Polynomial Formal Verification of Approximate Functions

Martha Schnieber*, Saman Froehlich*, Rolf Drechsler*†
*Institute of Computer Science, University of Bremen, Bremen, Germany
†Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
{schnieber, froehlich, drechsler}@uni-bremen.de

***Abstract*—During the development of digital circuits, their functional correctness has to be ensured, for which formal verification methods have been developed. However, the verification process using formal methods can have an exponential time and space complexity, potentially leading to a failure due to time or space constraints. Thus, in the past years, it has been shown that multiple circuits are guaranteed to be verifiable efficiently in polynomial time and space using specific formal methods. However, the polynomial verifiability of approximate functions, which are beneficial for applications with hardware or time constraints, has not been focused by research yet.**

In this paper, we present two methods for generating polynomially verifiable circuits for an approximate function g, if g approximates another function f such that they differ for a polynomial amount of input assignments and a polynomially verifiable circuit for f exists. Using BDDs, we prove the polynomial verifiability of both circuits for g with respect to the number of inputs. Furthermore, for several error metrics, we explore the correlation between the BDD sizes during the verification process of both circuits and the error between f and g. Finally, we experimentally evaluate the given upper bounds for the BDD sizes during the verification process of both circuit architectures for several approximate functions.

I. Introduction

Digital circuits play a significant role in modern systems, as they are responsible for a numerous amount of calculations. They can be found in a variety of applications, including safety-critical systems like cars or medical equipment. Thus, ensuring that the designed circuit correctly implements the specified function is critical. However, as there exist an exponential amount of possible assignments to a circuit, testing every assignment is not feasible.

Consequently, formal verification methods have been established to prove the correctness of a circuit [1] [2]. This can be done using several formal methods, such as SMT, SAT, BDDs [3] or other kinds of decision diagrams such as *BMDs [4]. Here, the circuit correctly implements the specification if e.g. the BDDs for the circuit and the specification are equal. Hence, the BDD for the circuit has to be constructed.

However, during the computation of the formal representation of the function realized by the circuit, the formal representation can reach an exponential size and therefore, the verification can fail due to time or space constraints [5]. Thus, predicting the time and space complexity of the verification process of specific circuits is critical, which is a challenge that has not been heavily focused by research yet. Nonetheless, in the past years, it has been shown for several circuits that they can be verified efficiently in polynomial time and space. So far, research has only focused on proving the polynomial verifiability for a fixed set of classes. Methods applicable to a wide range of functions have not been published yet.

This work was supported by the German Research Foundation (DFG) within the Project *PLiM* (DR 287/35-1) and the Reinhart Koselleck Project *PolyVer* (DR 287/36-1).

For some applications which have hardware or time restrictions, the exact output of a specific function is not required but an approximate output is sufficient. For these applications, the function can be approximated, resulting in a similar function which computes a different output in some cases. In return, the circuit for the approximate function is more area-efficient or has a lower delay, which is benefitting if there exist area or time restrictions [6] [7].

Several methods have been proposed for the development process of such approximate circuits [8] [9]. However, no research has yet focused on proving their polynomial verifiability and thus, even if the exact circuit is polynomially verifiable, the polynomial bounds are not guaranteed to hold for the approximate circuit. Thus, in this paper, we present two polynomially verifiable circuits for a function g with respect to the number of inputs, if g is an approximation of a function f, for which a polynomially verifiable circuit is known and where the number of assignments for which g and f differ scales polynomially with the number of inputs. We present both synthesis strategies for the generation of polynomially verifiable circuits for g and give upper bounds for the size of the BDDs during the verification process of these circuits. Furthermore, we give an upper bound for both the time and space complexity of the verification process. Thus, using the methods presented in this paper, circuits for approximate functions can be developed which are guaranteed to be polynomially verifiable. Furthermore, the maximum BDD size during the verification process, as well as the verification time and space can be estimated beforehand using the given upper bounds.

As approximate functions are typically evaluated using error metrics [10], we also explore the correlation of the error between f and g and the BDD sizes during the verification process of both circuits for four error metrics. The results can be used to enhance the development process of approximate circuits in such a way that the resulting circuits are guaranteed to be verifiable in polynomial time and space with respect to the number of inputs. In the experiments, we evaluate the upper bounds for the BDD sizes during the verification, as well as the proposed circuit architectures with respect to area and depth.

II. Preliminaries

A. Boolean Functions

Let $f : \{0,1\}^n \to \{0,1\}^m$ be a Boolean function with n inputs and m outputs. Then, $\alpha = (\alpha_1, \ldots, \alpha_n) \in \{0,1\}^n$ is

an assignment, which assigns a truth value to every input variable. Furthermore, $\alpha(f)$ applies the assignment to f, meaning every input variable x_i is is assigned the truth value α_i. For each assignment α, there exists a minterm m_α, which is a conjunction of all input variables, where every variable is either a positive or a negated literal [11].

B. Error Metrics

Several error metrics have been proposed to measure the difference between two functions and evaluate an approximation. In this paper, we examine four error metrics, where f and g are functions with n inputs and m outputs and α is an assignment. All error metrics are given normalized, meaning they are divided by the number of possible assignments [10] [12].

- Bit threshold: $bt(f,g) = \frac{\sum_\alpha \sum_{i<m} \alpha(f)_i \neq \alpha(g)_i}{2^n}$
- Error rate: $er(f,g) = \frac{\sum_\alpha \alpha(f) \neq \alpha(g)}{2^n}$
- Average-case error: $ace(f,g) = \frac{\sum_\alpha |\alpha(f) - \alpha(g)|}{2^n}$
- Mean-squared error: $mse(f,g) = \frac{\sum_\alpha (\alpha(f) - \alpha(g))^2}{2^n}$

C. Binary Decision Diagrams

A *Binary Decision Diagram* (BDD) is a directed acyclic graph that represents a Boolean function. A BDD has internal nodes and terminal nodes, where the terminal nodes represent the values 0 and 1.

In an *Ordered Binary Decision Diagram* (OBDD), each level of the BDD consists of nodes for the same input variable, where the order in which the variables appear is given by the variable ordering $(x_1, x_2, \ldots, x_n)$. Furthermore, a *Reduced Ordered Binary Decision Diagram* (ROBDD) is ordered and consists of a minimal amount of nodes in the context of a given variable ordering, making it canonical [3].

We denote $|f|$ as the size of the ROBDD of the function f in the number of nodes. As every node in a level can have two successors in the next level, a BDD can have an exponential number of nodes.

Each internal node v represents an input variable x_i and has two output edges, one for $x_i = 0$, which is called the low-edge and one for $x_i = 1$, which is called the high-edge. The low-edge of v leads to the node $low(v)$ and the high-edge leads to the node $high(v)$. Thus, the function f for a terminal node is its respective value, meaning 0 or 1, whereas the function f for an internal node v is recursively given by the Shannon decomposition: $f = \overline{x}_i f_{x_i=0} + x_i f_{x_i=1}$.

D. ITE-Operator

The ITE-operator (If-Then-Else) is an operator on BDDs which can be used for synthesis operations, e.g. for AND, OR or XOR operations. It can be computed recursively:

$$ITE(f,g,h) = ITE(x_i, ITE(f_{x_i=1}, g_{x_i=1}, h_{x_i=1}), ITE(f_{x_i=0}, g_{x_i=0}, h_{x_i=0}))$$

where x_i is the top variable of f, g and h. The complexity of the ITE-operation is $\mathcal{O}(|f| \cdot |g| \cdot |h|)$ and it therefore has a polynomial worst-case behaviour. Thus, the basic operations like AND, OR and XOR can be carried out on BDDs in polynomial time and space with respect to the BDD size [13].

III. Related Work

Even though the topic of polynomial verification is relatively new, in recent years, some papers have explored the polynomial verifiability of specific circuits.

Firstly, much research has focused on the polynomial verifiability of several adders. The author of [5] shows that the RCA, the CSA and the CLA are all polynomially verifiable, whereas the authors of [14] give specific bounds for the time complexity of the verification process of the CSA. Furthermore, in [15], the polynomial verifiability of some prefix adders is shown, specifically of the Serial Prefix Adder, the Ladner-Fischer Adder and the Kogge-Stone Adder.

Apart from adders, the polynomial verifiability of multipliers has been researched as well in [16], where the authors show that Wallace-tree like multipliers are polynomially verifiable. Additionally, the authors of [17] have shown that the verification process of integer arithmetic circuits has a polynomial time and space complexity with respect to the circuit size. Finally, it has been shown in [18] that a polyomially verifiable circuit can be generated for every symmetric function.

Thus, the polynomial verifiability of several circuit classes has already been shown. However, no research has yet focused on the polynomial verifiability of approximate functions.

IV. Polynomial Verification

In this section, we show that if a circuit for a function f can be verified in polynomial time and space in the number of inputs and if g is an approximation of f, where the amount of assignments for which the outputs of f and g differ scales polynomially with the number of inputs n, we can specify two methods for generating a circuit for g that are guaranteed to be polynomially verifiable as well with respect to the number of inputs. If the amount of assignments for which f and g differ scales at most polynomially with n, we call the amount of assignments for which the outputs differ polynomial. Here, we focus mainly on functions with a single output. However, for functions with multiple outputs, the results can be applied to every output separately.

Let f be a function of n input variables and let α be an assignment of the input variables. Then, m_α is the minterm of the assignment α. The output of f for a specific assignment α can be complemented using the operation $m_\alpha \oplus f$.

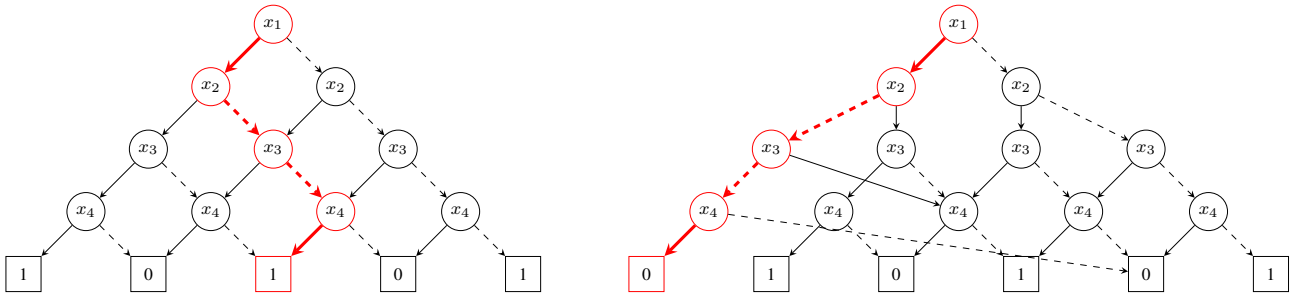
Theorem 1. *The XOR operation $m_\alpha \oplus f$ results in a BDD with at most $|f| + n$ nodes, where n is the number of inputs.*

Proof. On BDDs, $m_\alpha \oplus f$ can be calculated using $ITE(m_\alpha, \overline{f}, f)$. Let $(x_1, x_2, \ldots, x_n)$ be the variable ordering, where x_1 is the topmost variable. Using the definition of the ITE-operator, it holds that

$$ITE(m_\alpha, \overline{f}, f) = x_1 \cdot ITE(m_{\alpha x_1=1}, \overline{f}_{x_1=1}, f_{x_1=1}) + \overline{x}_1 \cdot ITE(m_{\alpha x_1=0}, \overline{f}_{x_1=0}, f_{x_1=0}).$$

As m_α is a conjunction of all input variables in either positive or negated form, either $m_{\alpha x_1=1}$ or $m_{\alpha x_1=0}$ evaluates to 0: $m_{\alpha x_1=0} = 0$ if $\alpha(x_1) = 1$, whereas $m_{\alpha x_1=1} = 0$ if $\alpha(x_1) = 0$.

Therefore, only one ITE-operation is left for which the next level has to be calculated, where again one of the two minterms



(a) BDD for f

(b) BDD for $g = m_\alpha \oplus f$

Fig. 1: BDD of a function with 4 inputs before and after complementing the output with $m_\alpha = x_1\overline{x}_2\overline{x}_3x_4$

evaluates to 0. This continues for every level, meaning in every level, only one ITE-operator has to be evaluated.

As there is exactly one additional ITE-operator in each level of the resulting BDD and as there exist at most n levels of internal nodes, there are at most n internal nodes in which an ITE-operation has to be evaluated, where n is the number of inputs. All other internal nodes in the BDD also exist in the BDD of f. Therefore, the $\oplus$-operation adds at most n internal nodes to the BDD of f. □

Note that the reduction of the resulting BDD can further reduce the amount of nodes. In most cases, no terminal nodes have to be added as they are already included in f. However, if f is one of the constant functions 0 or 1, only one terminal node is included in f and an additional terminal node has to be added. For the remainder of this section, we calculate the bounds for the case that f is not a constant function.

An example for the BDD of a function before and after the $\oplus$-operation is given in Figure 1. Here, we have used unreduced BDDs with multiple copies of the terminal nodes for clarity. Furthermore, $m_\alpha = x_1\overline{x}_2\overline{x}_3x_4$, where the corresponding path is marked in red. Figure 1(a) shows the BDD of a symmetric function f, whereas Figure 1(b) shows the BDD of $m_\alpha \oplus f$.

The exact number of assignments δ for which f and g differ can be computed with the satisfy-count operation on the BDD for $f \oplus g$, whereas the exact assignments can be computed with the satisfy-all operation on the BDD for $f \oplus g$, which calculates its satisfying set [3]. Let δ with $\delta \leq n^c$ for some constant c be the amount of assignments for which the outputs of f and g differ, meaning the number of assignments for which f and g differ scales at most polynomially with the number of inputs. Using Theorem 1, we can conclude that the BDD for g has a size of at most $|g| \leq |f|+n^c \cdot n = |f|+n^{c+1}$. If a circuit for f is polynomially verifiable in the number of inputs, $|f|$, meaning the size of the BDD for f, is polynomial and therefore, $|g|$ is also polynomial in the number of inputs.

Using the results from Theorem 1, we can now specify two circuit architectures for g that are polynomially verifiable, if the outputs of f and g differ for a polynomial amount of assignments and if a circuit for f is polynomially verifiable.

A. Polynomially Verifiable Multiplexer Circuit

Our first presented circuit architecture is the circuit which results from replacing all nodes in the BDD for g with multiplexers, which we call the multiplexer circuit. Specifically,

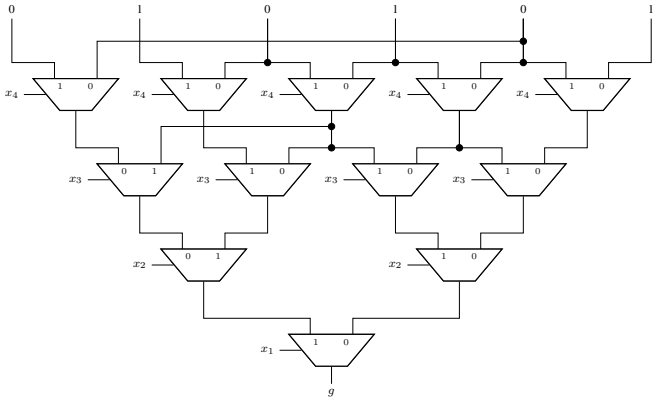


Fig. 2: Multiplexer circuit for Figure 1(b)

every node in the BDD is replaced by a multiplexer whose inputs are the high and low edges of the respective BDD node, whereas the select input is its input variable. This circuit has already proven to be polynomially verifiable in [19]. However, we provide an approach which uses the ITE-operator to represent the multiplexers and yields upper bounds for the BDD sizes, as well as for the time and space complexity.

The multiplexer circuit which results from the BDD in Figure 1(b) is shown in Figure 2. Every BDD node has been replaced by a multiplexer with their respective high and low edges as inputs and the input variables as select inputs.

As each multiplexer consists of 4 gates, the number of gates in the multiplexer circuit for g is at most $4 \cdot |g|$. Furthermore, the longest path in the circuit consists of at most n multiplexers and each multiplexer has a depth of 3. However, the longest path includes 3 gates for the first multiplexer and only two gates for all other multiplexers and therefore, the depth of the multiplexer circuit for g is at most $2 \cdot n + 1$.

The variable ordering of the BDDs during the verification of the multiplexer circuit is given by the order of the select inputs in the circuit. Other variable orderings may lead to an exponential verification complexity.

Theorem 2. *The multiplexer circuit for g is polynomially verifiable, if the circuit representation of f is polynomially verifiable and if the output is complemented for a polynomial amount δ of assignments.*

Proof. Every multiplexer with inputs g' and g'' and select input x_i can be represented by the ITE-operator: $ITE(x_i, g', g'')$. For the verification, we start by generating

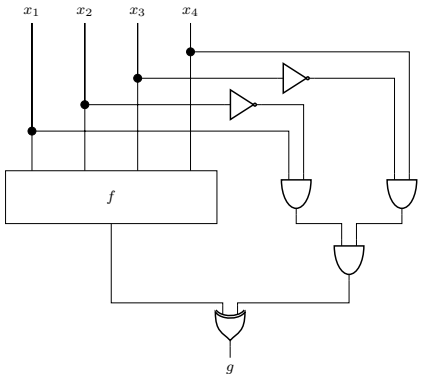


Fig. 3: XOR circuit for Figure 1(b)

BDDs for every multiplexer in the first layer of the circuit, where the inputs are 0 and 1. The resulting BDDs consist of three nodes of which two are terminal nodes. Using the ITE-operator, the BDDs for the multiplexers in the next layer are constructed. Here, every constructed BDD is at most as large as a subgraph of the BDD for g, where the root of the subgraph is the node which is represented by the current multiplexer in the multiplexer circuit. Therefore, the BDDs never exceed the size of the BDD for g during the construction process, as all intermediate results are subgraphs of the final BDD.

Thus, during the verification process of the multiplexer circuit for a function g, the size of the BDD does not exceed $|g|$ during its generation. As $|g| \leq |f| + n^{c+1}$ for some polynomial n^c, the BDD sizes during the verification process are always polynomial.

Furthermore, every operation $ITE(x_i, g', g'')$ can be carried out in time $|x_i| \cdot |g'| \cdot |g''|$, where g' and g'' are subgraphs of the BDD for g. Therefore, this circuit can be verified in time $\mathcal{O}(|g| \cdot |g|^2) = \mathcal{O}(|g|^3) \leq \mathcal{O}((|f| + n^{c+1})^3)$ as $|g|$ ITE operations have to be conducted.

The space complexity of the verification process of the multiplexer circuit is $\mathcal{O}(|g|^2) \leq \mathcal{O}((|f| + n^{c+1})^2)$, as there are $|g|$ intermediate results with a size of at most $|g|$. □

B. Polynomially Verifiable XOR Circuit

The second polynomially verifiable circuit architecture for g, which we call the XOR circuit, is based on the polynomially verifiable circuit for the function f. Each minterm of an assignment for which the output has to be complemented is realized by a negation of all negated literals, followed by a logarithmic tree of AND gates. Using a logarithmic tree of XOR gates, all these minterms are then XORed together with the output of the circuit for the function f.

The resulting XOR circuit for the function shown in Figure 1(b) is shown in Figure 3. The function f which is shown in Figure 1(a) is realized with a polynomially verifiable circuit. The input variables of the negated literals $\overline{x}_2$ and $\overline{x}_3$ in the minterm $x_1\overline{x}_2\overline{x}_3x_4$ are negated with a NOT gate and then, the minterm is realized using a tree of AND gates. Finally, the minterm is XORed with f.

If the output has to be complemented for δ assignments, the number of gates of the XOR circuit is at most $G(f) + n + n\delta$. The circuit for f is used, resulting in $G(f)$ gates. Furthermore, up to n NOT gates, $(n-1) \cdot \delta$ AND gates and δ XOR gates are required. Note that the trees of AND gates for multiple minterms can result in isomorphic subcircuits, potentially resulting in fewer gates if the circuit is optimized.

As the circuit for f and the trees of AND gates for the minterms can be computed in parallel, followed by the tree of XOR gates, the depth of the XOR circuit is at most $max(D(f), \lceil \log(n) \rceil + 1) + \lceil \log(\delta + 1) \rceil$.

For the verification of the XOR circuit, the variable ordering of all BDDs is equal to the ordering which is used during the polynomial verification of the circuit for f. Again, other variable orderings may result in an exponential verification complexity.

Theorem 3. *The XOR circuit for g is polynomially verifiable, if the circuit representation of f is polynomially verifiable and if the output is complemented for a polynomial amount δ of assignments.*

Proof. As the circuit for f is polynomially verifiable, we can construct the BDD for this part of the XOR circuit in polynomial time and space. Furthermore, the outputs of the AND trees can be constructed polynomially, as every subfunction is a conjunction of literals where every literal is either in positive or negated form and has a linear amount of nodes. Thus, during the construction of the BDDs of the trees of AND gates, the BDDs are never larger than n nodes, where the BDDs contain at most one node for every variable.

From Theorem 1, we can conclude that every assignment for which f and g differ adds at most n nodes to the BDD. Therefore, the size of the BDD after each XOR gate is polynomial. More accurately, if the maximum BDD size during the verification process of the circuit for f is $|f|_{max}$, the maximum BDD size during the verification process of the XOR circuit is $|f|_{max} + n\delta$, as the maximum BDD size can either be the maximum BDD size during the verification process of the circuit for f or the BDD size during the XOR operations. As both δ and $|f|_{max}$ are polynomial, the BDDs during the verification process have a polynomial size.

Let $\mathcal{O}(n^d)$ be the time complexity of the verification process of the circuit for f. Then, the overall time complexity for the verification process is the sum of the verification time of the circuit for f, as well as the computation time for the BDDs of the NOT, AND and XOR gates. This results in an overall time complexity of $\mathcal{O}(n^d + n + n^3\delta + \delta \cdot (|f|_{max} + n\delta)^3) = \mathcal{O}(n^d + \delta \cdot (|f|_{max} + n\delta)^3)$, which is polynomial as well.

Similarly, if $\mathcal{O}(n^d)$ is the space complexity of the verification process of the circuit for f, the space complexity of the verification process of the XOR circuit is $\mathcal{O}(n^d + n\delta + \delta \cdot (|f|_{max} + n\delta)) = \mathcal{O}(n^d + \delta \cdot |f|_{max} + n\delta^2)$, which is again polynomial. □

C. Error Metrics

Let f and g be two functions with n inputs and m outputs, where a circuit for f is polynomially verifiable and the maximum BDD size during the verification process of the circuit for f is $|f|_{max}$. For the four presented error metrics, given the function f and an error bound, we determine the maximum size of the BDD during the verification process of the two circuits for g resulting from our proposed method. Furthermore, given a specific polynomial n^c, we determine

the error ϵ for which the BDD for g increases the BDD size for f by at most n^c nodes. Both results can be used to enhance the development process of approximate circuits such that the resulting circuits are polynomially verifiable. Note that we compute the maximum BDD sizes for each output individually and calculate the results for unnormalized errors. For all four error metrics, the proofs and bounds are similar and therefore, we exemplarily provide the proof for the bit threshold.

Let ϵ be the unnormalized bit threshold. Then, ϵ is the sum of output bits that are complemented over all assignments. Therefore, at most ϵ minterms have to be XORed with the outputs of f in order to generate g. Thus, during the verification process of the multiplexer circuit, up to ϵ minterms have to be XORed with one output and therefore, the BDD does not exceed the size $|f| + n\epsilon$, whereas the BDD size does not exceed $|f|_{max} + n\epsilon$ during the verification process of the XOR circuit. Thus, if ϵ scales polynomially with respect to the number of inputs, the verification can be performed in polynomial time and space.

Given a polynomial n^c, we can determine a lower bound for ϵ such that the BDD does not exceed the size $|f| + n^c$ during the verification process of the multiplexer circuit and $|f|_{max} + n^c$ during the verification process of the XOR circuit. As the BDD size during the verification process of the multiplexer circuit does not exceed $|f| + n\epsilon$, we can conclude that it does not exceed $|f| + n^c$ if $\epsilon \geq n^{c-1}$. Furthermore, the upper bound for the BDD size during the verification process of the XOR circuit is $|f|_{max} + n\epsilon$ and therefore, it does not exceed $|f|_{max} + n^c$ if $\epsilon \geq n^{c-1}$.

V. Experiments

For the evaluation of the polynomial upper bounds, we have implemented the verification for both presented circuit architectures, meaning the multiplexer circuit and the XOR circuit, using CUDD 3.0.0 [20]. We evaluate the area and delay for both circuits, as well as the maximum BDD size during the verification process of both circuits for 9 approximate functions. Here, every evaluated function g is an approximation of a function f. This approximation is generated by XORing a function f with a function h, where f is a function for which a polynomially verifiable circuit exists and where h has a polynomial amount of minterms with respect to the number of inputs. Thus, the amount of assignments for which the outputs of f and g differ is polynomial in the number of inputs. All functions for f, g and h have 16 inputs. We evaluate all combinations of three different functions for f and three different functions for h, resulting in 9 approximate functions g. For functions f with multiple outputs, every output is XORed with the respective function for h.

The first function f_1 is the 0 function, which is trivially polynomially verifiable. Furthermore, the second function f_2 is the symmetric function $S^{16}(8-13)$, which means that at least 8 inputs and at most 13 inputs have to be 1 for this function to evaluate to 1, which is also polynomially verifiable [18]. Finally, the last function f_3 is the adder function, which is realized with a CLA, which is polynomially verifiable as well [5]. The CLA is implemented without a carry-in and with two 8-bit inputs, resulting in 16 input and 9 output bits in total.

TABLE I: Results for the multiplexer circuits

f	h	δ	Upper bound	Max BDD	Gates	Depth
f_1	h_1	16	14	14	36	25
f_1	h_2	105	42	42	97	33
f_1	h_3	560	57	57	126	33
f_2	h_1	16	120	120	258	33
f_2	h_2	105	213	213	446	33
f_2	h_3	560	105	105	227	33
f_3	h_1	$9 \cdot 16$	68	68	546	33
f_3	h_2	$9 \cdot 105$	136	136	1145	33
f_3	h_3	$9 \cdot 560$	158	158	1321	33
Average			101.44	101.44	466.89	32.11

TABLE II: Results for the XOR circuits

f	h	δ	Upper bound	Max BDD	Gates	Depth
f_1	h_1	16	258	21	86	8
f_1	h_2	105	1682	42	346	11
f_1	h_3	560	8962	68	1414	14
f_2	h_1	16	348	120	552	25
f_2	h_2	105	1772	213	782	28
f_2	h_3	560	9052	134	1846	31
f_3	h_1	$9 \cdot 16$	282	73	183	10
f_3	h_2	$9 \cdot 105$	1706	136	459	13
f_3	h_3	$9 \cdot 560$	8986	169	1551	16
Average			3672	108.44	812.11	17.33

The first function h_1 is the function which evaluates to 1, if all variables after the $\log n$-th variable are set to 1, which has n minterms. The second function h_2 evaluates to 1 for the pattern where the first variables are set to 1, followed by variables which are set to 0, followed by variables which are set to 1. This function has less than n^2 minterms. Finally, h_3 has less than n^3 minterms and is defined as the symmetric function $S^{16}(3)$, meaning it evaluates to 1, if exactly 3 inputs are 1.

A. Polynomially Verifiable Circuits

Table I and Table II show the results for all multiplexer circuits and XOR circuits respectively. Both tables show the values for δ, the computed upper bounds for the BDD sizes, the maximum BDD sizes using the respective variable ordering, the number of gates and the depth of the respective circuit.

As can be seen in Table I, for all approximate functions g, the maximum BDD sizes during the verification process of the multiplexer circuits do not exceed the upper bound, since the upper bound is the size of the BDD for g. Therefore, the upper bound is sharp in all test cases. For the XOR circuit, the maximum BDD sizes do not exceed the upper bound either. However, here, the upper bound overestimates the maximum BDD sizes, as can be seen in Table II. This overestimation is due to the fact that, as shown in Theorem 1, XORing a function with a minterm of an assignment increases the BDD for that function by at most n nodes. In most cases however, the BDD size is increased by less than n nodes. As the main objective of the upper bound is that it is never exceeded and to prove the polynomial verifiability of the circuit, the computed upper bound is suitable even if it overestimates the results.

Furthermore, the maximum BDD size during the verification process of the XOR circuit can be larger than the maximum BDD size during the verification process of the respective multiplexer circuit. This can be seen in the last row of both tables, as the BDD size never exceeds the size of the final

TABLE III: Error metrics and upper bounds

f	h	Max MUX	Max XOR	bt	Bound	er	Bound	ace	Bound	mse	Bound
f_1	h_1	14	21	16	258	16	258	16	258	16	258
f_1	h_2	42	42	105	1682	105	1682	105	1682	105	1682
f_1	h_3	57	68	560	8962	560	8962	560	8962	560	8962
f_2	h_1	120	120	16	348	16	348	16	348	16	348
f_2	h_2	213	213	105	1772	105	1772	105	1772	105	1772
f_2	h_3	105	134	560	9052	560	9052	560	9052	560	9052
f_3	h_1	68	73	144	2330	16	282	8048	128,794	4,048,304	64,772,890
f_3	h_2	136	136	945	15,146	105	1706	34,053	368,874	13,487,057	215,792,938
f_3	h_3	158	169	5040	80,666	560	8986	180,108	2,881,754	67,240,320	1,075,845,146

BDD during the verification process of the multiplexer circuit. However, for some test examples, the maximum BDD size is equal, e.g. in the second row of both tables. For these examples, the BDD sizes during the verification process of the XOR circuit do not exceed the size of the final BDD either, resulting in an equal maximum BDD size for both circuits.

Comparing the area and delay of both circuits, it is apparent that the XOR circuits require a higher amount of gates on average, as all gates from the base circuit have to be implemented, as well as NOT, AND and XOR gates. On average, the multiplexer circuit has 466.89 gates, whereas the average XOR circuit has 812.11 gates. However, the multiplexer circuits have a higher depth, as the depth of the multiplexer is linear in the amount of inputs, whereas the depth of the XOR circuit can have a logarithmic complexity, if the circuit for f has a logarithmic depth. This results in an average depth of 32.11 for the multiplexer circuits and 17.33 for the XOR circuits. Thus, the multiplexer circuits are preferable if a more area-efficient circuit is required, whereas the XOR circuits are more suitable if a low delay is essential.

B. Error Metrics

In addition to the upper bounds for the BDD sizes during the verification process of both circuits, we also evaluate the upper bounds for the BDD sizes given the four presented error metrics, meaning the bit threshold, error rate, average-case error and mean-squared error. The results are shown in Table III, which shows the maximum BDD sizes during the verification process of both circuits for all approximate functions g. Furthermore, for all four error metrics, the computed bounds for the BDD size, as given in Section IV-C, are shown as well. All error metrics are given unnormalized, meaning they are not divided by 2^n. The computed upper bounds for the BDD sizes are equal for both circuits, as $|f| = |f|_{max}$ for f_1, f_2 and f_3. For all approximate functions and all error metrics, the maximum BDD sizes during the verification process of both circuit architectures are lower than the computed upper bounds.

As can be seen in the first six rows, all error metrics have the same value for each single-output function, as well as the same resulting upper bound. However, for the functions with multiple outputs, which are shown in the last three rows, the error metrics and therefore the upper bounds differ. As can be seen, the bit threshold, the average-case error and the mean-squared error result in significantly larger bounds compared to the error rate. Thus, for functions with multiple outputs, the error rate provides the most accurate upper bound for the maximum BDD size during the verification process.

VI. Conclusion

In this paper, we have shown that if a function with a polynomially verifiable circuit is approximated such that a polynomial amount of input assignments result in a different output, there exist at least two polynomially verifiable circuits for the resulting function, meaning their verification is guaranteed to run in polynomial time and space with respect to the number of inputs. Polynomial upper bounds for the BDD sizes, as well as the time and space complexity were given for the verification process of both circuits. Additionally, as approximate functions are typically evaluated using error metrics, we have given upper bounds for the BDD sizes given the error between both functions. Using our results, the development of approximate circuits can be enhanced such that the resulting circuit is guaranteed to be polynomially verifiable and that the maximum BDD size during the verification process, as well as the verification time and space can be estimated beforehand. Furthermore, we have experimentally evaluated the bounds for the BDD sizes during the verification process.

References

[1] E. Seligman, T. Schubert, and M. V. A. K. Kumar, *Formal verification*. Boston: Morgan Kaufmann, 2015.

[2] R. Bryant, "Symbolic simulation-techniques and applications," in *DAC*, 1990, pp. 517–521.

[3] ——, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[4] R. E. Bryant and Y.-A. Chen, "Verification of arithmetic circuits with binary moment diagrams," in *DAC*, 1995, pp. 535–541.

[5] R. Drechsler, "Polyadd: Polynomial formal verification of adder circuits," in *DDECS*, 2021, pp. 99–104.

[6] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *ETS*, 2013, pp. 1–6.

[7] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *ICCAD*, 2011, pp. 667–673.

[8] R. Hrbacek, V. Mrazek, and Z. Vasicek, "Automatic design of approximate circuits by means of multi-objective evolutionary algorithms," in *DTIS*, 2016, pp. 1–6.

[9] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: Systematic logic synthesis of approximate circuits," in *DAC*, 2012, pp. 796–801.

[10] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *DATE*, 2017, pp. 258–261.

[11] I. Wegener, *The Complexity of Boolean Functions*. John Wiley & Sons, 1987.

[12] S. Fröhlich, S. Shirinzadeh, and R. Drechsler, "Multiply-accumulate enhanced BDD-based logic synthesis on RRAM crossbars," in *ISCAS*, 2020, pp. 1–5.

[13] K. Brace, R. Rudell, and R. Bryant, "Efficient implementation of a BDD package," in *DAC*, 1990, pp. 40–45.

[14] A. Mahzoon and R. Drechsler, "Late breaking results: Polynomial formal verification of fast adders," in *DAC*, 2021, pp. 1376–1377.

[15] ——, "Polynomial formal verification of prefix adders," in *ATS*, 2021, pp. 85–90.

[16] M. Keim, R. Drechsler, B. Becker, M. Martin, and P. Molitor, "Polynomial formal verification of multipliers," *Formal Methods Syst. Des.*, vol. 22, no. 1, pp. 39–58, 2003.

[17] M. Barhoush, A. Mahzoon, and R. Drechsler, "Polynomial word-level verification of arithmetic circuits," in *MEMOCODE*, 2021, pp. 1–9.

[18] R. Drechsler and C. Dominik, "Edge verification: Ensuring correctness under resource constraints," in *SBCCI*, 2021, pp. 1–6.

[19] R. Drechsler, "Polynomial circuit verification using bdds," in *ICEECCOT*, 2021, pp. 49–52.

[20] "CUDD 3.0.0," https://github.com/ivmai/cudd, accessed: 2021-10-15.