INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 3, PP. 727–732
Manuscript received May 19, 2024; revised JUly, 2024. doi: 10.24425/ijet.2024.149602

Complete synthesis of identity templates for quantum and reversible logic MCT circuits using SAT-solvers and proposal of suboptimality witness notion

Adam Jagielski

***Abstract*—In this study, we introduce a procedural generation technique for Identity Templates applicable to quantum and reversible logic circuits. These templates are recognized for their significant role in enhancing the efficiency of quantum and reversible logic optimization. Our approach enables the exhaustive synthesis of all potential templates up to a specified size. Leveraging the power of SAT-solver technology, we have verified the comprehensiveness of our template collections by confirming the full exploration of the search space. Additionally, we propose an innovative concept of Suboptimality Witnesses, which we anticipate will be instrumental in streamlining the search process in formal methods, akin to SAT-solvers, for the synthesis of reversible logic circuits.**

***Keywords*—quantum computing; circuit synthesis; cryptography; satisfiability problem; reversible logic**

I. Introduction

REVERISBLE logic is a fundamental component in various fields of applied mathematics and computer science. With the rapid advancement of quantum computing and IoT (Internet of Things) technologies, we are witnessing a significant shift in the perceived value of this branch of low-level logic. Consequently, there is a growing demand for the production of optimized reversible logic circuits.

Landauer in [1] proved that any kind of irreversible computation results with loss of energy above some theoretical threshold. Here, reversible logic emerges as a game-changer, offering a way to significantly reduce energy loss during computations. This can lead to prolonged devices' lifespans and lowered cost of maintenance. The efficiency brought by reversible logic is crucial for the sustainability of large-scale IoT networks, which face substantial energy demands. In this realm, the significance of energy-saving technologies cannot be overstated, especially for devices that rely on battery power.

Moreover, the role of reversible logic extends beyond energy efficiency to enhancing the security of symmetric ciphers, especially in the face of quantum computing threats. The National Institute of Standards and Technology, through initiatives like the Lightweight Cryptography Competition, underscores the importance of developing cryptographic standards resilient against such threats in [2] and [3], especially regarding lightweight cryptography. Representing ciphers as reversible circuits offers a promising avenue for assessing cryptographic algorithms in regards of quantum resilience.

Identity templates are key to optimizing reversible logic circuits by simplifying their complexity. These templates, which leave inputs unchanged, help identify parts of a circuit where there exists a chance of optimization by substituting a portion of circuit with a smaller equivalent, making the circuit more efficient and of lower cost.

Finally, we introduce the concept of suboptimality witnesses, which are gate sequences that, when identified within a circuit, serve as evidence of its non-optimality. These witnesses help eliminate specific gate combinations during exact or optimal synthesis by modeling to a satisfiability problem formulae or other different formal methods. This approach can significantly reduce the search space and potentially speed up the synthesis process by adding more constraints to a SAT formula representing synthesized circuit.

II. Preliminaries

This section is designated to introduce vital definitions and notations which we will use throughout the paper.

A. MCT circuits

In this paper we consider circuits constructed only from *Multiple Control Toffoli* (MCT) gate set - the set contains *NOT, CNOT* and *Toffoli* gates with any number of control lines.

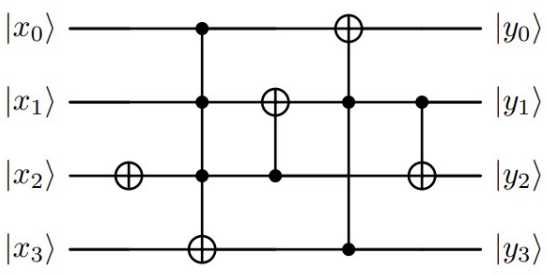

Fig. 1. C Example MCT circuit

For our purposes, without a loss of generality we will consider fully sequential circuits, i. e. in each layer there is exactly one gate and therefore the circuit is equivalent to a

A. Jagielski is with Military University of Technology, Warsaw, Poland (e-mail: adam.jagielski@wat.edu.pl).

sequence of composed gates. The number of gates in circuit is called *Gate Count (GC)* and number of lines (input and output variables) is called its *Width (W)*

Furthermore we will use strict equality relationship between circuits. That means, that two circuits have to be **identical** to consider them equal. In many cases it is convenient to use looser definitions of equality, for example two circuits can be considered equal if there exists permutation of lines which renders them identical or when removing an empty line yields the same result. In this case the most strict definition of equality will be the most useful one.

B. Identity Templates

Identity templates, first presented and used in [4] and [5] are a family of reversible logic circuits with a special property that they do not introduce any change to the input regardless of what the value of input is, i. e. they perform identity function. Although research on this kind of circuits may seem counter intuitive, the templates are crucial for optimizing circuits. Let us focus on the template in figure 2:

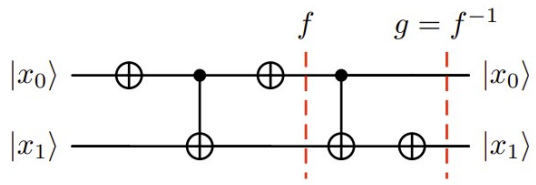

Fig. 2. Example template

The template has been split into two subsequences. The first sequence performs some reversible boolean function f and the second one implements function g, since we know that the whole circuit implements identity function - $g \circ f = id$ follows. That means that g "undoes" f and in fact $g = f^{-1}$. We can invert g function to obtain a different implementation of f because $g^{-1} = f^{-1^{-1}} = f$.

Given some implementation of function as an MCT circuit we can easily find a circuit that implements its inversion simply by mirroring it - which follows from every MCT gate being its own inverse. Finally we achieve two different but equivalent circuits that both implement function f but have ditinct cost of implementation and gate count:

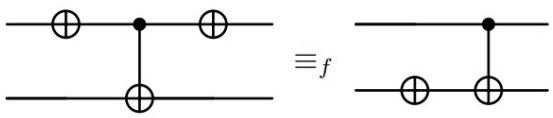

Fig. 3. Equivalent circuits

Now for every MCT circuit we can check if it contains this sequence of gates and replace it with the latter sequence one thus reducing its cost in gates.

This method of optimization was first presented in [5] and further developed in [6], [7], [8], [9].

In section IV we describe further how from single identity template produce more than one substitution rule.

III. SAT-solver synthesis

To synthesize identity templates we will use procedure from [10] and modified in [11]. On input we will define the desired size of identity template and the identity function as a function to be implemented by synthesized circuit. We will denote this method as function $synth(W, GC, f) \rightarrow c/\perp$ where:

- W - Width of circuit;
- GC - Number of gates of circuit;
- f - Function to be implemented by a circuit;
- $c/\perp$ - A circuit of width W and **exactly** GC gates which implements a function f or $\perp$ if there is no such a circuit with given size.

Having some set of already known circuits $C = \{c_1, c_2 \cdots, c_n\}$ which implement f we have to modify the procedure $synth$ such that it can produce a new f-implementing circuit $c' \notin C$ or $\perp$ if C is complete set of such circuits and there is no new circuit possible to synthesize, in particular C may be an empty set.

This can be easily done be simply adding a single clause to SAT problem formula for every $c \in C$ that removes it as a possible solution. We will denote this modified procedure as $synth(W, GC, f, C) \rightarrow c/\perp$ where:

- W, GC, f - As previously;
- $c/\perp$ - As previously, with distinction that $c \notin C$ or $\perp$ if there is no such a circuit, i.e. C is complete set of circuits implementing f with given dimensions.

Finally we can construct a crude algorithm that will produce a full set of identity templates of given dimensions:

Algorithm 1 Complete synthesis of Identity Templates

Require: W, GC
1: $C = \{\}$
2: **while** True **do**
3: $c = synth(W, GC, Id, C)$
4: **if** $c = \perp$ **then**
5: **return** C
6: **else**
7: $C := C \cup \{c\}$
8: **end if**
9: **end while**

While this algorithm is correct it is very ineffective since we have to solve a new SAT problem for each circuit found. In next chapter we will describe a crucial optimization that vastly reduces synthesis time.

IV. Identity Templates unrolling

As previously mentioned solving a new instance of SAT problem might not be the best way to produce whole collection of templates. Here we will describe some procedures that can help us omit some of the computational burden. All of the procedures below describe how given a single $W \times GC$ identity template we might cheaply produce many more with exactly the same dimensions:

A. Mirroring

The simplest method to produce a new template being given one is to simply mirror it. Since mirroring a MCT circuit always yields a circuit implementing an inverse function and $id^{-1} = id$ we know that mirrored template is a template itself.

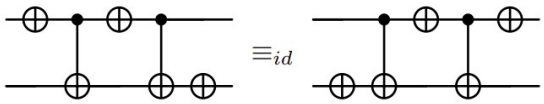

Fig. 4. Mirrored templates

B. Permutation

The second one method to obtain different templates is to permute lines of circuit. The fact that line-wise permutation of template is also a template follows from the fact that permuted function $P(f)$ can be written as a combination of initial permutation of input ρ, the function f itself and inverse permutation ρ^{-1} on output:

$$P(f) = \rho \circ f \circ \rho^{-1} \tag{1}$$

For $f = id$:

$$\begin{aligned} P(id) &= \rho \circ id \circ \rho^{-1} \\ &= \rho \circ \rho^{-1} = id \end{aligned} \tag{2}$$

Example of permuted templates:

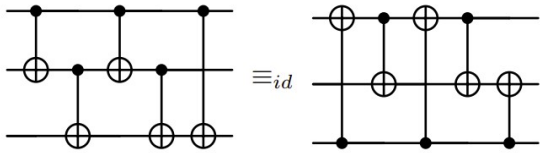

Fig. 5. Permuted templates

C. Rotation

Let template $c = g_n \circ \cdots \circ g_2 \circ g_1$ be a sequence of gates $[g_i]$. We can adjoin gate g_1 to the left side and g_1^{-1} to the right side of the circuit resulting with:

$$\begin{aligned} id &= g_n \circ \cdots \circ g_2 \circ g_1 \\ g_1 \circ id \circ g_1^{-1} &= g_1 \circ g_n \circ \cdots \circ g_2 \circ g_1 \circ g_1^{-1} \\ g_1 \circ g_1^{-1} &= g_1 \circ g_n \circ \cdots \circ g_2 \circ id \\ id &= g_1 \circ g_n \circ \cdots \circ g_2 \end{aligned} \tag{3}$$

The left side of equation did not change preserving the identity function. On the right side the g_1 gate has been moved to the leftmost position. Repeating this process allows us to rotate the template while preserving its properties. The rotation can be also thought as splitting the template into two consecutive sequences and joining them in swapped order. For given $W \times GC$ template we can produce up to $GC - 1$ new templates using this method.

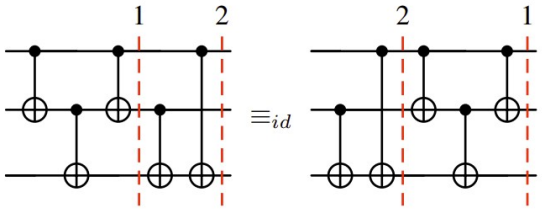

Fig. 6. Rotated templates

D. Gate-swap and DFS

From [4] we know that adjacent gates g_i, g_{i+1} can be swapped if and only if:

$$\begin{cases} t_i \notin Ct_{i+1} \\ t_{i+1} \notin Ct_i \end{cases} \tag{4}$$

where t_i is index of target line and Ct_i are indices of control lines of gate g_i. This swap does not bear any effect whatsoever on the function implemented by the circuit. The circuit from 5 and 6 does have two last gates fitting to be swapped. The swap results with:

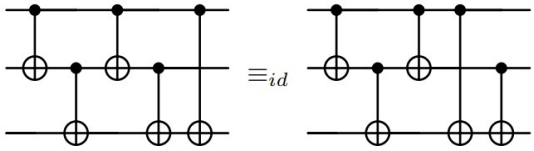

Fig. 7. Gate-swapped templates

For every circuit there might be multiple pairs of gates which can be swapped and swapping two gates my render a new pairs of potential gates for this transformation. For that reason to fully explore the space of possible templates it is necessary to implement DFS algorithm which will recursively search and generate new templates using gate-swap operation.

E. Full unroll

Now we can execute all of the operations described earlier to produce a class of templates according to the algorithm below:

The order of transformations has been chosen experimentally, the one above yielded results in the shortest time. The returned set C is a class of abstraction of templates related to each other *iff* there is some composition of operations described above between them. Figure 4 template class is shown in a figure 8.

Clearly we can see that unrolling procedure is capable of producing many more templates. Even the simplest non-trivial template - one that is not equivalent to gate repetition or gate swap - can be unrolled to up to 20 new circuits, and for templates of bigger sizes the benefit is even greater. In fact for the complete set of 2×5 templates can be found with only one SAT synthesis and unrolling the result and what is

Algorithm 2 Identity Templates Unrolling

Require: c_t
1: $C := \{c_t\}$
2: **for** $c_i \in C$ **do**
3: $\quad C := C \cup DFS(c_i)$
4: **end for**
5: **for** $c_i \in C$ **do**
6: $\quad C := C \cup ROTATE(c_i)$
7: **end for**
8: **for** $c_i \in C$ **do**
9: $\quad C := C \cup MIRROR(c_i)$
10: **end for**
11: **for** $c_i \in C$ **do**
12: $\quad C := C \cup PERMUTE(c_i)$
13: **end for**
14: **return** C

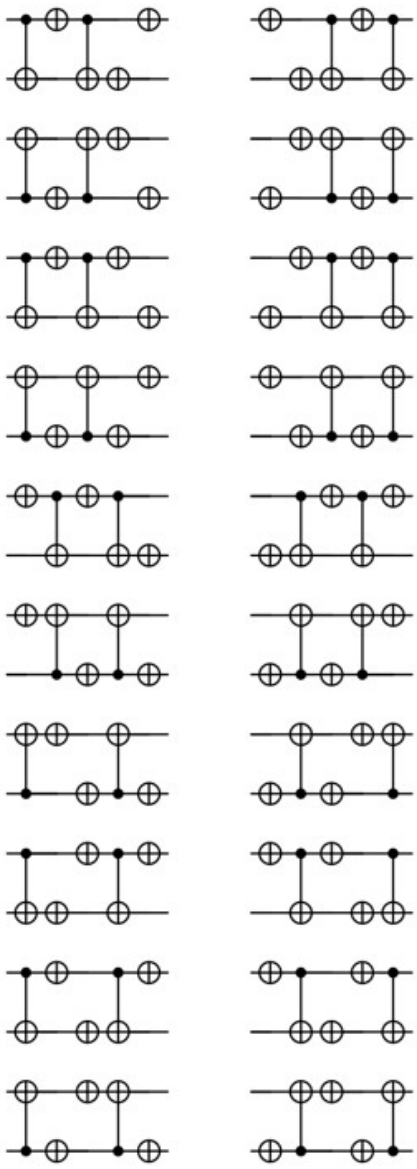

Fig. 8. Template abstraction class

important it does not matter which circuit is originally found in synthesis as all of them would be unrolled to the same set.

Considering that finding solving each subsequent SAT problem takes more time as it has more restrictions, with this method we are able to skip a trailing sequence of the last and the hardest iterations in complete synthesis loop.

Contrary to the synthesis of new circuits, unrolling methods are a simple procedures that simply manipulate list of gates instead of solving complex SAT-problems. The speed-up benefit of this optimization is presented in table I.

TABLE I
COMPARISON OF COMPLETE SYNTHESIS TIMES

Dimensions	T. w/o unroll	T. with unroll	Speedup
2×6	1103ms	73ms	$\times 15.1$
2×7	2113ms	84ms	$\times 25.2$
3×6	1087s	8.49s	$\times 128.0$
3×7	5468s	8.61s	$\times 635.1$

As table presents this optimization is crucial as it reduces expected time of complete synthesis by orders of magnitude. Without it it would not be possible to achieve complete results for templates in greater sizes.

V. SUBOPTIMALITY WITNESSES

A. Motivation

The motivation for this concept is the fact that identity templates, have found usage in circuit post-processing optimization. This approach is presented first in [5]. This process can be used after synthesizing a circuit that implements the desired function with heuristic methods which usually return sub-optimal results.

Our proposition is aimed to harness the usefulness of templates in methods that formally produce optimal results, as aforementioned SAT solver synthesis being one of them.

In our paper we used a method of excluding whole solutions from SAT problem to yield unique solutions in each subsequent iteration. But we are not limited to restricting whole solution from modeled SAT formula. Since modeling those formulae is restriction based we can actually exclude any combination of gates from occurring in synthesized circuit.

Returning back to figure 3 presenting two equivalent circuits up to a function they are implementing. It can be clearly seen, that any optimal circuit of width two, which has a subsequence of gates equal to the circuit at the left-hand-side is not optimal as it could be replaced with right-hand-side sequence resulting with a circuit with lower gate count.

Therefore while modeling a formula for optimal synthesis of some circuit we can explicitly exclude this sequence of gates. This restriction does not change the set of solution for formula as it is implicitly defined in the fact the formula represents an actually optimal circuit, but stating it explicitly can speed up the time of solving.

B. Definition

For any unique identity template of size $W \times GC$ we define its suboptimality witness as a circuit which is composed from $\lfloor \frac{GC}{2} \rfloor + 1$ first gates of the template.

The size of the witness is especially chosen to be as small as possible but still to be representing a proper sequence of gates, which, if substituted, results in circuit size reduction. The smaller the size of witness the more restrictions it puts on the SAT formula as smaller combination of gates is able to render the formula satisfiable. Therefore the "just above a half" size is optimal for construction of witnesses.

Unique witnesses created from class in figure 8 are presented in figure 9.

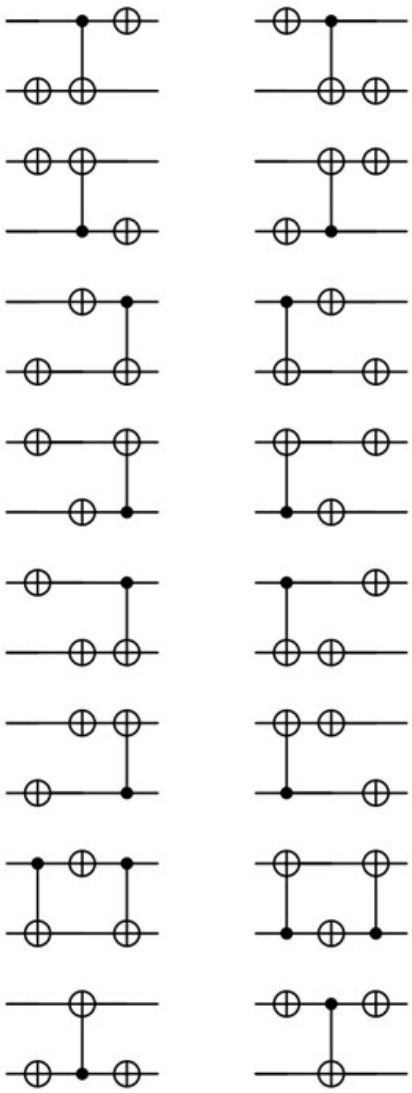

Fig. 9. Witness abstraction class

This class generates 16 witnesses from the total number of 22 2×3 witnesses. The remaining 6 are generated from 2×4 templates.

All of the 20 Templates yield 16 witnesses due to the fact that the last 4 presented in figure 9 are duplicates appearing twice.

VI. Witnesses Distillation Procedure

Distillation procedure is fairly simple. Being given collection of **all possible** (otherwise it would not be sufficient to just take leftmost one sequence from each) identity templates up to size $W \times GC$, each template has to be truncated to $\lfloor \frac{GC}{2} \rfloor + 1$ gates. After that duplicates are to be removed. The last procedure is to remove redundant witnesses. For example, suppose that after truncation process in our witnesses collection we find the two circuits in figure 10.

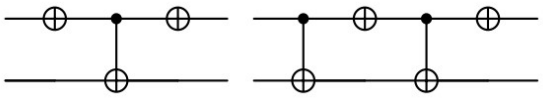

Fig. 10. Witness redundancy

On the same merit why we have chosen the minimal size of witness we can discard the witness on the right side. The combination of gates from the larger witness occurring in synthesized circuit implies that the combination from the smaller also occurs. Therefore the restrictions generated by the bigger circuit are redundant and the whole witness can be discarded as it does not provide any additional information about potential circuit to be synthesized.

VII. Experimental Results

Using methods above we synthesized **complete** collection of identity templates up to a size of 6×7. After synthesis the collection has been distilled into a complete collection of non-redundant witnesses of sizes up to 6×4. The numbers of templates and witnesses of appropriate sizes are listed in tables II and III.

TABLE II
Unique Identity Templates

GC\W	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	4	24	176	1540	13757
3	0	0	0	0	0	0
4	1	34	348	3296	33220	323273
5	0	20	240	2400	24800	244600
6	1	360	13104	269744	4626800	66741377
7	0	504	28644	674352	12343240	185127880

Please note that those results may seem contrary to results presented in [4] where it is stated that no templates of $GC = 7$ exist. This refers only to identity templates which are not constructible from other smaller templates by concatenation. Here we have numbers of **all possible** templates.

The templates were synthesized using winning SAT-solvers from most recent SAT-solvers competition. Template synthesis procedure took a few days of computation.

Witness distillation procedure implemented in low level C++ and vastly optimized took about two weeks due to $\mathcal{O}(n^2)$ complexity of filtering redundant witnesses and duplicates, where n is the number of circuits.

TABLE III
UNIQUE SUBOPTIMALITY WITNESSES

GC\W	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	4	12	32	80	192
3	0	22	252	1952	12720	75072
4	0	10	1992	50028	840300	11715270

VIII. FURTHER RESEARCH

In our future endeavors we plan to verify the hypothesis we have put forward. By preparing appropriate set of benchmarks we are going to test whether the concept of sub-optimality witnesses can bear a significant effect on efficiency of formal methods of synthesis and what is the level of potential gain.

The next step is to optimize further the procedure of restricting formulae using witnesses. We might expect that including all of possible witness based restrictions might occur with bloated formulae that are to complex to solve efficiently. We will try to determine which subsets of witnesses are the most effective in reducing synthesis time and how this set changes depending on the properties of to-be synthesized function.

IX. CONCLUSIONS

In our paper we present concrete and measurable results in tables I, II, III. The first table represents the speedup in synthesis of complete set of Identity Templates using unrolling methodology. The second table represent for the first time the exact numbers of unique and strictly distinct templates. The last table represents the numbers of witnesses of given sizes. The results should be easily reproducible and verifiable. For interested readers we will share on demand both complete lists of fully described circuits presented here and source code which resulted in creating those results.

Finally in this paper for the first time we described the notion of suboptimality witnesses and we put forward the hypothesis of using them in the methods of formally optimal synthesis procedures which we will put to testing in our future research.

REFERENCES

[1] R. Landauer, "Irreversibility and heat generation in the computing process," vol. 5, no. 3, pp. 183–191, 1961. [Online]. Available: http://ieeexplore.ieee.org/document/5392446/

[2] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on lightweight cryptography," p. NIST IR 8114, 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf

[3] M. S. Turan, "Status report on the final round of the NIST lightweight cryptography standardization process," p. NIST IR 8454, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8454.pdf

[4] G. W. Dueck, D. M. Miller, and D. Maslov, "Templates for toffoli network synthesis," 2003.

[5] D. Miller, D. Maslov, and G. Dueck, "A transformation based algorithm for reversible logic synthesis," 2003, pp. 318–323.

[6] D. Maslov, G. Dueck, and D. Miller, "Simplification of toffoli networks via templates," 2003, pp. 53–58.

[7] D. Maslov, C. Young, D. Miller, and G. Dueck, "Quantum circuit simplification using templates," in *Design, Automation and Test in Europe*. IEEE, 2005, pp. 1208–1213. [Online]. Available: http://ieeexplore.ieee.org/document/1395758/

[8] D. Maslov, G. Dueck, and D. Miller, "Toffoli network synthesis with templates," vol. 24, pp. 807–817, 2005.

[9] D. Maslov, D. M. Miller, and G. W. Dueck, "Techniques for the synthesis of reversible toffoli networks," 2006. [Online]. Available: http://arxiv.org/abs/quant-ph/0607166

[10] R. Wille and R. Drechsler, *Towards a Design Flow for Reversible Logic*. Springer Netherlands, 2010. [Online]. Available: http://link.springer.com/10.1007/978-90-481-9579-4

[11] A. Jagielski, "Optimal SAT solver synthesis of quantum circuits representing cryptographic nonlinear functions," vol. 69, no. 3, pp. 261–267, 2023. [Online]. Available: https://journals.pan.pl/dlibra/publication/144359/edition/127355/content