ &+ \Big(\mathbf{P}(\epsilon_i^j = 1|\epsilon_i^k = 1) - \mathbf{P}(\epsilon_i^j = -1|\epsilon_i^k = 1)\Big)\mathbf{P}(\epsilon_i^k = 1) \\ =& \, 0\end{aligned}$$

Given $\mathbf{E}[(\epsilon_i^k)^2]$ and $\mathbf{E}[\epsilon_i^j\epsilon_i^k]$, the WMSE is finally derived as

$$\text{WMSE} = (1/n)\sum\nolimits_{i=0}^{n-1}\sum\nolimits_{k=0}^{l-1}\left(4^k p_{\text{ct},i}^k\right) \quad (9)$$

If we have $p_{\text{ct},i}^k = p_{\text{ct},j}^k$ for any k and any $i \neq j$, i.e., the corresponding bits for different data set in the same word have the same transition probability, the above expression can be simplified to

$$\text{WMSE} = \sum\nolimits_{k=0}^{l-1}\left(4^k p_{\text{ct},i}^k\right) \quad (10)$$

The above expression implies that minimizing WMSE can be achieved by minimizing the cell transition probability. Moreover, higher-order cells contribute more to the WMSE. Interestingly, although derived from two different perspectives, the proposed WMSE is similar to the upper bound on the mutual information between the original data and the stored data from an information theoretic perspective, as proposed in [13].

4. UEPECC

In Section 2 we have briefly introduced the construction flow for our proposed UEPECC: the data bits are first partitioned into several blocks, and EEPECC is then applied to each block separately. In this section we will present the construction flow in more detail. We first introduce the EEPECC that is applied to different data blocks in this work, and then present an optimization algorithm based on dynamic programming for the construction of UEPECC, using the proposed WMSE as the design objective.

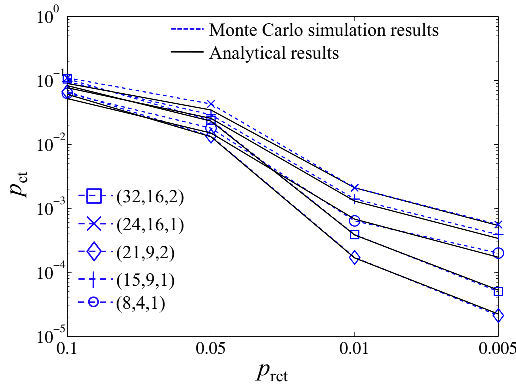


Figure 6: Comparison between analytical expression for p_{ct} of the OLSC derived in Equation 12 and p_{ct} of the OLSC obtained from Monte Carlo simulations

4.1 Adopted EEPECC

In our proposed UEPECC, each block can be equipped with one of two types of EEPECC, the repetition code or the orthogonal Latin square code (OLSC) [19]. The repetition and the OLSC codes are chosen because they are both one-step majority decodable. As one-step majority decoding is the fastest parallel decoding method [19], these codes incur less performance penalty than the traditional Hamming code and BCH code, although they require more check bits to achieve the same error correction ability t. Clearly, this is a tradeoff between area and performance. As will be presented in the optimization framework, if the area is of primary concern, our UEPECC construction can also be based on other EEPECCs like the BCH code. Between the repetition code and the OLSC, the former has low area efficiency but no design limitations, whereas the latter has comparatively higher area efficiency with the limitation in the maximum error correction ability t for fixed number of data bits. Therefore, they are complementary approaches that can enrich UEPECC construction.

Repetition code: The repetition code is a coding scheme that repeats the data bits to achieve high reliability, whose encoding and decoding latency is the delay of 0 and 1 level of XOR gate, respectively [8, 24]. Specifically, for a (m', m, t) repetition code, each bit of m is repeated $2t$ times and hence $m' = (2t+1)m$. Under this condition, each data bit x_i now has $2t+1$ copies of its value, and the decoder will declare the corresponding value of y_i as the majority value among the $2t+1$ bits. Clearly, a decoding error for x_i, i.e., $y_i \neq x_i$ will occur if there are more than t transitions among the $2t+1$ copies. Therefore,

$$p_{ct} = 1 - \sum_{k=0}^{t} \binom{(2t+1)}{k} p_{rct}^{k} (1-p_{rct})^{2t+1-k}. \tag{11}$$

Orthogonal Latin square code (OLSC): OLSC was proposed by Hsiao *et al.* in 1970 [19]. Similar to the repetition code, the OLSC is also based on the principle of majority voting. However, instead of simply generating $2t$ copies of the original bit as the check bits, OLSC encodes the orthogonal group of data bits to form the check bits. For brevity, we refer the reader to [19] for the details of OLSC construction. In order to construct a t-error correcting OLSC for $m = a^2$ data bits, the number of required check bits is $2ta$. However, whereas t can be arbitrarily large as long as there are more check bits ($m' = (2t+1)m$) for the repetition code, the maximum t for the OLSC code is generally less than or equal to $(a+1)/2$ [19]. The encoding and decoding latency of the OLSC, as discussed in [8], are the delay of $\lceil \log_2 \sqrt{m} \rceil$ and $\lceil \log_2 \sqrt{m} \rceil + 2$ levels of 2-input XOR gates, respectively, where $\lceil \rceil$ is the ceiling function. The accurate cell transition probability p_{ct} for the OLSC is difficult to derive and there is currently no analytical expression for p_{ct} of the OLSC. However, motivated by the approximation of p_{ct} for the traditional Hamming code [25], p_{ct} for a (m', m, t) OLSC is approximated by the following expression in this work:

$$p_{ct} \approx p_{rct} \left(1 - \sum_{k=0}^{t-1} \binom{(m'-1)}{k} p_{rct}^{k} (1-p_{rct})^{m'-k} \right) \tag{12}$$

As shown in Figure 6, the p_{ct} derived from the analytical approximation in Equation 12 matches well with the p_{ct} obtained from Monte Carlo simulations.

4.2 UEPECC construction

We consider an SRAM word consisting of at most m' cells, where m cells are data cells. The SRAM word stores n data sets, where each data set has a data length of $l = m/n$ bits. In our UEPECC construction, the corresponding bits for different data sets always receive the same protection, i.e., they are always grouped in the same block. We also require that the order of the bits has to be continuous in a block. For example, we can group the bits of the highest, the second highest, and the third highest order in the same block, but we cannot only group the bits of the highest and the third highest order in the same block because the order of the bits is not continuous. Under such conditions, the construction of the UEPECC can be described by the following optimization problem

$$\min_{(b,m_0,\ldots,m_{b-1},m'_0,\ldots,m'_{b-1})} \left(\text{WMSE} = \sum_{k=0}^{l-1} \left(4^k p_{ct,i}^k \right) \right)$$
$$\text{s.t.} \left\{ \begin{array}{c} b \leq b_{max}, \sum_{k=0}^{b-1} m_k = m, \sum_{k=0}^{b-1} m'_k \leq m', \text{ and} \\ p_{ct} \text{ is evaluated using Equations 11 and 12.} \end{array} \right\} \tag{13}$$

Here b_{max} is the maximum allowed number of blocks. Clearly, if $b_{max} = 1$, our proposed UEPECC is EEPECC. b is the number of blocks that the data is partitioned into; m_i and m'_i ($i = 0, 1 \ldots b-1$) are the number of data bits and the codeword length of each block, respectively. The p_{ct} of all the bits in the same block is evaluated according to Equations 11 and 12 depending on which EEPECC we use for the particular block.

In this work, the optimization formulation for UEPECC construction is solved using the dynamic programming algorithm in 1. We first construct a table $T[C, L, B]$ to store the best WMSE that can be achieved for a data piece having maximum allowed check bit length C, nL bits of data, and maximum allowed block number B. The evaluation of $T[C, L, B]$ is accomplished by considering three ECC schemes: (i) we do not partition the data set and apply the repetition code to it, (ii) we do not partition the data set and apply the OLSC to it, and (iii) we partition the data set into two blocks and look at the best ECC scheme for each block separately. The ECC scheme that presents the best WMSE is chosen and the best WMSE is written to $T[C, L, B]$. Note that for the third ECC scheme, all possible partitions will be considered, including different check bit length C, different bits of data L, and different maximum allowed block B for each partitioned block.

4.3 UEPECC construction study

In this subsection we study the construction of the proposed UEPECC under various design conditions. We consider an SRAM word having $m = 16$, $n = 2$, and $l = 8$. By default, $p_{rct} = 0.01$, $b_{max} = 8$, and $m' = 32$, which means we can have a maximum of 16 check bits with up to 100% area overhead.

We first study the effect of b_{max} on the construction of the UEPECC using Figure 7. Intuitively, a UEPECC with a larger maximum al-

Algorithm 1: UEPECC construction$(m, n, l, m', b_{\max})$

input : m data bits, n data sets, data set length l, maximum allowed codeword length m', and maximum allowed block number $b_{\max}$
output : WMSE

Initialize table T of size $(m' - m + 1) \times l \times b_{\max}$;
/* $T[C, L, B]$ stores the best WMSE that can be achieved for a data piece having maximum allowed check bit length C, nL bits of data, and maximum allowed block number B.*/

Call function $f(m' - m, l, b_{\max})$;
/* $f(C, L, B)$ returns the value of $T[C, L, B]$ */

$f(C, L, B)$:
if $T[C, L, B]$ **then**
  return $T[C, L, B]$;
else
  Evaluate $t_{\max}$ for repetition code for C, L.
  Apply $t_{\max}$-error correcting repetition code to the whole nL data bits.
  Evaluate p_{ct} and the corresponding WMSE.
  $T[C, L, B] =$ WMSE.
  Evaluate $t_{\max}$ for OLSC for C, L.
  Apply $t_{\max}$-error correcting OLSC to the whole nL data bits.
  Evaluate p_{ct} and the corresponding WMSE.
  $T[C, L, B] = \min(T[C, L, B], \text{WMSE})$.
  foreach $i = 0, \ldots, C, j = 1, \ldots, L - 1, k = 1, \ldots, B - 1$ **do**
    WMSE1 = $f(i, j, k)$
    WMSE2 = $f(C - i, L - j, B - k)$
    WMSE = 4^{L-j} ×WMSE1+WMSE2
    $T[C, L, B]$ = min($T[C, L, B]$, WMSE)
  return $T[C, L, B]$;

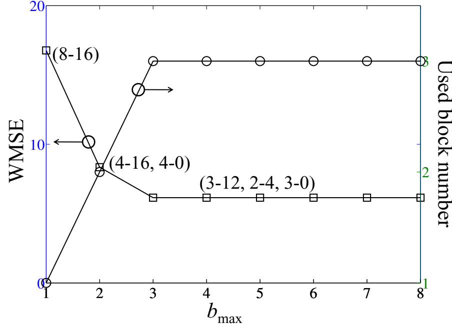


Figure 7: WMSE, number of used blocks, and UEPECC configuration for different $b_{\max}$. The numbers in parentheses represent the partitioning strategy for the SRAM word, referring to ($\frac{m_0}{n}$-c_0, ..., $\frac{m_{b-1}}{n}$-c_{b-1}), where c_i is the number of check bits for block i.

lowed block number $b_{\max}$ will always yield a WMSE no worse than the UEPECC with a smaller $b_{\max}$. However, the used number of blocks b does not necessarily increase with $b_{\max}$, but saturates to a fixed number. As shown in the figure, when $b_{\max} = 1$, only one block is allowed and the WMSE is the highest. When $b_{\max} = 2$, the 4 higher-order bits of the two data sets consume all the 16 check bits, while the 4 lower-order bits of the two data sets do not receive any check bit. Under such conditions, the WMSE is modest. When $b_{\max} = 3$, the 3 higher-order bits receive 12 check bits, the 2 medium-order bits receive 4 check bits, while the 3 lower-order bits do not receive any check bits. The WMSE is the lowest under such conditions. As $b_{\max}$ becomes larger than 3, the SRAM word is still partitioned into three blocks. This indicates that a larger $b_{\max}$ cannot always result in better WMSE, and that there is an optimal number of blocks for the UEPECC.

We next study the effect of p_{rct} on UEPECC construction for different p_{rct} in Figure 8. Intuitively, WMSE decreases with the reduction of p_{rct}, and this is confirmed in the results. Furthermore, we notice that the number of check bits assigned to higher-order bits

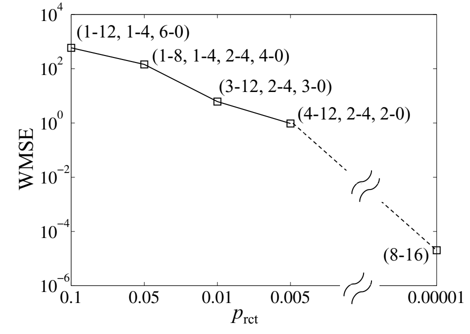


Figure 8: WMSE and UEPECC configuration for different p_{rct}. The numbers in parentheses represent the partitioning strategy for the SRAM word, referring to ($\frac{m_0}{n}$-c_0, ..., $\frac{m_{b-1}}{n}$-c_{b-1}), where c_i is the number of check bits for block i.

decreases with the reduction of p_{rct} while the number of check bits assigned to lower-order bits increases. This indicates a decrease in protection level for higher-order bits and an increase in protection level for lower-order bits. As p_{rct} further decreases, all the bits receive the same protection level and the UEPECC becomes EEPECC. As shown in the figure, when $p_{\text{rct}} = 0.1$, the 2 higher-order bits consume all 16 check bits while the 6 lower-order bits do not receive any check bits. However, when $p_{\text{rct}} = 0.00001$, the highest-order bits and the lowest-order bits are in the same block and receive the same protection level. This is due to the fact that the lower-order bits start to dominate the WMSE as p_{ct} of the higher-order bits decreases. Since p_{ct} is to the first order approximately proportional to $(p_{\text{rct}})^t$, p_{ct} decreases faster with t under low p_{rct}. Therefore, fewer check bits are assigned to the higher-order bits before the lower-order bits start to dominate WMSE.

5. Applications

In this section, we study two representative multimedia applications to demonstrate the advantage of our proposed UEPECC over the traditional EEPECC. The SRAM raw cell failure probability p_{rct} is varied from 0.1 to 0.005, corresponding to a supply voltage ranging from approximately 0.4 V to 0.6 V. The SRAM word is configured as $m = 64$, $n = 8$, $l = 8$, and $m' = 96$, with the maximum allowed number of blocks $b_{\max}$ set to 8.

5.1 Image processing

Twenty 512×512 grayscale images from the USC test image database [26] are stored into two different SRAMs configured with (i) the traditional EEPECC and (ii) the proposed UEPECC schemes. For the EEPECC, a 2-error correcting OLSC is applied to the SRAM word. In contrast, the proposed UEPECC will adjust the protection scheme according to different p_{rct}. In Figure 9 we compare the average PSNR of the stored image for both ECC schemes. It is observed that UEPECC achieves 8 dB higher PSNR than EEPECC across all p_{rct} on average.

5.2 Optical flow

Optical flow is the pattern of apparent motion caused by the relative motion between an observer and the scene, and it is widely applied in computer vision, image interpolation, and video coding. In a typical implementation of the optical flow, a sequence of image frames at different time steps are first stored in SRAMs, and an optical flow core retrieves different frames to compute the motion vectors [14]. Obviously, bit failures in the SRAMs will distort the image frame and further distort the computed motion vectors.

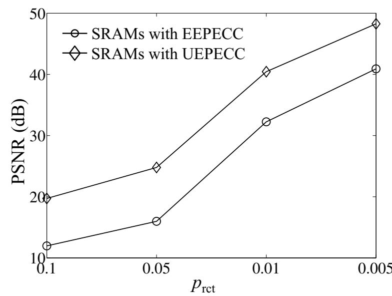


Figure 9: Comparison of average PSNR between traditional EEPECC and the proposed UEPECC across 20 test images from the USC test image database [26]

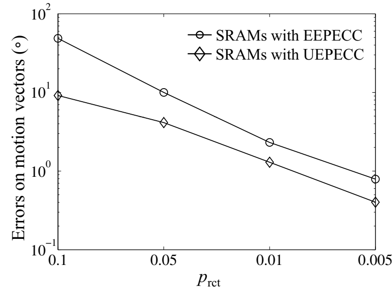


Figure 10: Comparison of introduced errors in the mption vectors between traditional EEPECC and the proposed UEPECC on the Yosemite fly-through grayscale image sequence [27]

In our evaluation, we use the open-source optical flow algorithm in [27] and 6 frames of the widely used benchmark Yosemite fly-through grayscale image sequence. The ideal motion vectors are obtained assuming that all the SRAM cells are completely error-free, i.e., $p_{\mathrm{rct}} = 0$. In Figure 10 we compare the computational errors in the motion vector due to SRAM failures with EEPECC and the proposed UEPECC scheme. It is observed that the proposed UEPECC results in 60% less errors compared to the EEPECC on average. Hence, the advantage of UEPECC is also confirmed in the optical flow application.

In both applications, since the total number of SRAM cells in one word is equal for both ECC schemes, the power consumption for both ECC schemes can also be considered equal for the same supply voltage. Furthermore, the encoding/decoding latency of the UEPECC, which is measured by the longest encoding/decoding latency among all the blocks, is also smaller than or equal to that of the EEPECC. As indicated in Section 4, the encoding (decoding) latency is 0 (1) and $\lceil \log_2(\sqrt{m}) \rceil$ ($\lceil \log_2(\sqrt{m}) \rceil + 2$) levels of 2-input XOR gates for the repetition code and the OLSC, respectively. Since the SRAM word with EEPECC has larger m than any block of the SRAM word with UEPECC, UEPECC will not incur a larger encoding/decoding latency than EEPECC.

6. Conclusions

This paper introduces UEPECC to improve the SRAM reliability at low supply voltages for mobile multimedia applications. We (i) proposed WMSE as the metric to measure the reliability of SRAM when different bits in the same word are not equally significant, and (ii) designed an optimization algorithm based on dynamic programming to construct the UEPECC. Although this paper focused on the integer data format for brevity, the proposed UEPECC can be potentially applied to other data formats including the floating-point format, through appropriate changes to the WMSE metric. The advantage of the proposed UEPECC over the traditional EEPECC is demonstrated using two representative multimedia applications. Compared to the traditional EEPECC, the proposed UEPECC can achieve an average 8 dB improvement in PSNR in image processing and incur 60% less errors in optical flows for the same area, power, and encoding/decoding latency. Thus, UEPECC enables further supply voltage scaling and reduction in power consumption, which is very attractive and promising in low-power multimedia applications.

References

[1] K. Zhang, *Embedded memories for nano-scale VLSIs*. Springer, 2009.

[2] H. Qin *et al.*, "SRAM leakage suppression by minimizing standby supply voltage," in *Intl. Symposium on Quality Electronic Design*, pp. 55–60, 2004.

[3] J. Kulkarni *et al.*, "A 160 mV robust Schmitt trigger based subthreshold SRAM," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 2303–2313, 2007.

[4] L. Chang *et al.*, "Stable SRAM cell design for the 32 nm node and beyond," in *Symposium on VLSI technology*, pp. 128–129, 2005.

[5] B. Calhoun and A. Chandrakasan, "A 256 kb sub-threshold SRAM in 65nm CMOS," in *IEEE Intl. Solid-State Circuits Conference*, pp. 480–481, 2006.

[6] W. Y. Choi *et al.*, "Tunneling field-effect transistors (TFETs) with subthreshold swing (SS) less than 60 mV/dec," *IEEE Electron Device Letters*, vol. 28, pp. 743–745, 2007.

[7] J. Kim *et al.*, "Multi-bit error tolerant caches using two-dimensional error coding," in *Intl. Symposium on Microarchitecture*, pp. 197–209, 2007.

[8] Z. Chishti *et al.*, "Improving cache lifetime reliability at ultra-low voltages," in *Intl. Symposium on Microarchitecture*, pp. 89–99, 2009.

[9] S. Zhou *et al.*, "Minimizing total area of low-voltage SRAM arrays through joint optimization of cell size, redundancy, and ECC," in *IEEE Intl. Conference on Computer Design*, pp. 112–117, 2010.

[10] R. Datta and N. Touba, "Post-manufacturing ECC customization based on orthogonal Latin square codes and its application to ultra-low power caches," in *IEEE Intl. Test Conference*, pp. 1–7, 2010.

[11] I. Chang *et al.*, "A voltage-scalable & process variation resilient hybrid SRAM architecture for MPEG-4 video processors," in *Design Automation Conference*, pp. 670–675, 2009.

[12] M. Cho *et al.*, "Accuracy-aware SRAM: A reconfigurable low power SRAM architecture for mobile multimedia applications," in *Asia and South Pacific Design Automation Conference*, pp. 823–828, 2009.

[13] X. Li, "Maximum-information storage system: Concept, implementation and application," in *Intl. Conference on Computer-Aided Design*, pp. 39–46, 2010.

[14] C. Claus *et al.*, "High performance FPGA based optical flow calculation using the census transformation," in *IEEE Intelligent Vehicles Symposium*, pp. 1185–1190, 2009.

[15] M. Horigochi and K. Itoh, *Nanoscale memory repair*. Springer, 2011.

[16] Z. Wang *et al.*, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Processing*, vol. 13, pp. 600–612, 2004.

[17] S. Borade *et al.*, "Unequal error protection: An information-theoretic perspective," *IEEE Trans. Information Theory*, vol. 55, pp. 5511–5539, 2009.

[18] F. MacWilliams and N. Sloane, *The theory of error correcting codes*. North–Holland, 1981.

[19] H. Hsiao *et al.*, "Orthogonal Latin square codes," *IBM Journal of Research and Development*, vol. 14, pp. 390–394, 1970.

[20] J. Wang *et al.*, "SRAM parametric failure analysis," in *Design Automation Conference*, pp. 496–501, 2009.

[21] Y. Cao *et al.*, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Custom Integrated Circuits Conference*, pp. 201–204, 2000.

[22] K. Agarwal and S. Nassif, "Statistical analysis of SRAM cell stability," in *Design Automation Conference*, pp. 57–62, 2006.

[23] B. Masnick and J. Wolf, "On linear unequal error protection codes," *IEEE Trans. Information Theory*, vol. 13, pp. 600–607, 1967.

[24] J. Ihm *et al.*, "An 80nm 4Gb/s/pin 32b 512Mb GDDR4 graphics DRAM with low-power and low-noise data-bus inversion," in *IEEE Intl. Solid-State Circuits Conference*, p. 492, 2007.

[25] B. Sklar, *Digital communications: fundamentals and applications*. Prentice Hall, 2001.

[26] *The USC-SIPI Image Database*. http://sipi.usc.edu/database/.

[27] T. Gautama *et al.*, "A phase-based approach to the estimation of the optical flow field using spatial filtering," *IEEE Trans. Neural Networks*, vol. 13, pp. 1127–1136, 2002.