Motion estimation performance models with application to hardware error tolerance

Hye-Yeon Cheong and Antonio Ortega

Dept. of Electrical Engineering, Signal and Image Processing Institute
Univ. of Southern California, Los Angeles, CA

ABSTRACT

The progress of VLSI technology towards deep sub-micron feature sizes, e.g., sub-100 nanometer technology, has created a growing impact of hardware defects and fabrication process variability, which lead to reductions in yield rate. To address these problems, a new approach, system-level error tolerance (ET), has been recently introduced. Considering that a significant percentage of the entire chip production is discarded due to minor imperfections, this approach is based on *accepting imperfect chips that introduce imperceptible/acceptable system-level degradation*; this leads to increases in overall *effective* yield. In this paper, we investigate the impact of hardware faults on the video compression performance, with a focus on the motion estimation (ME) process. More specifically, we provide an analytical formulation of the impact of single and multiple stuck-at-faults within ME computation. We further present a model for estimating the system-level performance degradation due to such faults, which can be used for the error tolerance based decision strategy of accepting a given faulty chip. We also show how different faults and ME search algorithms compare in terms of error tolerance and define the characteristics of search algorithm that lead to increased error tolerance. Finally, we show that different hardware architectures performing the same metric computation have different error tolerance characteristics and we present the optimal ME hardware architecture in terms of error tolerance. While we focus on ME hardware, our work could also applied to systems (e.g., classifiers, matching pursuits, vector quantization) where a selection is made among several alternatives (e.g., class label, basis function, quantization codeword) based on which choice minimizes an additive metric of interest.

Keywords: Error tolerant compression, Motion estimation, Computation error tolerance, Multiple stuck-at fault

1. INTRODUCTION

As semiconductor manufacturing technology advances, yield rate has decreased and it is becoming economically more difficult to produce close to 100% correct circuits as they become more highly condensed and integrated towards nanoscale. Lower yield rates can increase the chip manufacturing and verification cost and delay their time-to-market. To deal with these problems and improve yields, defect and fault tolerance techniques at the design and manufacturing stages have been widely studied and practiced.[1,2] While these approaches aim at masking all effects of defects such that no error occurs at any output of a faulty system, a recently introduced application-oriented approach to error tolerance (ET) is based on not discarding those systems that produce *erroneous yet acceptable* output,[3] so that overall effective yield can be improved. ET is based on determining whether a system is acceptable or unacceptable by analyzing the system-level impact of faults and accepting minor defects which result in slightly degraded performance within some application specific ranges of acceptability. This is a particularly promising solution for highly integrated LSI chips used for video compression applications, because the analysis can exploit the limitations in human perception and the fact that compression itself is a lossy processs. However, a critical factor for this approach to be successful is to be able to cost-efficiently test and accurately predict if a defective chip will provide acceptable system-level performance without having to perform application level testing.

Further author information:
Hye-Yeon Cheong, E-mail: hyeyeonc@usc.edu
Antonio Ortega, E-mail: ortega@sipi.usc.edu

In our work to date we have studied the impact of hardware defects that lead to faults at circuit interconnects and soft errors produced by voltage scaling. Hardware faults such as those arising in a typical fabrication process can potentially lead to "hard" errors, since some of the functionality in the design is permanently impaired, whereas "soft" errors may arise when a circuit operates at a voltage lower than specified for the system. Our previous works[4,5] showed that certain range of hardware defects within the motion estimation (ME) and discrete cosine transform (DCT) with quantization subsystems lead to acceptable quality degradation. We also proposed a novel ET based testing strategy for such systems. Our other recent work[6] showed that the ME process exhibits significant error tolerance in both hard and soft errors and further proposed simple error models to provide insights into what features in ME algorithms lead to increased error tolerance.

In this paper, based on the same ET concept, we further extend our previous work and provide an analytical formulation of the impact of multiple hardware faults on the motion estimation (ME) process. This provides estimates of system-level performance degradation due to such faults, which can be used in deciding whether to accept a given faulty chip. Furthermore, based on this model, we investigate the error tolerance behavior of the ME process in the presence of multiple hardware faults from both an algorithmic and a hardware architecture point of view. In comparing different motion search algorithms we observe that high performance, low complexity techniques can in fact more error tolerant than other methods. As an example, in our experiments, enhanced predictive zonal search (EPZS)[7] exhibits minimum degradation with respect to full search (FS) in the fault-free case (e.g., $0.01dB$ loss) but can perform significantly better in the presence of faults (e.g., up to $2.5dB$ gain compared to a faulty FS in some cases). When comparing different hardware architectures to perform the same metric computation, we also observe significant variations in error tolerance. We show that the optimal ME hardware architecture in terms of error tolerance is a perfectly balanced binary tree structure, which is also effective in terms of enabling parallel computation. Our simulations show that, if the optimal structure is used, the expected error due to a fault can be reduced by up to 95%, as compared to other architectures, and that more than 99.2% of fault locations within ME circuits result in less than $0.01dB$ performance degradation. While we focus on ME hardware, our work could also applied to systems (e.g., classifiers, matching pursuits, vector quantization) where a selection is made among several alternatives (e.g., class label, basis function, quantization codeword) based on which choice minimizes an additive metric of interest.

Our previous work as well as most previous research has relied on the single stuck-at (SA) fault assumption, which has been well-studied due to its simplicity and high fault coverage and has worked fairly well in practice. However, with decreasing feature sizes and increasingly aggressive design styles, single SA fault has become a rather restrictive model since it allows the defect to influence only one net, while defects in modern devices tend to cluster and affect multiple lines in the failing chip. Multiple SA fault model on the other hand covers a greater percentage of physical defects, and can also be used to model defects of certain different fault types such as multiple bridging faults or multiple transition faults. However, the multiple SA fault case has not been studied extensively due to the issues of complexity and functional error inter-dependency characteristics. In this paper, we present an analytical model of the system-level fault effect for any arbitrary multiple SA faults in a metric computation circuit and study error tolerance properties of the ME process at both algorithmic and hardware architecture level based on multiple SA fault assumption.

The rest of this paper is organized as follows. In Section 2, we begin with a brief description of motion estimation and a corresponding generalized hardware architecture model, followed by the formulation of multiple SA fault effect estimation problem on ME process. In Section 3, we present a model for measuring the impact of multiple SA fault on the accuracy of both the metric computation and matching process. In Section 4, we analyze and define the characteristics of ME search algorithms that lead to increased error tolerance. In Section 5, we provide analysis on ME hardware architecture pertaining to the error tolerance and show the optimal structure is perfectly balanced tree structure. Furthermore, we present experimental results and discussion on actual ET based decision example in the context of H.264/AVC, comparing different ME hardware architectures and search algorithms. Section 6 summarizes our conclusions and main results.

2. MOTION ESTIMATION PROCESS AND MULTIPLE SA FAULT MODEL

Block-based motion compensation is by far the most popular technique to exploit temporal redundancy in video coding, and has been adopted in many video compression standards. Block-based techniques require estimating

the motion vector (MV) of a block of pixels, which points to the best matching block in the reference picture. Throughout this paper we assume block-based techniques are used. The ME process comprises a search strategy and a matching metric computation. The search strategy aims at selecting a set of candidate MVs and then proceeds to compute the matching metric for each candidate. Finally, it selects the one that minimizes a relevant metric, e.g., the sum of absolute/squared differences (SAD/SSD), or absolute hadamard transformed differences ($SATD$). After the encoder selects the MV that minimizes the cost metric, it encodes the difference block (prediction residual) between the original and motion compensated blocks. Each residual block is transformed, quantized, and entropy coded. All simulations throughout this paper were performed using the SAD metric. However, our analysis and models are generic to any distortion metric.

There are several types of hardware implementation architectures[8] for computing a ME matching metric, with different levels of parallelism. We will refer to them as matching metric computation (MMC) architectures. Figure 1 illustrates a few examples of MMC architectures[8] which can be viewed as arrays of cascaded adders and represented as binary tree graphs, where each inner node represents an adder, an edge connecting two inner nodes represents a data bus, and a leaf node corresponds to the processing element (PE) that computes the pixel-level prediction error, e.g., absolute/squared difference for SAD/SSD metric.

Throughout this paper we consider only faults in the interconnect data bus that affect the data transfer between PEs within a ME hardware architecture. These interconnect faults are modeled with the stuck-at (SA) fault model, a well-known structural fault model that assumes that the design contains a fault that will cause a line in the circuit to behave as if it is permanently stuck at a logic value 0 (stuck-at-0 fault) or 1 (stuck-at-1 fault). The SA fault model covers 80-90% of the possible manufacturing defects in CMOS circuits,[9] such as missing features, source-drain shorts, diffusion contaminants, and metallization shorts, oxide pinholes, etc.

Faults in a MMC architecture can lead to errors in the cost metric computation. We will refer to this as a metric computation (MC) error and denote its magnitude as $\Delta(D, \{f_i\}) = \hat{D} - D$ where $\hat{D}$ and D denote the computed costs with and without MC error. In general $\Delta(D, \{f_i\})$ is a function of the input sequence and the characteristics of the faults (e.g., their types, locations, and their dependencies). However, to simplify the notation, from here on we omit $\{f_i\}$, a set of parameter vectors representing each fault's characteristics. We will refer to this function $\Delta(D)$ as the *MC error function*. Note that MC errors do not necessarily lead to matching process (MP) errors. We say a block suffers a MP error if $MV_f \neq MV_{min}$ where MV_f and MV_{min} represent

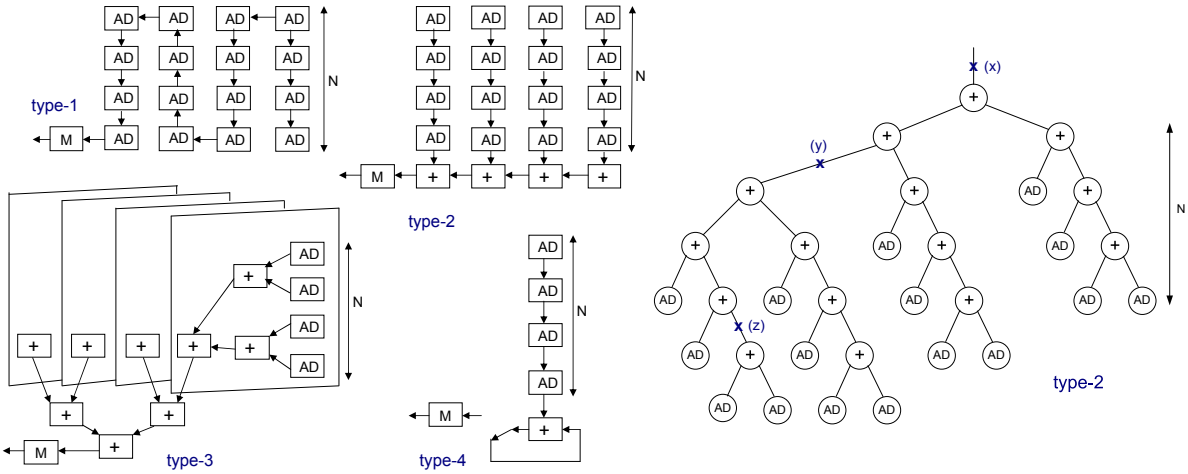

Figure 1. (Left) Examples of MMC architectures for the computation of prediction error (SAD) in ME, represented as a dependence graph, where only processing elements (PEs) are shown for simplicity. AD denotes an absolute difference and an addition, and M denotes a minimum value computation. (Right) Tree structured flow graph corresponding to the type-2 MMC architecture on the left. In this graph, the absolute difference (leaf nodes) and addition (inner nodes) computations are separated, thus AD denotes only an absolute difference. If SSD is used as a cost metric, leaf nodes become PEs computing a square difference.

the selected and best MV, respectively . Since MP errors do not occur for all blocks, we also define the block MP error rate ($P_E = prob(MV_f \neq MV_{min})$) which represents how often these MP errors occur, and the block MP error significance ($S_E = D(MV_f) - D(MV_{min})$, where $D(MV_f)$ and $D(MV_{min})$ are the computed costs corresponding to MV_f and MV_{min} respectively). S_E represents the additional error energy in the residual block due to faults in the motion estimation process.

3. FAULT EFFECT MODELING FOR MOTION ESTIMATION

In this section we present a complete analysis of the effect of interconnect faults in a ME system. We start by providing a model to capture the effect of any fault (or combination of multiple faults) on the matching metric (SAD). We then describe how these errors in metric computation lead to errors in the matching process. Our motivation is twofold. First, we will use this analysis to predict the behavior of ME algorithms and MMC architectures in the presence of faults (see Sections 4 and 5, respectively). Second, this analysis is a required step towards developing comprehensive testing techniques to identify acceptable error behavior. In particular, it will provide tools to tackle multiple fault scenarios, by discarding specific sets of faults (e.g., when one of the multiple faults is by itself unacceptable), establishing equivalence classes across faults (e.g., cases when dependent and independent faults may have the same impact), or determining how the parameters of the faults involved affect overall behavior (e.g., multiple faults with similar parameters may lead to relatively worse errors than sets of faults with differing characteristics).

3.1. Multiple Fault Effect Modeling for Metric Computation

Each SA fault in any MMC architecture can be fully characterized by three attributes, a) the fault type $t \in \{0, 1\}$ (SA0 or SA1), b) the bit position of the faulty data line $p \in \{0, \cdots, P\}$ where 0 and P correspond to the LSB and MSB of faulty data bus respectively, depending on the MMC architecture, and c) the position of that data bus. The last attribute can be parameterized as a ratio $\alpha \in (0, 1\,]$ of the number of leaves in the subtree rooted at a faulty edge Figure 1 (right), if a fault is positioned at (x), its corresponding α, say $\alpha_{(x)}$ is 1. Similarly, $\alpha_{(y)} = 1/2$ and $\alpha_{(z)} = 1/8$ and so forth. Therefore each fault i can be represented as $f_i = (t_i, p_i, \alpha_i)$, where $t_i \in \{0, 1\}$, $p_i \in \{0, \cdots, P\}$ and $\alpha_i \in (0, 1\,]$. Note that these fault parameters completely capture the fault effect, so that two faults with the same parameters in two different architectures have, on average, the same effect.

Consider an interconnect fault i affecting a data bus. If the input to the data bus is x then we can represent the effect of the fault as a function $h_i(x) \triangleq \hat{x} - x$, where $\hat{x}$ is the output of the faulty data bus. For fault parameters p_i and t_i the shift induced by SA fault i can then be described as

$$h_i(x) = \hat{x} - x = (-1)^{t_i+1} \cdot 2^{p_i} \cdot I_{w^i_{t_i}}(x), \quad \text{where}$$

$$w^i_0 = \bigcup_{k \in 2N_0+1} [\, k \cdot 2^{p_i}, (k+1) \cdot 2^{p_i}\,) \quad \text{and} \quad w^i_1 = \bigcup_{k \in 2N_0} [\, k \cdot 2^{p_i}, (k+1) \cdot 2^{p_i}\,),$$

where N_0 denotes the set of non-negative integers. Note that $h_i(x)$ is a function of x, i.e., an intermediate term in the computation of the final matching metric cost D. Thus, modeling the final observed metric computation error would in principle require modeling the distribution of intermediate values x for a given D. Instead, as a simplification, we assume that, on average and for a given α_i and D, we will tend to have $x \cong \alpha D$, i.e., the intermediate metric value x at the fault position is proportional to the number of pixel-errors accumulated up to that position*. With this approximation we define the error at the output corresponding to fault i as $H_i(D) \cong h_i(\alpha D)$, which can be written as:

$$H_i(D) \triangleq (-1)^{t_i+1} \cdot 2^{p_i} \cdot I_{W^i_{t_i}}(D), \tag{1}$$

$$\text{where} \qquad W^i_0 = \bigcup_{k \in 2N_0+1} \left[\, \frac{k \cdot 2^{p_i}}{\alpha_i}, \frac{(k+1) \cdot 2^{p_i}}{\alpha_i} \,\right) \quad \text{and} \quad W^i_1 = \bigcup_{k \in 2N_0} \left[\, \frac{k \cdot 2^{p_i}}{\alpha_i}, \frac{(k+1) \cdot 2^{p_i}}{\alpha_i} \,\right)$$

*This would hold if pixel-errors were iid with mean dependent on D. However, based on the generalized central limit theorem even if there exist some weak dependencies between pixel-errors, this still holds true if none of pixel-error variables exert a much larger influence than the others.

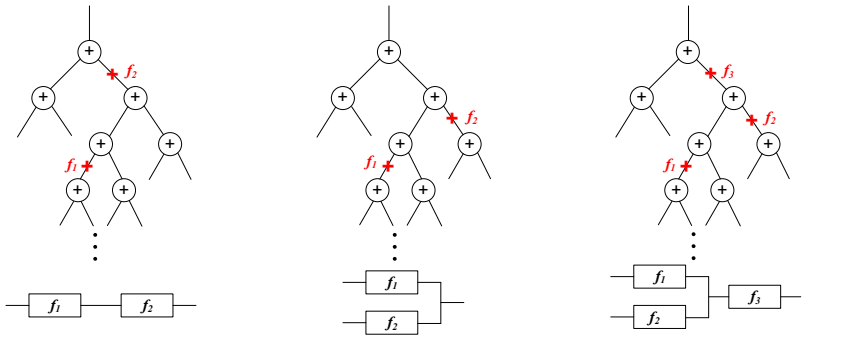

Figure 2. Examples of multiple SA fault cases with different dependency relations between faults. (a) two serially placed faults in the same path of circuit, (b) two parallel faults placed in separate paths, and (c) simple combination of (a) and (b) cases. These relations are illustrated more clearly in the fault dependency graphs shown below the binary trees. Metric computation (MC) error function $\Delta(D)$ can be formulated via functions of $\{H_i(D)\}_{i=1}^{M}$ defined in (1) using two basic operators $\{\odot, +\}$ (function addition and a variant of function composition operator defined in (2)), according to the dependency relation of a given set of faults as shown above, where D denotes the final cost value (e.g., SAD).

$H_i(D)$ is a periodic piecewise constant square wave function of D taking values in the set $R_i = \{0, \quad 2^{p_i} \cdot (-1)^{t_i+1}\}$ with period $T_i = 2^{p_i+1}/\alpha_i$. If there is a single SA fault $f_i = (t_i, p_i, \alpha_i)$ in a MMC architecture, then with the above approximation the error at the output is $\Delta(D) \cong H_i(D)$, which leads to D being shifted by $\pm 2^p$ or remaining unchanged depending on D and f_i.

However, when we consider multiple SA faults, $\Delta(D)$ cannot be accurately modeled unless extra information regarding the dependency relation between faults is also provided. In other words, two sets of faults with identical fault configurations, $\{f_i\}_{i=1}^{M}$ with $f_i = (t_i, p_i, \alpha_i)$ could produce different MC error function $\Delta(D)$ depending on the dependency relation between faults. Figure 2 illustrates simple examples of multiple faults with different dependencies. If faults are placed serially in the same computation path, the output of previous faults along the path could affect the input to the following fault position. We refer to such faults as *dependent* faults (Figure 2(a)). Note that the relative position of the faults in the architecture matters, i.e., the computations closer to the root of the tree are affected by the errors introduced closer to the leaves, but not the other way around. On the other hand, if faults are placed in separate paths such that the output of each fault position is independent from one another, we refer to them as *independent* faults (Figure 2(b)). Figure 2(c) shows the case of a simple combination of both independent and dependent faults. For each of these three scenarios, Figure 2 provides an example of multiple faults locations in the MMC hardware architecture represented as a binary tree and a fault dependency graph illustrating their dependency relation. Similarly, any dependency relation of an arbitrary set of faults $S = \{f_i\}_{i=1}^{M}$ in a MMC architecture, where $f_i = (t_i, p_i, \alpha_i)$, can be represented as a combination of parallel and serial cascade of faults.

In order to formulate $\Delta(D)$ for any given set of faults with arbitrary fault dependency relations, we first introduce $(\mathcal{F}, +, \odot)$, an algebraic structure where the elements of $\mathcal{F}$ are periodic piecewise constant functions which are closed under two operations $\{+, \odot\}$. The operator $+$ denotes linear function addition and $\odot$ denotes a function composition operator defined as

$$f(x) \odot g(x) \triangleq g(f(x) + x) + f(x) \tag{2}$$

This operator $\odot$ combines $f(x)$ and $g(x)$, where the input to $g(x)$ depends on the output of $f(x)$. These two operators $\{+, \odot\}$ essentially combine two MC error functions for two different sets of faults into one function providing the total cost shift incurred by both sets of faults. If two sets of faults are independent, the overall error function is the sum of the two error functions corresponding to each set of faults, thus the $+$ operator is used. Similarly, the $\odot$ operator is applied when two sets of faults are dependent. Figure 2 provides simple examples of multiple faults and their corresponding $\Delta(D)$ representation using two operators $\{+, \odot\}$. Figure 3 illustrates more graphically for each of the three cases discussed in Figure 2 how $\Delta(D)$ function is linked to

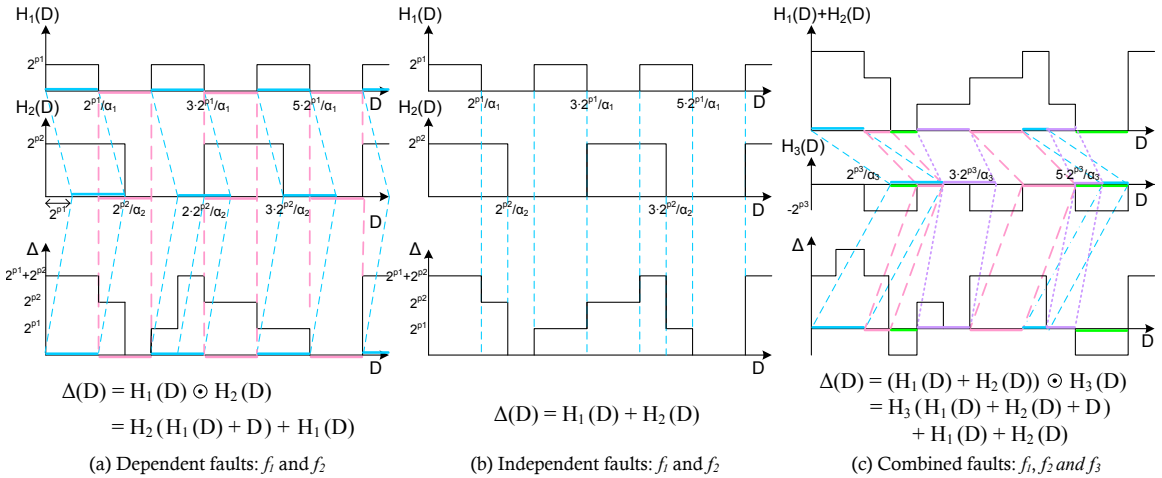

Figure 3. $\Delta(D)$ computation for each of the three cases described in Figure 2. Faults f_1 and f_2 are SA1 faults with parameters $\{1, p_1, \alpha_1\}$, $\{1, p_2, \alpha_2\}$ respectively and f_3 is SA0 fault with $\{0, p_3, \alpha_3\}$. (a) a fault f_1 affects the input to the next fault position f_2. Thus, the total cost shift at D (MC error function $\Delta(D)$) is the summation of the cost shift due to f_1 at D, $(H_1(D))$ and that of f_2 at shifted D by $H_1(D)$, which is $(H_2(H_1(D) + D))$. (b)$\Delta(D)$ is the simple linear summation of $H_1(D)$ and $H_2(D)$. (c) only dependent relation between faults $\{f_1, f_2\}$ and f_3 is depicted which is essentially the same process as (a).

the dependency relation of multiple faults. When two sets of faults are independent as in Figure 3 (b), $\Delta(D)$ function becomes linearly additive. On the other hand, when they are dependent, as in Figure 3 (a)(c), the error function will have to be computed by using the $\odot$ operator.

More formally, the MC error function $\Delta(D)$ of an arbitrary set of faults $S = \{f_i\}_{i=1}^{M}$ in any MMC architecture, where $f_i = (t_i, p_i, \alpha_i)$, with any dependency relation can be estimated using operators $\{+, \odot\}$ on the set of functions $\{H_i(D)\}_{i=1}^{M}$ resulting also in a periodic piecewise constant function of cost D, taking 2^M different possible values in a finite set R_S with period T_S, where[†]

$$R_S = \left\{ \sum_{i \in \mathcal{A}} 2^{p_i} \cdot (-1)^{t_i+1} \right\}_{\mathcal{A} \in P(\{1,\cdots,M\})} \qquad T_S = lcm \left\{ T_i = \frac{2^{p_i+1}}{\alpha_i} \right\}_{i=1}^{M} \tag{3}$$

Since the $\odot$ operator is noncommutative and nondistributive over addition, order is important in computing $\Delta(D)$. Based on its fault dependency graph (examples are shown in Figure 2 which can be also represented as a tree in which each node i correspond to a fault i), $\Delta(D)$ need to be updated iteratively at each node i of the tree from the leaves towards the root by combining each $H_i(D)$. The combination process at each node i is performed using $+$ and $\odot$ operators to combine siblings $(\{\Delta_j(D)\}_{j \in \mathcal{J}_i})$ and children $(\sum_{j \in \mathcal{J}_i} \Delta_j(D))$ with parent $(H_i(D))$, respectively, where $\Delta_i(D)$ represents the updated $\Delta(D)$ for a subtree rooted at node i and $\mathcal{J}_i$ is a set of all child nodes of node i.

$$\Delta_i(D) = (\sum_{j \in \mathcal{J}_i} \Delta_j(D)) \odot H_i(D)$$

Therefore for a given arbitrary set of multiple SA faults, we can obtain MC error function $\Delta(D)$ which indicates whether there is a MC error $(\Delta \neq 0)$ and how much shift occurred due to faults $(\Delta = \hat{D} - D)$ for any given final cost value D. Note that once we obtain MC error function $(\Delta(D))$, the dependency relation does not have to be reconsidered afterwards.

3.2. Multiple Faults Effect Modeling for the Matching Process

SA faults alter the computed cost values $\hat{D} = D + \Delta(D)$ according to the MC error function which we have analytically formulated in the previous section. However, this does not necessarily lead to matching process (MP)

[†] $P(\cdot)$ and $lcm(\cdot)$ denote the power set and the least common multiple, respectively.

errors, which is the case where the selected candidate $MV(MV_f)$ based on the altered cost value $\hat{D}$ is not equal to the best $MV(MV_{min})$. A MP error occurs if and only if, $D(MV_f) > D(MV_{min})$ and $\hat{D}(MV_f) < \hat{D}(MV_{min})$.

In this section, based on the MC error function $\Delta(D)$ we model the MP error with two assumptions: i) minimum SAD follows a distribution P_{minSAD} which has different characteristics for different classes of video sequences (e.g., low motion vs. high motion video), and ii) the set of candidates N to be tested for the matching process can be modeled as N iid samples drawn from a distribution P_{SAD}. Based on these two assumptions, for a given $\Delta(D)$, MP error rate P_E and the expected MP error significance $\bar{E}$ are formulated as follows.

SA faults shift all computed costs by $\Delta(D)$ which in turn alter P_{SAD} into $\hat{P}_{SAD}$ where

$$\hat{P}_{SAD}(\hat{D}) = \sum_{\forall D: D+\Delta(D)=\hat{D}} P_{SAD}(D) \tag{4}$$

Figure 4 (left) illustrates example of how a given P_{SAD} is mapped into $\hat{P}_{SAD}$ using $\Delta(D)$ function. MP error occurs if the shifted minimum SAD ($\widehat{minSAD}$) is not the minimum of $\hat{P}_{SAD}$ in $\hat{D}$ domain and if there exists at least one candidate that falls within the shaded region shown in Figure 4. We refer to this shaded area as *error region* which essentially indicates the range of SAD values satisfying $SAD > minSAD$ and $\widehat{SAD} < \widehat{minSAD}$ conditions and therefore determined by $\Delta(D)$. This *error region* is bounded by *error region bounding interval* τ which is a function of $minSAD$ value and defined as

$$\tau(D) \triangleq \Delta(D) - minR_S \tag{5}$$

where $minR_S$ denotes the minimum of a set R_S defined in (3). We denote $\hat{F}_{SAD}$ as a cumulative distribution function of $\hat{P}_{SAD}$ and define a new function $\xi_N(D) \triangleq (1 - \hat{F}_{SAD}(\hat{D}))^N$ that gives the probability that shifted cost values of all N candidates fall higher than $\hat{D}$. Then we can represent MP error rate for a given $minSAD$ (denoted as D_{min}) as $P_{E|D_{min}} = 1 - \xi_N(D_{min})$. Therefore,

$$P_E = \sum_D P_{minSAD}(D) \cdot P_{E|D} = \sum_D P_{minSAD}(D) \cdot (1 - \xi_N(D)) \tag{6}$$

Similarly, expected value of MP error significance for a given D_{min} can be represented as

$$\mathbb{E}_{D_{min}}(S_E) = \sum_{d \in \mathcal{X}_{D_{min}}} (d - D_{min}) \cdot \xi_{N-1}(d) \cdot P_{SAD}(d)$$

where $\mathcal{X}_{D_{min}} = \left\{ d \mid D_{min} < d < D_{min} + \tau(d),\ \hat{d} < \hat{D}_{min} \right\}$ is a set of d corresponding to the blue shaded area of $P_{SAD}(d)$ in Figure 4 (Left).

Similarly, the expected MP error can be expressed as

$$\bar{E} = \mathbb{E}(S_E) = \sum_D P_{minSAD}(D) \cdot \mathbb{E}_D(S_E) = \sum_D P_{minSAD}(D) \sum_{d \in \mathcal{X}_D} (d - D) \cdot \xi_{N-1}(d) \cdot P_{SAD}(d) \tag{7}$$

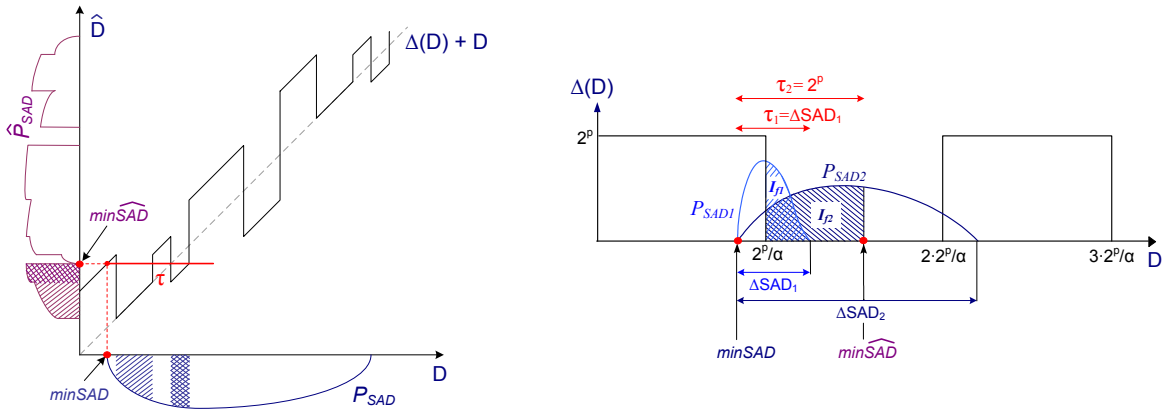

Figure 4. (Left) Analysis of matching process error due to multiple SA faults. (Right) A comparison of two different sets of MV candidates with respect to the MP error due to a single SA fault.

4. ERROR TOLERANCE OF MOTION ESTIMATION SEARCH ALGORITHMS

In Section 3, we introduced a function, $\Delta(D)$, to model the distortion in metric computation caused by a given set of multiple SA faults. Based on this model, we modeled the impact of faults on matching process by additionally considering the interaction between $\Delta(D)$ and the characteristics of a candidate set in terms of the number N and quality (distribution P_{SAD}). These attributes can be largely controlled by the ME search algorithm design while $\Delta(D)$ is determined solely by fault parameters i.e., locations, types, and dependencies. Therefore, in this section, based on our previous studies of MP error we define the characteristics of the search algorithm that lead to increased error tolerance and show how fault parameters and design choices made for the search process are related in terms of error tolerance. Note that our interest is not in accurate modeling of ME search algorithms but in characterizing their average behavior with respect to the parameters of our interest that are related to the error tolerance property.

The multiple faults effect model for the matching process in terms of P_E and $\bar{E}$ in (5),(6) shows that error robustness depends on both the number N and quality P_{SAD} of the candidates tested by ME algorithm. Note that both P_E and $\bar{E}$ are a function of $\xi_N(d) = (1 - \hat{F}_{SAD}(\hat{d}))^N$, which is the only term associated with N. Since $\xi_N(d)$ decays exponentially as a function of N, for large enough N the difference in performance between different N values is negligible. Thus, in practice, for values of N encountered in most practical algorithms (e.g., $N > 10$) differences in overall error behavior are mostly determined by differences in quality of the candidates, rather than their number.

As to the quality of the candidates, P_{SAD} in our model, we approximate it by the range of P_{SAD} distribution (a coverage of SAD values) defined as $\Delta SAD = maxSAD - minSAD$ based on two reasons: i) Although it varies from block to block, our experiments show that average distribution given the same ΔSAD exhibits close to uniform distribution, and ii) *error region bounding interval* τ defined in (7) over which both P_E and $\bar{E}$ expectation computations are performed is bounded by ΔSAD. In other words, both P_E and $\bar{E}$ can be directly controlled by ΔSAD. Therefore, we approximate the quality of the candidates with one parameter ΔSAD in this section for the purpose of studying error tolerance property of ME algorithms since it generally captures the main feature of P_{SAD} that is most related to the P_E and $\bar{E}$.

For given M multiple faults, there can be 2^M different τ values $\tau_i \in \mathcal{T} = \{\tau | \tau = r - minR_S, r \in R_S\}$. If $\Delta SAD > \tau_i \ \forall i$, both P_E and $\bar{E}$ are determined dominantly by $\Delta(D)$ function but not by ΔSAD. In other words, if a set of faults has all small p_i which decides R_S and consequently $\mathcal{T}$, the choice of ME algorithm will not change the impact of faults and the impact itself of such faults is generally already insignificant due to their small p_i. Figure 6 (left) shows that three ME algorithms with different ΔSADs result in similar $\bar{E}$ for small p. On the other hand, if $\Delta SAD < \tau_i \ \exists i$, then $\tau_i = \Delta SAD \ \forall i \in \{i \,|\, \tau_i > \Delta SAD\}$, resulting the *error region bounding interval* τ_i for P_E and $\bar{E}$ computation to be bounded by ΔSAD for some region of D. Therefore, the choice of ME algorithm can influence the impact of faults considerably. If $\Delta SAD < \tau_i, \forall i$, all τ_i are completely bounded by ΔSAD and the error depends primarily on ΔSAD.

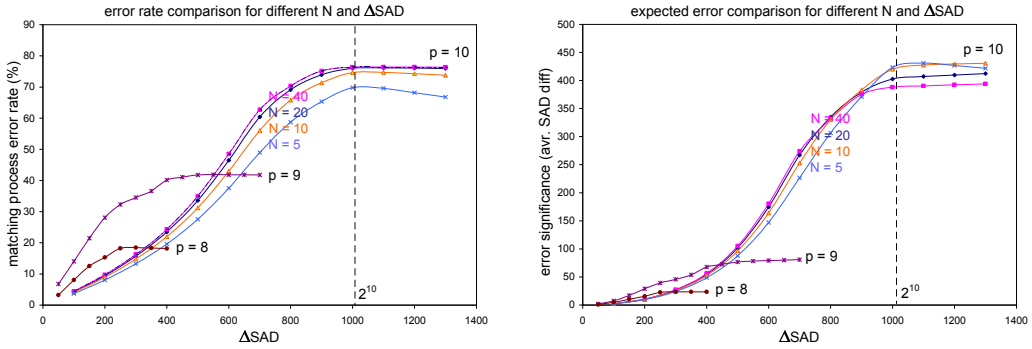

Figure 5. Effect of ΔSAD, N and fault location parameter p with respect to their impacts on the error rate P_E and expected error significance $\bar{E}$ based on our SA fault effect model (6),(7) and $P_{minSAD}(D)$ obtained from the CIF *foreman* sequence.

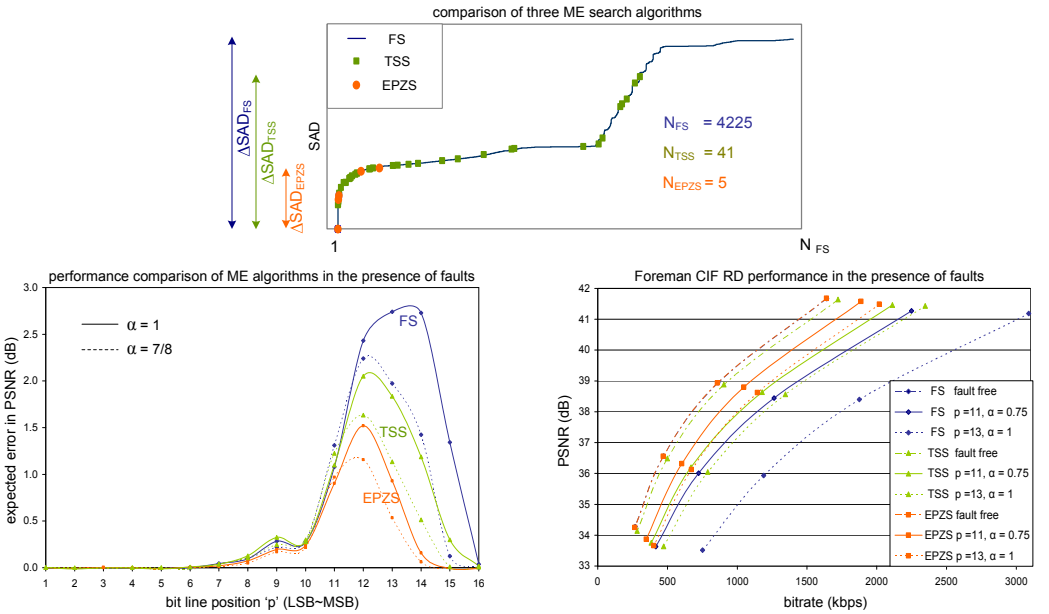

Figure 6. (top) comparison of three search algorithms FS, TSS, and EPZS with respect to ΔSAD and N. (left) ME algorithm comparison in coding efficiency loss for different SSA faults with parameters, $p = 0 \cdots 15$ from LSB to MSB at $\alpha = 1$ and 7/8. (right) RD impact of faults for different ME search algorithms.

Figure 4 (right) provides a simple comparison of two candidate sets with different ΔSADs in the presence of a single SA fault and show how ΔSAD is linked with the expected error. Figure 5 illustrates our conclusions drawn from our model for a single SA fault case. It shows that P_E and $\bar{E}$ increase almost exponentially with ΔSAD and saturate when ΔSAD reaches $\tau = 2^p$ while the impact of N is minimal especially for large N.

Therefore, if our goal is to choose an ME algorithm that reduces the impact of faults, we should choose one such that typical sets of MV candidates are as close as possible to optimal value (i.e., small ΔSAD). In practice, search algorithms satisfying this characteristic also show significant complexity reduction (small N) without having to compromise with the performance.

To evaluate the impact of ME algorithm on error tolerance, three representative ME search algorithms, full search (FS), three step search (TSS),[10] and enhanced predictive zonal search (EPZS)[7] are tested as they provide different combinations of N and ΔSAD parameters. FS exhaustively searches all candidates within the search window, and thus has the largest number of candidates N_{FS} and SAD range ΔSAD_{FS}. TSS successively evaluates sparsely distributed candidates and tries to follow the direction of minimum distortion to locate the smallest SAD. Although the number of candidates N_{TSS} is small, its SAD range ΔSAD_{TSS} remains relatively large. On the other hand, EPZS, a state of the art ME algorithm, considers a combination of optimized predictors and refinement process to locate the minimum distortion location. Unlike TSS, both the number of candidates N_{EPZS} and SAD range ΔSAD_{EPZS} tend to be quite small. As an example, when a search window of ± 32 is used, $N_{FS} = 4225$, $N_{TSS} = 41$, and $N_{EPZS} = 8.8$ on average for *Foreman* CIF sequence. For the same sequence, we have $\Delta SAD_{FS} = 1.5 \times \Delta SAD_{TSS} = 9.75 \times \Delta SAD_{EPZS}$ on average. These properties are illustrated in Figure 6 (top), which shows the distribution of the SADs for all candidates of a given block, sorted by magnitude.

With these three algorithms, various sequences were tested with a series of fault parameters using a H.264 /MPEG-4 AVC baseline encoder. Only 16x16 block partitions, a single reference, and only integer-pel search were used for ME. Note that all experimental results presented from this point forward were performed under these same constraints. Figure 6 (left) provides the comparison of three algorithms in terms of PSNR degradation[‡] due to a single SA fault with different parameters α and p while Figure 6 (right) depicts their RD performance. The CIF resolution *Foreman* sequence was used for both graphs and other sequences tested showed similar results.

[‡]BDPSNR (Bjontegaard Delta PSNR)[11] was used to calculate average PSNR difference between RD curves.

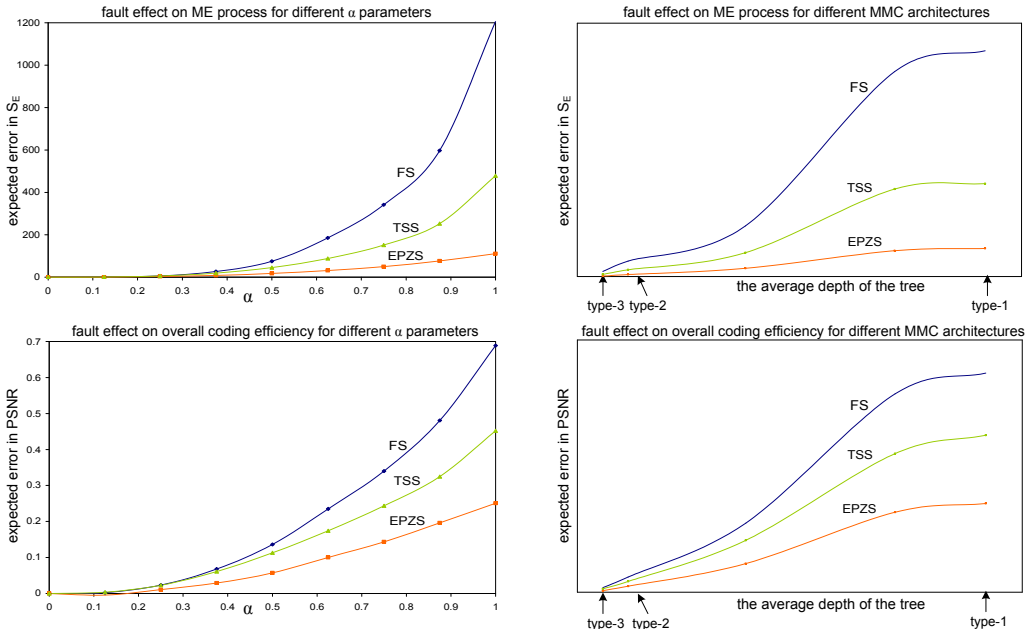

Figure 7. Average fault effect $\bar{E}$ on ME performance (top) and overall coding efficiency (bottom) are shown for different α parameters (left) and for different average binary tree depths ($D = \mathcal{N} \cdot E(\alpha)$) which describe different MMC architectures (right). Perfectly balanced trees (type-3 in Figure 1) show the minimum expected error introduced by a single SA fault.

Our experimental results are consistent with our earlier conclusions that the fault location parameters p and α mainly determines the fault impact on ME performance while for a given fault location, ME algorithm operating on smaller ΔSAD reduces the fault impact.

5. OPTIMAL ME HARDWARE ARCHITECTURE FOR ERROR TOLERANCE

In addition to any error tolerance provided by a ME algorithm, the different MMC architectures can also significantly influence the degree of error tolerance. For example, $\bar{E}$ can be reduced by more than 95% if a type-3 MMC architecture is used instead of type-1 (shown in Figure 1), when 16×16 block size is used for ME. In this section we show that MMC architecture with a perfectly balanced binary tree[§] structure (type-3) provides the highest error tolerance to SA faults.

As shown previously in Figure 1, there are several types of MMC architectures with different levels of parallelism, which can be grouped into either whole-block based (type-1,2,3) and sub-block based (type-4) structures. While whole-block based architectures allow more parallelized form, sub-block based ones reuse the architecture multiples times to perform whole block error computation. These two types of structures provide different error tolerance behavior and will be discussed separately. From here on, we omit the term "whole-block based" unless required for clarity.

Based on the fact that all MMC architectures have the same number of data buses and bit lines and the general assumption that random defects are distributed with equal probability across the ME circuit, the probability of fault occurrence is equal for all architectures. Therefore the expected value of additional error energy introduced by a single SA fault, $\mathbb{E}\left(\bar{E}|SSAF\right)$ indicates the error tolerance level of a given MMC architecture in the presence of a single SA fault. It can be represented as,

$$\mathbb{E}\left(\bar{E}|SSAF\right) = \sum_{\alpha_i \in A_k} \sum_{p_j \in P_{\alpha_i}} \bar{E}\left(\alpha_i, p_j\right) \cdot q = \sum_{\alpha_i \in A_k} \bar{E}_{\alpha_i} \cdot q = \sum_{\alpha} p\left(\alpha\right) \cdot \bar{E}_{\alpha}$$

[§]a full binary tree in which all leaves are at depth n or n-1 for some n.

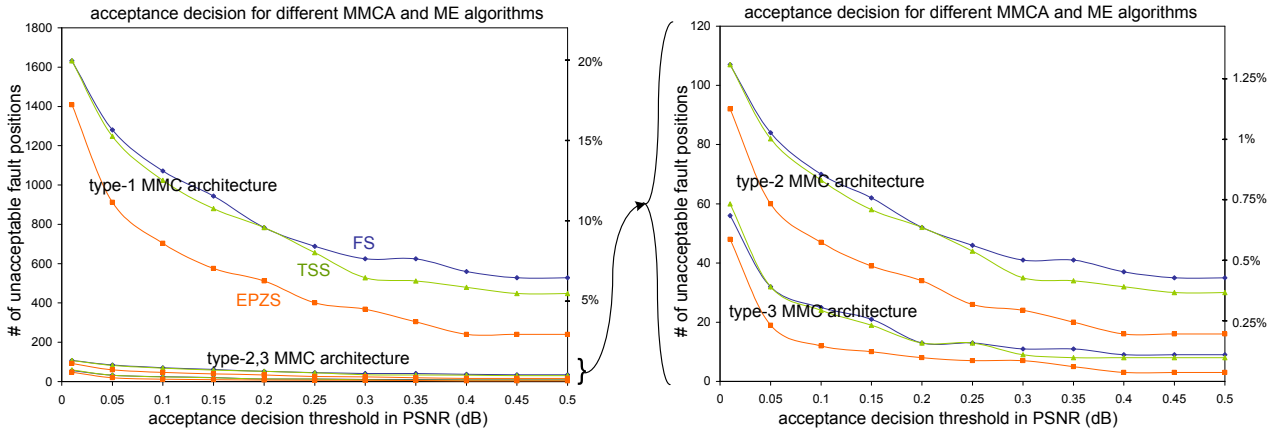

Figure 8. Experimental results on actual ET based decision example with single SA fault assumption in the context of H.264/AVC comparing different ME hardware architectures and search algorithms with respect to the percentage of unacceptable fault positions given different the decision thresholds. With a perfectly balanced tree MMC architecture (type-3), less than 1% of fault positions result in more than 0.01dB quality degradation.

where $\bar{E}_{\alpha_i} = \sum_{p_j \in P_{\alpha_i}} \bar{E}(\alpha_i, p_j)$ which is the same for all MMC architecture given α_i. A_k and P_{α_i} are the set of all data bus in MMC circuit k and of all bit lines belong to data bus α_i respectively. q is a probability of having a fault at certain position and it is equal probability over all fault locations. $p(\alpha)$ denotes the distribution of α and it is determined by the MMC architecture. Based on our fault effect model studied in Section 3, $\bar{E}$ increases linearly with the fault parameter α with an assumption that both $P_{SAD}(D)$ and $P_{minSAD}(D)$ are uniform distributions over the entire range of D. Then,

$$\arg\min_{p_k(\alpha)} \mathbb{E}\left(\bar{E}|SSAF\right) = \arg\min_{p_k(\alpha)} \sum_{\alpha} p_k(\alpha) \cdot \bar{E}_{\alpha} = \arg\min_{p_k(\alpha)} D_k$$

where $D_k = \mathcal{N} \cdot \sum_{\alpha} p_k(\alpha) \cdot \alpha = \mathcal{N} \cdot E(\alpha)$ which is equivalent to the average depth of the binary tree that corresponds to the MMC architecture k. If $p_k(\alpha)$ determined by the MMC architecture k minimizes $\mathbb{E}\left(\bar{E}|SSAF\right)$, it also minimizes the average depth of the binary tree corresponding to that MMC architecture k. Therefore the MMC architecture corresponding to the binary tree with minimum average depth[¶] (perfectly balanced binary tree) leads to the highest error tolerance and furthermore also maximize parallel computation. This conclusion holds more strongly when $\bar{E}$ increases with α superlinearly which is the case in practice as the above assumption is not generally true (Figure 7 (left)) mainly due to the $P_{minSAD}(D)$ distribution being more concentrated at the lower D such that minimum SAD becomes more likely to fall in the error region of $\Delta(D)$ function with higher α. More specifically, the relative reduction rate of $\bar{E}$ of the type-3 MMC structure compared to the others increases if the increasing rate of $\bar{E}$ vs. α is higher (e.g., exponential vs. linear increase) or if $\mathcal{N}$ increases[‖]. Figure 7 illustrate how $\bar{E}$ increases with α (left) and with average binary tree depth describing different MMC architecture (right) for different ME algorithms in actual simulation.

The same conclusion holds true for the multiple SA faults case although a formal argument establishing this property is somewhat involved, but its validity is essentially a consequence of the fact that $\bar{E}$ is an increasing function with each α_i of $\{f_i\}_{i=1}^{M}$.

Figure 8 shows how the percentage of unacceptable SSA fault locations given a MMC structure and search algorithm varies with the acceptance decision thresholds. It also provides comparison for three representative MMC architectures and search algorithms for the same process. This simulation result shows that even if

[¶]sketch of proof: a binary tree with $\mathcal{N}$ leaves in which the difference between maximum and minimum depths $D_{max} - D_{min}$ is greater than 1, can be reformed into the perfectly balanced tree by iteratively moving two leaves at D^i_{max} to D^i_{min} leaf. Each iteration i reduces average depth by $\left(D^i_{max} - D^i_{min} - 1\right)/\mathcal{N}$. Therefore perfectly balanced trees have the minimum average depth.

[‖]for example, $\bar{E}$ of type-1, 2, and 3 increases in the order of $O\left(\mathcal{N}^2\right)$, $O(\mathcal{N})$, and $O(log_2\mathcal{N})$ respectively.

acceptance decision threshold is as small as 0.01dB**, still more than 99% of fault locations produce imperceptible degradation. In other words, more than 99% of defective ME circuits with SSA fault having type-3 architecture only produce less than 0.01dB degradation. This result confirms that the MMC architecture can affect the system level error tolerance property quite significantly.

Similarly for sub-block based MMC architectures which reuses its structure L times to perform one whole block error computation, the expected error for a SA fault f_i can be represented as

$$\bar{E}_{sub} = P_{sub}(f_i) \cdot \mathbb{E}_{sub}(S_E|f_i) = \frac{1}{L} P_{wh}(f_i) \cdot (\mathbb{E}_{wh}(S_E|f_i))^L$$

Since a single SA fault in sub-block based structure has the same effect as that of L multiple independent SA faults with the same fault parameters in whole block based one, its impact increases by a factor of L. Therefore the fault impact for the sub-block based architectures increases exponentially with L, resulting in reduced error tolerance compared to the whole block based ones.

6. CONCLUSIONS

Based on the system-level error tolerance concept, this paper has proposed a model for estimating the impact of multiple hardware faults on the motion estimation process, which can be used for the ET based decision strategy of accepting a given faulty chip. Furthermore, based on this model, we investigated the error tolerance behavior of ME process in the presence of multiple hardware faults from both an algorithmic and a hardware architecture point of view by defining the characteristics of the search algorithm and hardware architecture that lead to increased error tolerance. We showed that error robustness depends on the number and quality of the candidates tested by ME algorithm but the quality primarily influences ET level. We also showed that the optimal ME hardware architecture in terms of error tolerance is perfectly balanced tree structure.

ACKNOWLEDGMENTS

This work is based upon work supported in part by the National Science Foundation under Grant No. 0428940

REFERENCES

1. I. Koren and Z. Koren, "Defect tolerant VLSI circuits: Techniques and yield analysis," in *Proceedings of the IEEE, Vol. 86*, pp. 1817–1836, (San Francisco, CA), Sept. 1998.
2. R. Karri, K. Hogstedt, and A. Orailoglu, "Computer-aided design of fault-tolerant VLSI systems," *IEEE Design & Test of Computers* **13**, pp. 88 – 96, 1996.
3. M. A. Breuer, S. K. Gupta, and T. M. Mak, "Defect and error tolerance in the presence of massive numbers of defects," *IEEE Design & Test of Comp.* **21**, pp. 216–227, 2004.
4. H. Chung and A. Ortega, "Analysis and testing for error tolerant motion estimation," in *Proc. of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT'05*, pp. 514–522, 2005.
5. I. Chong and A. Ortega, "Hardware testing for error tolerant multimedia compression based on linear transforms," in *Proc. of IEEE Int. Symp. on Defect and Fault ToleranceDFT'05*, pp. 523–534, 2005.
6. H. Cheong, I. Chong, and A. Ortega, "Computation error tolerance in motion estimation algorithms," in *Proc. of IEEE International Conference on Image Processing*, 2006.
7. H.-Y. Cheong and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc.IEEE ICME*, July 2003.
8. P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression-a survey," in *Proc. IEEE*, pp. 220–246, Feb. 1995.
9. O. Stern and H. J. Wunderlich, "Simulation results of an efficient defect analysis procedure," in *Proc. IEEE Int. Test Conf. on TEST: The Next 25 Years*, pp. 729–738, Oct. 1994.
10. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *IEEE NTC*, pp. 531–534, 1981.
11. G. Bjontegaard, "Calculation of average psnr differences between rd-curves," in *ITU-T Video Coding Experts Group, document VCEG-M33*, 2001.

**In general, 0.1-0.2dB is considered to be an imperceptible quality difference in typical image/video coding applications.