Formal Verification of Analog and Mixed Signal Designs in Mathematica

Mohamed H. Zaki, Ghiath Al-Sammane, and Sofiène Tahar

Dept. of Electrical & Computer Engineering, Concordia University
1455 de Maisonneuve W., Montréal, Québec, H3G 1M8, Canada
{mzaki,sammane,tahar}@ece.concordia.ca

Abstract. In this paper, we show how symbolic algebra in Mathematica can be used to formally verify analog and mixed signal designs. The verification methodology is based on combining induction and constraints solving to generate correctness for the system with respect to given properties. The methodology has the advantage of avoiding exhaustive simulation usually utilized in the verification. We illustrate this methodology by proving the stability of a $\Delta\Sigma$ modulator.

Keywords: AMS Designs, Formal Verification, Mathematica.

1 Introduction

With the latest advancement of semiconductors technology, the integration of the digital, analog and mixed-signal (AMS) designs into a single chip was possible and led into the development of System on Chip (SoC) designs. One of the main challenges of SoC designs is the verification of AMS components, which interface digital and analog parts. Traditionally, analyzing the symbolically extracted equations is done through simulation [1]. Due to its exhaustive nature, simulation of all possible scenarios is impossible, and hence it cannot guarantee the correctness of the design. In contrast to simulation, formal verification techniques aim to prove that a circuit behaves correctly for all possible input signals and initial conditions and that none of them drives the system into an undesired behavior. In fact, existing formal methods [2] are time bounded, where verification is achieved only on a finite time interval. We overcome this limitation by basing our methodology on mathematical induction, hence any proof of correctness of the system is time independent. In this paper, we show how symbolic algebra in Mathematica can be used to formally verify the correctness of AMS designs. We illustrate our methodology by applying it to prove the stability of a $\Delta\Sigma$ modulator [3].

The proposed verification methodology is based on combining induction and constraints solving to generate correctness proof for the system. This is achieved in two phases; modeling and verification, as shown in Figure 1. Starting with an AMS description (digital part and analog part) and a set of properties, we extract, using symbolic simulation, a System of Recurrence Equations (SRE) [4]. These are combined recurrence relations that describe each property in terms of the behavior of the system. SRE is used in the verification phase along with an inductive based proof with constraints defined inside *Mathematica* (details can be found in [4]). If a proof is obtained, then the

Y. Shi et al. (Eds.): ICCS 2007, Part II, LNCS 4488, pp. 263–267, 2007.

property is verified. Otherwise, we provide counterexamples for the non-proved properties using *reachability criteria*. If the counterexample is realistic (strong instance), then we have identified a problem (bug) in the design, otherwise the counterexample is spurious (weak instance) and should be eliminated from the verification process.

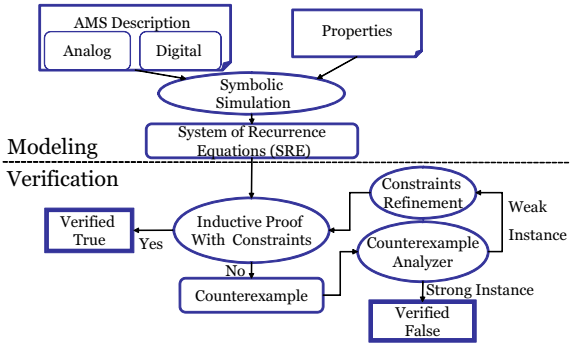

Fig. 1. Overview of the Methodology

2 Implementation in Mathematica

An SRE is a system of the form: $X_i(n) = f_i(X_j(n-\gamma)), (j,\gamma) \in \varepsilon_i, \forall n \in \mathbb{Z}$, where $f_i(X_j(n-\gamma))$ is a generalized `If-formula` (see [5] for a complete definition). The set ε_i is a finite non empty subset of $1,\ldots,k \times \mathbb{N}$. The integer γ is called the delay. A property is a relation of the form: P = $quanta(X, cond, expr)$, where $quanta \in \{\forall, \exists\}$, X is a set of variables. *cond* is a logical proposition formula constructed over X and *expr* is an `If-formula` that takes values in the Boolean Domain $\mathbb{B}$.

Proving Properties. Mathematical induction is then used to prove that a property $P(n)$ holds for all nonnegative integers $n \geq n_0$, where n_0 is the time point after which the property should be *True*:

- Prove that $P(n_0)$ is *True*.
- Prove that $\forall n > n_0, P(n) \Rightarrow P(n+1)$.

The induction algorithm is implemented in Mathematica using functions like *Reduce*, *Assuming* and *Refine*. It tries to prove a property of the form $quanta(X, cond, expr)$, otherwise it gives a counterexample using $FindCounterExample$:

If $Prove(quanta(X, cond, expr)) = True$ then

$Return(True)$

else

$FindCounterExample(cond \wedge \neg expr, var)$

Finding Counterexamples. The basic idea is to find particular variable values for which the property is not satisfied. This is implemented using the Mathematica function $FindInstance[expr, vars, assum]$. It finds an instance of *vars* that makes *expr* True

if an instance exists, and gives {} if it does not. The result is of the following form $\{\{v_1 \to instance_1, v_2 \to instance_2, \ldots, v_m \to instance_m\}\}$, where $vars=\{v_1, v_2, \ldots, v_m\}$. *FindInstance* can find instances even if *Reduce* cannot give a complete reduction. For example, the Mathematica command *FindInstance* $[x^2 - ay^2 == 1 \;\&\&\; x > y, \{a, x, y\}]$ returns $\{\{a \to -\frac{1}{2}, x \to -\frac{1}{\sqrt{2}}, y \to -1\}\}$

We need to insure that the instance is reachable by the SRE before considering it as a counterexample. For example, we verify the recurrence equation $U_n = U_{n-1} + 1$ against the property $\forall n > 0$. $P_n = U_n > 0$. *FindInstance* transforms P_n to an algebraic one and gives the instance $U_{n-1} \to -2$. However, this instance will never be reachable by U_n for $U_0 = 0$. Depending on the reachability, we name two types of SRE instances:

- *Strong instance*: if it is given as a combination of of the design input values. Then, the counterexample is always reachable.
- *Weak instance*: if it is a combination of input values and recurrence variables values. In this case,there in no guarantee that the counterexample is reachable.

If the recurrence equations are linear and if the conditions of the *If-formula* are monotones, then we can search directly for a reachable *strong instance*. We can solve these equations in Mathematica using the function *RSolve*$[Eq_1, Eq_2, \ldots, X_i(n), \ldots, n]$. It returns an explicit solution of the SRE $\{Eq_n\}$ in terms of time relations where the time n is an explicit parameter. We use *RSolve* to identify a time point at which a desired behavior is reached.

3 First-Order ΔΣ Modulator

ΔΣ modulators [3] are used in designing data converters. It is stable if the integrator output remains bounded under a bounded input signal. Figure 2 shows a first-order ΔΣ of one-bit with two quantization levels, $+1$V and -1V. The quantizer (input signal $y(n)$) should be between -2V and $+2$V in order to not be overload. The SRE of the ΔΣ is :

$$y(n) = y(n-1) + u(n) - v(n-1)$$
$$v(n-1) = IF(y(n-1) > 0, 1, -1)$$

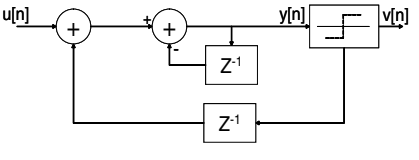

Fig. 2. First-order ΔΣ Modulator

The stability is expressed with the following properties:

Property 1. $\forall |u| \leq 1 \wedge |y(0)| \leq 1 \Rightarrow |y(n)| \leq 2$. To ensure that the modulator will always be stable starting from initial conditions. In Mathematica to prove the property at time n we write:

```
in[1]:= Reduce[
  ForAll[{u,y[n-1]}, And[-1< u < 1, -2< y[n-1] < 2 ],
        And[(-1+u+y[n-1] ⩽ 2) ,(1+u+y[n-1] ⩾ -2)]], {u,y[n-1]}, Reals]
out[1]:= True
```

Property 2. $\forall |u| > 1 \wedge |y(0)| \leq 1 \Rightarrow |y(n)| > 2$. If the input to the modulator does not conform to the stability requirement in Property 1, then the modulator will be unstable:

```
in[1]:= FindInstance[And[ 1<u , 1> y>0 ,(-1+u+y>2) ],u,y]
out[1]:= {u → 7/2, y → 1/2 }
```

As $y = \frac{1}{2}$ is already a valid state for $y[n]$, then the instance is weak. We refine the instance by adding it to the constraints list and restart the proof:

```
in[1]:= Assuming[And[ u → 7/2, 1> y>0 ] ,Refine[(-1+u+y>2)]]
out[1]:= True
```

Thus, the instance $u \rightarrow \frac{7}{2}$ is a strong instance for any $y[n]$.

Property 3. $\forall |u| \leq 1 \wedge |y(0)| > 2 \Rightarrow \exists n_0 > 0 \wedge \forall n > n_0.\ |y(n)| < 2$. If the input of the quantizer is distorted and cause the modulator to be temporarily unstable, the system will return to stable region and stay stable afterwards; which means that there exist an n for which the modulator will be stable for all $n > n_0$. *Rsolve* is used along with *FindInstance* to search for this n_0. We have two cases: $y[n-1] > 0$ and $y[n-1] < 0$. In Mathematica to prove the property, we write:

```
in[1]:= Eq=y[n+1]==(-1+u+y[n]);
    RSolve[Eq&&y[0]== a ,y[n],n]
out[1]:= y[n] → a-n+n u
in[2]:= Reduce[a+n+n u>-2 && u>-1 && a ∈ Reals,n]
out[2]:= a ∈ Reals && u > (-1) && n > (-2-a)/(1+u)
in[3]:= FindInstance[a < -2 && n > 2 && 1 > u > 0.5 &&
         n > (-2-a)/(1+u), {a, u,n}]
out[3]:= {a → -5.5, u → 0.75, n → 4}
```

Thus, we have found a time value which provides a proof for the property: $n > \frac{-2-a}{1+u}$. As the property is formalized using the existential quantifier, it is enough to find one instance: $n_0 \rightarrow 4$.

4 Conclusions

We have presented how Mathematica can be used efficiently to implement a formal verification methodology for AMS designs. We used the notion of SRE as a mathematical model that can represent both the digital and analog parts of the design. The induction based technique traverses the structure of the normalized properties and provides a correctness proof or a counterexample, otherwise. Our methodology overcomes the time bound limitations of conventional exhaustive methods. Additional work is needed in

order to integrate the methodology in the design process, like the automatic generation of the SRE model from design descriptions given in HDL-AMS languages.

References

1. Gielen, G.G.E., Rutenbar, R.A.: Computer-aided Design of Analog and Mixed-signal Integrated Circuits. In: Proceedings of the IEEE. Volume 88. (2000) 1825–1852
2. Zaki, M.H., Tahar, S., Bois, G.: Formal Verification of Analog and Mixed Signal Designs: Survey and Comparison. In: NEWCAS'06, Gatineau, Canada, IEEE (2006)
3. Schreier, R., Temes, G.C.: Understanding Delta-Sigma Data Converters. IEEE Press-Wiley (2005)
4. Al-Sammane, G., Zaki, M.H., Tahar, S.: A Symbolic Methodology for the Verification of Analog and Mixed Signal Designs. In: DATE'07, Nice, France, IEEE/ACM (2007)
5. Al-Sammane, G.: Simulation Symbolique des Circuits Decrits au Niveau Algorithmique. PhD thesis, Universite Joseph Fourier, Grenoble, France (2005)