Mod–2–OBDD's
A Generalization of OBDD's and EXOR–Sum–of–Products *

Jordan Gergov, Christoph Meinel
FB IV – Informatik
Universität Trier
D – 54 286 Trier
Germany

Abstract

We present a data structure — the Mod–2–OBDD's that considerably extend OBDD's (ordered binary decision diagrams) as well as ESOP's (EXOR–sum–of–products). Many Boolean function of practical interest like hidden weighted bit function, indirect storage access function, important symmetric functions have exponential size optimal OBDD's and/or ESOP's (even multilevel EXOR–expressions) but (low degree) polynomial size Mod–2–OBDD's.
The manipulation of Mod–2–OBDD's is at least as efficient as the manipulation of OBDD's. Apply operation, quantification, composition [Bry86] have the same complexity as in the case of OBDD's. Moreover, since the size of a minimal Mod–2–OBDD–representation of a Boolean function is, in general, smaller than the size of its optimal OBDD–representation manipulation is more efficient. Moreover, EXOR–operation and complementation can be performed much better — namely in $O(1)$ *time.*
However, the price of $O(1)$ *time EXOR–apply operations is the canonicity of the Mod–2–OBDD–representation. A Boolean function can have different reduced Mod–2–OBDD's of the same variable ordering similar to the case of mixed polarity RME's. In order to allow in spite of this fact efficient analysis of Mod–2–OBDD we present a probabilistical equivalence test (one–sided bounded error probability) for ESOP's and Mod–2–OBDD's which performs only linear many arithmetical operations.*
A Mod–2–OBDD–package is in preparation.

*Proc. *IFIP-Workshop on Application of Reed-Muller Expansion in Circuit Design*, TR WSI-93-2, 170–175, Univ. Tübingen, 1993

1 Introduction

Mod–2–representations of switching functions, known as Reed–Muller Expansions (RME) or EXOR–sum–of–products (ESOP) are considered already by Reed [Ree54] and Muller [Mul54]. These representations are easily testable [Red72, SR75, DB93] (e.g. the change of any input to an EXOR gate always propagates to the output). There are also some advantages in realizing symmetric and arithmetical functions by ESOP's [SB90, Sau92].

In the recent years various software tools and algorithms were developed for optimizing EXOR–expressions [FTY87, PCSS92, Sas92]. The increasing interest in RME's is caused in particular by the look–up table field programmable gate arrays (FPGA) where EXOR–gates do not produce extra costs in terms of chip area [BFRV92, LS93].

Recently, a compact graph-based representation of RME's , the ordered FDD's, has been introduced [KSR92, KR93]. The FDD's provide canonical form similarly to the OBDD's [KSR92, Bry86]. The application of binary Boolean operations on FDD's requires time proportional to the number of paths in the input FDD's. It was proven that certain operations, e.g. EXOR, can be performed in at most quadratic time [BD93]. However, all known algorithms for the AND operation have an exponential worst case behavior. It was shown that the concept of ordered FDD's can be extended to free FDD's, too [BD93].

OBDD's provide canonical representations for Boolean functions (and, hence, efficient satisfiability and equivalence tests) and allow efficient (quadratic time) applications of binary Boolean operations (the *apply* operation) [Bry86]. These algorithmic properties are the fundamental reason for the success of OBDD applications such as, for instance, in sequential circuit verification [e.g CMB90, BCMD90, Fil91, Eve91], in testing [e.g. Bec92, KBS93], or in logic optimization and multi-level synthesis [MWBS88, Kar89]. For an excellent survey see [Bry92].

Here, we present a powerful general purpose data structure for representing Boolean functions — the Mod–2–OBDD's ($\oplus$OBDD). Mod–2–OBDD's, as an extension of OBDD's as well as of ESOP's, preserve the OBDD property of efficient manipulation. Apply operation, quantification, composition [Bry86] have the same complexity as in the case of OBDD's. Further, EXOR, logical equivalence and complementation can be performed even better — namely in $O(1)$ time. That is why Mod–2–OBDD's are very suitable for describing the behavior of circuits containing many EXOR–gates. This algorithmic property can be used, for instance, to perform efficiently probabilistical verification in the same sense as in [JBAF92]. Since the Mod–2–OBDD's do not provide canonical form we describe an efficient probabilistical equivalence test which requires linear many arithmetical operations. The same test can be used to check the equivalence of general ESOP's.

Mod–2–OBDD's are much more powerful than OBDD's and ESOP's. Any Boolean function which can be represented efficiently by OBDD's or ESOP's can be represented efficiently by Mod–2–OBDD's but the converse is not true. There are Boolean function of practical interest like the hidden weighted bit function [Bry91], the indirect storage access function [BHR91], important symmetric functions which have exponential size optimal OBDD's and/or ESOP's (even multilevel EXOR–expressions) [Raz87] but polynomial size Mod–2–OBDD's. The succinctness of Mod–2–OBDD–representations can be deduced also from the fact that Mod–2–OBDD's are closer to circuits than OBDD's. We can consider Mod–2–OBDD's as special circuits build on from multiplexors and EXOR–gates.

A Mod–2–OBDD–package is in preparation. Most of the programming techniques used e.g. in the OBDD–package developed by Brace, Rudell and Bryant [BRB90] can be applied also in the case of Mod–2–OBDD's.

For the sake of space almost all proofs are omitted.

2 The Computational Power of Mod–2–OBDD's Compared with OBDD's and Mod–2–Expressions

Let us first define Mod–2–OBDD's.

Definition (Mod–2–OBDD's.) *Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables, and let π be an ordering of X_n. A* Mod–2–OBDD *over X_n is a directed acyclic connected graph P where each node has outdegree 2 or 0. There is a distinguished node, the source, which has indegree 0, and at most two nodes of outdegree 0, the 0-sink and the 1-sink, which are labeled by the Boolean constants 0 and 1, respectively. The remaining nodes v are labeled either by Boolean variables $x \in X_n$ or by the binary operation $\oplus$ (EXOR). In the following, we denote by $l(v)$ and $size(P)$ the label of v and the number of inner nodes of P. The two edges starting in a non-sink node are labeled by 0 and 1. The 0–successor and the 1–successor nodes of v are denoted by $v0$ and $v1$, respectively. If v_2 is a successor of v_1 in P and if $l(v_1), l(v_2) \in X$ then $l(v_1) < l(v_2)$ according to π.*

In order to illustrate the concept of Mod–2–OBDD's let us consider the following example. Let $f \in \mathbb{B}_3$ be defined by

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

and let π be the natural order on the set of variables, $\pi = (x_1 < x_2 < x_3)$. Then f can be represented by the Mod–2–OBDD cited in Figure 1.

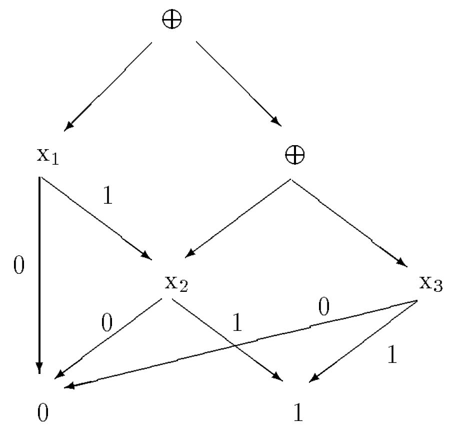

Figure 1. *Mod–2–OBDD computing f.*

If an input assignment is given then the Boolean values assigned to the sinks of a Mod–2–OBDD P extend to Boolean values associated with all nodes of P

in the following way: If both successor nodes v_0, v_1 of a node v of P carry the Boolean values δ_0, δ_1, and if $l(v) = x_i$ then we associate with v the value δ_0 or δ_1 according to $x_i = 0$ or $x_i = 1$. If $l(v) = \oplus$ then we associate with v the value $\oplus(\delta_0, \delta_1) = (\delta_0 + \delta_1)$ *mod* 2. P *computes* the value 1 (0) on an input $a \in \{0,1\}^n$ if the source of P associates with 1 (0) under a. The value $f_P(a)$ of the Boolean function f_P represented by a Mod–2–OBDD P can be computed easily in time $O(size(P))$.

Each Boolean function can be represented by means of Mod–2–OBDD's. For instance, using the graph–based representation of the Reed–Muller expansion developed by Kebschull and Rosenstiel [KR92] we can derive a Mod–2–OBDD–representation of a Boolean function f in the following way: We start with a tree–like OBDD whose sinks are labeled by 1 if the monom consisting of all variables that are tested positively on the path from the source to that sink appears in the Reed–Muller expansion. All other sinks are labeled by 0. Reducing this program similar as OBDD's [Bry86] we obtain a Mod–2–OBDD P if

- we replace each 1–edge starting at an inner node v by a $\oplus$–node that is connected to v and the 0–successor $v0$ of v, and if
- we replace each 0–edge by a 0–edge that leads to a 0–sink.

It is easy to check that P is a Mod–2–OBDD that computes f.

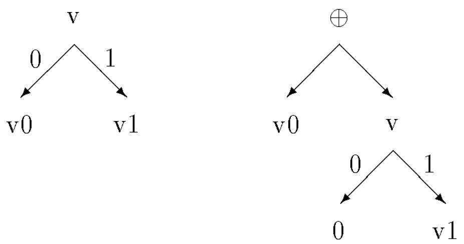

Figure 2. *Transformation of FDD (i.e. a graph-based description of RME) to Mod–2–OBDD.*

Before giving evidence why Mod–2–OBDD's beat OBDD's as well ESOP's let us only remind that the ESOP's have some advantages over the sum–of–products representations realizing symmetric and arithmetical functions [SB90, Sau92]. For instance, if we consider (sequences of) symmetric functions with polynomial size (n^k) SOP's then $1/2^{2k} \leq$ ESOP/SOP–size ratio ≤ 1 [Dam93]. Since the products in an ESOP as well the EXOR can be simulated, for each variable ordering, by OBDD's, and nodes labeled by $\oplus$, respectively, the function represented by an ESOP can be expressed in the worst case by equal size Mod–2–OBDD. The size of the Mod–2–OBDD can be reduced further using reductions as in the case of OBDD's (see **Section 4**).

Theorem 1.
The size of an optimal Mod–2–OBDD for a given Boolean function f is not greater than the minimum of the size of an optimal OBDD and an optimal ESOP of f. There exist Boolean functions with small (low degree polynomial) Mod–2–OBDD's whose optimal OBDD's or ESOP's are of exponential size. ■

It is easy to show that many interesting Boolean functions can be represented by small (linear or quadratic size, even for each variable ordering) Mod–2–OBDD's but need, for principal reasons, exponential size ESOP representations. For a lack of space we consider here only one example — the MAJORITY which needs always exponential size multilevel and/exor expression (unbounded fan–in and constant number of levels) [Raz87]. MAJORITY gives one as output if the number of input assignments having the value one is greater than the number of input assignments having the value zero. For each variable ordering we can construct an OBDD of quadratic size which counts the number of ones. Therefore MAJORITY can be represented efficiently in terms of Mod–2–OBDD's.

An OBDD is always a Mod–2–OBDD. In order to transform a Mod–2–OBDD to an OBDD, we have to eliminate all $\oplus$–nodes. A straightforward approach for doing this is based on the *apply* procedure of Bryant [Bry86]. However, as the next examples show this technique can lead to an exponential blow–up.

In the following, we explain how to design quadratic Mod–2–OBDD's for the hidden weighted bit function [Dam93]. If $wt(x)$ is the number of ones in the input assignment $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$, and if for simplicity x_0 denotes 0 then the *hidden weighted bit function* HWB [Bry91] is defined via the equality $HWB(x) = x_{wt(x)}$. In [Bry91], it was proven that each OBDD–representation of HWB is of exponential size. Let us show that for each variable ordering π there is a Mod–2–OBDD for HWB of at most cubic size. The equality $HWB(x) = \bigoplus_{k=1}^{n} x_k \wedge E_k(x)$ where $E_k(x)$ equals one if x contains exactly k ones can easily be verified. Since, for each variable ordering π $x_k \wedge E_k$ can be represented by an OBDD of at most quadratic size the above equality can be transformed easily into a cubic size Mod–2–OBDD of HWB.

There exist functions which possess moderate Mod–2–OBDD's but each FBDD (free binary decision diagram, for definition and basic properties see [GM92]) of them is of exponential size. The *clique*–function $\oplus$–$cl_{n,3}(x)$, $x = (x_{i,j})$ where $1 \leq i < j \leq n$) decides whether a given undirected n–node graph $G = G(x)$ contains an odd number of triangles (3–cliques). It was proven that any FBDD that computes $\oplus$–$cl_{n,3}$ is of exponential size [A&86]. This result immediately implies an exponential lower bound for any OBDD since OBDD's are in particular FBDD's. A cubic size Mod–2–OBDD for $\oplus$–$cl_{n,3}$ of any variable ordering can be constructed straightforward [e.g. Mei89].

3 Efficient Manipulation of Mod–2–OBDD's

The "graceful degradation" property of OBDD's (i.e. the quadratic time *apply* operation) is essential for almost all applications [Bry92]. Here we show that Mod–2–OBDD's preserve this advantage of the OBDD–based data structures. The following theorem is a consequence of results in [GM93].

Theorem 2.
Let P_1 and P_2 be two Mod–2–OBDD's of the variable ordering π, and let $*$ be a binary Boolean operation.

1. *A Mod–2–OBDD P of the variable ordering π with $f_P = f_{P_1} * f_{P_2}$ can be constructed in time $O(size(P_1) \cdot size(P_2))$;*
2. *If $* \in \{\oplus, \equiv\}$ then a Mod–2–OBDD P of the variable ordering π with $f_P = f_{P_1} * f_{P_2}$ can be constructed in time $O(1)$.* ∎

Since the restriction of a Mod–2–OBDD to a partial assignment of the input variables can be computed easily in linear time we obtain the following corollary.

Corollary 3.
Let P_1 and P_2 be two Mod–2–OBDD's of the variable ordering π, and let x be an input variable. Then, a Mod–2–OBDD of the variable ordering π for

1. **(Quantification)** *$\exists x f_{P_1}$ and $\forall x f_{P_1}$ can be computed in time $O(size(P_1)^2)$;*
2. **(Composition)** *$f_{P_1}|_{x \leftarrow f_{P_2}}$ can be computed in time $O(size(P_1)^2 \cdot size(P_2))$; and*
3. **(Complementation)** *the complement of P_1 can be constructed in $O(1)$ time.* ∎

4 Efficient Equivalence Test for Mod–2–Sum–of–Products and Mod–2–OBDD's

Mod–2–OBDD's can be reduced in the same way as OBDD's. We need reduction rules for $\oplus$ nodes additionally to the OBDD reduction rules [e.g. Bry92] The *merging rule* restricted to $\oplus$–nodes consists in the merging of two $\oplus$–nodes v and w such that $\{v0, v1\} = \{w0, w1\}$. The *deletion rule* restricted to $\oplus$–nodes is the replacement of a $\oplus$-node v such that $v0 = v1$ by the zero sink. The reduction rules can be further extended using complemented edges similarly to [MIY90]. It can be easily proved that the reduced diagram is uniquely determined and can be computed efficiently. However, Mod–2–OBDD's do not build a canonical representation and the reduced diagram can not help us to solve the equivalence problem. Nevertheless, we can prove the following theorem.

Theorem 4.
Let P_1, P_2 be two Mod–2–OBDD's, two ESOP's or a Mod–2–OBDD and an ESOP over $X_n = \{x_1, x_2, \ldots, x_n\}$. There is an $O((size(P_1)+size(P_2)) \cdot log(n) \cdot log^2(n) \cdot log^3(n))$ time (i.e. linear many arithmetical operations) probabilistical equivalence test for the equivalence of the Boolean functions represented by P_1 and P_2. If P_1 and P_2 compute the same Boolean function the test gives always the answer "yes". If P_1 and P_2 are not equivalent the error probability of the test is bounded by 1/2. ∎

Proof.
Here, we give only a brief sketch of the algorithm in the case of ESOP's. Let $m \geq log(n) + 1$ be a natural number.

Algorithm.
Input: Two ESOP's $P_1 = \bigoplus_{i \in I_1} m_{1,i}$ and $P_2 = \bigoplus_{i \in I_2} m_{2,i}$ where $m_{j,i}$ denote the corresponding (AND–) monoms.

1. Generate independently $a_1, a_2, \ldots, a_n$ uniformly distributed over $GF(2^m)$;
2. Compute $d := (\sum_{i \in I_1} m_{1,i}(a_1, \ldots, a_n)) - (\sum_{i \in I_2} m_{2,i}(a_1, \ldots, a_n))$ in $GF(2^m)$;
3. If $d = 0$ then "YES" else "NO". ∎

5 References

[A&86] M. Ajtai, L. Babai, P. Hajnal, J. Komlos, P. Pudlak, V. R"odl, E. Szemeredi, G. Turan: Two Lower Bounds for Branching programs, Proc. 18th ACM STOC, 30–38, 1986.

[BCMD90a] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill: Symbolic Model Checking: 10^{20} states and beyond, Proc. of 5th IEEE Symposium on Logic in Computer Science, (Philadelphia, June), 428–439, 1990.

[BCMD90b] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill: Sequential Circuit Verification Using Symbolic Model Checking, Proc. of 27th ACM/IEEE Design Automation Conference (Orlando, June), 46–51, 1990.

[BD93] B. Becker, R. Drechsler: On the Computational Power of Functional Decision Diagrams, Interner Bericht 5/93, Universität Frankfurt, 1993.

[Bec92] B. Becker: Synthesis for Testability: Binary Decision Diagrams, Proc. of 9th Annual Symposium on Theoretical Aspects of Computer Science (Feb.), Lecture Notes in Computer Science 577, 501–512, 1992.

[BFRV92] S. D. Brown, R. J. Francis, J. Rose, Z. G. Vranesic: Field-Programmable Gate Arrays, Kluwer Academic Publisher, 1992.

[BHR91] Y. Breitbart, H. B. Hunt III, D. Rosenkrantz: The Size of Binary Decision Diagrams Representing Boolean Functions, submitted to Inf. and Comp., 1991.

[BRB90] K. S. Brace, R. L. Rudell, R. E. Bryant: Efficient Implementation of a BDD Package, Proc. of 27th ACM/IEEE Design Automation Conference (Orlando, June), 40–45, 1990.

[Bry86] R. E. Bryant: Graph-Based Algorithms for Boolean Function Manipulation, IEEE Trans. Comput. C-35, 6 (Aug.), 677–691, 1986.

[Bry91] R. E. Bryant: On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Applications to Integer Multiplication, IEEE Trans. Comput. 40, 2 (Feb.), 205–213, 1991.

[Bry92] R. E. Bryant: Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams, ACM Computing Surveys, Vol. 24, No. 3 (Sep.), 293–318, 1992.

[CMB90] O. Coudert, J.-C. Madre, C. Berthet: Verifying Temporal Properties of Sequential Machines without Building Their State Diagrams, Proc. of Computer-Aided Verification (Rutgers, N.J., June), 75–84, 1990.

[Dam93] C. Damm: personal communication, 1993.

[DB93] R. Drechsler, B. Becker: Rapid Prototyping of Fully Testable Multi–Level AND/EXOR Networks, Interner Bericht 4/93, Universität Frankfurt, 1993.

[Eve91] H. Eveking: Verifikation digitaler Systeme, Teubner, Stuttgart, 1991.

[Fil91] T. Filcorn: A Method for Symbolic Verification of Sunchronous Circuits, In Computer Hardware Description Languages and their Applications (Marseilles, Apr.), Proc. of IFIP WG 10.2, Tenth International Symposium Amsterdam, North-Holland, D. Borrione and R. Waxman, Eds., 229–239, 1991.

[FTY87] H. Fleischer, M. Travel, J. Yeager: A Computer Algorithm for Minimizing Reed–Muller Canonical Forms, IEEE Trans. Comput. vol. c–36, 247–250, 1987.

[GM92] J. Gergov, Ch. Meinel: Efficient Boolean Manipulation with OBDD's Can be Extended to FBDD's, to appear in IEEE Transactions on Computers, July 1992.

[GM93] J. Gergov, Ch. Meinel: Frontiers of Feasible and Probabilistic Feasible Boolean Manipulation with Branching Programs, Proc. of 10th Annual Symposium on Theoretical Aspects of Computer Science (Feb.), Lecture Notes in Computer Science 665, 576–585, 1993.

[JBAF92] J. Jain, J. Bitner, J. Abraham, D. S. Fussell: Functional Partitioning for Verification and Related Problems, Proc. Brown/MIT VLSI Conference (March), 210-226, 1992.

[Kar89] K. Karplus: Using if-then-else DAGs for Multi-Level Logic Minimization, Proc. of Advance Research in VLSI, C. Seitz Ed., MIT Press, Cambridge, Mass., 101–118, 1989.

[KBS93] R. Krieger, B. Becker, R. Sinković: A BDD-based Algorithm for Computation of Exact Fault Detection Probabilities, to appear in International Symposium on Fault-Tolerant Computing, 1993.

[KR93] U. Kebschull, W. Rosenstiel: Efficient Graph-Based Computation and Manipulation of Functional Decision Diagrams, Proceedings of EDAC'93, 278–282, 1993.

[KSR92] U. Kebschull, E. Shubert, W. Rosenstiel: Multilevel Logic Synthesis Based on Functional Decision Diagrams, Proc. EDAC'92, 43–47, 1992.

[LS93] N. L. K. Lester, J. M. Saul: Technology Mapping of Mixed Polarity Reed-Muller Representations, Proc. EDAC'93, 305–309, 1993.

[MWBS88] S. Malik, A. Wang, R. K. Brayton, A. Sangiovanni-Vincentelli: Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment, Proc. IEEE International Conference on Computer-Aided Design (Santa Clara, Calif., Nov.), 6–9, 1988.

[Mei89] Ch. Meinel: Modified Branching Programs and Their Computational Power, Springer-Verlag, LNCS 370, 1989.

[MIY90] H. Minato, N. Ishiura, S. Yajima: Shared Binary Decision Diagrams with Attributed Edges for Efficient Boolean Function Manipulation, Proc. 27th ACM/IEEE Design Automation Conference (Orlando, June), 52–57, 1990.

[Mul54] D. E. Muller: Application of Boolean Algebra to Switching Circuit Design and to Error Corection, IRE Trans. Electr. Comp., vol. 3, No. 3 (Sep.), 6–12, 1954.

[PCSS92] M. A. Perkowski, L. Csansky, A. Sarabi, I. Schaäfer: Fast Minimization of Mixed Polarity AND/XOR Canonical Networks, Proceedings of ICCD'92, 33–36, 1992.

[Raz87] A. A. Razborov: A Lower Bound on the Size of Bounded Depth Networks over a Complete Basis with Logical Addition, Mat. Zametki, 41(4), 598-607, 1987, (in Russion); English translation in: Math. Notes 41(4), 333–338, 1987.

[Red72] S. M. Reddy: Easily Testable Realizations for Logical Functions, IEEE Trans. Comput., vol. C–21 (Nov.), 1183–1188, 1972.

[Ree54] I. S. Reed: A Class of Multiple-Error-Correcting Codes and Their Decoding Scheme, IRE Trans. Inf. Theory, vol. PGIT-4, 38–49, 1954.

[Sas92] T. Sasao: Optimization of Multi-Valued And–Exor Expressions Using Multiple–Place Decision Diagrams, Proceedings of the 22nd Int. Symp. of Multi-Valued Logic, 451–458, 1992.

[Sau92] J. M. Saul: Logic Synthesis for Arithmetic Circuits Using Reed–Muller Representation, Proc. EDAC'92 (March), 1992.

[SB90] T. Sasao, Ph. W. Besslich: On the Complexity of Mod–2 Sum PLA's, IEEE Trans. Comput., vol. 39, No. 2 (Feb.), 269–266, 1990.

[SR75] K. K. Saluja, S. M. Reedy: Fault Detecting Test Sets for Reed–Muller Canonic Networks, IEEE Trans. Comput., vol. C–24 (Oct.), 995–998, 1975.