Reverse BDD-based Synthesis for Splitter-free Optical Circuits

Robert Wille Oliver Keszocze Clemens Hopfmuller Rolf Drechsler
Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
Cyber Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{rwille,keszocze,drechsle}@informatik.uni-bremen.de

***Abstract*—With the advancements in silicon photonics, optical devices have found applications e.g. for ultra-high speed and low-power interconnects as well as functional computations to be realized on-chip. Caused by the increasing complexity of the underlying functionality, also the need for computer-aided design methods for this technology rises. Motivated by that, initial work on the development of synthesis methods for optical circuits has been performed. But all approaches proposed thus far suffer e.g. from unsatisfactory synthesis results or restricted scalability. In particular, splittings in the resulting circuits which degrade the optical signals into hardly measurable fractions prevent an efficient and scalable synthesis for optical circuits. In this work, we present a synthesis approach based on *Binary Decision Diagrams* (BDDs) that overcomes these obstacles. The approach yields circuits that rely on a total of zero splitters – at the expense of a moderate increase in the number of optical gates. Experiments confirm that, by this, an efficient and scalable synthesis scheme for optical circuits eventually becomes available.**

I. Introduction

In the last decades, the semiconductor industry witnessed impressive developments. Devices continued to become smaller and faster (known as Moore's law [1]) – a trend that will remain valid in the near future, but already is showing its limitations [2]. Consequently, researchers and engineers already started exploring alternative directions and technologies to be used in order to efficiently perform logical computations.

As a result, silicon-based integrated optics (also known as *silicon photonics*) received significant interest. This resulted in the physical design of optical interconnects and optical functional on-chip units [3] which allow for ultra-high-speed networks while having beneficial low-power properties. VLSI chips with ultra-fast optical interconnects have already been announced and will enter the market soon [4]. Innovations like [5], [6], [7], amongst others, allow for optical devices to be easily available. This, in turn, enables to extend the domain of optics from optical interconnects, the main application today (see e.g. [8]), to full-scale optical computations.

Motivated by this, initial work on the logic design of optical circuits for important Boolean functions has been conducted. This led to designs of (half) adders (see e.g. [9], [10]) or ternary logic operators (see e.g. [11]). While those approaches focused on single applications only, first attempts towards generic approaches have been considered in [12]. Here, different synthesis schemes aiming for the realization of an arbitrary function are discussed. However, the presented schemes mainly suffer from no or only poor support for expression sharing and, hence, rather large synthesis results as well as a restricted scalability (this is discussed in more detail later in Section II-B).

At the same time, the number of so called garbage outputs and particularly the splittings of optical signals constitute serious obstacles. In fact, a design scheme based on *Binary Decision Diagrams* (BDDs, [13]) and originally proposed in [12] was one of the the most promising approaches towards an efficient and scalable synthesis of optical circuits. Here, each node of the BDD is substituted by an optical gate eventually leading to a circuit realizing the desired function. But the generated circuits included a very large number of splitters, i.e. logical devices which divide an optical signal into two optical signals – each with only half the power. As a result, optical signals are split into hardly measurable fractions and, hence, this direction was eventually considered infeasible [12]. As a consequence, no scalable synthesis scheme for optical circuits exists until today.

In this work, we address this issue. We show that, despite the previous setbacks, the exploitation of BDDs still is a suitable direction. But in contrast to the previously proposed scheme, we introduce a new approach which generates the desired circuits in a reverse fashion, compared to the previously proposed scheme. For this purpose, paths of the BDD are explicitly considered. The proposed scheme eventually results in circuit structures where so called combiners rather than splitters are applied. This enables, for the first time, a scalable synthesis scheme which overcomes the obstacles of previously proposed solutions.

A conceptional discussion as well as an experimental evaluation confirm the advantages of the proposed solution: At the expense of a moderate increase in the number of optical gates, an optical circuit without any splitters is generated. At the same time, even improvements with respect to further cost criteria such as garbage outputs can be observed. Overall, this eventually establishes BDD-based synthesis as a suitable synthesis scheme for optical circuits without the serious obstacles observed in previous work.

The remainder of this paper is structured as follows: Section II provides an overview on the applied models and cost metrics used in the logic design of optical circuits as well as a brief review on previously proposed synthesis solutions. Afterwards, BDD-based synthesis as introduced in [12] including its major obstacles are discussed and the proposed solution is sketched in Section III. Details on the implementation of our solution are then provided in Section IV. Finally, the proposed approach is discussed from a conceptional point of view as well as experimentally evaluated in Section V and Section VI, respectively, before the paper is concluded in Section VII.

II. Design and Synthesis of Optical Circuits

Recent advances in the physical realization of optical devices motivated the development of corresponding, automated methods for their design and synthesis. For this purpose, corresponding logical models and optimization criteria have been derived. This section briefly reviews the resulting gate library and cost metrics. Afterwards, previously proposed synthesis schemes for optical circuits are discussed.

A. Applied Circuit Model and Cost Metrics

Optical devices allow to realize Boolean functionality by means of so called *crossbar gates* which route the optical signals between two parallel paths. The inputs of both paths can either be sourced by light (representing the logical value "1") or darkness (representing the logical value "0"). Furthermore, the routing of both paths may be switched depending on the value of an electrical signal. The output of each optical signal can be read using optical receivers. Logically, this leads to the following definition of a crossbar gate.

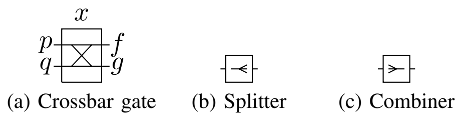


Fig. 1: Optical gates

Definition 1. *A* crossbar gate *realizes a Boolean function* $\mathbb{B}^3 \rightarrow \mathbb{B}^2$ *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 which, depending on the value of* x*, realize either the identity or a switch of the input values, i.e.*

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

is realized. Fig. 1a provides the graphical representation of a crossbar gate.

Note that, except for the crossbar gate as defined above, an optical signal cannot directly modify the value of an electrical signal and vice versa. For such purposes, an opto-electrical interface would be required which, however, is considered as 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.

Besides that, also *splitter* and *combiner* are utilized in order to realize logic functions.

Definition 2. *A* splitter *divides an optical signal into two optical signals – each with only half the signal power. A* combiner *merges two optical signals into a single one and, by this, inherently realizes the OR-function. Fig. 1b and Fig. 1c provide the graphical representation of both gates. Splitters (combiners) 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.*

Having a gate library composed of crossbar gates, splitters, and combiners, arbitrary Boolean functions can be realized.

Example 1. *Fig. 2 shows an optical circuit composed of six crossbar gates, two splitters, and one combiner.*

In order to measure the costs of an optical circuit, usually the *number of crossbar gates* is counted. This metric is simply motivated by the fact that each gate needs to be physically realized. Besides that, also the *number of splitters* has explicitly been considered. Although splitters (as well as combiners) are easy to realize physically, each splitter decreases the strength of an optical signal considerably. Hence, keeping the splitting of optical signals as small as possible is an important objective.

However, simply counting the number of splitters does not accurately measure the impact of splitters in a circuit. For the physical implementation, it will make a difference if e.g. ten signals are separately split once, or one signal is split ten times. While this has also been acknowledged in previous work [9], it was not explicitly addressed in terms of a metric thus far. In our work, we explicitly consider this effect by additionally introducing the *worst case fraction* of a single optical signal as a further metric. The worst case fraction is determined as follows: If we have a splitter with i outputs, each output is assumed to have a signal strength of $1/i$ of the incoming signal. A combiner will not change the fraction, as the worst case is assumed, i.e. all other inputs of the combiner are assumed to be 0. While this 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.

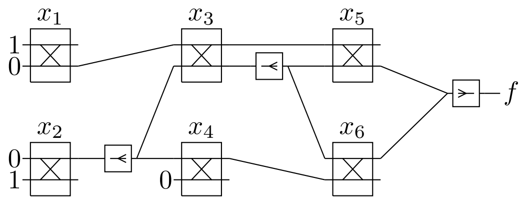


Fig. 2: Optical circuit

Finally, the number of *garbage outputs* – defined by the number of crossbar outputs, which are neither needed as input for another gate nor as primary output of the entire circuit anymore – has been considered in the past. Keeping the number of garbage outputs as small as possible is motivated by the fact that the respective optical signals still have to be disposed. This requires additional routing and, hence, overhead that should be avoided.

Example 2. *The costs of the circuit depicted in Fig. 2 are defined by 6 crossbar gates, 2 splitters, a worst case fraction of 4, and 5 garbage outputs.*

B. Previously Proposed Synthesis Approaches

As stated in Section I, logic design of optical circuits is an emerging area in which the development of corresponding automatic methods for synthesis is at the beginning. Nevertheless, first attempts have been made to automatically realize circuits based on the model and considering (some of) the cost metrics reviewed above.

In particular, several such methods for synthesis have been introduced and discussed in [12]. The first one employs so called *virtual gates*. Virtual gates are sub-circuits composed of crossbar gates that realize important building block-functions such as AND, OR, etc. They can be combined in a hierarchical fashion while, at the same time, maintain certain properties, e.g. the absence of garbage outputs. In contrast, virtual gates allow expression sharing only to a very limited extent and, hence, the resulting circuits are either of significant size or rely on a very large amount of splitters. To address this problem, an algorithm performing XOR decomposition is additionally employed. Here, parts of the function are synthesized as a virtual gate while sharing is explicitly employed whenever XORs expressions occur. For this purpose, function decomposition aiming for XOR expressions is performed. All these solutions are, however, bounded by scalability issues.

The same problem occurs for the solution proposed in [14], which introduces an exact synthesis scheme, i.e. an approach guaranteeing minimality with respect to the number of gates. Since guaranteeing minimality is a computationally hard problem, solvers for *Quantified Boolean Formulas* are utilized for this purpose. This allows for the synthesis of *minimal* optical circuits for Boolean functions of up to 6 variables. While this is ideal to confirm the minimality of virtual gates or to determine further building blocks, there obviously is a strong limitation with respect to scalability, too.

Thus far, the most promising approach towards scalable synthesis of optical circuits constitutes the BDD-based synthesis scheme proposed in [12]. Similar to corresponding schemes e.g. for the synthesis of conventional as well as reversible circuits in [15] and [16], respectively, BDDs representing the function to be synthesized are generated first. Afterwards, a mapping scheme is employed substituting each BDD node by a corresponding crossbar gate. This allows for a scalable synthesis but, eventually, leads to rather poor results. In fact, the generated circuits suffer from a very high decrease in signal strength – as mentioned above a crucial issue when it comes to the physical realization. Also, the number of garbage outputs was not satisfactory. Hence, this idea eventually was dropped by the authors of [12].

In this work, we propose a solution which overcomes and explicitly addresses these drawbacks. We will show that BDD-based synthesis indeed is a suitable scheme for synthesis of optical circuits – it only has to be applied in an entirely different fashion.

III. Motivation and General Idea

BDD-based synthesis as introduced in [12] represents a promising direction for scalable synthesis of optical circuits, but suffers from the very large number of splitters and an unsatisfactory number of garbage outputs. In this section, we describe the previously suggested (but eventually discarded) synthesis scheme in more detail and explicitly discuss its essential weakness. Afterwards, we present the general idea of our synthesis scheme which overcomes this weakness.

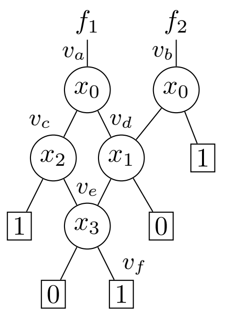
A. BDD-based Synthesis of Optical Circuits

BDD-based synthesis of optical circuits (originally proposed in [12]) relies on *Binary Decision Diagrams* as an efficient and scalable representation of Boolean functions. 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 two succeeding nodes $\mathrm{low}(v)$ and $\mathrm{high}(v)$. 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 the *co-factors* $f_{x_i=0}$ ($\mathrm{low}(v)$) and $f_{x_i=1}$ ($\mathrm{high}(v)$). Thus, the Shannon decomposition is naturally realized. 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*, i.e. nodes with more than one predecessor.

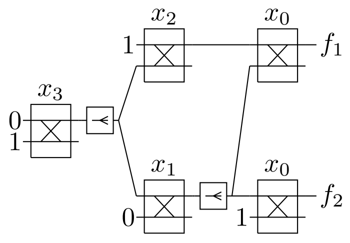
Having a BDD of the function to be realized, a corresponding optical circuit can easily been derived from it. In fact, the Shannon decomposition applied in each BDD node is directly realized by a crossbar gate as introduced in Def. 1. Hence, an optical circuit can be derived by traversing the BDD in a depth-first fashion and substituting each node $v \in V$ with a corresponding crossbar gate. If a shared node occurs, the respective optical signal has to be split accordingly. Combining all gates and connecting the respective signals eventually lead to an optical circuit realizing the desired function.

Example 3. *Fig. 3a shows a BDD representing the function $f : \mathbb{B}^4 \rightarrow \mathbb{B}^2$ with $f_1 = \bar{x}_0\bar{x}_2 + \bar{x}_0 x_2 x_3 + x_0 \bar{x}_1 x_3$ and $f_2 = x_0 + \bar{x}_0 \bar{x}_1 x_3$ as well as the respective co-factors resulting from the application of the Shannon decomposition. Fig. 3b shows the optical circuit obtained from this BDD. The co-factor represented by the node v_e can easily be realized by the left-most crossbar gate. Since v_e is a shared node, a splitter is necessary in order to realize the co-factors represented by v_c and v_d. More precisely, the splitter provides the respective input values which can be used by the suceeding gates in order to realize the co-factors of nodes v_c and v_d. Following this scheme eventually leads to the circuit shown in Fig. 3b.*

Considering practically relevant functions, the respective BDD representation usually includes a large amount of shared nodes which eventually result in splitters. Hence, applying the synthesis method sketched above leads to optical circuits where certain output signals are constituted by a negligible fraction of power only (this is evaluated later in Section VI in detail). Because of this essential weakness (and the fact that also the number of garbage outputs is rather large), BDD-based synthesis has been dropped as useful synthesis scheme.



(a) BDD



(b) Obtained circuit

Fig. 3: BDD-based synthesis (previous approach)

B. Proposed Idea

Dealing with the problem of realizing complex Boolean functionality with no or at least as few as possible splitters constitutes one of the biggest obstacles in the design of optical circuits. In the past, researchers tried to come up with alternative decomposition schemes such as the Ashenhurst-Curtis decomposition or functional bi-decomposition [14]. But all these directions eventually resulted in new challenges to be solved. Instead, this work shows that BDD-based synthesis still is a suitable scheme – if applied in an entirely different fashion.

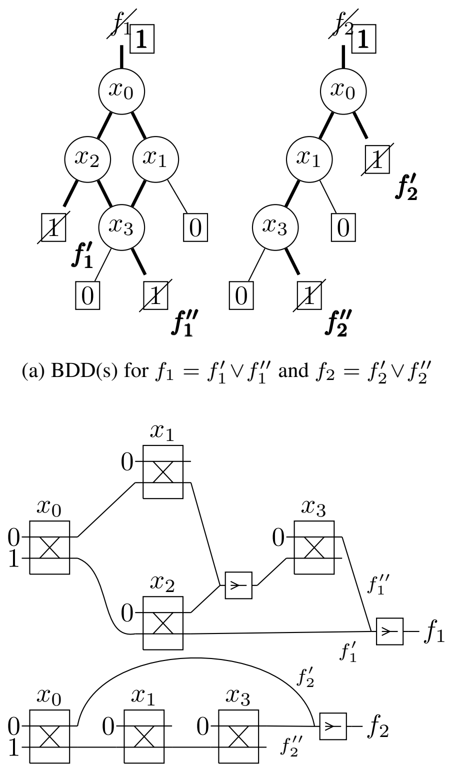
More precisely, we propose an alternative BDD-based synthesis scheme which exploits the fact that a function f to be realized is defined by the 1-paths of the corresponding BDD. A *1-path* $p = (v_1, v_2, \ldots, v_l)$ with $v_i \in V$ for a given BDD $G = (V, E)$ representing f is a sequence of nodes obtained by traversing the BDD from the root node to a $\boxed{1}$-terminal. A single path can be realized, as described in Section III-A, i.e. by simply traversing each node of the path and adding a corresponding crossbar gate. Based on that, a given function f can be realized by separately synthesizing all 1-paths of the corresponding BDD and eventually ORing the respective outputs[1].

However, generating optical logic for all 1-paths separately obviously would lead to a very expensive circuit realization. Hence, another property is exploited: the direction in which a 1-path is traversed during synthesis does not matter, i.e. the path can be traversed from the root node to the $\boxed{1}$-terminals or vice versa. In contrast to straight-forward BDD schemes, we propose to traverse the 1-paths in a *reverse* fashion, i.e. from the root nodes to the $\boxed{1}$-terminals. By this, shared nodes of the BDD can fully be exploited, i.e. no redundant logic has to be created, while, at the same time, combiners instead of splitters can be applied for their realization. This reduces the number of cases an optical signal is split to zero and, hence, overcomes the weakness of BDD-based synthesis reviewed above.

The following example illustrates the idea.

Example 4. *Consider again the function f_1 as discussed in Example 3 which is solely represented by the BDD shown at the left-hand side of Fig. 4a. The edges highlighted bold indicate that this function is defined by three 1-paths. Considering these paths in the direction from the root to the terminals and additionally exploiting the sharing within the BDD would eventually lead to (1) a root edge assumed to be constant 1 and (2) two functions f'_1 and f''_1 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 Fig. 4b.*

[1]Note that ORing optical signals can be realized by a single combiner, i.e. the output of the combiner emits light (i.e. the logical value 1) if at least one of its optical inputs sources light.



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

(b) Obtained circuit(s)

Fig. 4: Proposed BDD-based synthesis

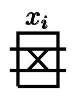
The general idea of the splitter-free BDD-based synthesis can only be applied to BDDs representing functions $f : \mathbb{B}^n \to \mathbb{B}^1$, i.e. functions with one output only. In order to realize multi-output functions such as considered in Example 3, all respective outputs have to be treated separately. The right-hand side of Fig. 4a and the bottom of Fig. 4b show the BDD and the resulting circuit for the function f_2 considered in Example 3, respectively. This restriction of course prevents the full exploitation of sharing within the BDD. However, discussions and experimental results show that this only leads to a minor drawback with respect to the costs of the resulting circuits. Before this is addressed in detail in Section V and Section VI, a detailed description of the implementation of this scheme is provided in the next section.

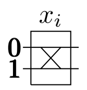
IV. Implementation

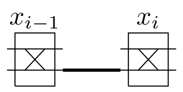
The observations and the general idea sketched in the previous section eventually resulted in an implementation of the proposed synthesis scheme as shown in Algorithm 1. The algorithm starts by creating the BDD for the function f (line 1). This can efficiently be done using existing BDD packages such as *CUDD* [17]. In this BDD representation, we are particularly interested in all 1-paths which are stored in the set P (line 2) and, afterwards, individually processed (line 5).

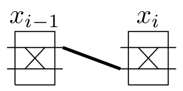
As discussed in Section III-B, single paths can be realized by simply traversing each node of the path and adding a crossbar gate accordingly. Hence, for each 1-path $p = (v_0, v_1, \ldots, v_l)$, each single node v_i ($1 \leq i \leq l$) is considered (line 7). We additionally consider to which sub-function f_j of f with $f : \mathbb{B}^n \to \mathbb{B}^m$ and $1 \leq j \leq m$ this path belongs (line 6): If the respective node has not been considered before in combination with function f_j (line 13),

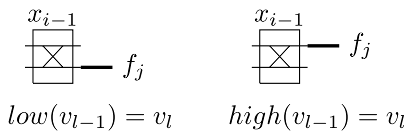
Algorithm 1: Overall reverse BDD synthesis algorithm

```
   Data: Function $f : \mathbb{B}^n \to \mathbb{B}^m$
   Result: Splitter free optical circuit realizing $f$
1  $G = (V, E) \leftarrow$ BDD representation of $f$
2  $P \leftarrow$ all 1-paths in $G$
3  $visited \leftarrow \emptyset$
4
5  foreach $p = (v_1, v_2, \ldots, v_l) \in P$ do
6      $f_j \leftarrow$ function belonging to the path $p$
7      for $i = 1$ to $l$ do
8          $x_i \leftarrow label(v_i)$
9          $x_{i-1} \leftarrow label(v_{i-1})$
10
11         // new entities in each step are highlighted in bold
13         if $(v_i, f_j) \notin visited \ \wedge \ i \neq l$ then
14
```


```
15             $visited \leftarrow visited \cup \{(v_i, f_j)\}$
16         else
17             reuse existing gate
18         switch do
19             case $i = 1$
20
```


```
21             case $v_i = low(v_{i-1})$
22
```


```
23             case $v_i = high(v_{i-1})$
24
```


```
25             case $i = l$
26
```


a corresponding crossbar gate is added to the circuit structure (line 14). Otherwise, the crossbar gate from the previous traversion can be re-used (line 17). The information whether a node has already been considered before in combination with f_j is stored in the set $visited$ which is initially set empty (line 3) and constantly updated (line 15). This case differentiation is illustrated by the following example.

Example 5. *Consider again the BDD shown in Fig. 3a. The node v_e is traversed by three 1-paths, namely*

- $p_1 = (v_a, v_c, v_e, v_f)$ *(belonging to f_1),*
- $p_2 = (v_a, v_d, v_e, v_f)$ *(belonging to f_1), and*
- $p_3 = (v_b, v_d, v_e, v_f)$ *(belonging to f_2).*

First, p_1 is traversed which, next to others, leads to the creation of a crossbar gate for v_e (see rightmost crossbar gate in the top circuit shown in Fig. 4b). The same gate can *be re-used when traversing p_2 since it belongs to the same sub-function f_1. However, as p_3 belongs to a different sub-function (namely f_2), a re-use is not possible. Instead, a new gate is created (see the rightmost crossbar gate in the bottom circuit shown in Fig. 4b).*

By following this scheme, the independent consideration of each function f_j as discussed in Fig. 4 is done implicitly.

Finally, the connections between the previously created gates have to be generated. Again, a case-by-case analysis is applied here (lines 18-25). Since we consider the BDD in a reverse fashion (as discussed in Section III-B and illustrated in Fig. 4a), the first node v_1 of each path is realized as shown in line 19, i.e. a simple identity function (at output f) and a negation function (at output g) with respect to x_1 (the label of v_1) is realized. Those two functions are eventually re-used by the next node v_2 in the path and, hence, the next crossbar gate. If v_2 is reached through a low-edge (i.e. if $v_2 = low(v_1)$), then the negation output is used, i.e. a connection as shown in line 22 is created. Otherwise, the identity output is applied, i.e. a connection as shown in line 24 is created. This eventually leads to co-factors at the output f and the output g. Which one is chosen here again depends on the next node v_3. This procedure continues until all nodes of the currently considered path are traversed. If a crossbar gate's input p or q is used multiple times, we implicitly create a combiner gate for those inputs.

The continuation of this procedure terminates if the last node v_l of the path p is considered. For this node, no crossbar gate is created but the output of the previous gate is used as a primary output for the function f_j. The choice of the output (f or g) is, again, determined by whether v_l was reached through a low-edge or a high-edge (see line 26).

Example 6. *Consider again the paths of the BDD shown in Fig. 3a. Applying the scheme described above exactly leads to the connections as shown in Fig. 4b.*

V. Discussion

As reviewed in Section II-A, the number of crossbar gates, the worst case fraction, and the number of garbage outputs are the most important optimization criteria in the logic design of optical circuits. In particular, the splitting of optical signals constituted a crucial obstacle for scalable synthesis thus far. The synthesis scheme proposed in this work aims to address this obstacle, thereby overcoming one of the main weaknesses of previously proposed solutions. Whether this has been accomplished and, if yes, at what costs is discussed in this section from a conceptional point of view[2]. For this purpose, arbitrary functions $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ composed of m single output functions denoted as f_j $(0 \leq j < m)$ are considered. The respective BDDs are denoted by $G = (V, E)$ and $G_j = (V_j, E_J)$ for f and f_j, respectively.

First, it can be observed that the proposed BDD-based synthesis scheme has an obvious drawback: Each function f_j of f has to be considered separately. Because of this, sharing between those functions cannot be exploited. Since additional BDD nodes directly translate to additional crossbar gates, this eventually leads to an overhead of crossbar gates which can be measured by

$$\frac{\sum_{0 \leq j < m} |V_i|}{|V|},$$

i.e. the combined number of all nodes of single-output BDDs G_j divided by the number of nodes of the BDD G representing all functions. That is, this overhead is clearly bounded by the number m of f_j-functions, meaning that no more than $m \cdot |V|$ gates are needed. Moreover, usually less overhead can be observed. For example, the number of nodes in the BDD G discussed in Fig. 3 is five, while the combined number of nodes in all BDDs G_j in Fig. 4 is seven. Hence, this results in an overhead of a factor of 1.4 only, instead of the maximal overhead of 2. Experiments summarized in Section VI confirm that this similarly holds also for larger BDDs.

On the other hand, this small overhead allows for addressing the main weaknesses of the previously proposed BDD-based synthesis. In fact, the number of garbage outputs is significantly reduced in the majority of the cases. BDD-based synthesis as proposed in [12] resulted in a garbage output for each BDD node (due to the fact that in each corresponding crossbar gate, only the f-output is used). In the proposed scheme, only BDD nodes leading to a $\boxed{0}$-terminal cause a crossbar gate with a garbage output. Hence, the newly proposed scheme generates as much as $\sum_{j=1}^{m} |\{v | v \in V_j \wedge \ (low(v) = 0 \vee high(v) = 0\}|$ garbage outputs – usually, this number is smaller than the total of $|V|$ garbage outputs generated by the previously proposed BDD-based synthesis.

Finally, and most important, the proposed synthesis scheme does not require any splitters. That is, not a single optical signal is split and, hence, the worst case fraction remains 1 for all signals in the circuit. Overall, this eventually establishes BDD-based synthesis as a suitable synthesis scheme for optical circuits without the serious obstacles observed in previous work and only at the expense of a moderate increase in the number of crossbar gates.

VI. Experimental Evaluation

In order to experimentally confirm the discussions from the previous section, BDD-based synthesis as described in Section IV has been implemented in Java and applied to a selection of benchmarks taken from the LGSynth-package. All experiments have been conducted on a 2.6 GHz Intel Core i5 machine with 8 GB of memory running Linux.

Table I summarizes the obtained results. The first columns provide details on the considered functions and their BDD representation, i.e. their name, number of primary inputs (n), number of primary outputs (m), the number of nodes of the entire BDD ($|V|$), as well as the combined number of nodes over all BDDs ($\sum |V_i|$). Afterwards, the results obtained by the previous synthesis approach from [12] and the newly proposed scheme from Section IV are reported, i.e. the number of crossbar gates as well as the number of garbage outputs. For the synthesis scheme from [12], we additionally listed the number of splitters and the worst case fraction (the proposed scheme always leads to a total of zero splitters and, hence, a worst case fraction of 1). For the proposed scheme, we additionally list the number of combiners (the previously proposed scheme always leads to a total of zero combiners). Finally, the last two columns list the differences (as factors) in the synthesis results with respect to the number of crossbar gates and garbage outputs. All results have been obtained in negligible run-time, i.e. just a few CPU seconds.

The results confirm that the actual overhead caused by the newly proposed synthesis scheme in terms of crossbar gates indeed is moderate. In the worst case (*table3*), a little bit more than twice the number of crossbar gates is needed – in the majority of benchmarks, the additional overhead is even smaller.

In contrast, this moderate overhead enables significant improvements with respect to other metrics. In fact, the number of garbage outputs can often be reduced to a fraction of

[2]Precise results based on experimental evaluations are provided later in Section VI.

TABLE I: Experimental Evaluation

					Previously proposed synthesis scheme [12]				Proposed synthesis scheme			Difference	
Function	n	m	$\|V\|$	$\sum \|V_i\|$	Crossbar	Splitter	w.c. fract.	Garb.	Crossbar	Comb.	Garb.	Crossbar	Garb.
apex2	39	3	7102	7182	7102	1756	$> 10^7$	7102	7182	6176	1235	1.01	0.17
frg1	28	3	203	203	203	26	720	203	203	189	98	1.00	0.48
cps	24	109	2318	3606	2318	280	184320	2318	3606	1279	2590	1.56	1.12
cordic	23	2	80	82	80	29	10240	80	82	64	20	1.03	0.25
mux	21	1	131070	131070	131070	255	$> 10^7$	131070	131070	256	550	1.00	0.00
cm151a	19	9	93	94	93	14	16	93	94	67	50	1.01	0.54
parity	16	1	31	31	31	28	16384	31	31	29	2	1.00	0.06
spla	16	46	681	1090	681	114	6480	681	1090	316	767	1.60	1.13
cm163a	16	13	50	70	50	8	60	50	70	23	38	1.40	0.76
pcler8	16	5	58	59	58	16	4	58	59	29	17	1.02	0.29
pdc	16	40	705	1118	705	124	18900	705	1118	413	765	1.59	1.09
t481	16	1	32	32	32	11	1296	32	32	29	6	1.00	0.19
cmb	16	4	47	48	47	1	2	47	48	2	26	1.02	0.55
ryy6	16	1	23	23	23	7	96	23	23	20	9	1.00	0.39
ham15	15	15	119	183	119	72	1024	119	183	87	75	1.54	0.63
in0	15	11	526	625	526	143	99792	526	625	307	137	1.19	0.26
f51m	14	8	1219	1605	1219	407	244992	1219	1605	683	98	1.32	0.08
alu4	14	8	1352	1534	1352	339	$> 10^7$	1352	1534	438	125	1.13	0.09
misex3c	14	14	847	970	847	202	367200	847	970	556	222	1.15	0.26
misex3	14	14	1301	1976	1301	349	$> 10^7$	1301	1976	1320	672	1.52	0.52
tial	14	8	1354	1677	1354	397	$> 10^7$	1354	1677	726	184	1.24	0.14
cu	14	11	65	97	65	10	10	65	97	12	82	1.49	1.26
table3	14	14	941	1996	941	212	$> 10^7$	941	1996	618	969	2.12	1.03
co14	14	1	27	27	27	12	4096	27	27	13	14	1.00	0.52

Function: Name of the function n: Number of primary inputs m: Number of primary outputs
$|V|$: Number of nodes of the entire BDD $\sum |V_i|$: Combined number of nodes over all BDDs G_i
Crossbar: Number of crossbar gates Splitter: Number of Splitter gates w.c. fract.: Worst case fraction
Garb.: Number of garbage outputs Comb.: Number of combiner gates
The proposed synthesis scheme always leads to circuits with a total of none splitters and, hence, a worst case fraction of 1.
All results have been obtained in negligible run-time, i.e. just a few CPU seconds.

what was needed by previous solutions. Exceptions are *cps*, *spla*, *pdc*, and *table3* due to their special BDD structure. But even here, no serious increase can be documented. Even more important, the main weakness of BDD-based synthesis has been solved: No splitter and, hence, no signal splitting is required anymore. As the numbers clearly show, this was the main obstacle of previous solutions. Signals have been split to hardly measurable fractions. All this does not happen anymore in the newly proposed synthesis scheme, i.e. all signals keep their initial signal strength.

VII. Conclusions

In this paper, a BDD-based synthesis scheme for splitter-free optical circuits has been proposed. By this, serious obstacles of previously introduced solutions are addressed. Experiments showed that our approach completely solves the issue of signal spliting while, at the same time, still provides a scalable synthesis scheme. In many cases, also the number of garbage outputs can signficantly be reduced. All these benefits come at the expense of a moderate overhead in terms of crossbar gates only. This eventually establishes BDD-based synthesis as a suitable and scalable synthesis scheme for optical circuits without the serious obstacles observed in previous work. In future work, we intend to apply one-path minimization techniques to further improve the results.

References

[1] G.E. Moore. Cramming more components onto integrated circuits. *Journal of Electronics*, 38(8):183–191, 1965.

[2] 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. Silicon CMOS devices beyond scaling. *IBM Journal of Research and Development*, 50(4.5):339–361, 2006.

[3] P.K. Kaliraj, P. Sieber, A. Ganguly, I. Datta, and D. Datta. Performance Evaluation of Reliability Aware Photonic Network-on-Chip Architectures. In *Intl. Green Computing Conference*, pages 1–6, 2012.

[4] A. Shacham, K. Bergman, and L. P. Carloni. Photonic Network-on-Chip for Future Generations of Chip Multi-Processors. *IEEE Transactions on Computers*, 57(9):1246–1260, 2008.

[5] William M Green, Michael J Rooks, Lidija Sekaric, and Yurii A Vlasov. Ultra-compact, low RF power, 10 Gb/s silicon Mach-Zehnder modulator. *Optics express*, 15(25):17106–17113, 2007.

[6] Ling Liao, Dean Samara-Rubio, Michael Morse, Ansheng Liu, Dexter Hodge, Doron Rubin, Ulrich Keil, and Thorkild Franck. High speed silicon Mach-Zehnder Modulator. *Optics Express*, 13(8):3129–3135, 2005.

[7] Haisheng Rong, Richard Jones, Ansheng Liu, Oded Cohen, Dani Hak, Alexander Fang, and Mario Paniccia. A continuous-wave Raman silicon laser. *Nature*, 433(7027):725–728, 2005.

[8] Ray Beausoleil, J Ahn, N Binkert, Al Davis, David Fattal, Marco Fiorentino, Norman P Jouppi, Moray McLaren, CM Santori, Robert S Schreiber, et al. A Nanophotonic Interconnect for High-Performance Many-Core Computation. In *IEEE Symposium on High Performance Interconnects, 2008.*, pages 192–189. IEEE, 2008.

[9] Abdallah K. Cherri and Ayman S. Al-Zayed. Circuit designs of ultra-fast all-optical modified signed-digit adders using semiconductor optical amplifier and Mach-Zehnder interferometer. *Optik - International Journal for Light and Electron Optics*, 121(17):1577 – 1585, 2010.

[10] D. K. Gayen and T. Chattopadhyay. Designing of Optimized All-Optical Half Adder Circuit using Single Quantum-Dot Semiconductor Optical Amplifier Assisted Mach-Zehnder Interferometer. *Journal of Lightwave Technology*, 31(12):2029–2035, 2013.

[11] T. Chattopadhyay. All-optical symmetric ternary logic gate. *Optics and Laser Technology*, 42(6):1014–1021, 2010.

[12] Christopher Condrat, Priyank Kalla, and Steve Blair. Logic Synthesis for Integrated Optics. In *Great lakes symposium on VLSI*, pages 13–18. ACM, 2011.

[13] Randal E Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 100(8):677–691, 1986.

[14] Christopher Condrat, Priyank Kalla, and Steve Blair. Exploring Design and Synthesis for Optical Digital Logic. In *International Workshop on Logic Synthesis*, 2010.

[15] R. Drechsler, J. Shi, and G. Fey. MuTaTe: An efficient design for testability technique for multiplexor based circuits. In *Great lakes symposium on VLSI*, pages 80–83, 2003.

[16] R. Wille and R. Drechsler. BDD-based synthesis of reversible logic for large functions. In *Design Automation Conference*, pages 270–275, 2009.

[17] Fabio Somenzi. *CUDD: Colorado University Decision Diagram package*. University of Colorado at Boulder, http://vlsi.colorado.edu/~fabio/CUDD/, 2012.