The Biddy BDD package

Robert Meolic[1]

1 Faculty of Electrical Engineering and Computer Science, University of Maribor

DOI: 10.21105/joss.01189

Software

- Review
- Repository
- Archive

Submitted: 15 January 2019
Published: 11 February 2019

Summary

A **Binary Decision Diagram** (BDD) is a data structure used in different areas including but not limited to the design, testing, optimization, and verification of digital circuits, communications protocols, and distributed systems (Bryant, 1986). There exist many different types of BDDs intended for different applications. The most notable types are used to represent Boolean functions and combination sets. For these purposes, BDDs can be very efficient, for example, they enable the representation and manipulation of set of sparse cubes with 10^{47} cubes (Minato, 2013).

In a BDD, every internal node contains a variable while leafs contain constants 0 and 1, respectively. For a Reduced Ordered Binary Decision Diagram (ROBDD), each edge to internal node n with variable $var(n)$, left successor $else(n)$, and right successor $then(n)$ corresponds to the Boolean function $f(n)$ that is calculated as:

$$f(n) = \overline{var(n)} \;\&\; else(n) + var(n) \;\&\; then(n)$$

For the ROBDD in Figure 1 this is applied as follows: $F3[088] = \overline{x1} \;\&\; (\overline{x3} \;\&\; 0 + x3 \;\&\; 1) + x1 \;\&\; (\overline{x2} \;\&\; (\overline{x3} \;\&\; 1 \; + x3 \;\&\; 0) \; + x2 \;\&\; 0)$. The same result can be obtained if every path starting in the root and leading to a leaf with constant 1 is considered to be a product of variables in which a negative literal is included if the path continues in the else successor and a positive literal is included if the path continues in the then successor. The resulting Boolean function is a sum of the obtained products. In this way, for the ROBDD in Figure 1 we directly obtain a minimal sum-of-products form, but in general, the result is not a minimal form. For the explanation of other types of BDDs we refer to the given references.

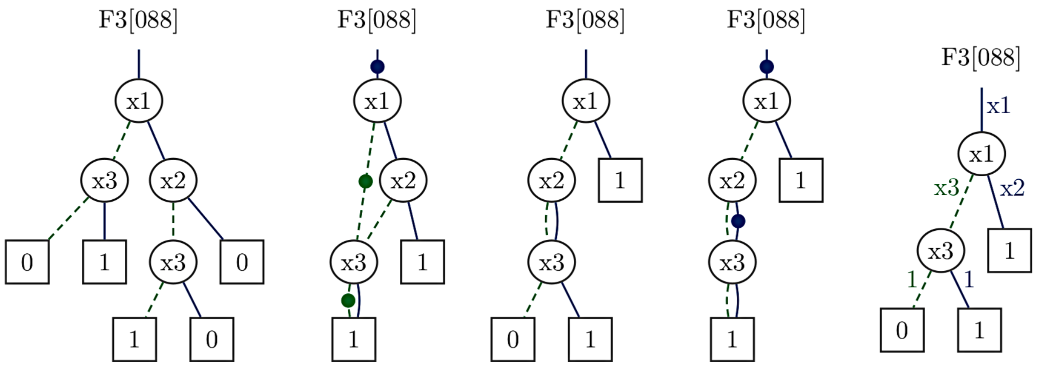

Figure 1: Representation of Boolean function $F3[088] = \overline{x1} \;\&\; x3 \; + x1 \;\&\; \overline{x2} \;\&\; \overline{x3}$ with different types of BDDs. From left to right there are an ROBDD, an ROBDD with complemented edges, a 0-sup-BDD, a 0-sup-BDD with complemented edges, and a tagged 0-sup-BDD.

Boolean functions are primarily used in digital circuit design where a thousand and more variables are not unusual. It is impossible to represent such large Boolean functions with vectors and similar explicit representations of the truth table. Strings are also not an option because they are not canonical. On the other hand, many huge Boolean functions of practical importance have a managable representation with BDDs. Nowadays, however, logic synthesis is not the only, or even not the main, target for Boolean functions and BDDs. They are also used as characteristic functions of sets and relations, which enables the encoding of combinatorial problems and their symbolic solution (Knuth, 2009) (Minato, 2013) (Meolic & Brezočnik, 2018). A very special method that has profited a lot from BDDs is model checking (Chaki & Gurfinkel, 2018). Application areas where BDDs can also be applied are approximate string matching, fault tree analysis, scheduling algorithms, security algorithms, reversible computing, and many others.

An efficient implementation of algorithms for BDDs is a rather complicated task, but several free BDD packages are avaliable online. Herein we describe the cross-platform **Biddy BDD package** (Meolic, 2012), one of the oldest continually developed of these BDD packages. From version 1.8.2, many standard features are implemented, such as automatic garbage collection, complemented edges, and a management system. Dynamic variable ordering with a sifting algorithm and an exhaustive search over all the possible variable orderings are provided, too. Various statistics about global properties and the individual Boolean functions are available.

Some distinguishing properties of the Biddy BDD package are:

- can be built on various platforms using native environments, including **gcc**, **mingw**, and **Visual Studio**;
- it follows a strict implementation style;
- it has a refined **C** API;
- it offers a uniform support for classical **reduced ordered BDDs** (ROBDDs) and **zero-suppressed BDDs** (0-sup-BDDs);
- at this moment, Biddy is the only package thoroughly supporting the **tagged zero-suppressed BDDs** (Meolic, 2016) (Dijk, Wille, & Meolic, 2017).

The Biddy BDD package is a part of the Biddy project that also focuses on the visualization of BDDs. The application **BDD Scout**, which is bundled with the Biddy BDD package, is an interactive tool (Figure 2). Its key features are:

- the creation of a BDD from a Boolean expression,
- node manipulation and variable reordering in the displayed BDD,
- conversion between the supported types of BDDs,
- exports to LaTeX, and
- integrated Tcl scripting.

These features make BDD Scout a unique tool for teaching and exploring properties of BDDs. For an example, check the generated BDD Encyclopedia (Meolic, 2019a).

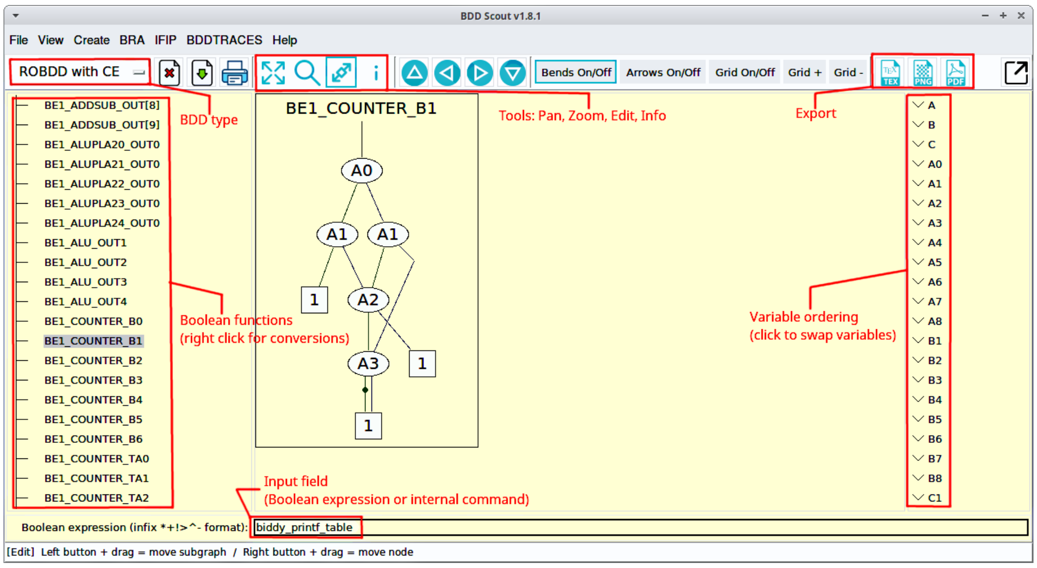

Figure 2: An annotated screenshot from BDD Scout

In conclusion, the Biddy BDD package is free software to be used in projects that need to manipulate Boolean functions or combination sets. It is a complete and efficient product suitable for many academic and commercial settings. The binary executables, the user manual, and the other documentation can be obtained from Biddy's Homepage (Meolic, 2019b).

Acknowledgements

The development of the Biddy BDD package and BDD Scout application so far was supported by University of Maribor, Faculty of Electrical Engineering and Computer Science.

References

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. IEEE transactions on computers, 35(8), pp. 677-691, 1986. doi:10.1109/TC.1986.1676819

Chaki, S., & Gurfinkel, A. (2018). BDD-based symbolic model checking. In handbook of model checking, springer, pp. 219-245, 2018. doi:10.1007/978-3-319-10575-8_8

Dijk, T. van, Wille, R., & Meolic, R. (2017). Tagged BDDs: Combining reduction rules from different decision diagram types. In proc. FMCAD 2017, pp. 108-115, 2017. doi:10.23919/FMCAD.2017.8102248

Knuth, D. E. (2009). *The art of computer programming, Volume 4, Fascicle 1: Bitwise tricks and techniques; Binary decision diagrams. Addison-Wesley Professional, 2009.*

Meolic, R. (2012). Biddy - a multi-platform academic BDD package. Journal of software, 7(6), pp. 1358-1366, 2012. doi:10.4304/jsw.7.6.1358-1366

Meolic, R. (2016). Implementation aspects of a BDD package supporting general decision diagrams. University of Maribor, Tech. rep., 2016. Retrieved January 15, 2019, from https://dk.um.si/IzpisGradiva.php?id=68831&lang=eng

Meolic, R. (2019a). BDD Encyclopedia. Retrieved January 15, 2019, from http://svn.savannah.nongnu.org/viewvc/*checkout*/biddy/bddscout/ENCYCLOPEDIA/bddencyclopedia.html

Meolic, R. (2019b). Biddy's homepage. Retrieved January 15, 2019, from http://biddy.meolic.com/

Meolic, R., & Brezočnik, Z. (2018). Flexible job shop scheduling using zero-suppressed binary decision diagrams. Advances in production engineering and management, 13(4), pp. 373-388, 2018. doi:10.14743/apem2018.4.297

Minato, S. (2013). Techniques of BDD/ZDD: Brief history and recent activity. IEICE trans. on information and systems, E96D(7), pp. 1419-1429, 2013. doi:10.1587/transinf.E96.D.1419