# Formal Verification of Integer Multiplier Circuits

Jonathan Wang
*Price College of Engineering*
*University of Utah*
Salt Lake City, Utah
u1306458@utah.edu

Henry Silverman
*Price College of Engineering*
*University of Utah*
Salt Lake City, Utah
henry.silverman@utah.edu

Garrett Slack
*Price College of Engineering*
*University of Utah*
Salt Lake City, Utah
u1315263@utah.edu

Dmitry Panin
*Price College of Engineering*
*University of Utah*
Salt Lake City, Utah
dmitry.panin@utah.edu

***Abstract*—This paper employs advanced mathematical techniques, specifically polynomials, and ideals, to rigorously verify the correctness of an integer multiplier circuit. By leveraging algebraic methods, this approach will provide a deeper understanding of the circuit's behavior and enable a more robust verification process.**

***Index Terms*—polynomials, ideals, multiplier circuit**

## I. Introduction

The paper presents the testing and verification of a 2-bit, 3-bit, 16-bit, and 32-bit integer multiplier circuit. We manually designed the 2-bit and 3 bit multiplier from scratch and derived a polynomial spec. We generated the larger circuits utilizing the ABC synthesis tool.

## II. Approach

Our approach to generate the 2-bit and 3-bit multiplier is simple for circuit generation,. We added on to the 2-bit multiplier provided by Dr. Kalla using the GEdit text editor. Then ran on command line in VMA RedHat Linux. For the larger 16-bit and 32-bit circuits, we generated the circuits using the ABC synthesis tool shown in Fig. 1.

```
abc 01> gen -N 32 -m 32BitMult.blif
Hierarchy reader flattened 2080 instances of logic boxes and left 0 black boxes.
abc 02> strash
abc 03> ifraig
abc 04> resyn2
abc 08> dch
abc 09> read lib2.genlib
Warnings: genlib library reader cannot detect the buffer gate.
Some parts of the supergate-based technology mapper may not work correctly.
Entered genlib library with 5 gates from file "lib2.genlib".
abc 09> map
abc 10> write_blif Mapped32BitMult.blif
```

Fig. 1. Steps to generate a 32-bit circuit in ABC.

## III. Algorithms and Techniques

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections below for more information on proofreading, spelling and grammar.

## IV. Software implementations

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections below for more information on proofreading, spelling and grammar.

## V. Concepts learned

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections below for more information on proofreading, spelling and grammar.

## VI. Labour Division

Jonathan: a lot
Henry: a lot
Garrett: a lot
Dmitri: a lot

## References

[1] D. Ritirc, A. Biere and M. Kauers, "Column-wise verification of multipliers using computer algebra," 2017 Formal Methods in Computer Aided Design (FMCAD), Vienna, Austria, 2017, pp. 23-30, doi: 10.23919/FMCAD.2017.8102237.

[2] T. Pruss, P. Kalla and F. Enescu, "Efficient Symbolic Computation for Word-Level Abstraction From Combinational Circuits for Verification Over Finite Fields," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 7, pp. 1206-1218, July 2016, doi: 10.1109/TCAD.2015.2501301.