# Evaluation of SAT Solvers and Bit-Blasting to challenges numerical bugs in programs

---

**Thesis advisors:**

- David DEFOUR, Full Professor, Univ de Perpignan Via Domitia
- Blaise MADELINE, Associate Professor, IMERIR, Perpignan

**Location:** University of Perpignan Via Domitia, France

**Duration**: 4-6 months

**Keywords:** Bit Blasting, SAT Solver, Polynomial evaluation

**Compensation:** ~630€/months

---

## 1. Introduction

Numerical problems often involve finding inputs that lead to specific desired outputs. Traditionally, these problems are solved using numerical methods, approximations, or optimization techniques. However, in numerical computation, where the representation of numbers is done in a finite precision (i.e., binary), there arises an opportunity to employ combinatorial techniques to search for specific bit patterns in inputs and outputs.

Bit blasting, a technique used in formal verification, translates a problem into a Boolean formula that can be solved by SAT solvers. In recent years, SAT solvers have shown great success in solving complex combinatorial problems. The ability of bit blasting to handle large, complex numerical problems in a bitwise manner can potentially open new avenues for generating inputs that exhibit specific bit patterns.

This thesis aims to explore the feasibility of using bit blasting, in combination with SAT solvers, to solve numerical problems where specific bit patterns are required in both the inputs and the outputs of a given formula. The primary focus will be on polynomial evaluations where the coefficients are fixed, and specific bit patterns are imposed on both the input and output. This exploration could lead to new methods of finding challenging numerical example and counter example in areas such as cryptography, number's theory or formal verification.

---

## 2. Problem Statement and Research Objectives

The goal of this research is to investigate whether bit blasting, paired with modern SAT solvers, can be effectively used to find numerical inputs that yield specific bit patterns in their outputs. The objectives of this thesis are:

1. **Bit Blasting for Polynomial Evaluation**: Investigate how polynomial evaluation problems can be represented using bit blasting. Start with simple polynomials and gradually increase the complexity by adding more variables and higher-degree polynomials.

2. **SAT Solver Integration**: Use modern SAT solvers to solve the bit-blasted Boolean formulas and find numerical solutions that exhibit specific bit patterns in their binary representation.
3. **Performance Evaluation**: Evaluate the performance of SAT solvers in terms of time complexity and resource utilization for solving these problems and compare various solver for this class of problem. Measure the scalability of the approach by varying the bit-width, polynomial degree, and input-output bit pattern constraints.
4. **Challenging Problem Generation**: Investigate whether this approach can be used to generate challenging numerical problems, such as finding inputs that lead to outputs with rare or predefined bit patterns such as hard to round case (i.e. Table Maker's Dilemma[1])

---

## 3. Literature Review

The literature on SAT solvers and bit-blasting spans several decades, with applications ranging from hardware verification to software model checking. This section will focus on foundational work in SAT solving, recent advancements in bit-blasting techniques, and their use in program verification.

1. **SAT Solvers and Bit-Blasting**:
    - Overview of SAT solver algorithms (DPLL, CDCL) and their evolution.
    - Examination of bit-blasting as a technique for converting high-level arithmetic and logical operations into SAT problems.
    - Insights from SAT competitions and performance benchmarks to understand the current state-of-the-art in SAT solver efficiency.
2. **Formal Verification with SAT Solvers**:
    - Applications of SAT solvers in software verification and bug detection.
    - Comparative analysis of SAT-based formal methods versus symbolic execution and model checking.
    - Case studies where SAT solvers were successfully used in discovering software bugs (e.g., integer overflow detection, buffer overflows, race conditions).
3. **Challenges and Limitations** :
    - Known challenges with bit-blasting, including formula size explosion and solver performance degradation in large or complex software systems.
    - Discussion of trade-offs between precision and efficiency when using SAT solvers and bit-blasting for bug detection.

---

## 4. Methodology

1. **Experimental Setup**:

[1] https://en.wikipedia.org/wiki/Rounding#Table-maker's_dilemma

- Selection of target SAT solvers, including but not limited to Z3, MiniSAT, and CryptoMiniSat.
- Development of an encoding pipeline to convert high-level formulae into bit-level representations suitable for SAT solving.

2. **Polynomial Representation**:
   - Start by representing simple polynomials (e.g., linear, quadratic) as Boolean formulas using bit-blasting techniques. Each variable and constant will be represented in binary, and operations like multiplication and addition will be converted into their Boolean equivalents.
3. **Bit-Blasting Process**:
   - Solve the Boolean formulas generated by the bit blasting. The solver will output bit vector's that satisfy both the polynomial equation and the bit pattern constraints.
4. **Performance Evaluation**:
   - Measure and compare SAT solver performance under various conditions while focusing on time to solution, memory usage, scalability, against satisfiability to unsatisfiability constraints.
5. **Optimization Techniques**:
   - Explore optimization methods for reducing the complexity of the SAT problem generated by bit-blasting on this class of problem (i.e. Techniques such as abstraction, formula simplification, and lazy bit-blasting)

---

## 5. Expected Contributions

1. A clear methodology for representing numerical problems (polynomial evaluation) as Boolean formulas using bit blasting.
2. Insights into the effectiveness of SAT solvers for finding numerical solutions that exhibit specific bit patterns.
3. A better understanding of the strengths and limitations of using bit blasting and SAT solvers for generating challenging numerical solutions.
4. Potential extensions of this approach to other domains, such as cryptography and number's theory.

---

## 6. Applications

The candidate should demonstrate skills or, at a minimum, a strong interest in software development, complexity analysis, SAT Solver and computer arithmetic.

Motivated candidates should send their CV and transcript to:

david.defour@univ-perp.fr, blaise.madeline@imerir.com

---

## 7. References


- Brain, M., Schanda, F., Sun, Y. (2019). Building Better Bit-Blasting for Floating-Point Problems. In: Vojnar, T., Zhang, L. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2019. Lecture Notes in Computer Science(), vol 11427. Springer, Cham. https://doi.org/10.1007/978-3-030-17462-0_5
- Gadelha, M.R., Cordeiro, L.C., Nicole, D.A. (2020). An Efficient Floating-Point Bit-Blasting API for Verifying C Programs. In: Christakis, M., Polikarpova, N., Duggirala, P.S., Schrammel, P. (eds) Software Verification. NSV VSTTE 2020 2020. Lecture Notes in Computer Science(), vol 12549. Springer, Cham. https://doi.org/10.1007/978-3-030-63618-0_11
- Brain, M., (2021). Further Steps Down The Wrong Path : Improving the Bit-Blasting of Multiplication. SMT'21: 19th International Workshop on Satisfiability Modulo Theories, https://ceur-ws.org/Vol-2908/short16.pdf