SAT-based Exact Synthesis of Ternary Reversible Circuits using a Functionally Complete Gate Library

Abhoy Kole *, Kamalika Datta‡, Indranil Sengupta *† Rolf Drechsler ‡§
*Department of Computer Science and Engineering, JIS University, India
†Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India
‡German Research Centre for Artificial Intelligence (DFKI), Bremen, Germany
§Institute of Computer Science, University of Bremen, Bremen, Germany
Email: abhoy.kole@jisuniversity.ac.in, kamalika.datta@dfki.de, indranil.sengupta@jisuniversity.ac.in, drechsler@uni-bremen.de

***Abstract*—The problem of synthesis and optimization of reversible and quantum circuits have drawn the attention of researchers for the last two decades due to increasing interest in quantum computing. Although lot of works have been done on the synthesis of binary reversible circuits, very less works have been reported on the synthesis of ternary reversible circuits. Ternary circuits have lower cost of implementation as compared to their binary counterparts. However, the synthesis approaches that exist for ternary reversible circuits either use too many circuit lines (qutrits) or too many gates. Only one prior work has discussed the problem of generating cost-optimal ternary reversible circuits, but for a very restrictive gate library, which limits the approach to a specific subset of ternary reversible functions and often the solution becomes sub-optimal due to the imposed restrictions. The present paper overcomes that restriction, and uses multiple control ternary Toffoli gates with all possible ternary target operations as the gate library. This gate library is functionally complete and can be used to synthesize any arbitrary function. The proposed SAT-based synthesis approach provides low cost solutions in terms of the number of gates for any arbitrary ternary reversible function. Experimental results on various randomly generated permutations as well as standard ternary benchmarks establish this claim. The results can be used as template for other synthesis approaches by observing how far they deviate from the optimal solutions.**

***Index Terms*—Ternary reversible circuit, SAT, exact synthesis**

I. Introduction

Over the last few decades there has been an increase in interest towards quantum computing research, more so with the emergence of prototype quantum computing machines [1]. As possible alternatives to classical CMOS, technologies like quantum-dot cellular automata (QCA) has also been explored for designing low-power high-speed digital circuits [2]. The availability of quantum computers is expected to solve certain computationally hard problems in polynomial time like integer factorization [3], database search [4], etc. With such motivations researchers have explored the problem of quantum circuit synthesis from different perspectives. A quantum circuit essentially consists of a number of elementary gate operations that are applied on quantum bits or *qubits* in sequence. In binary quantum circuits, a qubit can exist in two basis states, say $|0\rangle$ and $|1\rangle$, and also in states that are superposition of the basis states. In ternary quantum circuits, the basic unit of information is called a *qutrit*, which has three basis states $|0\rangle$, $|1\rangle$ and $|2\rangle$ respectively, and can also exist in a state of superposition. The information content per qutrit is more than that of a qubit, and the realization of logic functions requires 63% less number of qutrits as compared to qubits [5].

The synthesis of Boolean functions using binary reversible logic has been a well-studied topic. However, synthesis using ternary reversible logic has received much less attention in the literature. Recently, a few reversible synthesis methods have been proposed to obtain the ternary circuit realization for a given function [6]–[15]. However, most of these approaches produce sub-optimal quantum circuit realizations, and as such it is difficult to assess the quality of the solutions. Exact synthesis approaches are important in this regard that guarantee optimal solutions, and provide a benchmark against which other synthesis approaches can be compared. They can also be used for template-based or local optimizations for a given gate netlist of large functions. However, such approaches come with the overhead of exploring the entire search space, very time consuming and therefore can be applied only to very small functions. Only one prior work exists for the exact synthesis of ternary reversible logic that uses a restrictive gate library [11]. Due to the restriction, the gate library cannot be used to realize all ternary functions.

Various exact synthesis approaches have been proposed in the literature for binary reversible logic [16]–[19]. Many of these works model the problem of synthesis as a Boolean satisfiability (SAT) problem, and use a SAT solver to generate the solution. The main idea of SAT is to determine whether there exists some assignment of variables that satisfies a given Boolean formula. In [11], Kole et al. for the first time presented two exact synthesis methods for ternary reversible functions, using a level-constrained A* heuristic search and a SAT-based method, respectively. In the first approach, heuristic search is used to find a solution, where the circuit depth is increased progressively. In the SAT-based approach, the authors use a SAT formulation that realizes a given function F_z using d number of Ternary Multiple Control Toffoli (TMCT) gates. To obtain minimal gate realization, the search starts with the initial value of $d = 1$, and d is incremented by one at every

step until the specification becomes satisfiable. However, this method is restrictive in the sense that a gate library with only ternary $+1$ shift operation as the target has been used, which is not functionally complete and cannot realize any arbitrary function. However, if we also incorporate the other four target operations $+2$, 01, 02, and 12, the method will become more general and any ternary function can be realized. This is the specific objective of the present paper.

In this paper, we present a synthesis approach for ternary reversible functions, with the following contributions.

a) An exact algorithm for the synthesis of ternary reversible functions has been presented, which generates minimum-gate realizations.
b) The target gate library consists of TMCT gates with all five possible target operations, which is functionally complete.
c) The synthesis problem has been mathematically expressed as a Boolean satisfiability problem, and the solution obtained using a SAT solver.

The rest of the paper is organized as follows. Section II presents a review of ternary reversible circuits and the Boolean satisfiability problem. Section III discusses the proposed exact synthesis approach, the SAT formulation along with all the constraints, and also the overall algorithm for synthesis. Section IV discusses the experimental results on some ternary benchmark circuits, and section V concludes the paper.

II. Preliminaries

In this section, we briefly discuss ternary reversible gates, the various inversion operations that can be used in the target, and also explain the Boolean satisfiability problem. We also discuss various existing ternary reversible synthesis works.

A. Ternary Reversible Operations

The basic unit of information in three-valued or ternary quantum system is called a *qutrit*. The information content per qutrit is more than that of a qubit in conventional binary quantum system, and the realization of logic functions requires 63% less number of qutrits as compared to qubits [5]. An n-input ternary reversible logic gate realizes an $n \times n$ bijective function, and generates a unique n-digit output vector for every unique n-digit input vector.

A ternary reversible operator whenever acts on a qutrit, the state of the qutrit gets modified according to one of the six ternary inversion operations that are depicted in Table I. Here the notation $Z(t)$ is used to indicate a ternary reversible operation Z acting on a qutrit t. The operation Z is classified into three categories, I, $+x$ and xy where I represents the identity operation; $+x$ (for $x = 1$ and $x = 2$) performs modulo-3 addition of x with the current state of the circuit line; and xy (for $x, y \in \{0, 1, 2\}$ and $x \neq y$) changes the state of the circuit line from x to y, or from y to x. The inversion operation is defined in the following way.

TABLE I: Ternary Z inversion operations

	$Z(t)$					
t	I	$+1$	$+2$	12	01	02
0	0	1	2	0	1	2
1	1	2	0	2	0	1
2	2	0	1	1	2	0

Definition 1. *A ternary inversion operation $Z(t)_\delta$ on a qutrit t performs the following state transformations*

$$\begin{aligned} Z(t)_I &= (t+0) \ mod\ 3 \\ Z(t)_{+1} &= (t+1) \ mod\ 3 \\ Z(t)_{+2} &= (t+2) \ mod\ 3 \\ Z(t)_{12} &= (2t+0) \ mod\ 3 \\ Z(t)_{01} &= (2t+1) \ mod\ 3 \\ Z(t)_{02} &= (2t+2) \ mod\ 3. \end{aligned} \quad (1)$$

where $\delta \in \{I, +1, +2, 12, 01, 02\}$.

B. Ternary Reversible Gates

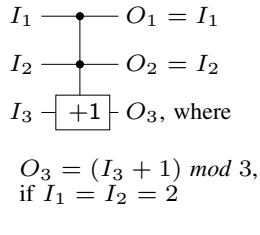
Ternary reversible circuits consists of a cascade of ternary reversible gates. In a ternary reversible circuit, every circuit line can exist in one of three states, 0, 1 or 2. For synthesizing any arbitrary ternary function, various ternary reversible gates like NOT, Feynman, Toffoli, Fredkin, etc. have been used by researchers [20]–[23].

- *A Ternary Toffoli gate*, denoted as $T(c_1, c_2; t)$, has one target line t and two control lines c_1 and c_2. When both c_1 and c_2 are in state 2, the target line changes to $t' = Z(t)$; otherwise, $t' = t$. The states of the control lines c_1 and c_2 do not change.
- *A Ternary Controlled-$Z(t)$ gate*, denoted as $T : (c; t)$, has one target line t and one control line c. When control line c is in state 2, the target line changes to $t' = Z(t)$; otherwise, $t' = t$. The state of the control lines c does not change.
- *A Ternary $Z(t)$ gate*, is a single-qutrit gate $T : (t)$, which always changes the state of the target to $t' = Z(t)$.

Depending on the type of inversion operation considered for a ternary reversible gate, the state transformation of target qutrit also varies. In graphical representation of these gates, the inversion type is specified inside a rectangular box to indicate the applied inversion operation. The following example illustrate one of such ternary reversible gate operation with corresponding graphical representation.

Example 1. *Fig. 1 shows the schematic diagram and truth table of a 2-control ternary Toffoli gate $T_{+1}(\{I_1, I_2\}; I_3)$ that performs the $+1$ inversion operation on I_3 when both I_1 and I_2 are in state 2.*

In the context of the present work, for synthesis we consider ternary multiple control Toffoli (TMCT) gates, i.e. Toffoli gates with arbitrary number of inputs, with the target capable of carrying out one of the six Z operations when activated. A TMCT gate with no control line, one control line, and



(a)

I_1	I_2	I_3	O_1	O_2	O_3
0	0	X	0	0	X
0	1	X	0	1	X
0	2	X	0	2	X
1	0	X	1	0	X
1	1	X	1	1	X
1	2	X	1	2	X
2	0	X	2	0	X
2	1	X	2	1	X
2	2	0	2	2	1
2	2	1	2	2	2
2	2	2	2	2	0

(b)

Fig. 1: Ternary Toffoli gate: (a) Schematic diagram, (b) Truth table where $X \in \{0, 1, 2\}$.

two control lines (i.e. $|C| = 0, 1$ and 2) respectively are also referred to as ternary NOT, ternary Feynman and ternary Toffoli gate, respectively.

C. Boolean Satisfiability Problem

Let $f(x_1, x_2, \ldots, x_n)$ denote an n-variable Boolean function. The Boolean satisfiability problem (SAT problem) is to determine whether there exists an assignment to the variables $x_1, x_2, \ldots, x_n$ such that f evaluates to true, or to prove that such an assignment does not exists. The given function is often represented in conjunctive normal form (CNF). A CNF is a conjunction of a set of disjunctions or clauses where each clause is a set of literals. The literals in the clauses can be either Boolean variables or their negations. The CNF expression is satisfiable (true) if and only if every clause is satisfiable.

Example 2. *Consider a Boolean function in CNF representation as* $f = (x_1 + x_3)(\overline{x}_2 + x_3 + \overline{x}_4)(x_2 + \overline{x}_3)$. *There are 7 literals and 3 clauses in the CNF (corresponding to the three product terms). One possible assignment of the variables that results in* f *to be true is* $x_1 = 1$, $x_2 = 1$, *and* $x_4 = 0$.

SAT is one of the first-known NP-complete problems [24]. However, a number of very efficient SAT solvers exist that are capable of handling large number of variables subject to various constraints. This also make use of Boolean constraint propagation and efficient learning approaches to speed up the inference process [25]. In addition to traditional theorem proving, SAT solvers have been used by researchers in various applications like classical and reversible logic synthesis, automatic test pattern generation, etc.

Typically, SAT solvers operate on the CNF representation of a given function. The solver produces the output as *satisfiable* or *unsatisfiable* depending on the input function. If the CNF expression is *satisfiable*, then it also produces a satisfying assignment to the variables.

Example 3. *Consider the pair of CNF expressions* $f_1 = (x_1 + x_2)(\overline{x}_1 + \overline{x}_2)$ *and* $f_2 = (x_1)(\overline{x_1})$. *If we run the SAT solver by passing the function* f_1 *as input, it will produce a satisfiable assignment, either* $x_1 = 1, x_2 = 0$, *or* $x_1 = 0, x_2 = 1$. *However, for the function* f_2, *the solver will declare the function as unsatisfiable.*

Some research has been carried out on ternary reversible logic synthesis in recent years [26], [27]. Synthesis of ternary reversible functions to elementary quantum gate libraries are typically carried out in two steps, ternary reversible synthesis followed by elementary quantum gate decomposition. Various approaches exist to synthesize ternary reversible logic circuits, viz. group theory based approach [6], [13], exact synthesis approach [11], ternary max-min projection approach [28], ternary Galois field sum-of-products expression (TGFSOP) approach [5], transformation based approach [21], [29], and other kind of approaches [14], [15]. From the generated netlist of ternary reversible gates, each gate is decomposed into ternary elementary gates like *Muthukrishnan-Stroud (M-S) Gate* and *Ternary Shift Gate*, which can be realized using technologies like ion-trap [30].

In [6] Rani et al. showed how to synthesize any Boolean functions into cascade of ternary reversible gates. They have generated 3-cycles from the permutations and then synthesized each of these cycles into ternary reversible gates. Some group theoretic based approaches like [6], [13] have used various rules to synthesize 3-cycles and 2-cycles into cascade of ternary reversible gates. Although these methods seems straightforward to implement but results in huge number of gates in the final netlist. In [14] Monfared et al. proposed a Quantum Ternary Multiplication Gate (QTMG), and showed the design of 1-qutrit and 2-qutrit multiplier circuits. Peres type Ternary Toffoli gate is introduced in [15], where after synthesis they used Barenco-like decomposition for various cases. In [29] Miller et al. used transformation based synthesis for 3-valued reversible functions. They have synthesized a given function using two methods, one in descending order and another in ascending with respect to the inputs. Although this method provides a way to synthesize any arbitrary Boolean function but the solutions generated might not be close to optimal.

In this paper we focus on a exact SAT-based ternary reversible logic synthesis approach considering multiple control ternary Toffoli gates with all possible ternary target operations.

III. SAT Based Exact Synthesis of Ternary Reversible Circuits

There exists some earlier works where SAT-based techniques have been used for the exact synthesis and test pattern generation of binary reversible circuits [17], [31]. In this paper, we present a functionally complete SAT-based approach for the exact synthesis of ternary reversible circuits. We consider TMCT gates as the technology library with all the five possible Z inversion operations $+1$, $+2$, 01, 02 and 12 as listed in Table I. Since the identity operation I does not change the state of the target line, we do not consider it in the formulation.

The synthesis problem concerns the identification of the set of d TMCT gates that realizes a given ternary function f, where d denotes the depth of the circuit. The process is

iterative, where we start with $d = 1$ and progressively increase the value of d by 1. We stop when either a solution is obtained, some maximum value of d is reached, or the time budget runs out. The SAT formulation for the synthesis problem is discussed in the following subsections.

A. SAT Formulation

For a ternary reversible function f, we construct the SAT formulation in terms of a Boolean function S_d that is satisfiable iff there exists a TMCT gate sequence of size d that realizes f. We discuss the various constraints in the formulation below.

1) Representing the chosen TMCT gate: Let n denote the number of lines in the ternary circuit, and d the number of TMCT gates (numbered 0 to $d-1$). The chosen TMCT gate at depth k, where $0 \leq k \leq d-1$, can be expressed as:

$$t^k = (t^k_{\lceil \log_2 n \rceil} \cdots t^k_1), \quad (2)$$

$$c^k = (c^k_{n-1} \cdots c^k_1). \quad (3)$$

For n lines, the position of the target is specified by the bit-vector t^k of size $\lceil \log_2 n \rceil$ bits. Control connections can exist in any of the remaining $n-1$ lines, as specified by the $(n-1)$-bit vector c^k. If $([t^k]_2 + l) \mod n$ is a control line, we set $c^k_l = 1$; otherwise, we set $c^k_l = 0$.

Example 4. *Fig. 2 shows the representation of a TMCT gate, $T_{01}(\{0,1\},3)$ at depth k for a 4-qutrit ternary reversible circuit. The bit-vector for target of the TMCT gate on qutrit 3 will be $t^k = (11)$. Similarly, the controls can be represented by a bit-vector $c^k = (011)$, since $([t^k]_2 + l) \ mod \ n$ for $l = 1, 2$ and $n = 4$ indicate control qutrits $0((3+1) \ mod \ 4)$ and 1 respectively.*

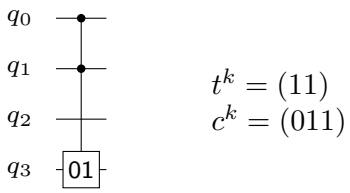


Fig. 2: An example gate representation.

2) Encoding of target operation: The set of five possible target operations (except the identity mapping I) and their binary encoding used in the formulation are shown in Table II. Three bits are required to encode one of the five target operations.

The bit-vector for the target operation of a gate at depth k, where $0 \leq k \leq d-1$ is defined as

$$o^k = (o^k_3 o^k_2 o^k_1). \quad (4)$$

Example 5. *Consider again the TMCT gate, $T_{01}(\{0,1\},3)$ shown in Fig. 2. According to the binary encoding presented in Table II, the target operation 01 is expressed by the bit-vector $o^k = (010)$.*

TABLE II: Encoding of target operations

Operation	Binary Code	Decimal Equivalent
+1	0 0 0	0
+2	0 0 1	1
01	0 1 0	2
02	0 1 1	3
12	1 0 0	4

3) Exclusion constraints for target operation: The exclusion constraint ensures that an illegal target operation at depth k ($0 \leq k \leq d-1$) must not be assigned to o^k. Since there are only five basic operations, the constraint can be written as:

$$\bigwedge_{k=0}^{d-1} [o^k]_2 < 5. \quad (5)$$

Example 6. *During synthesis, an assignment like $o^k = (110)$ for k-th gate operation should not be considered, since the operation $[110]_2 = 6$ violets the constraint* (5)*.*

4) Input-output constraints: The input-output constraints set the input and output pairs of each line of the truth table as specified by the reversible function f. For specifying these constraints, we first state how the ternary logic values on the circuit lines are encoded in binary. Each ternary state is encoded in two bits (x, y) as shown in Table III.

TABLE III: Encoding of ternary states

Ternary State	Binary Encoding x	y
0	0	0
1	0	1
2	1	0

The input/output constraints for an n-input ternary refersible function can be written as:

$$\bigwedge_{i=0}^{3^n-1} \bigwedge_{j=0}^{n-1} x^0_{ij} = i[j]_{MSB} \wedge y^0_{ij} = i[j]_{LSB} \wedge x^d_{ij} = f(i)[j]_{MSB} \wedge y^d_{ij} = f(i)[j]_{LSB}. \quad (6)$$

where x^0_{ij} (y^0_{ij}) is assigned to 0 or 1 according to the MSB (LSB) of the i^{th} row and j^{th} column of the ternary truth table input, whereas x^d_{ij} (y^d_{ij}) is assigned to 0 or 1 according to the i^{th} row and j^{th} column of the ternary truth table output.

Example 7. *The encoding for a 2-input ternary function ($n = 2$) is illustrated in Table IV. The ternary truth table is shown in Table IV(a), and the corresponding binary encoding obtained after Boolean transformation in Table IV(b). For circuit depth $d = 3$ and truth-table row $i = 5$, the input/output constraint* (7) *is added.*

$$(x^0_{50} = 0 \wedge y^0_{50} = 1) \wedge (x^0_{51} = 1 \wedge y^0_{51} = 0) \wedge (x^3_{50} = 1 \wedge y^3_{50} = 0) \wedge (x^3_{51} = 0 \wedge y^3_{51} = 0). \quad (7)$$

TABLE IV: Example encoding of ternary reversible function f: (a) Ternary truth table, (b) Binary encoding.

(a) Ternary Function f

i	$j=0$	$j=1$	$j=0$	$j=1$
0	0	0	1	0
1	0	1	2	1
2	0	2	1	1
3	1	0	2	2
4	1	1	0	1
5	1	2	2	0
6	2	0	0	0
7	2	1	0	1
8	2	2	0	2

(b) Function f After Encoding

i	$j=0$		$j=1$		$j=0$		$j=1$	
	x	y	x	y	x	y	x	y
0	0	0	0	0	0	1	0	0
1	0	0	0	1	1	0	0	1
2	0	0	1	0	0	1	0	1
3	0	1	0	0	1	0	1	0
4	0	1	0	1	0	0	0	1
5	0	1	1	0	1	0	0	0
6	1	0	0	0	0	0	0	0
7	1	0	0	1	0	0	0	1
8	1	0	1	0	0	0	1	0

5) Functional constraints: The functional constraints specify the way the line values are updated across levels as a result of the gate computations. At depth k ($0 < k < d$), for input line states (x_i^k, y_i^k) and gate operation $T(c^k, t^k, o^k)$, the output lines (x_i^{k+1}, y_i^{k+1}) are computed as (see Fig. 3):

$$\bigwedge_{i=0}^{3^n-1}\bigwedge_{k=0}^{d-1} \quad \left[x_i^{k+1} = T_x(x_i^k, y_i^k, c^k, t^k, o^k)\right] \wedge \left[y_i^{k+1} = T_y(x_i^k, y_i^k, c^k, t^k, o^k)\right] \tag{8}$$

where the functions $T_x()$ and $T_y()$ together realize the operation of a ternary reversible gate $T(c^k, t^k, o^k)$.

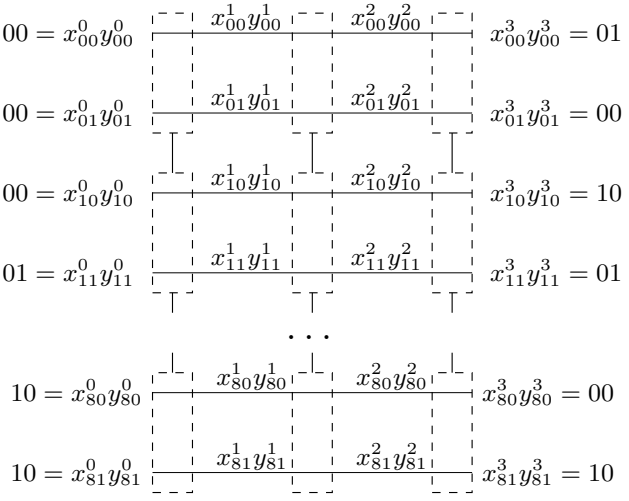
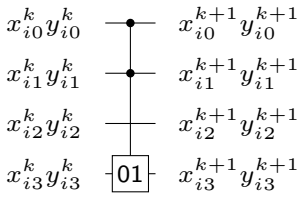


Fig. 3: Symbolic conventions during gate computations.

Example 8. *Consider the TMCT gate $T(c^k, t^k, o^k)$ as shown in Fig. 4a operates at depth k with control lines $c^k = (011)$, target line $t^k = (11)$ and operation $o^k = (010)$. The functions $T_x(x_i^k, y_i^k, c^k, t^k, o^k)$, and $T_y(x_i^k, y_i^k, c^k, t^k, o^k)$ add the constraint* (9) *for each truth table entry i to implement the state change of the target qutrit as shown in Fig. 4b when control lines* 0 *and* 1 *are assigned to* 2*.*

$$\begin{aligned} t^k = (11) \wedge c^k = (011) \wedge o^k = (010) \\ \Longrightarrow \\ \left(x_{i0}^{k+1} = x_{i0}^k\right) \wedge \left(y_{i0}^{k+1} = y_{i0}^k\right) \\ \wedge \left(x_{i1}^{k+1} = x_{i1}^k\right) \wedge \left(y_{i1}^{k+1} = y_{i1}^k\right) \\ \wedge \left(x_{i2}^{k+1} = x_{i2}^k\right) \wedge \left(y_{i2}^{k+1} = y_{i2}^k\right) \\ \wedge \left(x_{i3}^{k+1} = x_{i3}^k\right) \\ \wedge \left(y_{i3}^{k+1} = \overline{x_{i3}^k} \wedge \overline{y_{i3}^k} \wedge x_{i0}^k \wedge \overline{y_{i0}^k} \wedge x_{i1}^k \wedge \overline{y_{i1}^k}\right). \end{aligned} \tag{9}$$



(a) T_{01}

x_{i3}^k	y_{i3}^k	x_{i3}^{k+1}	y_{i3}^{k+1}
0	0	0	1
0	1	0	0
1	0	1	0
1	1	×	×

(b) $x_{i3}^k y_{i3}^k \rightarrow x_{i3}^{k+1} y_{i3}^{k+1}$

Fig. 4: An example of gate computation.

6) Exclusion constraints for illegal assignments: These constraints ensure that illegal assignments for target t^k are excluded, i.e. t^k never exceeds the number of circuit lines n, and can be specified as:

$$\bigwedge_{k=0}^{d-1} [t^k]_2 < n. \tag{10}$$

Example 9. *During synthesis of a reversible ternary circuit comprising $n = 9$ lines, the target line is represented by a 4-bit vector, i.e. $t^k = (t_4 t_3 t_2 t_1)$. An assignment like $t^k = (1101)$ leads to line $[1101]_2 = 13$ that does not refer to a valid circuit line and must be excluded.*

B. The Overall Algorithm

Based on the SAT formulation as discussed above, the procedure for exact synthesis of a ternary reversible function is presented as *Algorithm 1*. The inputs to the algorithm are the truth table of the ternary reversible function f, and the maximum depth of realization d_{max}. As mentioned earlier, the SAT solver is invoked several times, starting with the circuit depth d initialized to 1, which is incremented by 1 at every step. The process is repeated until a solution is found or the maximum preset value of d_{max} is reached. Any satisfiable solution to the given CNF for the smallest value of d gives the circuit realization with minimum cost. The process terminates

if no solution is obtained within d_{max}, or the time budget expires.

Algorithm 1: Exact Synthesis of Ternary Reversible Function

Input: a) Given ternary function f
b) Maximum depth of realization d_{max}
Output: TMCT gate realization of f

begin
  status = failure;
  $d = 1$;
  while (*status* = failure **and** $d \leq d_{max}$) **do**
    S_d = *GenerateCNF* (f, d);
    status = *SAT_Solver* (S_d);
    if (*status* = success) **then**
      assign = *ExtractAssignment*();
      G = *GenerateNetlist* (*assign*);
      return (G);
    else
      $d = d + 1$;
    endif
  end while
  return (NULL); // Solution not found
end

In the algorithm, the function *GenerateCNF (f, d)* generates the CNF corresponding to a given function f and specified circuit depth d, which is fed to the SAT solver using the function *SAT_Solver()*. If the SAT solver is able to find a solution within depth d, it returns the status *success* along with the corresponding variable assignment; otherwise, it returns the status *failure*. If the status returned is *success*, then the function *ExtractAssignment()* is called to get the variable assignments provided by the SAT solver, which is used by the function *GenerateNetlist()* to obtain the final gate netlist. The value of circuit depth d is initialized to 1, and is increased by 1 across iterations. If the value of d exceeds some preset threshold d_{max} and no solution is found, the algorithm terminates by returning a NULL netlist that indicates no solution has been found.

The proposed SAT approach is better than the previously devised SAT approach [11] in two ways. Firstly, the consideration of $+2$, 12, 01 and 02 target operations in addition to $+1$ minimizes the number of gates in realized circuit for certain ternary reversible functions as illustrated by the following example.

Example 10. *The realization of the permutation* $F = (5, 3, 8, 6, 7, 2, 0, 1, 4)$ *using SAT approach [11] requires* 6 2*-qutrit gates, whereas it requires only* 4 2*-qutrit gates using the proposed approach as shown in Fig 5.*

Secondly, the proposed approach can also realize certain permutations that are not synthesizable using earlier SAT approach [11] as illustrated in the following example.

Example 11. *For permutation* $F = (7, 4, 3, 1, 2, 0, 5, 8, 6)$*, the SAT based approach of [11] is unable to provide any solution,*

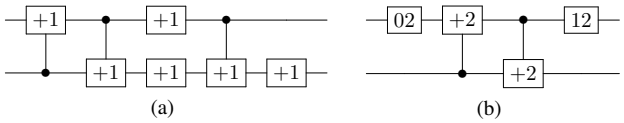


Fig. 5: Realizing $F = (5, 3, 8, 6, 7, 2, 0, 1, 4)$ using (a) SAT approach presented in [11], (b) Proposed approach.

whereas the proposed approach gives the realization as shown in Fig 6.

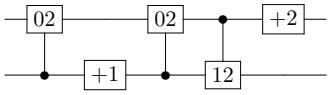


Fig. 6: Realizing $F = (7, 4, 3, 1, 2, 0, 5, 8, 6)$ using proposed approach.

A detail analysis of the proposed synthesis approach is presented in the following section.

IV. Experimental Results

The proposed synthesis approach has been implemented in C++ and the experiments run on a core-i3 machine with 2.4GHz clock and 4GB memory. We have used the MiniSAT SAT solver [25] for obtaining solutions to the SAT formulation problem as discussed in the previous section. In order to evaluate the effectiveness of the proposed approach, we have conducted two sets of experiments where the algorithm is executed by setting the threshold $d_{max} = 100$, and maximum runtime of 32 hours.

A. Synthesis Results for Randomly Generated Benchmarks

In our first set of experiments we have generated a random set of permutations, and run both the synthesis approaches (viz. the approach presented in [11], and the proposed approach). The synthesis results are presented in Table V. In the table, the first column shows the names of the random permutations, and the second column gives the number of qutrits (n) required to realize the permutations. In the next three columns, the synthesis results in terms of the number of TMCT gates (G), Muthukrishnan-Stroud (M-S) gate count (*Cost*) and run-time in seconds (*Time*) respectively, obtained using the approach [11] are given. The corresponding values for the proposed approach are presented in the next three columns. The final column shows the % improvement in *Cost* that is achieved by the proposed approach over [11].

For all these permutations, the proposed approach is able to generate the corresponding netlists, whereas the previous approach [11] either provides solutions with more number of gates or is unable to provide any solution.

B. Synthesis Results for Standard Benchmarks

For another set of ternary benchmarks, the synthesis results are presented in Table VI. In the table, the names of the benchmarks are shown in the first column, with the second

TABLE V: Comparative study of SAT-based approaches.

Name	n	Approach [11]			Proposed Approach			Improv.
		G	Cost	Time	G	Cost	Time	(%)
p-2-3-1	2	12	12	1.43	2	2	0.02	83.4
p-2-3-2	2	4	4	0.02	2	2	0.01	50.0
p-2-4-1	2	4	4	0.01	3	3	0.05	25.0
p-2-4-2	2	-	-	-	3	3	0.03	-
p-2-5-1	2	-	-	-	3	3	0.03	-
p-2-5-2	2	5	5	0.02	3	3	0.04	40.0
p-2-6-1	2	8	8	0.11	5	5	0.12	37.5
p-2-6-2	2	-	-	-	5	5	0.15	-
p-3-3-1	3	4	4	0.06	3	3	0.19	25.0
p-3-3-2	3	-	-	-	3	3	0.18	-
p-3-4-1	3	6	6	0.19	4	4	0.40	33.3
p-3-4-2	3	-	-	-	3	7	0.19	-
p-3-5-1	3	6	6	0.21	5	5	0.77	16.7
p-3-5-2	3	-	-	-	5	13	1.44	-
p-3-6-1	3	-	-	-	4	4	0.36	-
p-3-6-2	3	7	15	0.28	4	8	0.36	46.7
p-4-3-1	4	-	-	-	3	11	0.83	-
p-4-3-2	4	4	16	0.33	3	11	1.13	31.3
p-4-4-1	4	5	21	0.83	4	16	2.12	23.8
p-4-4-2	4	-	-	-	4	20	2.16	-
p-4-5-1	4	6	26	1.04	5	17	5.53	34.6
p-4-5-2	4	-	-	-	5	17	8.59	-
p-4-6-1	4	-	-	-	6	22	6.59	-
p-4-6-2	4	9	13	12.19	6	10	14.60	23.1

column showing the number of circuit lines (i.e. qutrits). The next three columns show the number of M-S gates required for realization using the approaches reported in [9], [32] and our proposed SAT-based approach, respectively. The % improvements of the proposed approach over the best known existing approach are shown in the next column. The last column gives the runtime of the proposed method in seconds. In the best case, 62.5% reduction in M-S gate count has been found. On an average, 37% improvement in M-S gate count has been observed in the proposed synthesis approach as compared to [9].

It may be noted that the proposed approach is guaranteed to provide a minimum-gate solution with respect to the extended ternary gate library, since we use a SAT-solver to look for a solution by progressively increasing the value of circuit depth d. The smallest value of d for which a solution is found gives the optimal solution. Also, we are able to synthesize any given ternary function within the specified time budget, as the method uses a functionally complete gate library. The execution time increases rapidly due to the increase in number of qutrits and value of d. For example, the realization of 4-qutrit benchmark *Mul2* is obtained for $d = 7$ and M-S gate count 11 as shown in Fig. 7, whereas the realization of 4-qutrit benchmark *sum4* has much higher run-time due to higher value of d, i.e. d and M-S gate count both are 9 that can be verified from Table VI.

V. Conclusion

In this paper, a SAT-based approach for the exact synthesis of ternary reversible circuits has been presented. The method uses a functionally complete gate library as compared to a previous SAT-based approach that used a restrictive gate library, and is therefore capable of handling a much wider range of functions. In this approach we have used ternary multiple control Toffoli (TMCT) gates, with all five possible Z inversion operations +1, +2, 01, 02 and 12. Results have been reported for various random ternary permutations as well as ternary benchmark functions. We have also compared the results with two previous approaches available in the literature. The method generates optimal solution for the various benchmarks, however, as expected it incurs large run times for some of the benchmarks. This method can serve as a template to assess the quality of solutions generated by other synthesis methods, and also for local optimization methods (e.g. window optimization).

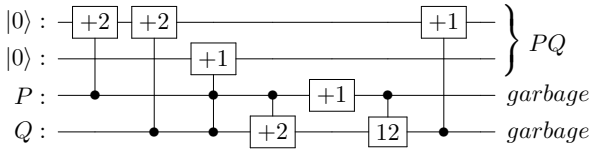


Fig. 7: Netlist realizing the benchmark *Mul2*.

TABLE VI: Comparative study for ternary benchmarks [33].

Benchmark	n	M-S Gate Count			Reduction	Time
		[9]	[32]	SAT	(%)	(sec.)
sum2	2	4	5	3	25.00	0.13
sqsum2	3	15	10	5	50.00	2.74
avg2	3	11	15	9	18.18	3.71
sum3	3	8	10	6	25.00	14.49
prod2	3	12	20	6	50.00	18.85
sqsum3	4	36	15	6	60.00	74.15
Mul2	4	13	25	11	15.38	255.74
tfadd	4	26	55	21	19.23	3089.20
sum4	4	12	15	9	25.00	34831.71
prod3	4	24	65	9	62.50	95923.70
avg3	4	38	40	17	55.26	115148.56

References

[1] IBM Q. https://www.research.ibm.com/ibm-q. [Accessed: 2019-03-20].

[2] G. Cocorullo, P. Corsonello, F. Frustaci, and S. Perri. Design of efficient BCD adders in quantum-dot cellular automata. *IEEE Transactions on CAS-II: Express Briefs*, 64(5):575–579, 2017.

[3] L.K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. ACM Symp. on Theory of Computing*, pages 212–219, Jul 1996.

[4] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. Symp. on Foundations of Computer Science*, pages 124–134, Nov 1994.

[5] M. H. A. Khan, M. A. Perkowski, M. R. Khan, and P. Kerntopf. Ternary GFSOP minimization using Kronecker decision diagrams and their synthesis with quantum cascades. *Journal of Multi Valued Logic and Soft Computing*, 11:567–602, 2005.

[6] P. M. N. Rani, A. Kole, K. Datta, and A. Chakrabarty. Realization of ternary reversible circuits using improved gate library. In *Proc. Intl. Conference on Advances in Computing & Communications*, pages 153–160, Cochin, India, September 2016.

[7] M. Khan and J. E. Rice. Ternary Max-Min algebra for representation of reversible logic functions. In *Proc. Intl. Symposium on Circuits and Systems (ISCAS)*, pages 1670–1673, Montreal, Canada, May 2016.

[8] M. Haghparast, R. Wille, and A. T. Monfared. Towards quantum reversible ternary coded decimal adder. *Quantum Information Processing*, 16(11):284/1–25, November 2017.

[9] S. Basu, S. B. Mandal, A. Chakrabarti, S. Sur-Kolay, and A. K. Choudhury. An efficient synthesis method for ternary reversible logic. In *Proc. Intl. Symposium on Circuits and Systems (ISCAS)*, pages 2306–2309, May 2016.

[10] A. T. Monfared and M. Haghparast. Design of new quantum/reversible ternary subtractor circuits. *Journal of Circuits, Systems and Computers*, 25(02):1650014:1–8, 2016.

[11] A. Kole, P. M. N. Rani, K. Datta, I. Sengupta, and R. Drechsler. Exact synthesis of ternary reversible functions using ternary toffoli gates. In *Proc. 47th Intl. Symposium on Multiple-Valued Logic (ISMVL)*, pages 179–184, Novi Sad, Serbia, May 2017.

[12] P. M. N. Rani, A. Kole, and K. Datta. A ternary decision diagram (TDD)-based synthesis approach for ternary logic circuits. *Journal of The Institution of Engineers (India), Series B*, 100(4):295–307, 2019.

[13] P. M. N. Rani and K. Datta. Improved ternary reversible logic synthesis using group theoretic approach. *Journal of Circuits, Systems and Computers*, 29(12):2050192:1–2050192:24, 2020.

[14] A.T. Monfared and M. Haghparast. Quantum ternary multiplication gate (QTMG): Toward quantum ternary multiplier and a new realization for ternary Toffoli gate. *Journal of Circuits, Systems and Computers*, 29(5):2050071:1–2050071:22, 2020.

[15] C. Moraga. Ternary Toffoli-type reversible gates: Control alternatives and quantum models. In *Proc. Intl. Symp. on Multiple-Valued Logic*, pages 101–106, May 2021.

[16] R. Wille and D. Große. Fast exact Toffoli network synthesis of reversible logic. In *Proc. Intl. Conference on CAD (ICCAD)*, pages 60–64, San Jose, California, November 2007.

[17] D. Große, X. Chen, G. W. Dueck, and R. Drechsler. Exact SAT-based Toffoli network synthesis. In *Proc. 17th ACM Great Lakes Symposium on VLSI*, pages 96–101, Stresa-Lago Maggiore, Italy, March 2007.

[18] R. Wille, D. Große, M. Soeken, and R. Drechsler. Using higher levels of abstraction for solving optimization problems by boolean satisfiability. In *Proc. IEEE Computer Society Annual Symposium on VLSI*, pages 411–416, Montpellier, France, April 2008.

[19] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact synthesis of elementary quantum gate circuits. *Journal of Multiple-Valued Logic and Soft Computing*, 15(4):283–300, January 2009.

[20] A. B. Khlopotine, M. A. Perkowski, and P. Kerntopf. Reversible logic synthesis by iterative compositions. In *IWLS*, pages 261–266, 2002.

[21] E. Curtis and M. Perkowski. A transformation based algorithm for ternary reversible logic synthesis using universally controlled ternary gates. *Proc. IWLS*, pages 2–4, 2004.

[22] S. Kotiyal, H. Thapliyal, and N. Ranganathan. Design of a ternary barrel shifter using multiple-valued reversible logic. In *Conf. on Nanotechnology*, pages 1104–1108, 2010.

[23] X. Li, G. Yang, and D. Zheng. Logic synthesis of ternary quantum circuits with minimal qutrits. *Journal of Computers*, 8(3):1941–1946, December 2013.

[24] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. Symp. on Theory of Computing*, page 151–158, 1971.

[25] N. Eén and N. Sörensson. MiniSAT SAT solver. MiniSAT is available at http://minisat.se.

[26] M. H. A. Khan and M. A. Perkowski. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. *Journal of Systems Architecture*, 53:453–464, 2007.

[27] M. M. Khan, A. K. Biswas, S. Chowdhury, M. Hasan, and A. I. Khan. Synthesis of GF(3) based reversible/quantum logic circuits without garbage output. In *Proc. Intl. Symp. on Multiple-Valued Logic*, pages 98–102, 2009.

[28] S. B. Mandal, A. Chakrabarti, and S. Sur-Kolay. Synthesis techniques for ternary quantum logic. In *Proc. Intl. Symp. on Multiple-Valued Logic*, pages 218–223, 2011.

[29] D. M. Miller and G.W. Dueck. Descending order transformation-based synthesis of MVL reversible circuits. In *Proc. Intl. Symp. on Multiple-Valued Logic*, pages 107–112, May 2021.

[30] A. Muthukrishnan and Jr C. R. Stroud. Multivalued logic gates for quantum computation. *Phys. Rev. A*, 62(5):052309/1–8, 2000.

[31] H. Zhang, R. Wille, and R. Drechsler. Sat-based atpg for reversible circuits. In *Proc. 5th Intl. Design and Test Workshop (IDT)*, pages 149–154. IEEE, 2010.

[32] M.H.A. Khan and M. Perkowski. Evolutionary algorithm based synthesis of multi-output ternary functions using quantum cascade of generalized ternary gates. *International Journal on Multiple-Valued Logic and Soft Computing*, 2005.

[33] M. H. A. Khan, M. A. Perkowski, and M. R. Khan. Ternary Galois field expansions for reversible logic and kronecker decision diagrams for ternary GFSOP minimization. In *Proc. 34th Intl. Symposium on Multiple-Valued Logic*, pages 58–67, 2004.

APPENDIX: THE RANDOM PERMUTATIONS

The random permutations for which synthesis results have been reported in Table V are given below.

Name	Permutation
p-2-3-1	3 4 5 0 1 8 6 7 2
p-2-3-2	2 0 1 5 3 4 7 8 6
p-2-4-1	2 0 4 5 3 8 6 7 1
p-2-4-2	0 1 8 3 4 7 2 6 5
p-2-5-1	8 7 0 2 1 3 5 4 6
p-2-5-2	8 6 7 0 1 2 5 3 4
p-2-6-1	2 3 4 8 5 7 6 0 1
p-2-6-2	7 4 3 1 2 0 5 8 6
p-3-3-1	0 1 8 3 4 2 7 5 6 9 10 17 12 13 11 16 14 15 21 22 20 25 23 24 18 19 26
p-3-3-2	21 22 23 18 19 20 24 25 26 12 13 14 9 10 11 15 16 17 3 5 4 0 2 1 6 8 7
p-3-4-1	0 1 5 3 4 8 24 25 20 9 10 14 12 13 17 2 6 7 19 23 18 22 26 21 11 15 16
p-3-4-2	0 1 2 3 4 5 15 16 26 9 10 11 12 13 14 6 7 8 19 20 18 22 23 21 25 17 24
p-3-5-1	9 10 22 12 13 8 26 24 19 23 21 2 6 7 5 20 18 17 0 1 11 3 4 14 15 16 25
p-3-5-2	21 22 23 6 7 8 0 1 2 12 13 5 15 16 17 9 10 11 3 4 14 26 25 24 18 19 20
p-3-6-1	0 1 15 3 4 14 8 7 11 9 10 21 12 13 26 17 16 20 18 19 6 24 25 5 23 22 2
p-3-6-2	21 22 14 24 25 17 18 19 11 3 4 23 6 7 26 0 1 20 12 13 2 15 16 5 9 10 8
p-4-3-1	0 1 74 3 4 77 6 7 80 9 10 65 12 13 68 15 16 71 18 19 56 21 22 59 24 25 62 27 28 2 30 31 5 33 34 8 36 37 11 39 40 14 42 43 17 45 46 20 48 49 23 51 52 26 72 73 29 75 76 32 78 79 35 63 64 38 66 67 41 69 70 44 54 55 47 57 58 50 53 61 60
p-4-3-2	0 1 29 3 4 32 6 7 35 9 10 38 12 13 41 15 16 44 18 19 20 21 22 23 24 25 26 27 28 56 30 31 59 33 34 62 36 37 65 39 40 68 42 43 71 45 46 50 48 49 53 51 52 47 54 55 2 57 58 5 60 61 8 63 64 11 66 67 14 69 70 17 75 76 74 78 79 77 72 73 80
p-4-4-1	18 19 20 21 22 23 25 26 24 0 1 2 3 4 5 7 35 6 9 10 11 12 13 14 16 44 15 45 46 47 48 49 50 52 53 51 27 28 29 30 31 32 34 62 33 36 37 38 39 40 41 43 71 42 72 73 74 75 76 77 79 80 78 54 55 56 57 58 59 61 8 60 63 64 65 66 67 68 70 17 69
p-4-4-2	0 1 2 3 4 5 6 7 17 9 10 11 12 13 14 15 16 8 18 19 26 21 22 20 24 25 23 27 28 29 30 31 32 33 34 44 36 37 38 39 40 41 42 43 35 45 46 53 48 49 47 51 52 50 60 61 62 57 58 59 54 55 65 69 70 71 66 67 68 63 64 56 78 79 74 75 76 80 73 77 72
p-4-5-1	0 1 2 3 4 5 33 34 35 9 10 11 12 13 14 42 43 44 45 46 47 48 49 50 79 53 78 27 28 29 30 31 32 60 61 62 36 37 38 39 40 41 69 70 71 75 76 77 24 25 26 73 80 72 57 58 59 6 7 8 54 55 56 66 67 68 15 16 17 63 64 65 18 19 20 21 22 23 52 74 51
p-4-5-2	0 1 5 3 4 8 6 7 2 9 10 14 12 13 17 15 16 11 24 25 74 18 19 77 21 22 80 54 55 59 57 58 62 60 61 56 63 64 68 66 67 71 69 70 65 78 79 26 72 73 23 75 76 20 27 28 32 30 31 35 33 34 29 36 37 41 39 40 44 42 43 38 51 52 47 45 46 50 48 49 53
p-4-6-1	0 1 2 3 4 5 6 7 80 9 10 11 12 13 14 15 16 71 18 19 20 21 22 23 25 24 62 27 28 29 30 31 32 60 61 8 36 37 38 39 40 41 69 70 17 45 46 47 48 49 50 79 78 26 63 64 65 66 67 68 42 43 44 56 73 72 59 76 75 53 51 52 54 55 74 57 58 77 33 34 35
p-4-6-2	11 9 37 14 12 40 5 3 31 20 18 46 23 21 49 17 15 43 2 6 34 26 0 28 8 24 52 38 36 64 41 39 67 32 30 58 47 45 73 50 48 76 44 42 70 29 33 61 53 27 55 35 51 79 65 63 10 68 66 13 59 57 4 74 72 19 77 75 22 71 69 16 56 60 7 80 54 1 62 78 25