Equivalence Checking of Circuits with Parameterized Specifications

Eugene Goldberg

Cadence Berkeley Labs, USA, egold@cadence.com

Abstract. We consider the problem of equivalence checking of circuits N_1,N_2 with a common specification (CS). We show that circuits N_1 and N_2 have a CS iff they can be partitioned into toggle equivalent subcircuits that are connected "in the same way". Based on this result, we formulate a procedure for checking equivalence of circuits N_1 and N_2 with specifications S_1 and S_2. This procedure not only checks equivalence of N_1 and N_2 but also verifies that S_1 and S_2 are identical. The complexity of this procedure is linear in specification size and exponential in the value of a specification parameter. Previously we considered specifications parameterized by the size of the largest subcircuit (specification granularity). In this paper we give a more general parameterization based on specification "width".

1 Introduction

Studying equivalence checking (EC) of combinational circuits is of great practical and theoretical importance. A key problem of electronic chip design is to build a circuit that implements a Boolean function f and at the same time optimizes a cost function. In a typical design flow, this task is performed by a logic synthesis procedure. Such a procedure starts with some circuit N implementing f, and then gradually modifies N trying to improve the value of the cost function. Each optimization step requires EC to prove that the modified implementation of f is functionally equivalent to the previous one. Typically, due to high complexity of EC, a logic synthesis procedure uses only very simple transformations. So any discovery of a new powerful EC procedure will enable more powerful logic optimization steps thus drastically improving the quality of synthesized circuits.

Studying EC may be also very useful from a theoretical point of view for the following reasons. In terms of EC, the problem of checking if a circuit N is satisfiable (further referred to as Circuit-SAT) reduces to testing inequivalence of N and a trivial circuit $N(0)$ implementing constant 0. In [5] we showed that EC of circuits with a "similar" structure is easy. This result implies that it may be unreasonable to check for equivalence circuits N and $N(0)$. (Because N may have a very involved structure while $N(0)$ does not have any structure at all and so N and $N(0)$ are definitely not "similar".) Instead, one may try to build a sequence of circuits $N_1,\ldots,N_k$ where $N_1 = N$ and $N_k = N(0)$ such that EC of a pair of consecutive circuits N_i,N_{i+1} is easy. Performing this kind of transitions

from a "hard" circuit to an "easy" one requires a good understanding of EC complexity.

In [5] we studied EC of circuits with a common specification (CS). A CS S of N_1 and N_2 is just a circuit of multi-valued gates (mv-gates for short) such that N_1 and N_2 are different implementations of S. An example of a CS is given in Figure 1. Circuits N_1 and N_2 have a CS of 3 mv-gates shown on the left. Subcircuits N_1^i, N_2^i are different implementations of multi-valued mv-gate G_i of S. Circuit N_m^i (m=1,2) implements a multi-output Boolean function whose truth table is obtained from that of G_i by replacing values of multi-valued variables with their binary codes. So the difference between N_1^i and N_2^i is in the choice of binary encodings for the variables of S.

The main result of [5] is that, given circuits N_1 and N_2 and their CS represented by partitions N_1^1,.., N_1^k and N_2^1,.., N_2^k, there is a short "specification driven" resolution proof that N_1 and N_2 are equivalent. This proof is linear in the number of subcircuits k and exponential in size of the largest subcircuit N_i^j, i=1,2, j=1,..., k (granularity of specification). (The main point of [5] was to show that there is a very important class of practical formulas of very low "non-deterministic" complexity that are most likely hard for any deterministic resolution based SAT-solver.)

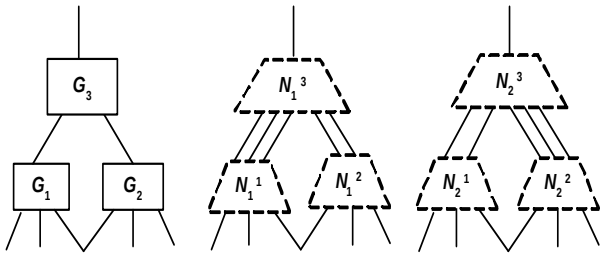

Fig. 1. Circuits N_1 and N_2 with a common specification of three mv-gates

In this paper we develop further the theory of [5] and report the following new results.

- In [5] we gave an "explicit" definition of a CS S of circuits N_1 and N_2 where one needs to know not only the functionality of mv-gates of S but also the binary encodings of multi-valued variables. In this paper we show that N_1 and N_2 have a CS iff N_1 and N_2 can be partitioned into toggle equivalent subcircuits that are connected "in the same way". This result allows one to represent a CS of N_1 and N_2 "implicitly" as a partitioning of N_1 and N_2 into subcircuits.
- Based on the result above, we give a new procedure for EC of circuits with a CS.
- We give parameterization of our EC procedure in terms of specification width that generalizes the notion of specification granularity introduced in [5].

– We clarify the relation between fixed-parameter tractability and parameterized EC

The novelties of our EC procedure are as follows.

– In contrast to [5], when checking for equivalence of circuits N_1 and N_2 with specifications S_1 and S_2, our EC procedure does not assume that S_1 and S_2 are identical. Instead, it verifies the identity of S_1 and S_2 "on the fly".
– Our procedure works not only if circuits N_1 and N_2 are equivalent (as in [5]) but also when they are anti-equivalent (a special case of inequivalence).
– In contrast to [5], no restrictions are imposed on the length of binary encodings of multi-valued variables (in [5] we considered only minimal length encodings).

For lack of space, this paper does not contain any experimental results. A comparison of our EC procedure with a powerful industrial equivalence checker is given in [6]. The report [6] also describes application of our theory to logic synthesis. Namely, in [6] we formulate a powerful logic synthesis procedure that, given a circuit N_1 with a specification (represented as partition of N_1 into subcircuits), generates a circuit N_2 that implements the same specification as N_1. Today's state-of-the-art algorithms can neither generate such a circuit N_2 nor prove it to be equivalent to N_1.

This paper is structured as follows. In Section 2 the relation between fixed-parameter tractability and parameterized complexity of EC is discussed. In Section 3 we recall the notion of circuits with specifications. Section 4 introduces the notion of toggle equivalence of Boolean functions. In Section 5 we show that circuits N_1 and N_2 have a CS iff they can be partitioned into toggle equivalent subcircuits that are connected "in the same way" in N_1 and N_2. In Section 6 we give a procedure for EC of circuits N_1 and N_2 with predefined specifications. In Section 7 we discuss the complexity of our EC procedure and introduce width based parameterization of EC. In Section 8 we explain why it is hard to find a CS. Finally, we draw some conclusions in Section 9.

2 Fixed parameter tractability and parameterized complexity of equivalence checking

In this section, we discuss (very informally) the relation between fixed parameter tractability and parameterized complexity of equivalence checking. The notion of parameterized tractability [4] is an elegant way to form tractable subproblems of computationally hard problems. To parameterize a problem K means to build a function $\xi(K)$ that maps each instance I of K to an integer $p{=}\xi(I)$. A parameterized problem K is called fixed-parameter (FP) tractable if there is an algorithm that solves every instance I of K in time $\mathrm{O}(g(p) * L^m)$. Here $g(p)$ is an arbitrary function of p, L is the length of I, and m is a constant. It is not hard to see that any set of instances of the problem K that has the same value of p can be solved by this algorithm in time $\mathrm{O}(L^m)$. (The same applies to any

set of instances of K that may have different values of p but the number of these values is finite.)

The EC procedure we give in this paper implies that EC has features of an FP tractable problem. Let K be the EC problem and I be an instance of K that is to check Boolean circuits N_1 and N_2 for equivalence. The function $\xi(K)$ is defined as follows. If N_1 and N_2 are functionally equivalent or anti-equivalent, then $p = \xi(I)$ is the width of a CS of N_1 and N_2 (see Section 7 for the definition). (Anti-equivalence means that Boolean functions implemented by N_1 and N_2 are complements of each other). As we will see if N_1 and N_2 are inequivalent but not anti-equivalent, then they do not have a CS. In that case, the value of $\xi(I)$ is arbitrary. Our EC procedure tests equivalence or anti-equivalence of N_1, N_2 or rejects the instance I (in case specifications of N_1 and N_2 are not identical) in time $O(g(p) * L)$ where L is the instance length. A major difference between parameterized EC and an FP tractable problem is that our EC procedure needs an "advice". This advice is given in the form of specifications $S_1 = \{N_1^1, .., N_1^k\}$ and $S_2 = \{N_2^1, .., N_2^k\}$ represented as partitions of N_1 and N_2 into subcircuits.

Let us explain the relation between parameterized EC and FP tractability by the example of Circuit-SAT. Circuit-SAT is an NP-complete problem which is to test if a circuit N is satisfiable. Circuit-SAT is known to be an FP tractable problem (see for example [3, 8]. The satisfiability of N can be solved in time $O(exp(c * w) * |N|)$ where w is the circuit width, c is a constant and $|N|$ is the circuit size. (We do not clarify which definition of circuit width we use because the discussion below applies to any of them.) The quick growth of $exp(c * w)$ makes it possible to efficiently solve Circuit-SAT only for very "narrow" circuits. On the other hand, our EC procedure can be efficiently applied to circuits N_1 and N_2 of arbitrary width. The limiting factor is the width of each pair N_1^i, N_2^i, $i = 1, .., k$ of corresponding subcircuits of S_1 and S_2. In other words, in the case of Circuit-SAT, the parameter describes the **"absolute"** circuit width. In the case of EC the parameter describes the **"differential"** width of circuits N_1 and N_2 that indicates how different N_1 and N_2 are.

One can view width-based tractability of Circuit-SAT as a "special case" of width-based tractability of EC. Indeed, on the one hand, testing the satisfiability of a circuit reduces to testing inequivalence of this circuit to constant 0. On the other hand, if N_2 is a trivial implementation of constant 0, then the relative width of N_1 and N_2 is actually the absolute width of N_1. While FP tractability looks for circuits that are "immediately" simple, parameterized EC implies the possibility of making circuits simpler "gradually" by performing "narrow" (and so easily verifiable) changes. For example, if the absolute width of N_2 is smaller than that of N_1 and the width of their CS is small, then one can replace $Circuit\_SAT(N_1)$ with $Circuit\_SAT(N_2)$ easily proving that this replacement is correct.

3 Boolean circuits with specifications

In this section, we recall the "old" definition of a CS of two circuits N_1 and N_2 given in [5]. In Section 5 we will give a new definition allowing to represent CS

"implicitly" as partitioning of N_1 and N_2 into toggle equivalent subcircuits. We will also show that these two definitions are equivalent.

An mv-circuit (here ***mv*** stands for ***multi-valued***) is a network of ***mv-gates***, exactly as a Boolean circuit is a network of Boolean gates. An mv-gate of n inputs is a device that implements an mv-function of n mv-variables. (Similarly, a Boolean gate of n inputs is a device implementing a Boolean function of n Boolean variables.)

We will call a Boolean function $\boldsymbol{f}$ (or a Boolean circuit implementing $\boldsymbol{f}$) specifying a mapping $\{0,1\}^m \rightarrow \{0,1\}^p$ a ***multi-output function*** (a multi-output circuit respectively) if $p \geq 1$ and a ***single-output function*** (a single output circuit respectively) if $p = 1$. Let X be a multi-valued variable. Let $\boldsymbol{q}(X)$ be a Boolean function mapping each value X' of X to a point of $\{0,1\}^n$. The function $\boldsymbol{q}(X)$ is called an (***n-bit***) ***encoding*** of X if different values X' and X'' of X are mapped to different points i.e. $\boldsymbol{q}(X') \neq \boldsymbol{q}(X'')$. The value of $\boldsymbol{q}$ at X' is called the ***code*** of X'.

Let $G(X_1, \ldots, X_n)$ be the mv-function implemented by an mv-gate G and Y be the mv-variable describing the output of G. A Boolean function $\boldsymbol{f}$ ***implements*** G if there are $n+1$ Boolean encodings $\boldsymbol{q}_1(X_1), \boldsymbol{q}_2(X_2), \ldots, \boldsymbol{q}_n(X_n), \boldsymbol{q}(Y)$ such that a) $Y = G(X_1, .., X_n)$ implies $\boldsymbol{q}(Y) = \boldsymbol{f}(\boldsymbol{q}_1(X_1), .., \boldsymbol{q}_n(X_n))$ and b) each combination of output values produced by $\boldsymbol{f}$ is a code of a value of Y. If a Boolean circuit N implements a Boolean function $\boldsymbol{f}$ implementing an mv-gate G, we will say that N implements G.

A multi-valued circuit S of k gates $G_1, .., G_k$ is called a ***specification*** of a Boolean circuit N if N is a composition of k subcircuits $N_1, .., N_k$ such that a) subcircuit N_i is an implementation of mv-gate gate G_i; b) subcircuits of N are connected as corresponding gates of S (for example, the circuit N_1 shown in the center of Figure 1 consists of three subcircuits N_1^1, N_1^2, N_1^3 connected as the corresponding mv-gates G_1, G_2, G_3 of the specification shown on the left.)

Definition 1. *An mv-circuit S is called a* ***common specification (CS)*** *of Boolean circuits N_1 and N_2 if S is a specification of both N_1 and N_2.*

In this paper we make the following ***assumptions***.

1. Each gate of a Boolean circuit has only two inputs (no assumptions are made about the number of inputs of an mv-gate).
2. Circuits N_1 and N_2 to be proven equivalent have only one output.
3. Every specification has only one output and this output takes only Boolean values. Every specification has only Boolean inputs. (This means, for example, that the inputs of gates G_1 and G_2 and the output of gate G_3 in Figure 1 take only Boolean values.) In other words, only "internal" variables of a specification can be multi-valued. All the "external" variables are Boolean.

4 Toggle equivalence of Boolean functions

According to Definition 1, to show that circuits N_1, N_2 have a CS one has to present an mv-circuit S and binary encodings of the mv-gates of S that prove

N_1 and N_2 to be implementations of S. In Sections 4 and 5 we show that one can represent a CS of N_1 and N_2 "implicitly" as a partitioning of N_1 and N_2 into toggle equivalent subcircuits. In this case, no explicit knowledge of encodings or the functionality of mv-gates is necessary.

In this section, we introduce the notion of toggle equivalence. We show that toggle equivalent Boolean functions can be considered as different implementations of the same multi-valued function.

4.1 Toggle equivalence of functions with identical sets of variables

Definition 2. *Let $\boldsymbol{f}_1$:$\{0,1\}^n \rightarrow \{0,1\}^m$ and $\boldsymbol{f}_2$:$\{0,1\}^n \rightarrow \{0,1\}^k$ be m-output and k-output Boolean functions of the same set of variables. Functions $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are called* ***toggle equivalent*** *if $\boldsymbol{f}_1(\boldsymbol{x}) \neq \boldsymbol{f}_1(\boldsymbol{x'}) \Leftrightarrow \boldsymbol{f}_2(\boldsymbol{x}) \neq \boldsymbol{f}_2(\boldsymbol{x'})$. Circuits N_1 and N_2 implementing toggle equivalent functions $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are called* ***toggle equivalent circuits.***

Informally, toggle equivalence means that for any pair of input vectors $\boldsymbol{x}$,$\boldsymbol{x'}$ for which at least one output of N_1 "toggles", the same is true for N_2 and vice versa.

Definition 3. *Let $\boldsymbol{f}$ be a multi-output Boolean function of n variables. Denote by Part($\boldsymbol{f}$) the partition of the set $\{0,1\}^n$ into disjoint subsets $B_1,\ldots,B_k$ such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x'})$ iff $\boldsymbol{x}$, $\boldsymbol{x'}$ are in the same subset B_i.*

Proposition 1. *Two Boolean functions $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are toggle equivalent iff Part($\boldsymbol{f}_1$)=Part($\boldsymbol{f}_2$) i.e. iff for each element B_i of the partition Part($\boldsymbol{f}_1$) there is an element B'_j of the partition Part($\boldsymbol{f}_2$) such that B_i=B'_j and vice versa.*

Proof. If $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are toggle equivalent, there cannot exist a pair of vectors $\boldsymbol{x}$,$\boldsymbol{x'}$ such that $\boldsymbol{x}$,$\boldsymbol{x'}$ are in the same subset of one partition and in different subsets of the other partition. (Because that would mean that one function produces two identical output assignments while the other function toggles.)

Proposition 2.
Let $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ be toggle equivalent single output Boolean functions. Then $\boldsymbol{f}_1$=$\boldsymbol{f}_2$ or $\boldsymbol{f}_1$=$\overline{\boldsymbol{f}}_2$ where $\overline{\boldsymbol{f}}_2$ is the negation of $\boldsymbol{f}_2$.

Proof. From Proposition 1 it follows that $Part(\boldsymbol{f}_1)$=$Part(\boldsymbol{f}_2)$. Since $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are single output Boolean functions, $Part(\boldsymbol{f}_1)$ and $Part(\boldsymbol{f}_2)$ consist of two elements each. So $\boldsymbol{f}_1$=$\boldsymbol{f}_2$ or $\boldsymbol{f}_1$=$\overline{\boldsymbol{f}}_2$.

Proposition 3. *Let $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ be toggle equivalent. Then $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are two different implementations of the same multi-valued function of Boolean variables.*

Proof. According to Proposition 1, $Part(\boldsymbol{f}_1)$=$Part(\boldsymbol{f}_2)$. Let $Part(\boldsymbol{f}_1)$, $Part(\boldsymbol{f}_2)$ consist of k elements each. Then $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are implementations of the function F: $\{0,1\}^n \rightarrow \{1,..,k\}$ where $F(\boldsymbol{x})$=m, iff $\boldsymbol{x} \in B_m$, i.e. iff $\boldsymbol{x}$ is in the m-th element of $Part(\boldsymbol{f_1})$.

4.2 Toggle equivalence of functions with different sets of variables

In this subsection, the notion of toggle equivalence is extended to the case of Boolean functions with different sets of variables that are related by constraint functions.

Definition 4. *Let X and Y be two disjoint sets of Boolean variables (the number of variables in X and Y may be different). A function Cf(X,Y) is called a* ***correlation function*** *if there are subsets $A^X \subseteq \{0,1\}^{|X|}$ and $A^Y \subseteq \{0,1\}^{|Y|}$ such that*

1. *$|A^X| = |A^Y|$ and*
2. *Cf(X,Y) specifies a bijective mapping M: $A^X \rightarrow A^Y$. Namely Cf($\boldsymbol{x}$, $\boldsymbol{y}$)=1 iff $\boldsymbol{x} \in A^X$ and $\boldsymbol{y} \in A^Y$ and $\boldsymbol{y} = M(\boldsymbol{x})$.*

Remark 1. Informally, $Cf(X,Y)$ is a correlation function if it specifies a bijective mapping between a subset A^X of $\{0,1\}^{|X|}$ and a subset A^Y of $\{0,1\}^{|Y|}$. So one can view $Cf(X,Y)$ as relating two different encodings of an $|A^X|$-valued variable.

As Proposition 4 below shows, one can check if a Boolean function $H(X,Y)$ is a correlation one without explicitly finding subsets A^X and A^Y.

Proposition 4.
Let X and Y be two disjoint sets of Boolean variables. A Boolean function $H(X,Y)$ is a correlation one iff the following two conditions hold.

1. *There do not exist three vectors $\boldsymbol{x}$, $\boldsymbol{x'}$,$\boldsymbol{y}$ (where $\boldsymbol{x}$, $\boldsymbol{x'}$ are assignments to variables of X and $\boldsymbol{y}$ is an assignment to variables of Y) such that $\boldsymbol{x} \neq \boldsymbol{x'}$ and $H(\boldsymbol{x},\boldsymbol{y})=H(\boldsymbol{x'},\boldsymbol{y})=1$.*
2. *There do not exist three vectors $\boldsymbol{x},\boldsymbol{y},\boldsymbol{y'}$ such that $\boldsymbol{y} \neq \boldsymbol{y'}$ and $H(\boldsymbol{x},\boldsymbol{y})= H(\boldsymbol{x},\boldsymbol{y'})=1$.*

Proof. Only if part. If $H(X,Y)$ is a correlation function, the fact that conditions 1) and 2) hold, follows from Definition 4.

If part. If conditions 1) and 2) hold then, $H(X,Y)$ specifies a bijective mapping between subsets A^X and A^Y defined in the following way. Subset A^X consists of all the assignments $\boldsymbol{x} \in \{0,1\}^{|X|}$ such that $H(\boldsymbol{x}, \boldsymbol{y})=1$ for some $\boldsymbol{y} \in \{0,1\}^{|Y|}$. Subset A^Y consists of all the assignments $\boldsymbol{y} \in \{0,1\}^{|Y|}$ such that $H(\boldsymbol{x}, \boldsymbol{y})=1$ for some $\boldsymbol{x} \in \{0,1\}^{|X|}$.

Remark 2. Checking if $H(X, Y)$ is a correlation function reduces to solving two instances of the satisfiability problem (SAT). Checking condition 1) of Proposition 4 reduces to testing the satisfiability of the expression $H(X, Y) \wedge H(X', Y') \wedge Neq(X, X') \wedge Eq(Y, Y')$. Here $H(X', Y')$ is a "copy" of $H(X, Y)$ where variables of X', Y' are independent of those of X, Y. $Neq(\boldsymbol{x},\boldsymbol{x'})$ is equal to 1 iff $\boldsymbol{x} \neq \boldsymbol{x'}$. Function $Eq(Y, Y')$ is the negation of $Neq(Y, Y')$. Checking condition 2) of Proposition 4 reduces to testing the satisfiability of $H(X, Y) \wedge H(X', Y') \wedge Eq(X, X') \wedge Neq(Y, Y')$. If either expression is constant 0 (i.e. unsatisfiable), then H is a correlation function.

Definition 5.
Let $\boldsymbol{f}_1$*:*$\{0,1\}^n \rightarrow \{0,1\}^m$ *and* $\boldsymbol{f}_2$*:*$\{0,1\}^p \rightarrow \{0,1\}^k$ *be m-output and k-output Boolean functions and X and Y specify their sets of Boolean variables where* $|X| = n$ *and* $|Y| = p$*. Let* $D_{inp}(X,Y)$ *be a Boolean function. Functions* $\boldsymbol{f}_1$ *and* $\boldsymbol{f}_2$ *are called* ***toggle equivalent under constraint function*** $D_{inp}(X,Y)$ *if* $(\boldsymbol{f}_1(\boldsymbol{x}) \neq \boldsymbol{f}_1(\boldsymbol{x'}) \wedge (D_{inp}(\boldsymbol{x}, \boldsymbol{y}) = D_{inp}(\boldsymbol{x'},\boldsymbol{y'})=1)) \Rightarrow (\boldsymbol{f}_2(\boldsymbol{y}) \neq \boldsymbol{f}_2(\boldsymbol{y'})$ *and vice versa* $(\boldsymbol{f}_2(\boldsymbol{y}) \neq \boldsymbol{f}_2(\boldsymbol{y'}) \wedge (D_{inp}(\boldsymbol{x},\boldsymbol{y})=D_{inp}(\boldsymbol{x'},\boldsymbol{y'})=1)) \Rightarrow \boldsymbol{f}_1(\boldsymbol{x}) \neq \boldsymbol{f}_1(\boldsymbol{x'})$*.*

Proposition 5. *Let X,Y be sets of Boolean variables and* $\{X_1, \ldots, X_s\}$ *and* $\{Y_1, \ldots, Y_s\}$ *be partitions of X and Y respectively. Let* $Cf(X_1,Y_1),..,Cf(X_s,Y_s)$ *be correlation functions. Let* $\boldsymbol{f}_1(X)$ *and* $\boldsymbol{f}_2(Y)$ *be toggle equivalent under the constraint function* $D_{inp}(X,Y)=Cf(X_1,Y_1) \wedge \ \ldots \ \wedge Cf(X_s,Y_s)$*. Then* $\boldsymbol{f}_1$ *and* $\boldsymbol{f}_2$ *are implementations of the same multi-valued function of s multi-valued variables.*

Proof follows from Proposition 3 and Remark 1.

4.3 Testing toggle equivalence

In this subsection, we show how to test toggle equivalence of multi-output Boolean circuits N_1 and N_2. Namely we show that testing the toggle equivalence of N_1 and N_2 reduces to checking if function $D_{out}(N_1, N_2)$ specified by Definition 8 (see below) is a correlation one. This test can be performed by two satisfiability checks as described in Remark 2.

Definition 6. *Let N be a Boolean circuit. Denote by* $v(N)$ *the set of Boolean variables associated with the output or an input of a gate of N. Denote by* ***Sat(v(N))*** *the Boolean function such that* $Sat(z)=1$ *iff the assignment z to variables* $v(N)$ *is "possible" i.e consistent. For example, if circuit N consists of just one AND gate* $y = x_1 \wedge x_2$*, then* $v(N)=\{y, x_1, x_2\}$ *and* $Sat(v(N))=(\overline{x}_1 \vee \overline{x}_2 \vee y) \wedge (x_1 \vee \overline{y}) \wedge (x_2 \vee \overline{y})$*.*

Definition 7. *Let f be a Boolean function. We will say that function* f^* *is obtained from f by* ***existentially quantifying away variable*** *x if* $f^* = f(\ldots, x=0,\ldots) \vee f(\ldots, x=1,\ldots)$*.*

If f is represented as a CNF formula F, then to existentially quantify away a variable x one just needs to add to F all the clauses produced by resolving clauses of F in x and to remove from F all the clauses having a literal of x.

Definition 8. *Let* N_1 *and* N_2 *be Boolean circuits whose inputs are specified by sets of variables X and Y respectively. Let* $D_{inp}(X,Y)$ *be a Boolean function. Denote by* $\boldsymbol{D_{out}(N_1, N_2)}$ *the Boolean function obtained from the Boolean function H, where* $H=Sat(v(N_1)) \wedge Sat(v(N_2)) \wedge D_{inp}(X,Y)$*, by existentially quantifying away all the variables of H but the output variables of* N_1 *and* N_2 *(i.e. the variables associated with outputs of* N_1 *and* N_2*).*

Proposition 6. *Let N_1 and N_2 be Boolean circuits with input variables specified by sets X, Y respectively. Let $D_{inp}(X,Y)$ be a Boolean function relating X and Y. Let $D_{inp}(X,Y)$ be a correlation function. Then N_1 and N_2 are toggle equivalent under constraint function $D_{inp}(X,Y)$ iff the function $D_{out}(N_1,N_2)$ specified in Definition 8 is also a correlation function.*

Proof. Only If part. Let N_1 and N_2 be toggle equivalent under constraint function $D_{inp}(X,Y)$. Then $D_{out}(N_1,N_2)$ satisfies either condition of Proposition 4 and hence it is a correlation function. For example, there cannot exist Boolean vectors $\boldsymbol{z}$, $\boldsymbol{z'}$ and $\boldsymbol{h}$ (where $\boldsymbol{z} \neq \boldsymbol{z'}$ and $\boldsymbol{z}$, $\boldsymbol{z'}$ are output assignments of N_1 and $\boldsymbol{h}$ is an output assignment of N_2) such that $D_{out}(\boldsymbol{z},\boldsymbol{h})=D_{out}(\boldsymbol{z'},\boldsymbol{h})=1$. Indeed, it would mean that there exist pairs of vectors $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{x'}$, $\boldsymbol{y'}$ such that a) $\boldsymbol{z}=N_1(\boldsymbol{x})$, $\boldsymbol{z'} = N_2(\boldsymbol{x'})$ and $\boldsymbol{h} = N_2(\boldsymbol{y})=N_2(\boldsymbol{y'})$;b) $D_{inp}(\boldsymbol{x}, \boldsymbol{y})=1$ and $D_{inp}(\boldsymbol{x'}, \boldsymbol{y'})=1$; c) $\boldsymbol{x} \neq \boldsymbol{x'}$ and $\boldsymbol{y} \neq \boldsymbol{y'}$; d) $N_1(\boldsymbol{x}) \neq N_1(\boldsymbol{x'})$ while $N_2(\boldsymbol{y}) = N_2(\boldsymbol{y'})$. But this is impossible because N_1 and N_2 are toggle equivalent.

If part can be proven in a similar manner.

5 Common specification and toggle equivalence

In this section, we show that the existence of a CS of single output combinational circuits N_1 and N_2 means that N_1, N_2 can be partitioned into toggle equivalent subcircuits that are connected in N_1 and N_2 "in the same way". This result is formulated in Proposition 7.

Definition 9. *Let $N=(V,E)$ be a directed acyclic graph (**DAG**) representing a Boolean circuit. (Here V,E are sets of nodes and edges of N respectively.) A subgraph $N^*=(V^*, E^*)$ of N is called a **subcircuit** if the following two conditions hold:*

1. *if g_1, g_2 are in V^* and there is a path from g_1 to g_2 in N, then all the nodes of N on that path are in V^* ;*
2. *if g_1, g_2 of V^* are connected by an edge in N, then they are also connected by an edge in N^*.*

Definition 10. *Let N^* be a subcircuit of N. An input of a gate g of N^* is called **an input** of N^* if it is not connected to the output of some other gate of N^*. The output of a gate g is called **an output** of subcircuit N^* if a) it is the output of N or b) it is connected to an input of a gate of N that is not in N^*.*

Definition 11. *Let a Boolean circuit N be partitioned into k subcircuits N^1, ..., N^k . Let T be a directed graph of k nodes such that nodes G_i and G_j of T are connected by a directed edge (from G_i to Gj) iff an output of N^i is connected to an input of N^k in N. T is called **the communication specification** corresponding to the partition N^1, ..., N^k. The partition N^1, ..., N^k is called **topological** if T is a DAG (i.e. if T does not contain cycles)*

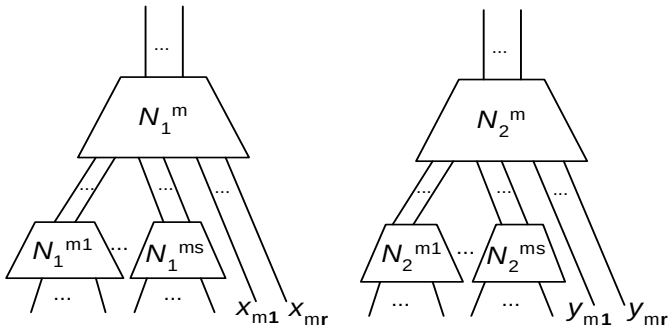

Fig. 2. Illustration to Definition 13

Definition 12. *Let T be the communication specification of circuit N with respect to a topological partition N^1, ..., N^k. Let G_i be the node of T corresponding to subcircuit N^i. The length of the longest path from an input of T to G_i is called the* ***level*** *of G_i and N^i (denoted by level(G_i) and level(N^i) respectively).*

Definition 13. *Let N_1^1, ...,N_1^k and N_2^1, ..., N_2^k be topological partitions of single output Boolean circuits N_1,N_2. Let communication specifications of N_1 and N_2 with respect to partitions N_1^1, ..., N_1^k and N_2^1, ..., N_2^k be identical. Denote by $\boldsymbol{D_{out}(N_1^m,N_2^m)}$, $\boldsymbol{m=1,\ldots,k}$ functions computed by induction in topological levels. Namely, we first compute functions D_{out} for the subcircuits of level 1, then for those of level 2 and so on. Function $D_{out}(N_1^m,N_2^m)$ is obtained from function H_m=$Sat(v(N_1^m)) \wedge Sat(v(N_2^m)) \wedge D_{inp}(N_1^m,N_2^m)$ by existentially quantifying away all the variables except the output variables of N_1^m, N_2^m. The function $D_{inp}(N_1^m,N_2^m)$ is equal to $D_{out}(N_1^{m1},N_2^{m1}) \wedge \ldots \wedge D_{out}(N_1^{ms},N_2^{ms}) \wedge Eq(x_{m1},y_{m1}) \wedge \ldots \wedge Eq(x_{mr},y_{mr})$. Here $N_1^{m1},..,N_1^{ms},N_2^{m1},..,N_2^{ms}$ are the s subcircuits (if any) whose outputs are connected to inputs of N_1^m,N_2^m respectively. (See illustration in Figure 2.) Variables $x_{m1},\ldots,x_{mr},y_{m1},..,y_{mr}$ are the r input variables of N_1 and N_2 (if any) that feed N_1^m and N_2^m respectively. Function $Eq(x_{mt},y_{mt})$, $1 \leq t \leq r$ is equal to 1 iff x_{mt} is equal to y_{mt}.*

Proposition 7. *Let N_1, N_2 be two single-output Boolean circuits and T be a DAG of k nodes. Circuits N_1 and N_2 are implementations of a specification S whose topology is given by T iff there is a partition $Spec(N_1) = \{N_1^1, ..., N_1^k\}$ of N_1 and a partition $Spec(N_2) = \{N_2^1, ..., N_2^k\}$ of N_2 into k subcircuits such that*

- *Communication specifications T_1,T_2 of N_1 and N_2 with respect to partitions $Spec(N_1)$, $Spec(N_2)$ are equal to T;*
- *Each pair of circuits N_1^m, N_2^m is toggle equivalent under constraint function $D_{inp}(N_1^m, N_2^m)$ specified by Definition 13.*

Sketch of the proof. If part. It is proven by induction (in levels) using Proposition 5 and Proposition 6. ***Only if part*** is proven by induction using the

fact that two Boolean functions implementing the same multi-valued function are toggle equivalent.

Proposition 7 allows one to specify a CS S of circuits N_1 and N_2 "implicitly" by giving partitions $\{N_1^1,.., N_1^k\}$ and $\{N_2^1,\ldots, N_2^k\}$. Of course, knowing these partitions one can always obtain the functionality of all mv-gates of S and find the binary encodings using which circuits N_1 and N_2 can be produced from S.

From Proposition 7 it follows that if N_1 and N_2 have a CS, then N_1^k and N_2^k (i.e. the "last" subcircuits of N_1 and N_2 respectively) and hence N_1 and N_2 are toggle equivalent. From Proposition 2 it follows then that N_1 and N_2 are either equivalent or anti-equivalent. In other words, if N_1 and N_2 are neither equivalent nor anti-equivalent, they have no CS.

6 A Procedure for equivalence checking of circuits with a common specification

In this section, we describe a procedure for EC of circuits N_1, N_2 with predefined specifications S_1, S_2. We will refer to this procedure as ECCS (EC of circuits with a CS). The ECCS procedure essentially mimics a specification guided resolution proof introduced in [5]. However there are a few differences.

- The ECCS procedure does not assume that specifications S_1 and S_2 are identical. Instead, using Proposition 7, it checks if S_1 and S_2 are identical on the fly.
- The ECCS procedure not only proves that N_1 and N_2 are functionally equivalent but can also prove that N_1 and N_2 are anti-equivalent.
- The ECCS procedure is described in terms of functions and existential quantification independently of the way these functions are represented.

The pseudocode of the ECCS procedure is shown in Figure 3. The procedure *topol_partition* checks if *Spec(*N_1*)* and *Spec(*N_1*)* are topological partitions (see Definition 11). The procedure *equiv_commun_specs* checks if communication specifications T_1 of N_1 (with respect to *Spec(*N_1*)*) and T_2 of N_2 (with respect to *Spec(*N_2*)*) are identical.

In the main loop, functions $D_{out}(N_1^i, N_2^i)$ are computed in topological order as described in Definition 13. Before computing $D_{out}(N_1^i, N_2^i)$ the procedure *constr_func* forms the expression D_{inp} (see Definition 13). The *exist_quantify* procedure existentially quantifies away from the function $H_i = Sat(v(N_1^i)) \wedge Sat(v(N_2^i)) \wedge D_{inp}$ all the variables except the output variables of N_1^i and N_2^i. Then the *correlation_function* procedure checks if the result of quantification D_{out} is a correlation function. This check reduces to two SAT-checks as described in Remark 2.

Finally, the ECCS procedure checks if the correlation function of $D_{out}(N_1^k, N_2^k)$ relating outputs N_1^k and N_2^k (and so outputs of N_1 and N_2) implies that N_1 and N_2 are equivalent or anti-equivalent. If $D_{out}(N_1^k, N_2^k)$ is equal to $(\overline{z}_1 \vee z_2) \wedge (z_1 \vee \overline{z}_2)$ (here z_1 and z_2 are Boolean variables associated with the output of N_1^k and N_2^k respectively), then N_1 and N_2 are functionally equivalent. If $D_{out}(N_1^k,$

```
/* Spec(N1)= {N1^1,..,N1^k},Spec(N2)= {N2^1,..,N2^k} */
ECCS(N1, N2, Spec(N1),Spec(N2)) {
      if (topol_partition(N1,N2,Spec(N1),Spec(N2)) == 'no')
        return('reject');
      if (equiv_commun_specs( N1,N2,Spec(N1),Spec(N2)) == 'no')
        return('reject');
      for (i=1; i <= k ; i++) {
        D_inp = constr_func(N1^i,N2^i,N1,N2);
        D_out(N1^i, N2^i) = exist_quantify(N1^i,N2^i, D_inp);
        if (correlation_function(D_out) == 'no')
          return('reject');}

/* At this point we know that D_out(N1^k, N2^k) is a correlation function */
      if (D_out(N1^k, N2^k) implies equivalence of N1 and N2 )
        return('equivalent');
      if (D_out(N1^k, N2^k) implies anti-equivalence N1 and N2)
        return('anti-equivalent');}
```

Fig. 3. Pseudocode of the ECCS procedure

N_2^k) is equal to $z_1 \wedge z_2$ (respectively $\overline{z}_1 \wedge \overline{z}_2$) then N_1 and N_2 are functionally equivalent and implement constant 1 (respectivley 0). If $D_{out}(N_1^k, N_2^k)$ is equal to $(z_1 \vee z_2) \wedge (\overline{z}_1 \vee \overline{z}_2)$ or $z_1 \wedge \overline{z}_2$ or $\overline{z}_1 \wedge z_2$, then functions implemented by N_1 and N_2 are complements of each other. It is not hard to show that any Boolean function of z_1,z_2 different from the six functions above is not a correlation function.

ECCS procedure returns the *'reject'* answer if any of the checks performed by *topol_partition, equiv_commun_specs* and *correlation_function* fails. The rejection means that partitions *Spec*(N_1) and *Spec*(N_2) do not represent a CS of N_1 and N_2.

7 Parameterization of EC by specification width

In this section we discuss the complexity of the ECCS procedure and give a new parameterization of EC based on specification width.

First we recall the parameterization based on specification granularity [5].

Definition 14. *Let N_1, N_2 be two Boolean circuits with a CS S represented by partitions Spec(N_1)=$\{N_1^1,..,N_1^k\}$, and Spec(N_2)= $\{N_2^1,..,N_2^k\}$. The* ***granularity*** *of S is the size (i.e. the number of gates) of the largest subcircuit N_i^j, i=1,2, j=1,..,k.*

The ECCS procedure is exponential in the granularity p of S and linear in the number of subcircuits in *Spec*(N_1), *Spec*(N_2) (i.e. in the number of mv-gates in S). The exponentiality in p is due to procedures *exist_quantify* and *correlation_function*. The reason why the ECCS procedure is exponential only

in p and not in the circuit size is that the two exponential procedures above are applied only to subcircuits N_1^i, N_2^i whose size is bounded by p. So the time complexity of the ECCS procedure is the same as the size of a specification guided resolution proof from [5].

The granularity based parameterization can be easily improved using the following two observations. First, suppose that the number of outputs of a subcircuit N_i^j, i=1,2, j=1,..,k is bounded by a constant w. Then the complexity of the *correlation_function* procedure is bounded by 3^{4w}. Indeed, the number of variables of the function $D_{out}(N_1^i, N_2^i)$ is bounded by $2 * w$. Testing if $D_{out}(N_1^i, N_2^i)$ is a correlation function reduces to two SAT-checks over functions of $4 * w$ variables (see Remark 2) . In general resolution, the complexity of either check is bounded by 3^{4w} (the total number of clauses of $4 * w$ variables).

Second, to quantify from the function H_i=$Sat(v(N_1^i)) \wedge Sat(v(N_2^i)) \wedge D_{inp}(N_1^i, N_2^i)$ all the variables except the output variables of N_1^i, N_2^i, it suffices to perform no more than 2^{2w} SAT-checks. Each SAT-check just tests if a function $H_i \wedge A$ is satisfiable. Here A is a Boolean function that is equial to 1 iff a particular set of assignments to the outputs of N_1^i, N_2^i is made. So if the compelxity of each SAT-check is bounded by a function $f(w')$ where w' is another parameter, then the complexity of the ECCS procedure is bounded by $(f(w')*2^{2w}+3^{4w}) * k$ where k is the number of subcircuits in $Spec(N_1)$ and $Spec(N_2)$. So, if the values of w and w' are bounded, the complexity of the ECCS procedure is linear in the size of N_1 and N_2.

Let $CNF(H_i \wedge A)$ be a CNF representing $H_i \wedge A$. To parameterize the complexity of checking the satisfiability of $H_i \wedge A$ we use the results on FP tractability of SAT [8]. This FP tractability is achieved by parameterizing SAT with the width of a graph relating variables and clauses of a CNF formula. Given a formula F, one can build different graphs relating clauses and variables of F (e.g. incidence graph, primal graph, hypergraph [8]). Besides, one can use different definitions of graph width (e.g. cut-width [2, 3, 7], tree width, branch width [1, 8]). We do not choose a particular width-based parameterization here because the reasoning below is mostly independent of such a choice.

Denote by $G(H_i \wedge A)$ a graph of choice to describe a relation between clauses and variables of $CNF(H_i \wedge A)$. To complete parametrization we show how one can compute $G(H_i \wedge A)$. Of course its computing is trivial if $CNF(H_i \wedge A)$ is given explicitly. However, this is not the case. Nevertheless one can "approximate" $G(H_i \wedge A)$ as follows. $CNF(H_i \wedge A)$ can be represented as $CNF(Sat(v(N_1^i))) \wedge CNF(Sat(v(N_2^i))) \wedge CNF(D_{inp}(N_1^i, N_2^i)) \wedge CNF(A)$. Knowing circuits N_1^i and N_2^i one can easily build CNF formulas $CNF(Sat(v(N_1^i)))$, $CNF(Sat(v(N_2^i)))$. So their contribution to $G(H_i \wedge A)$ can be easily computed. $CNF(A)$ can be represented as a conjunction of unit clauses, each clause describing an assignment one has to make to an output of N_1 or N_2. Since the number of outputs of N_1^i and N_2^i is known, the contribution of $CNF(A)$ to $G(H_i \wedge A)$ is easy to compute. To compute the contribution of $CNF(D_{inp}(N_1^i, N_2^i))$ to $G(H_i \wedge A)$ we can just assume the "worst" case. Namely, if outputs of the circuit N_1^j (respectively N_2^j) are connected to inputs of N_1^i (respectively to inputs

of N_2^i) we can assume that $CNF(D_{inp}(N_1^i, N_2^i))$ contains 3^q clauses each clause containing all the variables corresponding to the outputs of N_1^j and N_2^j. Here q is the total number of outputs of N_1^j and N_2^j. (By assumption, $q \leq 2 * w$.) The definition below summarizes our effort to parameterize EC by specification width.

Definition 15. *Let N_1 and N_2 be two Boolean circuits and partitions Spec(N_1)= $\{N_1^1,.., N_1^k\}$ and Spec(N_2)=$\{N_2^1,.., N_2^k\}$ represent their specifications. The **specification width** is the maximum of w and w' where w is the maximum number of outputs of a circuit N_i^j,i=1,2, j=1,., k and w' is the maximum width of graph $G(H_i \wedge A)$ for all i. (Graph $G(H_i \wedge A)$ is built as described above.)*

8 Why it is hard to find a common specification

As we mentioned in Section 2 in contrast to an FP tractable problem, parameterized EC of circuits N_1, N_2 needs an "advice" in the form of a CS. A natural question is whether this advice is crucial or there is an efficient algorithm that, given circuits N_1, N_2, can efficiently find a CS of width w. In [5] it was conjectured that finding a CS is hard for any deterministic algorithm. (So EC of circuits may serve as a source of formulas that have linear complexity, say, in general resolution but are extremely hard for a deterministic algorithm.) Armed with the new definition of CS based on toggle equivalence we can better justify this conjecture.

Suppose that N_1 and N_2 are two circuits to be checked for equivalence and we only know that N_1 and N_2 have a CS of width w. To find such a CS one needs to find a partition $\{N_1^1,.., N_1^k\}$ of N_1 and a partition $\{N_2^1,\ldots, N_2^k\}$ of N_2 into subcircuits that are connected in the "same way" and are toggle equivalent. Here we face the following two big problems. The first problem is that finding a pair of subcircuits that are toggle equivalent is computationally very expensive. Even though the parameter w limits the number of outputs a subcircuit can have, the number of candidate subcircuits in N_j,j=1,2 is roughly speaking proportional to $|N_j|^w$ (here $|N_j|$ is the number of gates in N_j).

The second problem is that if N_1^i, N_2^i is a part of a CS S, they have to satisfy the following two conditions.

- One should be able to check toggle equivalence of N_1^i, N_2^i in terms of their local inputs restricted by previously computed constraint function D_{inp}.
- The same should hold for the subcircuits of S whose inputs are connected to outputs of N_1^i, N_2^i.

Thus, even if we find two subcircuits N_1^i, N_2^i of width w that are toggle equivalent, it does not necessarily mean that they are a part of any CS of N_1 and N_2 of width w. (It may be the case that they are toggle equivalent "accidentally" and so we will not be able to find any toggle equivalent subcircuits of width w that are connected to outputs of N_1^i and N_2^i.) Due to the two problems above, it is very unlikely that there is an efficient algorithm for finding a CS of bounded width.

9 Conclusions

We give an efficient procedure for checking the equivalence of Boolean circuits with a common specification of bounded width. This result implies that one can consider parameterized equivalence checking as a generalization of FP tractability of Circuit-SAT. Instead of looking for circuits that are easy in absolute terms (e.g circuits of bounded width) one can search for pairs of circuits that are easy "relatively" i.e. with respect to each other (e.g circuits with a CS of bounded width). One can view parameterized EC as a way to study "natural" transformations of a circuit i.e transformations that preserve its functionality and can be easily verified.

10 Acknowledgments

The author would like to thank Edward Hirsch of Steklov Institute of Mathematics at St.Petersburg, Felice Balarin of Cadence Berkeley Labs, and anonymous reviewers for their valuable feedback.

References

1. A. Alekhnovich, A.Razborov. *Satisfiability, Branch-width and Tseitin Tautologies.* FOCS-2002,pp. 593-603.
2. C.L.Berman. *Circuit width, Register Allocation and Ordered Binary Decision Diagrams.* IEEE Trans. CAD. 10(8),pp.1059-1066,Aug. 1991.
3. E. A. Broering, S. V. Lokam. *Width-Based Algorithms for Satisfiability.* Proceedings of SAT 2003. Lecture Notes in Computer Science (LNCS), Volume 2919, pp. 162-171, 2004.
4. R.G. Downey, M.R.Fellows. *Parameterized complexity.* Springer-Verlag 1999.
5. E.Goldberg,Y.Novikov. *How good can a resolution based SAT-solver be?* SAT-2003, LNCS 2919,pp.37-52.
6. E.Goldberg. *On equivalence checking and logic synthesis of circuits with a common specification.* Cadence Berkeley Labs, Technical report, CDNL-TR-2004-1220,August 2004, (http://eigold.tripod.com/papers/tr-2004-1220.pdf).
7. M.R.Prasad,P.Chong, and K. Keutzer. *Why is Combinatorial ATPG Efficiently Solvable for Practical VLSI Circuits?* Journal of Electronic Testing: Theory and Applications, vol. 17,pp.509-527,2001.
8. S. Szeider. *On Fixed-parameter Tractable Parameterizations of SAT.* Proceedings of SAT-2003. Lecture Notes in Computer Science (LNCS) Volume 2919, pp. 188-202, 2004.