COMPUTATION ERROR TOLERANCE IN MOTION ESTIMATION ALGORITHMS

Hye-Yeon Cheong, In Suk Chong and Antonio Ortega

Signal and Image Processing Institute, Department of Electrical Engineering-Systems,
University of Southern California, Los Angeles, CA

ABSTRACT

In this paper we study the computation error tolerance properties of motion estimation algorithms. We are motivated by two scenarios where hardware systems may introduce computation errors. First, we consider hardware faults such as those arising in a typical fabrication process. Second, we consider “soft” errors due to voltage scaling, which can arise when operating at a lower voltage than specified for the system. Current practice is to discard all faulty systems. However there is an increasing interest in tools that can identify faulty systems which provide acceptable performance. We show that motion estimation (ME) algorithms exhibit significant error tolerance in these two scenarios. We propose simple error models and use these to provide insights into what features in these ME algorithms lead to increased error tolerance. Our comparison of the full search ME and a state of the art fast ME approach in the context of H.264/AVC shows that while both techniques are error tolerant, the faster algorithm is in fact more robust to computation errors.

Index Terms— Error tolerant compression, Motion estimation, Computational error tolerance, Stuck-at fault, Soft errors

1. INTRODUCTION

The progress of VLSI technology towards deep sub-micron feature sizes, e.g., sub-100 nanometer technologies, is resulting in a growing impact of hardware defects and fabrication process variability. Our work is motivated by two scenarios where this trend leads to imperfect hardware systems. For both cases we study the computation error tolerance properties of motion estimation algorithms.

First, we consider hardware defects that lead to faults at circuit interconnects. These can potentially lead to “hard” errors, since some of the functionality in the design is permanently impaired. Traditionally, systems having these kinds of faults would be discarded after testing. Defect tolerance techniques at the design and manufacturing stages have been widely studied and used in practice [1]. However, our focus is on system-level error tolerance (ET) [2], which accepts systems exhibiting errors at their outputs, as long as these result in only slight degradation in performance, e.g., in terms of coding efficiency. After appropriate testing, any imperfect system that is deemed as providing acceptable quality can be considered usable, thus increasing the effective yield of the fabrication process.

Second, we consider “soft” errors produced by voltage scaling. This second class of errors may arise when a circuit operates at a voltage lower than originally specified. Due to variability in the fabrication process, each version of a fabricated system may require a slightly different voltage to guarantee error-free operation. This can be addressed by “binning” together systems with similar voltage needs and attaching to each of these batches a different voltage specification. Our motivation in exploring error tolerance in this context is that, for certain applications, such as video coding, these systems may be able to operate at voltages below those originally specified, leading to lower power consumption (and thus longer battery life).

Note that we are not considering fault tolerance techniques which aim at achieving the same performance with a faulty system as would be achieved with a fault-free system. Instead, we consider systems that produce errors at their outputs (e.g., the motion vectors of the faulty system can be different from those of the fault-free one) and evaluate the impact of these errors on overall performance (e.g., coding efficiency). Our previous work [3] showed that certain range of deterministic faults within the motion estimation subsystem lead to acceptable quality degradation. We also proposed a novel ET based testing strategy for such a system. Other recent work [4] has shown that testing for acceptability can be done with reasonable complexity, which in some cases is comparable to that required for standard testing that aims at classifying systems into perfect and imperfect. In this paper we focus on the behavior of different ME algorithms in the presence of errors. We further extend our earlier work of [3] to consider soft errors.

The ME process comprises a strategy to search for the motion displacement offset, i.e., the motion vector (MV), and a matching cost metric computation, such as the sum of absolute differences (SAD) or the sum of squared differences (SSD). The search strategy aims at selecting a set of candidate MVs and then proceeds to compute the cost metric for the candidates. Finally, it selects the one with the minimum cost. 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.

There are several types of hardware architectures [3] for computing a ME matching metric, with different levels of parallelism. We will refer to them as matching metric computation (MMC) architectures. Most MMC architectures can be viewed as arrays of cascaded adders and represented as a binary tree graph, where each inner node represents an adder, a leaf node represents absolute difference or squared difference computation, and an edge connecting two inner nodes represent a data bus. For simplicity we use absolute difference, and therefore SAD, throughout this paper although our analysis could apply equally to other distortion metrics. Refer to [3] for a more detailed description of these architectures and associated testing.

In this paper we consider i) faults in the interconnect data bus (edge) and ii) soft errors in the adders (inner nodes) within a MMC architecture. We model interconnect faults with the single stuck-at (SSA) fault model, a well-known structural fault model which 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

This paper is based upon work supported in part by the National Science Foundation under Grant No. 0428940. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

0) or 1 (stuck-at 1). The SSA fault model covers 80-90% of the possible manufacturing defects in CMOS circuits [5], such as missing features, source-drain shorts, diffusion contaminants, and metallization shorts, oxide pinholes, etc.

With the assumption that ripple carry adders are used for SAD computation, we model the soft error related failures caused by the input voltage change to ripple carry adders. Soft errors are introduced due to deep submicron (DSM) noise and voltage scaling which cause probabilistic and input-dependent errors in multimedia systems [6, 7]. We focus on voltage scaling as it may arise in situations where energy-efficient processing is required [8].

We estimate the impact of these two types of errors on the accuracy of the metric computation, and their overall impact on ME performance. In Section 2 we briefly describe models for both stuck-at faults and soft errors. In Section 3 we will present a model for measuring the impact of both SSA faults and soft errors. This will provide some insight into the properties of ME that minimize the impact of faults. Finally, experimental results comparing several ME algorithms in terms of error tolerance will be given in Section 4.

2. MODELS FOR SSA FAULTS AND SOFT ERRORS

Faults or soft errors in a MMC architecture can lead to errors in the computed SAD values. We will refer to this as a distortion computation (DC) error and denote its magnitude as $\Delta = SAD' - SAD$ where SAD' is the computed SAD value with DC error. In general Δ is a function of the input sequence and the characteristics of the fault or error (e.g., fault location for a SSA fault or voltage input for a soft error). Note that DC errors do not necessarily lead to ME errors, which is the case where the MV selected (MV_f) is not equal to the best MV (MV_{min}). We say a block suffers a ME error if $MV_f \neq MV_{min}$. Since ME errors do not occur for all blocks, we also define and consider the block ME error rate ($P_E = prob(MV_f \neq MV_{min})$) which represents how often these ME errors occur, and the block ME error significance ($S_E = SAD_f - minSAD$, where SAD_f and $minSAD$ are the SAD values corresponding to MV_f and MV_{min} respectively). S_E represents the additional error energy in the residual block due to an error in motion estimation.

SSA faults in a MMC architecture can be fully characterized by three attributes, a) the fault type (SSA0 or SSA1), b) the bit position p of the faulty data line, which ranges from 0 to 15 depending on the MMC architecture, and c) the position of that data bus. The last attribute can be parameterized as a ratio α of the number of leaf nodes connected towards a faulty edge (data bus) to the total number of leaf nodes. Since SSA0 and SSA1 faults at the same position produce identical results in both P_E and S_E [9], only p and α have to be considered to describe a fault. If the input to a faulty position (p, α) is 0 for SSA1 (1 for SSA0), then the input is shifted by 2^p, resulting in $\Delta = 2^p$ (-2^p for SSA0). Otherwise the output SAD remains unchanged ($\Delta = 0$).

A ME error occurs in the SSA fault case if and only if, a) $\Delta = 2^p$ for MV_{min} and 0 for MV_f for SSA1 case, or $\Delta = 0$ for MV_{min} and -2^p for MV_f for SSA0 case. b) $0 < S_E \leq 2^p$. Note that P_E and S_E depend highly on the fault position (p, α) with a certain variation due to the input sequence characteristics. This is further discussed in Section 3.1.

Soft errors can be modeled using techniques proposed in [6]. A ripple carry adder operating at an input voltage (Vdd) below the normal operation range, generates soft errors [6] whose values are a function of Vdd and the adder inputs. In the context of SAD computation we assume that all adders operate under the same Vdd. Therefore, the DC error in the final SAD value, Δ, depends on the Vdd value and the intermediate values of the SAD computation, whose probability density function (pdf) can be modeled as a function of final SAD, using the methods proposed in [10]. Thus, Δ can be also represented as a function of Vdd and final SAD value. While the DC error Δ due to a SSA fault has a value of either 0 or $\pm 2^p$ such that only an input SAD in the range $[minSAD, minSAD + 2^p]$ could result in a ME error, Δ in the soft error case can take any arbitrary discrete value (in multiples of 2^{Vdd}) within the range $[-SAD, 0]$ which allows *any* input SAD value to lead to a ME error.

3. FAULT EFFECT MODELING

3.1. Stuck-at Fault Effect Modeling

To analyze the error tolerance of different ME algorithms in the presence of SSA faults, we introduce a simplified model to describe ME algorithms. We assume that ME algorithms differ in the structure of the set of candidates within which they search for the best MV. For a given block, each ME algorithm operates on a different number of candidates N, a different coverage of SAD values ΔSAD (defined as the difference between the largest SAD value in the candidate set and the minimum SAD, $minSAD$, corresponding to the best MV), and a different SAD distribution, P_{SAD}, within the range $[minSAD, minSAD + \Delta SAD]$.

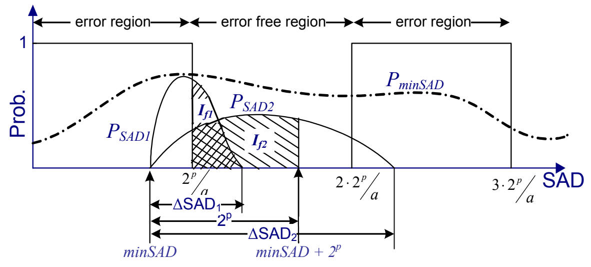

Fig. 1. Probability distribution of SADs

Figure 1 illustrates our model where *error regions* and *error-free regions* represent the range of final SAD values that will or will not suffer DC errors by a SSA1 fault with parameters (p, α). This is an approximation to the actual error behavior. It is possible for a DC error to occur for *any* final SAD value if $\alpha \neq 1$, but this is triggered only for certain combinations of intermediate SAD computations. Our simplified model is based on the assumption that α closely approximates the ratio of the intermediate SAD value accumulated up to the faulty data bus to the final SAD value (this would be a correct assumption if pixel errors followed a spatially uniform distribution.) Additional details on this model can be found in [9]. P_{minSAD} is the pdf of $minSAD$ for a given video sequence, which will be different, for example, for low and high motion scenes. Finally, we denote $P_{min,R}$ the distribution of the *minimum* SAD value in a given region R, which will be a function of the distribution of SAD values in the region R. P_{SAD1} and P_{SAD2} in Figure 1 represent two different ME algorithms, with respective SAD ranges, ΔSAD_1 and ΔSAD_2, such that $\Delta SAD_1 < \Delta SAD_2$, and $\Delta SAD_2 > 2^p$.

With this model, as shown in Figure 1, if $minSAD$ is in the *error free region* there will be no DC error, resulting in no ME error even though the system is faulty. If $minSAD$ is in the *error region* a DC error will occur. A ME error will then occur if there is a MV candidate whose SAD value is in the interval I_f corresponding to the shaded area. Since DC errors add 2^p to $minSAD$, a suboptimal motion vector MV_f will be selected instead of MV_{min} iff i) it belongs to the *error free region* and ii) its SAD, SAD_f, is such that $SAD_f < minSAD + 2^p$. We denote by I_f the interval containing SAD values meeting these two conditions. Thus $S_E = |SAD_f - minSAD| \leq \Delta SAD$ when $\Delta SAD < 2^p$, otherwise $S_E \leq 2^p$. If multiple candidates have SAD values within I_f, the one with the minimum SAD is selected.

We can represent the probability of ME error P_E for a given block, if $\Delta SAD < 2^p$, as

$$P_E = \sum_{i=1}^{\Delta SAD} \sum_{k=0}^{\infty} P_{minSAD}(\frac{(2k+1)2^p}{a} - i) \sum_{j=i+1}^{\Delta SAD} P_{min,I_f}(j),$$

which can be written as:

$$P_E = \sum_{i=1}^{\Delta SAD} \sum_{k=0}^{\infty} P_{minSAD}(\frac{(2k+1)2^p}{a} - i)(1-(\sum_{t=1}^{i} P_{SAD}(t))^N).$$

The expected value of SAD difference ($D(MV_f) - D(MV_{min})$), which can be used to predict the coding efficiency loss, can be represented as,

$$\bar{E} = \sum_{i=1}^{\Delta SAD} \sum_{k=0}^{\infty} P_{minSAD}(\frac{(2k+1)2^p}{a} - i) \sum_{j=i+1}^{\Delta SAD} P_{min,I_f}(j) \cdot (j-1)$$

which can be approximated by

$$\bar{E} \approx \sum_{i=1}^{\Delta SAD} \sum_{k=0}^{\infty} P_{minSAD}(\frac{(2k+1)2^p}{a} - i) \cdot (i-1) + c,$$

where c is a small constant which increases with N. Similar expressions can be obtained for $\Delta SAD > 2^p$. This model was experimentally verified.

Assume α and p are given and consider how $\bar{E}$ varies as a function of the ME algorithm. If $\Delta SAD > 2^p$, $\bar{E}$ is determined predominantly by 2^p. On the other hand, if $\Delta SAD < 2^p$, the error is bounded by a function that depends primarily on ΔSAD (i.e., the worst case is to replace the best match by the worst match among the candidate vectors, which introduces a ΔSAD error). Clearly p and α are both hardware related and do not depend on the ME algorithm. Thus, 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 the optimal value (i.e., ΔSAD small). Note that if ΔSAD is sufficiently small, the error introduced when $2^p < \Delta SAD$ will depend on p but will also be small. In conclusion, we should favor ME with small ΔSAD, which can (as will be verified experimentally) limit the degradation due to faults, even when p is large.

The above model provides important insight into what features of ME algorithms lead to increased error tolerance. However a better model would be needed to provide accurate estimation of the degradation in coding efficiency due to a given SSA fault. This would require additional information regarding the variation from block to block of the ME parameters (i.e. N, ΔSAD, P_{SAD}), as well as taking into consideration other coding issues. Our more recent work [9] provides such model and can be used as part of the decision strategy for accepting a given faulty chip.

3.2. Soft Error Analysis

While in the previous section we have provided a generic model for evaluating the error tolerance properties of ME algorithms given a SSA fault, for the soft error case we directly compare three representative ME algorithms for the same purpose. More specifically, the full search (FS), three step search (TSS) [11], and enhanced predictive zonal search (EPZS) [12] algorithms are considered. FS exhaustively searches all candidates within the search window, thus has the largest number of candidates (N_{FS}) and SAD range (ΔSAD_{FS}). TSS is a heuristic algorithm and can be seen as a sub-sampling of the FS SAD space. It successively evaluates sparsely distributed candidates and tries to follow the direction of minimum distortion for locating the smallest SAD. Although the number of candidates (N_{TSS}) for TSS reduces significantly, 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, adaptive thresholding, and refinements to locate the minimum distortion location. Unlike TSS, both number of candidates N_{EPZS} and the SAD range ΔSAD_{EPZS} tend to be quite small. These properties are illustrated in Figure 2, which shows the distribution of the SADs for all candidates of a given block, sorted by magnitude.

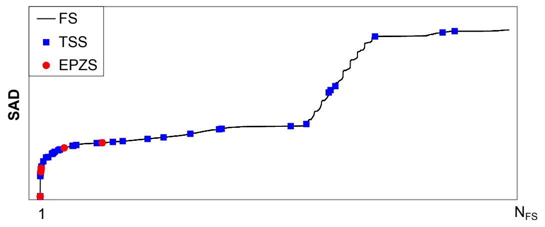

Fig. 2. Comparison of three search algorithms

Denote SAD'_i the computed SAD value with a soft error for a candidate i. Then, the overall error rate for a given ME algorithm with a number of candidates N can be represented as:

$$P_E = 1 - \prod_{k=1}^{N} Pr(minSAD' \leq SAD'_k)$$

Considering that both TSS and EPZS can be seen as subsets of FS in terms of the candidates examined, the above equation suggests that the P_E of both of these algorithms is smaller than that of FS. That is, FS is more error prone than either of these algorithms.

Let us also denote $P^i = Pr(SAD'_i < minSAD)$ and $\hat{P}^i$ as the probability of SAD'_i being the minimum among all SAD values for a given ME algorithm. We can safely assume that the probability of having multiple candidates with a computed SAD value smaller than $minSAD$ is very small, which allows us to approximate $\hat{P}^i$ as:

$$\hat{P}^i \approx Pr(SAD'_i \leq minSAD; SAD'_k \geq minSAD, \forall k \neq i)$$

Therefore, the probabilities $\hat{P}^r_{FS}$ and $\hat{P}^r_{FME}$ of a candidate r having the minimum SAD for FS and a fast ME (e.g. TSS or EPZS) respectively, resulting in a ME error of magnitude $SAD_r - minSAD$, and assuming that r is examined by both algorithms, can be computed as:

$$\hat{P}^r_{FS} \approx P^r \prod_{k=1,k\neq r}^{N_{FS}} (1-P^k) \;,\; \hat{P}^r_{FME} \approx P^r \prod_{k=1,k\neq r}^{N_{FS},k\in I_{FME}} (1-P^k)$$

where I_{FME} is the set of candidates examined by the fast ME. We observe that $\hat{P}^r_{FS} \leq \hat{P}^r_{FME}$. However, as can also be seen from Figure 2, apart from FS examining considerably more candidates than the fast ME, its candidates are also far more densely distributed. That is, the likelihood of having multiple other candidates m with SAD_m close to SAD_r for FS is high, such that $\hat{P}^m_{FS} \approx \hat{P}^r_{FS}$. Therefore the probability that any one of those candidates near r for FS has smaller SAD than $minSAD$ tends to be higher than that of the fast ME. This leads to a larger mean ME error ($\overline{E}$, thus worse coding efficiency) for FS than a fast ME. Therefore, we can draw a conclusion that for soft errors a higher N leads to greater degradation.

Soft errors are also bounded by the maximum SAD value of the ME algorithm. Although EPZS would evaluate a similar or smaller number of candidates compared to TSS, since ΔSAD_{EPZS} is usually significantly smaller than ΔSAD_{TSS}, it is expected that the error induced by using EPZS, especially for sequences where MV prediction is most effective, will be less significant.

In summary, for the soft error case, the number of candidates N is shown to be critical in determining error tolerance, i.e. a smaller N would usually lead to higher tolerance. While a smaller ΔSAD can also contribute to error tolerance to certain extent, its impact is not as significant as that of N.

4. EXPERIMENTAL RESULTS AND DISCUSSION

To evaluate our fault/soft error impact measurement models, various sequences were tested with a series of fault and soft error parameters using a H.264/MPEG-4 AVC baseline encoder and the FS, TSS, and EPZS algorithms. Only 16x16 block partitions and a single reference were considered for ME. We have used a search window of ± 32 resulting in $N_{FS} = 4225$ and $N_{TSS} = 41$. N_{EPZS}, however, varies depending on the sequence and fault characterists but on average for Foreman sequence, it was 8.8. Figures 3 and 4 depict the RD performance for the CIF resolution *Foreman* sequence for stuck-at faults and soft errors, respectively. Other sequences tested showed similar results.

Note that most SSA faults produce imperceptible degradation. In fact more than 90% of SSA fault locations lead to less than 1% bitrate increase with some variation depending on the ME and MMC architecture. To more clearly demonstrate the effect of faults in different ME algorithms we have selected fault parameters which result in significant quality degradation. Further simulation results can be found in [9]. From figure 3 it can be seen that p is the primary factor determining the level of error significance. However, for the same p, the algorithms tested demonstrate significant differences. Further evaluation of the value of ΔSAD for the three ME algorithms demonstrates that $\Delta SAD_{FS} = 1.5 \times \Delta SAD_{TSS} = 9.75 \times \Delta SAD_{EPZS}$. These numbers support our earlier conclusion that ΔSAD is the primary factor affecting robustness.

As for soft errors, the number of T_{FA} (time for one full adder operation) per clock cycle is commonly used [6] to simulate soft errors, since it directly indicates the Vdd level. Given that 16 or higher T_{FA} per cycle result in no error, we have selected values of 9 and $11T_{FA}$ in our simulations. It can be observed from figure 4 that performance of both EPZS and TSS is not very much affected by such errors. On the other hand, performance using FS is considerably degraded. This also supports our earlier conclusion that N plays a key role in determining error tolerance for soft errors.

5. CONCLUSION

In this paper we have investigated the computation error tolerance properties of ME algorithms and presented simple error models that provide insights into what features lead to increased error tolerance for both SSA faults and soft errors. Further analysis of fast ME algorithms, such as TSS and EPZS, suggests that error robustness depends on the number and quality of the candidates tested by each algorithm. However, we have observed that the number of candidates is more important for soft errors, while their quality is the primary consideration in the SSA case. This suggests different strategies to design error tolerant ME algorithms for hard or soft error scenarios.

6. REFERENCES

[1] I. Koren and Z. Koren, "Defect tolerant vlsi circuits: Techniques and yield analysis," in *Proceedings of the IEEE, Vol. 86*, San Francisco, CA, Sept. 1998, pp. 1817–1836.

[2] 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.*, vol. 21, pp. 216–227, 2004.

[3] H. Chung and A. Ortega, "Analysis and testing for error tolerant motion estimation," in *Proc. of IEEE Int. Symp. on Defect Fault Tolerance in VLSI Syst.*, 2005, pp. 514–522.

[4] Z. Jiang and S. Gupta, "An ATPG for threshold testing: Obtaining acceptable yield in future processes," in *International Test Conference*, 2002.

[5] 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*, Oct. 1994, pp. 729–738.

[6] R. Hedge and N. R. Shanbhag, "Soft digital signal processing," *IEEE Trans. on VLSI systems*, vol. 9, no. 6, 2001.

[7] K. L. Shepard and V. Narayanan, "Noise in deep submicron digital design," in *ICCAD*, Nov. 1996, pp. 524–531.

[8] D. Ernst et al, "Razor: A low-power pipelined based on circuit-level timing speculation," *MICRO-36*, Dec. 2003.

[9] H. Y. Cheong and A. Ortega, "Motion estimation performance models with application to hardware error tolerance," in *Proc. SPIE VCIP*, 2007. submitted.

[10] K. Lengwehasatit and A. Ortega, "Probabilistic partial-distance fast matching algorithms for motion estimation," *IEEE Trans. CSVT.*, vol. 11, no. 2, pp. 139–152, Feb. 2001.

[11] T. Koga et al, "Motion-compensated interframe coding for video conferencing," in *IEEE NTC*, 1981, pp. 531–534.

[12] H-Y. Cheong and A. M. Tourapis, "Fast motion estimation within the H.264 codec," in *Proc.IEEE ICME*, July 2003.

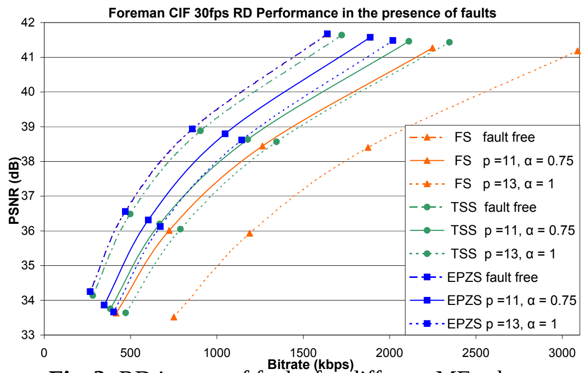

Fig. 3. RD impact of faults for different ME schemes

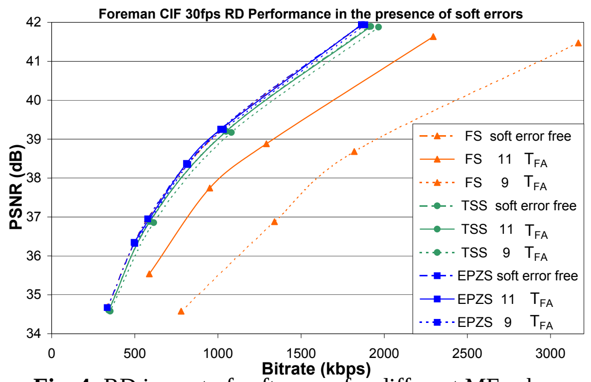

Fig. 4. RD impact of soft errors for different ME schemes