Information and Guidance for Class Projects
ECE/CS 6740/5740: CAD of Digital Circuits
Logic Synthesis and Optimization

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

Latest Update: February 20, 2019

I. Introduction

Students in the class are required to work on a term project. You may work on the projects either individually, or as a team of 2 students. Of course, a team of 2 students will do twice as more collective work than an individual. The objective of the project is to gain more experience – outside of lectures/HW – about electronic design automation (EDA) technologies in general, and modern logic optimization problems in particular.

Projects can be of the following type:

- *Take a well-defined CAD problem, study algorithmic solutions and implement it. It is okay if the problem is "text-bookish."* However, this is not a very exciting option. Most core-technology solvers are readily available in public domain, so it might make more sense to study applications of these CAD implementations. However, I may still allow it if you have a solid justification to pursue this type of a project.
- *Take a contemporary research problem in CAD, and perform some preliminary investigations.* These investigations can be experimental and/or theoretical, or survey the available literature. Based on the investigations, the team should propose a (justifiable) research direction. For many such problems, it is not hard to perform a rudimentary implementation in existing tools such as with BDDs, or with the ABC tool, and so on.
- Analyze some new and emerging areas in VLSI, SoC, embedded design, etc., and see if there is a need for CAD/automation/optimization tools. Examples of such areas are: reversible/quantum logic, binarized neural networks and their interpretations as circuits, logic design in integrated optics.

Applications may range from combinational circuits, sequential circuits, RTL designs, or any problem from combinatorics that can be formulated using circuits, CNF (propositional logic) formulae, Boolean relations, Boolean polynomials, etc. The technologies can be a mixture of symbolic computation, SAT and SMT solvers, any variation of BDDs, And-Invert Graphs, or the more recent Majority-Inverter graphs, etc.

The project will require that you develop a CAD framework for synthesis and/or verification, conduct some experiments with various tools, gain some experience about how CAD techniques are applied on practical circuits/systems. You may also develop core synthesis engines (solvers), and benchmark their execution on available circuits/designs. It is also possible to formulate your project around a theoretical investigation.

Keep in mind that this is a 1.5-month long class project, not a Master's thesis. The choice of the project is yours — remember, however, that I am the boss, and I will have the final say about your deliverables ☺.

II. Important Time-lines

Project time-lines are as follows:

1) Decide if you wish to pursue a team project or an individual one. *My suggestion: If you are new to this area, you should work as a team of 2. If you have prior experience in some aspects of EDA/Verification, and you have some ideas of what you wish to explore, then you may pursue an individual investigation.* Feel free to use Canvas discussion board to scout for team members.
2) Schedule a first meeting with me either this week, or early next week. I will open up a Doodle poll where you may sign up. This is important, we have to have this meeting.
3) We will discuss your interests and your project options, and I may help you with some reading materials, tools, benchmarks, etc.
4) By March 9, you will submit a (potentially tentative) project title and 1-2 page(s) long abstract/summary of your project investigations, based on our discussions.
5) By March 22, your project objectives should be finalized, and you should be working on it now.
6) Around the 10th of April, we will have 1 more meeting with each group to monitor progress.
7) Classes end Tuesday April 23. Our final exam is scheduled for Friday April 26. Exams end May 1. Grades are due by May 14. Based on these deadlines, I am setting the final project submission deadline as **Friday May 3**. The report, and any tools developed, will be due by 5pm on May 3. Details on project submission will be sent out later.
8) We can have project meetings and consultations as and when required. So please do not hesitate to request another meeting.

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 UofU network, you have free access to almost all of these papers. In case you cannot find any of these references, contact me.

A. Two-Level Logic Synthesis

The obvious idea would be to implement an `Espresso-Lite` two level optimization tool. The boring case would be to implement the vanilla branch-and-bound table covering version. Those who are good at programming may actually consider `Espresso-Lite` using the *unate recursive paradigm*. For this version, I can give you information on a very nice data-structure called the *positional cube notation* and how to perform the co-factor and consensus operations on it. Details are also given in the textbooks [1] [2]. For benchmarking, you could compare the quality of your tool's optimization against Espresso's.

You may also consider two-level encoding problems, such as Dichotomy-based Input Encoding Tool (DIET), which uses the "multi-valued input" representation capability of Espresso to solve some interesting problems. The main journal paper reference is [3].

These projects are least researchy, probably less intellectually challenging, **and maybe these topics have lost relevance**. But they are somewhat non-trivial with regards to implementation. Honestly, these type of projects are not something that I am very excited about.

Zero-Suppressed BDDs (ZDDs): In the mid-90s, Shin-Ichi Minato discovered that BDDs are not that cool for sparse combinatorial problems for characteristic sets. So he invented ZDDs which can be very powerful for a whole bunch of logic optimization problems, such as *unate cube set algebra* [4], irredundant covers and other combinatorial problems [5]. Fortunately, ZBDDs are already available as a library within the CUDD [6] package. Why not implement some of the combinatorial optimization problems described by Minato in his papers using CUDD?

B. Multi-Level Synthesis

The obvious project options:

- Implement kernel set computation, weak division and common subexpression extraction using the algebraic model. Compare against SIS. Not so much fun.
- Implement simple disjunctive decomposition and Ashenhurst-Curtis decomposition using BDDs. Use your decomposition engine as a complement to SIS.
- Implement the BDD-based Bi-decomposition system: try AND, XOR and MUX decompositions. Everything is given in the paper [7]. Implement these algorithms using the CUDD package.

We will study these topics in class from now onwards.

C. Boolean function decomposition using SAT and AIGs

Over the last few years, a new approach to scalable Logic Synthesis is emerging. This approach does not use a functional representation as the data-structure for synthesis. Instead, a simplified representation of the circuit called And-Invert-Graph (AIG) is used and the decompositions are derived using SAT solvers!

For example, suppose that you are interested in bi-decompositions of XOR-type ($f(x, y, z) = f_1(x, y) \oplus f_2(y, z)$). Can such a decomposition with the given variable partition $[(x, y); \quad (y, z)]$ be identified? In [8], the authors present the **conditions of bi-decomposability** in terms of a (UNSAT) QBF formula. *Jiang et al* [9] [10] [11] derive the decomposition by translating the QBF into an equisatisfiable SAT formula, and then analyzing its UNSAT core proof and Craig interpolant to identify the decomposed components.

While we will study these concepts of Boolean function decomposition in class [7] [8] [11] [10] [9] [12], some bi-decomposition problems using AIGs and SAT can be explored as class projects. It turns out that the AIG-based logic synthesis tool is already available to us, it is called ABC [13] and the full source code can be downloaded from Alan Mishchenko's website at http://www.eecs.berkeley.edu/~alanmi/abc/.

The journal paper [9] describes these concepts in more details.

D. Sequential Logic Synthesis

I'll update this section later. Still trying to see if there is anything interesting for a class project option – i.e. doesn't explode in complexity.

E. Logic Synthesis for Approximate Computing

Approximate computing is a new paradigm in hardware design and there are applications in speech, natural language and image processing that require design tools and technologies to realize them. Such applications can tolerate a few errors in their outputs. For example, the implemented circuit may match the specification at all but a very few inputs. The task is to introduce a few errors (given an error bound) so that the circuit can be further optimized for area, delay or power consumption. Not surprisingly, the logic synthesis community has begun to study these problems. In fact, some of them have used SIS and Synopsys design compiler to formulate the problems. How about you re-create one of these papers, i.e. study their formulation and recreate their experiments albeit at a smaller scale. Start with [14]. There is also an interesting PhD thesis [15] that you can look at. You don't need to read the full thesis, just look at the publications of Jin Miao (UT Austin), browse through the related references in [15] and pick a topic that interests you.

Recently, there has been some interesting work from the research group of Prof. W. Qian of Shanghai Jiao Tong University on Approximate computing. You can look at his publications at http://umji.sjtu.edu.cn/~wkqian/index.html, under the publications tab. The following topics might be intriguing: Two-level approx. logic synthesis [16], Multi-level approx. logic synthesis for area [17], same for delay optimization [18], and while we are at it, how about approx. bi-decomposition of Boolean functions [19]?

F. Logic Design/Synthesis for Hardware Security

We usually optimize digital circuits for area, delay, testability, power, etc. Now a days, a new problem has appeared – that of "secure computation". So security is being introduced as a metric for circuit synthesis. This is an emerging area of research, so there are not too many publications. If you are interested in browsing through this problem, here is an overview paper [20]. I am looking for other related logic synthesis papers in this area, so stay tuned for more info....

Another problem dimension is **logic locking to avoid piracy**. It is also a circuit design and logic synthesis/CAD problem. If you are interested in studying this problem, here is a journal paper [21] that describes the problem and gives a solution. For the class project, one could try to understand the concept of logic locking and implement a simplified version of the logic locking algorithm. Of course, there are quite a few logic locking techniques, a good list of references can be found in the same paper.

It looks like recently there have been SAT-solver based attacks on logic locking. So there are now publications, such as [22], trying to mitigate SAT attacks on such designs. On similar lines, there is approximate deobfuscation of ICs [23].

These are the new, glamorous applications of logic synthesis, they are worth a look.

G. Synthesis of Rectification Patches for Buggy Circuits, or for Engineering Change Orders – using Partial Logic Synthesis

When formal verification (equivalence checking) detects a bug in a design, it is required to **rectify** the circuit at some nets. Instead of resynthesizing the whole design to correct it, modern approaches perform logic synthesis locally (partial synthesis) at rectification target nets. This problem is similar to that of synthesis for engineering change orders (ECO) – where an implementation needs to be minimally modified to make it conform to an ECO-modified *spec*. The early work of [24] (which could be implemented with BDDs) was revisited using SAT and Craig Interpolation based models in [25] [26] [27]. The latest paper on this topic is that of [28].

These problems are not very hard, and these topics are very much doable as a class project!

H. For those who studied Gröbner Bases with me

Only for those of you who studied Gröbner bases with me (Fall 2014 or 2017), I may have some interesting topics for investigations. Primarily, these are related to building polynomial reduction algorithms $f \xrightarrow{f_1,\ldots,f_s}_+ r$, using ZDDs or using AIGs. Here, $f, f_1, \ldots, f_s$ are Boolean polynomials where the coefficients are 0 or 1, and $x^2 = x$ for all variables, and the computations are performed modulo 2. Therefore, these polynomials resemble AND-XOR circuits.

My students have investigated the use of ZDDs to compute Gröbner bases [29] or to perform $f \xrightarrow{f_1,\ldots,f_s}_+ r$ [30]. Analogously, you could consider if such operations could make use of AIGs as the data-structure. Actually, AIGs are not very efficient for XOR-dominated circuits. XOR-AND-invert graphs (called XAIGs) [31] [32] or Majority-inverter graphs (MIGs) [33] might be more suitable. Maybe you could try to implement such symbolic algebra procedures using XAIGs or MIGs? I think the libraries are available with us for use.

Why not use your past course (ECE 6745) to study the link to Logic Synthesis for Boolean polynomials modeled over the ring $\mathbb{F}_2[x_1, \ldots, x_n] \pmod{x_i^2 - x_i}$, where $\mathbb{F}_2 = \{0, 1\}$.

I. Logic synthesis and CAD for emerging technologies

Most logic optimization techniques that we consider are geared toward static CMOS. Due to the limits of CMOS scaling, the world is analyzing new, post-CMOS technologies for designing logic. These include integrated optics, reversible and quantum computing, among others. Some of the problems that we encounter in these technologies requires us to rethink the classical Boolean function model slightly. For example, reversible logic circuits need to have same number of inputs and outputs, and every gate needs to be reversible. See the work of Robert Wille at JKU Linz http://www.jku.at/iic/content/e289049. I also have a copy of Robert's thesis which has a good tutorial on reversible logic.

Similarly, in integrated optics, it is possible to build logical circuits using optical switches. But the technology requires a special case of XOR decomposition – see my papers on Logic Synthesis for Integrated Optics [34] [35]. Such problems can be formulated using K-maps, BDDs and SAT.

IV. Project Proposal Submission Guidelines

Each student, or group of students needs to submit a project proposal by March 9. The proposal should be a 1-2 page 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 synthesis, sequential circuit or fundamental combinatorial optimization techniques? Will you learn algebraic models 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 synthesis engine, say kernel extraction, 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 your best to narrow down the focus.

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.
6) What tools do you think you plan to develop, and if you already have some implementation ideas, you should write about it.
7) Try to project a realistic time-line about tasks/milestones you have decided to accomplish.
8) Finally, there's got to be a "division of labour" among the team-mates. Of course, each team member will have to perform the same preliminary study so all of you understand the problem being addressed; but when you get to implementation, not all of you have to do the same things. It is always good to agree upon each team member's responsibilities. It is okay to shuffle it around later based on any logistical issues, but it is good to put it in writing so that fairness in the division of labour is maintained.
9) It would be good to reflect on these issues in your final report — whether or not the project turned out to be the way you had planned it.

Do not worry if the 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! ☺.

References

[1] G. DeMicheli, *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, 94.

[2] S. Devadas, *Logic Synthesis.* McGraw-Hill, Inc., 1993.

[3] S. Yang and M. Ciesielski, "Optimum and Suboptimum Algorithms for Input Encoding and its Relationship to Logic Minimization," *IEEE Trans. on CAD*, pp. 4–12, Jan. 1991.

[4] S. Minato, "Calculation of Unate Cube Set Algebra using Zero-Suppressed BDDs," in *Proc. Design Automation Conference (DAC)*, 1994, pp. 420–424.

[5] ——, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," in *DAC*, 93, pp. 272–277.

[6] F. Somenzi, "Colorado Decision Diagram Package," Computer Programme, 1997.

[7] C. Yang and M. Ciesielski, "BDS: A BDD-Based Logic Decomposition System," *IEEE Trans. CAD*, vol. 21, no. 7, pp. 866–876, 2002.

[8] A. Mishchenko, B. Steinbach, and M. Perkowski, "An algorithm for bi-decomposition of logic functions," in *Proc. Design Automation Conference (DAC)*, 2001, pp. 103–108.

[9] J.-H. R. Jiang, C.-C. Lee, A. Mishchenko, and C.-Y. Huang, "To SAT or Not to SAT: Scalable Exploration of Functional Dependency," *IEEE Trans. Comp.*, vol. 59, no. 4, pp. 457–466, April 2010.

[10] R.-R. Lee, J.-H. R. Jiang, and W.-L. Hung, "Bi-Decomposing Large Boolean Functions via Interpolation and Satisfiability Solving," in *Proc. Design Automation Conference (DAC)*, 2008, pp. 636–641.

[11] ——, "To SAT or not to SAT: Ashenhurst Decomposition in a Large Scale," in *Proc. Intl. Conf. on CAD (ICCAD)*, 2008, pp. 32–37.

[12] M. Choudhury and K. Mohanram, "Bi-decomposition of large Boolean functions using blocking-edge graphs," in *Proc. Intl. Conf. on CAD, ICCAD*, 2010, pp. 586–591.

[13] "Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification," Available at htpp://www.eecs.berkeley.edu/alanmi/abc.

[14] S. Venkataramani, A. Sabne, V. Kozhikottu, K. Roy, and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in *Proc. Design Automation Conf.*, 2012, pp. 796–801.

[15] J. Miao, "Modeling and Synthesis of Approximate Digital Circuits," Ph.D. dissertation, Univ. of Texas, Austin, 2014.

[16] A. Su, C. Zou, W. Kong, J. Han, and W. Qian, "A Novel Heuristic Search Method for Two-Level Approximate Logic Synthesis," *to appear, in IEEE. Trans. on CAD*, 2019, available at:http://umji.sjtu.edu.cn/~wkqian/papers/Su_Zou_Kong_Han_Qian_A_Novel_Heuristic_Search_Method_for_Two_level_Approximate_Logic_Synthesis.pdf.

[17] Y. Wu and W. Qian, "An efficient method for multi-level approximate logic synthesis under error rate constraint," in *Proc. Design Automation Conference*, 2016, pp. 128:1–128:6.

[18] Z. Zhao, Y. Yao, S. Huang, S. Su, C. Meng, and W. Qian, "DALS: Delay-Driven Approximate Logic Synthesis," in *Proc. Intl. Conf. Computer-Aided Design (ICCAD)*, 2018, pp. 86:1–86:7.

[19] Y. Yao, S. Huang, C. Wang, Y. Wu, and W. Qian, "Approximate disjoint bi-decomposition and its application to approximate logic synthesis," in *Proc. Intl. Conf. Computer Design (ICCD)*, 2017, pp. 517–524.

[20] D. Demmler, G. Dessouky, F. Koushanfar, A. Sadeghi, T. Schneider, and S. Zeitouni, "Automated synthesis of optimized circuits for secure computation," in *Proc. SIGSAC Conf. Computer & Communications Security*, 2015, pp. 1504–1517.

[21] S. M. Plaza and I. Markov, "Solving the third-shift problem in ic piracy with test-aware logic locking," *IEEE Trans. on CAD*, vol. 34, no. 6, pp. 961–971, June 2015.

[22] X. Yang and A. Shrivastava, "Mitigating SAT-Attack on Logic Locking," in *Intl. Conf. on Cryptographic Hardware and Embedded Systems (CHES)*, 2016.

[23] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Pan, and Y. Jin, "AppSAT: Approximately Deobfuscating Integrated Circuits," in *Proc. Intl. Symposium on Hardware Oriented Security and Trust (HOST)*, 2017.

[24] C. C. Lin, K. C. Chen, S. C. Chang, and M. Marek-Sadowska, "Logic Synthesis for Engineering Change," in *Proc. Design Automation Conf. (DAC)*, 1995, pp. 647–652.

[25] K. F. Tang, C. A. Wu, P. K. Huang, and C. Y. Huang, "Interpolation-Based Incremental ECO Synthesis for Multi-Error Logic Rectification," in *Proc. Design Automation Conf. (DAC)*, 2011, pp. 146–151.

[26] S. Jo, T. Matsumoto, and M. Fujita, "SAT-Based Automatic Rectification and Debugging of Combinational Circuits with LUT Insertions," in *IEEE 21st Asian Test Symposium*, 2012.

[27] M. Fujita, "Toward Unification of Synthesis and Verification in Topologically Constrained Logic Design," *Proceedings of the IEEE*, 2015.

[28] A. Dao, N.-Z. Lee, L.-C. Chen, M. Lin, J.-H. Jiang, A. Mishchenko, and R. Brayton, "Efficient Computations of ECO Patch Functions," in *Proc. Design Auto. Conf.*, 2018.

[29] M. Brickenstein and A. Dreyer, "PolyBoRi: A Framework for Gröbner-basis Computations with Boolean Polynomials," *Journal of Symbolic Computation*, vol. 44, no. 9, pp. 1326–1345, 2009.

[30] U. Gupta, P. Kalla, and V. Rao, "Boolean Gröbner Basis Reductions on Datapath Circuits Using the Unate Cube Set Algebra," *IEEE Trans. CAD*, vol. 38, no. 9, pp. 576–588, March 2018.

[31] I. Háleček, P. Fišer, and J. Schmidt, "Towards and/xor balanced synthesis: Logic circuits rewriting with xors," *Microelectronics Reliability*, vol. 81, pp. 274–286, Feb 2018.

[32] ——, "Are xors in logic synthesis really necessary?" in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2017, pp. 134–139.

[33] L. Amaru, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A new paradigm for logic optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 806–819, 2016.

[34] C. Condrat, P. Kalla, and S. Blair, "Exploring Design and Synthesis for Optical Digital Logic," *International Workshop on Logic Synthesis*, 2010.

[35] ——, "Logic Synthesis for Integrated Optics," *Great Lakes Symposium on VLSI*, 2011.