Information and Guidance for Class Projects

ECE/CS 5745/6745: Testing and Verification of Digital Circuits

Priyank Kalla
ECE Dept., Univ. of Utah
kalla@ece.utah.edu

Note: As of October 30, 2019, this document is a work in progress. I will continue to update it with more project options as and when I get more ideas.

I. Introduction

Students in the class are required to work on a term project. You will work on the project as a team of 2 students. As we have an even number of students in the class, we’ll be able to accommodate everyone with a team of 2 students. The project should address hardware verification, core computational technologies for verification (SAT/SMT-solvers, Gröbner Bases, extensions of BDDs, AIGs, etc.), or consider other applications of verification technologies – such as hardware security, logic encryption, etc. Verification applications may range from combinational circuit verification, sequential circuit verification, RTL datapath verification, to any problem than can be formulated as such. The core methods should include concepts from testing and verification, such as algebraic geometry and symbolic computation (Gröbner basis reasoning), SAT-based function abstraction, post-verification debugging (bug location and characterization) of incorrect circuits, but not necessarily limited to it. Your formulations can be a mixture of algebraic geometry, symbolic computation, SAT, AIGs and BDDs. Since computational algebraic geometry is a core component of the course, studying some advanced Gröbner basis algorithms – or tweaking them for application to circuits or SAT solving – is also allowed.

The project will require that you develop a CAD framework for verification, conduct some experiments with various tools, gain some experience about how verification techniques are applied on practical circuits/systems. You may also develop core verification engines (solvers), and benchmark their execution on available circuits/designs. It is also possible to formulate your project around a theoretical investigation. The choice of the project is yours — remember, however, that I am the boss, and I will have the final say about your project deliverables ☺.

II. Time-line

Project time-lines are as follows:
1) Form a team of 2 members. Feel free to use the Canvas discussion board to scout for team members.
2) Schedule a first meeting with me either this week or next. [I will send you my availability after I setup a Doodle-like poll for you to pick a time.]
3) We will discuss your interests and your project options, and I may help you with some reading materials, tools, benchmarks, etc.
4) By Nov 8, your project should be finalized.
5) By Nov 13, you will submit a project title and 1-2 page(s) abstract/summary of your project.
6) The final project report, and any tools developed, will be due soon after exam ends, i.e. by December 15. Details on project submission will be sent out later.
7) We can have project meetings and consultations as and when required.

III. Project Options

In this section, I will provide you with some suggestions on what kind of projects you can work on. I am also providing you with some references, which can be found on IEEExplore or ACM Digital Library. From the Univ. of Utah network, you have free access to almost all of these papers. In case you cannot find these references, contact or me. All papers authored by me are available on my website too.

A. Verification Core Technologies

Core technologies for verification include SAT solvers, BDDs, SMT-solvers, and various Symbolic Computing engines such as Gröbner basis engines, number-theoretic solvers, etc. You will notice that solvers – along with their source code – are already available to us. Therefore, just re-designing/re-writing the same solver engines is probably not a fun exercise. However, building extensions on these base engines is acceptable – but is not my favourite option.

B. Gröbner Bases and SAT

Solving SAT problems using Gröbner bases (GB) is a very fundamental problem, it does have a rich history, but not rich enough to the liking of most SAT-researchers. A good start is my paper [1], where you may find references to many other SAT+GB frameworks. I can also provide you with Condrat’s M.S. thesis on this same topic.

GB computation is very susceptible to the term ordering used in Buchberger’s algorithm. SAT is also quite dependent on the various decision heuristics used for solving. You could study and learn about constraint partitioning (by way of hypergraphs) and variable ordering from the SAT domain, and apply these techniques to derive efficient term ordering for GB engines. Again, a good start is my paper [2], and it’s extensions [3].

One of the most important contribution recently in SAT is about conflict clauses. Finding conflict clauses is akin to information mining from constraints. Buchberger’s S-polynomials in the Gröbner basis algorithm also “mine” more information about ideals, in terms of polynomials with new leading terms. What relationships do S-polynomials have with conflict clauses? This problem is not well understood, so it might be a good idea to study this problem. A relevant paper is [4].

In class, we have also studied about UNSAT cores of CNF formulas. Can the Gröbner basis algorithm be used to identify UNSAT cores? You could try modifications to the techniques presented in my paper [5]. On similar lines, one can also solve #$SAT$, i.e. count the number of SAT solutions. See Gao’s M.S. thesis [6].

UNSAT cores are a type of a “minimum UNSAT (min-UNSAT)” problem. Its dual is MAXSAT. [Do a search on the internet on the MAXSAT problem, and contact me for some tutorial slides]. Wouldn’t you be interested in finding the Gröbner basis analog of the MAXSAT problem, or finding UNSAT cores?
C. On Gröbner Basis Engines

Buchberger's algorithm is old, circa 1963. In the last decade or so, a new algorithm has been devised by J.-C. Faugère called the F₄ algorithm [7]. It is based on the same principles as Buchberger's algorithm (reduce all $\text{Spoly}(f, g) \xrightarrow{G} 0$). However, it performs a simultaneous reduction of many S-polynomials in one-go, based on a Gaussian-elimination Matrix reduction. A team may actually study and implement the F₄ algorithm. It is a very do-able project, may also turn out to be kind of easy. If you are interested, talk to me and we can probably even think of a few extensions of this algorithm and/or tweak it for circuit verification. The original F₄ paper is going to be very hard to read. However, there is a nice M.S. thesis [8], also available on the web (I have a copy), describes F₄ in Chapters 1 and 2. Prof. Cox's book [9] (its latest 4th edition) has a description of a simplified version of the F₄ algorithm.

After F₄, Faugère came up with it's extension called the F₅ algorithm for Gröbner basis. That algorithm can also be a class project.

D. Boolean Gröbner Bases

Since we all want to exploit the power of GB engines for Boolean circuits, there have been attempts to build Boolean Gröbner Basis engines for model checking and SAT solving. This could be a good project as it combines a BDD-like DAG representation (Zero-Suppressed BDDs or ZDDs) and GB engines [10] – so you can learn both BDD-type unique tables and Gröbner Bases. What’s cool is that the ZDD library is available in CUDD, so you should already be familiar with it. This project requires representing Boolean polynomials with BDDs, and modeling arithmetic operations (mod 2) as AND-XOR operations on BDDs – it’s pretty cool. You can refer the following papers by Utkarsh [11], [12]. This could be a good project for those of you who have already studied BDDs with me.

If one can use BDDs for (AND/XOR) polynomial manipulation, then why not use AIGs instead? That could be another option: *Boolean Gröbner basis computation on digital circuits using AIGs*. Other similar ideas include [13] [14].

E. Combinational Circuit Verification

Recent work has seen very successful application of GB engines to combinational circuit verification. The approach can be applied over Galois fields $\mathbb{F}_{2^k}$ or over finite integer rings $\mathbb{Z}_{2^k}$. Over $\mathbb{F}_{2^k}$, Buchberger's algorithm works as-is, no problems; however, $\mathbb{Z}_{2^k}$ a variation of Buchberger's algorithm has to be applied. However, GB computation over large circuits can be computationally infeasible due to the complexity of the GB algorithm. There have been many investigations on how this complexity can be avoided, with some very cool results.

One paper that completely obviates the need to compute a GB is my own [15], download the extended version from my website. [This paper is very easy to read and is written in a way that an average computer engineer can understand, with good tutorial content.] Then there is another paper that solves the reverse problem: Given a circuit C, identify the function implemented by it, by efficient application of Gröbner Basis computations [16] [17]. Study/extensions of these techniques is also going to be a very good project.
F. Verification of Integer Multipliers

While the aforementioned papers [15] [16] solve verification problems over Galois field circuits, there are very similar applications over integer arithmetic circuits, such as [18]. This paper was then extended for SMT-solving too [19]. Any contribution of these techniques to RTL verification will have a lot of impact!

In the past 1 year, the world has also solved the verification of integer multipliers. This was considered a very hard problem, almost unsolvable, but is now pretty much solved due to the recent papers: i) Fast Computer Algebra re-writing based on And-Invert-Graphs [unpublished, email me if you want to see a manuscript], and its previous version [20]; ii) a later paper [21]; and iii) a paper published very recently [22] which you can download from here: http://fmv.jku.at/papers/RitircBiereKauers-FMCAD17.pdf. This paper won the best paper award at FMCAD!

While we will study the basic ideas of [22] in class, a detailed study of this paper can be a class project.

G. SMT-solving for Word-Level RTL datapaths using non-GB engines

GB can help in solving a system of polynomial equations, but it cannot actually solve a system of polynomial equations. Therefore, some sort of a SAT/SMT or a number theoretic solver is still needed. One way to solve the RTL datapath verification problem is using Wedler et al [19]. Another orthogonal approach is based on vNewton's p-adic iteration [23]. Probably, a sweet spot is a mixture of both number-theoretic and GB-based approaches.

If you are interested in working over RTL-datapaths, and don’t mind exploring a slightly more investigative project, talk to me about it, and we could explore if bit-level techniques such as [18] can be explored over word-level RTLs.

H. Sequential Circuit Verification

If you are interested in sequential circuit verification, then the first thing to do is to study the sequential circuit verification chapter from the textbook [24] (available in our library), and understand how BDDs are used for implicit state enumeration of sequential circuits (we will study these in the class too, but a little later). Then, you can explore GB engines to do just that — implicit state enumeration. Two papers that address such issues [25] [26].

If you have taken courses in model checking, then there is a world of knowledge out there that applies GB engines for model checking [14] [13].


Consider a combination circuit C, which is correctly designed. Assume that all the gates in the circuit are known, except only 1-gate is unknown. You are asked to identify the function implemented by that gate. This is a problem of rectification that we have studied in class. The corresponding papers are [27] that used Craig interpolation, and another which uses iterative SAT solving [28]. One of the projects could be to use these ideas and recreate his experiments using a SAT solver.

Another option would be to use the Gröbner basis engine instead of a SAT solver for rectification – my students have recently studied these problems using: i) Craig interpolation [29], for which I have a copy of
an extended book chapter that I wrote for it. ii) using extended ideal membership testing [30]. So maybe you could study these papers too.

This problem formulation of [28] actually inspires the post-verification debugging problem: Given a buggy circuit, where we may assume that only one logic gate (say, AND-gate) is mistakenly replaced by another one (say, OR-gate). How would you identify which gate is faulty, and provide the correct replacement? This is the debugging and auto-correction problem. The papers [31] [32] have tried to apply these to arithmetic circuits, but we have recently concluded that the techniques of [31] and [32] are incomplete. If you would like to study this problem and try to find solutions, feel free to talk to me about this project.

V. Craig Interpolation in Algebraic Geometry

Just like Craig interpolation exists over Boolean (un)SAT problems, similarly, polynomial algebra over finite fields also admits Craig interpolation. We have built an entire theory of Craig interpolation in finite fields. We will look at it in the class too, using Gröbner basis with LEX term orders. ABC computes interpolation using a heuristic on UNSAT proofs. I am interested in finding analogues of UNSAT proofs in algebraic geometry and efficiently computing an interpolant. If you have any interest in these ideas, let me know and I will email you a manuscript that I wrote that sketches these ideas – and it has examples too!

VI. Applications in Hardware Security – Updated 11/05/17

In hardware security, SAT-solvers and ATPG engines are used very often for security validation, for attacks on hardware ICs to break obfuscation or logic locking. As Gröbner bases complement SAT solving, I am very interested in studying some of these topics by replacing a GB engine for a SAT solver and asking questions such as: Can GB engines provide a mechanism or a metric to evaluate (or measure) hardware security provided by a technique? Can GB engines help derive a formal model of hardware security? Does classification of a variety (corresponding to the number of input vectors applied to a circuit) give us more information about security/logic encryption implemented to secure a circuit than what is provided by a SAT solver? These type of projects can be very investigative, and a learning experience, as it involves modeling the problems by trial-experimentation-error-repeat, and I’ll be interested in them.

An interesting application in hardware security is to crack-open a logic obfuscation or IC-camouflaging. A recent paper [33] uses a SAT solver to reverse engineer camouflaged gates in an obfuscated circuit. You could try the same experiments as given in this paper, or try to come up with a Gröbner basis formulation for the same problem. A Gröbner basis formulation definitely exists, and relates to the concepts of projections of varieties; it can be solved by computing a Gröbner basis with a specific LEX term order!

You can also try to verify hardware implementations of Cryptography primitives such as AES or DES.

VII. Any other ideas you have?

Such as computing Gröbner bases for Neural networks, like binarized NN, or CNNs, etc., for verification? Any idea that you may have? Talk to me about it, and we’ll see if it makes sense for a class project.

VIII. My Availability for Project Meeting

Let me send you a Doodle-like poll. If those times don’t work for you, email me and we’ll find some time.
IX. Project Proposal Submission Guidelines

Each group of students needs to submit a project proposal by Nov 13. But before that, every team needs to meet with me at least once. You can send it to me via email. The proposal should be a 1-2 page(s) document that should describe:

1) A tentative title of your project and the constituent team members.
2) The general topic area that you plan to study, and the core computational technologies that you wish to employ. For example, are you interested in combinational circuit verification, sequential circuit or RTL verification? Will you learn Gröbner basis concepts and their application or will you use BDDs, SMT solvers, SAT solvers or a combination of these?
3) If you are interested in developing a core engine, say, the $F_4$ algorithm and applying it to a specific domain — such as over Boolean rings for bit-level circuit verification — then describe the objective clearly.
4) If you have already refined/finalized the exact problem you wish to solve, then state the problem as best as you can. If you only have a general idea of the problem, and plan to refine the exact problem later with my help, it is okay to say so. But, do you best to narrow down the focus as best as you can.
5) Once the problem is described, what approach will you take to address your project. What topic areas require further study and which papers will you refer to? Provide a (short/relevant) reference list.

Do not worry if your proposal is not a very precise and mathematically clean document for now. Give it your best shot. I will help you finesse it, and I am expecting that many of you will have to go through one more iteration with my feedback to refine it. Good luck!

X. Final Project Submission Guidelines – will update later

References


