Late Breaking Results: Towards Efficient Formal Verification of Dot Product Architectures

Lennart Weingarten
Institute of Computer Science
University of Bremen
Bremen, Germany
len_wei@uni-bremen.de

Kamalika Datta
Institute of Computer Science
University of Bremen/DFKI
Bremen, Germany
kdatta@uni-bremen.de

Rolf Drechsler
Institute of Computer Science
University of Bremen/DFKI
Bremen, Germany
drechsler@uni-bremen.de

***Abstract*—The popularity of compute intensive applications, like AI/ML, has driven the design of processors with complex functionality. The *Dot Product* (DP) is one of the most essential operations in modern neural processors, although no complete formal verification technique exists that can ensure its 100% correctness. In this paper we show the first step towards formally verifying DP using *Symbolic Computer Algebra* (SCA). The verification process is performed without the need of a reference model generation which is a key factor in verification. Experimental results show the efficiency and scalability of SCA-based verification for DP architectures.**

***Index Terms*—Dot Product (DP), Formal Verification, Symbolic Computer Algebra (SCA)**

I. Introduction

In the ever evolving world of hardware development, together with the growth of AI, ensuring correctness of a design is of vital importance. One of the fundamental blocks in many AI and DSP applications is the *Dot Product* (DP) unit. Verification of such blocks is essential to avoid costly errors. While there exists a few design approaches for DP [1], [2], none can provide a complete formal verification that can ensure 100% correctness. In a recent work, an approach to formally verify the *Dot Product Accumulate Systolic Unit* (DPA) has been presented [3]. Equivalence checking is used to verify the RTL code against a golden reference model, which is yet to be proven complete. In this paper a verification approach is presented that does not require a complex reference model. Here, we verify the circuit by comparing it with the *Specification Polynomial* (SP), which is a simple high-level representation. *Symbolic Computer Algebra* (SCA) (see [4]–[7]) is found to be the best proof engine for formal verification of scalable multipliers. So far no complete formal verification techniques exist for DP. In DP, many multiplier terms are present, and as SCA-based methods are well suited in this case, we perform formal verification of DP architectures exploiting SCA.

II. Verification Methodology for Dot Product

A. Dot Product Architecture

The DP is realized by adding multiple product terms resulting in a single value. Eqn. (1) represents the DP of m product terms.

$$DP := (a_0 \times b_0) + (a_1 \times b_1) + \cdots + (a_{m-1} \times b_{m-1}) \quad (1)$$

Each term consists of a multiplication of two n-bit numbers, resulting in a product of $2n$-bits. The addition is performed by an adder tree structure shown in Fig. 1. Initially the first two terms are added, thereafter further terms are added one by one until the final result is calculated. Each addition generates an additional carry out, which can be added as a carry input to the next adder (as shown in Fig. 1). In this work, the carry bit is truncated.

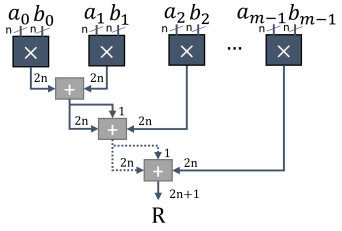

Fig. 1: DP architecture

B. Application in Matrix Vector Multiplication Operation

The DP is part of many complex arithmetic operations which are essential for AI applications, like e.g. *Matrix-Vector Multiplication* (MVM), *Matrix-Matrix Multiplication* (MMM) and *Multiply-Accumulate* (MAC). Eqn. (2) shows an example for MVM. The number of DP terms varies depending on the operation.

$$\begin{bmatrix} m_{[0,0]} & m_{[0,1]} & \cdots & m_{[0,k]} \\ m_{[1,0]} & m_{[1,1]} & \cdots & m_{[1,k]} \\ \vdots & & \ddots & \vdots \\ m_{[j,0]} & m_{[j,1]} & \cdots & m_{[j,k]} \end{bmatrix} \times \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_k \end{bmatrix} \quad (2)$$

Each r_k entry (result vector) from Eqn. (2) represents individual DPs, consisting of many product terms for a particular row matrix, as can be seen in Eqn. (3).

$$\begin{aligned} r_1 &= (m_{[0,0]} \times v_0) + (m_{[0,1]} \times v_1) + \cdots + (m_{[0,k]} \times v_k) \\ r_2 &= (m_{[1,0]} \times v_0) + (m_{[1,1]} \times v_1) + \cdots + (m_{[1,k]} \times v_k) \\ &\vdots \\ r_k &= (m_{[j,0]} \times v_0) + (m_{[j,1]} \times v_1) + \cdots + (m_{[j,k]} \times v_k) \end{aligned} \quad (3)$$

For all r_j in Eqn. (3) the SP can be expressed as shown in Eqn. (4). Provided that each DP can be verified using SCA, then the entire MVM in Eqn. (2) can be completely verified. Similarly, other operations like MMM or MAC can also be handled.

$$\begin{aligned} SP_{r_1} &:= r_1 - (m_{[0,0]} \times v_0) - (m_{[0,1]} \times v_1) - \cdots - (m_{[0,k]} \times v_k) = 0 \\ SP_{r_2} &:= r_2 - (m_{[1,0]} \times v_0) - (m_{[1,1]} \times v_1) - \cdots - (m_{[1,k]} \times v_k) = 0 \\ &\vdots \\ SP_{r_k} &:= r_k - (m_{[j,0]} \times v_0) - (m_{[j,1]} \times v_1) - \cdots - (m_{[j,k]} \times v_k) = 0 \end{aligned} \quad (4)$$

C. Verification Methodology

The first step in SCA-based verification is the definition of the SP of the circuit. The SP is described solely on the basis of the primary inputs and outputs of the circuit. Eqn. (5) shows the SP of the DP consisting of m product terms.

$$SP_{DP} := R - (a_0 \times b_0) - (a_1 \times b_1) - \cdots - (a_{m-1} \times b_{m-1}) = 0 \quad (5)$$

978-3-9826741-0-0/DATE25/© 2025 EDAA

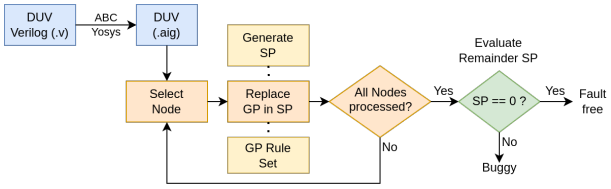

Fig. 2: Overview of Verification Methodology

Fig. 2 shows the overall verification methodology of our tool `DP-Verifier`. The *Device Under Verification* (DUV) given in Verilog is converted into an *And-Inverter-Graph* (AIG) representation, and the SP is generated. The verification is then performed by iterating from primary outputs to primary input in reverse topological order. For each AIG node, its *Gate Polynomial* (GP) is evaluated by replacing it in the SP (see [5] for more details of the substitution process). After all nodes are processed, the final remainder polynomial is evaluated. The circuit is correct, if it is a zero polynomial, otherwise it is faulty.

III. Experiments

The SCA-based `DP-Verifier` is implemented using C++. All experiments were performed on an AMD Ryzen 7 PRO 4750U with 40GB main memory. We generate two variations of the DP architecture for 8 and 16-bit.

- DT_SE_RC: an unsigned *Dadda Tree* (DT) multiplier with *Serial prEfix* (SE) adder for the final stage and *Ripple Carry* (RC) for the adder tree
- AR_RC_RC: an unsigned *ARray* (AR) multiplier with RC for the final stage adder and for the adder tree

To show the efficiency and scalability of SCA-based verification for DP architectures the experimental results for two example DP circuits are presented in Fig. 3. Each multiplier term has a bit width of 8 or 16 bits.

In the top and bottom sub-figures the verification time and the maximum specification polynomial size is presented, respectively. For all experiments, product terms ranging from 2 to 128 are evaluated. This implies that we can handle DPs with up to 4096 bits. For all sub-figures the x-axis shows the number of product terms. In the top figures the y-axis gives the verification time (vTime) in seconds and in the bottom the maximum SP size.

From this diagram it can be inferred that both DP designs have similar behavior. The verification time shows polynomial growth whereas the maximum SP size grows linearly only. The SP size for larger terms is correlated to the memory usage, which grows linear, whereas the verification time grows polynomial. This shows the power of the SCA-based verification methodology for complex circuits. While dealing with an NP-complete problem, the solutions provided turn out to be very efficient.

The substitution trajectory for 8 and 16 bit DP architectures is depicted to show how the SP size grows over the substitution steps. The top diagrams in Fig. 4 show the DP with the smallest product terms (2) and the bottom with the largest product terms (128). It can be observed that the trajectory curve behaves identically for larger numbers of product terms, which validates the scalability of SCA-based verification for DP architecture.

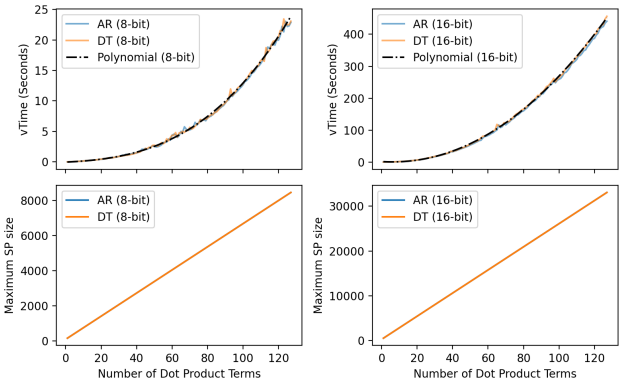

Fig. 3: Verification results for 8 and 16-bit dot product

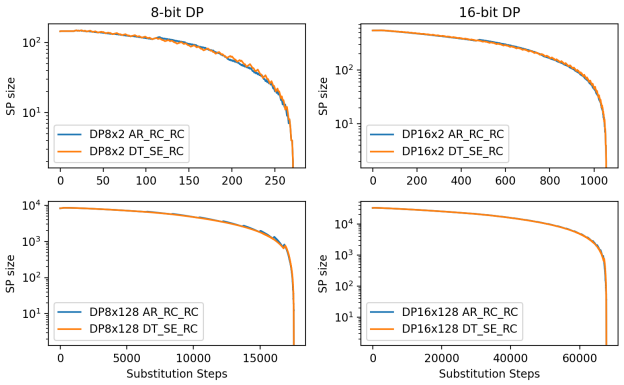

Fig. 4: Substitution trajectory for 8 and 16 bit DP

IV. Conclusion

In this work, we have shown the suitability and scalability of SCA-based verification for DP architectures. Circuits where up to 128 product terms (4096 bits) could be verified efficiently in a very short time. This shows the power of our approach which can verify large and complex DP circuits without the need for a reference model. As future work verification of MVM and MMM could be considered.

Acknowledgment

This work was supported in part by DFG within the Reinhart Koselleck Project PolyVer (DR 287/36-1) and partly by the German Federal Ministry of Education and Research (BMBF) within the ECXL project under grant no. 01IW22002.

References

[1] S. Prebeck *et al.*, "A Scalable, Configurable and Programmable Vector Dot-Product Unit for Edge AI," in *MBMV*, 2022, pp. 1–9.
[2] D. Puri *et al.*, "Raising the Bar: Achieving Formal Verification Sign-Off for Complex Algorithmic Designs, with a Dot Product Accumulate Case Study," in *DVCon India*, 2023.
[3] E. Morini *et al.*, "Achieving End-to-End Formal Verification of Large Floating-Point Dot Product Accumulate Systolic Units," *DVCon*, 2024.
[4] D. Kaufmann, A. Biere, and M. Kauers, "Verifying large multipliers by combining SAT and computer algebra," in *FMCAD*, 2019, pp. 28–36.
[5] A. Mahzoon, D. Große, and R. Drechsler, "RevSCA-2.0: SCA-Based Formal Verification of Nontrivial Multipliers Using Reverse Engineering and Local Vanishing Removal," *TCAD*, vol. 41, no. 5, 2022.
[6] A. Konrad and C. Scholl, "Symbolic Computer Algebra for Multipliers Revisited-It's All About Orders and Phases," in *FMCAD*, 2024.
[7] H. Liu *et al.*, "Parallel Gröbner Basis Rewriting and Memory Optimization for Efficient Multiplier Verification," in *DATE*, 2024, pp. 1–6.