Approximate Logic Synthesis of Very Large Boolean Networks

Jorge Echavarria, Stefan Wildermann, Jürgen Teich
Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany
Email:{jorge.echavarria, stefan.wildermann, juergen.teich}@fau.de

***Abstract*—For very large Boolean circuits most approximate logic synthesis techniques successively apply local approximation transformations affecting only a portion of the whole design. Hence, allowing such transformations to be implemented in polynomial time and to gain better control of the introduced error. A key issue here is to derive efficient techniques for selecting from all the possible portions of the design those more likely to yield better trade-offs between hardware resources and quality.**

Due to the likelihood of *error masking* growing with increasing circuit complexity, we expect the likelihood of a local transformation *reaching*—or being observable at—the primary outputs to decrease at a similar rate. Comparatively, the closer a portion undergoing a local transformation is to the primary outputs, the more likely the error introduced can be observed at the primary outputs. Based on this observation, this paper proposes a novel methodology for the selection of portions—or *sub*-functions—of Boolean circuits—represented by Boolean networks—for approximation according to their *degree of connectivity* with other portions of the design. Our selection criterion is based on that a Boolean *sub*-function shall be a better candidate for approximation when it drives many other *sub*-functions, especially those being driven by many other *sub*-functions.

We introduce, integrate, and compare our connectivity-based selection methodology with a state-of-the-art approximate logic synthesis framework. Experimental results show that our selection technique yields better trade-offs between hardware resources and accuracy of the resulting approximated circuits. Moreover, our technique is efficient and can speed up the design space exploration of the aforementioned framework.

I. Introduction

Approximate computing (AC) techniques let us circumvent hardware limitations, like, for example, resources and timing constraints by means of accuracy degradation. Such limitations are forcing us to come up with more clever ideas to pack a required piece of logic with specific timing and power budgets in pieces of hardware that cannot be upgraded to more sophisticated and expensive chips. A popular approach for designing approximate hardware is logic falsification [1], which may simplify the Boolean function representing a circuit to procure an inexact equivalent with a certain gain. Other techniques focus on don't care simplifications [2], or the pruning of gate-level netlists by selecting circuit portions for their subsequent removal [3]–[7]. Certainly, for these approximation frameworks to be efficient, a robust and scalable error estimation after applying such transformations is most importantly [8].

Due to the large design space of Boolean network approximation, many approximate logic synthesis (ALS) methodologies successively apply local approximate transformations (LATs). At the end of each iteration, these techniques evaluate the induced error, such as the error rate (ER), the bit error rate (BER), the error distance (ED), the mean error distance (MED), or the maximum absolute error (MAE); and the hardware gains, assessed by cubes and literals, area, delay, and/or power consumption.

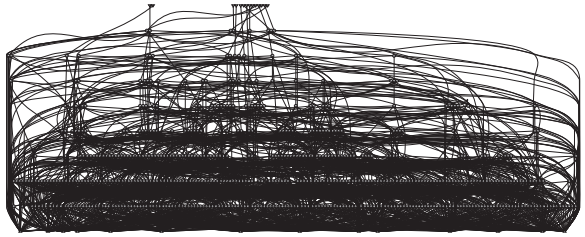

Figure 1: Boolean network of Benchmark *misex3* taken from IWLS'93 [9]. This still rather small example has 14 primary inputs (PIs) and 14 primary outputs (POs). Yet, the Boolean network is made of 603 4-feasible Boolean *sub*-functions.

In this paper, we present a *sub*-circuit[1] selection algorithm to bypass the non-scalable complexity of the design space exploration (DSE) of LATs. We showcase the performance enhancement induced by our selection algorithm in a state-of-the-art Boolean simplification-by-approximation framework [1]. When guided by our selection algorithm a significant reduction in resources for the inexact circuits and further speedup derived by the method presented in [1] is observed. Our approach is applicable to optimize any of the aforementioned statistical error metrics, we opted, however, for reporting only the ER, as proposed in [8], since that the ER is the backbone of many other error metrics and due to space constraints.

Moreover, our methodology can be applied to any graph-based representation of circuits, like, for example: a) AND-inverter graphs (AIGs), direct acyclic graphs (DAGs) representing a structural implementation of the logical functionality of a circuit or network; b) majority-inverter graphs (MIGs), logic representation structures consisting of three-input majority nodes and regular/complemented edges [10]; or c) post-technology mapping netlists.

In this paper, we will restrict our experimental assessments and exemplifications to very large Boolean networks, as the one portrayed in Fig. 1.

The rest of the paper is organized as follows: First we introduce the necessary notations and conventions to be able to follow the paper. Then, in Section III we present a brief survey of the related work. Section IV contains the main contribution of our work. It details the proposed selection methodology and briefly describes the DSE and the error estimation technique

[1]In the remainder of this paper, we will use the terms "portion of a circuit", "*sub*-circuit", and "*sub*-function" interchangeably.

978-3-9819263-5-4/DATE21/©2021 EDAA

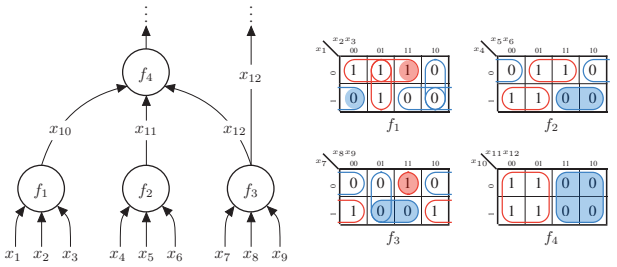

Figure 2: Part of a Boolean network consisting of *sub*-functions f_1, f_2, f_3, and f_4. The on-sets and the off-sets are highlighted with colors red ○ and blue ○, respectively. Similarly, the on-sets and the off-sets falsification candidates are represented by the contours filled in red ● and blue ●, respectively.

adopted to showcase our main contribution. Then, we discuss our empirical findings in Section V. Finally, we draw conclusions and discuss our plans for future work in Sections VI and VII, respectively.

II. Preliminaries

A. Boolean networks

Let $f_i : \{0,1,*\}^{n_i} \rightarrow \mathbb{B}$ describe a Boolean *sub*-function and f_i' describe an approximation of f_i. The union $\bigcup_i (f_i')$ corresponds to the n-input m-output Boolean network $B : \{0,1,*\}^n \rightarrow \mathbb{B}^m$.

The direct inputs (or outputs) of a Boolean *sub*-function within the Boolean network are called its fan-ins (or fan-outs). A combinatorial circuit, described by a Boolean network can be represented as a DAG $G(V,E)$, where each node $p \in V$ corresponds to a Boolean *sub*-function f_i and each edge $e(q,p) \in E$ to a connection—or transitive fan-in (TFI)—between nodes such that one of the fan-ins of p is a fan-out of q, as shown in Fig. 2. The TFI cone of a node includes the node itself and its TFIs. The PIs are source nodes and the POs are a subset of V.

For a node $p \in V$, a *k-feasible cone at* p, denoted $\check{p}$, is a *sub*-graph of the TFI of p such that the number of inputs of $\check{p}$ is less than or equal to k [11].

Each implicant t_i is described by a row vector $c = [c_1, \ldots, c_{n_i}]$, where c_k is 0 (or 1) if $\bar{x}_k$ (or x_k) appears in t_i, or $*$ if neither, $\bar{x}_k$ nor x_k appear in t_i. A set of cubes $\mathcal{C}_i = \{c^1, \ldots, c^l\}$ represents a cover of the on-set (or the off-set) of a logic Boolean *sub*-function f_i' iff every minterm (or maxterm) is covered by at least one cube in $\mathcal{C}_i$ and no cube $c \in \mathcal{C}_i$ covers any member of the complement of $\mathcal{C}_i$.

B. Logic falsification

Modifying the implicants of a given Boolean function f, thereby introducing errors in the definition of the function on purpose, is considered a *logic falsification* [1]. Each logic value 0 (or 1) disposed to a 1 (or 0) during falsification is called a falsified minterm (or maxterm) in the following. Similarly to [8], we consider four sets of implicants of a function f_i', namely: a) the on-set ${}_{\alpha}F_{f_i'}$, which represents the set of input values x such that $f_i(x)$ and $f_i'(x)$ evaluate to 1, b) the off-set ${}_{\alpha}R_{f_i'}$, which represents the set of input values x such that $f_i(x)$ and $f_i'(x)$ evaluate to 0, c) a falsified on-set ${}_{\varepsilon}F_{f_i'}$, which represents the set of input values x such that $f_i'(x)$ evaluates to 1 whilst $f_i(x)$ evaluates to 0, and d) a falsified off-set ${}_{\varepsilon}R_{f_i'}$, which represents the set of input values x such that $f_i'(x)$ evaluates to 0 whilst $f_i(x)$ evaluates to 1.

III. Related work

There is no common agreement on how to best select nodes as candidates for LAT in Boolean networks. Yet, many attempts have been made. In this section, we will briefly describe some of these works.

In [3], the authors present an AIG-based approach. Their proposed approximation technique is able to reduce the delay by selecting the critical paths (CPs), and the area of the circuits by selecting cuts. The selection of the cuts is done by performing cut enumeration on the selected CPs. The authors assume, however, that each node has an equal probability of inducing errors at the POs. Under this observation, the errors introduced shall then, by intuition, be related to the size of the cuts. Each LAT is then applied according to an increasing order of each cut size without violating the given error metrics.

The authors in [4] present a post-technology mapping DAG-based approach. Node pruning is performed based on three criteria: a) significance, which is a structural parameter, b) toggle count (TC), or c) the significance-activity product. These knobs allow better control over the ED and the ER. This approach, however, only considers node removal, that is, entire Boolean *sub*-functions may be erased from the Boolean network if selected as candidates for approximation.

In [5], the authors partition a graph representing a Boolean network by means of a search tree where each level within the tree represents a node and each branch corresponds to the inclusion or exclusion of the node itself into the set of nodes for approximation. Each path corresponds to an individual partition. An inclusion branch will be selected if its corresponding weight—indicating the maximum difference visible at the circuit outputs—is not greater than a given error threshold. Otherwise, the algorithm backtracks to the previous node.

The authors in [6] present an algorithm that first partitions a graph representing the circuit and then determines an error-propagation model of the resulting *sub*-graphs. The error-propagation model labels each node with the MAE induced by its removal.

These methodologies, nevertheless, only work for arithmetic units due to the bit-significance of the POs for weights required for the error-propagation model. More importantly, the complexity of these approaches increases with the cut-feasibility and number of outputs per *sub*-graph.

Naïve error-bound iterative selection approaches may ignore candidates that would yield low error and better gains. This occurs because as long as such an approach finds a single approximation candidate satisfying the error bound it would simply keep looking for the next LAT. Our methodology, on the other hand, applies the internal Boolean simplification-by-approximation LATs in every node candidate for approximation as proposed in [1].

Another novel approach is presented in [12]. Here, the authors *re*substitute a node based on an approximation of its care set obtained by analyzing its TFI.

Unfortunately, the latter three approaches still rely on simulations. Thus, trying to tackle the scalability issues the authors of [12] express the care patterns in terms of internal nodes instead of the PIs.

IV. DSE-driven ALS

Before introducing our novel selection algorithm, we briefly explain its integration into a DSE framework for multi-output Boolean logic function approximations as presented in [1]. In that work, the on-set of a Boolean *sub*-function f_i represented in its canonical disjunctive normal form (CDNF) is approximated by *over*expanding irredundant sum-of-products (ISOP) of a Boolean function, that is, a sum-of-products (SOP) expression in which each product term is a prime implicant. An on-set prime implicant is *over*expanded by selecting cubes from a set (R^c) made of only those off-set implicants which guarantee that by selecting them the number and/or size of prime implicants in the respective cover are reduced, hence reducing the size of the on-set cover in terms of cubes and literals. The off-set, on the other hand, is approximated by assimilating maximally-reduced cubes from a set made of on-set implicants with a greater likelihood of not becoming redundant by subsequent *over*expansions (F^c). The aforementioned falsification techniques introduced in [1] reduce the search space without sacrificing the quality of the result while improving their scalability. The DSE is based on NSGA-II, a nondominated sorting multi-objective genetic algorithm (GA) [13].

A. Baseline approach

We use binary encoding, where each falsification candidate is represented by one bit. The length of a chromosome of the baseline approach introduced in [1] is equivalent to $\sum_i |R_i^c| + |F_i^c|$ as it requires to perform an exploration of the whole design space. Consider the Boolean network shown in Fig. 2 with the falsification candidates $\bar{x}_1x_2x_3$, $x_1\bar{x}_2\bar{x}_3$, x_4x_5, $\bar{x}_7x_8x_9$, x_7x_9, and x_{11}. A valid chromosome would, for example, be 101001, representing the inclusion of cubes $\bar{x}_1x_2x_3$, x_4x_5, and x_{11} into the set of candidates for falsification. Clearly, such an approach would not scale well with very large Boolean networks.

B. Connectivity-Based approach

In ALS, node selection is a tricky process that demands great acquaintance over the Boolean network. Thus, an automated, as well as an efficient method for determining promising node selections, is of utmost importance. Currently, however, there is no common understanding to formally comprehend the factors behind an effective node selection, as it is necessary to predict prior to the approximation stage which nodes would induce fewer errors at the POs for a particular LAT. We identify as a foremost quality of such an algorithm to recognize complex relationships between nodes such that we are able to identify those exhibiting higher correlations. Our instinct tells us that a high correlation between a node and its transitive fan-out (TFO) increases the likelihood of a LAT applied at this specific node to face masking scenarios that may conceal any effects it may have on the POs.

In this paper, we examine the potential of optimization progress of a DSE when exploiting the observability of LAT at the POs during node selection by introducing a graph-based centrality technique. We evaluated our node selection algorithm exploiting the connectivity of a circuit for a set of significantly large benchmark circuits.

In the following, we describe our method for node selection. We integrated it into the framework ABC [14] allowing it to handle the k-feasible cuts of Boolean networks. The **i**ntegrated **F**PGA (if) mapper of ABC supports up to 32-feasible cut sizes.

One of the main challenges in probability propagation through Boolean networks is reachability, also known as observability. While estimations trough simulations offer unlimited visibility into every node of a Boolean network, fully accurate techniques have no such luxury. An analogous approach to node selection is to consider the TFOs driving large parts of the circuit to be more important than those driving smaller parts. A DAG, as shown in Fig. 2, may encapsulate the information of such connectivity from a flattened netlist of a Boolean network.

In this paper, we use the hyperlink-induced topic search (HITS) method [15], also known as *hubs and authorities*, to quantify the relative importance of each node in the DAG. HITS is an iterative method originally proposed to rank Web pages. Nowadays, it is employed outside the context of Web search in an ever-growing number of application domains, like, for example, topic distillation, word stemming, and—like in our case—item selection, making it a reference algorithm for link analysis [16].

Given a DAG $G(V, E)$, let $h^{\langle p \rangle}$ and $a^{\langle p \rangle}$ respectively represent the hub and authority scores of G:

$$h^{\langle p \rangle} \propto \sum_{q:(p,q)\in E} a^{\langle q \rangle}, \tag{1}$$

$$a^{\langle q \rangle} \propto \sum_{p:(p,q)\in E} h^{\langle p \rangle}. \tag{2}$$

The hub and authority scores quantify the relative importance of each node in the graph. Loosely speaking, *the mutually reinforcing relationship* exhibited by Eqs. (1) and (2) may be interpreted as follows: a good *hub* is a node pointing to many good authorities, whilst a good *authority* is a node being driven by many good hubs.

For a Boolean network with $|V|$ nodes, its connectivity graph $G(V, E)$ can be captured as an adjacency matrix A with dimensions $|V|$-by-$|V|$, where each $A_{i,j}$ entry is 1 or 0 for whether an edge $e(p_i, p_j) \in E$ exists between nodes p_i and p_j. Take, for example, the Boolean network shown in Fig. 2 with an adjacency matrix A:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots \\ 0 & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots \\ 0 & 0 & 0 & 1 & \cdots & 1 & \cdots & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots & \cdots \end{bmatrix}.$$

Thus, in matrix notation, the hub and authority scores may be computed as $h = \beta Aa$ and $a = \alpha A^T h$, respectively, where a and h represent the hub and authority score eigenvectors $\{a^{\langle p \rangle}\}$ and $\{h^{\langle p \rangle}\}$ of $A^T A$ and AA^T, and α and β represent constants. In our context, $A^T A_{ij}$ and AA^T_{ij} represent the cardinality of the TFI and the TFO, respectively, of both p_i and p_j.

Algorithm 1: HITS algorithm.

Data: Every node within a Boolean network collected in a DAG G. Stop condition: $s \in \mathbb{Z}$.
Result: (h_s, a_s)

1 $a_0 := (1, \ldots, 1)^T$
2 **for** $i = 1, 2, \ldots, s$ **do** // Repeat looking for convergence
3 $\quad h_i := Aa_{i-1}$ // Update h-weights, see Eq. (1)
4 $\quad a_i := A^T h_i$ // Update a-weights, see Eq. (2)
5 $\quad h_i := \frac{h_i}{|h_i|}$ // Normalize h
6 $\quad a_i := \frac{a_i}{|a_i|}$ // Normalize a
7 **end**

Note that the normalization steps shown in Lines 5 and 6 in Algorithm 1 are well-defined assuming, as we shall do throughout the rest of the paper, that the DAG G has at least one edge $e(p, q)$. This algorithm has many other applications, such as in data mining, and it has influenced other link analysis algorithms like Google's PageRank, and Ask (previously Teoma) showcasing its scalability. In addition to its greater scalability over other centrality measures, such as closeness and betweenness, the rationale behind choosing HITS over other network centrality measures is the capability HITS possesses of selecting those nodes in the network which are important for the network, but that are not central themselves. Moreover, instead of simply considering the influence of the immediate TFO, this approach also considers the downstream relationship of its TFI, which is itself affected by its own TFI. The main advantage of this approach is that instead of accurately computing the observability of specific LATs at the POs, we can obtain a similar measurement efficiently even for large circuits.

Fig. 3 shows an example of the hub and authority scores of another small example from IWLS'93 [9] for purpose of illustration. Algorithm 1 was run for $s = 20$ iterations to obtain the shown values. As one may observe, those nodes closer to the PIs share a similar hub score, except for the rightmost node. The reason behind this occurrence lays on the fact that the TFOs of the five leftmost nodes are larger than the TFO of the rightmost node. We may even draw a similar observation about the uppermost node. Note, however, that the only two inner—non PI—nodes driving the uppermost node do not share the same hub score. The reason corresponds to the difference in the size of the TFIs of both nodes.

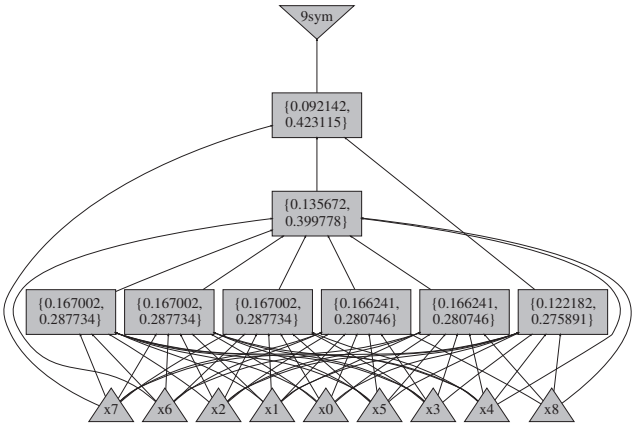

Figure 3: 8-feasible HITS-centrality graph of Benchmark *9sym* taken from IWLS'93 [9] after $s = 20$ iterations. The values within each node p represent their $\{h^{\langle p \rangle}, a^{\langle p \rangle}\}$ scores.

Given that TFOs driving large parts of the circuit shall be considered more important than those driving smaller parts, our selection algorithm takes into account only the hub scores of each node within the Boolean network. The condition of inclusion for a given node p_i as falsifiable is chosen as follows:

$$h^{\langle p_i \rangle} \geq \frac{\max\limits_{p \in V} \left(h^{\langle p \rangle}\right) - \min\limits_{p \in V} \left(h^{\langle p \rangle}\right)}{2}. \tag{3}$$

Considering the intuitive rationale behind our selection methodology, the node representing Boolean *sub*-function f_3, as shown in Fig. 2, would most likely be considered for its inclusion for falsification, as it seems to belong to a greater, or be part of many, TFIs. To continue with our example, let us consider that nodes representing the Boolean *sub*-functions f_1 and f_2 also satisfy Eq. (3), then a valid chromosome with five genes would be sufficient to represent each individual within each population, as any possible candidate from f_4 would be ignored due to its apparent proximity to the POs.

1) Time complexity: As seen in Algorithm 1, the computation of the hub and authority weights $h_i = Aa_{i-1}$ and $a_i = A^T h_i$—Lines 3 and 4—and the normalizations $h_i = h_i/|h_i|$ and $a_i = a_i/|a_i|$—Lines 5 and 6—share the same computational complexity of $O(N)$. Therefore, the complexity of the HITS algorithm when executed for s iterations is $O(4N \times s)$.

C. Fitness evaluation

We consider $m + 2$ objectives during the fitness evaluation: the BER of each of the m POs, the number of cubes—see Eq. (4)—and the number of non-$*$ literals—see Eq. (5). The ED is the most natural approach to evaluate the outputs of arithmetic circuits since a clear notion of distance is available for these functions. For generic functions, on the other hand, probabilistic metrics, such as the ER, seem to be a more sensible choice. We have empirically found, however, that the implementation of LATs in very large Boolean networks generally affects more than one single PO simultaneously. Therefore, we consider the BER to better represent the inaccuracies induced by any given LAT. Moreover, the BER allows

us to better steer the DSE according to specific weights given to each of the POs in accordance to their bit-significance.

Statistical errors, such as the BER, are usually computed using a) binary decision diagrams (BDDs), b) Monte-Carlo (MC) simulations, or c) analytical methods. The former two techniques are too expensive and do not scale well. The latter, on the other hand, based on signal probability propagation, has shown to scale better [8]. More importantly, this methodology has been shown to be capable of estimating the BER in the order of milliseconds of every benchmark it has been tested for, allowing us to incorporate it in our framework with tight timeouts.

This methodology characterizes all error masking scenarios to compute and propagate the so-called *error densities* $\mathcal{D}_{\varepsilon F_{f_i'}}$, and $\mathcal{D}_{\varepsilon R_{f_i'}}$ of each node p_i through the Boolean network. A remarkable feature of this approach is that only the error densities within the TFO of node p_i have to be updated after applying a LAT to Boolean *sub*-function f_i represented by such node, considerably reducing the number of computations.

For the sake of fair comparison and to better showcase our methodology, we restricted our following experiments to the minimization of, besides the BER, the number of cubes, which is trivially evaluated by counting the number of row vectors of each on-set cover:

$$\sum_{i=1}^{|V|} |_{\alpha}F_{f_i'}| + |_{\varepsilon}F_{f_i'}|; \tag{4}$$

and the number of non-$*$ input literals, with an equally trivial computation:

$$\sum_{i=1}^{|V|} \sum_{j=1}^{|_{\alpha}F_{f_i'} \cup_{\varepsilon} F_{f_i'}|} |\{l, 1 \leq l \leq n_i : c_j^l \neq *\}|. \tag{5}$$

V. Results

As discussed in Section III, we are not able to compute the exact observability of any of the nontrivial benchmark circuits used in this paper. We therefore have no means of directly measuring the accuracy of our selection algorithm. Therefore, we discuss instead the observable gains of our algorithm over a state-of-the-art ALS framework with regard to the accuracy, amount of required resources—in terms of cubes and literals—and run-time.

We set the stop condition s—see Algorithm 1—to 20 as this is known to be a good figure that allows the hub and authority scores to converge to steady values [15]. All the experiments reported in this paper were steered using NSGA-II [13] for the performed DSE. The GA was set up to stop after 512 generations with an initial population of the same size. To be able to report the $m+2$ objectives of our DSE, we make use of the hypervolume indicator $\mathcal{I}_H$ [17], a measure that estimates the volume of the dominated portion of the objective space. In the sake of consistency and better readability, we report the hypervolume indicators after normalization within the range $[0, 1]$.

First, similarly to [1], [7], [18], we analyze the resource gains—with regard to cubes and literals—against the induced error—with regard to the BERs, following the algorithm presented in [8]. Fig. 4 shows the normalized hypervolume indicators $\mathcal{I}_H$ of the Pareto sets of 4 different benchmarks after finalizing all the generations. As we can observe, after performing a DSE under the same GA setup and after the same number of generations, our selection algorithm is able to drastically improve the quality of the ALS framework introduced in [1]. We find these results quite encouraging as it shows that our initial reasoning on error masking due to larger TFOs holds empirically true.

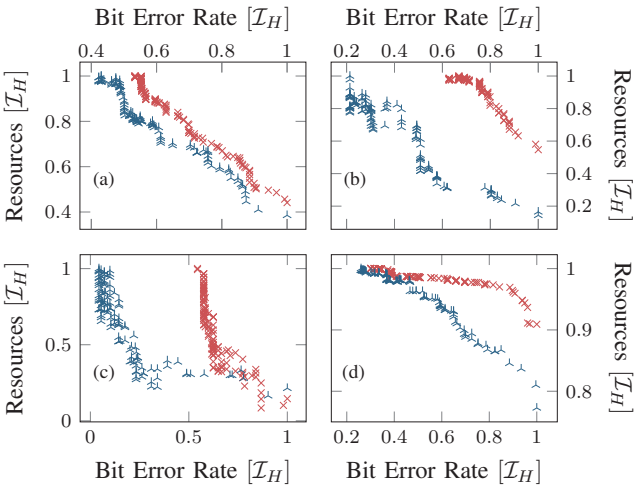

Figure 4: Normalized hypervolume indicators $\mathcal{I}_H$ [17] of resources—in terms of cubes and literals—as shown in the y-axes, and the BER, as shown in the x-axes, of each feasible population. The ⅄ marks represent the proposed approach performing HITS-based selection whilst the × marks represent the baseline approach [1]. The benchmarks a) *clip*, b) *misex1*, c) *t481*, and d) *sao2* were taken from IWLS'93 [9].

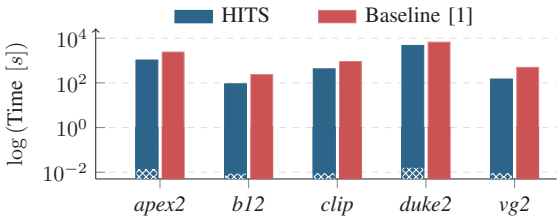

Figure 5: The required time in logarithmic scale to approximate and evaluate each of the individuals of each of the 512 generations of our GA using the approximation methodology introduced in [1] applied to five different benchmarks taken from IWLS'93 [9]. The blue—left—bars correspond to our selection algorithm. The white cross-hatch at the bottom of the bars represents the time taken to compute the hub and authority scores. The red—right—bars correspond to the baseline approach.

Fig. 5 compares the run-time in seconds of the DSE of five different benchmarks using our node-selection algorithm against the baseline approach described in Section IV-A. As may be observed, our selection algorithm constantly beats the time required to finalize the 512 generations. We may also notice with the cross-hatch shown at the bottom of the blue bars the small fraction of time needed to compute the hub and authority scores of each tested benchmark.

Clearly, the fraction of time required to compute the hub

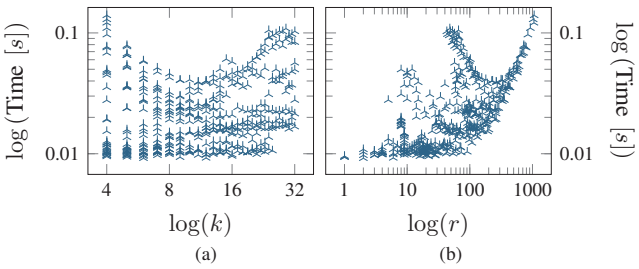

Figure 6: Relation between required time to compute the hub and authority scores of every benchmark from IWLS'93 [9] and a) k-feasibility cuts with $k = 4, ..., \min(32, n)$, and b) the resulting number of r nodes, with $r = |U|$ and $U \subseteq V$. Note that the number of nodes is a byproduct of the k-feasible cuts. Both axes are shown in logarithmic scale.

and authority scores may vary according to the number of nodes within the Boolean network, or the k-feasibility cuts. Figs. 6a and 6b show this relation. Along with the fact that the hub and authority scores computation is performed only once, one may still notice that such calculation takes no longer than 124.56 ms for any of the benchmarks taken from [9]. Moreover, as previously discussed, the larger the value of k, the fewer the number of nodes within the Boolean network, and consequently, the shorter the run-time of our node selection algorithm.

VI. Conclusion

As motivated in this paper, the large design space of approximate logic synthesis for large Boolean networks requires the investigation of efficient node selection algorithms. Such kind of algorithms can greatly improve the quality of the results of local approximate transformations applied successively in Boolean networks targeting multiple objectives, as demonstrated in Section V. In this paper, we presented an efficient node selection algorithm that exploits the observation that the likelihood of *error masking* increasing with circuit complexity reduces the probability of a local approximation transformation to reach, or be observed, at the primary outputs. For the presented Algorithm 1, a consistent speedup in the design space exploration has been shown for Boolean networks taken from the IWLS'93 set of benchmarks. We have assessed this claim with extensive experimentation incorporating our methodology in a well-established simplification-by-approximation framework [1], using a very well-known multi-objective genetic algorithm [13], and a state-of-the-art error propagation technique [8].

VII. Future Research

We have identified the following milestones for future work: a) Perform design space exploration with an automated selection of the k value for k-feasible cuts. b) Revisit and further improve the hub-based condition of inclusion for approximation shown in Eq. (3). c) Develop ideas for better selecting the initial weights of the hub and authority scores—as shown in Algorithm 1. d) Carry out further research on the effects of each node external don't cares to improve local minimization. e) Finally, investigation of other network centrality techniques, such as, for example, eigenvector centrality, degree centrality, PageRank, closeness centrality, and betweenness centrality.

References

[1] J. Echavarria, S. Wildermann, and J. Teich, "Design space exploration of multi-output logic function approximations," in *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018*, I. Bahar, Ed. ACM, 2018, p. 52.

[2] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012*, P. Groeneveld, D. Sciuto, and S. Hassoun, Eds. ACM, 2012, pp. 796–801.

[3] A. Chandrasekharan, M. Soeken, D. Große, and R. Drechsler, "Approximation-aware rewriting of aigs for error tolerant applications," in *Proceedings of the 35th International Conference on Computer-Aided Design, ICCAD 2016, Austin, TX, USA, November 7-10, 2016*, F. Liu, Ed. ACM, 2016, p. 83.

[4] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, "Design and applications of approximate circuits by gate-level pruning," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, no. 5, pp. 1694–1702, 2017.

[5] I. Scarabottolo, G. Ansaloni, and L. Pozzi, "Circuit carving: A methodology for the design of approximate hardware," in *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, J. Madsen and A. K. Coskun, Eds. IEEE, 2018, pp. 545–550.

[6] I. Scarabottolo, G. Ansaloni, G. A. Constantinides, and L. Pozzi, "Partition and propagate: an error derivation algorithm for the design of approximate circuits," in *Proceedings of the 56th Annual Design Automation Conference 2019, DAC 2019, Las Vegas, NV, USA, June 02-06, 2019*. ACM, 2019, p. 40.

[7] S. Su, C. Zou, W. Kong, J. Han, and W. Qian, "A novel heuristic search method for two-level approximate logic synthesis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 39, no. 3, pp. 654–669, 2020.

[8] J. Echavarria, S. Wildermann, O. Keszöcze, and J. Teich, "Probabilistic error propagation through approximated Boolean networks," in *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*. IEEE, 2020, pp. 1–6.

[9] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Tech. Rep., May 1993.

[10] L. G. Amarù, P. Gaillardon, and G. D. Micheli, "Majority-inverter graph: A new paradigm for logic optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 806–819, 2016.

[11] J. Cong and Y. Ding, "Flowmap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 1–12, 1994.

[12] C. Meng, W. Qian, and A. Mishchenko, "ALSRAC: Approximate logic synthesis by resubstitution with approximate care set," in *Proceedings of the 57th Annual Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 19-23, 2020*. IEEE, 2020.

[13] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[14] B. L. Synthesis and V. Group, "ABC: A system for sequential synthesis and verification," http://www.eecs.berkeley.edu/~alanmi/abc/, Release 200717.

[15] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[16] E. Peserico and L. Pretto, "HITS can converge slowly, but not too slowly, in score and rank," *SIAM J. Discret. Math.*, vol. 26, no. 3, pp. 1189–1209, 2012.

[17] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of pareto- compliant indicators via weighted integration," in *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007, Proceedings*, 2006, pp. 862–876.

[18] J. Miao, A. Gerstlauer, and M. Orshansky, "Approximate logic synthesis under general error magnitude and frequency constraints," in *The IEEE/ACM International Conference on Computer-Aided Design, ICCAD'13, San Jose, CA, USA, November 18-21, 2013*, J. Henkel, Ed. IEEE, 2013, pp. 779–786.