Approximate logic synthesis 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* — Error tolerance formally captures the notion that – for a wide variety of applications including audio, video, graphics, and wireless communications – a defective chip that produces erroneous values at its outputs may be acceptable, provided the errors are of certain types and their severities are within application-specified thresholds. All previous research on error tolerance has focused on identifying such defective but acceptable chips during post-fabrication testing to improve yield. In this paper, we explore a completely new approach to exploit error tolerance based on the following observation: If certain deviations from the nominal output values are acceptable, then we can exploit this flexibility during circuit design to reduce circuit area and delay as well as to increase yield. The specific metric of error tolerance we focus on is error rate, i.e., how often the circuit produces erroneous outputs. We propose a new logic synthesis approach for the new problem of identifying how to exploit a given error rate threshold to maximally reduce the area of the synthesized circuit. Experiment results show that for an error rate threshold within 1%, our approach provides 9.43% literal reductions on average for all the benchmarks that we target.**

Keywords* — *error tolerance, logic synthesis, approximate logic function, functional yield

I. INTRODUCTION

As the VLSI fabrication technology scaling is reaching nano-scale, dramatic improvements in most attributes of circuits, especially delay and yield, provided by scaling are beginning to decrease. One of the main reasons for this trend is the increase in non-idealities, such as defect rates and variations due to the manufacturing process [1]. To mitigate the effects of these non-idealities, researchers have proposed concepts of *fault tolerance* and *defect tolerance*. However, the disadvantage of these techniques is that they require additional hardware hence increase the complexity of the circuit, when compared to the original implementation. The concept of *error tolerance* was proposed to combat non-idealities without increasing circuit complexity [2].

Traditional test techniques classify chips as either *perfect*, i.e., without any error-producing defects, or *imperfect*, i.e., those that have one or more error-producing defects or variability. In traditional testing, every imperfect chip is discarded. The core concept of error tolerance is that, for a wide range of applications including image, video, audio, graphics, games, and error-correcting codes for wireless communication, an imperfect chip can still be used, provided the types and severities of errors are within certain application-specified threshold. These imperfect but useable chips (i.e., chips that produce only acceptable errors) are classified as *acceptable* chips [3, 4].

Two quantitative metrics have been previously proposed to measure severity of errors [3]. *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. *Error rate (ER)* is the percentage of vectors for which values at a set of outputs deviate from the error free response, during normal operation. Composite metrics have also been defined using ER and ES.

In [5, 6], faults in circuits used for multimedia applications are analyzed and it is shown that for a significant percentage of faults the output quality degradation is not significant. Also in [7] an algorithm has been developed to generate tests that detect unacceptable chips without rejecting acceptable chips, for applications that use ER as the metric to define acceptability.

The main focus of all previous research on error tolerance was to identify chips during post-fabricated testing that are imperfect but still acceptable. However, [8] we showed that exploiting error tolerance during the design of the circuit can reduce circuit's complexity, i.e., delay and area. Since reduction in circuit complexity can be translated into yield improvement and reduction in fabrication cost, this represents a new way of exploiting error tolerance.

In this paper we propose a logic synthesis approach to design circuits that implement approximate versions of the given function. We consider ER as the metric for error tolerance. The applications for which the chip is to be used must be analyzed to determine the threshold on error rate. Chips that have error rates smaller than this threshold would then be deemed acceptable. The objective is to obtain designs that have minimum area (minimum number of literals) for a given error rate threshold. Assuming that every input vector is equally likely during normal operation, the application-specified error rate threshold determines the number of minterms in the care-set of the given function for which the output value can be complemented. With this observation, we can define the objective of our algorithm as: ***Identify minterm complements that produce an approximate circuit version that has the smallest number of literals for a given error rate threshold.***

Even though a composite metric of ES and ER is useful for many applications, analyzing the each output's ES for multiple output functions is beyond the scope of this paper and a subject of our ongoing research.

First we investigate this problem using exhaustive search where we enumerate all possible approximate functions, each obtained by complementing a different minterm. This simple experiment showed that significant literal reduction is possible even when a single minterm of the original function is complemented. However, the main challenge with exhaustive search is that its time complexity increases exponentially with the number of circuit inputs, and the increase in time complexity with given error rate threshold. In this paper, we identify several new properties and integrate these into a new heuristic approach to synthesize approximate circuit versions for

higher error rate thresholds.

This paper is organized as follows. In the next section, we show the results of our exhaustive search experiments. In Section III, we develop several new properties to reduce the number of candidate minterms that can be complemented to reduce the number of literals. In Section IV, we propose our heuristic approach to identify minterms that can be complemented to maximally reduce the number of literals. Finally in Section V, we present experimental results for benchmark circuits and report the yield improvements provided by our approach.

II. EXHAUSTIVE SEARCH TO FIND OPTIMAL MINTERM COMPLEMENTS

A. *Example of reducing literals in a circuit by complementing minterms of the original function.*

Definition 1) **Minterm complement threshold** (C_t): The maximum number of minterms in the care-set (on-set and off-set) of the Boolean function that can be complemented under the given error rate threshold.

For example, for an n input function, if the error rate threshold is r, then we are allowed to complement up to $C_t = r \cdot 2^n$ minterms in its care-set.

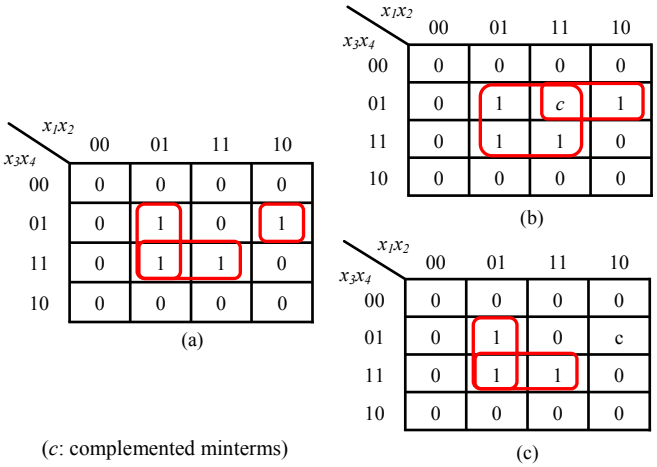


Figure 1. Literal reductions obtained by complementing a minterm.

The cover in Figure 1(a) is the minimum sum-of-product (SOP) cover. Suppose $C_t = 1$. Then we can complement any one minterm from the *original function*. To reduce the number of literals from original minimum cover, we choose minterm $x_1 x_2 \overline{x_3} x_4$ to complement from 0 to 1. The minimum cover for this simplified function is shown in Figure 1(b). The complemented minterm expands the prime implicant (PI) $x_2 x_3 x_4$ to $x_2 x_4$, expands $x_1 \overline{x_2} \overline{x_3} x_4$ to $x_1 \overline{x_3} x_4$ and removes $\overline{x_1} x_2 x_4$. This reduces the number of literals by 1, 1, and 3, respectively, with respect to the original PIs. Hence, the total literal reduction is six for this minterm complement, since removing a PI also reduces 1 literal in the "OR" part of SOP representation. A significant reduction in the number of literals can hence be expected by selecting 0 to 1 minterm complements that can expand many original PIs. With the same constraint of $C_t = 1$, Figure 1(c) shows an example where complementing a minterm from 1 to 0 reduces the number of literals. In this case PI $x_1 \overline{x_2} \overline{x_3} x_4$ can be removed, which reduces five literals from the original minimum cover.

B. *Exhaustive search to find a minterm complement for maximum literal reduction*

This sub-section presents the results of our exhaustive search to identify the minterm complements that reduce the greatest number of literals for a given C_t. Each possible combination of C_t or fewer minterm complements produces one approximate version of the function. Hence, assuming C_t minterms have been complemented to generate each approximate function, we have $C(2^n, C_t)$ possible approximate functions, where n is the number of inputs of the function. (Note: $C(p, q)$ denotes all combinations of q items selected from a given set of p items.) We synthesize each approximate function and count the number of literals in the synthesized circuit. For example, if $C_t = 1$ (i.e., error rate threshold is $1/2^n$), first we complement minterm (000…000) to obtain an approximate version of the function, synthesize it, and count the number of literals in the circuit. This experiment is repeated for all single complements from minterm (000…000) to (111…111). We use two-level logic synthesis tool ESPRESSO-MV [9].

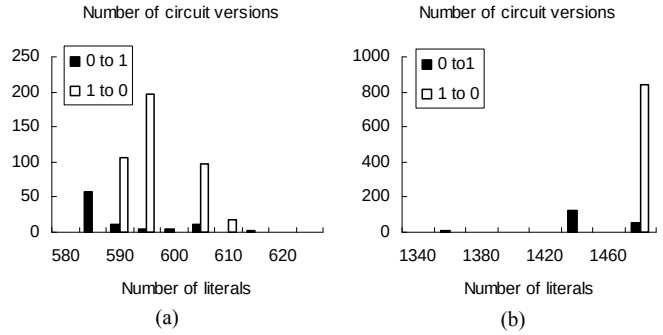


Figure 2. Number of literals in different circuit versions when *Ct* = 1: (a) Z9sym.pla, and (b) sym10.pla.

Figure 2 shows the histograms of number of literals in different circuit versions obtained in this manner for (a) Z9sym.pla (a 9-input, 1-output function), and (b) sym10.pla (a 10-input, 1-output function), which are benchmark circuits from the two-level synthesis suite. The numbers of literals in the original minimum covers are 610 and 1470, respectively. We achieve 4.8% and 8.5% literal (area) reduction for Z9sym.pla and sym10.pla, respectively, when we complement a single minterm that reduces the number of literals by the maximum amount, compared to the minimum cover for the corresponding original function.

One important observation from Figure 2 is that all approximate versions of circuits that have the smallest numbers of literals are obtained from approximate functions obtained by complementing a minterm in the off-set (i.e., by a 0 to 1 complement). This is because to reduce the number of literals by exploiting a 1 to 0 complement, we have to remove an existing PI. With one 1 to 0 complement, we can only remove at most one PI in the original minimum cover. However, a 0 to 1 complement can expand many PIs in the original minimum cover. Also, if an expanded PI covers all the minterms in another PI, then the PI that has been covered becomes redundant and hence can be removed from the minimum cover for approximate function. Figure 1(b) shows the removal of a PI made redundant by a 0 to 1 complement. A similar reasoning also suggests that 0 to 1

complements often provide greater reductions in number of literals for cases where we are allowed to complement multiple minterms of the function.

In Figure 2, we can also observe that several circuit versions have larger numbers of literals than the original minimum cover. There are two reasons for this: (i) a 0 to 1 minterm complement adds a new PI to the original minimum cover, and (ii) a 1 to 0 minterm complement reduces the size of PIs that were used in the original minimum cover and this necessitates the use of multiple PIs, each with more literals.

Such an exhaustive search can only be used for circuits with small number of inputs and to select a small number of minterms to complement, since complexity grows exponentially with the number of inputs and increase in complexity with C_t. Hence, we perform exhaustive search only when $C_t = 1$ or 2 for most of the benchmarks to compare with the heuristic approach that we describe in Section IV.

III. DETERMINISTIC PRUNE OF MINTERM COMPLEMENT CANDIDATES

In the previous section, we quantified literal reductions obtained by exhaustively enumerating all possible approximations. From the results, we notice that 0 to 1 complements are typically more beneficial than 1 to 0 complements and suggested possible reasons for this phenomenon. In the rest of this paper, we will focus only on 0 to 1 complements. In this section, we introduce some useful properties to eliminate 0 to 1 complement candidates that **cannot** reduce the number of literals.

Definition 2) **Adjacent minterms**: Consider two minterms m_i and m_j. m_i and m_j are said to be ***adjacent*** if the hamming distance between m_i and m_j is 1 in the sense that m_i and m_j differ only in one bit.

For example, in a four-input function, minterms $x_1\overline{x}_2\overline{x}_3x_4$ and $x_1\overline{x}_2x_3x_4$ are adjacent to each other.

Definition 3) **Minterm cluster** (M_c): A minterm cluster is a set of minterms complemented from the original function from 0 to 1 such that for any two minterms, m_i and $m_j \in M_c$, there exists a sequence of minterms (m_i, m_{i1}, m_{i2},… ,m_{ia}, m_j) where m_{i1}, m_{i2},… ,m_{ia} belong to M_c, and every pair of consecutive minterms in the sequence are ***adjacent***.

For developing properties, let us denote the set of all on-set minterms in the original function by a set M_o and the set of PIs newly generated by complementing the set of minterms M_c as $PI(M_c)$. Also, denote the set of minterms in a prime implicant PI_j as M_{PIj}.

Property 1) Consider a 0 to 1 minterm complement m_f. If $M_{PI\delta} \cap M_o = \phi$, for all PI such that $m_f \in PI_\delta$, then there exist other approximate versions of the original function with fewer literals and smaller sets of minterm complements that do not include the minterm complement m_f.

Proof) To cover m_f in the new minimum cover, we need at least one PI_δ. Since $M_{PI\delta} \cap M_o = \phi$, if we do not complement minterms that are only covered by PI_δ, including m_f, we can remove PI_δs from the cover for the approximate function. This leads to a new circuit version with fewer literals and fewer minterm complements, because without PI_δ, the number of literals reduces and we complement fewer minterms from the original function.

Hence, under the given minterm complement threshold, there exists another approximate circuit version with fewer literals and fewer minterm complements where we do not complement m_f. □

Above property shows us that if there exists a minterm complement m_f and for all the PI_δs for which $m_f \in PI_\delta$ don't overlap with M_o, then we do not have to consider complementing m_f from the original function. For example, in Figure 3(a), it is unnecessary to complement minterm $x_1\overline{x}_2\overline{x}_3x_4$ with $x_1x_2\overline{x}_3x_4$ because when we complement both minterms, we need an additional PI $x_1\overline{x}_3x_4$ to cover minterm $x_1\overline{x}_2\overline{x}_3x_4$. Even though the minimum cover for the approximate function obtained by complementing both minterms has fewer literals than the original minimum cover, there exists another approximate version of circuit, obtained by complementing only $x_1x_2\overline{x}_3x_4$, which has fewer literals and fewer minterm complements.

Corollary 1) If $M_{PI\delta} \cap M_o = \phi$, for all $PI_\delta \in PI(M_c)$, then the number of literals in the minimum cover of approximate function obtained by complementing M_c is greater than the number of literals in the original minimum cover.

Proof) For $M_c \cup M_o$, the original minimum cover is still the minimum cover for M_o because $M_{PI\delta} \cap M_o = \phi$, for all $PI_\delta \in PI(M_c)$. Since we need additional PIs to cover M_c in addition to the PIs that cover M_o, the literals for these additional PIs are added to the number of literals in the original minimum cover. □

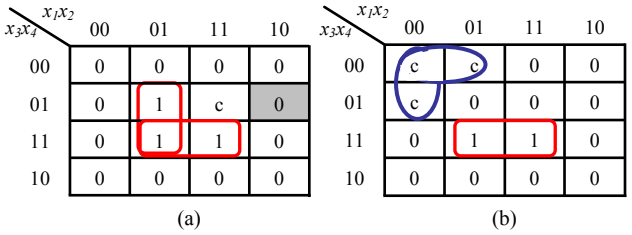


Figure 3. Examples of minterm complement candidates.

Figure 3 (b) shows that complementing $M_c = \{\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4, \overline{x}_1x_2\overline{x}_3\overline{x}_4, \overline{x}_1\overline{x}_2\overline{x}_3x_4\}$ cannot reduce the number of literals from the original minimum cover and M_c requires additional PI to cover. Conversely, for a M_c to reduce number of literals from original minimum cover, there must exist $PI_\delta \in PI(M_c)$ such that $M_{PI\delta} \cap M_o \neq \phi$ (i.e., at least one PI in $PI(M_c)$ must overlap with M_o).

Corollary 2) Let d be the hamming distance between a 0 to 1 minterm complement m_f and a on-set minterm in M_o that is at the minimum distance from m_f. For the new approximate function obtained by complementing m_f to reduce the number of literals in the minimum cover, $\boldsymbol{d \leq log_2(C_t+1)}$.

Proof) Let m_c be a closest on-set minterm in M_o to m_f. Suppose we complement m_f to reduce literals from the original minimum cover. For m_f to reduce literals from the original minimum cover, $\exists\ PI_\delta$ such that $m_f \in M_{PI\delta}$ and $M_{PI\delta} \cap M_o \neq \phi$. The prime implicant PI_δ that contains m_f, overlaps with M_o and contains the smallest number of minterm complements is the PI which consists of literals common to minterms m_f and m_c. All the minterms in this PI must have the literals common to m_f and m_c and the distance from each minterm in the PI (except for m_f) to m_c must be smaller than the distance from m_f to m_c. Since we assume that m_c is the on-set minterm in M_o that is the closest to m_f, all the other minterms except m_c in the PI are not in M_o. This in turn means all these are off-set minterms for the original

function. Since to complement all minterms in the PI except m_c requires 2^d-1 complements, 2^d-1 $\leq C_t$. Hence, we see that $d \leq log_2(C_t+1)$. □

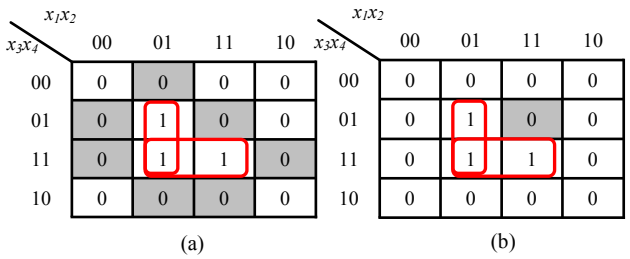


Figure 4. Candidate minterm flips when C_t = 1: (a) All the minterms in *CM*, (b) Minterms that can expand the original PIs.

Figure 4(a) shows the possible minterm complements when C_t = 1. The shaded off-set minterms are the ones for which $d \leq log_2(C_t+1)$ (note that d is a non-negative integer). Only the off-set minterms among these candidates can reduce the number of literals compared to the original minimum cover when C_t = 1. We can try different combinations of C_t or fewer complements out of these candidates and synthesize each approximate function to obtain the approximate version of circuit that has the minimum number of literals, for the given C_t. Assuming we complement all the possible C_t complements for the approximate function with above proprieties, we can only consider $C(CM, C_t)$ cases to complement, where ***CM*** is the set of off-set minterms such that, for each minterm, $d \leq log_2(C_t+1)$.

IV. APPROXIMATE LOGIC SYNTHESIS BY ASSISTING EXPANSIONS

With the observations in Section II and properties derived in Section III, we identify the facts that 0 to 1 complements that are within a particular hamming distance of the minterms in the original function are possible candidate minterms to complement to reduce the number of literals from the original minimum cover. However, a search algorithm that considers $C(CM, C_t)$ approximate functions still has exponential worst-case run-time complexity in the number of function inputs, n, and a significant complexity with C_t.

In this section, we propose a heuristic approach to select 0 to 1 minterm complements to find an approximate circuit version that has the minimum number of literals for a given value of C_t. Among the minterms in *CM*, the minterms that we focus on are ***the minterms that can assist expanding PIs in the original minimum cover***. The main reason is because when we complement only minterms from 0 to 1 that can expand one or more PIs in the original minimum cover, the given minimum cover of the corresponding approximate function is guaranteed to have a smaller number literals than the minimum cover for the original function. On the other hand, complementing minterms in *CM* that cannot expand any PI in the original cover requires new PIs to cover the complemented minterms which may increase the total number of literals. In some cases, complementing minterms that cannot expand any original PI can also reduce the total number of literals by producing a different cover for the original function if minterm complements are in *CM* (minterm complements that can expand one or more original PIs are also in *CM*). In this case, the total literal reduction is the difference between the increase in the number of literals due to the new PIs that cover minterm complements and the decrease in the number of literals due to the use of a different cover for the original function. Figure 4 shows the candidate minterm complements when we consider (a) all minterms in *CM* for C_t = 1, and (b) the candidate minterms that can expand one or more PI in the original cover, for the case where C_t = 1. Typically, the latter set contains much fewer candidate minterms. Our heuristic exploits the fact that in most cases, a combination of minterm complements that can assist PI expansions can provide an approximate circuit version with the minimum number of literals at given C_t. However, there exist some original functions where minterm complements that do not expand any PI in the given original function minimize the number of literals for a given C_t.

Based on above observations, we develop procedures to enumerate all the possible expansions for the original PIs and for identifying approximate circuit versions with the minimum number of literals for a given C_t.

Definition 3) **Minterm set to expand original PIs** (*MSEOP*): A set containing all off-set minterms that prevent expanding one or more PIs in a certain direction.

Since a PI can expand by eliminating different literals in the PI, multiple *MSEOP*s can exist for a PI. For example, in Figure 4(b), $\{x_1x_2\overline{x}_3x_4\}$, $\{\overline{x}_1x_2\overline{x}_3\overline{x}_4, \overline{x}_1x_2x_3\overline{x}_4\}$, and $\{\overline{x}_1\overline{x}_2\overline{x}_3x_4, \overline{x}_1\overline{x}_2x_3x_4\}$ are a few different *MSEOP*s that each expand (by removing one literal) the PI $\overline{x}_1x_2x_4$ in the original cover. We first enumerate all *MSEOP*s that can expand PIs in the original minimum cover to identify the *MSEOP* that leads to an approximate circuit version for a given C_t. Let us denote the cardinality of *MSEOP* by |*MSEOP*|. If |*MSEOP*| $\leq C_t$, we store the original PI, the expanded PI, and the number of literals reduced by the expansion to a list, *L(MSEOP)*. After searching all the PIs that can expand in the given C_t, the list contains the information about PIs that can be expanded by complementing *MSEOP*s. Figure 5 shows the procedure *Generate-list*.

```
//Generate-list ( )
  begin
  foreach PI in the original minimum cover i,
        foreach possible expansion of PI_i j,
            if |MSEOP_ij| ≤ C_t, then add info about PI_ij to expansion
            list L(MSEOP_ij)
  end
```

Figure 5. Procedure to generate list of *MSEOP* that can expand PI.

```
//Generate-union-list (MSEOP_i)
  begin
  foreach L(MSEOP) j,
    if | MSEOP_i ∪ MSCOP_j | ≤ C_t,
    Merge L(MSEOP_i) and L(MSEOP_j) into L(MSEOP_i ∪
    MSEOP_j)
    Generate-union-list (MSEOP_i ∪ MSEOP_j)
  end
```

Figure 6. Procedure for generating union list for union *MSEOP*.

Lists from *Generate-list* consider *MSEOP*s that can expand at least one PI in the original minimum cover. We can consider complementing union of multiple *MSEOP*s, such that $|MSEOP_i \cup MSEOP_j \cup MSEOP_k \ldots| \leq C_t$.

Figure 6 shows the procedure for generating such an union list. These generated lists contain information regarding the literal reduction when complementing an *MSEOP* or union of multiple *MSEOP*s. However, for accurately estimating the literal reduction, we have to examine PIs after expansion. This is because, in many cases, the expanded PIs can make other PIs redundant and counting the literals of these redundant PIs can underestimate the total reduction in the number of literals. Hence for accurate estimation of literal reduction, we develop and use following properties.

Let us denote two different PIs (PI_1 and PI_2) and a *MSEOP* for PI_1, $MSEOP_1$, and a *MSEOP* for PI_2, $MSEOP_2$. Also denote the number of literals reduced by expanding PI_1 using $MSEOP_1$ by l_1 and PI_2 using $MSEOP_2$ by l_2.

Property 2) If $MSEOP_1 = MSEOP_2$ and the PIs after expanding PI_1 and PI_2 are **not identical**, then the number of literals reduced by complementing $MSEOP_1$ is $(l_1 + l_2)$.

Property 3) If $MSEOP_1 = MSEOP_2$ and the PIs after expanding PI_1 and PI_2 are **identical**, then the number of literals reduced by complementing $MSEOP_1$ is: l_1 + (all the literals in PI_2) + 1.

Property 4) If $MSEOP_1 \supset MSEOP_2$ and $PI_1 = PI_2$, then the number of literals reduced by complementing $MSEOP_1$ ($\supset MSEOP_2$) = l_1.

Property (2) shows that if the *MSEOP* is the same for the different PIs, we add the literal reduction for the two PIs, if no PI becomes redundant after expansion. In Property (3), the number of reduced literals in PIs that become redundant is added to the total literal reduction estimation. Property (4) prevents overestimating the number of literals when multiple *MSEOP*s exist for the same PI, and one of the *MSEOP*s is a superset of the other. In such case, expanded PI obtained by the superset *MSEOP* is only the PI that is counted for computing literal reduction. Our list generating procedure and literal estimating properties provide tight estimation of the lower bound on literal reduction by complementing certain *MSEOP* and union of *MSEOP*s.

V. Experimental Results

A. Comparison of assisting expansion heuristic and exhaustive search

To show the near-optimality and efficiency of our heuristic, we performed exhaustive search for six benchmark circuits. The results can be obtained only for C_t = 1 and 2 under our experimental environment due to the high time complexity of the exhaustive search. The experiments were performed on a 2.66 Ghz quad core Intel Xeon processor with 2 GB of main memory. For the sum of runtime for 11 cases, assisting expansion is 87.43 times faster than exhaustive search. Also the percentage difference in number of literals between the two approaches is 2.27%, on average.

From Table 1, we see that multiple output benchmarks have greater difference in the numbers of literals between the exhaustive and heuristics than single output benchmarks. This is because for multiple output functions, our heuristic partitions the function into single output functions and generates input and output relationship for each output separately. After partitioning the function, we select minterm complements that can reduce the number of literals for each single output function. The minterm complements that have the greatest total estimated value of literal reductions for separate output functions is selected. However, above heuristic for multiple output functions does not take into account multiple output implicants, i.e., implicants that have output value of 1 for more than one output [13].

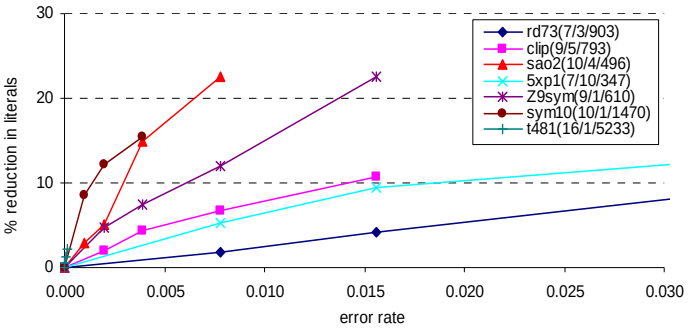
Table 1. The numbers of literals and runtimes.

Benchmark	C_t	# of literals		Runtime(sec)	
		Exhaustive	Heuristic	Exhaustive	Heuristic
Z9sym (9,1, 610)	1	581	581	1.18	0.32
	2	554	564	81.94	1.20
sym10 (10, 1, 1470)	1	1344	1345	6.99	6.39
	2	1290	1290	947.32	26.40
rd73 (7, 3, 903)	1	870	886	0.74	0.24
	2	845	866	46.86	2.73
clip (9, 5, 793)	1	763	777	8.08	0.18
	2	745	759	2162.40	0.25
sao2 (10, 4, 496)	1	449	482	6.50	0.12
5xp1 (5, 7, 347)	1	329	339	0.92	0.08
	2	300	314	60.35	0.10

(Legend: # of I/Ps, # of O/Ps, number of literals in the original function.)

B. Literal reductions for benchmark circuits

We used the heuristic described in Section IV to identify minterm complements that maximally reduce the number of literals for various values of C_t. We synthesized seven benchmark circuits from the two-level synthesis suite that implement arithmetic functions since the idea of error tolerance has been shown to be applicable to datapath functions. To calculate the number of literals in each approximate version of the circuit, minterms selected by our heuristics are complemented in the original minimum cover to obtain an approximate cover, and two-level synthesis is performed. For each circuit, we perform above experiment for C_t = 1, 2, 4, 8. Since error rate threshold is $C_t / 2^n$ for a given C_t, where n is number of inputs, identical C_t values corresponds to different error rate thresholds for different circuits. Typically, as the number of circuit inputs increases, larger literal reduction is achieved for the same error rate threshold. This occurs because the same error rate threshold allows exponentially larger number of minterm complements for a circuit with larger number of inputs. The results in Figure 7 show that, on an average, for an error rate threshold of 0.2% we achieve 3.75% literal reduction, and for an error rate threshold of 1% we achieve 9.43 % literal reduction.



(legend: #of inputs/#of outputs/number of literals in original function.)

Figure 7. Literal reduction for different error rates.

C. Functional Yield improvement due to literal reduction

We use the negative binomial yield model [17] to estimate functional yield improvement due to circuit area reduction that we calculated from the heuristic. Since we don't have any information

regarding manufacturing process, first we assume the yield of the original circuit is 0.7 and the clustering factor (α) is 4, for all the benchmark circuits. These assumptions enable us to calculate the value of defect density for the manufacturing process. Functional yield is then calculated using above defect density and the reduced area of approximately synthesized circuits. Since the circuit size is smaller then original circuit, number of defect in the circuit is smaller then original circuit that we can expect the functional yield improvement of the circuit. Table 2 shows the result of the experiment for different benchmarks. On average, we obtain 3.26% yield improvement with 1% error rate threshold and 4.35% yield improvement with 2% error rate threshold.

Table 2. Yield improvement assuming circuit area is proportional to number of literals.

Benchmarks	Original	Error rate = 0.01		Error rate = 0.02	
		Approximate	% improved	Approximate	% improved
Z9sym	0.7	0.726	3.77	0.754	7.72
sym10	0.7	0.738	5.42	0.738	5.42
clip	0.7	0.716	2.35	0.726	3.74
rd73	0.7	0.704	0.64	0.71	1.41
sao2	0.7	0.757	8.09	0.757	8.09
t481	0.7	0.705	0.77	0.705	0.77
5xp1	0.7	0.713	1.79	0.723	3.31

Yield improvement obtained by approximate synthesis grows with the defect density. In addition to assuming original yield as 0.7 for above experiment, we also recalculate yield improvements by assuming the original yield values as being 0.9, 0.5, 0.3, and 0.1 and measure the yield improvements with approximate logic synthesis. The decrease in original yield represents the increase in defect density. The results show that the % improvement in yield increase drastically as original yield decreases. This means that for the high defect density, we can expect huge % yield improvements using approximate logic synthesis. Figure 8 shows the yield improvement for different values of yield for the original design. Since the main objective of error-tolerance technique is to combat yield decrease due to increasing defect densities in future nano-scale fabrication processes, these curves clearly show that as the problem of yield decrease become more serious, our technique becomes more useful.

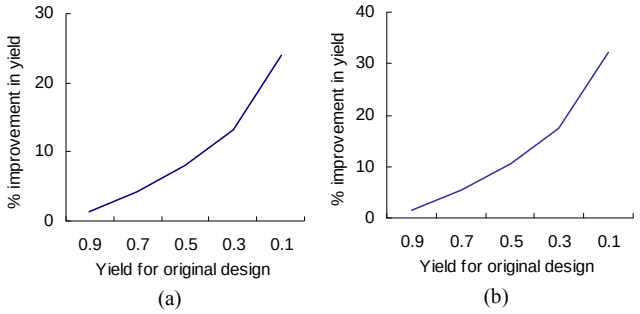


Figure 8. Yield improvement: (a) Z9sym and (b) sym10 for different original yield values (error rate threshold = 0.01).

VI. CONCLUSIONS

In this paper, we present a new logic synthesis approach for error tolerant applications. The major contributions of this paper are (i) Proposing the idea of using error tolerance threshold during the design phase and demonstrating that it can provide dramatic improvements in chip cost and yield. On average, a 0.2% error rate threshold can reduce the number of literals by more than 3.75% and a 1% error rate threshold can reduce the number of literal by 9.43%. (ii) Derivation of properties that eliminate large number of 0 to 1 complements that cannot reduce the number of literals for a given original minimum cover. (iii) Development of a heuristic approach that is practically useable for larger circuits and high values of C_t and demonstration of the fact that our heuristic is near-optimal for all cases that we have tried. (iv) Calculation of the yield improvement using our technique and the demonstration that yield improvement becomes more significant as the original yield decreases.

This work focused on synthesis of approximate two-level circuits starting with a given logic function. Since most of the well known multi-level synthesis heuristics factorize the function using boolean or algebraic division [11, 12], for some cases, a small number of literals in the two-level SOP representation does not mean a small number of literals in the factored form or small area after technology mapping in multi-level synthesis. To find minterm complements that can reduce area and delay of multi-level circuits is a subject of our ongoing research. We are also developing methods to maximally simplify a given implementation of a circuit for implementation at a given value of C_t.

VII. REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS) 2003 [Online]. http://public.itrs.net/Files/2003ITRS/Home2003.htm

[2] M. A. Breuer and S. K. Gupta, "Intelligible testing," In *Proc Int'l Workshop on Microprocessor Test and Verification*, 1999.

[3] M. A. Breuer, "Intelligible test techniques to support error tolerance," In *Proc. Asian Test Symposium*, 2004, pp. 386- 393.

[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] 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.

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

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

[8] 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.

[9] R. Rudell, "Multiple-Valued Logic Minimization for PLA Synthesis," *Technical Report, University of. California, Electronics Research Laboratory, Berkeley*, 1986.

[10] R. K. Brayton, A. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel, *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[11] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 1062–1081, Nov. 1987.

[12] A. Mishchenko, S. Chatterjee, and R. K. Brayton, "DAG-aware AIG rewriting: A fresh look at combinational logic synthesis," *In Proc. DAC*, 2006, pp. 532-536.

[13] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*, Kluwer Academic Publishers, 2000.
[14] S. Devadas, A. Ghosh, and K. Keutzer, *Logic Synthesis*, McGraw-Hill, 1994.
[15] N. Jha and S. K. Gupta, *Testing of Digital Systems*, Cambridge University Press, 2003.
[16] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, John Wiley and Sons, 1995.
[17] I. Koren, Z. Koren, and C. H. Stepper, "A unified negative-binomial distribution for yield analysis of defect-tolerant circuits," *IEEE Trans. Computers*, vol.42, no.6, pp.724-734, Jun. 1993.