Approximate Logic Synthesis by Symmetrization

Abstract—**Approximate synthesis is a recent trend in logic synthesis that changes some outputs of a logic specification to take advantage of error tolerance of some applications and reduce complexity and consumption of the final implementation. We propose a new approach to approximate synthesis of combinational logic where we derive its closest symmetric approximation, i.e., the symmetric function obtained by injecting the minimum number of errors in the original function. Since BDDs of totally symmetric functions are quite compact, this approach is particularly convenient for BDD-based implementations, such as networks of MUXes directly mapped from BDDs. Our contribution is twofold: first we propose a polynomial algorithm for computing the closest symmetric approximation of an incompletely specified Boolean function with an unbounded number of errors; then we discuss strategies to achieve partial symmetrization of the original specification while satisfying given error bounds. Experimental results on classical and new benchmarks confirm the efficacy of the proposed approach.**

I. Introduction

Approximate Logic Synthesis is a novel approach to the synthesis of Boolean functions based on the idea of exploiting the given error tolerance of applications *during design and synthesis* of circuits to implement approximate designs with smaller area, delay, or lower power consumption. Indeed, even though a circuit that exploits error tolerance at the design stage can produce erroneous responses, for a wide range of applications including image, video, audio, machine learning, pattern recognition, and error-correcting codes for wireless communication, errors could be acceptable, provided that their types and severities are within an application-specified threshold [18], [24].

Approximate logic synthesis has been studied under different error metrics, and therefore different cost functions. The two main error constraints that have been considered are *Error magnitude (EM)* and *Error rate (ER)*. The *error magnitude* for a set of outputs is defined as the maximum amount by which the numerical value at the outputs of a circuit can deviate from the exact value, and it is used typically in arithmetic circuits to quantify the numerical error. Instead, the *error rate* represents the percentage of all input vectors that produce the erroneous outputs in the approximate circuits. Composite metrics have also been defined using ER and EM. In this paper we use metrics based on the error rate.

In recent years, many heuristic techniques for synthesizing approximate logic circuits have been investigated with promising results, see for instance [3], [5], [14], [15], [16], [20], [21], [22]. They derive approximate variants of a given combinational Boolean function, by modifying some of its outputs so that the modified circuit has a reduced complexity and power consumption, while the error is within the tolerance bounds (targeting two-level logic, XOR-AND-OR forms, multi-level logic, etc.).

In this paper we propose a new approach to approximate synthesis, by which we change the outputs of a given function with the goal to obtain the closest totally symmetric function. The optimality criterion is given by the chosen error metrics, i.e., we change the original function within the tolerance allowed by the given error bounds, by computing the function that achieves total symmetry with the fewest changes, and then relaxing the symmetrization when the error bounds are not met until we reach a feasible solution. The reason why we target symmetric functions is that they have compact realizations in some logic architectures. For instance, a totally symmetric Boolean function with n input variables has a BDD of size $O(n^2)$ (independent from the variable order) which maps into a network of MUXes of the same size [6].

Technically, we contribute a *polynomial algorithm* for computing the *closest totally symmetric approximation* of a given *incompletely specified multi-output* Boolean function with an unbounded number of errors. This is achieved by exploiting properties of the disjoint covers of a Boolean function and efficient BDD constructions, so that we can build polynomially the characteristic vector of the closest symmetric function and its BDD. Then we propose heuristic strategies to relax total symmetrization in order to guarantee a bounded approximation scheme.

Experimental results on classical and new benchmarks have confirmed the efficacy of the proposed approach: the average gain in the BDD size for the unbounded approximation scheme is 69% with an average error rate of about 15%, while in the bounded context, with a fixed error rate 5%, the average gain is about 31%.

The paper is organized as follows: after preliminaries on symmetric Boolean functions and metrics in Sec. II, the results on the symmetric function closest to a given one are presented in Sec.III, and the heuristics on partial symmetrization under bounded error rates are in Sec. IV. Experimental results are discussed in Sec. V, and final conclusions in Sec. VI.

II. Preliminaries

A. Symmetric Boolean functions

The concept of symmetry has been extensively studied and applied in several contexts, such as function classification, functional decomposition in technology-independent logic synthesis, Boolean matching in technology mapping, formal verification, and binary decision diagram (BDD) minimization (see for instance [9], [12], [13], [17], [19]).

Let $f : \{0,1\}^n \to \{0,1\}$ be a completely specified Boolean function, and $X = \{x_1, x_2, \ldots, x_n\}$ be the set of its input variables. Recall that the function f is generally called *symmetric*, or *totally symmetric*, if it is invariant under all permutations of the input variables, whereas f is said to be *partially symmetric* if it remains invariant under any permutation of a proper subset, of size at least 2, of the input variables.

A function can be partially symmetric with respect to different subsets of variables. More precisely, symmetry of a completely specified Boolean function f in pairs of input variables leads to an equivalence relation on the set X. Thus, there exists a unique minimal partition P of X into disjoint subsets S_1, S_2, …, S_k, with $k \leq n$, given by the equivalence classes of this relation. These sets are called *symmetry sets*. If $k = 1$, then f has only one symmetry set which coincides with X, thus f is totally symmetric, while if $k = n$, f presents no symmetries. In all other cases $1 < k < n$, f is partially symmetric.

Observe that a totally symmetric function f depends only on the number of ones in the input minterm, thus it can be described by its *value vector* $v(f) = [v_0, v_1, \ldots, v_n]$, where $v_i = 1$ if and only if $f(x_1, x_2, \ldots, x_n) = 1$ for all minterm such that $x_1 + x_2 + \ldots + x_n = i$, i.e., all minterms of Hamming weight i.

Very efficient algorithms are known in literature for detecting various type of symmetries and computing the symmetry sets for Boolean functions [17], [19]. Methods to detect symmetries are usually based on checking the equality of two-variable cofactors of the function, in order to find all pairs of symmetric variables. Using this information, larger sets of symmetric variables can be constructed by applying the transitivity of the symmetry relation. This basic approach has been improved in several ways, as reviewed in [17], which presents a new improved method with worst-case complexity cubic in the size of the BDD representing the input function, but close to linear for practical benchmarks.

B. Error metrics for Approximate Synthesis

Following the approach from [20], in [3] the concept of error rate has been further specialized with the notions of *bit threshold* B_t and *minterm threshold* M_t. The first metric evaluates the overall number of complemented (wrong) output bits, while the second metric evaluates the number of input vectors on which the output computed by the circuit differs from the exact one by *at least one bit*. These two error measures coincide for single-output functions, while $B_t \geq M_t$ for multi-output ones.

In this work, we will consider only the bit threshold metric B_t, which can be computed starting from the error rate threshold as follows. Let $f : \{0, 1\}^n \rightarrow \{0, 1, -\}^m$ be a multi-output Boolean function with n inputs and m outputs, and let r denote the error rate, defined as the maximum percentage allowed of erroneous output bits. Then, $B_t = r \cdot m \cdot 2^n$.

In our study, we will also adopt a slightly different approach. In the first part of the paper, we approximate the function f by the totally symmetric function with a minimal Hamming distance from f. Thus, instead of fixing an error rate r, we will first compute the minimum number e of output bits that must be complemented in order to transform f into a totally symmetric function. Then, from e, we will derive the error rate r induced by this transformation as follows. If the target function f is a single output function, with n inputs, then $r = e/2^n$. If the target function f is a multi-output function with n inputs and m outputs, then

$$r = \frac{\sum_{i=1}^{m} e_i}{m\, 2^n} \,,$$

where e_i denotes the minimum number of output bits that must be complemented to transform the i-th output function into a totally symmetric one. The value of r should then be compared with the error rate that is considered acceptable for the application under study, to verify whether the approximation is feasible or not.

In the second part of the paper, we discuss different strategies that can be adopted to *enhance* the symmetry of the target function within a given error rate r defined in advance.

III. Unbounded approximation with a totally symmetric function

In the context of approximate logic synthesis, we study how to modify some outputs of a function f, in order to derive a *symmetric approximation* of f, whose logic implementation might be of reduced complexity, smaller area and delay. This approach is particularly convenient for logic implementation derived from BDDs, as when mapping directly the BDD of a Boolean function onto a network of MUXes [2]. Indeed, the size (i.e., the number of nodes) of a reduced BDD of any totally symmetric function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is bounded by $O(n^2)$ [8], [10], [23].

To this aim, we propose here an algorithm that, given a single output Boolean function f, computes the *minimum number* of outputs that must be changed in order to transform f into a totally symmetric function. In other words, the proposed algorithm computes the totally symmetric function f' closest to f, implicitly assuming an unbounded error rate threshold. We will discuss in Section IV some strategies that can be adopted to enhance the symmetry of the target function f within a given error rate r defined in advance.

A. Completely specified functions

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a completely specified Boolean function depending on n binary variables. Our algorithm for computing the closest symmetric approximation of f starts from a minimal disjoint sum of products form (DSOP) representing f [4]. Recall that in a DSOP representation, each minterm in the on-set of f is covered by exactly one product. We refer the reader to [4] for more details and references on DSOP forms and their applications.

The first step of the algorithm consists in computing, for each $0 \leq w \leq n$, the number of minterms in the on-set of f with Hamming weight w, where the Hamming weight of a minterm $x_1 x_2 \cdots x_n$ is defined as the number of ones among $x_1, \ldots, x_n$. This task can be accomplished using the following result.

Proposition 1: Let p be a product in a DSOP representation of f, containing $d \leq n$ literals, k of which are positive. Let $h = n - d$ be the number of don't care variables, i.e., the variables that do not appear in p. Then, for each $0 \leq w \leq n$, the number $T_p(w)$ of on-set minterms with Hamming weight w covered by p is given by

$$T_p(w) = \begin{cases} \binom{h}{w-k} & k \leq w \leq k+h \\ 0 & o/w\,. \end{cases}$$

Proof. The Hamming weight of the 2^h minterms covered by p is at least k and at most $k+h$, as p contains k positive literals and h don't cares variables. Then, the proposition follows immediately since, for any $0 \leq i \leq h$, there are exactly $\binom{h}{i}$ ways to choose the i variables to set to 1, out of the h variables that do not appear in p. ■

For example, in the Boolean space B^4, consider the function f with DSOP representation $\overline{x}_2\overline{x}_3x_4 + \overline{x}_1\overline{x}_3\overline{x}_4 + \overline{x}_1\overline{x}_2x_3\overline{x}_4 + x_2x_3\overline{x}_4$, and its product $p = \overline{x}_2\overline{x}_3x_4$. For p we have that $k = 1$ and $h = 1$. Therefore, $T_p(0) = T_p(3) = T_p(4) = 0$, $T_p(1) = \binom{1}{0} = 1$, and $T_p(2) = \binom{1}{1} = 1$. In fact, p covers two minterms: one, 0001, with Hamming weight 1 (i.e., $T_p(1) = 1$) and one, 1001, with Hamming weight 2 (i.e., $T_p(2) = 1$).

Summing the contribution $T_p(w)$ of each product p in the DSOP representation of f, we then derive the exact number $T(w)$ of on-set minterms of Hamming weight w, for each $0 \leq w \leq n$. Observe that, f is a totally symmetric function if and only if, for all $0 \leq w \leq n$, $T(w) = 0$ or $T(w) = \binom{n}{w}$.

For instance, considering all the products in the previous example, we have that $T(0) = 1$, $T(1) = 3$, $T(2) = 2$, $T(3) = 1$, and $T(4) = 0$. We can note that f is not symmetric since some of the non-zero values of T are not equal to the corresponding binomial: i.e., $T(1) \neq \binom{4}{1} = 4$, $T(2) \neq \binom{4}{2} = 6$, $T(3) \neq \binom{4}{3} = 4$.

The second phase of the algorithm consists in deriving the totally symmetric function f' closest to f. For all $0 \leq w \leq n$, we compute the minimum number $E(w)$ of minterms on which the output bit of f must be complemented to make the function constant on all inputs of Hamming weight w. Observe that $E(w)$ is simply given by

$$E(w) = \min\left\{T(w), \binom{n}{w} - T(w)\right\}\,.$$

In particular, if $E(w) = T(w)$, then f' is derived from f by a 1 to 0 complement of the output bit on the on-set minterms of Hamming weight w, otherwise f' is derived by a 0 to 1 complement on the off-set minterms of Hamming weight w. The overall minimum number of output bits that must be changed in order to transform f into a totally symmetric function, is then given by $e = \sum_{w=0}^{n} E(w)$.

Following the previous example, we have that $E(0) = \min\{1, \binom{4}{0} - 1\} = 0$, $E(1) = \min\{3, \binom{4}{1} - 3\} = 1$, $E(2) = \min\{2, \binom{4}{2} - 2\} = 2$, $E(3) = \min\{1, \binom{4}{3} - 1\} = 1$, and $E(4) = \min\{0, \binom{4}{4} - 0\} = 0$. In particular, the points with Hamming weight 1 are: 3 in the on-set and 1 (i.e., 1000) in the off-set of f. With 1 error we set all of them to 1. Moreover, the points with Hamming weight 2 are: 2 in the on-set and 4 in the off-set. With 2 errors we set all of them to 0. Finally, the points with Hamming weight 3 are: 1 in the on-set and 3 in the off-set, with 1 error we set all of them to 0. This means that the value vector of the closest symmetric function f' is $v(f') = [1, 1, 0, 0, 0]$, and $e = 4$

Observe that the symmetric approximation f' of f is not necessarily unique. Indeed, whenever $T(w) = \frac{1}{2}\binom{n}{w}$, f' can be derived either by a 1 to 0 complement of f on the on-set minterms of Hamming weight w, or by a 0 to 1 complement of f on the off-set minterms of weight w.

B. Incompletely specified functions

We now discuss how to transform an incompletely specified Boolean function in a totally symmetric one, introducing the minimum number of errors. Since any don't care condition can represent a 0 or a 1, often, the generalization of a problem to incompletely specified Boolean function implies a growth of the complexity of the resolution algorithm. Fortunately, we can show that, in this case, the resolution procedure is still polynomial. The intuition behind this fact is that, for any Hamming weight w, all don't cares should be set to 0 or all the don't cares should be set to 1. Therefore, the choice is performed for any w between 0 and n, and not for any don't care point.

More formally, let $f : \{0,1\}^n \to \{0,1,-\}$ be an incompletely specified Boolean function, and $X = \{x_1, x_2, \ldots, x_n\}$ be the set of its input variables. The minimum number of errors is given by

$$E(w) = \min\left\{T(w), \binom{n}{w} - T(w) - D(w)\right\}\,,$$

where $D(w)$ is the number of don't care minterms of Hamming weight w, for each $0 \leq w \leq n$, computed in the same way as $T(w)$, considering a DSOP representation of the don't care minterms of f. Note that if $E(w) = T(w)$, then f' is derived from f by a 1 to 0 complement of f on all on-set and don't care-set points of Hamming weight w and the number of errors is given only by the number $T(w)$ of on-set minterms of weight w, otherwise f' is derived by a 0 to 1 complement of f on the off-set minterms of Hamming weight w. In this last case, the number of errors is given by the number of off-set minterms of Hamming weight w, i.e., $\binom{n}{w} - T(w) - D(w)$.

C. Algorithms to represent totally symmetric functions

Algorithm 1 describes, in pseudocode, the strategy discussed in the previous subsections, in the general case of incompletely specified Boolean functions.

Let $f : \{0,1\}^n \to \{0,1,-\}$ be an incompletely specified Boolean function, and let f^{on} and f^{dc} denote its on-set and don't care-set, respectively. For the sake of simplicity, suppose that $f^{on} \cap f^{dc} = \emptyset$; otherwise, following the usual semantics, we consider $f^{on} \setminus f^{dc}$ as the on-set of f.

The proposed algorithm starts from the DSOP representations of the on-set and of the don't care-set of the target function f. These representations can be derived by applying a heuristic algorithm for DSOP synthesis as for instance the one described in [4], or by building the BDD representation of f^{on} and f^{dc} and using them to derive the requested DSOP forms, as proposed in [7]. Observe in fact that a DSOP form can be extracted in a straightforward way from a BDD, as different one-paths correspond to disjoint cubes.

Algorithm 1: Algorithm for computing the totally symmetric function closest to an incompletely specified target function under the bit threshold metric.

```
SymmetricApproximation (function $f$)
INPUT: An incompletely specified function $f = (f^{on}, f^{dc})$
OUTPUT: The value vector $v$ of the totally symmetric
function $f'$ closest to $f$, and the number $e$ of output bits
of $f$ that must be complemented to derive $f'$

compute a DSOP representation of $f^{on}$ and of $f^{dc}$
$e = 0$
$v$ = new array of $n+1$ integers, initialized to 0
$T$ = new array of $n+1$ integers, initialized to 0
$D$ = new array of $n+1$ integers, initialized to 0

forall product $p$ in DSOP($f^{on}$) or in DSOP($f^{dc}$)
  compute the number $k$ of positive literals in $p$
  compute the number $h$ of don't care variables in $p$
  for $w = k$ to $k + h$ do
      if ($p \in$ DSOP($f^{on}$))   $T[w] = T[w] + \binom{h}{w-k}$
      else $D[w] = D[w] + \binom{h}{w-k}$

for $w = 0$ to $n$ do
    if ($T[w] > \binom{n}{w} - T[w] - D[w]$)
      $v[w] = 1$
      $e = e + \binom{n}{w} - T[w] - D[w]$
    else
      $v[w] = 0$
      $e = e + T[w]$
return $v$, $e$
```

The algorithm has a time complexity polynomial in the number n of input variables and in the number of products in the DSOP forms representing f^{on} and f^{dc}. The correctness is proved in the following theorem.

Theorem 1: Let f be a Boolean function represented in DSOP form. The proposed algorithm computes the totally symmetric function closest to f, i.e., a totally symmetric function f' that can be derived complementing the minimum number e of output bits of f.

Proof. The correctness of the algorithm follows from Proposition 1 and from the fact that each on-set minterm is covered by one and only one product in the DSOP representation of f^{on}, as well as each don't care-set minterm is covered by one and only one product in the DSOP representation of f^{dc}. Thus, the numbers $T(w)$ and $D(w)$ of the on-set and of the don't care-set minterms of Hamming weight w, for each $0 \leq w \leq n$, can be correctly computed summing the contribution of each product in the DSOP representations of f^{on} and f^{dc}. ■

Observe that in an approximate logic synthesis scenario, the minimum number e of output bits that must be changed to make f totally symmetric provides a sort of lower bound to the bit threshold B_t required in order to allow the approximation of f with f'. That is, whenever $e \leq B_t$, we can synthesize f' instead of f. In terms of error rate, we can consider the approximation feasible if the error rate $r = e/2^n$ induced by the substitution of f with f', is less or equal to the error rate considered acceptable for the application under study.

We finally describe a dynamic programming procedure, based on the recursive approach described in [11] (pp. 65-66), which can be used to build the BDD representation of the totally symmetric function f' computed by Algorithm 1, starting from the value vector v of f'. The idea is to first build an $(n+1) \times (n+2)$ matrix M of BDDs, where $M[i][j+1]$ is the BDD representing the set of all i-dimensional vectors of Hamming weight j, and then to select, according to the value vector v, the BDDs in the last row of the matrix M whose union corresponds to f'.

Algorithm 2: Algorithm for computing the BDD representation of a totally symmetric function, starting from its value vector.

```
ValueVectorToBDD (function $f$)
INPUT: The value vector $v$ of a totally symmetric function
with $n$ inputs
OUTPUT: The reduced BDD representation of $f$

/* Dynamic programming computation of the matrix $M$ */
$M$ = new matrix of dimension $(n+1) \times (n+2)$
$M[0][1]$ = the BDD terminal node 1
for $i = 0$ to $n$ do
    $M[i][0]$ = the BDD terminal node 0
for $i = 0$ to $n$ do
    for $j = i+2$ to $n+1$ do
      $M[i][j]$ = the BDD terminal node 0
for $i = 1$ to $n$ do
    for $j = 1$ to $i+1$ do
      $M[i][j]$ = BDD-ITE($x_i$, $M[i-1][j-1]$, $M[i-1][j]$)

/* BDD construction */
$SymDD$ = the BDD terminal node 0
for $i = 0$ to $n$ do
  if ($v[i] == 1$)  $SymDD$ = BDD-OR($SymmDD$, $M[n][i+1]$)
return $SymDD$
```

D. Multi-output functions

The algorithms that we designed for single-output functions can be applied also to multi-output functions, by simply considering each output as a separate Boolean function. Indeed, once each output function has been transformed into a totally symmetric function, the overall multi-output function becomes totally symmetric as well, as proved in the following Proposition 2.

Let $f : \{0,1\}^n \to \{0,1,-\}^m$ be an incompletely specified Boolean function with n inputs and $m > 1$ outputs. For each i, $1 \leq i \leq m$, let f_i denote the single output function corresponding to the i-th output of f, and let f'_i be the totally symmetric function derived complementing the minimum number e_i of output bits of f_i, i.e., applying Algorithm 1 to each output of f. Finally, let f' be the multi-output function obtained substituting each output function f_i in f with the totally symmetric function f'_i, $1 \leq i \leq m$.

Proposition 2: The multi-output function f' is a totally symmetric function.

Proof. A multi-output function is totally symmetric if and only if it is invariant under all permutations of the input variables, i.e., if and only if it computes the same output vector on all minterms with the same Hamming weight. Now consider the function f' and observe that each single output of f' is totally symmetric, thus each single output is constant on all minterms with weight w, $0 \leq w \leq n$. As a consequence, f' outputs the same vectors on all minterms with Hamming weight w and the thesis immediately follows. ■

We now prove that not only f' is totally symmetric, it is also the totally symmetric function closest to f.

Proposition 3: The function f' derived applying Algorithm 1 to each output of f is the totally symmetric function that can be derived complementing the minimum number of output bits of f.

Proof. For each i, $1 \leq i \leq m$, let e_i denote the number of output bits complemented in order to transform the i-th output function of f into a totally symmetric one applying Algorithm 1. Moreover, let $e = \sum_{i=1}^{m} e_i$ denote the overall number of output bits complemented to make the whole f totally symmetric. Suppose that there exists a totally symmetric multi-output function g' closer to f. Then, there exists at least one output of f, say f_i, that can be made totally symmetric complementing $e'_i < e_i$ output bits, in contradiction with the minimality of e_i proved in Theorem 1. ■

Finally, the error rate r induced by the approximation of f with the totally symmetric function f' can be computed as

$$r = \frac{e}{m\,2^n} = \frac{\sum_{i=1}^{m} e_i}{m\,2^n},$$

where m is the number of outputs of f.

IV. Symmetry-based approximation under a bounded error rate

In this section we discuss some possible strategies that can be adopted to enhance the symmetry of a function f within an error rate r defined in advance.

Let $f : \{0,1\}^n \to \{0,1,-\}^m$ be an incompletely specified function with n inputs and m outputs, and let r denote the error rate. From r, we can compute the bit threshold $B_t = r \cdot m \cdot 2^n$, i.e., the maximum number of output bits that we are allowed to complement to derive a *symmetric* approximation f' of f.

A first greedy strategy consists in simply sorting the outputs of the multi-output function with respect to the number of errors needed to transform each of them into a totally symmetric function. Starting from the output function with the lowest number of errors, we transform in symmetric functions the outputs till we reach the given bit threshold B_t. In this way, we enhance the symmetry of the overall function making some of its outputs totally symmetric.

An alternative method could consist in making a function constant on all minterms with the same Hamming weight. The idea is basically to apply Algorithm 1 to f, and to stop the computation as soon as the number of complemented output bits reaches the bit threshold B_t. But instead of processing the subsets of minterms in order of increasing Hamming weight, we first compute, for each weight w and each output function f_i, the minimum number $E_i(w)$ of minterms on which the output bit of f_i must be complemented to make this output constant on all inputs of weight w. Then, we start from the weight w and from the output f_i on which f can be made constant (0 or 1) complementing the *least number* of output bits, and we proceed in a greedy way in ascending order of $E_i(w)$, until we reach the threshold B_t. At the end of the process, we obtain a function f' with some outputs that assume a constant value on minterms with equal weight for a subset of possible weights. This approach presents an evident limitation, as making a function constant only for some weights does not necessarily enhance its overall symmetry and may not provide realizations of reduced complexity. This limitation has been confirmed by our experimental results, that have clearly shown how only the greedy strategy based on the total symmetrization of some outputs of the function provides interesting results.

Finally, we observe that a different approach could be based on enhancing the partial symmetry of a function, instead of its total symmetry, increasing the number of variables that can be permuted without changing the output. This task could be accomplished for instance increasing the cardinality of the biggest symmetry set adding a new variable to it. The variable should be selected in a greedy way, as the variable whose insertion in the currently biggest symmetry set causes the least number of erroneous output bits. Due to space limitation, this methodology will be discussed in more details in the long version of the paper.

V. Experimental results

In this section we discuss the experimental results obtained by applying the algorithms described in Sections III and IV. We have considered the classical Espresso benchmark suite [25] and the new EPFL combinational benchmark suite [1]. The computational experiments were performed on a Linux Intel Core i7-7700 CPU with 8 GB of RAM. The algorithms have been implemented in C, using the CUDD library for BDDs to represent Boolean functions.

Table I compares the BDD dimension for the benchmark functions and for their closest symmetric approximations in the unbounded and bounded error models. Due to the limited space available, we report only a significant subset of the experiments. The first column reports the name of the benchmarks and the number of their inputs and outputs. The second column shows the dimension (number of nodes) of the BDDs representing the on-set of the benchmark functions. The first group of columns, labeled *Sym Unbounded*, refers to the closest totally symmetric function in the unbounded model (Section III). The first column in this group shows the BDD size of the symmetric function, the second column contains the required error rate, and the third column reports the percentage gain of the BDD of the closest symmetric function. The second group, labeled *Sym Bounded*, provides the results for the bounded model based on the symmetrization of a subset of the outputs (Section IV) with a fixed error rate $r = 5\%$. The last column provides the computational time in seconds (for computing both approaches). In general, the average gain for the unbounded model is 69% with an average error rate 15%. In the bounded context, with a fixed error rate 5%, the average gain is about 31%. Due to the polynomial nature of the approaches, the computational times are very limited.

The second bounded approach described in Section IV, which makes a function constant only for some weights, has been tested on the same benchmarks with low quality results, i.e., the BDD size increases by 150%. This confirms the theoretical intuition that fixing a constant value for just some weights does not give an interesting "symmetry" property to the function.

TABLE I
EXPERIMENTAL COMPARISON WITH THE CLOSEST-SYMMETRIC IN THE UNBOUNDED AND BOUNDED MODELS.

		Sym Unbounded			Sym Bounded (5%)		
bench.(in/out)	**BDD**	**BDD**	**ER (%)**	**gain**	**BDD**	**gain**	**time (s)**
alu2 (10/8)	242	42	9.7	82.6	177	26.9	0.01
alu3 (10/8)	237	43	9.7	81.9	170	28.3	0.01
b2 (16/17)	4424	79	29.0	98.2	4215	4.7	0.15
b9 (16/5)	173	80	37.9	53.8	173	0.0	0.01
bc0 (26/11)	590	490	32.9	17.0	635	-7.6	0.02
bca (26/46)	1428	52	0.1	96.4	52	96.4	0.02
bcb (26/39)	1268	52	0.1	95.9	52	95.9	0.01
bcc (26/45)	1116	27	0.1	97.6	27	97.6	0.02
bcd (26/38)	843	27	0.1	96.8	27	96.8	0.01
cavlc (10/11)	508	37	10.3	92.7	291	42.7	0.01
chkn (29/7)	742	234	7.8	68.5	577	22.2	0.02
ctrl (7/26)	101	27	13.4	73.3	65	35.6	0.01
dec (8/256)	510	16	0.4	96.9	16	96.9	0.01
i2c (147/142)	2873	8306	22.1	-189.1	7287	-153.6	2.24
in0 (15/11)	518	89	23.2	82.8	410	20.9	0.01
in1 (16/17)	4424	79	29.0	98.2	4215	4.7	0.10
in2 (19/10)	2361	101	12.1	95.7	1195	49.4	0.05
in5 (24/14)	492	144	10.6	70.7	200	59.4	0.01
in7 (26/10)	235	229	23.0	2.6	166	29.4	0.01
int2float (11/7)	359	51	25.5	85.8	345	3.9	0.01
mark1 (20/31)	253	153	0.0	39.5	153	39.5	0.01
max1024 (10/6)	261	93	43.2	64.4	261	0.0	0.01
max128 (7/24)	130	78	26.0	40.0	144	-10.8	0.01
max46 (9/1)	75	1	12.1	98.7	75	0.0	0.01
max512 (9/6)	148	65	38.0	56.1	168	-13.5	0.01
opa (17/69)	513	172	8.0	66.5	118	77.0	0.01
pdc (16/40)	695	102	4.0	85.3	102	85.3	0.06
router (60/30)	231	551	0.2	-138.5	231	0.0	2.77
t2 (17/16)	149	81	7.9	45.6	44	70.5	0.01
test2 (11/35)	4817	77	9.9	98.4	2694	44.1	0.14
test3 (10/35)	2619	79	9.8	97.0	1445	44.8	0.07
test4 (8/30)	941	89	8.5	90.5	476	49.4	0.02
tial (14/8)	1307	153	37.7	88.3	1029	21.3	0.01
ts10 (22/16)	4391	1	6.3	100.0	1331	69.7	0.09
vg2 (25/8)	219	213	21.2	2.7	175	20.1	0.01
vtx1 (27/6)	241	228	14.9	5.4	182	24.5	0.01
x9dn (27/7)	271	229	12.8	15.5	186	31.4	0.01
Z9sym (9/1)	25	25	0.0	0.0	25	0.0	0.01

VI. Conclusions

In this paper we presented a new approach to approximate synthesis of combinational logic, by deriving a polynomial algorithm to find the symmetric approximation that injects the minimum number of errors in a given incompletely specified Boolean function, and then relaxing the symmetrization until the given error bounds are met. This approach is particularly convenient for BDD-based implementations, such as networks of MUXes directly mapped from BDDs. Experimental results on classical and new benchmarks confirm the efficacy of the proposed approach. Future work includes the investigation of bounded error approximation methods to enhance the partial symmetry of a function, increasing the number of symmetric variables as mentioned in Section IV.

References

[1] "The EPFL Combinational Benchmark Suite." [Online]. Available: http://lsi.epfl.ch/benchmarks

[2] B. Becker, "Synthesis for testability: Binary decision diagrams," in *STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, France, February 13-15, 1992, Proceedings*, 1992, pp. 501–512.

[3] A. Bernasconi and V. Ciriani, "2-SPP approximate synthesis for error tolerant applications," in *17th Euromicro Conference on Digital System Design, DSD 2014, Verona, Italy, August 27-29, 2014*, 2014, pp. 411–418.

[4] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Compact DSOP and partial DSOP forms," *Theory Comput. Syst.*, vol. 53, no. 4, pp. 583–608, 2013.

[5] A. Chandrasekharan, M. Soeken, D. Große, and R. Drechsler, "Approximation-aware rewriting of aigs for error tolerant applications," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2016, pp. 1–8.

[6] R. Drechsler and W. Günther, *Towards One-Pass Synthesis*. Kluwer Academic Publishers, 2002.

[7] G. Fey and R. Drechsler, "Utilizing BDDs for disjoint SOP minimization," in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, vol. 2, 2002.

[8] M. A. Heap, "On the exact ordered binary decision diagram size of totally symmetric functions," *J. Electronic Testing*, vol. 4, no. 2, pp. 191–195, 1993.

[9] L. Heinrich-Litan and P. Molitor, "Least upper bounds for the size of OBDDs using symmetry properties," *IEEE Trans. Computers*, vol. 49, no. 4, pp. 360–368, 2000. [Online]. Available: https://doi.org/10.1109/12.844348

[10] L. Heinrich-Litan, P. Molitor, and D. Möller, "Least upper bounds on the sizes of symmetric variable order based OBDDs," in *6th Great Lakes Symposium on VLSI (GLS-VLSI '96), March 22-23, 1996, Ames, IA, USA*, 1996, p. 126.

[11] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, *Synthesis of Finite State Machines: Functional Optimization*. Kluwer Academic Publishers, now Springer, 1996.

[12] B. Kim and D. L. Dietmeyer, "Multilevel logic synthesis of symmetric switching functions," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 10, no. 4, pp. 436–446, 1991.

[13] V. N. Kravet and K. A. Sakallah, "Constructive library-aware synthesis using symmetries," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, March 2000, pp. 208–213.

[14] Y. Lai, C. Lin, C. Wu, Y. Chen, and C. Wang, "Efficient synthesis of approximate threshold logic circuits with an error rate guarantee," in *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, 2018, pp. 773–778.

[15] J. Miao, A. Gerstlauer, and M. Orshansky, "Approximate logic synthesis under general error magnitude and frequency constraints," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 779–786.

[16] ——, "Multi-level approximate logic synthesis under general error constraints," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2014, pp. 504–510.

[17] A. Mishchenko, "Fast computation of symmetries in Boolean functions," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 11, pp. 1588–1593, 2003.

[18] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016. [Online]. Available: http://doi.acm.org/10.1145/2893356

[19] C. Scholl, D. Möller, P. Molitor, and R. Drechsler, "BDD minimization using symmetries," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 18, no. 2, pp. 81–100, 1999.

[20] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *DATE*, 2010, pp. 957–960.

[21] D. Shin and S. Gupta, "A new circuit simplification method for error tolerant applications," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2011, pp. 1–6.

[22] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: Systematic logic synthesis of approximate circuits," in *49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2012, pp. 796–801.

[23] I. Wegener, "Optimal decision trees and one-time-only branching programs for symmetric Boolean functions," *Information and Control*, vol. 62, no. 2/3, pp. 129–143, 1984.

[24] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb 2016.

[25] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Microelectronic Center, User Guide, 1991.