Hierarchical Synthesis of Reversible Circuits Using Positive and Negative Davio Decomposition

Mathias Soeken Robert Wille Rolf Drechsler
Institute of Computer Science, University of Bremen
28359 Bremen, Germany
{msoeken,rwille,drechsle}@informatik.uni-bremen.de

***Abstract*—Synthesis of reversible circuits is an important research area providing the basis for a design flow of this emerging technology. Recently, in the development of scalable synthesis approaches a significant step forward has been made by a hierarchical method in combination with Shannon decomposition. However, this approach leads to circuits with high costs. In this paper, we propose an alternative that additionally makes use of positive Davio and negative Davio decomposition. We show that the usage of these decomposition types offers several advantages for the synthesis of reversible circuits. Using the proposed approach, on average the number of lines can be reduced by 22%, the number of gates by 22%, and the quantum cost by 32%. In the best case, even reductions of more than 60% are possible.**

I. Introduction

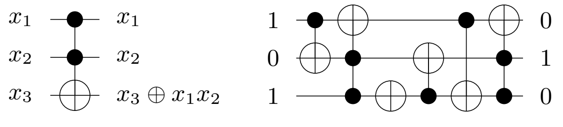
Nowadays, the continuing trend towards power dissipation is proving an ever more difficult problem in the development of smaller and more efficient digital circuits. A significant part of this problem arises from the non-ideal behavior of transistors and materials. This can be tackled by higher levels of integration and new fabrication processes. However, a much more fundamental problem has already been observed by Landauer in 1961 [1]. He proved that each time information is lost, energy is dissipated regardless of the underlying technology. More precisely, exactly $k \cdot T \cdot \log 2$ Joules of energy are dissipated for each "lost" bit of information where k is the Boltzmann constant and T is the temperature. While this amount of power currently does not sound significant, it may become crucial considering that (1) in today's circuitry millions of operations are performed in every second (and increasing processor frequencies continue to increase this number) and (2) more operations are performed with smaller transistor sizes (i.e. in a smaller area).

As a consequence, Landauer (and later Bennett [2], Fredkin [3], Toffoli [4], and others) suggested the use of reversible circuits, i.e. circuits with an equal amount of input and output signals, whereby each input assignment maps to a unique output assignment (i.e. the function represented by the circuit is a bijection). Since reversible circuits are by definition information-lossless, power dissipation resulting from Landauer's principle, as described above, can be decreased or even eliminated. Recently, a physical implementation of reversible circuits exploiting these observations has been successfully realized [5]. Furthermore, research in this domain is continuing to gain more interest because of the recent achievements in the development of quantum circuits, where reversible circuits can also be exploited [6].

Driven by this, the development of future CAD tools for this kind of circuits is an active research area. In particular, synthesis of reversible circuits is being intensely studied. Exact approaches [7], [8], [9], [10], [11] and heuristic approaches [12], [13], [14], [15] have been introduced in the past. However, they are only applicable for functions with up to about 30 variables (see also Section III for a more detailed treatment). This is mainly caused by the fact that most of these synthesis approaches rely on a truth table description. As a consequence, a hierarchical synthesis approach [16] has recently been proposed which works on a symbolic representation of the function to be synthesized, thereby entailing the use of *Binary Decision Diagrams* (BDD) [17]. BDDs decompose a given function into smaller sub-functions using the Shannon decomposition. These sub-functions can be handled by the above synthesis methods. Composing the resulting sub-circuits produces a reversible circuit representing the overall function to be synthesized. In this sense, [16] provided the first method which can realize circuits for functions with more than 100 inputs – an important step towards the development of scalable synthesis approaches. However, the resulting circuits are of high cost.

In this paper, we propose an alternative hierarchical synthesis approach for reversible circuits. In contrast to [16], positive Davio and negative Davio decomposition are additionally utilized. It has been shown that the usage of positive Davio and negative Davio leads to optimization in classical circuit synthesis [18], [19], [20]. We show why the usage of these decomposition types offers several advantages during synthesis especially when working with reversible gates. Motivated by this, a hierarchical synthesis approach is introduced exploiting these benefits. Experimental results show that with our alternative, reversible circuits with 22% fewer circuit lines, 22% fewer gates, and 32% fewer quantum cost can be realized on average. In the best case, reductions of more than 60% are possible. The algorithm is public available in the RevKit toolkit [21].

The remainder of this paper is structured as follows. Section II provides the background on reversible circuits while Section III briefly reviews recently introduced synthesis approaches. Afterwards, the application of positive Davio and negative Davio decomposition is motivated in Section IV leading to an alternative synthesis approach presented in Section V. Finally, experimental results obtained by the proposed approach are reported in Section VI and the paper is concluded in Section VII, respectively.



(a) Toffoli gate (b) Toffoli circuit

Fig. 1. Toffoli gate and Toffoli circuit

II. Reversible Circuits

Reversible circuits are digital circuits with the same number of input signals and output signals. Furthermore, reversible circuits realize bijections only, i.e. each input assignment maps to a unique output assignment. Accordingly, computations can be performed in both directions (from the inputs to the outputs and vice versa).

Reversible circuits are composed as cascades of reversible gates. The Toffoli gate [4] is widely used in the literature and also considered in this paper. A Toffoli gate over the inputs $X = \{x_1, \ldots, x_n\}$ consists of a (possibly empty) set of *control lines* $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ and a single *target line* $x_j \in X \setminus C$. The Toffoli gate inverts the value on the target line if all values on the control lines are assigned to 1 or if $C = \emptyset$, respectively. All remaining values are passed through unaltered.

Example 1: Fig. 1(a) shows a Toffoli gate drawn in standard notation, i.e. control lines are denoted by ●, while the target line is denoted by $\oplus$. A circuit composed of several Toffoli gates is depicted in Fig. 1(b). This circuit maps e.g. the input 101 to the output 010 and vice versa.

To measure the cost of a reversible circuit, different metrics are applied (sometimes depending on the addressed technology). In particular, the number of circuit lines, the number of gates, and *quantum cost* [22] are considered in the literature. The latter considers gates with more control lines to be more costly. For example, a Toffoli gate with no or only one control line has quantum cost of 1, while a Toffoli gate with two control lines has quantum cost of 5.

III. Recent Synthesis Approaches

In the last years, synthesis of reversible circuits has been intensely considered resulting in dozens of different approaches. Exact synthesis methods determine a *minimal* solution with regard to a cost metric, e.g. the number of gates or quantum cost, respectively. Methods based on a depth-first traversal [7], group theory [8], reachability analysis [9], Boolean satisfiability [11], and Quantified Boolean functions [10] have been proposed. However, since ensuring minimality causes an enormous computational overhead, all these approaches are only applicable to functions with up to six variables.

In contrast, heuristic methods enabled the synthesis of larger functions. Here, in particular the transformation-based approach introduced in [12] is of interest. The basic idea of this approach is to traverse each line of the truth table (representing the function f to be synthesized) and add gates to the circuit until the output values match the input values (i.e. until the identity is achieved). Gates are thereby chosen so that they do not alter already considered truth table lines. This strategy has been adopted and further extended leading to approaches that additionally incorporate decision diagrams [13], positive-polarity Reed-Muller expansion [14], or Reed-Muller spectra [15]. However, even with these extensions only functions with up to about 30 variables can be synthesized.

To overcome this limitation, hierarchical synthesis approaches provide a solution. Here f, the (possibly very large) function to be synthesized, is decomposed into smaller sub-functions. This decomposition is repeatedly applied until the respective sub-functions evaluate to a constant. Then, for each decomposition, a sub-circuit representing this operation can be synthesized. By composing all sub-circuits, a circuit representing the desired function f results.

Recently, an approach applying such a scheme has been introduced in [16]. Shannon decomposition has been thereby applied, whereby the sub-functions are the corresponding co-factors of f, i.e.

$$f = \overline{x}_i \cdot f_{x_i=0} + x_i \cdot f_{x_i=1}$$

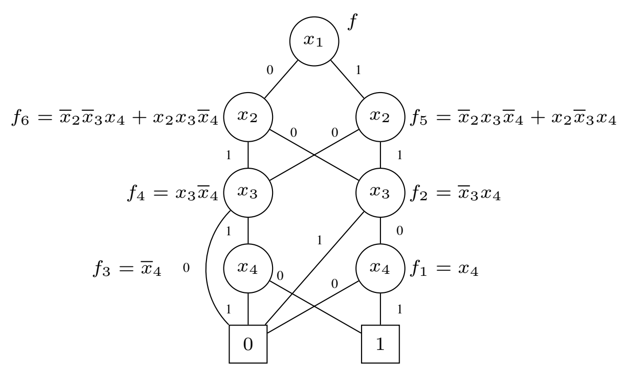
where $f_{x_i=0}$ ($f_{x_i=1}$) is the negative (positive) co-factor of f obtained by assigning x_i to 0 (1).

For this purpose, *Binary Decision Diagrams* (BDDs) [17] have been utilized. A BDD is a directed graph $G = (V, E)$ where each terminal node represents the constant 0 or 1 and each non-terminal node represents a (sub-)function. Each non-terminal node $v \in V$ has thereby two succeeding nodes $\mathrm{low}(v)$ and $\mathrm{high}(v)$. If v is representing the function f and labeled with the variable x_i, then the corresponding sub-functions represented by the succeeding nodes are the co-factors $f_{x_i=0}$ ($\mathrm{low}(v)$) and $f_{x_i=1}$ ($\mathrm{high}(v)$). Thus, a BDD naturally exposes the Shannon decomposition and therefore can be used for hierarchical synthesis. Having a BDD representing a function f as well as its sub-functions derived by Shannon decomposition, a reversible circuit for f can be obtained as shown by the following example.

Example 2: Figure 2(a) shows a BDD representing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$ as well as the respective co-factors resulting from the application of the Shannon decomposition. The co-factor f_1 can easily be represented by the primary input x_4. Having the value of f_1 available, the co-factor f_2 can be realized by the first two gates depicted in Fig. 2(b)[1]. In this manner, respective sub-circuits can be added for all remaining co-factors until a circuit representing the overall function f results. The remaining steps are shown in Fig. 2(b).

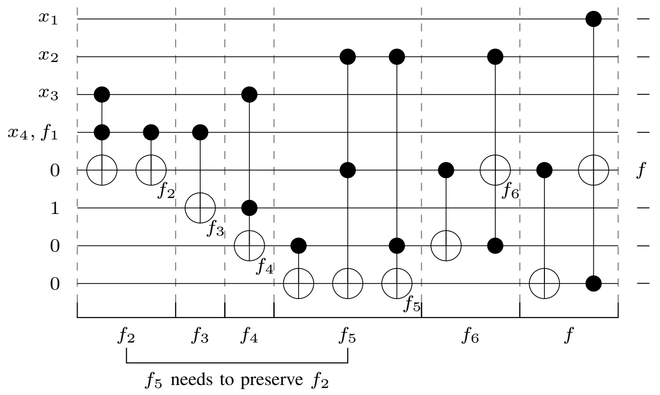
That is, to realize (possibly large) functions, decomposition is applied leading to smaller sub-functions for which existing synthesis approaches can be applied. Then, the resulting sub-circuits can be composed to realize the overall function. As can be seen, this method sometimes requires additional circuit lines with constant inputs in order to preserve (temporary) values. For example, as already shown above, an additional line is required to realize the co-factor f_2 without losing the

[1]Note that an additional circuit line is added to preserve the values of x_4 and x_3 which are still needed by the co-factors f_3 and f_4, respectively.



f
x_1
$f_6 = \overline{x}_2\overline{x}_3x_4 + x_2x_3\overline{x}_4$
$f_5 = \overline{x}_2x_3\overline{x}_4 + x_2\overline{x}_3x_4$
$f_4 = x_3\overline{x}_4$
$f_2 = \overline{x}_3x_4$
$f_3 = \overline{x}_4$
$f_1 = x_4$
0
1

(a) BDD



(b) Resulting circuit

Fig. 2. Example for BDD-based synthesis

input value x_4 (which is still needed to realize f_3). A similar issue occurs for the co-factor f_5. Here, the values of f_2 and f_4 have to be preserved since they are still needed later to realize co-factor f_6. Despite this "overhead", the approach enables synthesis of functions with more than 100 variables for the first time.

IV. Applying Positive and Negative Davio Decomposition

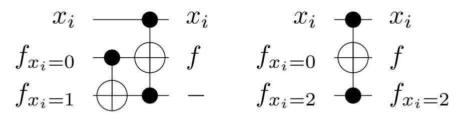
Besides Shannon, further decompositions of Boolean functions exist. In particular, positive Davio and negative Davio decomposition defined by

$$f = f_{x_i=0} \oplus x_i \cdot f_{x_i=2} \qquad \text{(pos. Davio)}$$

$$f = f_{x_i=1} \oplus \overline{x}_i \cdot f_{x_i=2} \qquad \text{(neg. Davio)}$$

with $f_{x_i=2} = f_{x_i=0} \oplus f_{x_i=1}$ have been established in the past[2]. For certain types of functions they allow more compact decompositions than the Shannon method, i.e. they enable to decompose a given function into a smaller number of different sub-functions (see e.g. [24]). Besides that, in particular for synthesis of reversible circuits these decomposition types provide some interesting properties.

[2]In fact, it has been proven that Shannon, positive Davio, and negative Davio decomposition are sufficient to efficiently decompose Boolean functions [23].



(a) Shannon (b) Positive Davio

Fig. 3. Reversible circuits realizing different decompositions

To illustrate this, consider Fig. 3 showing the minimal circuits representing the Shannon decomposition, (i.e. computing $f = \overline{x}_i \cdot f_{x_i=0} + x_i \cdot f_{x_i=1}$) and positive Davio decomposition (i.e. computing $f = f_{x_i=0} \oplus x_i \cdot f_{x_i=2}$), respectively. While for the Shannon decomposition two gates (and quantum cost of 6) are required, the positive Davio decomposition can be realized by one single gate (and quantum cost of 5) only. Furthermore, applying positive Davio, the value of $f_{x_i=2}$ is preserved. Thus, the additional circuit lines needed to preserve the respective values (such as those in the example of Fig. 2) can be saved.

Taking all this into account, it is worth considering positive and negative Davio decomposition in the context of hierarchical synthesis of reversible circuits since

- for certain functions they enable decomposing a given function into a smaller number of different sub-functions,
- in many cases they enable more compact realizations as reversible circuits, and
- they enable the preservation of the values of some co-factors without additional circuit lines so that the overall line count can be kept small.

In the next section, a hierarchical synthesis approach is introduced that utilizes positive and negative Davio decomposition and, thus, exploits these advantages.

V. Synthesis Approach

Based on the observations from the previous section, a new hierarchical synthesis approach is derived which additionally utilizes positive Davio and negative Davio decomposition. To efficiently perform the respective decompositions *Kronecker Functional Decision Diagrams* (KFDDs) [25] are applied. Similar to BDDs, KFDDs provide an efficient data-structure for the representation of Boolean functions. Additionally, KFDDs enable the decomposition of a (sub-)function not only with respect to Shannon but also with respect to positive and negative Davio.

Therefore, a decomposition type is assigned to each variable of the corresponding Boolean function. To obtain the function f represented by a node v the formula $f = \text{low}(f) \oplus x_i \cdot \text{high}(f)$ is used for positive Davio decomposition and $f = \text{low}(f) \oplus \overline{x}_i \cdot \text{high}(f)$ is used for negative Davio decomposition, respectively, where $\text{low}(f)$ and $\text{high}(f)$ are the functions represented by the descending nodes of v.

Having available a KFDD $G = (V, E)$ representing the function to be synthesized, the following steps are performed.

1) The KFDD is traversed in a depth-first manner, i.e. each node $v \in V$ is visited.

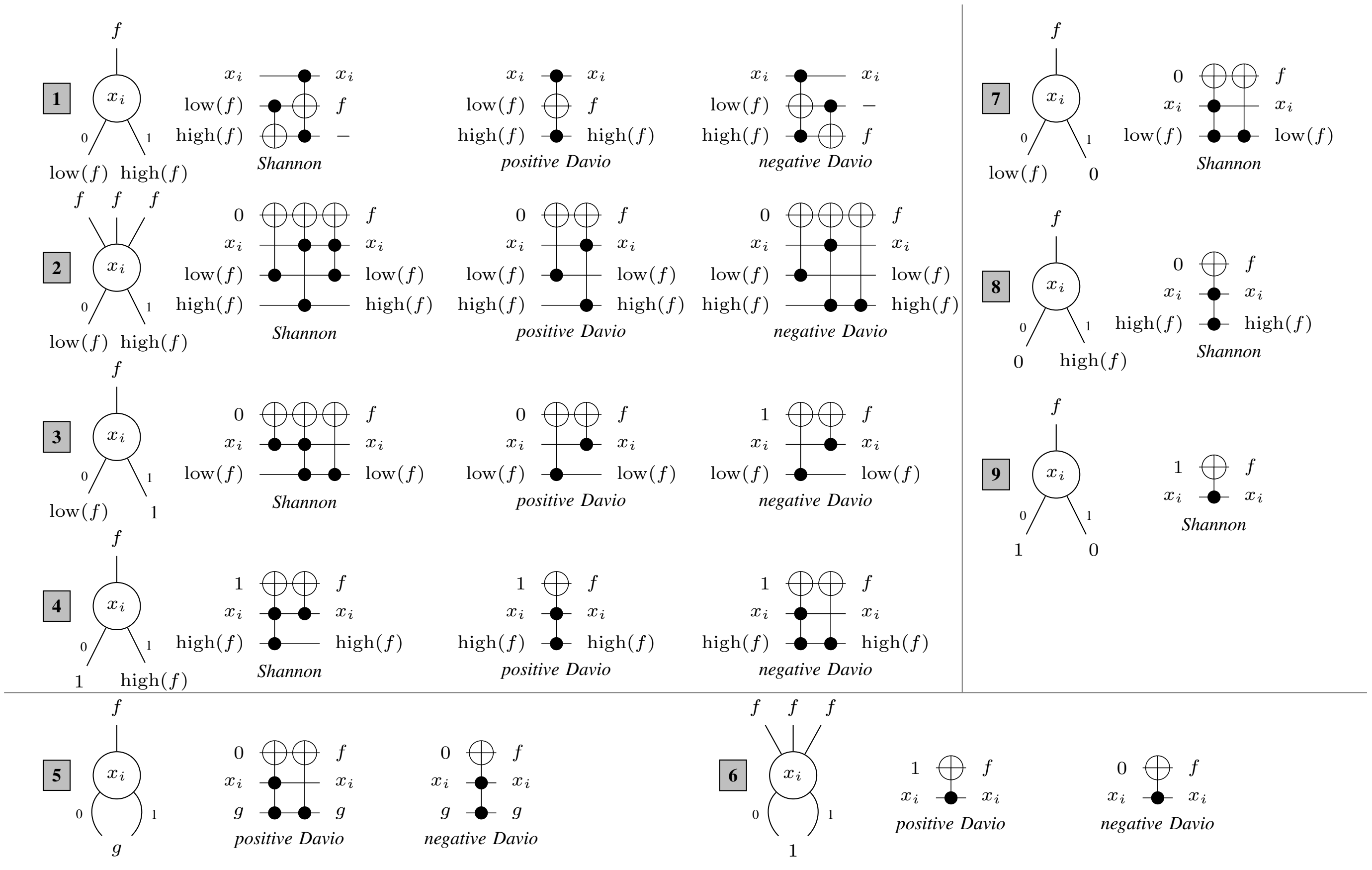


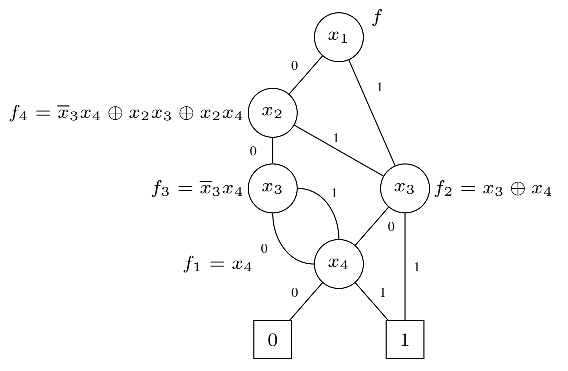
Fig. 4. Reversible cascades representing the different decompositions

2) For each node v, a cascade of reversible gates representing the respective co-factor is generated. Temporary values from previously traversed nodes (or co-factors, respectively) are thereby utilized.
3) Finally, all generated sub-circuits are composed.

Depending on the decomposition type applied to the respective nodes, different cascades result in Step 2. Fig. 4 lists the most frequent cases which may occur in the proposed approach[3]. Note thereby that some structures only occur for certain decomposition types. For example, Case 5 and Case 6 only occur if Davio decomposition is applied. Similarly, Case 7, Case 8, and Case 9 is possible with Shannon decomposition only.

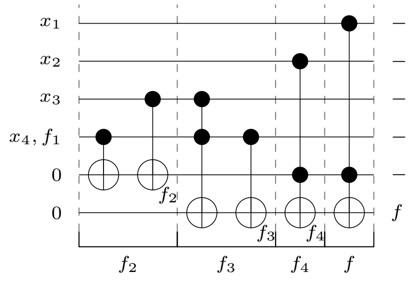
Example 3: Figure 5(a) shows a KFDD representing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$ from Example 2. Positive Davio decomposition is applied to each node. Traversing the data-structure as described above, first the co-factor f_1 is considered. This can be represented by the primary input x_4. Then, the co-factor f_2 can be realized (Case 3 from Fig. 4). Continuing this process until the depth first-traversal is completed all resulting sub-circuits can be composed. This leads to the final circuit depicted in Fig. 5(b) and representing the desired function f.

As shown by the example, using the proposed approach a more compact reversible circuit can be realized for the



f
x_1
$f_4 = \overline{x}_3x_4 \oplus x_2x_3 \oplus x_2x_4$
x_2
$f_3 = \overline{x}_3x_4$
x_3
$f_2 = x_3 \oplus x_4$
$f_1 = x_4$
x_4
0
1

(a) KFDD



(b) Resulting circuit

Fig. 5. Example for synthesis exploiting Davio decomposition

[3]Due to page limitations, cases including e.g. complement edges are omitted.

TABLE I
GATE COUNT AND QUANTUM COST OF DIFFERENT REALIZATIONS

Case	Shannon		pos. Davio		neg. Davio	
	Gates	Cost	Gates	Cost	Gates	Cost
1	2	6	**1**	**5**	2	6
2	3	11	**2**	**6**	3	7
3	3	7	**2**	**2**	**2**	**2**
4	2	6	**1**	**5**	2	6
5	–	–	2	6	1	5
6	–	–	1	1	1	1
7	2	6	–	–	–	–
8	1	5	–	–	–	–
9	1	1	–	–	–	–

considered function. More precisely, the number of lines is reduced by 2, the number of gates by 5, and the quantum cost by 17 for this simple example. However, also in general, the proposed approach leads to circuits more likely to be compact. Besides the fact that certain functions can often be decomposed into a smaller number of different sub-functions, in particular this is caused by the more compact realizations which are possible if Davio decomposition is applied.

To further illustrate this, Table I compares the resulting gates and quantum cost for the respective cases (according to Fig. 4). Even if this comparison is only an approximation (since according to the decomposition type, the respective co-factors are completely different), a clear trend can be observed: The positive Davio decomposition usually can be realized with a significantly smaller number of gates and costs, respectively. Sub-circuits representing the negative Davio decomposition are also more compact than their Shannon equivalent.

The experimental evaluation in the next section confirms these general observations by showing that significant improvements can be achieved using the proposed method.

VI. EXPERIMENTAL EVALUATION

To evaluate the proposed approach, the hierarchical synthesis method introduced in the last section has been implemented in C++ using RevKit [21] and compared to an implementation exploiting Shannon decomposition only (based on the concepts of [16]). Therefore, the CUDD package [26] has been used to perform the respective decompositions for BDDs and the PUMA decision diagram package [27] has been used to perform the respective decompositions for KFDDs. Standard optimization techniques (e.g. sifting [28]) have been thereby utilized. Benchmarks from the LGSynth package have been used. All experiments have been carried out on an Intel Core 2 Duo 2.26 GHz with 3 GB of main memory.

Table II lists the resulting numbers. The first column denotes the name of the considered functions. The following columns give the respective number of lines (*Lines*), the number of gates (*Gates*), the resulting quantum cost (*Cost*), the number of sub-functions (or nodes respectively) resulting from the decomposition (*Nodes*), as well as the run-time (*Time*, in seconds). It is thereby distinguished between the results obtained by the method from [16] (where Shannon decomposition only is applied) and the proposed method, where all three decompositions are applied. The differences are reported in the rightmost columns.

First of all, it can be concluded that run-time is negligible for all considered benchmarks. That is, performing the decompositions and adding the respective sub-circuits can be done very quickly.

Besides that, the results confirm the major assumptions made in Section IV. Using Davio decomposition enables the decomposition of the majority of the functions into a smaller number of different sub-functions (as can be seen by comparing the number of nodes). That is, a smaller number of sub-functions has to be realized and, thus, a smaller number of sub-circuits has to be composed, respectively.

Furthermore, the generally more compact realizations of the respective decompositions pay off. As a special case, consider the function *apex4*. Even though the number of nodes increases, i.e. Davio decomposition leads to more sub-functions in this case (which is an exception from the previous observation), the quantum cost can be significantly reduced. This is because the additional overhead is compensated by the much more compact realization of the Davio decomposition.

Overall, these two aspects in particular are the reasons why, in the majority of the cases, significantly smaller reversible circuits can be synthesized using the proposed approach. Only the circuits for the functions *ex4p*, *sao2*, and *table3* are exceptions. Additionally, for a small number of benchmarks (e.g. *cordic*, *rd73*, or *vg2*) only some of the cost criteria (e.g. the quantum cost) can be improved while, for the other, some increases are reported. Nevertheless, all these increases are marginal. In contrast, over all circuits, the number of lines can be reduced by 22%, the number of gates by 22%, and the quantum cost by 32% on average. In the best case (i.e. for *seq*) reductions of 57% (lines), 57% (gates), and even 65% (quantum cost) can be achieved, respectively.

VII. CONCLUSION

In this paper, we introduced an alternative hierarchical synthesis approach based on positive and negative Davio decomposition. In comparison with Shannon decomposition, this leads to the following advantages: (1) certain functions can be decomposed into a smaller number of sub-functions, (2) the respective decompositions can be realized with smaller costs, and (3) more values are implicitly preserved keeping the number of additional circuit lines small. Overall, using the proposed approach, on average, the number of lines can be reduced by 22%, the number of gates by 22%, and the quantum cost by 32%. In the best case, even reductions of more than 60% are possible.

ACKNOWLEDGMENT

This work was supported by the German Research Foundation (DFG) (DR 287/20-1).

REFERENCES

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, vol. 5, p. 183, 1961.
[2] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev*, vol. 17, no. 6, pp. 525–532, 1973.
[3] E. F. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3/4, pp. 219–253, 1982.

TABLE II
EXPERIMENTAL EVALUATION

	Shannon decomposition only [16]					Proposed Approach							
Function	Lines	Gates	Cost	Nodes	Time	Lines	Gates	Cost	Nodes	Time	ΔLines	ΔGates	ΔCost
5xp1	30	90	254	42	0.02	36	89	229	36	0.01	6	-1	-25
9sym	27	62	206	25	0.00	29	60	148	26	0.00	2	-2	-58
alu4	541	2186	7222	804	0.03	334	1149	3265	490	0.07	-207	-1037	-3957
apex1	1084	3586	11058	1394	0.42	910	3218	8350	1166	0.18	-174	-368	-2708
apex2	498	1746	5922	652	0.18	333	1011	2883	417	2.32	-165	-735	-3039
apex4	547	2551	8343	909	0.02	552	2645	7325	915	0.06	5	94	-1018
apex5	1025	2909	10349	1092	0.06	490	1262	3002	429	0.16	-535	-1647	-7347
b12	61	156	468	66	0.00	58	125	353	54	0.00	-3	-31	-115
bw	87	307	943	104	0.00	81	265	673	91	0.00	-6	-42	-270
clip	66	228	704	87	0.00	66	185	517	85	0.01	0	-43	-187
con1	16	32	96	16	0.00	15	25	77	13	0.00	-1	-7	-19
cordic	52	101	325	43	0.02	59	105	285	46	0.00	7	4	-40
cps	923	2763	8487	1071	0.03	628	2259	5771	789	0.06	-295	-504	-2716
duke2	325	975	2751	387	0.01	226	765	1989	287	0.02	-99	-210	-762
e64	195	387	907	132	0.02	193	383	891	128	0.05	-2	-4	-16
ex1010	670	2982	9766	1080	0.03	658	2883	9403	1062	0.06	-12	-99	-363
ex4p	476	1138	3494	489	0.03	525	1302	3930	543	0.22	49	164	436
ex5p	206	647	1843	242	0.02	204	635	1663	234	0.02	-2	-12	-180
inc	53	187	579	73	0.01	56	196	544	70	0.00	3	9	-35
misex1	35	104	304	36	0.00	33	87	255	32	0.00	-2	-17	-49
misex2	99	220	588	86	0.00	86	181	449	66	0.00	-13	-39	-139
misex3c	441	1513	4769	618	0.03	302	985	2853	407	0.06	-139	-528	-1916
misex3	428	1473	4661	602	0.04	431	1486	4426	612	0.08	3	13	-235
pdc	619	2080	6500	794	0.09	542	1722	4918	675	0.11	-77	-358	-1582
rd53	13	34	98	17	0.00	15	30	62	13	0.00	2	-4	-36
rd73	25	73	217	31	0.00	25	52	108	21	0.00	0	-21	-109
rd84	33	103	299	42	0.00	32	70	154	29	0.00	-1	-33	-145
sao2	74	211	667	86	0.00	77	226	726	91	0.01	3	15	59
seq	1617	5990	19362	2163	0.54	694	2529	6697	884	0.27	-923	-3461	-12665
spla	489	1709	5925	590	0.06	484	1594	4358	583	0.09	-5	-115	-1567
sqrt8	30	76	240	35	0.00	29	63	163	29	0.00	-1	-13	-77
squar5	28	81	253	35	0.00	27	62	154	26	0.00	-1	-19	-99
t481	30	52	152	21	0.01	26	35	91	16	0.01	-4	-17	-61
table3	554	1988	6276	782	0.02	592	2110	6514	827	0.08	38	122	238
table5	551	1893	5737	704	0.02	468	1749	4865	624	0.11	-83	-144	-872
vg2	198	532	1728	233	0.01	200	555	1491	236	0.04	2	23	-237
xor5	6	8	8	6	0.00	6	6	6	5	0.00	0	-2	-2
$\sum$	12152	41173	131501								-2630	-9069	-41913

[4] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*, W. de Bakker and J. van Leeuwen, Eds. Springer, 1980, p. 632, technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.

[5] B. Desoete and A. D. Vos, "A reversible carry-look-ahead adder using control gates," *INTEGRATION, the VLSI Jour.*, vol. 33, no. 1-2, pp. 89–104, 2002.

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

[7] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 22, no. 6, pp. 710–722, 2003.

[8] G. Yang, X. Song, W. N. N. Hung, and M. A. Perkowski, "Fast synthesis of exact minimal reversible circuits using group theory," in *ASP Design Automation Conf.*, 2005, pp. 1002–1005.

[9] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis." *IEEE Trans. on CAD*, vol. 25, no. 9, pp. 1652–1663, 2006.

[10] R. Wille, H. M. Le, G. W. Dueck, and D. Große, "Quantified synthesis of reversible logic," in *Design, Automation and Test in Europe*, 2008, pp. 1015–1020.

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

[12] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Design Automation Conf.*, 2003, pp. 318–323.

[13] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis," in *Design Automation Conf.*, 2004, pp. 834–837.

[14] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, 2006.

[15] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the synthesis of reversible Toffoli networks," *ACM Trans. on Design Automation of Electronic Systems*, vol. 12, no. 4, 2007.

[16] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conf.*, 2009, pp. 270–275.

[17] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. on Comp.*, vol. 35, no. 8, pp. 677–691, 1986.

[18] S.-B. Ko and J.-C. Lo, "Efficient Decomposition Techniques for FPGAs," in *HiPC*, ser. Lecture Notes in Computer Science, S. Sahni, V. K. Prasanna, and U. Shukla, Eds., vol. 2552. Springer, 2002, pp. 630–642.

[19] S.-B. Ko, "A new partitioning method for LUT-based FPGAs," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, vol. 1, 4-7 2003, pp. 103 – 106 vol.1.

[20] S.-B. Ko and J.-C. Lo, "Efficient Realization of Parity Prediction Functions in FPGAs," *J. Electronic Testing*, vol. 20, no. 5, pp. 489–499, 2004.

[21] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: A toolkit for reversible circuit design," in *Workshop on Reversible Computation*, 2010, RevKit is available at http://www.revkit.org.

[22] A. Barenco, C. H. Bennett, R. Cleve, D. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *The American Physical Society*, vol. 52, pp. 3457–3467, 1995.

[23] B. Becker and R. Drechsler, "How many decomposition types do we need?" in *European Design & Test Conf.*, 1995, pp. 438–443.

[24] B. Becker, R. Drechsler, and M. Theobald, "OKFDDs versus OBDDs and OFDDs," in *ICALP*, ser. LNCS, vol. 944. Springer Verlag, 1995, pp. 475–486.

[25] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. Perkowski, "Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams," in *Design Automation Conf.*, 1994, pp. 415–419.

[26] F. Somenzi, *CUDD: CU Decision Diagram Package Release 2.4.2*. University of Colorado at Boulder, 2001.

[27] R. Drechsler and B. Becker, "PUMA: An OKFDD-package and its implementation," in *European Design & Test Conf.*, 1995, university Booth.

[28] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Int'l Conf. on CAD*, 1993, pp. 42–47.