Enabling AMS Simulation using Recurrence Notations

Naeem Abbasi, Rajeev Narayanan, Ghiath Al Sammane, Mohamed H. Zaki and Sofiène Tahar
Hardware Verification Group,
Concordia University, Montreal, Quebec, Canada

May, 2008

1 Abstract

System of Recurrence Equations (SRE) have been shown to accurately model Analog and Mixed Signal (AMS) circuits and systems which link digital systems to the analog world. In this paper we present an efficient SRE simulation algorithm and its prototype implementation. A comparison of speed of simulation of several benchmark circuits shows that modeling the AMS design using SRE and simulation using the proposed simulation algorithm can significantly speed up the simulation with out loosing accuracy of simulation results as compared to traditional HDL AMS simulators.

Contents

2 Introduction

Traditionally, simulation was used where the evaluation of the results is often done manually in an informal fashion and the search of the state space is not complete. In general, simulation requires large memory and CPU resources as both the digital and the analog parts of the design are translated to very detailed level of abstraction. Simulation based techniques were then complemented by symbolic techniques where the effect of parameters variations on the system behavior is analyzed. For a simulator, tradeoff exists between accuracy of the results and the simulation speed. In selecting a level of abstraction the verification engineer should consider the circuit and the system property being verified to obtain accurate results and thus can lead to smaller simulation times without sacrificing the accuracy. One such modeling technique is the use of system of recurrence equations (SRE). The behavior of a complete AMS system can be described using a system of recurrence equations (SRE) and then simulated using one simulator. It has the additional advantage that symbolic simulation steps can be easily incorporated to improve the overall efficiency of the simulation.

In this paper we present an algorithm (C-SRE Simulator) for the simulation of AMS SRE models. We present the simulation cycle, and describe an implementation of the simulation algorithm. We investigated and compared various AMS HDL languages for describing the behavior of a system of recurrence equations in terms of accuracy and speed of simulation of the AMS design. We show the superiority of the proposed simulator over traditional AMS HDL simulators in terms of simulation speed without compromising on accuracy.

The rest of the report is organized as follows: We discuss about current simulators, methodology/framework for AMS simulation in 3 section. In Section 4 we give an overview of the proposed C-SRE simulation framework, followed by symbolic simulation and SRE in Section 5. In Section 6 we describe the C-SRE simulation cycle and implementation. We conclude the paper with experimental results in Section 7 and a comparison of simulation run-times in Section 7.3, followed by conclusions in Section 8.

3 Related Work

During the past few decades, several work in the Computer-aided design (CAD) literature were concerned with studying possible frameworks/methodology for the simulation of mixed signal designs. For instance, *FASTSPICE* [24] simulator is a best choice if the design contains circuit blocks which cannot be partitioned easily. Cadence *Virtuoso* Analog Design Environment [25] is capable of accepting description in Verilog-AMS, VHDL-AMS as well as various netlist formats like SPICE and Spectre, or combinations of these languages and formats. On the otherhand, ModelSim [26] is also capable of handling behavioral simulation using VHDL, Verilog and SystemC. However, researchers have tried to develop methodology by combining different simulators. For instance, in [1], the authors proposed a SystemC/Simulink co-simulation framework for embedded system that relies on Simulink for the continuous simulation

and SystemC for the discrete simulation based on one or more synchronization model. While in [2], the authors provides a co-simulation environment based on SPICE and SAVANT. Another mixed-domain simulation framework was proposed in [3] based on VHDL and ELDO. The commercial tool Nexus-PDK [4] supports co-simulation of cycle accurate C/C++ with SystemC, MATLAB/Simulink, and VHDL/Verilog simulators. All the above papers, are not capable of handling the SRE based symbolic simulation efficiently. For mixed-level simulation, a new type of simulator which can should be capable of handling symbolic simulation is needed.

4 The C-SRE Simulator Framework

Figure 1 shows the C-SRE simulator Framework. The AMS description is composed in general of a digital part and an analog part. For the analog part, it could be described using recurrence equations or a set of differential algebraic equations (DAEs) that can be converted into an equivalent set of difference equations. For the digital part, it could be described using a hardware description language (HDL) like VHDL. The AMS description, the circuit topology along with design components are input to a symbolic simulator that performs a set of transformations by rewriting rules in order to obtain a normal mathematical representation called System of Recurrence Equations (SRE to be described in Section 5.1). The symbolic simulator are implemented inside the computer algebra system *Mathematica*. These are combined recurrence relations describing the behavior of AMS system in terms of the continuous time (CT), the discrete time (DT), and the discrete event (DE) SREs. The combined recurrence equation along with signal source are applied as an input to the C-SRE Scheduler/Solver. The scheduler/solver makes sure that the CT, DT, or DE SREs are executed at an appropriate instant of time to simulate the correct transient behavior of the circuit. The simulation output of the solver are then plotted using MAPLE script generator.

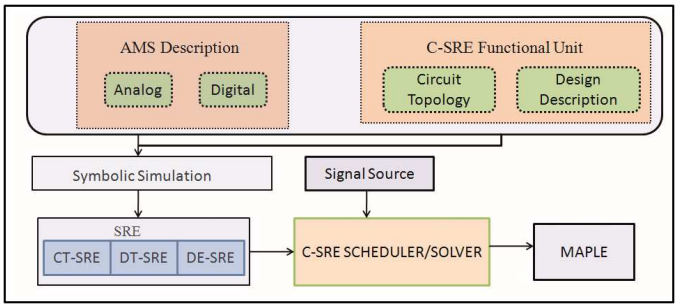


Figure 1: C-SRE Simulation Framework

5 C-SRE based Symbolic Simulation

A mixed signal system consists of three parts; a continuous-time part, a discrete-time part, and a discrete-event part. The three parts of the system can be modeled using SRE's at various levels of abstractions. Recurrence equations at behavioral level of abstraction are of interest to us because of their simplicity of expression and speed of simulation. For discrete-event part and continuous-time part of the design the time step is not *uniform* and is not known in advance. For the discrete-time part of the design time steps are uniform and are known in advance. The time step is chosen to find a good compromise between the accuracy (to keep Local Truncation Error(LTE) below acceptable level) [8] and the speed of the simulation.

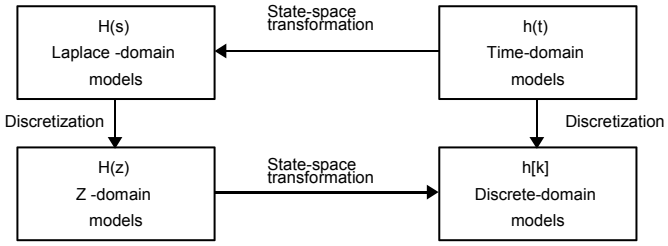


Figure 2: Discretization of differential algebraic equations

The simulator clock provides the reference in the event based framework where the SRE describing the behavior of a mixed signal system can interact with each other by scheduling events. The interactions between various recurrence equation processes are managed by the simulator using a single event queue. The event queue keeps track of events based on which part of the design do they belong to, and at what time it is to be triggered. The events are sorted and then scheduled in earliest to latest order and with the restriction that T_{CT} is always the smallest time step in the simulation.

Figure 2 shows two possible ways a continuous time domain model can be discretized [9]. First involves numerical integration or discretization in time domain and the other involves transformation to frequency domain, followed by s to z transformation and then transformation back to discrete time domain.

5.1 The System of Recurrence Equations: SRE

A recurrence equation or a difference equation is the discrete version of an analog differential equation. A recurrence equation defines a relation between consecutive elements of a sequence. In [10], the notion of recurrence equation is extended to describe digital circuits using the normal form: generalized `If-formula`.

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

- *A variable $X_i(n)$ or a constant C that take values in $\mathbb{K}$*
- *Any arithmetic operation $\diamond \in \{+,-,\div,\times\}$ between variables $X_i(n)$ that take values 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, $IF(x,y,z) : B \times \mathbb{K} \times \mathbb{K} \longrightarrow \mathbb{K}$ satisfies the axioms:*
  (1) $IF(True,X,Y) = X$
  (2) $IF(False,X,Y) = Y$

We define a System of Recurrence Equations as follows:

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: $3^{rd}Order\Delta\Sigma$ modulator

$\Delta\Sigma$ modulators with single-bit quantizers have made possible the construction of robust high-resolution analog-to-digital and digital-to-analog converters. The design of the $\Delta\Sigma$ 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: $v(k) = IF(c3x3(k) + u(k) \geq 0, -a, a)$. The equivalent SRE of the system is then:

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

A $\Delta\Sigma$ modulator is said to be stable if the integrator output remains bounded under a bounded input signal.

$$Always\ -1 < x_3 < 1$$

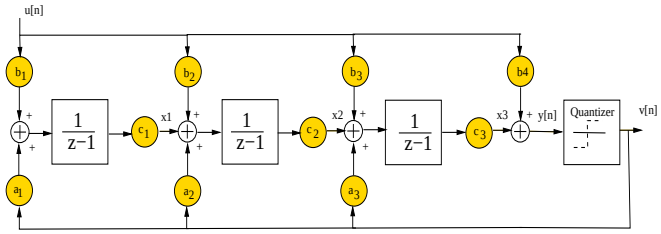


Figure 3: Third Order ΔΣ Modulator

Transforming Ordinary Differential Equations (ODEs) into difference equation is well known in the domain of analog circuits simulation. For the digital part, [10] has defined a method and implemented tool to extract a set of recurrence equations (difference equations) from a digital synchronized VHDL descriptions. We use the same framework to extract the recurrence equation of the digital part. The proposed C-SRE algorithm provides a practical platform for carrying out SRE based Symbolic Simulation. The tool allows one to quickly build custom executable models of AMS circuits describing the behavior of AMS systems using recurrence equations at various levels of abstraction.

The SRE If-formulae operator is extremely useful in describing the non-linear behavior such as saturation, hysteresis and dead-band. For real circuits the signal output generally cannot exceed the supply voltage and the output of the circuit levels off to a value either equal to or a few volts below the supply voltage. This non-linear behavior of the circuit is some times called saturation or saturation non-linearity. The saturation non-linearity can be modeled using a piece-wise linear approximation of the non-linear behavior using the SRE If-formulae construct.

5.2 C-SRE Methodology

The overall methodology involves partitioning the design into continuous-time, discrete-time and discrete-event parts, and then generating recurrence equation as described in Section 5.2.1 and its equivalent dependency graph as described in Section 5.2.2. The procedure described for generating an executable simulation model has eight main steps. Figure 5 shows the methodology steps.

- Partition the design into small manageable parts preferably along the continuoustime, discrete-time, and discrete-event lines as shown in 4.
- Extract Recurrence Equations from each part.
- Simplify Recurrence Equations if needed (this step is optional)
- Code each Recurrence Equation as C function.
- From the initial design description or schematic extract the Dependency Graph

- Arrange the C Functions based on the Dependency Graph (From inputs to outputs)
- Include the above code into the C template for the program. Update stimulus and any other simulator parameters if needed. Compile and Run the executable model. The program generates a skeleton MATLAB script. Modify the script if needed.
- Plot results using the MATLAB script.

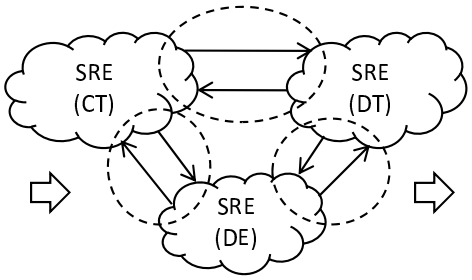


Figure 4: A mixed signal SRE model

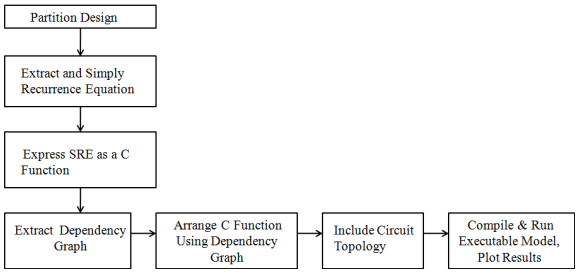


Figure 5: C-SRE Methodology

5.2.1 Generating Recurrence Equation

There are several methods available to transform continuous-time behavior of an analog component into a system of recurrence equations. Some of these methods are based on frequency domain techniques [13] and some are based on time domain techniques. We follow the following steps to derive the recurrence equations.

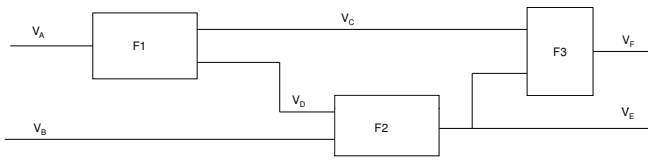
- Write down time domain differential algebraic equations
- Transform the equations into frequency domain using Laplace transform

- Using bilinear transformation [13] find a discrete time approximation of the continuous time transfer function
- Take the inverse Z transform and convert it into a difference equation.

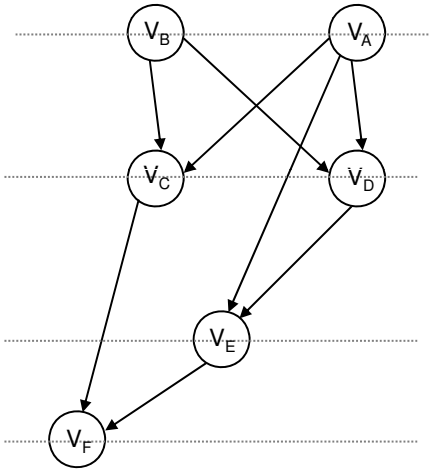
The process involves derivation of transfer function from nodal equations using symbolic rewriting in *Mathematica* [14]. This is followed by S to Z bilinear transformation using *Mathematica* [15] as well and finally conversion from Z transform to difference equations.

5.2.2 Understanding Dependency Graph

The following example shows how the dependency graph is obtained from the description of the circuit such as from the circuit schematic. For the general circuit shown in Figure 6(a) (F1, F2, F3 represents some functional blocks), Figure 6(b) shows the dependency graph.



(a)



(b)

Figure 6: (a)General Circuit Example, (b) Dependency Graph

In these graphs each node corresponds to a node of interest in the circuit and corresponds to the Left-Hand Side (LHS) of the recurrence equation. The recurrence equations in Tables 1 represent the behavior of the each of the components in the respective

Table 1: Recurrence Equations

Recurrence Equation	Logic Function
$V_F = ITE[V_A, 0, V_B]$	and
$V_F = ITE[V_A, NOT[V_B], V_B]$	xor
$V_F = ITE[V_A, 1, NOT[V_B]]$	nand

Recurrence Equation	Comment
$V_X = ITE[V_A, 0, V_B]$	and
$V_H = ITE[V_X, 1, NOT[V_B]]$	xor
$V_G = ITE[V_X, NOT[V_H], V_H]$	nand

circuit and the order in which they should execute. The order of execution can be easily determined from the dependency graphs shown in Figures 6(a) and (b) respectively. A close inspection of the recurrence equation shows that the dependence is actually built into the way behavior of the circuit is described in the recurrence equations. In these examples the order of execution from input to output for recurrence equations is $[V_A, V_B] \rightarrow V_C \rightarrow V_D \rightarrow V_E \rightarrow V_F$. ($F_1$ shall be evaluated first then F_2 and then F_3)

5.2.3 SRE Coding as C Function

The C Function returns the value of the RHS of the recurrence equation to the calling function and is usually assigned to the values of one of the nodes of interest. Recurrence equation in assumed to have the following general form:

$Vnode =$**ITE**(Condition,Statement(TRUE),Statement(FALSE))

Most C function implementations use the If-Then-Else construct.

Table 2 shows a few examples of C functions for basic logic gates.

6 C-SRE Implementation

This section gives an overview about the C-SRE simulation cycle and implementation details.

6.1 C-SRE Simulation Cycle

Assuming that all three parts of the analog and mixed signal system have been transformed into SRE, and that the simulation takes place in an event driven framework. An elaborated SRE design consists of an interconnection of processes, one corresponding to each equation in the AMS design description.

Let T_{CT}, T_{DT}, and T_{DE} be the continuous-time, discrete-time, and discrete-event time steps. Let T_{CT} be the smallest time step required to provide a desired time resolution and accuracy for the numerical solution of the Ordinary Differential Equation (ODE) [8]. T_{CT} and T_{DE} are not fixed and depend on the inputs to and the state of the SRE's of the continuous-time and discrete-event part of the designs respectively.

Table 2: C functions, Pseudo Code

C Functions	Comment

`char ANDGate(char &A, char &B){`
`if(A==1){`
`return(0);`
`} else {`
`return(B);`
`}`
`}`
	AND

`char XORGate(char &A, char &B){`
`if(A==1){`
`return(not(B));`
`} else {`
`return(B);`
`}`
`}`
	XOR

`char NANDGate(char &A, char &B){`
`if(A==1){`
`return(1);`
`} else {`
`return(not(B));`
`}`
`}`
	NAND

`char CircuitVH(char &A, char &B){`
`char VX;`
`VX=ANDGate(&A, &B);`
`return (NANDgate(&VX, &VB));`
`}`
	V_H

`char CircuitVG(char &A, char &B){`
`char VX, VH;`
`VX=ANDGate(&A, &B);`
`VH=CircuitVH(&VX, &VB);`
`return XORGate((&VX, &VB));`
`}`
	V_G

T_{DT} on the other hand is uniformly spaced in time and is known in advance. T_{CT} is computed using SRE, and occurrence of external events at any given time. Its value can be greater, equal or smaller than the T_{DT} and T_{DE} time steps.

In a mixed signal design the three parts of the design may interact with each other. The discrete-time part of the design may only interact at intervals of T_{DT} with either discrete-event or continuous-time parts. The continuous-time and discrete-event parts may interact with each other at any time. As long as T_{CT} is forced to be smaller than T_{DT} and T_{DE}, the simulator will never miss an interaction between the discrete-event and continuous-time parts of the design [12]. Figure 7 shows the SRE simulation

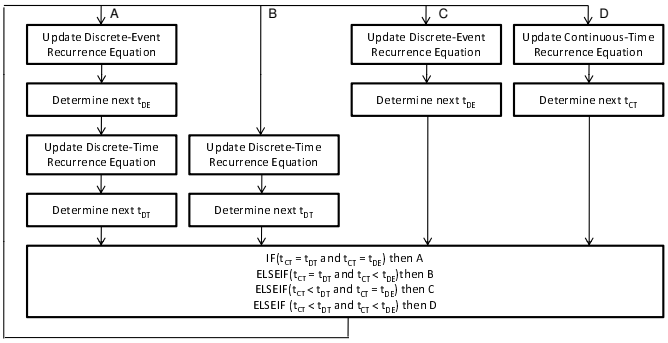


Figure 7: Recurrence Equation simulation flow

cycle. The main steps in the simulation are summarized below:

1. Initialize recurrence equations.
2. Determine initial analog solution at t_{CT}. Determine t_{CT}, t_{DT}, and t_{DE}. Where $t_{DE} = t_{DE} + T_{DE}$, $t_{DT} = t_{DT} + T_{DT}$, $t_{CT} = t_{CT} + T_{CT}$, and $T_{current} = MIN(t_{DE}, t_{DT}, t_{CT})$. T_{CT} should always be the smallest time step in the simulation.

   Use the following four rules to determine the order in which the simulation cycle should proceed.

   - **IF**$[(t_{CT} = t_{DT})$**AND**$(t_{CT} = t_{DE})]$ then update DE and then DT recurrence equations
   - **IF**$[(t_{CT} = t_{DT})$**AND**$(t_{CT} < t_{DE})]$ then update DT recurrence equations
   - **IF**$[(t_{CT} < t_{DT})$**AND**$(t_{CT} = t_{DE})]$ then update DE recurrence equations
   - **IF**$[(t_{CT} < t_{DT})$**AND**$(t_{CT} < t_{DE})]$ then update CT recurrence equations
3. The simulation proceeds in simulation cycles until end of simulation is reached.

6.2 C-SRE Simulation Algorithm

C-SRE is a prototype implemented in C langauge that validates the simulation algorithm of AMS system using the notion of SRE and explicate trace enumeration.

- Design Components: Each design component is an SRE model describing its function. The blocks has several inputs and produces one output. The assumption is that the SRE is an expression of the form:
  $V = \mathbf{ITE}(Condition, Statement(TRUE), Statement(FALSE))$.
  Three parameters passed to the **ITE** function. If the condition evaluates to TRUE, Statement(TRUE) is executed, while Statement(FALSE) is executed in case the condition is FALSE. A very basic library of components has been created. A more extensive library of components will make the task of design and verification easier.

- Signal Source: Various basic signal sources can be created in a similar way as the design components.

The SRE Scheduler/Solver has three main blocks: Initialization block, TimeSync block, and the ExecuteRecurrenceEquations block.

- The Initialization block initializes all the internal VCircuit data structure. Simulation results are stored in this data structure.

- The TimeSync block and the ExecuteRecurrenceEquation block together implement the scheduling algorithms and make sure that various functions are called and recurrence equations executed in the correct sequence and order.

In addition to the design components and signal sources or stimuli the user must provide the three time step values, TCT, TDT, and TDE. The scheduling algorithm takes as input, TCT, TDT, and TDE and determines the instances of execution for a properly synchronized simulation.

The algorithm output or the transient simulation results are stored in a two dimensional array. The results are written to a MATLAB script file through the MATLAB Script generator block . This can be used to plots the simulation results.

The main program flow is shown in Figure 8. The simulations starts with initialization. The simulation then proceeds guided by the algorithm described in subsection 6.1. The simulation terminates when the current simulation time either exceeds or becomes equal to the maximum simulation time provided by the user. The simulation progresses in discrete time steps and is snapshot of the state of the system at a particular instant of time. Figure 9 shows the time instants at which continuous-time, discrete-time, and discrete-event SRE have to be evaluated. The figure also shows how the scheduling algorithm makes sure that the simulation of the system is synchronized. Execution of recurrence equations is in an order such that correct behavior is simulated. The numbers show the sequence of operation. The square, the triangle, and the circle represents the instants of time at which an SRE must be executed. The simulation progresses in discrete time steps and is a snapshot of the state of the system at a particular instant of time. Consider a block diagram of a mixed signal system described as an interconnection of SRE. In a given system there may be more than one set of SREs for each type. Lets further assume that each set of recurrence equations belong to either continuous-time, discrete-time, or discrete-event part of the design and that the level of abstraction has already been appropriately chosen for each part.

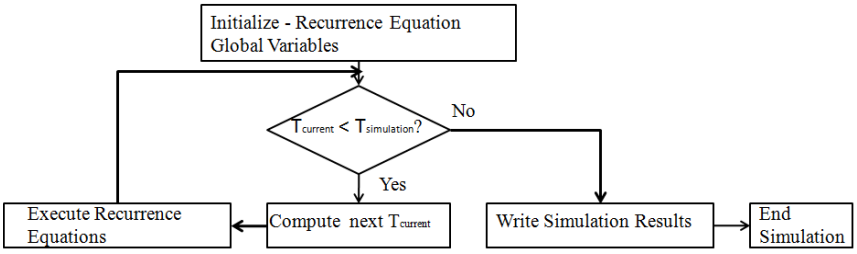


Figure 8: Simulation Algorithm Flow

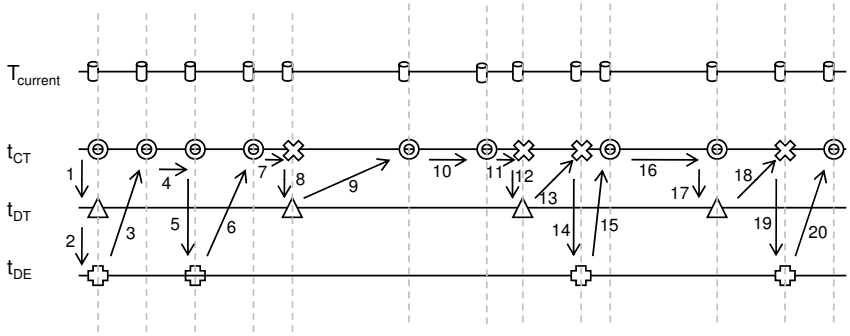


Figure 9: Timing diagram

There are two main challenges in simulating the behavior of an AMS system described as an interconnection of a number of sets of systems of recurrence equations. They are to:

- Determine the correct sequence of execution of each set of recurrence equation with reference to a real time reference. This sequence of execution is also referred to as the schedule. The algorithm described in section 4 correctly sequences the execution of sets of recurrence equations in time.
- Determine the correct order of execution within each set of recurrence equations such that the correct behavior of the circuit is modeled and simulated. The recurrence equations have a built in dependency mechanism. Correct extraction of recurrence equations guarantees that the recurrence equations shall execute in the correct order to simulate the behavior of the circuit.

The sequence of execution refers to when during simulation, a time step takes place between the execution of two sets of recurrence equations. The order of execution on the other hand refers to when no time step takes place between the execution of two or more recurrence equations. The C-SRE implementation solves these two challenges and correctly orders and sequences the execution of SREs.

7 Application and Simulation results

7.1 Example 1:Pulse Width Modulator Circuit

A pulse width modulator circuit is shown in Figure 10. It consists of an analog filter, a first order switch capacitor filter and an analog comparator a digital logic gate and a flip-flop. The circuit is modeled at circuit level and simulated using HSPICE. the design is also modeled using generalized SRE and simulated using CSRE simulator.

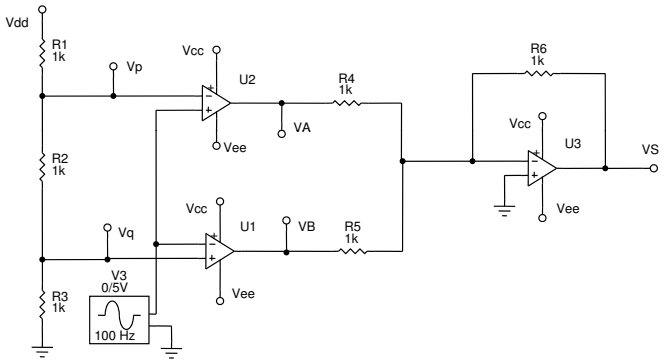
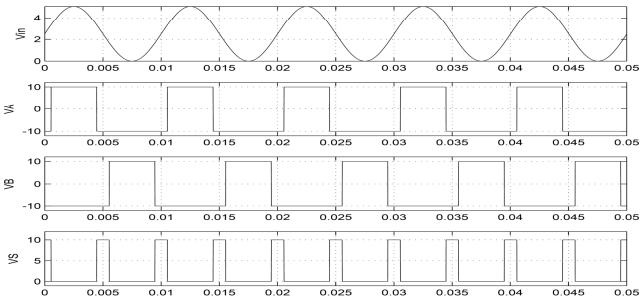
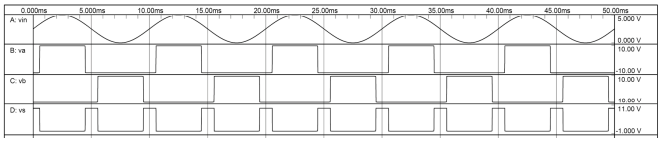


Figure 10: The circuit diagram of a simple PWM circuit



(a)



(b)

Figure 11: Simulation Results: (a) CSRE. (b) SPICE.

The transient analysis simulation results of the mixed signal circuit generated by this C-SRE and SPICE are shown in Figures 7.1 (a) and (b) respectively and the two simulation results are virtually identical. We see the accuracy of the proposed simulator in comparison to SPICE. In the next section we compare simulation speed of C-SRE with other AMS simulators.

7.2 Example 2: Window Comparator Circuit

In this circuit an analog input signal is compared against two fixed dc reference values (low and high threshold) values . The circuit consists of a voltage divider circuit used to generate the two reference voltages. The circuit has three mutually exclusive binary outputs which indicate wether the input signal is below, above or between the lower and upper threshold values. The operational amplifiers U1 and U2 compare input signal Vin to the two threshold reference values and generate the Vhigh and Vlow outputs. The Vhigh and Vlow are added together using the analog adder to generate the Vmid value. WE simulated this circuit using HSPICE and C-SRE simulators . The circuit model uses ideal operational amplifier models and simulated using HSPICE. The CSRE model describes the behavior of the circuti using generalized IF Formula. Figures 13, and 13 show that the SPICE and CSRE simulation results which are virtually identical.

7.3 Experimental Results

We model several benchmark circuits [16] using SREs and describe the behavior in several AMS HDL languages. We compare the ease of modeling the behavior and

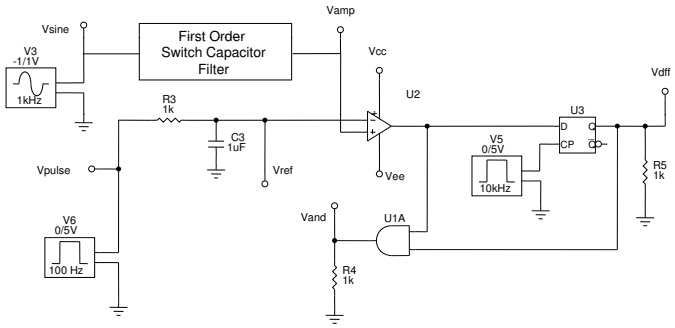


Figure 12: The circuit diagram of a simple mixed signal circuit

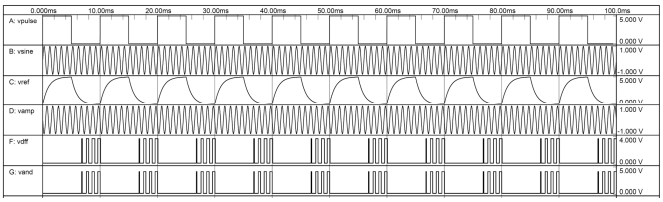


Figure 13: SPICE simulation results

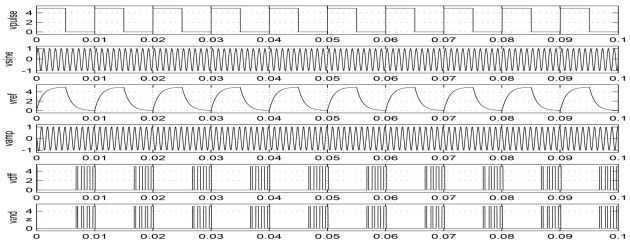


Figure 14: CSRE simulation results

speed of simulation. sinusoidal inputs of general critical frequencies for circuits are used as inputs to simulate the designs. We provide here some experimental results to measure the performance of the proposed C-SRE algorithm using benchmark circuits. As mentioned above, the algorithm is sensitive to the dynamic changes in the values of continuous signals. Thus, we provide several experiments to observe the fluctuation of the performance regarding the frequency of the simulated circuits and the different configurations of the simulation parameters.
The selected circuits are small to medium sized analog, switch capacitor, and mixed-signal circuits. The circuits simulated are:

1. Low Pass Active Filter (LPAF)
2. First Order Switch Capacitor Filter (FOSCF)
3. Continuous-Time State space Filter (CTSF)
4. Leap Frog Filter (LFF)
5. Phase Lock Loop (PLL)

The circuit diagrams are given in Figure 15, 16, 17, and 18. These are low to medium complexity analog and mixed-signal circuits, containing both active and passive, and linear and non-linear components. Circuits are described using HSPICE [17], VHDL-AMS [18], VERILOG-AMS [19], SystemC-AMS [20] and C [21]. Circuits are modeled in HSPICE at the component level for reference purpose. The behavior of the circuits is described both at component level and using SRE in the three AMS HDLs. The SRE model is used to describe behavior in C and simulated using the C-SRE simulation prototype. For all the filter circuits the small signal behavior can be described with just difference equations.

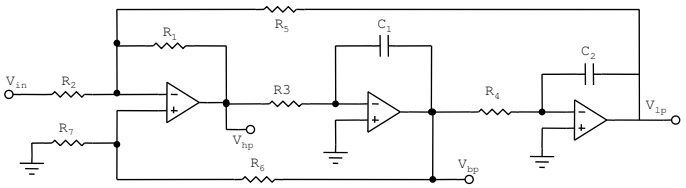


Figure 15: Continuous-Time State Filter (CTSF)

The SRE model of the circuits are verified by comparing the SRE Transient Simulation results with those obtained from HSPICE. Simulation of circuits is run for a fixed duration of time (T_{Sim}). In each simulation cycle, the simulation progresses by one real time unit. The size of this unit is fixed in SRE models. Where as the simulation step size in case of circuit models is determined by the simulator during the simulation based on the smallest time constant of the circuit, input signal rate of change of amplitude or frequency, and the time resolution and accuracy requirements provided to the simulator.

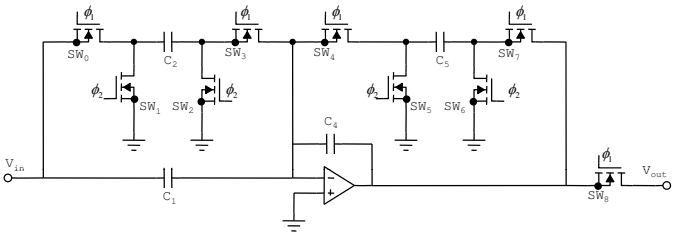


Figure 16: First Order Switch Capacitor Filter (FOSCF)

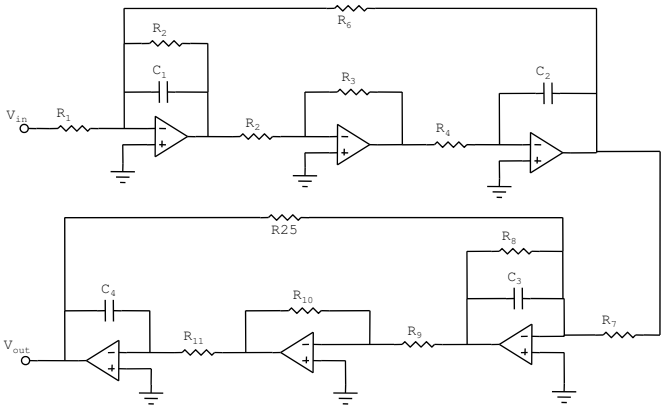


Figure 17: Leap Frog Filter (LFF)

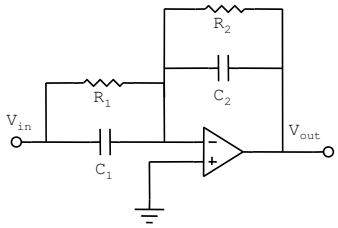


Figure 18: Low Pass Active Filter (LPAF)

Circuit	Frequency (Hertz)	Verilog-AMS (Seconds)	VHDL-AMS (Seconds)	SystemC-AMS (Seconds)	CSRE (Seconds)
Low Pass	1k	0.43	2.34	0.06	0.01
Active	2k	0.41	3.48	0.06	0.01
Filter	4k	0.59	3.53	0.03	0.01
(LPAF)	40k	2.05	4.37	0.07	0.01
First Order	500	0.40	1.19	0.04	0.01
Switch Cap.	1k	0.39	2.95	0.04	0.01
Filter	2k	0.34	2.82	0.04	0.01
(FOSCF)	4k	0.62	2.92	0.05	0.01
Continuous	100	0.52	3.23	0.06	0.01
Time State	795	0.54	3.52	0.05	0.01
Space	1k	0.49	3.96	0.04	0.01
Filter	10k	1.33	3.73	0.09	0.01
(CTSF)	40k	2.67	4.83	0.05	0.01
Leap	1k	0.49	3.36	0.05	0.01
Frog	1.4k	0.53	3.41	0.05	0.01
Filter	10k	1.13	5.13	0.05	0.01
(LFF)	100k	5.27	8.75	0.04	0.01
PLL Mixed-Signal (SRE)	1MHz	0.31	0.09	0.06	0.08

Table 3: SRE Simulation Run-Times (1000 simulation cycles)

SPICE [22] is used for circuit level simulations. Mentor Graphics AdvanceMS [23] tools are used for VHDL-AMS and Verilog-AMs simulations. For SystemC-AMS most recent libraries from [20] are used. The simulation is run for 10ms duration.

Table 3 shows simulation run-times for four circuits for various frequency sinusoidal inputs. In these simulatins the SRE models are used for all circuits and the simulation is run for exactly 1000 simulation cycles. The sampling Period T_{smp} and the total simulation time T_{sim} used in the above simulations is 15.625μs and 15.625ms respectively which corresponds to exactly 1000 simulation cycles. As can be seen from the table that C-SRE simulation run-times are an order of magnitude faster than those obtained from description of SRE models in the three HDLs.

For higher frequency inputs the simulation run time is slightly higher than for low frequency inputs. This is because when the input signal changes at a faster rate (higher frequency) the analog solver requires more iterations to converge to an analog solution point for a given accuracy requirements and hence results in a slight increase in simulation time. This is seen for each circuit described in the three HDLs as one looks at the simulation run-time numbers starting from low frequency values to high frequency values.

Table4 shows simulation run-times obtained from circuit level simulations. In these simulations the the sampling period T_{smp} for First Order Switch Capacitor Filer is 15.625μs. The sampling period T_{smp} for the remaining circuits is 10ns. Total number

Circuit	Frequency (Hertz)	Verilog-AMS (Seconds)	VHDL-AMS (Seconds)	SystemC-AMS (Seconds)	HSPICE (Seconds)	CSRE (Seconds)
Low	1k	0.12	0.13	48.24	48.72	1.10
Pass	2k	0.21	0.17	48.45	48.73	1.13
Active	4k	0.26	0.26	48.16	48.74	1.10
Filter	40k	1.32	0.96	48.20	48.75	1.02
First	500	21.04	6.72	70.28	184.34	1.06
Order	1k	21.94	6.84	70.27	185.65	1.09
Switch	2k	19.98	6.97	70.39	186.13	1.09
Capacitor	4k	18.77	7.06	70.40	185.38	1.12
Continuous	100	0.09	0.07	49.20	57.24	1.24
Time	795	0.07	0.07	48.26	56.61	1.34
State	1k	0.13	0.1	49.07	56.62	1.26
Filter	10k	0.50	0.38	49.71	56.61	1.28
	40k	1.95	1.34	49.55	56.66	1.28
Leap	1k	0.22	0.09	50.26	66.85	1.21
Frog	1.4k	0.15	0.12	50.56	66.89	1.14
Filter	10k	0.82	0.52	50.66	66.70	1.20
	100k	6.92	4.99	51.27	66.73	1.15

Table 4: Circuit Simulation Run-Times

of simulation cycles is equal to 1000,000. The simulation duration is 10 ms.

The seventh column in this table contains simulation run-times obtained by simulating SRE models using CSRE rudimentary simulation tool. The simulation run-time for CSRE is more or less constant because the number of number of operation performed per simulation cycle are constant.

The circuit simulation times of the first order switch capacitor filter are larger because of the non-linear switches in the filter circuit which cause the simulator to iterate more often at the instants of time when the switches change states from ON to OFF or vice versa. Since the switches are turned ON and OFF a fixed number of times in a 10ms simulation the simulation run-time is independent of the input signal frequency but rather depends on the clock signal frequency used for controlling the switches. That clock frequency is fixed and hence the number of ON/OFF instances for each switch and consequently the simulation run-times.

For the three HDL languages, and for the three analog filter circuits, the simulation run-times increase as the input signal frequency increases. This again is due to the fact that the simulator requires more iteration for each analog solution point if the input signal changes faster as compared to a slowly varying signal for a given time resolution and accuracy requirements.

The SPICE and SystemC-AMS run times are comparable for the three filter circuits and are higher than VHDL-AMS and VERILOG-AMS. C-SRE simulation results are shown in the last column for comparison.

Table5 shows the simulation run-times for SRE models. The simulation duration is 10ms. The sampling period T_{smp} for First Order Switch Capacitor Filer is 15.625μs and for the the rest of the circuits it is 10ns. The simulation is run for exactly 1000,000 sim-

Circuit	Frequency (Hertz)	Verilog-AMS (Seconds)	VHDL-AMS (Seconds)	SystemC-AMS (Seconds)	CSRE (Seconds)
Low	1k	272.74	244.67	30.62	1.10
Pass	2k	280.34	247.01	30.61	1.13
Active	4k	303.27	248.73	30.43	1.10
Filter	40k	315.47	250.00	30.30	1.02
Continuous	100	347.38	286.22	51.21	1.24
Time	795	383.27	354.31	50.27	1.34
State	1k	392.64	430.57	50.62	1.26
Filter	10k	425.05	347.17	50.75	1.28
	40k	421.85	326.32	49.42	1.28
Leap	1k	308.51	257.81	33.99	1.21
Frog	1.4k	311.83	256.64	34.15	1.14
Filter	10k	322.83	260.12	34.09	1.20
	100k	322.02	260.76	34.13	1.15
Phase Locked Loop	f_o=1MHz	120.19	38.25	28.70	2.52

Table 5: SRE Simulation Run-Times for 1000,000 simulation cycles

ulation cycles. In these simulations specific language constructs from VHDL-AMS and VERILOG-AMS are used for modeling the circuit behavior as SREs. For SystemC-AMS the circuit behavior is described using SREs embedded in design units sensitive to an external sampling clock. Last column contains CSRE simulation results from comparison. CSRE simulation runt-time is one to two orders of magnitude faster than SRE models described in the three HDLs.

The description of behavior in terms of SRE in VHDL-AMS and VERILOG-AMS takes far fewer lines of code and is more concise and easy to read and understand but simulates slower. Both VHDL, and VERILOG to a lesser extent provides use of reals and integer data types for modeling of behavior at higher levels of abstraction. The circuit designer thus has an alternative to describe the SRE behavior in another way using real and integer data types rather than terminals and quantities, and attributes of quantities such as ‘delayed(Tsmp) and ‘zoh(Tsmp) in VHDL-AMS or using transfer function related functions in VERILOG-AMS. The design are described in SystemC-AMS and CSRE by encapsulating the SRE behavior in a design unit sensitive to an external sampling clock signal T_{smp}.

In table 6 all circuits are described using SRE models. In all HDL languages and in C implementations the SRE behavior is described in a design unit sensitive to an external sampling clock signal.

The design description is much longer compared to using AMS specific language constructs in VHDL-AMS and VERILOG-AMS. The accuracy of simulation results is the same. Introduction of external sampling clock required more time to plan, describe and potentially debug the SRE simulation model. But it simulates faster.

Circuit	Frequency (Hertz)	Verilog-AMS (Seconds)	VHDL-AMS (Seconds)	SystemC-AMS (Seconds)	CSRE (Seconds)
Low	1k	31.18	19.31	30.62	1.10
Pass	2k	31.18	19.24	30.61	1.13
Active	4k	31.57	19.05	30.43	1.10
Filter	40k	31.15	18.94	30.30	1.02
Continuous	100	32.40	21.57	51.21	1.24
Time	795	31.90	22.62	50.27	1.34
State	1k	32.23	21.83	50.62	1.26
Filter	10k	32.69	21.88	50.75	1.28
	40k	32.21	21.75	49.42	1.28
Leap	1k	31.25	18.54	33.99	1.21
Frog	1.4k	31.65	18.76	34.15	1.14
Filter	10k	31.70	18.57	34.09	1.20
	100k	31.48	18.43	34.13	1.15
Phase Locked Loop	f_o=1MHz	120.19	38.25	28.70	2.52

Table 6: SRE Simulation Run-Times for 1000,000 simulation cycles, Behavior of each circuit is embedded in a module or design unit which is sensitive to an external sampling clock signalT_{smp}

The trade-off here is between less coding and debug time when using AMS specific language constructs to less simulation run-time when embedding the behavior in design units sensitive to external sampling clock.

In all these simulations the simulation run-times are more or less comparable for each circuit. The CSRE simulation run-times are 30 to over 200 times faster.

Tables 7 presents a simulation run-time comparison of the two SRE behavioral models of various circuits described in VHDL-AMS. Model # 1 used AMS language specific constructs for describing the behavior of the circuits. Model # 2 embeds the SRE behavior in a design unit sensitive to an external sampling clock. Simulation run-times for Model # 2 are an order of magnitude faster. The last column provides CSRE simulation results for comparison.

Tables 8 presents a simulation run-time comparison of the two SRE behavioral models of various circuits described in VERILOG-AMS. Model # 1 used AMS language specific constructs for describing the behavior of the circuits. Model # 2 embeds the SRE behavior in a design unit sensitive to an external sampling clock. Simulation run-times for Model # 2 are an order of magnitude faster. The last column provides CSRE simulation results for comparison.
We observe the following summarizes the above results:

- The simulation of analog and mixed signal circuits is both memory and CPU intensive.
- The simulation speed depends on the complexity of circuit, length of simulation,

Circuit	Frequency (Hertz)	VHDL-AMS (Model # 1) (Seconds)	VHDL-AMS (Model # 2) (Seconds)	C-SRE C-SRE (Seconds)
Low Pass	1k	244.67	19.31	1.10
Active	2k	247.01	19.24	1.13
Filter	4k	248.73	19.05	1.10
(LPAF)	40k	250.00	18.94	1.02
Continuous	100	286.22	21.57	1.24
Time State	795	354.31	22.62	1.34
Space	1k	430.57	21.83	1.26
Filter	10k	347.17	21.88	1.28
(CTSF)	40k	326.32	21.75	1.28
Leap	1k	257.81	18.54	1.21
Frog	1.4k	256.64	18.76	1.14
Filter	10k	260.12	18.57	1.20
(LFF)	100k	260.76	18.43	1.15
PLL Mixed-Signal (SRE)	1MHz	¿1000	38.25	2.52

Table 7: Simulation run-rimes for the two SRE behavioral models described in VHDL-AMS(1000,000 simulation cycles). Model # 1 uses AMS specific VHDL-AMS language constructs to model the SRE behavior. Model # 2 used only VHDL language constructs.

Circuit	Frequency (Hertz)	VERILOG-AMS (Model # 1) (Seconds)	VERILOG-AMS (Model # 2) (Seconds)	C-SRE C-SRE (Seconds)
Low Pass	1k	272.74	31.18	1.10
Active	2k	280.34	31.18	1.13
Filter	4k	303.27	31.57	1.10
(LPAF)	40k	315.47	31.15	1.02
Continuous	100	347.38	32.40	1.24
Time State	795	383.27	31.90	1.34
Space	1k	392.64	32.23	1.26
Filter	10k	425.05	32.69	1.28
(CTSF)	40k	421.85	32.21	1.28
Leap	1k	308.51	31.25	1.21
Frog	1.4k	311.83	31.65	1.14
Filter	10k	322.83	31.70	1.20
(LFF)	100k	332.02	31.48	1.15
PLL Mixed-Signal (SRE)	1MHz	¿1000	120.19	2.52

Table 8: Simulation run-times for the two SRE behavioral models in VERILOG-AMS(1000,000 simulation cycles). Model # 1 uses AMS specific VERILOG-AMS language constructs to model the SRE behavior. Model # 2 used only VERILOG language constructs except for in VCO module where cosine function was needed.

shape and frequency of the input signals.

- A faster change in input in general results in an increase in simulation time.
- Non linear components in a design usually take more time to simulate.

8 Conclusion

The SRE model provides a mathematical means to computes the symbolic trace for the combined system (design + assertions). The SRE If-formulae operator is extremely useful in describing the non-linear behavior such as saturation, hysteresis and dead-band. For real circuits the signal output generally cannot exceed the supply voltage and the output of the circuit levels off to a value either equal to or a few volts below the supply voltage. This non-linear behavior of the circuit is some times called saturation or saturation non-linearity. The saturation non-linearity can be modeled using a piece-wise linear approximation of the non-linear behavior using the SRE If-formulae construct.

In this paper we presented an SRE simulation algorithm and its prototype implementation for simulating AMS circuits and systems. This simple tool allows one to very quickly build custom executable models of AMS circuits describing behavior of AMS systems using recurrence equations at various levels of abstraction. The ability to describe various design components at different levels of abstraction makes it possible for the verification engineer to describe important components of the design at an appropriate level of abstraction thus guaranteeing accuracy of simulation results. Rest of the design components can be modeled at as high a level of abstraction as possible. This attractive feature makes sure that accuracy requirements are never sacrificed (an extremely important concern in analog and mixed signal designs) while at the same time where ever possible advantage of higher level of abstraction is taken to speed up the simulation and thus indirectly allowing verification process to speed up as well. The simple simulation algorithm described in this paper is one reason for the gain in the speed of simulation when compared with other AMS HDL tools.

References

[1] F. Bouchhima, M. Brire1 G. Nicolescu1 M. Abid E. M. Aboulhamid. A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation, In Proc. Behavioral Modeling and Simulation, IEEE, pp. 1-6, 2006.

[2] D.E. Martin, P.A. Wilsey, R.J. Hoekstra, E.R. Keiter, S.A. Hutchinson, T.V. Russo, L.J. Waters. Integrating multiple parallel simulation engines for mixed-technology parallel simulation, In Proc. Simulation Symposium, IEEE, pp. 45-52, 2002.

[3] H. El Tahawy, D. Rodriguez, S. Garcia-Sabiro, J.J. Mayol. VHD_ELDO: A new mixed mode simulation, In Design Automation Conference, IEEE/ACM, pp.546-551, 1993.

[4] Celoxia *Website: http://www.celoxica.com/*

[5] T.E. Bonnerud, B. Hernes, T. Ytterdal. A Mixed-signal Functional Level Simulation Framework based on SystemC for System-on-a-Chip Applications, In Proc.Custom Integrated Circuits,IEEE, pp. 541-544, 2001.

[6] E. Markert, M. Schlegel, G. Herrmann,D. Mller. Subproject A2: Examination of the Applicability of SystemCAMS for the Description of MEMS, Technical Report, TU Chemnitz, Faculty of Electrical Engineering and Information Technology, 2004. [3 Pages]

[7] H. Al-Junaid, T. Kazmierski. HDL Models of Ferromagnetic Core Hysteresis Using Timeless Discretisation of the Magnetic Slope. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 12, pp. 2757-2764, 2006

[8] Jan Ogrodzki. Circuit simulation methods and algorithms.

[9] Jaidip Singh Resve A. Saleh, Brian A. A. Antao. Multilevel and mixed- domain simulation of analog circuits and systems.

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

[11] G. Al Sammane, M. Zaki, and S. Tahar: A Symbolic Methodology for the Verification of Analog and Mixed Signal Designs; Proc. IEEE/ACM Design Automation and Test in Europe, pp. 249-254, 2007.

[12] Martin Padeke Werner Haas Thomas Buerner Herbert Braize Thomas Guetner Alexander Grassman Ulrich, Heinkel. The VHDL reference, A practical guide to computer aided integrated circuit design including VHDL-AMS (simulation and synthesis), (Chapter 6 , VHDL-AMS tutorial), TK7885.7 V44. Wiley, 2000.

[13] Franklin, G.F., J.D. Powell, and M.L. Workman, Digital Control of Dynamic Systems, Second Edition, Addison-Wesley, 1990.

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

[15] Mathematica. Website: http://www.wolfram.com/products/mathematica/index.html, 2008.

[16] B. Kaminska K. Arabi, I. Bell, P. Goteti, J.L. Huertas, B. Kim, A. Rueda, M. Soma. Analog and Mixed-Signal Benchmark Circuits-First Felease, In Proc. Test Conference, IEEE, pp. 183-190, 1997.

[17] Website: http://www.synopsys.com/products/mixedsignal/hspice/hspice.html, 2008.

[18] VHDL-AMS IEEE Standard. Website: http://www.eda.org/vhdl-ams/, 2007

[19] Verilog Analog and Mixed Signal Language Reference Manual (2004). Available: http://www.eda.org/verilog-ams/

[20] SystemC AMS. Website: http://www.systemc-ams.org/, 2007.

[21] B. W. Kernighan and D. M. Ritchie, The C Programming Language, Prentice-Hall, Englewood Cliffs New Jersey, 1978.

[22] SPICE University of California Berkely.

[23] Website: http://www.mentor.com/products/fv/ams/, 2008.

[24] Berkeley Design Automation, Inc. *http://www.berkeley-da.com/*, 2008.

[25] Cadence Design Systems Inc. *http://www.cadence.com/*, 2008.

[26] Mentor Graphics Corporation. *http://www.mentor.com/*, 2008.