Transaction-Level Analysis and Optimization of Decision Diagram Packages on RISC-V

Rune Krauss[1], Jan Zielasko[1], Christoph Lüth[1,2], Rolf Drechsler[1,2]

[1]Cyber-Physical Systems, DFKI, Germany
[2]Institute of Computer Science, University of Bremen, Germany

Abstract

The complexity of modern electronic systems has increased significantly over the past decades due to continuous technological advances. To cope with this growing complexity, data structures, algorithms, and the underlying hardware platforms used in Electronic Design Automation (EDA) must be continuously improved. Decision Diagrams (DDs) constitute a fundamental graph-based structure for formal verification, enabling efficient representation and algorithmic manipulation of switching functions. Owing to their practical relevance, numerous optimizations have been incorporated into existing DD software packages. However, these optimizations are typically designed in an architecture-agnostic manner and do not explicitly exploit characteristics of a specific target platform. As a consequence, architecture-specific optimization opportunities may remain untapped. In this work, a transaction-level analysis of a representative DD package is conducted using a RISC-V-based trace analysis tool to investigate this potential. The study reveals recurring instruction sequences with strong potential for hardware-level aggregation, enabling more efficient hardware designs. Furthermore, the derived insights provide guidance for higher-level software optimizations.

1 Introduction

The increasing complexity of digital circuits requires continuous improvements in *Electronic Design Automation* (EDA) to meet time-to-market constraints [1]. *Decision Diagrams* (DDs) play a central role in formal verification, as they compactly represent switching functions and allow algorithmic manipulation [1]. Numerous data structures and algorithms have been developed based on DDs and implemented in software packages for EDA [1, 2]. These optimizations primarily target software, leaving potential hardware-level improvements largely unexploited.

To explore this potential, a transaction-level analysis is conducted using the SystemC-based *Virtual Prototype* (VP) tool *RISC-V Opt-VP* [3, 4].[1] The modular and efficient DD package *FrEDDY* [2] is employed as a case study.[2] Its application includes symbolic simulation, a precursor to EDA tasks such as equivalence checking [1]. This study demonstrates that recurring instruction sequences can be identified, enabling hardware/software optimizations and significant speedups. In summary, the main contributions are:

1. Transaction-level analysis of FrEDDY conducted using the RISC-V Opt-VP platform
2. Identification of hardware-oriented optimization opportunities for DD packages

[1] RISC-V Opt-VP is an application analysis platform available at `https://github.com/agra-uni-bremen/opt-vp` (accessed May 30, 2026).
[2] FrEDDY, an open-source software package, is publicly available at `https://github.com/runekrauss/freddy` (accessed May 30, 2026).

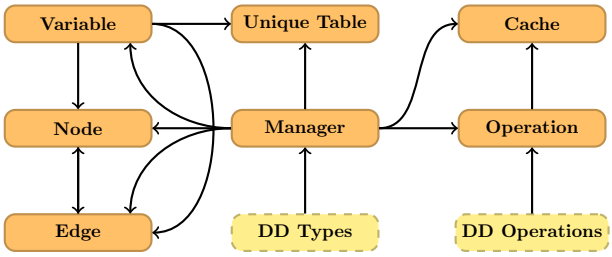

Figure 1: *Dependency graph of modules in FrEDDY.*

The extended abstract is organized as follows: Section 2 and Section 3 introduce FrEDDY and RISC-V Opt-VP, respectively. The case study is presented in Section 4, and the results are discussed in Section 5.

2 FrEDDY

FrEDDY is a C++23 framework for the efficient manipulation of DDs, distinguished particularly from other packages by its usability and extensibility. Its key concepts are illustrated as a dependency graph in Figure 1 and briefly summarized in the following. Using the manager and operation modules, arbitrary, user-defined DD types and corresponding operations can be implemented by specifying node and edge entities. Edges point to nodes, either labeled with variables or representing constants, which are organized levelwise in separate unique tables implemented as hash tables. A cache stores results of frequently invoked recursive operations to avoid redundant computations.

Further details on FrEDDY's modular architecture and functionality can be found in [2].

Table 1: *Excerpt of optimizations identified by RISC-V Opt-VP and recommended for FrEDDY.*

I	w_I	score(I, w_I)	#Deps.	PCs	Coverage (%)	Total Speedup
`ADDI → BLTU → SW → SW → SW → SW → ADDI`	141,978	993,846	8	`0x756D8`	40.70	1.42
`SW → SW → ADDI → BLTU → SW → SW`	141,978	851,868	3	`0x756D2`	34.89	1.26
`SW → SW`	537,997	1,075,994	0	`0x756D2`, ...	44.07	1.28

3 RISC-V Opt-VP

RISC-V Opt-VP is a RISC-V VP-based extension for tailoring hardware to application requirements. It traces the application to generate a bounded execution tree for each instruction. Next, the trees are processed by an analysis module using a scoring function:

$$\text{score}(I, w_I) = |I| \cdot w_I \tag{1}$$

In Equation (1), I denotes a specific instruction sequence, and w_I is its associated weight defined as the execution count of I during the application run. Evaluating the instruction trees ultimately identifies promising sequences, if any, and creates corresponding optimization recommendations.

A more comprehensive overview is available in [4].

4 Case Study

To evaluate the hardware optimization potential of FrEDDY using RISC-V Opt-VP, the following benchmark case was symbolically simulated:

$$f(x_1, x_2, \ldots, x_{2n}) = (x_1 \vee x_2) \wedge \ldots \wedge (x_{2n-1} \vee x_{2n})$$

where

n number of variable pairs

f switching function $\mathbb{B}^{2n} \to \mathbb{B}$

Figure 2 shows a tree of maximum depth 20 generated for the `ADDI` instruction under the RV32IMAC configuration, where $n = 5$. A promising sequence extracted from this tree is detailed in Table 1. In this case study, the subsequence `SW → SW` stands out in particular. By tracing the Program Counter (PC), it was determined that this pattern is related to hashing.

One possible hardware optimization is integrating a 64-bit bus into the RISC-V core and replacing the frequently used `memset` function at the corresponding program locations with a dedicated instruction via inline assembly. This modification not only yields a total speedup of about 28 %, but also positively affects other instruction sequences.

5 Discussion and Future Work

To the best of the authors' knowledge, this work presented the first transaction-level analysis of a DD package on RISC-V. The case study demonstrated

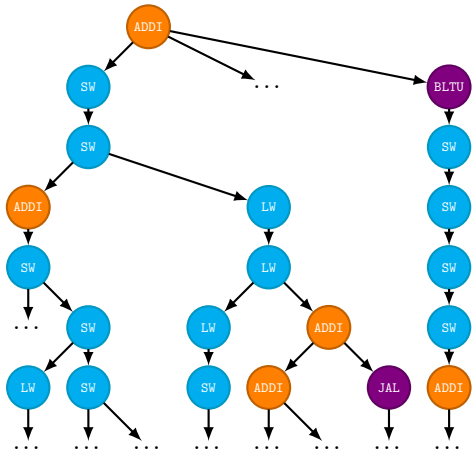

Figure 2: *A bounded execution tree based on FrEDDY.*

that, using RISC-V Opt-VP, significant hardware optimizations are possible for FrEDDY under symbolic simulation. As the identified patterns are related to hashing, the achieved results can be generalized to other DD packages. Since hashing techniques themselves offer further optimization potential, the derived insights enable additional software improvements. Future work will also include hardware realizations, potentially leading to a dedicated DD processor.

Acknowledgments

This work was supported in part by the *Federal Ministry of Research, Technology and Space* through the projects DI-OCDCPro (grant no. 16ME0938), ExaVerse (grant no. 01IW25003), and FAIRe (grant no. 01IS23074).

References

[1] C. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design*. Springer, 2012.

[2] R. Krauss, J. Zielasko, and R. Drechsler. "FrEDDY: Modular and Efficient Framework to Engineer Decision Diagrams Yourself". In: *Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–2.

[3] W. Herdt et al. "RISC-V Based Virtual Prototype: An Extensible and Configurable Platform for the System-Level". In: *Journal of Systems Architecture* 109 (2020), pp. 1–13.

[4] J. Zielasko et al. *RISC-V Opt-VP: An Application Analysis Platform Using Bounded Execution Trees*. RISC-V Summit Europe Poster. 2024. URL: `https://riscv-europe.org/summit/2024/posters`.