USC–SIPI REPORT #354

System Level Fault Tolerance for Motion Estimation

by

Hyukjune Chung and Antonio Ortega

July 2002

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 400
Los Angeles, CA 90089-2564 U.S.A.

Project report: System Level Fault Tolerance for Motion Estimation

USC-SIPI #354

Hyukjune Chung and Antonio Ortega

July 19, 2002

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Widespread deployment of multimedia applications is continuing to create a need for low cost chip designs that can be incorporated into all sorts of consumer devices, such as portable communicators or home appliances. This surge in the use of digital multimedia information has been enabled to a large extent by the emergence of standard algorithms for compression of speech, audio, images and video.

In this work, we will focus on digital video systems, as they are the most challenging in terms of both computation and memory requirements, but many of the results of our work will be easily applicable to other types of media (including emerging ones such as stereo and multiview video), as well as to future compression standards.

For the video compression systems, video encoder is much more complicated, and expensive than video decoder. In the past, there were few applications where customers should have video encoder, therefore, reducing the cost of video encoder did not draw much attention. However, recently, many applications like TiVo service, and mobile video telephony which require both video encoder and decoder have emerged. Therefore, it is very important to reduce the cost of video encoder. One of the main reason of expensive video encoder is that the yield rate of video encoder is very low due to the complex algorithm of video encoding. By employing fault tolerance technique, the yield rate can be increased, and the manufacturing cost can be reduced.

To overcome the effect of faults, algorithm-based fault tolerance scheme has been proposed in many literatures like [1], [2], [3], and [4]. [1] and [2] are based on the algorithmic level computations on redundant data. In [3] and [4], fault location and correction for the multiprocessor system have been suggested. In this algorithm-based fault tolerance technique, the algorithm itself is modified to detect and correct faults.

For video compression systems, not all the faults in the implementation of a video encoder are intolerable from the view point of the system level. Some faults can induce catastrophic errors, but some faults have tolerable errors. A block diagram of video encoding stages is shown in Figure 2.1. If there are faults in the motion compensation circuit, the effect is not catastrophic. Even if there are faults in the motion compensation stage, the video encoding can be done, however resulting in the increased bit rate for the output compressed video stream. In this work, we will analyze the effect of faults in the motion compensation stage of video encoding process, and propose the fault concealment technique which can compensate for the effect of some faults. Also, in this work, we will propose a testing algorithm which requires small number of test vectors.

This work is organized as follows. In Chapter 2, we introduce motion estimation implementation issues briefly. In Chapter 3, we propose the fault concealment technique and the novel testing algorithm. In Chapter 4, we describe future works.

Chapter 2

MOTION ESTIMATION

Motion compensation is the process which reduces the temporal redundancy during video encoding. Motion estimation is the process to estimate motion vectors which are used to reduce temporal redundancy in motion compensation. In this chapter, we will explain basics of motion estimation, and also, we will cover some implementation issues for motion estimation.

2.1 Motion Estimation and Motion Compensation

To reduce the temporal redundancy between frames, current frame is divided into non-overlapping $N \times N$ (usually $N = 16$) blocks (macro-blocks), then the best matches for these blocks are searched in the previous frame. A motion vector indicates the best matching block for the given macro-block. Then these motion vector information and block differences are encoded and transmitted to the decoder. The efficiency of the motion compensation depends on the accuracy (integer pel, half pel), the motion vector, and levels of search (16×16, 8×8). In Figure 2.1, the schematic diagram of a generic MPEG encoding algorithm is shown. The motion compensation and the motion estimation process is located in the feedback loop of the encoding algorithm. As one can see, a frame that is available at the decoder side is fed through the feedback loop, therefore the motion compensation uses this decoded frame as reference frame rather than the original frame. The purpose of this feedback loop is to maintain the synchronization between the encoder side and the decoder side that is required to reduce the temporal redundancy.

The motion estimation is the process to find motion vectors which correspond to the best matches. For block matching motion estimation algorithm

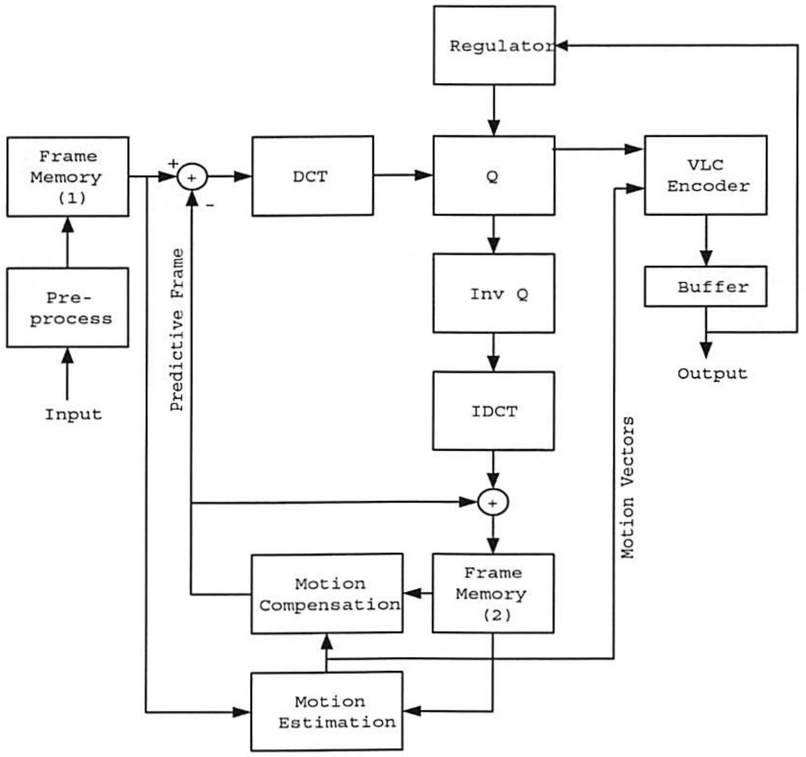

Figure 2.1: Schematic diagram of MPEG encoder

with an exhaustive search, a block of size $N \times N$(reference macro block X) of the current image is matched with all the blocks(candidate blocks Y) in the search window of size $(2w+1)^2$. The motion estimation can be described as,

$$D(m,n) = \sum_{(i,j)\in A} F(x(i,j) - y(i+m, j+n)) \tag{2.1}$$

$$v = arg \min_{(m,n)\in S} D(m,n), \tag{2.2}$$

where,

$$A = [0, N-1] \times [0, N-1],$$
$$S = [-w, w] \times [-w, w].$$

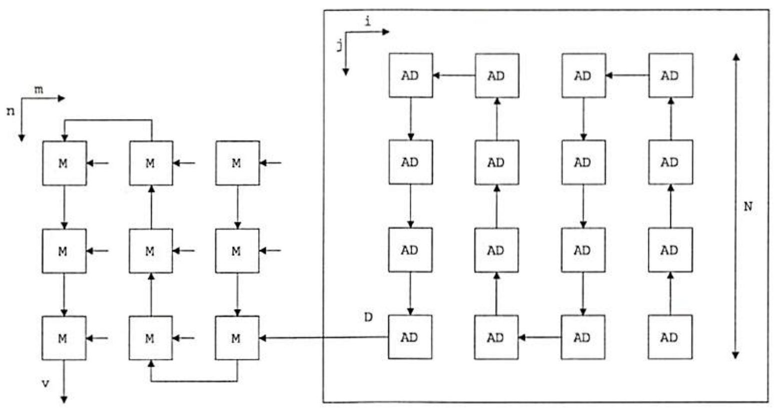

Figure 2.2: Dependence graph of a motion estimation algorithm[5].

A motion estimation algorithm is composed of a searching stage and a matching stage. The searching stage is corresponding to (2.2), and the matching stage is corresponding to (2.1). In the matching stage, if $F(\ \cdot\ )$ is the absolute function, then the matching metric is called as SAD (Sum of Absolute Difference), and if $F(\ \cdot\ )$ is the square function, then the matching metric is called as SSD (Sum of Square Difference).

2.2 Hardware Implementation of Motion Estimation stage

2.2.1 Search and matching for motion estimation

The motion estimation stage consists of matching and search processes. The matching process computes the metric between the given macro-block and candidate blocks, and keeps the minimum metric values and the corresponding position of the best match. The search process selects a subset of candidate blocks for motion estimation, and determines the order of candidate blocks in the subset of candidate blocks.

The search process supply pixel values of the candidate blocks to the matching process. Therefore search process is implemented using data busses and registers. For the given macro-block and a candidate block, a matching process calculates $N \times N$ difference values for each pixel, and then

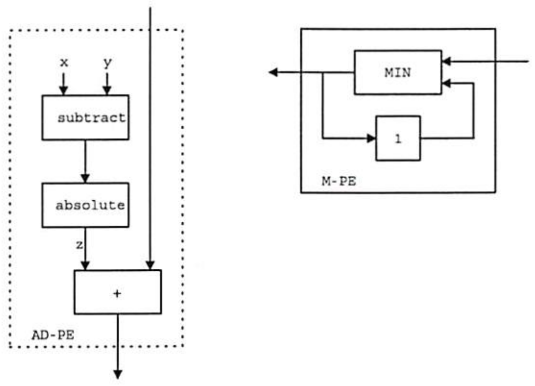

Figure 2.3: Internal Architecture of AD-PE and M-PE [5].

sums these values to find the distortion. For a practical implementation of a matching process, a metric value between the pixel of a macro-block and the pixel of a candidate block is calculated, added to the partial sum value, and then the partial sum is fed to the next PE.

The computations and data flow can be depicted using dependence graph (DG). A possible dependence graph by directly mapping the motion estimation is shown in Figure 2.2. In this figure, left side DG represents the searching process, and right side DG represents the matching process. In this figure, AD represents a processing element (PE) which contains an absolute difference and an addition, and M represents a PE for minimum value computation. A schematic diagram of AD-PE and M-PE is shown in Figure 2.3, and more detailed version of AD-PE, M-PE, and motion estimation implementation is shown in Figure 2.4.

2.2.2 Various structures to implement the matching process

Due to the computational nature of a matching process, a matching process can be implemented using various computational architectures as can be seen from Figure 2.5. In this figure, architecture type-1 is implemented using sequential computation. For architecture type-2 and type-3 more parallel computation is used than type-1. For architecture type-4, a 1-D sequential circuit is used to calculate the partial distortion for each column of data. Due to the degree of parallelism in each architecture, the required time to complete a metric computation also varies. Let us assume that each PE

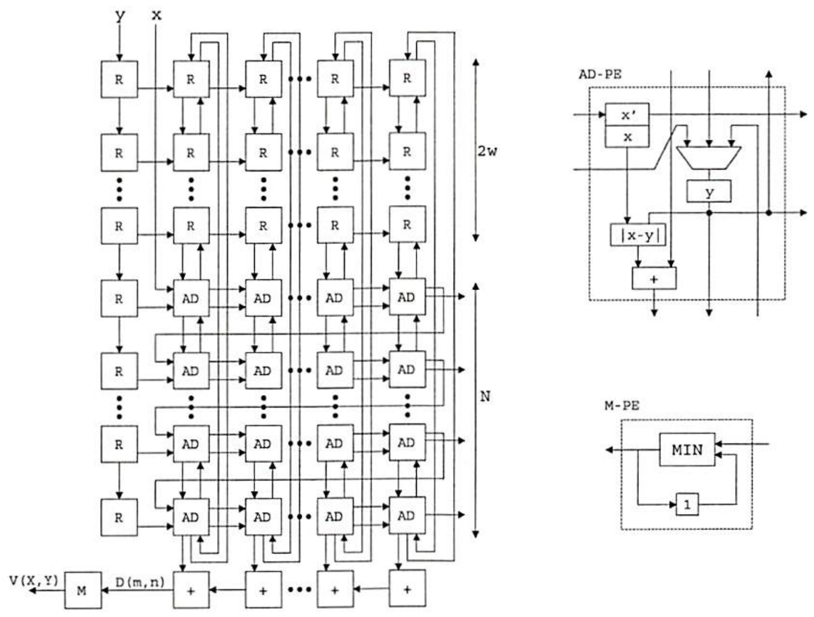

Figure 2.4: Detailed version of AD-PE, M-PE, and ME implementation[5]. R represents registers used to access pixel values.

can produce output in a cycle time. Then, architecture type-1 and type-4 requires N^2 cycles to complete a metric computation for a given macro-block and a candidate block. However, architecture type-2 requires $2N-1$ cycles, and architecture type-3 requires $\lceil 2\log_2 N \rceil$ cycles. As one can see from this figure, to implement a matching process adders are required in addition to AD-PE components, and the number of adders required are different depending on the architecture type used. To implement architecture type-1, no additional adders are required. However, to implement architecture type-3, $N^2 - 1$ additional adders are required. Likewise, type-2 requires $N-1$ adders, and type-4 requires 1 adder. The number of PE and adders, and required number of cycles to complete the metric computation is shown in Table 2.1.

We also can see the difference between these architectures by introducing a worst case fault, and then see the different effects for each architecture. Let's assume that there exist a single fault in the matching stage of a motion

Table 2.1: Number of processing elements and required cycles for each architecture type of matching process.

	type-1	type-2	type-3	type-4
AD-PE	N^2	N^2	N^2	N
M-PE	1	1	1	1
Adder	0	$N-1$	N^2-1	1
# of cycles	N^2	$2N-1$	$\lceil 2\log_2 N \rceil$	N^2

estimation algorithm, and the defective adder produces fixed output for all the input combinations. Then the effect of the single fault in the motion estimation stage can be interpreted as the dimensional reduction for the matching stage of the motion estimation algorithm. In other words, due to a single fault, (2.1) and (2.2) can be modified as follows, if $F(\ \cdot\ ) = |\ \cdot\ |$,

$$D(m,n) = \sum_{(i,j)\in A'} |x(i,j) - y(i+m, j+n)| \tag{2.3}$$

$$v = arg \min_{(m,n)\in S} D(m,n), \tag{2.4}$$

where,

$$A' \subset A,$$
$$S = [-w,w] \times [-w,w].$$

The number of the reduced dimension will be analyzed in the following. For the following analysis, it is assumed that the single fault is uniformly distributed in the given architecture.

Architecture type-1

Architecture type-1 is for the direct implementation of the 2D block matching motion estimation algorithm. The pixel difference at the current position is added to the partial sum of the differences, and then is fed into the following AD-PE. For architecture type-1, the number of adder used is N^2, when the macro-block size is $N \times N$. The average number of the dimensional

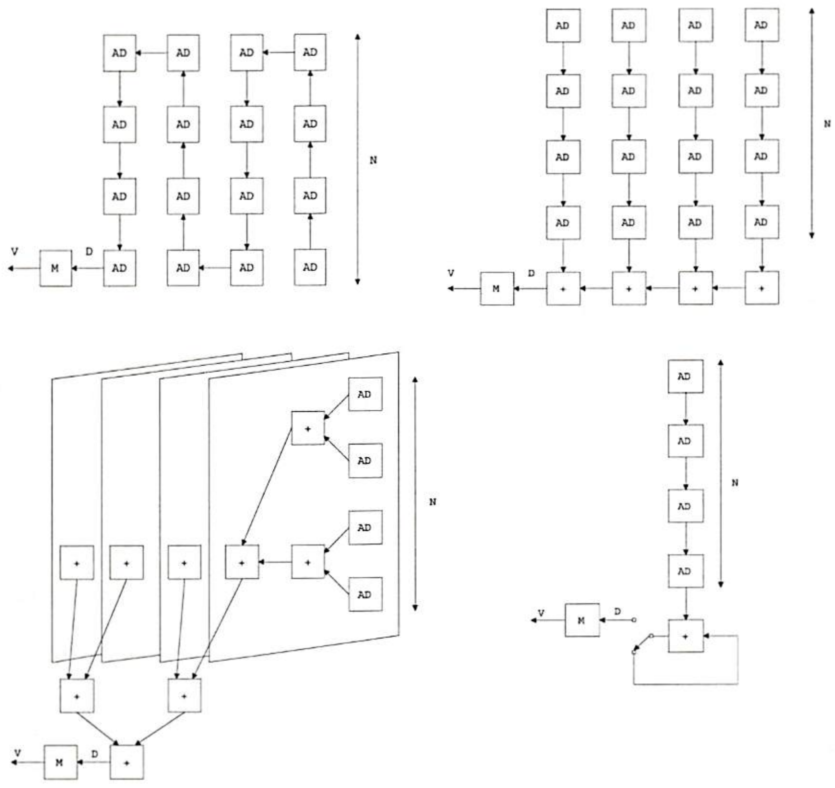

Figure 2.5: Dependence graphs for motion estimation implementation(upper left: type-1, upper right: type-2, lower left: type-3, lower right: type-4)[5]

reduction when a single fault occurs is,

$$
\begin{aligned}
&E\{\#\text{ of dimensional reduction}|\text{single fault occurs}\} \qquad (2.5)\\
&= \frac{1}{N^2}(1+2+3+\cdots+N^2)\\
&= \frac{N^2+1}{2}.
\end{aligned}
$$

Therefore,

$$\begin{aligned}&E\{\#\text{ of resulting dimension}|\text{single fault occurs}\} \qquad (2.6)\\&= N^2 - \frac{N^2+1}{2}\\&= \frac{N^2-1}{2}.\end{aligned}$$

Architecture type-2

Architecture type-2 is a slight modification of the architecture type-1. Architecture type-2 employs row-wise(or column-wise) partial summation and then calculates the total sum of the differences($D(m,n)$). Therefore by modifying Architecture type-2, it is possible to implement deterministic fast matching(DTFM) criterion. For architecture type-2, the number of adder used is N^2+N. The average number of the dimensional reduction when a single fault occurs is,

$$\begin{aligned}&E\{\#\text{ of dimensional reduction}|\text{single fault occurs}\} \qquad (2.7)\\&= \frac{1}{N^2+N}(1+2+3+\cdots+N)\cdot N\\&+ \frac{1}{N^2+N}(N+2N+\cdots+N^2)\\&= N.\end{aligned}$$

Therefore,

$$\begin{aligned}&E\{\#\text{ of resulting dimension}|\text{single fault occurs}\} \qquad (2.8)\\&= N^2 - N.\end{aligned}$$

Architecture type-3

Architecture type-3 is similar to architecture type-2 from the view point of row-wise(or column-wise) summation of the partial differences, but for architecture type-3, additions are done hierarchically, that is, an adder sums up only two results from the previous addition stage, and AD-PE does not need to contain addition operation. For architecture type-3, the number of adder used is N^2-1 , and $N=2^p$. The average number of the dimensional reduction when a single fault occurs is,

$$\begin{aligned}
&E\{\#\text{ of dimensional reduction}|\text{single fault occurs}\} \qquad (2.9)\\
&= \frac{1}{N^2-1}(2^{p-1}\cdot 2 + 2^{p-2}\cdot 2^2 + \cdots + 2^0\cdot 2^p)\cdot N\\
&+ \frac{1}{N^2-1}(2^{p-1}\cdot 2N + 2^{p-2}\cdot 2^2 N + \cdots + 2^0\cdot 2^p N)\\
&= \frac{2N^2 log_2 N}{N^2-1}\\
&\cong 2log_2 N.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&E\{\#\text{ of resulting dimension}|\text{single fault occurs}\} \qquad (2.10)\\
&= N^2 - \frac{2N^2 log_2 N}{N^2-1}\\
&\cong N^2 - 2log_2 N.
\end{aligned}$$

Architecture type-4

Architecture type-4 is an 1D implementation of the block matching motion estimation algorithm. Every row-wise(or column wise) summation is calculated using the same computational structure, and after N cycles, minimization operation is performed to find the best motion vector V. For architecture type-4, the number of adder used is $N+1$. The average number of the dimensional reduction when a single fault occurs is,

$$\begin{aligned}
&E\{\#\text{ of dimensional reduction}|\text{single fault occurs}\} \qquad (2.11)\\
&= \frac{1}{N+1}(N + 2N + \cdots + N^2) + \frac{N^2}{N+1}\\
&= \frac{N^2}{2} + \frac{N^2}{N+1}\\
&\cong \frac{N^2}{2} + N.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&E\{\#\text{ of resulting dimension}|\text{single fault occurs}\} \qquad (2.12)\\
&= \frac{N^2}{2} - \frac{N^2}{N+1}\\
&\cong \frac{N^2}{2} - N.
\end{aligned}$$

From the above analysis, one can see that the dimensional reduction effect due to a single fault is lesser for architecture type-2, and architecture type-3 than for architecture type-1, and architecture type-4. From the view point of semiconductor processing, the required silicon area for implementing architecture type-4 is smallest, but because the severe dimensional reduction is possible to occur, architecture type-4 is not recommended for implementation.

2.3 Conclusion

In this chapter, the basics of motion compensation and motion estimation are explained, and various hardware implementation architectures for motion estimation are shown and analyzed. In this work, our work will be focused on the hardware implementation issues of matching process for motion estimation. As one can see from this chapter, depending on the type of architecture used, the number of processing elements and the required number of cycles are different. Also, by using the worst case scenario, the effect of faults for each architecture is analyzed and compared with.

In the next chapter, fault effect for matching process implementation will be analyzed in more detail. Also, in the next chapter, a technique to compensate some of fault effects will be proposed.

Chapter 3

FAULTS IN MATCHING PROCESS

In the previous chapter, we showed some implementation architectures for the matching process of motion estimation. In this chapter, we will analyze the faults effect in the matching process implementation, and propose a technique to compensate for some effects of these faults. In this chapter, we will use architecture type-2 which is introduced in the previous chapter, and SAD as a matching metric to derive a technique to compensate for some effects of faults. However, the proposed technique can be easily applied to the matching process implementation which uses the other architecture types and the other metrics.

3.1 Assumptions and Basic Ideas

3.1.1 Assumptions

For the analysis, let us assume the followings. First, the pixel value is integer, and the range of the value is from 0 to 255, therefore each pixel is represented by 8 bit data. This assumption is reasonable because usually luminance component for each pixel is represented by 8 bit data, and the motion estimation is done for luminance components. Second, all the outputs of AD-PEs and adders are 16 bit wide, therefore all the adders are 16 bit adders. The adders and the corresponding input and output data lines are specified in Figure 3.4. In this figure, the width of each data line is specified. This assumption is reasonable because maximum difference input for each AD-PE can be represented by 8 bit data, therefore maximum distortion at the output of a matching process is 16 bit wide (because usually $N = 16$). That is, to implement the architecture type-2, same N^2 AD-PEs and same $N-1$ adders which have same architectural structure are used. Third, there is no fault in the absolute difference operation. This assumption is needed, because our work will be focused on adders. Fourth, the fault dealt with in this work is permanent fault like the 1-stuck fault and the 0-stuck fault.

3.1.2 Basic ideas

The purpose of the matching process is to find the best matching block for the given macro-block. In a matching stage, a metric (distortion) computation is done for N^2 macro-block pixels and N^2 candidate block pixels. Therefore if X is the set of possible pixel values, and Y is the set of possible distortion values when there is no fault, then a metric computation is the mapping D such that,

$$D: \; X^{N^2} \times X^{N^2} \to Y. \tag{3.1}$$

After the metric computation, a matching process compares the current distortion with the minimum distortion found so far. If the current block results in the smaller distortion, then a matching process updates the minimum distortion. Therefore the output of a matching process is the motion vector which corresponds to the minimum distortion value. Let us assume that there are some faults in a matching process hardware, but despite of these faults, this faulty hardware produces the motion vector which corresponds to the minimum distortion for all the combinations of inputs. Then, if we use this matching process for a video encoder, then we cannot see the effects of these faults. Now the question is "Is it possible to conceal the effects of hardware faults for a matching process implementation?". The answer is "It is possible.". Then, the remaining question is "How?". Let us define ξ as a nonlinear mapping which accounts for the effects of the faults in a matching process implementation, roughly speaking, if there exists a nonlinear mapping ϕ which satisfies

$$\phi(\xi(D(p,q))) \leq \phi(\xi(D(p',q'))) \quad \text{if and only if} \quad D(p,q) \leq D(p',q') \tag{3.2}$$
$$\forall \;\; p,q,p',q' \in X^{N^2}$$

then we can conceal the effects of the faults. To explain the basic idea clearly, let us take a simple example. In Figure 3.1, the architecture type-2 matching process is shown. In this figure, output data range of each AD-PE is shown. Let us assume that the 9th bit output data line from LSB is

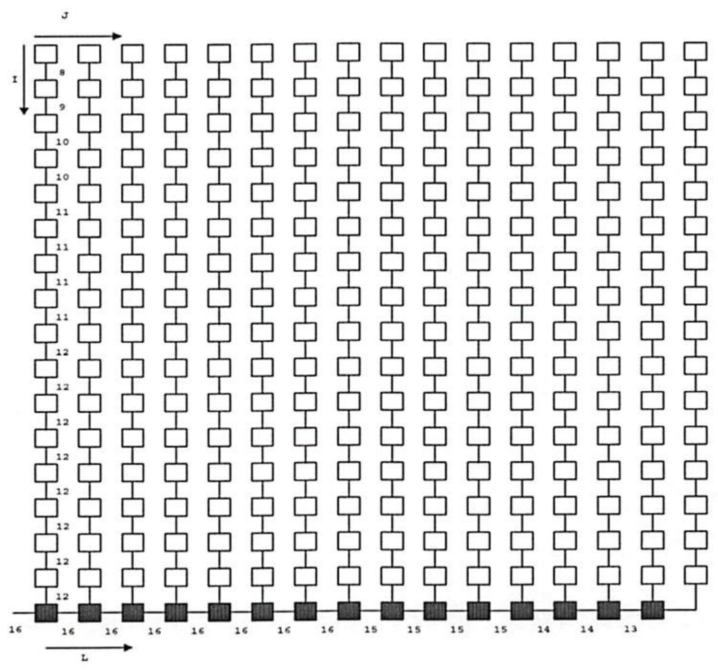

Figure 3.1: Architecture type-2 matching process. Output data range of each AD-PE is specified. Blank square is AD-PE, and shaded square is an adder. I and J are the indices for AD-PEs, and L is the index for the adders. $(1 \leq I, J \leq 16,\ 1 \leq L \leq 15)$

stuck to 1 for the AD-PE which corresponds to $I = 1$ and $J = 1$. Then, every distortion values for all the combinations of inputs will be increased by 2^8. However, this faulty matching process will satisfy the condition (3.2). Therefore, there exists a way to conceal the effect of the fault for this case.

In this chapter, we will analyze the effect of faults in a matching process implementation in detail, and propose the technique to conceal the effects of the faults. To analyze the effect of faults, we will use SAD as a matching metric, and architecture type-2 as an implementation architecture, but the results can be easily applied to the other metrics and implementation architectures.

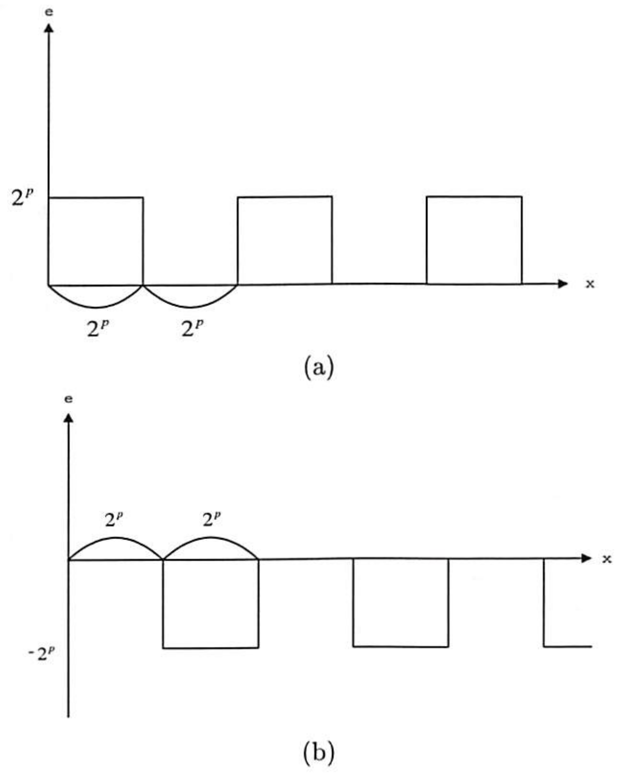

Figure 3.2: (a)Error due to the 1-stuck fault in a data line ($n = p$). (b)Error due to the 0-stuck fault in a data line ($n = p$).

3.2 Fault Effect for I/O of Adders

For an adder, faults can affect the input data lines, output data lines, and carry generation processes. The faults in the input (output) data part of an adder can be mapped to the faults in the input (output) data lines. Therefore, in this work, we will not deal with input (output) data part and input (output) data lines separately. In addition, because the adders in this problem are cascaded, the input data lines of an adder are the output data lines of the previous stage adder, therefore, we do not need to consider the input data lines and the output data lines separately. Also, because we will focus on the faults in the input and output data parts of adders, we will assume that the carry generation processes do not contain any fault.

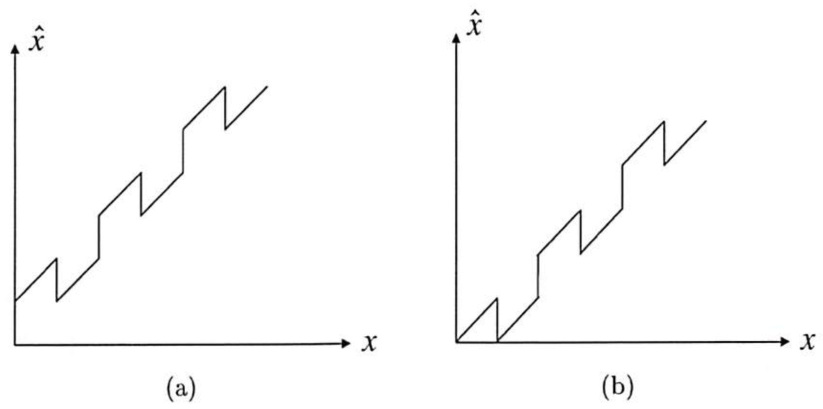

Figure 3.3: (a)Input vs. output when there is the 1-stuck fault in a data line ($n = p$). (b)Input vs. output when there is the 0-stuck fault in a data line ($n = p$).

For a bunch of m bit data lines, let us define $n = 0$ as the LSB data line, and $n = m - 1$ as the MSB data line. Also, let us define x as the input to the data lines, $\hat{x}$ as the output of the data lines. Then, the error e between x and $\hat{x}$ is $e = \hat{x} - x$. If there is a single 1-stuck fault in the $n = p$ data line, then the relationship between x and e is as follows.

$$e = \begin{cases} 2^p, & 2k \cdot 2^p \leq x \leq (2k+1) \cdot 2^p - 1 \\ 0, & (2k+1) \cdot 2^p \leq x \leq (2k+2) \cdot 2^p - 1 \end{cases} \tag{3.3}$$
$$\text{where, } k = 0, 1, \ldots, 2^{m-p-1} - 1$$

Likewise, if the single fault is 0-stuck fault, then the relationship between x and e is as follows.

$$e = \begin{cases} 0, & 2k \cdot 2^p \leq x \leq (2k+1) \cdot 2^p - 1 \\ -2^p, & (2k+1) \cdot 2^p \leq x \leq (2k+2) \cdot 2^p - 1 \end{cases} \tag{3.4}$$
$$\text{where, } k = 0, 1, \ldots, 2^{m-p-1} - 1$$

The relationship between e and x is shown in Figure 3.2, and the relationship between $\hat{x}$ and x is shown in Figure 3.3.

If there are more than one fault in a bunch of data lines, then the error will be additive, that is, if e_1 and e_2 are the error due to a single fault in a single bunch of data lines, then the total error e due to these faults is

$$e = e_1 + e_2 \ . \tag{3.5}$$

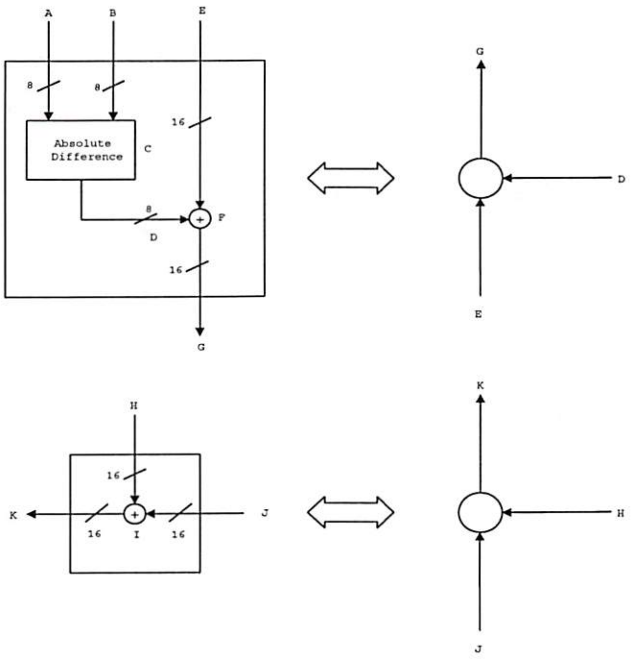

Figure 3.4: Node models for AD-PE (top) and a adder (bottom).

The fault effect of an adder can be described using the data line concept discussed above. However, we can not explain the effect of multiple faults located in different adders using above analysis. The main difference between multiple faults in a single adder and multiple faults in different adders is that the error due to multiple faults located in the different adders is not additive. Therefore, to analyze the effect of the error due to multiple faults in multiple adders, we should use different analysis approach. For this, we will use the dynamic range concept. Let us define the dynamic range in this work as follows. The dynamic range is a set of all the possible values of a signal for that part of a hardware implementation. Because the adders are cascaded for the architectures of a matching process, we can simplify a matching process implementation as a cascaded array of adders. Also, because we assumed that there are no faults in the absolute difference process, we can simplify the input parts of AD-PE as a difference input. For analysis, we will model the AD-PE and the adder as a node of a directed tree graph.

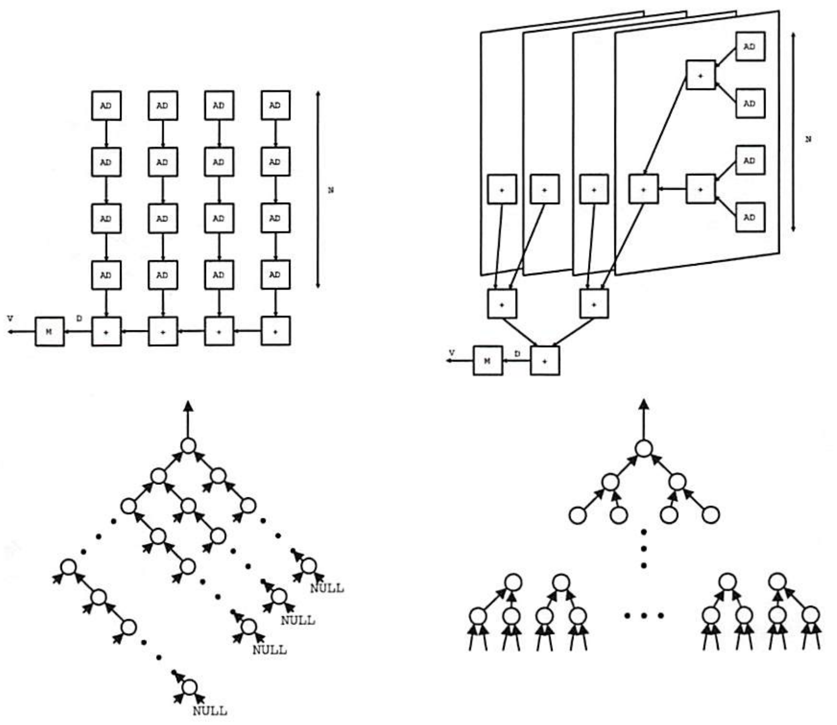

Figure 3.5: Architecture type-2 (top left) and type-3 (top right) for the matching process are shown with corresponding the tree structured flow graphs (type-2: bottom left, type-3: bottom right).

This is shown in Figure 3.4.

By this node concept, we can represent the architectures of the matching process implementation as the tree structured flow graph. The tree structured flow graph is a directed tree graph whose nodes represent AD-PEs or adders. This tree structured flow graph is useful for the analysis of multiple faults in the output of different adders. also, by using this tree structured flow graph, we can design a test vector generation algorithm for a generic matching process implementation. In Figure 3.5, architecture type-2 and type-3 for the matching process are shown with the corresponding tree structured flow graphs. Each node in the tree structured flow graph has two inputs. The inputs can be the outputs from the previous nodes or the differ-

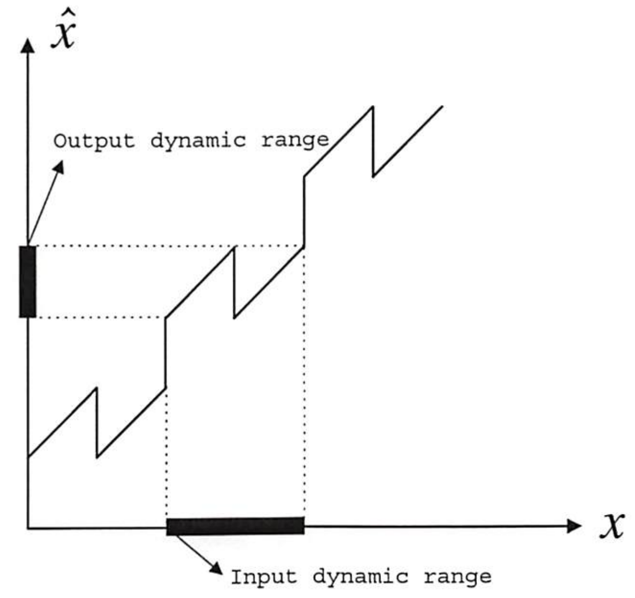

Figure 3.6: Dynamic range transform.

ence inputs for each AD-PE. The depth of tree structured flow graph varies depending on the degree of parallelism of the implementation architecture used.

Let us assume that there is a fault at the output of an adder, then the effect of the fault can be thought of as the dynamic range transform as can be seen from Figure 3.6. As one can see from this figure, the output dynamic range of this example is the shifted and shrunken version of the given input dynamic range. The output dynamic range due to this dynamic range transform is one of the followings.

Type-1 transform: Shifted version of the input dynamic range.

Type-2 transform: Nonlinearly transformed version of the input dynamic range. There exists one-to-one correspondence between the elements of the input dynamic range and that of the output dynamic range.

Type-3 transform: Shifted and shrunken version of the input dynamic range.

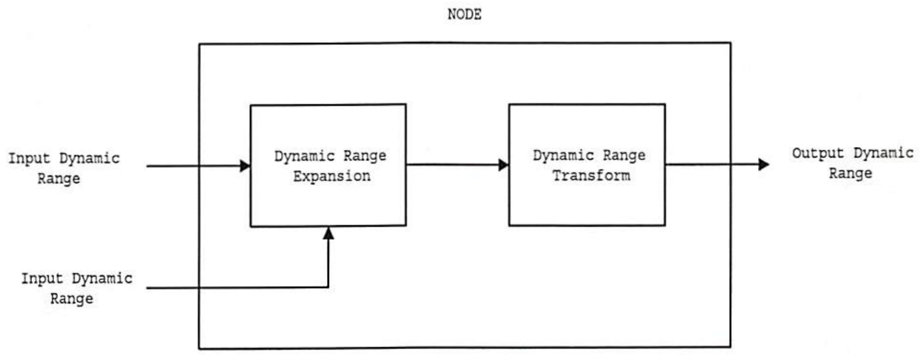

Figure 3.7: Interpretation of each node as the cascaded operations on input dynamic ranges.

Type-4 transform: Nonlinearly transformed version of the input dynamic range. There is no one-to-one correspondence between the elements of the input dynamic range and that of the output dynamic range.

For type-1 transform, we will say that the input dynamic range is preserved, and for type-3 and type-4 transforms, we will say that the input dynamic range is distorted. Type-2 transform is a special case, so, we will explain type-2 transform later. If the output dynamic range is preserved, then there exists the one-to-one mapping between the sum of the input values and the output values. Therefore, if the output dynamic range is preserved, then we can retrieve the true value of the output because there is one-to-one correspondence between the real metric values and the distorted metric values. However, if the output dynamic range is distorted, then we can not retrieve the true output value from the distorted one.

Now, let us think of the operation of an adder. An adder has two inputs, and produces the sum of these two inputs. Each of the input has its own dynamic range. The addition of two inputs has the effect of expanding the dynamic range. Therefore, each node can be thought of as the cascaded operations on input dynamic ranges. This can be seen from Figure 3.7. From the view point of the effect of faults, faults in the carry generation processes have the effects on the dynamic range expansion, and the faults in the output data part of an adder have the effects of the dynamic range transform. Therefore, from the leaf nodes to a root node of the tree structured flow graph, the input dynamic ranges of the leaf nodes are expanded

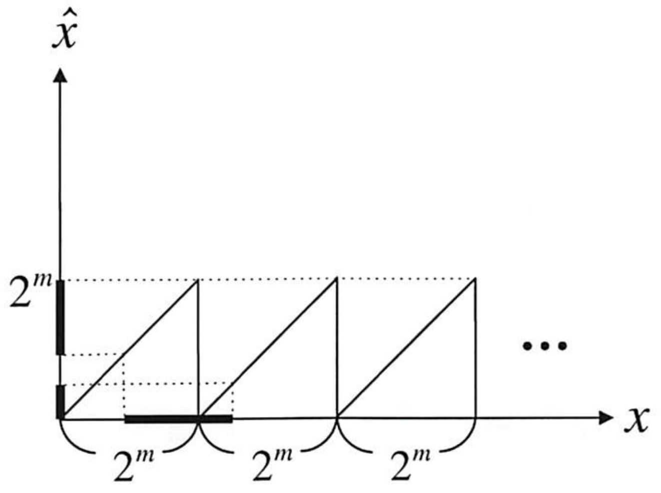

Figure 3.8: Input & output relationship due to the finite bit width of an output.

and transformed.

To use the dynamic range concept introduced above for our problem, we should consider the finite bit width of each output. Because the bit width of each output is finite, we can model each output as the data lines whose high bit data lines ($n \geq m$, where m is the bit width of the output, and n is the bit position index) are fixed to zero. The effect of the finite bit width is shown in Figure 3.8. This transform corresponds to Type-2 or Type-4 dynamic range transform.

In this section, we analyzed the effect of multiple faults for a matching process implementation by employing the dynamic range concept. In the next section, we will propose a method to conceal the fault effects by using the dynamic range concept.

3.3 Fault Concealment Technique

Fault effect concealment technique can be thought of as a lossless scheme in the sense that even if there are faults in the hardware implementation, by using the concealment technique, the faulty matching process always finds the best matching block for a given macro-block. The other scheme is a lossy

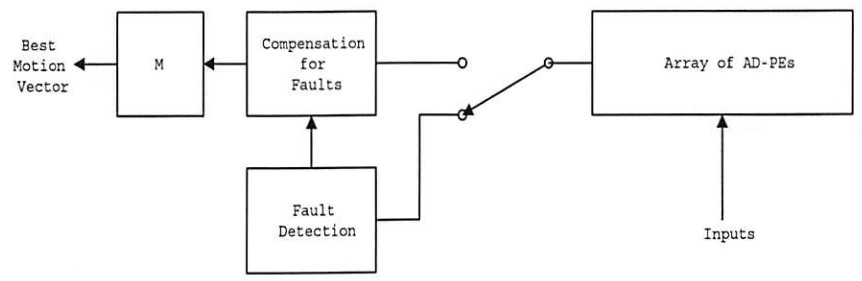

Figure 3.9: Block diagram for the proposed fault concealment technique.

scheme. In the lossy scheme, it is likely that the faulty matching process does not match all the macro-blocks with the best matching blocks. Therefore, the required bit rate is increased. However, the bit rate increase is in the acceptable range. In this work, we will deal with the lossless scheme, and we will deal with the lossy scheme in the future work.

A block diagram for the proposed fault concealment technique is shown in the Figure 3.9. Theoretically, if $\psi(\cdot)$ in (3.2) is an one-to-one mapping, then we can always find $\phi(\cdot)$ in (3.2), therefore, we can always conceal the effect of faults. However, to store $\phi(\cdot)$ in a hardware implementation, a huge storage space is needed, therefore it is practically impossible. In this work, we propose a practical fault concealment technique to conceal the effect of some faults, not all the faults whose effect can be concealed.

For the proposed fault concealment technique, the following conditions should be satisfied.

Condition-1 : All the expanded dynamic ranges at every nodes (except the root node) in the tree structured flow graph should be uniformly shifted by the dynamic range transforms.

Condition-2 : At the root node, the dynamic range transform should be the one-to-one mapping for the expanded dynamic range.

For the condition-1 to be satisfied, the expanded dynamic range should be contained in the uniform offset interval of the dynamic range transform. Because all the nodes except the root node should satisfy the condition-1, every expanded intervals should be connected, therefore, if the minimum and the maximum values of an expanded dynamic range are contained in the same uniform offset interval, then the whole expanded dynamic range is contained in the same uniform offset interval. Due to this fact, the proposed test vector generation is based on the minimum and the maximum difference inputs not all the difference inputs for each node. Due to this fact, the required number of test vectors can be greatly reduced comparing with the number of test vectors generated by all the combinations of difference inputs.

If the expanded dynamic range for the root node is $[\alpha, \alpha+1, \ldots, \alpha + 2^{16} - 2^8]$, then the output dynamic range of the root node is $[0, 1, \ldots, \alpha - 2^8] \cup [\alpha, \alpha+1, \ldots, 2^{16}-1]$. This can be seen from Figure 3.8. For the interval $[0, 1, \ldots, \alpha - 2^8]$, we can retrieve the true distortion value by adding $2^{16} - \alpha$, and for the interval $[\alpha, \alpha+1, \ldots, 2^{16}-1]$, we can retrieve the true distortion value by subtracting α. Therefore in the fault detection phase, the test circuit should detect the value of α, and α should be stored in the fault concealment circuit in Figure 3.9.

The maximum value of acceptable α depends on the used metric (SAD, SSD, etc.) and the used implementation architecture type. For the matching process implementation which employs SAD, and type-2 architecture, the maximum value of acceptable α is $2^{12}+2^8-2^4-1$, but due to the condition-1, the maximum value is 2^{12}.

In this section, we propose a technique to compensate for the effect of some faults in the implementation of a matching process. In the next section, we will propose a test vector generation algorithm for the proposed fault concealment technique.

3.4 Testing Algorithm

In the precious section, we proposed the fault concealment technique. This technique is based on the dynamic range concept. An output dynamic range is determined by an expanded dynamic range and a dynamic range transform of the given node. An expanded dynamic range is determined by the input dynamic ranges and the faults in the addition operation, specifically the faults in the carry generation operation. However, because we assumed that there is no fault in the carry generation operation, the expanded dynamic ranges are determined by the input dynamic ranges. These input dynamic ranges are output dynamic ranges of the child nodes in the tree structured flow graph. A dynamic range transform is determined by the faults in the output of adders for the given node. Therefore, to test the effect of faults at a node, we should find all the faults in the sub-tree of the given node. Let us define the inside range faults and the outside range faults. The inside faults are the faults which correspond to the dynamic range transform whose smallest uniform offset interval is smaller than the dynamic range for that part, and the other faults are defined as the outside faults. In Figure 3.1, we showed the number of bits required to represent each dynamic range. For the lossless scheme, the inside faults should be avoided, because there is no way to compensate for these faults. Therefore, the fault testing proposed in this work is composed of two phases. The first phase is to check the inside faults to ensure that at each node, the minimum uniform offset interval is greater than the expanded dynamic range of the node, and the second phase is to ensure that the minimum and the maximum elements of an expanded dynamic range are contained in the same uniform offset interval.

The first testing phase is a top-down approach. Top represents the root node, and the bottom represents the leaves in a tree structured flow graph. To check if there is any fault in the output of a given node, we should know that there are no faults in the path from the node to the root node, because the distortion value is observed from the root node. The first testing phase is composed of the lower 8 bit testing (for the case that SAD is used for a metric) and the higher bit testing. This is because that the higher bit positions are tested by using the maximum difference input (0xFF for the case that SAD is used for a metric) combinations of nodes in the sub-tree. For example, if $x(I,J)$ is the difference input of the (I,J) node, and D is the total distortion in Figure 3.1, to check the fault in the 9th bit from LSB, we should check if the 9th bit of D is '0' when $x(I,J) = 0x00,\ 1 \leq\ I,J\ \leq 16$, and then we should check if the 9th bit of D is '1' when $x(I,J) = 0xFF$, $x(I-1,J) = 0xFF$, and $x(i,j) = 0x00\ i,j \neq I,J$. Using this way, we can check the faults in the higher bit positions than lower 8bit.

The second testing phase is a bottom-up approach. This is because that an expanded dynamic range at a node is determined by the output dynamic ranges of nodes in the sub-tree of a tree structured flow graph. Because only the nodes whose minimum uniform offset interval is larger than the expanded dynamic range pass the first phase testing, in the second testing phase, if the offset values of the minimum and maximum elements of an expanded dynamic range are same, then the expanded dynamic range is contained in the uniform offset interval of the dynamic range transform, that is, every elements of the expanded dynamic range are shifted by the same amount. Let us define D_{min} as the distortion when all the difference inputs of the sub-tree of the node are $0x00$, and D_{max} as the distortion when all the difference inputs of the sub-tree of the node are $0xFF$. In the second testing phase of a node, if $D_{max} - D_{min}$ is the same as the size of the expanded dynamic range of the node, then the expanded dynamic range is contained in the uniform offset interval of the dynamic range transform for

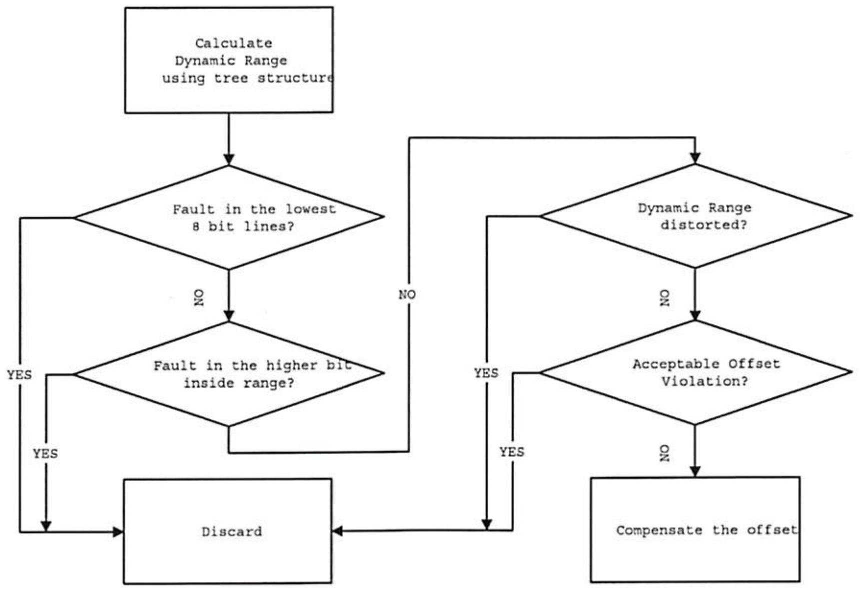

Figure 3.10: Schematic flow chart for the proposed testing algorithm.

that node. In this way, we can check the dynamic range distortions for all nodes. Also, in the second testing phase, we need to check if all the offset values are acceptable as discussed in the previous section. A schematic flow chart for the proposed testing algorithm is shown in Figure 3.10.

One of the main advantage of the proposed testing algorithm is that it can reduce the number of test vectors significantly. For example, for the case when SAD is used as a matching metric, and the implementation architecture is type-2, the proposed test algorithm requires approximately 1500 test vectors, however, the number of all the combinations of difference inputs is 2^{2048}. Also, because the proposed testing algorithm is based on the minimum ($0x00$) and the maximum ($0xFF$) difference inputs, we can represent a pair these difference inputs by single bit. Therefore, the proposed testing algorithm requires small storage space needed to store all the test vectors. For example, 96 kbit storage space is required for the example above. This is important because usually test is performed using a testing hardware, and the test vector should be generated by the testing hardware, therefore if the required storage space for test vectors are huge, then the cost of the testing increases.

3.5 Conclusion

In this chapter, we analyzed the faults effect using the dynamic range concept, and proposed the fault concealment technique and the testing algorithm. The proposed algorithm is based on a lossless scheme in the sense that the performance of fault compensated circuits are exactly same as that of the faultless circuit. The proposed algorithm is a generic algorithm by the help of the tree structured flow graph interpretation of matching process implementations.

Because the proposed algorithm is based on the minimum and the maximum difference inputs for each node, the number of test vector is very small, therefore small storage space is needed to implement a test circuit. Also, because there exists a regularity in the test vectors, we can further reduce the required storage space.

Chapter 4

FUTURE WORK

In this work, we proposed the lossless fault concealment technique. By using this technique, we can increase the yield rate to manufacture a matching process. However, if we accept some faults which slightly degrade the performance in terms of frame residual energy, then we can increase the yield rate more. This lossy scheme should be based on the statistics of the real distortion values. For example, if the average difference between the smallest distortion values and the second smallest distortion values is, for example, 50, then the effect of a fault in the LSB position of a node will be very small. Using this kind of approach, we will do the research on the lossy scheme.

In this work, we assumed that the carry generation operations are correct, that is, there is no fault. If there are faults in the carry generation operations, the dynamic range expansion operations are affected by using the concept in Figure 3.7, and there will be some loss. In [6], Soft DSP technique is proposed to compensate for the effect of faults in the carry generation operations for digital filter implementations. In the future work, we will modify the Soft DSP technique for our problem, and propose an algorithm to compensate for the effect of faults in the carry generation operations.

In this work, we assumed that there are no faults in the difference operations in AD-PEs. In the future work, we will do the research on the faults effect for this process. Also, in the future work, we will propose an algorithm to further reduce the storage space to store the test vectors by using the regularity of the proposed sequential test algorithm.

Bibliography

[1] K.-H. Huang and J. A. Abraham "Algorithm-Based Fault Tolerance for Matrix Operations," *IEEE Trans. Computers*, vol. 33, no. 6, pp. 518-528, Jun. 1984.

[2] Y.-H. Choi and M. Malek "A Fault-Tolerant FFT Processor," *IEEE Trans. Computers*, vol. 37, no. 5, pp. 617-621, May 1988.

[3] R. Sitaraman and N. K. Jha "Optimal Design of Checks for Error Detection and Location in Fault-Tolerant Multiprocessor Systems," *IEEE Trans. Computers*, vol. 42, no. 7, pp. 780-793, Jul. 1993.

[4] A. Roy-Chowdhury and P. Banerjee "Algorithm-Based Fault Location and Recovery for Matrix Computations on Multiprocessor Systems ," *IEEE Trans. Computers*, vol. 45, no. 11, pp. 1239-1247, Nov. 1996.

[5] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architectures for video compression-a survey," *Proc. IEEE*, vol. 83, no. 2, pp. 220-246, Feb. 1995.

[6] R. Hegde and N. R. Shanbhag , "Soft Digital Signal Processing," *IEEE Trans. VLSI systems*, vol. 9, no. 6, pp. 813-823, Dec. 2001.