Studies on Synthesis Methods for Efficient Optical Logic Circuits

Ryosuke Matsuo

Abstract

Optical circuits using nanophotonic devices attract significant interest due to their ultra-high-speed operation. As a consequence, the synthesis methods for optical circuits also attract increasing attention. In order to implement general and large-scale logic functions, automated design methods are proposed based on various schemes. Especially, a binary decision diagram(BDD)-based method has attracted significant interest due to the ultra-high speed and area-efficient characteristics compared with the other synthesis methods. However, the strategy of simply mapping a BDD to an optical circuit sometimes results in an explosion of size and involves significant power consumption due to branches of the waveguide and optical logic gates. In this thesis, we propose some methods to address these issues, and experimental results demonstrate that our methods are effective.

First, we propose a method for reducing the size of BDD-based optical logic circuits exploiting wavelength division multiplexing (WDM). This thesis also proposes a method for reducing the number of waveguide branches, which reduces the power dissipation in laser sources.

Next, we consider the method for reducing power consumption by exploiting a property of a BDD. We demonstrate that power consumption largely depends on the variable order of a BDD. Unfortunately, an optimization problem of finding the variable order to minimize the power consumption has large time complexity. Therefore, we propose an algorithm that utilizes an efficient reordering method based on an adjacent variable swap to reduce the execution time.

We then turn to a method for reducing power consumption by exploiting an Optical-to-Electrical(OE) converter. Our method divides the target logic function into multiple sub-functions. An optical logic circuit implements each sub-function. The optical logic circuits are connected with OE converters. This method can exponentially reduce power consumption. Although

OE converters have an issue of large computational delay, the proposed synthesis method mitigates the OE converter delay overhead by parallelizing sub-circuits.

We finally propose a cross-bar gate logic (CBGL) as a new scheme for optical logic circuits without a signal power loss at a waveguide branch. This thesis enumerates CBGL with the minimum number of gates for all three-input functions by an exhaustive search. The enumeration algorithm incorporates a technique to efficiently prune the vast search space to reduce the execution time.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Shin-ichi Minato. He always gives me various insightful comments and suggestions, which are essential to improve my research. I also would like to express special thanks to Hidetoshi Onodera, who is my former supervisor.

I would like to sincerely thank my coauthors: Tohru Ishihara, Jun Shiomi, Akihiko Shinya and Masaya Notomi. My appreciation also goes to current and former members of our laboratory. Especially, I am grateful to Jun Kawahara, Yuni Iwamasa and Yu nakahata. I also thank the secretaries of our laboratory, Seiko Jinno and Sachie Oomae.

I appreciate the financial support by JSPS KAKENHI Grant Number JP21J12765 and JST-CREST JPMJCR18K3.

Finally, I would like to thank my family and friends for their kind support.

List of Publications

Publications Included in this Thesis

This thesis includes contents of the following six publications.

Refereed Journal Articles

1. Ryosuke Matsuo, Jun Shiomi, Tohru Ishihara, Hidetoshi Onodera, Akihiko Shinya, Masaya Notomi.
   Methods for Reducing Power and Area of BDD-Based Optical Logic Circuits.
   IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E102-A, 12, pp. 1751–1759, 2019.

2. Ryosuke Matsuo, Jun Shiomi, Tohru Ishihara, Hidetoshi Onodera, Akihiko Shinya, Masaya Notomi.
   A Synthesis Method Based on Multi-Stage Optimization for Power-Efficient Integrated Optical Logic Circuits.
   IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E104-A, 11, pp. 1546–1554, 2021.

Refereed Conference Proceedings

1. Ryosuke Matsuo, Jun Shiomi, Tohru Ishihara, Hidetoshi Onodera, Akihiko Shinya, Masaya Notomi.
   BDD-based Synthesis of Optical Logic Circuits Exploiting Wavelength Division Multiplexing.
   Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 203–209, 2019.

2. Ryosuke Matsuo, Jun Shiomi, Tohru Ishihara, Hidetoshi Onodera, Akihiko Shinya, Masaya Notomi.
   A Synthesis Method for Power-Efficient Integrated Optical Logic Circuits Towards Light Speed Processing.
   IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 488–493, 2020.

3. Ryosuke Matsuo, Shin-ichi Minato.
   BDD Variable Ordering for Minimizing Power Consumption of Optical Logic Circuits.
   IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 96–101, 2021.

4. Ryosuke Matsuo, Shin-ichi Minato.
   Space and Power Reduction in BDD-based Optical Logic Circuits Exploiting Dual Ports.
   Design, Automation and Test in Europe Conference (DATE), pp. 1077–1082, 2022.

Contents

Chapter 1

Introduction

1.1 Background

Computing on phothonic integrated circuit (PIC) has attracted significant interest as a promising alternative to traditional CMOS electronics as Moore's law dreaks down. Although the LSI technologies have made remarkable progress since their appearance, the effective resistivities of local level wires increase rapidly due to size effects at ultra-scaled dimensions [1]. Post-layout analysis using predictive technology models [2] shows that interconnect performance degradation may dominate over the device speed improvement in a 22 nm technology node and below. As a result, the reduction of the total delay per gate saturates around 10 ps [1]. Optical communication technologies also have been rapidly growing over the past decades. With recent advances in nanophotonics, the optical communication technologies have gradually migrated into ever-shorter distances and moved onto silicon chips as on-chip optical interconnects [3]. More recently, integrated optical circuits using nanophotonic devices attract significant interest due to its ultra-high-speed nature. The delay of an optical gate based on a nanophotonic directional coupler is on the order of a few hundreds of femtosecond [4], which is more than 10 times faster than that of the CMOS logic gates. Therefore, PICs are expected to realize ultra-high-speed computing. Previous research on optical computing by PICs explored two computing paradigms: digital and analog computing. Digital computing performs boolean logic. In this thesis, we call a PIC implementing boolean logic as an optical logic circuit. Analog computing, on the other hand, interprets optical signals as continuous values in the real or complex domain and uses linear optics to perform analog-style computations. In the field of digital computing, research efforts have been made on basic bitwise operations such as (N)AND, (N)OR, and X(N)OR gates [5–9], logic circuits such as adders [10–17], comparators [10], multiplexers [18], multipliers [19–22], subtractors [23–25], decoders [26, 27], and encoders [28, 29]. In order to implement general and larger-scale logic fucntions, logic synthesis methods are proposed based on various schemes: from virtual gates [30], sum of products form [31, 32], exclusive sum of products form [32], and-inverter graph (AIG) [33], or-inverter graph (OIG) [34], and binary decision diagram(BDD) [35–38]. In the field of analog computing, optical neural networks (ONNs) have attracted significant interest [39–41]. ONNs have advantages over CMOS electronics in both speed and power efficiency since they can directly utilize linear optics to perform neuromorphic manipulations. On the other hand, since digital optical computing has fundamental disadvantages in that optical devices lack the capability of logic-level restoration, the scalability of optical logic circuits is very limited. The signal power attenuations at an optical logic gate and a waveguide branch cause a significant power loss in the circuit for a large-scale logic function, which results in huge power consumption. Moreover, the size of optical devices is also a scalability issue. Although the size of optical devices has been drastically reduced in the last decades thanks to the advancement of nanophotonics technologies, it is still very large [4]. Therefore, the realization of an all-optical computer is expected to be difficult, and computers with both electrical devices and optical devices have been recently intensively investigated. However, studies on circuit designs for low-power and area-efficient optical logic circuits are still essential. Although logic synthesis is essential to implement various applications with optical logic circuits, researches on logic synthesis are not mature compared to studies on circuit designs for specific logic functions. Therefore, we study logic synthesis methods that enhance the scalability of optical logic circuits while maintaining the ultra-high-speed characteristic by exploiting the property of data structures such as BDDs and utilizing techniques regarding optimization algorithms. This thesis is focused on synthesis methods based on a BDD since BDD-based methods have attracted significant interest due to their ultra-high-speed and area-efficient characteristics compared with the other synthesis methods.

1.2 Related Works

Synthesis Methods for Optical Logic Circuits

Let us describe related works for synthesis methods for an optical logic circuit with cross-bar gates. This thesis uses a cross-bar gate as a logic gate in an optical circuit. In the following, an optical logic circuit indicates a logic circuit with cross-bar gates. A cross-bar gate has two optical inputs and two optical outputs, in which an electrical signal controls the optical routing from inputs to outputs. We describe a cross-bar gate in detail in chapter 2. Shamir and Caulfield investigated the use of cross-bar gates as Fredkin gates [42,43]. The Fredkin gate model assumes that an optical signal can also drive a control input, which is different from the cross-bar gate model we assume. Hardy demonstrated that a circuit based on non-Fredkin cross-bar gates (this gate model is the same as our assumption) could implement some simple logic functions [44]. Unfortunately, the proposed method can not be applied to large-scale complex logic functions. Therefore, to implement an arbitral logic function, Condrat proposed a synthesis method based on a virtual gate (VG) [30]. In this paper, VG denotes a circuit implementing two-input logic functions: AND, OR, XOR, and so on. Utilizing nested structures of VG can implement an arbitral logic function. Then, Deb proposed a synthesis method based on the sum of products (SOP) form and the exclusive sum of products (ESOP) form [32]. This method exploits VG for AND function to implement product terms. In chapter 2, we explain these methods in detail. Condrat mentioned a BDD-based method as well. He concluded that a BDD-based method has a fundamental disadvantage of large power consumption due to a large amount of power loss at a waveguide branch (a waveguide branch is called a splitter) compared to a VG-based method. In contrast with power consumption, BDD-based circuits have ultra-high speed and area-efficient characteristics compared to circuits based on the other methods. Therefore, Wille proposed a synthesis method for a splitter-less BDD-based optical logic circuit. This method results in a large number of combiners [35]. Since combiners also have a power loss, this methods can not improve the issue of power consumption. Zhao also proposed a method for reducing power loss by eliminating splitters [36]. However, this method can not drastically reduce power consumption and has the drawback of increasing the number of gates. BDD-based optical logic circuits also have issues regarding signal power attenuation at a gate. Zhao proposed a method for mitigating optical signal power attenuation with the help of signal power restoration by exploiting O-E-O repeaters [38]. Unfortunately, since O-E-O repeaters' operation speed is significantly larger than cross-bar gates, this method spoils the advantage of ultra-high-speed operation of optical logic circuits. Research efforts have been made on the size of BDD-based optical logic circuits. In optical logic circuits, the number of cross-bar gates represents the size. Zhao proposed a method for reducing the number of gates by exploiting wavelength division multiplexing [37].

Binary Decision Diagram

Let us introduce related works regarding a BDD since this thesis proposes synthesis methods for efficient optical logic circuits exploiting some techniques for a BDD. BDDs were proposed in [45, 46], and then reduced and ordered BDDs(ROBDDs) were proposed in [47]. ROBDDs have a canonical form and can be synthesized by each other. Since this property is important for various applications of BDDs, the term BDD generally indicates a ROBDD. BDDs are exploited in various research fields such as logic synthesis [48–50], model checking [51], and logic optimization [52]. Several techniques to efficiently implement BDDs have been proposed, for example, variable ordering [53–58],attributed edges [59], and shared-BDD [59]. These techniques are deeply related to the synthesis methods proposed in this thesis. ZDDs were proposed in [60] as a variant of BDDs. Since ZDDs are suitable for sparse set families compared with BDDs, ZDDs have been applied to wide research fields such as data mining and combinatorial optimization. Several variations of BDDs/ZDDs are proposed. Sequence BDDs [61] are suitable for sets of sequences. πDDs [62], rot-πDDs [63] and Group Decision Diagrams (GDDs) [64] are suitable for sets of permutations. Multi-valued decision diagrams (MDDs) [65] are suitable for multi-valued logic functions. Sentential decision diagrams(SDDs) [66] are generalizations of BDDs. Zero-suppressed SDDs (ZSDDs) are generalizations of ZDDs.

1.3 Our Contributions

In contrast with the advantage in speed, the fundamental disadvantages of the optical logic circuits against the CMOS LSI logic circuits are size and power consumption. To address these issues, we propose methods for reducing the power consumption and the size of optical logic circuits while maintaining the ultra-high-speed characteristic. Our contributions are summarized below. Chapters 3,4, and 5 are focused on a BDD-based synthesis method.

Capter 3 We propose a method for reducing the size of BDD-based optical logic circuits exploiting wavelength division multiplexing (WDM). Although WDM is known as an essential technology in the field of optical communication, there are few studies exploiting WDM in optical logic circuits. This chapter demonstrated that WDM has the potential to improve the efficiency of optical logic circuits. In addition, we propose a method for reducing the number of splitters in a BDD-based circuit, which reduces power consumption. Since power loss in splitters is a serious issue in a BDD-based optical logic circuits, methods for reducing power loss in splitters are proposed in [35, 36].

Capter 4 We consider the method for reducing power consumption by exploiting a property of a BDD. This chapter focuses on BDD variable ordering. Although research efforts have been made on variable ordering for a compact BDD, to the best of our knowledge, there is no research variable ordering for reducing the power consumption of BDD-based optical logic circuits. We demonstrate that power consumption largely depends on the variable order of a BDD. However, an optimization problem of finding the variable order to minimize the power consumption has significant time complexity. Therefore, we propose an algorithm that utilizes an efficient reordering method based on an adjacent variable swap to reduce the execution time.

Capter 5 We propose a method for eliminating a splitter in the circuit, focusing on nodes that satisfy certain conditions. This method is especially effective for the XOR function or logic function that includes XOR function as a sub-function. Although a BDD representing the XOR function is compact, a naive BDD-based optical logic circuit for the XOR function has significant power consumption due to many splitters. XOR function can be implemented by a compact optical logic circuit with low power consumption [44]. Our method can modify a BDD-based optical logic circuit for XOR function into a power-efficient circuit shown in [44]. Experimental results demonstrate that our method works effectively even for general logic functions.

Capter 6 We propose a method for reducing power consumption by exploiting an Optical-to-Electrical(OE) converter. Our method divides the target logic function into multiple sub-functions with OE converters. Each sub-function has a smaller number of inputs than that of the original function, which enables exponential reducing the power dissipated by an optical logic circuit representing the sub-function. Unfortunately, the OE converter has a low-speed operation compared with an optical logic gate. Therefore, the proposed synthesis method mitigates the OE converter delay overhead by parallelizing sub-functions.

Capter 7 We propose a circuit design without a garbage output, which is a signal power loss at a waveguide branch. We define this circuit design as a cross-bar gate logic (CBGL). Although a method for synthesizing a circuit without garbage outputs is proposed, this method can not minimize the number of gates. We find the circuit with the minimum number of gates for all three-input functions by an exhaustive search. Since the search space is vast, we introduce a technique to prune it efficiently.

Let us explain the dependencies of methods proposed in each chapter. Chapters 3, 4, and 5 discuss an optimization for BDD-based optical logic circuits, and the methods proposed in these chapters can be applied simultaneously. However, this thesis does not discuss the optimization of exploiting these methods simultaneously, which will be our future work. Chapter 7 proposes a CBGL as a new scheme for logic circuits, then discuss the optimization for a CBGL. Since a BDD and a CBGL are different logic structures, the techniques proposed in chapter 7 are independent of methods proposed in chapters 3, 4, and 5. Chapter 6 propose a method for reducing power consumption by dividing the logic function into sub-functions. In chapter 6, the sub-functions are implemented by only BDD-based optical logic circuits. However, we can utilize synthesis methods based on the other scheme, such as CBGL, SOP, AIG. In summary, the method proposed in chapter 6 can incorporate both BDD-based methods (chapters 3, 4, and 5) and CBGL-based methods (chapter 7). Accordingly, we can improve the circuits by choosing an appropriate scheme for the sub-functions, which will be our future work.

1.4 Organization

The rest of this thesis is organized as follows. Chapter 2 presents a background to optical logic circuits and then describes that optical logic circuits have fundamental issues regarding size and power consumption. Chapter 3 proposes a method for reducing the size of BDD-based optical logic circuits exploiting wavelength division multiplexing. Chapter 4 clarifies that the power consumption of a BDD-based optical logic circuit largely depends on the BDD variable order, then proposes an algorithm for optimizing the variable order in terms of the power consumption. Chapter 5 demonstrates that the optical logic circuit with a large number of garbage outputs (e.g., the circuit for the XOR function) consumes huge power. We propose a method for eliminating a garbage output by focusing on BDD nodes satisfying certain conditions, drastically reducing power consumption. Chapter 6 proposes a method for exponentially reducing power consumption by exploiting OE converters. Chapter 7 considers the circuit design without garbage output. We propose a new scheme and enumerate the circuits with the minimum number of logic gates for all three-input boolean functions. Chapter 8 concludes this thesis.

Chapter 2

Preliminary–Logic Circuits based on Nanophotonic Devices

This chapter explains nanophotonic devices and convectional logic synthesis methods for optical logic circuits. Then we describe the disadvantages of the conventional synthesis methods.

2.1 Nanophotonic Devices

2.1.1 Optical Pass Gates

An optical pass gate uses a photonic crystal device to control the ON/OFF of light or change the direction of light. There are two types of optical pass gates: one that controls the ON/OFF of light with one optical input and one optical output [4, 67] and one that has multiple optical inputs and optical outputs and controls the optical routing. In this thesis, we call an optical pass gate with one optical input and one optical output 1×1 optical pass gate and an optical pass gate with multiple optical inputs and optical outputs 2×2 optical pass gate. We introduce an electro-absorption modulator(EAM) type and a Mach-Zehnder-interferometer(MZI) type as representative 1×1 optical pass gates. In the electro-absorption type, when a voltage is applied to the control terminal, the absorption coefficient in the semiconductor changes and the intensity of the light is modulated, enabling ON/OFF control of the light [67]. In the Mach-Zehnder interferometer type, the coupler on the input side splits the input light one-to-one, and the coupler on the output side

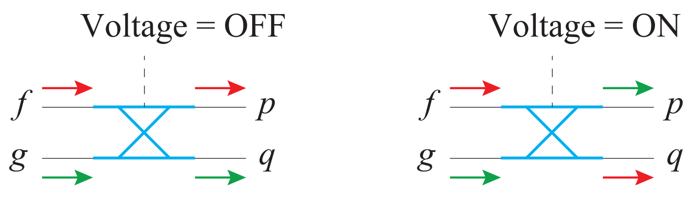


Figure 2.1: Conceptual diagram of a directional coupler.

multiplexes and interferes the two lights, enabling ON/OFF control of the light. The phase of the light is adjusted by controlling the refractive index of one of the two branched optical waveguides with voltage. The intensity of the output light can be controlled since the interference when the two split lights are combined can be controlled by the phase of the light [68,69]. 2×2 optical pass gates include directional coupler type and Mach-Zehnder interferometer type. Here, we describe a directional coupler-type pass gate [4]. Fig. 2.1 shows a conceptual diagram of a directional coupler type 2×2 pass gate. This device uses the phenomenon that the transmitted optical power is transferred between two adjacent parallel optical waveguides. The output destination of light can be controlled by changing the coupling constant of light with the applied voltage. When light is input from $f(g)$, when a voltage of 0 V is given, light propagates to $p(q)$ (this operation is called cross-state), and when a voltage of $V_c(> 0)$ is given, light propagates to $q(p)$ (this operation is called bar-state) [68]. Logic gates with such operations are called cross-bar gates. In this way, the optical pass gate functions as a switch that controls the ON/OFF of light and optical routing. This thesis uses a directional coupler (DC) type 2×2 optical pass gate as a basic building block for logic circuits. We assume that a propagation delay is 1 ps and a signal power attenuation is -1 dB as the light passes through the DC [70].

2.1.2 Ring Resonator

A ring resonator is a ring-shaped optical element that controls the propagation direction of an optical signal having a resonant wavelength. The resonance wavelength of a ring resonator can be controlled by changing the free electron density in the semiconductor of the ring resonator or by changing the temperature [31]. Fig. 2.2 shows a conceptual diagram of ring resonators. The ring resonator corresponding to wavelength λ_t is assumed to resonate

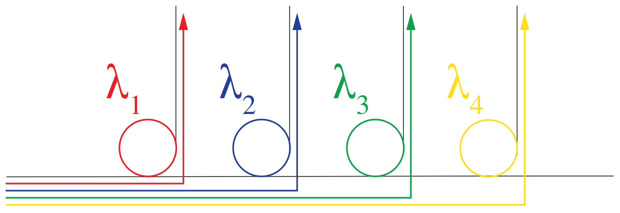
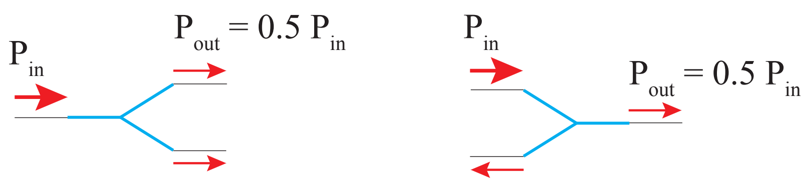


Figure 2.2: Extract a specific light by a ring resonator.

with light having wavelength λ_t. Waveguides are arranged orthogonally along the circumference of the ring resonator. Four types of light having wavelength $\lambda_1, \lambda_2, \lambda_3$, and λ_4 are input from the horizontal direction. The light having wavelength λ_1 is vertically bent by the ring resonator corresponding to wavelength λ_1. This operation is called drop. The lights having wavelength λ_2, λ_3 and λ_4 do not resonate with the ring resonator corresponding to wavelength λ_1 and pass through in the horizontal direction. This operation is called through. Similarly, the lights having wavelength λ_2, λ_3 and λ_4 are dropped by the different ring resonators, then each light propagates to different waveguides. The radius and width of a ring resonator are assumed to be 1.5 μm and 450 nm. The signal power attenuation at a ring resonator is negligibly small compared with the signal power attenuation at a DC [31].

2.1.3 Splitter and Combiner

As shown in Fig. 2.3 (a), a splitter is an optical element that splits a waveguide. When light is input to a splitter, its signal power is distributed to the outputs in a certain ratio. This signal power distribution ratio is called the splitting ratio. A splitting ratio is assumed to be equal. Such a splitter equally splits the signal power of an input light into the outputs. In other words, when light passes through a splitter, the light is attenuated to the original signal power divided by the number of outputs [71]. When there are many splitters on the path, the input light must have large signal power in order to obtain sufficient signal power at the output of the circuit, resulting in increased power consumption of the circuit. A combiner, shown in Fig. 2.3 (b), is an optical element with a structure similar to a splitter. A combiner attenuates the signal power as well as a splitter. In the splitter



(a) Splitter.

(b) Combiner.

Figure 2.3: Power losses in splitter and combiner.

with one input and two outputs shown in Fig. 2.3 (a), light having half the signal power of the input light propagates to the two outputs, which means that a signal power attenuation is 50% at one output. In the combiner with two inputs and one output shown in Fig. 2.3 (b), an input light propagates to both the input and the output, resulting in a signal power attenuation of 50% at the output [71]. Therefore, the signal power attenuation at a splitter with one input and two outputs is −3 dB. The signal power attenuation at a combiner with two inputs and one output is −3 dB as well. Since there is no difference between a splitter and a combiner regarding the signal power loss, we use only a splitter. In this thesis, we assume that a splitter has one input and two outputs, and the splitting ratio is equal unless otherwise noted.

2.1.4 Photodetector

A photodetector(PD) is a device that converts the received light into an electrical signal. A PD using a photonic crystal with an internal capacitance of 0.6 fF and an element length of 1.7 μm has been developed. Its operating speed and photo-electric conversion efficiency are 40 GHz and 1A/W [72]. If the signal power of the light received by the PD is not sufficiently high, noise such as dark current makes it impossible to convert the logic value into a recognizable current value, and the signal detection time increases. In this thesis, the minimum detectable signal power of a PD is assumed to be 10 μW.

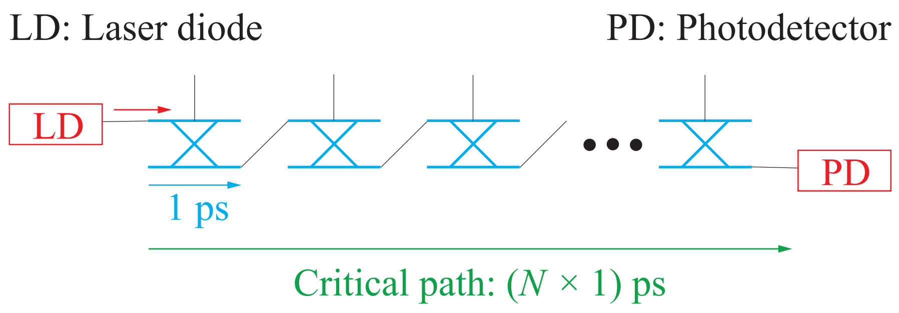
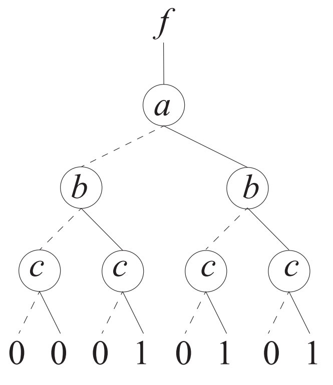


Figure 2.4: Naive optical logic circuit for AND function.

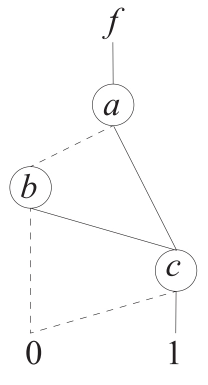
2.2 A Synthesis Method based on Binary Decision Diagram

2.2.1 Naive Optical Logic Circuits

First, we describe a naive optical logic circuit with cross-bar gates. In this thesis, we assume to use a DC as a cross-bar gate. In a DC, the optical routing from inputs to outputs is digitally controlled by the voltage-control signal. Like the pass transistor logic, the Boolean logic can be constructed by serially connected DCs, an optical source, and an optical output. Since an electrical signal digitally controls the path from the optical source to the optical output, the voltage bias corresponding to the value of the input variable in the target logic function is given to the control input of a DC. At the optical output, the PD receives the light from the optical source given from a laser diode (LD). Once the input voltage is given, the latency of the DC is determined by only the speed of the light passing through the DC. In the optical circuit for N-input AND operation, as shown in Fig. 2.4, the latency is N picoseconds (1 ps delay per DC in this paper). Note that the PD in Fig. 2.4 receives the light from the LD if and only if all the voltage-inputs are in the ON state. Therefore the optical circuit in Fig. 2.4 implements the N-input AND function. In order to implement general and larger-scale logic functions, synthesis methods exploiting the ultra-high-speed nature of optical logic circuits are proposed based on various schemes. We especially focus on synthesis methods based on a BDD due to the high-speed and area-efficient



(a) BDT.



(b) Reduced BDD.

Figure 2.5: Bibary decision diagram.

characteristics.

2.2.2 Binary Decision Diagram

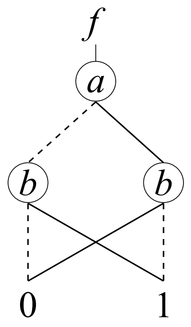
Before explaining BDD-based optical logic circuits, we describe a BDD itself in detail. A binary decision diagram (BDD) is a data structure that is used to represent a Boolean function. A Boolean function can be represented as a rooted, directed, acyclic graph, which consists of variable nodes and two terminal nodes called 0-terminal and 1-terminal. Each variable node is labeled by an input variable of the function and has two child nodes called 0-child and 1-child. The edge from a node to a 0(1)-child represents an assignment of the variable to 0(1). An edge from the parent to a 0(1)-child is called a 0(1)-edge. Fig. 2.5 (a) shows a binary decision tree(BDT) for representing the function $F = \bar{a}bc + ac$. A BDT is one type of BDDs, in which the BDD reduction rules are not applied. We will explain the BDD reduction rules later. The value of the function can be determined for a given variable assignment by following a path down from the root to the terminal. A Dotted line represents a 0-edge, and a solid line represents a 1-edge.

A BDD is called 'ordered' if different variables appear in the same order on all paths from the root to the terminal. A BDD is called 'reduced' if the following two operations are applied: (1)Given node v, if the 0-edge and the 1-edge of node v go to the same node, node v is called a redundant node and can be removed. (2)Given nodes v, u with the same label, if the 0-edges of nodes v and u go to the same node, and the 1-edges of nodes v and u go to the same node, nodes v, u are called equivalent nodes and can be merged. By applying these two reduction rules to the BDT shown in Fig. 2.5 (a), we can obtain the reduced BDD shown in Fig. 2.5 (b). In popular usage, the term BDD almost always refers to Reduced Ordered Binary Decision Diagram (ROBDD). The ROBDD is known to be unique [46]. The advantage of an ROBDD is that it is canonical for a particular functionality. This property makes it useful in functional equivalence checking and other operations like functional technology mapping. A path from the root node to the 1-terminal represents a variable assignment for which the represented Boolean function is true.

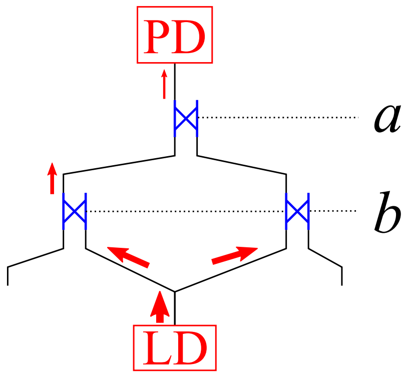
2.2.3 BDD-Based Optical Logic Circuits

Fig. 2.6 (a) shows a BDD representing $f = (\neg a \wedge b) \vee (a \wedge \neg b)$. Fig. 2.6 (b) shows the circuit based on the BDD shown in (a). From BDD-based designs, we consider the root and 1-terminal node as the optical output and optical source, respectively. In the optical implementation of a BDD, we use a DC as the BDD node and a splitter as the branch. The electrical inputs corresponding to input variables are fed into DCs corresponding to the BDD nodes to control the direction of the light. The speed of the BDD-based circuit is very fast, as determined by the speed of the light passing through serially connected DCs. The computational delay of the circuit for an n-input function is n ps (1 ps delay per DC in this paper). This ultra-high-speed characteristic is the basic motivation underlying the use of BDD-based design for optical logic circuits. The number of BDD nodes is assumed to represent the area of the circuit since the number of DCs is equal to the number of nodes, and the area of the splitter is negligible compared with the DC. BDDs can represent general logic functions with a small number of nodes. Therefore, BDD-based optical logic circuits have a high area efficiency. These ultra-high-speed and area-efficient characteristics are the basic motivation for using BDD-based design for optical logic circuits.

In contrast with the advantage in speed, the fundamental disadvantage of the optical circuits against the CMOS LSI circuits is their size. Although the size of optical devices has been drastically reduced in the last decade thanks to the advancement of nanophotonics technologies, it is still very large [4]. For example, the size of nanophotonic directional couplers is several tens of



(a) BDD.



(b) Optical circuit.

Figure 2.6: BDD-based Optical Logic Circuit.

micrometers [4], which is more than two orders of magnitude bigger than CMOS logic gates. From a circuit-level point of view, what can be done to reduce the size of the circuit is to reduce the number of nodes in a BDD. To reduce the number of nodes, a method of sharing a common sub-graph among multiple logic functions is proposed [59]. This shared-graph representation is called shared BDD. The BDD manipulation method largely contributes to reducing the size of BDDs. However, if there is only a small part of the graph that can be shared among multiple logic functions, the size cannot be reduced sufficiently. This paper, for the first time, proposes a method of more widely sharing a BDD among multiple logic functions exploiting an idea of wavelength division multiplexing (WDM). This method can reduce a large number of gates in a wider range of conditions than the method based on shared BDD. This method is described in chapter 3.

Another big issue of optical circuits based on BDD is a large number of branches, which causes a large amount of power loss in the corresponding power splitters. This results in a large power dissipation in laser sources to maintain a specific signal power level at the outputs of the circuit. In [35], a technique to synthesize splitter-free BDD representations is proposed. This reduces the signal power loss when the light passes through the optical power splitter, which corresponds to a branch in BDDs. However, the splitter-free BDD representations require combiners which also involve a large amount of power loss. As explained in the previous section, a combiner causes a significant signal power loss as well as a splitter. This corresponds to the fan-in loss of the combiner. Therefore, even if there is no splitter in a path, the signal power is degraded by -3 dB if there is a combiner in the path. In [36], a more practical method to reduce the insertion losses from optical combiners and switches is proposed. However, the reduction of power loss by this method is limited. In this thesis, we propose a more radical solution for reducing the power loss at splitters in chapters 3, 4, and 5. However, these methods can not completely eliminate splitters, and there is no research that proposes a method for completely eliminating splitters. Therefore, we consider a circuit design without splitters in chapter 7.

2.3 Synthesis Methods for Optical Logic Circuits

This section describes synthesis methods based on a virtual gate, the sum of products (SOP) form, and the exclusive sum of products (ESOP) form. The circuits based on these methods operate in the same principle as BDD-based optical logic circuits. The light from the optical source passes through the path consisting cross-bar gate controlled by an electrical signal. Then we can obtain the output value of the target logic function by detecting the light at the optical output.

Virtual Gate

A synthesis method based on a virtual gate (VG) is proposed [30]. A VG denotes a circuit implementing a 2-input logic function, such as AND, OR, and XOR, with cross-bar gates. Since cross-bar gates in a VG can be replaced with arbitral VG, a VG-based synthesis can implement arbitral logic function by exploiting nested structures. This synthesis method has the advantage of being splitter-less. Unfortunately, VG-based circuits have a large number of loops, which increases the number of gates from the optical source to the optical output. Therefore, a VG-based circuit for a large-scale logic function results in significant power loss due to signal power attenuation at a cross-bar gate. As for the size, VG-based circuits generally have a larger number of gates compared with BDD-based circuits.

Sum of Products and Exclusive Sum of Products

Synthesis methods based on an SOP form and an ESOP form are proposed [32]. An SOP represents a Boolean function with a disjunction (OR) of conjunction (AND) of literals. An ESOP form represents a Boolean function with an exclusive disjunction (XOR) of conjunction (AND) of literals. A literal is either an input variable of the logic function or its negation. In both SOP- and ESOP-based circuits, all product terms are implemented by a virtual gate corresponding to the AND function. SOP-based circuits sum up product terms with a gate implementing OR operation. ESOP-based circuits sum up product terms with a gate implementing XOR operation. These synthesis methods based on SOP and ESOP result in compact circuits compared with a straightforward mapping from the truth table. However, these methods have a fundamental disadvantage: the number of virtual gates corresponding to AND function increases as the number of product terms increases. Even when the target logic function can be expressed by a compact BDD, the SOP form sometimes has a large number of product terms, which increases the number of gates in the SOP-based circuit.

Comparison with BDD-based Methods

Compared with synthesis methods based on VG, SOP form, and ESOP form, BDD-based circuits have high-speed and area-efficient characteristics due to a short path from the root node to the terminal node and strong reduction rules in a BDD. On the other hand, BDD-based circuits have more splitters than the other methods, which results in significant power loss. Therefore, our research direction of addressing the issues regarding splitters is essential to enhance the advantage of BDD-based synthesis methods.

2.4 Serial-Connection and Cascade-Connection

There are two ways to connect DCs: serial-connection and cascade-connection. As shown in Fig. 2.7, serial-connection is a way to connect the optical output of a DC to the optical input of the next DC, and cascade-connection is a way to connect the optical output of a DC to the voltage control input of the next DC via the OE converter. In order to take advantage of the ultra-high speed characteristic of a PIC, optical logic circuits generally consist of a serial-connection. However, connecting DCs with serial connection exponentially

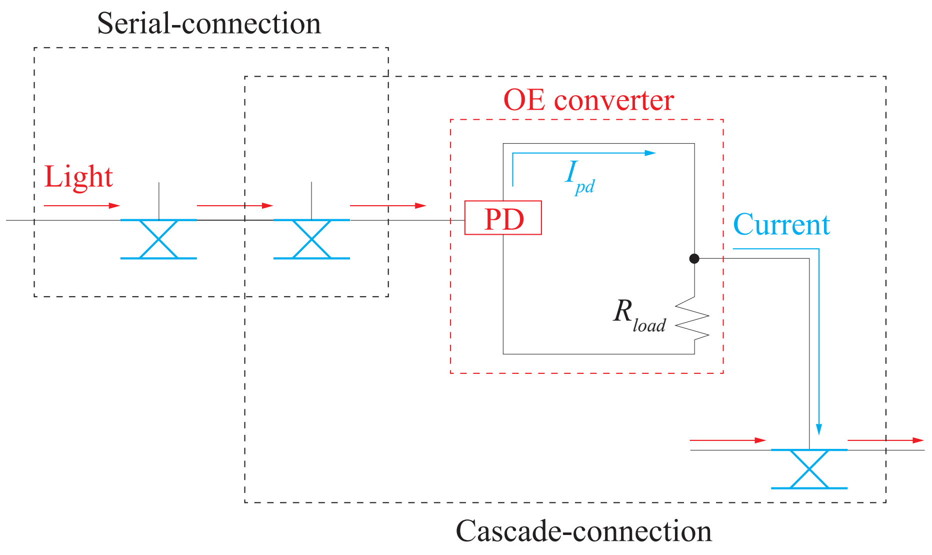


Figure 2.7: Serial-connection and cascade-connection.

increases the signal power attenuation to the number of DCs, resulting in significant power consumption. On the other hand, cascade-connection can restore the signal, which reduces the signal power attenuation. As shown in Fig. 2.8, the optical input of the DC given an electrical signal via the OE converter is input light from the LD. This circuit can convert a low-power signal from the DC with serial-connection to a high-power signal from the LD. In this thesis, such an optical amplification mechanism is called an OE/EO converter. Note that an OE/EO converter can not maintain the wavelength or phase of light. In contrast with the power consumption, an OE/EO converter has a fundamental disadvantage of a low operation speed. The OE conversion delay is much larger than the light propagation time in the serial-connection. For example, the serial connection delay of the DC presented in [69] is 1 ps per DC, while the OE conversion delay is 25 ps [72]. Although a method for reducing the power consumption of BDD-based optical logic circuits by OE/EO converters [38], this method simply inserts OE/EO converters in a path from the LD to the PD, which increases the delay of the circuit. Therefore, we propose a method for mitigating the OE converter delay overhead by parallel execution in chapter 6. Our method can drastically

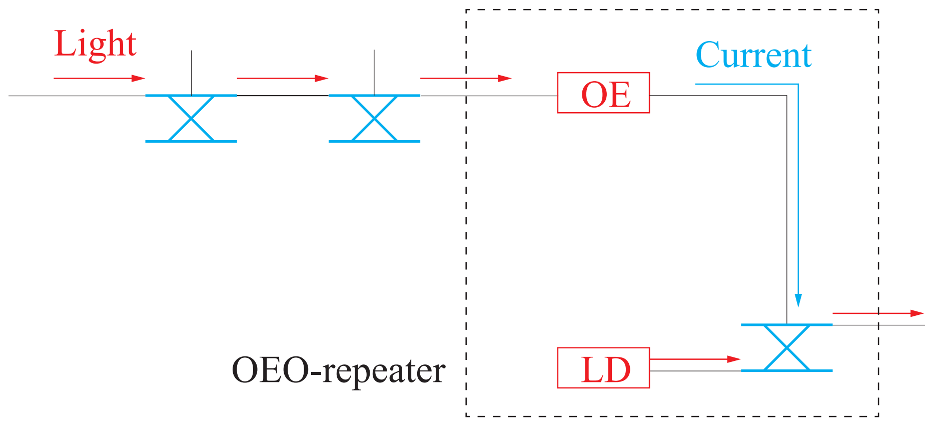


Figure 2.8: OEO-repeater.

reduce the power consumption of a large-scale optical logic circuit without increasing the delay.

Chapter 3

Wavelength Division Multiplexing and Splitter Elimination

3.1 Introduction

3.1.1 Background and Our Contribution

In contrast with the advantage in speed, the fundamental disadvantage of the optical circuits against the CMOS LSI circuits is the size and power consumption. In this chapter, we propose a method of more widely sharing a BDD among multiple logic functions by exploiting the idea of wavelength division multiplexing (WDM). The optical waves having different wavelengths do not interfere with each other. Therefore, by using different wavelengths for different logic sub-functions, an optical sub-circuit can be shared among multiple different logic sub-functions. This largely reduces the size of the BDD-based optical circuit. Although a similar idea is proposed in [37], this method applies WDM to share the circuit among different functions. On the other hand, our method can be applied to sub-functions in a single function, which is a big difference from the method proposed in [37]. In addition, we propose a method for reducing power consumption by reducing power losses at a splitter. Although methods for reducing power loss at a splitter are proposed [35, 36], the reduction in power losses by these methods is limited. Our method is a more radical solution that eliminates splitters by replacing them with additional DCs in an optical circuit. This reduces the power losses

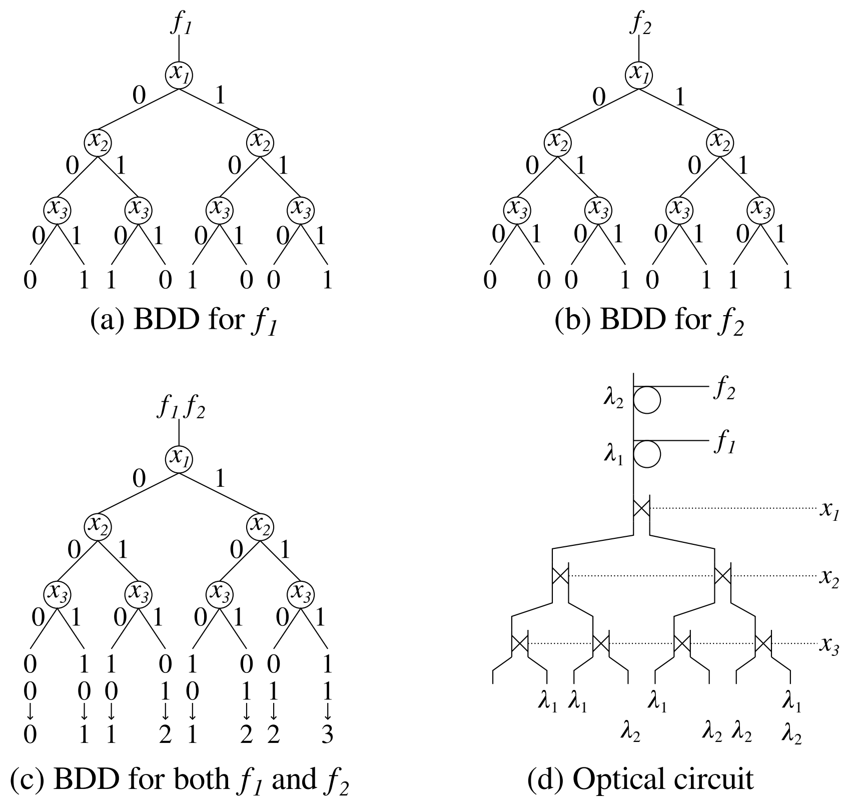


Figure 3.1: Sharing BDDs exploiting WDM. (Copyright(C)2019 IEICE)

in the optical circuit since the power loss in a DC is much smaller than that of a splitter.

3.1.2 Motivational Example

By using optical signals with different wavelengths to each other, an optical circuit can be shared by the multiple optical signals without mutual interference. A minimum spacing of neighboring wavelengths is around 1.3 nm [73]. It is experimentally demonstrated that optical inter-channel interference is negligible for channels with a wavelength spacing of 1.3 nm [73]. In [74], a DC which operates over a wide wavelength range from 1500 nm to 1600 nm is proposed. In this case, the DC can be functional for 75 WDM optical signals. Suppose we have two different logic functions $f_1(x_1, x_2, x_3)$ and $f_2(x_1, x_2, x_3)$ as shown in Fig. 3.1 (a) and (b). By using wavelengths of λ_1 and λ_2 for the

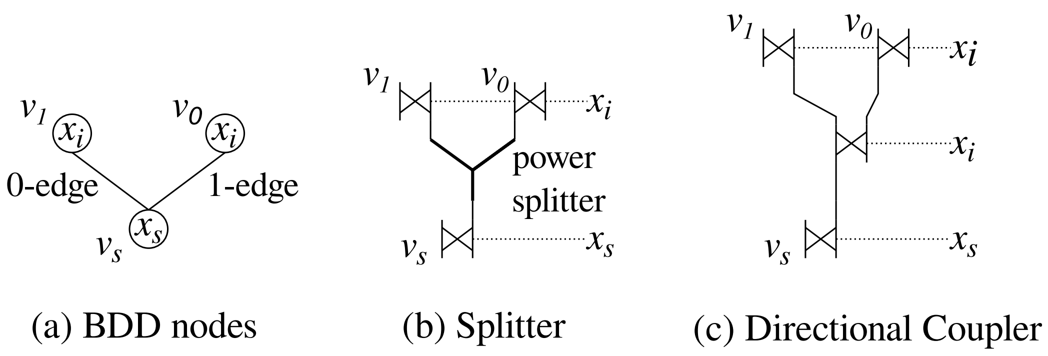


Figure 3.2: Optical implementations for a branch in BDDs. (Copyright(C)2019 IEICE)

terminal nodes (i.e. leaves) in f_1 and f_2, respectively, a single optical circuit can be shared by the functions f_1 and f_2 as shown in Fig. 3.1 (d). This largely reduces the number of optical devices used in an optical circuit. However, this shared structure needs wavelength filters at the output to extract an optical signal having a specific wavelength as shown in Fig. 3.1 (d). Typically a ring resonator is used for the wavelength filter [31]. The diameter of the ring resonator is an order of wavelength which is one order of magnitude smaller than DCs. However, the resonator involve a few picoseconds delay to extract a specific optical signal. Therefore, it is very important to consider the tradeoffs among area, power and delay of the circuit when applying the WDM technique.

Since BDDs inherently have a large number of branches if their size is well reduced. This causes significant power losses in the corresponding optical circuit. Let v_s in Fig. 3.2 (a) be a BDD node. The v_s has two predecessors v_0 and v_1 having the same control input x_i as shown in Fig. 3.2 (a). Let us suppose a signal is propagated from v_s to v_0 and v_1. In this case, a naive optical implementation of the graph needs a splitter as shown in Fig. 3.2 (b). This involves a power attenuation by half (i.e. -3 dB attenuation). However, if the output of v_s is split into the 1-edge of v_1 and the 0-edge of v_0, the splitter can be replaced by a DC as shown in Fig. 3.2 (c) since the output of v_s is used in v_1 only if $x_i = 1$ and used in v_0 only if $x_i = 0$. The power loss involved in the DC is smaller than that of the splitter since the DC directionally outputs the optical signal while the splitter provides the signal power to both outputs. This replacement reduces the -3 dB power loss in the splitter to -1 dB loss in the DC [70] [69]. However, this replacement involves a delay overhead since the delay of the DC is around 1 ps while that of the splitter is negligible. The size of the splitter is also negligible compared with that of the DC. Therefore, it is also very important to consider the tradeoffs among area, power and delay of the circuit when replacing splitters by DCs.

3.2 BDD Manipulation for Optical Circuits

3.2.1 Reduction with Wavelength Division Multiplexing

This section presents a method of synthesizing an optical logic circuit exploiting WDM. The method consists of the following 3 operations; i) Inter-function sharing, ii) BDD reduction, and iii) Intra-function sharing. The operations above sometimes increase the delay and/or power consumed in the circuit. This subsection provides the operations to reduce the size of optical circuits considering the tradeoffs among the area, delay and power, which help designers optimize the circuit according to the design goals and constraints given by the designers.

Inter-Function Sharing

Fig. 3.1 shows an example of synthesizing an optical circuit which implements two logic functions f_1 and f_2 together. This idea can be generalized into the following procedure for synthesizing an optical circuit shared by k different logic functions exploiting WDM as shown in Fig. 3.1 (d).

The procedure receives Binary Decision Trees (BDTs) for $f_0, f_1, ..., f_k$ which are all Boolean functions with n-input as inputs of the procedure. It outputs an optical circuit implementing the BDTs for the logic functions all together. If the value of the i-th terminal node of the BDT implementing the function f_l is 1, a light with the wavelength λ_l is given to an optical source of the circuit corresponding to the i-th terminal node of the BDT. This means that a different function uses a different wavelength. A ring resonator shown at the output of Fig. 3.1 (d) is needed to extract an optical signal with a specific wavelength to detect a result of the corresponding function. The extraction of a specific optical signal with the ring resonator involves a considerable delay when the signal is bent in the ring resonator.

BDD Reduction

The idea behind BDDs can be easily adapted to more general binary decision diagrams such as Multi-Terminal BDD (MTBDD) [75]. The MTBDD is like an ordinary BDD except that the terminal nodes can be arbitrary integer values instead of just 0 and 1. The circuit for the MTBDD can be constructed using the inter-function sharing method in Sec. 3.1.1. As a result, the ordinal BDD reduction technique can be applied to the MTBDD structure, which enables to further reduce the number of DCs. This section presents the ordinal BDD reduction method for the optical MTBDD circuit.

MTBDDs can be used to represent functions that map vectors with binary values into the integers. Let D_n be the set $\{0,...,2^{n+1}-1\}$ of integers that can be represented with n+1 bits, and let B be the set consisting of the Boolean values 0 and 1. Let f: $B^m \rightarrow D_n$ be a function that maps Boolean vectors of length m into the set D_n. The function f is expressed as a summation

$$f(\bar{x}) = \sum_{i=0}^{n} f_i(\bar{x}) \cdot 2^i, \tag{3.1}$$

where each f_i has value 0 or 1 and is represented as a BDD. For example, considering f_1 shown in Fig. 3.1 (a) and f_2 shown in Fig. 3.1 (b) as $f_0(\bar{x})$ and $f_1(\bar{x})$ in (3.1), respectively, the possible values for $f(\bar{x})$ can be obtained by (3.1) as integers. If we give the integer values obtained by $f(\bar{x})$ in (3.1) to the terminal nodes of a BDD as shown in Fig. 3.1 (c), an MTBDD which implements f_1 and f_2 together can be constructed. This functionality of handling integers in BDD manipulation is very useful for the design and verification of arithmetic circuits [75]. Reduction methods applied to ordinary BDDs can be also applied to MTBDDs, which effectively reduces the number of nodes without using ring resonators unlike inter-function sharing. However, the BDD reduction usually causes an increase of the number of branches which are implemented by splitters in an optical circuit. This results in an increase of the power dissipated in laser sources.

Intra-Function Sharing

Inter-function sharing can be also applied to a sub-graph of a BDT. Suppose we have a BDT with the co-factors f_1 and f_2 as shown in Fig. 3.3. The co-factors f_1 and f_2 can be merged into an optical sub-circuit by assigning different wavelengths to terminal nodes of f_1 and those of f_2. This method

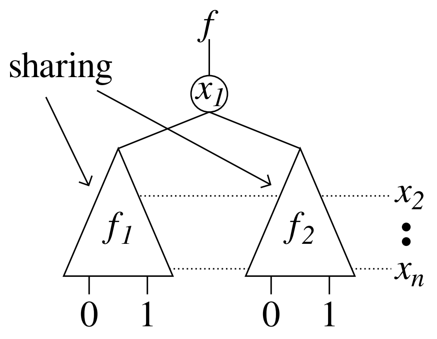


Figure 3.3: Intra-function sharing. (Copyright(C)2019 IEICE)

can be applied even if the values in the terminal nodes of the f_1 and f_2 are different from each other. On the other hand, BDD reduction can only be applied if the values in the terminal nodes of the f_1 and f_2 are the same. This needs ring resonators internally to select the output of f_1 and f_2 so that the results of f_1 and f_2 can be separately given to 0-edge and 1-edge of the node labeled x_1. This involves a delay overhead similar to the inter-function sharing. Therefore, making too many intra-function sharing sub-circuits leads to a significant degradation of the circuit performance.

More specific example is shown in Fig. 3.1 (d) and Fig. 3.4. Fig. 3.1 (d) shows an optical circuit where the possible outputs of the corresponding function are ranging from 0 to 3, which can be encoded as 2-bit binary numbers. Therefore, two wavelengths λ_0 and λ_1 are sufficient for representing the values given to the leaves as shown in Fig. 3.1 (d). Fig. 3.4 shows an optical circuit whose size is reduced from the circuit shown in Fig. 3.1 (d) using the intra-function sharing method. The DCs located in the third stage are merged into one DC. In this case, four times more wavelengths are needed so that multiple lights given to the DC do not interfere to each other. Since the lights have to be redirected to the second stage DCs, ring resonators are located between the first and the second stages. In this thesis, the wavelength band of the light extracted by the ring resonator is defined as a pass-band. The four ring resonators should be designed such that the pass-band includes multiple wavelengths used by the lights. For example, the lights with wavelengths λ_6 and λ_7 are extracted by the resonator R3, and the lights with wavelengths λ_2 and λ_3 are extracted by R1. The possible outputs of the corresponding function are encoded as 2-bit binary numbers. For example, the optical signals having λ_6 and λ_2 correspond to 2^0, while λ_7 and λ_3 correspond

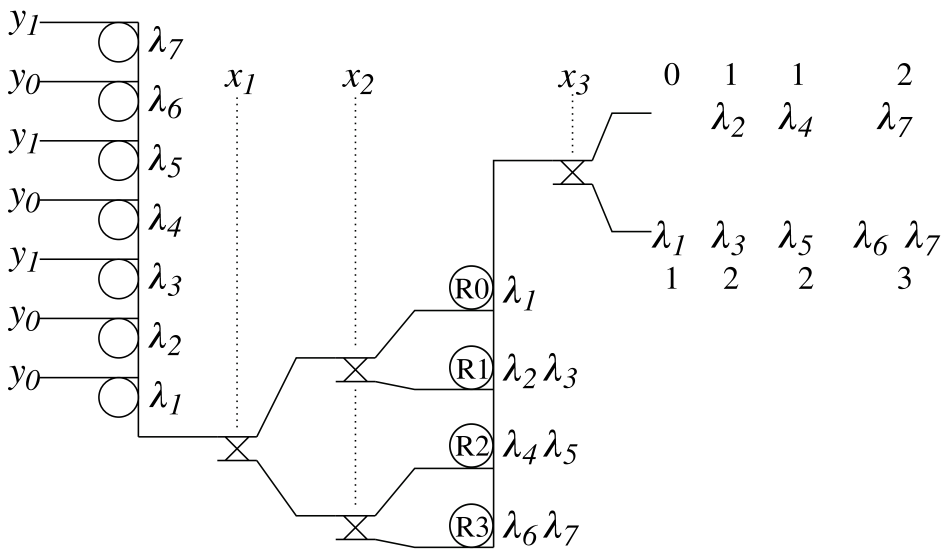


Figure 3.4: Example of one-stage intra-function sharing. (Copyright(C)2019 IEICE)

to 2^1. At the outputs of the circuit, y_0 corresponds to 2^0 and y_1 corresponds to 2^1. For example, if the input variables $(x_1, x_2, x_3) = (1, 1, 1)$, the light having wavelength λ_6 propagates to the output y_0 and the light having wavelength λ_7 propagates to the output y_1. The output value is 11 as a 2-bit binary number since the lights are detected at both y_0 and y_1. If the input variables $(x_1, x_2, x_3) = (0, 0, 1)$, the light having wavelength λ_1 propagates to the output y_0. The output value is 01 as a 2-bit binary number since the light is detected at the output y_0 and the light is not detected at the output y_1. Such a pass-band setting can be realized by using a ring resonator having a low quality-factor or placing multiple ring resonators in parallel.

3.2.2 Splitter Elimination

As explained in Fig. 3.2, the splitter can be replaced by a DC. However, as shown in Fig. 3.5, if the v_s is connected to the edges of the same type such as 0-edges of predecessors (i.e. v_a and v_b in this case), the splitter elimination explained with Fig. 3.2 cannot work. Even in this case, however, if a path from v_s to one type of edge of a node controlled by x_n and another path from the v_s to another type of edge of a node controlled by the x_n do not

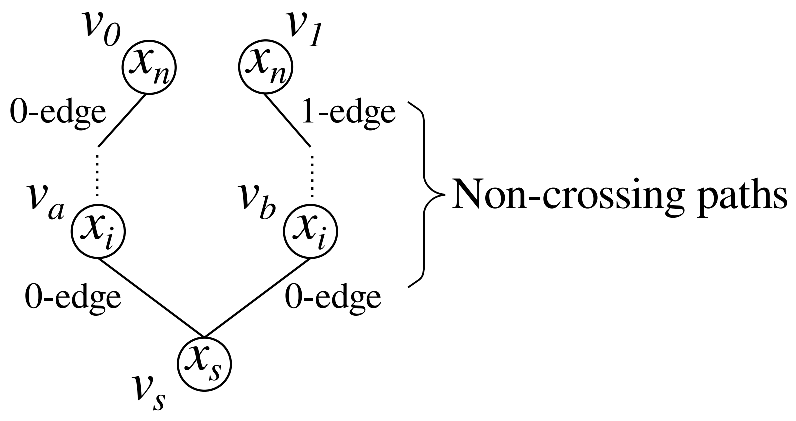


Figure 3.5: Splitter elimination in more complicated cases. (Copyright(C)2019 IEICE)

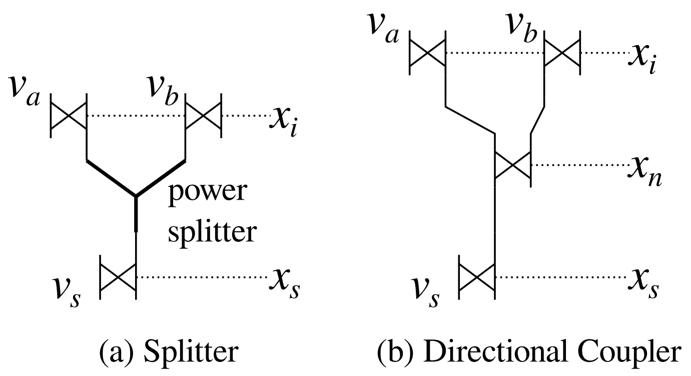


Figure 3.6: Splitter replacement by DC. (Copyright(C)2019 IEICE)

intersect each other as shown in Fig. 3.5, the splitter can be replaced by a DC controlled by the x_n as shown in Fig. 3.6. This is because the output of v_a affects the final result of the function only if the x_n=0 and the output of v_b affects the result only if the x_n=1. This condition includes a case that the two nodes v_0 and v_1 are identical.

If there is a branch with more than 2 outputs, the procedures presented above cannot work effectively. For example, the black node V in Fig. 3.7 has a branch with 3 outputs. The naive optical implementation for the 3-output branch is a 3-output splitter. Let us consider to replace the 3-output splitter with a DC-tree like an optical circuit shown in Fig. 3.8 (c). The A, B and C at the bottom of Fig. 3.8 (c) correspond to the 3-output of the DC-tree. The

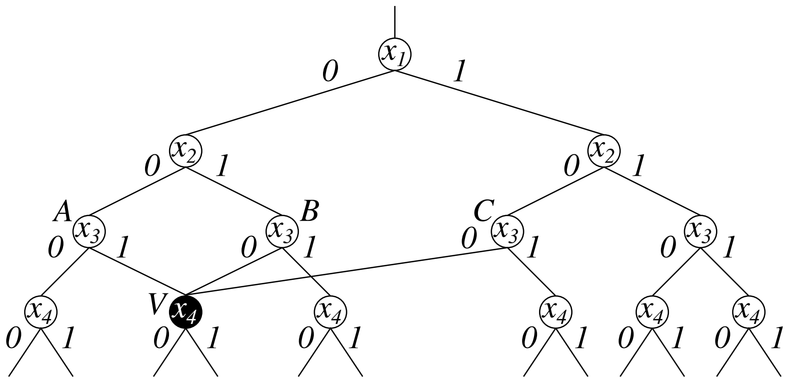


Figure 3.7: Example of a 3-output branch in BDD. (Copyright(C)2019 IE-ICE)

first step to synthesize the DC-tree is to give labels on the three predecessors of the node V. If we give labels A, B and C on the end points of the three paths directed from the root to the node V, respectively, we obtain the BDD as depicted in Fig. 3.8 (a). The sub-graph shown in Fig. 3.8 (a) is the most naive graph representation of the DC-tree. The root corresponds to the input and the terminal nodes A, B and C correspond to the outputs of the DC-tree, respectively. If we use the root of the sub-graph shown in Fig. 3.8 (a) as an input and A, B and C as outputs, we can use the counterpart optical circuit as a replacement of the 3-output splitter. This reduces the signal power attenuation in the circuit. However, the sub-graph shown in Fig. 3.8 (a) has several redundant nodes which can be eliminated by BDD reduction. We can treat the terminal nodes other than the nodes labeled with A, B and C as "don't care" as shown in Fig. 3.8 (a). This is because, if the sub-graph is used as a replacement of the 3-output splitter, the don't care nodes do not affect the final result of the logic function represented by Fig. 3.7 at all. If we apply BDD reduction to the graph in Fig. 3.8 (a), we can obtain a graph in Fig. 3.8 (b), which has the minimum number of nodes required for the replacement of the 3-output splitter. An optical circuit corresponding to the graph shown in Fig. 3.8 (b) is depicted in Fig. 3.8 (c). If we use the top port of the circuit shown in Fig. 3.8 (c) as an input and A, B and C as outputs, we can use this optical circuit as a replacement of the 3-output splitter.

Let us generalize this splitter elimination method. Suppose we have a n-output branch in a BDD where n is more than 1. This branch can be implemented by a n-output splitter. The n-output splitter can be replaced

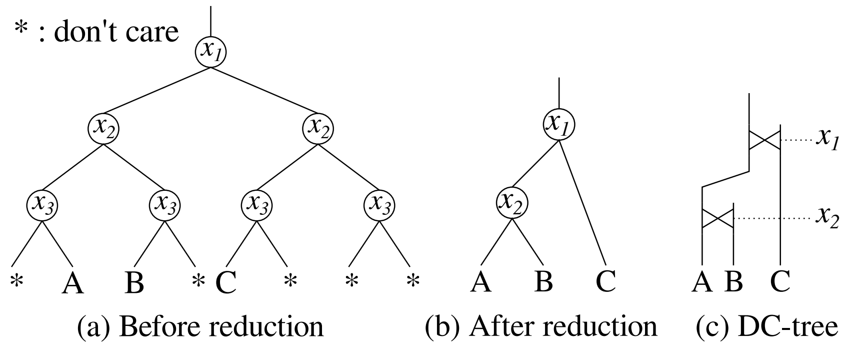


Figure 3.8: BDD and corresponding optical circuit for splitter elimination. (Copyright(C)2019 IEICE)

by DCs in the following procedure. First, we extract a sub-graph which includes all the paths without excess from the outputs of the branch to the root of the BDD. Second, we apply BDD reduction to the sub-graph for minimizing the number of nodes in the sub-graph. This also minimizes the number of nodes connected in series, which leads to a minimization of the delay in the optical circuit corresponding to the minimized sub-graph. Third, we implement an optical circuit based on this minimized sub-graph. Finally, we replace the n-output splitter with the optical circuit corresponding to the minimized sub-graph by using the root of the circuit as input and the leaves as outputs.

3.3 Application to Arithmetic Logic Function

3.3.1 Experimental Setup

The propagation delay of a light passing through a DC is assumed to be 1 ps [69]. We also assume that the delay of a ring resonator is 2 ps when a light with a specific wavelength is extracted and the delay of just passing straight is negligible [31]. The number of DCs used in a circuit is assumed to represent the area of the circuit since the area of the DC [69] is about two orders of magnitude larger than those of the ring resonator [31] and the splitter [76]. We assume that the power attenuation in a DC is -1 dB [70] [69] and that of the splitter is -3 dB [71]. When a light is passing through a ring resonator, the power attenuation of a light is negligible. Also, when a light with a specific wavelength is extracted, the light is propagated to the crossing waveguide without the power attenuation [31]. For the circuits based on Shared-BDD, we apply an idea proposed in [36] which optimizes the splitting ratios of the splitters to reduce the insertion losses of the splitters. Based on the assumptions on the power attenuation for the optical devices, we estimate the power dissipated in laser sources. The output power of the laser sources are determined so that the signal power level of every primary output is no less than 10 μW. The synthesis of the circuit is performed manually.

3.3.2 Parallel Counter

The circuit diagram of a 7-3 parallel counter is shown in Fig. 3.9. The circuit counts the number of 1s in inputs (i.e. $x_1, x_2, ..., x_6, x_7$). The synthesis results are summarized in Table 3.1. The power values represent the total power consumption in the laser sources. The results of the circuit synthesized based on shared-BDD are shown in the bottom of Table 3.1. Shared-BDD method is often used for synthesizing circuits based on BDD. We start the circuit synthesis from a naive BDT (Binary Decision Tree) which does not incorporate any modification. The specification of this circuit corresponds to the top row in Table 3.1. We then apply inter-function sharing. Since the 7-3 parallel counter has 3-bit outputs, there are two more logic functions behind of this circuit. The optical circuit shown in Fig. 3.9 is shared by three logic functions using WDM, which means that the inter-function sharing is incorporated. This circuit then incorporates the BDD reduction. Finally, the splitter elimination is applied. The result is shown in the row of "Ad-hoc". The power consumption is largely reduced compared to the results of Shared-BDD while the delay is more than double and the number of DCs is unchanged. Given a specific objective function such as area or power to be minimized, either the area or the power can be further reduced, although the other criteria may be sacrificed. In this experiment, we show the synthesis results against two different objective functions; one is size minimization and the other is power minimization. For minimizing the number of DCs, inter-function sharing and BDD reduction are applied. The number of DCs can be reduced to one third by sharing one optical circuit by three different logic functions using inter-function sharing. Moreover, the number of DCs is reduced by BDD reduction. Note that splitter elimination is not applied for the objective of size minimization since the number of DCs

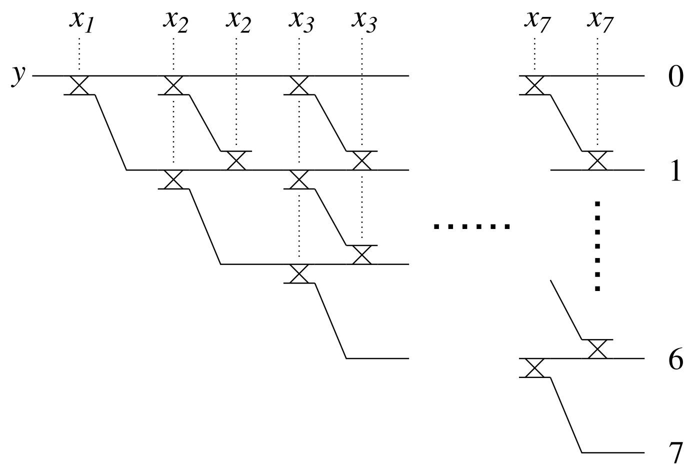


Figure 3.9: Circuit diagram of 7-3 parallel counter. (Copyright(C)2019 IEICE)

Table 3.1: 7-3 parallel counter

	Power [mW]	Delay [ps]	# DCs
Original BDT	22.9	7	381
Ad-hoc	4.45	15	49
Minimum area	69.0	9	28
Minimum power	2.28	13	86
Shared-BDD	30.3	7	49

increases if we replace splitters with DCs. The result is shown in the row of “Minimum area”. For minimizing the power, BDD reduction and splitter elimination are applied. The number of leaves which correspond to the laser sources is reduced by BDD reduction. Although this reduces the number of laser sources, the total power dissipated in the laser sources is unchanged if splitters are used for delivering a strong optical signal to multiple nodes. However, the power can be reduced by replacing the splitters with DCs since it reduces the attenuation of the light going through the path. Note that inter-function sharing is not applied for the objective of power minimization. This is because the number of leaves which correspond to the laser sources increases if the inter-function sharing is applied. This result is shown in the row of “Minimum power”.

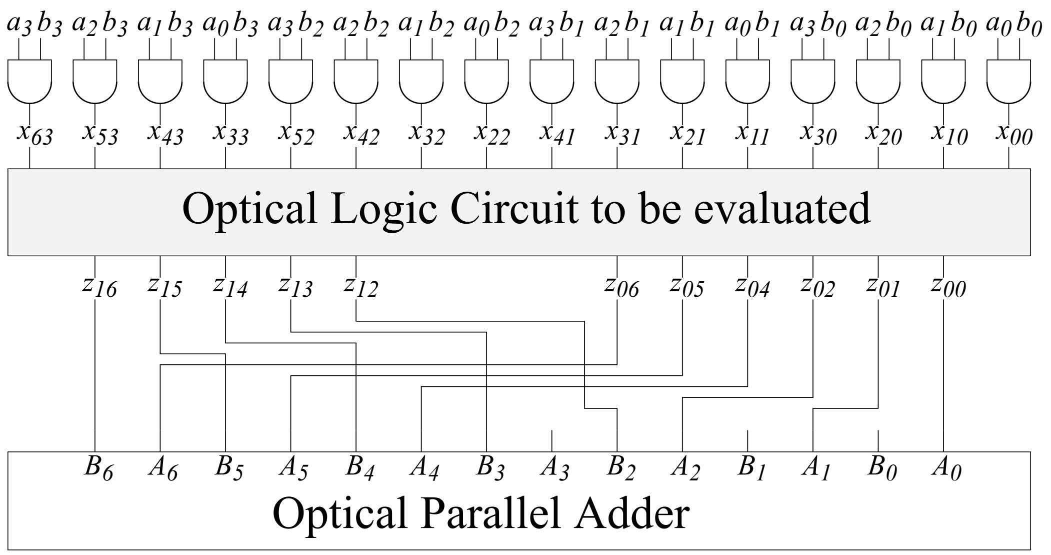

a3 b3 a2 b3 a1 b3 a0 b3 a3 b2 a2 b2 a1 b2 a0 b2 a3 b1 a2 b1 a1 b1 a0 b1 a3 b0 a2 b0 a1 b0 a0 b0
x63 x53 x43 x33 x52 x42 x32 x22 x41 x31 x21 x11 x30 x20 x10 x00
Optical Logic Circuit to be evaluated
z16 z15 z14 z13 z12 z06 z05 z04 z02 z01 z00
B6 A6 B5 A5 B4 A4 B3 A3 B2 A2 B1 A1 B0 A0
Optical Parallel Adder

Figure 3.10: Entire circuit structure for 4-bit parallel multiplier. (Copyright(C)2019 IEICE)

3.3.3 4-bit Multiplier

Let us explain our synthesis method using the following 4-bit parallel multiplication example.

$$\begin{array}{ccccccc}
 & & & a_3 & a_2 & a_1 & a_0 \\
\times & & & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & x_{30} & x_{20} & x_{10} & x_{00} \\
 & & x_{41} & x_{31} & x_{21} & x_{11} & \\
 & x_{52} & x_{42} & x_{32} & x_{22} & & \\
x_{63} & x_{53} & x_{43} & x_{33} & & & \\
\hline
z_{06} & z_{05} & z_{04} & z_{03} & z_{02} & z_{01} & z_{00} \\
z_{16} & z_{15} & z_{14} & z_{13} & z_{12} & z_{11} & z_{10}
\end{array}$$

Fig. 3.10 shows an entire optoelectronic circuit structure for the 4-bit parallel multiplier. We assume that the partial products are generated by CMOS AND-gates as shown in Fig. 3.10. Once the electrical signals corresponding to the partial products are generated by the AND-gates, those are given to the control terminals of the optical logic circuit which accumulates the partial products and outputs two binary numbers; $(z_{06}, z_{05}, z_{04}, z_{03}, z_{02}, z_{01}, z_{00})_2$ and $(z_{16}, z_{15}, z_{14}, z_{13}, z_{12}, z_{11}, z_{10})_2$. In the optical logic circuit shown in Fig. 3.10,

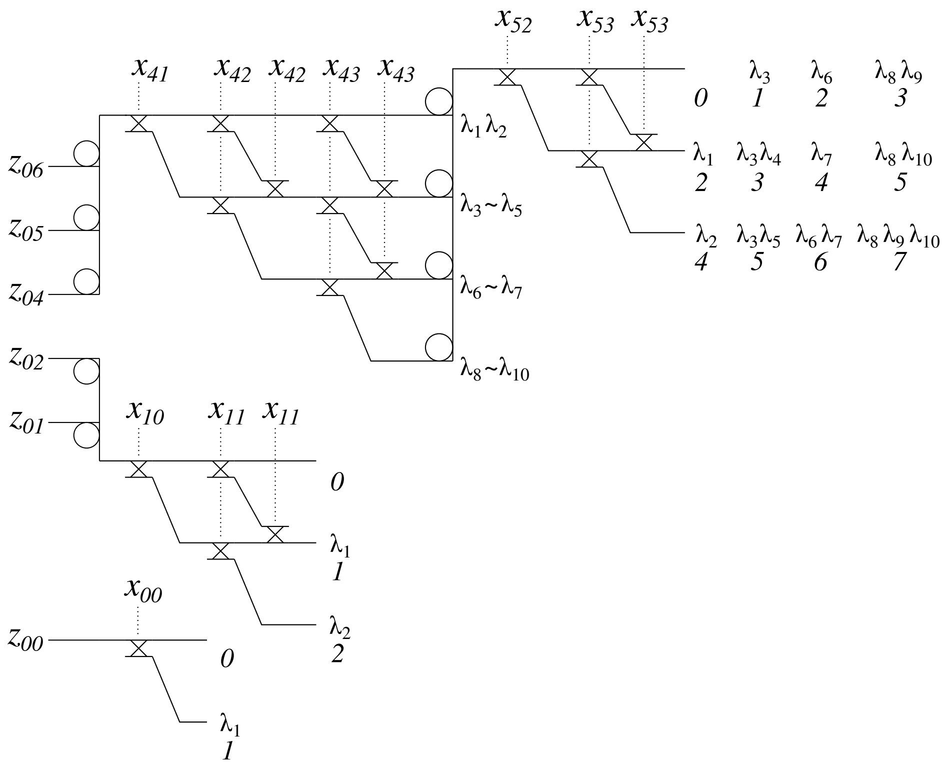


Figure 3.11: Circuit calculating the weighted sum for partial products of the 1st and 2nd digits and the 5th and 6th digits in parallel. (Copyright(C)2019 IEICE)

we accumulate the partial products using two separate circuit blocks in parallel. One circuit block accumulates the partial products in the 1st, 2nd, 5th and 6th digits as shown in Fig. 3.11 and generates the binary number $(z_{06}, z_{05}, z_{04}, z_{03}, z_{02}, z_{01}, z_{00})_2$. The other circuit block accumulates the partial products in 3rd, 4th and 7th digits as shown in Fig. 3.12 and generates the binary number $(z_{16}, z_{15}, z_{14}, z_{13}, z_{12}, z_{11}, z_{10})_2$.

The optical parallel adder presented in [17] can be used for obtaining the final sum in this multiplication. However, the parallel adder circuit for the final sum is out of scope for this evaluation. Fig. 3.11 shows a circuit which calculates the weighted sum for the partial products of the 1st, 2nd, 5th and 6th digits and Fig. 3.12 shows a circuit calculating the weighted sum for the partial products of the 3rd, 4th and 7th digits. These circuits already incorporate inter-function sharing, BDD reduction, splitter elimination and

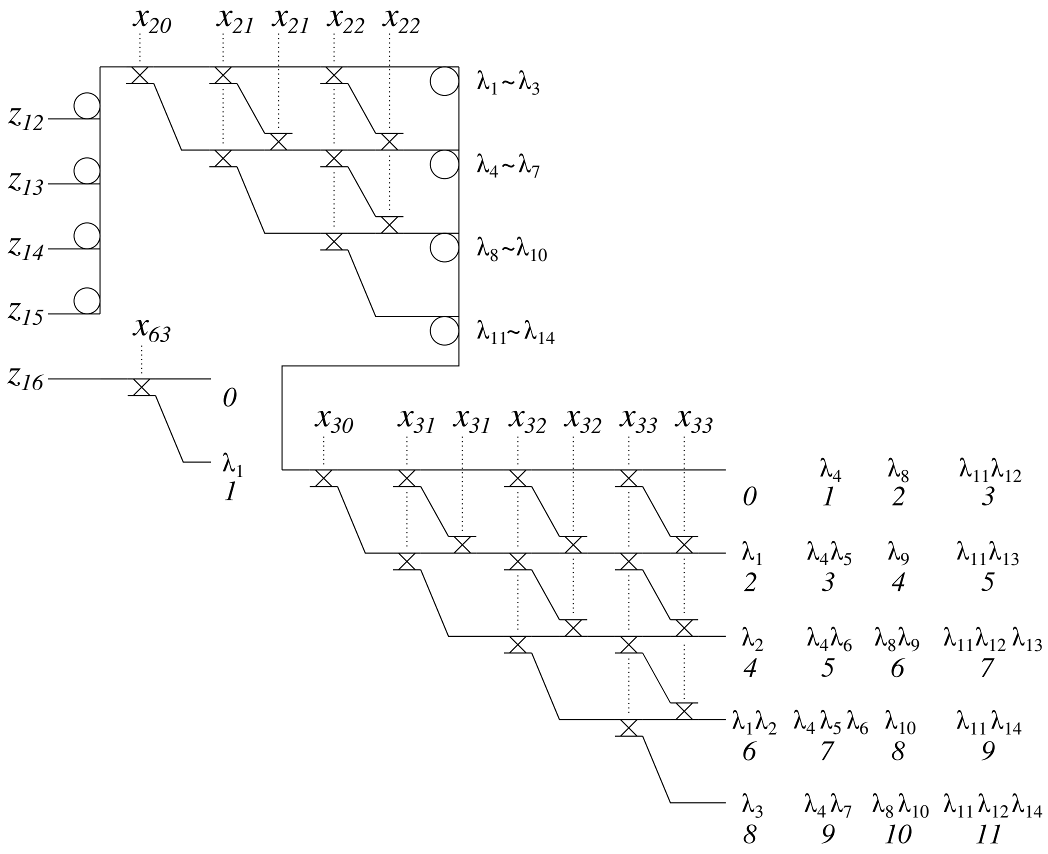

x_{20} x_{21} x_{21} x_{22} x_{22}
z_{12}
z_{13}
z_{14}
z_{15}
z_{16}
x_{63}
$\lambda_1 \sim \lambda_3$
$\lambda_4 \sim \lambda_7$
$\lambda_8 \sim \lambda_{10}$
$\lambda_{11} \sim \lambda_{14}$
0
λ_1 1
x_{30} x_{31} x_{31} x_{32} x_{32} x_{33} x_{33}
0 | λ_4 1 | λ_8 2 | $\lambda_{11}\lambda_{12}$ 3
λ_1 2 | $\lambda_4\lambda_5$ 3 | λ_9 4 | $\lambda_{11}\lambda_{13}$ 5
λ_2 4 | $\lambda_4\lambda_6$ 5 | $\lambda_8\lambda_9$ 6 | $\lambda_{11}\lambda_{12}\lambda_{13}$ 7
$\lambda_1\lambda_2$ 6 | $\lambda_4\lambda_5\lambda_6$ 7 | λ_{10} 8 | $\lambda_{11}\lambda_{14}$ 9
λ_3 8 | $\lambda_4\lambda_7$ 9 | $\lambda_8\lambda_{10}$ 10 | $\lambda_{11}\lambda_{12}\lambda_{14}$ 11

Figure 3.12: Circuit calculating the weighted sum for partial products of the 3rd and 4th digits and the 7th digit in parallel. (Copyright(C)2019 IEICE)

intra-function sharing. In Fig. 3.11, the circuit calculating the weighted sum for partial products of the 1st and 2nd digits and the circuit calculating the weighted sum for partial products of 5th and 6th digits are performed in parallel, since results of 1st and 2nd digits and those of 5th and 6th digits are independent from each other. The circuit shown in Fig. 3.12 is also parallelized, which reduces the number of serially connected DCs and, as a result, reduces the delay of the circuit and the power consumption of laser sources.

Fig. 3.13 shows the circuit calculating the weighted sum for partial products of 5th and 6th digits. Let us define the sub-circuit accumulating the partial products of 5th digit as a first-stage counter and the sub-circuit accumulating the partial products of 6th digit as a second-stage counter. The

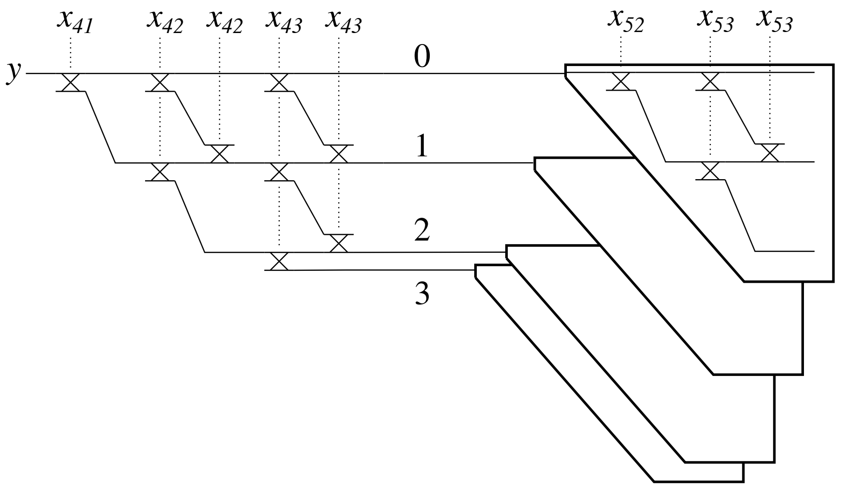


Figure 3.13: Circuit calculating the weighted sum for partial products of the 5th and 6th digits. (Copyright(C)2019 IEICE)

Table 3.2: 4-bit multiplier

	Power [mW]	Delay [ps]	# DCs
Ad-hoc	10.5	16	43
Minimum area	60.8	11	29
Minimum power	3.59	12	137
Shared-BDD	29.9	7	89

circuit already incorporates inter-function sharing, BDD reduction and splitter elimination. As can be seen from the figure, the circuit structures of the second-stage counters are same from each other. Therefore, in this circuit, the four second-stage counters can be merged into one using the intra-function sharing method described in Section 3 as shown in Fig. 3.14.

The synthesis results are summarized in Table 3.2. This results correspond to the results obtained using “optical logic circuit to be evaluated”, which is depicted in the middle of Fig. 3.10. Therefore, delay, power and the number of DCs of the adder circuit for calculating the final sum are not included.

For minimizing the number of DCs, inter-function sharing, BDD reduction and intra- function sharing are applied. The result is shown in the row

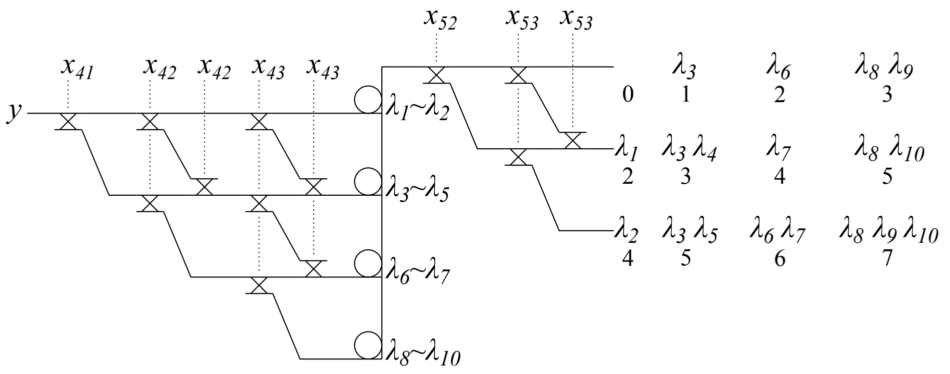

x_{41}
x_{42} x_{42} x_{43} x_{43}
x_{52} x_{53} x_{53}
y
λ_1~λ_2
λ_3~λ_5
λ_6~λ_7
λ_8~λ_{10}
0 λ_3 1 λ_6 2 λ_8 λ_9 3
λ_1 2 λ_3 λ_4 3 λ_7 4 λ_8 λ_{10} 5
λ_2 4 λ_3 λ_5 5 λ_6 λ_7 6 λ_8 λ_9 λ_{10} 7

Figure 3.14: Circuit incorporating intra-function sharing with WDM. (Copyright(C)2019 IEICE)

of "Minimum area". For minimizing the power, BDD reduction and splitter elimination are applied. The result is shown in the row of "Minimum power". Finally, with the ad-hoc approach which incorporates inter-function sharing, MTBDD reduction, splitter elimination and intra-function sharing in an ad-hoc manner, the number of DCs is reduced by 52% and the power dissipation is reduced by 64% compared to the results of shared-BDD although the delay is more than double. With the objective of minimizing the number of DCs, our methods reduce the number of DCs by 67% compared to the result of shared-BDD. With the objective of minimizing the power, the power dissipation in the laser source is reduced by 88% compared to the result of shared-BDD. Note that the primary goal of this thesis is providing new approaches to reduce the size and the power of optical circuits considering the tradeoffs among the area, delay and power, which help designers optimize the circuit according to their design goal and constraints.

3.4 Conclusion

The strategy of simply mapping a BDD to an optical circuit sometimes results in an explosion of its size and involves significant power losses in splitters and optical devices. To address these issues, this thesis proposed a size reduction method based on BDDs for optical logic circuits exploiting WDM. Since BDDs inherently have a large number of branches if their size is well reduced, optical circuits based on the BDDs need a lot of splitters. This causes significant power losses in optical circuits based on the BDD. The thesis also proposed a method for reducing the number of branches in a BDD-based optical circuit, which significantly reduces the power dissipation in laser sources. Experimental results obtained using a partial product accumulation circuit which is used in a 4-bit parallel multiplier demonstrated advantages of our method over existing approaches in terms of area and power consumption.

Chapter 4

BDD Variable Ordering for Minimizing Power Consumption

4.1 Introduction

4.1.1 Background and Our Contribution

As explained in the previous chapter, power consumption is a fundamental issue of BDD-based optical logic circuits. This chapter considers a method for reducing power consumption by optimizing a BDD itself. Here, we focus on the variable order of a BDD. Since the power consumption largely depends on the variable order, we can reduce the power consumption by variable ordering. However, existing BDD variable ordering aims only to reduce the number of nodes and does not consider power consumption. Little is known about the effect of variable ordering on the power consumption of BDD-based optical logic circuits, and to date, there is no efficient variable ordering algorithm for minimizing the power consumption. Therefore, in this thesis, we address these challenges and propose a method for finding the variable order that minimizes power consumption. Since the design methods proposed in [35, 36] and chapter 3 do not consider a variable order, these methods are expected to reduce the power consumption more by incorporating the method proposed in this thesis. Our contributions are summarized below.

- We clarify the variable order that minimizes the power consumption and the variable order that minimizes the number of nodes are quite different, and demonstrate that selecting the former rather than the latter drastically reduces the power consumption while keeping the number of nodes comparable.

- We demonstrate that an optimization problem for finding the variable order that minimizes the power consumption has a huge time complexity. To address this issue, we drastically reduce the execution time by adopting the idea, proposed in [55], of exploiting an adjacent variable swap algorithm and an algorithm of generating all $n!$ permutationa by making just $n! - 1$ adjacent interchanges. Experimental results demonstrate that this strategy enables us to find the optimal variable order in a reasonable execution time for 10-input logic functions.

4.1.2 BDD Variable Ordering

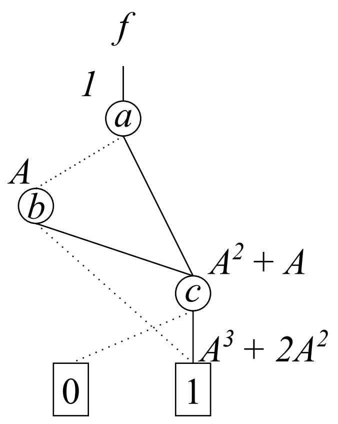
A well-known characteristic of BDDs is that the number of nodes largely depends on the variable order. Unfortunately, the problem for finding the variable order that minimizes the number of nodes for arbitrary n-input logic functions is NP-hard [58]. One simple algorithm builds a BDD based on the canonical sum of products (SOP) form and calculates the number of nodes for all $n!$ variable orders. The time complexity of this algorithm is obviously greater than $O(n!)$. To address this issue, [53,54] proposed reducing the time complexity to $O(n^2 3^n)$ by utilizing a Dynamic Programming (DP) technique.

4.2 Effect of Variable Ordering on Power Consumption

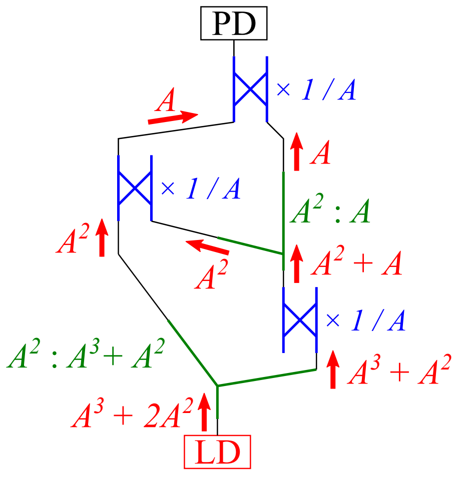
First, we propose a method of calculating the power consumption and the optimal splitting ratio by focusing on the insertion loss, which we define as the total amount of signal power lost on the path from the root to node v. The insertion loss $loss_v$ is formulated by

$$loss_v = A \times \sum loss_{parent}. \tag{4.1}$$

where A is the value for compensating for the signal power attenuation at DCs. We assume the signal power attenuation is -1 dB, so we must have



(a) BDD.



(b) Optical circuit.

Figure 4.1: Example of calculating power consumption. (© 2021 IEEE)

$A = 1.25$. $loss_{parent}$ is defined as the insertion loss of the parent nodes of node v. Therefore, Eq. (5.1) means that $loss_v$ can be obtained by multiplying A by the sum of the insertion loss of parent nodes. However, when the splitting ratio optimization, proposed in [36], is not applied, the insertion loss is larger than $loss_v$. Given the two parent nodes x and y of node v, setting the splitting ratio $SR_x : SR_y$ to $loss_x : loss_y$ minimizes the signal power loss at the splitter, where $SR_x(SR_y)$ denotes the ratio of signal power split into node $x(y)$. Similarly, given the three parent nodes x, y and z, when $SR_x : SR_y : SR_z = loss_x : loss_y : loss_z$, which minimizes the signal-power loss at the splitter. The same is true when the number of parent nodes is larger than 3. As an example, take the splitter connected to the DC labeled c in Fig. 4.1. In this case, we define node a as the node labeled by a and the same definition is used for nodes b and c. We then obtain the optimal splitting ratio $SR_a : SR_b = 1 : A$ from $loss_a = 1$ and $loss_b = A$. Next, we explain how to calculate the power consumption and the optimal splitting ratio of all splitters by calculating Eq. (5.1) for all nodes from the root to the 1-terminal node. Consider the example of Fig. 4.1. First, $loss_a$ is obviously 1. Second, $loss_b$ is obtained by multiplying A by $loss_a$ since the signal power is attenuated by $1/A$ through the DC labeled by a. Third, since node c has the two parent nodes a and b, $loss_c = A \times (loss_a + loss_b) = A^2 + A$. Then

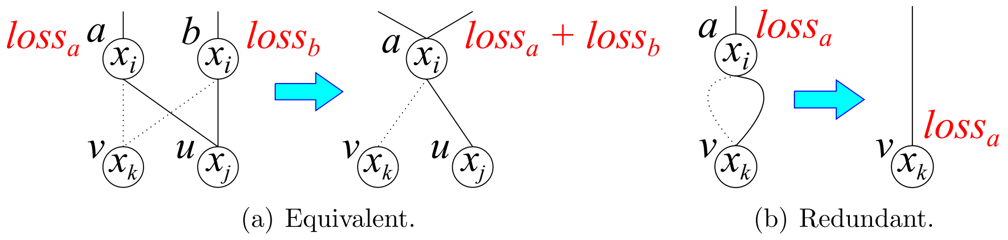


Figure 4.2: BDD reduction rule. (© 2021 IEEE)

we have the optimal splitting ratio $SR_a : SR_b = 1 : A$. Finally, we have $loss_t = A \times (loss_b + loss_c) = A^3 + 2A^2$, where t denotes the 1-terminal node, and $SR_b : SR_c = A : A^2 + A$. Fig. 4.1 (b) shows the optical circuit based on the optimal splitting ratio and the power consumption calculated above. Red arrows represent light propagation, blue letters represent signal attenuation at a DC, and green letters represent the splitting ratio. We can see here that sufficient signal power can be obtained at a PD, no matter which path an input light passes through.

Equation (5.1) helps us to understand the effect of variable ordering on the power consumption of BDD-based optical logic circuits. First, we explain the equivalent node and the redundant node since these nodes are related to the power consumption. As shown in Fig. 4.2 (a), node a and b are defined as equivalent nodes when 0-child of node a and 0-child of node b are the same, 1-child of node a and 1-child of node b are the same. As shown in Fig. 4.2 (b), a node is defined as a redundant node when its 0-child and 1-child are the same. The number of equivalent nodes and redundant nodes depends on a variable order. When a BDD has a larger number of equivalent nodes and redundant nodes, the number of nodes can be reduced, as these nodes can be eliminated by the BDD reduction rule. Therefore, a variable ordering for minimizing the number of nodes aims to select the order that has a larger number of equivalent nodes and redundant nodes. On the other hand, equivalent nodes and redundant nodes have different effects on the power consumption. We explain this nature using Eq. (5.1) and Fig. 4.2. First, consider the case when equivalent nodes are shared in Fig. 4.2 (a). Originally, both $loss_v$ and $loss_u$ equal $A \times (loss_a + loss_b)$. When node b is merged into node a, the insertion loss of node b is added to the insertion loss of node a, which makes the insertion loss of node a $loss_a + loss_b$. Here, node v and u have only node a as the parent node and their insertion loss is $A \times (loss_a + loss_b)$. Therefore, sharing equivalent nodes does not alter the power consumption. On the other hand, when we eliminate a redundant node as shown in Fig. 4.2 (b), the insertion loss of node v is reduced from $2A \times loss_a$ to $loss_a$. This means that eliminating a redundant node reduces the power consumption. Therefore, in terms of the power consumption, we do not need to consider the number of equivalent nodes but rather should select a variable order having a larger number of redundant nodes. From the above explanation, we can see that sometimes the variable order with the minimum number of nodes and the variable order with the minimum power consumption are quite different. Therefore, existing variable orderings for minimizing the number of nodes are not sufficient for optimizing the variable order in terms of the power consumption. It is thus essential to clarify the characteristic of an optimization problem for finding the variable order that minimizes the power consumption and build an efficient algorithm for this optimization problem.

4.3 Optimization Algorithm

4.3.1 Difficulty of Variable Ordering for Power Minimization

First we explain why we can utilize the DP technique for variable ordering for minimizing the number of nodes. As shown in Fig. 4.3, a N-input BDD is divided into an upper part and a lower part with level k as a boundary. Since changing the variable order of an upper part dose not change the number of nodes in a lower part and vice versa, we can consider them separately. The number of nodes in an entire BDD is the sum of the number of nodes in an upper part and a lower part. Therefore, the variable order that minimizes the number of nodes in a lower part obviously leads to minimize the number of nodes in an entire BDD. That is, the optimization problem for finding the optimal variable order of a lower part is the sub-problem for finding the optimal variable order of an entire BDD, which make it possible to utilize the DP technique for variable ordering for minimizing the number of nodes.

Let us demonstrate why the DP technique cannot be used for variable ordering for minimizing the power consumption by using Fig. 4.3. By focusing on nodes v_i defined as a node belongs to a lower part and has a edge

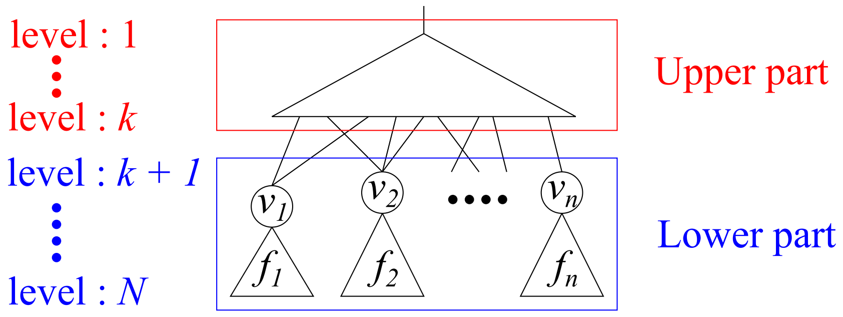


Figure 4.3: Concept of DP technique for a BDD. (© 2021 IEEE)

connected to a node in an upper part, the power consumption is represented by

$$10\ \mu\text{W} \times \sum (loss_{root \to v_i} \times loss_{v_i \to terminal}). \tag{4.2}$$

where $loss_{root \to v_i}$ is the insertion loss from the root to a node v_i and $loss_{root \to v_i}$ is the insertion loss from a node v_i to the 1-terminal node. Since any path from the root to the 1-terminal node passes through one of nodes v_i, Eq. (4.2) considers all paths from the root to the 1-terminal node and exactly expresses the power consumption. The insertion loss $loss_{root \to v_i}$ depends on a variable order of an upper part. On the other hand, the insertion loss $loss_{v_i \to terminal}$ depends on a variable order of a lower part. We can see that the variable order of a lower part that leads to minimize the power consumption change according to the variable of an upper part, since the insertion loss from the root to the 1-terminal node is multiplication of $loss_{root \to v_i}$ and $loss_{v_i \to terminal}$. That is, we cannot divide the optimization problem of finding the optimal variable order of an entire BDD into the optimization problems for an upper part and a lower part. Therefore, we cannot use the DP technique for variable ordering to minimize the power consumption. We need to examine the $n!$ variable order, which causes a huge execution time. To address this issue, we focus on a fast reordering algorithm called adjacent variable swap.

4.3.2 Adjacent Variable Swap

First, we explain a simple method of building a BDD based on a canonical SOP form. [47] proposed a fast algorithm that takes BDDs representing logic functions f_1 and f_2 and, a binary operator $\langle op \rangle$ (e.g., AND and OR) and then produces a reduced BDD representing the function $f_1\langle op\rangle f_2$. When the number of nodes of two BDDs are $|G_1|$ and $|G_2|$ respectively, the time complexity of this algorithm is $O(|G_1|\cdot|G_2|)$. Any BDD can be built by using this algorithm based on the canonical SOP form. The number of operations for BDDs increases as the number of products and literals in each product increases. Take the function $f = abcd + abce + \bar{a}\bar{b}c + b\bar{c} + \bar{d}\bar{e}$ as an example. The number of products is 5 and the number of literals in each product is 4, 4, 3, 2, and 2 in order from the left. We need 4 times OR operations and $3 + 3 + 2 + 1 + 1 = 10$ times AND operations in order to build a BDD corresponding to function f. Therefore, when the number of inputs of the target logic function is large, it increases both the number of BDD nodes and the operations for BDDs, resulting in a huge execution time.

Let us explain the fast reordering algorithm based on adjacent variable swap proposed in [56] by using Fig. 4.4. We can swap adjacent variables x_i and x_j in time proportional to the number of nodes labeled by x_j, without affecting, or even accessing, the nodes below x_i or above x_j, as long as we know which node, above x_j, has an edge connecting to x_i or x_j. Let f be the function corresponding to a node labeled by x_j. If we express f using the Shannon expansion, as

$$f = x_j f_1 + \bar{x}_j f_0 = x_j(x_i f_{11} + \bar{x}_i f_{10}) + \bar{x}_j(x_i f_{01} + \bar{x}_i f_{00}).$$

swapping x_j and x_i corresponds to rearranging these terms as:

$$f = x_i(x_j f_{11} + \bar{x}_j f_{01}) + \bar{x}_i(x_j f_{10} + \bar{x}_j f_{00}) = x_i j_1 + \bar{x}_i j_0.$$

Thus, the node encoding f must be relabeled and overwritten with new content: its variable change from x_j to x_i, and its children change from

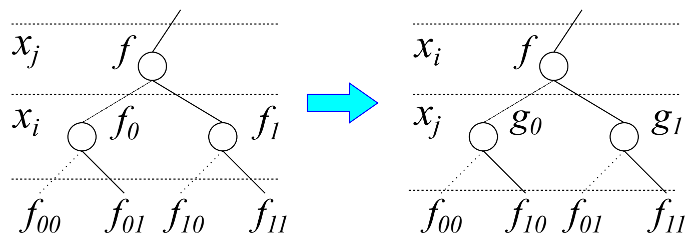


Figure 4.4: Adjacent variable swap. (© 2021 IEEE)

the nodes encoding f_0 and f_1 to the nodes encoding g_0 and g_1. The nodes encoding f_{11}, f_{10}, f_{01} and f_{00} are reused without changing them.

Considering the time complexity of the above two algorithms, the execution time of reordering based on adjacent variable swap is very fast compared to that of building a BDD based on a canonical SOP form. Therefore, we attempt to reduce the execution time for finding the variable order that minimizes the power consumption by utilizing adjacent variable swap.

4.3.3 Calculating Power Consumption

In order to find the variable order that minimizes the power consumption, we need to calculate the power consumption for $n!$ BDDs having different variable orders. Therefore, we propose an algorithm based on the method of calculating the power consumption and the optimal splitting ratio proposed in section 7.2.

Our algorithms are expressed in a pseudocode. Each BDD node is represented by a record declared as follows:

```
type node{
low, high: node;
level: 1...$N+1$
val: $(0, 1, X)$
id: integer
loss: real number;
parent: array; };
```

Field	Terminal	Nonterminal
low	null	$low(v)$
high	null	$high(v)$
level	$N+1$	$level(v)$
val	$value(v)$	X

Both nonterminal and terminal nodes are represented by the same type, but the field values for a node v depend on the node type as given in the table above. Let N be the number of input variables of a BDD. The id field contains an integer identifier that is unique to node v in the BDD. The low(high) field specifies the child node connected by 0-edge(1-edge). The level field shows which variable node v is labeled by. The loss field stores the value corresponding to the insertion loss of v formulated by Eq. (5.1). The parent field is an array that stores the id of the parent node.

Algorithm 1 shows the procedure for calculating the power consumption. The time complexity is proportional to the number of BDD nodes. First, nodes are collected into lists according to their levels. This can be done by Depth-First Search (DFS) from the root. The time complexity is proportional to the number of nodes. Then we process these lists from the one containing

Algorithm 1 Calculating power consumption

```
1: procedure POWER_CALCULATION
2:     push all nodes in $nlist[i][j]$
3:     for $level$ := 1 upto $N$ do
4:         for each $u$ on $nlist[level]$ do
5:             add $u.loss \times A$ to $u.low.loss$
6:             add $u.loss \times A$ to $u.high.loss$
7:             push $u.id$ to $u.low.parent$ and $u.high.parent$;
8:             examine $u.low.level$ and $u.high.level$
9:         end for
10:    end for
11:    process 1-terminal node $t$ on $nlist[N + 1]$
12:    power consumption := $t.loss \times 10\mu$W
13: end procedure
```

the root up to the one containing the N-level nodes. For each node u on the list, we add $u.loss \times A$ to both $u.low.loss$ and $u.high.loss$. Also, we push $u.id$ to $u.low.parent$ and $u.high.parent$. This information is needed to calculate the optimal splitting ratio. Then, we examine $u.low.level$ and $u.high.level$. When $u.low.level$ or $u.high.level$ matches the levels to be processed by adjacent variable swap, information of $u.id$ is needed by the adjacent variable swap algorithm. Finally, we process the 1-terminal node t. The power consumption of the BDD-based optical logic circuit is $t.loss \times 10$ μW.

4.3.4 Generating All Variable Orders

A simple algorithm makes it possible to generate all $n!$ permutation by making just $n! - 1$ adjacent interchanges [77]. Thanks to this algorithm, we can build BDDs with $n!$ variable orders by $n! - 1$ adjacent variable swap, as summarized below.

1. Build a BDD based on a canonical SOP form.
2. Calculate the power consumption by Algorithm 1.
3. Determine which variables to swap by the algorithm introduced in [77].
4. Apply the adjacent variable swapping algorithm.

5. Calculate the power consumption by Algorithm 1.

6. Iterate from steps(3) to (5) $n! - 1$ times.

The time complexity of our algorithm is $n! \times |G|$, where $|G|$ is the number of BDD nodes. Examination of the $n!$ variable order is inevitable for minimizing the power consumption. However, our algorithm is considerably faster than a simple method of building a BDD based on a canonical SOP form thanks to the adoption of the adjacent variable swap.

4.4 Experimental Results

In this section, we demonstrate how our algorithm resolves the issues discussed above. In our evaluation of the performance of optical logic circuits, we assume that the power attenuation in a DC is -1 dB [69]. The minimum optical power that the PD can detect is assumed to be 10 μW. For BDD-based optical logic circuits, we apply splitting ratio optimization to the splitters to reduce their insertion loss.

The proposed algorithm is applied to logic functions obtained by applying an LUT technology mapper provided by ABC [78] to an ISCAS'85 C7552 benchmark circuit. This circuit-transformation method described in chapter 6 can reduce the power consumption of large circuits such as ISCAS'85 benchmark circuits by several orders of magnitude. Since the maximum number of LUT inputs in this technology mapper is 10, several logic functions with 1–10 input variables are obtained. The synthesis results are summarized in Table 6.1 separately according to the number of input variables. The numerical values in the table are the average value of those synthesis results. The time values represent the execution time of our algorithm. As we can see, our algorithm can obtain the optimal variable order in a reasonable execution time even when the number of inputs of a target function is 10. For a 10-input function, the simple algorithm for building 10! BDDs having different variable orders based on the canonical SOP form results in an execution time of just a few hours. As one of our goals is to build a fast variable ordering algorithm, these results demonstrate that the strategy of exploiting adjacent variable swap works well and our goal is achieved.

Let us discuss our second goal, which is reducing the power consumption by variable ordering. In Tabel 6.1, the No. of nodes values represent the number of nodes of BDDs and corresponds to the number of DCs of

Table 4.1: Experimental results

10-input function						
time = 63.1 [s]						
			Min. node			
	Avg.	Freq.	Min.	Max.	Avg.	Optimal
No. of node	38.2	34.5	15.7			16.7
Power [mW]	5.97	2.65	1.92	2.76	2.17	1.84

9-input function						
time = 4.24 [s]						
			Min. Node			
	Ave.	Freq.	Min.	Max.	Ave.	Optimal
No. of node	25.3	17.1	12.0			12.2
Power [mW]	2.27	0.857	0.710	0.944	0.779	0.666

8-input function						
time = 0.419 [s]						
			Min. Node			
	Ave.	Freq.	Min.	Max.	Ave.	Optimal
No. of node	20.7	17.1	10.5			10.8
Power [mW]	1.42	0.803	0.672	0.952	0.788	0.653

7-input function						
time = 0.0368 [s]						
			Min. Node			
	Ave.	Freq.	Min.	Max.	Ave.	Optimal
No. of node	18.2	15.2	9.76			9.76
Power [mW]	0.591	0.346	0.324	0.405	0.358	0.324

optical circuits. The power values represent the power consumption of optical logic circuits. The Avg. column corresponds to the average value of the results of $n!$ BDDs having different variable orders, where n represents the number of inputs of the target logic function. The Freq. column corresponds to the specification of the circuit that applies variable ordering that assigns the variables from the top level in descending order of appearance frequency in the canonical SOP form. This variable ordering is well-known as a good heuristic for reducing the number of nodes. Moreover, it is also effective in terms of the power consumption, since a BDD that applies this heuristic is likely to have many redundant nodes, which reduces the power consumption. The No. of nodes values in the Min. node column represent the minimum number of nodes among the results of all variable orders. There are several variable orders that minimize the number of nodes, and each has a different power consumption. In Table 6.1, the power values in the Min. column, the Max. column, and the Avg. column respectively represent the minimum power consumption, the maximum power consumption, and the average power consumption among the results of the minimum number of nodes. The power values in the Optimal column represent the minimum power consumption among the results of all variable orders. There are also several variable orders that minimize the power consumption, and each has a different number of nodes. The No. of nodes values in the Optimal column represent the minimum number of nodes among the results of the minimum power consumption. In our algorithm, we consider the variable order that has the number of nodes and the power consumption summarized in the Optimal column as the optimal one. When the number of inputs of the target function is 7, the Min. values equal the Optimal values. However, when the number of inputs of the target function is larger than 7, the Min. values do not equal the Optimal values, which demonstrates the theory, discussed in Section 7.2, that the variable order minimizing the number of nodes and the variable order minimizing the power consumption are quite different hold for general logic functions. Taking a look at the synthesis results of the 10-input functions, we can see that our method reduces the power consumption by 31% and the number of nodes by 52% compared with the Freq. results. It also increases the number of nodes by 6% compared with the Min. node results. Unfortunately, the power consumption is reduced by only 4% compared with the Min. results. However, our method reduces the power consumption by 15% compared with the Avg. results and by 33% compared with the Max. results. Existing variable orderings aim to reduce the number of nodes but do not consider the power consumption. Therefore, considering the Max. results and the Avg. results, we can conclude that our method reasonably reduces the power consumption compared with existing variable ordering techniques while keeping the number of nodes comparable.

4.5 Conclusion

We clarified that optimizing the variable order drastically reduced the power consumption of BDD-based optical logic circuits and the optimization problem of finding the variable order that minimizes the power consumption had a large time complexity. Then we proposed the variable ordering algorithm exploiting adjacent variable swap to resolve the issue of the time complexity. Experimental results demonstrated that our variable ordering algorithm drastically reduced the power consumption without increasing the number of nodes compared to the results of the minimum number of nodes. Since the methods proposed in [35, 36] are based on a BDD with a variable order that reduces the number of nodes, these methods are expected to reduce the power consumption more by incorporating our method. Moreover, our algorithm found the optimal variable order in a reasonable execution time for the 10-input function. However, the strategy of finding the variable order that minimizes the power consumption was not practical when the number of inputs of the target function is larger than 10.

Chapter 5

Dual Port Node Sharing

5.1 Introduction

5.1.1 Background and Our Contribution

Continuing from the previous chapter, we study a method for reducing power loss at a splitter. We propose a method of eliminating splitters by focusing on a node that satisfies a certain condition. We call this node a *dual port node (DP node)*, and we call our method for eliminating a splitter based on DP nodes as *DP node sharing*. DP node sharing eliminates splitters by sharing DCs corresponding to DP nodes, which results in a significant power reduction and a small reduction in space without increasing delay. We also propose a synthesis method for BDD-based circuits that incorporate DP node sharing. The execution time of the proposed synthesis method is comparable to that of a BDD-based synthesis method that does not incorporate DP node sharing. Finally, we conducted an experiment involving 10-input logic functions obtained by applying an LUT technology mapper to an ISCAS'85 C7552 benchmark circuit [79] to evaluate our DP node sharing. We examine all variable orders by the mothod proposed in chapter 4 to fully exploit DP node sharing. Although splitter elimination method proposed in chapter 3 can be applied simultaneously with DP node sharing, we apply only DP node sharing to the target function to evaluate DP node sharing in this chapter. The experimental results indicate that DP node sharing reduces power consumption exponentially and reduces circuit size slightly, i.e., the number of DCs.

5.1.2 Power Consumption Model

A method for calculating the optimal splitting ratio and power consumption of an entire circuit is described in chapter 4. Since we describe the effect of DP node sharing by using this calculation method, this section briefly explains this calculation method again. This method focuses on an insertion loss defined as the total amount of signal power lost on the path from the root to node v. This insertion loss $loss_v$ is formulated by

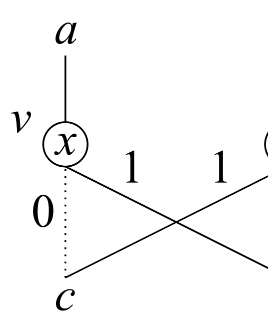
$$loss_v = A \times \sum loss_{parent}. \tag{5.1}$$

where A is the value for compensating for the signal-power attenuation at DCs. We assume the signal-power attenuation is -1 dB, so we must have $A = 1.25$. The $loss_{parent}$ denotes the insertion loss of the parent nodes of node v.

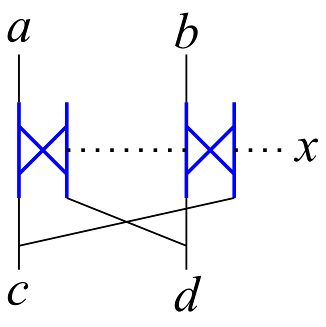
5.2 DP Node Sharing

5.2.1 Concept and Effect

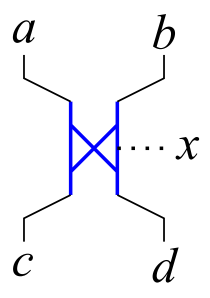
In this section, we describe DP node sharing then discuss its effect on power consumption and number of nodes. Given two nodes v and u labeled with the same variable, we define the two nodes as DP nodes, if the two nodes satisfy the following two conditions: (1) 0-child of v and 1-child of u are the same and (2) 1-child of v and 0-child of u are the same. In Fig. 5.1 (a), node v and u are a pair of DP nodes. Fig. 5.1 (b) shows the conventional BDD-based circuit corresponding to the pair of DP nodes v and u. This circuit has two DCs and two splitters. The insertion loss of node c is equal to $A \times (loss_a + loss_b)$ from Eq .5.1. The same is true for node d. Although it is inevitable that two splitters exist in the part of the circuit corresponding to DP nodes, we can eliminate the two splitters by using a dual port that is not used. Take a look at the left DC. The upper left port and the upper right port are dual. In other words, when the light from node $c(d)$ reaches the upper left port, the light from node $d(c)$ reaches the upper right port. As shown in Fig. 5.1 (c), we can implement the circuit by a single DC by connecting the right upper port with node b. When $x = 0$, the light from node $c(d)$ reaches node $a(b)$. When $x = 1$, the light from node $c(d)$ reaches node $b(a)$. It is obvious that this operation is logically correct from Fig. 5.1 (a). In every DC, two output



(a) DP nodes.



(b) Conventional.



(c) Share.

Figure 5.1: Concept of DP node Sharing.

ports are dual as well as two input ports. Therefore, we can use the circuit shown in Fig. 5.1 (c) for implementation of DP nodes. We define this method as DP node sharing. Note that DP node sharing does not increase the delay of the circuit since there is no extra DC inserted into a path from the LD to PD. Let us consider the insertion loss of node c. The light from node c passes through one DC and reaches node a or b in accordance with the input variable x. No matter which node the light reaches, the signal power has to be larger than the insertion loss of the node that the light reaches. Therefore, the insertion loss of node c is equal to $A \times (\max(loss_a, loss_b))$. The same is true for node d. Since the insertion loss of the circuit with DP node sharing is obviously smaller than that of a conventional BDD-based circuit, DP node sharing can reduce power consumption. Consider the case of $loss_a = loss_b$. In the conventional BDD-based circuit, $loss_c = 2A \times loss_a$. In the circuit incorporating DP node sharing, however, $loss_c = A \times loss_a$. Therefore, the power consumption is reduced by half for every pair of DP nodes, resulting in exponentially smaller power consumption of circuits incorporating DP node sharing compared with that of conventional BDD-based circuits. DP node sharing can also reduce the number of DCs. The number of DCs in Fig. 5.1 (c) is one less than in Fig. 5.1 (b), which means that every pair of DP nodes reduces the number of DCs by one. However, the reduction in the number of DCs has a smaller impact than power consumption. Accordingly, DP node sharing is suitable for reducing power consumption rather than the number of DCs. In summary, DP node sharing can reduce both power consumption and number of DCs without increasing delay. Therefore, a circuit is optimized by applying DP node sharing to all DP nodes.

5.2.2 Best Case Example

When we implement a function based on a conventional BDD-based optical logic circuit, even if the function can be expressed by a BDD having a small number of nodes, the power consumption varies greatly for each function. Consider examples of AND and XOR. Figs. **??** (b) and 5.2 show the circuit for AND and XOR, respectively. The number of nodes of AND and XOR is n and $2n-1$, respectively, where n is the number of input variables. Accordingly, the number of DCs of AND and XOR is n and $2n-1$, respectively. Let us consider the power consumption based on an insertion loss. In the circuit for AND, the power consumption is obviously A^n from the calculation method mentioned in Sec 6.1.2. Consider DCs labeled x_{i+1} and DCs labeled x_i in Fig 5.2. Since there is a splitter between the DCs, $loss_{x_{i+1}} = A \times loss_{x_i} + A \times loss_{x_i} = 2A \times loss_{x_i}$ $(i > 1)$. We must obtain the power consumption of $A^n \times 2^{n-1}$ by calculating an insertion loss from the root node to 1-terminal node. In the circuit for XOR, the increase in the number of DCs is linear, however, that in power consumption is exponential. From the above discussion, the power consumption of the circuit for XOR is exponentially larger than that of AND. However, DP node sharing can drastically reduce the power consumption of XOR. Fig. 5.3 shows the circuit for XOR incorporating DP node sharing. Consider DCs labeled x_{i+1} and DCs labeled x_i. Since there is no splitter between the DCs, $loss_{x_{i+1}} = A \times loss_{x_i}$, which results in smaller power consumption of A^n. The number of DCs in an entire circuit is n since the number of DCs labeled x_i is 1. In other words, DP node sharing exponentially reduces the power consumption of XOR to that of AND while reducing the number of DCs without increasing delay. In summary, logic functions with huge power consumption, such as XOR, are barriers to practical use of BDD-based optical logic circuits. However, when splitters cause huge power consumption, DP node sharing can sometimes drastically reduce power consumption, which resolves the issue regarding the practicality of BDD-based optical logic circuits.

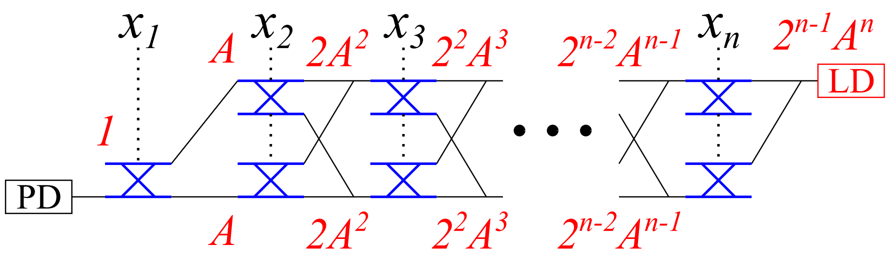


Figure 5.2: Conventional BDD-based Circuit.

5.3 Logic Synthesis

In this section, we introduce our proposed synthesis method for BDD-based optical logic circuits incorporating DP node sharing. The flow of our synthesis method is as follows. First, we transform a target logic function into a BDD then design the circuit on the basis of this BDD. When designing the circuit, our synthesis method optimizes connections between DCs and the splitting ratios of splitters.

5.3.1 BDD Optimization

A BDD incorporating *input inverters* [59] is suitable for the proposed synthesis method. An input inverter is one of the attribute edges, which has a different operation from a complemented edge. Fig. 5.4 (b) shows a BDD incorporating input inverters corresponding to a BDD shown in Fig. 5.4 (a). Input inverters indicate an operation of swapping a 0-edge and 1-edge at the next node. This operation is regarded as complementing an input variable. The abuse of input inverters breaks a property that each subgraph uniquely represents a function. However, the uniqueness is maintained under the constraint that the id of a 0-child is smaller than that of a 1-child. Due to

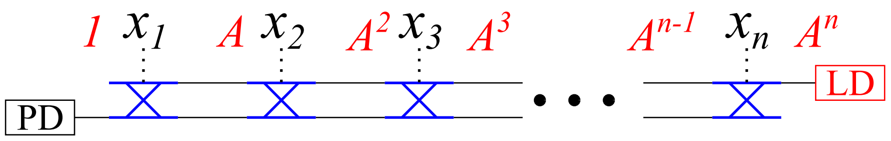
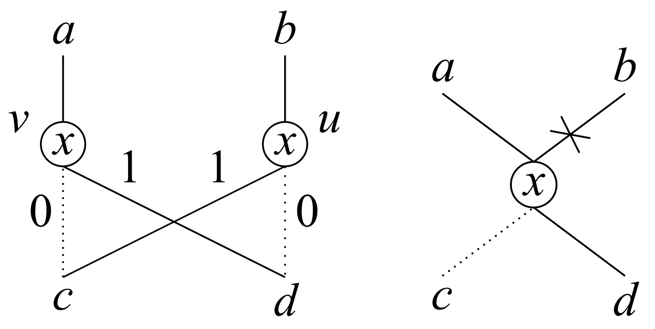


Figure 5.3: BDD-based Cicuit Incorporating DP Node Sharing.



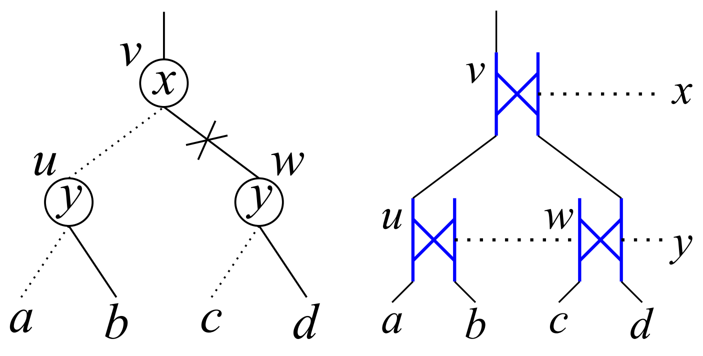
(a) BDD.　(b) Input inv.

Figure 5.4: Concept of Input Inverter.

this constraint, even if a node does not have a DP node, sometimes that node is connected to its parent node with an input inverter. In a standard BDD, we check whether a certain node can be removed by using a unique id for each node. Similarly, we can check whether a certain node can be connected to its parent node with an input inverter by examining the id of its child nodes. Therefore, implementation of input inverters does not incur additional computational cost compared with the standard BDD. There is a one-to-one correspondence between DP nodes and input inverters, which is clear from the definitions of DP nodes and input inverters as well as Figs. 5.4 and 5.1 (a). In a BDD fully incorporating input inverters, every pair of DP nodes is merged into one node. As explained in Section 7.2, circuits incorporating DP node sharing are optimized by applying DP node sharing to all DP nodes. Therefore, the proposed synthesis method applies input inverters to a BDD for the target logic function to fully exploit DP node sharing. Applying input inverters to a BDD makes it is easy to calculate the number of DCs from a BDD since the number of nodes in a BDD incorporating input inverters is equal to the number of DCs in a circuit incorporating DP node sharing.

5.3.2 Circuit Optimization

We now describe connections between DCs and optimization of splitting ratios. The other factors in circuit design are the same as the conventional BDD-based circuit design explained in Section 6.1.2. First, we explain how

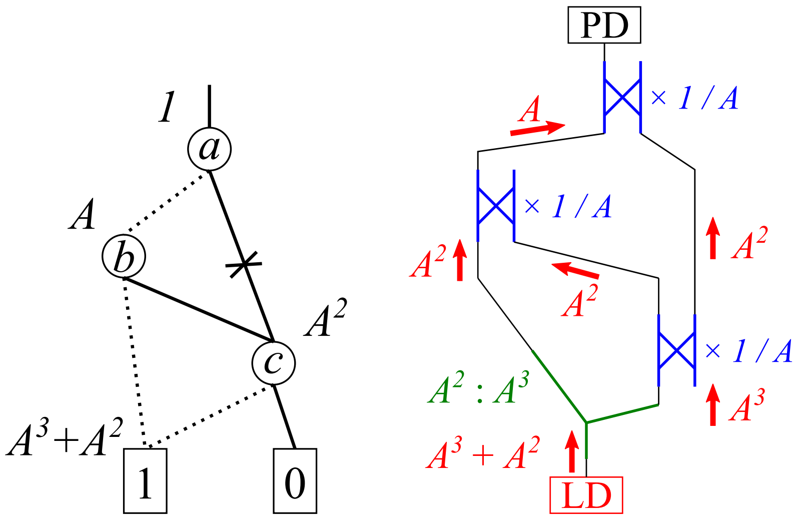


(a) BDD. (b) Circuit.

Figure 5.5: Implementation of Input Inverter.

to connect DCs corresponding to nodes connected with an input inverter. Fig. 5.5 (a) shows a part of a BDD incorporating input inverters. Note that even if a node does not have a DP node, sometimes that node is connected with an input inverter. Consider the operation of input inverters. At a node that is connected to the parent node with a normal edge, when the input variable is equal to 0(1), the 0(1)-child is selected. At a node that is connected to the parent node with an input inverter, when the input variable is equal to 0(1), the 1(0)-child is selected. We use a wiring method to implement this operation. Fig. 5.5 (b) shows the circuit implementing the BDD shown in Fig. 5.5 (a). At node $u(w)$, the lower left ports are connected to node $a(c)$ corresponding to the 0-child, and the lower right ports are connected to node $b(d)$ corresponding to the 1-child. We determine which upper port is connected to the parent node depending on whether the node is connected to the parent node with a normal edge or input inverter. Node u is connected to the parent node with a normal edge. In this case, the upper left port is connected to node v so that the light from node $a(b)$ can reach node v when $y = 0(1)$. However, node w is connected with an input inverter. In this case, the upper right port is connected to node v so that the light from node $d(c)$ can reach node v when $y = 0(1)$. We can see that this circuit is logically correct by assigning values to input variables x and y.

Next, we explain a method for calculating an optimal splitting ratio. Section 6.1.2 presented a method for calculating an optimal splitting ratio in a conventional BDD-based circuit based on an insertion loss. However, that calculation method does not incorporate DP node sharing. Therefore,



(a) BDD. (b) Optical circuit.

Figure 5.6: Example of Optimal Splitting Ratio with DP Node Sharing.

to address DP node sharing, we incorporate the method for calculating the insertion loss at a DP node, explained in Section 7.2, into the calculation method explained in Section 6.1.2. When we calculate the insertion loss of node v, the correct value can be obtained by just checking the insertion loss of the parent nodes regardless of whether node v is a DP node. Therefore, DP node sharing does not increase the execution time when calculating an optimal splitting ratio. In summary, the execution time of the proposed synthesis method incorporating DP node sharing is comparable to that of a conventional BDD-based synthesis method since the execution time of BDD optimization and circuit optimization does not incur additional computational cost. Let us show an example of calculating the optimal splitting ratio in a circuit incorporating DP node sharing. Fig. 5.6 (a) shows a BDD incorporating input inverters. For sake of simplicity, we define the node labeled by input variable a as node a and the same for nodes b and c. First, $loss_a$ is obviously 1. Second, $loss_b$ is obtained by multiplying A by $loss_a$. Third, since we can apply DP node sharing to node c, $loss_c = A \times max(loss_a + loss_b) = A^2$. Finally, we have $loss_t = A \times (loss_b + loss_c) = A^3 + A^2$, where t denotes 1-terminal node, and $SR_b : SR_c = A^3 : A^2$. Fig. 5.6 (b) shows the circuit incorporating DP node sharing based on the optimal splitting ratio and power consumption calculated above. Red arrows represent light propagation, blue letters represent signal attenuation at a DC, and green letters represent the splitting ratio. We can see here that sufficient signal power can be obtained at the PD, no matter which path an input light passes through.

5.4 Experiment and Results

In this section, we discuss the experiment we conducted on how our DP node sharing reduces power consumption and number of nodes. We applied DP node sharing to functions having the same number of inputs to remove the difference in the number of inputs from the experimental results. Therefore, we used the circuits obtained by applying an LUT technology mapper provided by ABC [78] to an ISCAS'85 C7552 benchmark circuit [79]. This circuit-transformation method described in chapter 6 can reduce the power consumption of large circuits such as ISCAS'85 benchmark circuits by several orders of magnitude. As the number of inputs of an LUT is 10, several logic functions with 1-10 input variables are obtained. We show the results of 10-input logic functions. In our evaluation of the performance of optical logic circuits, we assumed that the power attenuation in a DC is -1 dB [69]. The minimum optical signal-power that the PD can detect is assumed to be 10 μW. We compared the synthesis results of conventional BDD-based circuits and BDD-based circuits incorporating DP node sharing. Since the power consumption of a BDD-based optical circuit largely depends on the variable order, we examine all variable orders by the mothod proposed in chapter 4 to fully exploit DP node sharing.

Fig. 5.7 shows the results regarding power consumption when the BDDs have variable orders that minimize power consumption. "Conventional" represents the synthesis results of the conventional BDD-based circuits, and "Dual Share" represents those of the BDD-based circuits incorporating DP node sharing. The functions are labeled a number from 1 in ascending order of power consumption of "Conventional". "Function number" represents the number assigned to each function. From the results of "Conventional", we can see that the power consumption varied greatly from function to function in the conventional BDD-based circuits. The smallest power consumption was 93.1 μW, which is the smallest among all feasible 10-input functions. DP node sharing could not reduce the power consumption for this function since there was no splitter. Similarly, DP node sharing can not largely reduce

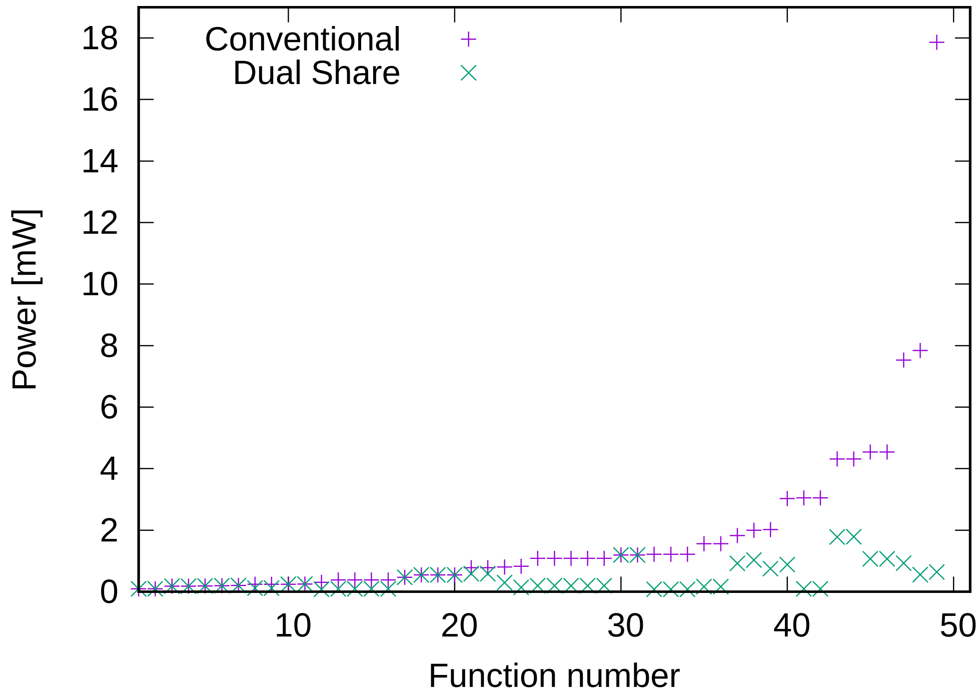


Figure 5.7: Results regarding Power Consumption.

the power consumption when circuits have a small number of splitters. However, conventional BDD-based circuits for such functions consume a small amount of power due to a small number of splitters. Consider the power consumption of function numbers 1-22. DP node sharing did not largely reduce the power consumption. However, since the power consumption of the conventional BDD-based circuits was small, these functions are not barriers to practical use of BDD-based optical logic circuits. Note that the primary goal of DP node sharing is to reduce the power consumption of functions for which conventional BDD-based circuit consumes a large amount of power. Consider the power consumption of function number 40-49. DP node sharing did significantly reduce the power consumption, which means that the cause of high power consumption in these functions is splitters corresponding to DP nodes. The highest power consumption was 17.9 mW, which was reduced to 643 μW with DP node sharing. In other words, DP node sharing, which

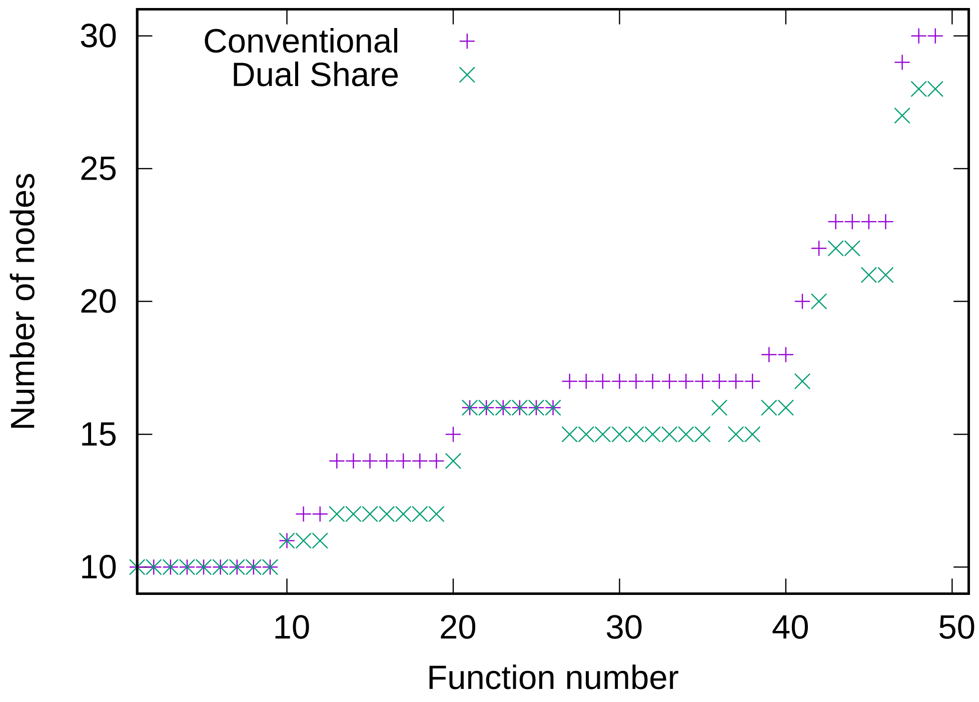


Figure 5.8: Results regarding number of Nodes.

does not require complicated implementation of software or hardware, can drastically reduce the power consumption of a general logic circuit such as ISCAS'85 C7552 benchmark circuit. Let us consider the results regarding the number of nodes. Note that the optimal variable order of "Dual Share" and "Conventional" can differ. Therefore, the number of nodes of "Dual Share" may be larger than that of "Conventional". However, this was the case for just five functions. In four functions, the number of nodes increased by just 1 or 2. In one function, the number of nodes increased by 12. Therefore, we also consider the number of nodes in optimization for variable ordering when a design constraint includes circuit size.

Next, we discuss the results when the BDDs have variable orders that minimize the number of nodes. In all functions, the power consumption of "Dual Share" is not larger than that of "Conventional". Fig. 5.8 shows the results regarding the number of nodes. The functions are labeled a number from 1 in ascending order of the number of nodes of "Conventional". "Function number" represents the number assigned to each function. We can see that reduction in the number of nodes was much smaller than that of power consumption, which is consistent with the discussion in Section 7.2. Therefore, DP node sharing is suitable for reducing power consumption.

5.5 Conclusion

We clarified that circuits having a large number of splitters, such as XOR, consume a large amount of power. To address this issue, we proposed DP node sharing as a method for eliminating splitters and that can be easily implemented in an optical circuit. We also proposed a synthesis method that does not incur additional computational cost compared with a conventional BDD-based synthesis method optimization. Experimental results obtained using ISCAS'85 C7552 benchmark circuit indicate that our DP node sharing drastically reduces the power consumption of optical logic circuits that consume a large amount of power. In other words, DP node sharing is key to breaking down the barrier to practical applications of BDD-based optical logic circuits.

Chapter 6

Multi-Stage Optimization for Optical Logic Circuits

6.1 Introduction

6.1.1 Background and Our Contribution

In chapters 3,4, and 5, we proposed the method for reducing the signal power loss at splitters to reduce the power consumption of optical logic circuits. However, the signal power attenuation at DCs is also a fundamental issue in terms of power consumption. Light signals from a laser source exponentially attenuate as they propagate through optical devices. As a result, the power dissipation of typical optical logic circuits is exponential to the number of inputs of the target logic functions, which may limit the scalability of optical logic circuits. To address this issue, the design method for signal restoration using Optical-to-Electrical (OE) converters is proposed [38]. Although this method largely reduces the signal power attenuation, the insertion of OE converters leads to an increase in delay since the OE conversion delay is considerably larger than the light propagation delay of DCs. This chapter proposes a synthesis method for power-efficient and high-speed optical logic circuits based on signal restoration using OE converters. The proposed synthesis method divides a target function into multiple sub-functions. As a result, the proposed synthesis method reduces power consumption to a polynomial order of the number of inputs of the target logic function. Moreover, our method can mitigate the OE converter delay overhead by parallelly executing the sub-functions. The proposed method thus exponentially reduces

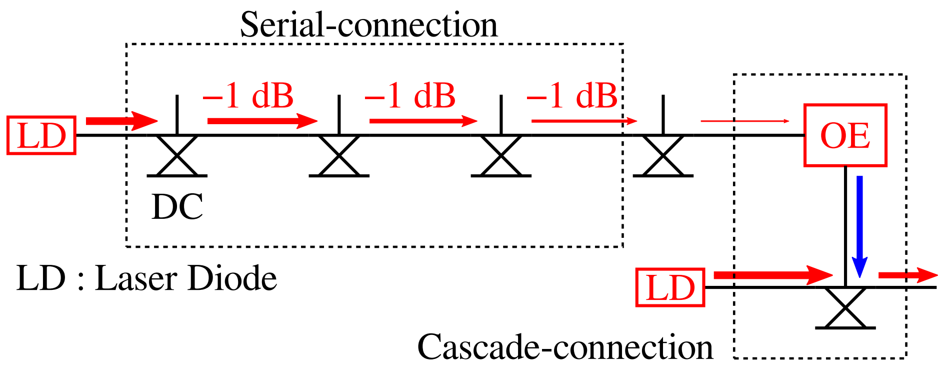


Figure 6.1: Serial-connection and cascade-connection. (Copyright(C)2021 IEICE)

power consumption without sacrificing the high-speed property of light.

6.1.2 Serial-Connection and Cascade-Connection

This thesis uses a DC as a basic building block. There are two ways to connect DCs: serial-connection and cascade-connection. In the serial-connction, once the input voltages are given, the latency is determined by the speed of the light passing through the serially connected DCs. In the cascade-connection, on the other hand, the OE conversion based on a PD is required in order to control the direction of the light passing through the DCs. The OE conversion delay is much larger than the light propagation time in the serial-connection. For example, the serial connection delay of the DC presented in [69] is 1 ps per DC, while the OE conversion delay is 25 ps [72]. In order to exploit the light speed, conventional design methods thus mainly use the serial-connection. However, one of the biggest issues in the serial-connection is a large insertion loss of the final output since the attenuation of the final output signal power is exponential to the number of DCs connected in series. If the intensity of the output optical signal is not sufficiently large, the signal detection time may increase and the circuit may result in failure of detection due to noise. This issue can be mitigated by inserting cascade-connection. Cascade-connection largely reduces the signal power attenuation since it repeats the light signal as shown in Fig. 6.1 [80].

Let us define an optical logic circuit consisting of serial-connection only as a single-stage optical logic circuit. Single-stage optical logic circuits have a

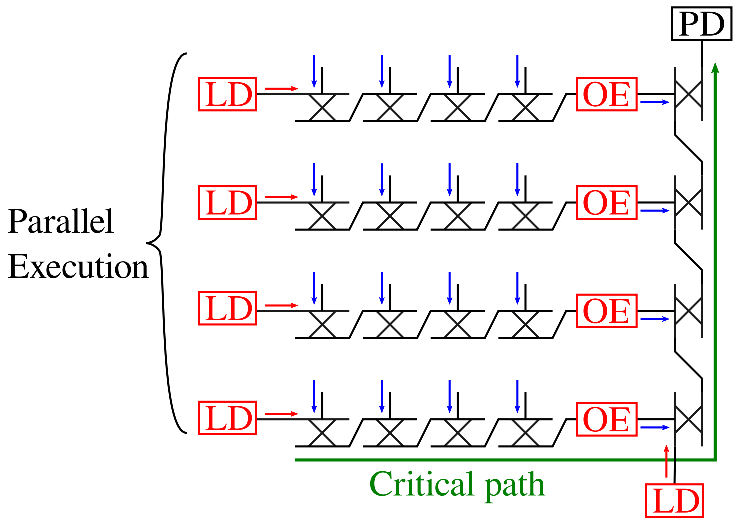
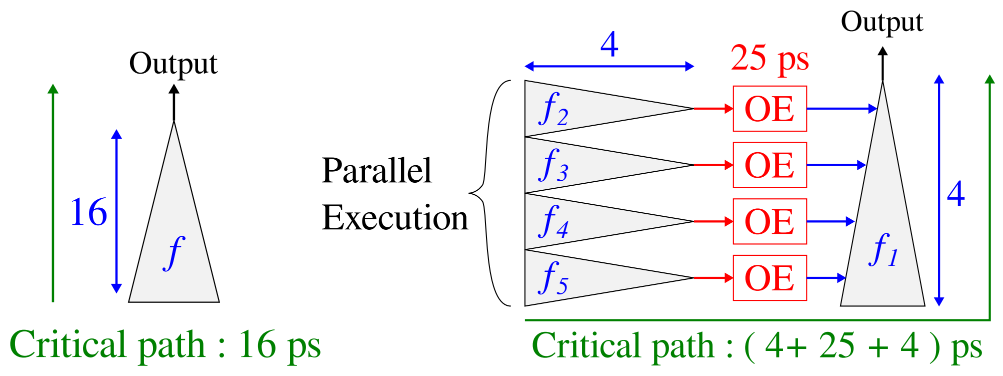


Figure 6.2: Multi-stage optical logic circuit (16-input AND operation). (Copyright(C)2021 IEICE)

high-speed characteristic but have a power-consuming characteristic. Optical logic circuits based on BDD are single-stage optical logic circuits. Although the speed of the BDD-based circuit is very fast as it is determined by the speed of the light passing through serially connected DCs, the power dissipation is exponential to the number of inputs (1 dB per DC in this thesis). The strategy of the node elimination based on BDD reduction rules can effectively reduce the number of DCs. However, it is hard to avoid the explosion of the power consumption.

The cascade-connection can reduce the power dissipation of the laser source. In the best case, the latency can also be improved [80]. Let us define an optical circuit which has cascade-connection on its critical path as a multi-stage optical logic circuit. Fig. 6.2 shows the multi-stage optical logic circuit for 16-input AND operation. In a single-stage optical logic circuit, the number of DCs connected in series is 16. On the other hand, in the circuit shown in Fig. 6.2, the number of DCs connected in series is 4, which exponentially reduces the power consumption. The power dissipated by a laser source in the single-stage optical logic circuit is 355 μW while the total power dissipated by the five laser sources in the multi-stage optical logic circuit is 97.7 μW. Note that the minimum detectable optical power at PDs is assumed to be 10 μW in this example. Moreover, since the OE



(a) Single-stage. (b) Multi-stage (This work).

Figure 6.3: Concept of BDD-based multi-stage optical logic circuit. (Copyright(C)2021 IEICE)

conversion delay can be mitigated by parallel execution of sub-circuits, in the best case, delay of a multi-stage optical logic circuit is also reduced compared to that of a single-stage optical logic circuit [80]. Synthesis methods for the optimization of multi-stage optical logic circuits are proposed in [80]. In this thesis, "multi-stage optimization" is defined as the optimization of multi-stage optical logic circuits. However, the target logics of the synthesis method proposed by [80] are limited to simple logic functions such as AND, OR, and XOR. To solve this problem, this thesis proposes a synthesis method of optical logic circuits for general and complex functions.

6.2 BDD-based Multi-Stage Optical Circuit

6.2.1 Concept

The concept of a BDD-based multi-stage optical logic circuit is shown in Fig. 6.3. Fig. 6.3 (a) shows a BDD-based single-stage optical logic circuit. The circuit corresponds to the function f with 16 inputs. In the BDD-based multi-stage optical logic circuit, as shown in Fig. 6.3 (b), the original 16-input BDD is divided into multiple 4-input BDDs in order to avoid an explosion of the power consumption since the power dissipation of optical circuits is exponential to the number of inputs. In each sub-circuit corresponding to the sub-functions f_1, f_2, f_3, f_4 and f_5, four DCs are serially connected. The

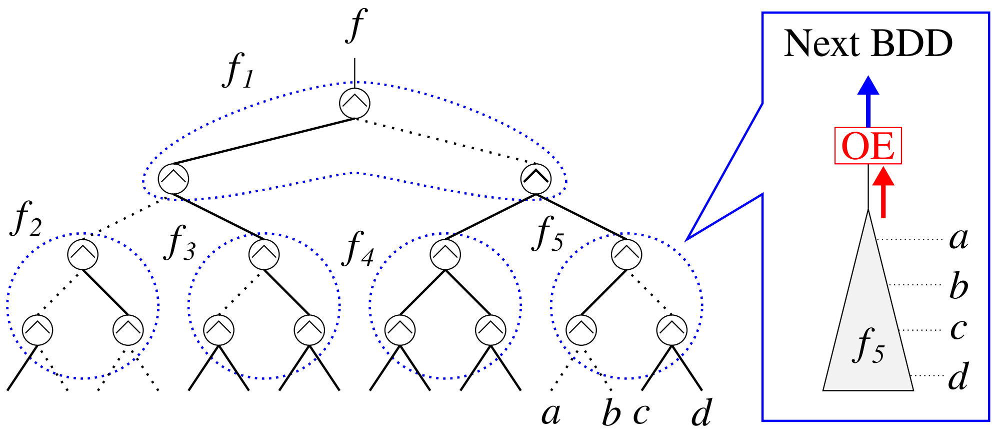


Figure 6.4: Example of division of AIG. (Copyright(C)2021 IEICE)

attenuation in the power of the light signals is largely reduced compared to a BDD-based single-stage optical logic circuit. The optical output signals of the divided BDDs (f_2, f_3, f_4, and f_5) are converted using OE converters. The converted electrical signals are then fed into the directional couplers in the successor BDD (f_1) as shown in Fig. 6.3 (b). The state-of-the-art integrated OE converter achieves a 25 ps conversion latency [72]. However, the 25 ps latency is still much larger than the light propagation time. This thesis assumes the directional coupler propagation delay is 1 ps per DC [69]. The proposed synthesis method can mitigate this OE converter delay overhead by parallelizing the divided BDDs as shown in Fig. 6.3 (b). In the best case, the propagation delay of the circuit based on the proposed design is smaller than that of the BDD-based single-stage circuit. For example, if the function f is a 64-input AND operation, the original 64-input BDD can be divided into multiple 8-input BDDs. The delay of the circuit based on the proposed design is $8 + 25 + 8 = 41$ ps, which is smaller than that of the 64-input BDD-based circuit.

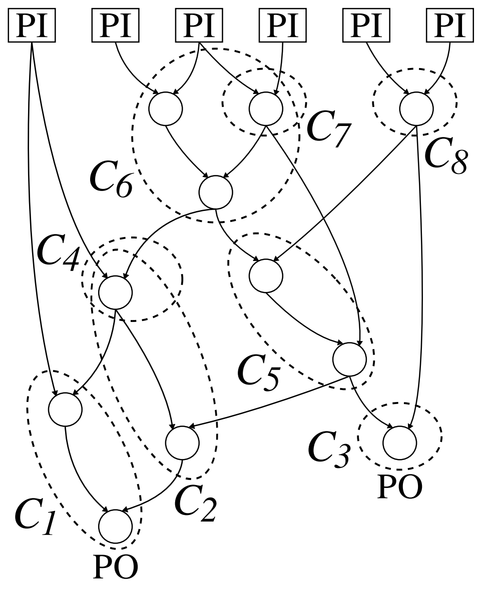
In the rest of this subsection, this thesis presents the overview of the synthesis method for multi-stage optical logic circuits. Initially, we apply a logic optimization based on And-Inverter Graph (AIG) to the target function f. Fig. 6.4 shows the AIG represents target function f. An AIG is expressed as a directed acyclic graph (DAG) in which each node except for leaves has two children and operates as AND function of the two children. The edges with a dotted line represent logical negation. Primary inputs are given to leaves. For example, f_5 shown in Fig. 6.4 represents $(\neg a \wedge \neg b) \wedge \neg(c \wedge d)$. Then, we divide the AIG into several sub-graphs corresponding to sub-functions f_1, f_2, f_3, f_4 and f_5 with blue dotted curves in Fig. 6.4. Finally, each sub-function is implemented by a BDD-based optical logic circuit. The optical output signal of a sub-circuit is fed into the directional coupler in the successor sub-circuit since the optical routing from inputs to outputs at a DC is controlled by the electrical signal. The BDD labeled as f_5 in Fig. 6.3 (b) is obtained by the above procedure. The rest of this section discusses a synthesis method for reducing the power consumption considering the tradeoff between the delay and the power consumption.

6.2.2 Synthesis Method

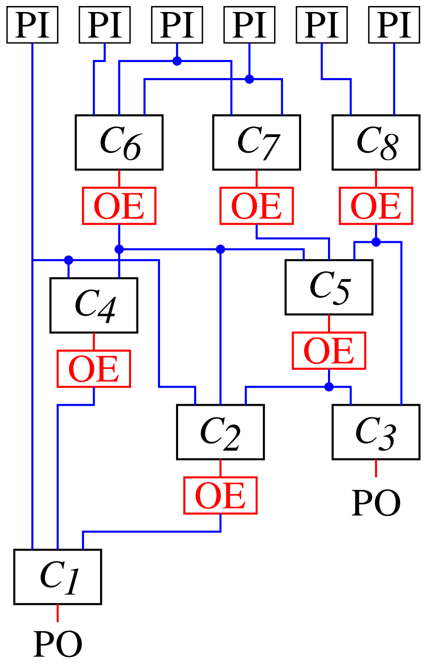
The synthesis flow is summarized below.

1. Apply an AIG-based logic optimization to the target function.
2. Divide the AIG to multiple sub-functions.
3. Implement a single-stage optical logic circuit to several sub-functions.

Firstly, this thesis discusses an optimization method for dividing the AIG. Let us define this optimization problem as the partition mapping problem. The partition mapping in our method is similar to a LookUp-Table-based (LUT-based) Field-Programmable Gate Array (FPGA) technology mapping. Accordingly, this thesis gives a precise problem formulation based on the problem formulation for a LUT-based FPGA technology mapping in [81]. AIG can be represented as a DAG called Boolean network where each node represents a logic element. A directed edge (i, j) exists if the output of logic element i is an input of logic element j. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. We use $input(v)$ to denote the set of nodes which are fanins of logic element v. Given a sub-graph H of the Boolean network, $input(H)$ denotes the set of distinct nodes outside H which supply inputs to the logic element in H. For a node v in the network, a *cone* at v, which is simply denoted by C_v, is a sub-graph, which consists of v and its predecessors, such that there exist paths connecting from any node in C_v to v. The *level* of a node v is the length of the longest path from any PI node to v. The level of a PI node is zero. The *depth* of a network is the largest node level in the network. In the Boolean network shown in Fig. 6.5 (a), the depth is 6.



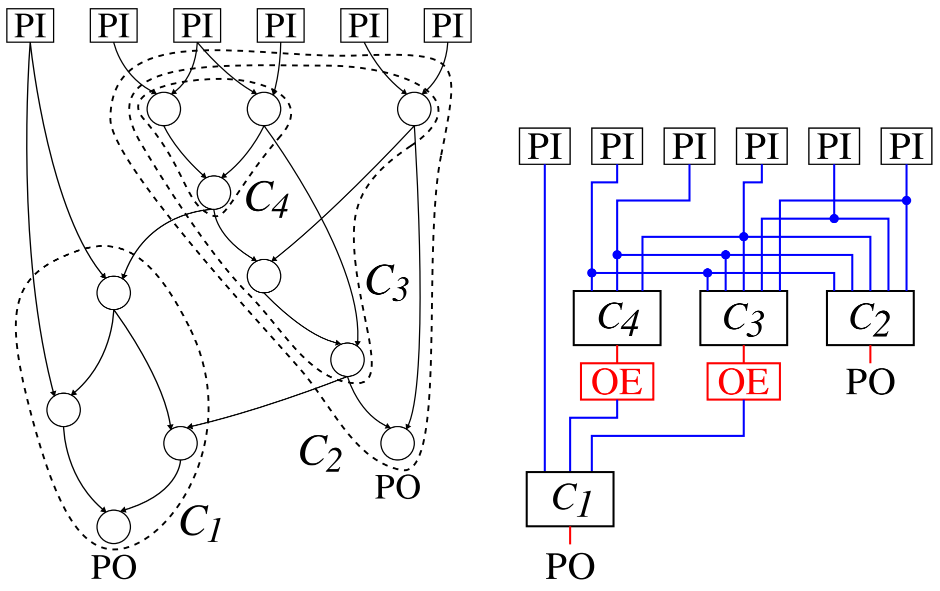
(a) Boolean network.



(b) Optical circuit.

Figure 6.5: Partition mapping solution and implementation by optical circuit. (Copyright(C)2021 IEICE)

In the proposed method, each sub-circuit in a multi-stage optical circuit is a BDD-based optical circuit that can operate as any Boolean function. Thus, any cone of a Boolean network can be implemented with single-stage BDD-based optical circuits. The partition mapping problem for multi-stage optical circuits is defined as the problem to cover a given Boolean network with cones. A partition mapping solution S is a DAG where each cone is replaced with a node. A directed edge (C_u, C_v) exists if there is a edge between the two cones. In Fig. 6.5 (a), dotted curves show the example of the mapping solution, where the depth is 4 and the number of cones is 8. Let us explain the procedure to generate a mapping solution. Let L be the set of nodes. A cone C_v $(v \in L)$ is to be mapped. Initially, L contains all the PO nodes. We process the nodes in L one by one. We remove a node v from L and map the C_v, then insert the nodes u $(\in \mathit{input}(C_v))$ to L. When a node u has been already inserted, we do not insert a node u to L. We iterate this



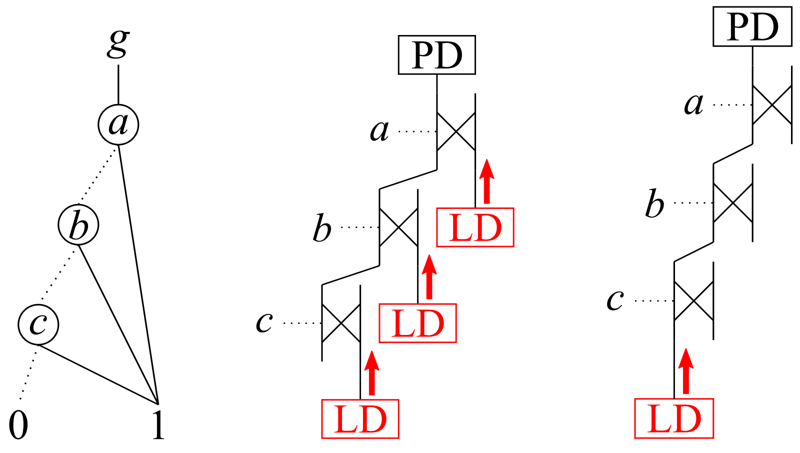
(a) Boolean network. (b) Optical circuit.

Figure 6.6: Partition mapping solution and implementation by optical circuit in which the constraint of the number of inputs of an LUT is relaxed. (Copyright(C)2021 IEICE)

manipulation. This procedure ends when L consists of only PI nodes of the original network.

Fig. 6.5 (b) shows the circuit based on the mapping solution shown in Fig. 6.5 (a). Each cone is implemented by a BDD-based optical circuit and each edge is implemented by a cascade-connection in BDD-based multi-stage optical logic circuits. The delay of a BDD-based multi-stage optical logic circuit is determined by two factors: the delay in sub-circuits and the delay in the interconnection paths (i.e., the OE conversion delay). Each BDD-based sub-circuit has a constant delay of n ps (where n is the number of inputs) which is independent of original target function. Each edge in the mapping solution corresponds to a constant delay of 25 ps since each edge is implemented with a cascade-connection. Note that this thesis assumes that the OE conversion delay is 25 ps [72]. In this case, the delay largely depends on the depth of the mapping solution. The power consumption of a BDD-based multi-stage optical logic circuit is determined by two factors: the number of sub-circuits and the power consumption in sub-circuits. Intuitively, a large number of sub-circuits cause the increase of the power consumption. Since the power consumption of a BDD-based optical circuit is exponential to the number of inputs, we limit the number of inputs of cones in the mapping solution in order to avoid the exponential power explosion. Therefore, the primary objective of our method is to find a mapping solution that satisfies the constraint of the number of inputs in a cone. The secondary objective is to reduce the depth and the number of cones in the mapping solution. A LUT-based FPGA technology mapping algorithm is suitable for this optimization since it is demonstrated that a LUT-based FPGA technology mapping algorithm has the same objectives [81, 82]. Therefore, we use the mapping algorithm in the synthesis flow. Synthesis results largely depend on a constraint of the number of inputs in a cone, which corresponds to the number of inputs of an LUT in FPGA technology mapping. The delay and the power consumption are reduced and increased as the number of inputs of an LUT increases, respectively. The mapping solution, as shown in Fig. 6.5 (a), satisfies the constraint that the number of inputs in a cone is 3. Fig. 6.6 (b) shows the circuit based on the mapping solution shown in Fig. 6.6 (a), in which the constraint is relaxed to 5. In the circuit shown in Fig. 6.6 (b), the number of OE conversions on the critical path and the number of sub-circuits are reduced compared to the circuit shown in Fig. 6.5 (b). When LUTs can have a larger number of inputs, the depth of the mapping solution may be reduced, which reduces the number of time-consuming OE conversions on the critical path. However, the number of inputs in each sub-circuits increases. This may lead to a significant increase in the power consumption since the power consumption of each sub-circuit is exponential to the number of its inputs. Therefore, it is very important to consider the tradeoff between the delay and the power consumption.

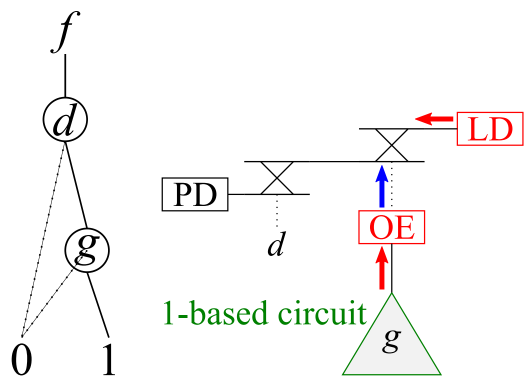
In the proposed method, the sub-graphs obtained by the above procedure are implemented by the BDD-based design method. In Section 6.1.2, we explained that the 1-terminal nodes corresponded to the optical source. However, in some cases, we can reduce the power consumption by using the 0-terminal nodes as the optical source. Consider the example of 3-input OR function $g = a \vee b \vee c$ shown in Fig 6.7 (a). The edges with a solid(dotted) line represent 1(0)-edge. Figs. 6.7 (b) and 6.7 (c) show the two circuit design in which an optical source is located at the leaf corresponding to the 1-terminal node and the 0-terminal node, respectively. Let us define these two circuit design as 1-based circuit design and 0-based circuit design, respectively. The



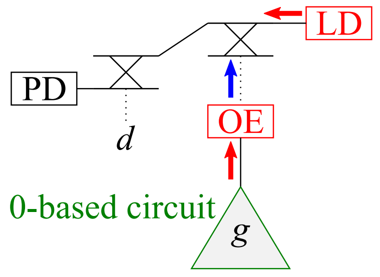
(a) BDD. (b) 1-based. (c) 0-based.

Figure 6.7: Implementation of BDD g. (Copyright(C)2021 IEICE)

1-based circuit design shown in Fig. 6.7 (b) has three paths from the output to the optical source. On the other hand, since the 0-based circuit design shown in Fig. 6.7 (c) has a single path from the output to the optical source, the power consumption of the 0-based circuit design is smaller than that of the 1-based circuit design. Therefore, we compare the power consumption of the 1-based circuit design with the 0-based circuit design, then adopt the circuit design that has smaller power consumption. When we mix these two circuit designs for implementing sub-graphs, the connection between a BDD and its subsequent BDDs should be modified. This is because the output optical signal is inverted if we use 0-based circuits. In this thesis, we propose the design method to address this issue. The DC has two optical inputs and two optical outputs. Let us define the optical input corresponding to a 1(0)-edge in a BDD as 1(0)-edge input, and the optical output on the side of 1(0)-edge input as 1(0)-edge output. Taking an example of the DC corresponding to the node labeled by b in Fig. 6.7 (b), the lower left port is a 0-edge input. The lower right port is a 1-edge input. The upper left port is a 0-edge output. The upper right port is a 1-edge output. Consider the example of the multi-stage BDD-based optical logic circuit for the logic function $f = d \wedge g$ ($g = a \vee b \vee c$) shown in Fig. 6.8 (a). When we adopt the 1-based circuit design, we connect the downside optical output (0-edge output) of the DC corresponding to the node labeled by g (DC labeled by



(a) BDD. (b) 1-based.



(c) 0-based.

Figure 6.8: Implementation of BDD f. (Copyright(C)2021 IEICE)

g in the following) to the DC corresponding to the node labeled by d (DC labeled by d in the following) in the next circuit as shown in Fig. 6.8 (b). When $a = 1, b = 0, c = 0, d = 1$ ($f = 1$), an electrical signal is given to the DC labed by g and the input light of the next circuit passes through that DC, then the light is detected at the PD. On the other hand, when we adopt the 0-based circuit design, we connect the upside optical output (1-edge output) of the DC labeled by g to the DC labeled by d as shown in Fig. 6.8 (c). When $a = 0, b = 0, c = 0, d = 1$ ($f = 0$), an electrical signal is given to the DC labeled by g and the input light of the next circuit does not pass through that DC, then the light is not detected at the PD. In summary, we connect the 0(1)-edge output of the DC controlled by 1(0)-based circuit design to the DC corresponding to the parent node in a next BDD-based circuit.

The proposed design method can reduce the power consumption while maintaining a correct output value of a multi-stage optical logic circuit.

In summary of this section, we show the synthesis flow in Fig. 6.9. Initially, we apply a logic optimization based on AIG to the target function. Then we divide the AIG into several sub-graphs using an LUT-based FPGA technology mapping algorithm. Next, we convert each sub-graph to a BDD. We compare the power consumption of the 1-based circuit design with the 0-based circuit design, then adopt the circuit design that has smaller power consumption. In oreder to obtain the tradeoff between the delay and the power consumption, we vary the number of inputs of an LUT in the mapping algorithm.

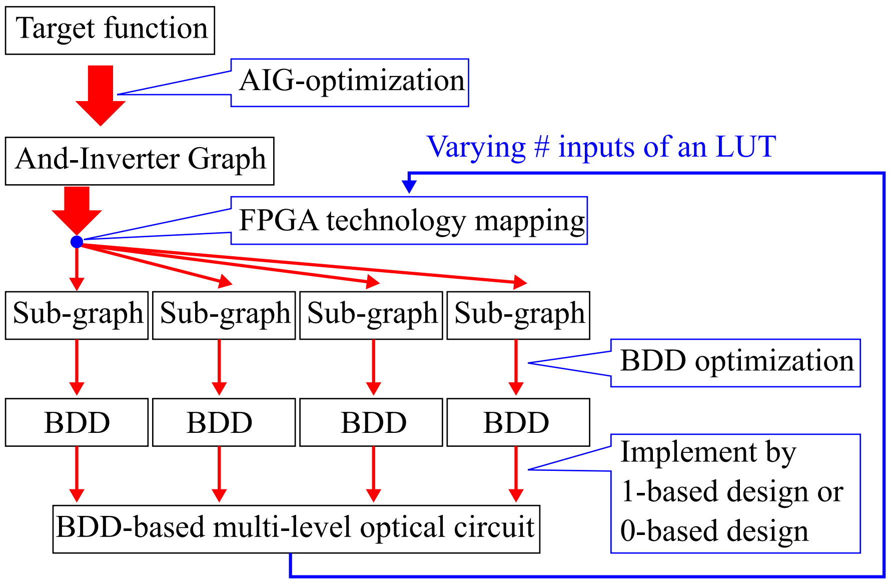


Figure 6.9: Synthesis flow. (Copyright(C)2021 IEICE)

6.3 Performance Evaluation Based on ISCAS'85 Benchmark

In this section, the proposed synthesis method is applied to the ISCAS'85 benchmark circuits [79]. In order to evaluate the performance of optical logic circuits, this thesis assumes that the propagation delay of a light passing through a DC is 1 ps [69]. The delay of the OE conversion is assumed to be 25 ps [72]. We assume that the power attenuation in a DC is -1 dB [69]. The minimum optical power which the PD can detect is assumed to be 10 μW. In this thesis, we assume that the energy overhead of the OE conversion is negligible compared to the power dissipation of the laser source.

6.3.1 Comparison between Single-stage Circuits with Multi-stage Circuits

In our synthesis method, we use the AIG-optimization and the LUT technology mapping algorithm provided by "ABC" [78]. In BDD-based multi-stage optical logic circuits, we utilize "CUDD" [83] by building BDDs corresponding to each sub-graph. The circuit specifications of ISCAS'85 benchmark are

Table 6.1: Specification of ISCAS'85 benchmark circuits.

Circuit	Function	PI / PO	Max input
c432	27-channel interrupt controller	36 / 7	36
c880	8-bit ALU	60 / 26	45
c1355	32-bit SEC circuit	41 / 32	41
c1908	16-bit SEC/DED circuit	33 / 25	33
c2670	12-bit ALU and controller	223 / 140	122
c3540	8-bit ALU	50 / 22	50
c5315	9-bit ALU	178 / 123	67
c6288	16×16 multiplier	32 / 32	32
c7552	32-bit adder/comparator	207 / 108	194

Table 6.2: Synthesis result of ISCAS'85 benchmark circuits (small set).

Circuit name	c432	c880	c1355	c1908	c2670
Power (Multi) [mW]	13.8	15.2	58.8	26.0	19.8
Delay (Multi) [ps]	145	107	74	135	103
Delay (Single) [ps]	36	45	41	33	122

summarized in Table 6.1. The PI/PO value is the total number of PIs/POs in the benchmark circuits. A target circuit has multiple POs. Each logic function corresponding to each PO is individually implemented by one optical logic circuit. For example, since c432 has 7 POs, 7 optical logic circuits are required to design c432. In Table 6.2 and 6.3, the "Delay (single)" values represent the delay of benchmark circuits based on single-stage BDD. The delay is less than or equal to (# PIs) $\times$ 1 ps since not all PIs are necessarily given to the optical circuits corresponding to each PO. For example, although c880 has 60 PIs, the delay is 45 ps. Benchmark circuits based on single-stage BDD have huge power consumption compared to CMOS logic circuits. For example, the power consumption of c6288 based on single-stage BDD is on the order of megawatts. Although the circuit size of c432 is small compared to other benchmark circuits, the power consumption is expected to be on the order of kilowatts.

The proposed synthesis method based on the LUT technology mapping algorithm is applied to the ISCAS'85 benchmark circuits, varying the number

Table 6.3: Synthesis result of ISCAS'85 benchmark circuits (large set).

Circuit name	c3540	c5315	c6288	c7552
Power (Multi) [mW]	84.3	43.1	79.6	68.4
Delay (Multi) [ps]	150	135	635	135
Delay (Single) [ps]	50	67	32	194

of inputs of an LUT from 2 to 10 in the partition mapping phase. The synthesis results are summarized in Table 6.2 and 6.3. The specification of the circuits based on the proposed synthesis method corresponds to the first row and the second row in Tabel 6.2 and 6.3. The "Power (multi)" values represent the total power dissipated by the laser sources in the multi-stage optical logic circuits. In this thesis, we compare a 1-based circuit design and a 0-based circuit design based on a BDD built by "CUDD" for implementing each sub-graph. An ordering algorithm provided by "CUDD" makes each BDD compact, which reduces the power consumption of BDD-based optical circuits. The "Delay (multi)" values are estimated by total delay of sub-circuits and the number of OE conversions, on the critical path. The delay value shown in Table 6.2 and 6.3 is the smallest among synthesis results under a power constraint of a 100 mW. The proposed synthesis method reduces the power consumption to under 100 mW. The delay of the circuits synthesized by the proposed method is kept less than about four times the delay of the single-stage circuits except for c6288. Especially, in the circuits for c2670 and c7552, our method reduces the delay and power simultaneously since the strategy of parallelizing the sub-circuits works well. In the circuit for c6288, although the proposed method increases the delay by 20 times, it reduces the power consumption by seven orders of magnitude. Note that the primary goal of this thesis is providing new approaches to improve the power consumption of optical logic circuits to the level of practical values without sacrificing the ultra-high-speed characteristic. If we simply design c6288 based on single-stage optical logic circuits, the power consumption is on the order of megawatts which is not an acceptable value for practical use. The proposed method thus can resolve the scalability issue in terms of the power explosion.

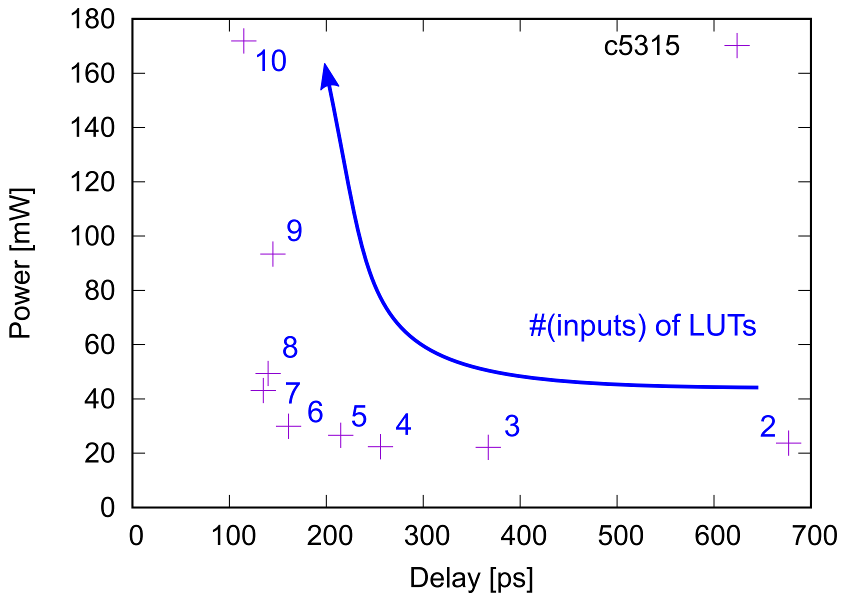


Figure 6.10: Characteristics of delay and power consumption (c5315). (Copyright(C)2021 IEICE)

6.3.2 Tradeoff between the Power Consumption and the Delay

Fig. 6.10 and Table. 6.4 show synthesis results obtained by varying the number of inputs of an LUT from 2 to 10 in the partition mapping phase. The values of partition mapping solutions are also summarized in Table. 6.4. "Max # inputs" represents the maximum number of inputs of an LUT in the partition mapping phase. "Average # inputs" represents the average number of inputs of a cone in the mapping solution. Fig. 6.10 and Table. 6.4 show the synthesis results for c5315. Experiments for other benchmark circuits also show a similar characteristic in terms of the delay and the power consumption. It is observed that the delay is reduced and the power consumption is exponentially increased as the number of inputs of an LUT in the partition mapping phase increases. Note that the proposed method can reduce the power consumption of optical circuits considering the tradeoff between the delay and the power consumption, which helps designers optimize the circuit depending on their design goals and constraints.

Table 6.4: Partition mapping solution (c5315).

Max # input	Depth	# cones	Average # inputs
2	26	1389	1.98
3	14	768	2.66
4	9	501	3.29
5	8	362	3.95
6	6	304	4.32
7	5	273	4.89
8	5	237	5.06
9	5	236	5.56
10	4	228	5.89

6.3.3 Polynomial Relationship between the Power Consumption and the Number of Inputs

We demonstrate that our method reduces the power consumption to a polynomial order of the number of inputs of the target logic functions. It is hard to see the relationship between the number of inputs and the power consumption from the number of PIs and the power consumption of benchmark circuits in Table 6.1. This is because the logic complexity and the number of POs are considerably different between benchmark circuits. Therefore, we focus on one benchmark circuit. Logic functions corresponding to each PO in one benchmark circuit are separately extracted. We consider the relationship between the number of PIs and the power consumption in each PO. For this experiment, we use the c7552 benchmark circuit since c7552 has a variety number of PIs associated with each PO compared with other benchmark circuits. Figs. 6.11, 6.12, and 6.13 show the results obtained by mapping 3-input LUTs to c7552 and the results obtained by mapping 10-input LUTs to c7552, respectively. Note that the proposed optimization method is allowed to use smaller LUTs than 3-/10-input LUTs. The smaller LUTs are typically utilized for functions which are not on a critical path. Experiments for other benchmark circuits also show a similar characteristic. Fig. 6.11 shows characteristics of the power consumption and the number of PIs associated with each PO in a log-log graph.

The power consumption is a polynomial of the number of PIs in both two cases since they are on a straight line in a log-log plot. Fig. 6.12 shows

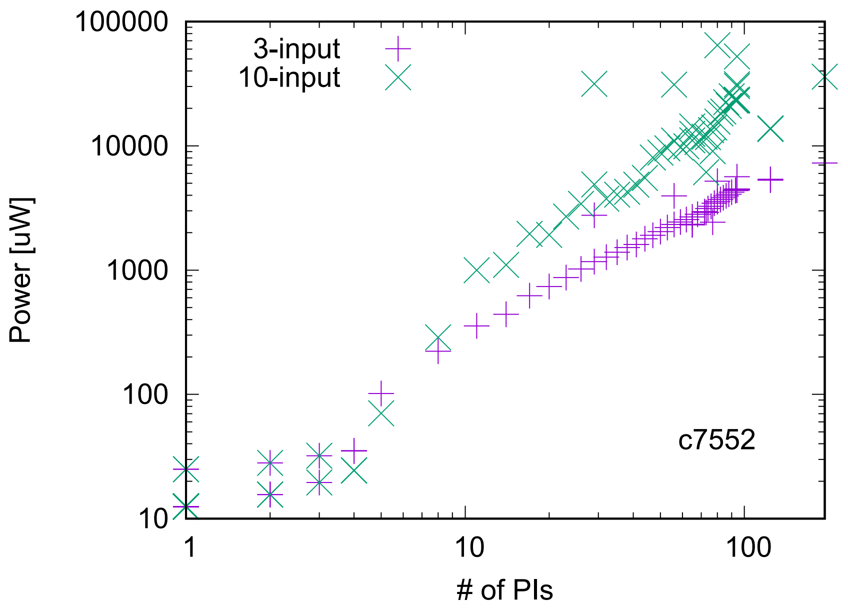


Figure 6.11: Characteristics of power consumption and # PIs. (Copyright(C)2021 IEICE)

characteristics of the number of sub-circuits and the number of PIs given to each PO in a log-log graph. Fig. 6.13 shows characteristics of the average number of inputs of sub-circuits and the number of PIs given to each PO in a semi-log graph. First, we consider the results obtained by mapping 3-input LUTs. The two graphs shown in Figs. 6.11 and 6.12 have almost the same slope. It is observed that the average number of inputs of sub-circuits does not depend on the number of PIs from Fig. 6.13. Since the power consumption of BDD-based optical circuits is dominated by the number of inputs, the average power consumption of sub-circuits also does not depend on to the number of PIs. Therefore, the power consumption of multi-stage optical logic circuits increases with the same slope in a log-log plot as the number of sub-circuits increases. On the other hand, in the results obtained by mapping 10-input LUTs, the average number of inputs of sub-circuits increases as the number of PIs increases from Fig. 6.13. This is because the ABC uses small LUTs for sub-circuits on non-critical paths. This makes the slope in Fig. 6.11 steeper than the 3-input LUT design, since power-hungry large sub-circuits are utilized more than small sub-circuits. The above

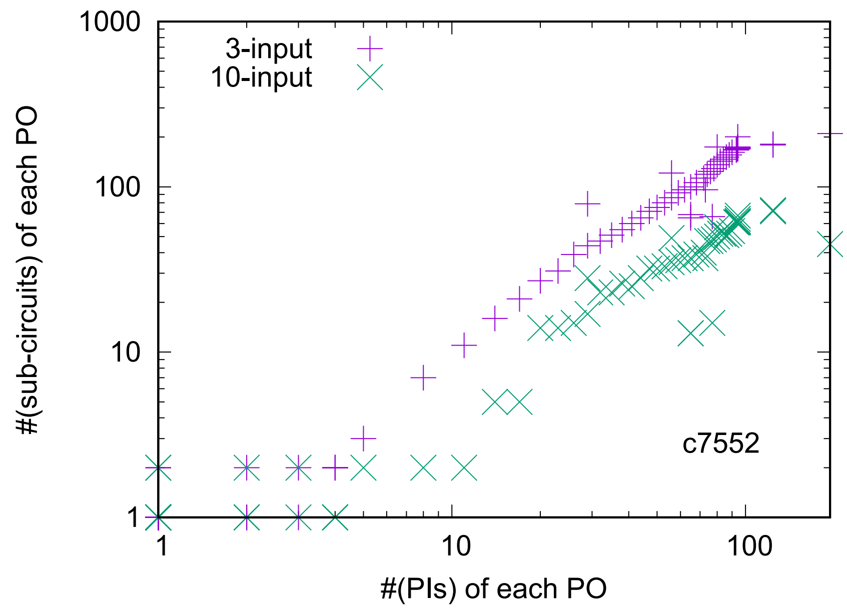


Figure 6.12: Characteristics of # sub-circuits and # PIs. (Copyright(C)2021 IEICE)

discussion shows that the power consumption of logic functions is reduced to the polynomial order of the number of their inputs.

6.4 Conclusion

Existing synthesis methods for optical logic circuits result in an explosion of the power consumption. To address this issue, this thesis proposed the power consumption reduction method based on multi-stage optimization using OE converters while exploiting the high-speed nature of optical logic circuits. The proposed synthesis method can mitigate the OE converter delay overhead by parallelizing sub-circuits. Experimental results obtained using ISCAS'85 benchmark circuits demonstrated that our method reduces the power consumption of optical logic circuits to a polynomial order of the number of inputs of the target logic function. without sacrificing the ultra-high-speed characteristic. It also offers designers the tradeoff between the delay and the power consumption. Our future work will be focused on de-

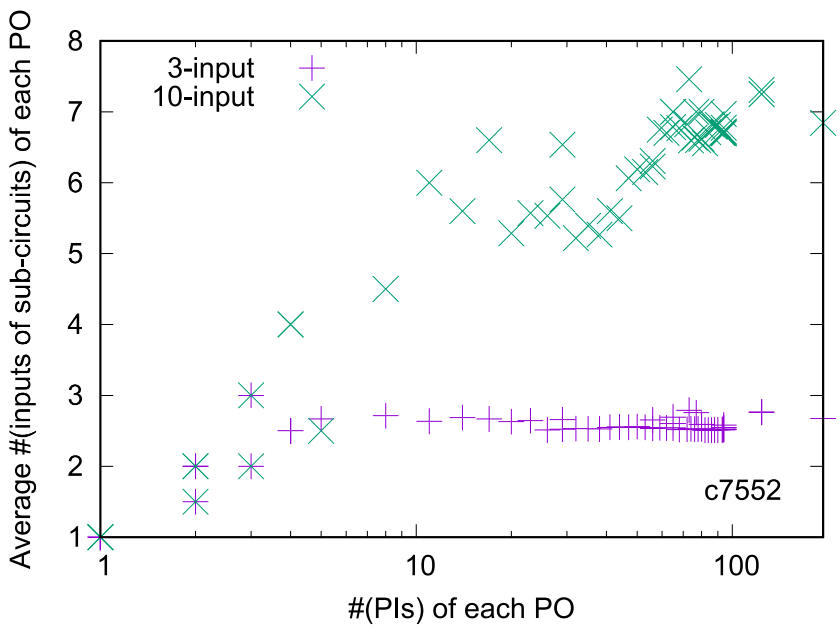


Figure 6.13: Characteristics of average # inputs and # PIs. (Copyright(C)2021 IEICE)

veloping algorithms which optimize the optical circuit based on their design goals and constraints.

Chapter 7

Optical Logic Circuits without Garbage Outputs

7.1 Introduction

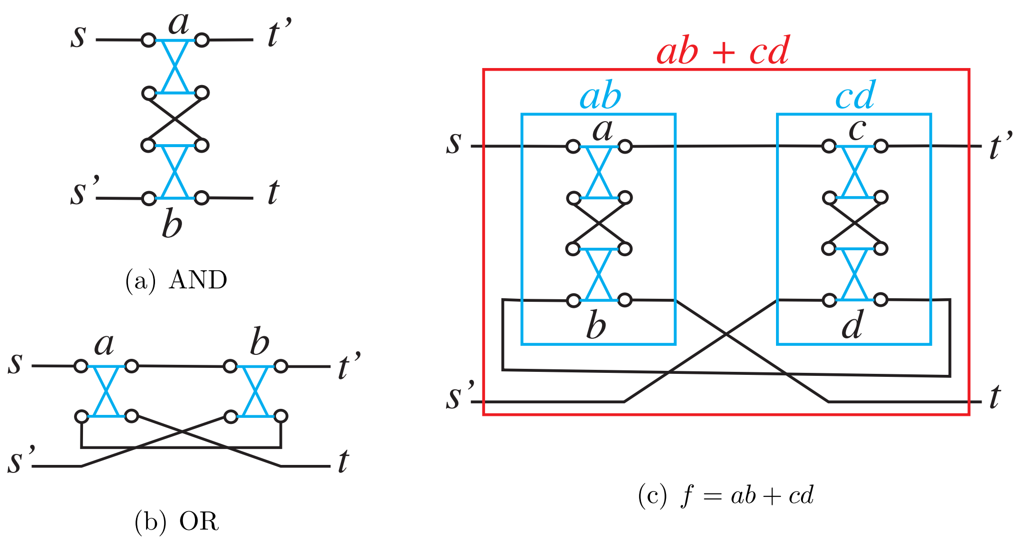
7.1.1 Background and Our Contribution

BDD-based optical logic circuits have a fundamental disadvantage of a large amount of signal power discarded in the circuit, which causes a large power consumption. This power loss is called a garbage output. Let us explain why garbage outputs exist in a BDD-based circuit. A splitter splits the signal power of an input light into two outputs. Due to the property of a BDD, both of the two output lights never reach the PD simultaneously, which means that one of the output lights is always discarded in the circuit. Therefore, many splitters result in a large amount of garbage outputs. Though methods for eliminating splitters have been proposed in [35,36], and in chapters 3 and 5, these methods can not address all splitters. In other words, a garbage output inevitably exists in a BDD-based optical logic circuit based on existing synthesis methods. In some logic functions, garbage outputs are still a fundamental issue of optical logic circuits. Therefore, we propose a cross-bar gate logic (CBGL) as a new scheme for optical logic circuits without garbage outputs. Then, we enumerate CBGL with the minimum number of gates for all three input functions by an exhaustive search and make a library of the optimal CBGL.

7.1.2 Existing Method for Circuits without Garbage Outputs

A method for synthesizing an optical logic circuit without garbage outputs is proposed [30]. This method uses a circuit implementing two-input functions such as AND, OR, and XOR as a basic building block. It synthesizes a logic function with a large number of input variables based on the basic building block. In this method, the basic building block is called a virtual gate (VG). Here, we introduce a synthesis method utilizing circuits for AND and OR as the basic building block. Fig. 7.1 (a) and (b) show VGs for logic function $f = ab$ (AND) and logic function $f = a + b$ (OR) respectively. In a VG, logic values 1 and 0 are assigned to input s and s'. Depending on the logical values given to the control inputs a and b, the state of the gate is determined to be cross or bar, forming a path, and the output value of the logical function f is detected at t. At the same time, $\bar{f}$ is detected at t'. When $f = 0$, input s and s' reach output t' and t, respectively. When $f = 1$, input s and s' reach output t and t', respectively. Thus, in VG, inputs s, s' always reach t or t' in any case. Therefore, the optical signal input from inputs s and s' is always detected at output t or t'. In other words, the circuit has no garbage output. Existing methods implement arbitrary functions by the nested structure of VG. For example, Fig. 7.1 (c) shows the circuit for $f = ab + cd$, where two AND circuits are nested within an OR circuit. Operations of AND, OR, and NOT are a complete system, and any gate in VG can be replaced with VG, so we can implement any logic function by using AND and OR as VG. Existing method [30] synthesizes a circuit based on the logic expression for the target logic function. The number of gates is equal to the number of literals in the logic expression. Therefore, obtaining logic expressions with a small number of literals is essential in the existing method. However, since it is not the subject of this thesis, we consider the case of logical synthesis based on the sum-of-products form. For example of Fig. 7.1 (c), the function is expressed as $f = ab + cd$ in the sum of products form. When the existing method synthesizes the circuit based on this function expression, we obtain the circuits shown in Fig. 7.1 (c), and the number of gates is four.

Although the existing method based on VGs for two-input functions can implement any logic functions, the result circuits are not optimal at least in terms of the number of gates. Fig. 7.2 shows the circuit with three gates for logic function $f = \bar{a}\bar{b} + \bar{a}\bar{c} + abc$. On the other hand, synthesizing the logic function using VGs for AND and OR from the sum of products form



(a) AND

(b) OR

(c) $f = ab + cd$

Figure 7.1: Existing method for circuits without garbage outputs.

results in a circuit with seven gates. Thus, when synthesizing logic functions with three or more inputs, the circuits obtained by the existing method can be far from optimal. Therefore, we propose a cross-bar gate logic (CBGL) as a new scheme for optical logic circuits without garbage outputs. Then, we enumerate CBGL with the minimum number of gates for all three input functions by an exhaustive search and make a library of the optimal CBGL. Since CBGLs include the circuit obtained by the existing method, the number of gates of the CBGL circuit with the minimum number of gates enumerated is equal to or smaller than that of the existing method.

7.2 Cross-Bar Gate Logic

This section proposes a CBGL as the circuit without garbage outputs.

7.2.1 Concept

Let us consider a circuit that consists of cross-bar gates, two inputs s, s' and two output t, t', and have the following two properties. A cross-bar gate is a logic gate with has two inputs, two outputs, and one control input that

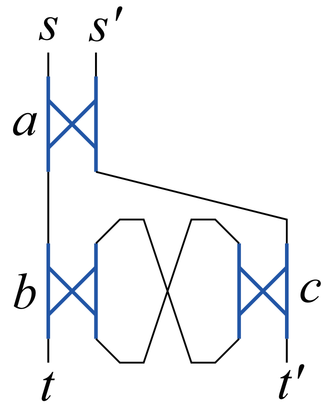


Figure 7.2: Example of CBGL with the minimum number of gates.

has the following operation. The control input determines whether the input signals pass straight to the outputs (bar-state) or whether the input signals pass to the outputs after crossing each other (cross-state). A DC is one type of cross-bar gates.

Property 1: In a cross-bar gate, the signal flow is fixed.

Property 2: When input s reaches output $t(t')$, input s' reaches output $t'(t)$. In other words, inputs s and s' are completely routed to outputs.

For example, the circuits shown in Figs. 3 and 4 are CBGL circuits. Therefore, we explain properties 1 and 2 by using the circuit in Fig. 3(a). In the cross-bar gate, the signal always flows from left to right regardless of the logic values of a and b. In other words, property 1 is satisfied. This property is necessary for nested structures since the circuit output results in the incorrect value if the signal flows in an unintended direction, such as from s to s'. The nested structure enables logic synthesis for large-scale logic functions from basic building blocks. Next, consider Property 2. Property 2 means that signals from inputs s, s' always reach the output, so this circuit has no garbage output.

Let us consider sufficient conditions for properties 1 and 2. First, we consider a sufficient condition for property 1. When a circuit satisfies the following condition, the circuit has the property 1.

Condition 1: We distinguish each port as an input port pair and an output port pair, connecting an input port and an output port.

Let us consider the situation when condition 1 is satisfied. We input a signal from input s, then the signal passes through an input port and propagate to one of the output port according to the control input of the gate. After that, the signal propagates to the input port of the next gate. Like this, the signal passes through an input port and an output port alternately until reaching the output t or t'. Therefore, the signal always flows from an input port to an output port in a gate, corresponding to property 1. Next, consider a sufficient condition for property 2 when condition 1 is satisfied. For simplicity, in the following, we consider a circuit in which cross or bar configurations are assigned to each gate as a directed graph. For example, Fig. 7.3 shows the graph corresponding to the AND circuit shown in Fig. 7.1 (a). Nodes represent ports of gates, and edges represent the direction of signal flow. In order to distinguish each port as an input port and an output port, we define nodes corresponding to input ports as in-nodes and nodes corresponding to output ports as out-nodes, respectively. Here, v_{in} denotes an in-node and v_{out} denotes an out-node. Node s, s' represent input s, s', and node t, t' represent output t, t'. In a CBGL, the signal propagates from input s or s' to an input port, then the signal finally propagates from an output port to output t or t'. Therefore, $d^+(s) = d^+(s') = d^-(t) = d^-(t') = 1$ and $d^-(s) = d^-(s') = d^+(t) = d^+(t') = 0$, where $d^+(v)$ and $d^-(v)$ denote an outdegree and an indegree of node v respectively. The signal always flows from an input port to an output port in a gate. The signal always flows from an output port to an input port between gates. Therefore, $d^+(v_{in}) = d^-(v_{out}) = 1$. Note that $d^-(v_{in}) \geq 1$ and $d^+(v_{out}) \geq 1$ since a port is sometimes connected to multiple ports. Moreover, when a circuit satisfies the following condition, the circuit has property 2.

Condition 2: Each port is connected to just one port.

Since the number of input ports is equal to that of output ports, it is possible to satisfy condition 2, and we can obtain $d^-(v_{in}) = 1$ and $d^+(v_{out}) = 1$. From an indegree and an outdegree of each node, we can see that the graph consists of two paths from the input to the output and some cycles (sometimes there is no cycle). Therefore, the circuit has property 2. We define a circuit that satisfies conditions 1 and 2 as a CBGL. Since CBGLs have properties 1 and 2, optical circuits based on CBGL have no garbage outputs and can incorporate nested structures. We focus on the relationship between the routing from the input to the output and the value of the logic function f. The whole circuit can be regarded as a cross-bar gate with a bar-state when $f = 0$ and a cross-state when $f = 1$. The name cross-bar gate logic is derived from this

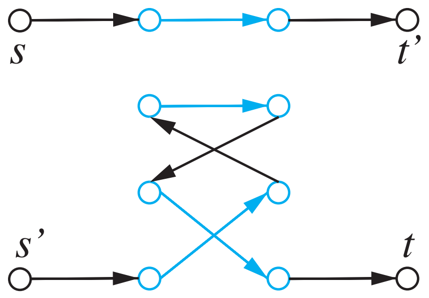


Figure 7.3: Graph corresponding to AND circuit.

property.

7.2.2 Path Length

The delay and power consumption of an optical circuit based on a CBGL are determined by the number of cross-bar gates in the path from input ports to output ports. The delay and the power consumption are n ps and $A^n \times 10\mu$W, where n is the number of gates in the path and A is the signal power attenuation at a cross-bar gate(DC), and the minimum detectable signal power at a PD is assumed to be 10 μW. When we input a light from the input s and detect the light at the output t, n is equal to $(L-1)/2$, where L is the length of the st-path in the graph. Although the length of the st-path is quite different between each function, we can give an upper limit of the length. Let us consider the length of the st-path in a CBGL with g gates. The indegree and outdegree of a node are one except for node s, s', t and t', which means that all nodes are visited just once. Here, we define the path from s' to t' as L'. When there is no cycle, $L+L' = |E| = 4g+2$, where $|E|$ is the number of edges in the graph, since the graph consists of only the path from input s to output t and the path from input s' to output t'. When a cycle exists, which reduces the number of edges included in the st-path or $s't'$-path, resulting in $L + L' \leq |E| = 4g + 2$. Since input s' must not be connected to output t' directly, the path from input s' to output t' has at least one in-node and one out-node, which means $L' \geq 3$. From the above discussion, $L \leq 4g - 1$. Therefore, finding the CBGL with the minimum number of gates can reduce the delay and the power consumption.

7.3 Enumerate Minimum Circuits

The flow of a method for enumerating the CBGL with the minimum number of gates is summarized below.
Step 1: Enumerate combinations of wiring between gates. We call this combination of wiring between gates a layout.
Step 2: Assign input variables to each gate for layouts obtained by the first step, then check the logic function implemented by the obtained circuit. We apply these processes to all layouts obtained by the first step.
Step 3: Iterate Steps 1 and 2 while increasing the number of gates from one.

7.3.1 Enumerate Layouts

First, describe a simple method for enumerating layouts having g gates that satisfy conditions 1 and 2. As shown in Fig. 7.4, we label in-nodes and out-nodes with numbers separately. In-nodes are labeled with blue numbers, and out-nodes are labeled with red numbers in Fig. 7.4. We label each gate with a number from 1 to g. We label two in-nodes of i-th gate with $2i$ and $2i+1$, respectively. Since node t and t' have one indegree and zero outdegree, we consider output t and t' as in-nodes, labeling node t and t' with 0 and 1. We also label two out-nodes of i-th gate with $2i$ and $2i+1$, respectively. Since node s and s' have 0-indegree and 1-outdegree, we consider input s and s' as out-nodes, labeling node s and s' with 0 and 1. When i-th gate is bar-state, $2i$-th in-node reaches $2i$-th out-node and $2i+1$-th in-node reaches $2i+1$-th out-node. When i-th gate is cross-state, vice versa. As explained in Section 7.2, when all out-nodes are respectively connected to a different in-node with one edge, the layout satisfies conditions 1 and 2. Let us describe a simple method for enumerating such layouts. We define 0-th out-node as a

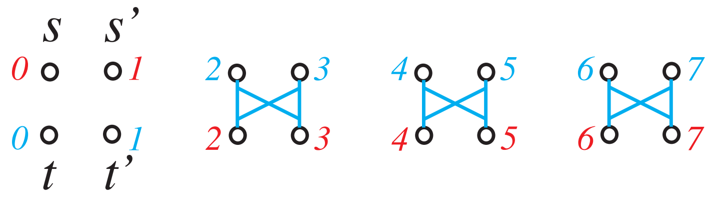


Figure 7.4: Labeling nodes with a number.

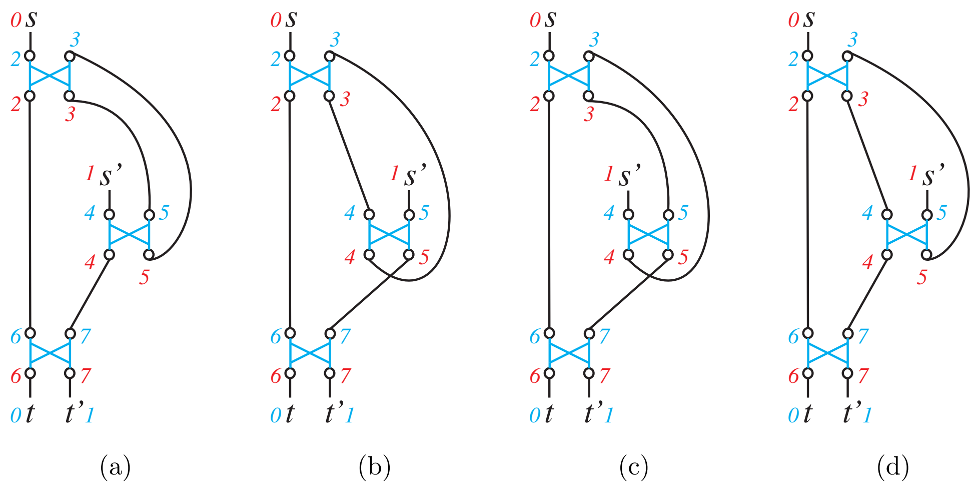


Figure 7.5: Equivalent layouts.

target node, then choose an in-node and place an edge between the in-node and the target. We can obtain a layout satisfying conditions 1 and 2 by iterating this operation from 0-th out-node to $2g+1$-th out-node. Note that we can choose each in-node just once. Let us call the in-nodes that can be chosen in each iteration as candidate in-nodes. We can enumerate layouts by examining all candidate in-nodes in each step. The number of layouts obtained by this simple method is $(2(g+1))!$. However, we can reduce the number of layouts by removing redundant or equivalent ones. First, consider the wiring between the same gate. When $2i+1$-th in-node and $2i+1$-th out-node are connected, $2i$-th in-node always reaches $2i$-th out-node regardless of the value of the control input of i-th gate, which means that such wiring is redundant. Next, consider the four layouts shown in Fig. 7.5. Although the layouts (a) and (b) have different topologies, these two layouts are logically equivalent. Take a look at layout (a). When the value of the control input is 0, node s' and 3-th out-node reach 7-th in-node and 3-th in-node. When the value of the control input is 1, node s' and 3-th out-node reach 3-th in-node and 7-th in-node. Consider layout (b). Assigning logic values to the control input shows that the layout (b) is logically equivalent to layout (a). Similarly, layouts (c) and (d) are logically equivalent. In order to remove such equivalent layouts, we enumerate layouts under a constraint: $O_{2i} < O_{2i+1}$, where O_j is the number labeled with out-node that is connected to j-th

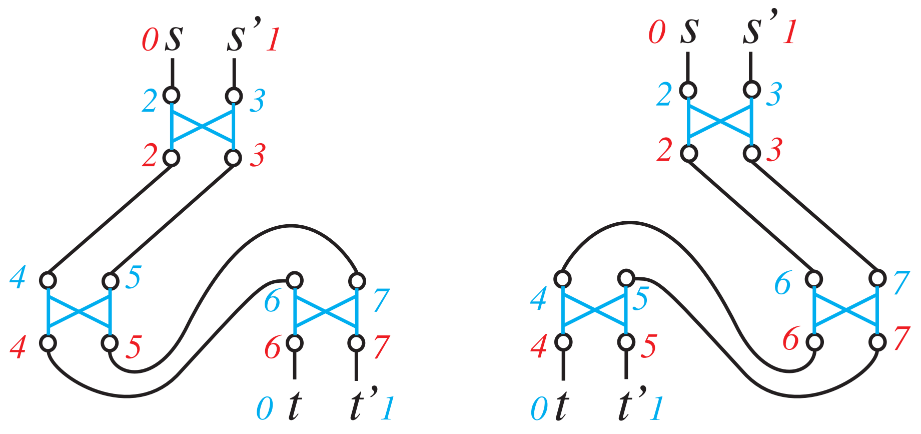


Figure 7.6: Equivalent layouts.

in-node. We can remove the layouts (b) and (d) by this constraint. Let us call this constraint 1. Moreover, the remaining two layouts are logically equivalent, considering the negation of the control input. Therefore, in order to remove such equivalent layouts, we enumerate layouts under a constraint: $I_{2i} < I_{2i+1}$, where I_j is the number labeled with in-node that is connected to j-th out-node. Let us call this constraint 2. Finally, consider the layouts shown in Fig. 7.6. It is obvious that these two layouts are logically equivalent. However, such layouts can be generated when we enumerate layouts under the constraint mentioned above. Therefore, we need an extra constraint: When the number of gates corresponding to an in-node that are already chosen is k, candidate in-nodes must be up to $2k + 3$-th in-nodes. Let us call this constraint 3. By this constraint, we can remove the layout shown in Fig. 7.6 (b).

Our method for enumerating layouts is summarized below. Like the simple method, we chose an in-node among candidate in-nodes for the target out-node. Then we iterate this operation from 0-th out-node to $2g + 1$-th out-node. Although the simple method examines all candidate in-nodes in each step, our method removes some in-nodes from candidate in-nodes based on the constraints mentioned above. Let us explain the examples of defining $2i$-th out-node and $2i + 1$-th out-node as a target out-node since there are some differences in the constraints between the two nodes. Consider when

Table 7.1: The number of layouts

g : # of gates	Simple method : (2(g+1))!	Our method
2	720	3
3	40320	14
4	3.6×10^5	91
5	4.8×10^8	722
6	8.7×10^{10}	6733
7	2.1×10^{13}	71639

we chose the destination of $2i$-th out-node. Since $2i$-th in-node and $2i+1$-th in-node are in the same gate as $2i$-th out-node, we remove these two in-nodes from candidate in-nodes. If $2l$-th in-node is not chosen, we remove $2l+1$-th in-node from candidate in-nodes, corresponding to constraint 1. If the number of gates corresponding to the in-node already chosen is k, we remove in-nodes after $2k+4$-th in-nodes, corresponding to constraint 3. We consider the case where j-th in-node is chosen for $2i$-th out-node. Then, we choose ab in-node for $2i+1$-th out-node. We remove in-nodes before j-th in-node, corresponding to constraint 2. If $2m+1$-th in-node is already selected, we remove $2m$-th in-node from the candidates, corresponding to constraint 1. The process regarding constraint 3 is the same as $2i$-th out-node. We can obtain all valid layouts with g gates by iterating these operations from 0-th out-node to $2g+3$-th out-node. Table 7.1 shows the number of layouts enumerated by the simple method and our method. It is observed that the number of layouts enumerated by our method is significantly smaller than that of the simple method, which drastically reduces the computational cost for the enumeration of CBGL with the minimum number of gates.

7.3.2 Variable Assignment and Function Checking

We give a brief explanation for assigning input variables to each gate and checking the function implemented. We can also assign the negation of an input variable to a gate. Therefore, when we consider CBGL for n-input functions, the number of candidates of the control input assigned to each gate is $2n$. Note that each input variable must be assigned to at least one gate. After assigning input variables to each gate, we check the function implemented. We can obtain the function by examining all possible com-

Table 7.2: The number of enumerated functions

g : # of gates	# of logic functions	# of NPN-equivalent classes
0	2	1
1	6	1
2	30	2
3	114	5
4	80	3
5	24	2
Total	256	14

binations of values of the control inputs such that input s reaches output t.

7.4 Experiment

We enumerated CBGL for three-input functions with the minimum number of gates by our method described in Section 7.3. We iteratively ran our method while increasing the number of gates from 0. Table 7.2 shows the results. The second column shows the number of functions that are found for the first time when the number of gates is set to g. The "Total" row corresponds to the total number of logic functions found in each iteration. Since the number of all three-input logic functions is $2^{2^3} = 256$, it is observed that we enumerated the CBGL for all three-input functions. Here, we introduce the concept of NPN equivalence classes. It is said that two logic functions are NPN equivalent if one can be obtained by the other by negating inputs, permutating inputs, or negating the output. Since these three manipulations can be applied to CBGL, a layout obtained by our method can implement all functions in one NPN equivalence class. The third column shows the number of NPN equivalence classes that are found for the first time when the number of gates is set to g. Since the number of NPN equivalence classes for three-input logic functions is 14, it is observed that we enumerated all NPN equivalence classes for three input logic functions. We extract one logic function from each of the 14 NPN equivalence classes and apply the synthesis method based on AND and OR. Table 7.3 shows the results. The second column represents the number of gates required to implement 14 logic

Table 7.3: The number of gates for NPN equivalence classes

Logic function	AND/OR based	Optimal
#1: 1	0	0
#2: a	1	1
#3: $a\bar{b} + \bar{a}b$	4	2
#4: ab	2	2
#5: $a \oplus b \oplus c$	12	3
#6: $a\bar{b} + a\bar{c}$	4	3
#7: abc	3	3
#8: $\bar{a}\bar{b} + \bar{a}b\bar{c} + abc$	8	3
#9: $\bar{a} + a\bar{b}\bar{c} + abc$	7	3
#10: $\bar{a}\bar{b} + a\bar{c}$	4	4
#11: $\bar{a}\bar{b} + ab\bar{c}$	5	4
#12: $\bar{a}b + \bar{a}c + a\bar{b} + a\bar{c}$	8	4
#13: $\bar{a}bc + \bar{a}b\bar{c} + a\bar{b}\bar{c}$	9	5
#14: $ab + bc + ac$	6	5

functions shown in the first column. As explained in Section 6.1.2, since the number of gates in the circuit synthesized based on AND and OR depends on the function expression, we can reduce the number of gates compared to the results in Table 7.3. However, this thesis is not focused on a method for a compact function expression. Accordingly, we show the results based on a sum of products form. The third column represents the minimum number of gates required to implement 14 logic functions in the first column, which corresponds to the results obtained from the enumeration of CBGL. The results demonstrate that using our library of CBGL can largely reduce the number of gates.

Table 7.4 shows the number of gates, power consumption, and delay of naive BDD-based circuits and CBGL. In order to evaluate the performance of optical logic circuits, this thesis assumes that the propagation delay of a light passing through a DC is 1 ps [69]. We assume that the power attenuation in a DC is -1 dB [69]. The minimum optical power which the PD can detect is assumed to be 10 μW. The column of "logic function" indicates the logic function to be synthesized, and the number corresponds to the logic function number in Table 3. The "BDD" column represents the results of

Table 7.4: BDD-based circuits and CBGLs with the minimum number of gates

	# of gates		Power [μW]		Delay [ps]	
Logic function	BDD	CBGL	BDD	CBGL	BDD	CBGL
#1	0	0	0	0	0	0
#2	1	1	12.5	12.5	1	1
#3	3	2	31.2	15.6	2	2
#4	2	2	15.6	19.5	2	3
#5	5	3	78.1	19.5	3	3
#6	5	3	54.6	24.4	3	4
#7	4	3	51.5	24.4	3	4
#8	3	3	19.5	30.5	3	5
#9	3	3	35.1	24.4	3	4
#10	3	4	31.2	30.5	2	5
#11	4	4	35.1	30.5	3	5
#12	5	4	70.3	38.1	3	6
#13	5	5	74.2	38.1	3	6
#14	4	5	54.6	47.6	3	7

the circuit based on BDD, and the "CBGL" column represents the CBGL results with the minimum number of gates enumerated in this paper. The power consumption of the CBGL with the minimum number of gates can be much smaller than that of the BDD-based circuit. This is because there are many garbage outputs in BDD-based circuits but no garbage outputs in CBGL circuits. However, the delay of CBGL circuits with the minimum number of gates is larger than that of BDD-based circuits. This is because the number of gates the input light passes through is larger in CBGL circuits than in BDD-based circuits. In the worst case, the number of gates the input light passes through in the CBGL is $2g-1$ (g is the number of gates). On the other hand, the number of gates the input light passes through in the BDD-based circuit is n (n is the number of input variables). Since CBGL has pros and cons compared to BDD, it is necessary to exploit CBGL appropriately according to design constraints and target functions.

7.5 Conclusion

A garbage output is a fundamental issue of optical logic circuits. To address this issue, we proposed the concept of CBGL as the synthesis method for optical logic circuits without garbage outputs. Although a method for synthesizing CBGL is based on two-input logic functions, this method can not build the optimal CBGL for logic functions with more than two input variables. Therefore, we enumerated CBGL with the minimum number of gates. Experimental results demonstrated that the enumerated circuits have a smaller number of gates compared with the existing method. We also showed that the enumerated circuits could consume smaller power than the BDD-based circuits.

Chapter 8

Conclusion

In this thesis, we propopse synthesis methods for efficient optical logic circuits. Below, we conclude this thesis by summarizing each contribution and presenting future works and future directions.

Capter 3: Wavelength Division Multiplexing and Splitter Elimination. We proposed inter-function sharing and intra-function sharing. These methods can reduce the size of BDD-based optical logic circuits by exploiting wavelength division multiplexing (WDM). We also propose a method for replacing a splitter with a logic gate, which reduces the power dissipation in laser sources. Experimental results obtained using a partial product accumulation circuit used in a 4-bit parallel multiplier demonstrate significant advantages of our method over existing approaches in terms of area and power consumption. Our future work will be focused on developing more formal algorithms which optimize the optical circuit considering the tradeoffs among area, power, and delay appropriately.

Chapter 4: BDD Variable Ordering for Minimizing Power Consumption. We proposed a variable ordering algorithm for minimizing the power consumption. To the best of our knowledge, this is the first study of an optimization method of BDDs for optical logic circuits. In this thesis, we demonstrate that the power consumption largely depends on the variable order of a BDD; however, an optimization problem of finding the variable order to minimize the power consumption has large time complexity. Our algorithm utilizes an efficient reordering method based on the adjacent variable swap to reduce the execution time. Experimental results using 10-input logic functions obtained by applying an LUT technology mapper to an ISCAS'85 c7552 benchmark circuit demonstrate that our algorithm can reduce the power consumption by an average of 30% within a reasonable amount of time compared to the results of variable orders that minimize the number of nodes. Our future work will focus on developing a variable ordering algorithm for sufficiently reducing power consumption within a reasonable execution time for functions with more than ten inputs.

Chapter 5: Dual Port Node Sharing. We proposed a method for eliminating a splitter exploiting this dual port. We define a BDD node corresponding to a dual port as a dual port node (DP node) and call the proposed method DP node sharing. We demonstrated that DP node sharing reduces power consumption drastically and circuit size slightly without increasing delay. We conducted an experiment involving 10-input logic functions obtained by applying an LUT technology mapper to an ISCSA ' 85 C7552 benchmark circuit to evaluate our DP node sharing. The experimental results demonstrated that DP node sharing reduces the power consumption by two orders of magnitude of circuits that consume a large amount of power.

Chapter 6: Multi-Stage Optimization for Optical Logic Circuits. We proposed a synthesis method reducing power dissipation to a polynomial order of the number of inputs while exploiting the high-speed nature. Our method divides the target logic function into multiple sub-functions with Optical-to-Electrical (OE) converters. Each sub-function has a smaller number of inputs than that of the original function, which exponentially reduces the power dissipated by an optical logic circuit representing the sub-function. The proposed synthesis method can mitigate the OE converter delay overhead by parallelizing sub-functions. We apply the proposed synthesis method to the ISCAS'85 benchmark circuits. The power consumption of the conventional circuits based on the Binary Decision Diagram (BDD) is at least three orders of magnitude larger than that of the optical logic circuits synthesized by the proposed method. The proposed method reduces the power consumption to about 100 mW. The delay of almost all the circuits synthesized by the proposed method is kept less than four times the delay of the conventional BDD-based circuit.

Chapter 7: Optical Logic Circuits without Garbage Outputs. We considered a circuit design with two optical inputs and two optical outputs in which an input light always propagates to an optical output; in other words, this circuit has no garbage output. We define this scheme as a cross-bar gate logic (CBGL). Although a method for synthesizing cross-bar gate logic is proposed, this method can minimize the number of gates for only functions with up to two input variables. This paper finds the circuit with the minimum number of gates for all three-input functions by an exhaustive search. Since the search space is vast, we introduce a technique to prune it efficiently. Experimental results demonstrate that the maximum number of gates required to implement each function is five among all three-input functions. In the best case, the number of gates in the optimal circuit obtained by our exhaustive search is one-half compared to the existing method. Our future work will be focused on developing a synthesis method for large-scale logic functions based on the library of the optimal CBGL obtained by this research.

Future Works

Although combining methods in each chapter is expected to improve the synthesis result, the optimization is not easy since each method has different characteristics. For example, each method works effectively for different functions. Methods in chapte 3 work effectively for symmetric functions since symmetric functions have common structures among the sub-graph and a lot of branches in its BDD. On the other hand, optimization for variable ordering in chapter 4 has no effect on symmetric functions since the structure in its BDD does not depend on the variable order. Although DP node sharing in chapter 5 is effective against the XOR function, which is one of the symmetric functions, XOR function is a special case, and there are some symmetric functions for which DP node sharing can not work. Therefore, it is essential to study more deeply the effect of each method on various functions. In addition, combining each proposed method have issues with the optimization. Here, we describe future works regarding these issues.

- The power consumption, size, and delay of BDD-based circuits incorporating the proposed methods depend on the BDD variable order, like naive BDD-based circuits. In naive BDD-based circuits, since a variable order with a larger number of redundant nodes results in small power consumption, we can build a heuristic algorithm of variable ordering for power-efficient BDD-based circuits. On the other hand, we do not know the effect of variable ordering on BDD-based circuits incorporating the proposed methods, which is essential for heuristic variable ordering for efficient BDD-based circuits.

- Inter-function sharing and intra-function sharing result in a smaller number of equivalent nodes and redundant nodes, reducing the number of nodes to which DP node sharing can be applied. Therefore, it is essential to study an optimization algorithm for inter-function sharing and intra-function sharing considering DP node sharing.

- Although the multi-stage optimization in chapter 6 adopt a BDD-based circuit for implementing sub-functions, we can also adopt CBGL circuits. Experimental results in chapter 7 demonstrate that CBGL circuits sometimes have advantages in power consumption or area efficiency compared with BDD-based circuits. Accordingly, we can improve the multi-level optimization by choosing an appropriate scheme for the sub-functions. In order to reach this goal, it is essential to build a library of optimal BDD-based circuits and optimal CBGL circuits for all logic functions with a small number of variable inputs.

Future Directions

Finally, we give future directions of optical logic circuits.

- From the power consumption perspective, it is challenging to implement a large-scale and complex logic function with only serial-connections of optical logic gates due to signal attenuation and power loss. We can mitigate signal attenuation by exploiting cascade-connections, which spoils the advantage of ultra-high-speed operation. Therefore, optical logic circuits are expected to be utilized for small-scale functions that require ultra-high-speed operation.

- Logic synthesis methods have been studied based on various schemes. Among them, synthesis methods for circuits without splitters are promising. Circuits without splitters tend to have a long path from the optical source to the optical output compared to circuits with splitters, which results in large power consumption of circuits without splitters. However, with the development of technology for nanophotonic devices, signal power attenuation at optical gates may be drastically reduced in the future. On the other hand, power loss at a splitter can not be improved. Therefore, it is essential for the practical application of optical logic circuits in the future to study logic synthesis for optical circuits without splitters.

Bibliography

[1] A. Ceyhan, M. Jung, S. Panth, S. K. Lim, and A. Naeemi, "Impact of Size Effects in Local Interconnects for Future Technology Nodes: A Study Based on Full-Chip Layouts," in *Proc. IEEE Interconnect Technology Conference/Advanced Metallization Conference*, pp. 345–348, May 2014.

[2] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for sub-45nm Early Design Exploration," *IEEE Transactions on Electron Devices*, vol. 53, pp. 2816–2823, Nov. 2006.

[3] Y. Vlasov, "Silicon Photonics for Next Generation Computing Systems," in *Proc. European Conference on Optical Communications*, no. Tu.1.A.1, Sept. 2008.

[4] N. Yamamoto, T. Ogawa, and K. Komori, "Photonic Crystal Directional Coupler Switch with Small Switching Length and Wide Bandwidth," *Optics Express*, vol. 14, pp. 1223–1229, Feb. 2006.

[5] H. Dong, H. Sun, Q. Wang, N. Dutta, and J. Jaques, "80gb/s all-optical logic and operation using mach-zehnder interferometer with differential scheme," *Optics Communications*, vol. 265, pp. 79–83, 09 2006.

[6] P. Zhou, L. Zhang, Y. Tian, and L. Yang, "10 ghz electro-optical or/nor directed logic device based on silicon micro-ring resonators," *Optics Letters*, vol. 39, pp. 1937–1940, Apr 2014.

[7] L. Yang, L. Zhang, C. Guo, and J. Ding, "Xor and xnor operations at 12.5 gb/s using cascaded carrier-depletion microring resonators," *Optics Express*, pp. 2996–3012, 2014.

[8] M. Xu, W. Yangguang, M. Zhai, X. Lu, K. Wang, and J. Ji, "Ultrafast passive all-optical full function logic gates on micro-silicon-on-insulator waveguide chip," *Journal of Nanophotonics*, vol. 12, p. 1, 12 2018.

[9] S. Kita, K. Nozaki, K. Takata, A. Shinya, and M. Notomi, "Ultrashort low-loss Ψ gates for linear optical logic on si photonics platform," *Communications Physics*, vol. 3, 03 2020.

[10] P. Dutta, C. Bandyopadhyay, C. Giri, and H. Rahaman, "Mach-zehnder interferometer based all optical reversible carry-lookahead adder," in *2014 IEEE Computer Society Annual Symposium on VLSI*, pp. 412–417, 2014.

[11] K. Datta, T. Chattopadhyay, and I. Sengupta, "All optical design of binary adders using semiconductor optical amplifier assisted mach-zehnder interferometer," *Microelectron. J.*, vol. 46, p. 839–847, sep 2015.

[12] R. Das, C. Bandyopadhyay, and H. Rahaman, "All optical reversible design of mach-zehnder interferometer based carry-skip adder," in *2016 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pp. 73–78, 2016.

[13] Z. Ying, Z. Wang, S. Dhar, Z. Zhao, D. Z. Pan, and R. T. Chen, "On-chip microring resonator based electro-optic full adder for optical computing," p. JW2A.147, 01 2017.

[14] Y. Tian, L. Zhang, J. Ding, and L. Yang, "Demonstration of electro-optic half-adder using silicon photonic integrated circuits," *Optics Express*, pp. 6958–6965, 2014.

[15] Z. Ying, Z. Wang, Z. Zhao, S. Dhar, D. Z. Pan, R. Soref, and R. T. Chen, "Silicon microdisk-based full adders for optical computing," *Optics Letters*, 2018.

[16] Z. Ying, Z. Wang, Z. Zhao, S. Dhar, D. Pan, R. Soref, and R. Chen, "Comparison of microrings and microdisks for high-speed optical modulation in silicon photonics," *Applied Physics Letters*, vol. 112, p. 111108, 03 2018.

[17] T. Ishihara, A. Shinya, K. Inoue, K. Nozaki, and M. Notomi, "An integrated nanophotonic parallel adder," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, July 2018.

[18] K. Datta and I. Sengupta, "All optical reversible multiplexer design using mach-zehnder interferometer," in *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pp. 539–544, 2014.

[19] J. Roy, T. Chattopadhyay, and T. Sarkar, "All-optical multiplication using soa-mzi based programmable logic device (pld)," 12 2010.

[20] S. Kumar, A. Bisht, G. Singh, and A. Amphawan, "Implementation of 2-bit multiplier based on electro-optic effect in mach-zehnder interferometers," *Optical and Quantum Electronics*, vol. 47, 08 2015.

[21] A. Manna, S. Saha, R. Das, C. Bandyopadhyay, and H. Rahaman, "All optical design of cost efficient multiplier circuit using terahertz optical asymmetric demultiplexer," in *2017 7th International Symposium on Embedded Computing and System Design (ISED)*, pp. 1–5, 2017.

[22] S. Sharma, K. Chakrabarty, and S. Roy, "On designing all-optical multipliers using mach-zender interferometers," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, pp. 672–679, 2018.

[23] B. Chakraborty and S. Mukhopadhyay, "All-optical method of developing half and full subtractors by the use of phase encoding principle," *Optik - International Journal for Light and Electron Optics*, vol. 122, p. 2207–2210, 12 2011.

[24] A. Kumar, S. Kumar, and S. Raghuwanshi, "Implementation of full-adder and full-subtractor based on electro-optic effect in mach–zehnder interferometers," *Optics Communications*, vol. 324, p. 93–107, 08 2014.

[25] S. Abdulnabi and M. Abbas, "Design an all-optical combinational logic circuits based on nano-ring insulator-metal-insulator plasmonic waveguides," *Photonics*, vol. 6, p. 30, 03 2019.

[26] A. Cherri, "Designs of all-optical higher-order signed-digit adders using polarization-encoded based terahertz-optical-asymmetric-demultiplexer (toad)," 06 2014.

[27] Nivedita, S. Kaur, and R. Goyal, "All-optical decoder/demultiplexer with enable using soa based mach-zehnder interferometers," in *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 780–784, 2019.

[28] R. Kaler and R. S. Kaler, "Implentation of optical encoder and multiplexer using mach–zehnder inferometer," *Optik - International Journal for Light and Electron Optics*, vol. 122, pp. 1399–1405, 08 2011.

[29] R. Rajasekar, R. Latha, and S. Robinson, "Ultra-contrast ratio optical encoder using photonic crystal waveguide," *Materials Letters*, vol. 251, 05 2019.

[30] C. Condrat, P. Kalla, and S. Blair, "Logic Synthesis for Integrated Optics," in *Proc. Great lakes symposium on VLSI*, pp. 13–18, May 2011.

[31] Q. Xu and R. Sorei, "Reconfigurable Optical Directed-Logic Circuits Using Microresonator-Based Optical Switches," *Optics Express*, vol. 19, pp. 5244–5259, Mar. 2011.

[32] A. Deb, R. Wille, O. Keszocze, S. Hillmich, and R. Drechsler, "Gates vs. splitters: Contradictory optimization objectives in the synthesis of optical circuits," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, pp. 1–13, 06 2016.

[33] A. Deb, R. Wille, and R. Drechsler, "Dedicated synthesis for mzi-based optical circuits based on and-inverter graphs," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 233–238, 2017.

[34] A. Deb, R. Wille, and R. Drechsler, "Or-inverter graphs for the synthesis of optical circuits," in *2017 IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL)*, pp. 278–283, 2017.

[35] R. Wille, O. Keszocze, C. Hopfmuller, and R. Drechsler, "Reverse BDD-based Synthesis for Splitter-Free Optical Circuits," in *Proc. Asia-South Pacific Design Automation Conference*, pp. 172–177, Jan. 2015.

[36] Z. Zhao, Z. Wang, Z. Ying, S. Dhar, R. T. Chen, and D. Z. Pan, "Logic Synthesis for Energy-Efficient Photonic Integrated Circuits," in *Proc. Asia-South Pacific Design Automation Conference*, Jan. 2018.

[37] Z. Zhao, D. Liu, Z. Ying, B. Xu, C. Feng, R. T. Chen, and D. Z. Pan, "Exploiting wavelength division multiplexing for optical logic synthesis," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1567–1570, March 2019.

[38] Z. Zhao, J. Gu, Z. Ying, C. Feng, R. T. Chen, and D. Z. Pan, "Design technology for scalable and robust photonic integrated circuits: Invited paper," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, Nov 2019.

[39] Y. Shen, N. C. Harris, S. Skirlo, D. Englund, and M. Soljačić, "Deep learning with coherent nanophotonic circuits," in *2017 IEEE Photonics Society Summer Topical Meeting Series (SUM)*, pp. 189–190, 2017.

[40] Z. Zhao, D. Liu, M. Li, Z. Ying, L. Zhang, B. Xu, B. Yu, R. T. Chen, and D. Z. Pan, "Hardware-software co-design of slimmed optical neural networks," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ASPDAC '19, (New York, NY, USA), p. 705–710, Association for Computing Machinery, 2019.

[41] T. Ishihara, J. Shiomi, N. Hattori, Y. Masuda, A. Shinya, and M. Notomi, "An optical neural network architecture based on highly parallelized wdm-multiplier-accumulator," in *2019 IEEE/ACM Workshop on Photonics-Optics Technology Oriented Networking, Information and Computing Systems (PHOTONICS)*, pp. 15–21, 2019.

[42] J. Shamir, H. J. Caulfield, W. J. Micelli, and R. J. Seymour, "Optical computing and the fredkin gates.," *Applied optics*, vol. 25, pp. 1604–1607, 1986.

[43] A. Poustie and K. Blow, "Demonstration of an all-optical fredkin gate," *Optics Communications*, vol. 174, pp. 317–320, jan 2000.

[44] J. Hardy and J. Shamir, "Optics Inspired Logic Architecture," *Optics Express*, vol. 15, pp. 150–165, Jan. 2007.

[45] C. Y. Lee, "Representation of switching circuits by binary-decision programs," *The Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, 1959.

[46] Akers, “Binary decision diagrams,” *IEEE Transactions on Computers*, vol. C-27, no. 6, pp. 509–516, 1978.

[47] R. E. Bryant, “Graph-Based Algorithms for Boolean Function Manipulation,” *IEEE Transactions on Computers*, vol. C-35, pp. 677–691, August 1986.

[48] M. Fujita, H. Fujisawa, and N. Kawato, “Evaluation and improvement of boolean comparison method based on binary decision diagrams,” in *[1988] IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*, pp. 2–5, 1988.

[49] S. Malik, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, “Logic verification using binary decision diagrams in a logic synthesis environment,” in *[1988] IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*, pp. 6–9, 1988.

[50] O. Coudert and J. Madre, “A unified framework for the formal verification of sequential circuits,” in *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pp. 126–129, 1990.

[51] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill, “Sequential circuit verification using symbolic model checking,” in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, DAC ’90, (New York, NY, USA), p. 46–51, Association for Computing Machinery, 1991.

[52] Y. Matsunaga and M. Fujita, “Multi-level logic optimization using binary decision diagrams,” in *1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pp. 556–559, 1989.

[53] R. Drechsler, N. Drechsler, and W. Gunther, “Fast exact minimization of BDDs,” in *Proceedings 1998 Design and Automation Conference. 35th DAC. (Cat. No.98CH36175)*, pp. 200–205, 1998.

[54] S. J. Friedman and K. J. Supowit, “Finding the optimal variable ordering for binary decision diagrams,” *IEEE Transactions on Computers*, vol. 39, no. 5, pp. 710–713, 1990.

[55] N. Ishiura, H. Sawada, and S. Yajima, “Minimization of binary decision diagrams based on exchanges of variables,” in *1991 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*, pp. 472–475, 1991.

[56] M. Fujita, Y. Matsunaga, and T. Kakuda, "On variable ordering of binary decision diagrams for the application of multi-level logic synthesis," in *Proceedings of the European Conference on Design Automation.*, pp. 50–54, 1991.

[57] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, pp. 42–47, 1993.

[58] S. Tani, K. Hamaguchi, and S. Yajima, "The complexity of the optimal variable ordering problems of shared binary decision diagrams," in *International Symposium on Algorithms and Computation*, 1993.

[59] S. Minato, N. Ishiura, and S. Yajima, "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation," in *Proceedings of the 27th ACM/IEEE Design Automation Conference*, DAC '90, (New York, NY, USA), pp. 52–57, 1991.

[60] S. Minato, "Zero-suppressed bdds for set manipulation in combinatorial problems," in *Proceedings of the 30th International Design Automation Conference*, DAC '93, (New York, NY, USA), p. 272–277, Association for Computing Machinery, 1993.

[61] E. Loekito, J. Bailey, and J. Pei, "A binary decision diagram based approach for mining frequent subsequences," *Knowl. Inf. Syst.*, vol. 24, p. 235–268, aug 2010.

[62] S. Minato, "πdd: A new decision diagram for efficient problem solving in permutation space," in *International Conference on Theory and Applications of Satisfiability Testing*, 2011.

[63] Y. Inoue and S. Minato, "An efficient method for indexing all topological orders of a directed graph," vol. 8889, pp. 103–114, 12 2014.

[64] T. Maehara and Y. Inoue, "Group decision diagram (gdd): A compact representation for permutations," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19, AAAI Press, 2019.

[65] A. Srinivasan, T. Ham, S. Malik, and R. Brayton, “Algorithms for discrete function manipulation,” in *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pp. 92–95, 1990.

[66] A. Darwiche, “Sdd: A new canonical representation of propositional knowledge bases.,” *IJCAI International Joint Conference on Artificial Intelligence*, pp. 819–826, 01 2011.

[67] K. Nozaki, A. Shakoor, S. Matsuo, T. Fujii, K. Takeda, A. Shinya, E. Kuramochi, and M. Notomi, “Ultralow-energy electro-absorption modulator consisting of ingaasp-embedded photonic-crystal waveguide,” *APL PHOTONICS 2*, 2017.

[68] G. I. Stegeman, E. M. Wright, N. Finlayson, R. Zanoni, and C. T. Seaton, “Third order nonlinear integrated optics,” *Journal of Lightwave Technology*, vol. 6, pp. 953–970, Jun 1988.

[69] H. C. Nguyen, N. Yazawa, S. Hashimoto, S. Otsuka, and T. Baba, “Sub-100 μm Photonic Crystal Si Optical Modulators: Spectral, Athermal, and High-Speed Performance,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, pp. 127–137, June 2013.

[70] L. O’Faolain, D. M. Beggs, T. P. White, T. Kampfrath, K. Kuipers, and T. F. Krauss, “Compact Optical Switches and Modulators Based on Dispersion Engineered Photonic Crystals,” *IEEE Photonic Journal*, vol. 2, pp. 404–414, June 2010.

[71] O. Solgaard, *Photonic Microsystems: Micro and Nanotechnology Applied to Optical Devices and Systems.* New York, NY: Springer, 2008.

[72] K. Nozaki, S. Matsuo, T. Fujii, K. Takeda, A. Shinya, E. Kuramochi, and M. Notomi, “Femtofarad Optoelectronic Integration Demonstrating Energy-Saving Signal Conversion and Nonlinear Functions,” *Nature Photonics*, vol. 13, pp. 454–459, July 2019.

[73] Q. Xu, B. Schmidt, J. Shakya, and M. Lipson, “Cascaded Silicon Micro-ring Modulators for WDM Optical Interconnection,” *Optics Express*, vol. 14, pp. 9431–9436, Oct. 2006.

[74] Z. Lu, H. Yun, Y. Wang, Z. Chen, F. Zhang, N. A. F. Jaeger, and L. Chrostowski, “Broadband Silicon Photonic Directional Coupler using

Asymmetric-Waveguide based Phase Control," *Optics Express*, vol. 23, pp. 3795–3808, Feb. 2015.

[75] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. Yang, "Spectral Transforms for Large Boolean Functions with Applications to Technology Mapping," in *Proc. Design Automation Conference*, pp. 54–60, June 1993.

[76] K. Yamada, T. Tsuchizawa, T. Watanabe, J. Takahashi, E. Tamechika, M. Takahashi, S. Uchiyama, H. Fukuda, T. Shoji, S. Itabashi, and H. Morita, "Microphotonics Devices Based on Silicon Wire Waveguiding System," *IEICE Transactions on Electronics*, vol. E87-C, pp. 351–358, Mar. 2004.

[77] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 2: Generating All Tuples and Permutations.* Addison-Wesley Professional, 2005.

[78] Robert Brayton and Alan Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *Proc. of International Conference on Computer Aided Verification*, pp. 24–40, 2010.

[79] B. F. and F. H., "A neural netlist of ten combinational benchmark circuits and translator in Fortran," in *Int. Symp. Circuits and Systems (ISCAS)*, pp. 663–698, June 1985.

[80] T. Egawa, T. Ishihara, H. Onodera, A. Shinya, S. Kita, K. Nozaki, K. Takata, and M. Notomi, "Multi-Level Optimization for Large Fan-In Optical Logic Circuits using Integrated Nanophotonics," in *IEEE International Conference on Rebooting Computing*, pp. 43–50, Nov. 2018.

[81] J. Cong and Y.Ding, "FlowMap: An optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Tran. CAD*, vol. 13, no. 1, pp. 1–12, 1994.

[82] J. Cong and Y. Hwang, "Simultaneous depth and area minimization in LUT-based FPGA mapping," in *FPGA '95 Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays*, pp. 68–74, 1995.

[83] F. Somenzi, "CUDD:CU Decision Diagram Package". available online.