A Best-Fit Mapping Algorithm to Facilitate ESOP-Decomposition in Clifford+$T$
Quantum Network Synthesis

Giulia Meuli∗ Mathias Soeken∗ Martin Roetteler† Nathan Wiebe† Giovanni De Micheli∗
∗Integrated Systems Laboratory, EPFL, Lausanne, Switzerland †Microsoft Research, Redmond, WA, USA

Abstract—Currently, there is a large research interest and a significant economical effort to build the first practical quantum computer. Such quantum computers promise to exceed the capabilities of conventional computers in fields such as computational chemistry, machine learning and cryptanalysis. Automated methods to map logic designs to quantum networks are crucial to fully realizing this dream, however, existing methods can be expensive both in computational time as well as in the size of the resultant quantum networks. This work introduces an efficient method to map reversible single-target gates into a universal set of quantum gates (Clifford+$T$). This mapping method is called best-fit mapping and aims at reducing the cost of the resulting quantum network. It exploits $k$-LUT mapping and the existence of clean ancilla qubits to decompose a large single-target gate into a set of smaller single-target gates. In addition this work proposes a post-synthesis optimization method to reduce the cost of the final quantum network, based on two cost-minimization properties. Results show a cost reduction for the synthesized EPFL benchmark up to 53% in the number of $T$ gates.

I. INTRODUCTION

The recent prospect of practical quantum computers [1], [2], [3] is pushing the design automation community to develop suitable tools that are able to address the peculiarities of quantum circuits. In fact, there are many aspects in which quantum computing differs from standard computing. First, quantum computers process qubits instead of bits. They do not only have the classical values 0 and 1, but can represent a superposition of these. The state of a qubit cannot be copied, so it is impossible to have a quantum gate with multiple fanout. All quantum circuits are reversible. During design it is possible to consider all the inputs as Boolean values—even though when embedded as part of a quantum algorithm entangled states in superposition are being applied. A quantum circuit performing a Boolean function is called reversible quantum network. This network is composed by reversible gates. In this paper, we consider the frequently used single-target gates and multiple-controlled Toffoli gates. Those are high-level abstractions of the real operations that can be performed on qubits. During design it is necessary to lower the level of abstraction and map this reversible gates into a universal set of quantum gates, the Clifford+$T$ set [4], in an efficient way.

In this paper, we present a method that maps single-target gates into Clifford+$T$ networks by applying ESOP (exclusive sum-of-products) decomposition [5]. To further reduce the costs of the Clifford+$T$ networks, we propose a post-synthesis optimization method based on graph matching that optimizes the Toffoli networks resulting from ESOP decomposition.

We evaluate our mapping algorithm within the LUT-based hierarchical reversible logic synthesis (LHRS) algorithm proposed in [6]. LHRS is a reversible synthesis algorithm that maps classical (irreversible) logic networks into reversible networks composed of single-target gates. Embedded into LHRS, experimental results show that we can obtain a reduction of up to 53% in the number of $T$ gates when synthesizing the EPFL arithmetic benchmarks. Experiments also show that the proposed mapping approach can significantly speed up the execution time of LHRS.

II. PRELIMINARIES

A. Reversible Network

A reversible network is a set of reversible gates realizing a reversible function. A multi-output Boolean function is reversible if each input pattern uniquely maps to an output pattern. To accomplish this specification, a reversible function has the same number of inputs and outputs. In addition, a reversible network that corresponds to a quantum network is required to be garbage free: intermediate values must not appear at any output terminal of the network. This is because the quantum networks typically need to be run on a superposition of different inputs and measuring and resetting garbage bits can betray the path that the quantum computer took, which can collapse the quantum state that encodes the data. To describe reversible gates, consider the following notation. A reversible gate performs an $n$-variable reversible function which is applied to qubits (lines) $X = \{1, \ldots, n\}$. We further consider literals based on the numbers in $X$, i.e., given $x \in X$, we can have $l$ as the positive literal and $\bar{l}$ as the negative of $x$. Note that $\bar{l} = l$, and we define $|l| = |\bar{l}| = x$. Finally, let $l \oplus 0 = l$ and $l \oplus 1 = \bar{l}$. Also, for a given set of literals $L$, we use $|L| = \{|l| \mid l \in L\}$ to refer to all variables of $L$.

Definition 1 (Single-target gate): Let $c : \mathbb{B}^k \to \mathbb{B}$ be a Boolean function, called control function. Also, let $C = \{c_1, \ldots, c_k\} \subset X$ be a set of control lines and let $t \notin C$ be a target line. Then the single-target gate $T_c(C, t) : \mathbb{B}^n \to \mathbb{B}^n$ is a reversible Boolean function which maps

$$(x_1, \ldots, x_n) \mapsto \begin{cases} x_i & \text{if } i \neq t, \vspace{1em} \\ x_t \oplus c(x_{c_1}, \ldots, x_{c_k}) & \text{otherwise}. \end{cases}$$
In other words, it inverts the target, if and only if the control function evaluates to true.

Definition 2 (Multiple-controlled Toffoli gate): If \( c \) can be expressed as a single product term \( c = \bigwedge_{i=1}^{k} (x_c_i \oplus p_i) \) in a single-target gate \( T_c(C, t) \), where \( p_i \) are the polarities of the controls, then we call the gate a multiple-controlled Toffoli gate. Since we consider these gates as special cases, we introduce a special notation \( T(C', t) \) where \( C' = \{ l \oplus p_l \mid l \in C \} \). An example of this special notation is in Fig. 2. For Toffoli gates, we will use \( C' \) and \( c \) interchangeably.

B. LUT mapping

In algorithms such as LHRS, the control function of a single-target gate is represented symbolically, e.g., using an and-inverter graph (AIG). An AIG is a logic network composed of AND gates and inverters (see Fig. 1(a) for an AIG representing the function \( \text{prime}_6(x_1, \ldots, x_6) = [(x_6 \cdots x_1)_2 \text{ is prime}] \)). Each non-terminal node in an AIG represents an AND gate with two operands and an edge between two nodes is complemented when is drawn dashed.

The \( k \)-LUT mapping describes the problem of mapping an AIG into \( k \)-LUTs, which are gates with \( k \) inputs that can represent any \( k \)-input Boolean function. Several algorithms have been presented to obtain a \( k \)-LUT mapping (see, e.g., [7], [8], [9]). Figs. 1(b) and 1(c) show a 3-LUT and 4-LUT mapping of the AIG. Note that vertices with the same color belong to the same LUT. There are cases in which nodes of the initial AIG were copied such that they can belong to two different LUTs. The 3-LUT and 4-LUT mappings contain 12 and 14 LUTs, respectively.

III. MAPPING OF SINGLE-TARGET GATES

We target the synthesis of Clifford+\( T \) circuits, a gate library consisting of the 2-qubit CNOT (controlled NOT) gate, and the single-qubit Hadamard (\( H \)) and \( T \)-gate. The \( T \) gate is by

\[
T_{1\wedge 2}(\{1, 2\}, 3)
\]

A single-target gate is represented symbolically, e.g., using an AIG into \( \text{prime}_6 \),

Fig. 2. Notations for the multiple-controlled Toffoli: the left gate is \( T_{1\wedge 2}(\{1, 2\}, 3) \) in the complete notation and \( T(\{1, 2\}, 3) \) in the special notation; for the second gate \( T_{1\wedge 2}(\{1, 2\}, 3) \) and \( T(\{1, 2\}, 3) \).

Fig. 1. LUT mapping for the function \( \text{prime}_6 \).

Fig. 3. Example of mapping a single-target gate into Toffoli gates using ESOP decomposition.

far the most expensive in fault tolerant quantum computing, so that it is often the only one defining the cost of a quantum algorithm [10]. The number of \( T \)-gates in the quantum network is called \( T \)-count and we are interested in minimizing this quantity. In this section we propose mapping techniques that solve the following problem in reversible logic and quantum circuit synthesis.

Problem 1: Given a single-target gate \( T_c(C, t) \), a set of clean ancilla lines \( X_{\text{clean}} \), find a Clifford+\( T \) network that realizes the function \( c \) on line \( t \) and restores the initial values on all other lines.

Ancilla are helper lines that can be used to map reversible gates into quantum gates more efficiently. In large reversible circuits many ancilla lines are available because each reversible gate locally interacts only with a portion of all the qubits. If the value of the ancilla is known to be zero when used in the realization of a gate, then the ancilla is called clean. We first introduce existing mapping methods. The first two are direct methods and do not make use of any ancilla line: one based on ESOP decomposition, and one based on precomputed optimal networks. We propose then a novel method which exploits ancillae by means of \( k \)-LUT networks, selecting the most suitable LUT size. This novel method is called best-fit mapping of single-target gates.

A. Direct Mapping Methods

ESOP-decomposition based mapping. One can always decompose a single-target gate \( T_c(\{x_1, \ldots, x_k\}, x_{k+1}) \) into a cascade of Toffoli gates

\[
T_{c_1}(X_1, x_{k+1}) \circ T_{c_2}(X_2, x_{k+1}) \circ \cdots \circ T_{c_l}(X_l, x_{k+1}),
\]

where \( c = c_1 \oplus c_2 \oplus \cdots \oplus c_l \), each \( c_i \) is a product term or 1, \( X_i \subseteq \{x_1, \ldots, x_k\} \) is the support of \( c_i \) and, \( \circ \) performs
a function composition. This decomposition of $c$ is also referred to as ESOP decomposition [11], [12], [13]. ESOP minimization techniques can be applied to reduce the size of the ESOP expression. An example is given in Fig. 3. Eventually, each of the multiple-controlled Toffoli gates is mapped into a Clifford+$T$ with the mapping described in [10].

**Near-optimum mapping.** For small functions it is practical to store optimal Clifford+$T$ realizations, found by applying optimization algorithms such as [14], [15], [16], [17], and store them into a database. In order to reduce the number of optimal results to be stored, affine classification of Boolean function is exploited [18], [19], [20]. There are $2^2$ Boolean functions on $n$ variables and they can be partitioned into equivalence classes by allowing transformations on the inputs that only require CNOT gates and therefore do not account to the number of $T$ gates in the quantum circuit. Two functions that are equivalent under the condition of affine equivalence under negation are called AN-equivalent. The exploitation of this classification method enables to scale the use of the database up to 5 input-LUTs. Indeed for $n = 1, 2, 3, 4$ and $5$ there are only $2, 3, 6, 18$ and 206 classes of AN-equivalent functions, respectively [21].

**B. LUT-based Mapping Methods**

We describe techniques that exploit LUT mapping to translate a single-target gate into a network of multiple-controlled Toffoli gates. A LUT-based method performs $k$-LUT mapping, which consists of dividing the network into subnetworks, each one having maximum $k$ inputs. If the mapped LUT network is composed of $\ell$ LUTs, then it is possible to map the single-target gate into a network of single-target gates with at most $k$ inputs, by using $\ell - 1$ clean ancilla lines. Fig. 4 shows how the LUT mapping in Fig. 1(c), where $k = 4$, can be mapped into a single-target gates reversible network. Note that for the LUT mapping in Fig. 1(b), where $k = 3$, 11 clean ancilla are needed.

**Hybrid LUT-based mapping.** The hybrid method is a previously proposed LUT-based mapping that aims at exploiting the near-optimal precomputed networks [6]. Given the input AIG it performs 4-LUT mapping to match the 4-input functions affine equivalent classes. However, it can happen that there are not enough clean ancilla to store all the intermediate values of the mapping. In that case the hybrid method merges two LUTs into a larger one, thereby requiring one fewer clean ancilla. This procedure is repeated until the number of available clean ancilla suffices. This procedure leads to one very large LUT. In fact, this LUT can have more inputs than primary inputs.

**Best-fit mapping.** The best-fit mapping method addresses the limitations of both the direct and the hybrid methods. It exploits extra ancilla, it applies near-optimal pre-computed networks, and reduces the size of the functions that are directly synthesized. The idea is to find a suitable value for $k$. This value is chosen to be the smallest for which the LUT size fits the available number of clean ancilla. Starting from $k = 4$, $k$ is incremented until the above mentioned condition is satisfied. Once the mapping is obtained, each $k$-LUT needs to be mapped in Clifford+$T$ gates. If the number of a LUT’s inputs is small enough, the LUT can be replaced by the precomputed optimal Clifford+$T$ network, otherwise ESOP-based decomposition is applied.

**IV. Post-Synthesis Optimization**

In addition we propose a technique to reduce the $T$-count of reversible networks composed of multiple-controlled Toffoli gates. It is particularly useful for our best-fit technique. In fact, the optimization is effective for reversible circuits obtained with ESOP-decomposition; since these circuits consist of a set of multiple-controlled Toffoli gates all acting on the same target lines. Networks with these characteristics can be optimized exploiting gate pair combinations described in the next section. In addition it is important to note that the position of a gate is irrelevant in these networks, in fact all the gates have the same target and for this reason they can be moved in any order.

**A. Exploited Properties**

The rules here described are possible ways to combine two gates with specific characteristics of their control lines. All the rules apply on two gates which share the same target line.

**Rule 1:** Let $g_1 = g_2 = T(C, t)$. Then $costs(g_1 \circ g_2) = 0$. If two gates are identical, they can both be removed from the network. This property is called deletion rule in [22].

**Rule 2:** Let $g_1 = T(C_1, t)$ and $g_2 = T(C_2, t)$ with $A = C_1 \cap \overline{C_2}$ and $B = C_2 \cap \overline{C_1}$.

If the following conditions are verified:

- if $l \in C_1$ then $l \not\in C_2$,
- if $l \in C_2$ then $l \not\in C_1$,
- $\#B = 1$, \quad $B = \{c\}$

where $\#B$ is the cardinality of the set of controls $B$. Then $g_1 \circ g_2 = T(A, |c|) \circ g_2 \circ T(A, |c|)$.

Rule 2 is a generalization of some rules presented in [23]. An example is shown in Fig. 5, explaining the rules using identities from [22]. Given two Toffoli gates, it applies if one gate has a single control on a qubit that is not a control of the other one. In the example this is $x_5$. It is possible to substitute the second gate with two identical gates applied before and after.

**Fig. 4.** Example of mapping a single-target gate into Toffoli gates using LUT mapping.

**Fig. 5.** Rule 2 example. Equivalence rules from [22]: (a) D1, (b) D7, (c) D1.
the remaining one. They are controlled by the ones not in common with the first gate. This rule leads to cost reduction only if the two initial gates have some identical controls in common: $x_1$ and $x_2$ in the example.

Rule 3 ([24]): Let $g_1 = T(C_1, t)$ and $g_2 = T(C_2, t)$ with $\#C_1 = \#C_2$, i.e., $g_1$ and $g_2$ share the same set of controls. Let $D = C_1 \cap C_2 = \{c_1, \ldots, c_k\}$ be the set of controls that occur in different polarities in $g_1$ and $g_2$, and let $\#D > 0$. Then

$$g_1 \circ g_2 = \bigcap_{i=2}^{k} T(c_i, |c_i|) \circ T(C_1 \cap C_2, t) \circ \bigcap_{i=2}^{k} T(c_i, |c_i|).$$

An example is shown in Fig. 6. This rule applies when the first and the second gates have controls on the same lines, with different polarities. It uses two identical CNOT gates before and after the initial gates to complement the polarity of one control (see rule D2 in [22]). This is done until only one control with different polarity remains. Then the pair is equivalent to a single gate with this control removed and all the identical controls kept (see rule D3 in [22]).

B. Graph Matching Problem

Direct single-target gate mapping using ESOP decomposition leads to reversible networks with multiple-controlled Toffoli gates that all have the same target line. There are many pair of gates that could be combined and this paragraph describes the algorithm used to select which pairs to combine. The exploited method derives an optimization graph from the circuit and performs graph matching in a similar fashion to how it has been done in [25].

Definition 3 (Optimization graph): Given a set of generalized Toffoli gates $g_1 = T(C_1, t)$, $g_2 = T(C_2, t)$, \ldots, $g_m = T(C_m, t)$, we define the undirected graph $G = (V, E)$ with edge weights $q : E \to \mathbb{N}_0$ as follows:

$$V = \{g_1, \ldots, g_m\}$$

$$E = \{(v, w) \mid v, w \in V \land \text{costs}(v \circ w) < \text{costs}(v) + \text{costs}(w)\}$$

$$q(e) = \text{costs}(v) + \text{costs}(w) - \text{costs}(v \circ w)$$

where $e = \{(v, w)\}$

In other words, vertices in $G$ are all gates and two gates are connected by an edge if their cost when combined together is smaller than their accumulated individual cost. The weights on an edge $e = \{(v, w)\}$ describe the cost savings that can be achieved when composing the gates $v$ and $w$ together. We refer to this graph as optimization graph. We use graph matching to find the set of graph edges corresponding to the set of combined pairs that leads to the largest gain in terms of cost. The following theorem follows trivially.

Theorem 1: Let $G = (V, E)$ be an optimization graph as defined above. Let $M = \{e_1, \ldots, e_D\}$ be a graph matching for $G$. Then it is possible to realize all generalized Toffoli gates in a circuit with

$$\text{costs}(g_1 \circ \cdots \circ g_m) = \sum_{(v, w) \in M} \text{costs}(v \circ w) + \sum_{v \in V_r} \text{costs}(v)$$

where $V_r = V \setminus (e_1 \cup \cdots \cup e_D)$.

A greedy algorithm is used to compute a graph matching with a maximal weight. Given the matching, that corresponds to a set of edges in the optimization graph. Each edge refers to a pair of Toffoli gates in the initial circuit to be combined together exploiting Rule 2 or 3. The final circuit is computed considering the reduced cost of combined gates.

V. RESULTS

We implemented the algorithm in C++ on top of RevKit [26]. Experiments were run on an Intel Core i3 with 3.06 GHz and 4 GB RAM running Mac OS X Yosemite.

A. Best-Fit Mapping Method

The efficiency of the best-fit mapping method is compared to the hybrid mapping, a method already implemented in LHRS [6]. The EPFL arithmetic benchmark is used for the evaluation and the results are shown in Table I. For each benchmark the table shows the results for three different methods: HY – all single-target gates with hybrid mapping; BF – all single-target gates with best-fit mapping; PB – pick-best method. Pick-best selects, for every single-target gate, the one between the hybrid and the best-fit technique which results in the lower $T$-count. All the final networks are post-optimized with the method proposed in Section IV.

In order to perform the synthesis of a complex circuit, the LHRS framework performs an initial LUT-mapping, then associates each LUT with a single-target gate to be synthesized. This part of the procedure is independent from the mapping method evaluation, but the selection of the $k_1$ parameter for this initial LUT mapping has an impact on the complexity of the single-target gates and on the number of qubits. Different initial $k_1$ selections are shown in Table I: 16-LUT, 22-LUT and 28-LUT. It is important to not confuse this $k_1$ parameter, that is used to map the initial benchmark AIG into single-target gates, with the $k$ used for mapping of each single-target gate into Clifford$^+T$ networks.

The results show that the best-fit method is always superior to the hybrid method in terms of the number of $T$ gates in the final network. While the second method applies best-fit to all the single-target gates that compose the circuit; the pick-best method selects for each single-target gate the one between hybrid and best-fit that results in the smaller cost. Most of the time using pick-best helps in getting a smaller $T$-count, even if the gain is always small. This proves that for most of the possible, isolated, single-target gates the mapping method best-fit is superior to the hybrid method.

All the simulations have been performed with a timeout of 30 minutes. The LUT-based mapping techniques are particularly slow whenever direct mapping has to be performed.

1https://msoeken.github.io/revkit.html
This happens whenever the number of inputs of the LUT exceeds the AN-classification capabilities. Even if there is not any runtime difference between the three methods with a fixed number of available qubits; the situation changes when some extra qubits are available for the synthesis. In the LHRS framework, the number of available ancillae for the mapping of each single-target gate is fixed by means of the first LUT-mapping. Nevertheless, because the mapping algorithm can also be used outside the proposed framework, it includes the possibility to specify a different number of ancillae. In this experiment this is used to show the best-fit algorithm superior capabilities in exploiting additional ancillae. Indeed the best-fit mapping, varying its $k$ parameter, is able to take advantage from extra qubits, reducing the number and the size of the LUTs that are mapped with the direct method. Note that some of the results in hyp, log2, multiplier, and square are marked with the symbols * and **. This means that it is possible to get these results by adding 20 qubits or 40 qubits, respectively, whenever running into a timeout. Wherever in the table there is a timeout indication, that means that not even with 40 additional qubits a mapping is possible in 30 minutes. This is often the case for the hybrid method.

**B. Post-Synthesis Optimization**

The post-synthesis optimization technique proposed in Section IV is evaluated on the EPFL arithmetic benchmark. It is synthesized with the LHRS framework with direct mapping method. Results are shown in Table II and Table III. The first table shows a comparison between greedy and an exact matching algorithms, for example the blossom algorithm [27], with $k_1 = 6$. A small $k_1$ is chosen in order to be able to compute the exact matching in a reasonable time. It proves that the greedy algorithm, whose complexity is $\Theta(E)$ where $E$ is the number of edges in the graph, is capable of obtaining satisfying maximal matching. The exact approach only leads to very small improvements, always less than 1% in the case of $k_1 = 6$. We are then confident that for the purpose of the reversible circuit optimization, the greedy approach is accurate enough and that the results of the exact approach are not worth the complexity it has. The second table shows the case in which the initial LUT mapping has been performed with $k_1 = 16$. With a larger $k_1$, each single-target gate has a more complex control function in terms of number of gates and inputs and more room for improvements. The optimization leads to a 53% reduction in terms of $T$-count.

**VI. CONCLUSION**

We propose an efficient method to map single-target gates into Clifford+T logic networks. This method has been integrated in the open-source LHRS framework for Hierarchical Reversible Synthesis and its performances have been compared with other mapping methods proving its superiority. We also provide a post-synthesis optimization method to reduce the cost of reversible quantum networks, reaching up to 53% reduction in $T$-count. Both the proposed method have been evaluated on the EPFL arithmetic benchmark.

**Acknowledgments:** This research was supported by H2020-ERC-2014-ADG 669354 CyberCare and the Swiss National Science Foundation (200021-169084 MAJesty and and 200021-146600).
TABLE I. MAPPING TECHNIQUES COMPARISON

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>$k_1$</th>
<th>runtime [s]</th>
<th>qubits</th>
<th>$\langle$T-count$\rangle$</th>
<th>$\bar{k}_1$</th>
<th>runtime [s]</th>
<th>qubits</th>
<th>$\langle$T-count$\rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>adder</td>
<td>HY</td>
<td>0.03</td>
<td>463</td>
<td>$16788$</td>
<td>max</td>
<td>0.18</td>
<td>796</td>
<td>$101063$</td>
</tr>
<tr>
<td>bar</td>
<td>HY</td>
<td>0.05</td>
<td>439</td>
<td>$30738$</td>
<td>max</td>
<td>0.83</td>
<td>720</td>
<td>$119359$</td>
</tr>
<tr>
<td>BF</td>
<td>0.02</td>
<td>463</td>
<td>5280</td>
<td>max</td>
<td>BF</td>
<td>0.16</td>
<td>796</td>
<td>$60927$</td>
</tr>
<tr>
<td>bar</td>
<td>BF</td>
<td>0.03</td>
<td>439</td>
<td>8352</td>
<td>BF</td>
<td>0.74</td>
<td>84927</td>
<td></td>
</tr>
<tr>
<td>bar</td>
<td>BF</td>
<td>0.06</td>
<td>415</td>
<td>12432</td>
<td>BF</td>
<td>0.43</td>
<td>707</td>
<td>$84406$</td>
</tr>
<tr>
<td>PB</td>
<td>0.1</td>
<td>439</td>
<td>8002</td>
<td>max</td>
<td>PB</td>
<td>0.63</td>
<td>65224</td>
<td></td>
</tr>
<tr>
<td>PB</td>
<td>0.18</td>
<td>415</td>
<td>12082</td>
<td>max</td>
<td>PB</td>
<td>2.31</td>
<td>707</td>
<td>$80991$</td>
</tr>
<tr>
<td>PB</td>
<td>0.23</td>
<td>415</td>
<td>12082</td>
<td>max</td>
<td>PB</td>
<td>2.32</td>
<td>707</td>
<td>$80587$</td>
</tr>
</tbody>
</table>

TABLE II. DIRECT MAPPING - $k_1 = 6$ - GREEDY VS EXACT MATCHING ALGORITHM

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>depth</th>
<th>$t_{count}$</th>
<th>size</th>
<th>runtime [s]</th>
<th>$\langle$T-count$\rangle$</th>
<th>gates</th>
<th>$\langle$T-count$\rangle$</th>
<th>%Tcount</th>
</tr>
</thead>
<tbody>
<tr>
<td>adder</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
<tr>
<td>bar</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
<tr>
<td>BF</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
<tr>
<td>BF</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
</tbody>
</table>

TABLE III. DIRECT MAPPING - $k_1 = 16$ - NAIVE VS POST-OPTIMIZED SYNTHESIS

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>depth</th>
<th>$t_{count}$</th>
<th>size</th>
<th>runtime [s]</th>
<th>$\langle$T-count$\rangle$</th>
<th>gates</th>
<th>$\langle$T-count$\rangle$</th>
<th>%Tcount</th>
</tr>
</thead>
<tbody>
<tr>
<td>adder</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
<tr>
<td>bar</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
<tr>
<td>BF</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
<tr>
<td>BF</td>
<td>28</td>
<td>2.05</td>
<td>32170428</td>
<td>858.40</td>
<td>60927</td>
<td>1620</td>
<td>518</td>
<td>3385470</td>
</tr>
</tbody>
</table>

* = 20 qubits, ** = 40 qubits