Abstract—We consider the partial equivalence checking problem (PEC), i.e., checking whether a given partial implementation of a combinational circuit can (still) be extended to a complete design that is equivalent to a given full specification. To solve PEC, we give a linear transformation from PEC to the question whether a dependency quantified Boolean formula (DQBF) is satisfied.

Our novel algorithm to solve DQBF based on quantifier elimination can therefore be applied to solve PEC. We also present first experimental results showing the feasibility of our approach and the inaccuracy of QBF approximations, which are usually used for deciding the PEC so far.

I. INTRODUCTION

Verification of incomplete (or partial) system designs has received a lot of research efforts during the last decade [1], [2], [3], [4], [5]. In a partial system design some parts are so-called black boxes, i.e., modules of which the internal structure and behavior is not known. The concept of incomplete or partial designs can be used, 1) if parts of the system have not been implemented yet, 2) if the complexity of the verification task is too high and therefore some parts which are supposed not to influence the validity of some properties (e.g., multiplier or memory modules) have been removed to make verification feasible, and 3) if a designer wants to localize errors (then one can remove parts of the design and if for all possible implementations of the removed parts the error does not disappear, the remaining parts must be erroneous).

For circuits with black boxes (i.e. circuits where parts of the implementation are not (yet) available), we ask whether the implementation is equivalent to the specification for some realization of the black box parts. If this is the case, then we call the specification realizable. We call the corresponding problem the partial equivalence checking problem (PEC). If it turns out that there is no feasible extension, the already implemented parts are erroneous. This helps detecting errors in an early stage of a design.

As in [1], we assume that the specification and the partial design are combinational circuits, where the partial design additionally contains black boxes. There are also existing generalizations to sequential circuits (based on bounded model checking) [4], which we do not consider in this work.

Several approximate and exact methods to solve PEC are presented in [1]. If an approximate algorithm reports that there is no implementation for the black boxes, such that the specification can be realized, the desired specification is indeed not realizable. However, if such an algorithm is not able to prove unrealizability, this can be due to the approximate nature of the method, and the desired functionality may nevertheless be not realizable. The algorithms in [1] are based on solving SAT or QBF formulations of PEC. The SAT formulations are efficient to solve, but also rather inaccurate due to a coarse approximation. Their accuracy is improved in several steps, leading to a QBF formulation that can solve PEC for a single black box exactly. The authors of [1] additionally give an exact characterization of realizability of PEC for multiple black boxes. However, no feasible algorithmic method for solving the problem is given.

We show that for solving PEC with multiple black boxes exactly, an extension of QBF called dependency quantified Boolean formulae (DQBF) can be used. A DQBF is a propositional formula with a quantifier prefix containing Henkin-quantifiers [6]. In QBF an existentially quantified variable depends on all universal quantifiers appearing on the left of this variable in the prefix, defining a linear order on the variables. Contrarily, in DQBF the universally quantified variables on which an existential one depends are specified explicitly, allowing partially ordered quantifier prefixes.

In [7] the complexity of PEC is proven by showing that PEC is polynomially equivalent to DQBF. Therefore PEC lies in the same complexity class as DQBF, namely both are NEXPTIME-complete.

The first algorithmic approach that considers DQBF is stated in [8], but no detailed experimental evaluation is given. The algorithm is based on the QBF-extension QDLL [9] for the search-based DLL [10] algorithm for SAT. In [11] an algorithm is presented which evaluates QBF by encoding the function tables of the Skolem functions for the existential variables into a propositional SAT-formula. In principle this can also be applied to solve DQBF [12].

In the QBF domain 1.) variable elimination based algorithms tend to be beneficial and 2.) And-Inverter graphs (AIG) [13] as symbolic representation of circuit related verification problems turn out to be fruitful [14], [15]. Also there is no DQBF solver publicly available so far. Hence, in this paper we present a new approach for solving DQBF by using variable elimination [16], give some details of our implementation using AIGs, and show that our algorithm is sound and complete.

In the experiments with a prototypical implementation of our DQBF algorithm we check both artificial and realistic PEC instances for realizability using exact DQBF and approximate QBF formulations. The results show the inaccuracy of QBF in comparison with DQBF, demonstrating clearly that QBF gives
The remainder of the paper is structured as follows. In Section II we give the foundations of equivalence checking of partial designs and DQBF. In Section III we show how to translate a partial design into a DQBF. Then we state an algorithm to solve DQBF and give proof for its soundness and completeness in Section IV. Finally, we present first experimental results in Section V and conclude the paper in Section VI.

II. FOUNDATIONS

In this section we introduce dependency quantified Boolean formulae and equivalence checking for partial circuits.

A. Dependency Quantified Boolean Formulae

Let $V := \{v_1, \ldots, v_n\}$ be a set of Boolean variables. A variable assignment for $V$ is a function $\nu : V \rightarrow \{0, 1\}$. We denote the set of variable assignments for $V$ by $A_V$.

If $\varphi$ is a Boolean expression containing the variable $v \in V$, and $\psi$ an expression not containing $v$, we denote by $\varphi[\nu/v]$ the expression that results from $\varphi$ by replacing each occurrence of $v$ with $\psi$. Replacing each $v \in W \subseteq V$ by an expression $\psi_v$ is denoted by $\varphi[\psi_v/v \forall v \in W]$. In this case we require that the expressions $\psi_v$ do not contain any $w \in W$ such that the resulting formula does not depend on the replacement order.

In the following we use the symbols $x_1, \ldots, x_n$ for universally quantified variables and $y_1, \ldots, y_m$ for existentially quantified variables.

Definition 1: Let $\varphi$ be a Boolean formula, containing the Boolean variables $x_1, \ldots, x_n, y_1, \ldots, y_m$, and $D_1, \ldots, D_m \subseteq \{x_1, \ldots, x_n\}$ sets of Boolean variables. A dependency-quantified Boolean formula (DQBF) $\psi$ has the form:

$$\psi := \forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 (D_1) \exists y_2(D_2) \ldots \exists y_m(D_m) : \varphi.$$ 

The sets $D_i$ are called dependency sets of $y_i$, and the formula $\varphi$ is called the matrix of $\psi$.

We denote $V^x_\varphi = \{y_1, \ldots, y_m\}$ as the set of existential variables and $V^y_\varphi = \{x_1, \ldots, x_n\}$ the set of universal variables. If $y_i \in V^y_\varphi$ is an existential variable with dependency set $D_i$, a Skolem function for $y_i$ is a function $s_{y_i,D_i} : A_{D_i \setminus \{x\}} \rightarrow \{0, 1\}$. In this case, $\varphi[s_{y_i,D_i}/y_i]$ denotes the expression resulting from $\varphi$ by replacing each occurrence of $y_i$ by a Boolean expression for the Skolem function $s_{y_i,D_i}$.

For a variable $x \in D_i$ we denote by $s_{y_i,D_i|x=0}$ the Skolem function $s_{y_i,D_i \setminus \{x\}} : A_{D_i \setminus \{x\}} \rightarrow \{0, 1\}$ which results from $s_{y_i,D_i}$ by setting the variable $x$ constantly to 0. Accordingly for $s_{y_i,D_i|x=1}$.

Definition 2: Let $\psi := \forall x_1 \forall x_2 \ldots \forall x_n \exists y_1 (D_1) \exists y_2(D_2) \ldots \exists y_m(D_m) : \varphi$ be a DQBF. It is satisfied (written $\models \psi$) if and only if there are Skolem functions $s_{y_i,D_i}$ for $i = 1, \ldots, m$ such that $\varphi[s_{y_i,D_i}/y_i] \forall y_i \in V^y_\varphi$ is a tautology.

Note that DQBF is a generalization of quantified Boolean formulae (QBF). A QBF of Boolean variables $\{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ has the form: $\psi := \forall X_1 \exists Y_1 \ldots \forall X_n \exists Y_n : \varphi$, where $X_i \subseteq \{x_1, \ldots, x_n\}$ and $Y_i \subseteq \{y_1, \ldots, y_m\}$ are disjoint sets of variables. An existential variable $y_j \in Y_i$ always depends on all universal variables which are stated in the prefix left of $y_j$, and hence QBF is limited to a linear order of existential variable dependencies.

B. Partial Equivalence Checking

Equivalence checking considers the problem to decide whether two combinational circuits always produce the same outputs, given the same inputs. In case that one of the circuit is not completely given, but contains missing parts, so-called black boxes, we ask if there are implementations of the black boxes such that the two circuits become equivalent. If these implementations exist, we say the partial design is realizable, otherwise unrealizable. We call this the partial equivalence checking problem (PEC).

We introduce notations for partial combinational circuits $P$:

- $x_1, \ldots, x_n$ are the primary inputs of the circuit.
- $BB_1, \ldots, BB_m$ are the black boxes of the circuit.
- $I_i$ are the inputs of $BB_i$ $(i = 1, \ldots, m)$.
- $Y_i$ are the outputs of $BB_i$ $(i = 1, \ldots, m)$.
- $F_i(x_1, \ldots, x_n, Y_1, \ldots, Y_{i-1})$ is the vector of functions defining $I_i$.
- $R(x_1, \ldots, x_n, Y_1, \ldots, Y_m)$ is the output function of the circuit.

These notations are illustrated in Fig. 1.

Example 1: Consider $x_1 \oplus x_2$ as the specification and as an implementation a partial circuit with three gates and two black boxes as given in Fig. 2. The PEC problem asks: Is there a realization of both black boxes $BB_1$ and $BB_2$ such that the implementation is equivalent to the specification for every value of the inputs $x_1$ and $x_2$? To answer this question, we add an additional equivalence (or XOR) gate for each corresponding output of the specification and implementation and require that the outputs of these equivalence gates be constantly 1.

We will revisit this example in the next sections and show how to formulate an appropriate DQBF to determine whether the PEC is realizable.

1To guarantee that the circuit is combinational, we assume that $BB_1, \ldots, BB_m$ are in topological order, i.e., $BB_i$ does not depend on $BB_j$ for $i < j$. 

Fig. 1. Notation for partial designs
III. FROM PARTIAL DESIGNS TO QBF AND DQBF

A. DQBF Formulation

In order to decide PEC we specify a linear transformation from PEC to DQBF such that the resulting DQBF is satisfied if and only if the PEC is realizable. Together with a linear transformation in the opposite direction this not only yields a way to solve PEC but also a complexity-theoretic characterization.

Consider a PEC with black boxes \( BB_1, \ldots, BB_m \). We assume that the partial circuit and the specification have already been combined into a single circuit with the requirement that the output has to be constantly 1 (cf. Example 1). We follow the notations introduced in Section II-B, i.e., a black box \( BB_i \) has outputs \( \vec{Y}_i \) and inputs \( \vec{I}_i \) etc.

We assume w.l.o.g. that \( \vec{Y}_i \cap \vec{I}_j = \emptyset \) for all \( i, j \). That means no output of a black box is directly connected to an input of another black box. Since we need to use universal quantification for black box inputs, but existential quantification for black box outputs, having \( \vec{Y}_i \cap \vec{I}_j \neq \emptyset \) would lead to a contradiction. If \( \vec{Y}_i \cap \vec{I}_j \neq \emptyset \), we insert a buffer, “computing” the identity function, between \( BB_i \) and \( BB_j \) to separate the outputs of \( BB_i \) and the inputs of \( BB_j \). This does not change the functionality of the circuit and causes at most a linear blow-up.

We first construct the quantifier prefix of the DQBF. The primary inputs \( x_1, \ldots, x_n \) and the black box inputs \( \vec{I}_1, \ldots, \vec{I}_m \) are universally quantified, all other variables are existentially quantified. The dependency set of black box output \( y_{i,j} \) contains exactly the inputs \( \vec{I}_i \) of \( BB_i \). Hence the quantifier prefix is

\[
\forall x_1 \ldots \forall x_n \forall \vec{I}_1 \ldots \forall \vec{I}_m \exists \vec{Y}_1(\vec{I}_1) \ldots \exists \vec{Y}_m(\vec{I}_m).
\]

If the black boxes are not directly connected to the primary inputs but to internal signals, we have to take into account that not all possible combinations of values may arrive at the inputs of the black boxes. Since we use universal quantification for the black box inputs we have to ensure that our formula is satisfied if the value of the black box inputs \( \vec{I}_i \) deviates from the values obtained as a function \( \vec{F}_i(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_{i-1}) \).

\[
\varphi := (\vec{I}_1 \neq \vec{F}_1(x_1, \ldots, x_n)) \lor \cdots \lor (\vec{I}_m \neq \vec{F}_m(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_{m-1})) \lor R(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_m).
\]

By applying Tseitin transformation [17], which is essentially introducing auxiliary variables \( \vec{A} = (a_1, \ldots, a_p) \) for the internal signals of the circuit, one can obtain a CNF \( \varphi' \) that is satisfiability equivalent to \( \varphi \) and whose size is linear in the size of \( \varphi \). The variables in \( \vec{A} \) are existentially quantified in the quantifier prefix. Their dependency set encompasses all universally quantified variables.

The resulting DQBF is:

\[
\psi := \forall x_1 \ldots \forall x_n \exists \vec{Y}_1 \ldots \exists \vec{Y}_m (\vec{I}_1) \ldots \exists \vec{Y}_m(\vec{I}_m) \\
\exists \vec{A}(x_1, \ldots, x_n, \vec{I}_1, \ldots, \vec{I}_m) : \varphi'.
\]

The formula \( \psi \) is satisfied if and only if we can replace all \( \vec{Y}_i \) with Skolem functions \( s_{\vec{Y}_i, \vec{I}_i} \) such that \( \varphi' \) becomes a tautology. The Skolem functions \( s_{\vec{Y}_i, \vec{I}_i} \) exist if and only if there are implementations for the black boxes \( BB_i \) of the PEC, such that the specification is realized. Therefore any PEC can be translated with linear effort into a DQBF such that the PEC is realizable if and only if the DQBF is satisfied. Using Tseitin transformation, it is always possible to obtain a matrix of the QBF such that the PEC is realizable.

Lemma 1: Any PEC can be translated into an equivalent DQBF with linear effort.

We illustrate this transformation with an example (which for simplicity omits the conversion into CNF by Tseitin transformation):

Example 2: Consider again the PEC in Example 1. The corresponding DQBF is:

\[
\psi_{\text{DQBF}} = \forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : \\
((y_1 \lor y_2) \lor (x_1 \land \neg x_2)) \equiv (x_1 \oplus x_2).
\]

The primary inputs \( x_1 \) and \( x_2 \) get universally quantified. The input functions \( F_1 \) and \( F_2 \) are the identity functions of \( x_1 \) and \( x_2 \), respectively. The black box outputs are directly connected to the primary inputs and therefore we do not need additional variables for them. The black box outputs \( y_1 \) and \( y_2 \) are existentially quantified, whereby signal \( y_1 \) depends on \( x_1 \), since \( BB_1 \) has this signal as input. Accordingly for \( y_2 \) and \( x_2 \).

The three gates of the implementation (cf. Example 1) are represented by the Boolean expression \(((y_1 \lor y_2) \lor (x_1 \land \neg x_2))\).

For the matrix we require that either the inputs of the black boxes are inconsistently assigned (which is trivially not the case, thus we can omit the corresponding contradictions in the disjunctive formula (1) above) or the requirement has to be satisfied, i.e., the implementation has to be equal to \((x_1 \oplus x_2)\).
We will use this example again in the next section illustrating our proposed algorithm and thereby show whether the PEC is realizable.

Following Lemma 1, the formulation of a PEC as DQBF leads to an approach for solving this problem.

For the sake of completeness we state the following lemma, which allows to show the complexity of the PEC problem.

**Lemma 2:** Any DQBF can be translated into an equivalent PEC with linear effort.

**Proof:** Consider a DQBF
\[
\psi := \forall x_1 \exists y_2 \ldots \forall x_n \exists y_1(D_1) \exists y_2(D_2) \ldots \exists y_m(D_m) : \varphi.
\]

The matrix \( \varphi \) can be easily transformed into a combinational circuit \( C(\varphi) \) with inputs \( x_1, \ldots, x_n \) and \( y_1, \ldots, y_m \) by replacing the logical connectives \( \lor, \land, \neg \) with the corresponding gates. The input \( y_i \) of \( C(\varphi) \) is the output of a new black box \( BB_i \) in the PEC. The inputs of \( BB_i \) are exactly the signals in \( D_i \). Requiring the output of \( C(\varphi) \) to be constantly 1 is equivalent to comparing the incomplete circuit \( P \) with the 1-function as the specification \( S \). The translation is illustrated in Fig. 3. It can be shown that the resulting PEC is realizable if and only if the DQBF is satisfied [7]. Its size (number of gates, signals, and black boxes) is linear in the length of the DQBF.

We have shown that for each PEC \( P \) there is a DQBF \( \psi \) whose size is linear in the size of \( P \) such that \( P \) is realizable if and only if \( \psi \) is satisfied (cf. Lemma 1). Conversely, for each DQBF \( \psi \) there is a PEC \( P \) whose size is linear in the size of \( \psi \) such that \( \psi \) is true if and only if \( P \) is satisfiable (cf. Lemma 2). This is captured in the following theorem:

**Theorem 1:** PEC and DQBF are polynomially equivalent.

Finally we can state the following corollary using the known complexity class of DQBF:

**Corollary 1:** PEC is NEXPTIME-complete.

**Proof:** Since DQBF is NEXPTIME-complete [18] and PEC and DQBF are polynomially equivalent (Theorem 1), PEC is also NEXPTIME-complete.

**B. QBF Approximations**

In [1] QBF formulations for PEC have been defined. Here we will show the relationship between QBF and DQBF formulations.

**Definition 3:** Let
\[
\psi_{\text{QBF}} := \forall x_1 \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_m(D_m) : \varphi
\]

be a DQBF and
\[
\psi := \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \exists y_k : \varphi
\]
a QBF with the same matrix, such that \( \{ X_i \subseteq \{ x_1, \ldots, x_n \} \mid i = 1, \ldots, k \} \) is a partition of the universal and \( \{ Y_i \subseteq \{ y_1, \ldots, y_m \} \mid i = 1, \ldots, k \} \) a partition of the existential variables. \( \psi_{\text{QBF}} \) is an approximation of \( \psi_{\text{DQBF}} \) (written \( \psi_{\text{DQBF}} \preceq \psi_{\text{QBF}} \)) if \( \psi_i \in Y_j \) implies \( D_i \subseteq \bigcup_{i=1}^{m} X_\ell \) for all \( i = 1, \ldots, m \).

That means, \( \psi_{\text{QBF}} \) is an approximation of \( \psi_{\text{QBF}} \) if for all existential variables \( y_i \) of \( \psi_{\text{QBF}} \), the universal variables in \( D_i \) appear in the quantifier prefix of \( \psi_{\text{QBF}} \) on the left of \( y_i \).

**Lemma 3:** If \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \), then \( \models \psi_{\text{QBF}} \Rightarrow \models \psi_{\text{QBF}} \).

**Proof:** If \( y \) is an existential variable of \( \psi_{\text{QBF}} \) and \( \psi_{\text{DQBF}} \preceq \psi_{\text{QBF}} \) and \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \), then each Skolem function for \( y \) in \( \psi_{\text{QBF}} \) is also a Skolem function for \( y \) in \( \psi_{\text{QBF}} \).

If a QBF approximation is unsatisfied, we can therefore conclude that the original DQBF is also unsatisfied, but a satisfied QBF approximation does not give us any information about the satisfaction of the DQBF.

**Definition 4:** Let \( \psi_{\text{QBF}} := \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \exists y_k : \varphi \) and \( \psi_{\text{QBF}} := \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \exists y_k : \varphi \) be two approximations of \( \psi_{\text{QBF}} \). \( \psi_{\text{DQBF}} \) is stronger than \( \psi_{\text{QBF}} \) (written \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \)) if for all existential variables \( y \) holds: \( y \in Y_1 \cap Y_2 \) implies \( \bigcup_{i=1}^{m} X_\ell \subseteq \bigcup_{i=1}^{m} X_\ell \).

Stronger approximations are more favorable in terms of approximation quality:

**Lemma 4:** If \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \), then \( \models \psi_{\text{QBF}} \Rightarrow \models \psi_{\text{QBF}} \).

**Proof:** If \( y \) is an existential variable of \( \psi_{\text{QBF}} \) and \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \) and \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \), then each Skolem function for \( y \) in \( \psi_{\text{QBF}} \) is also a Skolem function for \( y \) in \( \psi_{\text{QBF}} \).

**Definition 5:** Let \( \psi_{\text{DQBF}} \) be a DQBF and \( \psi_{\text{QBF}} \) a QBF approximation of \( \psi_{\text{DQBF}} \). \( \psi_{\text{QBF}} \) is a strongest approximation or strongest formulation if \( \psi_{\text{QBF}} \preceq \psi_{\text{QBF}} \) implies \( \psi_{\text{QBF}} = \psi_{\text{QBF}} \) for all approximations \( \psi_{\text{QBF}} \).

That means the strongest approximations are the ones that are closest to the original DQBF formula.

**Remark 1:** If the PEC contains a single black box, the corresponding (unique) strongest QBF approximation is equivalent to the DQBF formulation (i.e. PEC with single black boxes can be solved exactly by QBF, see also [1]).

**Example 3:** Consider again the DQBF from the previous example:
\[
\psi_{\text{DQBF}} := \forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : (y_1 \lor y_2) \land (x_1 \lor \neg x_2).
\]

In order to obtain a QBF approximation, we have to take the dependencies into account: \( y_1 \) has to be placed right of \( x_1 \) and \( y_2 \) right of \( x_2 \) in the quantifier prefix. Therefore we obtain three different QBF approximations, where \( \varphi \) is the matrix of \( \psi_{\text{QBF}} \):
\[
\begin{align*}
\psi_{\text{QBF}}^1 &= \forall x_1 \exists y_1 \forall x_2 \exists y_2 : \varphi, \\
\psi_{\text{QBF}}^2 &= \forall x_2 \exists y_2 \forall x_1 \exists y_1 : \varphi, \\
\psi_{\text{QBF}}^3 &= \forall x_1 \forall x_2 \exists y_1 \exists y_2 : \varphi.
\end{align*}
\]

where \( \varphi \) is the matrix as in \( \psi_{\text{QBF}} \). Note, in \( \psi_{\text{QBF}}^1 \) the variable \( y_2 \) depends on both universal variables \( x_1 \) and \( x_2 \), whereas \( y_1 \) only
depends on \(x_1\) and vice versa for \(\psi_{\text{QBF}}^2\). In \(\psi_{\text{QBF}}^3\), both \(y_1\) and \(y_2\) depend on both \(x_1\) and \(x_2\).

There are two strongest approximations, namely \(\psi_{\text{QBF}}^1\) and \(\psi_{\text{QBF}}^2\). In both formulations all universal variables appearing left of \(y_1\) (\(y_2\)) also appear left of \(y_1\) (\(y_2\)) in \(\psi_{\text{QBF}}^3\). Therefore \(\psi_{\text{QBF}}^3\) is not a strongest approximation.

In our experiments in Section V we will compare the results obtained for a series of PEC case studies using DQBF and only the strongest QBF approximations.

IV. ELIMINATION-BASED DQBF SOLVING

In this section we describe variable elimination procedures for DQBF and prove their correctness. They yield an algorithm to decide whether a given DQBF is satisfied.

Let in the following

\[
\psi := \forall x_1 \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_m(D_m) : \varphi
\]

be a DQBF with \(D_i \subseteq \{x_1, \ldots, x_n\}\) for \(i = 1, \ldots, m\). For a set \(V\) of Boolean variables let \(V' = \{x' \mid x \in V\}\) of new Boolean variables, indicating that \(x'\) is a copy of \(x\).

To eliminate a universal variable \(x_i\), we construct the conjunction of the two co-factors of \(\varphi\) and replace in one co-factor the variables in \(E_{x_i}\) with their copy. Therefore we have to double all existential variables which depend on \(x_i\), i.e., all variables in \(E_{x_i}\).

**Theorem 2 (Elimination of universal variables):** Let \(E_{x_i} = \{y_j \in V^3_\psi \mid x_i \in D_j\}\) be the set of existential variables which depend on the universal variable \(x_i\). Then \(\psi\) is equivalent to the following DQBF:

\[
\psi' := \forall x_1 \ldots \forall x_{i-1} \forall x_{i+1} \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_m(D_m) : \varphi
\]

\[
\exists y'_1(D_1 \setminus \{x_i\}) \ldots \exists y'_m(D_m \setminus \{x_i\}) \exists y'_j(D_j \setminus \{x_i\}) : \varphi[0/x_i] \land \varphi[1/x_i][y'_j/y_j \forall y_j \in E_{x_i}].
\]

**Proof:** To simplify notation, w.l.o.g. assume \(i = 1\), i.e., we eliminate \(x_1\). Then we have:

\[
\psi \\
\Leftrightarrow \exists y_1, D_1, \ldots, s_{ym}, D_m \text{ with } \\
\Leftrightarrow \forall x_1 \ldots \forall x_n : \varphi[s_{yj}, D_j/y_j \forall y_j \in V^3_\psi] \\
\Leftrightarrow \exists y_1, D_1, \ldots, s_{ym}, D_m \text{ with } \\
\Leftrightarrow \forall x_2 \ldots \forall x_n : \varphi[s_{yj}, D_j/y_j \forall y_j \in V^3_\psi][0/x_1] \\
\Leftrightarrow \exists y_1, D_1, \ldots, s_{ym}, D_m \text{ with } \\
\Leftrightarrow \forall x_2 \ldots \forall x_n : \varphi[0/x_1][s_{yk}, D_k/y_k \forall y_k \in V^3_\psi \setminus E_{x_1}] \\
\[(s_{yk}, D_k|_{x_1=0}/y_k \forall y_k \in E_{x_1}) \\
\land \varphi[1/x_1][s_{yk}, D_k/y_k \forall y_k \in V^3_\psi \setminus E_{x_1}] \\
\[(s_{yk}, D_k|_{x_1=1}/y_k \forall y_k \in E_{x_1}) \\
\Leftrightarrow \exists y_1, D_1, \ldots, s_{ym}, D_m \text{ with } \\
\Leftrightarrow \forall x_2 \ldots \forall x_n : \left(\varphi[0/x_1] \land \varphi[1/x_1][y'_j/y_j \forall y_j \in E_{x_1}]\right) \\
\[(s_{yk}, D_k/y_k \forall y_k \in V^3_\psi \setminus E_{x_1}) \\
\[(s_{yk}, D_k|_{x_1=0}/y_k \forall y_k \in E_{x_1})[s_{yj}, D_j|_{x_1=1}/y_j \forall y_j \in E_{x_1}].
\]

In the following we state elimination rules for two special cases. First, consider the case of eliminating a universal variable \(x_i\) with \(E_{x_i} = \emptyset\), i.e., there is no existential variable depending on \(x_i\). We obtain the following elimination rule:

**Corollary 2:** If \(E_{x_i} = \emptyset\), \(\psi\) is equivalent to

\[
\psi' := \forall x_1 \ldots \forall x_{i-1} \forall x_{i+1} \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_m(D_m) : \varphi[0/x_i] \land \varphi[1/x_i].
\]

In a second case consider an existential variable depending on all universal variables. For this we apply the elimination rule which is stated in the following:

**Lemma 5 (Elimination of existential variables):** Consider the following DQBF:

\[
\psi := \forall x_1 \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_m(D_m) : \varphi
\]

If \(D_i = \{x_1, \ldots, x_n\}\), i.e., if \(y_i\) depends on all universal variables, \(\psi\) is equivalent to:

\[
\forall x_1 \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_{i-1}(D_{i-1}) \exists y_{i+1}(D_{i+1}) \ldots \exists y_m(D_m) : \varphi[0/y_i] \lor \varphi[1/y_i].
\]

This is the standard QBF variable elimination rule for existential variables on the innermost quantifier level [16].

Algorithm 1 shows how to apply quantifier elimination to decide a given DQBF. It takes the set \(V_\psi\) of universally quantified variables, the set \(V_3\) of existentially quantified variables together with their dependency sets, and the matrix \(\varphi\) of the DQBF as inputs. First we compute for each universal variable \(x \in V_\psi\) which existential variables depend on \(x\). As long as the formula contains universal variables we repeat the following elimination process:

We first check if there are existential variables which depend on all universal variables (cf. Line 5). These are eliminated first by using Lemma 5, because they would otherwise be doubled for each universal variable that is eliminated. This in particular applies in a PEC to the Tseitin variables which are introduced to generate a matrix in conjunctive normal form. The function \(\exists\)-eliminate takes care of this elimination. We have to remove the eliminated variables from \(V_3\) and all \(E_x\) for \(x \in V_\psi\).

If no existential variables are left that can be eliminated, we choose a universal variable \(x^*\) for elimination upon which a minimal number of existential variables depend. This heuristic choice is based on the fact that the smaller the number of depending existential variables the less variables have to be doubled. The elimination is carried out by the function \(\forall\)-eliminate. Here, we first expand \(\varphi\) by duplicating the existential variables depending on \(x^*\). Then \(x^*\) is substituted by
Algorithm 1 Solving DQBF using quantifier elimination

SolveDQBF(ψ) := ∀x₁...∀xₙ∃y₁(D_y₁)...∃yₘ(D_yₘ) : φ)
begin
  Vᵥ := \{x₁,...,xₙ\}
  V₃ := \{(y₁,D_y₁),...,(yₘ,D_yₘ)\}
  Eₓ := \{y \, | \, (y,D_y) \in Vᵥ \land x \in D_y\} for all \, x \in Vᵥ
  while Vᵥ ≠ \emptyset do
    // eliminate existential variables
    P := \{y \, | \, (y,D_y) \in Vᵥ \land D_y = Vᵥ\}
    if P ≠ \emptyset then
      φ := 3-eliminate(φ,P)
      Eₓ := Eₓ \ P for x \in Vᵥ
      Vᵥ := Vᵥ \ \{(y,D_y) \, | \, y \in P\}
    end if
    // variable selection and elimination:
    x* := arg minₓ∈Eₓ |Eₓ|
    φ := v-eliminate(φ,x*,Eₓ)
    // update the variable and dependency sets:
    Vᵥ := Vᵥ \ \{x^*\}
    V₃ := \{(y,D \, | \, x^* \in Vᵥ) \cup \{(y',D_y') \, | \, (y,D_y) \in Eₓ^* \land D_y' = D_y \land x^*\}\}
    Eₓ := Eₓ \ \{y' \, | \, y \in Eₓ^* \land Eₓ\} for all \, x \in Vᵥ
  end while
  return SAT(φ)
end

0 and 1 as described in Theorem 2 and finally a logical AND of both sub-formulae is built. Afterward the sets Vᵥ, V₃ and Eₓ for x \in Vᵥ have to be adjusted. The eliminated variable x* has to be removed from Vᵥ and from all dependency sets. Additionally we have to insert all newly created existential variables y' into V₃ and into the Eₓ sets.

This algorithm terminates after finitely many iterations of the while-loop since in each iteration the number of universally quantified variables decreases by one.

If we have obtained a formula without universal variables, we can use a propositional SAT-solver to decide if the formula is satisfied (cf. Line 17).

Example 4: Consider again our running example from the previous sections. We want to show whether

∀x₁∀x₂∃y₁(x₁)∃y₂(x₂) : (((y₁ \lor y₂) \lor (x₁ \land \neg x₂)) \equiv (x₁ \land x₂))

is satisfied. Applying Algorithm 1 we first eliminate one of the universal variables according to Theorem 2, say x₁, since there is no existential variable which depends on every universal one. This yields

∀x₂∃y₁(0)∃y₁'(0)∃y₂(x₂) : (((y₁ \lor y₂) \equiv x₂) \land (∀y₁ \land ¬y₂'))

Now y₂ depends on all remaining universal variables and gets eliminated (cf. Lemma 5):

∀x₂∃y₁(0)∃y₁'(0) : (((y₁ \equiv x₂) \land (∀y₁ \land ¬y₂'))

Now the algorithm eliminates x₂. Note that we do not have to double any existential variable because none of them depends on x₂. This finally yields the formula (cf. Corollary 2):

ψ₁^{QBF} := ∀x₁∃y₁∀x₂∃y₂ : (((y₁ \lor y₂) \lor (x₁ \land \neg x₂)) \equiv (x₁ \land x₂))

which is obviously not satisfied and hence, the PEC is not realizable.

Now we consider the two strongest QBF formulations from Example 3, first

ψ₁^{QBF} := ∀x₁∃y₁∀x₂∃y₂ : (((y₁ \lor y₂) \lor (x₁ \land \neg x₂)) \equiv (x₁ \land x₂))

We can see that ψ₁^{QBF} is satisfied by giving appropriate Skolem functions for the existential variables. Note that the Skolem functions depend on all universal variables on the left of the existential variable. We use s_{y₁,x₁}(x₂) = 0 for y₁ and s_{y₂,x₁,x₂}(x₁,x₂) = x₁ \lor x₂ for y₂. Replacing y₁ and y₂ (or, equivalently, the left and the right black box) by their Skolem functions, we get

∀x₁∀x₂ : (((0 \lor (x₁ \land x₂)) \lor (x₁ \land \neg x₂)) \equiv (x₁ \land x₂))

which is satisfied.

For the second QBF formulation

ψ₂^{QBF} := ∀x₁∃y₁∀x₂∃y₂ : (((y₁ \lor y₂) \lor (x₁ \land \neg x₂)) \equiv (x₁ \land x₂))

we can use the Skolem functions s_{y₂,x₂}(x₂) = 0 for y₂ and and s_{y₁,x₁,x₂}(x₁,x₂) = x₁ \lor x₂ for y₁. Replacing y₁ and y₂ by their Skolem functions yields

∀x₁∀x₂ : (((x₁ \land x₂) \lor (x₁ \land \neg x₂)) \equiv (x₁ \land x₂)),

which is the same formula as before and therefore also satisfied.

We observe that both strongest QBF formulations give the wrong answer, due to their approximate character, and only the DQBF formulation is correct.

Note that for the third (non strongest) QBF approximation ψ₃^{QBF} from Example 3 we could use either the Skolem functions for ψ₁^{QBF} or for ψ₂^{QBF} leading to a tautological matrix.
V. EXPERIMENTS

In this section we first describe some implementation details of our DQBF solver, followed by a short description of the experimental setup. Finally we present experimental results and their evaluation.

We have implemented Algorithm 1 in C++ in a prototypical solver called henaig. As the basic data structure we use functionally reduced And-Inverter graphs (FRAIGs) [13], [19], [20]. They are essentially circuits consisting of AND gates and inverters only. A FRAIG is a ‘semi-canonical’ form of AIGs, i.e., there are no two distinct gates in the FRAIG representing the same (or inverse) function. Nevertheless FRAIGs still allow multiple structurally different representations of the same function. FRAIGs support all necessary operations like conjunction, disjunction, and replacing an input by an arbitrary FRAIG (in particular the constant 0- and 1-FRAIG).

Currently our solver can handle PECs with a few hundred gates, but in this paper we focused on showing the qualitative differences between QBF and DQBF formulations. So far our available instances tend to fall into two classes: instances which are rather fast to solve and secondly instances which cannot be solved due to memory or timing constraints. This is a similar situation as in the early days of related solving engines for SAT or QBF and we expect to see scalability in the near future.

We have generated all $2^{16} = 65,536$ possible Boolean functions with four inputs and used an implementation of them as the four-input circuit in Fig. 4. We checked if there are realizations of the black boxes such that the implementation becomes equivalent to the XOR of the primary inputs. For this we used the DQBF formulation as well as both strongest approximate QBF formulations as seen in Example 3. We abstain from making a comparison with SAT-based approaches, since they are even less accurate than the QBF-based approach.

We extended this example by considering 3 black boxes and primary inputs as well as 4 black boxes and primary inputs. The demanded specification is again an XOR of all primary inputs and we compare the DQBF results with all strongest QBF approximations. Due to the mere number of possible functions we did not consider all of them and ran only 50,000 randomly picked instances for both 3 and 4 black boxes. Each of these instances could be solved in significantly less than one second.

All results are given in Table I. The first column states the number of black boxes (“BB”) followed by the number of total instances and their classification (“SAT” or “UNSAT”, i.e., realizable or unrealizable) obtained from our DQBF solver. The fourth column shows the total number of instances classified as UNSAT and the last three columns show the results of the QBF formulation compared with the unsatisfied DQBF one. Here the number of instances are given where all strongest QBF formulations return the same result as the DQBF version (“correct”), where all strongest QBF formulations return a different result (“wrong”), and finally where some QBF formulations report the same result and some a different one (“depends”). The given percentages are related to the total number of unsatisfied DQBF instances, since the satisfied DQBF instances are also (correctly) stated as satisfied in any QBF approximation.

If the DQBF formulation reports realizability, the QBF always detects realizability, too. For 2 black boxes, this happened in 32378 cases. The remaining 33168 cases, where DQBF detects unrealizability, can be partitioned in the following three cases:

In 16798 cases (50.6 %) the result of all strongest QBF formulations are correct, i.e., also unsatisfiable. In all other cases at least one of the QBF formulations returns an incorrect result: In 4584 cases (13.8 %), both QBF formulations reported the contrary result, and in 11776 additional cases (35.5 %), one of the QBF formulations correctly reported unrealizability, but the other QBF formulation was satisfied.

This means, in 13.8 % of these cases, DQBF is the only way to obtain a correct result, and in additional 35.5 % of these cases, one only obtains a correct result by chance.

The number of possible implementations decreases significantly with the number of black boxes—from about 50% with 2 black boxes to 199 out of 50,000 cases with 4 black boxes—and at the same time the number of incorrect QBF results increases. For the unsatisfied instances with 4 black boxes in only 16 out of 49 801 cases all strongest QBF formulations return the correct result and 43.4 % return that the PEC is realizable although it is not.

We applied a similar scenario using PEC problems described in [1]. These problems consist of a carry ripple adder circuit as specification and a copy of this specification as implementation. In addition at least one and up to six gates are removed from the implementation and replaced by a distinct black box for each
TABLE II

RESULTS FOR CARRY RIPPLE ADDER

<table>
<thead>
<tr>
<th>#BBs</th>
<th>#DQBF instances total</th>
<th>SAT</th>
<th>UNSAT</th>
<th>#QBF instances total</th>
<th>SAT</th>
<th>UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>96</td>
<td>24</td>
<td>72</td>
<td>96</td>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>56</td>
<td>21</td>
<td>35</td>
<td>112</td>
<td>95</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>48</td>
<td>14</td>
<td>34</td>
<td>288</td>
<td>153</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>10</td>
<td>18</td>
<td>672</td>
<td>294</td>
<td>378</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>960</td>
<td>275</td>
<td>685</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2880</td>
<td>751</td>
<td>2119</td>
</tr>
<tr>
<td>total</td>
<td>240</td>
<td>74</td>
<td>168</td>
<td>5008</td>
<td>1664</td>
<td>3324</td>
</tr>
</tbody>
</table>

gate. There are 20 different versions of this circuit, which are obviously realizable. For each version there exist 11 additional variations, where random faults have been added to the non black boxed parts of the implementation, resulting in 240 benchmarks in total.

For all of these 240 instances we have generated the corresponding DQBF as well as all 5008 strongest QBF approximations. The results are shown in Table II. Each instance could be solved within three seconds. Again the first column shows the number of black boxes in the design ("#BBs"). The next three columns show the total number of different PEC versions as well as their classification ("SAT" and "UNSAT") by the DQBF formulation. The last three columns show the number of different QBF versions ("total") and their result compared to the DQBF formulation ("correct" and "wrong"). Note, that for n black boxes there are n! different strongest QBF approximations.

For one black box all strongest QBF versions return the correct result, since it is accurate for one black box. But one can clearly observe that the number of incorrect QBF answers increases again significantly with the number of black boxes—up to 73.6% for 6 black boxes. We observed that for these particular benchmarks there is no instance for which all strongest QBF approximations provide the wrong answer, but there is a significant amount of strongest prefix variations returning the wrong result for most of the instances. Consider in particular the 3 unrealizable DQBF instances with 6 black boxes. We considered 6! = 720 different strongest QBF approximations for each. In only 31 out of in total 2 160 instances (1.4%) the correct result is returned that the PEC is unrealizable.

VI. CONCLUSION AND FUTURE WORK

We have shown how to decide exactly whether a partial combinational circuit can be extended such that it becomes equivalent to a complete specification. Our approach is based on a transformation from PEC to DQBF. We have presented an algorithm to solve DQBF based on variable elimination. Preliminary experimental results show the feasibility and necessity of this approach.

Future work will encompass making the solver more efficient by transferring more of the techniques commonly used in state-of-the-art QBF solvers to the domain of Henkin quantifiers. Preprocessing of DQBF to simplify the formula is also a current research topic. We expect from both considerably improved scalability to large-scale circuits. Additionally we plan to use DQBF in bounded model checking of sequential circuits.

Acknowledgments

We thank Florian Pigorsch, University of Freiburg, for providing us with his AIG package, QBF solver (AIGsolve) and support during the solver development.

REFERENCES