Enumerating Optimal Quantum Circuits using Spectral Classification

Giulia Meuli* Mathias Soeken*† Martin Roetteler† Giovanni De Micheli*
*LSI, EPFL, Lausanne, Switzerland †Microsoft, Redmond, WA, USA

***Abstract*—This work targets fault-tolerant quantum computing and focuses on the problem of mapping reversible circuits into the Clifford+T quantum gate library. We present an automatically-generated database containing minimal-cost quantum circuits for Boolean functions up to 5 inputs. The database contains three circuits for each spectral-equivalent class representative, which are respectively optimized for the T-count, the T-depth, and the number of qubits. We show that any Boolean function can be derived from the implementation of its class representative without increasing any of the stated cost functions.**

I. Introduction

Few quantum systems have recently been developed by technology companies and their academic partners [1]–[3]. Current fabrication technologies only enable systems with few noisy qubits, referred to as Noisy-Intermediate-Scale Quantum (NISQ) systems. While the capabilities of such systems may suffice in some specific applications, their size and noise levels do not allow to compute fault-tolerantly, i.e., using quantum error-correcting codes. As the technology is developing, it is believed that fault-tolerant quantum computing will revolutionize the way computation is performed through the development of quantum algorithms, e.g., [4]–[6], which can break the lower bound complexities of their classical counterparts.

Quantum algorithms often require the computation of large classical logic functions to be performed directly on the quantum system. Such logic functions are usually specified using a high-level description language that has to be mapped into native and application-specific operations. This process is called *quantum compilation*. In fault-tolerant quantum computing, native operations are grouped into the universal Clifford+T library. This library contains the T gate, which is very expensive to be implemented fault-tolerantly and is often the only one accounted for when estimating the cost of a quantum algorithm [7], [8]. In this setting, compilation aims at minimizing both the T-count, which is the number of T gates and the T-depth, which is the maximum number of T gates that cannot be performed in parallel [9]. Finally, compilation targets the minimization of the total number of qubits of the resulting quantum circuit.

In this work, we present a database of quantum circuits for all the representatives of the spectral-equivalent classes of Boolean functions with 4 and 5 inputs. To generate the circuits, we use three different compilation algorithms: two existing ones, designed to minimize the T-count and the number of qubits, and a new algorithm to minimize the T-depth.

The database can be used to implement any Boolean function. First, any 4- and 5-input function can be directly generated from the spectral-equivalent entry without using any additional qubit or T gate. Second, larger Boolean functions can be automatically compiled from the database exploiting hierarchical methods as, e.g., the one proposed in [10].

II. Preliminaries

A. Boolean functions

A Boolean function over n variables is defined as $f : \mathbb{B}^n \to \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$. A Boolean function can be represented by its truth table, which is a bitstring $b_{2^n-1}b_{2^n-2}\dots b_0$ of size 2^n where $b_x = f(x_1, \dots, x_n)$ when $x = (x_1x_2\dots x_n)_2$. For large functions, it is convenient to use a hexadecimal encoding of the bitstring.

Example 1: The truth table of the majority-of-three function $\langle x_1x_2x_3\rangle$ is 1110 1000 or $^{\#}$`e8` in hexadecimal encoding.

Every Boolean function can also be represented in terms of an *exclusive sum-of-products* (ESOP) expression. This representation is not unique and many heuristic and exact minimization methods have been proposed [11]–[14].

B. Logic networks

Multi-level logic networks are scalable representations of Boolean functions. A logic network is represented by a graph in which each node performs a Boolean operation and edges define data dependencies. The inputs of the function are the primary inputs of the graph. Networks are characterized by their size, i.e., the number of nodes, and by their depth, i.e., the number of levels in the graph. Two nodes are in the same level if they have the same maximum distance from the primary inputs. According to the characteristics of the network, we define different graph representations. In this work, we use Xor-And-inverter Graphs (XAG), in which nodes implement the 2-input XOR, the 2-input AND. and inversion. Fig. 2 shows the XAG network for the majority-of-three function $^{\#}$`e8`, where dashed edges represent inversion.

C. Reversible circuits

The automatic compilation of a Boolean function f usually proceeds by embedding f in a reversible function.

Definition 1 (Reversible Boolean function): A multi-output Boolean function $f : B^n \to B^m$ is reversible, iff f is a bijection: each input pattern uniquely maps to an output pattern. A reversible function is implemented using reversible gates; in particular, we define the single-target gate and the multiple-control Toffoli gate.

Definition 2 (Single-target gate): Let $c : \mathbb{B}^k \to \mathbb{B}$ be a Boolean function, called the *control function*. Also, let $C = \{c_1, \dots, c_k\}$ be a set of *control lines* and let $t \notin C$ be a *target*

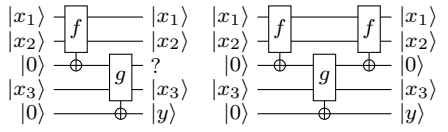


Fig. 1. (a) The function $y = g(f(x_1, x_2))$ is computed using two single-target gates. An unknown intermediate result is generated. (b) Garbage-free circuit where the intermediate result has been uncomputed by applying f twice.

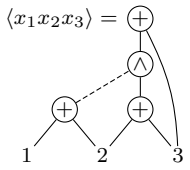


Fig. 2. An XAG for the majority of 3-inputs.

line. Then the *single-target gate* $\mathrm{T}_c(C, t) : \mathbb{B}^n \to \mathbb{B}^n$ is a reversible Boolean function which maps

$$(x_1, \ldots, x_n) \mapsto (x'_1 \ldots, x'_n) \quad \text{where}$$

$$x'_i = \begin{cases} x_i & \text{if } i \neq t, \\ x_t \oplus c(x_{c_1}, \ldots, x_{c_k}) & \text{otherwise.} \end{cases}$$

In other words, it inverts the target if and only if the control function evaluates to true.

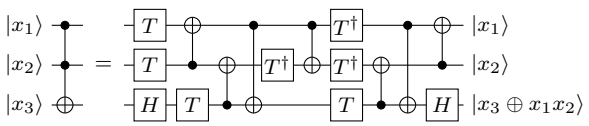
Definition 3 (Multiple-control Toffoli gate): A *multiple-control Toffoli gate* is a single-target gate $\mathrm{T}_c(C, t)$ whose function is a single product term.

Quantum compilation requires reversible circuits to be *garbage-free*, which means that all intermediate results need to be uncomputed. This is due to the fact that quantum circuits are run on a superposition of different inputs (*quantum parallelism*), and measuring and resetting garbage bits can collapse the quantum state that encodes the data. Fig. 1 shows a reversible circuit in which the intermediate state is uncomputed by repeating a single-target gate.

D. Quantum circuits

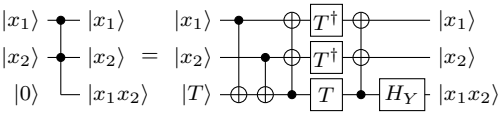
Quantum circuits describe a sequence of operations, represented by quantum gates. In fault-tolerant quantum computing, we consider the Clifford+T universal library which consists of the CNOT gate, the Hadamard gate, abbreviated H, as well as the T gate, and its inverse $T^\dagger$. The T gate is sufficiently expensive that it is customary to neglect all other gates when costing a quantum algorithm. The CNOT gate is a 2-qubit gate which inverts its target if the control qubit is in the one state. Classically, it would correspond to a single-control Toffoli gate. A CNOT gate with zero controls is an X gate. For more details on quantum gates, we refer the reader to [15].

Automatic quantum compilation requires quantum implementations of the reversible gates. The 2-control Toffoli gate has a Clifford+T implementation that requires 7 T gates [7], which is optimum [8], [16]:



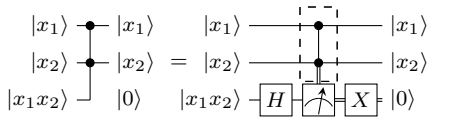
(1)

When the Toffoli gate is computed on a qubit initialized to $|0\rangle$, it can be implemented using 4 T gates, with a T-depth of 2, and without requiring any additional qubit [17], [18]:



(2)

where $H_Y = SH$ and $|T\rangle = TH|0\rangle$. Besides, when the result of the Toffoli is uncomputed, this can be performed without the use of any T gate, exploiting measurement-based uncomputation, as shown:



(3)

Several works from the literature describe how to map larger multiple-control Toffoli gates into Clifford+T gates (see, e.g., [7], [8], [19], [20]). Among them, a method proposed by *Barenco et al.* [21] allows us, provided an additional qubit, to map any multiple-control Toffoli gate into a sequence of 2-control Toffoli gates, which can be implemented using the optimum circuits in (1) and (2).

E. Spectral operations and classification

We define the five operations that are used to partition the set of all n-variable Boolean functions into equivalence classes.

Definition 4 (Spectral invariant operations [22]):

1) *Swapping two variables* ($f \xrightarrow{x_i \leftrightarrow x_j} g$).
2) *Complementing a variable* ($f \xrightarrow{\bar{x}_i} g$).
3) *Complementing the function* ($f \xrightarrow{\neg} g$).
4) *Translational operation* ($f \xrightarrow{x_i \oplus x_j} g$). It replaces one input x_i with $x_i \oplus x_j$.
5) *Disjoint translational operation* ($f \xrightarrow{\oplus x_i} g$). It is obtained by XOR-ing f with an input x_i .

These operations partition all n-variable Boolean functions into equivalence classes by means of the following equivalence relation.

Definition 5 (Spectral equivalence [23]): Two n-variable Boolean functions f and g are *spectral-equivalent*, if there exist operations $o_1, \ldots, o_k$ (from Definition 4) such that:

$$f \xrightarrow{o_1} \cdots \xrightarrow{o_k} g.$$

Using this equivalence relation, the set of all n-variable Boolean functions for $n = 1, 2, 3, 4, 5, 6$ collapses into just $1, 2, 3, 8, 48, 150\,357$ equivalence classes, respectively [24], [25]. We refer the reader to the literature (e.g., [26]–[28]), for further information on spectral classification techniques.

III. Spectral equivalence in quantum compilation

In this section, we explain how our database can be used to compile any Boolean function, by only adding Clifford gates.

Given two Boolean functions f and f' such that $f \xrightarrow{o} f'$ where o is a spectral operation, the optimal T-count (T-depth) for a single-target gate controlled by f' must equal the optimal T-count (T-depth) for a single-target gate controlled by f. The

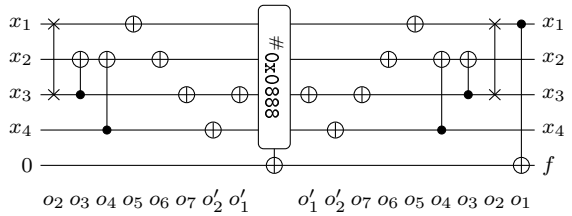


Fig. 3. Synthesized reversible circuit for the function f, obtained performing all the spectral operation to retrieve f from the optimal implementation available in the database f_r ($^{\#}$0x0888)

reason is that the five spectral operations can be implemented only using X and CNOT gates.

Example 2: Assume we want to map $\mathrm{T}_f(\{x_1, x_2, x_3, x_4\}, x_5)$ with $f = {}^{\#}\texttt{acab}$. Knowing that $f \in [{}^{\#}0888]$. Thus let $f_r = {}^{\#}0888$ be the database entry for this class. The spectral canonization algorithm in [22] finds the operation sequences to transform f and f_r in a canonical representative function of the equivalent class ($^{\#}$8880). Respectively, $o_1 = \oplus x_1, o_2 = x_1 \leftrightarrow x_3, o_3 = x_2 \oplus x_3, o_4 = x_2 \oplus x_4, o_5 = \bar{x}_1, o_6 = \bar{x}_2, o_7 = \bar{x}_3$ and $o'_1 = \bar{x}_3, o'_2 = \bar{x}_4$. We obtain a circuit as illustrated in Fig. 3. First the operations o'_1, o'_2 are applied to transform $f_r \rightarrow {}^{\#}8880$ then $o_7, \ldots, o_1$ to transform ${}^{\#}8880 \rightarrow f$.

The database can be used in combination with decomposition-based compilation techniques, as the look-up table (LUT) based method proposed in [10]. This method takes a logic network with an arbitrary number of inputs and decomposes it in sub-networks with maximum k inputs, using k-LUT-based decomposition. The algorithm can be extended to fetch implementations for each sub-network from the proposed database. As a consequence, any improvement in the database would positively affect the result of the compilation.

IV. Algorithms for the synthesis of single-target gates

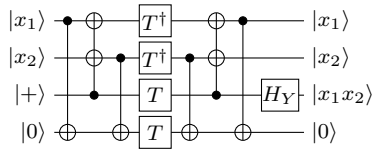
This section presents three different compilation methods. Each one has been developed to minimize a different cost function for fault-tolerant quantum computing: the T-count, the T-depth, and the number of qubits.

A. An XAG-based algorithm to optimize T-count

The algorithm that we employ to generate circuits with minimal T-count is a constructive hierarchical method based on Xor-And-inverter Graphs (XAGs) [29]. This compilation algorithm is capable of generating a quantum circuit using $O(N)$ qubits and $O(N)$ gates, where N is the size of the graph. The method guarantees an upper bound on the T-count, which depends on the multiplicative complexity $\tilde{c}$ of the Boolean function representation. This is the number of AND nodes in the network. In particular, the T-count is at most equal to $4 \times \tilde{c}$. The algorithm takes advantage of the low T-count implementation of the AND function proposed by [17], [18], and shown in (2), (3).

B. An XAG-based algorithm to optimize T-depth

We propose a hierarchical constructive method that aims to minimize the circuit T-depth and uses XAGs as inputs. We exploit an AND gate implementation with T-depth = 1, which combines the AND circuit from [18] and the T-depth one Toffoli gate implementation in [19]. The circuit requires one extra qubit with respect to the implementation in (2):



(4)

where $|+\rangle = H|0\rangle$.

Given an XAG with d levels containing at least one AND node, our algorithm would generate circuits with T-depth equal to d. As a consequence, depth optimization may be applied to minimize the resulting T-depth. Our synthesis method is based on the fact that each AND node is driven by two multi-input parity functions, which can be computed in-place using CNOT gates. It proceeds level by level and in topological order, and:

1) finds the parity functions that are input to AND nodes;
2) computes the parity functions in-place using CNOT gates;
3) if the function is input to more AND nodes, uses a CNOT to copy it on a new qubit initialized to $|0\rangle$;
4) implements inversions using X gates;
5) implements all AND nodes in the level using (4).

By copying some of the qubits, we enable parallel computation of all the AND nodes in one level.

C. An ESOP-based algorithm to optimize the number of qubits

To generate the database entries optimized for the number of qubits, we select an algorithm that generates a Toffoli network without adding any additional line. The algorithm builds a single-target gate controlled by the given Boolean function. Then, it performs the ESOP-based decomposition of the single-target gate into a cascade of Toffoli gates. Finally, each multiple-control Toffoli gate is mapped into 2-control Toffoli using Barenco decomposition [21] and in Clifford+T using the methods described in Section II-D. The algorithm for ESOP synthesis that we apply is a portfolio method proposed in [30]. It selects the best ESOP-decomposition strategy between Positive Polarity Reed Muller (PPRM) [31], Pseudo-Kronecker Reed Muller (PKRM) [32], and the exact method proposed in the same publication.

V. Database of optimal quantum circuits

We run the three compilation algorithms described in Section IV on all the spectral-equivalent class representatives of 4- and 5-input Boolean functions. Results are presented in Table I. It is clear that each algorithm succeeds in optimizing its respective cost function. To synthesize the all benchmark, the ESOP-based method uses a total of 345 qubits, against 475 and 625 qubits required by the other methods. The XAG method that targets T-count optimization achieves a total of 712 T gates, against the 10913 produced by the ESOP-based method. Besides, the XAG-based method for T-depth optimization achieves the same T count as the previous method (712) but reduces the T-depth from 220 to 143, paying the price of additional Cliffords and qubits. Note that, when measurement-based uncomputation is used, the number of

TABLE I

	optimal num. of qubits				optimal T-count				optimal T-depth			
truth table	cliffords	qubits	T-count	T-depth	cliffords	qubits	T-count	T-depth	cliffords	qubits	T-count	T-depth
#8000	120	6	70	37	51	7	12	4	81	11	12	3
#8080	48	5	28	19	23	6	8	3	29	8	8	2
#0888	132	6	77	42	47	7	12	4	85	10	12	3
#8888	12	5	7	5	9	5	4	2	17	7	4	1
#7080	60	5	35	24	25	6	8	3	45	9	8	2
#7880	182	6	105	55	70	8	12	4	89	11	12	3
#7888	24	5	14	9	22	7	8	2	35	8	8	1
#6ac06ac0	24	6	14	9	26	8	8	2	31	9	8	1
#6ac8e000	720	7	413	210	125	10	16	5	162	15	16	2
#80008000	120	6	70	37	43	8	12	4	85	12	12	3
#80808080	48	6	28	19	19	7	8	3	33	9	8	2
#88808000	582	7	336	159	72	9	12	4	99	11	12	3
#88808080	314	7	182	92	61	9	16	4	77	12	16	3
#88808880	170	6	98	50	65	8	12	4	97	11	12	3
#88888888	12	6	7	5	9	6	4	2	13	8	4	1
#a8808000	848	7	490	232	110	10	16	5	155	13	16	3
#a8808080	292	6	168	88	53	8	12	4	85	11	12	3
#a8808880	436	7	252	123	78	10	16	5	119	12	16	3
#a880a880	148	6	84	47	29	7	8	3	49	10	8	2
#a8888880	548	7	315	160	57	8	12	4	89	11	12	3
#a888a080	258	6	147	74	59	8	12	4	85	11	12	3
#a8e0c800	560	7	322	156	110	10	16	5	135	12	16	3
#aa808080	360	7	210	105	57	9	16	4	73	12	16	3
#b884a880	686	7	392	188	82	9	12	4	101	11	12	3
#bc88a080	709	7	399	192	104	10	16	5	137	13	16	3
#e0a8c880	240	6	140	75	86	10	16	5	107	13	16	2
#e1808880	372	6	217	117	86	9	12	4	113	12	12	3
#e8808000	798	6	448	226	69	8	12	4	105	12	12	3
#e8808002	721	7	420	222	92	10	16	5	113	12	16	3
#e8808080	484	7	280	142	98	10	16	4	127	13	16	3
#e8808880	410	6	238	120	82	9	12	4	113	12	12	3
#e880a880	414	7	238	120	92	10	16	5	125	12	16	3
#e880e880	250	6	140	79	98	9	12	4	125	12	12	3
#e8818880	974	7	560	259	84	10	16	4	101	12	16	3
#e881e880	504	7	287	143	121	10	16	5	154	13	16	3
#e8888880	526	7	301	151	103	10	16	5	140	13	16	3
#e8a08880	524	7	301	146	118	10	16	5	143	13	16	3
#e8c0a880	268	6	147	79	86	10	16	5	123	12	16	3
#e9a0c088	468	7	266	134	122	10	16	5	167	13	16	3
#e9c0a880	540	7	308	152	124	10	16	5	153	13	16	3
#ea808080	216	6	126	70	56	9	12	3	75	12	12	2
#eca08880	414	6	238	120	113	10	16	5	150	13	16	3
#f8808880	528	7	308	155	80	10	16	4	117	13	16	3
#f8888880	594	7	343	168	78	9	12	4	105	12	12	3
#fca08880	578	7	329	155	122	10	16	5	155	13	16	3
#2888a000	96	6	56	33	45	8	12	3	65	11	12	2
#6ac8e240	198	6	112	59	74	10	16	5	111	12	16	3
#78888888	60	6	35	19	56	9	12	4	75	13	12	2
#80000000	264	7	154	78	43	9	16	4	57	11	16	3
#80808000	316	7	182	91	59	9	16	4	73	11	16	3
#88888880	282	7	161	82	58	10	16	4	75	11	16	3
#e9808080	204	6	119	66	57	8	12	4	101	13	12	2
#eac86240	148	6	84	47	44	9	12	4	67	11	12	3
#ee84a060	198	6	112	61	90	10	16	5	119	15	16	2
Total	19002	345	10913	5506	3842	475	712	220	5260	625	712	143

Clifford gates varies according to the probabilistic outcomes of the measurements.

VI. Conclusions

We present a database containing automatically generated quantum circuits for all Boolean functions up to 5 inputs. We consider the specifications of fault-tolerant quantum computing: low T-count, low T-depth, and low number of qubits. To generate the database, we combine optimal reversible gates implementations with compilation methods in the literature. Furthermore, we propose a new XAG-based synthesis strategy for T-depth optimization. The database can be used to compile larger Boolean functions if combined with hierarchical methods. In this direction, we are working on extending the database to 6-input functions. The database is available at *https://github.com/gmeuli/stg-benchmark* and can serve as benchmark for new synthesis and optimization algorithms.

Acknowledgments

This research was supported by the Swiss National Science Foundation (200021-169084 MAJesty).

REFERENCES

[1] C. Nay, "IBM unveils world's first integrated quantum computing system for commercial use," *IBM Research Communications*, 2019.

[2] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[3] P. Ball, "First commercial ion-based quantum computer built," *Physics World*, vol. 32, no. 2, pp. 5–5, feb 2019.

[4] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, pp. 63–66, 2016.

[5] P. J. J. O'Malley *et al.*, "Scalable quantum simulation of molecular energies," *Physical Review X*, vol. 6, p. 031007, 2016.

[6] E. A. Martinez, C. A. Muschik, P. Schindler, D. Nigg, A. Erhard, M. Heyl, P. Hauke, M. Dalmonte, T. Monz, P. Zoller, and R. Blatt, "Real-time dynamics of lattice gauge theories with a few-qubit quantum computer," *Nature*, vol. 534, pp. 516–519, 2016.

[7] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," *Physical Review A*, vol. 93, p. 022311, 2016.

[8] 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.

[9] M. Amy, D. Maslov, and M. Mosca, "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1476–1489, 2014.

[10] G. Meuli, M. Soeken, M. Roetteler, and G. De Micheli, "ROS: Resource constrained oracle synthesis for quantum circuits," in *Quantum Physics and Logic*, 2019.

[11] G. Bioul, M. Davio, and J.-P. Deschamps, "Minimization of ring-sum expansions of Boolean functions," *Philips Research Reports*, vol. 28, pp. 17–36, 1973.

[12] A. Mishchenko and M. A. Perkowski, "Fast heuristic minimization of exclusive-sum-of-products," in *Reed-Muller Workshop*, 2001.

[13] S. Stergiou, K. Daskalakis, and G. K. Papakonstantinou, "A fast and efficient heuristic ESOP minimization algorithm," in *ACM Great Lakes Symposium on VLSI*, 2004, pp. 78–81.

[14] T. Sasao, "AND-EXOR expressions and their optimization," in *Logic Synthesis and Optimization*, T. Sasao, Ed. Kluwer Academic, 1993.

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

[16] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, "An algorithm for the T-count," *Quantum Information and Computation*, vol. 14, no. 15–16, pp. 1261–1276, 2014.

[17] C. Jones, "Low-overhead constructions for the fault-tolerant Toffoli gate," *Physical Review A*, vol. 87, no. 2, p. 022328, 2013.

[18] C. Gidney, "Halving the cost of quantum addition," *Quantum*, vol. 2, no. 74, pp. 10–22 331, 2018.

[19] P. Selinger, "Quantum circuits of T-depth one," *Physical Review A*, vol. 87, p. 042302, 2013.

[20] N. Abdessaied, M. Amy, M. Soeken, and R. Drechsler, "Technology mapping of reversible circuits to Clifford+T quantum circuits," in *Int'l Symp. on Multiple-Valued Logic*, 2016, pp. 150–155.

[21] 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," *Physical Review A*, vol. 52, no. 5, p. 3457, 1995.

[22] C. R. Edwards, "The application of the Rademacher-Walsh transform to Boolean function classification and threshold logic synthesis," *IEEE Trans. on Computers*, vol. 24, no. 1, pp. 48–62, 1975.

[23] R. J. Lechner, "Harmonic analysis of switching functions," in *Recent Developments in Switching Theory*, A. Mukhopadhyay, Ed. Academic Press, 1971, pp. 121–228.

[24] E. R. Berlekamp and L. R. Welch, "Weight distributions of the cosets of the $(32, 6)$ Reed-Muller code," *IEEE Trans. on Information Theory*, vol. 18, no. 1, pp. 203–207, 1972.

[25] J. A. Maiorana, "A classification of the cosets of the Reed-Muller code $\mathcal{R}(1, 6)$," *Mathematics of Computation*, vol. 57, no. 195, pp. 403–414, 1991.

[26] J. L. Walsh, "A closed set of normal orthogonal functions," *American Journal of Mathematics*, vol. 45, no. 1, pp. 5–24, 1923.

[27] M. A. Thornton, R. Drechsler, and D. M. Miller, "Computation of spectral coefficients," in *Spectral Techniques in VLSI CAD*. Springer, 2001, pp. 83–116.

[28] S. L. Hurst, D. M. Miller, and J. C. Muzio, "Spectral techniques in digital logic," 1985.

[29] G. Meuli, M. Soeken, E. Campbell, M. Roetteler, and G. De Micheli, "The role of multiplicative complexity in compiling low T-count oracle circuits," *arXiv preprint arXiv:1908.01609*, 2019.

[30] G. Meuli, B. Schmitt, R. Ehlers, H. Riener, and G. De Micheli, "Evaluating ESOP optimization methods in quantum compilation flows," in *Reversible Computation*, 2019.

[31] I. Zhegalkin, "The technique of calculation of statements in symbolic logic," *Mathe, Sbornik*, vol. 34, pp. 9–28, 1927.

[32] R. Drechsler, "Pseudo-Kronecker expressions for symmetric functions," *IEEE Transactions on Computers*, vol. 48, no. 9, pp. 987–990, 1999.