Biconditional BDD: A Novel Canonical BDD for Logic Synthesis targeting XOR-rich Circuits

Luca Amarú, Pierre-Emmanuel Gaillardon, Giovanni De Micheli
Integrated Systems Laboratory (LSI), EPFL, Switzerland

***Abstract*—We present a novel class of decision diagrams, called *Biconditional Binary Decision Diagrams* (BBDDs), that enable efficient logic synthesis for XOR-rich circuits. BBDDs are binary decision diagrams where the Shannon's expansion is replaced by the *biconditional* expansion. Since the *biconditional* expansion is based on the XOR/XNOR operations, XOR-rich logic circuits are efficiently represented and manipulated with canonical *Reduced and Ordered BBDDs* (ROBBDDs). Experimental results show that ROBBDDs have 37% fewer nodes on average compared to traditional ROBDDs. To exploit this opportunity in logic synthesis for XOR-rich circuits, we developed a BBDD-based *One-Pass Synthesis* (OPS) methodology. The BBDD-based OPS is capable to harness the potential of novel XOR-efficient devices, such as ambipolar transistors. Experimental results show that our logic synthesis methodology reduces the number of ambipolar transistors by 49.7% on average with respect to state-of-art commercial logic synthesis tool. Considering CMOS technology, the BBBD-based OPS reduces the device count by 31.5% on average compared to commercial synthesis tool.**

I. Introduction

In logic synthesis, the choice of the logic representation form is a crucial point to achieve small and fast circuits. Traditionally, functions are represented using the *Sum Of Products* (SOP) representation [1]. However, the SOP form is not efficient for XOR-rich circuits that are heavily used in arithmetic, error correcting and telecommunication circuits. Indeed, considering XOR-dominated applications, a XOR-based representation form is preferable to achieve better circuit realizations. Motivated by this fact, we investigate data structures having the XOR/XNOR functions as primitive elements.

Reduced and Ordered Binary Decision Diagrams (ROBDDs) [4] are canonical logic representation forms that are easy to manipulate and also compact for most functions. The compactness of ROBDDs inspired several works on *One-Pass Synthesis* (OPS) [5], i.e., a synthesis methodology where logic optimization and technology mapping tasks are carried out directly on the ROBDD structure. To further increase the ROBDDs compactness, and consequently enhance the efficiency of the corresponding OPS, several BDD extensions have been proposed [6]–[9]. Among them, *EQuational BDDs* (EQ-BDDs) [7] natively support the XNOR operation and therefore are well suited to directly synthesize XOR-rich logic circuits. Unfortunately, EQ-BDDs as proposed in [7] are not canonical and therefore lose some of the desirable properties of BDDs for logic synthesis.

In this paper, we present *Biconditional BDDs* (BBDDs): a new canonical BDD extension for efficient logic synthesis targeting XOR-intensive circuits. Inspired by the EQ-BDDs concept in [7], we construct BBDDs using a different logic expansion, the *biconditional expansion*, rather than the standard Shannon's expansion used in BDDs [4]. Novel reduction and ordering rules make *Reduced and Ordered BBDDs* (ROBBDDs) canonical and very compact. Experimental results show that ROBBDDs have 37% fewer nodes on average as compared to traditional ROBDDs.

Emerging XOR-efficient devices, such as ambipolar transistors [10]–[12], can take advantage of the representation compactness thanks to direct mapping onto BBDD structures. To exploit this opportunity, we present in this paper a BBDD-based *One Pass Synthesis* method. While we focus on the application of the BBDD-based OPS to ambipolar technology, the proposed synthesis method also supports standard CMOS technology. Experimental results show that the BBDD-based OPS produces logic circuits, based on ambipolar transistors, with 49.7% fewer devices on average compared to state-of-art commercial synthesis tool. Considering CMOS technology, the average transistor count reduction of the proposed synthesis method with respect to commercial synthesis tool is 31.5%.

The remainder of this paper is organized as follows. Section II provides a background on ambipolar technology and BDDs. In Section III, *Biconditional BDDs* are introduced with corresponding variable ordering and reduction rules. Section IV presents a *One-Pass Synthesis* methodology based on ROBBDDs targeting XOR-rich functions. Then, in Section V, experimental results for the proposed synthesis methodology are presented and compared to state-of-art commercial synthesis tool. The paper is concluded in Section VI.

II. Background and Motivation

This section provides relevant background on ambipolar transistors and binary decision diagrams.

A. Ambipolar Transistors

Ambipolar transistors are *Double-Independent Gate* (DIG) *Field Effect Transistors* (FETs) having one gate controlling on-line the device polarity (Fig. 1(a)). Ambipolar transistors have been experimentally fabricated in several novel technologies, such as carbon nanotubes [10], graphene [11] and *Silicon NanoWires* (SiNWs) [12]. The on-line configuration of DIG ambipolar FETs polarity is enabled by the regulation of *Schottky Barriers* on source/drain junctions through the additional gate. Logic gates based on ambipolar devices can implement XOR-based functions with low physical resources. In Fig. 1(b), the full-swing XNOR-2 (A⊙B) ambipolar gate proposed in [13] is reported. The XNOR ambipolar implementation requires 4 transistors while the traditional *full-swing static* CMOS implementation uses 8 transistors [14]. Note that by swapping $V_{cc} \rightleftharpoons B/A$ and $V_{ss} \rightleftharpoons C$ in the XNOR-2 in Fig. 1(b), it is possible to obtain the majority function MAJ_3(A,B,C) as depicted in Fig. 1(c). The CMOS counterpart of the majority function requires 2.5x more transistors than in the proposed implementation. Thanks to their improved expressive power, ambipolar transistors can be directly mapped in XOR-based logic representation forms enabling compact realizations for XOR-dominated circuits.

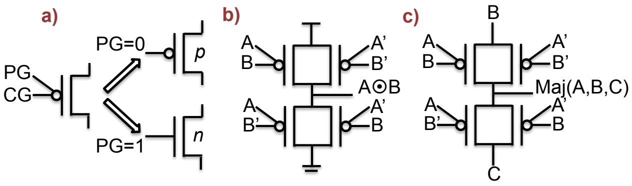

Fig. 1: Polarity control in double-gate ambipolar transistors (a), XNOR-2 gate (b), 3-input majority gate (c).

B. Binary Decision Diagrams

BDDs were first introduced by Lee [2] and Akers [3]. The notions of ordering and reduction of BDDs were introduced by Bryant in [4], where it was shown that, with these restrictions, BDDs are a canonical logic representation form. Canonical *Reduced and Ordered BDDs* are often compact[1] and easy to manipulate, and are therefore widely used in electronic design automation and in other fields. Several BDD extensions have been proposed in literature to enhance the representation compactness [6]–[9]. Among them, we describe here only *Transformation BDDs* (TBDDs) [8], [9] and *Equational BDDs* (EQ-BDDs) [7] as they are the closest to our BDD extension and useful to understand the following material.

The core idea of TBDDs is the transformation of a function into another function in a new domain using an injective mapping (e.g., linear XNOR transformation), and then represent the new function with a standard ROBDD. TBDDs are canonical since the injective mapping operation maintains the unique representation of objects with ROBDD. Despite TBDDs preserve canonicity and potentially improve compactness, other operations on BDD become more complex due to the domain transformation, e.g., variable ordering. EQ-BDD proposed in [7], are a BDD extension where the *If-Then-Else* (ITE) formula of a *non-terminal* node contains an equation in place of a single variable. EQ-BDDs considered in [7] have a branching condition with equation $x = y$, where x and y are input variables. Unfortunately, the ordering conditions proposed in [7] leads to a non-canonical representation of reduced and ordered EQ-BDDs. The input variable ordering represents a major limitation regarding canonicity for EQ-BDDs and dynamic reordering for TBDDs.

To address the aforementioned issues, we propose a new BDD extension that intrinsically embeds the XNOR operation while maintaining canonicity and simple ordering conditions.

III. Biconditional Binary Decision Diagrams

This section introduces *Biconditional Binary Decision Diagrams* (BBDDs). First, it presents the core logic expansion that drives BBDDs. Then, it gives ordering and reduction rules that makes *Reduced and Ordered BBDDs* (ROBBDDs) canonical.

A. Biconditional Expansion

In standard *Binary Decision Diagrams* (BDDs), each *non-terminal* node represents a Shannon's expansion:

$$f(x,y,..,z) = x\cdot f(1,y,..,z) + x'\cdot f(0,y,..,z) \quad (1)$$

In BBDDs, the Shannon's expansion is replaced by the *biconditional expansion*:

$$f(x,y,..,z) = (x \oplus y)\cdot f(y',y,..,z) + (x \odot y)\cdot f(y,y,..,z) \quad (2)$$

with $\oplus$ and $\odot$ representing the XOR and XNOR operations, respectively. Note that the *biconditional expansion* is a special case of the (x_i,p)-decomposition in [15] that extends the Shannon's expansion. As per the *biconditional expansion* in Equation 2, only functions that have two or more variables can be decomposed. Indeed, in single variable functions, the terms $(x \oplus y)$ and $(x \odot y)$ cannot be computed. In such a condition, the *biconditional expansion* of a single variable function can reduce to a Shannon's expansion by fixing the second variable y to logic 1. With this boundary condition, any Boolean function can be fully represented in terms of *biconditional expansions*.

B. BBDD Structure and Ordering

A *Biconditional Binary Decision Diagram* (BBDD) is a BDD driven by the *biconditional expansion* in place of Shannon's expansion. Each non-terminal node in a BBDD has the branching condition *biconditional* on two variables. We call these two variables the *Primary Variable* (PV) and the *Secondary Variable* (SV).

[1]Note that some logic circuits have not a compact representation with ROBDDs, e.g., for multipliers, the ROBDD size depends exponentially on the number of inputs.

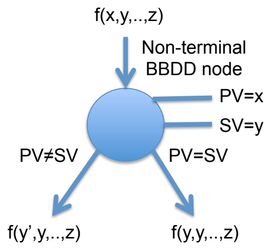

Fig. 2: BBDD *non-terminal* node.

An example of a BBDD *non-terminal* node is provided by Fig. 2. We refer hereafter to $PV \neq SV$ and $PV = SV$ edges in a BBDD node simply as $\neq$*-edges* and $=$*-edges*, respectively.

To achieve *Ordered BBDDs* (OBBDDs), a variable order must be imposed for PVs and a rule for the other variables assignment must be provided. We propose the following *Chain Variable Order* (CVO) to address this task. Given a Boolean function f and an order $\pi = (\pi_0, \pi_1, .., \pi_{n-1})$ for the support variables of f, the CVO assigns PVs and SVs as:

$$\begin{cases} PV_i = \pi_i \\ SV_i = \pi_{i+1} \end{cases} \text{with } i = 0,1,..,n-2; \begin{cases} PV_{n-1} = \pi_{n-1} \\ SV_{n-1} = 1 \end{cases} \quad (3)$$

Note that if we swap $\pi_i \rightleftharpoons \pi_j$ in the initial order π, due to some reordering operation, this simply translates through the CVO as PV$_i$ $\rightleftharpoons$PV$_j$ and SV$_{i-1}$ $\rightleftharpoons$SV$_{j-1}$.

<u>*CVO Example*</u>: From $\pi = (\pi_0, \pi_1, \pi_2)$, the corresponding CVO ordering is obtained by the following method. First, $PV_0 = \pi_0$, $PV_1 = \pi_1$ and $SV_0 = \pi_1$, $SV_1 = \pi_2$ are assigned. Then, the final boundary conditions of Equation 3 are applied as $PV_2 = \pi_2$ and $SV_2 = 1$. The consecutive ordering by couples (PV$_i$, SV$_i$) is thus $((\pi_0, \pi_1), (\pi_1, \pi_2), (\pi_2, 1))$.

The *Chain Variable Order* (CVO) is a key factor enabling unique representation of ordered biconditional decision structures. For the sake of clarity, we first consider the effect of the CVO on *complete*[2] *Binary Biconditional Decision Trees* (BBDTs) (*i.e.* with no sharing between nodes and thus an exponential size) and then we move to generic reduced BBDDs.

<u>*Lemma 1:*</u> For a given Boolean function f and a variable order π, there exists only one *complete* BBDT ordered with the CVO.
<u>*Proof:*</u> Each couple (PV$_i \leftarrow \pi_i$, SV$_i \leftarrow \pi_{i+1}$) assigned by the CVO to a BBDT node can be seen as a single variable $\mu_i = \pi_i \odot \pi_{i+1}$ of a corresponding *Binary Decision Tree* (BDT) node. Note that $\pi \in \mathbb{B}^n \rightarrow \mu \in \mathbb{B}^n$ is a bijective transformation since (i) $\pi \neq \pi' \Rightarrow \mu \neq \mu'$ and (ii) π and μ have the same number of elements. Then, a *complete Ordered BBDT* (OBBDT) is equivalent to a Bijective Transformation *complete Ordered BDT* (OBDT) [9]. Since *complete* OBDTs are unique for a given function, it follows that also *complete* OBBDTs are a unique logic representation form. *Q.E.D.*

We refer hereafter to ordered binary biconditional decision structures as BBDDs ordered by the CVO.

C. BBDD Reduction

In order to improve the representation efficiency, OBBDDs should be reduced according to a set of rules. The straightforward extension of OBDD reduction rules [4] to OBBDDs, leads to *weak Reduced OBBDDs* (ROBBDDs). This kind of reduction is called *weak* due to the partial exploitation of OBBDD reduction opportunities. A *weak* ROBBDD is an OBBDD respecting the two following rules:

R1) It contains no isomorphic subgraphs.
R2) It contains no nodes with isomorphic children.

[2]We refer to a complete tree as a full binary tree in which all leaves are at the same level, and in which every parent has two children.

In addition, the OBBDD representation exhibits supplementary interesting features enabling further reduction opportunities. First, levels with no nodes may occur in OBBDDs for XOR-intensive functions. Such levels must be removed to compact the original OBBDD. Second, subgraphs that represent functions of a single variable must be collapsed into a single BDD node. These two additional reductions lead to *strong* ROBBDDs. Formally, a *strong* ROBBDD is an OBBDD respecting R1 and R2 rules and in addition:

R3) It contains no empty levels.

R4) Subgraphs representing functions of a single variable are collapsed into a single BDD node.

Weak and *strong* reduced OBBDDs are canonical, as per:

Lemma 2: For a given Boolean function f and a variable order π, there exists only one *weak* ROBBDD.
Proof: *Weak* ROBBDDs share the same reduction rules than ROBDDs. The proof of *Lemma 1* is valid using Bijective Transformation ROBDDs that are equivalent to *weak* ROBBDDs. *Q.E.D.*

Theorem 1: A *strong* ROBBDD is a canonical representation for any Boolean function f.
Proof: *Strong* ROBBDDs can be derived by *weak* ROBBDDs applying reduction rules R3 and R4. Reduction rules R3 and R4 are injective operations as they produce unique and distinct results. *Strong* ROBBDDs are canonical since they can be obtained applying injective operations to canonical *weak* ROBBDDs. *Q.E.D.*

Unless specified otherwise, we refer hereafter to ROBBDDs as *strong* ROBBDDs.

IV. One-Pass Logic Synthesis

One Pass Synthesis (OPS) [5] is a logic synthesis methodology where logic optimization and technology mapping phases are combined in a single step carried out through a common data structure, e.g., BDDs. In this work, we use *Biconditional BDDs* (BBDDs) as data structure supporting OPS targeting XOR-rich functions.

In BBDD-based OPS, logic optimization corresponds to the *strong* ROBBDD construction. Note that most of the algorithms for ROBDDs construction, e.g., BUILD, APPLY [4] *etc*., can be adapted to ROBBDD, hence to support the *biconditional expansion* in place of Shannon's expansion. Standard dynamic variable reordering algorithms can be applied also with the *Chain Variable Order* (CVO).

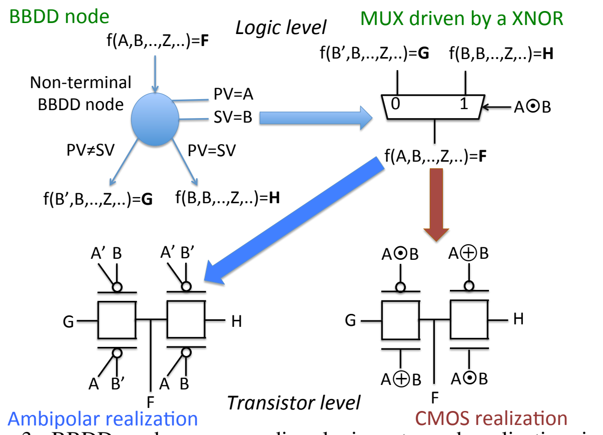

Fig. 3: BBDD node corresponding logic gate and realization in ambipolar and CMOS technologies.

Reduced OBBDDs are a compact logic representation structure. Such compactness can be preserved in the implemented logic circuit by directly mapping ROBBDDs components onto basic logic elements. In particular, a BBDD node is efficiently implemented by a TG-based MUX having the selection signal driven by a XNOR (MUX-$\odot$). In Fig. 3, the MUX-$\odot$ structures for ambipolar and CMOS technologies are depicted.

V. Experimental Results

In this section, we present experimental results for the proposed BBDD-based *One-Pass Synthesis* (OPS) methodology. Ambipolar and standard CMOS are considered as target technologies. Area results, in terms of device count, are compared with *Synopsys Design Compiler* synthesis tool fed with a reference library.

A. Methodology

Implementation details for the OPS flow and settings for the reference *Design Compiler* flow are presented.

1) BBDD-based OPS: The proposed OPS methodology consists of logic optimization and technology mapping operations both carried out on a reduced and ordered BBDD.

Logic Optimization: Logic optimization in BBDD-based OPS is equivalent to ROBBDD construction. In this work, *weak* and *strong* ROBBDDs are considered for this purpose. We showcase *strong* ROBBDD potential compactness for majority and adder functions via custom construction algorithms written in C language. Since an integrated decision diagram package for generic function *strong* ROBBDD construction is not available at the current time, we use *weak* ROBBDDs to have a first insight about automated BBDD-based OPS potential. Indeed, we automatically generate *weak* ROBBDDs through the procedure described in the proof of *Lemma 2* that is implemented on top of ABC [16] software.

Technology Mapping: Technology mapping in BBDD-based OPS consists of the direct assignment of MUX-$\odot$ structures to BBDD nodes. This is accomplished by a program written in C language.

2) Buffering in OPS: Circuits synthesized by OPS require buffering to avoid excessive series stacking of transistors. For this reason, we insert a buffer every 4 cascaded MUX-$\odot$ to limit the maximum stack to 4, similarly to traditional standard-cell approach.

3) Reference Synthesis Flow: For comparison purposes, a reference library has been built for *Design Compiler* with INV, XOR-{2,3}, XNOR-{2,3}, MAJ_3, NAND-{2,3,4}, NOR-{2,3,4}, AOI/OAI and generalized AOI/OAI [13] standard cells. No timing information is provided in the reference library to prevent area/timing trade-off. The given area information is the transistors count for the implementation of each cell in static complementary style. The command used for logic synthesis is *compile -area_effort high*.

B. Results and Discussion

Experimental results are shown in Table I. Adders and majority benchmarks are manually created. Other benchmarks are arithmetic based circuits selected from the MCNC suite. We first introduce results about the ROBBDD representation compactness. Then, we present synthesis results for the BBDD-based OPS considering ambipolar and CMOS technologies.

1) ROBBDD Compactness: The size of *strong* ROBBDDs for majority and adders functions is about 50% less than their corresponding ROBDDs. On the other hand, *weak* ROBBDDs, automatically generated for the MCNC benchmarks, have 28% fewer nodes compared to standard ROBDDs. On average, ROBBDDs reduce by 37% the number of nodes with respect to their ROBDDs counterpart.

TABLE I: Experimental results for BBDD-based OPS and comparison with Synopsys DC

		Input, Output and Node numbers				Ambipolar Technology			CMOS Technology		
						OPS no buff.	OPS with buff.	DC	OPS no buff.	OPS with buff.	DC
OBBDD Reduction	Benchmarks	In	Out	|BBDD|	|BDD|	Transistors count			Transistors count		
Strong	Adder 8	16	17	24	40	110	114	226	224	228	328
Strong	Adder 16	32	33	48	80	222	230	498	450	456	674
Strong	Adder 32	64	65	96	160	442	458	1042	898	914	1378
Strong	Adder 64	128	129	192	320	894	926	2130	1802	1834	2786
Strong	Adder 128	256	257	384	640	1790	1854	4306	3574	3638	5602
Strong	Majority 11	11	1	25	38	98	122	234	248	272	306
Strong	Majority 13	13	1	33	51	130	162	320	308	340	472
Strong	Majority 15	15	1	42	66	166	198	302	372	436	378
Strong	Majority 17	17	1	52	83	206	246	460	452	492	648
Strong	Majority 19	19	1	63	102	250	290	534	536	576	662
Weak	xor5	5	1	3	5	24	24	32	36	36	48
Weak	rd53	5	3	14	18	64	72	70	96	108	96
Weak	rd73	7	3	20	32	102	110	112	144	152	152
Weak	parity	16	1	8	16	80	88	106	142	150	180
Weak	9symml	9	1	17	23	105	121	174	194	210	232
Weak	misex1	8	7	28	36	163	179	240	220	236	240
Weak	cordic	23	2	65	82	316	388	494	502	574	526
Weak	f51m	8	8	36	56	186	226	510	256	296	514
Average	-	32.2	29.7	65.7	104.3	303.1	329.0	602.9	580.7	608.2	847.1
Reduction	-	-	-	-37.0%	**100%**	-49.7%	-45.4%	**100%**	-31.4%	-28.2%	**100%**

2) Ambipolar Technology: Considering ambipolar technology, the BBDD-based OPS reduces the average number of devices by 49.7% compared to *Design Compiler*. With buffer insertion, the number of devices increases by less than 5%, therefore maintaining a large advantage over *Design Compiler* also in this condition. For adder and majority benchmarks, the BBDD-based OPS produces the largest improvements thanks to the *strong* reduction of OBBDDs.

3) CMOS Technology: With CMOS technology, the BBDD-based OPS generates logic circuits having 31.5% fewer devices on average compared to *Design Compiler*. The buffer insertion only increases by 3% the transistor count. This is because the area overhead associated with the buffer insertion has less impact in CMOS circuits that have larger transistor count.

4) Ambipolar vs. *CMOS:* The BBDD-based OPS is capable to take the largest advantage of the tunable polarity opportunity offered by ambipolar technology over standard CMOS. Indeed, the BBDD-based OPS for ambipolar technology reduces by 46% the number of devices with respect to its CMOS counterpart. On the other hand, *Design Compiler* for ambipolar technology decreases the transistor count by only 28% compared to its CMOS counterpart. This result confirms that the BBDD-based OPS is well suited to take advantage of XOR-efficient emerging devices at the logic synthesis level.

5) Discussion: Experimental results highlighted the effectiveness of BBDDs employed in one-pass logic synthesis for XOR-rich functions targeting both ambipolar and standard CMOS technologies. The *strong* ROBBDD construction algorithms employed for majority and adder functions, fully exploit the BBDD advantage in the corresponding OPS. On the other hand, results for general MCNC benchmarks can be further improved by creating (or adapting) a decision diagram package for BBDDs. In this way, the BBDD construction leads to a *strong* reduced OBBDD rather than to a *weak* reduced structure and dynamic variable re-ordering can be applied to further reduce the representation size. Even though this work focuses on area-minimal results, the number of logic levels, and the associated timing, of an ambipolar circuit synthesized on a monolithic ROBBDD is predictable. Indeed, given the number of inputs n_I and a buffering condition every k BBDD nodes, the number of logic levels n_L is bounded as: $\frac{n_I}{2k} \leq n_L \leq \frac{n_I}{k}$. The lower bound happens if reduction rule R3 apply for half inputs, e.g., parity functions. The upper bound happens if reduction rule R3 never apply, e.g., majority functions.

VI. Conclusions

We proposed a new canonical BDD extension, the *Biconditional BDD* (BBDD), capable to efficiently support *One-Pass Synthesis* (OPS) for XOR-rich logic circuits. BBDDs are based on the *biconditional expansion* that natively embed the XOR/XNOR operations. We directly synthesize XOR-intensive circuits on a *Reduced and Ordered BBDD* (ROBBDD) using a one-pass synthesis methodology. In this context, the quality of the synthesized circuit depends on the ROBBDD compactness. Ordering and reduction rules make the ROBBDD construction efficient and unique. Experimental results show that ROBBDDs have on average 37% fewer nodes compared to standard ROBDDs. CMOS circuits directly synthesized on ROBBDDs have 31.5% fewer transistors compared to the same circuits synthesized by *Design Compiler*. Considering ambipolar devices, the direct synthesis on ROBBDDs reduces by 49.7% the devices count with respect to logic circuits synthesized by *Design Compiler*.

References

[1] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, New York, 1994.

[2] C.Y. Lee, *Representation of Switching Circuits by Binary-Decision Programs*, Bell Systems Technical Journal, 1959.

[3] S.B. Akers, *Binary Decision Diagrams*, IEEE Trans. Comp., C-27(6):509-516, June 1978.

[4] R.E. Bryant, *Graph-based algorithms for Boolean function manipulation*, IEEE Trans. Comput., C-35: 677-691, 1986.

[5] R. Drechsler, W. Gunther, *Toward One-Pass Synthesis*, Springer, 2002.

[6] I. Wegener, *Branching Programs and Binary Decision Diagrams: Theory and Applications*, SIAM, 2000.

[7] J.F. Groote, J. Van de Pol, *Equational Binary Decision Diagrams*, Proc. LPAR, pp.161-178, 2000.

[8] C. Meinel, F. Somenzi, T. Theobald, *Linear Sifting of Decision Diagrams*, Proc. DAC, pp. 202-207, 1997.

[9] E.I. Goldberg, Y. Kukimoto, R.K. Brayton, *Canonical TBDD's and Their Application to Combinational Verification*, Proc. IWLS, 1997.

[10] Y. Lin *et al.*, *High-Performance Carbon Nanotube Field-Effect Transistor with Tunable Polarities*, IEEE Trans. Nanotech., 4(5): 481-489, 2005.

[11] N. Harada *et al.*, *A polarity-controllable graphene inverter*, Applied Physics Letters, 96(1): 012102 - 012102-3, 2010.

[12] M. De Marchi *et al.*, *Polarity control in Double-Gate, Gate-All-Around Vertically Stacked Silicon Nanowire FETs*, Proc. IEDM, 2012.

[13] M.H. Ben-Jamaa *et al.*, *An Efficient Gate Library for Ambipolar CNTFET Logic*, IEEE Trans. CAD, 30(2): 242-255, February 2011.

[14] J.M. Rabaey *et al.*, *Digital Integrated Circuits*, Prentice Hall, 2003

[15] A. Bernasconi *et al.*, *On Decomposing Boolean Functions via Extended Cofactoring*, Proc. DATE, pp. 1464-1469, 2009.

[16] ABC Logic Synthesis Tool [Avilable online].