A new circuit simplification method for error tolerant applications*

Doochul Shin and Sandeep K. Gupta

Electrical Engineering Department, University of Southern California, Los Angeles, CA 90089
{doochuls, sandeep}@usc.edu

Abstract— **Starting from a functional description or a gate level circuit, the goal of the multi-level logic optimization is to obtain a version of the circuit that implements the original function at a lower cost. For error tolerant applications - images, video, audio, graphics, and games - it is known that errors at the outputs are tolerable provided that their severities are within application-specified thresholds. In this paper, we perform application level analysis to show that significant errors at the circuit level are tolerable. Then we develop a multi-level logic synthesis algorithm for error tolerant applications that minimizes the cost of the circuit by exploiting the budget for approximations provided by error tolerance. We use circuit area as the cost metric and use a test generation algorithm to select faults that introduce errors of low severities but provide significant area reductions. Selected faults are injected to simplify the circuit for the experiments. Results show that our approach provides significant reductions in circuit area even for modest error tolerance budgets.**

Keywords- Error tolerance, circuit optimization, ATPG, DCT, redundancy removal

I. INTRODUCTION

The classical notion of yield distinguishes between perfect and imperfect chips and defines yield as the percentage of perfect chips, i.e., chips without any erroneous behavior. Since process variability increases as technology node continues to scale down, yield decreases in terms of the classical definition. The main motivation behind the concept of *error tolerance* is the fact that for some applications - images, video, audio, graphics, and games - many faulty versions of a chip can be used, provided the severities of the errors at outputs are within application-specified thresholds [1, 2]. Error tolerance is hence an increasingly attractive approach for improving yield as the classical yield decreases, since its exploitation enables us to use a fraction of imperfect chips that would have otherwise been discarded. We use the term *effective yield* to denote yield that includes perfect chips as well as imperfect-but-acceptable chips, or, simply, *acceptable chips*.

Studies in our group have been performed to define metrics that can be used to distinguish unacceptable chips from acceptable ones [3, 4]. Testing techniques have also been developed to identify acceptable chips using these metrics and corresponding application-specified thresholds. At circuit level, researchers have focused on developing new test generation algorithms that identify faults that cause errors with severities greater than a given threshold [5, 6]. At application level, people have been defined metrics for error tolerance and verified the relationship between the metric and the quality of the application level outputs [1, 2]. All of these approaches exploit error tolerance for improving yield, via new types of testing, after a chip is designed and manufactured using classical approaches.

An alternative way of exploiting error tolerance is to use the error tolerance threshold during design and synthesis of circuits to reduce cost, reduce delay, and increase yield [7, 8]. Heuristic approaches for re-designing datapath modules and synthesizing approximate logic circuits have been proposed. The key concept behind these techniques is that even though a circuit that exploits error tolerance at the design stage can produce erroneous responses, any such errors are acceptable per the given thresholds. Specifically, we perform researches to synthesize two-level approximate circuits and an approach to reduce delays of widely used datapath components, such as adders and multipliers, which are designed using full-custom architectures.

Previous works of designing and synthesizing circuit exploiting error tolerance propose a new way of exploiting error tolerance. The main difference between this research and previous researches is that in this work we are proposing a synthesis algorithm for generic multi-level circuit. So the algorithm can be applied to any type of gate level implementation of the circuit if the circuit for error tolerance applications. Our goal is to minimize the area of a given multi-level circuits that implement the original function by simplifying its design to exploit the given thresholds for error tolerance. Since the error tolerance threshold is pre-identified via application level analysis, our re-design effort and the approximate circuits it generates are both application-specific.

To measure the severities of errors for a circuit with faults, several metrics have been defined in previous works [3, 4]. **Error rate (ER)** is the percentage of vectors for which the values at a set of outputs deviate from the error free responses, during normal operation. **Error significance (ES)** for a set of outputs is defined as the maximum amount by which the numerical value at the outputs of an imperfect circuit version can deviate from the corresponding value for the perfect version. Since it has been verified that composite metric of ER and ES is a better metric for error tolerance [9], we use a composite metric of ES and ER. In particular, we define **rate-significance (RS)**. The RS value of a re-designed version of a circuit can be computed in many different ways. In this research, we will use a simplified version of RS. Let T_f denote the set of all possible tests for a fault f in the circuit. A test is a vector that if applied to the circuit's primary inputs produces an output response that is different for the circuit with the fault, compared to the response for the version without the fault. Also, let ES for each vector in T_f are $ES_1, ES_2, ..., ES_k$.

$$RS = ER*ES, \tag{1}$$

where ER = $|T_f|/2^n$ and ES = max ($ES_1, ES_2, ..., ES_k$). Our estimation of RS is conservative since the circuit might not produce max ($ES_1, ES_2, ..., ES_k$) at the rate of $|T_f|/ 2^n$.

With that as the background, we define the specific objective of this paper: **For a given original combinational circuit and an RS threshold, derive a minimum-area simplified circuit version, such that errors it produces are within the given threshold.**

This paper has six sections. In the next section, we show the existence of a substantial error tolerance budget and show that RS

* This research is supported by National Science Foundation

978-3-9810801-7-9/DATE11/©2011

is indeed an effective metric for this purpose. In Section III, we describe core idea of design simplification. Section IV shows heuristic approach that maximally reduces circuit area within the given RS threshold. Section V is for experiment results and make conclusion in Section VI.

II. Error tolerance at application level

Most error tolerant applications are the ones where the final consumers of information are humans. Since human perception of information is more at an abstract level than at the level of numerical data, some errors in the data are acceptable. Image and video applications are hence good candidates of error tolerance. In this section, we analyze error tolerance of a DCT module (discrete cosine transform; a key module in many image/video compression systems, especially JPEG and MPEG) in a video/image encoder to show the existence of significant levels of error tolerance, particularly in terms of the RS metric.

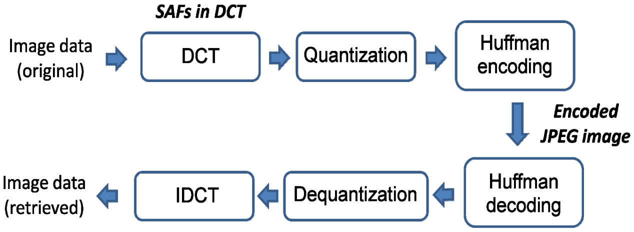


Figure 1. JPEG image compression process.

Figure 1 shows the flow of image compression and decompression using JPEG. The original image is divided into 8 x 8 blocks of image data. For each 8 x 8 image data, a DCT module separates the coefficients from low frequency to high frequency. Since human eyes are more sensitive to low frequency coefficients, image compression is accomplished by ignoring some information pertaining to the high frequency coefficients. The information regarding these high frequency coefficients is processed by the quantization module whose effectiveness is measured in terms of the compression ratio. Greater quantization provides more compression but degrades image quality. After the original matrix has been quantized, a Huffman encoder selects only the non-zero coefficients in the image data and generates an encoded image. Image retrieval from the encoded JPEG image is performed by reversing the encoding process. Due to quantization, JPEG compression naturally loses some of the information in the original image. If we have faults in the DCT module, or use some approximate designs for some datapath components, we may lose some additional information.

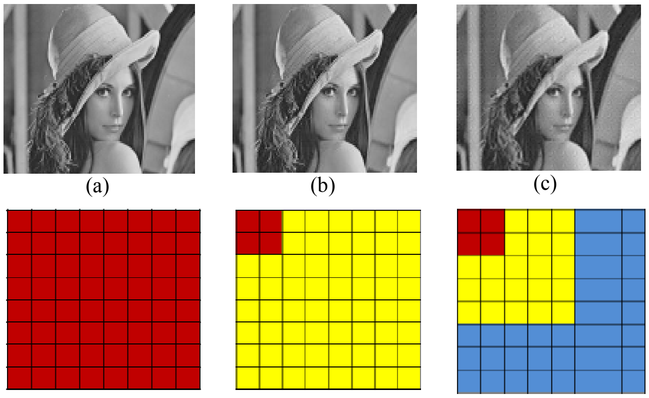
In our experiment, we assume that the quantization and the Huffman encoding modules are fault-free. We inject faults in the DCT module and examine the visual image quality as well as the *Peak Signal to Noise Ratio* (PSNR) which can be defined as

$$PSNR = 10 \cdot \log_{10}(\frac{MAX_I^2}{MSE}), \qquad (2)$$

where MAX_I is maximum possible pixel value and MSE is mean squared error between original image and compressed image. Note that, at the application level, a circuit version with a fault is equivalent to the corresponding simplified circuit version obtained as described in the next section.

In our experiment, the compression ratio is set to 90% which means that 10% of the data is compressed from the original. We use a direct 2D DCT architecture for the experiment. This architecture is faster than the more commonly used 1D transpose type DCTs; however, it uses more hardware. Inside the 2D DCT hardware, we inject faults into the adders at the final stage, just before the outputs of the DCT. For the direct 2D DCT, there are 64 input buses and 64 output buses. In Figure 2, grids below the Lena image represent this architecture of 2D DCT. Each square cell represents the output bus of an 8 x 8 block DCT.

It is well known that output values from the top-left corner cells are more sensitive to human eyes but the outputs from the cells that are far from the top-left corner are less critical. So in our study, we use perfect adders for several cells in the top-left corner and use circuits with faults in other cells. As we move farther from the top-left corner, we use adders with faults corresponding to larger RS values.



🟥 : perfect adder at the final stage
🟨 : adder with fault that RS value = 46.4
🟦 : adder with fault that RS value = 30690.0

RS(Sum) =0	RS(Sum) =2784	RS(Sum) =1197885
PSNR(dB) =31.831	PSNR(dB) =31.833	PSNR(dB) = 24.914

Figure 2. Image quality and RS value

From Figure 2, image (b), which uses circuits with faults in 60 cells, is acceptable in terms of visual image quality as well as the PSNR value. According to [10], we consider that PSNR values greater than 30dB are acceptable. However, quality of image and PSNR of (c) is not acceptable. Hence, the threshold of acceptability lies somewhere between cases (b) and (c). For any imperfect version of the 8 x 8 DCT architectures, we calculate the sum of RS value for all the faulty adders. Note that while the RS is a good metric of error tolerance for individual adders, the sum of all RS values for the DCT architectures is a weaker approximation. As we mentioned above, values from the cells close to the top-left corner have higher importance.

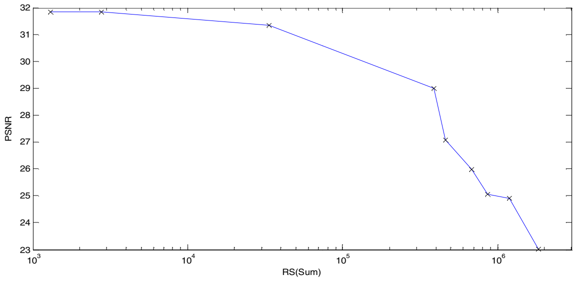


Figure 3. RS(Sum) vs. PSNR

However, finding an optimal metric is out of the scope of this paper. Instead we assume that our architecture for a given RS (Sum) is designed carefully such that among different alternatives with the same RS (Sum) value, the architecture that we study has the best image quality.

Figure 3 shows the trend of PSNR vs. RS (Sum) for above DCT for 11 different faulty versions of the DCT architecture. The figure shows that there is a clear inverse relationship between these two metrics. In other words, RS (Sum) is a useful metric if used with the condition described above. Next, we establish the practical significance of the RS (Sum) threshold value. Since RS (Sum) ≈ 10^5 when PSNR = 30, this value can be set to RS threshold. RS = 10^5 means that if we decide to simplify 60 adders like the architecture for image (b), then for each final stage adder the RS threshold = $10^5/60 = 1666.7$. This means that, assuming ER = 1 (its maximum possible value), each adder can tolerate elimination of up to 9 least significant bits of data. Since we have 27-bit adder in each final stage, this simple calculation tells us that more than 30% of the area of the final stage adders can be removed for 60 (out of 64) adders.

In summary, the above case study shows that RS is an appropriate metric for capturing the error tolerance threshold and that the RS threshold identified provides for significant circuit simplification using our approach described next.

III. Circuit Simplification by injecting faults

In this section, we describe our new approach to simplify a circuit. Our approach starts with a given original circuit, which implements the desired function accurately. Also, a given RS threshold value that captures the level of error tolerance for the desired application. Then, we identify a set of multiple stuck-at faults (SAF) and use them these to simplify the circuit to obtain a minimum-area approximate version of the original circuit that is within the given RS threshold. We start by describing our procedure for circuit simplification for a stuck-at fault.

A. Procedure of circuit simplification for a stuck at fault

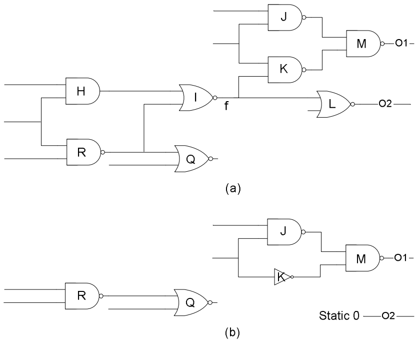


Figure 4. Design simplification with line *f* SA1.

To simplify a circuit, we inject a selected set of SAFs in the original circuit, assign a static 0 or static 1 at each line that is a site of a selected SA0 and SA1 fault, and modify the circuit to maximally exploit the static values assigned. According to the direction of the simplification, we divide our circuit simplification approach into two different procedures.

1) Backward simplification

Starting at a line, this procedure carries out simplification towards primary inputs (backward). At a line, the procedure traverses each fanin of the line. If that line is not the stem of a fanout branch system, we mark it to be deleted. From each fanin marked, we iteratively traverse all its fanins and mark it for deletion as above, until the line traversed is either a stem of a fanout or a primary input. Once the above process terminates, we remove all the traversed lines marked for deletion and the intervening gates. This procedure is independent of type of the stuck at fault (SA0 or SA1) or the types of gates encountered during traversal. In Figure 4, by injecting a fault in line f, gate I and gate H are removed by the backward simplification procedure.

2) Forward simplification

Starting at a line, this procedure carries out simplification towards primary outputs (forward). For forward simplification, the type of stuck at fault matters, since we perform forward implication of constant logic values assigned by the SAF. The exact nature of simplification at each gate traversed by logic implication also depends on the type of gate, as summarized in Table I. Once the simplification is complete for a gate, if the constant logic value at its input was the gate's controlling value, we assign an appropriate constant value at the gate's output and then perform backward simplification at all the inputs of the gate. In Figure 4, if we inject a SA1 fault at line f, gate L is removed and gate K become an inverter.

TABLE I. Forward Simplification of a gate

	Constant 0 at an input	Constant 1 at an input
NAND	Remove gate, perform forward implication with output 1; Perform backward simplification at every other input of the gate	Disconnect and remove input, stop forward implication
AND	Remove gate, perform forward implication with output 0; Perform backward simplification at every other input of the gate	Disconnect and remove input, stop forward implication
NOR	Disconnect and remove input, stop forward implication	Remove gate and perform forward implication with output 0; Perform backward simplification at every other input of the gate
OR	Disconnect and remove input, stop forward implication	Remove gate and perform forward implication with output 1; Perform backward simplification at every other input of the gate
XOR	Disconnect and remove input, change the gate to n-1 XOR gate	Disconnect and remove input, change the gate to n-1 XNOR gate

B. Problem definition of circuit simplification

Definition 1) Two different circuit networks *C* and *C'* are ***functionally equivalent*** if and only if they implement the **same** Boolean function.

For a given Boolean function, the classical problem of circuit optimization can be defined as: Among the functionally equivalent set of circuits, find the circuit that has the minimum cost. The cost typically incorporates area, delay, power consumption, and testability as the main objectives. Different cost functions lead to different optimization algorithms. The most common objective is area, which can be a good basis for subsequent reductions for minimizing power and delay. Several methods have been developed for such optimizations, including those that use algebraic/Boolean division, those that use BDDs, those that decompose and rewrite sub-circuits, and so on. One of the well studied multi-level circuit optimization is to remove redundant stuck at faults in the given circuit network *C* to minimize cost function [13, 14]. To identify redundant faults, automatic test pattern generators (ATPG) are used to generate tests for the faults. If the ATPG fails to generate any test for a target fault, then the fault is redundant, which means that for every input vector applied at the primary inputs the node will have a static constant value. For the classical redundancy removal, core problems are: (i) identify all redundant faults in the given network C, and (ii) simplify C by applying corresponding static value at each node which is the site of a redundant fault.

Considering the similarity between our approximate circuit simplification problem and the classical redundancy removal problem, we formally define our circuit simplification objective as: *Simplify a given original circuit (which implements the desired function accurately) to derive a simplified circuit with minimum area that produces errors within the given RS threshold.* We are expanding the classical redundancy removal problem, since our candidate circuits are not limited to those functionally equivalent to the original. The main objective for us is to identify candidate faults that (i) produce no errors, or (ii) produce errors but only within the given threshold. Hence, our set of candidate faults is a superset of the set of redundant faults, because in the analysis that we perform, a redundant fault is simply a candidate that has zero ES and ER values.

C. Theoretical ER and ES bounds for multiple SAFs

Since our approach deals with multiple SAFs, we now explore ways of characterizing the values of ER, ES, and RS for multiple faults, given the values of each constituent single fault. For the analysis we assume that the combinational circuit consists of primitive gates only.

Definition 2) The **five-valued system** uses the symbols 1_5, 0_5, D_5, $\bar{D}_5$, X_5 where D_5 represents that the value of the node without the fault is 1 and the value with the fault is 0, and $\bar{D}_5$ represents the opposite. X_5 represents unknown value.

We are going to denote D_5 as D and $\bar{D}_5$ as $\bar{D}$ for this paper.

Definition 3) **Faulty value** is either D or $\bar{D}$ in the five-valued system.

Definition 4) Faulty values are said to be **interacting** at a gate only when there exist faulty values at multiple inputs of the gate and the faulty values are propagated from at least two different faults.

For the maximum reduction via circuit simplification, we need to inject multiple faults in the circuit for the given error tolerance threshold. Assume that we know the ER and ES values for every single fault. This does not mean that we can compute ES and ER values for the circuit with multiple faults, because at interacting gates faulty values from different faults interact and mask the erroneous value at the gate's output. Masking at an interacting gate means that due to the multiple faulty values at the inputs of a gate, its output value becomes the same as the fault-free value. In this section, we analyze the interactions between two faults to compute upper bounds on the ER and ES values for double faults.

Definition 5) A line c_j is in **transitive fanout** of line c_i if starting from c_i one can reach c_j via a sequence of lines ($c_{\gamma 1}, c_{\gamma 2}, c_{\gamma 3}, \ldots, c_{\gamma n}$), where $c_{\gamma n}$ is fanout of $c_{\gamma n-1}$ and $c_{\gamma 1} = c_{\gamma i}$, $c_{\gamma n} = c_{\gamma j}$.

Transitive fanin can be defined in a similar manner to transitive fanout using the concept of fanin.

1) Double faults whose transitive fanouts are disjoint

Lemma 1) Consider two distinct lines c_i and c_j. Let the SAF on c_i be f_i and SAF on c_j be f_j. If the transitive fanout of c_i and the transitive fanout of c_j are disjoint, then

$$\text{abs}(ES_{ij}) \leq \text{abs}(ES_i) + \text{abs}(ES_j), \tag{3}$$

$$ER_{ij} = |T_i \cup T_j| / 2^n, \tag{4}$$

where ES_i and ER_i are error significance and error rate for the circuit with fault f_i. T_i is set of all possible test vectors for f_i, and abs(x) represents the absolute value of x.

Disjoint in transitive fanout means that structurally there is no chance for the two faults to interact with each other since there is no interacting gate. In this case, all the vectors in ($T_i \cup T_j$) are tests and they are the only tests for double fault, which leads to the above expression for ER for the double fault. Also for every vector in ($T_i \cap T_j$), the faulty values with the single fault f_i when applying vectors in ($T_i \cap T_j$) as well as faulty values with the single fault f_j when applying vectors in ($T_i \cap T_j$) will both appear at the primary outputs (POs). This leads to the above bound of ES for the double fault.

2) Upper bound of ES for double faults whose transitive fanouts are not disjoint

Definition 6) For a primary output, the **cone** is the set of all lines that are in its transitive fanin. Inputs of this cone are all the primary inputs that are a subset of the lines in the cone.

Definition 7) For a primary output, parity of a fault f_i is **odd** when f_i can produce only the faulty value D. The parity of f_i is **even** when f_i can produce only faulty value $\bar{D}$. Finally, the parity of f_i is **both** when f_i can produce D for some test vector and $\bar{D}$ for some other.

For example, in the circuit in Figure 4 (a), the parity of fault *f*-SA0 at the output O1 is **odd** and at the output O2 it is **even**. For the analysis of ES, the faulty value at each PO is important, since these values are added to calculate the total ES value for the circuit. Hence, first we focus on a single PO and later extend to multiple POs. Considering double fault f_i and f_j, we calculate an upper bound of ES for different cases for a single PO. If f_i and f_j are not in the same cone for the PO we consider, then there will be no interacting gates in the cone to change the faulty value at the PO compared to individual fault cases. For the following cases we assume that the cone of the PO that we are considering contains both f_i and f_j.

Case a) For a PO, parities of the f_i and f_j are identical but not **both**. In other words, the parities are either **even** and **even** or **odd** and **odd** for f_i and f_j.

For a primitive gate, if all the faulty values at the inputs of the gate are the same, either D or $\bar{D}$, the number of inputs that have faulty values doesn't matter in terms of the value at the gate's output. Since the parities are the same at the PO, for any interaction gate in the cone, the inputs can only have the same faulty value which will produce the same output values as in each individual fault case. In terms of the PO value, if a test vector t_v produces a faulty value at the output for a single fault, it will also produce the same faulty value at the output for the double fault.

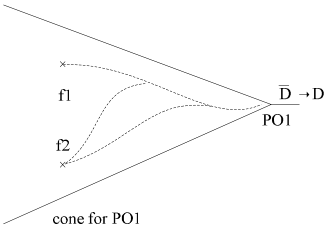


Figure 5. Two faults interact to change faulty value at PO

Case b) For a primary output, parity of double faults are different or either of the faults has a parity **both**.

In this case, an interaction gate can mask the faulty value. Due to the masking, for the circuit with double faults if a vector t_{v1} produces D at the output, it is possible for another vector t_{v2} to produce $\bar{D}$ at the output.

Definition 8) For a primary output, **weight** is numerical significance of its value.

For example, for 32 bit combinational adder, the least significant output bit has weight of 2^0, while the most significant output bit has weight of 2^{31}. Let us assume that the output weight the PO under consideration is w. For a vector t_{v1} with a fault i in the circuit, we assume that the PO under consideration has faulty value D and other outputs that have faulty value will have $\bar{D}$ only. Then, abs(ES_i) = Σ (weight of outputs with faulty value $\bar{D}$) - w. For a vector t_{v2}, we can assume a similar scenario with fault j in the circuit. In this case, we also assume that the only POs under consideration are in transitive fanout of both fault i and j. Now for some vector t_{v3} and faults i and j in the circuit, the circuit can produce $\bar{D}$ at all POs that where either D or $\bar{D}$ were produced with fault i and fault j in the above scenario. ES will be increased by $3w$ from abs(ES_i) + abs(ES_j) by removing -w from both abs(ES_i) and abs(ES_j). Also this adds +w to abs(ES_i) + abs(ES_j) .

Lemma 2) Let ES_{jk} be the error significance of the circuit with the existence of both faults f_j and f_k, and ES_j be the error significance with f_j, then for the general case,

$$\text{abs}(ES_{jk}) \leq \text{abs}(ES_j) + \text{abs}(ES_k) + 3W, \qquad (5)$$

where W is the sum of weights of the outputs where the parity of the two faults are different or where either of the two faults has the parity **both**.

3) Upper bound of ER for double faults whose transitive fanouts are not disjoint

For the ER case, if two faults are not in the same cone for all the outputs, then there are no interacting gates. So the upper bound of ER is as given by (4). Also if all primary outputs are in *Case a*) for the double fault, the upper bound is the same as (4). However, even if one of the outputs is in Case b), it is not possible to estimate ER for the double fault using only the ER values for the constituent single faults. There are two possible scenarios that can change the ER of double fault to an unexpected value. i) Some vectors in ($T_i \cup T_j$) are not tests for the double fault. ii) Some vectors not in ($T_i \cup T_j$) become tests for the double fault. Hence, for a generic circuit it is impossible to define an upper bound for ER for a double fault only in terms of the ER values of the constituent single faults. So we conclude there exist no efficient upper bounds for ER for double faults.

IV. HEURITSIC FOR MAXIMAL REDUCTION IN AREA

Due to the analysis in previous section, calculating ES and ER for all single faults in the original circuit is not sufficient to select multiple faults for simplification. ES bound can be very loose if many POs are in Case b) of previous section and ER does not have efficient upper bound. One way of maximally reducing circuit area for a given error tolerance threshold, is via an iterative method. In our iterative method, we select a fault to simplify circuit and then re-analyze ES and ER for the simplified version of the circuit. Then we repeat simplification iteratively until we reach the error tolerance threshold. At each iteration step, the fault used for simplification is chosen using a greedy heuristic.

```
//Circuit-simplify ( )
  begin
  while (error($C_S$) ≤ threshold)
      foreach single fault j in circuit calculate area FOM(j),
          find a fault f that has maximum value FOM
              Simplify circuit with fault f to produce $C_S$
  end
```

Figure 6. Greedy heuristic to simplify circuit

Figure 6 shows our heuristic to maximally reduce area. C_S is the simplified circuit by injecting a SAF to the circuit from previous iteration of the procedure. FOM(j) represents figure of merit for choosing fault j to simplify for greedy heuristic. We use FOM as (area reduction/RS) or (area reduction) and report better result between these two.

The simplified circuits that are obtained using the above heuristic is always smaller than the original circuit and always have larger values of RS than the original circuit, unless all the selected faults are redundant faults. Hence, the circuit area monotonically decreases with each iteration step. Also the time complexity of the above heuristics are O(kp) where k is the number of faults selected within the threshold for simplification and p is the total number of faults in the circuit. Since k<<p, these procedures have polynomial time complexities. A drawback of the above procedure is that it is heuristic that do not guarantee optimal area reduction. However, the experimental results presented in the next section clearly demonstrate their practical utility.

A. Estimating ER and ES for selected SAFs

To estimate ER for a given circuit, we use a parallel fault simulator. In the previous section, we have shown that for a general combinational circuit, the ER values for circuit versions with single faults cannot be used to compute the ER value for the circuit version with the corresponding double fault. Hence, when we estimate ER for multiple faults, we always compare the output values computed by the parallel simulator for the faulty circuit with the output values for the circuit without any fault. In this way, independent of whether masking at interacting gates, we can estimate the ER value accurately. Completely accurate estimation of ER requires repeating such simulation for all possible 2^n vectors where n is total number of inputs in the circuit. Since this has impractically high time complexity for most circuits, we perform parallel fault simulation for randomly selected vectors. The accuracy of the estimated ER values increases with the number of vectors simulated and the relationship between the two can be found in [15]. For nearly all circuits, we simulate 10,000 vectors as this provides sufficiently high level of accuracy.

To estimate ES, we have adapted a previously developed PODEM based ATPG for error significance testing [6]. This ATPG branches until either a lower-bound on ES is greater than a threshold; it bounds when an upper-bound on ES is lower than the threshold. This ATPG considers a vector as a test only when the vector produces ES greater than the provided threshold. To use this ATPG to estimate ES for a fault, we re-run it with different threshold values from 2^0 to 2^{m+1}, where m is the number of POs in the circuit. If the ATPG can generate a test for threshold of 2^j but not for 2^k ,where 0≤ j<k≤ m, then we consider 2^k as ES value for the fault, since the maximum possible ES that the fault can have is 2^k or less. Using this ATPG in this manner to compute ES gives us an ES value compared to the circuit on which we are running the ATPG. This means that if we run ATPG on a simplified version of the circuit obtained after several iterations of our circuit simplification procedure, then it will provide an ES value compared to that simplified version, and not the ES value compared to the original circuit. To avoid this inaccuracy, instead of running this ATPG on the simplified circuit, we run ATPG on the original circuit but with a set of multiple faults which includes all the faults selected in the previous iteration steps of our circuit simplification procedure. To enable such use of this ATPG, which was originally designed for single SAFs, we have modified it to consider multiple SAFs. This modification exploits the equivalence between multiple faults and single fault as shown in Figure 7. More details can be found in [16].

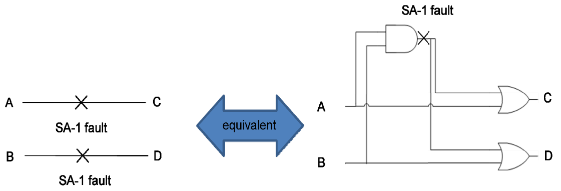


Figure 7. Equivalence between multiple faults and single fault

V. EXPERIMENTAL RESULTS

TABLE II. EXPERIMENTAL RESULT FOR DESIGN SIMPLIFICATION

c880 (Total Area :901, % datafaults 37.5)				
%RS	1	2	5	10
% area reduction	5.88	11.32	20.75	22.53

c1908 (Total Area: 1723, % datafaults 14.3)				
%RS	0.1	0.2	0.5	1
% area reduction	1.86	2.79	5.57	12.00

c3540(Total Area:3752, %datafaults 0.84)				
%RS	1	2	5	10
%area reduction	0.11	0.21	0.21	0.43

c5315(Total Area:5631, %datafaults 19.6)				
%RS	1	2	5	10
% area reduction	1.97	3.29	5.03	8.72

c7552(Total Area:7164, %datafaults 11.4)				
%RS	10^{-7}	$2*10^{-7}$	$5*10^{-7}$	$10*10^{-7}$
% area reduction	5.97	5.97	5.97	6.30

We have applied our design simplification approaches to five of the largest circuits in the ISCAS85 benchmark suite. These benchmarks are selected mainly because for each benchmark core modules are arithmetic circuits and hence belong to the domain where the notion of error tolerance is applicable [17]. In keeping with this fact, we simplify each benchmark circuit only using faults in its datapath. To accomplish this, we analyze the transitive fanins of control outputs and exclude all faults in the lines traversed from our candidate faults list. Note that the faults in transitive fanin of both a control and a data output are excluded from our list of candidates .

Each data output has a different weight and typically the weight increases exponentially and so does the maximum RS value. Since we are representing %RS value as percentage of the maximum RS, for circuits with very wide output data busses, %RS values in the range of 1% entail massive simplification. In such cases, particularly C7552, we pick a smaller limit on %RS.

The results of our simplifications presented in Table II clearly demonstrate the effectiveness of our approach. Note that significant area reductions are possible even for relatively low RS threshold values. (The only exception is C3540, where only 0.84% of its lines are datapath lines. In other words, our simplification procedure excludes itself from performing any simplification at 99.16% of its lines.)

V. CONCLUSION

In this paper, we present a new design simplification approach for error tolerant applications. First we analyze the architecture of a JPEG encoder and inject faults in its DCT module and demonstrate that a high error tolerance budget exists and that RS is a good metric for error tolerance. Then we develop a new heuristic approach to maximally reduce circuit area for a given RS threshold. Along the way, we derive important theoretical properties associated with ER and ES for multiple faults, modify an existing ES ATPG to consider multiple faults, and integrate these to implement our new circuit simplification procedures. Experimental results on the largest of ISCAS85 benchmark circuits clearly demonstrate the significant levels of simplifications that our new approach can provide.

VI. REFERENCES

[1] H. Chung and A. Ortega, “Analysis and testing for error tolerant motion estimation,” In *Proc. Defect and Fault Tolerance conference*, 2005, pp. 514- 522.

[2] I. S. Chong and A. Ortega, “Hardware testing for error tolerant multimedia compression based on linear transforms,” In *Proc. Defect and Fault Tolerance Conference*, 2005, pp. 523- 531.

[3] M. A. Breuer and S. K. Gupta, “Intelligible testing,” In *Proc Int’l Workshop on Microprocessor Test and Verification*, 1999.

[4] M. A. Breuer, S. K. Gupta, and T. M. Mak, “Defect and error-tolerance in the presence of massive numbers of defects,” *IEEE Design and Test Magazine*, 21, pp. 216-227, May 2004.

[5] S. Shahidi and S. K. Gupta, “ERTG: A test generator for error rate testing,” In *Proc. International Test Conference*, 2007, pp. 1-10.

[6] Zhigang Jiang, Sandeep K. Gupta, “Threshold testing: Covering bridging and other realistic faults, ” In *Proc. Asian Test Symposium ,* 2005, pp. 390-397.

[7] D. Shin and S. K. Gupta, “A Re-design Technique for datapath modules in error tolerant applications,” *In Proc. Asian Test Symposium*, 2008, pp. 431-437.

[8] D. Shin, Sandeep K. Gupta, “Approximate logic synthesis for error tolerant applications,” In *Proc. Design, Automation and Test in Europe,* 2010, pp. 957-960.

[9] I. Chong, H. Cheong and A. Ortega, “New Quality Metric for Multimedia Compression Using Faulty Hardware,” In *Proc. International Workshop on Video Processing and Quality Metrics for Consumer Electronics*, 2006.

[10] R. S. Asamwar, K. Bhurchandi and A.S. Gandhi, “Successive Image Interpolation Using Lifting Scheme Approach,” In *Proc. Journal of Computer Science*, 2010.

[11] N. Jha and S. K. Gupta, *Testing of Digital Systems*, Cambridge University Press, 2003.

[12] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, John Wiley and Sons, 1995.

[13] Luis A. Entrena and K.-T. Cheng, “Sequential Logic Optimization By Redundancy Addition And Removal,” In *Proc. International Conference on Computer-Aided Design*, 1993, pp. 310-315.

[14] P. R. Merion and H. Ahuja, “Redundancy Removal and Simplification of Combinational Circuits,” In *Proc. VLSI Test Symposium,* 1992, pp. 268- 273.

[15] Tong-Yu Hsieh, Kuen-Jong Lee, Melvin A. Breuer, “An Error-Oriented Test Methodology to Improve Yield with Error-Tolerance,” In *Proc. VLSI Test Symposium,* 2006, pp. 130-135.

[16] Yong Chang Kim, Kewal K. Saluja, Vishwani D. Agrawal, “Multiple Faults: Modeling, Simulation and Test,” In *Proc. International Conference on VLSI Design,* 2002.

[17] Mark C. Hansen, Hakan Yalcin, John P. Hayes, “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering,” In *Proc. IEEE Design and Test of Computers*, 1999, pp. 72-80.