Verification and Debugging of Gate-Level Arithmetic Circuits

Maciej Ciesielski, Michael Shliselberg, Jordan Kaplan
University of Massachusetts, Amherst, USA

Research Objectives

Identify and remove the residual expression from the extracted unique bit-level polynomial functions of arithmetic circuits.

Decrease runtime functional verification of gate-level, integer arithmetic circuits using an algebraic approach.

Progress and Results

Identified that the residual expression is created through transforming Boolean functions (extracted from the arithmetic circuits) to an algebraic function.

Currently exploring new methods to determine whether weights and other properties of the variables in forward rewriting are enough to be used for the verification and debugging process. This will increase the time efficiency of the current processes. For example, through weight propagation, the circuit progresses cut-by-cut only if certain variables contain certain weights. If it does not have the needed weight, there is an error within the inputs of the system that cannot propagate. Though the results of the tests conducted on small level linear circuits have shown promising results, further testing is needed on non-linear and larger circuits before a conclusion or more adaptations can be made.

Residual Expression

The residual expression is a zero-equivalent term created from the process of forward rewriting. It formulates when the zero term that is added on does not completely cancel out the previous terms from the previous cut.

A large amount of residual expressions are created from single OR gates because the multiplication of the inputs equal zero. Even still, through boolean logic this process would cancel it out immediately; when converted to an algebraic formula the terms are still zero. So far, the only way to determine if it is actually a residual expression and not an outlier term created by a bug is through backwards rewriting and to see if it truly does equal zero. The residual increases the amount of computation of the debugging process because it adds terms to the difference equation that needs to be evaluated. Therefore eliminating this residual is definitely beneficial.

Future Work

Continue to search for methods to identify and remove the residual expression as well as determine more efficient routes to reduce the time it takes to verify and debug a gate-level arithmetic circuit.

Conference and Journal Papers


Ghandali, Samaneh, Cunxi Yu, Duo Liu, Walter Brown, and Maciej Ciesielski. "Logic Debugging of Arithmetic Circuits."

Forward Rewriting

Forward rewriting is the process of going from the input signature to the output signature by adding terms that are equivalent to zero to the cut equation that will cancel terms of the previous cut and add terms of a new cut.

Backwards Rewriting

Backwards rewriting is the process where you begin at the output signature and proceed cut by cut to get to the input signature through the process of substituting each variable with the algebraic equation that represent the logic gate that the variable originated from.

Debugging

Debugging compares the forward rewriting and backwards rewriting to find the incorrect gate. The current approach finds the signature difference between backwards rewriting and forwards rewriting for each cut. If the signature difference does equate to zero and matches an expression on the table below, we are able to correct the detected bugs by substituting the correct gate in for the bug.

Special Thanks to:
Professor Ciesielski, Cunxi Yu, Walter Brown, Samaneh Ghandali, Duo Liu.