MODD: A New Decision Diagram and Representation for Multiple Output Binary Functions

Abusaleh M. Jabir
School of Technology
Oxford Brookes University
Oxford OX3 0BP, UK
E-mail: ajabir@brookes.ac.uk

Dhiraj K. Pradhan*
Department of Computer Science
University of Bristol
Bristol BS8 1UB, UK
E-mail: pradhan@cs.bris.ac.uk

Abstract

This paper presents a new decision diagram (DD), called MODD, for multiple output binary and multiple-valued functions. This DD is canonic and can be made minimal with respect to a given variable order. Unlike other reported DDs, our approach can represent arbitrary combination of bits at the word-level. The preliminary results show that our representation can result in considerable memory saving [1].

1. Introduction

In this paper we present a new representation and a DD, called MODD, for multiple-output functions. Unlike BMDs [5], our method can be used to represent an arbitrary combination of output bits. For example, the MODD can simultaneously represent the most-significant bit (MSB) and the least significant bit (LSB) outputs of a 32-output Boolean function or, for that matter, any arbitrary subset of these output bits in any order. This can be useful for verification of higher-levels of abstraction, e.g., word-level RTL. This overcomes critical shortcomings of earlier DDs such as [2, 5].

The MODD is based on finite fields [4]. Preliminary results in [1] indicate that, in general, the MODD has smaller size than a collection of BDDs [2]. The MODD has advantages over [3], which is not a DD. Also, unlike the MDD, which is constructed on MIN-MAX based post algebra, the MODD is based on finite fields.

2. Proposed Decision Diagram

A finite field (also called *Galois field*) is a set GF(N) of N elements ($N = p^k$, where p is a prime number, and k an integer) and two operators '+' and '$\cdot$' with neutral elements 0 and 1 respectively, such that GF(N) forms an Abelian group (AG) under '+', and its non-zero elements form AG under '$\cdot$' operation [4]. Circuits can be efficiently designed and represented in a finite field. This paper presents the generalized theoretical aspects of the MODD for representing multiple-output Boolean as well as word-level circuits.

As an example, consider a 4-input 8-output Boolean function with MSB and LSB being $G = \sum m(10,11,14,15)$, and $H = \sum m(4,5,6,7,8,9,10,11,12,13,14,15)$ respectively. The resulting reduced MODD appears in Fig. 1. The BDD based representation of this circuit will require a larger number of nodes. This process can be carried out by means of the MODD for any number of outputs in any arbitrary combinations. Techniques like [6], which only work for a field of fixed order, are not suitable for this purpose.

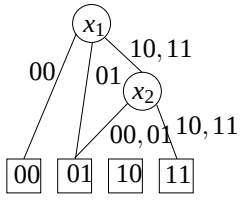

Fig. 1. **An MODD for 2 output bits.**

Any n-variable function $f(x_1,\ldots,x_k,\ldots,x_n)$ in GF(N) can be expanded as $f(x_1,\ldots,x_k,\ldots,x_n) = \sum_{e=0}^{N-1}[1-(x_k-\delta(e))^{N-1}]f|_{x_k=\delta(e)}$, where $\delta : I_N \rightarrow \mathrm{GF}(N)$ is a one-to-one mapping with $\delta(0) = 0$, and $I_N = \{0,1,\ldots,N-1\}$ [4]. The term, $[1-(x_k-\delta(e))^{N-1}]$ is a multiple-valued literal in GF(N). The MODD is based on this expansion and defined as a rooted directed acyclic graph (DAG) with a set of nodes V containing two types of nodes: (a) A set of N terminal nodes or leaves, with out-degree of 0, labeled with an element of GF(N). (b) A set of non-terminal nodes, with out-

* Supported in part by EPSRC, UK grant No. GR/S40855/01.

degree of N, having two attributes $var(v) = x_i$ $(1 \leq i \leq n)$, and $child_j(v) \in V$ $(j \in I_N)$ which represents each of the children (direct successor) of v.

An MODD rooted at v denotes a function f^v in GF(N) such that, if v is a terminal node, then f^v represents an element of GF(N). Otherwise, $var(v) = x_i$, and f^v is the function $f^v(x_1,\ldots,x_i,\ldots,x_n) = \sum_{e=0}^{N-1}[1-(x_i - \delta(e))^{N-1}]f^{child_e(v)}$.

Reduction If $\forall i,j \in I_N$ and $i \neq j$ $f|_{x_k=\delta(i)} = f|_{x_k=\delta(j)}$, then $f = f|_{x_k=\delta(0)} = f|_{x_k=\delta(1)} = \cdots = f|_{x_k=\delta(N-1)}$. Based on this, an MODD can be reduced as follows. (*a*) If all the N children of a node v point to the same node w, then delete v and connect the incoming edge of v to w. (*b*) Share equivalent subgraphs.

An MODD is said to be ordered if the expansion is recursively carried out in a certain linear variable order. It is said to be reduced iff: (*a*) There is no node $u \in V$ such that $\forall i,j \in I_N$ and $i \neq j$, $child_i(u) = child_j(u)$. (*b*) There are no two distinct nodes $u,v \in V$ which have the same variable names and same children, i.e., $var(u) = var(v)$ and $child_i(u) = child_i(v)$ $\forall i \in I_N$ implies $u = v$. A reduction algorithm can be constructed similar to a BDD [2].

We have the following regarding the canonicity and minimality of an MODD.

Theorem 2.1 *A reduced ordered MODD of a function in GF(N), based on the presented expansion in [4], is canonic up to isomorphism and minimal.*

Fig. 1 shows an example reduced MODD in GF(4), where $\delta(10) = \alpha$ and $\delta(11) = \beta$ are elements in GF(4).

Operations Let f and g be two functions in GF(N), and an algebraic operation '$\odot$' (e.g., addition, multiplication, subtraction, and division) in GF(N). Then, $f \odot g = \sum_{e=0}^{N-1}[1-(x_i-\delta(e))^{N-1}](f|_{x_i=\delta(e)} \odot g|_{x_i=\delta(e)})$. Based on this 'apply' operation, an MODD for $h = f \odot g$, can be obtained from the MODDs for f and g. As in the case for a BDD, a dynamic programming like approach can be incorporated to limit the complexity to $O(|G_f| \cdot |G_g|)$ ($|A|$ represents the number of nodes in a DAG A).

Also, $f|_{x_i=g} = \sum_{e=0}^{N-1}[1-(g-\delta(e))^{N-1}]f|_{x_i=\delta(e)}$. Based on this, an algorithm for composition of two functions can be formulated in a manner similar to that for a BDD [2].

Any path from the root node to a terminal node of an MODD with a value $\delta(s)$ constitutes a satisfying assignment with respect to $s \in I_N$. This can be determined by searching for a path from the root node to the terminal node with the value $\delta(s)$ in $O(|G_f|)$.

3. Applications to Synthesis and Verification

Each node in a reduced MODD can be represented by means of an N-input multiplexer, where each input can assume a value in GF(N). Therefore, an MODD can be translated to a network of word-level multiplexers on a one-to-one basis.

The MODD can be used to represent multiple output binary functions efficiently. Given an n-input, m-output Boolean circuit, any arbitrary combination of outputs and inputs can be grouped in chunks of r-bits and encoded in GF(N), from which a reduced DD can be constructed. Such a situation may arise in cases where certain outputs of a circuit requires separate treatment from others (e.g., an adder circuit with carry output). Owing to the canonicity of the DD, this can help verify arbitrary combinations of bits, which is not possible by techniques like [5]. An example appears in Fig. 1.

The DD can also represent word-level (e.g., RTL) combinational logic blocks, by means of the apply operation and topologically traversing a netlist of the blocks in GF(N).

The MODD can be used for factorization. For example, the MODD in Fig. 1 gives the factored form $(x_1^\alpha + x_1^\beta)((x_2^\alpha + x_2^\beta)\beta + (x_2^0 + x_2^1)) + x_1^1$, where $x_j^i = 1-(x_j - i)^{N-1}$.

This DD can also aid formal verification, e.g., model checking by efficiently storing the enormous number of internal states which may result. Often BDDs are used to store the internal states. However, the MODD, being multiple valued, can represent the internal states much more efficiently than a conventional BDD, if multiple-valued encoding in GF(N) is used.

Since the extension field over GF(p^k) can be constructed from the algebra of polynomials over the field GF(p) [4], this gives us the opportunity to verify bit-level implementations in GF(2) with higher levels of abstractions in GF(2^k) $(k > 1)$ with the help of the MODD.

Currently, work is underway for synthesis and verification under a unified framework based on the MODD.

References

[1] A. Jabir and D. Pradhan. A Theory of Finite Field Decision Diagrams. Technical report, Department of Computer Science, University of Bristol, UK, Nov. 2003.

[2] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.*, C–35(8):677–691, Aug. 1986.

[3] D.K. Pradhan, M. Ciesielski, and S. Askar. Mathematical Framework for Representing Discrete Functions as Word-Level Polynomials. In *Proc. HLDVT'03, San Fransisco, USA.*, pages 135–142, Nov. 2003.

[4] D. Pradhan. A Theory of Galois Switching Functions. *IEEE Trans. Comp.*, C–27(3):239–249, Mar. 1978.

[5] R.E. Bryant and Y.A. Chen. Verification of Arithmetic Functions with Binary Moment Diagrams. In *DAC-95*, 1995.

[6] R.S. Stanković and R. Dreschler. Circuit Design from Kronecker Galois Field Decision Diagrams for Multiple-Valued Functions. In *ISMVL-27*, pages 275–280, May 1997.