# System Verification Based on Modified Interval Analysis

I. Ugarte, P.Sanchez
Microelectronics Engineering Group. TEISA Department. ETSIIT
University of Cantabria
Avda. los Castros s/n. 39005 Santander. Cantabria. Spain
{ ugarte, sanchez }@teisa.unican.es

***Abstract* – Interval arithmetic was original developed to estimate rounding errors on floating-point computations but it is used in a wide variety of applications from constraint solvers and global optimizers to power and timing analysis of software processes. The objective of interval analysis is to determine the output ranges (or interval) of a computation set. The main problem of classical interval analysis is the overestimation of the output ranges and its dependency on the coding of the system behavior. In this paper, a modified interval analysis method is presented. The method reduces the interval overestimation and it is independent of the coding. This modified interval analysis is the kernel of a new verification technique that enables the verification of functional properties in system level descriptions and obtains functional test vectors.**

**Keywords – Interval Arithmetic, System Level Verification.**

## I. INTRODUCTION

As system complexity grows, designers describe it at higher abstraction levels and spend more effort on verification. In order to confront this growing complexity, it is essential to define verification methodologies that allow the validation of the design during the specification step, at system level. Simulation is the most widely used verification technique, but even if coverage metrics are used, it has several problems (test bench definition, completeness, etc). Another possibility is to use formal verification techniques. Some of these are based on transforming the system description into a functionally canonical form, such as BDDs, and deriving the solution from this structure. Other verification techniques test the satisfiability of properties (e.g. SAT-based verification [2, 3, 7]) or combine both approaches. However, these previously commented approaches generally suffer from exponential worst-case complexity, because they use Boolean representations of the system that increase the number of signals and operators during the verification processes.

This paper proposes a different type of solution that takes advantage of the structure of system-level description. At this level, the system is described with a set of statements that operate with integer or real data. Common arithmetic, relational and control statement are used.

The main objective of the proposed technique is to find an input range that verifies a set at properties and/or constraints. In order to do so, interval arithmetic is used [4]. Although this technique has been used previously in verification (e.g. timing and power analysis in software [6]), it has important drawbacks: overestimation of the output range and expression dependency. For example, if the behavior of the 'y' output is represented by the equation 'y=$x^2$-x' and the 'x' input is defined in the range *[-2,3]*, it is possible to derive that the output 'y' will be defined in the range *[-9,11]* using interval arithmetic [4]. However, if the output is defined by the equivalent equation 'y=x(x-1)', the derived range will be *[-9,6]*. Both approaches are overestimations of the correct range, *[-1/4,6]*.

In this paper, a modified interval analysis, which is independent of the expression and has low overestimation, is presented. This technique is used to determine the input space values that verify certain properties. These input values can be used as test benches and/or counter-examples.

## II. MODIFIED INTERVAL ANALYSIS

In our approach, we assume that the system is described at system level as a set of statements that operate with integer or real data. The addition, subtraction and multiplication operators are supported as well as 'if' control statements. Loop statements and other operators (Boolean operations, division, etc) are not currently supported.

At system level, the functionality can be represented for polynomial functions. Let $P(x_1, x_2, ..., x_N)$ be the polynomial that describes the behavior of a system 'S' with a input space of N inputs ($x_{I1} < x_1 < x_{S1}$, $x_{I2} < x_2 < x_{S2}$, …, $x_{IN} < x_N < x_{SN} \equiv X_I < X < X_S$). The inputs are positive. With negative inputs, a displacement is proposed. In order to obtain a maximum and minimum bound of this polynomial, it is split in two polynomials: the increasing (P+(X)) and the decreasing (P–(X)) polynomials. The increasing polynomial is composed of positive monomials (terms of the polynomial) and the decreasing polynomial by negative monomials. $P^+(X)$ and $P^-(X)$ can be non-linear functions, so they are bounded by two linear functions (hyperplanes): upper ($P_A(X)$) and lower ($P_B(X)$) bound hyperplanes. The addition of the bounds of the monotonous polynomials provides a hyperplanes that limit the polynomial $P(X)$. From these hyperplanes are obtained the range of values of the $P(X)$ in the input space.

## III. SYSTEM-LEVEL VERIFICATION BASED ON INTERVALS

Let 'S' be the system to verify. The objective is to verify that certain properties are true or false in an input value space (i.e. there is no overflow in operations, a special control path is executed, an assertion is asserted, etc).

The system description includes control and data statements as well as input assertions. The data statements affect interval values and the control statements and assertions are constraints that reduce the input value space. The current version of the algorithm only considers conditional control statements. In order to evaluate a property, all the conditional statements of the path between the inputs and the property have to be considered. These conditions (or requirements) are transformed into expressions of the form *P(X) > 0*, thus if the conditional expression takes a value greater than 0 the requirement will be satisfied. For example, the statement "*if (x + y < 50) then*" is transformed into "*if (50 - x - y > 0) then*" and the input values 'x = 10', 'y = 20' satisfy the requirement but the input values 'x = 30' and 'y = 25' do not.

Modified interval analysis is used to approach these constraints. Thus, the output range sign will determine the constraint satisfaction. The possible situations are: the maximum value of the approximated requirement is less than '0' on case (a). This means that there is no input that satisfies the requirement, therefore, the input value space will not be considered during the rest of the analysis. Case (c) occurs when the minimum value of the requirement approximation is greater than '0'. This means that all the inputs satisfy the requirement, thus this requirement is eliminated for this input space. Case (b) is the default case, in which the function could be positive and negative. In this case, the algorithm calculates a heuristic parameter (R) that measures the probability that the inputs generate a positive function value.

Concerning the property under verification, it has to be transformed into an expression of the form *P(X) > '0'*. For example, if we want to verify that the output 'z' is never greater than 255, the property will be defined as "z – 255 > 0". If case (a) occurs (there is not any value greater than 0), the property will be satisfied in the input value space. In case (b), an extreme point takes a value greater than '0'. If this point satisfies all the requirements, we will find a counter example that does not satisfy the property. In the other case, more searches are needed. Only in this case, a heuristic value (P) is calculated that indicates the probability that there is a point in which the property is greater than '0'.

The algorithm needs a cost function that guide the search of the counter-example. It is functions of the paramenters R and P, that estimate the existence of points that complies the requirements and the property.

The verification algorithm is a depth-first-search algorithm that explores the input value space. For a particular input value interval, the algorithm determines the hyperplanes that approximate the restrictions and properties. If it is not possible to provide a conclusion about the property satisfaction (case (c)), a new point inside the input value interval is selected. This process will generate several new intervals in which it is necessary to evaluate requirements and properties. In order to determine the interval in which the search will continue, a cost function is defined.

The input space with the maximum "cost" is chosen and it is split by the central point of the input space (the median value of the range in each one of the inputs). Again these zones are studied. The algorithm will finish if a counter-example point is found or if there is no any input space to study.

## IV. CONCLUSIONS

In this paper we present a technique for property verification in system level descriptions. The technique is based on the modified interval analysis method which is a better approximation than the classical interval analysis. This interval analysis technique is one of the main contributions of this paper.

For verification, a first-depth-search based algorithm is proposed. Only the basic arithmetic operators and the control sentence "if-then-else" are considered, but the potential for verifying a system can be appreciated.

## REFERENCES


[1] P. Sanchez, S. Dey, "Simulation-based system-level verification using polynomials", HLDVT'99, November 1999.

[2] F. Fallah, S. Devadas, K. Keutzer, "Functional Vector Generation for HDL Models Using Linear Programming and Boolean Satisfiability", IEEE Trans. Computer-Aided Design, vol. 20, pp. 994-1002, August 2001.

[3] Z. Zeng. P. Kalla, M. Ciesielski, "LPSAT: A Unified Approach to RTL Satisfiability", DATE-2001, pp. 398-402, March 2001.

[4] R.E. Moore. Interval analysis. Prentice-Hall, 1966.

[5] J. Smith, G. De Micheli, "Polynomial Methods for Component Matching and Verification", Proc. of the ICCAD'98 Conference. 1998.

[6] D. Ziegenbein, F. Wolf, K. Richter, M. Jersak, R. Ernst, "Interval-Based Analysis of Software Processes", *LCTES '2001*, pages 94-101, Snowbird, Utah, USA, June 2001.

[7] C. Huang, K. Cheng, "Assertion Checking by Combined Word-level ATPG and Modular Arithmetic Constraint-Solving Techniques", DAC 2000.

[8] T. Larrabee, "Efficient Generation of Test Patterns Using Boolean Difference", ITC 1989.

[9] A. Biere, A. Cimatti, E.M.Clarke, M. Fujita, Y. Zhu, "Symbolic Model Checking using SAT procedures instead of BDD", DAC 1999.