Boolean Gröbner Basis Reductions on Datapath Circuits using the Unate Cube Set Algebra

Utkarsh Gupta, Priyank Kalla, Vikas K. Rao

Electrical and Computer Engineering
University of Utah, Salt Lake City, UT 84112
utkarsh.gupta@utah.edu, kalla@ece.utah.edu, vikas.k.rao@utah.edu

Response to Reviewer's Comments

We are thankful to the reviewers for the detailed comments and the suggestions for improving our submission. We have made appropriate revisions to the manuscript based on the comments: $i)$ corrected the typographical errors; $ii)$ changed the structure of the section on limitations of our approach; $iii)$ furnished details to the theory as suggested $iv)$ provided answers to questions related to experiments.

The edits made in the manuscript are highlighted in red.

Following are the response to the reviewers and description of the revisions.

Response to Reviewer 1

A very well written paper that clearly explains and summarizes current computer algebra approach to formal verification of arithmetic circuits.

The main contribution of the paper is the use of ZBDDs which, by imposing the reverse topological term order exposes useful divisors for polynomial reduction. This allows one to cancel multiple monomials at a time, hence contributing to increased efficiency. The Experimental Results are strong, showing a number of good, practical designs. Here are my detailed comments.

Comment 1: Definitions missing: The term "Boolean polynomial" is used several times before it is formally defined on page 3; this should be done earlier. Also, the paper does not define the term "pseudo-Boolean polynomial", used freely in the paper several times.

Response: As defining these terms without the notations introduced in section III is difficult, we have provided informal definitions for these terms when these terms are first introduced in the manuscript (excluding the abstract).

Comment 2: The only real weak part of the paper is Section VI E, page 12, "Limitations of the approach ...", which is difficult to grasp at the first reading. In particular, the following sentence (lines 52-55) should be rewritten:
" Performing a detailed analysis reveals that for integer arithmetic datapath circuits, a bit-level reduction is not sufficient for verification efficiency, and a word-level approach is required."
I don't think that "not sufficient" correctly explains the problem.
It should basically say that it cannot be applied directly to integer arithmetic circuits, explain the reason (it performs reduction for each bit separately), and give your example.
I think it would help if you just changed the structure of this section; first show how inadequate your approach, designed specifically for finite field circuits, is for integer arithmetic, citing the number of monomials. Then, conclude that a better approach could be to perform the reduction on the word level, and show the reduction of word Z.

Response: We have changed the structure of this subsection. We have first described the reasoning behind the limitation of our GBR approach on integer arithmetic circuits and hinted that a word-level approach is better suited for them. We then summarize the experiment we performed on the 7x7 integer array multiplier providing the information on the number of monomials generated during the GBR of z_{13} and z_{12}. Lastly, we explain why a word-level approach is needed for integer arithmetic circuits as it can cancel the non-linear terms early during GBR and avoid intermediate blow-up.

Comment 3: Page 4, Definition III.2: Replace "there exists $i \in 1, ...t$ such that ..." into "there exists $g_i \in g_1, ...g_t$ such that ...". It is the existence of the polynomial g_i and not its index that is important.

Response: That was indeed the intention of the older definition. We have made the correction as suggested.

Comment 4: Page 4, line 26 right (below the theorem): the sentence "for all polynomials f" should read "for any polynomial f .." Here f is a single item, so it cannot be "polynomials".

Response: It is indeed a typo. We have made the correction.

Comment 5: Similarly, on page 5, line 9 (in Proposition III.2), "Let C be any arbitrary circuit" should be changed to "Let C be an arbitrary circuit" or "Let C be any circuit" to avoid redundancy (any / arbitrary).

Response: We have replaced the original sentence with "Let C be an arbitrary circuit".

Comment 6: Page 4, line 34 (in Proposition III.1): ”denote k-bit primary output variables” is incorrect. You probably meant ”denote one-bit of the k-bit primary output (word) ...”.

Response: Yes, we meant one bit of k-bit primary output. We have made the appropriate correction.

Comment 7: Page 7, Fig.6. It would be nice to show the final ZBDD; the current figure leaves the logical sum operation on the two subgraphs unfinished.

Response: We have modified the Fig. 6 to show the final ZBDD obtained from the sum operation.

Comment 8: Page 7, line 52 right: ”Reduction $f \rightarrow r$ requires to obtain” should be changed to ”Reduction $f \rightarrow r$ requires one to obtain”.

Response: We have made the suggested correction.

Comment 9: Page 8, lines 24-25 right. The statement ”if indices of top-most nodes of ZBDDs of z_i and f_i are equal, then $lm(f_i)$ divides $lt(z_i)$”. This is not obvious and should be formally proven, or at least you should provide intuition behind it.

Response: The equality of indices indicate that the top variables are the same (as indices are unique). In addition, as f_i is guaranteed to have that variable only (leading term is a singleton with only one variable) in the leading term, f_i can divide z_i. We have furnished this detail in the manuscript.

Comment 10: Later on the same page, Example V2, you say: ”As $index(top\,var(f2)) = index(top\,var(r1))$, $lm(f2)|lm(r1)$. In view of the earlier statement, you should have $lm(f2)|lt(r1)$, even if one understands that in Boolean polynomials, with coefficients in F_2, $lm(r1) = lt(r1)$. (Definition to the rescue !!) But to be mathematically precise, the notation should be consistent.

Response: We have changed the notation so that it is now consistent with the earlier use.

Comment 11: Page 1, lines 14-16 right: ”Reducing the primary outputs ...” should probably read ”Reducing the primary output polynomials ...

Response: Indeed the primary output polynomials (which are actually just output variables $= 0$ $i.e.$ $z_i = 0$) are reduced by the mentioned GB. We have made the correction to indicate this.

Comment 12: Equation on page 12, line 39 has an obvious typo: $4x_1 + 2x_2 + 2x + 3 + 4_4 \rightarrow 4x_1 + 2x_2 + 2x_3 + x_4$

Response: Yes it had a typo. We have made the correction.

Comment 13: Page 1, Abstract: ”we show that that imposition” ? $\rightarrow$ ”we show that (the) imposition”.

Response: We have made the suggested revision.

Response to Reviewer 2

The paper presents a method for Boolean GB reduction using ZBDD that is ordered by the circuit reverse topological order of the gates.

The paper is very well written and it is very easy to follow.

The experimental results on some cryptography circuits shows the advantages of the method in those circuit.

I have the following questions regarding the results:

Comment 1: The circuits that are used in the experiments are first optimized with ABC and then the ZBDD is generated. My question is about the role of logic optimization in the efficiency of the method. Given two circuits for equivalency checking, one of them optimized and the other one non-optimized, to the method, how the runtime differs for the two circuits?

Response: The benchmarks from [5] had lot of buffers and redundant variables. Since ZBDD is a complex data-structure, defining these unnecessary variables increased the runtime and memory consumption of GBR. Secondly, as mentioned in [15], the reduction of a output bit of a Galois field multiplier depends only on its logic cone, therefore, bit-level logic optimization reduces the GBR time. But optimization not only reduces GBR time for our approach but also for the approach in [15] and the PolyBori tool.

Comment 2: As the authors stated, the method may not work well in data path circuits based on integer arithmetics operations, and the reason seems having many XOR gates? I think it needs more discussion/elaboration.

Response: The GBR approach introduced in the manuscript is not directly applicable to the integer arithmetic circuits. Our approach performs a bit-level reduction of the output variables one at a time. The logic sharing among the output variables in the case of integer arithmetic circuits generates a large number of non-linear terms that leads to intermediate blow-up in the number of monomials. On the other hand, as explained using the Fig. 9 in subsection VI E, a word-level reduction can cancel these non-linear terms early in the GBR and controls the intermediate monomial explosion. We have changed the structure of this subsection as suggested by Reviewer 1 and it now does a better job of explaining the limitations of our approach and a possible solution.

Reviewer 3

This paper explores the application of computer algebra techniques on verification problems. In particular, equivalence checking of (field) arithmetic circuits is considered. Such circuits form the basis of many of cryptographic applications and thus formal methods for proving their correctness are important. The arithmetic nature of these circuits makes their formal verification with standard formal methods ineffective.

In recent years, methods based on computer algebra have shown promising results for these types of circuits and also the paper discussed here follows these lines. It chooses the usual approach of converting the circuit into a set of polynomials and choosing a term-order based on the topology of the circuit.

Here the paper borrows results from the existing literature that Gate-wise translation of a circuit into polynomials under this term-order results in a set of polynomials that is already a Grbner-Basis (GB) of its generated Ideal. In this manner decisions problems can be solved applying reduction algorithms directly without costly GB computations.

Like the public domain tool PolyBori, a general purpose algebra tool for boolean field algebra, the paper uses ZBDDs to represent polynomials. The authors however, restrict themselves to the above mentioned term order and can thus come up with a simplified version of the reduction algorithm that outperforms the existing methods.

The approach is evaluated on a comprehensive set of benchmarks and compared against PolyBori, prior work from the authors and a third approach using the Nullstellen-Satz. The experiments confirm that the simplified reduction technique outperforms existing algorithms.

The paper is well written and gives the right credits to borrowed ideas from existing literature and thus may serve as an overview over the existing work in the field. The key new insight of the paper is the simplified reduction algorithm. Thus technically the paper is rather a small delta with nonetheless a substantial gain in performance.

Response to Reviewer 4

In this paper, a novel approach is proposed to speed up the process of Groebner basis reduction in verification of Galois field arithmetic circuits. The authors take advantage of unate cube set and Boolean polynomials and convert all the operations on the polynomials to operations on ZBDDs. They have also proposed a new algorithm on ZBDDs to perform multiple divisions in just one step. They demonstrate their algorithms on several examples and finally provide an experimental evaluation on finite field multipliers which are highly relevant in crypto applications.

The idea of the paper is very interesting and the paper is very well written. The experimental results show significant improvement compared to other state-of-the-art approaches. However, the following points should be improved:

Comment 1: You have shown in the experimental results that the proposed method is only suitable for finite field (Galois field) arithmetic circuits and it fails for integer multipliers. Hence, I suggest to add "finite field datapath circuits" or "Galois field datapath circuits" explicitly to the title of the paper.

Comment 2: You mentioned in the paper, and it also can be concluded from Fig. 5, ZBDDs need less memory in comparison with classical presentation of polynomials. However, it would be nice to provide some (peak) memory number also in the experiments.

Response: We have provided the memory consumption (range) for the cases of 64-bit and (to) 571-bit Mastrovito/Montgomery multipliers. These values are for the case of sequential execution and give an idea about the memory consumption of other multiplier benchmarks. In addition, these values were also used to determine the number of parallel processes that can be executed simultaneously in the case of parallel execution.

Comment 3: Page 9, eq (15): you develop/conclude this equation from an example. Why is it correct in the general case?

Response: The equation is correct in general due to the structure of ZBDDs under RTTO. Let's understand it using some arbitrary polynomials r_1, f_2, and r_2 (unlike in the manuscript they can be any polynomials but their ZBDDs use the RTTO order) where we want to perform the operation $r_1 \xrightarrow{f_2}_+ r_2$. Note that the top variables of the ZBDDs of r_1 and f_1 are same and for this discussion we assume that it is y. The operation is carried out as follows,

$$r_2 = r_1 + quotient \cdot f_2 \quad \text{(where } quotient = \text{quotient of division of } r_1 \text{ by } f_2)$$
$$r_2 = y \cdot then(r_1) + else(r_1) + quotient \cdot (y + else(f_2))$$

notice that $quotient = then(r_1)$ as f_2 can only divide $y \cdot then(r_1)$ portion of r_1 $(= y \cdot then(r_1) + else(r_1))$,

$$r_2 = y \cdot then(r_1) + else(r_1) + then(r_1) \cdot (y + else(f_2))$$
$$r_2 = 2 \cdot y \cdot then(r_1) + else(r_1) + then(r_1) \cdot else(f_2)$$
$$r_2 = else(r_1) + then(r_1) \cdot else(f_2) \quad (\text{as } 2 \pmod 2 = 0)$$

We have arrived at the same equation using general r_1, r_2, and f_2.

Comment 4: Page 10, right column, line 41, the range of 'k' is not correct.

Response: This was a typo. The range is for i and not k where $0 \geq i \geq k-1$. We have made the correction in the manuscript.

Comment 5: For the second experiment in Table III, the verification time for Montgomery multiplier with 283 bit data size is considerably higher than other testcases. Please clarify what is the reason and why the run-time of your proposed method is higher than [15] just for this benchmark.

Response: The verification time for some Montgomery benchmarks is not in accordance with their bit-widths. For example, 163-bit multiplier reduction takes more time than 233-bit. Similarly 283-bit multiplier takes more time than 409-bit. The reason is that the bit-width of the circuit is not the only factor controlling the reduction cost. The irreducible polynomial $P(x)$ also plays an important role in the designing of the multiplier and consequently in the reduction as explained in [15]. We have included this information in the Montgomery multiplier subsection of Experiments section.

Comment 6: Table VI + VII: time unit is missing

Response: We have updated the Table titles (VI and VII) to include the time units.

Comment 7: Page 6, right column, line 54: number _of_ iterations

Response: We have included the missing "of".

Comment 8: Page 8, Example V.1 and Algorithm 3: use \mathit{ite} for "ite"

Response: The updated algorithm and example now uses the \mathit{ite} instead of just "ite".