Preprint from IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, October 6-9, 2013

Equivalence Checking of Partial Designs Using Dependency Quantified Boolean Formulae*

Karina Gitina, Sven Reimer, Matthias Sauer, Ralf Wimmer, Christoph Scholl, Bernd Becker
Albert-Ludwigs-Universität Freiburg, Germany
{gitina|reimer|sauerm|wimmer|scholl|becker}@informatik.uni-freiburg.de

***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

*This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS) and the Research Training Group "Embedded Microsystems" (GRK 1103).

978-1-4799-2987-0/13/$31.00 ©2013 IEEE

incorrect results in a significant number of cases.

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 sound- 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 $\nu : V \to \{0, 1\}$. We denote the set of variable assignments for V by $\mathcal{A}_V$.

If φ is a Boolean expression containing the variable $v \in V$, and ψ an expression not containing v, we denote by $\varphi[\psi/v]$ the expression that results from φ by replacing each occurrence of v with ψ. Replacing each $v \in W \subseteq V$ by an expression ψ_v is denoted by $\varphi[\psi_v/v\ \forall v \in W]$. In this case we require that the expressions ψ_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 φ 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)* ψ 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 φ is called the matrix of ψ.

We denote $V_\psi^\exists = \{y_1, \ldots, y_m\}$ as the set of existential variables and $V_\psi^\forall = \{x_1, \ldots, x_n\}$ the set of universal variables. If $y_i \in V_\psi^\exists$ is an existential variable with dependency set D_i, a *Skolem function* for y_i is a function $s_{y_i,D_i} : \mathcal{A}_{D_i} \to \{0, 1\}$. In this case, $\varphi[s_{y_i,D_i}/y_i]$ denotes the expression resulting from φ 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 $s_{y_i,D_i\setminus\{x\}} : \mathcal{A}_{D_i\setminus\{x\}} \to \{0, 1\}$ 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 $\vDash \psi$) if and only if there are Skolem functions s_{y_i,D_i} for $i = 1, \ldots, m$ such that $\varphi[s_{y_i,D_i}/y_i\ \forall y_i \in V_\psi^\exists]$ 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 $Y_i \subseteq \{y_1, \ldots, y_n\}$ 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.

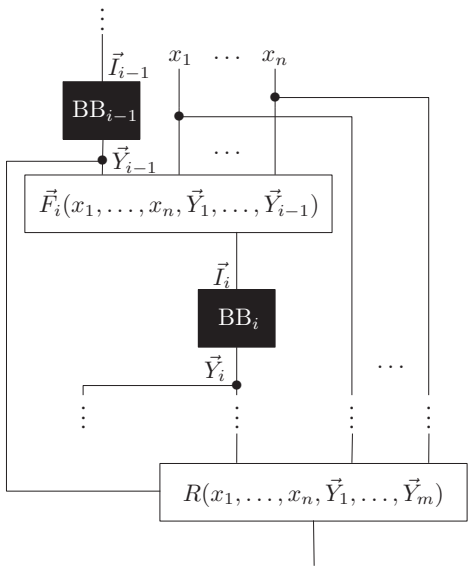

Fig. 1. Notation for partial designs

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[1].
- $\vec{I}_i$ are the inputs of BB_i $(i = 1, \ldots, m)$.
- $\vec{Y}_i$ are the outputs of BB_i $(i = 1, \ldots, m)$.
- $\vec{F}_i(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_{i-1})$ is the vector of functions defining $\vec{I}_i$.
- $R(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_m)$ 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 XNOR) gate for each corresponding output of the specification and implementation and require the outputs of these equivalence gates to 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.

[1]To 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 $i < j$

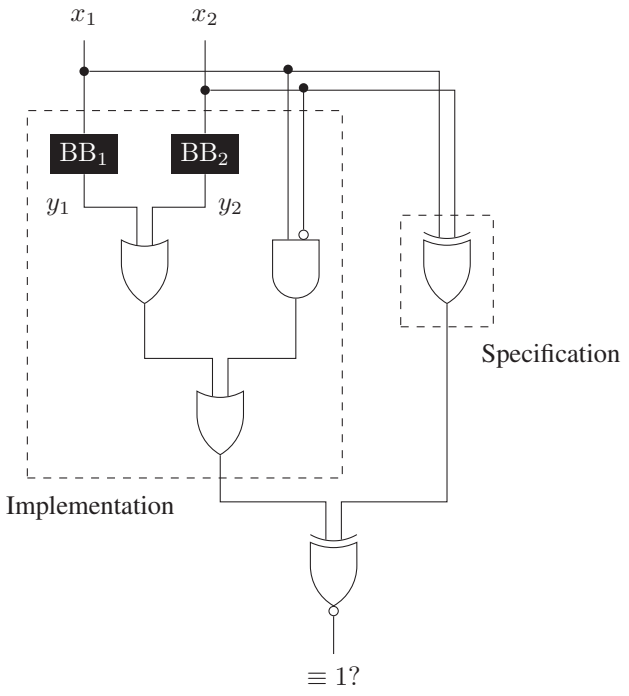

Fig. 2. Example for PEC

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 $\vec{F}_i(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_{i-1})$.

$$\begin{aligned}\varphi := \big(\vec{I}_1 \not\equiv \vec{F}_1(x_1, \ldots, x_n)\big) \vee \cdots \\ \vee \big(\vec{I}_m \not\equiv \vec{F}_m(x_1, \ldots, x_n, \vec{Y}_1, \ldots, \vec{Y}_{m-1})\big) \\ \vee R(x_1, \ldots x_n, \vec{Y}_1, \ldots, \vec{Y}_m). \qquad (1)\end{aligned}$$

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 φ' that is satisfiability equivalent to φ and whose size is linear in the size of φ. The variables in $\vec{A}$ are existentially quantified in the quantifier prefix. Their dependency set encompasses all universally quantified variables.

The resulting DQBF is:

$$\begin{aligned}\psi := \forall x_1 \ldots \forall x_n \forall \vec{I}_1 \ldots \forall \vec{I}_m \exists \vec{Y}_1(\vec{I}_1) \ldots \exists \vec{Y}_m(\vec{I}_m) \\ \exists \vec{A}(x_1, \ldots, x_n, \vec{I}_1, \ldots, \vec{I}_m) : \varphi'.\end{aligned}$$

The formula ψ is satisfied if and only if we can replace all $\vec{Y}_i$ with Skolem functions $\vec{s}_{\vec{Y}_i, \vec{I}_i}$ such that φ' becomes a tautology. The Skolem functions $\vec{s}_{\vec{Y}_i, \vec{I}_i}$ 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 ψ whose length is linear in the size of the circuit. This is captured in the following lemma:

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:

$$\begin{aligned}\psi_{\text{DQBF}} = \forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : \\ \big((y_1 \vee y_2) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2).\end{aligned}$$

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 inputs 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 $\big((y_1 \vee y_2) \vee (x_1 \wedge \neg x_2)\big)$. 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 $(x_1 \oplus x_2)$.

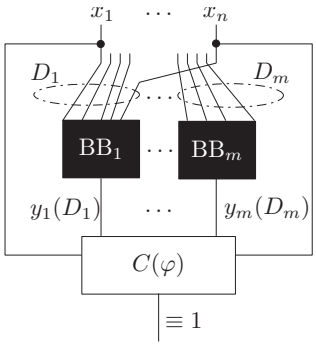

Fig. 3. Translation from DQBF to PEC

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

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

The matrix φ 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 $\vee$, $\wedge$, and $\neg$ 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 ψ whose size is linear in the size of P such that P is realizable if and only if ψ is satisfied (cf. Lemma 1). Conversely, for each DQBF ψ there is a PEC P whose size is linear in the size of ψ such that ψ 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

$$\psi_{\text{DQBF}} := \forall x_1 \ldots \forall x_n \exists y_1(D_1) \ldots \exists y_m(D_m) : \varphi$$

be a DQBF and

$$\psi_{\text{QBF}} := \forall X_1 \exists Y_1 \forall X_2 \exists Y_2 \ldots \exists Y_k : \varphi$$

a QBF with the same matrix, such that $\{X_i \subseteq \{x_1, \ldots, x_n\} \,|\, i = 1, \ldots, k\}$ is a partition of the universal and $\{Y_i \subseteq \{y_1, \ldots, y_m\} \,|\, i = 1, \ldots, k\}$ a partition of the existential variables. ψ_{QBF} is an *approximation* of ψ_{DQBF} (written $\psi_{\text{DQBF}} \preceq \psi_{\text{QBF}}$) if $y_i \in Y_j$ implies $D_i \subseteq \bigcup_{\ell=1}^{j} X_\ell$ for all $i = 1, \ldots, m$.

That means, ψ_{QBF} is an approximation of ψ_{DQBF} if for all existential variables y_i of ψ_{DQBF}, the universal variables in D_i appear in the quantifier prefix of ψ_{QBF} on the left of y_i.

Lemma 3: If $\psi_{\text{DQBF}} \preceq \psi_{\text{QBF}}$, then $\vDash \psi_{\text{DQBF}}$ implies $\vDash \psi_{\text{QBF}}$.

Proof: If y is an existential variable of ψ_{DQBF} and ψ_{QBF}, and $\psi_{\text{DQBF}} \preceq \psi_{\text{QBF}}$, then each Skolem function for y in ψ_{DQBF} is also a Skolem function for y in ψ_{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 ψ_{DQBF}. ψ_{QBF} is *stronger* than ψ'_{QBF} (written $\psi_{\text{QBF}} \preceq \psi'_{\text{QBF}}$) if for all existential variables y holds: $y \in Y_i \cap Y'_j$ implies $\bigcup_{k=1}^{i} X_k \subseteq \bigcup_{k=1}^{j} X'_k$.

Stronger approximations are more favorable in terms of approximation quality:

Lemma 4: If $\psi_{\text{QBF}} \preceq \psi'_{\text{QBF}}$, then $\vDash \psi_{\text{QBF}}$ implies $\vDash \psi'_{\text{QBF}}$.

Proof: If y is an existential variable of ψ_{QBF} and ψ'_{QBF}, and $\psi_{\text{QBF}} \preceq \psi'_{\text{QBF}}$, then each Skolem function for y in ψ_{QBF} is also a Skolem function for y in ψ'_{QBF}. ∎

Definition 5: Let ψ_{DQBF} be a DQBF and ψ_{QBF} a QBF approximation of ψ_{DQBF}. ψ_{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 ψ'_{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:

$$\psi_{\text{DQBF}} = \forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : \big((y_1 \vee y_2) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2).$$

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 φ is the matrix of ψ_{DQBF}:

$$\psi^1_{\text{QBF}} = \forall x_1 \exists y_1 \forall x_2 \exists y_2 : \varphi,$$
$$\psi^2_{\text{QBF}} = \forall x_2 \exists y_2 \forall x_1 \exists y_1 : \varphi,$$
$$\psi^3_{\text{QBF}} = \forall x_1 \forall x_2 \exists y_1 \exists y_2 : \varphi.$$

where φ is the matrix as in ψ_{DQBF}. Note, in ψ^1_{QBF} 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 ψ^2_{QBF}. In ψ^3_{QBF}, both y_1 and y_2 depend on both x_1 and x_2.

There are two strongest approximations, namely ψ^1_{QBF} and ψ^2_{QBF}. In both formulations all universal variables appearing left of y_1 (y_2) also appear left of y_1 (y_2) in ψ^3_{QBF}. Therefore ψ^3_{QBF} 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

$$\psi := \forall x_1 \dots \forall x_n \exists y_1(D_1) \dots \exists y_m(D_m) : \varphi$$

be a DQBF with $D_i \subseteq \{x_1, \dots, x_n\}$ for $i = 1, \dots, m$. For a set V of Boolean variables let V' denote the set $V' = \{x' \mid x \in V\}$ 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 φ 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 $E_{x_i} = \left\{y_j \in V^\exists_\psi \mid x_i \in D_j\right\}$ be the set of existential variables which depend on the universal variable x_i. Then ψ is equivalent to the following DQBF:

$$\begin{aligned}\psi' := \forall x_1 \dots \forall x_{i-1} \forall x_{i+1} \dots \forall x_n \\ \exists y_1(D_1 \setminus \{x_i\}) \dots \exists y_m(D_m \setminus \{x_i\}) \underbrace{\exists y'_j(D_j \setminus \{x_i\})}_{\text{for all } y_j \in E_{x_i}} : \\ \varphi[0/x_i] \wedge \varphi[1/x_i][y'_j/y_j \ \forall y_j \in E_{x_i}].\end{aligned}$$

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

$$\begin{aligned}
&\vDash \psi \\
&\Leftrightarrow \exists s_{y_1,D_1}, \dots, s_{y_m,D_m} \text{ with} \\
&\qquad \vDash \forall x_1 \dots \forall x_n : \varphi[s_{y_j,D_j}/y_j \ \forall y_j \in V^\exists_\psi] \\
&\Leftrightarrow \exists s_{y_1,D_1}, \dots, s_{y_m,D_m} \text{ with} \\
&\qquad \vDash \forall x_2 \dots \forall x_n : \varphi[s_{y_j,D_j}/y_j \ \forall y_j \in V^\exists_\psi][0/x_1] \\
&\qquad\qquad \wedge \varphi[s_{y_j,D_j}/y_j \ \forall y_j \in V^\exists_\psi][1/x_1] \\
&\Leftrightarrow \exists s_{y_1,D_1}, \dots, s_{y_m,D_m} \text{ with} \vDash \forall x_2 \dots \forall x_n : \\
&\qquad \varphi[0/x_1][s_{y_k,D_k}/y_k \ \forall y_k \in V^\exists_\psi \setminus E_{x_1}] \\
&\qquad\qquad [s_{y_j,D_j|x_1=0}/y_j \ \forall y_j \in E_{x_1}] \\
&\qquad \wedge \varphi[1/x_1][s_{y_k,D_k}/y_k \ \forall y_k \in V^\exists_\psi \setminus E_{x_1}] \\
&\qquad\qquad [s_{y_j,D_j|x_1=1}/y_j \ \forall y_j \in E_{x_1}] \\
&\Leftrightarrow \exists s_{y_1,D_1}, \dots, s_{y_m,D_m} \text{ with} \\
&\qquad \vDash \forall x_2 \dots \forall x_n : \Big(\varphi[0/x_1] \wedge \varphi[1/x_1][y'_j/y_j \ \forall y_j \in E_{x_1}]\Big) \\
&\qquad [s_{y_k,D_k}/y_k \ \forall y_k \in V^\exists_\psi \setminus E_{x_1}] \\
&\qquad [s_{y_j,D_j|x_1=0}/y_j \ \forall y_j \in E_{x_1}][s_{y_j,D_j|x_1=1}/y'_j \ \forall y_j \in E_{x_1}] \\
&\Leftrightarrow \vDash \forall x_2 \dots \forall x_n \ \underbrace{\exists y_k(D_k)}_{\text{for all } y_k \notin E_{x_1}} \ \underbrace{\exists y_j(D_j \setminus \{x_1\}) \exists y'_j(D_j \setminus \{x_1\})}_{\text{for all } y_j \in E_{x_1}} : \\
&\qquad \varphi[0/x_1] \wedge \varphi[1/x_1][y'_j/y_j \ \forall y_j \in E_{x_1}] \\
&\Leftrightarrow \vDash \forall x_2 \dots \forall x_n \\
&\qquad \exists y_1(D_1 \setminus \{x_1\}) \dots \exists y_m(D_m \setminus \{x_1\}) \underbrace{\exists y'_j(D_j \setminus \{x_1\})}_{\text{for all } y_j \in E_{x_1}} : \\
&\qquad \varphi[0/x_1] \wedge \varphi[1/x_1][y'_j/y_j \ \forall y_j \in E_{x_1}].
\end{aligned}$$

■

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 $E_{x_i} = \emptyset$, ψ is equivalent to

$$\begin{aligned}\psi' := \forall x_1 \dots \forall x_{i-1} \forall x_{i+1} \dots \forall x_n \exists y_1(D_1) \dots \exists y_m(D_m) : \\ \varphi[0/x_i] \wedge \varphi[1/x_i].\end{aligned}$$

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:

$$\psi := \forall x_1 \dots \forall x_n \exists y_1(D_1) \dots \exists y_m(D_m) : \varphi$$

If $D_i = \{x_1, \dots, x_n\}$, i. e., if y_i depends on all universal variables, ψ is equivalent to:

$$\begin{aligned}\forall x_1 \dots \forall x_n \exists y_1(D_1) \dots \exists y_{i-1}(D_{i-1}) \\ \exists y_{i+1}(D_{i+1}) \dots \exists y_m(D_m) : \varphi[0/y_i] \vee \varphi[1/y_i].\end{aligned}$$

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 $V_\forall$ of universally quantified variables, the set $V_\exists$ of existentially quantified variables together with their dependency sets, and the matrix φ of the DQBF as inputs. First we compute for each universal variable $x \in V_\forall$ 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 $V_\exists$ and all E_x for $x \in V_\forall$.

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 φ by duplicating the existential variables depending on x^*. Then x^* is substituted by

Algorithm 1 Solving DQBF using quantifier elimination

SolveDQBF($\psi := \forall x_1 \dots \forall x_n \exists y_1(D_{y_1}) \dots \exists y_m(D_{y_m}) : \varphi$)
begin
$V_\forall \leftarrow \{x_1, \dots, x_n\}$ (1)
$V_\exists \leftarrow \{(y_1, D_{y_1}), \dots, (y_m, D_{y_m})\}$ (2)
$E_x \leftarrow \{y \,|\, (y, D_y) \in V_\exists \wedge x \in D_y\}$ for all $x \in V_\forall$ (3)
while $V_\forall \neq \emptyset$ **do** (4)
// eliminate existential variables
$P \leftarrow \{y \,|\, (y, D_y) \in V_\exists \wedge D_y = V_\forall\}$ (5)
if $P \neq \emptyset$ **then** (6)
$\varphi \leftarrow \exists$-eliminate$(\varphi, P)$ (7)
$E_x \leftarrow E_x \setminus P$ for $x \in V_\forall$ (8)
$V_\exists \leftarrow V_\exists \setminus \{(y, D_y) \,|\, y \in P\}$ (9)
end if (10)
// variable selection and elimination:
$x^* \leftarrow \arg\min_{x \in V_\forall} |E_x|$ (11)
$\varphi \leftarrow \forall$-eliminate$(\varphi, x^*, E_{x^*})$ (12)
// update of the variable and dependency sets:
$V_\forall \leftarrow V_\forall \setminus \{x^*\}$ (13)
$V_\exists \leftarrow \{(y, D \setminus \{x^*\} \,|\, (y, D) \in V_\exists\}$
$\dot\cup \{(y', D_{y'}) \,|\, (y, D_y) \in E_{x^*} \wedge D_{y'} = D_y \setminus \{x^*\}\}$ (14)
$E_x \leftarrow E_x \,\dot\cup\, \{y' \,|\, y \in E_{x^*} \cap E_x\}$ for all $x \in V_\forall$ (15)
end while (16)
return SAT(φ) (17)
end

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

$$\forall x_1 \forall x_2 \exists y_1(x_1) \exists y_2(x_2) : \big((y_1 \vee y_2) \wedge (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2)$$

is satisfied. Applying Algorithm 1 we first eliminate one of the universal variables according to Theorem 2, say x_1, since there is no existential variable which depends on every universal one. This yields

$$\forall x_2 \exists y_1(\emptyset) \exists y_1'(\emptyset) \exists y_2(x_2) : \underbrace{\big((y_1 \vee y_2) \equiv x_2\big)}_{\text{setting } x_1 = 0} \wedge \underbrace{\big((y_1' \vee y_2 \vee \neg x_2) \equiv \neg x_2\big)}_{\text{setting } x_1 = 1 \text{ and replacing } y_1 \to y_1'}$$

Now y_2 depends on all remaining universal variables and gets eliminated (cf. Lemma 5):

$$\forall x_2 \exists y_1(\emptyset) \exists y_1'(\emptyset) : \underbrace{(y_1 \equiv x_2) \wedge \big((y_1' \vee \neg x_2) \equiv \neg x_2\big)}_{\text{setting } y_2 = 0} \vee \underbrace{\big((1 \equiv x_2) \wedge (1 \equiv \neg x_2)\big)}_{\text{setting } y_2 = 1}.$$

$$\Leftrightarrow \forall x_2 \exists y_1(\emptyset) \exists y_1'(\emptyset) : (y_1 \equiv x_2) \wedge \big((y_1' \vee \neg x_2) \equiv \neg x_2\big).$$

Now the algorithm eliminates x_2. Note that we do not have to double any existential variable because none of them depends on x_2. This finally yields the formula (cf. Corollary 2):

$$\exists y_1(\emptyset) \exists y_1'(\emptyset) : \underbrace{\big((y_1 \equiv 0) \wedge (1 \equiv 1)\big)}_{\text{setting } x_2 = 0} \wedge \underbrace{\big((y_1 \equiv 1) \wedge (y_1' \equiv 0)\big)}_{\text{setting } x_2 = 1},$$

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

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

$$\psi^1_{\text{QBF}} = \forall x_1 \exists y_1 \forall x_2 \exists y_2 : \big((y_1 \vee y_2) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2).$$

We can see that ψ^1_{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 $s_{y_1,\{x_1\}}(x_1) = 0$ for y_1 and $s_{y_2,\{x_1,x_2\}}(x_1, x_2) = x_1 \oplus x_2$ for y_2. Replacing y_1 and y_2 (or, equivalently, the left and the right black box) by their Skolem functions, we get

$$\begin{aligned} &\forall x_1 \forall x_2 : \big((0 \vee (x_1 \oplus x_2)) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2) \\ \Leftrightarrow\ &\forall x_1 \forall x_2 : \big((x_1 \oplus x_2) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2) \\ \Leftrightarrow\ &\forall x_1 \forall x_2 : \big(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)\big) \\ &\qquad\qquad\qquad\qquad \equiv (x_1 \oplus x_2) \\ \Leftrightarrow\ &\forall x_1 \forall x_2 : (x_1 \oplus x_2) \equiv (x_1 \oplus x_2), \end{aligned}$$

which is satisfied.

For the second QBF formulation

$$\psi^2_{\text{QBF}} = \forall x_2 \exists y_2 \forall x_1 \exists y_1 : \big((y_1 \vee y_2) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2)$$

we can use the Skolem functions $s_{y_2,\{x_2\}}(x_2) = 0$ for y_2 and and $s_{y_1,\{x_1,x_2\}}(x_1, x_2) = x_1 \oplus x_2$ for y_1. Replacing y_1 and y_2 by their Skolem functions yields

$$\forall x_2 \forall x_1 : \big(((x_1 \oplus x_2) \vee 0) \vee (x_1 \wedge \neg x_2)\big) \equiv (x_1 \oplus x_2),$$

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 ψ^3_{QBF} from Example 3 we could use either the Skolem functions for ψ^1_{QBF} or for ψ^2_{QBF} leading to a tautological matrix.

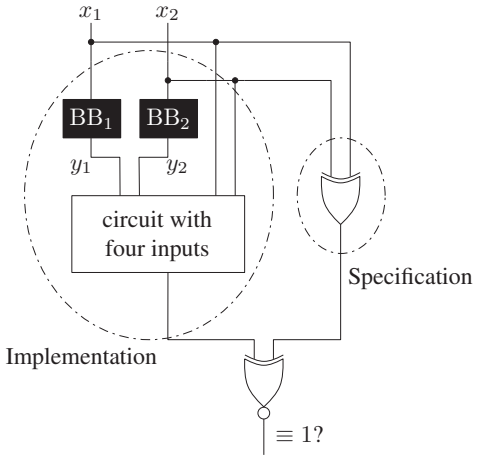

Fig. 4. Circuit template for two black boxes

TABLE I
RESULTS FOR XOR TEMPLATES

#BBs	total	SAT	UNSAT			
			total	correct	wrong	depends
2	65536	32378	33168	16798	4584	11776
			100 %	50.6 %	13.8 %	35.5 %
3	50000	9124	40876	2344	15979	22553
			100 %	5.7 %	39.1 %	55.2 %
4	50000	199	49801	16	21626	28159
			100 %	<0.1 %	43.4 %	56.5 %

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 16 798 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 11 776 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 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

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

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 instances ("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

[1] C. Scholl and B. Becker, "Checking equivalence for partial implementations," in *Design Automation Conference (DAC)*. ACM Press, 2001, pp. 238–243.

[2] T. Nopper and C. Scholl, "Approximate symbolic model checking for incomplete designs," in *Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, ser. LNCS, vol. 3312. Springer-Verlag, 2004, pp. 290–305.

[3] C. Scholl and B. Becker, "Checking equivalence for circuits containing incompletely specified boxes," in *Int'l Conf. on Computer Design (ICCD)*. IEEE, 2002, pp. 56–63.

[4] M. Herbstritt, B. Becker, and C. Scholl, "Advanced SAT-techniques for bounded model checking of blackbox designs," in *Int'l Workshop on Microprocessor Test and Verification (MTV)*. IEEE, 2006, pp. 37–44.

[5] T. Nopper, C. Scholl, and B. Becker, "Computation of minimal counterexamples by using black box techniques and symbolic methods," in *Int'l Conf. on Computer-Aided Design (ICCAD)*. IEEE, 2007, pp. 273–280.

[6] L. Henkin, "Some remarks on infinitely long formulas," in *Infinitistic Methods: Proceedings of the 1959 Symposium on Foundations of Mathematics*. Pergamon Press, Sep. 1961, pp. 167–183.

[7] K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker, "Equivalence checking for partial designs revisited," in *Workshop "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen"*. Universität Rostock, ITMZ, Germany, Mar. 2013, pp. 61–70.

[8] A. Fröhlich, G. Kovásznai, and A. Biere, "A DPLL algorithm for solving DQBF," in *Int'l Workshop on Pragmatics of SAT (POS)*, 2012.

[9] E. Giunchiglia, M. Narizzano, and A. Tacchella, "Backjumping for quantified Boolean logic satisfiability," *Artificial Intelligence*, vol. 145, no. 1-2, pp. 99–120, 2003.

[10] M. Davis, G. Logemann, and D. W. Loveland, "A machine program for theorem-proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[11] M. Benedetti, "Evaluating QBFs via symbolic Skolemization," in *Int'l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, ser. LNCS, vol. 3452. Springer-Verlag, Mar. 2005, pp. 285–300.

[12] V. Balabanov, H.-J. K. Chiang, and J.-H. R. Jiang, "Henkin quantifiers and boolean formulae," in *SAT*, ser. Lecture Notes in Computer Science, A. Cimatti and R. Sebastiani, Eds., vol. 7317. Springer, 2012, pp. 129–142.

[13] A. Kuehlmann, M. K. Ganai, and V. Paruthi, "Circuit-based Boolean reasoning," in *Design Automation Conference (DAC)*. ACM Press, 2001, pp. 232–237.

[14] F. Pigorsch and C. Scholl, "An AIG-based QBF-solver using SAT for preprocessing," in *Design Automation Conference (DAC)*. ACM Press, 2010, pp. 170–175.

[15] R. K. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *CAV*, ser. Lecture Notes in Computer Science, T. Touili, B. Cook, and P. Jackson, Eds., vol. 6174. Springer, 2010, pp. 24–40.

[16] A. Biere, "Resolve and expand," in *Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT)*, ser. LNCS, vol. 3542. Springer-Verlag, May 2004, pp. 59–70.

[17] G. S. Tseitin, "On the complexity of derivation in propositional calculus," *Studies in Constructive Mathematics and Mathematical Logic*, vol. Part 2, pp. 115–125, 1970.

[18] S. Azhar, G. Peterson, and J. Reif, "Lower bounds for multiplayer non-cooperative games of incomplete information," *Journal of Computers and Mathematics with Applications*, vol. 41, pp. 957–992, 2001.

[19] A. Mishchenko, S. Chatterjee, R. Jiang, and R. K. Brayton, "FRAIGs: A unifying representation for logic synthesis and verification," EECS Dept., UC Berkeley, Tech. Rep., 03 2005.

[20] F. Pigorsch, C. Scholl, and S. Disch, "Advanced unbounded model checking based on AIGs, BDD sweeping, and quantifier scheduling," in *Int'l Conf. on Formal Methods in Computer-Aided Design (FMCAD)*. IEEE Computer Society Press, Nov. 2006, pp. 89–96.