Exploiting Inherent Characteristics of Reversible Circuits for Faster Combinational Equivalence Checking

Luca Amarú[1], Pierre-Emmanuel Gaillardon[1], Robert Wille[2,3], Giovanni De Micheli[1]
[1]Integrated Systems Laboratory (LSI), EPFL, Lausanne, Switzerland
{luca.amaru, pierre-emmanuel.gaillardon, giovanni.demicheli}@epfl.ch
[2]Integrated Circuit and System Design, Johannes Kepler University, Linz, Austria
robert.wille@jku.at
[3]University of Bremen and DFKI GmbH, Bremen, Germany
rwille@informatik.uni-bremen.de

***Abstract*— Reversible circuits implement invertible logic functions. They are of great interest to cryptography, coding theory, interconnect design, computer graphics, quantum computing, and many other fields. As for conventional circuits, checking the combinational equivalence of two reversible circuits is an important but difficult (co$\mathcal{NP}$-complete) problem. In this work, we present a new approach for solving this problem significantly faster than the state-of-the-art. For this purpose, we exploit inherent characteristics of reversible computation, namely bi-directional (invertible) execution and the XOR-richness of reversible circuits. Bi-directional execution allows us to create an *identity miter* out of two reversible circuits to be verified, which naturally encodes the equivalence checking problem in the reversible domain. Then, the abundant presence of XOR operations in the identity miter enables an efficient problem mapping into XOR-CNF satisfiability. The resulting XOR-CNF formulas are eventually more compact than pure CNF formulas and potentially easier to solve. As previously anticipated, experimental results show that our equivalence checking methodology is more than one order of magnitude faster, on average, than the state-of-the-art solution based on established CNF-formulation and standard SAT solvers.**

I. Introduction

Reversible computing is a *non-conventional* computing style where all logic processing is conducted through bijective, i.e., invertible, Boolean functions. Reversible circuits implement invertible Boolean functions at the logic level and are represented as cascades of reversible gates. In conventional technologies, reversible circuits find application in cryptography [1], coding theory [2], interconnect design [3], computer graphics [4] and many other fields where the logic invertibility is a key asset. In emerging technologies, such as quantum computing [5], reversible circuits are one of the primitive computational building blocks.

Whether they are finally realized in conventional or emerging technologies, the design of reversible circuits faces two major conceptual challenges: synthesis and verification [6]. Synthesis maps a target Boolean function into the reversible logic domain while minimizing the number of additional information bits and primitive gates [7], [8]. Verification checks if the final reversible circuit conforms to the original specification [9].

In this work, we focus on reversible circuit verification and, in particular, on combinational equivalence checking. The problem of combinational equivalence checking consists of determining whether two given reversible circuits are functionally equivalent or not. As for conventional circuits, this is a difficult (co$\mathcal{NP}$-complete) problem [10]. We present a new approach for solving this problem significantly faster than the state-of-the-art verification approaches [9].

For this purpose, our methodology exploits, for the first time, inherent characteristics of reversible computation, i.e., its invertible execution and the XOR-richness of reversible circuits. This stands in contrast to previously proposed solutions such as introduced in [9] which only adapted established verification schemes for conventional circuits but ignored the potential of the reversible computing paradigm. Our proposed methodology consists of the following steps. First, we create an identity miter by cascading one circuit with the inverse of the other. If the two reversible circuits are functionally equivalent, then the resulting cascade realizes the identity function. Next, we encode the problem of checking whether the resulting circuit indeed realizes the identity into a mixed XOR-CNF satisfiability problem. The possibility to express natively XOR operations, frequently appearing in reversible circuits, reduces significantly the number of variables and clauses as compared to a pure CNF formulation. Finally, we solve the XOR-CNF satisfiability problem using CryptoMiniSat [11], a MiniSat-based solver handling XORs through Gaussian elimination [12]. Experimental results show that, on average, the proposed methodology is more than one order of magnitude faster than the state-of-the-art reversible circuit checker based on the established CNF-formulation and MiniSat solver [9]. Besides that, the proposed approach also provides potential for improving combinational equivalence checking of conventional circuits.

The remainder of this work is organized as follows. Section II provides the background on reversible circuits and on Boolean satisfiability. Section III presents the proposed methodology for equivalence checking of reversible circuits. Section IV describes the setup applied for our experimental evaluation and summarizes the obtained results. Section V discusses the future research directions – in particular for combinational equivalence checking of conventional circuits. Section VI concludes the paper.

II. Background

In this section, we briefly review the background on reversible circuits and on Boolean satisfiability.

A. Reversible Circuits

A logic function $f : \mathbb{B}^{n_i} \rightarrow \mathbb{B}^{n_o}$ is *reversible* if and only if it represents a bijection. This implies that:

- the number of inputs is equal to its number of outputs (i.e., $n_i = n_o$) and
- it maps each input pattern to a unique output pattern.

A reversible function can be realized by a circuit $G = g_1 g_2 \dots g_d$ comprised of a cascade of reversible gates g_i, where d is the number of gates. Multiple forks and feedback are not directly allowed [5]. Several different reversible gates have been introduced including the Toffoli gate [13], the Fredkin gate [14], and the Peres gate [15]. In accordance to the common approach in reversible circuit design (see e.g., [7], [8]), we focus on Toffoli gates in the following. Toffoli gates are universal, i.e., all reversible functions can be realized by means of this gate type alone [13].

A *Toffoli gate* has a *target line* t and *control lines* $\{c_1, c_2, \dots, c_n\}$[1]. Its behavior is the following: If all control lines are set to the logic value 1, i.e., $c_1 \cdot c_2 \cdot \dots \cdot c_n = 1$, the target line t is inverted, i.e., t'. Otherwise, the target line t is passed through unchanged.

[1]Toffoli gates have originally been introduced with just two control lines. However, their functionality is commonly extended to n control lines.

978-3-9815370-7-9/DATE16/

Hence, the Boolean function of the target line can be expressed as $(c_1 \cdot c_2 \cdot \ldots \cdot c_n) \oplus t$. All remaining signals (including the signals of the control lines) are always passed through unchanged. Fig. 1 depicts a Toffoli gate with its respective output functions. We follow the established drawing convention of using the symbol $\oplus$ to denote the target line and solid black circles to indicate control connections for the gate.

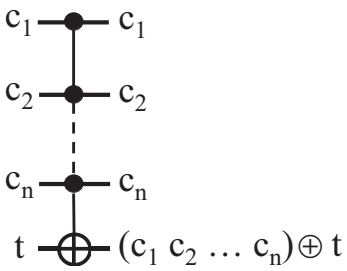

Fig. 1: A Toffoli gate.

A Toffoli gate with no control lines always inverts the target line and is a *NOT gate*. A Toffoli gate with a single control line is called a *controlled-NOT gate* (also known as the CNOT gate). The case of two control lines is the original gate defined by Toffoli [13].

Example 1. *Fig. 2 shows a reversible circuit composed of* $m = 3$ *circuit lines and* $d = 6$ *Toffoli gates. This circuit maps each input pattern into a unique output pattern. For example, it maps the input pattern* 111 *to the output pattern* 100*. Inherently, every computation can be performed in both directions (i.e., computations towards the outputs* and *towards the inputs can be performed).*

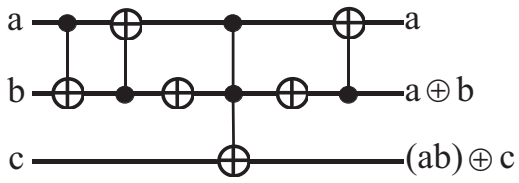

Fig. 2: A reversible circuit composed of Toffoli gates

B. Boolean Satisfiability

The Boolean *Satisfiability* (SAT) problem consists of determining whether there exists or not an assignment of variables so that a Boolean formula evaluates to true. In its most common formulation, Boolean SAT deals with *Conjunctive Normal Form* (CNF) formulas, i.e., a conjunction of disjunctions (clauses). For example, a standard CNF is

$$(a + b')(a + c')(a' + b + c)$$

which is satisfiable by $(a = 1, b = 1, c = 1)$.

Even though SAT for generic CNFs is a difficult ($\mathcal{NP}$-complete) problem, modern SAT solvers can handle fairly large problems in reasonable time [16]. The core technique behind most SAT solvers is the DPLL (Davis-Putnam-Logemann-Loveland) procedure, introduced several decades ago [17]. It basically performs a backtrack search in the space of partial truth assignments. Through the years, the main improvements to DPLL have been smart branch selection heuristics, a fast implication scheme, and extensions such as clause learning, randomized restarts, as well as well-crafted data structures such as lazy implementations and watched literals for fast unit propagation [16].

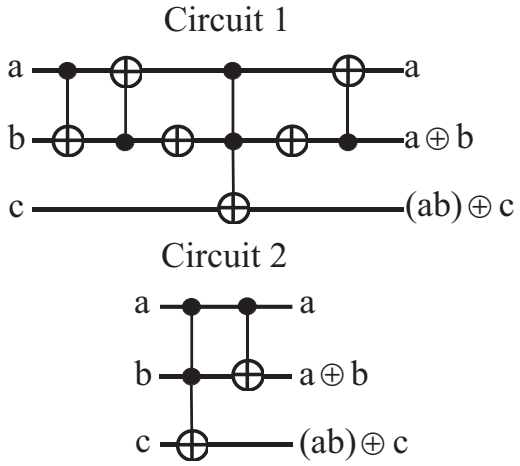

Fig. 3: Two functionally equivalent reversible circuits.

Recently, researchers considered SAT to solve other important problems in computer science, for example, cryptographic applications [19]. Here, SAT solvers are often faced with a large amount of XOR constraints. These XORs are typically difficult to handle using pure CNF and standard SAT solvers. However, the presence of these XOR constraints can be exploited within a DPLL solving framework by using on-the-fly Gaussian elimination [12]. Some SAT solvers have been proposed which exploit this potential and, hence, work on mixed XOR-CNF formulas rather than pure CNF formulas. For example, a mixed XOR-CNF is

$$(a \oplus b')(a \oplus c)(a' + b + c)$$

which is satisfiable by $(a = 1, b = 1, c = 0)$.

CryptoMiniSat [11] is one of the most popular solvers for XOR-CNF formulas based on MiniSat [24] and Gaussian elimination to handle XOR constraints [12].

III. Mapping Combinational Equivalence Checking for Reversible Circuits to XOR-CNF SAT

In this section, we present the proposed approach for checking the combinational equivalence between two reversible circuits. Without loss of generality, we consider reversible circuits composed only of Toffoli, CNOT, and NOT gates. Since Toffoli gates are universal, any other primitive reversible gate can be decomposed into a combination of those.

In the remainder of this section, we first describe how to create an *identity miter* out of two reversible circuits under test. Then, we propose an efficient encoding of the identity check problem into XOR-CNF satisfiability.

A. Creating an Identity Miter

In the considered scenario, two reversible circuits need to be checked for combinational equivalence. As an example, consider the circuits depicted in Fig. 3.

Following established verification schemes, both circuits are fed by the same input signals. Differences at the outputs are observed by applying XOR operations. This eventually leads to a new circuit specifically used for equivalence checking which is commonly called *miter circuit* [20]. If at least one output of the miter can evaluate to the logic value 1, for some input pattern, then the two circuits are functionally different. Otherwise, the two circuits are functionally equivalent.

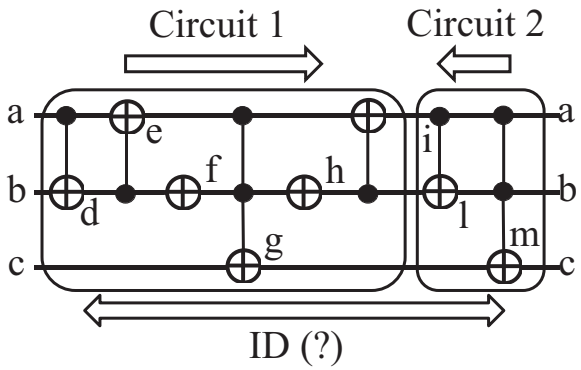

Fig. 4: The resulting identity miter.

The very same approach can be used to verify the combinational equivalence of two reversible circuits (and, in fact, has been done before in [9]). However, just an adaptation of this conventional scheme entirely ignores the potential that comes by following the reversible computing paradigm. In fact, properties of reversible circuits can be exploited to create a different type of miter. More precisely, a reversible circuit realizes a function f when considered from the inputs to the outputs. But thanks to the reversibility, it also realizes the inverse function f^{-1} when considered from the outputs to the inputs[2]. Therefore, by cascading one reversible circuit with the inverse (I/O flip) of a functional equivalent one always yields to a circuit realizing the identity function over all signal lines. This concept is illustrated in Fig. 4 which shows the resulting identity miter comprised from the example circuits of Fig. 3.

We call such composite circuit an *identity miter*. If at least one output of the identity miter does not represent the identity function, i.e., if $f(x) \neq x$, then the two reversible circuits are functionally different. Otherwise, the two circuits are functionally equivalent.

Note that the idea of creating an identity miter out of two reversible circuits is not new per se. Indeed, it has been already studied in [18]. However, in that work, the use of an identity miter did not lead to substantial improvements for equivalence checking of reversible circuits. This is because researchers used canonical data structures, decision diagrams and alike, to perform the identity checking task. The scaling limitations of canonical data structures severely confined the potential efficiency of using an identity miter.

Instead, in this work, we propose an innovative SAT formulation to describe the identity miter checking problem. SAT can handle much larger problems than canonical data structures before hitting serious scaling limitations. Moreover, we develop an ad-hoc mixed XOR-CNF formulation to natively handle the identity miter checking problem and significantly expedite its solving as compared to a pure CNF formulation.

B. XOR-CNF Formulation

To test the equivalence of two reversible circuits, we need to check whether their *identity miter* actually represents an identity function or not. If such an assignment can be determined, then the identity miter does not actually represent the identity function and the two reversible circuits under test are not functionally equivalent (in this case, the determined assignment works as counterexample). Otherwise, the identity miter represents the identity function and the two reversible circuits are functionally equivalent.

[2]This holds since self-inverse reversible gates such as Toffoli gates, CNOT gates, NOT gates, etc. are considered here. If other gate libraries are applied, similar observations can be made by additionally replacing each gate with its corresponding inverse gate.

Besides that, the XOR-richness of the considered circuits can be exploited. In fact, most of the reversible circuits are inherently composed of XOR operations only – caused by the applied Toffoli gate library as introduced in Section II-A. This allows for a formulation in terms of a mixed XOR-CNF satisfiability problem which, as reviewed in Section II-B, can be handled much better using dedicated solvers rather than the conventionally applied CNF satisfiability.

The resulting formulation is defined as follows: First, corresponding SAT variables are introduced. More precisely, for each primary input of the identity miter as well as for each reversible gate, a new free variable is introduced.

Example 2. *Consider again the identity miter as shown in Fig. 4. For the primary inputs, the variables a, b, c are introduced. The variables $d, e, \ldots, m$ represent reversible gates outputs.*

Afterwards, two types of constraints are introduced: The first type covers the functionality of the circuit, i.e., symbolically restricts the set of possible assignments to those which are valid with respect to the given gate functions and connections. The second type covers the objective, i.e., symbolically restricts the set of possible assignments to those which show, for at least one circuit line, the non-identity of the input x and the output $f(x)$ (in other words, assignments which violate $x = f(x)$).

Considering the functional constraints, there are as many functional constraints as Toffoli, CNOT, and NOT gates in the circuit. Each of them introduces its particular set of functional constraints which restrict the output value (denoted by o in the following) of the respective target lines. More precisely,

- a NOT gate with a target line t is represented by $(o = t')$,
- a CNOT gate with target line t and control line c is represented by $(o = c \oplus t)$, and
- a Toffoli gate with target line t and control lines $\{c_1, c_2, \ldots, c_n\}$ is represented by $(o = p \oplus t)$ and $(p = c_1 \cdot c_2 \cdot \ldots \cdot c_n)$.

All these constraints must simultaneously hold in order to properly represent the circuit functionality.

Example 3. *Consider again the identity miter shown in Fig. 4. For this circuit, the following functional constraints are created:*

$$\textit{Functionality} \begin{cases} d = b \oplus a \\ e = a \oplus d \\ f = d' \\ g = c \oplus p_1 \\ p_1 = f \cdot e \\ h = f' \\ i = e \oplus h \\ l = h \oplus i \\ m = g \oplus p_2 \\ p_2 = i \cdot l \end{cases} \tag{1}$$

As an example, consider the variable g which symbolically represents the output value of the fourth gate from Fig. 4. The functionality of this gate is represented by $g = c \oplus p_1$. The variable p_1 represents thereby the controlling part of this Toffoli gate and is accordingly represented as $p_1 = f \cdot e$. The remaining constraints in Eq. 1 are derived analogously.

Considering the objective constraints, there are as many objective constraints as lines in the reversible circuit. Here, the functional constraints as described above are utilized. For a generic $line_i$ ($i \leq m$), the primary outputs in the identity circuit are respectively defined by the cascade of gates $g_1 g_2 \ldots g_d$. The functional constraints represent these gates by means of a cascade of XOR operations so that $line_i$ is eventually defined as $line_i = h_1 \oplus h_2 \cdots \oplus h_d$ where each h_j ($j \leq d$) is either

- the product $p = c_1 \cdot c_2 \cdot \ldots \cdot c_n$ of the control connections of gate g_j (in case the corresponding gate g_j is a Toffoli gate),
- the control signal c (in case the corresponding gate g_j is a CNOT gate), or
- the logic value 1 (in case the corresponding gate g_j is a NOT gate).

Because of this cascade of XOR operations, the objective constraints only have to ensure that, for at least one $line_i$, its corresponding output assumes the logic value 1, i.e., behaves as an inverter rather than a buffer. This can be formulated as $\exists i \in \{1, 2, \ldots, m\} : line_i = 1$, where m is the number of lines.

Example 4. *Consider again the identity circuit considered above. For this circuit, the following objective constraints are added:*

$$\textit{Non-Identity} \begin{cases} \exists i \in \{1,2,3\} : line_i = 1 \\ line_1 = d \oplus h \\ line_2 = a \oplus 1 \oplus 1 \oplus i \\ line_3 = p_1 \oplus p_2 \end{cases} \quad (2)$$

As an example, consider the bottom (third) line of the reversible circuit from Fig. 4. We have that $line_3 = p_1 \oplus p_2$. The values of p_1 and p_2 are derived from the functional constraints, in particular from control lines of the respective Toffoli gates. The objective constraint asks for at least one of the three lines to evaluate to the logic value 1, thus to invert the corresponding input bit (so not being an identity).

As one can visually notice, the set of constraints in Eq. 1 and Eq. 2 are not yet in XOR-CNF form. Hence, some further transformations are needed. For this purpose, we exploit the fact that, in the Boolean domain, $(a = b)$ can equally be represented as $(a \oplus b' = 1)$. This allows us to transform most of the equalities directly into XOR clauses. In contrast, special treatment is required for the AND constraints caused by the representations of the control lines, i.e., for p. For these ones, it is more efficient to rely on the established Tseitin transformation [21]. Tseitin transformation sets a particular gate Boolean expression equal to constant 1 and transforms it into a conjunction of disjunctions. For this reason, Tseitin transformation encodes an AND function over k inputs into $k+1$ OR clauses. Finally, the constraint $\exists i \in \{1, 2, 3\} : line_i = 1$ is naturally mapped into a standard OR clause.

Example 5. *Following the example from above, all constraints from Eq. 1 and Eq. 2 are eventually transformed into the following single set of XOR-CNF clauses:*

$$\textit{XOR-CNF} \begin{cases} d' \oplus b \oplus a \\ e' \oplus a \oplus d \\ f' \oplus d' \\ g' \oplus c \oplus p_1 \\ p_1 + f' + e' \\ p_1' + f \\ p_1' + e \\ h \oplus f \\ i' \oplus e \oplus h \\ l' \oplus h \oplus i \\ m' \oplus g \oplus p_2 \\ p_2 + i' + l' \\ p_2' + i \\ p_2' + l \\ line_1' \oplus d \oplus h \\ line_2' \oplus a \oplus 1 \oplus 1 \oplus i \\ line_3' \oplus p_1 \oplus p_2 \\ line_1 + line_2 + line_3 \end{cases} \quad (3)$$

The resulting XOR-CNF problem is unsatisfiable as the considered identity miter shown in Fig. 4 indeed represents the identity. That means that the two original reversible circuits to be verified (shown in Fig. 3) are combinationally equivalent. This can be proved manually or, more efficiently, using a XOR-CNF satisfiability solver.

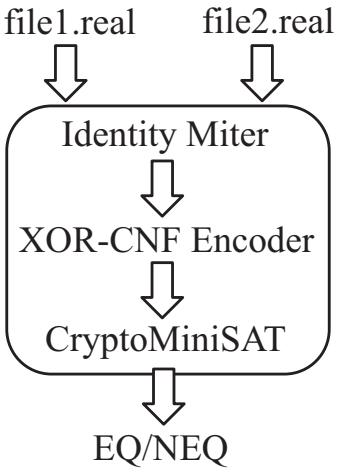

Fig. 5: The proposed equivalence checking flow.

Note that the XOR-CNF formulation in Eq. 3 is composed of 18 clauses and 16 variables. In contrast, the established formulation based on pure CNF requires 82 clauses and 34 variables [9]. This reduction alone is likely to lead to a solving speed-up. Moreover, the presence of more than 60% XOR clauses opens even more speed-up opportunities. Mixed XOR-CNF solvers take advantage of XOR clauses through fast Gaussian elimination. Results showed in the next section confirm the predicted improvement.

IV. Experimental Results

In order to evaluate the performance of the proposed approach, we implemented the techniques described above and compared them against the state-of-the-art solution presented in [9]. In this section, we summarize the respectively obtained results. Details on the applied methodology as well as the experimental setup are provided. Our experimental environment is publicly available and can be downloaded from [22].

A. Methodology and Setup

The proposed equivalence checking scheme has been implemented as a tool chain which is sketched by Fig. 5. Two reversible circuits (provided in the *.real-format [23]) are taken and re-arranged into an identity circuit as well as mapped into an equivalent XOR-CNF formulation. For this purpose, the concepts described in Section III have been implemented in terms of a C-program. Afterwards, the resulting formulation is passed to CryptoMiniSAT 2.0 – an XOR-CNF solver [11]. In case the solver proved the unsatisfiability of the instance, equivalence (EQ) has been proven; otherwise, it has been shown that the considered circuits are not equivalent (NEQ).

For comparison, we additionally considered the SAT-based reversible circuit checker presented in [9]. From a high-level perspective, this tool first creates an XOR-miter of the given reversible circuits. Then, it encodes the XOR-miter into a pure CNF formula which is eventually solved using MiniSAT [24]. Even though this flow has been explicitly tuned for verification of reversible circuits in [9], it still employs the state-of-the-art schemes as applied for verification of conventional circuits. To enable a fair runtime comparison, we

TABLE I: Experimental results (all run-times in CPU seconds)

		State-of-the-art [9]			Proposed solution			
Circuit1 (lines/gates)	Circuit2 (lines/gates)	Vars/Clauses	Answer	Runtime	Vars/Clauses	XOR%	Answer	Runtime
Unstructured Reversible Functions (from RevLib)								
urf3_1 (10/26k)	urf3_2 (10/26k)	133609/527485	EQ	98.85	104212/210085	32	EQ	14.20
urf3_1 (10/26k)	urf3_bug (10/26k)	133433/526926	NEQ	5.91	104212/210085	32	NEQ	1.69
urf1_1 (9/11k)	urf1_2 (9/6k)	58122/229437	EQ	17.89	35847/61885	60	EQ	2.54
urf1_1 (9/11k)	urf1_bug (9/6k)	58124/229390	NEQ	2.77	45438/91655	31	NEQ	0.52
urf5_1 (10/10k)	urf5_2 (10/10k)	51746/20401	EQ	15.85	40350/81455	31	EQ	3.75
urf5_1 (10/10k)	urf5_bug (10/9k)	51810/204249	NEQ	1.54	40312/81377	31	NEQ	0.42
urf6_1 (15/10k)	urf6_2 (15/10k)	54888/216888	EQ	5694.22	42565/85526	33	EQ	570.39
urf6_1 (15/10k)	urf6_bug (15/9k)	54682/216370	NEQ	2.64	42524/85445	33	NEQ	0.49
urf4_1 (11/32k)	urf4_2 (11/31k)	162247/636237	EQ	883.27	127254/255271	55	EQ	92.37
urf4_1 (11/32k)	urf4_bug (11/31k)	162349/636563	NEQ	6.04	127245/255252	55	NEQ	2.03
Total URF		921010/3443964	–	6728.98	709959/1418036	39	–	688.40
Components of the RISC CPU (from RevLib)								
alu1_1 (756/3k)	alu1_2 (756/10k)	82617/281684	EQ	5649.74	23128/99803	28	EQ	670.96
alu1_1 (756/3k)	alu1_bug (756/2k)	66182/216644	NEQ	67.84	10625/65921	21	NEQ	6.96
alu2_1 (6204/3k)	alu2_2 (6204/3k)	5568/20216	EQ	304.65	21254/22521	79	EQ	186.44
alu2_1 (6204/3k)	alu2_bug (6204/3k)	5657/20610	NEQ	369.49	21250/22517	80	NEQ	76.21
alu3_1 (255/10k)	alu3_2 (255/11k)	227505/752851	EQ	12751.02	35406/253957	12	EQ	728.98
alu3_1 (255/10k)	alu3_bug (155/8k)	209887/691117	NEQ	56.91	30424/232584	12	NEQ	9.90
alu4_1 (757/4k)	alu4_2 (757/7k)	28671/111480	EQ	8899.70	20941/40794	42	EQ	320.87
alu4_1 (757/4k)	alu4_bug (757/4k)	22140/85537	NEQ	825.71	16496/30987	55	NEQ	169.00
alu5_1 (256/9k)	alu5_2 (256/10k)	47290/185110	?	>1 day	33150/65249	45	EQ	6948.86
alu5_1 (256/9k)	alu5_bug (256/9k)	43966/171863	NEQ	51.56	30894/60329	51	NEQ	10.36
Total RISC CPU		739483/2537112	–	115376.62	243568/894662	42	–	9128.54
Grand Total		1660493/5981058	–	122105.60	953527/2312698	41	–	9816.94
Improvement compared to [9]		1/1	–	1	**1.74/2.58**×	–	–	**12.44**×

downloaded, compiled, and run the reference tool from [9] for our evaluations.

As benchmarks, we considered reversible circuits (provided in the *.real-format) from the RevLib benchmark library [23]. We neglected small reversible circuits for which the verification runtime was less than a second. We focused on complex reversible circuits (with >2k gates) for which the verification task required more computational effort. In particular, we give results for two classes of benchmarks: circuits realizing *Unstructured Reversible Functions* (URF) as well as circuits realizing arithmetic components of a RISC CPU. These classes are the largest and toughest benchmarks available at RevLib [23] and, hence, are appropriate to challenge the proposed verification scheme.

Whenever required, all gates in these circuits have been locally transformed into universal Toffoli gates. In order to consider both cases of equivalence as well as non-equivalence three versions of each circuit have been considered, namely (i) the original version, (ii) an optimized version, and (iii) an erroneous version.

All experiments have been conducted on a Dual Xeon 6 cores X5650 machine with 24GB RAM running under RHEL 5.8 - 64 bits OS. All benchmarks, the implementation, and the experimental setup can be downloaded from [22] for the sake of repeatability.

B. Results

Table I summarizes the experimental results. Considering the URF-benchmarks, equivalence checking can be conducted approx. 9 times faster compared to the reference verification scheme. If the CPU-benchmarks are considered, even better improvements can be observed; namely speed-ups of a factor of approx. 12. Here, particular the benchmark alu_5 is of interest. Applying the reference scheme proposed in [9], no result was obtained within 24 hours (its contribution to the total runtime nevertheless has been considered as 24 hours, i.e., 86400 in favor to the reference flow). In contrast, the proposed approach was able to check the equivalence in less than two hours. Over all benchmarks, an improvement of more than one order of magnitude (more precisely, a factor of 12.44) is observable.

We see the two reasons for this significant improvement: On the one hand, the number of variables and clauses are considerably smaller in the proposed XOR-CNF formulation compared to the pure CNF formulation (a reduction by the factor of 1.29 and 2.42, respectively). On the other hand, the richness of XOR-clauses in our formulation helps the solving engine in simplifying the formula early in the process (e.g., through Gaussian elimination). Further investigation is needed to numerically separate the contributions for each speedup source.

Besides that, non-equivalent cases have been solved quite faster than equivalent cases for both, the proposed scheme as well as the reference scheme. This is expected as SAT solvers are known to be very fast in detecting satisfying assignments rather than proving unsatisfiability.

V. Discussion

The proposed solution provides an alternative verification scheme for reversible logic which leads to significant improvements with respect to the state-of-the-art. Beyond that, it also opens promising new paths for improving verification of conventional designs. This section briefly discusses new research opportunities in this direction.

A. Application to the Verification of Conventional Circuits

The significant speed-up obtained in this work is enabled by intrinsic properties of reversible circuits such as bi-directional execution and XOR-richness. Conventional circuits usually do not inherit these

particular properties. Hence, at a first glance, the proposed verification scheme may seem applicable only to reversible computation paradigms.

However, conventional logic can also be represented in terms of reversible logic by using extra I/Os and extra gates. Previous studies explored this direction in order to (ideally) map any combinational design into reversible circuits [26]. This motivates us to consider a *new verification flow*. The core idea is to convert conventional circuits into reversible ones and perform the verification tasks in the reversible domain. In this way, the efficiency of the reversible equivalence checking flow proposed in this work can be further exploited. The conventional-to-reversible mapping may also be inefficient, from an optimization standpoint, but the benefits demonstrated so far are large enough to absorb such inefficiency and possibly leave room for a relevant improvement.

The main issue here is defining a robust and trustable conventional-to-reversible mapping technique. In this context, existing conventional-to-reversible mapping techniques [26] do not natively fit the requirements as they are intrinsically developed for logic optimization purposes. Our future research efforts are focused on the development of such reversible conversion method starting from arbitrary combinational logic circuits.

Provided that, conventional verification tasks will take full advantage of the reversible computing paradigm opening new exciting research directions.

B. Easy Exploitation of Parallelism

The proposed equivalence checking method can be further improved by exploiting concurrent execution. To speed-up SAT solvers, researchers are studying parallel and concurrent execution (e.g., [27]). This is motivated by the fact that, nowadays, multi-cores are widespread and computing resources are inexpensive. However, to fully exploit the potential offered by parallelization, also the respective SAT problems must be formalized in a parallel fashion. This is usually not obvious for the established equivalence checking solutions proposed in the past.

In contrast, a parallel consideration is simple for the solution proposed in this work. In fact, the formulation described in Section III can easily be split for each circuit line. By this, the overall equivalence checking problem is decomposed into m separate instances (with m being the number of circuit lines). These instances are smaller and can be solved independently from each other. As soon as one of the instances is found satisfiable, non-equivalence has been proven. Overall, this does not only allow for easier instances to be separately solved, but also enables the full exploitation of multiple-cores – something which is much harder to accomplish for almost all (conventional) verification schemes available thus far.

VI. Conclusions

Reversible circuits are of great interest to various fields, including cryptography, coding theory, communication, computer graphics, quantum computing, and many others. Checking the combinational equivalence of two reversible circuits is an important but difficult (co$\mathcal{NP}$-complete) problem. In this work, we presented a new approach for solving this problem significantly faster than the state-of-the-art. The proposed methodology explicitly exploited the inherent properties of reversible circuits, namely the bi-directional execution as well as their XOR-richness. This eventually enabled speed-ups of more than one order of magnitude on average. While this represents a substantial improvement for the verification of circuit descriptions aimed for reversible computation, it also offers promising potential to be exploited in the verification of conventional designs. Possible directions for future work in this regard have been outlined and briefly discussed.

Acknowledgements

This research was supported by the ERC-2009-AdG-246810 and by the EU COST Action IC1405.

References

[1] D. Kamalika, I. Sengupta. *Applications of Reversible Logic in Cryptography and Coding Theory*, Proc. 26th Intl. Conf. on VLSI Design. 2013.

[2] K. Czarnecki, *et al.*, *Bidirectional transformations: A cross-discipline perspective*, Theory and Practice of Model Transformations. Springer Berlin Heidelberg, 2009. 260-283.

[3] R. Wille, R. Drechsler, C. Oswald, A. Garcia-Ortiz, *Automatic Design of Low-Power Encoders Using Reversible Circuit Synthesis* In Design, Automation and Test in Europe (DATE), pg: 1036-1041, 2012.

[4] S. Lee, Sunil, C.Dong Yoo, T. Kalker, *Reversible image watermarking based on integer-to-integer wavelet transform*, IEEE Transactions on Information Forensics and Security, 2.3 (2007): 321-330.

[5] M. Nielsen, I. L. Chuang, *Quantum computation and quantum information*, Cambridge university press, 2010.

[6] R. Wille, R. Drechsler, *Towards a Design Flow for Reversible Logic*, Springer, 2010.

[7] R. Drechsler, R. Wille, *From Truth Tables to Programming Languages: Progress in the Design of Reversible Circuits*, IEEE International Symposium on Multiple-Valued Logic, pages 78-85, 2011.

[8] M. Saeedi, I. L. Markov, *Synthesis and optimization of reversible circuits – a survey*, ACM Computing Surveys (CSUR) 45.2 (2013): 21.

[9] R. Wille, D. Große, D.M. Miller, Rolf Drechsler, *Equivalence checking of reversible circuits*, IEEE International Symposium on Multiple-Valued Logic, 2009.

[10] S.P. Jordan, *Strong equivalence of reversible circuits is coNP-complete*, Quantum Information & Computation 14.15-16 (2014): 1302-1307.

[11] CryptoMiniSAT tool - http://www.msoos.org/cryptominisat2/

[12] M. Soos, *Enhanced Gaussian Elimination in DPLL-based SAT Solvers* POS@ SAT. 2010.

[13] T. Toffoli, *Reversible computing,* in Automata, Languages and Programming, W. de Bakker and J. van Leeuwen, Eds. Springer, 1980, p. 632, technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.

[14] E. F. Fredkin, T. Toffoli, *Conservative logic*, International Journal of Theoretical Physics, vol. 21, no. 3/4, pp. 219253, 1982.

[15] A. Peres, *Reversible logic and quantum computers*, Phys. Rev. A, no. 32, pp. 32663276, 1985.

[16] A. Biere, M. Heule, Hans van Maaren, eds. *Handbook of satisfiability*, Vol. 185. ios press, 2009.

[17] M, Davis, G. Logemann, D. Loveland, *A Machine Program for Theorem Proving*, in Communications of the ACM, vol. 5, n. 7, 1962, pp. 394397.

[18] S. Yamashita, I. L. Markov. *Fast equivalence-checking for quantum circuits*, Proceedings of the 2010 IEEE/ACM International Symposium on Nanoscale Architectures. IEEE Press, 2010.

[19] M. Soos, K. Nohl, C. Castelluccia, *Extending SAT solvers to cryptographic problems*, Theory and Applications of Satisfiability Testing-SAT 2009. Springer Berlin Heidelberg, 2009. 244-257.

[20] D. Brand. *Verification of large synthesized designs*, Proc. ICCAD 93, pp. 534 -537.

[21] G. S. Tseitin, *On the complexity of derivation in propositional calculus*, Automation of reasoning. Springer Berlin Heidelberg, 1983. 466-483.

[22] Reversible CEC Package Experiments – available for download at: http://lsi.epfl.ch/RCEC.

[23] R. Wille, D. Große, L. Teuber, G.W. Dueck, R. Drechsler, *RevLib: An Online Resource for Reversible Functions and Reversible Circuits*, International Symposium on Multiple-Valued Logic, pages 220-225, 2008. RevLib is available at http://www.revlib.org.

[24] MiniSat: open-source SAT solver http://minisat.se

[25] D.M. Miller, R. Wille, G.W. Dueck, *Synthesizing Reversible Circuits for Irreversible Functions*, In Euromicro Conference on Digital System Design (DSD), pages 749-756, 2009.

[26] R. Wille, O. Keszocze, R. Drechsler. *Determining the Minimal Number of Lines for Large Reversible Circuits.* In Design, Automation and Test in Europe (DATE), 2011.

[27] Y. Hamadi, J. Said, S. Lakhdar et al., *ManySAT: a parallel SAT solver*, Journal on Satisfiability, Boolean Modeling and Computation 6 (2008): 245-262.