May 2017

Analog and Mixed Signal Verification using Satisfiability Solver on Discretized Models

Nikita Ramesh Wanjale
University of Nevada, Las Vegas

ANALOG AND MIXED SIGNAL VERIFICATION USING SATISFIABILITY SOLVER ON DISCRETIZED MODELS

By

Nikita Ramesh Wanjale

Bachelor of Technology, Instrumentation and Control Engineering
Vishwakarma Institute of Technology, Pune, India
2014

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Engineering - Electrical Engineering

Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2017

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 17, 2017

This thesis prepared by

Nikita Ramesh Wanjale

entitled

Analog and Mixed Signal Verification using Satisfiability Solver on Discretized Models

is approved in partial fulfillment of the requirements for the degree of

Master in Science - Electrical and Computer Engineering
Department of Electrical and Computer Engineering

Henry Selvaraj, Ph.D.
Examination Committee Chair

Biswajit Das, Ph.D.
Examination Committee Member

Grzegorz Chmaj, Ph.D.
Examination Committee Member

Laxmi Gewali, Ph.D.
Graduate College Faculty Representative

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

ABSTRACT

ANALOG AND MIXED-SIGNAL CIRCUIT VERIFICATION USING SATISFIABILITY SOLVER ON DISCRETIZED MODELS

By

Nikita Ramesh Wanjale

Dr. Henry Selvaraj, Examination Committee Chair

Professor, Department of Electrical and Computer Engineering

University of Nevada, Las Vegas

With increasing demand of performance constraints and the ever reducing size of the IC chips, analog and mixed-signal designs have become indispensable and increasingly complex in modern CMOS technologies. This has resulted in the rise of stochastic behavior in circuits, making it important to detect all the corner cases and verify the correct functionality of the design under all circumstances during the earlier stages of the design process. It can be achieved by functional or formal verification methods, which are still widely unexplored for Analog and Mixed-Signal (AMS) designs.

Design Verification is a process to validate the performance of the system in accordance with desired specifications. Functional verification relies on simulating different combinations of inputs for maximum state space coverage. With the exponential increase in the complexity of circuits, traditional functional verification techniques are getting more and more inadequate in terms of exhaustiveness of the solution. Formal verification attempts to provide a mathematical proof for the correctness of the design regardless of the circumstances. Thus, it is possible to get 100% coverage using formal verification. However, it requires advanced mathematics knowledge and thus is not feasible for all applications.

In this thesis, we present a technique for analog and mixed-signal verification targeting DC verification using Berkeley Short-channel Igfet Models (BSIM) for approximation. The verification problem is first defined using the state space equations for the given circuit and applying Satisfiability Modulo Theories (SMT) solver to determine a region that encloses complete DC equilibrium of the circuit. The technique is applied to an example circuit and the results are analyzed in turns of runtime effectiveness.

ACKNOWLEDGEMENT

I would like convey my gratitude and thank Dr. Henry Selvaraj, my research advisor for all the guidance and support during my graduate studies at University of Nevada, Las Vegas. His confidence in me encouraged me to move in the right direction during this research. He has always made sure of providing me with the right amount of resources for useful research with good hands on experience.

This research could not have been completed without the sincere help and guidance from Dr. Grzegorz Chmaj, Dr. Biswajit Das and Dr. Laxmi Gewali. I would like to extend my gratitude to them for serving on my committee and reviewing my thesis.

The research was a very challenging experience for me and would not have been completed without the help and support from my parents who have given me the motivation to learn and opportunities to grow. I will forever be in your debt.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

An analog and mixed-signal circuit (AMS) is an integrated circuit which encompasses digital and analog circuits on a single chip. Its design is crucial for embedded system designs and microprocessors. AMS circuits can be found as fully functional units or sub-functions of a larger assembly. Pertaining to the ubiquity of embedded systems and microprocessors, it is extremely crucial that the AMS circuits adhere to design specifications.

Verification is a process to validate the performance of the system in accordance with desired design specifications. At present, digital circuits have well-developed and explored verification tools and techniques available. Although the Computer Aided Design (CAD) tools for analog circuits have developed significantly in recent years, the verification process, for the most part, has remained limited to series of simulations to estimate noise and variation metrics. The process still relies solely on the experience of the design engineer for the exhaustiveness of simulations. Moreover, due to being labour intensive with little automation, circuit simulations take a significantly longer time than its digital counterparts [1].

The topic of this master's thesis is to implement Satisfiability (SAT) solver, which is a well researched digital verification technique, on discretized models of AMS circuits to achieve maximum state space coverage.

1.2 Motivation

During my summer internship as a field application engineering intern at Aldec, which is a design verification company, I had the opportunity to work on the Zynq FPGA board and study various digital verification techniques. I realized how crucial time-to-market can be in a commercial setting, making it extremely important to incorporate a faster verification methodology.

In the semester following the internship, I got familiar with an algorithm which uses FSMs and Petri-Nets to represent the AMS circuit [2]. This motivated me to see whether such discretized models can be used with a suitable verification techniques, thus allowing more automation in the AMS verification process.

In this work I focus on Satisfiability (SAT) solver and different algorithms that can be used for effective and timely verification. SAT is the problem of determining whether there exists an interpretation of variable to satisfy a given formula [3]. All the results and experiments were done using software simulations while the work largely depends on hardware aspects.

1.3 Main Goal

Main goal of this work is to explain how and why SAT solvers combined with suitable discretized models can increase the level of automation and reduce the runtime of AMS circuit verification. Multiple simulations have been performed to implement different algorithms and compare the results.

Understanding the advantages and limitations of these algorithms help decide whether it is commercially favourable compared to current pure simulation-based methodologies.

1.4 Scope of Work

This thesis contains 7 main chapters. In this chapter, the problem area is introduced and motivations and main goals are described. Then the problem background is discussed, where different verification methods are briefly discussed. Third chapter discusses different discretization methods and discrete state space modeling is introduced. Fourth chapter talks in detail about the SAT solver and its algorithms. Fifth and Sixth chapter describe DC and transient verification techniques respectively. The last chapter summarizes all results and observations and concludes the work.

CHAPTER 2

BACKGROUND

In recent years, the growth in compact electronic devices has been tremendous. Moore's law states that the number of transistors on integrated circuit chips has doubled every year since its invention. With a steady compression of circuits, the complexity has increased, making the validation process absolutely vital. Automation tools have benefitted the EDA industry for circuit design, validation and testing for years. However, continuous nature of AMS circuits make them unsuitable for these tools. Customized methods need to be used for such circuits. Let us discuss the contemporary methods for analog verification.

2.1 Functional Verification

Typically, an RTL code is written to interpret the functional description of the circuit. Functional Verification checks an RTL design from a functional perspective. It checks the correspondence between the RTL description and design specification. Verilog-AMS or VHDL-AMS can then be used to simulate different input and state variable combinations.

In [4], S. Steinhorst and L. Hedrich developed a stimuli generation algorithm to simulate different conditions on a discretized state space model. A graph structure is generated as a discretized state space model where each state is represented by a vertex. All the vertices eventually converge to the DC operating points on the graph. The stimuli generation algorithm traverses on the graph to reach all the vertices in optimal number of transitions.

Pertaining to the high number of potential design states in a large AMS design, functional verification is often unable to provide exhaustiveness necessary for such designs.

2.2 Formal Verification

Formal verification techniques take into consideration, all the possible input and state variable conditions and generate a state space for the system. Since it inherently considers the entire range of values for input and state variables, proofs given by the formal verification techniques hold true for the complete state space [5]. Formal verification can be broadly classified into two types: *Theorem Proving Method* and *State Space Exploration Method* [6].

Theorem proving method develops a mathematical model and proves its correctness according to required design specifications. Due to its completeness, automated theorem proving (ATP) computer programs are being explored. [7] uses *MetiTarski,* an ATP for inequalities on real-valued elementary functions to verify properties of AMS circuits by first obtaining a closed form solution to the discrete circuit model and checks properties concerning changes in gain and oscillations.

Dang et al. (2004) demonstrated a formal verification methodology to deal with the dynamic behavior of AMS circuits is described by a DAE system:

$$F(x(t), \dot{x}(t), u(t), p) = 0 \tag{1}$$

where $x \in R_n$ denotes the state variables (internal voltages, currents, and outputs), $\dot{x}$ denotes their time derivatives, $p \in P \subset R_m$ is the parameter vector, and $u : R+ \rightarrow U$ is the input signal. Due to the uncertainty of input, external disturbances and noise can be modeled [8]. To verify the time domain properties of the circuit, the set of solutions of the above equation is characterized for all possible inputs and all parameter values p.

State space exploration method relies on the state space representation of the circuit and validates it for all the inputs over the entire range on states. This method allows a crucial advantage over other analog verification methods - ability to automate. Since the number of states of a system depends on the number of storage element, it is possible, through loop equations, to automate the state space generation. However, it faces an issue of state space explosion [9].

State space exploration method can further be divided in: *Equivalence* and *model checking*.

Equivalence checking method analyzes the functional equivalency of two models of the same circuit. The purpose is to replace more complex AMS circuit with a simplified model in a system, provided that the two models are validated to be equivalent. Equivalence checking can also be done between models with different levels of abstraction. For example, a netlist can be checked again a behavioral model.

In [10], a linear analog circuit is represented by transfer functions in the s-domain and demonstrated an equivalence checking algorithm, while taking parameter variation into account. Although this work is limited to only linear circuits, it has been extended to accommodate nonlinear circuits in [11].

Model checking method is well suited for testing dynamic properties of the AMS system. A circuit model is used to check if a certain state is reachable in the complete state space of the system. The property to be checked and the state space of the model are both mathematically formulated and state space exploration is achieved by reachability analysis [12][13].

In [14], reachability analysis is performed on charge pump phase-locked loops (PLLs) for model verification. The main problem of bounded uncertain parameters is resolved by over-approximating the effects of the switching conditions with uncertain parameters in linear continuous models.

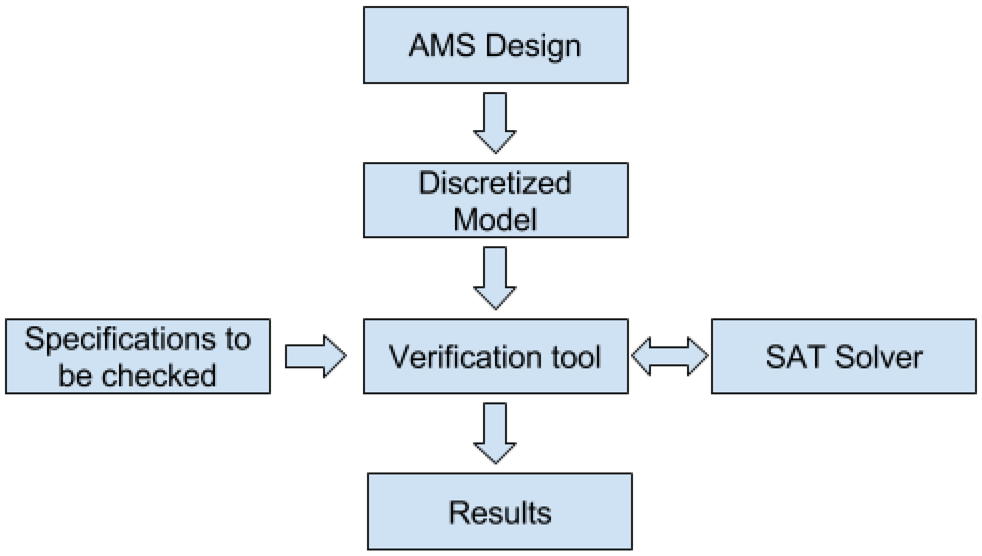


Fig. 1. General Block Diagram of AMS Verification using SAT Solver

2.3 Overview

In this work, two approaches with a similar flow are studied for AMS circuit verification. In the first approach, stable DC operating points are determined and the transistor-level circuit behavior is inspected. To overcome complex nonlinear equations of certain modern transistors, this approach has an intermediate stage to apply SAT solver to simple bound models before computing final solution, which includes accurate BSIM model information. The approach is then tested on an example circuit to demonstrate the results.

Second approach iteratively calculates the next reachable space starting from the initial range of the state space. For a large AMS circuit, it is required to consider a conservative bounded behavioral model considering parameter variations and modeling errors. An SAT solver is applied to the model to check conservative dynamic properties. The functionality is then demonstrated using an example circuit.

CHAPTER 3

DISCRETIZED MODEL GENERATION

In order to achieve successful verification, discretized model generation method can be seen as a bottleneck problem. Modeling the correct behavior is extremely necessary for AMS verification. A lot of work has been done to ensure accuracy of the models and gotten impressive results [15].

In [16], various Piecewise Linearization (PWL) approaches, like simplicial, piecewise and constant linearization, are compared against each other in the context of DC equilibrium points and transient analyses. This work, although effective, does not take into account the noise and process variations in a practical setting. [1] uses stochastic differential equations to model these disturbances for modeling and verification of AMS design.

The discrete model generation technique used in this work is largely motivated by [17]. Discrete state space graphical representation has been proven to be an accurate way to capture the behavior of AMS circuits. Figure 2 shows the sequential process of obtaining the discrete state space for the AMS design.

3.1 Obtaining DAE System

Equation (1) depicts description of a nonlinear analog system in terms of a differential algebraic equation (DAE). Depending on the number of system variables, majority of AMS circuits can be represented by DAEs. Nonlinearities in a system can be arbitrary in theory, though they are usually bounded in practice. DAEs of a system can be obtained from model behavior or netlist.

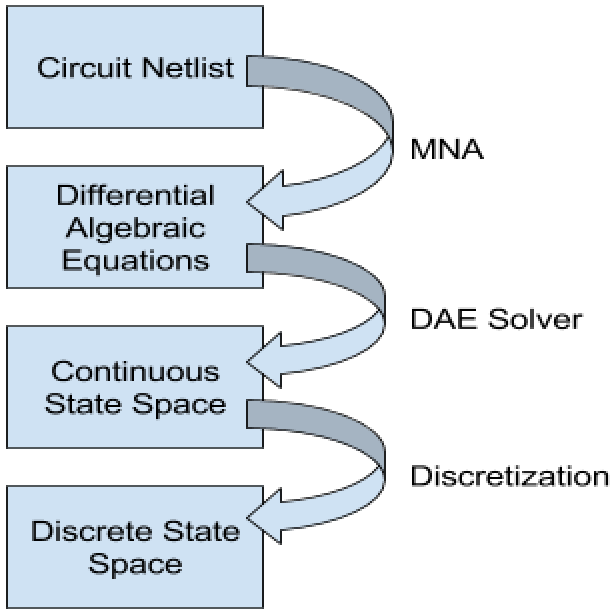
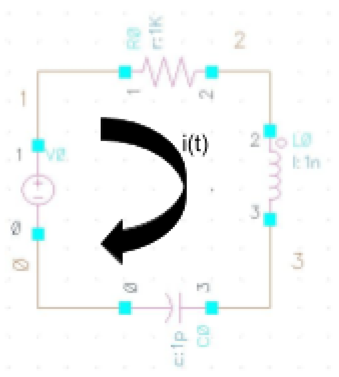


Fig. 2. Discrete State Space Representation of AMS Circuit

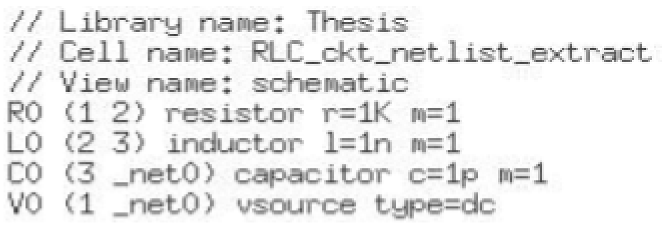
First, node reference, which is usually ground, in assumed to be node 0 and rest of the nodes are numbered consecutively. Then their voltage/current characteristics and Kirchoff's laws are used to develop a system of equal numbers of unknowns and equations. Let us observe these steps through an example.

Fig. 3. (a) shows the schematic of a series RLC circuit. The netlist of the circuit is obtained from Analog Design Environment (ADE) of Cadence Virtuoso and is used to develop a DAE system with Modified Nodal Analysis (MNA) as described above.

Once the DAE system is obtained, it need to be modified prior to further processing. The DAE system index is directly proportional to the numerical complexity of the system. The higher the index, the more difficult it is to solve the DAE system. This work uses topological index reduction technique to bring down the number of indices to one. The importance of index reduction and details of the technique are described in [16].

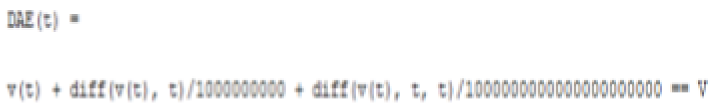


(a) Circuit schematic



```
// Library name: Thesis
// Cell name: RLC_ckt_netlist_extract
// View name: schematic
R0 (1 2) resistor r=1K m=1
L0 (2 3) inductor l=1n m=1
C0 (3 _net0) capacitor c=1p m=1
V0 (1 _net0) vsource type=dc
```

(b) Netlist extraction from ADE



```
DAE(t) =

v(t) + diff(v(t), t)/1000000000 + diff(v(t), t, t)/1000000000000000000000 == V
```

(c) Equivalent DAE system

Fig. 3. Obtaining DAEs from Circuit Netlist

Apart from index minimization, it is also important to remove singularity in MNA equations to avoid mathematical errors while solving the DAE system, as matrix inverse calculations are often involved. This is done with an additional step to eliminate excess variables and systematically obtaining nonsingular DAE system as described in [17].

Once the conditioned DAE system is obtained, a state space representation can be achieved by solving the equations.

3.2 Discrete State Space Modeling

Since DAE system solution usually involves numerical integrations, it is well known hurdle for SPICE and similar simulators. Various techniques have been proposed to solve a DAE system numerically [18]. Using discrete time steps, the system is solved at t_{n+1} using backward Euler formula -

$$F(t_{n+1},\, y_{n+1},\, \tfrac{y_{n+1}-y_n}{h_{n+1}}) \;=\; 0\,; \qquad \text{where } h_{n+1} = t_{n+1} - t_n \tag{2}$$

For this work, Differential Algebraic system solver (DASSL) code has been used to solve AMS DAE systems. DASSL approximates the derivative using Backward Differential Formula (BDF). Step size is chosen at every step according to the local measurement of integration error. If the integration errors exceeds a threshold, the step size is reduced.

Having variable step size increases overhead, as DASSL uses a stepsize variable to implement BDF formulas and advance the solution for each successor step. To simplify this issue, the continuous state space can be divided in sub-parts having homogenous tendencies. The step size within each subpart is kept constant. This way, any point in a state space can be traced to its successor location through local stepsize control [19].

CHAPTER 4

SATISFIABILITY SOLVER ALGORITHM

Satisfiability (SAT) solver checks for an interpretation that satisfies a given property. If there exists an interpretation, then it is considered a satisfiable assignment, else an unsatisfiable assignment. For example for an SR flip-flop, $\phi = (S \text{ and } R)$ is an unsatisfiable condition because both S and R cannot be assigned a HIGH value simultaneously. However, $\phi = (S \text{ and } \neg R)$ is satisfiable as {S = 1, R = 0} is a valid interpretation.

The exhaustiveness of underlying search algorithm of SAT solvers has proven to be their greatest asset for digital hardware verification. Various SAT solver algorithms used for digital verification have been discussed in [20]. This work is largely motivated by [21], in which a linear SAT solver, *fSPICE*, is used for analog verification. A nonlinear SAT solver, *iSAT* has been used, since it can handle both linear and nonlinear constraints. It conveniently represents nonlinearity with the use of nonlinear functions instead of approximated linearization methods.

4.1 DPLL Algorithm

The Davis-Putnam-Logemann-Loveland (DPLL) algorithm is a foundation of majority of contemporary SAT solvers. The ultimate aim of the algorithm is to either provide a satisfiable interpretation or prove that the assignment is unsatisfiable [22]. If satisfiable, the it returns assignment ρ to a given problem ϕ , which is specified in conjunctive normal form (CNF). CNF in boolean domain is similar to product of sums in circuits' perspective. Figure 4 outlines the DPLL algorithm.

Fig. 4. DPLL Algorithm for Boolean Domain

The input to the algorithm is a boolean expression ϕ. The first step is to preprocess the input boolean formula and check if there is a non-existent clause or a variable. After the preprocess step, the algorithm checks if there exists any unassigned clause. If no such clause exists, then a satisfiable assignment has been found and the output ρ is returned. Otherwise, the algorithm will select an unassigned variable and assign it with a truth value, either TRUE or FALSE. Decision step is followed by deduction step. In the deduction step, the algorithm locates each unit clause and makes an assignment to let the unit clause to be true. If there is no unit clause left, then the algorithm will go back to the decision step. After the implied assignment is made for the unit clause, the formula will be evaluated. If the result is UNSAT, a source of conflict ϕ_c is combined with the function ϕ to form a comprehensive function. If the resulting function comprises of the entire state space, it is proven that it is unsatisfiable and UNSAT is returned. Otherwise, the DPLL algorithm will backtrack and undo all the decisions stemmed from the conflict source and restart the process.

4.2 iSAT Solver Algorithm

Interval Constraint Propagation (ICP) locates the region in state space containing all the solutions returned from the algorithm, based on interval arithmetic. For example, for problem constraint $a + b = c$ where $a \in [-3, 2]$, $b \in [-2, 1]$ and $c \in [-10, 10]$, with interval arithmetic, region of c can be contracted from [-10, 10] to [-5, 3].

ICP can be incorporated with the DPLL algorithm to contract the real domain. The skeleton of the algorithm for iSAT solver is similar to DPLL as shown in figure 5.

The input to the iSAT solver algorithm is ϕ, an expression in boolean or real domain and a pre-defined threshold ε, for the interval length control with ICP. If the interval length of a clause is greater than ε, it is split into two equal parts. The first interval length is then assumed to be the interval length for that clause before testing satisfiability. If the algorithm returns UNSAT for both the interval lengths, a conflict set is determined and added to the existing formula ϕ.

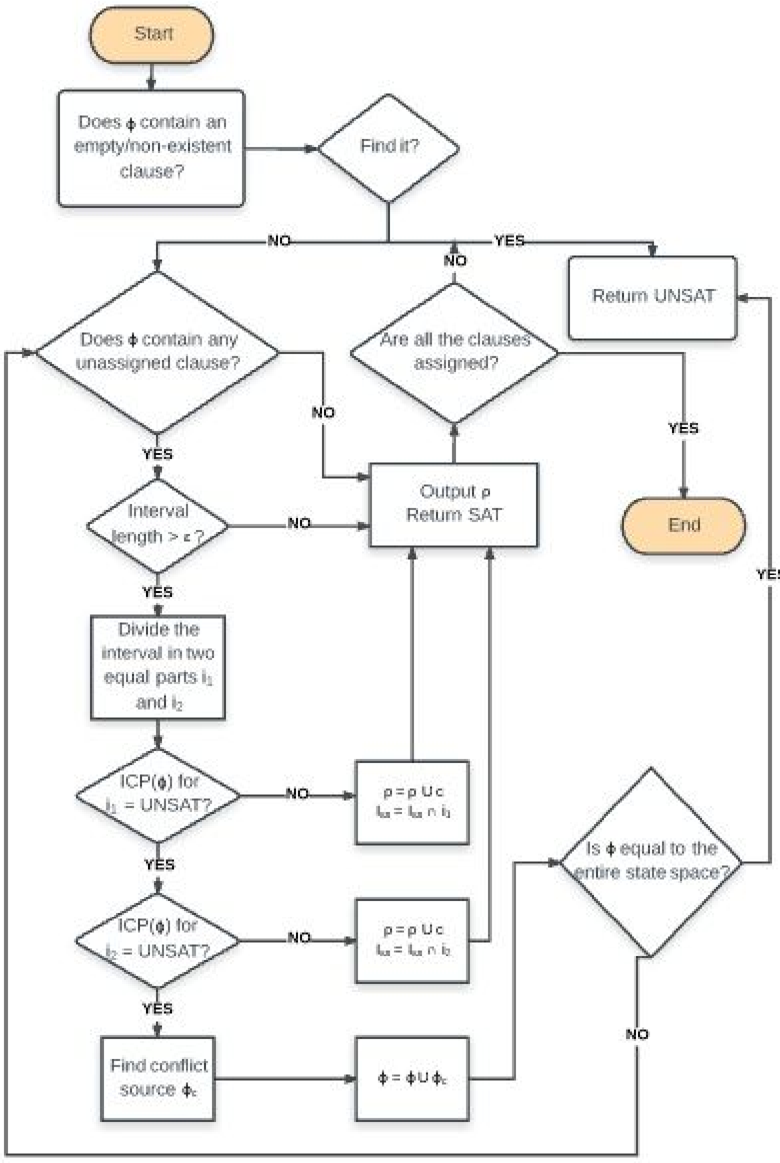


Fig. 5. iSAT Solver Algorithm for Real and Boolean Domain

If the resulting ϕ contains the entire state space, it implies that no solutions exists for the original ϕ. If not, the backtracking process from the DPLL algorithm is executed. Unlike DPLL algorithm, the solution given by iSAT solver is a space range containing the point solution, instead of an exact point solution. If the threshold ε is small enough, the solution can be approximated to the exact point solution.

A notable advantage of iSAT solver is that it guarantees unsatisfiability. If the algorithm returns UNSAT, it is proven that no solution to the given ϕ exists. This feature is used in this work as explained in later chapters.

CHAPTER 5

APPROACH I: DC VERIFICATION

For the DC analysis of any circuit, we need to identify all the stable DC operating points in the circuit. A DC operating point of a circuit is a set of states at which the system converges eventually for constant input and remains in that state. Transient verification, which is discussed in chapter 6, takes the DC operating point as an initial state of the circuit to linearize the nonlinear circuit behavior. It is possible to have no DC operating points, such as in a ring oscillator, where the state keeps oscillating with time. On the other hand, a circuit can have multiple DC operating points, as in a Schmitt trigger.

As discussed in chapter 3, a circuit can be represented by a DAE system shown in equation (3).

$$\mathrm{F}(x(t), \dot{x}(t), u(t)) = 0 \tag{3}$$

$$||x(0) - x(\infty)|| < \varepsilon \qquad \text{where } \varepsilon > 0 \tag{4}$$

$$\lim_{t \to +\infty} x(t) = x(\infty) \tag{5}$$

For constant input, i.e. *u(t) = u(0)*, $x(\infty)$ is called the equilibrium point of a circuit. A stable equilibrium point is the DC operating point, which may not be the case always. Equation (5) formalizes the condition for stability of an equilibrium point [23].

As seen above, DC analysis involves two steps. First step is to identify all the equilibrium points and the second step is to find the DC operating points, i.e. stable equilibrium points.

5.1 Device Approximation

SPICE uses device models like BSIM3/4 and PTM, expressed in C or Fortran, which are often complex. Such models cannot be handled by the proposed iSAT Solver, and need to be abstracted to a simpler form. This can be achieved by curve fitting on BSIM models from [24]. A bounded device model is formed in the following form.

$$I_{ds} \geq Lowerbound(V_{gs}, V_{ds}, V_{sb}, P) \tag{6}$$

$$I_{ds} \leq Upperbound(V_{gs}, V_{ds}, V_{sb}, P) \tag{7}$$

The device model shown by equations (6) and (7) is simpler compared to the complex BSIM models in SPICE, but it is guaranteed to bound the accurate I-V characteristics of the devices. The parameter term, P, from the equations can be eliminated for modeling fixed device parameters, or can be used to model effects of parameter variation. For example, the effect of gate width on the transistor performance can be modeled by plugging it in the bounded model formulae above.

In this work, we only model I-V characteristics of devices, but it can be extended to include the effects of process variations as well.

5.2 Problem Generation and Solution

Besides having an accurate model for a circuit or a system, it is important to develop an appropriate problem formula ϕ to express the desired behavior precisely. The bounded models from equations (6) and (7), along with the Kirchoff's laws (KVL and KCL), can be used as constraint to generate the expression for ϕ. iSAT Solver is used to find a solution to ϕ. Since ϕ

represents approximation of the circuit, the solution to ϕ itself is not the solution to the circuit, but it is guaranteed to bound the solution to the circuit. This can be explained using an example shown in figure 6. The shaded region is the solution to ϕ and the point of intersection is the DC operating point or the solution to the circuit.

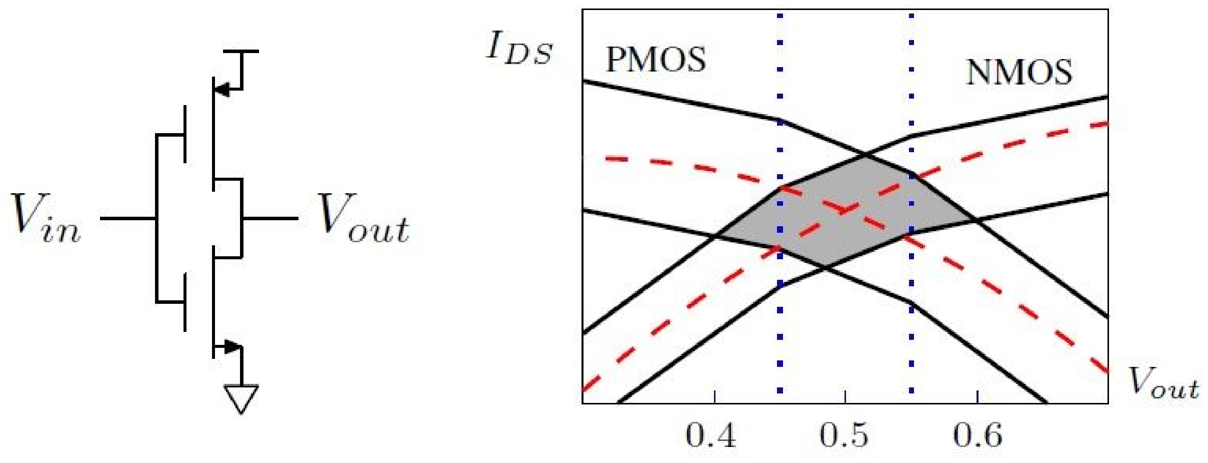


Fig. 6. DC Problem Formulation with Bounded Models

Since any point within the solution region of ϕ will eventually converge to the DC operating point, i.e. the circuit solution, iSAT outputs a point within the solution region. In figure 6, any point within the shaded region can be given as a solution by iSAT, due to its eventual convergence.

Once a point is found, the subdivision or a *box* of the continuous state space as discussed in chapter 3, containing the point is eliminated from the state space region and the iSAT algorithm is re-run to find additional DC operating points, if any.

5.3 DC Verification Algorithm

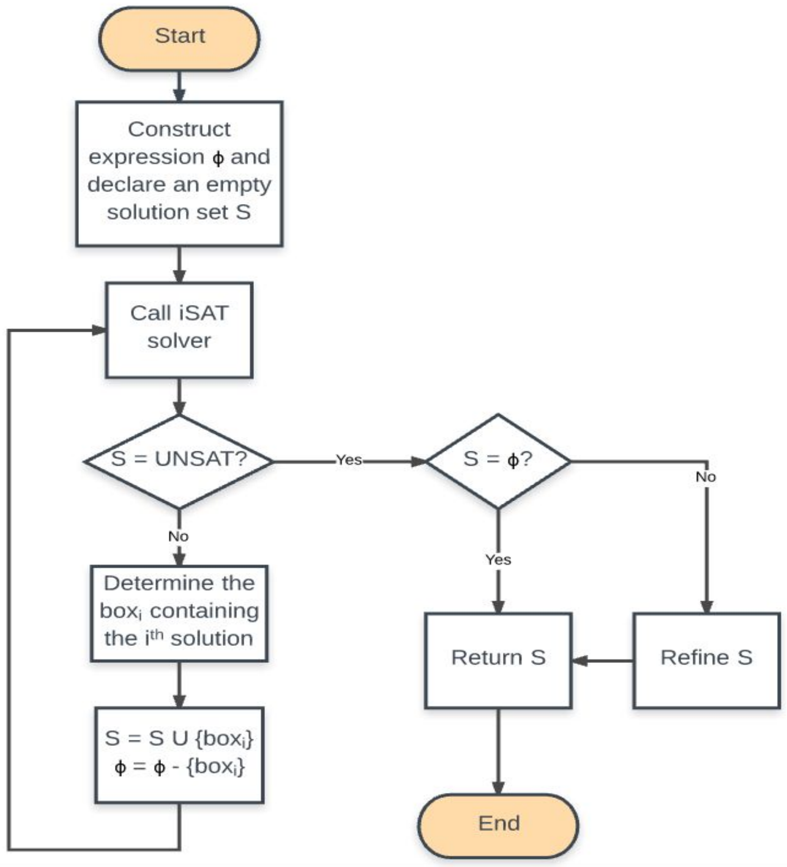

Start
Construct expression ϕ and declare an empty solution set S
Call iSAT solver
S = UNSAT?
Yes
No
S = ϕ?
Yes
No
Determine the box_i containing the i^{th} solution
$S = S \cup \{box_i\}$
$\phi = \phi - \{box_i\}$
Return S
Refine S
End

Fig. 7. DC Verification Algorithm

The input to the DC verification algorithm shown in figure 7 is a set of boxes obtained from the state space representation in chapter 3. The set of boxes are described as a set of constraints.

In the first step, problem formula ϕ is constructed and fed to the iSAT solver. As described earlier, iSAT solver will return a point in the state space within the solution for ϕ. With the help of pre-defined constraints of the boxes, the box corresponding to the output of the iSAT solver is determined. A constraint is added to the formula ϕ, to curtail the box region. The modified ϕ

is fed again to the iSAT solver to find additional solutions, until the solver returns UNSAT. Since the iSAT output guarantees a lack of solution for UNSAT, it is intuitive that the region occupied by the set of boxes, containing solutions to ϕ, is the superset of the solution interval for the circuit. In this work, this region is referred to as the *candidate region*.

The run-time and the resolution of the candidate region for the circuit are primarily dependant on the size of the boxes. With larger boxes, the candidate region can be found out with much lesser iterations of the iSAT, compared to smaller boxes. However, the candidate region for larger boxes is significantly larger compared to the smaller boxes.

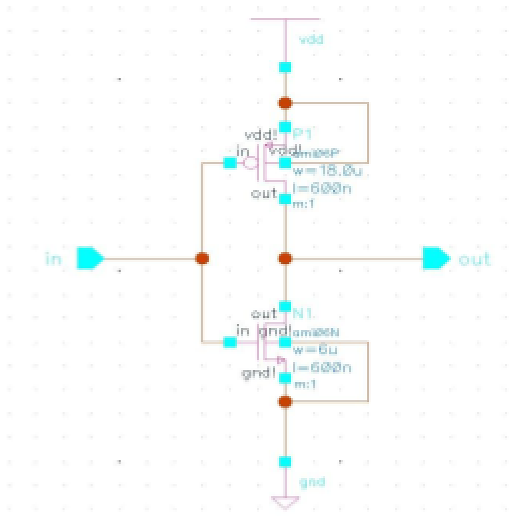
An optimal size needs to be chosen depending on the type of application and its requirements. For instance, for a time-intensive application, larger boxes can be chosen to minimize the run-time, while compromising the resolution.

Another way to reduce the size of candidate region is described in [21]. This work combines larger boxes with the size reduction technique from [21] to achieve faster run-time with optimal resolution. Since both the methods guarantee the inclusion of all possible solutions, output region *S* is guaranteed to bound all DC solutions.

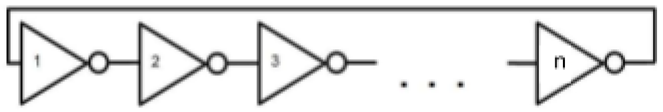
Next section discusses the results of this algorithm using examples of Schmitt trigger and ring oscillator.

5.4 Results: DC Verification of Ring Oscillators

Ring oscillator is a series of inverters, where the output of the last inverter is fed to the input of the first. It is important to check the stability of equilibrium points of the ring oscillator in order to avoid it being locked in steady state. Figure 8 shows the schematic of an n-stage ring oscillator. Since the carrier mobility of PMOS transistors is 1/3rd of that of NMOs transistors, if the width of the PMOS transistor is thrice the width of the NMOS transistor, there is an equilibrium point at $V_{dd}/2$. Naturally, since the ring oscillator for odd number of stages keeps oscillating, it will not have another equilibrium point.



(a) Inverter Schematic



(b) n-stage Ring Oscillator

Fig. 8. Ring Oscillator

On the other hand, for a ring oscillator with even number of stages, there will intuitively be two additional equilibrium points, i.e. at 0 and V_{dd}.

For the implementation of the DC verification algorithm in figure 7, the transistors are modeled using upper/lower bounds and curve fitting of BSIM3 model simulation results. The results for 3-stage and 4-stage oscillators with V_{dd} = 5V are shown in equations 8 and 9 respectively.

$$S = \{V_1 \varepsilon [2.499, 2.5], V_2 \varepsilon [2.499, 2.5], V_3 \varepsilon [2.499, 2.5]\} \tag{8}$$

$$\begin{aligned} S = \{&V_1 \varepsilon [2.499, 2.5], V_2 \varepsilon [2.499, 2.5], V_3 \varepsilon [2.499, 2.5]; \\ &V_1 \varepsilon [0, 0.001], V_2 \varepsilon [0, 0.001], V_3 \varepsilon [0, 0.001]; \\ &V_1 \varepsilon [4.999, 5], V_2 \varepsilon [4.999, 5], V_3 \varepsilon [4.999, 5]\} \end{aligned} \tag{9}$$

Run-time (In seconds)			
No. of stages	**Method in [26]**	**The proposed method**	**Method in [21]**
11	36.73	41.56	51.85
12	110.76	86.45	129.09
13	64.89	134.04	368.93
14	86.85	251.56	1226
15	134.74	456.87	4072
16	118.58	535.88	17223

Table 1. Comparison of Run-time for Ring Oscillators

The run-time performance of the proposed algorithm is also compared against the algorithm in [21] and [26] by varying the number of stages of the oscillator. It is clearly evident in table 1 that using this method reduces the run-time significantly compared to [21]. The run-time for [26] which uses fixed grid boxes, is less than the proposed method due to simpler algorithm. However, designating a fixed size to each box gives rise to over-approximation. The proposed method minimizes the over-approximation at the cost of runtime and finds an optimal solution.

As discussed in chapter 4, DPLL algorithm incorporates ICP to minimize the solution interval. As the ICP interval threshold (ε) is reduced, the number of iterations and calls to the SAT solver increase exponentially as demonstrated in the algorithm in figure 5. The run-time performance of the proposed DC verification algorithm is shown in table 2 for different values of ε .

Run-time (in Seconds)				
No. of stages	**ε = 0.1**	**ε = 0.01**	**ε = 0.001**	**ε = 0.0001**
11	17.88	24.34	41.56	84.58
12	32.11	55.96	86.45	188.31
13	64.27	85.49	134.04	225.00
14	122.17	178.54	251.56	386.47
15	226.86	311.76	456.87	672.18
16	290.04	381.63	535.88	882.33

Table 2. Comparison of Run-time for a range of values of ICP interval threshold (ε)

It is thus clear that the selection of ε largely depends on the type of application and its needs. For instance, smaller value of ε will result in faster run-time performance for a time-sensitive application.

CHAPTER 6

APPROACH II: TRANSIENT VERIFICATION

Upon finding the DC operating points of the circuit, it is necessary to analyse the transient response of time dependent voltage/ current signal to a given input. Starting from the initial condition, which can be either the DC operating point discussed earlier or a user-defined condition, the DAE solver described by equation (2) in chapter 3 can be used to calculate the next reachable state of the circuit. In the end, the trajectory approximation can be obtained from the given initial condition.

During transient verification, usually a range of initial conditions is taken to observe the dynamic response of the circuit. Naturally, an efficient scheme has to be designed to get samples of the initial range of interest. The circuit is then simulated for these samples of the initial range. In this work, two methods of transient verification are proposed with the combination of simulation and SAT solver.

6.1 Reachability Analysis

Implementing state space exploration method described in chapter 2 for transient analysis, the extend of state space coverage is an important consideration. In [21], circuit equations (2) are combined together and solved at once with a strategy known as *unroll* strategy. Due to the long simulation times usually necessary for the observation of dynamic behavior of the circuit, scalability becomes a major issue for the unroll strategy. When thousands of points in the state space are involved, it is impractical to use the strategy.

Another choice, the *reachability analysis*, is proposed in [25]. In this approach, the circuit equations (2) are solved one by one as we will see later in the chapter.

In this work, the homogenous boxes obtained in chapter 3 are used to represent the reachable space in the reachability analysis. Figure 9 depicts the algorithm for box-merging.

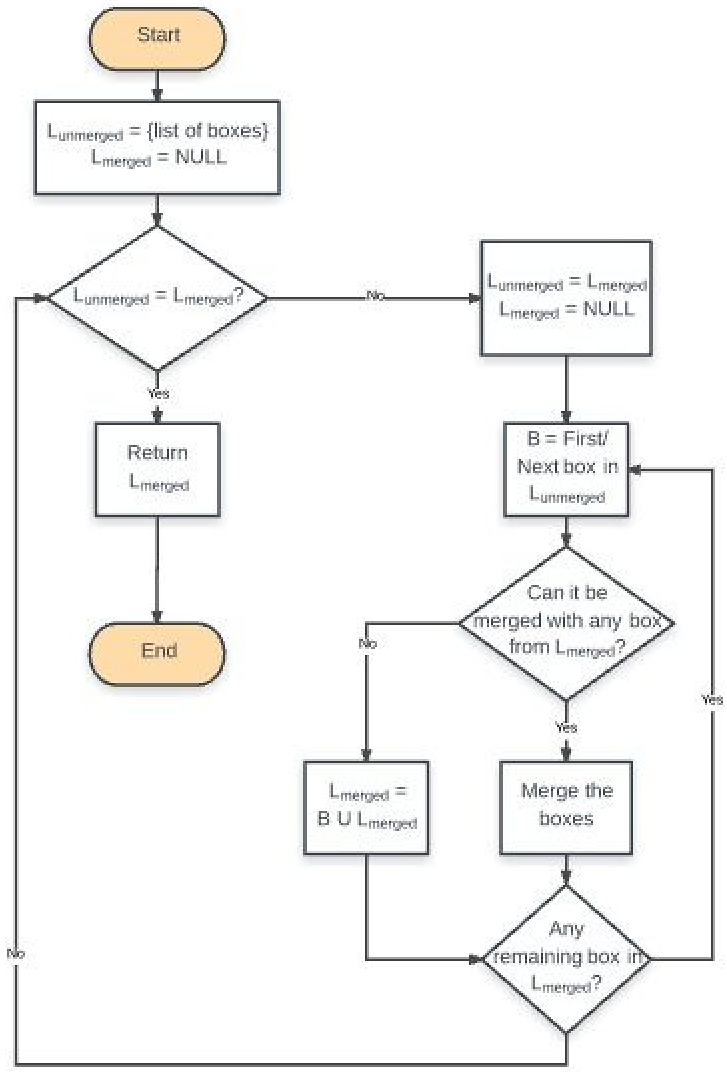

Start
$L_{unmerged}$ = {list of boxes}
L_{merged} = NULL
$L_{unmerged} = L_{merged}$?
No
$L_{unmerged} = L_{merged}$
L_{merged} = NULL
Yes
Return L_{merged}
End
B = First/ Next box in $L_{unmerged}$
Can it be merged with any box from L_{merged}?
No
Yes
L_{merged} = B U L_{merged}
Merge the boxes
Yes
Any remaining box in L_{merged}?
No

Fig. 9. Box Merging for Reachability Analysis

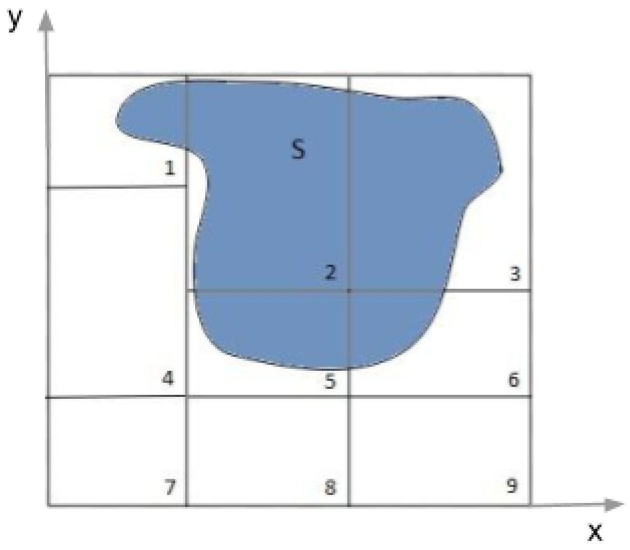


Fig. 10. Box-grid Representation

Let us take an example to explain the algorithm and its purpose. Figure 10 shows an example of a box-grid obtained from the state space subdivision process discussed in chapter 3. The colored enclosed space S is the current reachable space. To represent S in terms of S, the union on 5 boxes - 1, 2, 3, 5, 6 can be used as shown in equation (8) below.

$$(x,\ y)\ \varepsilon\ B_1 \vee B_2 \vee B_3 \vee B_4 \vee B_5 \qquad (10)$$

As we can see from the example, using a geometrical representation method introduces over-approximation. There are spaces in the representation of S which do not belong in S. While the variable sized box representation has lesser over-approximation compared to fixed size boxes in [26], the number of times the SAT solver is called in the algorithm is higher since it has more number of boxes. Therefore, reducing the number of boxes by merging provides a solution for better efficiency.

Let us now discuss how the box-merging algorithm works with the help of the example in figure 10. The order in which $L_{unmerged}$ is loaded with boxes can affect the result of the merge.

Suppose the list is generated by traversing through the state space horizontally, from top to bottom, B_1 will be the head and B_6 will be the tail and the $L_{unmerged}$ would be {B_1, B_2, B_3, B_5, B_6}. Initially, L_{merged} is NULL. The algorithm checks if the first box in $L_{unmerged}$ can be merged with any boxed from L_{merged}. Since, L_{merged} is empty, B_1 is added to L_{merged}. The next box checked in the algorithm is B_2. Since it can be merged with B_1, L_{merged} will be modified with a larger box B_{12} in place of B_1. After going through all the boxes in $L_{unmerged}$ once, L_{merged} is loaded as {B_{123}, B_{56}}. The algorithm will then load the L_{merged} as $L_{unmerged}$ and will empty L_{merged} for another cycle of execution. Every time the algorithm has gone through all the elements of $L_{unmerged}$, it will check if the unmerged list is same as the merged list, as that would indicate the saturation of solution. The execution will then be stopped and L_{merged} will be return for further operations.

On the other hand, if the list is generated by traversing through the state space vertically, from top to bottom, The head and tail would still be B_1 and B_6 respectively. However, $L_{unmerged}$ in this case would be {B_1, B_2, B_5, B_3, B_6} and at the end of the execution, L_{merged} will be {B_1, B_{2356}}. Figure 11 shows the contrast between the result of these two box-merging variations for the given example.

Once the merged boxes are obtained, reachability analysis, introduced earlier in this chapter, is used by iteratively calculating the next reachable space from the current reachable space by the equation (2). Equation (2) can be seen as a function of the current state, x_k and the next state x_{k+1}, i.e. $F(x_k, x_{k+1}) = 0$.

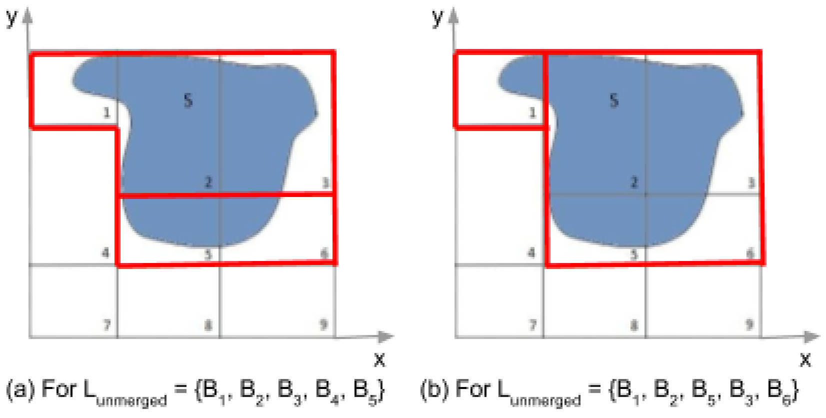

y
x
1
2
3
4
5
6
7
8
9
S
(a) For $L_{unmerged} = \{B_1, B_2, B_3, B_4, B_5\}$
(b) For $L_{unmerged} = \{B_1, B_2, B_5, B_3, B_6\}$

Fig. 11. Box-merging Variations

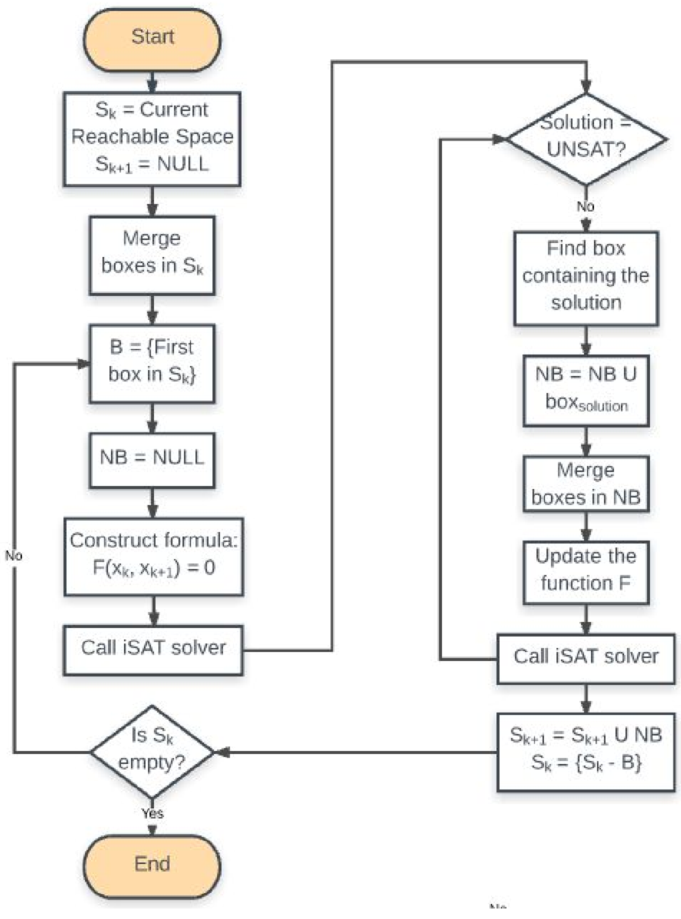

Start
S_k = Current Reachable Space S_{k+1} = NULL
Merge boxes in S_k
B = {First box in S_k}
NB = NULL
Construct formula: $F(x_k, x_{k+1}) = 0$
Call iSAT solver
Solution = UNSAT?
No
Find box containing the solution
NB = NB U $box_{solution}$
Merge boxes in NB
Update the function F
Call iSAT solver
$S_{k+1} = S_{k+1}$ U NB $S_k = \{S_k - B\}$
Is S_k empty?
No
Yes
End

Fig. 12. Reachability Analysis using iSAT Solver

Figure 12 shows the algorithm for reachability analysis using iSAT solver. The input to the algorithm is the current reachable set, which is a set of boxes, S_k. S_{k+1}, the next reachable space is set to be NULL. The boxes in S_k are merged using the algorithm in figure 9. For each element in S_k, formula F is constructed with the constraints on current and next reachable space and iSAT solver is called iteratively until the entire reachable space from that element is retrieved. Set S_{k+1} accumulates the solutions after each iteration and returns the next reachable space at the end of execution.

While the solution of transient verification using iSAT solver algorithm is complete, which is a strong advantage over simulation-based techniques, it can be combined with transient simulation to improve the performance in terms of efficiency. Figure 13 demonstrates this algorithm.

In contrast to the algorithm in figure 12, this algorithm generates a uniform set of sample points from the state space at the beginning. Using simulations, the next projected point is calculated for each sample point. The union of boxes containing these projected points is considered to be the next reachable state space. The remaining unidentifiable space is then determined using the iterative execution of iSAT solver. This way, the number of iteration for iSAT execution is minimized.

6.2 Results: Transient Analysis of Tunnel Diode Oscillator Using Reachability Analysis

Tunnel diode oscillator uses tunnel diode operation for obtaining oscillation. We use the reachability analysis algorithm in figure 13 to verify the startup condition of the oscillator.

Fig. 13. Reachability Analysis using iSAT solver and Transient Simulation

Figure 14 (a) shows the schematic diagram of the tunnel diode oscillator. The input voltage is set to be 2.6V. The time step chosen for the reachability analysis is $\Delta t = 0.2$ ns. The state space is represented by inductor current (I_L) and capacitor voltage (C_V). The initial range of interest in the experiment is $I_L \varepsilon$ [1.6mA, 2.2mA]. The sub-divided state space and the reachability analysis is shown in figure 14(b). The result verifies that the oscillator can generate oscillation for any initial state in the given initial range.

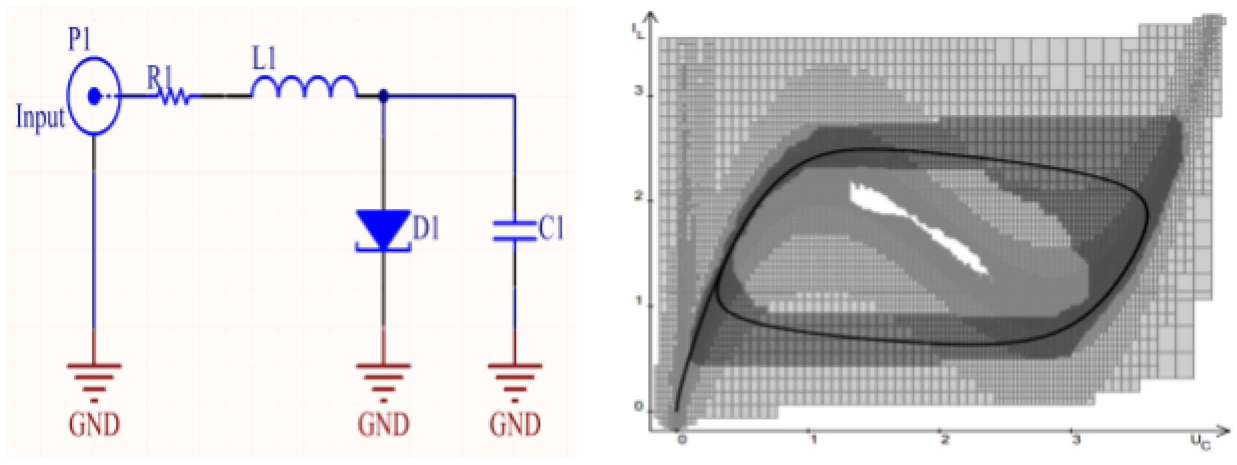


Fig. 14. Reachability Analysis of Tunnel Diode Oscillator

Sample interval	No of simulations	Simulation time (s)	No. of calls to iSAT solver	iSAT solver time (s)	Total time (s)
1/2	0.7M	14.7	29477	7224.6	7239.3
1/3	1.2M	27.4	26061	6254.0	6281.4
1/4	1.9M	41.9	23782	4806.6	4848.5
1/5	2.7M	59.3	21764	4174.1	4233.4
1/6	3.6M	80.0	19943	3712.4	3792.4
1/7	4.7M	105.1	21367	4055.2	4160.3
1/8	5.9M	137.5	22856	4669.4	4806.9
1/9	7.2M	175.6	23224	4782.1	4957.7
1/10	8.5M	227.4	24049	5666.0	5893.4

Table 3. Run-time Performance for a range of sampling interval

In table 3, we observe the effect of sampling rate on the number of calls to the iSAT solver and the run-time performance. As can be seen, the increase in the sampling rate leads to increase in the number of simulations and the simulation time. The best suitable sampling rate can be found out by observing the trend in the number of calls to the iSAT solver. In this example, sampling rate ⅙ is the most suitable since the number of calls reaches its minimum value at this sampling rate, thus minimizing the total solution time.

6.3 Possible alternative

While iSAT guarantees complete coverage, it is also possible to achieve complete coverage by using the DC analysis from chapter 5 and generating a stimuli for transient analysis.

In [4], input stimuli signal is generated using the DC operating points determined and traversing through the state space. This method can be combined with the solution of iSAT solver to provide a conservative solution and complete coverage. This method can be seen as an extension to the DC verification method. First the sub-divided state space is converted to a transition system which can be represented by graphical representation, where each box is represented by a vertex and a set of arrows representing transitions of states. Figure 15 shows an example of the graphical structure for a simple circuit with two state variables. The dark coloured vertices represent the DC operating points of the circuit. Since the circuit will be in steady state at operating point, they have a loop transitioning to themselves. The unidirectional arrows display the dynamic behavior of the circuit.

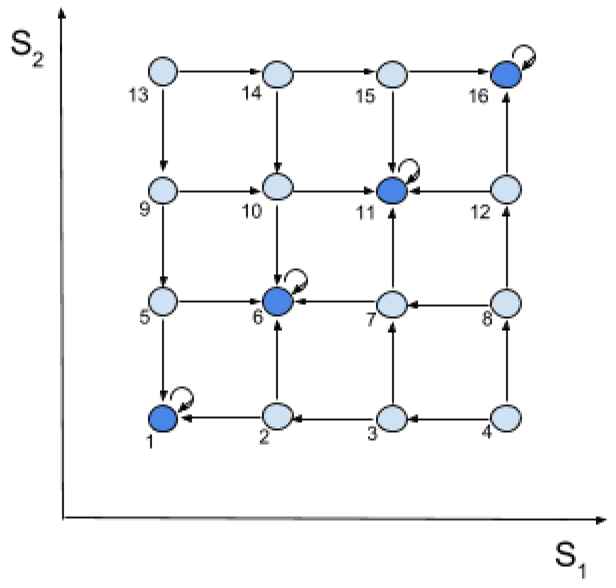


Fig. 15. Representation of Transition System by Graphical Structure

Once the transition system is obtained, input stimuli is generated in such a way that each reachable state and transition, i.e. each vertex and edge is traveled at least once. The number of transitions need to be minimized to increase the efficiency and run-time of the process. Moreover, it is also necessary to periodically visit the DC operating point to recover from the extreme vertices in the transition system. The transient analysis is performed using the algorithm shown in figure 16.

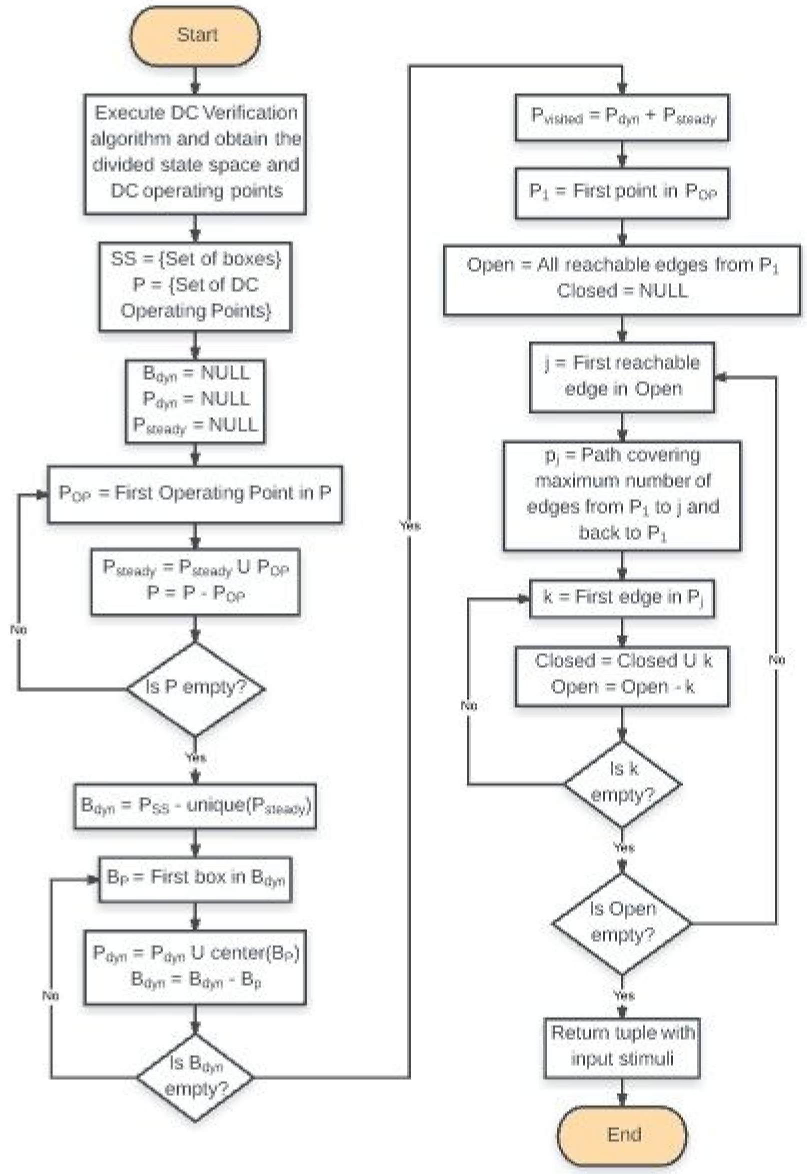

Start
Execute DC Verification algorithm and obtain the divided state space and DC operating points
SS = {Set of boxes}
P = {Set of DC Operating Points}
B_{dyn} = NULL
P_{dyn} = NULL
P_{steady} = NULL
P_{OP} = First Operating Point in P
P_{steady} = P_{steady} U P_{OP}
P = P - P_{OP}
Is P empty?
No
Yes
B_{dyn} = P_{SS} - unique(P_{steady})
B_P = First box in B_{dyn}
P_{dyn} = P_{dyn} U center(B_P)
B_{dyn} = B_{dyn} - B_p
Is B_{dyn} empty?
No
Yes
$P_{visited}$ = P_{dyn} + P_{steady}
P_1 = First point in P_{OP}
Open = All reachable edges from P_1
Closed = NULL
j = First reachable edge in Open
p_j = Path covering maximum number of edges from P_1 to j and back to P_1
k = First edge in P_j
Closed = Closed U k
Open = Open - k
Is k empty?
No
Yes
Is Open empty?
No
Yes
Return tuple with input stimuli
End

Fig. 16. Stimuli Generation Algorithm

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this work, we proposed two primary approaches for analog and mixed-signal circuit verification using satisfiability solver, i.e. DC verification and transient verification.

In DC verification, operating points of the circuit are determined using iSAT solver, a nonlinear satisfiability solver. BSIM model data is used with bounding conditions to simplify the system equations. The algorithm is implemented on a ring oscillator with odd and even number of inverters. It is shown that this algorithm reduces the speed-up by approximately 20% compared to [21] and the speedup keeps getting higher with increasing number of stages.

In transient verification, iSAT solver and tradition simulation technique are combined together for transient analysis of a circuit. The simulation technique finds out the next reachable space before the implementation of iSAT solver to find the remaining reachable space. This reduces the number of calls to the solver, thereby making it efficient. The approach is applied on a tunnel diode oscillator and the reachability analysis is demonstrated using the state space graphical representation.

In addition, an alternative technique for transient analysis is suggested. It combines this work with [4] to generate a stimuli to apply on the circuit to force the space to traverse through the entire state space region for complete coverage.

REFERENCES

[1] R. Narayanan, I. Seghaier, M. Zaki, S. Tahar, "Statistical Run-Time Verification of Analog Circuits in Presence of Noise and Process Variation" in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, October 2013, Vol. 21, No. 10.

[2] S. Little, N. Seegmiller, D. Walter, "Verification of Analog/Mixed-Signal Circuits Using Labeled Hybrid Petri Nets" in Computer-Aided Design, ICCAD 2006, IEEE/ACM International Conference, February 2007, 1558-2434.

[3] R. Dreschler, D. K. Pradhan, "Recent Advances in Verification, Equivalence Checking and SAT Solvers" in The IEEE International Symposium on Circuits and Systems, August 2005, 0-7803-7991-8.

[4] S. Steinhorst, L. Hedrich, "Improving Verification Coverage of Analog Circuit Blocks by State Space-Guided Transient Simulation" in Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), August 2010, 978-1-4244-5309-2.

[5] F. Jiao and A. Doboli, "A Causal Reasoning-based Approach for Analog Circuit Verification" in Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD) International Conference, September 2015, 978-1-4673-9184-9.

[6] M. H. Zaki, S. Tahar, G. Bois, "Formal Verification of Analog and Mixed-Signal Designs: Survey and Comparison" in IEEE North-East Workshop on Circuits and Systems, November 2006, pp.1395-1404.

[7] W. Denman et al., "Formal Verification of Analog Designs using MetiTarski" in Formal Methods in Computer-Aided Design (FMCAD), December 2009, pp. 93-100.

[8] T. Dang, A. Donze, O. Maler, "Verification of Analog and Mixed-Signal Circuits using Hybrid System Techniques" in Hu A.J., Martin A.K. (eds) Formal Methods in Computer-Aided Design (FMCAD), Lecture Notes in Computer Science, vol 3312. Springer, Berlin, Heidelberg.

[9] D. Kulkarni, "Improved Model Generation and Property Specification for Analog/ Mixed-Signal Circuits" in ProQuest Dissertation and Theses (PQDT) Global Database.

[10] L. Hedrich, E. Barke, "A Formal Approach to Verification of Linear Analog Circuits With Parameter Tolerances" in Proceedings of Design, Automation and Test in Europe, August 2002, pp. 649-654.

[11] L. Hedrich, E. Barke, "A Formal Approach to Nonlinear Analog Circuit Verification" in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), August 2002, pp. 123-127.

[12] W. Hartong, L. Hedrich, E. Barke, "Model Checking Algorithm for Analog Verification" in 39th Proceeding of Design Automation Conference, August 2002, pp. 542-547.

[13] S. Little et al., “Analog/Mixed-Signal Circuit Verification using Models Generated from Simulation Traces” in Eighth International Workshop on Microprocessor Test and Verification, September 2008, pp. 191-210.

[14] M. Althoff et al., “Formal Verification of Phase-Locked Loops using Reachability Analysis and Continuization” in IEEE/ACM International conference on Computer-Aided Design (ICCAD), December 2011, pp. 659-666.

[15] A. Antoulas, D. Sorensen, S. Gugercin, “A Survey of Model Reduction Methods for Large Scale Systems” in Contemporary Mathematics, September 2003, Vol. 280, pp. 193-219.

[16] R. Srinivasan, H. Carter, “Occurrence and Simulation of Index-3 DAEs in VLSI Circuits” in IEEE International Workshop on Behavioral Modeling and simulation, September 2008, pp. 61-65.

[17] Q. Chen, S. Weng, C. Cheng, “A Practical Regularization Technique for Modified Nodal Analysis in Large-Scale Time-Domain Circuit Simulation” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, July 2012, Vol. 31, No. 7, pp. 1031-1040.

[18] L. Liu, F. Felgner, G. Frey, “Comparison of 4 Numerical Solvers and Hybrid Systems Simulation” in IEEE Conference on Emerging Technologies and Factory Automation (ETFA), September 2010, 978-1-4244-6850-8.

[19] K. E. Brenan, S. L. Campbell, L. R. Petzold, “Numerical Solution of Initial-value Problems in Differential-algebraic Equations”, Philadelphia: SIAM, 1996.

[20] M. Prasad, A. Biere, A. Gupta, “A Survey of Recent Advances in SAT-based Formal Verification” in International Journal on Software Tools for Technology Transfer, January 2005, Vol. 7, No. 2, pp. 156-173.

[21] S. Tiwary et al., “First Step towards SAT-based Formal Analog Verification” in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), November 2009, pp. 1-8.

[22] M. Davis, G. Logemann, D. Loveland, “A Machine Program for Theorem Proving” in CACM 1962, pp. 394-397.

[23] Wikipedia contributors, “Stability Theory” in Wikipedia, the Free Encyclopedia, Retrieved from https://en.wikipedia.org/w/index.php?title=Stability_theory&oldid=765887538, February 2017.

[24] “Star-HSPICE Manual”, Avant! Co., Fremont CA, July 1998, Ch. 16, pp. 1-7.

[25] G. Frehse, B. H. Krogh, R. A. Rutenbar, “Verifying Analog Oscillator Circuits Using Forward/ Backward Abstraction Refinement” in the Proceedings of the Design Automation and Test in Europe Conference, March 2006, Vol. 1, pp. 257-262.

[26] Y. Deng, “SAT-based Verification for Analog and Mixed-Signal Circuits”, Retrieved from http://oaktrust.library.tamu.edu/bitstream/handle/1969.1/ETD-TAMU-2012-05-11221/DENG-THESIS.pdf?sequence=2.

CURRICULUM VITAE

GRADUATE COLLEGE
UNIVERSITY OF NEVADA, LAS VEGAS
NIKITA RAMESH WANJALE

Contact Details:

Email ID: wanjale@unlv.nevada.edu

Degrees:

- Bachelor of Technology in Instrumentation and Control Engineering, 2010

  Vishwakarma Institute of Technology

- Master of Science in Electrical and Computer Engineering, 2017

  University of Nevada, Las Vegas

Thesis Title: **Analog and Mixed-Signal Circuit Verification using Satisfiability Solver on Discretized Models**

Thesis Examination Committee:

- Committee Chair, Henry Selvaraj, Ph.D
- Committee Member, Biswajit Das, Ph.D
- Committee Member, Grzegorz Chmaj, Ph.D
- Graduate College Representative, Laxmi Gewali, Ph.D

Experience:

- Electrical Engineer I, Scientific Games, March 2017 - Present
- Graduate Teaching Assistant, ECE Dept., UNLV, August 2015 - March 2017
- Trainee Decision Scientist, Mu Sigma, August 2014 - May 2015