2015 IEEE 45th International Symposium on Multiple-Valued Logic

Fredkin-Enabled Transformation-based Reversible Logic Synthesis

Mathias Soeken[1,3] Anupam Chattopadhyay[2]
[1] Department of Mathematics and Computer Science, University of Bremen, Germany
[2] Nanyang Technological University, Singapore
[3] Cyber-Physical Systems, DFKI GmbH, Bremen, Germany

***Abstract*—Reversible computation is gaining increasing relevance in the context of several post-CMOS technologies, the most prominent of those being quantum computing. The problem of implementing a given Boolean function using a set of elementary reversible logic gates is known as reversible logic synthesis. Due to the presumed hardness of the reversible circuit synthesis problem, different heuristics have been proposed in the literature to reduce quantum cost (QC), gate count, and logical depth without using ancilla lines. Interestingly, none of these heuristics systematically utilize Fredkin gates. In this paper, we demonstrate, both theoretically and empirically, that accommodating Fredkin gates significantly improves the performance of reversible circuits.**

I. Introduction

Reversible logic synthesis is an important research area to address the synthesis issues of emerging nano-technologies and post-CMOS technologies, where the primitive computing elements are based on reversible logic gates. Besides quantum computing, several classical computing problems, where gain-free elementary computing structures are mandatory (e.g., photonic computing and encoder-decoder circuits) also require reversible circuits.

Due to the difficulty of scaling up practical quantum circuits for a large number of qubits, an important research problem is to perform ancilla-free reversible logic synthesis, within which we limit the scope of the current paper. For a complete survey of reversible logic synthesis methods, readers are kindly referred to [1].

Multiple synthesis methods have been proposed to perform what can be grossly classified as first, *optimal and non-scalable methods* [2], [3] and second, *heuristic methods* [4], [5]. Heuristic methods can be further classified based on their input Boolean function representation format such as truth-tables, decision diagrams, and permutations. Ancilla-free reversible logic synthesis, so far, is reported for truth-table based representation [4], binary decision diagrams [6], quantum multiple-valued decision diagrams [7], and permutations [8]. Interestingly, none of these approaches include Fredkin gate in the synthesis flow, rather it is used as a post-synthesis optimization [9], [10].

Let us consider a simple synthesis scenario of the prominent transformation-based method, known as MMD [4], after its authors. MMD proceeds in a row-wise fashion altering each bit in the output, which needs to be matched with the corresponding input bit. Assuming we can use a Fredkin gate successfully, instead of two Toffoli gates, one gate suffices to match two bits. This allows a straightforward reduction of 1 reversible gate. Corresponding *reduction* in QC, assuming both Toffoli and Fredkin gates employ n control lines, is

$$24(n-3).$$

We refer to QC as the T-depth of the circuit according to [11] which can be expanded for multiple-controlled Toffoli gates using [12]. The above simple formulation provides a positive reduction, i.e., an improved QC when $n \geq 4$. This observation also matches with the experiments done for template-based post-processing schemes [9], where multiple templates with swap gates are used. Another recent work attempting inclusion of Fredkin gates in transformation-based synthesis is proposed at [5], which reported improved performance for selected benchmark circuits. However, a detailed experimental study with large benchmark circuits as well as theoretical analysis are missing so far. Both of these are addressed in this paper.

The rest of the paper is organized as following. The following Sections II and III provide the background on reversible logic synthesis in general and transformation-based reversible logic synthesis in particular. Section IV introduces our key idea of Fredkin-enabled synthesis. Sections V and VI provide theoretical and experimental results supporting the presented technique. The paper is concluded and future work is outlined in Section VII.

II. Preliminaries

A Boolean function f is of the form $f : \{0,1\}^n \rightarrow \{0,1\}$. The output of the Boolean function f can be represented as a string of size 2^n consisting of ones and zeros. It can also be represented as a multivariate polynomial over $\mathbf{GF}(2)$. This polynomial can be expressed as a exclusive disjunction (EXOR) of a constant a_0 and one or more conjunctions of the function argument. This is called the Exclusive Sum-Of-Product (ESOP) representation. A less general representation of the ESOP form is known as the Algebraic Normal Form (ANF). The general ANF for a function $f(x_1,..,x_n)$ over n-variables can be written as,

$$\begin{aligned} f(x_1,\dots,x_n) =& a_0 \oplus a_1x_1 \oplus \cdots \oplus a_ix_i \oplus \cdots \oplus a_nx_n \\ & \oplus \cdots \oplus a_{1,2,\dots,n}x_1x_2\cdots x_n \end{aligned} \quad (1)$$

Reversible and irreversible Boolean functions: An n-variable vectorial Boolean function is *reversible* if all its output patterns map uniquely to an input pattern and vice versa; otherwise it is called *irreversible*. It can be expressed as an n-input, n-output bijection or alternatively, as a Boolean permutation function over the truth value set $\{0,1,\dots,2^{n-1}\}$.

An irreversible Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ can also be made reversible with the help of extra constant-initialized input lines known as *ancilla*. If those lines are not restored to their original constant values at the end of the computation, those are termed as *garbage* lines.

Reversible logic gates: A reversible gate library is a smallest complete set of reversible gates which can be used to build an arbitrary reversible circuit. Prominent reversible logic gates are NOT, Toffoli, Fredkin and in general, unitary transformations or rotations of qubits in the Bloch sphere. Any complex reversible gate can be decomposed in terms of smaller elementary gates [12].

III. Transformation-based Synthesis

MMD, a prominent transformation-based reversible logic synthesis, introduced in [4] and improved in [9], [5] served as the key benchmark synthesis technique for all the subsequent reversible logic synthesis approaches, in particular those which need zero ancilla lines. For the sake of completeness, we briefly review MMD in this section. To keep this discussion simple, optimizations of MMD (including bi-directional synthesis) are skipped here. However, these are included in our implementation and experimental benchmarking.

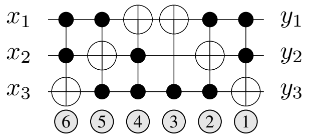

Fig. 1. Resulting circuit from example in Table I

Algorithm T (*Transformation-based synthesis*). Given a truth table as a list of input-output mappings $(i^{(1)}, o^{(1)}), (i^{(2)}, o^{(2)}), \dots, (i^{(2^n)}, o^{(2^n)})$, this algorithm finds a circuit by prepending gates to the empty circuit until the truth table has been transformed to match the identity function. The truth table is sorted by its input patterns.

T1. [Initialize.] Set $j \leftarrow 1$.

T2. [Create masks.] Set $p \leftarrow (i^{(j)} \oplus o^{(j)})$ & $i^{(j)}$, $q \leftarrow (i^{(j)} \oplus o^{(j)})$ & $o^{(j)}$, and $m \leftarrow o^{(j)}$.

T3. [Change 0's to 1's.] For each k such that $p_k = 1$, prepend a Toffoli gate with target on k and controls according to the bits that are set in m, also flip $o_k^{(\ell)}$ for all ℓ such that $m \subseteq i^{(\ell)}$. Set $m_k \leftarrow 1$.

T4. [Change 1's to 0's.] For each k such that $q_k = 1$, prepend a Toffoli gate with target on k and controls according to the bits that are set in m, also set $m_k \leftarrow 0$, also flip $o_k^{(\ell)}$ for all ℓ such that $m \subseteq i^{(\ell)}$.

T5. [Done?] If $j = 2^n$, terminate, otherwise set $j \leftarrow j+1$ and return to step 2. ▮

We visit each truth table entry in order using the loop variable j. For each row gates are inserted to match the output pattern with the input pattern such that previous entries are not modified. For this purpose, first bits that are 0 in the output but 1 in the input are changed (their indexes are stored in the mask p), and afterwards 1's are changed to 0's (their indexes are stored in the mask q). The control lines of the gates are assigned according to the mask m: We add a control line to line k, if and only if $m_k = 1$.

Note that several optimizations can be incorporated into the algorithm which we omitted to keep its description simple. As an example, for the first row the insertion of NOT gates is sufficient since there are no previous rows that may be changed. This step is explicitly described in the original work [4]. Furthermore, before entering steps 2–4 one can check, whether the input pattern matches already the output pattern. (Note that in this case, both p and q are 0 and therefore no gates are added in steps 3 and 4.) For each row $\nu(p \mid q)$ Toffoli gates are prepended to the circuit, where ν denotes the sideways sum.

Example 1: An example application of Algorithm T for the reversible function '*miller*' is illustrated by means of Table I. The initial truth table is represented by the columns labeled $x_1x_2x_3$ and $y_1y_2y_3$. Six steps need to be applied which consecutively lead to new functions $y_1^i y_2^i y_3^i$ until eventually $y_1^6 y_2^6 y_3^6$ represents the identity function. The current considered row is marked by '$\triangleright$' and bits affected by the gate operation are highlighted in blue. The resulting circuit is depicted in Fig. 1.

IV. Optimized Synthesis using Fredkin Gates

We propose to extend Algorithm T such that Fredkin gates are explicitly considered, since they are able to modify two bits at the same time. For this purpose the following steps are added after step 2 in Algorithm T.

T2a. [Loop over k and l.] For all k, l such that $p_k = 1$ and $q_l = 1$, perform step 2b.

T2b. [Add Fredkin gate.] Set $m' \leftarrow m$, $m'_k \leftarrow 0$, and $m'_l \leftarrow 0$. If m' is a valid mask, prepend a Fredkin gate with targets on k and l and controls according to the bits that are set in m'. Also, set $p_k \leftarrow 0$, $q_l \leftarrow 0$, $m_k \leftarrow 1$, and $m_l \leftarrow 0$.

We loop over all pairs k and l such that k refers to a position in the output pattern in which a 0 needs to be changed to a 1 and l refers to a position in the output pattern in which a 1 needs to be changed to a 0. If these two bits can be changed using a Fredkin gate, only one gate instead of two needs to be added to the circuit. We can only add a Fredkin gate if the mask m' allows for a valid set of control lines, i.e. a gate application does not changed previous input-output mappings. There are different strategies to check whether m' is a valid mask. A simple strategy is to check whether it is lexicographically larger than the input of the current row, i.e.

$$m' > i^{(j)}.$$

If that is the case, no previous row can be modified by the gate application. This strategy is not exhaustive. A Fredkin gate may change a previous row, however, it may swap two 0's or two 1's. In this case, the previous rows are not modified either. This can be checked by the following condition:

TABLE I. EXAMPLE APPLICATION OF THE MMD TRANSFORMATION-BASED SYNTHESIS APPROACH

		$\xrightarrow{1}$	$\xrightarrow{2}$	$\xrightarrow{3}$	$\xrightarrow{4}$	$\xrightarrow{5}$	$\xrightarrow{6}$
$x_1x_2x_3$	$y_1y_2y_3$	$y_1^1y_2^1y_3^1$	$y_1^2y_2^2y_3^2$	$y_1^3y_2^3y_3^3$	$y_1^4y_2^4y_3^4$	$y_1^5y_2^5y_3^5$	$y_1^6y_2^6y_3^6$
000	000	000	000	000	000	000	000
001	▷110	▷111	▷101	001	001	001	001
010	010	010	010	010	010	010	010
011	011	011	011	▷111	011	011	011
100	100	100	100	100	100	100	100
101	101	101	111	011	▷111	101	101
110	001	001	001	101	101	▷111	110
111	111	110	110	110	110	110	111

TABLE II. EXAMPLE APPLICATION OF THE MMD TRANSFORMATION-BASED SYNTHESIS APPROACH USING FREDKIN GATES

		$\xrightarrow{1}$	$\xrightarrow{2}$	$\xrightarrow{3}$	$\xrightarrow{4}$
$x_1x_2x_3$	$y_1y_2y_3$	$y_1^1y_2^1y_3^1$	$y_1^2y_2^2y_3^2$	$y_1^3y_2^3y_3^3$	$y_1^4y_2^4y_3^4$
000	000	000	000	000	000
001	▷110	▷101	001	001	001
010	010	010	010	010	010
011	011	011	▷111	011	011
100	100	100	100	100	100
101	101	110	110	▷110	101
110	001	001	101	101	110
111	111	111	011	111	111

$$(m' > i^{(j)}) \vee \bigwedge_{\substack{r\leq j \\ m' \& i^{(r)} = m'}} \left(o_k^{(r)} = o_l^{(r)}\right),$$

i.e. either m' is lexicographically larger than the currently considered input pattern $i^{(j)}$ (such as in the simple case) or for all previous rows r that do match the mask, i.e. m' & $i^{(r)} = m'$, the output bits at the positions k and l must be equal.

Example 2: When applying the extensions discussed in this section to the function from Example 1, we obtain the circuit in Fig. 2. The transformations are given in Table I. Note that the exhaustive mask validity checking has been used. Its effect can be observed from the last column, in which previous rows are affected but their values are not changed.

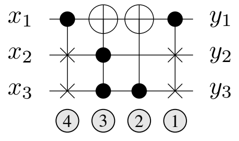

Fig. 2. Resulting circuit from example in Table II

V. THEORETICAL RESULTS

Despite significant works on the practical reversible logic synthesis, there have been very few notable attempts [13], [14], [15] to identify the theoretical bounds of different approaches, which, if determined, would serve as the key indicator of strength/weaknesses. For theoretical upper bound determination, we resort to the following reversible logic gate libraries:

- Multiple-Controlled Toffoli (MCT)
- Mixed-Polarity Multiple-Controlled Toffoli (MPMCT)
- Mixed-Polarity Multiple-Controlled Toffoli-Fredkin (MPMCF)

We do not consider the extension of bi-directional synthesis in the following discussions.

A. Theoretical Upper Bound for MMD using the MPMCT Library

The theoretical upper bound on the gate counts for MMD is determined for the MCT library in [13]. There, the authors also identified circuits achieving this bound. For an n-variable reversible Boolean function, the bound is

$$(n-1)2^n + 1 \tag{2}$$

However, in view of recent results, deploying the MPMCT library significantly improves quantum cost and thus, has been adopted in all synthesis methods [7], [6], [16]. Therefore, we first present some results on the theoretical upper bound of MMD using MPMCT. This is followed by the analysis for MPMCF library.

Lemma 5.1: The worst-case gate count for MMD [4] using the MPMCT library is $(n-1)2^n + 1$.

Proof: In MMD, when the value of the binary expression of the first output row is $(2^n - 1)$, it will require the maximum number of n gates to flip all the n bits. After matching the first output row, assume the second output row is $(2^n - 2)$, then n gates are needed as well. In matching the upper half of the input-output table, n gates are assumed to be needed for each row. From the $2^{n-1} + 1$-th output row, $n - 1$ gates are needed since the MSB is already fixed. Proceeding in this manner, the MMD upper bound is obtained.

The aforementioned derivation [13] does not detail anything about the control. It is assumed that for every row, there will be an available set of *positive* control signals that alter the target bit without affecting previously matched rows. Implicitly, this is guaranteed since, for an unmatched row, it cannot have same number of 1s in the same bit positions, compared to the matched rows. Arguing along the same lines, it can be deduced that MPMCT library does not reduce the number of gates, since the gate count is linked to the number of target bits and not the number or the polarity of the controls. Hence, the upper bound on gate count for MMD using MPMCT is $(n-1)2^n + 1$. ■

From the fact that the QC of a Toffoli gate with n control lines is $24(n-3)$ [11], [12], we obtain the following corollary.

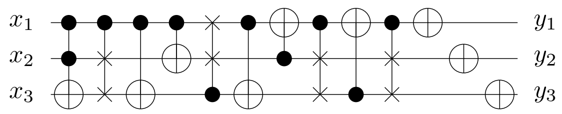

Fig. 3. Worst-case circuit for fredkin-enabled MMD using MPMCF Library

Corollary 5.2: The upper bound of QC for MMD [4] using the MPMCT library is $((n-1)2^n+1)\cdot 24(n-3)$.

B. Theoretical Upper Bound for Proposed Fredkin-enabled MMD using the MPMCF Library

The key idea for deriving this result is that, in Fredkin-enabled MMD, application of *one Fredkin gate* matches two bits compared to two Toffoli gates, which MMD would have used. We show that there is a possibility of applying Fredkin gates in nearly all the rows, considering the worst-case scenario. We use the term *quadrant* here: The first quadrant refers to the first 2^{n-1} output rows, the second quadrant refers to the following 2^{n-2} output rows and so on.

Theorem 5.3: The upper bound of gate count for the proposed Fredkin-enabled MMD using the MPMCF library is $(n-2)2^n+2+n$.

Proof: In the worst-case scenario, the first output row will require the maximum n NOT gates to match with the first input row, 0, if its binary value is 2^n-1. In this row, Fredkin gates cannot be applied. Now let us assume the second output row is 2^n-2. Here we can apply one unconditional Fredkin gate (basic swap gate) and $n-2$ Toffoli gates, totaling to $n-1$ gates. Assuming, we apply one Fredkin gate to each row of the first quadrant, we need a total of $(n-1)2^{n-1}+1$ gates. At this point, from the second quadrant onward, all the output rows have 1 in its MSB position. To match the $2^{n-1}+1$-th output row, again, we cannot apply any Fredkin gate.

Thus, it can be deduced that, in the first output row of each quadrant, Fredkin gates cannot be employed. More importantly, in all the other rows, at least one Fredkin gate can be applied when we consider the worst-case scenario. To have the case for Fredkin gate, we need either a $0\rightarrow 1$ or a $1\rightarrow 0$ scenario, where $\rightarrow$ indicates desired alteration from more significant bit-position to less significant bit-position. Let us assume that we first apply all the necessary Toffoli gates to match all other bits with the input row. In that case, if we apply all the other bits as controls, only $1\rightarrow 0$ is a valid change and it does not affect the previous rows.

Thus, the worst-case count of gates (T_g) produced by the algorithm can be computed as follows:

$$\begin{aligned}T_g &= (n-1)2^{n-1}+(n-2)2^{n-2}+\cdots+0\cdot 2^{n-n}+n\\ &= (n-1)2^{n-1}+(n-2)2^{n-2}+\cdots+2+n\end{aligned}$$

With

$$2\cdot T_g=(n-1)2^n+(n-2)2^{n-1}+\cdots+2^2+2n$$

we simplify

$$\begin{aligned}T_g &= 2\cdot T_g - T_g\\ &= (n-1)2^n-(2^{n-1}+2^{n-2}+\cdots+2^2+2)+n\\ &= (n-1)2^n-(2^n-1-1)+n\\ &= (n-2)2^n+2+n\end{aligned}$$

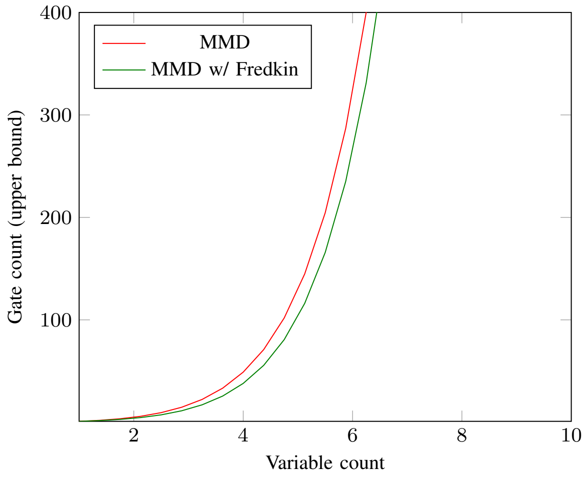

Fig. 4. Benchmarking gate count upper bounds

■

In an attempt to determine the circuit with worst-case gate count, the series of transformations as assumed, are applied on a 3-variable Boolean function. This is shown in the Table III. The corresponding reversible circuit is presented in Fig. 3.

The improvement of the upper bound for our proposed approach is pictorially depicted for increasing number of variables in the Fig. 4. The worst-case count of Fredkin (T_f) gates, while applying Fredkin-enabled MMD for MPMCT is then,

$$\begin{aligned}T_f &= \sum_{i=1}^{n}(2^{n-i}-1)\\ &= (2^{n-1}-1)+(2^{n-2}-1)+(2^{n-3}-1)+\cdots+(2^{n-n}-1)\\ &= (2^{n-1}+2^{n-2}+2^{n-3}+\cdots+2^{n-n})-n\\ &= (2^{n-1}+2^{n-2}+2^{n-3}+\cdots+2+1)-n\end{aligned}$$

With

$$2\cdot T_f=(2^n+2^{n-1}+2^{n-2}+\cdots+2-2n)$$

we simplify

$$\begin{aligned}T_f &= 2\cdot T_f - T_f\\ &= 2^n-1-n\end{aligned}$$

The worst-case number of NOT gates (T_{NOT}) is n, for the first row. Thus, the maximum number of Toffoli gates with at least one control line is

$$\begin{aligned}T_t &= T_g - T_f - T_{\text{NOT}}\\ &= (n-2)2^n+2+n-2^n+1+n-n\\ &= (n-3)2^n+n+3\end{aligned}$$

Based on the above individual results, it can be deduced that the worst-case QC for MMD using MPMCF is

TABLE III. TRANSFORMATION-BASED SYNTHESIS FOR WORST-CASE 3-VARIABLE CIRCUIT

$x_1x_2x_3$	$y_1y_2y_3$	$y_1^1y_2^1y_3^1$	$y_1^2y_2^2y_3^2$	$y_1^3y_2^3y_3^3$	$y_1^4y_2^4y_3^4$	$y_1^5y_2^5y_3^5$	$y_1^6y_2^6y_3^6$	$y_1^7y_2^7y_3^7$	$y_1^8y_2^8y_3^8$	$y_1^6y_2^6y_3^6$	$y_1^{10}y_2^{10}y_3^{10}$	$y_1^{11}y_2^{11}y_3^{11}$
000	▷111	000	000	000	000	000	000	000	000	000	000	000
001	001	▷110	▷101	001	001	001	001	001	001	001	001	001
010	110	001	001	▷101	▷110	010	010	010	010	010	010	010
011	011	100	100	100	100	▷100	▷101	011	011	011	011	011
100	101	010	010	010	010	110	111	▷111	▷110	100	100	100
101	100	011	011	111	111	011	011	101	100	▷110	101	101
110	010	101	110	110	101	101	100	100	101	111	▷111	110
111	000	111	111	011	011	111	110	110	111	101	110	111

TABLE IV. EXPERIMENTAL RESULTS

		w/o Fredkin (uni)			w/o Fredkin (bi)			w/ Fredkin (uni)			w/ Fredkin (bi)			w/ Fredkin+ (uni)			w/ Fredkin+ (bi)		
Benchmark	Lines	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time
ham3	3	8	9	0.00	8	9	0.00	7	9	0.00	7	9	0.00	6	9	0.00	5	9	0.00
miller	3	6	15	0.00	6	15	0.00	6	15	0.00	6	15	0.00	4	9	0.00	4	9	0.00
3_17	3	11	12	0.00	11	12	0.00	8	12	0.00	8	12	0.00	11	12	0.00	6	6	0.00
hwb4	4	24	108	0.00	24	108	0.00	19	90	0.00	18	87	0.00	9	18	0.00	9	18	0.00
majority	6	14	246	0.00	14	246	0.00	21	258	0.00	21	258	0.01	14	246	0.00	21	258	0.01
sym6	7	351	11292	0.03	299	8484	0.03	304	9045	0.07	276	7476	0.06	285	8085	0.03	236	6363	0.05
urf2	8	790	39642	0.36	639	26706	0.30	647	30342	0.39	548	22617	0.26	595	25428	0.27	493	20304	0.24
con1	8	844	36516	0.16	713	27363	0.14	677	28338	0.31	575	22035	0.28	662	24738	0.15	549	21024	0.26
hwb9	9	2050	118551	0.79	1624	80106	0.61	1726	91233	1.60	1410	66432	1.29	884	37080	0.42	752	30948	0.65
adr4	9	1597	96645	0.59	1238	64728	0.46	1310	79686	1.16	1085	57831	0.93	1034	53550	0.45	926	48498	0.95
urf5	9	681	44571	0.24	626	38835	0.22	522	33501	0.47	445	28263	0.34	543	33177	0.24	400	24879	0.41
urf1	9	1804	114444	0.73	1422	75837	0.57	1541	90507	1.33	1182	61518	1.06	1323	69963	0.56	1114	57723	1.10
5xp1	10	4730	340692	3.68	3590	212877	2.70	3897	263034	7.92	3161	186555	5.95	3410	203790	3.02	2932	172878	5.38
urf3	10	3426	258045	2.56	2728	176727	1.98	2780	197532	5.72	2304	148929	4.97	2599	168795	2.18	2128	138303	4.02
sym9	10	4566	320940	3.50	3483	207138	2.64	3926	262137	8.15	3072	182337	5.63	3333	197742	2.81	2879	170595	6.51
rd84	11	10452	880071	18.18	7816	547011	12.38	8826	714717	34.15	6861	482220	27.04	7538	529110	13.38	6465	454059	38.71
clip	11	9948	865272	16.40	7492	539985	11.77	8412	687129	31.35	6632	478263	24.54	7205	517527	12.65	6219	446301	24.15
urf4	11	10527	914817	19.27	7800	563442	12.93	8908	722418	32.13	6980	495126	25.66	7460	537411	13.93	6495	459351	27.19
sym10	11	10192	859314	19.10	7540	523143	14.98	8639	707556	36.44	6658	476910	26.03	7387	516723	14.28	6243	438315	26.81
cycle10_2	12	19	1218	0.10	19	1218	0.09	19	1218	0.11	19	1218	0.10	19	1218	0.09	19	1218	0.10
dc2	13	49949	5784213	325.68	36354	3455097	228.80	42443	4652268	657.68	31761	3022143	497.75	35107	3354747	244.91	30556	2905056	557.79

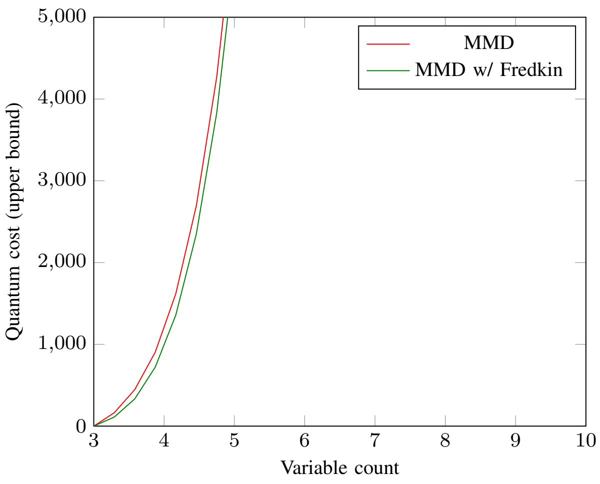

Fig. 5. Benchmarking quantum cost upper bounds

$$\begin{aligned} QC &= QC(T_t) + QC(T_f) + QC(T_{\mathrm{NOT}}) \\ &= ((n-2)2^n - n + 1) \cdot 24(n-3) \\ &+ (2^n - 1 - n) \cdot 24(n-3) \\ &= 24(n-3)(2^n n - 2n - 2^n) \end{aligned}$$

Graphically, the improvement in QC is shown in the Fig. 5.

VI. EXPERIMENTAL RESULTS

The synthesis approach we propose has been implemented in C++ on top of RevKit [17].[1] The results are listed in Table IV and were generated using the RevKit program '*transformation_based_synthesis*'. Standard benchmark circuits with varying I/O sizes are taken from www.revlib.org. For each circuit, the number of Toffoli gates, the quantum costs in terms of T-depth in a Clifford+T mapping according to [11] and the run-time (in seconds) are presented. Since this paper is on demonstrating the effect of incorporating Fredkin gates inside synthesis for cost reductions, we have not compared the approach to other synthesis algorithms of a different kind, but only to the classical transformation-based synthesis algorithm. Future work will consider the incorporation of Fredkin gates to these approaches.

The first two columns represent the standard MMD operations without Fredkin, in uni-direction or bi-direction mode. The next two columns show the Fredkin-enabled synthesis algorithm using the simple strategy for checking whether the control mask is valid. The last two columns also show results for applying the Fredkin-enabled synthesis algorithm but using the exhaustive strategy to check mask validity. Table V shows the percentage improvements. Notice that uni-directional Fredkin-enabled synthesis is compared to uni-directional MMD and bi-directional Fredkin-enabled synthesis is compared to bi-directional MMD.

The application of Fredkin gates clearly shows significant gate count and T-depth reduction without any noticeable impact on the run-time. In average the run-time increases by

[1]The source code that has been used to perform this evaluation is available at www.revkit.org (version 2.1).

TABLE V. Experimental results (percentage improvements)

		w/o Fredkin (uni)			w/o Fredkin (bi)			w/ Fredkin (uni)			w/ Fredkin (bi)			w/ Fredkin+ (uni)			w/ Fredkin+ (bi)		
Benchmark	Lines	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time	Gates	T-depth	Run-time
ham3	3	8	9	0.00	8	9	0.00	12.50	0.00	0.00	12.50	0.00	0.00	25.00	0.00	0.00	37.50	0.00	0.00
miller	3	6	15	0.00	6	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	33.33	40.00	0.00	33.33	40.00	0.00
3_17	3	11	12	0.00	11	12	0.00	27.27	0.00	0.00	27.27	0.00	0.00	0.00	0.00	0.00	45.45	50.00	0.00
hwb4	4	24	108	0.00	24	108	0.00	20.83	16.67	0.00	25.00	19.44	0.00	62.50	83.33	0.00	62.50	83.33	0.00
majority	6	14	246	0.00	14	246	0.00	-50.00	-4.88	0.00	-50.00	-4.88	0.00	0.00	0.00	0.00	-50.00	-4.88	0.00
sym6	7	351	11292	0.03	299	8484	0.03	13.39	19.90	-133.33	7.69	11.88	-100.00	18.80	28.40	0.00	21.07	25.00	-66.67
urf2	8	790	39642	0.36	639	26706	0.30	18.10	23.46	-8.33	14.24	15.31	13.33	24.68	35.86	25.00	22.85	23.97	20.00
con1	8	844	36516	0.16	713	27363	0.14	19.79	22.40	-93.75	19.35	19.47	-100.00	21.56	32.25	6.25	23.00	23.17	-85.71
hwb9	9	2050	118551	0.79	1624	80106	0.61	15.80	23.04	-102.53	13.18	17.07	-111.48	56.88	68.72	46.84	53.69	61.37	-6.56
adr4	9	1597	96645	0.59	1238	64728	0.46	17.97	17.55	-96.61	12.36	10.66	-102.17	35.25	44.59	23.73	25.20	25.07	-106.52
urf5	9	681	44571	0.24	626	38835	0.22	23.35	24.84	-95.83	28.91	27.22	-54.55	20.26	25.56	0.00	36.10	35.94	-86.36
urf1	9	1804	114444	0.73	1422	75837	0.57	14.58	20.92	-82.19	16.88	18.88	-85.96	26.66	38.87	23.29	21.66	23.89	-92.98
5xp1	10	4730	340692	3.68	3590	212877	2.70	17.61	22.79	-115.22	11.95	12.36	-120.37	27.91	40.18	17.93	18.33	18.79	-99.26
urf3	10	3426	258045	2.56	2728	176727	1.98	18.86	23.45	-123.44	15.54	15.73	-151.01	24.14	34.59	14.84	21.99	21.74	-103.03
sym9	10	4566	320940	3.50	3483	207138	2.64	14.02	18.32	-132.86	11.80	11.97	-113.26	27.00	38.39	19.71	17.34	17.64	-146.59
rd84	11	10452	880071	18.18	7816	547011	12.38	15.56	18.79	-87.84	12.22	11.84	-118.42	27.88	39.88	26.40	17.29	16.99	-212.68
clip	11	9948	865272	16.40	7492	539985	11.77	15.44	20.59	-91.16	11.48	11.43	-108.50	27.57	40.19	22.87	16.99	17.35	-105.18
urf4	11	10527	914817	19.27	7800	563442	12.93	15.38	21.03	-66.74	10.51	12.12	-98.45	29.13	41.25	27.71	16.73	18.47	-110.29
sym10	11	10192	859314	19.10	7540	523143	14.98	15.24	17.66	-90.79	11.70	8.84	-73.77	27.52	39.87	25.24	17.20	16.22	-78.97
cycle10_2	12	19	1218	0.10	19	1218	0.09	0.00	0.00	-10.00	0.00	0.00	-11.11	0.00	0.00	10.00	0.00	0.00	-11.11
dc2	13	49949	5784213	325.68	36354	3455097	228.80	15.03	19.57	-101.94	12.63	12.53	-117.55	29.71	42.00	24.80	15.95	15.92	-143.79
Average								12.42	15.53	-68.22	10.72	11.04	-69.20	25.99	34.00	14.98	22.58	25.24	-68.37

about 70%, for uni-directional synthesis using the exhaustive strategy for mask validity checking, the run-time decreases by about 15% in average. Gate count can be decreased by up to 26% in average and T-depth can be decreased by up to 34% in average (uni-directional, exhaustive strategy). Best cases are 62.5% for gate count and 83.33% for T-depth ('*hwb4*', exhaustive strategy). Only for the benchmark '*majority*' negative or no improvements are obtained with all four configurations.

VII. Conclusions

In this paper, transformation-based reversible logic synthesis algorithm for constructing ancilla-free reversible circuits is revisited. By introducing Fredkin gates in the synthesis method, the theoretical worst-case bound is reduced. This is corroborated by experimental evidence. In future, introduction of Fredkin gates in other synthesis methods will be studied.

Acknowledgments

Second author sincerely acknowledges the contributions of Nahian Chowdhury, MPSoC Architectures Research Group, RWTH Aachen for helpful discussions regarding the worst-case circuit constructions and derivation of theoretical bounds.

References

[1] M. Saeedi and I. L. Markov, "Synthesis and optimization of reversible circuits - a survey," *ACM Comput. Surv.*, vol. 45, no. 2, p. 21, 2013.

[2] O. Golubitsky and D. Maslov, "A study of optimal 4-bit reversible toffoli circuits and their synthesis," *IEEE Trans. Computers*, vol. 61, no. 9, pp. 1341–1353, 2012.

[3] D. Große, R. Wille, G. W. Dueck, and R. Drechsler, "Exact multiple-control toffoli network synthesis with SAT techniques," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 703–715, 2009.

[4] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003*, 2003, pp. 318–323.

[5] C. Chandak, A. Chattopadhyay, S. Majumder, and S. Maitra, "Analysis and improvement of transformation-based reversible logic synthesis," in *43rd IEEE International Symposium on Multiple-Valued Logic, ISMVL 2013, Toyama, Japan, May 22-24, 2013*, 2013, pp. 47–52.

[6] M. Soeken, L. Tague, G. W. Dueck, and R. Drechsler, "Ancilla-free synthesis of large reversible functions using binary decision diagrams," *Journal of Symbolic Computation*, 2015, accepted.

[7] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler, "Synthesis of reversible circuits with minimal lines for large functions," in *Proceedings of the 17th Asia and South Pacific Design Automation Conference, ASP-DAC 2012, Sydney, Australia, January 30 - February 2, 2012*, 2012, pp. 85–92.

[8] Z. Sasanian, M. Saeedi, M. Sedighi, and M. S. Zamani, "A cycle-based synthesis algorithm for reversible logic," in *Proceedings of the 14th Asia South Pacific Design Automation Conference, ASP-DAC 2009, Yokohama, Japan, January 19-22, 2009*, 2009, pp. 745–750.

[9] D. Maslov, G. W. Dueck, and D. M. Miller, "Fredkin/toffoli templates for reversible logic synthesis," in *2003 International Conference on Computer-Aided Design (ICCAD'03), November 9-13, 2003, San Jose, CA, USA*, 2003, pp. 256–261.

[10] R. Wille, A. Lye, and R. Drechsler, "Optimal SWAP gate insertion for nearest neighbor quantum circuits," in *19th Asia and South Pacific Design Automation Conference, ASP-DAC 2014, Singapore, January 20-23, 2014*, 2014, pp. 489–494.

[11] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, 2013.

[12] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Phys. Rev. A*, vol. 52, pp. 3457–3467, Nov 1995.

[13] D. Maslov, G. W. Dueck, and D. M. Miller, "Toffoli network synthesis with templates," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 807–817, 2005.

[14] A. Chattopadhyay, S. Majumder, C. Chandak, and N. Chowdhury, "Constructive reversible logic synthesis for boolean functions with special properties," in *Reversible Computation - 6th International Conference, RC 2014, Kyoto, Japan, July 10-11, 2014. Proceedings*, 2014, pp. 95–110.

[15] N. Abdessaied, M. Soeken, M. K. Thomsen, and R. Drechsler, "Upper bounds for reversible circuits based on young subgroups," *Inf. Process. Lett.*, vol. 114, no. 6, pp. 282–286, 2014.

[16] R. Wille, M. Soeken, N. Przigoda, and R. Drechsler, "Effect of negative control lines on the exact synthesis of reversible circuits," *Multiple-Valued Logic and Soft Computing*, vol. 21, no. 5-6, pp. 627–640, 2013.

[17] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "Revkit: An open source toolkit for the design of reversible circuits," in *Reversible Computation - Third International Workshop, RC 2011, Gent, Belgium, July 4-5, 2011. Revised Papers*, 2011, pp. 64–76.