Exact Template Matching Using Boolean Satisfiability

Nabila Abdessaied* Mathias Soeken*§ Robert Wille*§ Rolf Drechsler*§
*Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
§Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{nabila,msoeken,rwille,drechsle}@informatik.uni-bremen.de

***Abstract*—Reversible logic is an emerging research area that has shown promising results in applications such as quantum computing, low power design, and optical computing. Since the synthesis of minimal circuits is a cumbersome task, many synthesis algorithms apply heuristics and can therefore not provide a minimal solution. As a consequence, post synthesis methods such as window optimization and template matching are being applied.**

Template matching algorithms explore the circuits for gate cascades that can be replaced by smaller ones using a special class of identity circuits, so called templates. The determination of cascades applicable for substitution is the bottleneck of the template matching algorithm and problem-solving methods have been proposed in the recent past. Since these algorithms are based on heuristics, it cannot be ensured that a matching cascade can always be found.

In this paper, we propose a new approach that determines matching cascades based on Boolean satisfiability and therefore ensures that these cascades are always found if they exist. Experimental results demonstrate that template matching yields smaller circuits when applying the new method for cascade determination.

I. Introduction

Reversible logic is an emerging technology in the domain of quantum computing [1], [2], [3], low power design [4], [5], optical comuting [6], as well as nanotechnologies [7]. In reversible circuits, all computations are performed in an inverted manner which provides new promising enhancements for computation technologies. Motivated by the benefits brought by these circuits, researchers started to develop new design methods.

Several synthesis approaches have been proposed to determine a circuit realization for a given function. Significant improvements and achievements [8], [9], [10], [11] have been made since the truth table based approach has taking place. However, the majority of the synthesis approaches do not guarantee optimal realizations, in fact, the algorithms that do guarantee an optimal solution (e.g. [12]) are only applicable to small circuits of about 4 to 6 lines.

As a result, several post synthesis optimization approaches [9], [13] have been proposed to minimize a given circuit after it has been synthesized. The template matching algorithm explained in [9] is one of these optimization techniques. Given a set of templates which is a special class of identity circuits, the algorithm tries to determine sub-circuits that match a part of a template. In this case, the determined sub-circuit can be replaced with the inverted remaining part of the template due to reversibility. If the remaining part is smaller, the overall circuit size can be reduced.

The efficiency of the template matching algorithm highly depends on the strategy used for matching the template in a circuit. Since the considered search space that should be inspected in order to find a match for a template is usually very large, many heuristic approaches have been investigated that work efficiently but cannot guarantee that a matching sub-circuit can be found if it exists.

To overcome this limitation, we propose a new approach that exploits Boolean satisfiability techniques allowing an exhaustive but yet efficient determination of cascades according to a given set of templates. For this purpose, the search for a cascade is formulated as a decision problem and encoded as a Boolean formula that is afterwards solved using an SMT solver. If the instance is satisfiable, the matching cascade can be replaced by the second half of the template. Otherwise, it can be concluded that the template cannot be used to further reduce of the circuit cost.

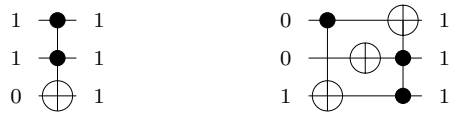
We have implemented the algorithm and compared it to the search method presented in [9]. Experimental results show that cost reductions of up to about 60% compared with respect to the initial circuit and up to 28% with respect to the method in [9] can be achieved.

The remainder of the paper is organized as follows. Section II introduces the basics of reversible logic, the template matching algorithm, and SMT. In Section III the general structure of our approach is explained, while details on the SMT encoding are provided in Section IV. Experimental results are given in Section V and the paper is concluded in Section VI.

II. Background

A. Reversible Circuits

A Boolean function is reversible if it maps each input assignment to a distinct output assignment. As a result, such a function must have the same number of input and output variables $X := \{x_1, \ldots, x_n\}$. A circuit realizing a reversible function is a cascade of reversible gates. In the literature, several types of reversible gates have been introduced. Besides the Fredkin gate [14] and the Peres gate [15], multiple controlled



(a) Reversible gate (b) Reversible circuit

Fig. 1. Reversible circuitry

Toffoli gates [16] are widely used. In this paper we consider only Toffoli gate circuits.

A Toffoli gate has the form $\mathrm{T}(C,t)$ with a set of control lines $C = \{x_{i_1}, \cdots, x_{i_k}\} \subset X$ and a target line $t \in X \setminus C$. C may be empty. The target line t is inverted if and only if all control lines are assigned 1, i.e. a gate maps an input assignment $(x_1, \ldots, t = x_j, \ldots, x_n)$ to $(x_1, \ldots, x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k} \oplus x_j, \ldots, x_n)$.

Quantum cost are often used to measure the cost of a reversible gate. Every reversible gate can be transformed into a sequence of elementary quantum gates [17] where each elementary gate has a quantum cost of one.

Example 1: Fig. 1(a) shows a Toffoli gate with two control lines. The control lines are denoted by ●, while the target line is denoted by $\bigoplus$. The annotated values demonstrate the computation of the gate for a given input assignment. Fig.1(b) shows different Toffoli gates in a cascade forming a reversible circuit.

B. Template Matching Procedure

Templates have initially been proposed in [9] and their definition has been adjusted several times in the past. The current widely accepted definition states that a template is an identity circuit with m gates such that each sub-circuit of size less than $\lceil \frac{m}{2} \rceil$ cannot be reduced by another template. Furthermore, a template consists of two different line types which we call C-lines and T-lines and which should not be mistaken for control lines and target lines of a gate (cf. Fig. 2). A C-line of a template is a line in which all gates only have control lines but no target lines, all other lines are T-lines. This separation is of use since any C-line can be duplicated, removed, or replaced without changing the functionality of the circuit. This is illustrated by means of Figs. 3(a), 3(b), and 3(c), respectively. Further, the order of the gates can be rotated being wrapped at the circuit boundaries, again without changing the functionality. This is illustrated in Fig. 3(d).

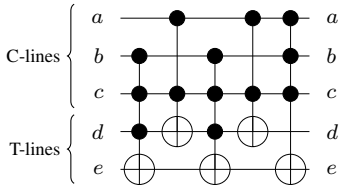


Fig. 2. Original template

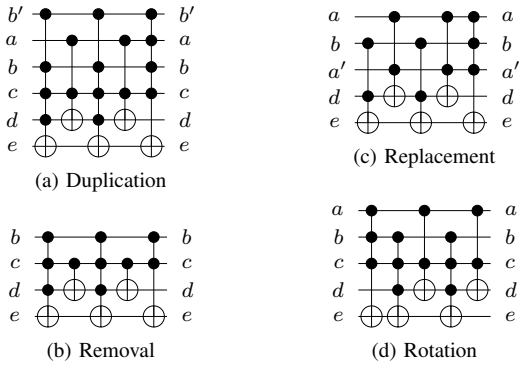


Fig. 3. Template disposition

Since a template is an identity circuit it can be split anywhere in the middle and the left part equals the inverse of the right part and vice versa. The template matching algorithm is applied in order to reduce the circuit size or its costs. It takes a circuit and tries to find sub-circuits in it that match a part of a template. If that part is longer or more expensive than the remaining part the sub-circuit is replaced by the inverse of it. The matching procedure tries to find the first gate and then subsequently looks for the other gates which do not need to be necessarily adjacent. Using the *moving rule*, gates can be moved in the circuit without changing the circuit's functionality [9].

Example 2: Fig. 4 illustrates the basic template matching algorithm. It shows a template in the lower left corner with a left and a right part. A sub-circuit matching the left part of the template is found in the original circuit, hence, this sub-circuit can be replaced with the inverse of the right part of the template yielding a smaller simplified circuit of less cost.

The degree of optimization in template matching depends on the considered templates that are used in the process and the search method for finding matching sub-circuits. The latter task is the most complex part in the template matching algorithm and not feasible when being applied in an exhaustive manner. In order to determine matching templates, the circuit is rearranged by applying the time consuming moving rule. As a consequence, heuristics are used to find sub-sequences

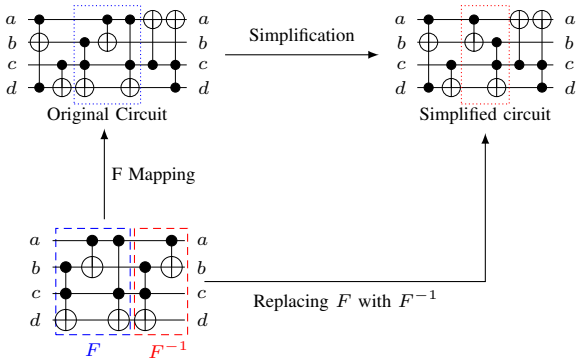


Fig. 4. Template matching

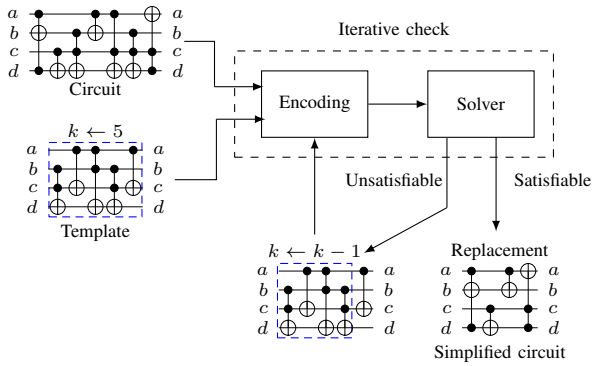


Fig. 5. Encoding the search method as an SMT instance

that do not guarantee that a sub-circuit is found if it exists. For example, some of them take a circuit and try to match its gates to the template from the input to the output side of a that circuit, other ones apply the search additionally in the reverse direction in order to increase the chances of matching the template in the circuit.

C. Satisfiability Modulo Theories

Boolean satisfiability (SAT) is a decision problem that determines if the variables of a given propositional formula can be assigned in such a way that the formula evaluates to true. Equally important is to determine whether no such assignments exist, which would imply that the formula is the contradiction. In the latter case, the function is said to be unsatisfiable, otherwise it is satisfiable. Most modern SAT solvers require the formula to be in conjunctive normal form, i.e. a conjunction of clauses which in turn are disjunctions of literals.

Satisfiability Modulo Theories (SMT) is also a decision problem but with more complex theories rather than only propositional logic. A detailed introduction is given in [18].

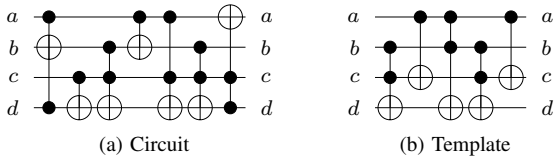
SMT is is about checking the satisfiability of first-order formulas containing operations from various theories such as the Booleans, bit-vectors, arithmetic, arrays, and recursive data-types.

SMT solvers are available that handle complex formulas such as Z3 [19], MathSAT [20], Boolector [21], and SWORD [22].

III. General Idea

In this work, we are proposing a new search method that determines matching sub-circuits in a circuit given a template while keeping the general concepts for template matching as they have been explained in [9]. Instead of applying heuristics for the search, we suggest an exact and efficient template matching algorithm based on SMT which allows to exhaustively explore the full search space and therefore guarantees that a matching sub-circuit is found if it exists.

Fig. 5 outlines the proposed approach. Given a template with m gates and the original circuit, the proposed algorithm creates a Boolean formula encoding the decision problem



(a) Circuit (b) Template

Fig. 6. Reversible circuit and template

whether there exists a sub-circuit in the original circuit that matches the first k gates of the given template. While initially setting $k \leftarrow m$, k is decremented by 1 as long as the Boolean formula is unsatisfiable. If the formula is satisfiable the matching sub-circuit can be extracted from the satisfying assignment that is returned from the SMT solver. The template length k is only decreased as long as the left part of the template is larger in cost than the right one.

IV. Encoding Using SMT

This section describes the proposed approach in detail. The encoding as a decision problem is explained by listing all constraints that are necessary for searching a correct match of a given template in a circuit.

A. Decision Problem

The overall decision problem can be stated as follows. Let $G = g_1 \cdots g_d$ be a circuit of n lines and $T = g'_1 \cdots g'_{d'}$ be a template of n' lines with $g_j = (C_j, t_j)$ and $g'_i = (C_i, t_i)$ for $j = 1, \ldots, d$ and $i = 1, \ldots, d'$, respectively. The considered lines are defined over the variables $\{x_1, \ldots, x_n\}$ and $\{x'_1, \ldots, x'_{n'}\}$, respectively. Can the first $k \leq d'$ gates of the template T be matched in G, i.e. can positions $m_1, \ldots, m_k$ in G be found such that the gates can be moved together and resemble the first k gates of T. Notice that also the order of lines in the template does not necessarily need to match the original order of lines in the circuit. As a result, besides the matching positions $m_1, \ldots, m_k$ also a line reordering $l_1, \ldots, l_{n'}$ is part of the solution, if the template can be matched.

Example 3: Figs. 6(a) and 6(b) show a circuit and a template, respectively. The first $k = 4$ gates of the template can be matched to the gates in the circuit at positions $m_1 = 3$, $m_2 = 4$, $m_3 = 5$, and $m_4 = 6$ when a line mapping $l_1 = 1$, $l_2 = 3$, $l_3 = 2$, and $l_4 = 4$, i.e. $x_1 \mapsto x'_1$, $x_2 \mapsto x'_3$, $x_3 \mapsto x'_2$, and $x_4 \mapsto x'_4$.

B. Gate Positions and Line Mapping

The variables $m_1, \ldots, m_k$ and $l_1, \ldots, l_{n'}$ represent gate and line indexes in the intervals $[1, d]$ and $[1, n']$, respectively. A one-hot encoding is used for all these variables, i.e.

$$\vec{m}_i \in \mathbb{B}^d \quad \text{for} \quad i = 1, \ldots, k$$

and

$$\vec{l}_j \in \mathbb{B}^{n'} \quad \text{for} \quad j = 1, \ldots, n'$$

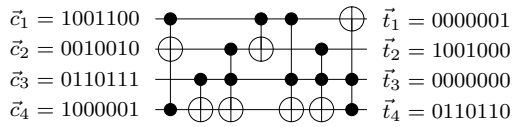


Fig. 7. Circuit encoding

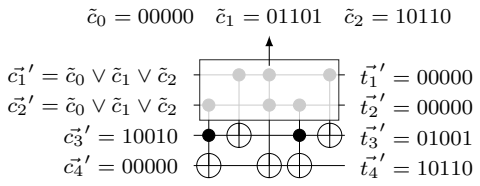


Fig. 8. Template encoding

such that $\nu\vec{m}_1 = \cdots = \nu\vec{m}_k = \nu\vec{l_j} = \cdots = \nu\vec{l}_{n'} = 1$, where the *sideways sum* ν denotes the number of all 1 bits in a bit-vector. The single bit that is set in the vectors denotes the respective index.

Two constraints are sufficient to enforce both the ordering of the positions m_i and guarantee the one-hot encoding, i.e.

$$0 < \vec{m}_i < \vec{m}_{i+1} \quad \text{for} \quad i = 1, \ldots, k-1$$

and

$$\bigwedge_{i \neq j} (\vec{m}_i \wedge \vec{m}_j = 0) \wedge \nu\vec{M} = k$$

with $\vec{M} = \bigvee_{i=1}^{k} \vec{m}_i$ being the bit-mask containing all position indexes. The line mapping variables do not have to follow an order, however all bit-vectors need to be one-hot encoded and they must all differ, i.e.

$$\bigwedge_{i \neq j} (\vec{l}_i \wedge \vec{l}_j = 0) \wedge \bigwedge_{j=1}^{n'} \vec{l}_j \neq 0 \wedge \bigwedge_{j=1}^{n'} \vec{l}_j = \underbrace{1 \ldots 1}_{n' \text{ times}}.$$

C. Circuits and Templates

In order to represent circuits, we are making use of two different encodings which are both used later for constraining the mapping. They both share the property that control lines and targets are separately represented as bit-mask, however once in vertical and once in horizontal orientation. For the circuit's lines, we are following the horizontal scheme, i.e. for each control line and for each target line we add bit-vectors

$$\vec{c}_i \in \mathbb{B}^d \quad \text{with} \quad \vec{c}_i[j] \Leftrightarrow x_i \in c_j$$

and

$$\vec{t}_i \in \mathbb{B}^d \quad \text{with} \quad \vec{t}_i[j] \Leftrightarrow x_i = t_j$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, d$. Note that the bit-vector indices start from 1.

Example 4: The encoded bit-vectors for the circuit lines of the initial circuit from Fig. 6(a) are illustrated in Fig. 7.

For the templates we are using a slightly different encoding that captures the different modification possibilities that exists for templates and have been illustrated in Fig. 3. Further, for the encoding we assume that the template can be extended to at most n lines such that it fits the circuit. Also, let us assume that the template has τ T-lines. Then the first $n - \tau$ circuit lines for the template are defined as

$$\vec{c_i}' \in \mathbb{B}^k \quad \text{with} \quad \vec{c_i}' = \bigvee_{j=0}^{n'-\tau} \tilde{c}_j \qquad (i \leq n - \tau)$$

and

$$\vec{t_i}' \in \mathbb{B}^k \quad \text{with} \quad \vec{t_i}' = 0 \ldots 0 \qquad (i \leq n - \tau)$$

with

$$\tilde{c}_j \in \mathbb{B}^k \quad \text{with} \begin{cases} 0...0 & \text{if } j = 0, \\ \tilde{c}_j[\ell] \Leftrightarrow x'_j \in c_\ell & \text{otherwise.} \end{cases}$$

That is, there are no targets on these lines and a control line can be either one of all possible C-lines (encoded as $\tilde{c}_{>0}$) or also $0 \ldots 0$ (encoded as $\tilde{c}_0$) indicating that the line is not used by the template. The lines for $i > n - \tau$ are encoded in the exact same manner as for the original circuit lines.

Example 5: The encoded bit-vectors for the circuit lines of the template from Fig. 6(b) are illustrated in Fig. 8. Notice that the order of the C-lines as it is given in Fig. 6(b) does not matter anymore and is encapsulated in the $\tilde{c}_j$ variables.

From the bit-vectors $\vec{c_i}'$ and $\vec{t_i}'$ we are deriving new bit-masks $\check{c_j}'$ and $\check{t_j}'$ that describe the circuit in a vertical orientation, i.e. each bit-vector corresponds to a gate instead of a line. These are bit-vectors

$$\check{c_j} \in \mathbb{B}^n \quad \text{with} \quad \check{c_j}'[i] = \vec{c_i}'[j]$$

and

$$\check{t_j} \in \mathbb{B}^n \quad \text{with} \quad \check{t_j}'[i] = \vec{t_i}'[j].$$

D. Mapping

Given all bit-vector encodings from above, we can define the constraints that map template gates to circuit gates with respect to a line mapping. For this purpose, we make use of the function $@ : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}$ with

$$@ : (\vec{a}, \vec{b}) \mapsto \vec{a} \wedge \vec{b} \neq 0$$

where $\vec{a}, \vec{b} \in \mathbb{B}^n$. We are using the function only in cases where $\vec{b}$ is one-hot encoded, hence, the function evaluates to true if and only if the one bit that is set in $\vec{b}$ is also set in $\vec{a}$. Given that, we can formalize the most important mapping for the encoding which maps template gates to circuit gates, expressed as

$$\check{c_j}' @ \vec{l_i} = \vec{c_i} @ \vec{m}_j \quad \text{and} \quad \check{t_j}' @ \vec{l_i} = \vec{c_i} @ \vec{m}_j$$

with $i = 1, \ldots, n$ and $j = 1, \ldots, k$. Taking the control lines, that is the formula on the left hand side, it means the following. Assuming there is a control line in the j^{th} gate of the template at the line where line i maps to. Then, there must also be a control line in the j^{th} gate chosen by the mapping $\vec{m}_j$ in the original circuit at line i and vice versa. The same applies for target lines. Notice that the fact that we have vertical and horizontal encodings for template and circuit gates plays a key role in this encoding.

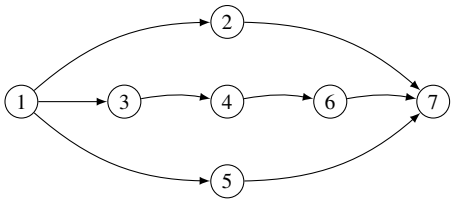


Fig. 9. Moving rule graph for circuit in Fig. 6(a)

E. Moving Rule

The encodings given so far are not sufficient yet, since at the current state arbitrary gates in the original circuits can be matched although it might not be possible to move them together. As a consequence, the moving rule needs to be encoded into the SMT instance as well.

In order to have a formal representation of the moving rule, we are making use of the moving rule graph that has been introduced in [23]. In a moving rule graph each vertex represents a gate of the circuit and two gates are connected by an edge if and only if the gates cannot be interchanged by moving.

Example 6: The moving rule graph for the circuit in Fig. 6(a) is depicted in Fig. 9.

We write $g_i < g_j$ if g_j cannot be moved before g_i and gate g_i cannot be moved past g_j. Since the moving rule is transitive, $g_i < g_j$ if and only if there exists a path between the vertex representing g_i and the vertex representing g_j in the moving rule graph.

This leads to the final constraint which ensures the moving capabilities. For all $0 < i < j < k \leq d$ such that $g_i < \cdots < g_j < g_k$ and there is no j' such that $g_i < g_{j'} < g_k$, add the constraint

$$\vec{M}[i] \wedge \vec{M}[k] \Rightarrow \vec{M}[j].$$

These constraints can easily be generated with the help of the moving rule graph.

Example 7: Given the moving rule graph in Fig. 9, the extracted constraints are as follows:

$$\begin{array}{ll} \vec{M}[1] \wedge \vec{M}[7] \Rightarrow \vec{M}[2] & \vec{M}[1] \wedge \vec{M}[7] \Rightarrow \vec{M}[5] \\ \vec{M}[1] \wedge \vec{M}[7] \Rightarrow \vec{M}[6] & \vec{M}[1] \wedge \vec{M}[6] \Rightarrow \vec{M}[4] \\ \vec{M}[1] \wedge \vec{M}[4] \Rightarrow \vec{M}[3] & \vec{M}[3] \wedge \vec{M}[7] \Rightarrow \vec{M}[6] \\ \vec{M}[3] \wedge \vec{M}[6] \Rightarrow \vec{M}[4] & \vec{M}[4] \wedge \vec{M}[7] \Rightarrow \vec{M}[6] \end{array}$$

V. Experimental Results

The proposed approach has been implemented in C++. In order to read the reversible circuits as well as the templates, we used the open source toolkit for reversible circuit design *RevKit* [24]. The SMT problem, i.e. the template matching problem is encoded using the *metaSMT* [25] framework which provides the use of SAT and SMT solvers directly over its API through the language. Different solvers can be used within the metaSMT framework, for our experiments the *Boolector* [21] SMT solver turns out to be the most efficient one. The approach has been evaluated on a Dual-Core AMD Processor with 4 GB of main memory. We used circuits provided in *RevLib* [26] as benchmarks. We have compared our approach to the one proposed in [9] and used the same set of templates that is presented in the work.

Table I summarizes the obtained results for the conducted experiments. The first three columns give the name of the initial not optimized circuits as well as its number of gates and quantum cost (QC). In the following columns, the obtained results for the heuristic template matching approach and the proposed approach are presented. For each the resulting number of gates, the quantum cost, the run-time, and the improvement compared to the initial circuit is given (Impr/IC). For the proposed approach additionally the improvement compared to the heuristic approach is listed (Impr/HTM).

Considering quantum cost, for most of the circuits significant cost reduction can be seen when applying the new approach. Applying the heuristic approach reduces the quantum costs by around 29.14% when considering all circuits. It is clearly observed that these results can be improved when applying the approach based on Boolean satisfiability. The proposed approach leads to an additional cost reductions of 11.42% on average and 27.46% in the best case. The results clearly confirm the impact of the exhaustive search to the circuit costs. The effect is in particular observable for large circuits.

Heuristic template matching is already a time consuming process. However, as it is shown, the new approach needs an enormous computation time compared to the heuristic method. This is expected due to the fact that the match is determined exhaustively by the SAT solver, which needs higher run-time to provide the corresponding answer.

VI. Conclusion

In this paper, a new search method for applying templates in the template matching algorithm has been proposed. Instead of using heuristics for matching gates, the problem is encoded into an instance of Boolean satisfiability and therefore ensures an exhaustive examination of the search space. The proposed approach leads to improvements in terms of circuit costs compared to the heuristic template matching approach. Experimental results clearly confirm these improvements.

References

[1] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[2] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Foundations of Computer Science*, pp. 124–134, 1994.

[3] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," *Nature*, vol. 414, p. 883, 2001.

[4] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, vol. 5, p. 183, 1961.

[5] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev*, vol. 17, no. 6, pp. 525–532, 1973.

[6] R. Cuykendall and D. R. Andersen, "Reversible optical computing circuits," *Optics Letters*, vol. 12, no. 7, pp. 542–544, 1987.

[7] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology*, vol. 4, pp. 21–40, 1993.

TABLE I
EXPERIMENTAL RESULTS

			Heuristic Template Matching (HTM, [9])				SMT Based Template Matching				
Bench	Gates	QC	Gates	QC	Time	Impr/IC	Gates	QC	Time	Impr/IC	Impr/HTM
3_17	6	14	6	14	0.01	0.00	6	14	0.7	0.00	0.00
4gt5_21	3	19	3	19	0.01	0.00	4	16	0.08	15.79	15.79
mod5mils_18	9	21	7	11	0.02	47.62	6	10	0.8	52.38	9.09
4gt10_22	3	47	4	44	0	6.38	4	36	0.17	23.40	18.18
aj-e11_81	22	118	22	102	0.15	13.56	24	96	13.98	18.64	5.88
hwb4_12	24	144	24	92	0.14	36.11	24	92	9.55	36.11	0.00
ex2_151	15	219	21	145	0.11	33.79	18	131	12.95	40.18	9.66
4_49_7	32	220	31	187	0.67	15.00	39	151	336.47	31.36	19.25
1-2-3_27	25	285	33	221	0.23	22.46	34	218	140.58	23.51	1.36
hwb5_132	39	339	40	320	0.79	5.60	51	295	97.88	12.98	7.81
mini-alu_84	35	483	36	392	0.29	18.84	48	348	255.12	27.95	11.22
wrd53_683	41	548	50	282	2.22	48.54	39	284	577	48.18	-0.71
hwb5_131	36	576	43	475	1.52	17.53	51	419	1371.03	27.26	11.79
C17_1172	38	654	45	509	0.99	22.17	63	419	5465.24	35.93	17.68
sym6_63	36	777	51	461	0.8	40.67	30	337	240.68	56.63	26.90
hwb5_13	75	915	83	803	10.32	12.24	87	714	6704	21.97	11.08
sym9_714	51	949	73	478	1.14	49.63	49	392	3021.39	58.69	17.99
hwb6_142	50	1002	56	992	0.66	1.00	64	896	459.54	10.58	9.68
C17_1171	31	1003	35	735	0.28	26.72	52	708	81.97	29.41	3.67
C17_117	69	1657	80	1244	1.87	24.92	112	1128	3963.94	31.93	9.32
hwb6_141	50	1826	54	1414	2.41	22.56	81	1261	2109	30.94	10.82
cm82a_1263	53	2011	58	1718	1.29	14.57	72	1360	991.97	32.37	20.84
max46_1772	45	2556	61	1224	1.15	52.11	51	1153	583.12	54.89	5.80
cm82a_1262	53	2565	61	2349	2.06	8.42	83	1704	11403.1	33.57	27.46
cm82a_1264	53	2643	57	2464	1.32	6.77	72	1897	1536.02	28.23	23.01
cm82a_1261	53	3336	71	2615	1.24	21.61	78	2162	1170.06	35.19	17.32
hwb6_14	153	3465	169	2925	3.7	15.58	172	2620	4769	24.39	10.43
m152a_1302	43	5574	43	5454	2.65	2.15	43	5452	2927.56	2.19	0.04
cm152a_130	88	8754	99	7159	7.77	18.22	106	7154	4604	18.28	0.07
cm82a_126	313	12811	356	11224	30.91	12.39	443	8847	10639	30.94	21.18

[8] K. Fazel, M. A. Thornton, and J. E. Rice, "ESOP-based Toffoli gate cascade generation," in *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2007, pp. 206–209.

[9] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Design Automation Conf.*, 2003, pp. 318–323.

[10] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conf.*, 2009, pp. 270–275.

[11] M. Soeken, R. Wille, C. Otterstedt, and R. Drechsler, "A synthesis flow for sequential reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2012, pp. 299–304.

[12] R. Wille and D. Große, "Fast exact Toffoli network synthesis of reversible logic," in *Int'l Conf. on CAD*, 2007, pp. 60–64.

[13] M. Soeken, R. Wille, G. W. Dueck, and R. Drechsler, "Window optimization of reversible and quantum circuits," in *IEEE Symp. on Design and Diagnostics of Electronic Circuits and Systems*, 2010, pp. 341–345.

[14] E. F. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3/4, pp. 219–253, 1982.

[15] A. Peres, "Reversible logic and quantum computers," *Phys. Rev. A*, no. 32, pp. 3266–3276, 1985.

[16] 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.

[17] A. Barenco, C. H. Bennett, R. Cleve, D. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *The American Physical Society*, vol. 52, pp. 3457–3467, 1995.

[18] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Satisfiability*, 2009, pp. 825–885.

[19] L. M. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2008, pp. 337–340.

[20] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, and R. Sebastiani, "The MathSAT 4 SMT solver," in *Computer Aided Verification*, 2008, pp. 299–303.

[21] R. Brummayer and A. Biere, "Boolector: An efficient SMT solver for bit-vectors and arrays," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2009, pp. 174–177.

[22] R. Wille, G. Fey, D. Große, S. Eggersglüß, and R. Drechsler, "SWORD: A SAT like prover using word level information," in *VLSI of System-on-Chip*, 2007, pp. 88–93.

[23] N. Scott, G. Dueck, and D. Maslov, "Improving template matching for minimizing reversible toffoli cascades," in *Int'l Reed-Muller Workshop*, 2005.

[24] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: An Open Source Toolkit for the Design of Reversible Circuits," in *Reversible Computation 2011*, ser. Lecture Notes in Computer Science, vol. 7165, 2012, pp. 64–76, RevKit is available at www.revkit.org.

[25] F. Haedicke, S. Frehse, G. Fey, D. Große, and R. Drechsler, "metaSMT: Focus on Your Application not on Solver Integration," in *Int'l Workshop on Design and Implementation of Formal Tools and Systems*, 2011.

[26] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: an online resource for reversible functions and reversible circuits," in *Int'l Symp. on Multi-Valued Logic*, 2008, pp. 220–225, RevLib is available at http://www.revlib.org.