Exact Synthesis of Ternary Reversible Functions using Ternary Toffoli Gates

Abhoy Kole*, P. Mercy Nesa Rani†, Kamalika Datta†, Indranil Sengupta‡ Rolf Drechsler§

* B. P. Poddar Institute of Management and Technology, Kolkata, India
Email: abhoy.kole@gmail.com

† National Institute of Technology Meghalaya, Shillong, India
Email: {mercyranip, kdatta}@nitm.ac.in

‡ Indian Institute of Technology, Kharagpur, India
Email: isg@iitkgp.ac.in

§ University of Bremen, Germany
Email: drechsler@informatik.uni-bremen.de

***Abstract*—Realization of logic functions using ternary reversible logic is known to require lesser number of lines as compared to conventional binary reversible logic. This aspect of ternary reversible logic has motivated researchers to explore various synthesis approaches in the past. Existing synthesis methods require additional lines (called ancilla lines) for synthesis, which is expensive from the quantum implementation point of view. There is no reported work for ternary reversible logic synthesis that require the minimum possible number of gates and also lines. This class of synthesis methods is called exact synthesis. In this paper two exact synthesis methods for ternary reversible logic have been proposed for the first time, one based on boolean satisfiability (SAT) and the other based on level-constrained heuristic search technique. A permutation representing a reversible ternary truth table is given as input, and a reversible circuit consisting of generalized ternary Toffoli gates that implements the permutation is obtained as output. Experimental studies have been carried out on various randomly generated ternary reversible functions.**

Keywords: Ternary reversible logic, exact synthesis, boolean satisfiability, heuristic search

I. Introduction

Research in the area of reversible logic has been motivated by Landauer [1] and Bennett [2], and has drawn the attention of researchers for more than one decade in the search of an alternate technology that can provide very low power consumption. A number of synthesis approaches have been proposed for reversible circuits, that map a given function specification to a cascade of reversible gates. These methods can be broadly classified as *Exact*, which guarantee minimum cost solutions but are computationally very complex, and *Non-exact*, which do not guarantee optimality but can perform synthesis in less amount of time. Methods for exact synthesis have been explored for binary reversible circuits [3], [4], [5], [6], [7] mainly to compare the quality of the solutions generated by non-exact synthesis algorithms with respect to the optimal ones. However, no such exact synthesis methods exist in the literature for ternary reversible logic.

The reversible gates as synthesized are typically mapped to gates from some well-known quantum gate library to evaluate the cost of implementation. Research in multivalued reversible and quantum computing have picked up recent years, because they promise low cost implementations both in terms of number of gates and lines [8], [9], [10], [11], [12], [13], [14], [15], [16]. The basic unit of information in a multivalued quantum system is called a *qudit*. For an n-valued quantum system, a qudit can be expressed as a linear superposition of the basis states $|0\rangle, |1\rangle, ...|n-1\rangle$. A qudit for $n = 3$ is referred to as a *qutrit*, which corresponds to a ternary quantum system with states $|0\rangle$, $|1\rangle$, and $|2\rangle$.

A number of works have been reported for the synthesis of logic functions using ternary reversible gates, which generates sub-optimal solutions and use various ternary reversible gates for synthesis. However, no exact synthesis method for ternary reversible circuits exists in the literature that guarantees minimum gate solutions. In this context, the present work proposes two exact synthesis approaches for ternary reversible circuits for the first time. The first approach is based on mapping the synthesis problem into a boolean 0 problem and then using a SAT solver to get the solution. The second approach is based on mapping the synthesis problem as a state-space search problem, and using level-constrained heuristic search to get the solution. Results show that the heuristic search based approach is better only for very small circuit instances as compared to the SAT-based approach.

The rest of the paper is organized as follows. Section II provides a background of ternary reversible logic and some existing synthesis approaches. Section III presents the SAT-based exact synthesis approach, while section IV discusses the heuristic search based approach. Section V presents the experimental results, while section VI concludes the paper.

II. Background

In this section we briefly introduce ternary reversible gates and some the variations proposed in the literature, and some existing synthesis approaches for ternary reversible logic.

A. Ternary Reversible Functions and Gates

Definition 1. *An n-input ternary reversible function f represents a bijection $f : T^n \to T^n$, where T denotes the set of ternary logic values* $\{0, 1, 2\}$.

A ternary reversible function can be realized as a cascade of ternary reversible gates. Some of the ternary reversible gates that have been proposed in the literature are Toffoli gate, controlled-NOT gate, and NOT gate as shown in Figure 1.

- A ternary Toffoli gate $T : \{c_1, c_2; t\}$ contains two control lines c_1 and c_2 and one target line t. The target line changes to t' $(= t \oplus_3 1)$ when both c_1 and c_2 are in state $|2\rangle$.
- A ternary controlled-NOT gate $CNOT : \{c; t\}$ contains one control line c and a target line t. Here also the target line changes to t' $(= t \oplus_3 1)$ when c is in state $|2\rangle$.
- A ternary NOT gate always changes $(t \oplus_3 1)$ the state of the target line t.

Different versions of Toffoli gates are also available. In the present work we consider the one proposed in [17].

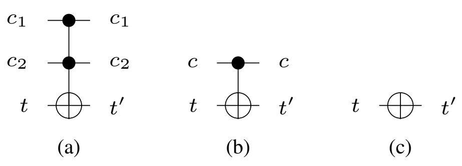


Fig. 1: Ternary reversible gates. (a) Toffoli gate, (b) controlled-NOT gate, (c) NOT gate

In the literature three kinds of conditions under which a Toffoli or CNOT gate triggers have been mentioned [10], [11], [18]: (i) when all the controls are in state $|2\rangle$, (ii) when all the controls are in state $|1\rangle$, and (iii) when one control is in state $|1\rangle$ and the other in state $|2\rangle$.

To evaluate the cost of implementation, reversible gates are realized using elementary quantum gates (each of unit cost). For conventional reversible logic we usually use the NCV quantum gate library [19]. For ternary reversible gates, one of the commonly used quantum elementary gate is the Muthukrishnan-Stroud (M-S) gate [20], which is a 2-qutrit gate MS(C;t), defined as follows:

$$\begin{aligned} t' &= Z(t), \text{ if } C = |2\rangle \\ &= t, \text{ otherwise} \end{aligned}$$

and depicted in Figure 2.

The state of the control line C remains unchanged. Z represents a unitary operation that can be one of +1, +2, 12, 01, and 02, with the corresponding mappings illustrated in Table I.

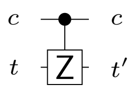


Fig. 2: Muthukrishnan-Stroud (MS) gate

TABLE I: Truth table of MS-gate

Target	t'				
t	Z(+1)	Z(+2)	Z(12)	Z(01)	Z(02)
$\|0\rangle$	$\|1\rangle$	$\|2\rangle$	$\|0\rangle$	$\|1\rangle$	$\|2\rangle$
$\|1\rangle$	$\|2\rangle$	$\|0\rangle$	$\|2\rangle$	$\|0\rangle$	$\|1\rangle$
$\|2\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|2\rangle$	$\|0\rangle$

A ternary Toffoli gate can be realized using M-S gates as shown in Figure 3, which only uses gates corresponding to $Z(+1)$ and $Z(+2)$.

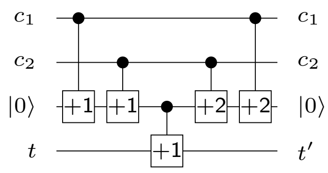


Fig. 3: Ternary Toffoli gate using M-S gates

B. Existing Works on Ternary Reversible Logic Synthesis

In this subsection, we discuss a few significant works related to ternary reversible logic synthesis. Broadly various ternary reversible synthesis methods can be classified into three main categories viz., group theory based synthesis, Ternary Galois Field Sum of Products (TGFSOP) based synthesis and some miscellaneous approaches (like, based on genetic algorithm).

In [10] Li et al. presented the synthesis of ternary reversible circuits with the minimum number of ancilla lines and garbage lines. They have converted conventional ternary logic circuits into ternary reversible logic circuits and the permutations are obtained from the truth table. The 3-cycle permutations are decomposed into the product of 0 3-cycles. Then the 0 3-cycles are synthesized using ternary NOT gate and ternary Toffoli gate. Here, ternary Toffoli gate fires when the control inputs are 2. The merit of this approach is the number of ancilla lines and garbage lines are less in the realization of ternary half and full adders as compared to [12], at the cost of an increase in the number of gates.

In [15], Mondal et al. presented the synthesis of balanced ternary reversible logic circuits. Balanced ternary reversible logic consists of 3 states, -1, 0 and $+1$. They have proposed balanced ternary NOT, CNOT and C^2NOT gates. The realizations for full adder, half adder, single-trit multiplier and double-trit multiplier circuits are shown. The hardware complexity of synthesis has been reduced significantly using this approach.

In [13] Khan et al. proposed constant ternary literals for the variables to represent functions. They have also proposed the composite ternary literals and reduced post literals. They used 16 Ternary Galois Field Expansions (TGFE) and three different types of ternary decision diagrams to realize ternary benchmarks for the first time.

In [21] Khanom et al. presented a GA based solution to realize ternary half adder using M-S gates. GA allows to find appropriate combination of the gates that realize ternary logic functions. Redundant gates presented in the synthesized circuit are eliminated using a post GA reduction process.

III. SAT Based Exact Synthesis

In this section we present a SAT based exact synthesis approach for ternary reversible circuits. This is basically an extension of existing SAT based exact synthesis approaches for binary reversible circuits to the ternary domain. The method exhaustively searches for an assignment of input variables satisfying constraints encoded in SAT formulation to realize a given function F_z using d number of Multiple Control Ternary Toffoli (MCTT) gates. In order to obtain minimal gate realization, the search starts with $d = 1$ and continues incrementing the value of d until the specification become satisfiable. The problem encoding is similar to the approach presented in [5], [6]. For the sake of completeness, we reintroduce all the constraints and extend them to the ternary domain.

a) **Encoding Toffoli gate**: For an n variable ternary reversible function F_z, a Toffoli gate $T(\mathbf{C}^k; \mathbf{t}^k)$ at depth k can be selected in $n.2^{n-1}$ ways where control ($\mathbf{C}^k$) and target ($\mathbf{t}^k$) lines can be selected 2^{n-1} and n ways respectively. The selection vectors for control and target lines are encoded as follows:

$$\mathbf{t}^k = (t^k_{\lceil \log_2 n \rceil} \dots t^k_1) \qquad \mathbf{C}^k = (c^k_{n-1} \dots c^k_1)$$

for $0 \le k < d$, where $\mathbf{t}^k$ represents an integer between 0 to $n-1$ in binary format, and $(t^k + l)$ mod n is a control line if $c^k_l = 1$.

b) **Input/Output constraints**: A ternary state z is realized as a pair of bits (x, y) representing the most and least significant bits respectively. For a given ternary reversible function F_z synthesized using d gates, the Boolean vectors representing the states of each circuit line are

$$X^k_i = (x^k_{i(n-1)} \dots x^k_{i0}) \qquad Y^k_i = (y^k_{i(n-1)} \dots y^k_{i0})$$

for $0 \le i < 3^n$ and $0 \le k \le d$, and the pair (X^k_i, Y^k_i) represents input, output, or intermediate ternary state vector Z^k_i for $k = 0$, $k = d$ or $0 < k < d$, respectively. The constraints for input and output of the truth table are set as:

$$\bigwedge_{i=0}^{3^n-1} X^0_i = i_x \wedge X^d_i = F_x(i) \wedge Y^0_i = i_y \wedge Y^d_i = F_y(i)$$

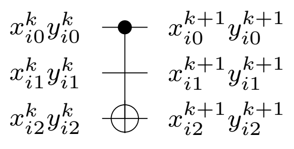
where the pairs (F_x, F_y) and (i_x, i_y) represent the MSBs and LSBs of the given function F_z and n-variable input i, respectively.

c) **Functional constraints**: Depending on the input Z^k_i (X^k_i, Y^k_i) of the k-th gate $T(\mathbf{C}^k; \mathbf{t}^k)$, the gate output Z^{k+1}_i (X^{k+1}_i, Y^{k+1}_i) is computed as

$$\bigwedge_{i=0}^{3^n-1} \bigwedge_{i=0}^{d-1} X^{k+1}_i = T_x(Z^k_i, \mathbf{C}^k, \mathbf{t}^k) \wedge Y^{k+1}_i = T_y(Z^k_i, \mathbf{C}^k, \mathbf{t}^k)$$

where the functions $T_x(Z^k_i, \mathbf{C}^k, \mathbf{t}^k))$ and $T_y(Z^k_i, \mathbf{C}^k, \mathbf{t}^k))$ together realizes the functionality of the MCTT gate $T(\mathbf{C}^k; \mathbf{t}^k)$ for the i-th truth table entry.

A MCTT gate $T(\mathbf{C}^k, \mathbf{t}^k)$ performs the operation $(Z^k_{it} \oplus_3 1)$ on the target line when all control lines are in state $|2\rangle$, i.e. $\bigwedge_{c \in C} Z^k_{ic} = 2 \Rightarrow \bigwedge_{c \in C} X^k_{ic} = 1 \wedge Y^k_{ic} = 0$. Figure 4a shows a ternary CNOT gate. When the control line is in state $|2\rangle$, the target line changes as shown in Figure 4b.



$x^k_{i0}y^k_{i0}$
$x^{k+1}_{i0}y^{k+1}_{i0}$
$x^k_{i1}y^k_{i1}$
$x^{k+1}_{i1}y^{k+1}_{i1}$
$x^k_{i2}y^k_{i2}$
$x^{k+1}_{i2}y^{k+1}_{i2}$

(a) CNOT

x^k_{i2}	y^k_{i2}	x^{k+1}_{i2}	y^{k+1}_{i2}
0	0	0	1
0	1	1	0
1	0	0	0
1	1	×	×

(b) $Z^k_{i2} \oplus_3 1$

Fig. 4: Operation of Ternary CNOT gate

Specifically, to realize a 3-variable ternary function, the following constraints for the k-th MCTT gate with $\mathbf{t}^k = (10)$ and $\mathbf{C}^k = (01)$ are added on i-th circuit instance corresponding to the i-th truth table entry.

$$\begin{aligned} \mathbf{t}^k = (10) \wedge \mathbf{C}^k = (01) \Rightarrow \quad & x^{k+1}_{i0} = x^k_{i0} \wedge x^{k+1}_{i1} = x^k_{i1} \\ & \wedge x^{k+1}_{i2} = \overline{x^k_{i2}} \wedge y^k_{i2} \wedge x^k_{i0} \\ & \wedge y^{k+1}_{i0} = y^k_{i0} \wedge y^{k+1}_{i1} = y^k_{i1} \\ & \wedge y^{k+1}_{i2} = \overline{x^k_{i2}} \wedge \overline{y^k_{i2}} \wedge x^k_{i0} \end{aligned}$$

d) **Target line constraints**: These constraints ensure that the target line ($\mathbf{t}^k$) never exceeds the number of circuit lines n:

$$\bigwedge_{k=0}^{d-1} \mathbf{t}^k < n$$

The formulation of the synthesis problem is illustrated by the following example.

Example 1. *Figure 5 shows the SAT formulation for the ternary function $F_z =$ (6 7 17 0 1 11 3 4 14 15 16 26 9 10 20 12 13 23 24 25 8 18 19 2 21 22 5) with $n = 3$ variables and depth $d = 3$. Each of $3^3 = 27$ truth table entries is represented by a circuit instance with input and output mapped to the corresponding truth table input and output. In each instance, all the 3 gates (marked with dashed rectangles) are defined by assigning values to their respective control ($\mathbf{C}^k$) and target ($\mathbf{t}^k$) lines.*

With all the above constraints defined appropriately, the synthesis flow can be stated as shown in Algorithm 1. One restriction of the SAT based approach is that it starts the search from $d = 1$ onwards. In other words, it will fail for an identity reversible function (i.e. $d = 0$), where zero number of gates are required. However, this is an extremal case that would show up with extreme rarity in practical scenarios.

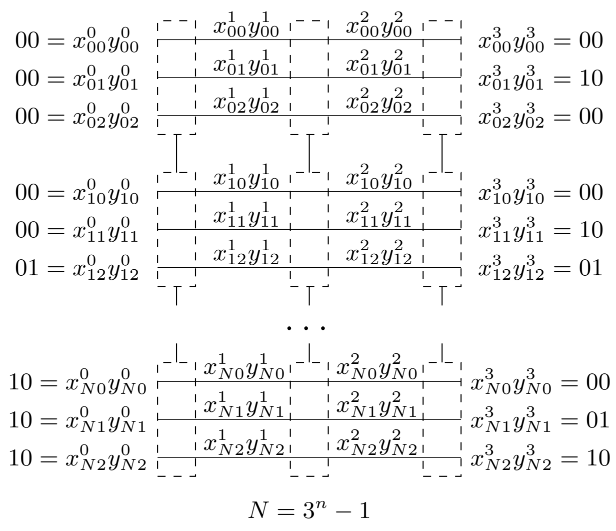


Fig. 5: Example SAT formulation

Algorithm 1: SAT-based Exact Synthesis

```
Input: Ternary reversible function $F_z$
Output: Sequence of MCTT gates that realize $F_z$
begin
    $found$ = false; $d$ = 1;
    while ($found$ = false) do
        $instance$ = $formulateProblem(F_z, d)$;
        $result$ = $callSolver(instance)$;
        if ($result$ is satisfiable)
            $asgn$ = $getAssignement()$;
            $circ$ = $createCircuit(asgn)$;
            $found$ = true;
        else
            $d = d + 1$;
        endif
    end
return $circ$;
```

IV. Heuristic Search Based Exact Synthesis

For an n-variable ternary reversible function, the input-output relationship can be specified as a permutation defined on the values $(0, 1, 2, \ldots, 3^{n-1})$. Every n-input MCTT gate G_i corresponds to a permutation $\Pi(G_i)$. A sequence of gates $G = \{G_1, G_2, \ldots, G_p\}$ implements the permutation $\Pi(G_1)$ o $\Pi(G_2)$ o $\cdots$ o $\Pi(G_p)$. Given a permutation Π_{given} to synthesize, the synthesis problem consists of finding a sequence of gates G that implements Π_{given}.

The synthesis problem can be mapped to a graph search problem, where each node of the graph represents a permutation. Every (directed) edge of the graph from v_i to v_j is labeled with a MCTT gate G_k that maps a permutation Π_{v_i} to (Π_{v_i} o $\Pi(G_k)$). Starting from an initial node that represents the identity permutation, the objective is to search for a path in the graph that leads to the desired permutation to synthesize.

For an n-input MCTT gate, the target can be placed in n different ways, while the controls can be placed in various ways on the $(n-1)$ other lines, resulting in $N = n2^{n-1}$ possibilities.

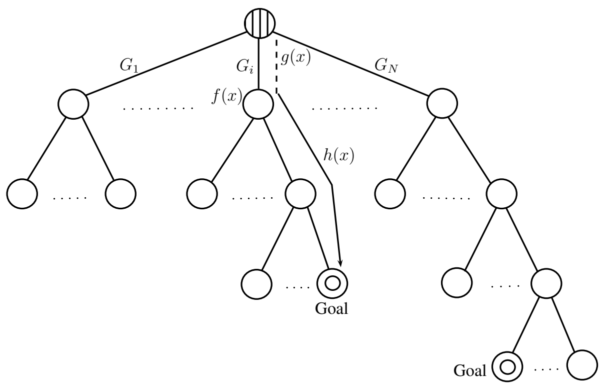


Fig. 6: Heuristic search for goal node

An n-input MCTT gate is encoded as an $\lceil log_2 n \rceil (n-1)$-bit number, where the first $\lceil log_2 n \rceil$ bits indicate the position of the target, while the last $(n-1)$ bits represent the positions of the control connections. In the search graph, from any node the number of outgoing edges will be N.

We have implemented the heuristic search algorithm A^* to explore the search graph. The cost of a node x is computed as $f(x) = g(x) + h(x)$, where $g(x)$ represents the cost of the current node x from the starting node, and $h(x)$ represents the heuristic function that gives an estimate of the cost from x to a goal node representing the desired permutation as shown in Figure 6. $g(x)$ is simply calculated as the number of edges traversed from the starting node (i.e. number of gates) .

In the implementation, $h(x)$ has been estimated by counting the number of mismatches in the permutation positions between the current node x and the goal node. Following the justification given in [22], for a particular bit position, if the bits change for $p/2^i$ positions, then $(i+1)$ is added to the heuristic estimate, where $p = 2^n$.

Algorithm 2 shows the steps of level-constrained A^*. The input provided is the desired permutation P, number of inputs n, and the maximum number of levels l_{max} up to which the search will proceed. The initial permutation Pi_{init} is inserted into a priority queue OPEN. The function add_to_OPEN(x) calculates the cost of a node x and enters it into OPEN. The function remove_from_OPEN() returns the node with the smallest cost from OPEN. The process is repeated until either the desired permutation is reached, or if the number of levels exceeds l_{max} or the time budget runs out.

Algorithm 2: Level Constrained A^* Synthesis

```
Input: Desired permutation P, number of inputs n, max. level $l_{max}$
Output: Sequence of MCTT gates that realize P
begin
    $found$ = 0;
    $\Pi_{init} = (0, 1, 2, \ldots, 3^n - 1)$
    $initial.perm$ = $\Pi_{init}$; $initial.level$ = 0;
    $add\_to\_OPEN$ ($initial$);
    $level$ = initial level;
    repeat
        $NODE$ = $remove\_from\_OPEN$();
        if ($NODE.perm$ = P) then
            $found$ = 1;
            Get solution by tracong path from NODE to S;
        else
            if ($NODE.level < level$) then
                $Q$ = Set of child nodes of NODE;
                foreach $q \in Q$ do
                    $child.perm = q$;
                    $child.level = NODE.level + 1$;
                    $add\_to\_OPEN$ ($child$);
                end
            endif
        endif
        $level = level + 1$;
    until (($found$ = 1) or ($level > l_{max}$));
end
```

In the algorithm, search proceeds with increasing values of *level*, starting from the initial node. If a solution is found for the first time at level i, it clearly means that there was no solution with levels less than i, and hence the solution obtained is exact (that is, minimal in number of gates).

V. Experimental Study

The exact synthesis tools based on A^* and SAT solver have been implemented in C and run on a core-i3 desktop with 2.4 GHz clock and 4 GB memory. The open access SAT solver MiniSAT [23] is used in the implementation.

To compare the performances of the two implementations, we have run the synthesis tools on a number of randomly generated ternary reversible functions in the form of permutations. The synthesis results are summarized in Table II, where results for up to 6 lines and 6 gates are shown. The first two columns give the name of the random permutation and the number of variables respectively. The permutation p-x-y-z indicates that the permutation corresponds to a randomly generated ternary gate netlist with x lines and y gates, where z is the instance of the random permutation generated. The next three columns give the number of gates G, the quantum cost in terms of the number of M-S gates required for implementation, and the run time in seconds for the SAT based implementation. The last three columns show the corresponding values for the A^* based implementation. The dashed ('-') entries indicate that results could not be obtained due to resource limitations.

The following can be concluded from the synthesis results.

a) Though the run time for the A^* based tool is less than that for the SAT based tool for several smaller permutations, it increases significantly for functions with larger number of variables and gates. For some of the permutations (e.g. p-4-5-1), the A^* tool ran out of memory and could not give any result.
b) The SAT based tool gave results for some larger functions that the A^* based tool failed to synthesize; however, the run time increases rapidly with increase in number of variables and gates.

TABLE II: Comparison of synthesis results

Permutations		SAT			A^*		
Name	n	G	$cost$	runtime	G	$cost$	runtime
p-2-3-1	2	3	3	0.01	3	3	0.00
p-2-3-2	2	3	3	0.01	3	3	0.00
p-2-4-1	2	4	4	0.02	4	4	0.00
p-2-4-2	2	4	4	0.02	4	4	0.00
p-2-5-1	2	5	5	0.01	5	5	0.00
p-2-5-2	2	5	5	0.02	5	5	0.00
p-2-6-1	2	5	5	0.03	5	5	0.00
p-2-6-2	2	6	6	0.03	6	6	0.00
p-3-3-1	3	3	3	0.05	3	3	0.00
p-3-3-2	3	3	11	0.04	3	11	0.00
p-3-4-1	3	4	4	0.07	4	4	0.04
p-3-4-2	3	4	12	0.05	4	12	0.04
p-3-5-1	3	5	5	0.11	5	5	0.22
p-3-5-2	3	5	13	0.08	5	13	0.19
p-3-6-1	3	6	6	0.14	6	6	1.25
p-3-6-2	3	6	10	0.15	6	10	3.20
p-4-3-1	4	3	11	0.17	3	11	0.12
p-4-3-2	4	3	11	0.16	3	11	0.21
p-4-4-1	4	4	12	0.28	4	12	2.75
p-4-4-2	4	4	12	0.28	4	12	1.29
p-4-5-1	4	5	13	0.44	-	-	-
p-4-5-2	4	5	17	0.39	5	17	43.13
p-5-3-1	5	3	15	0.74	3	15	5.40
p-5-3-2	5	3	11	0.70	3	11	7.91
p-5-4-1	5	4	16	1.12	-	-	-
p-5-4-2	5	4	12	1.17	-	-	-
p-5-5-1	5	5	17	2.82	-	-	-
p-5-5-2	5	5	13	2.33	-	-	-
p-5-6-1	5	6	14	5.10	-	-	-
p-5-6-2	5	6	22	5.73	-	-	-
p-6-3-1	6	3	15	3.93	-	-	-
p-6-3-2	6	3	19	3.37	-	-	-
p-6-4-1	6	4	20	5.31	-	-	-
p-6-4-2	6	4	24	5.91	-	-	-
p-6-5-1	6	5	25	18.73	-	-	-
p-6-5-2	6	5	29	19.72	-	-	-
p-6-6-1	6	6	42	31.66	-	-	-
p-6-6-2	6	6	54	33.69	-	-	-

n: no. of lines G: no. of MCTT gates $cost$: no. of M-S gates

VI. Conclusion

The problem of exact synthesis of ternary reversible circuits have been addressed for the first time in this paper. Two different exact synthesis methods have been presented, one based on the A^* heuristic search algorithm with level constraints, and the other using a SAT solver. Synthesis results with up to 6 variables have been reported. As expected, the run times increase rapidly as the number of variables or required number of gates increase. These approaches can provide a benchmark against with the performances of other non-exact synthesis techniques can be compared.

Non-availability of ternary benchmark has restricted the experimental analysis in the present work. As a future work, identifying a set of ternary benchmark functions from their binary equivalents shall be taken up.

References

[1] R. Landauer. Irreversibility and heat generation in the computing process. *Journal of IBM Research and Development*, (5):183–191, 1961.

[2] C. H. Bennett. Logical reversibility of computation. *Journal of IBM Research and Development*, 17(6):525–532, November 1973.

[3] R. Wille and D. Große. Fast exact Toffoli network synthesis of reversible logic. In *Intl. Conf. on CAD*, pages 60–64, 2007.

[4] D. Große, X. Chen, G. W. Dueck, and R. Drechsler. Exact SAT-based Toffoli network synthesis. In *ACM Great Lakes Symposium on VLSI*, pages 96–101, 2007.

[5] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares. In *Intl Symp. on Multi-Valued Logic*, pages 214–219, May 2008.

[6] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact multiple-control Toffoli network synthesis with SAT techniques. *IEEE Transactions on CAD*, 28(5):703–715, May 2009.

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

[8] M. H. A. Khan. GFSOP-based ternary quantum logic synthesis. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 7797- 16 V.2, pages 1–15, 2010.

[9] M. H. A. Khan. GFSOP based ternary quantum logic synthesis. In *In Proc. Optics and Photonics for Information Processing IV*, pages 1–15, 2010.

[10] 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.

[11] G Yang, X. Song, M. Perkowski, and J. Wu. Realizing ternary quantum switching networks without ancilla bits. *Journal of Physics A: Mathematical and General*, 38:1–10, 2005.

[12] S. Mandal, A. Chakrabarti, and S. Sur-Kolay. Synthesis techniques for ternary quantum logic. In *Proc. 41st Intl. Symp. on Multiple Valued Logic (ISMVL)*, pages 218–223, 2011.

[13] M. H. A. Khan, M. 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.

[14] F. S. Khan and M. Perkowski. Synthesis of ternary quantum logic circuits by decomposition. *Phys. Rev. A*, 62(5):052309/1–8, 2005.

[15] B. Mondal, P. Sarkar, P. K. Saha, and S. Chakraborty. Synthesis of balanced ternary reversible logic circuit. In *Proc. 43rd Intl. Symp. on Multiple Valued Logic*, pages 334–349, 2013.

[16] S. Basu, S. B. Mandal, A. Chakrabarty, and S. Sur-Kolay. An efficient synthesis method for ternary reversible logic. In *Intl. Symp. on Circuits and Systems (ISCAS)*, pages 2306–2309, 2016.

[17] M. H. A. Khan. Design of reversible/quantum ternary multiplexer and demultiplexer. *Engineering letters*, 13(2):65–69, 2006.

[18] P. M. N. Rani, A. Kole, K. Datta, and A. Chakrabarty. Realization of ternary reversible circuits using improved gate library. In *Proc. 6th Intl. Conf. on Advances in Computing and Communication*, pages 153–160, 2016.

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

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

[21] R. Khanom, T. Kamal, and M. H. A. Khan. Genetic algorithm based synthesis of ternary reversible/quantum circuit. In *Proc. 11th Intl. Conf. on Computer and Information Technology*, pages 270–275, 2008.

[22] K. Datta, G. Rathi, I. Sengupta, and H. Rahaman. Exact synthesis of reversible circuits using A* algorithm. *Journal of Institution of Engineers (Series B)*, 96(2):121–130, 2015.

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

APPENDIX: THE RANDOM PERMUTATIONS

Some of the random permutations for which synthesis results have been reported in Table II are given below.

p-2-3-1:	7 8 6 1 2 0 4 5 3
p-2-3-2:	3 4 6 7 8 2 0 1 5
p-2-4-1:	7 2 6 1 5 0 4 8 3
p-2-4-2:	3 4 7 8 6 2 0 1 5
p-2-5-1:	8 2 7 1 5 0 4 6 3
p-2-5-2:	6 7 1 2 0 5 3 4 8
p-2-6-1:	2 5 1 4 8 3 7 0 6
p-2-6-2:	7 8 2 0 1 3 4 5 6
p-3-3-1:	0 1 20 3 4 23 15 16 8 9 10 2 12 13 5 24 25 17 18 19 11 21 22 14 6 7 26
p-3-3-2:	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 21 22 26 24 25 20 19 23 18
p-3-4-1:	0 1 20 3 4 23 16 17 6 9 10 2 12 13 5 25 26 15 18 19 11 21 22 14 7 8 24
p-3-4-2:	1 2 0 4 5 3 7 8 6 10 11 9 13 14 12 16 17 15 22 23 24 25 26 18 20 21 19
p-3-5-1:	9 10 2 12 13 5 25 26 15 18 19 11 21 22 14 7 8 24 0 1 20 3 4 23 16 17 6
p-3-5-2:	10 11 9 13 14 12 16 17 15 19 20 18 22 23 21 25 26 24 4 5 6 7 8 0 2 3 1
p-3-6-1:	21 22 14 24 25 17 19 20 9 3 4 23 6 7 26 1 2 18 12 13 5 15 16 8 10 11 0
p-3-6-2:	10 14 9 13 17 12 11 15 16 19 23 18 22 26 21 20 24 25 4 8 6 7 2 0 5 3 1
p-4-3-2:	27 28 29 30 31 32 33 34 62 36 37 38 39 40 41 42 43 71 45 46 47 48 49 50 51 52 80 54 55 56 57 58 59 60 61 17 63 64 65 66 67 68 69 70 26 72 73 74 75 76 77 78 79 8 0 1 11 3 4 14 6 7 35 9 10 20 12 13 23 15 16 44 18 19 2 21 22 5 24 25 53
p-4-4-1:	0 1 2 3 4 5 15 16 17 9 10 11 12 13 14 79 80 78 45 46 47 48 49 50 6 7 8 27 28 29 30 31 32 42 43 44 36 37 38 39 40 41 25 26 24 72 73 74 75 76 77 33 34 35 54 55 56 57 58 59 69 70 71 63 64 65 66 67 68 52 53 51 18 19 20 21 22 23 60 61 62
p-4-4-2:	27 28 29 30 31 32 33 34 60 36 37 38 39 40 41 42 43 69 45 46 47 48 49 50 51 52 78 55 56 54 58 59 57 61 62 17 64 65 63 67 68 66 70 71 26 73 74 72 76 77 75 79 80 8 0 1 11 3 4 14 6 7 35 9 10 20 12 13 23 15 16 44 18 19 2 21 22 5 24 25 53
p-4-5-1:	0 1 2 3 4 5 15 16 17 9 10 11 12 13 14 80 78 79 46 47 45 49 50 48 6 7 8 27 28 29 30 31 32 42 43 44 36 37 38 39 40 41 26 24 25 73 74 72 76 77 75 33 34 35 54 55 56 57 58 59 69 70 71 63 64 65 66 67 68 53 51 52 19 20 18 22 23 21 60 61 62
p-4-5-2:	3 4 5 6 7 8 0 1 2 12 13 14 15 16 17 9 10 11 48 49 50 51 52 53 45 46 47 30 31 32 33 34 35 27 28 29 39 40 41 42 43 44 36 37 38 75 76 77 78 79 80 72 73 74 58 62 57 61 56 60 55 59 54 67 71 66 70 65 69 64 68 63 22 26 21 20 24 25 19 23 18
p-5-3-1:	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 19 20 18 22 23 21 25 26 24 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 46 47 45 49 50 48 52 53 51 54 55 56 57 58 59 60 61 143 63 64 65 66 67 68 69 70 152 73 74 72 76 77 75 79 80 159 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 100 101 99 103 104 102 106 107 105 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 127 128 126 130 131 129 133 134 132 135 136 137 138 139 140 141 142 224 144 145 146 147 148 149 150 151 233 154 155 153 157 158 156 160 161 240 162 163 164 165 166 167 177 178 179 171 172 173 174 175 176 187 188 186 181 182 180 184 185 183 168 169 170 189 190 191 192 193 194 204 205 206 198 199 200 201 202 203 214 215 213 208 209 207 211 212 210 195 196 197 216 217 218 219 220 221 231 232 71 225 226 227 228 229 230 241 242 78 235 236 234 238 239 237 222 223 62
p-5-3-2:	3 4 5 6 7 8 0 1 29 12 13 14 15 16 17 9 10 38 21 22 23 24 25 26 18 19 47 30 31 32 33 34 35 27 28 65 39 40 41 42 43 44 36 37 74 48 49 50 51 52 53 45 46 56 57 58 59 60 61 62 63 64 2 66 67 68 69 70 71 72 73 11 75 76 77 78 79 80 54 55 20 84 85 86 87 88 89 81 82 110 93 94 95 96 97 98 90 91 119 102 103 104 105 106 107 99 100 128 111 112 113 114 115 116 108 109 146 120 121 122 123 124 125 117 118 155 129 130 131 132 133 134 126 127 137 138 139 140 141 142 143 144 145 83 147 148 149 150 151 152 153 154 92 156 157 158 159 160 161 135 136 101 165 166 167 168 169 170 162 163 191 174 175 176 177 178 179 171 172 200 183 184 185 186 187 188 180 181 209 192 193 194 195 196 197 189 190 227 201 202 203 204 205 206 198 199 236 210 211 212 213 214 215 207 208 218 219 220 221 222 223 224 225 226 164 228 229 230 231 232 233 234 235 173 237 238 239 240 241 242 216 217 182
p-5-4-1:	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 19 20 18 22 23 21 25 26 24 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 46 47 45 49 50 48 52 53 51 63 64 65 66 67 68 69 70 152 72 73 74 75 76 77 78 79 161 55 56 54 58 59 57 61 62 141 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 100 101 99 103 104 102 106 107 105 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 127 128 126 130 131 129 133 134 132 144 145 146 147 148 149 150 151 233 153 154 155 156 157 158 159 160 242 136 137 135 139 140 138 142 143 222 162 163 164 165 166 167 177 178 179 171 172 173 174 175 176 187 188 186 181 182 180 184 185 183 168 169 170 189 190 191 192 193 194 204 205 206 198 199 200 201 202 203 214 215 213 208 209 207 211 212 210 195 196 197 225 226 227 228 229 230 240 241 80 234 235 236 237 238 239 223 224 60 217 218 216 220 221 219 231 232 71
p-5-4-2:	3 4 5 7 8 6 0 1 29 12 13 14 16 17 15 9 10 38 21 22 23 25 26 24 18 19 47 30 31 32 34 35 33 27 28 65 39 40 41 43 44 42 36 37 74 48 49 50 52 53 51 45 46 56 57 58 59 61 62 60 63 64 2 66 67 68 70 71 69 72 73 11 75 76 77 79 80 78 54 55 20 84 85 86 88 89 87 81 82 110 93 94 95 97 98 96 90 91 119 102 103 104 106 107 105 99 100 128 111 112 113 115 116 114 108 109 146 120 121 122 124 125 123 117 118 155 129 130 131 133 134 132 126 127 137 138 139 140 142 143 141 144 145 83 147 148 149 151 152 150 153 154 92 156 157 158 160 161 159 135 136 101 165 166 167 169 170 168 162 163 191 174 175 176 178 179 177 171 172 200 183 184 185 187 188 186 180 181 209 192 193 194 196 197 195 189 190 227 201 202 203 205 206 204 198 199 236 210 211 212 214 215 213 207 208 218 219 220 221 223 224 222 225 226 164 228 229 230 232 233 231 234 235 173 237 238 239 241 242 240 216 217 182
p-5-5-1:	0 1 11 3 4 14 6 7 17 9 10 20 12 13 23 15 16 26 19 2 18 22 5 21 25 8 24 27 28 38 30 31 41 33 34 44 36 37 47 39 40 50 42 43 53 46 29 45 49 32 48 52 35 51 63 64 74 66 67 77 69 70 161 72 73 56 75 76 59 78 79 143 55 65 54 58 68 57 61 71 141 81 82 92 84 85 95 87 88 98 90 91 101 93 94 104 96 97 107 100 83 99 103 86 102 106 89 105 108 109 119 111 112 122 114 115 125 117 118 128 120 121 131 123 124 134 127 110 126 130 113 129 133 116 132 144 145 155 147 148 158 150 151 242 153 154 137 156 157 140 159 160 224 136 146 135 139 149 138 142 152 222 162 163 173 165 166 176 177 178 188 171 172 182 174 175 185 187 170 186 181 164 180 184 167 183 168 169 179 189 190 200 192 193 203 204 205 215 198 199 209 201 202 212 214 197 213 208 191 207 211 194 210 195 196 206 225 226 236 228 229 239 240 241 62 234 235 218 237 238 221 223 233 60 217 227 216 220 230 219 231 232 80
p-5-5-2:	30 31 32 34 35 33 27 28 56 39 40 41 43 44 42 36 37 65 48 49 50 52 53 51 45 46 74 57 58 59 61 62 60 54 55 11 66 67 68 70 71 69 63 64 20 75 76 77 79 80 78 72 73 2 3 4 5 7 8 6 9 10 29 12 13 14 16 17 15 18 19 38 21 22 23 25 26 24 0 1 47 111 112 113 115 116 114 108 109 137 120 121 122 124 125 123 117 118 146 129 130 131 133 134 132 126 127 155 138 139 140 142 143 141 135 136 92 147 148 149 151 152 150 144 145 101 156 157 158 160 161 159 153 154 83 84 85 86 88 89 87 90 91 110 93 94 95 97 98 96 99 100 119 102 103 104 106 107 105 81 82 128 192 193 194 196 197 195 189 190 218 201 202 203 205 206 204 198 199 227 210 211 212 214 215 213 207 208 236 219 220 221 223 224 222 216 217 173 228 229 230 232 233 231 225 226 182 237 238 239 241 242 240 234 235 164 165 166 167 169 170 168 171 172 191 174 175 176 178 179 177 180 181 200 183 184 185 187 188 186 162 163 209
p-5-6-1:	0 1 11 3 4 14 6 7 17 9 10 18 12 13 21 15 16 24 20 2 19 23 5 22 26 8 25 27 28 38 30 31 41 33 34 44 36 37 45 39 40 48 42 43 51 47 29 46 50 32 49 53 35 52 66 67 75 69 70 159 63 64 72 76 77 59 79 80 143 73 74 56 58 68 57 61 71 141 55 65 54 81 82 92 84 85 95 87 88 98 90 91 99 93 94 102 96 97 105 101 83 100 104 86 103 107 89 106 108 109 119 111 112 122 114 115 125 117 118 126 120 121 129 123 124 132 128 110 127 131 113 130 134 116 133 147 148 156 150 151 240 144 145 153 157 158 140 160 161 224 154 155 137 139 149 138 142 152 222 136 146 135 162 163 173 165 166 176 168 169 179 171 172 180 174 175 183 177 178 186 182 164 181 185 167 184 188 170 187 189 190 200 192 193 203 195 196 206 198 199 207 201 202 210 204 205 213 209 191 208 212 194 211 215 197 214 228 229 237 231 232 78 225 226 234 238 239 221 241 242 62 235 236 218 220 230 219 223 233 60 217 227 216
p-5-6-2:	30 31 32 34 44 33 27 28 29 39 40 41 43 53 42 36 37 38 48 49 50 52 35 51 45 46 74 57 58 59 61 71 60 54 55 56 66 67 68 70 80 69 63 64 65 75 76 77 79 62 78 72 73 2 3 4 5 7 17 6 9 10 11 12 13 14 16 26 15 18 19 20 21 22 23 25 8 24 0 1 47 111 112 113 115 125 114 108 109 110 120 121 122 124 134 123 117 118 119 129 130 131 133 116 132 126 127 155 138 139 140 142 152 141 135 136 137 147 148 149 151 161 150 144 145 146 156 157 158 160 143 159 153 154 83 84 85 86 88 98 87 90 91 92 93 94 95 97 107 96 99 100 101 102 103 104 106 89 105 81 82 128 192 193 194 196 206 195 189 190 191 201 202 203 205 215 204 198 199 200 210 211 212 214 197 213 207 208 236 219 220 221 223 233 222 216 217 218 228 229 230 232 242 231 225 226 227 237 238 239 241 224 240 234 235 164 165 166 167 169 179 168 171 172 173 174 175 176 178 188 177 180 181 182 183 184 185 187 170 186 162 163 209