Polynomial Formal Verification of KFDD Circuits

Martha Schnieber
University of Bremen
Bremen, Germany
schnieber@uni-bremen.de

Rolf Drechsler
University of Bremen
Cyber-Physical Systems, DFKI GmbH
Bremen, Germany
drechsler@uni-bremen.de

ABSTRACT

As the complexity of digital circuits increases, their verification poses an increasingly difficult challenge. Simulation techniques cannot fully guarantee the correctness of a circuit and thus, formal verification techniques (such as BDDs or SAT) have to be used. However, their application can generally require exponential time and space. Consequently, the verification complexity of several circuits has been researched in recent years. Among other circuits, it has been proven that circuits derived from BDDs can be verified efficiently in polynomial time and space. However, for some functions, circuits derived from KFDDs have at most the same size and can even be exponentially smaller than BDD circuits. In this paper, we show that the verification complexity of KFDD circuits is linear and improve the previously proven upper bound for BDD circuits. The verification is carried out using KFDDs, where we give linear upper bounds for the KFDD size during the verification process, as well as for the overall time complexity. The theoretical results presented in this paper are supported by an experimental evaluation verifying several KFDD circuits.

CCS CONCEPTS

• **Hardware → Functional verification**.

KEYWORDS

polynomial formal verification, complexity, kronecker functional decision diagrams

ACM Reference Format:
Martha Schnieber and Rolf Drechsler. 2023. Polynomial Formal Verification of KFDD Circuits. In *21st ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE '23), September 21–22, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3610579.3611080

1 INTRODUCTION

During the design and synthesis of digital circuits, numerous bugs can appear, leading to faulty circuits on physical chips. Therefore, the verification of circuits remains an essential task for their design process. However, verification approaches like discrete simulation fail to be able to fully guarantee a correct design, as it is not feasible to test the circuit for all possible input assignments. Thus, only formal verification methods can formally guarantee the correctness of a circuit. Several formal verification techniques can be applied, such as techniques based on *Boolean Satisfiability* (SAT) [20], *Symbolic Computer Algebra* (SCA) [1, 24] or *Binary Decision Diagrams* (BDDs) [5], [11]. Apart from BDDs, other kinds of decision diagrams can be employed as well, such as *Kronecker Functional Decision Diagrams* (KFDDs) [16, 12] or *Binary Moment Diagrams* (BMDs) [8, 13].

While formal verification methods like decision diagrams can be used to verify a variety of circuits, formally verifying a circuit can generally have an exponential time and space complexity, potentially causing the verification process to fail due to time or space constraints. Even if the formal representation of the final output has a polynomial size, they can reach an exponential size during the verification process. For multipliers, it has been proven that the verification using BDDs always requires exponential time and space [6]. Thus, the possibility of an efficient verification of specific circuit classes using various verification methods has been researched in the past years. Here, *Polynomial Formal Verification* (PFV) has been established [9, 15], where it was proven that for several circuits and verification methods, the verification can be carried out in polynomial time and space. Some of the circuits included in this class of polynomially verifiable circuits are several adders [9, 23], [21], multipliers [19], tree-like circuits [10], integer arithmetic circuits [1] and circuits computing symmetric functions [14]. The research results on verification complexity are beneficial during design-time, as the scalability of using specific methods to verify circuit classes can be estimated beforehand, as well as the verification time and space. Furthermore, the design of circuits can be adjusted, such that the resulting circuit can be verified efficiently.

A circuit for a specific function can be derived from its BDD, where each node is replaced by a multiplexer [2]. In recent years, it has been shown that BDD circuits can be verified polynomially regarding the circuit size [10], where the circuit size directly depends on the BDD size. For some Boolean functions however, more efficient representations than BDDs exist. For some functions, the KFDD can even be exponentially smaller than the respective BDD [3]. Using multiplexers, AND, XOR and NOT gates, circuits can be derived from KFDDs as well [17]. As the circuit size for KFDD circuits directly depends on the KFDD size as well, the resulting KFDD circuit can also be exponentially smaller than the respective BDD circuit. Despite the potential decrease of the circuit size, it has not been researched yet how the employment of KFDDs affects the verification complexity. In general, the application of operations like AND and OR to KFDDs can require an exponential amount of steps [12]. Nonetheless, in this paper, we prove that circuits derived from KFDDs can be formally verified in linear time and space with respect to the circuit size. By this, we also improve the bound for BDD circuits given in [10]. For the verification, symbolic simulation

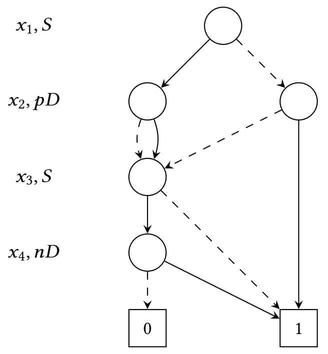


Figure 1: Example of a reduced and ordered KFDD

is used. We give upper bounds for the maximum KFDD size during the verification process and calculate the resulting linear time complexity. In addition to the theoretical results, we evaluate the upper bounds for the verification of 10,000 random KFDD circuits. The results of this paper also hold for BDDs, read-once KFDDs and Pseudo-KFDDs.

2 PRELIMINARIES

This section provides an overview of KFDDs, operations on KFDDs, circuits derived from KFDDs and the formal verification of circuits using symbolic simulation.

2.1 KFDDs

A KFDD [16, 12] is a directed acyclic graph with a root node and two terminal nodes that represent the values 0 and 1, where every non-terminal node has two outgoing edges and is associated with a variable. If the KFDD is ordered, the variables occur in the same order in every path from the root to a terminal node. Furthermore, a KFDD is reduced if it consists of a minimal amount of nodes. In the remainder of this paper, we operate on ordered and reduced KFDDs. We denote the size of an ordered and reduced KFDD F as $|F|$.

For KFDDs, three different decomposition types can be used to expand a node. Let f_i^0 be the cofactor of f, where x_i is set to 0 and analogously, let f_i^1 be the cofactor of f, where x_i is set to 1. The *Shannon* (S) decomposition expands the node for f with

$$f = \overline{x}_i f_i^0 + x_i f_i^1. \tag{1}$$

Furthermore, let f_i^2 be defined as the XOR operation performed on both cofactors $f_i^2 = f_i^0 \oplus f_i^1$. Then, the *positive Davio* (pD) decomposition is defined as

$$f = f_i^0 \oplus x_i f_i^2. \tag{2}$$

Analogously, the *negative Davio* (nD) decomposition is defined as

$$f = f_i^1 \oplus \overline{x}_i f_i^2. \tag{3}$$

If BDDs are used, every variable is decomposed using the Shannon decomposition. However, for KFDDs, every variable can be decomposed using one of the three decomposition types, which can reduce the size of the decision diagram exponentially for some functions [3]. The *Decomposition Type List* (DTL) of a KFDD specifies the decomposition type chosen for each variable. Figure 1 shows an

Algorithm 1: XOR Operation

```
Input  : $F, G$
Output : $F \oplus G$
1  if terminal case or $(F, G) \in$ computed-table then
2      return result
3  else
4      let $v$ be the top node of $F, G$
5      $low(v)$ = XOR($F_{low(v)}, G_{low(v)}$)
6      $high(v)$ = XOR($F_{high(v)}, G_{high(v)}$)
7      if Shannon(v) then
8          if $low(v) == high(v)$ then
9              return $low(v)$
10     else
11         if $high(v) == 0$ then
12             return $low(v)$
13     $R$ = find-or-add-unique-table($v, low(v), high(v)$)
14     insert-computed-table($F, G, R$)
15     return $R$
```

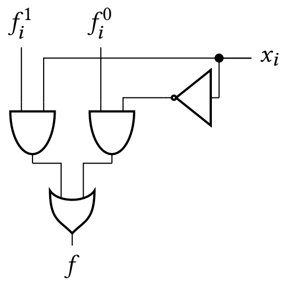
Algorithm 2: AND Operation

```
Input  : $F, G$
Output : $F \cdot G$
1  if terminal case or $(F, G) \in$ computed-table then
2      return result
3  else
4      let $v$ be the top node of $F, G$
5      $low(v)$ = AND($F_{low(v)}, G_{low(v)}$)
6      if Shannon(v) then
7          $high(v)$ = AND($F_{high(v)}, G_{high(v)}$)
8          if $low(v) == high(v)$ then
9              return $low(v)$
10     else
11         $high(v)$ = AND($F_{high(v)}, G_{high(v)}$) $\oplus$
            AND($F_{low(v)}, G_{high(v)}$) $\oplus$ AND($F_{high(v)}, G_{low(v)}$)
12         if $high(v) == 0$ then
13             return $low(v)$
14     $R$ = find-or-add-unique-table($v, low(v), high(v)$)
15     insert-computed-table($F, G, R$)
16     return $R$
```

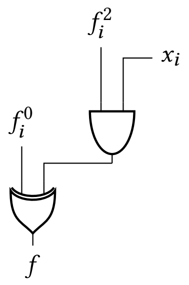
example of a reduced and ordered KFDD with the variable ordering (x_1, x_2, x_3, x_4), where the DTL (S, pD, S, nD) is used to specify the decomposition type for every variable.

Let f and g be two functions and let F and G be their respective KFDDs. The execution of the XOR operation on F and G has a quadratic time and space complexity of $O(|F| \cdot |G|)$ regarding the size of both KFDDs F and G, assuming an optimal hashing in $O(1)$. Similar to the XOR operation on BDDs, the XOR operation for nodes with the Shannon decomposition can be computed recursively as follows:

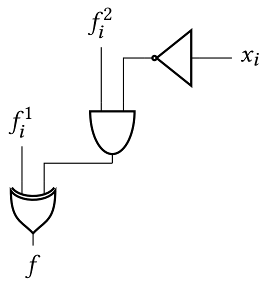
$$f \oplus g = \overline{x}_i \cdot (f_i^0 \oplus g_i^0) + x_i \cdot (f_i^1 \oplus g_i^1) \tag{4}$$



(a) Shannon decomposition



(b) Positive Davio decomposition



(c) Negative Davio decomposition

Figure 2: Circuits for the three decomposition types S, pD and nD

For nodes with the positive or negative Davio decomposition, the computation of the XOR operation is adjusted according to Equations (2) and (3), respectively. The XOR operation for the positive Davio decomposition is defined as follows:

$$\begin{aligned} f \oplus g &= (f_i^0 \oplus x_i f_i^2) \oplus (g_i^0 \oplus x_i g_i^2) \\ &= (f_i^0 \oplus g_i^0) \oplus x_i \cdot (f_i^2 \oplus g_i^2) \end{aligned} \tag{5}$$

To realize the XOR operation for the negative Davio decomposition, the variable and the cofactors have to be adjusted as in Equation (3):

$$\begin{aligned} f \oplus g &= (f_i^1 \oplus \overline{x}_i f_i^2) \oplus (g_i^1 \oplus \overline{x}_i g_i^2) \\ &= (f_i^1 \oplus g_i^1) \oplus \overline{x}_i \cdot (f_i^2 \oplus g_i^2) \end{aligned} \tag{6}$$

The algorithm for the XOR operation with the input KFDDs F and G is shown in Algorithm 1. Here, the result is returned immediately if a terminal case is reached or if the result has already been computed in an earlier step. A terminal case is reached if one of the inputs is a terminal. Otherwise, the two cofactors $low(v)$ and $high(v)$ are computed according to Equations (4) and (5), followed by a reduction based on the decomposition type.

Other operations may have an exponential time and space complexity on KFDDs, such as the AND operation, which is shown in Algorithm 2. For the Shannon decomposition, the cofactors $low(v)$ and $high(v)$ are computed similar to the XOR operation. However, for the positive and negative Davio decomposition, the AND operation has an exponential worst case time complexity. For the positive Davio decomposition, it is computed as follows:

$$\begin{aligned} f \cdot g &= (f_i^0 \oplus x_i f_i^2) \cdot (g_i^0 \oplus x_i g_i^2) \\ &= (f_i^0 \cdot g_i^0) \oplus x_i((f_i^2 \cdot g_i^2) \oplus (f_i^0 \cdot g_i^2) \oplus (f_i^2 \cdot g_i^0)) \end{aligned} \tag{7}$$

Again, for the negative Davio decomposition, the variable and the cofactors have to be adjusted according to Equation (3):

$$\begin{aligned} f \cdot g &= (f_i^1 \oplus \overline{x}_i f_i^2) \cdot (g_i^1 \oplus \overline{x}_i g_i^2) \\ &= (f_i^1 \cdot g_i^1) \oplus \overline{x}_i((f_i^2 \cdot g_i^2) \oplus (f_i^1 \cdot g_i^2) \oplus (f_i^2 \cdot g_i^1)) \end{aligned} \tag{8}$$

Therefore, $high(v)$ is computed differently for the Davio decomposition types, as shown in Line 11 of Algorithm 2.

The OR operation is computed analogously to the AND operation. Furthermore, the NOT operation can be realized using the XOR operation, as $\overline{f} = 1 \oplus f$, and is therefore efficient.

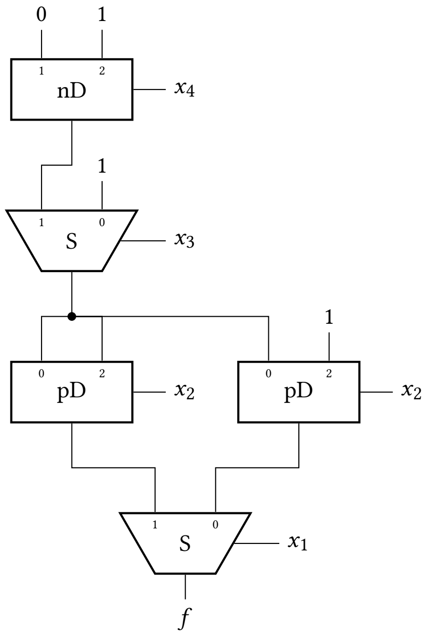


Figure 3: Example of a KFDD Circuit resulting from Figure 1

2.2 KFDD Circuits

To synthesize Boolean functions, their decision diagrams can be leveraged. BDD circuits are created by replacing the nodes in the BDD with multiplexers [2]. Figure 2(a) shows a multiplexer circuit, consisting of a NOT gate, two AND gates and an OR gate. However, as BDDs can be exponentially larger than KFDDs for some functions [3], KFDD circuits can drastically reduce the area. Here, every node that is decomposed using the Shannon decomposition is replaced by a multiplexer, whereas each positive Davio node is replaced by an XOR gate and an AND gate, as shown in Figure 2(b). For the negative Davio nodes, an additional NOT gate is required, as is illustrated in Figure 2(c) [17]. Figure 3 shows the KFDD circuit corresponding to the KFDD in Figure 1, where the Shannon nodes are replaced by multiplexers and the Davio nodes are realized using NOT, AND and XOR gates.

2.3 Symbolic Simulation

To formally verify a circuit using KFDDs, its KFDD can be computed according to the circuit design using symbolic simulation. First, the KFDDs for the input variables are created, which consist of two or three nodes in total, depending on the decomposition type. Then, the KFDDs for the outputs of the following gates are computed by applying the corresponding operations to the input KFDDs of the respective gates. This process is repeated until the output KFDD of the final gate(s) is created [7]. The resulting KFDD can then be compared to the KFDD of the specification. As KFDDs are canonical given a variable ordering and a DTL [12], the circuit correctly implements the specification if and only if the computed KFDD for the circuit is equal to the KFDD of the specification.

3 RELATED WORK

The verification complexity of several circuit classes has already been researched. Here, many approaches leverage BDDs, which can be used to polynomially verify multiple adder architectures, such as the ripple carry adder, the conditional sum adder and the carry look ahead adder [9]. The authors of [23] furthermore analyse the polynomial formal verification of several prefix adders.

In [10], it has been shown that circuits derived from BDDs can be verified in polynomial time and space regarding the circuit size. Here, it has also been proven that tree-like circuits can be efficiently verified using BDDs as well, where the gates of the tree-like circuits are limited to basic gates. In [22], this result was extended to include other gates such as XOR gates, where BDDs and KFDDs are used for the verification. Other decision diagrams can also be employed for verification, such as *BMDs, which can be used for the polynomial formal verification of Wallace-tree like multipliers [19]. Additionally, it has been proven that integer arithmetic circuits can be verified in linear space and quadratic time regarding the circuit size using BMDs and SCA [1].

Thus, the verification complexity of several circuits is already known, however, the verification complexity of circuits derived from KFDDs has not been investigated yet.

4 POLYNOMIAL VERIFICATION

To prove the polynomial verifiability of KFDD circuits, we first prove in Section 4.1 that the KFDDs resulting from each step of the symbolic simulation are at most as large as the KFDD of the final function. For the verification, the variable ordering and the DTL are chosen according to the KFDD circuit. In Section 4.2, we then prove that each step of the symbolic simulation can be carried out in constant time, leading to an overall linear time complexity. The linear space complexity of the verification process directly results from the linear time complexity.

4.1 Maximum KFDD Size

In the following, the upper bound for the KFDD size during the verification process of the KFDD circuit for a function f is proven to be $|F|$, if $|F| \geq 3$, where all intermediate KFDDs are considered. If $|F| < 3$, all intermediate results trivially have a constant size.

Theorem 1. *Let $|F| \geq 3$ be the size of the KFDD for a function f, where the variable ordering and the DTL are chosen as given by the KFDD circuit. Furthermore, let $|F'|$ be the size of an intermediate KFDD during the verification process. Then, it holds that $|F'| \leq |F|$.*

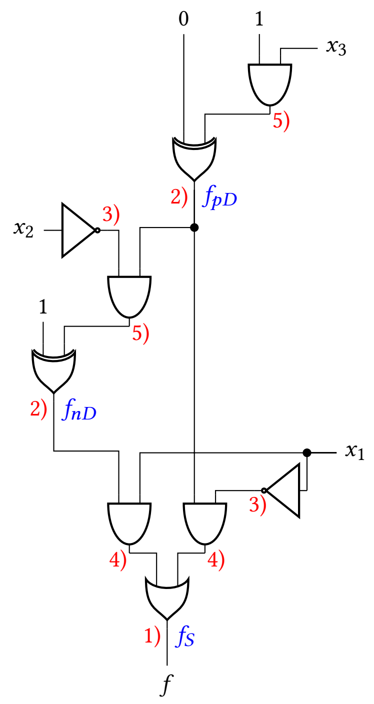
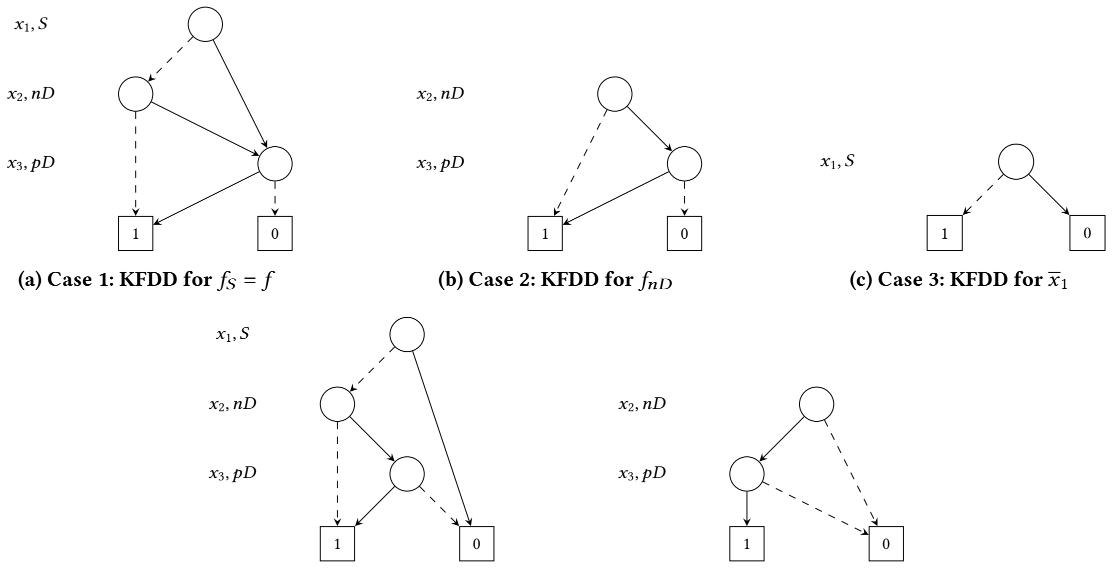


Figure 4: Example of the five cases for a KFDD Circuit

Proof. For the proof, we differentiate between all types of gates, meaning all gates in the multiplexers replacing the Shannon nodes, as well as the NOT, AND and XOR gates realizing Davio nodes. Here, five different cases for F' are examined, which are marked in Figure 4 for an exemplary KFDD circuit, where each gate output is marked with the respective case. Furthermore, the intermediate results f_{pD}, f_{nD} and f_S for the decomposition types pD, nD and S are marked. Exemplary KFDDs for the five cases during the verification process of the KFDD circuit displayed in Figure 4 are shown in Figure 5.

(1) Let F' be the KFDD of a multiplexer output, meaning the output of its OR gate. Then, F' is a subgraph of F, as the function f' corresponding to the KFDD F' is the input of a node in the final KFDD. Thus, $|F'| \leq |F|$. An example can be seen in Figure 5(a), where the KFDD for f_S is shown, which is also the final KFDD for the function f.
(2) Let F' be the KFDD for the output of an XOR gate for the negative or positive Davio decomposition. The KFDD for the output of the XOR gate of a negative Davio node is displayed in Figure 5(b). Again, F' is a subgraph of F and thus, it holds that $|F'| \leq |F|$.
(3) Now let F' be the KFDD of a NOT gate from the realization of Shannon or negative Davio nodes. The KFDD of the NOT gate is trivially constant, as a single variable x_i is negated, resulting in a KFDD of size $|F'| \leq 3$, which is shown in Figure 5(c). Thus, $|F'| \leq |F|$, as $|F| \geq 3$.



(a) Case 1: KFDD for $f_S = f$ **(b) Case 2: KFDD for f_{nD}** **(c) Case 3: KFDD for $\overline{x}_1$**

(d) Case 4: KFDD for $\overline{x}_1 \cdot f_{nD}$ **(e) Case 5: KFDD for $\overline{x}_2 \cdot f_{pD}$**

Figure 5: Illustration of the KFDDs for the five cases

(4) Let F' be the KFDD for the output of one of the AND gates in the multiplexer computing either $x_i \cdot g_i^1$ or $\overline{x}_i \cdot g_i^0$, where the KFDDs for g_i^1 and g_i^0 were examined in either Case 1 or Case 2. The AND operation merely adds one node to the KFDD of g_i^1 or g_i^0, as x_i is the topmost variable due to the bottom-up approach of the KFDD circuit. As the KFDDs of g_i^1 and g_i^0 are subgraphs of F, but are not equal to the final KFDD F, it holds that $|G_i^1| < |F|$ and $|G_i^0| < |F|$ and therefore $|F'| = |G_i^1| + 1 \leq |F|$ and $|F'| = |G_i^0| + 1 \leq |F|$. An example for the KFDD after the AND operation is shown in Figure 5(d).

(5) Finally, let F' be the KFDD for the output of the AND gate used for the positive or negative Davio decomposition. The AND gate computes either $x_i \cdot g_i^2$ for the positive Davio decomposition or $\overline{x}_i \cdot g_i^2$ for the negative Davio decomposition, where an example for $\overline{x}_i \cdot g_i^2$ is displayed in Figure 5(e). The KFDD of g_i^2 is again examined in either Case 1 or Case 2 and is therefore a subgraph of F, but not equal to F. As x_i is the topmost variable, only a single node is added to the KFDD and similar to Case 4, it holds that $|F'| = |G_i^2| + 1 \leq |F|$.

□

Thus, we have shown that the outputs of all gates can be represented by a KFDD which is smaller than or equal to the final KFDD.

4.2 Time Complexity

For the calculation of the time complexity, we first prove that each step of the symbolic simulation requires constant time, despite the general exponential time complexity of the OR and AND operations and the quadratic time complexity of the XOR and NOT operations.

Theorem 2. *Every operation of the symbolic simulation can be carried out in constant time.*

Proof. The same five cases described in Section 4.1 are considered, where we examine the complexity of the KFDD computation for each gate, meaning each step of the symbolic simulation. For all cases, let v be the topmost node of the KFDD resulting from the respective operation, where we calculate the complexity of computing $low(v)$ and $high(v)$. We show that the computation of each operation reaches a terminal case in a constant amount of steps.

(1) The KFDD corresponding to the OR gate in the multiplexer computing the function $f' = \overline{x}_i \cdot g_i^0 + x_i \cdot g_i^1$ is computed using the OR operation. Generally, the computation of the OR operation requires exponential time and space on KFDDs. However, in this case, a terminal case is reached in constant time for the computation of both $low(v)$ and $high(v)$:

$$\begin{aligned}
low(v) &= (\overline{x}_i \cdot g_i^0)_i^0 + (x_i \cdot g_i^1)_i^0 \\
&= 1 \cdot g_i^0 + 0 \cdot g_i^1 \\
&= g_i^0 + 0 = g_i^0 \\
high(v) &= (\overline{x}_i \cdot g_i^0)_i^1 + (x_i \cdot g_i^1)_i^1 \\
&= 0 \cdot g_i^0 + 1 \cdot g_i^1 \\
&= 0 + g_i^1 = g_i^1
\end{aligned} \tag{9}$$

Here, the inputs of the OR operation are inserted into the equation for the computation of $low(v)$ and $high(v)$. Note that $(\overline{x}_i \cdot g_i^0)_i^0 = 1 \cdot g_i^0$, as the value of x_i is already set to 0 in g_i^0 and therefore, g_i^0 doesn't depend on x_i. Analogously, $(x_i \cdot g_i^1)_i^0 = 0 \cdot g_i^1$. Thus, the OR operations that compute $low(v)$ and $high(v)$ reach a terminal case in constant time and therefore, also the KFDD for f' can be computed in constant time $O(1)$.

(2) The XOR gate $f' = g_i^0 \oplus x_i \cdot g_i^2$ for a positive Davio node can be calculated using the XOR operation.

$$\begin{aligned} low(v) &= (g_i^0)_i^0 \oplus (x_i \cdot g_i^2)_i^0 \\ &= g_i^0 \oplus 0 \cdot g_i^2 \\ &= g_i^0 \oplus 0 = g_i^0 \\ high(v) &= (g_i^0)_i^2 \oplus (x_i \cdot g_i^2)_i^2 \\ &= (g_i^0 \oplus g_i^0) \oplus ((x_i \cdot g_i^2)_i^0 \oplus (x_i \cdot g_i^2)_i^1) \\ &= 0 \oplus (0 \cdot g_i^2 \oplus 1 \cdot g_i^2) \\ &= 0 \oplus 0 \oplus g_i^2 = g_i^2 \end{aligned} \tag{10}$$

Here, $(g_i^0)_i^2 = (g_i^0 \oplus g_i^0)$, as f_i^2 is defined as $f_i^2 = f_i^0 \oplus f_i^1$ and the value of x_1 is already set to 0 in g_i^0 and therefore, it holds that $(g_i^0)_i^0 = (g_i^0)_i^1 = g_i^0$. For a negative Davio node, the XOR gate $f' = g_i^1 \oplus \overline{x}_i \cdot g_i^2$ is computed:

$$\begin{aligned} low(v) &= (g_i^1)_i^1 \oplus (\overline{x}_i \cdot g_i^2)_i^1 \\ &= g_i^1 \oplus 0 \cdot g_i^2 \\ &= g_i^1 \oplus 0 = g_i^1 \\ high(v) &= (g_i^1)_i^2 \oplus (\overline{x}_i \cdot g_i^2)_i^2 \\ &= (g_i^1 \oplus g_i^1) \oplus ((\overline{x}_i \cdot g_i^2)_i^0 \oplus (\overline{x}_i \cdot g_i^2)_i^1) \\ &= 0 \oplus (1 \cdot g_i^2 \oplus 0 \cdot g_i^2) \\ &= 0 \oplus g_i^2 \oplus 0 = g_i^2 \end{aligned} \tag{11}$$

Thus, for both the positive and negative Davio decomposition, the XOR operation reaches a terminal case in constant time $O(1)$.

(3) A NOT gate $f' = \overline{x}_i$ is computed using the XOR operation $1 \oplus x_i$, where the KFDD for the Shannon decomposition is computed as

$$\begin{aligned} low(v) &= 1 \oplus x_i^0 \\ &= 1 \oplus 0 = 1 \\ high(v) &= 1 \oplus x_i^1 \\ &= 1 \oplus 1 = 0 \end{aligned} \tag{12}$$

For the negative Davio decomposition, the NOT gates are computed analogously according to Equations (5) and (6). Thus, the computation of the NOT gates has a constant time complexity $O(1)$.

(4) The multiplexers contain two different AND gates. The KFDD for the AND gate $f' = x_i \cdot g_i^1$ located in the multiplexer can be computed as follows:

$$\begin{aligned} low(v) &= x_i^0 \cdot (g_i^1)_i^0 \\ &= 0 \cdot g_i^1 = 0 \\ high(v) &= x_i^1 \cdot (g_i^1)_i^1 \\ &= 1 \cdot g_i^1 = g_i^1 \end{aligned} \tag{13}$$

In general, the computation of the AND operation on KFDDs has an exponential time and space complexity. However, in this case, $low(v)$ is constant and $high(v)$ is set to g_i^1, meaning a terminal case is reached in constant time $O(1)$. Similarly, the AND operation for $f' = \overline{x}_i \cdot g_i^0$ is computed as:

$$\begin{aligned} low(v) &= \overline{x}_i^0 \cdot (g_i^0)_i^0 \\ &= 1 \cdot g_i^0 = g_i^0 \\ high(v) &= \overline{x}_i^1 \cdot (g_i^0)_i^1 \\ &= 0 \cdot g_i^0 = 0 \end{aligned} \tag{14}$$

Again, the computation terminates in constant time $O(1)$.

(5) For the AND gate of the realization of positive and negative Davio nodes, Equations (13) and (14) do not hold, as the Davio decomposition is used for the calculation of the AND operation. For the positive Davio decomposition, the AND operation $f' = x_i \cdot g_i^2$ is computed as follows:

$$\begin{aligned} low(v) &= x_i^0 \cdot (g_i^2)_i^0 \\ &= 0 \cdot g_i^2 = 0 \\ high(v) &= (x_i^2 \cdot (g_i^2)_i^2) \oplus (x_i^0 \cdot (g_i^2)_i^2) \oplus (x_i^2 \cdot (g_i^2)_i^0) \\ &= (1 \cdot 0) \oplus (0 \cdot 0) \oplus (1 \cdot g_i^2) \\ &= 0 \oplus 0 \oplus g_i^2 = g_i^2 \end{aligned} \tag{15}$$

Here, it holds that $x_i^2 = (x_i^0 \oplus x_i^1) = 0 \oplus 1 = 1$, whereas $(g_i^2)_i^2 = (g_i^2)_i^0 \oplus (g_i^2)_i^1 = g_i^2 \oplus g_i^2 = 0$. Thus, a terminal case is again reached in constant size $O(1)$. Similarly, for the negative Davio decomposition and the computation of $f' = \overline{x}_i \cdot g_i^2$:

$$\begin{aligned} low(v) &= \overline{x}_i^1 \cdot (g_i^2)_i^1 \\ &= 0 \cdot g_i^2 = 0 \\ high(v) &= (\overline{x}_i^2 \cdot (g_i^2)_i^2) \oplus (\overline{x}_i^1 \cdot (g_i^2)_i^2) \oplus (\overline{x}_i^2 \cdot (g_i^2)_i^1) \\ &= (1 \cdot 0) \oplus (0 \cdot 0) \oplus (1 \cdot g_i^2) \\ &= 0 \oplus 0 \oplus g_i^2 = g_i^2 \end{aligned} \tag{16}$$

As for the positive Davio decomposition, the AND operation for the negative Davio decomposition has a constant time complexity.

□

For every node in F, the KFDD circuit contains at most 4 gates (see Figure 2). Therefore, the circuit size c is linear to $|F|$, where $c \leq 4 \cdot |F|$. As one operation with a constant time complexity has to be carried out for every gate, the overall time complexity of the verification process is $O(c)$.

4.3 Other DD Types

The upper bounds presented in this paper also hold for other types of decision diagrams.

As BDDs are KFDDs, where only the Shannon decomposition is used, BDDs are a subclass of KFDDs and thus, the formal verification of BDDs has a linear time complexity $O(c)$ and the BDD sizes during the verification process are bounded by $|F|$.

The upper bounds for the maximum KFDD size during the verification process and for the time complexity can also be applied to the formal verification of circuits derived from read-once KFDDs, where the variable ordering can be encountered in different orders on different paths in the KFDD [4]. Generally, operations on

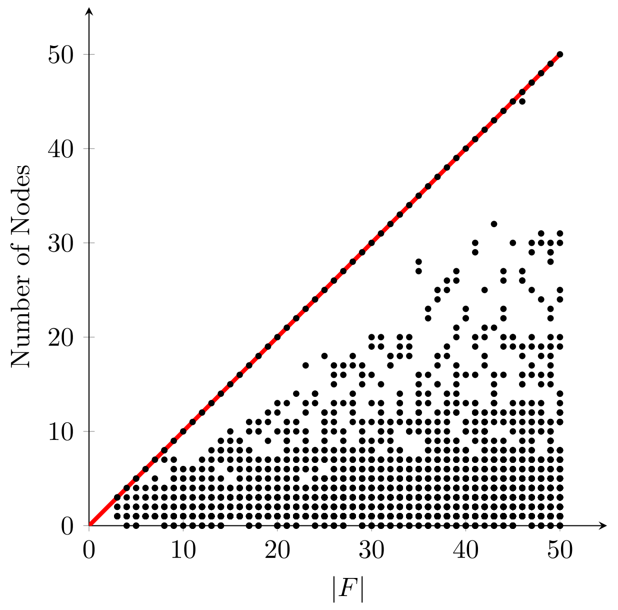


Figure 6: Size of KFDD during the verification process for $|F| \leq 50$

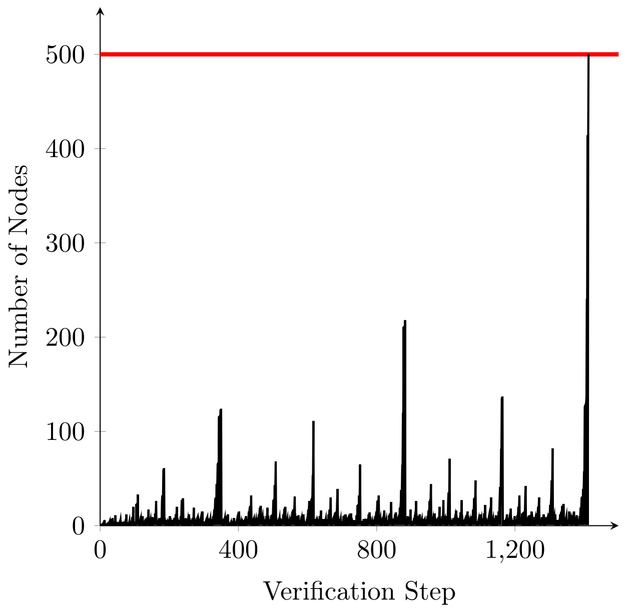


Figure 7: KFDD size during the verification steps of a KFDD circuit with $|F| = 500$

two KFDDs require the same variable ordering for both KFDDs. However, all operations during the verification of KFDD circuits terminate in constant time and only operate on the topmost node v, whereas $low(v)$ and $high(v)$ are merely set to g_i^0, g_i^1, g_i^2, 0 or 1 and therefore don't require further computation. Thus, the circuit can be polynomially verified using read-once KFDDs. This also holds for the verification of circuits derived from Pseudo-KFDDs, where nodes representing the same variable can have different decomposition types.

5 EXPERIMENTS

To evaluate the upper bounds proven in Section 4, we have implemented KFDDs in the wld package [18], a C++ library for decision diagrams. The Shannon decomposition, as well as the positive and negative Davio decomposition are implemented, along with the required operations AND, OR, XOR and NOT. The KFDD sizes after each step of the symbolic simulation and the verification time are evaluated for 10,000 KFDD circuits with up to 100,000 gates. The circuits are derived from KFDDs for random functions and with random variable orderings and DTLs.

Figure 6 shows the sizes of the KFDDs during the verification process for $|F| \leq 50$, meaning the KFDDs from which the circuits are derived have a size of up to 50 nodes. Every dot represent the KFDD size after an AND, OR, XOR or NOT operation. Here, the red line marks the upper bound proven in Section 4.1. As can be seen, the upper bound is met during the verification of all tested circuits, but is not surpassed for any example. These results also hold for all tested KFDD circuits with $|F| > 50$. Thus, the results obtained by the experimental evaluation support the upper bound for the KFDD sizes presented in Section 4.1.

The KFDD sizes during the verification process of a KFDD circuit with $|F| = 500$ and a circuit size of $c = 1414$ gates is shown in Figure 7. Again, the upper bound $|F| = 500$ for the KFDD size is marked in red, which is not surpassed during the verification process. The KFDD size only meets the upper bound during the last verification step, which results in the final KFDD of the circuit.

Finally, the verification time in milliseconds for random KFDD circuits with random variable ordering and DTLs and a circuit size of $c \leq 100,000$ is plotted in Figure 8. As can be seen, the experimental results support the linear time complexity proven in Section 4.2.

6 CONCLUSION

In this paper, we have shown that circuits derived from KFDDs can be formally verified in linear time and space with respect to the circuit size, despite the general exponential complexity of formally verifying circuits using BDD-based or KFDD-based techniques. We have given upper bounds for the size of all intermediate KFDDs during the verification process, as well as for the overall time complexity. The theoretical bounds presented in this paper were supported by an experimental evaluation on 10,000 circuits derived from random KFDDs with up to 100,000 gates. The upper bounds presented in this paper also hold for the formal verification of circuits derived from BDDs, read-once KFDDs and Pseudo-KFDDs.

ACKNOWLEDGMENTS

Parts of this work have been supported by DFG within the Reinhart Koselleck Project PolyVer: Polynomial Verification of Electronic Circuits (DR 287/36-1).

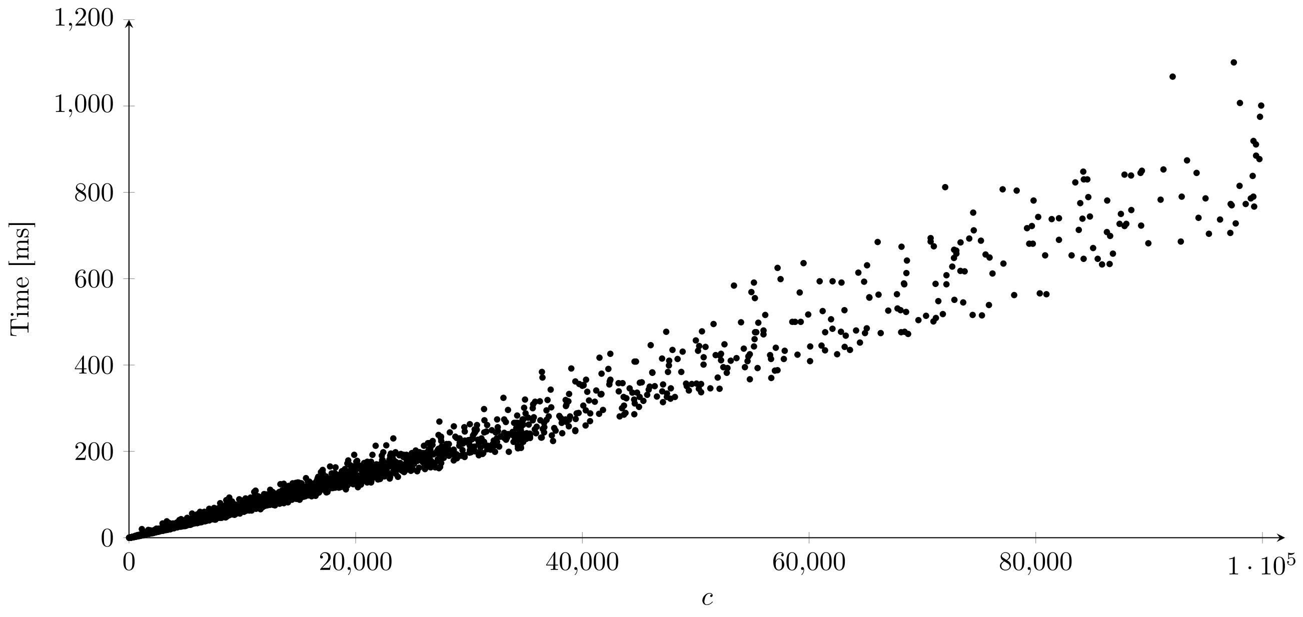


Figure 8: Verification time for random KFDD circuits with $c \leq 100,000$

REFERENCES

[1] Mohammed Barhoush, Alireza Mahzoon, and Rolf Drechsler. 2021. Polynomial word-level verification of arithmetic circuits. In *2021 19th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, 1–9.

[2] Bernd Becker. 1992. 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* (Lecture Notes in Computer Science). Alain Finkel and Matthias Jantzen, (Eds.) Vol. 577. Springer, 501–512.

[3] Bernd Becker, Rolf Drechsler, and Michael Theobald. 1997. On the expressive power of OKFDDs. *Formal Methods Syst. Des.*, 11, 1, 5–21.

[4] R.E. Bryant. 1995. Binary decision diagrams and beyond: enabling technologies for formal verification. In *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, 236–243.

[5] Randal E. Bryant. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35, 8, 677–691.

[6] Randal E. Bryant. 1991. On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication. *IEEE Transactions on Computers*, 40, 2, 205–213.

[7] Randal E. Bryant. 1990. Symbolic simulation - techniques and applications. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, 517–521.

[8] Randal E. Bryant and Yirng-An Chen. 1995. Verification of arithmetic circuits with binary moment diagrams. In *Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference (DAC)*, 535–541.

[9] Rolf Drechsler. 2021. PolyAdd: polynomial formal verification of adder circuits. In *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 99–104.

[10] Rolf Drechsler. 2021. Polynomial circuit verification using BDDs. In *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, 49–52.

[11] Rolf Drechsler and Bernd Becker. 2013. *Binary Decision Diagrams: Theory and Implementation*. Springer US.

[12] Rolf Drechsler and Bernd Becker. 1998. Ordered Kronecker functional decision diagrams-a data structure for representation and manipulation of Boolean functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17, 10, 965–973.

[13] Rolf Drechsler, Bernd Becker, and Stefan Ruppertz. 1997. The K*BMD: A verification data structure. *IEEE Design & Test of Computers*, 14, 2, 51–59.

[14] Rolf Drechsler and Caroline Dominik. 2021. Edge verification: ensuring correctness under resource constraints. In *2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*, 1–6.

[15] Rolf Drechsler and Alireza Mahzoon. 2022. Polynomial formal verification: ensuring correctness under resource constraints : (invited paper). In *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 1–9.

[16] Rolf Drechsler, Andisheh Sarabi, Michael Theobald, Bernd Becker, and Marek A. Perkowski. 1994. Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams. In *31st Design Automation Conference*, 415–419.

[17] Harry Hengster, Rolf Drechsler, Bernd Becker, Stefan Eckrich, and Tonja Pfeiffer. 1996. AND/EXOR-based synthesis of testable KFDD-circuits with small depth. In *Proceedings of the Fifth Asian Test Symposium (ATS'96)*, 148–154.

[18] Marc Herbstritt. 2003. Wld - a C++ library for decision diagrams. https://ira.informatik.uni-freiburg.de/software/wld/index.html. (2003).

[19] Martin Keim, Rolf Drechsler, Bernd Becker, Michael Martin, and Paul Molitor. 2003. Polynomial formal verification of multipliers. *Formal Methods in Systen Design*, 22, 1, 39–58.

[20] Andreas Kuehlmann, Viresh Paruthi, Florian Krohm, and Malay K. Ganai. 2002. Robust Boolean reasoning for equivalence checking and functional property verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21, 12, 1377–1394.

[21] Alireza Mahzoon and Rolf Drechsler. 2021. Late breaking results: polynomial formal verification of fast adders. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 1376–1377.

[22] Alireza Mahzoon and Rolf Drechsler. 2022. Polynomial formal verification of general tree-like circuits. In *2022 China Semiconductor Technology International Conference (CSTIC)*, 1–4.

[23] Alireza Mahzoon and Rolf Drechsler. 2021. Polynomial formal verification of prefix adders. In *2021 IEEE 30th Asian Test Symposium (ATS)*, 85–90.

[24] Alireza Mahzoon, Daniel Große, Christoph Scholl, and Rolf Drechsler. 2020. Towards formal verification of optimized and industrial multipliers. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 544–549.