Influence of the Reduction Order in Multiplier Verification using Computer Algebra

Daniela Kaufmann
Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria
daniela.kaufmann@jku.at

***Abstract*—Currently the most promising approach for automatically verifying multiplier circuits relies on computer algebra. In this approach the circuit as well as its specification are modeled as polynomials and a polynomial reduction algorithm is applied to show correctness. We want to elaborate on the influence of the reduction order by employing a wide range of valid orderings. In our experiments we measure and compare the size of the intermediate reduction results and gain a clear preference towards a column-wise ordering.**

I. Introduction & Preliminaries

The state of the art approach for fully automated verification of gate-level multipliers, calculating $S = A \cdot B$, is based on polynomial reasoning [1], [3], [4], [5], [6], [7].

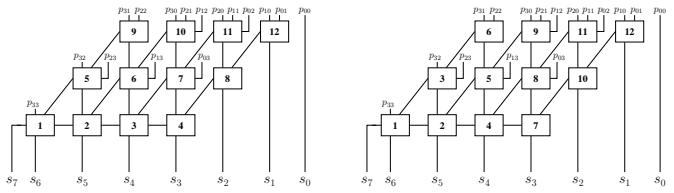
In this method each logical gate of the circuit is represented by a polynomial, describing the relation of the output and inputs of the gate. Additionally the specification of the multiplier is modeled as a polynomial. Correctness of the circuit is shown by reducing the specification polynomial by the gate polynomials, using polynomial division, until no further reduction is possible. The multiplier is correct if and only if the reduction returns zero. More details on the polynomial encoding are given, for instance, in [4].

The main issue of the algebraic approach is that without preprocessing the size (number of monomials) of the intermediate reduction results increases drastically, leading to a slow-down in the verification time. This was for instance analyzed in [5].

In order to overcome the issue of the monomial blow-up related works propose various heuristics which rewrite the polynomial representation of the circuit before reduction is applied [1], [3], [4], [6]. Although using different approaches, a commonality is to explicitly or implicitly detect full- and half-adders in the circuit. The polynomials of the internal adder gates are eliminated and only polynomials which define the relation between the outputs and inputs of an adder remain. This allows cancellation of common non-linear monomials.

After preprocessing the specification polynomial is reduced by the rewritten gate polynomials. The choice of the reduction order does not influence the correctness of the algorithm. However it is conjectured that it has great impact on the size of the intermediate reduction results. The authors of [7] show an experiment, where two reduction orderings for simple multiplier architectures are compared. However in this experiment no preprocessing is applied.

Supported by Austrian Science Fund (FWF), NFN S11408-N23 (RiSE), P31571-N32, SFB F5004, LIT AI Lab funded by the state of Upper Austria.



(a) Row-wise Order (b) Column-wise Order

Fig. 1. Different orders of full- and half-adders, with $p_{ij} = a_i b_j$.

In this work we elaborate on the impact of the reduction order after preprocessing, in particular after rewriting full- and half-adders in the circuit. In our experiments we select a simple and a complex multiplier architecture and choose different (arbitrary) reduction orderings and measure the size of the intermediate reduction results.

II. Reduction Order

Gates in acyclic circuits, such as multiplier circuits, can be ordered according to their reverse topological level. The reduction order of the corresponding gate polynomials should follow such a reverse topological ordering. This ensures that after a gate polynomial is used for reduction it never has to be considered again [7].

Given the shape of multipliers, two orderings seem natural, namely a column-wise and a row-wise ordering. Both orderings are depicted in Fig. 1 for a simple 4-bit carry-save-adder multiplier. The multipliers in Fig. 1 have been preprocessed, such that only full- and half-adders remain as boxes.

The idea of the row-wise order, cf. Fig. 1a, is to order the gates according to their backward level seen from the circuit inputs. The approaches of [1], [6], [7] use such an ordering.

In the column-wise order, cf. Fig. 1b, the multiplier is sliced vertically, such that each slice contains one output bit. The gates in the slices are ordered according to their topological level seen from the output bit. In our work [3], [4] we use a column-wise ordering, which allowed us to introduce an incremental checking method in [4]. In this approach the correctness of the circuit is shown by incrementally verifying each slice. The main advantage is that we always use only one small part of the global specification for reduction. However the column-wise reduction order remains the same.

The reduction orderings are not limited to these two corner cases, circuits support various arbitrary reverse topological orderings, e.g. in Fig. 1a the ordering $1 > 2 > 5 > 3 > 4 > 6 > 7 > 9 > 8 > 11 > 10 > 12$ is also reverse topological.

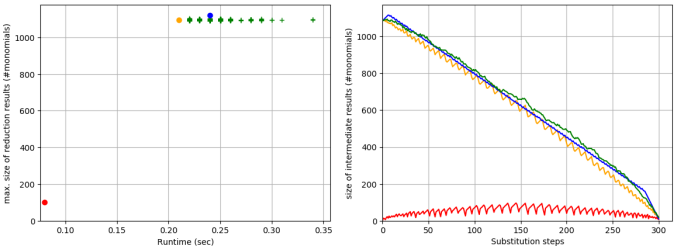


Fig. 2. Effect of reduction order on sp-ar-rc benchmarks

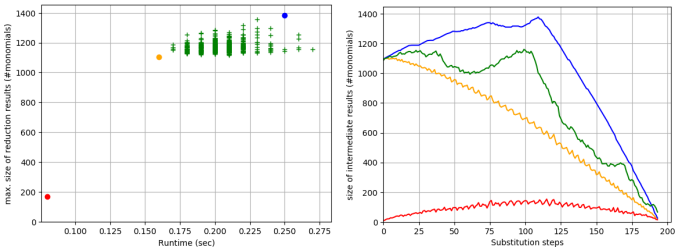


Fig. 3. Effect of reduction order on bp-wt-rc benchmarks

III. Experiments

In our experiments we compare row-wise and column-wise reduction orders to arbitrary reverse topological reduction orders. We select two different multipliers of the AOKI benchmarks [2] for our experimental evaluation. We perform preprocessing, as in [3], and measure the size and time of the final reduction process. All our experiments are run on a standard Ubuntu 16.04 Desktop machine with Intel i7-2600 3.40GHz CPU and 16 GB of main memory. Our experimental data is available at http://fmv.jku.at/redorder.

In our first experiment we select a simple 32-bit "sp-ar-rc" multiplier. The 4-bit version of this architecture is depicted in Fig. 1. The results can be seen in Fig. 2. The left side of Fig. 2 shows the time (in seconds) needed to verify the multiplier and the maximum size (number of monomials) of the intermediate reduction results of a row-wise (blue) and column-wise (orange) order and 500 arbitrary reverse topological orderings (green). Additionally we list the size and time of our incremental column-wise (red) approach.

It can be seen that the non-incremental approaches are in the same order of magnitude, i.e., the sizes span a range of around 30 monomials. However the incremental column-wise approach produces by far the smallest and fastest result, because it never considers the complete global specification. The right side of Fig. 2 shows the development of the size of the intermediate results for a complete reduction run. We only show the results of one of the 500 arbitrary orderings. Again the non-incremental orders behave similarly, but are out-rivaled by the incremental column-wise approach.

In our second experiment we consider a more complex 32-bit multiplier "bp-wt-rc", which uses Booth-encoding for generating the partial products and where the full- and half-adders are arranged in a Wallace-tree structure. Wallace-tree multipliers are faster than simple carry-save-adder multipliers, but the arrangement of the full- and half-adders is more involved. The results of this experiment can be seen in Fig. 3, where we use the same color-scheme as in Fig. 2.

In contrast to the "sp-ar-rc" multiplier, there is a gap of around 300 monomials between the column-wise and row-wise ordering. The sizes of the arbitrary reverse topological orderings are in between. Only the column-wise order has a linearly decreasing trend during reduction. Again our incremental approach outperforms the non-incremental approaches.

Interestingly in both experiments the row-wise ordering caused the biggest intermediate results.

IV. Conclusion & Future Work

In this work we investigated the effect of the reduction order after preprocessing. We compared the sizes of the intermediate reduction results as well as the reduction time for column-wise, row-wise or arbitrary reverse topological orderings.

Our experiments show that the effect of reduction order highly depends on the circuit architecture. For simple architectures there is almost no difference between the various non-incremental orderings. On the other hand the reduction order has a tremendous impact for complex multipliers. Interestingly in both experiments the column-wise ordering was very stable, and especially for complex multipliers, there is a clear trend to select a column-wise ordering for polynomial reduction.

However all non-incremental approaches are outperformed by our incremental column-wise approach [4]. Thus these experiments further support our observation, that an incremental approach, where the specification is divided into multiple polynomials helps to speed up computation.

In the future we want to further investigate the influence of the reduction order. We want to be able to understand what causes the differences in the intermediate reduction results, as this may help to successfully apply multiplier verification to synthesized multipliers. The structure of such multipliers is highly optimized, having the effect that the full- and half-adders cannot be extracted. This issue leads to a blow-up in the size of the reduction results. Thus synthesized multipliers still impose a big challenge for automated verification.

We would like to thank Alan Mishchenko for bringing up and discussing this interesting research question.

References

[1] M. Ciesielski, T. Su, A. Yasin, and C. Yu. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: a Bit-Flow Model. *IEEE TCAD*, 2019.

[2] N. Homma, Y. Watanabe, T. Aoki, and T. Higuchi. Formal Design of Arithmetic Circuits Based on Arithmetic Description Language. *IEICE Transactions*, 89-A(12):3500–3509, 2006.

[3] D. Kaufmann, A. Biere, and M. Kauers. Verifying Large Multipliers by Combining SAT and Computer Algebra. To appear in FMCAD'19.

[4] D. Kaufmann, A. Biere, and M. Kauers. Incremental Column-wise verification of arithmetic circuits using computer algebra. *FMSD*, 2019.

[5] A. Mahzoon, D. Große, and R. Drechsler. PolyCleaner: clean your polynomials before backward rewriting to verify million-gate multipliers. In *ICCAD*, page 129. ACM, 2018.

[6] A. Mahzoon, D. Große, and R. Drechsler. RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers. In *DAC*, pages 185:1–185:6. ACM, 2019.

[7] C. Yu, W. Brown, D. Liu, A. Rossi, and M. J. Ciesielski. Formal Verification of Arithmetic Circuits by Function Extraction. *IEEE TCAD*, 35(12):2131–2142, 2016.