Analysis and Testing for Error Tolerant Motion Estimation *

Hyukjune Chung and Antonio Ortega
Signal and Image Processing Institute
Department of Electrical Engineering-Systems, University of Southern California
Los Angeles, California, 90089-2564
{*hyukjunc,ortega*}*@sipi.usc.edu*

Abstract

We propose a novel system-level error tolerance approach specifically targeted for multimedia compression algorithms. In particular we focus on the motion estimation process performed by most video encoders. While current manufacturing process classifies fabricated systems into two classes, namely, perfect and imperfect, our proposed scheme employs categories which are based on acceptable/unacceptable performance degradation. By enabling the use of systems that would otherwise have been discarded we seek to increase the overall yield rate in the system fabrication process. To achieve this, we propose testing algorithms that aim at determining if faults in a given chip produce acceptable performance degradation, and we propose a technique which can cancel the effect of those among the acceptable faults that can be compensated.

1 Introduction

Widespread deployment of multimedia applications is continuing to create a need for highly integrated chips which support various multimedia functionalities. Also, as technologies advance the dimension of chips tends to decrease, which leads to increases in the effects of manufacturing defects and reductions in yield rate. Low yield rates can increase the cost of chips and delay their mass production stage [6]. To address these problems, fault tolerance (FT) techniques have been proposed to provide reliable operations in the presence of faults or errors. Defect tolerance (DT) techniques enhance the yield rate by using redundancy (spares) and/or defect avoidance techniques in layout and circuit design, etc. Examples of FT schemes are [8, 2, 16, 14]. Some approaches [8, 2] are based on algorithmic level computations using redundant data. In other cases [16, 14], the algorithm itself is modified to enable fault location and correction. In these systems the goal is to compensate the effect of the faults so that a system that contains a faulty subsystem behaves *exactly* as a fault-free one. In [1] a system-level error tolerance (ET) scheme was proposed to increase effective yield. ET is a new design and test paradigm, which takes into consideration whether erroneous outputs of defective circuits produce *acceptable* results. ET classifies a chip as being *acceptable/unacceptable* by estimating the performance degradation due to faults, rather than relying solely on the conventional perfect/imperfect classification. ET analyzes the system-level effects of faults, and accepts chips if the performance degradation they lead to is within some, application-specific, ranges of acceptability.

A common characteristic of all compression standards for digital video (and indeed for images, speech, or audio) is that they rely on lossy compression, that is, the decoded video is not an exact copy of the original. Thus, in this work, we view the effect of faults as potential additional distortion

*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 author(s) and do not necessarily reflect the views of the National Science Foundation.

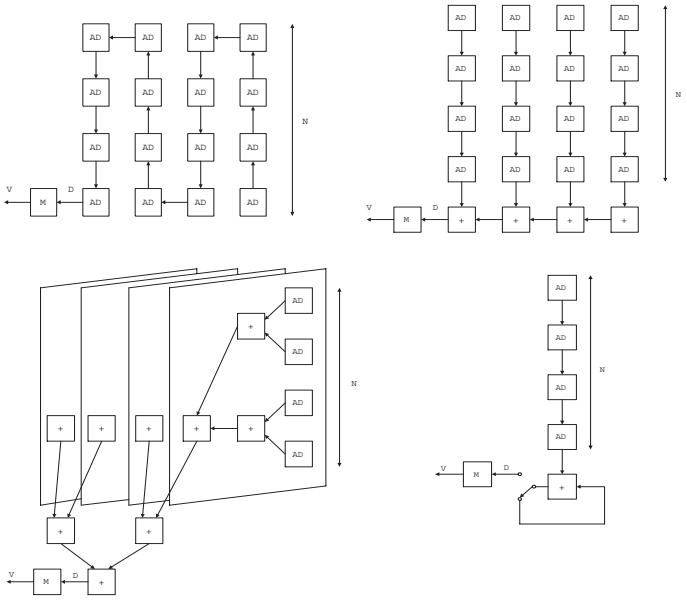

Figure 1. Examples of matching architectures for ME [13]. Only processing elements (PEs) are shown for simplicity. AD is a PE that computes the absolute difference (AD-PE) and M is a PE that computes the minimum (M-PE). Type-1: upper left, Type-2:upper right, Type-3: lower left, and Type-4: lower right.

suffered by the decoded video. This added distortion will in some cases still lead to an acceptable output. Therefore, multimedia compression systems are good applications for ET techniques. For example, our analysis of a complete video compression system [5] indicates that numerous errors at the encoder are non-catastrophic, i.e., a valid bitstream is generated and can be decoded, so that the impact of the faults can be measured in terms of video quality degradation.

In this paper, we propose an ET scheme specifically targeted for the motion estimation (ME) subsystem in video compression systems, as this is one of the most challenging subsystems in terms of both computation and memory requirements. While our focus is system-level analysis of ME techniques for acceptable degradation, testing can also be applied to lower level components of a system [10] and to other subsystems within a video encoder [3]. Our proposed schemes are aimed at separating the systems being tested into four classes, namely, (i) fault-free systems, (ii) faulty systems such that a simple programmable algorithmic level compensation can make them error-free (i.e., the output will then be the same as that of a fault-free system), (iii) faulty systems producing acceptable quality degradation, and (iv) faulty systems producing unacceptable quality degradation. Only systems in the last class will be discarded. Thus the main novelty of our approach is to identify systems in the third class, which do not produce an output identical to that of a fault-free system, but can still be used. Also, we provide a novel FT scheme for the second class. To achieve the proposed goal, we first analyze and model the system level effect of faults within the ME process (Section 2). We then propose test vector generation and testing algorithms for the proposed ET scheme (Section 3).

2 System Level Error Tolerance for Motion Estimation

For each block of size $N \times N$ pixels in the current frame (the *reference* macroblock X), ME seeks to find the best match among all the blocks (*candidate* macroblocks Y_i) in a search window located in the previous and/or future frames. ME algorithms include a search strategy and a matching metric computation. The search strategy (e.g., full search, three-step search [11], and two-dimensional logarithmic search [9]) decides the set of candidate blocks to be tested and the order in which their

metrics should be computed. The matching metric (e.g., the sum of absolute differences (SAD) or the sum of square differences (SSD) between a candidate block and the reference block) is computed for successive candidate blocks, until a sufficiently good candidate block has been found, i.e., one whose SAD or SSD with respect to the reference block is sufficiently low. The encoder then subtracts this chosen candidate from the reference block and transmits the resulting difference data (i.e., the prediction residual) to the decoder (as long as the energy in the difference is sufficiently low this leads to better compression efficiency than sending the reference block directly.)

It is important to note that even if the encoder does not choose the "best" candidate block (e.g., the one having the lowest possible SAD among all the candidates for a given reference block), encoding and decoding are still possible. There is a penalty in compression performance (i.e., more bits are needed to represent the residual signal when a sub-optimal candidate block is chosen) but otherwise the encoder and decoder operate normally. This is a key of observation in our analysis of motion estimation hardware: we will seek to estimate the impact of faults on the accuracy of the metric computation and hence on the quality of the candidates chosen by the encoder. Acceptable hardware faults will be such that the optimal candidate may not always be chosen, but such that, in general, sufficiently good candidates are chosen instead. It is important to note that the exact computed SAD metric is itself of little concern; what is important is the relative ranking of the candidate blocks. Thus, *certain faults in the matching metric computation result in no error in the resulting video encoding*. If the faulty systems only rarely affect the block ranking during ME, then the coding penalty is likely to be minimal.

Examples of implementation architectures with different levels of parallelism for the metric computation are illustrated by Fig. 1 [13]. While we focus our analysis on the architectures shown in Figure 1, and assume full search block matching with SAD metric, the results can be applied to other implementation structures as well.

We also make the following assumptions. First, all the outputs of AD (absolute difference) processing elements (AD-PEs) and adders are 16 bit wide. Usually ME is performed on 16×16 luminance macroblocks, where each luminance pixel is represented with 8 bits. Thus the absolute difference between two luminance values can also be represented by 8 bits. Therefore, the maximum SAD value when combining the ADs of N^2 pixels can represented using 16 bits[1]. Second, our work is focused on the interconnect faults that affect the data transfer between PEs. Therefore, we assume that the absolute difference operation and the carry generation process in an adder are error-free. These error-free processes can be achieved by well-known self checking design techniques [15, 7, 12]. Third, we assume that the faults in the interconnect between processing elements are stuck-at-0 (SF0) or stuck-at-1 (SF1) faults, which cause the given data line to produce a constant value (0/1) independent of other signal values in the circuit. Fourth, in our yield-rate analysis, we assume uniform spatial distribution of faults and use the Poisson distribution model.

2.1 Tree structured model of matching metric computation architectures

In a matching metric computation (MMC) architecture, the main component for each PE is an adder that computes the absolute pixel difference for the current pixel and adds it together with one or more partial SAD values. MMC architectures can be viewed as arrays of cascaded adders and represented as tree graphs (see Figure 2), where each node represents an adder, and edges connecting two nodes represent a data bus (i.e., a set of data lines). Tree graphs similar to that in Figure 2 can be constructed for all non-recursive architectures in Figure 1, i.e., all except Type 4. Under our assumption that numerical operations are correct, this tree model will be used to determine the errors caused at the output by faults in the data transfer between PEs.

Depending on the architecture used, the set of possible outputs of each PE is different. Here we define the *dynamic range* as the set of all possible signal values in a given data bus. Clearly, since the system computes the total SAD by adding partial SAD values, the dynamic range of nodes closer to the final output node (i.e., the root node) will tend to be larger. This can be seen in Figure

[1]This is obtained as $\lceil \log_2(N^2 \times (2^8 - 1)) \rceil$, $N = 16$, where $\lceil \cdot \rceil$ is the ceiling operator.

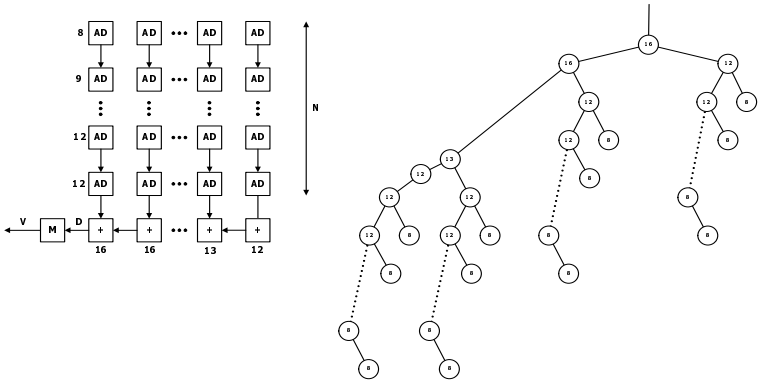

Figure 2. Dependence graph (left) of the Type-2 architecture and the corresponding tree structured flow graph (right). Next to each PE we show the number of bits needed to represent partial SADs.

2 where we include the bit-width needed to represent the output dynamic range of each node in the fault-free case. The dependence graphs and the tree graph models can be constructed in a similar way for other (non-recursive) architectures.

2.2 Fault effect modeling for the matching process

Let us first analyze a single stuck-at fault in an m-bit data bus, where data line 0 corresponds to the LSB. Let x and $\hat{x}$ denote the input and output of the data bus, respectively, and let the error in the data bus be $e = \hat{x} - x$. If there is a single SF1 fault in the p-th data line then we have that:

$$e = \begin{cases} 2^p, & 2k \cdot 2^p \le x \le (2k+1) \cdot 2^p - 1 \\ 0, & (2k+1) \cdot 2^p \le x \le (2k+2) \cdot 2^p - 1 \\ & \forall\ k = 0, 1, \ldots, 2^{m-p-1} - 1. \end{cases} \quad (1)$$

A similar relationship can be derived for the case of SF0 fault. As can be seen from (1), a single stuck-at fault causes some inputs to be shifted by a constant amount, while some other inputs remain unchanged. Thus all inputs belonging to an interval

$$I(k,p) = [k \cdot 2^p, (k+1) \cdot 2^p - 1], \quad (2)$$

are shifted by the same amount (2^p if k is even, 0 if odd). We will call $I(k,p)$ the *uniform offset intervals* for a fault at data line p.

Note that if all the inputs to a faulty data-line are shifted by the same amount (i.e., the dynamic rage of the input falls within one of these $I(k,p)$), then the ranking of values at the output remains unchanged in spite of the fault. This is illustrated by Fig. 3, where Fig. 3(a) represents the fault-free case and Fig. 3(b) shows the case when all input values are equally shifted. Conversely, when the input dynamic range is not completely enclosed in one of the $I(k,p)$, a *non-uniform offset* of the inputs is introduced (see Fig. 3(c) and (d)), which can lead to changes in the ranking at the output. Note that in the fault-free case the lower bound of all dynamic ranges is zero, but this is no longer the case if faults occur.

For the multiple fault case, the problem becomes more complex. At each individual node, the dynamic ranges are transformed by the mechanism shown in Figure 3, but the effect of multiple faults located in different data buses is not necessarily the sum of the effects of each individual fault. We denote $L_F(\eta)$ the width of the uniform offset interval at node η, e.g., $L_F(\eta) = 2^p - 1$ if p-th data line is the least significant faulty data line. If all faults present in the system lead to *uniform* shifts, then the ranking at the output will not be affected and the outcome of ME will be the same as in a fault-free system. Note that if the MMC architecture contains data lines that exceed the

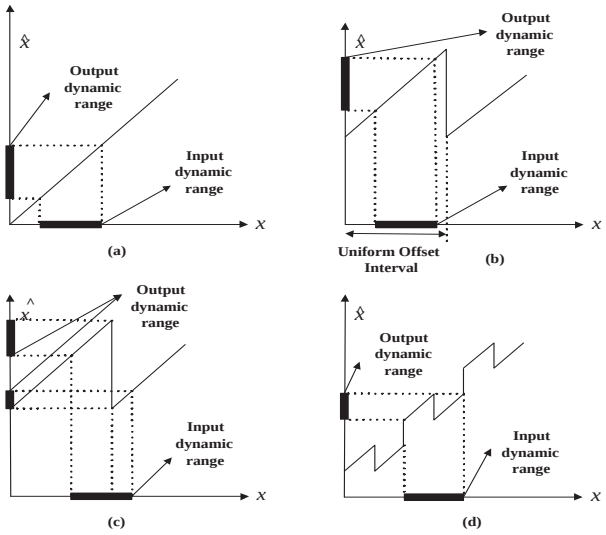

Figure 3. Dynamic range transform (a) no fault, (b) uniform shift, (c) and (d) non-uniform shifts. The faults in this example are single SF1 faults.

Table 1. Effect of SF1 faults in the cumulative adders in terms of PSNR degradation (dB). Negative entry represents PSNR degradation, and positive entry represents PSNR enhancement. These results are achieved by using MPEG-2 TM5 and the Type-2 matching process architecture with test sequences encoded at 400Kbps.

Bit/Column	0	4	8	12	15
0	0	0.007	0.002	0.011	0.006
2	0.037	0.014	0.001	0.049	-0.006
4	0.034	0.004	-0.010	0.005	-0.001
6	-0.005	-0.045	-0.021	0.009	-0.019
8	-0.007	-0.179	-0.126	-0.124	-0.234
10	0	-0.111	-1.019	-1.127	-0.879
12	0	0.014	-0.364	-2.466	-3.395
14	0	0	0	-0.001	-1.876

maximum dynamic range (for example there may be 16 data lines for an operation which is known to have only 12 bit outputs), then faults in these unused data lines do not affect the ME outcome (since clearly the dynamic range is always included within a uniform offset interval in this case). Note, that this may occur when the MMC system is designed using a single standard module for all AD operations.

To illustrate the impact on the ME output of those faults that produce non-uniform shifts as in Figs. 3(c) and (d), we simulated an MPEG-2 encoder using Type-2 MMC. In the results shown in Table 1, the same video sequence is repeatedly encoded at the same bit-rate, $400kbps$, with each table entry representing the result when a single faults affects a given PE (from rightmost, Column 0, to leftmost horizontal PE in Figure 2) and for SF1 at different bit positions. In the table, decreases in peak signal to noise ratio (PSNR) correspond to reduction in decoded video quality and are due to changes in the candidate ranking with respect to the fault free system. These results demonstrate that many faults have minimal impact on video quality[2]Since the actual degradation depends on both the video content and the type of fault, an accurate estimate of the effect of of

[2]PSNR is a widely used objective quality metric for video compression, and degradations of $0.1dB$ or less are essentially imperceptible.

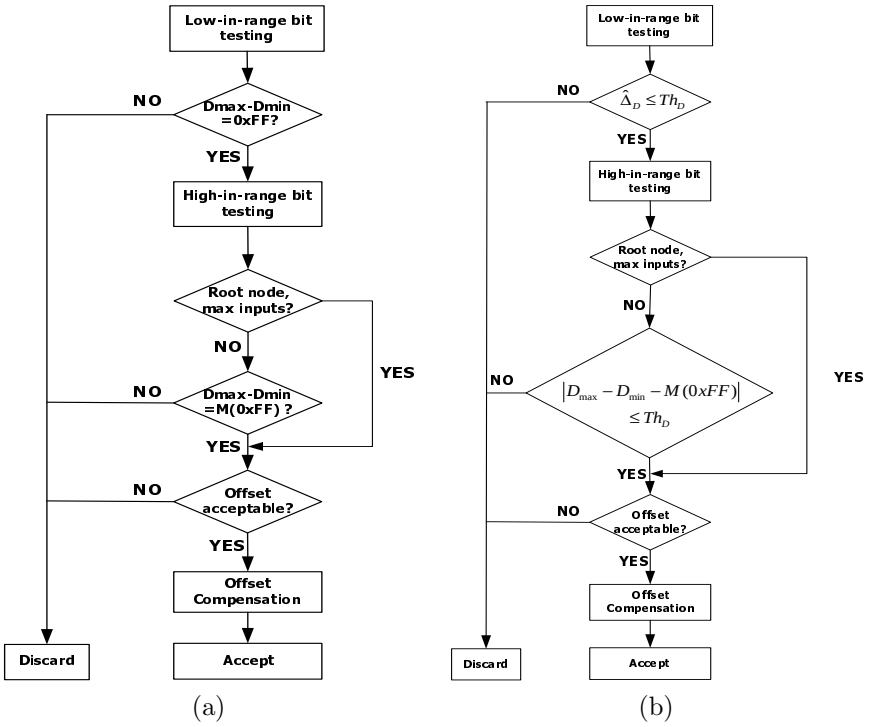

Figure 4. Schematic flow chart for the proposed testing algorithm. (a) lossless case and (b) lossy case.

a fault on the video output is not easy to obtain. Instead, we propose to use an upper-bound for Δ_D, the matching metric error, and use this to determine fault acceptability. Assume that there are multiple faults in the data path leading from a leaf node to the root node. The worst case is when all these faults affect the data in the path. Since each fault at bit-line p produces an offset 2^p, the maximum error introduced in the matching metric will be bounded as follows:

$$\Delta_D \le \sum_{\eta \in \Theta_f} \sum_{p \in B_f(\eta)} 2^p, \tag{3}$$

where Θ_f is the set of faulty nodes and $B_f(\eta)$ is the set of faulty data lines of node η. When the upper bound associated with a particular fault is relatively small (e.g., smaller than the lowest SAD values encountered during the search), then the resulting performance degradation is likely to be negligible. This will be used to design our testing algorithm.

3 Test Vector Generation and Testing Algorithm

In our testing we distinguish the *lossless case*, where all faults introduce uniform offsets, and the *lossy case*, where some non-uniform offsets occur. *In-range* faults are defined as faults occurring in data lines within the dynamic range of data being transferred. Note that even in the lossy case not all metric rankings are necessarily affected by the faults.

Our proposed testing algorithm is structured as follows. First, based on the tree graph corresponding to the MMC architecture, we compute the minimum size of the uniform offset interval for each node that guarantees lossless operation. Second, for each node, we test for potential in-range faults. Third, if there are no in-range faults, then the size of the uniform offset interval is larger than that of the input dynamic range. Therefore, input dynamic ranges overlap with at most two

Table 2. Number of tests for the proposed test and for exhaustive testing for Type-1, Type-2, and Type-3 architectures in Figure 1.

	Type-1	Type-2	Type-3
Proposed	2051	1091	512
Exhaustive	7682	5738	4606

uniform offset intervals. Thus, we need to check if the minimum and maximum outputs are shifted by the same amount. If they are, the dynamic range will be completely embedded within a single uniform offset interval. Otherwise, non-uniform offsets will occur, potentially leading to errors at the output.

Let $D_{max}(\eta)$ and $D_{min}(\eta)$ be the observed distortion values at the root node when we apply to node η the maximum and minimum inputs, respectively. Following step three above, we propose to use $D_{max}(\eta) - D_{min}(\eta)$ as the metric for the proposed testing algorithm. Note that, in contrast, most existing testing algorithms compare, for each node, the observed output values $D(\eta)$ at the root node with the expected output values, and if those are different, then mark the chip as a faulty circuit.

3.1 Testing algorithm: lossless case

For the lossless case, there should be no in-range faults, and the following condition should hold for each node η:

$$D_{max}(\eta) - D_{min}(\eta) = M_\eta \times (0xFF) \tag{4}$$

where M_η is the number of leaf nodes (inputs) contained in the subtree of η. This equation guarantees that the minimum and maximum outputs are contained in the same uniform offset interval. $D_{max}(\eta)$ is only observed at the root node, therefore, we need to test the data path from the parent node of η to the root node *before* we test for η. Thus our testing algorithm operates in a *top-down* manner, where the top is the root node, and the bottom contains the leaf-nodes. More specifically, the least significant k bit data lines of a given node are tested after all the data lines connecting these k bit lines to the root node need have been tested.

In Figure 4 (a), we show a flow chart of the proposed testing algorithm for the lossless case. "Low-in-range" bits are the 8 least significant bits, which are enough to represent the dynamic range of leaf nodes. "Hig- in-range" bits are those bit-lines above the 8th most significant one that are within the dynamic range of the node. We first test low-in-range bits (low 8 bit) in top-down manner to ensure that all the low 8 bit data path from the leaf nodes to the root node are fault-free. Low-in-range bits are needed to excite high-in-range bits.

Testing for high-in-range bits for each node is performed from lower significance to higher significance bits and also in a top-down manner. To excite $n = p$ bit ($n = 0$ is LSB) for a node, $\alpha(p)$ consecutive inputs in one subbranch of the node are excited to the maximum value $0xFF$, where $\alpha(p) = \lceil \frac{2^p}{2^8-1} \rceil$, and $\lceil \cdot \rceil$ is the ceiling operation. A detailed algorithm description and a pseudo-code implementation can be found in [4].

An input vector is composed of $0x$00s and $0xFF$s because the proposed testing metric is $D_{max}(\eta) - D_{min}(\eta)$. For low-in-range bit testing, from the top node to leaf-nodes, a single node is excited to the maximum input($0xFF$) in top-down manner. Because a single input is excited, the expected output is $0xFF$. For high-in-range bit testing, the test input sequence depends on how we traverse nodes and data buses for the tree graph model during testing. The expected response is determined by the expected size of the dynamic range of each node, which is determined by the number of excited leaf-nodes $\alpha(p)$ for each step.

In Table 2, we show the number of tests for the proposed testing and exhaustive testing for Type-1, Type-2, and Type-3 architectures in Figure 1. In this table the exhaustive testing method is not based on an analysis of the ME system and therefore, it views the given system as a black box, and checks for the correctness of each bit, so that the number of tests is proportional to

Table 3. The percentage of acceptable faults by the proposed error tolerance scheme for Type-1, Type-2, and Type-3 architecture in Figure 1. SSF: single stuck-at fault, DSF: double stuck-at fault.

	Type-1	Type-2	Type-3
$SSF_{Lossless}$	6.23%	34.58%	43.86%
$DSF_{Lossless}$	0.36%	8.54%	19.13%
SSF_{Lossy}	9.16%	56.06%	75.27%
DSF_{Lossy}	0.42%	10.84%	36.77%

the number of data lines which can be excited separately. Clearly, our proposed technique can be implemented with limited test complexity, as compared to exhaustive testing. Moreover, for the proposed testing algorithm, all inputs applied during testing are either the minimum ($0x00$) or maximum ($0xFF$) value, so that we can represent each node test input with a single bit, and then generate the corresponding test by using a multiplication by $0xFF$ in the testing hardware. Therefore, our proposed testing algorithm requires small storage space for the test vectors.

3.2 Testing algorithm: lossy case

To further increase the yield-rate, we will accept low-in-range faults which result in acceptable performance degradation. If maximum increase in the metric Δ_D for a given fault is less than a threshold Th_D, this fault will be acceptable. For single fault cases, Δ_D can be estimated using the proposed testing algorithm. However, because the testing algorithm is based on the observed output at the root node, multiple faults in the same bit positions at different nodes in the subtree may not be discriminated because our low-in-range testing uses maximum and minimum inputs for each node. Denote $\delta_D(\eta)$ the maximum distortion increase in the matching metric for a node η. During the low-in-range bit test, if we detect a single fault, then the upper-bound $\widehat{\delta}_D(\eta)$ for $\delta_D(\eta)$ is estimated as:

$$\widehat{\delta}_D(\eta, p) = N_s(\eta) \times 2^p, \tag{5}$$

where $N_s(\eta)$ is the number of nodes in the subtree of η and p is the bit position of the fault. This means that the effect of the single fault is weighted by the number of sub-tree inputs. Then, the upper-bound $\widehat{\Delta}_D$ of Δ_D in (1) can be estimated as:

$$\widehat{\Delta}_D = \sum_{\eta \in \Theta_f} \sum_{p \in B_f(\eta)} \widehat{\delta}_D(\eta, p), \tag{6}$$

where Θ_f is the set of faulty nodes which are first detected through top-down testing, that is, for the same bit fault, Θ_f only includes the nodes closest to the root node. $B_f(\eta)$ is the set of faulty data lines of node η. Therefore, if $\widehat{\Delta}_D \leq Th_D$, then we can guarantee that the performance degradation is less than Th_D. In Figure 4 (b), we show the flow chart of the proposed testing algorithm for the lossy fault tolerance case. The only difference between lossless and lossy cases is the dynamic range distortion checking routine: the test vectors used are the same.

3.3 Performance of the proposed fault tolerance scheme

In this section, we evaluate the yield-rate increase achievable with our proposed error tolerance techniques. In our evaluation we assume uniform spatial distribution of faults, a Poisson distribution model, and stuck-at faults and affecting only the data buses. Table 3, shows the percentage of acceptable faults using the proposed *lossless and/or lossy* techniques. For the lossy scheme, we set the threshold TH_D as 64 which is observed to result in less than 0.1 dB degradation as can be seen from Table 1. Note that the proposed testing scheme accepts a significant portion of chips which should have been discarded otherwise. Also, one can notice that the percentage of acceptance is much higher for Type-2 & Type-3 architectures than for Type-1 architecture. This can be explained

as follows: First, due to the lack of parallelism of Type-1 architecture, there are fewer redundant data buses, therefore, the percentage of acceptable faults is lower for Type-1 architecture. By employing variable width data line, we can completely remove the redundant out-range data buses. In this case lossless error tolerance cannot be achieved. Second, for lossy error tolerance, from (6), the upper-bound of performance degradation is proportional to the number of nodes $N_s(\eta)$ in the subtree. For Type-1 architecture due to the serial connection of all nodes, $N_s(\eta)$ is larger, therefore, fewer in-range faults are accepted.

4 Conclusion

In this paper, a novel *application oriented design and test scheme* is proposed and applied to the motion estimation process used in video compression. The proposed scheme, shows promising results in terms of yield rate improvement.

References

[1] 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 Computers*, 21:216–227, May–June 2004.

[2] Y.-H. Choi and M. Malek. A fault-tolerant FFT processor. *IEEE Trans. Computers*, 37(5):617–621, May 1988.

[3] I. Chong and A. Ortega. Harware testing for error tolerance in multimedia compression based on linear transforms. In *Proc. IEEE Intl. Symp. on Defect and Fault Tolerance in VLSI Systems, DFT'05*, Monterey, CA, Oct. 2005.

[4] H. Chung. *Complexity Scalable and Robust Motion Estimation for Video Compression, Doctorate Dissertation*. University of Southern California, Los Angeles, CA, 2005.

[5] H. Chung and A. Ortega. *System Level Fault Tolerance for Motion Estimation : Technical Report USC-SIPI 354*. Signal and Image Processing Institute, University of Southern California, Los Angeles, CA, 2002.

[6] B. El-Kareh, A. Ghatalia, and A. V. S. Satya. Yield management in microelectronic manufacturing. In *Proc. Electronic Components and Technology Conference*, pages 21–24, May 1995.

[7] M. Gossel and E. S. Sogomonyan. New totally self-checking ripple and carry look-ahead adders. In *Proc. 3rd Int. On-line Testing Workshop*, pages 36–40, 1997.

[8] K.-H. Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Computers*, 33(6):518–528, Jun. 1984.

[9] J. Jain and A. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans. on Comm.*, 29(12):1799–1808, Dec. 1981.

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

[11] J. Koga, K. Iiunuma, A. Hirani, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *Proceedings of the National Telecommunications Conference*, pages G5.3.1–5.3.5, 1981.

[12] M. Nicolaidis. Efficient implementations of self-checking adders and ALUs. In *Proc. FTCS 23*, pages 586–595, 1993.

[13] P. Pirsch, N. Demassieux, and W. Gehrke. VLSI architectures for video compression-a survey. *Proc. IEEE*, 83(2):220–246, Feb. 1995.

[14] A. Roy-Chowdhury and P. Banerjee. Algorithm-based fault location and recovery for matrix computations on multiprocessor systems. *IEEE Trans. Computers*, 45(11):1239–1247, Nov. 1996.

[15] F.W. Shih. High performance self-checking adder for VLSI processor. In *Custom Integrated Circuits Conference, Proc. IEEE*, pages 15.7/1–15.7/3, May 1991.

[16] R. Sitaraman and N. K. Jha. Optimal design of checks for error detection and location in fault-tolerant multiprocessor systems. *IEEE Trans. Computers*, 42(7):780–793, Jul. 1993.