Gates vs. Splitters: Contradictory Optimization Objectives in the Synthesis of Optical Circuits

ARIGHNA DEB, Institute of Computer Science, University of Bremen, Bremen, Germany
ROBERT WILLE, Institute for Integrated Circuits, Johannes Kepler University, Linz, Austria Cyber Physical Systems, DFKI GmbH, Bremen, Germany
OLIVER KESZÖCZE, Institute of Computer Science, University of Bremen, Bremen, Germany Cyber Physical Systems, DFKI GmbH, Bremen, Germany
STEFAN HILLMICH, Institute of Computer Science, University of Bremen, Bremen, Germany
ROLF DRECHSLER, Institute of Computer Science, University of Bremen, Bremen, Germany Cyber Physical Systems, DFKI GmbH, Bremen, Germany

Optical circuits are considered a promising emerging technology for applications in ultra-high-speed networks or interconnects. However, the development of (automatic) synthesis approaches for such circuits is still in its infancy. Although first generic and automatic synthesis approaches have been proposed, no clear understanding exists yet on how to keep the costs of the resulting circuits as small as possible. In the domain of optical circuits, this is particularly interesting for the number of gates and the effect of so-called splitters to the signal strength. In this work, we investigate this relation by considering a variety of (existing as well as proposed) synthesis approaches for optical circuits. Our investigations show that reducing the number of gates and reducing the number of splitters are contradictory optimization objectives. Furthermore, the performance of synthesis guided with respect to gate efficiency as well as synthesis guided with respect to splitter freeness is evaluated and an overhead factor between the contradictory metrics is experimentally determined.

Categories and Subject Descriptors: B.6.3 [**Logic Design**]: Design Aids

General Terms: Design

Additional Key Words and Phrases: Optical circuit, crossbar gate, splitter, worst-case fraction

ACM Reference Format:
Arighna Deb, Robert Wille, Oliver Keszöcze, Stefan Hillmich, and Rolf Drechsler. 2016. Gates vs. splitters: Contradictory optimization objectives in the synthesis of optical circuits. J. Emerg. Technol. Comput. Syst. 13, 1, Article 11 (June 2016), 13 pages.
DOI: http://dx.doi.org/10.1145/2904445

This research was supported by the European Commission in the framework of the Erasmus Mundus cLINK project.
Authors' addresses: A. Deb (corresponding author), Institute of Computer Science, University of Bremen, 28359 Bremen, Germany; email: deb@informatik.uni-bremen.de; R. Wille, Institute for Integrated Circuits, Johannes Kepler University, A-4040 Linz, Austria and Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany; email: robert.wille@jku.at; O. Keszöcze, Institute of Computer Science, University of Bremen, 28359 Bremen, Germany and Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany; email: keszocze@informatik.uni-bremen.de; S. Hillmich, Institute of Computer Science, University of Bremen, 28359 Bremen, Germany; email: hillmich@tzi.de; R. Drechsler, Institute of Computer Science, University of Bremen, 28359 Bremen, Germany and Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany; email: drechsler@uni-bremen.de.

DOI: http://dx.doi.org/10.1145/2904445

1. INTRODUCTION

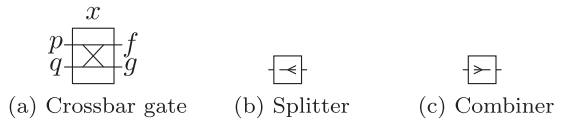
The semiconductor technology has evolved rapidly, resulting in smaller and faster devices. This has allowed Moore's law to remain valid over past decades as well as into the near future. However, the trend of miniaturizing device size is approaching its limits. Furthermore, an increasing emphasis on low-power, ultra-fast design restricts the computational capabilities of modern circuits and systems [Haensch et al. 2006]. This motivates researchers to explore alternative technologies in order to efficiently perform logical computations.

One of these alternatives, namely silicon-based integrated optics (also known as *silicon photonics*), has received significant attention in the recent past. In fact, optical technology has extensively been used in communication systems [Keiser 1991] and signal processing applications [Willner et al. 2014]. Besides that, it also finds applications in Very-Large-Scale Integration (VLSI) chips in the form of interconnects. This is evident from recent advancements in the physical design of optical interconnects and optical functional on-chip units [Kao and Chao 2014]. They allow for ultra-high-speed networks while having beneficial low-power properties. The design of corresponding VLSI chips with ultra-fast optical interconnects is in progress [Ho et al. 2013]. Based on that, the domain of optical computation finds, thus far, major applications for optical interconnects but also provides innovations to low-power and high-speed devices by expanding the application areas towards full-scale optical computing [Shimizu and Uenohara 2012; Alexoudi et al. 2013; LeGrange et al. 2014].

However, for any circuit technology (including optical circuits), synthesis is one of the most important steps in design flow. Here, for a given function, a corresponding netlist is determined with the focus on an *efficient* realization, that is, a realization at low costs. For optical circuits, the development of automatic approaches for synthesis is still in its early stages. Moreover, the underlying technological constraints are not entirely known yet (this is subject to current research). But initial models and corresponding gate libraries have already been proposed in order to start the investigation of design automation for this technology. First attempts have been introduced recently in Condrat et al. [2010]. Here, so-called *virtual gates*, that is, sub-circuits that realize important building block-functions such as AND, OR, and so on, have been considered. In a similar fashion, building blocks, such as multiplexers, adders, universal logic blocks, and so on, have been proposed in the recent past (see, e.g., Al-Zayed and Cherri [2010], Chattopadhyay [2010], Gayen and Chattopadhyay [2013], Cherri and Al-Zayed [2010], Ghasemi et al. [2012], Menezes et al. [2010], Nady et al. [2013], and Datta and Sengupta [2014]). However, just having a selected set of building blocks is not sufficient for an efficient realization of arbitrary functionality.

For this purpose, first generic and automatic synthesis approaches have been proposed. Among them is an exact solution proposed in Condrat et al. [2011], which guarantees an optimal result with respect to the number of gates in the resulting circuit, but is only applicable for Boolean functions of up to six variables. In contrast, several solutions based on *Binary Decision Diagrams* (BDDs) have been proposed in Condrat et al. [2011] and Wille et al. [2015], which do not guarantee minimality but are applicable to relatively large functions.

While these developments represent first accomplishments towards automatic synthesis of optical circuits, it remains open how to automatically realize arbitrary functions *and*, at the same time, keep the costs as small as possible. In this sense, particularly the number of gates as well as the number of splitters are of utmost importance: While the first metric corresponds to the area of the resulting chip, splitters have a significant impact on the signal strength. However, no detailed investigation on the relation between these two cost metrics has been conducted thus far.



(a) Crossbar gate (b) Splitter (c) Combiner

Fig. 1. Optical gates.

This issue is addressed in this article. For this purpose, we consider a variety of (complementary) synthesis approaches for optical circuits that are explicitly based on different function representations—including solutions reviewed above as well as two proposed synthesis methods. As a result of our observations, we can conclude that reducing the number of gates and reducing the number of splitters are contradictory optimization objectives. Following this conclusion, we present results of an experimental evaluation based on the considered synthesis approaches that provide further insights into the characteristics and properties of optical circuit synthesis. More precisely, the performance of synthesis guided with respect to gate efficiency as well as synthesis guided with respect to splitter freeness is evaluated and an overhead factor between these contradictory metrics is determined.

The remainder of this work is structured as follows. The next section reviews the basics on the models and cost metrics applied for the logic design of optical circuits. Sections 3 and 4 describe the considered synthesis approaches—first, solutions aiming for gate efficiency and, afterwards, solutions leading to splitter-free circuits are considered. Conclusions and evaluations that can be drawn from these considerations are discussed and presented in Section 5. Finally, possible directions for future work are outlined in Section 6, which also concludes this article.

2. PRELIMINARIES

To ease the development of automated methods for the synthesis of optical circuits, corresponding logic models have been proposed. This section briefly reviews the logic model as well as the correspondingly assumed cost metrics that are applied in this work.[1]

The main logic element of an optical circuit is a *crossbar gate*, which routes the optical signals between two parallel paths. The inputs of both paths can be sourced either by light (representing the logical value *true*) or darkness (representing the logical value *false*).[2] Furthermore, the routing of both paths is controlled by an electrical signal. The output of each optical signal can be read using optical receivers. Logically, a crossbar gate can be defined as follows.

Definition 2.1. A *crossbar gate* realizes a Boolean function $\mathbb{B}^3 \to \mathbb{B}^2$ composed of two optical inputs p and q, one electrical input x, and two optical outputs f and g. The signals p and q as well as f and g are connected by waveguides. Depending on the value of x, the light is assumed to move from p/q to f/g either as the identity or a switch of the input values, that is,

$$\overline{x} \Rightarrow (p \equiv f) \wedge (q \equiv g) \qquad \text{and} \qquad x \Rightarrow (p \equiv g) \wedge (q \equiv f)$$

is realized. Figure 1(a) provides the graphical representation of a crossbar gate.

Note that, in the above definition, an optical signal cannot directly be used to switch the electrical input of the crossbar gate. For this purpose, an opto-electrical interface

[1]Note that alternative logic models exists. However, for such models the results and conclusions of this work can be adapted accordingly. Hence, for sake of clarity, those gates are not explicitly considered in the following.

[2]In the following, we will denote these values by "1" and "0", respectively.

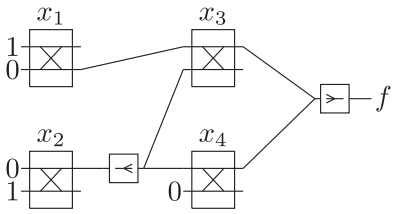


Fig. 2. Optical circuit.

would be required that, however, is considered expensive as well as slow. As a result, electrical and optical signals are never assumed to interact with each other except for the crossbar gate.

In addition to crossbar gates, *splitters* and *combiners* are also utilized as optical logic elements in order to realize logic functions.

Definition 2.2. A *splitter* divides an optical signal into two signals—each with only half of the incoming signal power. In contrast, a *combiner* merges two optical signals into a single one and, by this, inherently realizes the OR function. A splitter (combiner) may have more than two outputs (inputs). Then, in case of a splitter, the strength of the signal is divided by the number of outputs. Figures 1(b) and 1(c) provide the graphical representation of both elements.

Using these logic elements, a technology library is formed that allows us to realize arbitrary Boolean functions. As an example, Figure 2 shows an optical circuit composed of four crossbar gates, one splitter, and one combiner.

In the literature, the complexity of optical circuits is measured in terms of the following metrics [Condrat et al. 2010, 2011].

—The number of crossbar gates (denoted as gate costs): This metric is simply motivated by the fact that each gate needs to be physically realized.
—The number of splitters: A splitter causes a considerable decrease in the optical signal strength. Hence, keeping their number as small as possible is an important objective.

Although counting the number of splitters in a circuit is an obvious metric to judge the quality of a circuit, measuring their impact in the circuit is actually more important. In fact, the impact of splitters will significantly differ if, for example, 10 signals are separately split at once by a single splitter or 1 signal is split 10 times by 10 splitters. Because of this, a more elaborate metric, which has been used in Wille et al. [2015] for the first time, is considered more important in the following.

—Worst-case fraction: This metric measures the worst-case signal strength at one of the circuit's primary outputs. It is determined as follows: Each splitter with n outputs produces an output signal of strength $1/n$. The circuit's worst-case fraction is then determined by following all paths from the primary inputs to the primary outputs and multiplying all signal strengths produced by splitters visited on the current path. Afterwards, the minimum value of all paths is considered the worst-case fraction.

Although the above metric is just an approximation that may not correspond perfectly to the physical implementation, it gives a better idea of the resulting signal strength than just counting the number of splitters. Furthermore, to keep the model simple, combiners are assumed not to change the signal strength. This makes the worst-case fraction a pessimistic approximation, that is, the actual signal strength may be better than the computed one.

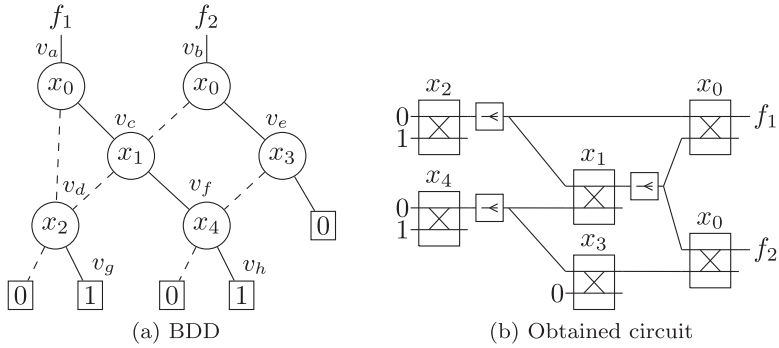


Fig. 3. Gate-efficient BDD-based synthesis.

3. GATE-EFFICIENT SYNTHESIS

In a first series of investigations, we consider synthesis approaches aiming for keeping the number of gates as small as possible. For each synthesis solution, the applied function description is reviewed before the actual synthesis process is described.

3.1. BDD-Based Synthesis

One of the first (scalable) synthesis approaches was proposed in Condrat et al. [2010, 2011], and it relies on BDDs. A BDD is a directed acyclic graph $G = (V, E)$ with the following properties:

(1) Each node $v \in V$ is either a terminal or a non-terminal.
(2) Each terminal node $v \in V$ is labeled by a value of either 0 or 1 and has no outgoing edges.
(3) Each non-terminal node $v \in V$ is labeled by a Boolean variable $x_i \in X$ and has exactly two succeeding nodes, low(v) and high(v).
(4) The directed edges to these succeeding nodes are called *low edge* and *high edge*, represented by dashed and solid lines, respectively.
(5) A Shannon decomposition [Shannon 1938]

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

is carried out in each non-terminal node. The non-terminal nodes (labeled by x_i) define the variable used in the decomposition, while the nodes low(v) and high(v) represent the functions $f_{x_i=0}$ and $f_{x_i=1}$, respectively.

Furthermore, if v is representing the function f and is labeled with the variable x_i, the corresponding sub-functions represented by the succeeding nodes are also called the *co-factors* $f_{x_i=0}$ (low(v)) and $f_{x_i=1}$ (high(v)). BDDs constitute an efficient representation for Boolean functions as they represent redundant sub-functions by the same sub-graph. This eventually leads to *shared nodes*, that is, nodes with more than one predecessor.

Having a BDD $G = (V, E)$, a corresponding optical circuit can easily be derived from it. In fact, the Shannon decomposition applied in each BDD node can directly be realized by a crossbar gate. Hence, an optical circuit can be derived by traversing the decision diagram in a depth-first fashion and substituting each node $v \in V$ with a corresponding crossbar gate. If a shared node occurs, then the optical signal has to be split accordingly. Combining all gates and connecting the respective signals eventually leads to an optical circuit realizing the desired function.

Example 3.1. Figure 3(a) shows a BDD representing the function $f : \mathbb{B}^5 \to \mathbb{B}^2$ with $f_1 = \bar{x}_0 x_2 + x_0 \bar{x}_1 x_2 + x_0 x_1 x_4$ and $f_2 = \bar{x}_0 \bar{x}_1 x_2 + \bar{x}_0 x_1 x_4 + x_0 \bar{x}_3 x_4$. Figure 3(b) shows

	x_1	x_2	x_3	f_1	f_2
C_1	1	–	1	1	1
C_2	0	1	–	1	0
C_3	1	–	0	0	1

(a) SOP

	x_1	x_2	x_3	h_1	h_2
C_1	–	0	1	1	1
C_2	1	0	–	1	1
C_3	1	–	1	0	1

(b) ESOP

Fig. 4. Two-level function descriptions.

the optical circuit obtained from this BDD. For example, the co-factor represented by the node v_d can easily be realized by the left-most crossbar gate shown at the top of Figure 3(b). Since this node also represents a shared node, a splitter is added in order to realize the co-factors represented by v_a and v_c. In a similar fashion, other nodes are realized, which finally results in the circuit shown in Figure 3(b).

In BDD-based synthesis, the need for splitters is an obvious drawback. Considering practically relevant functions, the BDD representation usually includes a large amount of shared nodes. This directly corresponds to the amount of splitters. Hence, applying this synthesis method leads to optical circuits where certain output signals have a barely noticeable signal strength. On the other hand, sharing allows for a rather compact realization with respect to the number of required gates.

3.2. Synthesis Based on Two-Level Descriptions

Besides BDDs, also so-called two-level function descriptions such as *Sum of Products* (SOP) or *Exclusive Sum of Products* (ESOP) are established means that also allow for the efficient representation of large Boolean functions. An SOP represents a Boolean function in terms of a disjunction (OR) of conjunctions (AND) of literals. A *literal* is either a propositional variable or its negation. That is, an SOP is an AND-OR expression. The conjunction of literals is usually called a *product* or a *cube* and is denoted by C in the following. Vice versa, an ESOP represents a Boolean function in terms of exclusive disjunction (XOR) of products and, hence, is an AND-XOR expression.

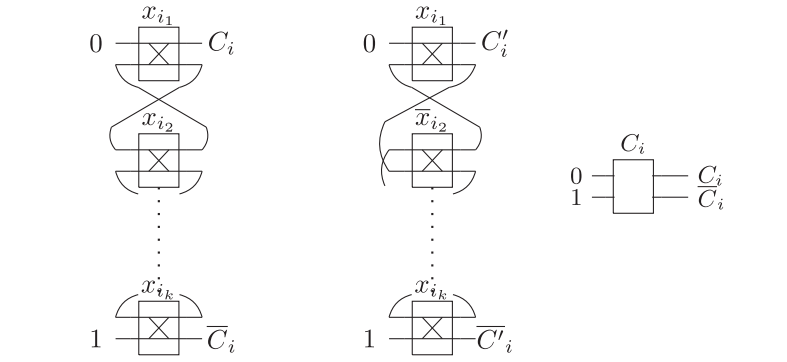
Example 3.2. We illustrate both two-level function representations by the functions f and h, which are represented by the

—SOP as shown in Figure 4(a) and
—ESOP as shown in Figure 4(b), respectively.

More precisely, the column on the left-hand side shows the respective products, where a "1" on the ith position denotes a positive literal (i.e., x_i) and a "0" denotes a negative literal (i.e., $\overline{x}_i$), respectively. A "—" denotes that the respective variable is not included in the product. The right-hand side gives the respective primary output patterns to which the OR or XOR is applied.

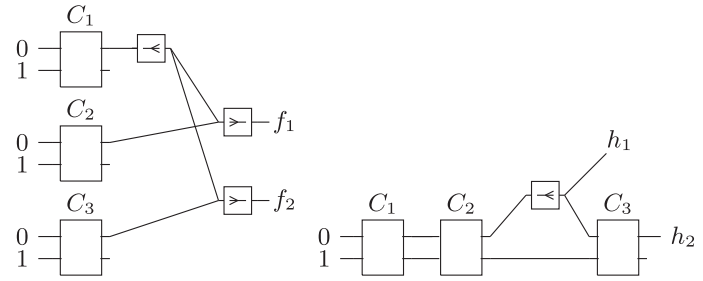
Since both SOP and ESOP rely on the realization of the respective products, we first describe the synthesis of the conjunction of literals. Afterwards, a way to realize the disjunction or exclusive disjunction of the resulting structures is described, respectively.

3.2.1. Realization of Product Terms. Without loss of generality, consider a product $C_i = x_{i_1} x_{i_2} x_{i_3} \cdots x_{i_k}$ composed of positive literals only. Such a product can be realized by an optical circuit of crossbar gates as shown in Figure 5(a). For each literal $x_{i_j} \in C_i$, a crossbar gate is added. The respective inputs/outputs are connected in a fashion so the input value 1 at the bottom of the circuit in Figure 5(a) is passed through all gates to the output C_i iff all literals are indeed assigned 1. In order to additionally support negative literals, the respective signal connections have to be flipped as illustrated in



(a) Positive literals only (b) Mixed literals (c) Functional block

Fig. 5. Realization of product terms.



(a) Result of SOP-based synthesis (b) Result of ESOP-based synthesis

Fig. 6. Realization of two-level descriptions.

Figure 5(b) for a product $C'_i = x_{i_1}\overline{x}_{i_2}x_{i_3}\cdots x_{i_k}$ with a negative literal $\overline{x}_{i_2}$. In the following, a realization of a product $x_{i_j} \in C_i$ is represented by a functional block as depicted in Figure 5(c).

3.2.2. Realization of the SOP. Having the building blocks of all product terms C_i of an SOP $f = C_1 \vee \cdots \vee C_d$, the entire function can be realized by adding logic that ORs the output signals of all those blocks. Since the OR operation is inherently realized by a combiner, this can easily be conducted. However, for SOPs representing multiple-output functions $f : \mathbb{B}^n \to \mathbb{B}^m$, products might be required more than once. Then, a splitter is added to make the intermediate results at the outputs of the building blocks available to all functions relying on it. The following example illustrates the idea.

Example 3.3. Consider again the SOP shown in Figure 4(a). Figure 6(a) shows the optical circuit obtained from this SOP. The building blocks denoted C_1, C_2, C_3 realize the corresponding products from the left-hand side of Figure 4(a). Since the first product C_1 belongs to both outputs, a splitter is added to derive two copies of the signal value C_1. Afterwards, combiners merge the optical signals in order to realize the disjunction of C_1 and C_2 (realizing f_1) and of C_1 and C_3 (realizing f_2).

3.2.3. Realization of the ESOP. For ESOPs, the respective exclusive-OR combination of the products is not as trivial as for SOPs. In fact, in order to realize a cascade of XOR operations, explicit crossbar gates as shown in Figure 7 are required. This structure can accordingly be applied to the building blocks representing the respective structure.

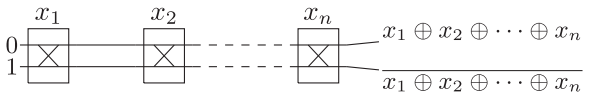


Fig. 7. XOR circuit.

If products are required more than once, then splitters are added. Again, an example illustrates the idea.

Example 3.4. Consider again the ESOP shown in Figure 4(b). Figure 6(b) shows the optical circuit obtained from this ESOP. The building blocks denoted by C_1, C_2, and C_3 realize the corresponding products from the left-hand side of Figure 4(b). To realize the exclusive-OR of products C_1 and C_2, corresponding blocks are cascaded according to the sketch in Figure 7. Since this exclusive-OR affects both outputs, a splitter is applied to derive the two copies of the corresponding signal. One copy realizes the function h_1, and the other one is used (together with the building block C_3) to realize the function h_2.

Note that the realizations of the AND and the XOR are similar to the virtual gate synthesis presented in Condrat et al. [2011], where variants working on two variables only were introduced. However, since no sharing is exploited, the corresponding results obtained by virtual gate synthesis would always be equal to or worse than that by the proposed (E)SOP-based synthesis. Because of that, virtual gate synthesis is not further considered in the following.

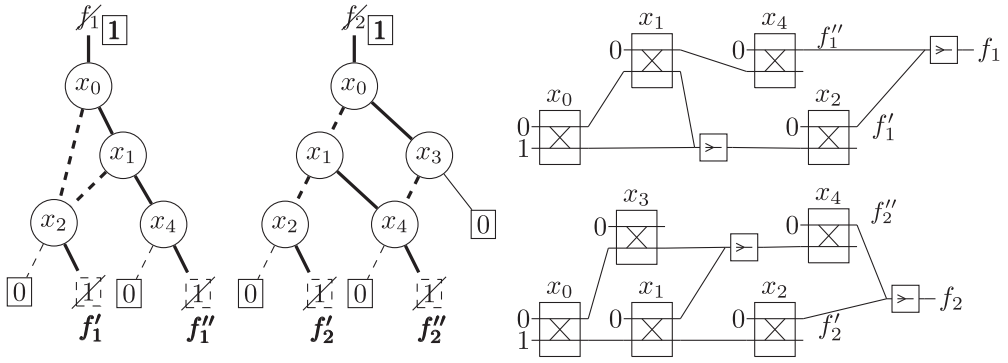
4. SPLITTER-FREE SYNTHESIS

The synthesis approaches reviewed and proposed above allow for an automatic synthesis of rather complex Boolean functionality. Redundancies are kept small (by means of shared nodes in the BDD-based approach or by means of shared products in the synthesis based on two-level descriptions). Hence, the resulting circuits are efficient with respect to the number of gates. But, in contrast, a significant amount of splitters is required. Hence, the resulting circuits may have severe issues with respect to the signal strength. Accordingly, in a second series of investigations, we consider approaches aiming for avoiding splitters at all.

4.1. BDD-Based Synthesis

Splitter-free synthesis using BDDs is possible as shown by the concepts introduced in Wille et al. [2015]. Here, the facts are exploited that (1) the function to be realized is defined by the 1-paths of the corresponding BDD and that (2) it does not matter whether these paths are traversed from the root to the leafs of the BDD or vice versa. For a given BDD, representing a function f, this fact can be exploited by a reverse consideration of the BDD, that is, by re-labeling the root nodes with $\boxed{1}$ and the $\boxed{1}$-terminals with f', f'', and so on. Then, synthesis can be conducted as described in Section 3.1 with the difference that now combiners rather than splitters are added for shared nodes. The overall function is eventually realized by ORing the functions f', f'', and so on (which again can be realized by a combiner). The following example illustrates the idea.

Example 4.1. Consider again the function f_1 as discussed in Example 3.1. Its corresponding BDD is shown at the left-hand side of Figure 8(a). The bold edges indicate the three 1-paths. Considering these paths in the direction from the root to the terminals yields the two functions $f_1' = x_2(\overline{x_0} \vee x_0\overline{x_1})$ and $f_1'' = x_0x_1x_4$ with $f_1' \vee f_1'' = f_1$. This interpretation of the BDD can be realized without any splitters as shown at the top of Figure 8(b). Following the 1 signal through the circuit corresponds directly to the 1-paths of the BDD.



(a) BDD(s) for $f_1 = f_1' \vee f_1''$ and $f_2 = f_2' \vee f_2''$ (b) Obtained circuit(s)

Fig. 8. Reverse BDD-based synthesis.

This scheme indeed enables BDD-based synthesis without the need for a single splitter. However, problems occur when multi-output functions are considered. Here, it remains unclear whether a $\boxed{1}$-terminal corresponds, for example, to a sub-function f_1, a sub-function f_2, or both. As a consequence, shared nodes in sub-trees that affect more than one sub-function cannot be handled and have to be explicitly realized. Again the example illustrates the idea.

Example 4.2. Consider again the BDD shown in Figure 3(a). The shared node v_c obviously affects both sub-functions f_1 and f_2. As a consequence, it cannot uniquely be determined whether the $\boxed{1}$-terminals are supposed to be associated to f_1 or to f_2—the splitter-free BDD-based synthesis as sketched above cannot be applied. Hence, in order to realize multi-output functions, corresponding BDDs such as shown in Figure 3(a) have to be considered as shown in Figure 8(a), that is, by means of separate BDDs for each sub-function. Then, the synthesis scheme sketched above can be applied and finally results in the circuit shown in Figure 8(b).

Obviously, such a splitter-free synthesis leads to additional redundancy and, hence, an increase in the number of gates.

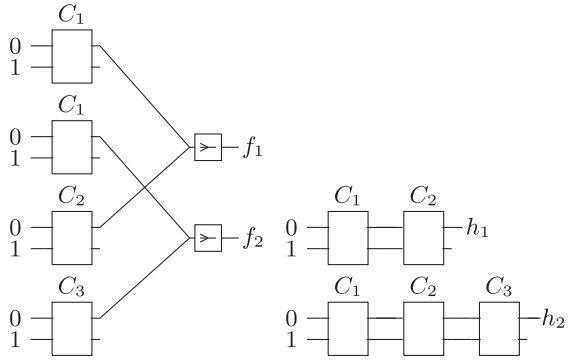
4.2. Synthesis Based on Two-Level Descriptions

In order to realize a splitter-free optical circuit from a two-level description, products are similarly realized as described in Section 3.2.1. Shared products, however, that is, products that are applied to more than one function, are handled in a different fashion. Since splitters shall not be used in order to apply copies of signals to more than one output, products have to be explicitly computed multiple times as needed. For this purpose, a building block realizing a product C_i might be added more than once to the circuit.

Example 4.3. Re-consider the functions depicted in Figure 4(a) and Figure 4(b), respectively. In Figure 4(a), the product C_1 is needed twice (by f_1 and f_2). To provide two copies of signal C_1, the corresponding functional block C_1 is created twice. This ultimately leads to the circuit depicted in Figure 9(a). In a similar fashion, ESOP-based synthesis is applied for Figure 4(b), leading to the circuit shown in Figure 9(b).

5. DISCUSSION AND EVALUATION

Automatic synthesis approaches are supposed to lead to efficient circuit realizations of the desired functionality. In the domain of logic design of optical circuits, “efficient” is



(a) Result of SOP-based synthesis (b) Result of ESOP-based synthesis

Fig. 9. Splitter-free realization of two-level descriptions.

particularly defined by the gate costs as well as the worst-case fraction of the resulting circuit description.[3] The sections above presented several methods of how to approach these objectives. As a result of these investigations, we can conclude that there does not yet exist a synthesis scheme which satisfies *both* objectives at the same time. Synthesis of optical circuits either

—reduces the gate costs of the circuit at the expense of splitters and, hence, a significantly larger worst-case fraction or
—keeps the fraction at a minimum (no splitters at all) at the expense of additional gates.

Besides that, it is unlikely that this situation ever can be changed substantially. In fact, a reduction in the number of gates can only be achieved if redundant structures are exploited so recurring intermediate results are re-used. This, however, requires a fanout that, in the domain of optical circuit design, can only be realized in terms of a splitter.

Hence, it remains to evaluate the overhead factor between these contradictory optimization objectives. To this end, we (re-)implemented all the approaches discussed in Section 3 and Section 4 and applied them[4] to benchmark functions taken from the LGSynth library. The obtained results are summarized in Table I, where the first columns provide the name of the benchmark (denoted by *Benchmark*), its number of primary inputs (denoted by *n*), as well as its number of primary outputs (denoted by *m*). The remaining columns give the number of gates (denoted by *Gates*) as well as the denominator of the worst-case fraction (denoted by *Fract.*) of the circuits generated by the approaches. Note that no worst-case fraction is provided in case of the methods from Section 4, since here always circuits without splitters and, hence, with a worst-case fraction of 1, are obtained. Runtimes are not shown as all results were obtained in negligible time, that is, just a few CPU seconds.

Based on these numbers, two clear conclusions on the relation between gate costs and the worst-case fraction can be drawn.

[3]Other metrics such as insertion loss, bending loss, dispersion, and so on, may be considered during the physical design of optical circuits.
[4]More precisely, we re-implemented the BDD-based approaches as described in Condrat et al. [2011] and Wille et al. [2015], while the SOP- and ESOP-based approaches have been developed from scratch.

Table I. Experimental Evaluation

			Gate-efficient (Section 3)						Splitter-free (Section 4)					
			BDD-based		ESOP-based		SOP-based		BDD-based	ESOP-based	SOP-based	Overhead factor		
Benchmark	n	m	Gates	Fract.	Gates	Fract.	Gates	Fract.	Gates	Gates	Gates	BDD	ESOP	SOP
cps	24	109	2318	184320	13684	$>10^3$	7141	1	3606	15432	7141	1.6	1.1	1.0
cordic	23	2	80	10240	10619	2	18369	1	82	21189	18369	1.0	2.0	1.0
cm151a	19	9	93	16	104	2	227	1	94	108	227	1.0	1.0	1.0
spla	16	46	681	6480	9213	40	34947	12	1090	9747	220971	1.6	1.1	6.3
cm163a	16	13	50	60	90	14	100	1	70	109	100	1.4	1.2	1.0
pcler8	16	5	58	4	51	6	145	1	59	57	145	1.0	1.1	1.0
pdc	16	40	705	18900	8802	16	33019	12	1118	9537	207376	1.6	1.1	6.3
t481	16	1	32	1296	40	1	4752	1	32	40	4752	1.0	1.0	1.0
cmb	16	4	47	2	36	2	126	1	48	48	126	1.0	1.3	1.0
ryy6	16	1	23	96	328	1	624	1	23	328	624	1.0	1.0	1.0
in0	15	11	526	99792	2048	18	897	6	625	2176	1872	1.2	1.1	2.1
f51m	14	8	1219	244992	2892	6	18141	1	1605	2955	18141	1.3	1.0	1.0
alu4	14	8	1352	$>10^7$	4150	12	7875	1	1534	4651	7875	1.1	1.1	1.0
misex3c	14	14	847	367200	11387	96	1304	4	970	11839	1764	1.2	1.0	1.4
misex3	14	14	1301	$>10^7$	11601	48	17971	1	1976	11826	17971	1.5	1.0	1.0
tial	14	8	1354	$>10^7$	4022	4	4530	2	1677	4632	5334	1.2	1.2	1.2
cu	14	11	65	10	102	8	114	1	97	160	114	1.5	1.6	1.0
table3	14	14	941	$>10^7$	8138	216	2001	11	1996	8610	7381	2.1	1.1	3.7
co14	14	1	27	4096	182	1	196	1	27	182	196	1.0	1.0	1.0
add6	12	7	371	20790	853	32	2196	1	372	960	2196	1.0	1.1	1.0
adr4	8	5	101	80	138	8	340	1	129	155	340	1.3	1.1	1.0
alu2	10	6	257	11616	454	4	2006	1	286	481	2006	1.1	1.1	1.0
alu3	10	8	143	168	307	6	279	2	159	310	286	1.1	1.0	1.0
apla	10	12	211	480	349	16	166	3	262	431	355	1.2	1.2	2.1
dc1	4	7	27	28	62	18	28	5	48	99	97	1.8	1.6	3.5
dc2	8	7	69	160	257	8	213	3	103	297	311	1.5	1.2	1.5
dist	8	5	195	1440	589	16	706	4	247	707	986	1.3	1.2	1.4
dk17	10	11	127	84	170	4	103	3	162	208	197	1.3	1.2	1.9
ex1010	10	10	1079	370944	16445	288	8100	6	1614	16785	14710	1.5	1.0	1.8
5xp1	7	10	88	288	191	6	296	1	113	198	296	1.3	1.0	1.0
apex4	9	19	1021	67932	27037	216	3703	11	1610	27494	14960	1.6	1.0	4.0
bw	5	28	114	156	646	$>10^3$	240	5	253	934	413	2.2	1.5	1.7
hwb6	6	6	71	192	362	8	378	6	136	392	1152	1.9	1.1	3.0
inc	7	9	89	33	252	32	189	5	119	317	562	1.3	1.3	3.0
rd84	8	4	59	384	295	3	2040	3	71	297	3286	1.2	1.0	1.6
sao2	10	4	154	10080	389	8	423	2	182	483	532	1.2	1.2	1.3
sqr6	6	12	72	60	169	6	202	4	104	193	304	1.4	1.1	1.5
urf1	9	9	741	12600	8155	128	4599	9	1030	8341	20736	1.4	1.0	4.5
urf2	8	8	386	3264	2926	64	2040	8	548	3028	8192	1.4	1.0	4.0

—*Gate-overhead for splitter-free circuits:* Obviously, generating circuits with no splitters at all (and, hence, with an optimal worst-case fraction) causes an increase in the number of gates. The evaluation unveils that the ESOP-based and the BDD-based synthesis show the best tradeoff in this regard (with an increase in the number of gates by an average factor of 1.2 and 1.3, respectively). Only in a few cases does that factor exceed 2. SOP-based synthesis performs a bit worse but has a similar gate overhead on average. That is, splitter-free synthesis of optical circuits generally causes an overhead factor between 1 and 2 regarding the number of gates.
—*Gate-efficient synthesis is infeasible:* Although gate-efficient BDD-based synthesis leads to better circuits with respect to the number of gates, almost all results obtained by it are infeasible. In fact, except for very few cases (e.g., *cmb*, *pcler8*), the resulting circuits lead to a worst-case signal fraction that will hardly be measurable in physical implementations. Similar observations can be made for ESOP-based synthesis (although not that drastically). Only the gate-efficient SOP-based synthesis leads to circuits with an acceptable worst-case fraction in most of the cases. However, here the gate costs are still larger than the gate costs in the splitter-free BDD-based synthesis. That is, considering the existing synthesis approaches, an explicit focus on gate-efficient synthesis methods (at the expense of fraction) is not recommendable.

6. CONCLUSION AND FUTURE WORK

In this work, we investigated the synthesis of optical circuits with respect to the two most important costs metrics: the number of gates and the worst-case fraction. To this end, we considered a variety of synthesis approaches that are optimized for either of these cost metrics.

Our investigations and evaluations eventually confirmed that both metrics represent *contradictory* optimization objectives. The results additionally show that, thus far, splitter-free synthesis is the most suitable synthesis paradigm, while gate-efficient synthesis often leads to circuits that are way beyond practical applicability (due to a dramatically bad worst-case fraction), while it does not lead to significantly smaller circuits with respect to the number of gates. The overhead caused by splitter-free synthesis compared to gate-efficient synthesis has also been observed to range between a factor of 1 and 2 on average. As a "best practice," the splitter-free BDD-based synthesis should be chosen.

These results give a better understanding of the characteristics and properties of optical circuit synthesis and, by this, provide the basis for various future work in this direction. In particular, the development of "tradeoff" approaches that realize circuits representing a "compromise" between the contradictory optimization objectives is a logical next step.

REFERENCES

A. Al-Zayed and A. Cherri. 2010. Improved all-optical modified signed-digit adders using semiconductor optical amplifier and Mach-Zehnder interferometer. *Optics Laser Technol.* 42, 5 (2010), 810–818.

T. Alexoudi, S. Papaioannou, G. T. Kanellos, A. Miliou, and N. Pleros. 2013. Optical cache memory peripheral circuitry: Row and column address selectors for optical static RAM banks. *J. Lightwave Technol.* 31, 24 (2013), 4098–4110.

T. Chattopadhyay. 2010. All-optical symmetric ternary logic gate. *Optics Laser Technol.* 42, 6 (2010), 1014–1021.

Abdallah K. Cherri and Ayman S. Al-Zayed. 2010. Circuit designs of ultra-fast all-optical modified signed-digit adders using semiconductor optical amplifier and Mach-Zehnder interferometer. *Optik (Stuttg)* 121, 17 (2010), 1577–1585.

Christopher Condrat, Priyank Kalla, and Steve Blair. 2010. Exploring design and synthesis for optical digital logic. In *International Workshop on Logic Synthesis*.

Christopher Condrat, Priyank Kalla, and Steve Blair. 2011. Logic synthesis for integrated optics. In *Great Lakes Symposium on VLSI*. ACM, New York, NY, 13–18.

K. Datta and I. Sengupta. 2014. All optical reversible multiplexer design using Mach-Zehnder interferometer. In *Intl. Conference on VLSI Design*. 539–544.

D. K. Gayen and T. Chattopadhyay. 2013. Designing of optimized all-optical half adder circuit using single quantum-dot semiconductor optical amplifier assisted Mach-Zehnder interferometer. *J. Lightwave Technol.* 31, 12 (2013), 2029–2035.

M. A. Ghasemi, R. Khodadadi, and H. A. Banaei. 2012. Design and simulation of all optical multiplexer based on one-dimensional photonic crystal for optical communications systems. *Int. J. Eng. Res. Appl.* 2, 6 (2012), 960–968.

Wilfried Haensch, Edward J. Nowak, Robert H. Dennard, Paul M. Solomon, Andres Bryant, Omer H. Dokumaci, Arvind Kumar, Xinlin Wang, Jeffrey B. Johnson, and Massimo V. Fischetti. 2006. Silicon CMOS devices beyond scaling. *IBM J. Res. Dev.* 50, 4.5 (2006), 339–361.

R. Ho, P. Amberg, E. Chang, P. Koka, J. Lexau, G. Li, F. Y. Liu, H. Schwetman, I. Shubin, H. D. Thacker, X. Zheng, J. E. Cunningham, and A. V. Krishnamoorthy. 2013. Silicon photonic interconnects for large-scale computer systems. *IEEE Micro* 33, 1 (2013), 68–78.

Y. H. Kao and H. J. Chao. 2014. Design of a bufferless photonic clos network-on-chip architecture. *IEEE Trans. Comput.* 63, 3 (2014), 764–776.

G. Keiser. 1991. *Optical Fiber Communications*. McGraw-Hill, New York, NY.

J. D. LeGrange, M. Dinu, T. Sochor, P. Bollond, A. Kasper, S. Cabot, G. S. Johnson, I. Kang, A. Grant, J. Kay, and J. Jaques. 2014. Cascaded all-optical operations in a hybrid integrated 80-Gb/s logic circuit. *Optics Express* 22, 11 (2014), 13600–13615.

J. W. M. Menezes, W. B. Fraga, A. C. Ferreira, G. F. Guimares, A. F. G. F. Filho, C. S. Sobrinho, and A. S. B. Sombra. 2010. All-optical half adder using all-optical XOR and AND gates for optical generation of sum and carry. *Fiber Integr. Optics* 29, 4 (2010), 254–271.

M. Nady, K. F. A. Hussein, and A. E. A. Ammar. 2013. Ultrafast all-optical full adder using quantum-dot semiconductor optical amplifier-based Mach-Zehnder interferometer. *Progr. Electromagn. Res. B* 54 (2013), 69–88.

C. E. Shannon. 1938. A symbolic analysis of relay and switching circuits. *Trans. AIEE* 57, 12 (1938), 713–723. DOI:10.1109/T-AIEE.1938.5057767

S. Shimizu and H. Uenohara. 2012. All-optical signal regenerator for differential phase-shift keying format employing a differential encoding circuit with SOA-MZIs. *IEEE J. Quant. Electron.* 48, 9 (2012), 1128–1136.

R. Wille, O. Keszöcze, C. Hopfmuller, and R. Drechsler. 2015. Reverse BDD-based synthesis for splitter-free optical circuits. In *Design Automation Conference (ASP-DAC), Asia and South Pacific*. 172–177.

A. E. Willner, S. Khaleghi, M. R. Chitgarha, and O. F. Yilmaz. 2014. All-optical signal processing. *J. Lightwave Technol.* 32, 4 (2014), 660–680.

Received October 2015; revised March 2016; accepted March 2016