

# Formal Verification of Synthesized Analog Designs

Abhijit Ghosh and Ranga Vemuri

*Address for Correspondence:*

Dr. Ranga Vemuri  
Digital Design Environments Laboratory  
ECECS Department, ML. 30  
University of Cincinnati  
Cincinnati, OH 45221-0030

Phone: 513-556-4784

Fax: 513-556-7326

Email: ranga.vemuri@uc.edu

**Track : Verification and Test**

---

This work is sponsored by the US Air Force Research Laboratory, Wright Patterson Air Force Base under contract number F33615-96-C-1911.

# Formal Verification of Synthesized Analog Designs

## Abstract

We present an approach for formal verification of the DC and low frequency behavior of synthesized analog designs containing linear components and components whose behavior can be represented by piecewise linear models. A formal model of the structural description of a synthesized design is extracted from the sized component netlist produced by the synthesis tool, in terms of characteristic behavior of the components and various voltage and current laws. For the synthesized implementation to be correct, it must imply a formal model extracted from a user given behavior specification. Circuit implementation and expected behavior are both modeled in the PVS higher-order logic proof checker as linear functions and the PVS decision procedures are used to prove the implication.

## 1 Introduction

The challenges in formally verifying an analog design are somewhat different from those in verifying digital designs. Analog components exhibit continuous time behavior often represented as an algebraic function of several parameters. This kind of behavior is expressed with linear/nonlinear algebraic functions and is often dependent on the frequency of signals applied. This leads to each component's behavior being abstracted into a set of models according to operating frequencies. For example, an operational amplifier has different models for different operating frequencies. Many analog devices have inherent nonlinearity in their behavior. For example, a transistor has at least three regions of operation (cutoff, linear, and saturation) distinct from one another [1], [2]. A greater difficulty of understanding arises when several such components interact with each other in an analog system. It becomes quite difficult to determine which mode a component is operating on.

The process of high level synthesis essentially involves the user specifying the functional and performance requirements in a high level language and the synthesis tool generating a circuit satisfying these requirements. There are several techniques for analog synthesis [3], [4], [5], [6]. Synthesis and verification are often viewed as the two sides of the same coin [7]. Synthesis tools often restrict the style of design and the operating modes of the components so that effective design space exploration and function/performance analysis is possible during synthesis. These very restrictions can be exploited to achieve automated formal verification of synthesized designs which is often not possible to achieve for hand-crafted designs. In case of analog designs, the restriction is in terms of composing components in a manner that the operating modes of each component can be correctly determined at any time. This in turn facilitates the use of piecewise linear models of component behavior in place of non-linear models while verifying the behavior of synthesized analog designs.

When using piecewise linear models, the behavior of an analog component can be modeled as a conjunction of Boolean predicates which are relations between the voltage and currents at the terminals and various attributes of the components. Interconnections of these components follow current and voltage laws which must be satisfied. These laws can again be modeled as boolean predicates which are relations among the voltages at the nodes and currents in the branches. A circuit composed of subcomponents can be modeled

as a conjunction of these component level Boolean predicates. If we know the expected behavior of such a circuit and are able to model it in a formal logic then we can use a mechanized verifier of that logic to prove that circuit behavior implies the specified functional behavior.

We discuss the existing methods of circuit analysis in Section 2. This is followed by the description of our technique in Section 3. In Section 4 we have given an implementation of our method in VASE [6]. We conclude with a discussion of the scope and limitations of this method and future extensions possible in Section 5.

## 2 Existing Methods

Theory of analog circuit analysis well established. There are a number of well understood techniques such as nodal analysis and tableau analysis which have been automated and used widely. The basic idea of these methods is as follows:

- Model a circuit in terms of primitive components such as independent voltage and current sources, controlled sources and resistive elements,
- Setup equations relating the above using Kirchoff's Voltage and Current Laws,
- Solve these equations using Cramer's rule (if the equations are linear) and use Numerical techniques like Newton Raphson (for nonlinear equations).

Typical components in an analog circuit have different regions of operation and correspondingly different models for them. In order to correctly certify the behavior of a circuit we must analyze all these possible regions of operation. The task of analog verification boils down to proving that one or more components never go into certain regions of operation based on the prevailing circuit conditions.

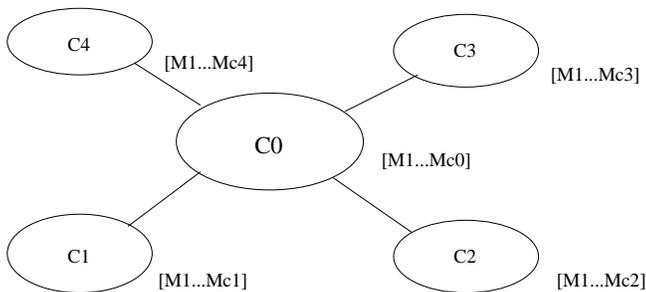


Figure 1: A Component in a General Circuit

An analog circuit, as shown in Figure 1, consists of several components  $C_0, \dots, C_k$  where each component can be in several modes of operation  $[M_1, \dots, M_j]$ . Here the mode in which a component goes into depends upon the modes of operation of other components to which it is connected. The circuit works correctly for particular configurations of modes among the components. In order to guarantee the correctness of the circuit we must be able to prove that those are the only configurations possible for the circuit. Hanna [8] addressed formal modeling and verification of analog and mixed-level circuits using rectilinear and piece wise linear approximations. This work illustrates how such an approach could be effective in verifying the DC behavior of circuits. We have used similar techniques on synthesized analog designs in a higher-order proof checking environment to verify DC and low frequency behaviors.

Traditional verification methods such as simulation using Spice [9] depend on numerical techniques which are accurate but highly compute intensive. Symbolic Analysis Techniques [10] go through multiple stages of approximations which is again time consuming. These methods fail for large designs where many equations need to be solved numerically or symbolic approximations need to be made. As a first line of defense in the design process we might be interested in verifying the functional correctness of the circuit in terms of correct interactions among the components before analyzing the precise magnitudes of the voltages and currents using a simulator. Analog simulation tools like Spice work at the level of device/process equations. However, since we have a good understanding of the terminal behavior and characteristic properties of many analog components such as current mirrors, differential amplifiers, operational amplifiers etc. used in the analog synthesis libraries we can model these components in terms their characteristic properties and use these models to verify the behavior a synthesized analog circuit made up of these components.

By raising the abstraction level from device/process behavior to component behavior we hope to handle larger designs. This approach takes particular significance in analog synthesis where we work with characterized library of analog components which are used to synthesize larger circuits.

### 3 The Verification Technique

An analog synthesis system takes a behavior specification and a library of characterized component topologies as input and produces a netlist of sized components. A verification tool must be able to prove that the sized netlist produced by the synthesis tool has the same behavior as the user given behavior specification. This is illustrated in Figure 2.

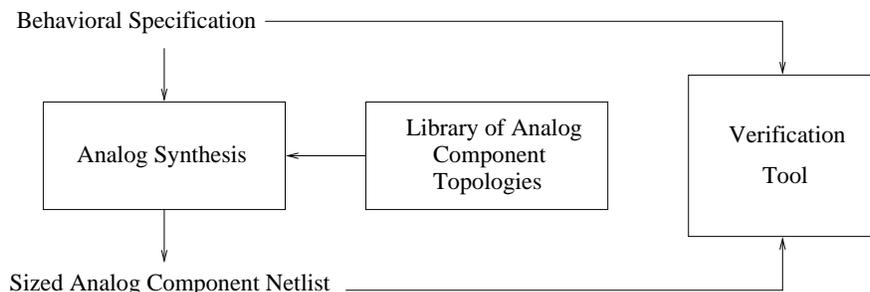


Figure 2: Verification in Analog Synthesis

In our technique as shown in Figure 3, we model the circuit in a higher-order logic theorem proving environment and use the theorem prover’s proof mechanism to prove the implication that the signals appearing at the external terminals of the circuit will satisfy a user given expected behavior of the circuit. Our technique consists of structural model extraction from the synthesized netlist, behavioral model extraction from a user’s HDL specification and a proof that the structural model implies the behavioral model in terms of voltages and currents at the external terminals. We describe the details in the following paragraphs.

#### 3.1 Structural Modeling

A circuit is given as a hierarchical netlist of components such that primitive components appear as the leaf nodes. In the following subsections we describe the modeling of such a circuit. We first need to characterize the primitive components. This is followed by the process of capturing the Kirchoff’s Voltage and Current Laws. Finally we compose the hierarchical components consisting of primitive ones and continue the

process upwards in the hierarchy till we can describe the complete circuit.

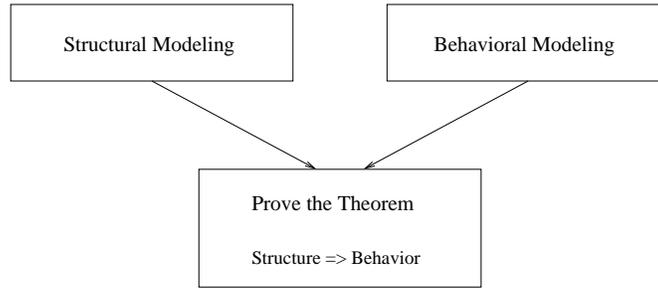


Figure 3: Verification Technique

### 3.1.1 Characterization of the primitive components

Analog synthesis is a process of selecting components from a library in which the components are characterized using their physical attributes and topologies. Given a circuit, when we try to analyze its behavior we first look at the nature of the signals being applied to it and what would be the operating points of such a device.

Each component is modeled by a Boolean-valued function whose parameters are the voltages and currents at its terminals and the attributes of the component. Body of the function must be the relation which must be satisfied by the parameters. Suppose we want to describe a simple analog device like a resistor. It has two terminals, say  $a$  and  $b$ . The voltage at the two terminals are  $V_a$  and  $V_b$ , there is a current of  $I$  flowing through it and it has a non zero resistance of  $R$ , then it will satisfy the following Boolean predicate.

*resistor*( $V_a, V_b, I, R: \text{real}$ ) : bool =  $V_a - V_b = I * R$

Here we have described an ideal resistor in terms of the terminal voltages ( $V_a, V_b$ ) and currents ( $I$ ) and its physical attribute (the resistance  $R$ ). Non-ideality in the behavior of the resistor, say the fact that its resistance changes with temperature may be similarly depicted.

*resistor\_nonideal*( $V_a, V_b, I, \alpha, \text{temp}, R: \text{real}$ ) : bool =  $V_a - V_b = I * R * (1 + \alpha * \text{temp})$

Following is the description of a transistor having Cutoff, Linear and Saturation regions of operation.

*transistor* ( $V_g, V_d, V_s, I_{ds}, V_t, \beta: \text{real}$ ) : bool =  
**Cutoffmode**( $V_g, V_d, V_s, V_t, I_{ds}, \beta$ ) and  
**Linearmode**( $V_g, V_d, V_s, V_t, I_{ds}, \beta$ ) and  
**Saturationmode**( $V_g, V_d, V_s, V_t, I_{ds}, \beta$ )

$V_g, V_d, V_s$  are the terminal voltages at the gate, drain and source respectively,  $I_{ds}$  is the drain to source current and threshold voltage  $V_t$  and  $\beta$  are the physical attributes of the transistor.

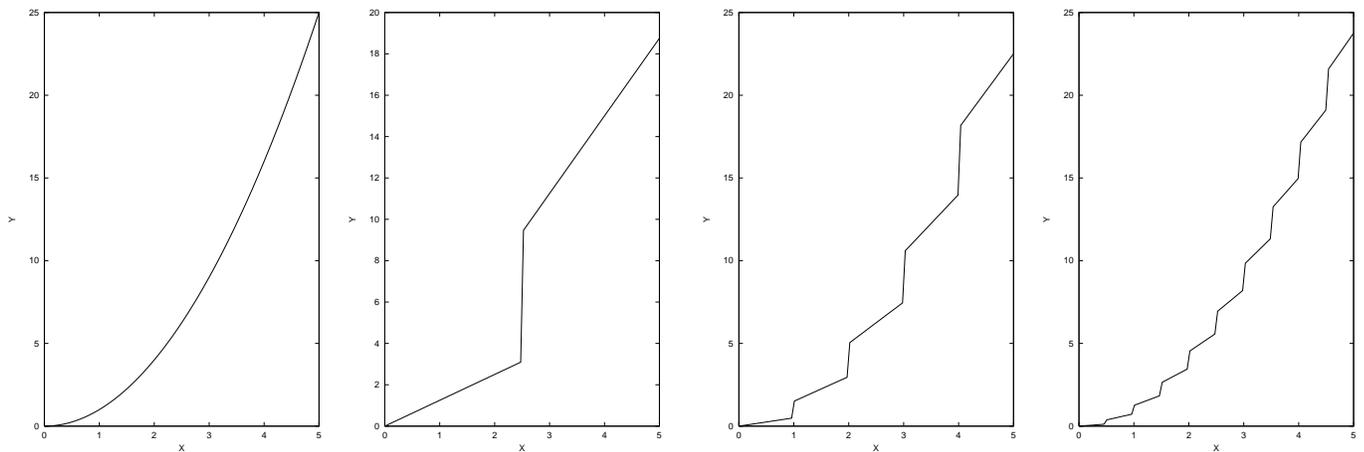
Transistor is in Cutoff if the gate to source bias is less than the threshold voltage, it is in linear region of operation if the gate to source bias is less than drain to source bias and is in saturation otherwise. This behavior of the transistor can be written as one single predicate such that the three regions of operation appear in separate conditional cases.

```

transistor(Vg,Vd,Vs,Ids,Vt,beta:real) : bool =
  if Vg-Vs-Vt <= 0
    then Ids=0                                     /* Cutoff */
    elseif Vg-Vs-Vt > Vd-Vs
      then Ids= beta*(Vd-Vs)*(Vg-Vs-Vt)          /* Linear */
      else Ids= beta*(Vg-Vs-Vt)*(Vg-Vs-Vt)      /* Saturation */
    endif
  endif

```

There is a nonlinear relation between the drain to source current  $I_{ds}$  and the terminal voltages in the linear and saturated regions of operation . We can approximate this behavior with a piecewise linear approximation. The ability to handle linear equalities and inequalities in decision procedures makes these approximations useful. In the following figure we have shown that linear approximation over sufficient number of intervals closely approximates a nonlinear function. Here a desired nonlinear function  $y = x*x$  is approximated linearly in **2, 5, 10** intervals successively. For each interval  $[A, B]$  the approximating linear function is  $y = x * (A + B) / 2$  .



Such an approximation over sufficiently large number of intermediate intervals justifies our following attempt to approximate similar nonlinear behavior of analog components by linear relations. We can describe the above transistor as,

```

transistor(Vg,Vd,Vs,Ids,Vt,beta:real) : bool =
  if Vg-Vs-Vt <= 0
    then Ids=0
    elseif Vg-Vs-Vt > Vd-Vs
      then if Vg-Vs-Vt<= V
            then Ids= beta*(Vd-Vs)*V/2
            else Ids= beta*(Vd-Vs)*(V+Vmax)/2
          endif
      else if Vg-Vs-Vt <= V
            then Ids= beta*(Vg-Vs-Vt)*V/2
            else Ids= beta*(Vg-Vs-Vt)*(V+Vmax)/2
          endif
    endif
  endif

```

In the above description we have approximated the behavior of the transistor in the linear and saturation regions in two intervals  $[0, V]$  and  $[V, Vmax]$ . Both  $V$  and  $Vmax$  are known real constants.  $Vmax$

is the maximum voltage in the circuit and  $V$  is an intermediate voltage which can be typically taken to be  $V_{max}/2$ . This approximation is improved many folds by making several such intervals and piecewise linearly approximating in each such interval. In our verification attempts we have described transistors approximated in five intervals.

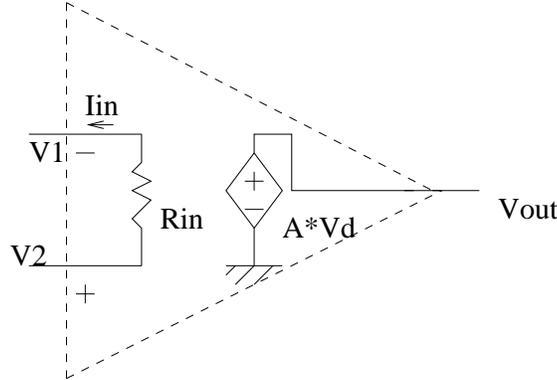


Figure 4: A Small signal model for the Opamp

In Figure 4 we have shown an Operational Amplifier model which consists of an input impedance  $R_{in}$  and a Voltage Controlled Voltage Source  $A * V_d$ . Here  $A$  is the open loop voltage gain and  $V_d$  is the differential input voltage  $V_2 - V_1$ . We have neglected the output impedance with no effect on the generality of our process. The Opamp is expressed as a Boolean function as follows.

```

Opamp (Vdd,Vss,V1,V2,Vout,Iin,A,Rin:real) : bool =
  if A*(V2-V1) >= Vdd
    then Vout = Vdd
    /* PositiveSaturation */
  elseif A*(V2-V1) <=Vss
    then Vout = Vss
    /* NegativeSaturation */
  else Vout = A*(V2-V1)
  /* LinearOperation */
endif and V2-V1 = Iin*Rin

```

We have described the analog components as predicates over terminal voltages, currents and other attributes and expressed them in terms of linear equations and inequalities.

### 3.1.2 Low Frequency Small Signal Modeling

In the low frequency range of operation a transistor can be modeled in terms of transconductance  $g_m$  and output conductance  $g_d$  [1]. Figure 5 shows the small signal model of a transistor. The transconductance  $g_m$  and output conductance  $g_d$  are defined as ,

$$g_m = \frac{dI_{ds}}{dV_{gs}} \mid V_{ds} = constant$$

$$g_d = \frac{dI_{ds}}{dV_{ds}} \mid V_{gs} = constant$$

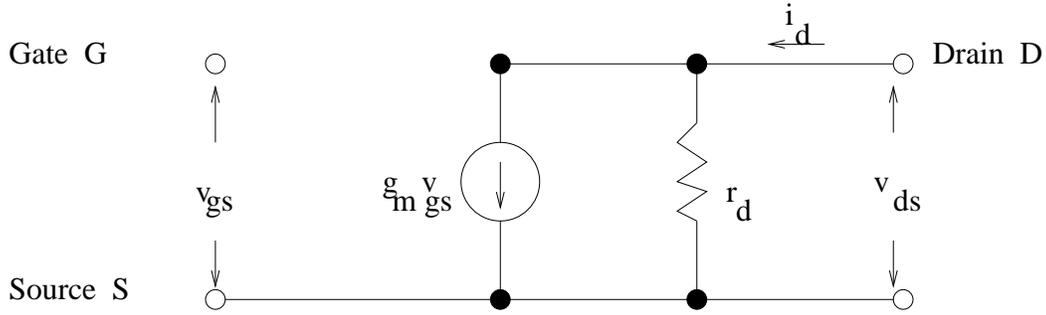


Figure 5: Low frequency small signal model of a FET

Differentiating the transistor equations we can get,

$$\begin{aligned}
 g_m = g_d = 0 & & / * \text{Cutoff} * / \\
 g_m = \beta V_{ds}, g_d = \beta(V_{gs} - V_t) & & / * \text{Linear} * / \\
 g_m = \beta(V_{gs} - V_t), g_d = 0 & & / * \text{Saturation} * /
 \end{aligned}$$

The incremental change in drain current  $i_{ds}$  is given in terms of incremental change in drain to source voltage  $v_{ds}$  and incremental change in gate to source voltage  $v_{gs}$  as,

$$i_d = g_m v_{gs} + g_d v_{ds}$$

The low frequency small signal characteristics is incorporated into our transistor characterization in the following way.

```

transistor (Vg,vg,Vd,vd,Vs,vs,Vt,Ids,ids,beta:real) : bool =
  Exists (gm,gd:real):
    if Vg-Vs-Vt <= 0
      then Ids=0 and gm=0 and gd=0
      elsif Vg-Vs-Vt > Vd-Vs
        then gm=beta*(Vd-Vs) and gd=beta*(Vg-Vs-Vt) and
          if Vg-Vs-Vt <= V
            then Ids= beta*(Vd-Vs)*V/2
            else Ids= beta*(Vd-Vs)*(V+Vmax)/2
          endif
        else gm=beta*(Vd-Vs) and gd=0 and
          if Vg-Vs-Vt <= V
            then Ids= beta*(Vg-Vs-Vt)*V/2
            else Ids= beta*(Vg-Vs-Vt)*(V+Vmax)/2
          endif
      endif and ids=gm*(vg-vs)+gd*(vd-vs)

```

Here  $v_g$ ,  $v_d$ ,  $v_s$ ,  $i_{ds}$  are small changes in gate voltage, drain voltage, source voltage and drain to source current which satisfy the above relation. The values of  $g_m$  and  $g_d$  depend on the operating point of the transistor determined by  $V_g$ ,  $V_d$ ,  $V_s$  and  $I_{ds}$ . This model of the transistor can be used to perform low frequency small signal analysis.

### 3.1.3 Interconnection of Components

In the previous sections primitive analog components were described as the relations being satisfied by the attributes of the components with respect to the terminal voltages and currents. A circuit is a collection of such components where in addition to the laws of the individual components being satisfied, Kirchoff's Voltage and Current Laws of the circuit must also be followed.

Let us consider a simple adder, as shown in Figure 6. The analog components Resistors ( $R_1, R_2, R_3, R_f$ ) and Opamp (*op*) are connected to the external terminals (a,b,Out) and internal nodes (1,2). We can see that for the circuit to function correctly, each of the components must satisfy their own behavior described by the functions of the previous section. The characteristic Boolean functions for the resistor  $R_1, R_2, R_3$  and  $R_f$  gives us the following equations,

$$\begin{aligned} V_a - V_1 &= I_1 * R_1 \\ V_b - V_1 &= I_2 * R_2 \\ V_2 - 0 &= I_3 * R_3 \\ V_{out} - V_1 &= I_f * R_f \end{aligned}$$

Characteristic Boolean functions for the Opamp will result in the following two equations,

$$\begin{aligned} V_{out} &= A * (V_2 - V_1) \\ V_1 - V_2 &= I_{in} * R_{in} \end{aligned}$$

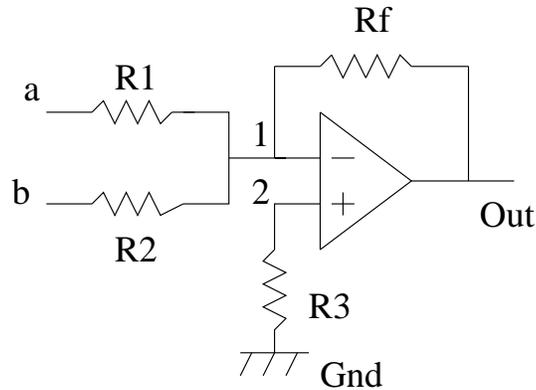


Figure 6: An Example to explain Interconnection of Components

In Figure 7 an expanded view of the adder circuit has been depicted. By Kirchoff's Current Law, we know that the sum of all currents at a node is zero. Hence at the internal nodes, we must have

$$\begin{aligned} \text{Currentlaw1}(I1, I2, I_f, I_{in} : \text{real}) &: \text{bool} = I1 + I2 + I_f + I_{in} = 0 && /* \text{ At Node 1 } */ \\ \text{Currentlaw2}(I_{in}, I3 : \text{real}) &: \text{bool} = I_{in} + I3 = 0 && /* \text{ At Node 2 } */ \end{aligned}$$

Using the above equalities we will be able to show the classic Adder result

$$V_{out} = - \frac{\frac{V_a}{R_1} + \frac{V_b}{R_2}}{\frac{1}{R_f} + \frac{1}{A * R_{in}} + \frac{R_{in} + R_3}{A * R_{in}}} * \left( \frac{1}{R_f} + \frac{1}{R_1} + \frac{1}{R_2} \right)$$

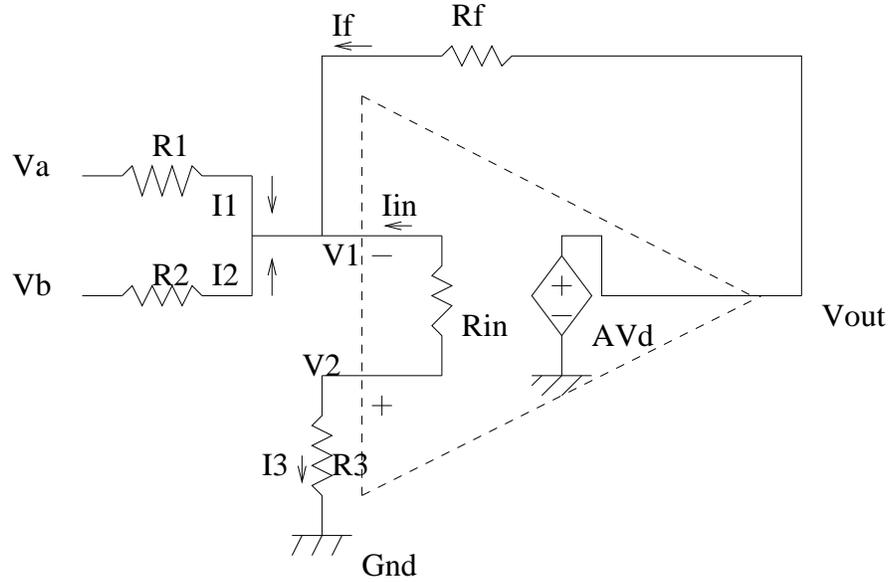


Figure 7: An Expanded view of the Adder Circuit

The Boolean functions `Currentlaw1` and `Currentlaw2` must be satisfied by the components of the adder circuit.

### 3.1.4 Composition of Components

There are many ways to represent circuits. We will view a circuit (see Figure 8) as consisting of;

- a set of nodes ( $n_1 \cdots n_k$ );
- a set of components ( $c_1 \cdots c_k$ ) whose terminals (called the internal terminals of the circuit) are connected to the nodes;
- a set of external terminals ( $e_1 \cdots e_k$ ) also connected to the nodes.

A sub-circuit can be written as a predicate over all voltages and currents at its external terminals such that there exist some voltages and currents at the internal nodes satisfying the constraints imposed by the characteristic behaviors of the individual subcomponents and the conserving the Current and Voltage Laws.

A circuit or a sub-circuit called `Hcomp` is shown in Figure 8. Attaching (voltage, current) tuples with all nodes, external as well as internal, `Hcomp` can be described as,

```

Hcomp(Ve1,Ie1,Ve2,Ie2,Ve3,Ie3,A1,A2,A3) : bool =
  Exists (Vn1,In1,Vn2,In2,Vn3,In3) :
    C1(Ve1,Ie1,Vn1,In1,Vn2,In2,Vn3,In3,A1) and
    C2(Vn1,In1,Vn2,In2,Vn3,In3,A2) and
    C3(Ve3,Ie3,Vn1,In1,A3) and
    VoltageCurrentLaws(Ve1,Ie1,Ve2,Ie2,Ve3,Ie3,Vn1,In1,Vn2,In2,Vn3,In3)

```

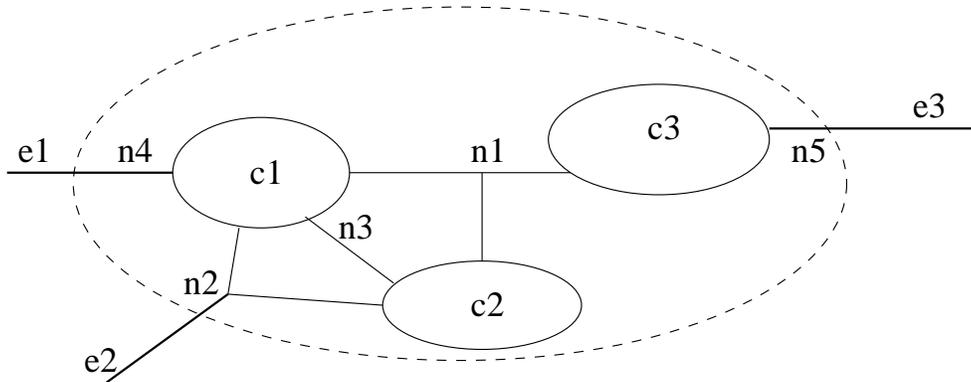


Figure 8: A Sub-Circuit

In the above description, attribute of a component  $C_j$  is given by  $A_j$ . We have shown only one attribute per component where as a component can have any number of them. Following is the model of the adder circuit. We have not used any explicit voltage laws here, but we have associated each node in the circuit with a single voltage variable which follows the Kirchoff's Voltage law.

```

adder (Va, I1, Vb, I2, Vgnd, I3, Vout:real, R1, R2, R3, Rin, Rf, A:nzreal) : bool =
  Exists (V1, V2, Iin, Ifb:real) :
    resistor(Va, V1, I1, R1) and
    resistor(Vb, V1, I2, R2) and
    resistor(V2, Vgnd, I3, R3) and
    resistor(Vout, V1, Ifb, Rf) and
    opamp(V1, V2, Vout, Iin, A, Rin) and
    Currentlaw1(I1, I2, If, Iin) and
    Currentlaw2(Iin, I3)

```

### 3.2 Behavioral Modeling

Behavior specification of the circuit can be given by the user in a specification language whose formal semantics is understood. We are analyzing circuits whose continuous time behavior is an algebraic relation between the input and output, hence algebraic operators (like addition, multiplication) can be used to describe them. We can also allow conditional constructs in such a specification. A language with algebraic operators and conditional cases provides the ability to specify a variety of behaviors.

Subset of VHDL-AMS [11] is used as the input specification language for some Analog Synthesis Tools. There has been effort on part of researchers [12], [13] to use a subset of this specification language for synthesis purposes. The idea here is to give the user ability to specify a wide range of analog behaviors and retain the ease of transformation of the language constructs into analog circuit. We have used a similar subset of VHDL-AMS for behavior specification. In our subset

- external terminals of an ENTITY are the PORTS;
- body of the ARCHITECTURE of an ENTITY consists of simple simultaneous statements which can be algebraic operations involving additions and multiplications or conditional statements;

An example of such a specification of a telephone receiver [14] is shown below.

```

ENTITY telephone IS
PORT(
  QUANTITY Vline:IN real;
  QUANTITY Vlocal:IN real;
  QUANTITY Vearph:OUT real)
END ENTITY;
ARCHITECTURE behavioral OF telephone IS
QUANTITY rvar:real;
  Vearph== (Vline+Vlocal)*rvar;
  IF (Vline'ABOVE(Vth))USE
    rvar==r1c;
  ELSE
    rvar==r1c+r2c;
  END USE;
END ARCHITECTURE;

```

Such a specification can be directly translated into a formal model as shown in Figure 9. PORTS are translated as the external terminals, local QUANTITIES used inside the ARCHITECTURES are existentially quantified in the boolean functions and simultaneous statements in the body of the ARCHITECTURE are directly translated as algebraic relations with conjunctions between them. We have shown the example for one ARCHITECTURE for an ENTITY.

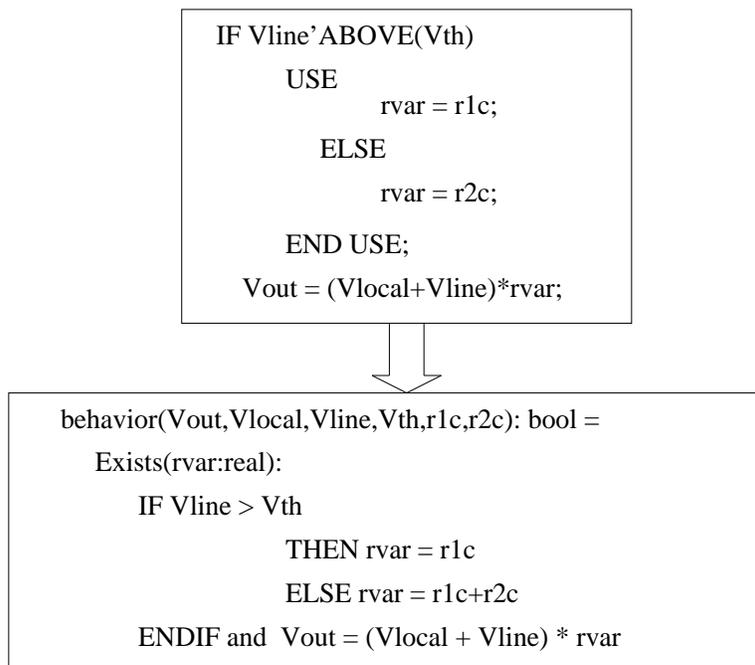


Figure 9: Behavioral Specification Extraction

## 4 Application of the technique in VASE

VASE [6], is an analog synthesis system, which takes a behavioral specification in VHDL-AMS as input [13] and comes up with an analog circuit consisting of components from a characterized library [15] satisfying the performance constraints. It goes through many optimization and estimation steps [16] to achieve the performance goals. The analog circuit thus produced is then traditionally simulated to verify the functional correctness.

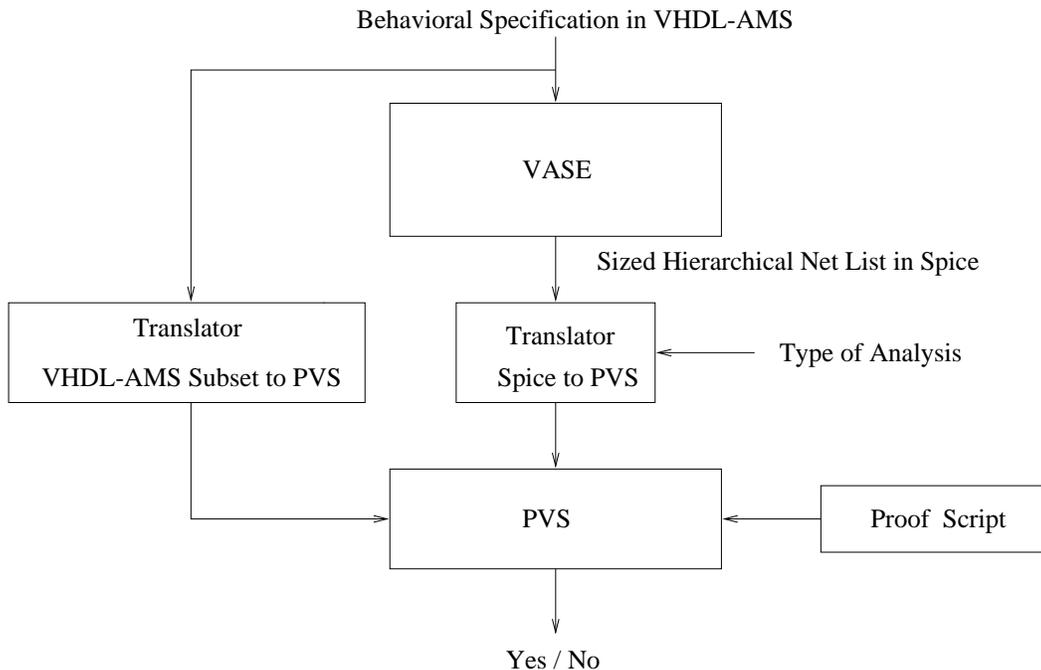


Figure 10: Verifier in VASE

In this research, we have formally verified the functional correctness of the synthesized circuits using Higher Order Logic Proof Checker PVS [17]. PVS decision procedures' support of linear arithmetic on real numbers and our technique of describing the multi-mode primitive components using piecewise linear models facilitated this process. Application of our technique in VASE is shown in Figure 10. Our technique consists of two major steps.

- Generating the PVS file, consisting of a structural model, a behavioral model and a theorem for Structural Model  $\Rightarrow$  Behavioral Model.
- Coming up with a proof strategy to automatically prove the correctness theorem.

### 4.1 PVS file generation

VASE produces a hierarchical spice netlist of sized transistors, which is used to extract the structural model. To generate the structural model we also take the user input to identify the type of analysis required. We are able to describe DC and small signal models of transistors and based on the user input we decide which model should be used to generate the structural description of the circuits. The circuit can be described in PVS from its spice netlist using the following algorithm.

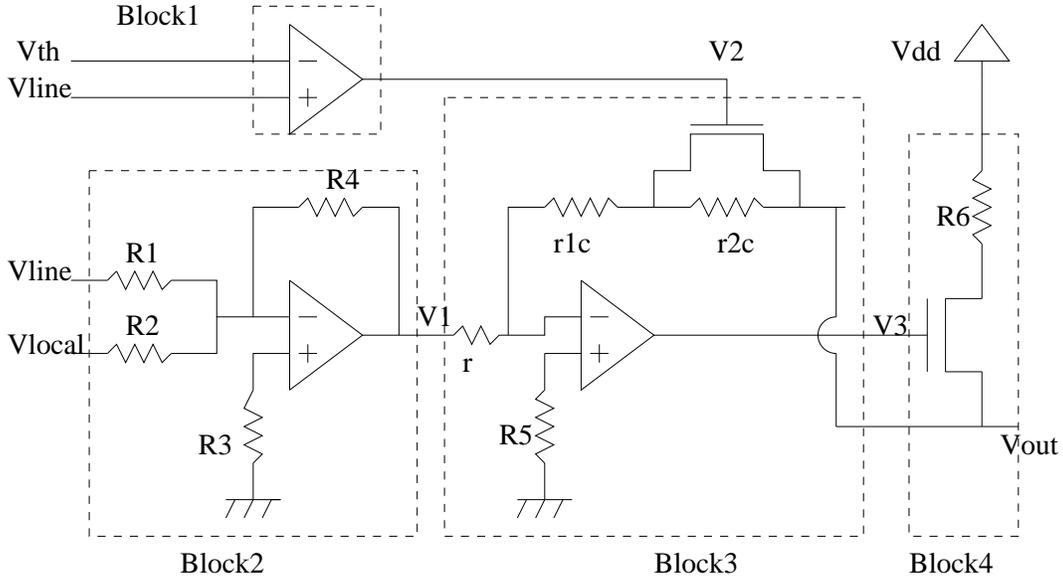


Figure 11: The Telephone Receiver

- Identify the primitive components used and write the corresponding Boolean functions based on the user given type of analysis;
- Write the functions for the higher level components using the composition technique described in Section 3.1, using the functions for lower level components;
- Continually repeat the previous step until the circuit is described which is the top level entity.

The behavioral model is extracted from VHDL-AMS specification using the translation mechanism described in section 3.2

For the telephone receiver behavioral specification given in section 3.2, circuit shown in Figure 11 was synthesized by the VASE tool. Generating the specification for the circuit with our above algorithm, we will get the following function representing the structure. We have generated similar functions for the individual blocks.

```

Recv (Vlocal,Vline,Vth,Vout:real,R1,R2,R3,R4,R5,R6,r,r1c,r2c,Rin,A,beta:nzreal):bool =
  Exists(V1,V2,V3,I:real):
    block1(Vline,Vth,V2) and
    block2(Vlocal,Vline,V1,R1,R2,R3,Rin,R4,A)and
    block3(V1,V2,V3,Vout,I,r,R3,r1c,r2c,Rin,A,beta) and
    block4(V3,Vout,I,R6,beta)

```

Our example in Figure 11 has the following behavior. This was given by the user in a VHDL-AMS specification as presented in section 3.2. We have restricted the VHDL-AMS to subset as discussed in that section, making a direct translation of such a specification into PVS possible.

```

IF Vline>Vth
  THEN Vout= (Vlocal+Vline) * (r1c/r)
  ELSE Vout= (Vlocal+Vline) * ((r1c+r2c)/r)

```

The theorem which we need to prove in order to ascertain that this implementation of the receiver will satisfy the behavior desired is the following.

**Recvtheorem: THEOREM**

```

forall (Vlocal,Vline,Vth,Vout,bound1,bound2:real):
  exists (R1,R2,R3,R4,R5,R6,r,r1c,r2c,Rin,A,beta:real):
    recv(Vlocal,Vline,Vth,Vout,R1,R2,R3,R4,R5,R6,r,r1c,r2c,Rin,A,beta) =>
      if Vline> Vth
        then (Vout-(Vlocal+Vline)*(r1c/r)<=bound1) and
              (Vout - (Vlocal+Vline)*r1c/r>=bound2)
        else (Vout-(Vlocal+Vline)*((r1c+r2c)/r)<=bound1) and
              (Vout - (Vlocal+Vline)*((r1c+r2c)/r)>=bound2)
      endif

```

We have used piecewise linear approximations to model some components. These approximations lead to errors in the difference between expected and available values. We have tried to prove that these differences lie within some error bounds. Here in this example we say that the absolute difference between the expected and available outputs is less than 0.001. These error bounds can be user given quantities. Once we are able to prove the above theorem, it means that for the given values of resistance, transistor betas, opamp gains the output of the above circuit will be close to the user expected output within the error bounds.

## 4.2 Proof Strategy

PVS maintains a proof tree where each node is a proof goal. Through the process of proving, user has to complete the tree such that all the leaves are true. Each node or proof goal is a sequent which is a sequence of formulas in the antecedents and consequents. Conjunction of antecedents must imply the disjunction of consequents.

If we look at our Structural model as the Implementation and the expected Behavior model as the Specification then the Theorem we have to prove has the following sequent :

$$\vdash \forall e ( \exists i,a : \text{Implementation}(e,i,a) \Rightarrow \text{Specification}(e) )$$

Here  $e$  represents the external variables,  $i$  represents the internal variables and  $a$  represents the attributes of the implementation. Our proof strategy essentially is,

- Remove the universal quantification using the rule *skolem* which leaves us with,
 
$$\vdash \exists i,a : \text{Implementation}(e!,i,a) \Rightarrow \text{Specification}(e!)$$
 , in the consequent.
- Use *flatten* to generate the following sequent
 
$$\exists i,a: \text{Implementation}(e!,i,a) \vdash \text{Specification}(e!)$$
- Remove the existential quantification in the antecedent using *skolem*, so that we have
 
$$\text{Implementation}(e!,i!,a!) \vdash \text{Specification}(e!)$$
- Recursively use *expand* to rewrite the functions at the circuit, block and component levels, so that we are left with the characteristic equations in the antecedent.
- Use *split*, *lift-if* to remove the conditional cases. Now we have a set of linear equalities and inequalities.

Input Circuit	Opamps	Resistors	Transistors
Telephone Receiver Topology 1	3	9	2
Telephone Receiver Topology 2	3	11	1
Telephone Transmitter	2	8	1
Neural Controller	5	12	0

Table 1: Results

- Use *assert* to invoke the PVS decision procedures to finally prove the consequents.

PVS allows us to write our own proof strategies with the above proof commands. Such a proof strategy can be extracted out of the design flow of the synthesis process.

### 4.3 Other Examples

We have applied our verification technique to other analog designs. We present them in Table 1. The circuits are categorized according to the number of different analog components used in them. The execution time of PVS on these examples is of the order of a few minutes on Sun Sparc Ultra 2 workstation (296 MHz, 128Mb RAM).

## 5 Conclusions

We have described a technique for formal verification of synthesized analog circuits using automated theorem proving. The technique is useful for DC and small-signal behavior verification and is based on piecewise linear approximation of the behavior of the analog components used in synthesis. Piecewise linear approximation facilitates automated verification using the linear arithmetic decision procedures in the PVS proof-checking system. We have verified several synthesized analog circuits representative of typical circuits (in terms of size and complexity) that current analog synthesis tools are capable of generating. We are currently exploring methods to verify large-signal behavior of synthesized circuits at higher frequency ranges.

## References

- [1] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design*. Addison Wesley Publishing Company, 1993.
- [2] Jacob Millman and Christos C. Halkias. *Integrated Electronics*. Tata Mcgraw Hill, 1991.
- [3] B.A. Antao and A.J. Brodersen. "Techniques for Synthesis of Analog Integrated Circuits". *IEEE Design & Test of Computers*, pages 8–18, March 1992.
- [4] L. R. Carley, G.E. Gielen, R. A. Rutenbar, and W. Sansen. "Synthesis Tools for Mixed-Signal ICs: Progress on Frontend and Backend Strategies". *Proceedings of the 33rd Design Automation Conference*, pages 298–303, June 1996.

- [5] R. Harjani, R. Rutenbar, and L. Carley. "OASYS: a framework for analog circuit synthesis". *IEEE Trans. CAD*, 8(12):1247–1266, December 1989.
- [6] Alex Doboli, N. Dhanwada, A. Nunez-Aldana, , Sree Ganesan, and R. Vemuri. "Behavioral Synthesis of Analog Systems using Two-Layered Design Space Exploration". *Proceedings of 36th Design Automation Conference*, June 1999.
- [7] S. Johnson, R. Wihmeister, and B. Bose. "On the Interplay of Synthesis and Verification". *Applied Formal Methods for Correct VLSI Design Proceedings of the IMEC-IFIP International Workshop*, pages 385–404, November 1989.
- [8] Keith Hanna. "Automatic Verification of Mixed-Level Logic Circuits". *Proceedings of Formal Methods in Computer-Aided Design*, pages 133–148, 1998.
- [9] L.W.Nagel. "SPICE2: A Computer Program to Simulate Semiconductor Circuits". Technical Report Memo ERL-M520, Berkley, Calif, May 1975.
- [10] G. E. Gielen, H. Walscharts, and W. Sansen. "ISAAC: A symbolic simulator for Analog Integrated Circuits". *IEEE J. Solid-State Circuits*, 24(6):1587–1597, December 1989.
- [11] "*IEEE Standard VHDL Language Reference Manual (Integrated with VHDL-AMS changes)*". IEEE Std.1076.1.
- [12] Alex Doboli and R. Vemuri. "Definition of a VHDL-AMS Subset for Behavioral Synthesis Analog System". *Proceedings of IEEE/VIUF International Workshop of Behavioral Modeling and Simulation*, 1998.
- [13] A. Doboli and R. Vemuri. "A VHDL-AMS Compiler and Architecture Generator for Behavioral Synthesis of Analog Systems". *Proceedings of Design Automation and Test in Europe*, March 1999.
- [14] J.Trontely, L.Trontelj, and G.Shenton. *Analog Digital Asic Design*. McGraw-Hill Book Company, 1989.
- [15] A. Nunez-Aldana and R. Vemuri. "An Analog Performance Estimator for Improving the Effectiveness of CMOS Analog Systems Circuit Synthesis". *Proceedings of Design Automation and Test in Europe*, March 1999.
- [16] N. Dhanwada, A. Nunez-Aldana, and R. Vemuri. "Hierarchical Constraint Transformation using Directed Interval Search for Analog System Synthesis". *Proceedings of Design Automation and Test in Europe*, March 1999.
- [17] N.Shankar, S.Owre, and J.M.Rushby. "*The PVS Proof Checker: A Reference Manual (Beta Release)*", March 1993.