Late Breaking Results: Polynomial Formal Verification of Fast Adders

Alireza Mahzoon[1] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
{mahzoon,drechsle}@informatik.uni-bremen.de

***Abstract*—Despite the recent success of formal verification methods, the computational complexity of most of them is still unknown. It raises serious questions regarding the scalability of the approaches. One of the most successful formal methods to prove the correctness of adders is Binary Decision Diagram (BDD)-based verification. It reports very good results for verification of different adder architectures. However, the computational complexity of BDD-based verification has not been yet fully investigated. In this paper, we calculate the complexity of verifying one of the fastest available adders, i.e., conditional sum adder. Then, we show that the verification of this architecture is possible in polynomial time. Finally, we confirm our theoretical calculations by experimental results.**

I. Introduction

The importance of arithmetic circuits is rapidly growing due to the demands for complex and extensive computations in modern systems. These circuits are usually complicated; thus, proving the correctness of them before implementation is of the utmost importance to avoid bugs. Several formal verification methods based on *Binary Decision Diagrams* (BDDs) [1], *Binary Moment Diagrams* (BMDs) [2], and *Symbolic Computer Algebra* (SCA) [3], [4] have been proposed to check the correctness of arithmetic circuits. They usually report very good results when it comes to the verification of sophisticated architectures.

Despite the practical success of formal verification methods, the computational complexity of most of them is still unknown. A good example of the unknown complexity is the formal verification of integer adders using BDDs. It has been shown in practice that BDDs are very efficient in proving the correctness of adders. However, the computational complexity of many adder architectures has not been yet calculated. To the best of our knowledge, PolyAdd [5] is the only work on the verification complexity of adders. It proves that verifying some adder architectures is possible in polynomial time. However, it does not provide the exact order of complexities.

The conditional sum adder is one of the important architectures whose verification complexity has not been yet fully investigated. It has the smallest delay among the adders thanks to its unique structure containing many multiplexers [6]. Therefore, it is the first choice for the designs in which the delay is the most important parameter. In this paper, we calculate the computational complexity of verifying a conditional sum adder using BDDs. We also prove that verifying this architecture is possible in polynomial time. Finally, we compare the theoretical calculations with the experimental results to confirm the correctness of the obtained complexities in practice.

II. Preliminaries

In this section, formal verification using BDDs and the structure of the conditional sum adder are reviewed.

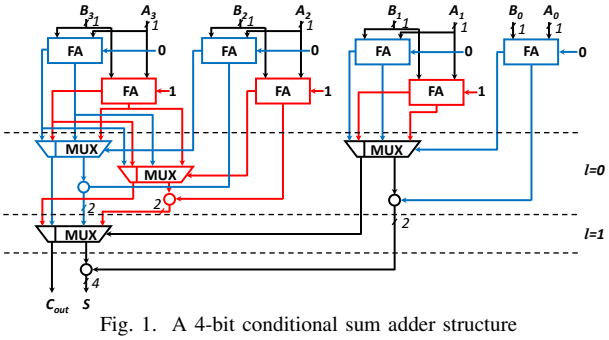


Fig. 1. A 4-bit conditional sum adder structure

BDD-based Verification: Formal verification of an adder architecture becomes possible by obtaining each output function in form of BDD. Since BDDs are canonical, the BDD of i^{th} output is always identical independent of the adder architecture. Thus, we can evaluate the BDDs to see whether they match the BDDs of an adder. Symbolic simulation is a powerful tool to get the output BDDs. First, the corresponding BDDs for each input signal are generated. Then, starting from primary inputs the BDD for the output of a gate (or a building block) is computed. This process continues until we reach the primary outputs and obtain their BDDs.

In order to compute the BDD of each gate (or building block) output during symbolic simulation, we use the ITE operator (If-Then-Else):

$$ITE(f, g, h) = (f \wedge g) \vee (\bar{f} \wedge h) \tag{1}$$

A recursive algorithm to compute the ITE operations has been proposed in [7]. This algorithm which is widely used in BDD packages (e.g., CUDD [8]) has a computational complexity of $O(|f| \cdot |g| \cdot |h|)$, where $|f|$, $|g|$ and $|h|$ denote the size of the BDDs in terms of the number of nodes.

Conditional Sum Adder: Fig. 1 shows the structure of a 4-bit conditional sum adder. In this architecture, two sets of outputs are generated for a given group of k operand bits. Each set contains k sum bits and one outgoing carry. For one of the sets, it is assumed that the eventual incoming carry will be zero, while for the other set it will be one. Once the incoming carry is known, we select the correct set of outputs using multiplexers.

The MUX blocks in Fig. 1 consist of two multiplexers: a multiplexer to select between two k-bit sums and a multiplexer to select between two 1-bit carries. These MUX blocks can be put in different levels based on their inputs. Fig. 2 shows the MUX blocks (boxes) of an 8-bit conditional sum adder in four levels. The number inside each box presents the size of the input sum bits, i.e., k.

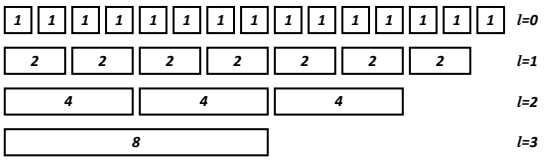


Fig. 2. MUX blocks in an 8-bit conditional sum adder

III. Computational Complexity

An n-bit conditional sum adder is divided into two main stages: 1) $2n-1$ *Full-Adders* (FAs) to generate initial sum and carry bits. 2) An array of MUX blocks (see Fig. 2) to select the sum and carry bits. We first calculate the computational complexity of the first stage; then, we focus on the second stage. Finally, we add up the two complexities to obtain the overall verification complexity.

A FA with a '0' input (blue FAs in Fig. 1) is a *Half-Adder* (HA); thus, the outputs can be obtained by two ITE operations:

$$\begin{aligned} s_i &= A_i \oplus B_i = \mathit{ITE}(A_i, \overline{B}_i, B_i), \\ c_i &= A_i \wedge B_i = \mathit{ITE}(A_i, B_i, 0) \end{aligned} \tag{2}$$

Similarly, the output BDDs of a FA with an '1' input (red FAs in Fig. 1) can be obtained as follows:

$$\begin{aligned} s'_i &= A_i \odot B_i = \mathit{ITE}(A_i, B_i, \overline{B}_i), \\ c'_i &= A_i \vee B_i = \mathit{ITE}(A_i, 1, B_i) \end{aligned} \tag{3}$$

The computational complexity of these two adders is the same and equals:

$$\begin{aligned} &\mathit{Complexity}(s_i) = \mathit{Complexity}(s'_i) = |A_i| \cdot |B_i| \cdot |\overline{B}_i| = 3 \cdot 3 \cdot 3 = 27 \\ &\mathit{Complexity}(c_i) = \mathit{Complexity}(c'_i) = |A_i| \cdot |B_i| = 3 \cdot 3 = 9 \end{aligned} \tag{4}$$

Since there are $2n-1$ FAs in the first stage of the adder, the overall complexity of the first stage is calculated as follows:

$$\mathit{complexity}_{[\mathit{stage1}]} = (2n-1) \cdot (27+9) = 72n - 36 \tag{5}$$

In order to calculate the computational complexity of the second stage, we first need to obtain the complexity of a single MUX block. A MUX block in level l (see Fig. 2) receives two inputs with $2^l + 1$ bits, i.e., $M[2^l : 0]$, and $N[2^l : 0]$. These inputs are the results of adding two 2^l-bit numbers. Then, the MUX selects between these two inputs based on the c signal, which is an output carry that resulted from adding two 2^l-bit numbers. Therefore, a MUX block can be translated into 2^l+1 ITE operations as follows:

$$o_0 = \mathit{ITE}(c, M_0, N_0), \quad \ldots, \quad o_{(2^l)} = \mathit{ITE}(c, M_{(2^l)}, N_{(2^l)}) \tag{6}$$

Thus, the overall complexity of a MUX block is calculated as follows:

$$\mathit{complexity}_{[\mathit{MUX}]} = |c| \cdot \sum_{i=0}^{2^l} |M_i| \cdot |N_i| \tag{7}$$

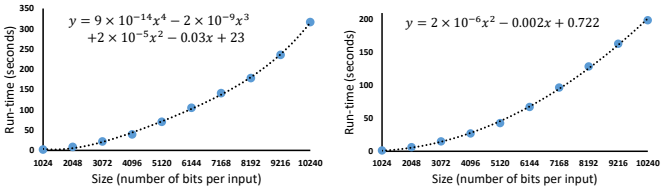
It has been proven in [9] that the BDD size of the i^{th} sum and carry bits are bounded by $3i+5$ and $3i+3$, respectively. Based on the facts that M_i and N_i are the sum bits, and c is the $(2^l)^{th}$ carry of an addition, we have:

$$\mathit{complexity}_{[\mathit{MUX}]} = (3 \cdot 2^l + 3) \cdot \sum_{i=0}^{2^l} (3i+5)^2 \tag{8}$$

The number of MUXs in each row and the number of rows in a conditional sum adder (see Fig. 2) are calculated as follows:

$$\begin{aligned} \mathit{number\_of\_MUXs\_in\_row} &= 2^{(\log_2 n - l + 1)} - 1, \\ \mathit{number\_of\_rows} &= \log_2 n + 1 \end{aligned} \tag{9}$$

where the equations are exact for all word length being a power of 2 (i.e., $n = 2^m$) [6].



(a) Conditional sum adder (b) Ripple carry adder

Fig. 3. Run-time graphs of the adders

Consequently, the overall computational complexity of the second stage is obtained:

$$\begin{aligned} \mathit{complexity}_{[\mathit{stage2}]} &= \sum_{l=0}^{\log_2 n} \left((2^{(\log_2 n - l + 1)} - 1)(3 \cdot 2^l + 3) \sum_{i=0}^{2^l} (3i+5)^2 \right) \\ &= \frac{384}{35} n^4 + \frac{720}{7} n^3 + 488n^2 + 399n \log_2 n - \frac{795}{7} n - 75 \log_2 n + \frac{1601}{35} \end{aligned} \tag{10}$$

Finally, the overall computational complexity of a conditional sum adder is calculated by adding up the complexity of the two stages in Eq. (5) and Eq. (10). Based on the calculations, we can observe that the order of the verification complexity is $O(n^4)$, i.e., it has quartic time complexity.

IV. Experimental Results

After calculating the verification complexity for a conditional sum adder, we check the correctness of the theoretical results in practice. Fig. 3(a) (Fig. 3(b)) presents the run-time of verifying conditional sum adders (ripple carry adders) with different sizes. Similar studies have shown that the verification of the ripple carry adder has quadratic time complexity. We fit a curve (dashed lines) to the points with an acceptable error and evaluate the curve function. We can fit a curve with the order of 4 to the verification run-times of conditional sum adders in Fig. 3(a). It confirms that the verification has quartic time complexity. On the other hand, a curve with the order of 2 can be fitted to the verification run-times of ripple carry adders in Fig. 3(b).

V. Conclusion

In this paper, we calculated the computational complexity of verifying a conditional sum adder using BDD-based method. Based on the calculations, we proved that verifying this adder is of polynomial time, i.e., quartic time complexity. We also confirmed the correctness of the complexity bounds obtained in our theoretical calculations by experimental results.

Acknowledgment: This work was supported by the German Research Foundation (DFG) within the Reinhart Koselleck Project *PolyVer* (DR 287/36-1).

References

[1] R. Drechsler and D. Sieling, "Binary decision diagrams in theory and practice," *Int. J. Softw. Tools Technol. Transf.*, vol. 3, no. 2, pp. 112–136, 2001.
[2] M. Keim, R. Drechsler, B. Becker, M. Martin, and P. Molitor, "Polynomial formal verification of multipliers," *Formal Meth. in Sys. Des.*, vol. 22, no. 1, pp. 39–58, 2003.
[3] A. Mahzoon, D. Große, and R. Drechsler, "PolyCleaner: clean your polynomials before backward rewriting to verify million-gate multipliers," in *ICCAD*, 2018, pp. 129:1–129:8.
[4] ——, "RevSCA: Using reverse engineering to bring light into backward rewriting for big and dirty multipliers," in *DAC*, 2019, pp. 185:1–185:6.
[5] R. Drechsler, "PolyAdd: Polynomial formal verification of adder circuits," in *DDECS*, 2021, pp. 99–104.
[6] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. dissertation, Swiss Federal Institute of Technology, 1997.
[7] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *DAC*, 1990, pp. 40–45.
[8] F. Somenzi, "CUDD: CU decision diagram package release 2.7.0," available at https://github.com/ivmai/cudd, 2018.
[9] I. Wegener, *Branching Programs and Binary Decision Diagrams*. SIAM, 2000.