Constraint-Based Verification of Delta-Sigma Modulators using Interval Analysis

Ghiath Al Sammane, Mohamed H. Zaki, Sofiène Tahar and Guy Bois[‡]
ECE Dept., Concordia University, Montreal, Quèbec, Canada
[‡] Genie Informatique, Ecole Polytechnique de Montreal, Montreal, Quèbec, Canada
Email: {sammane, mzaki, tahar}@ece.concordia.ca, guy.bois@polymtl.ca

***Abstract*— Recently, constraint based methods have been advocated as verification solution for Analog and Mixed Signal (ASM) designs. Constraint-based verification (CBV) is based on the concept of adding properties as constraints to the design such that detection of possible errors is due to constraint violation during the simulation. In this paper, we propose a new method to conduct constraint-based verification on a $\Delta\Sigma$ modulator using interval analysis. In particular, we are interested in the verification of the stability properties of $\Delta\Sigma$ modulators.**

I. Introduction

Development of computer aided design (CAD) methods for analog and mixed signal (AMS) system, is usually focussed to overcome challenges in the design process. Among the challenges encountered is the verification task, concerned to check whether an AMS design is robust with respect to different types of inaccuracies like parameters tolerances, nonlinearities, etc. Such inaccuracies can lead to erroneous behaviors of the system under development. In general, the verification relies largely on a mixture of formal analysis and simulation. For instance, one is interested in global properties connected to the dynamic behavior of the design. For example, we might be interested in properties like: *can we reach from the initial state a state where a certain condition holds?*.

One important class of AMS designs is the $\Delta\Sigma$ modulators [11], needed in data converters at the interface of the analog environment and the digital processing components. The principle of the $\Delta\Sigma$ architecture is to make rough evaluations of the input signal over several stages, to measure the error, integrate it and then compensate for that error. One important requirement in $\Delta\Sigma$ designs is therefore the stability of the output signal. A $\Delta\Sigma$ modulator is said to be stable if each integrator output remains bounded under a bounded input signal.

In this paper, we propose a new method to conduct constraint-based verification on a $\Delta\Sigma$ modulator using interval analysis. In general, *constraint-based verification* (CBV) [14] is based on the concept of adding properties as constraints to the design such that detection of possible errors is due to constraint violation during the simulation. In particular, we are interested in the verification of the stability properties of $\Delta\Sigma$ modulators.

The paper structure is as follows: We start by discussing relevant related work in Section II, afterwards, we describe the proposed verification methodology in Section III. An Illustrative $\Delta\Sigma$ modulator verification is presented in Section IV, before concluding with a discussion in Section V.

II. Related Work

AMS Formal Verification. Several approaches have been proposed for the formal verification of AMS designs. For instance, model checking and reachability analysis were proposed for validating AMS designs over a range of parameter values and a set of possible input signals. These approaches rely on the discretization of the state space by using over-approximating domains like polyhedra models, as was implemented in the tools d/dt [5], *Checkmate* [8] and *PHaver* [6], which have been used to verify different classes of AMS designs (e.g., oscillators, filters). In the approach we propose, we use interval domains, which are in general computationally less expensive than polyhedra models. In addition, we combine the verification with symbolic simulation methods to simplify the system equations at each time step.

In [9], the authors used interval methods to construct the abstract state space. They used heuristics to identify possible transitions between adjacent regions. This requires to divide the whole state space prior to verification, which can be computationally expensive. In addition, this approach is only suitable for analog designs, while our approach supports AMS designs.

$\Delta\Sigma$ Modulators Verification. Different computer aided techniques have been developed recently for the stability verification of $\Delta\Sigma$ modulators. In [5], the authors used an optimization technique in order to find bounds of the reachable states, hence checking for stability. The essence of the method is to reformulate bounded time reachability analysis to be solved by mixed-integer linear programming [3]. The verification idea is to compute a set of worst trajectories which safety implies the safety of all other trajectories. The main disadvantage of this method is that the translation of the system equations into an optimization problem is not generic and depends on the specific design. In our case, a canonical form is used to represent the system equations. In [8], the authors used the tool *Checkmate* to verify the stability of a $\Delta\Sigma$ modulator, by simulating the system over polyhedra models, instead of intervals as in our case. The induction based symbolic verification approach proposed in [1] was applied on the stability verification of

1-4244-1176-9/07/$25.00 ©2007 IEEE.

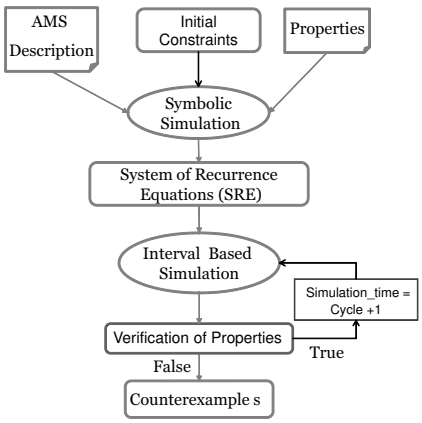

Fig. 1. Verification Methodology

$\Delta\Sigma$ modulators. While the results are time independent, the method requires the addition of design constraints which can be sometimes difficult to identify.

Another symbolic method for stability checking is based on invariant verification [12]. In [7], the authors proposed using iterative simulation to construct the invariant regions, which if constructed successfully, provides a stability proof of the modulator. The main drawback of this method is that stability violation cannot be detected as failure to construct such region does not provide a proof of instability.

III. Methodology

The proposed verification methodology is shown in Figure 1. Given an AMS described using standard recurrence equations and the properties as algebraic constraints between signals, a symbolic simulator performs a set of transformations by rewriting rules in order to obtain a normal mathematical representation called System of Recurrence Equations (SRE). These are combined recurrence relations that describe directly each property constraint in terms of the behavior equations of the system. The verification is then applied using interval analysis [10] over the normal structure of the SRE. Interval analysis is used to simulate the set of all input conditions with a given length that drives the discrete-time system from given initial states to a given set of final states satisfying the set of constraints. If for all time steps, the properties constraints are satisfied, then the stability of the modulator is ensured, otherwise, we provide counterexamples for the non-proved properties. The intervals based simulator is implemented inside the computer algebra system *Mathematica* [13], which provides useful functions to simplify and prove algebraic relations.

A. System Modeling

The notion of recurrence equation is extended to describe digital circuits using the normal form (called generalized `If-formula` in [2]). Such formalization was found practical and suitable in modeling AMS systems [1]. In this paper, we use a generalization of recurrence equations as a model for $\Delta\Sigma$ modulators.

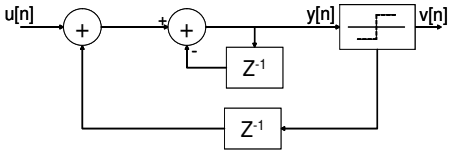

Fig. 2. First-order $\Delta\Sigma$ Modulator

Definition 1: ***Generalized `If-formula`***
In the context of symbolic expressions, the generalized `If-formula` is a class of expressions that extend recurrence equations to describe digital systems. Let $\mathbb{K}$ be a numerical domain ($\mathbb{N},\mathbb{Z},\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{B}$), a generalized `If-formula` is one of the following:

- A variable $X_i(n)$ or a constant $C \in \mathbb{K}$
- Any arithmetic operation $\diamond \in \{+,-,\div,\times\}$ between variables $X_i(n) \in \mathbb{K}$
- A logical formula: any expression constructed using a set of variables $X_i(n) \in \mathbb{B}$ and logical operators: $not, and, or, xor, nor, \ldots$, etc.
- A comparison formula: any expression constructed using a set of $X_i(n) \in \mathbb{K}$ and comparison operator $\alpha \in \{=, \neq, <, \leq, >, \geq\}$.
- An expression $IF(X,Y,Z)$, where X is a logical formula or a comparison formula and Y,Z are any generalized `If-formula`. Here, $IF(x,y,z) : B \times K \times K \longrightarrow K$ satisfies the axioms:
  (1) $IF(True,X,Y) = X$
  (2) $IF(False,X,Y) = Y$

A formalization for a uniform system of recurrence equations was defined as an extension of conventional recurrence equations in [2]:

Definition 2: ***A System of Recurrence Equations (SRE)***
Consider a set of variables $X_i(n) \in \mathbb{K}$, $i \in V = 1,\ldots,k$, $n \in \mathbb{Z}$, 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`. The set ε_i is a finite non-empty subset of $1,\ldots,k \times \mathbb{N}$. The integer γ is called the delay.

Example 1: Figure 2 shows a first-order $\Delta\Sigma$ modulator of one-bit with two quantization levels, $+1$V and -1V. Consider the constraint that the quantizer (input signal $y(n)$) should be between -2V and $+2$V in order to not be overloaded. The SRE of the $\Delta\Sigma$ is then described as:

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

B. Interval Analysis

Intervals are numerical domains that enclose the original states of a system of equations at each discrete step [10]. Such method produces bounded envelopes for the reachable states not only at some discrete time points but also for all continuous

ranges of intermediate states between any two consecutive time discrete points. These methods, also known as *validated methods*, are an attractive tool to use in the verification of the behavior of systems with uncertainty on the design parameters or initial conditions as it allows sound discretization. Basic interval arithmetic operators can be defined as follows: Let I_1 and I_2 be two real intervals (bounded and closed), the basic arithmetic operations on intervals are defined by:

$$I_1 \Phi I_2 = \{r_1 \Phi r_2 | r_1 \in I_1 \wedge r_2 \in I_2\}$$

with $\Phi \in \{+,-,,/\}$ except that I_1/I_2 is not defined if $0 \in I_2$.

$$\begin{aligned}
[a,b]^{\iota} &\triangleq [a+b] \\
[a,b] +^{\iota} [a',b'] &\triangleq [a+a', b+b'] \\
[a,b] -^{\iota} [a',b'] &\triangleq [a-b', b-a'] \\
[a,b] \times^{\iota} [a',b'] &\triangleq [min(aa',ab',ba',bb'), \\
&\quad\ max(aa',ab',ba',bb')] \\
1 \div^{\iota} [a,b] &\triangleq [1 \div b, 1 \div a] if 0 \notin [a,b] \\
[a,b] \div^{\iota} [a',b'] &\triangleq [a,b] \times 1 \div [a',b']
\end{aligned}$$

In addition, other elementary functions can be included as basic interval arithmetic operators. For example, *exp* may be defined as $exp([a,b]) = [exp(a), exp(b)]$. The guarantee that the real solutions for a given function are enclosed by the interval representation is described by the following property.

Property 1: ***Inclusion Function.***[10] Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function, then $F : \mathbb{I}^n \to \mathbb{I}$ is an interval extension (inclusion function) of f if

$$\{f(x_1,\ldots,x_n) | x_1 \in X_1, \ldots, x_n \in X_n\} \subseteq F(X_1,\ldots,X_n)$$

where $\mathbb{I}$ is the interval domain.

As a generalization of the inclusion function, interval analysis provides efficient and safe methods for checking truth values of Boolean propositions over intervals by using the notion of *inclusion test*.

Property 2: ***Inclusion Test.*** Given a constraint $c : \mathbb{R}^n \to \mathbb{B}$, we define $C_{\mathbb{I}} : \mathbb{I}^n \to \mathbb{B}_{\mathbb{I}}$ to be an inclusion test of c, with an interval domain defined with three values set; $\mathbb{B}_{\mathbb{I}} = \{0, 1, [0,1]\}$, where 0 stands for false, 1 for true and $[0,1]$ for indeterminate, iff:

$$\{c(x_1,\ldots,x_n) | x_1 \in X_1, \ldots, x_n \in X_n\} \subseteq C_{\mathbb{I}}(X_1,\ldots,X_n)$$

Inclusion test can be used during the verification algorithm to prove wether the reachable interval states satisfy a given property, or not. We define the inclusion test as follows: $C_{\mathbb{I}}(x_{\mathbb{I}}) = 1 \Rightarrow \forall x \in X, c(x) = 1$ and $C_{\mathbb{I}}(x_{\mathbb{I}}) = 0 \Rightarrow \forall x \in X, c(x) = 0$. For instance, given a set of reachable interval states and a property predicate, we remove the states that do not satisfy

$$1 \in C_{\mathbb{I}}(X_1,\ldots,X_n)$$

Therefore, if $C_{\mathbb{I}}(X_1,\ldots,X_n) = \emptyset$, then we have a guarantee that the property is not satisfied. A set of the main logical rules that define the inclusion test is given as follows:

$$\begin{aligned}
x_{\mathbb{I}} \leq^{\iota} y_{\mathbb{I}} = 1 &\Leftrightarrow b \leq a' \\
x_{\mathbb{I}} \in^{\iota} y_{\mathbb{I}} = 1 &\Leftrightarrow x_{\mathbb{I}} \in y_{\mathbb{I}} \\
&\Leftrightarrow a \geq a' \ and \ b \leq b' \\
x_{\mathbb{I}} \vee^{\iota} y_{\mathbb{I}} &\triangleq \{x \vee y | x \in x_{\mathbb{I}} \ and \ y \in y_{\mathbb{I}}\} \\
x_{\mathbb{I}} \wedge^{\iota} y_{\mathbb{I}} &\triangleq \{x \wedge y | x \in x_{\mathbb{I}} \ and \ y \in y_{\mathbb{I}}\} \\
\neg^{\iota} x_{\mathbb{I}} &\triangleq \{\neg x | x \in x_{\mathbb{I}}\}
\end{aligned}$$

C. Verification Algorithm

A simplified version of the CBV algorithm based on interval analysis is shown below (Algorithm 1):

Algorithm 1 Constraint Based Verification

Require: $\mathbf{x}(0) = Initialize\_States\_Equations$
Require: $\mathbf{x}(k+1) = f(\mathbf{x}(k), u, v)$
Require: $Q(k)$
Require: $R_{simp} = R_{Math} \cup R_{Logic} \cup R_{IF} \cup R_{Eq} \cup R_{Intervals} \cup R_{Int-Logic}$
1: $P(k) = SRE(\mathbf{x}(k), Q(k))$
2: **for all** k: $0 \leq k \leq N_{max}$ **do**
3: $P(k+1) = ReplaceRepeated(P(k), R_{simp})$
4: **if** $P(k+1) == False$ **then**
5: **return** Constraint is unsatified
6: **end if**
7: **end for**

The first step of the algorithm is to obtain the SRE describing the constraint $Q(k)$ in terms of the system equations $\mathbf{x}(k)$ (line 1), where $\mathbf{x}(k+1)$ is described in terms of state variables at previous time step $\mathbf{x}(k)$, the input signals u and the digital control signals v. The constraint based verification is applied for a N_{max} time steps (line 2), using the rewriting function *ReplaceRepeated* (line 3), unless the property constraint is unsatisfied: $P(k+1) == False$ at $k < N_{max}$ (line 4). *ReplaceRepeated* [1] is a symbolic rewriting function defined recursively over the property and a set of rewriting rules R_{simp}. These rules include the same rules defined in [1] and extend them by $R_{Intervals}$ which is a set of rewriting rules that define interval arithmetic expressions simplifications over SRE and $R_{Int-Logic}$ which is a set of rewriting rules that define the simplification of Boolean expressions over intervals.

IV. ΔΣ Modulator Verification

In this section, we apply the proposed verification methodology on a third-order ΔΣ modulator. The design of the ΔΣ modulator in Figure 3 is given using vector recurrence equations $X(k+1) = C\ X(k) + B\ u(k) + A\ v(k)$, where A, B and C are matrices providing the parameters of the circuit, $u(k)$ is the input and $v(k)$ is the digital part of the system. The condition of the threshold of the quantizer is computed to be equal to $c3x3(k) + u(k)$. The digital description of the quantizer is transformed into a recurrence equation. Thus, the equivalent recurrence equation that describes $v(k)$ is $v(k) = IF(c3x3(k) + u \geq 0, -a, a)$. A ΔΣ modulator is said to be stable if the integrator output remains bounded under a bounded input signal. The stability property P of the ΔΣ modulator is written

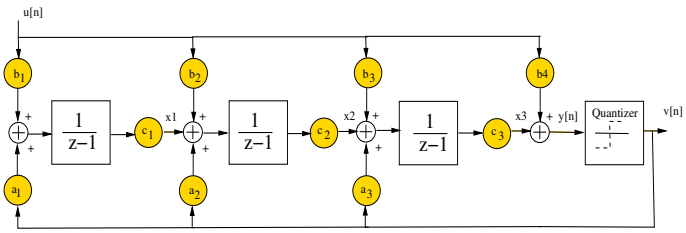

Fig. 3. Third-order ΔΣ Modulator

as: $P(k) = Always(-1 < x3(k) < 1$ for all $k \geq 0$. The modulator is described using the following recurrence equations:

$$\begin{aligned}
x_1(k+1) &= if(c_3x_3(k)+u >= 0, x_1(k)+b_1u-a_1a, \\
&\quad x_1(k)+b_1u+a_1a) \\
x_2(k+1) &= if(c_3x_3(k)+u >= 0, c_1x_1(k)+x_2(k)+b_2u(k) \\
&\quad -a_2a, c_1x_1(k)+x_2(k)+b_2u(k)+a_2a) \\
x_3(k+1) &= if(c_3x_3(k)+u >= 0, c_2x_2(k)+x_3(k)+b_3u(k) \\
&\quad -a_3a, c_2x_2(k)+x_3(k)+b_3u(k)+a_3a)
\end{aligned}$$

Applying the symbolic simulation, we obtain the following expression of the property:

$$\begin{aligned}
P(k+1) &= if(c_3x_3(k)+u >= 0, \\
&\quad c_2x_2(k)+x_3(k)+b_3u(k)-a_3a < 2, \\
&\quad -2 < c_2x_2(k)+x_3(k)+b_3u(k)+a_3a)
\end{aligned}$$

The correctness of the property $P(k+1)$ depends on the parameters A, B and C, the values of variables $x_1(k)$, $x_2(k)$ and $x_3(k)$, the time k, and the input signal $u(k)$. We verify the ΔΣ modulator for the following set of parameters inspired from the analysis in [8]:

$$\begin{cases}
a = 1 & a_1 = 0.044 & a_2 = 0.2881 \\
a_3 = 0.7997 & b_1 = 0.07333 & b_2 = 0.2881 \\
b_3 = 0.7997 & c_1 = c_2 = c_3 = 1
\end{cases}$$

We applied the Algorithm 1 in order to verify the ΔΣ modulator stability for the above set of parameters. Table I shows the verification results for the design given in Figure 3. We use the same circuit parameters set as in [8]. The experiments were performed on an Intel Core2 1900 MHz processor with 2GB of RAM. The initial constraints define the set of test cases over which interval based simulation is applied. If the property is *false*, as in the first and third cases in Table I, then the verification is completed and a counterexample is generated from the simulated intervals. On the contrary, when the property is *True*, we have a partial verification result as it is bounded in terms of simulation steps. The second case in Table I illustrates such limitation.

V. Conclusion

In this paper, we presented a semi-formal methodology for the stability verification of ΔΣ modulators under a set of given initial conditions and input signals. The main advantage of the method is the sound verification it provides over the simulation time. In addition, it generates counterexamples for the failed properties, which can be used for further analysis.

TABLE I

Verification Results

Initial Constraints	**Property Evaluation for $n=0$ to N_{max} Cycles**	**CPU time Used**
$0.028 \leq x_1(0) \leq 0.03$ $-0.03 \leq x_2(0) \leq -0.02$ $0.8 \leq x_3(0) \leq 0.82$, $u := 0.8$	$N_{max} = 40$ $n = 0$ to 15 True $n > 15$ False	1.5 sec $x_1[16] \mapsto 0.263$ $x_2[16] \mapsto 1.256$, $x_3[16] \mapsto 2.42$
$0.012 \leq x_1(0) \leq 0.013$ $0.01 \leq x_2(0) \leq 0.02$ $0.8 \leq x_3(0) \leq 0.82$, $u := 0.54$	$N_{max} = 38$ True	31 sec
$0.163 \leq x_1(0) \leq 0.164$ $-0.022 \leq x_2(0) \leq -0.021$ $0.8 \leq x_3(0) \leq 0.82$, $u := 0.6$	$N_{max} = 40$ $n = 0$ to 17 True $n > 17$ False	0.8 sec $x_1[19] \mapsto 0.163$ $x_2[19] \mapsto 0.886$, $x_3[19] \mapsto 2.47$

The method is still in primary phase and some issues need to be addressed to make it more practical. For instance, the interval based reachability computation is expensive and hence limits the maximum verification time steps as divergence can occur quickly. This is mainly caused by the *wrapping effect* [10] which appears when the results of a computation are overestimated when enclosed into intervals, hence leading to error accumulation at each time step.

To tackle some of the limitation mentioned, we plan to use a variant of interval analysis that can reduce the divergence problem. In addition, we plan to integrate this methodology with the induction based verification developed in [1].

References

[1] G. Al Sammane, M. Zaki, and S. Tahar: A Symbolic Methodology for the Verification of Analog and Mixed Signal Designs. In Proc. of IEEE/ACM Design, Automation and Test in Europe, April 2007.

[2] G. Al-Sammane. Simulation Symbolique des Circuits Decrits au Niveau Algorithmique. PhD thesis, Université Joseph Fourier, Grenoble, France, July 2005.

[3] A. Bemporad and M. Morari: Verification of Hybrid Systems via Mathematical Programming. In Hybrid Systems: Computation and Control, LNCS 1569, pp. 31-45, Springer, 1999.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled. Model Checking. MIT Press, 2000.

[5] T. Dang, A. Donze, O.Maler: Verification of Analog and Mixed-Signal Circuits using Hybrid System Techniques. In Formal Methods in Computer-Aided Design, LNCS 3312, pp. 14-17, Springer, 2004.

[6] G. Frehse, B. H. Krogh, R. A. Rutenbar. Verifying Analog Oscillator Circuits Using Forward/Backward Abstraction Refinement. In Proc. of IEEE/ACM Design, Automation and Test in Europe, pp. 257-262, 2006.

[7] M. Goodson R. Schreier and B. Zhang. An Algorithm for Computing Convex Positively Invariant Sets for Delta-Sigma Modulators. IEEE Transactions on Circuits and Systems I 44:38-44, January 1997.

[8] S. Gupta, B.H. Krogh, R.A. Rutenbar: Towards Formal Verification of Analog Designs, In Proc. of IEEE/ACM International Conference on Computer Aided Design, pp. 210-217, 2004.

[9] W. Hartong, *et. al*. Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking. Advanced Formal Verification, Kluwer: pp. 205-245, 2004, .

[10] R. E. Moore, Methods and Applications of Interval Analysis, Society for Industrial and Applied Mathematics, 1979.

[11] R. Schreier and G. C. Temes. Understanding Delta-Sigma Data Converters, Wiley-IEEE Press, 2005.

[12] P. Steiner and W. Yang. Stability Analysis of the Second Order Sigma–Delta Modulator. In Proc. of IEEE International Symposium on Circuits and Systems, pp. 365-368., 1994,

[13] S.Wolfram. Mathematica: A System for Doing Mathematics by Computer. Addison Wesley Longman Publishing, USA, 1991.

[14] J. Yuan, C. Pixley, A. Aziz. Constraint-Based Verification, Springer, 2006.