A Synchronization Algorithm for VHDL-AMS Simulation with ADA Feedback Effect

Hamid Reza Ghasemi
ECE Department
University of Tehran
Tehran 14399
Tel: +98214204078
Fax: +98218778690
hrghasemi@cad.ece.ut.ac.ir

Zainalabedin Navabi
ECE Department
University of Tehran
Tehran 14399
Tel: +98218009215
Fax: +98218778690
navabi@ece.neu.edu

ABSTRACT
The nature of analog and digital simulation is different, so they work with two different simulation kernels. Synchronization between these two kernels is an important challenge in the mixed signal simulation subject. In this paper we introduce an algorithm for mixed signal simulation that works based on reducing the analog and digital simulation interaction. When the simulator switches to analog simulation, the analog solver initiates the analog parameters to start analog simulation. Resuming the analog simulation needs some additional steps; we reduce some switching between analog and digital simulation. The efficiency of the new algorithm is shown by some examples.

1. INTRODUCTION
For many years the simulation of digital and analog circuits has been done separately. When the simulation of a mixed signal circuit is necessary, most of the designers tend to simulate analog and digital sections individually and they are urged to use different analog and digital simulation tools. One reason is that the concepts of analog and digital simulations are different and it is not reasonable to simulate a digital circuit with an analog simulation engine. Another reason is that there have not been common modeling languages that are suitable to model both analog and digital parts of design without limitations and therefore, suitable mixed signal simulation environments did not exist. Although VHDL-AMS [6] is a recent modeling language that is suitable for modeling mixed signal circuits, but the performance of its simulator strictly depends on the implementation. Interconnection of the analog and digital part of design can be viewed from two different aspects. Data transfer and synchronization between the analog and digital parts of design. Both of them should be done effectively. The resolution of the simulation should comply with both analog and digital circuit limitations.

In Section 2 we present the SIRE/M[1] intermediate format that is used for mixed signal simulation kernel. Then in Section 3, we briefly describe the implementation process from source code compilation to SIRE/M generation. In Section 4, we describe the scheduling algorithm which is used for synchronization between digital and analog engines. Section 5 contains experimental results and conclusions.

2. MIXED SIGNAL DATA STRUCTURE
This unique intermediate format can simulate both analog and digital circuit described in a common language such as VHDL-AMS.

2.1 SIRE/M Data Structure
In order to simulate a VHDL-AMS model, its SIRE/M intermediate format (IF) representation is needed. Figure. 1 illustrates the general SIRE/M structure, which is generated for a VHDL-AMS model. A SIRE/M digital block is defined for a block that represents a VHDL-AMS concurrent statement, and an analog block is defined for analog model representation correspondingly. Both the digital and analog blocks are made up of sub-blocks. Each digital sub-block is mapped directly to a concurrent statement in the VHDL-AMS model. But an analog sub-block is either mapped to an analog simultaneous statement or to an equation that will be extracted automatically from the analog circuit topology. These equations are KVL and KCL equations which are defined implicitly in the VHDL-AMS model as the topology of the analog model. Figure 1 suggests that all the sub-blocks are connected serially to build a coherent intermediate format. These blocks are ordered according to their order of appearance in the code.
When generating a sub-block each statement or equation is converted to its post order representation. Consequently the post order representations of all the sub-blocks are

[1] Simulation Intermediate Representation with Extensibility / Mixed signal

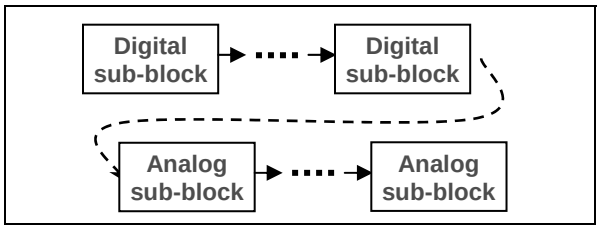

Figure 1: The general structure of SIRE/M.

concatenated to generate the SIRE/M format. This format enables the execution of the sub-blocks using a stack based execution scheme. For example consider the simple concurrent statement in Figure 2(a). The SIRE/M representation of the statement is seen in Figure 2(b). All the elements of the statement are portioned into operators and operands and used in a stack as depicted in Table 1. The result of executing the operations on all elements of Figure 2(b) is calculating and scheduling an event for the source signal. Relatively, all the digital and analog statements have a similar execution procedure. The format allows all the sub-blocks to be concatenated serially. Therefore the simulation engine can run the whole SIRE/M by executing the elements or run a single statement by initiating the execution from a given reference to the start of a sub-block.

An Analog block has two main parts, consisting of some analog statements and their topology equation. These sections

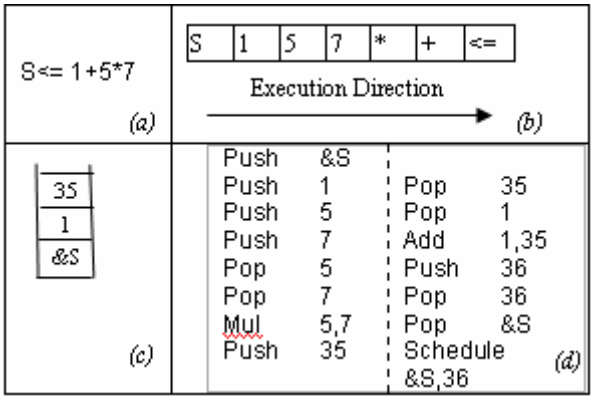

Figure 2: (a) a simple concurrent statement, (b) post order representation, (c) the execution stack after executing "push 35" in (d) part, (d) the operations that are needed to execute the concurrent statement.

Table 1: Operators and operands in SIRE/M and their stack operations

Element Type	Stack Operation
Operands	Push value or address to the stack
Operators	Pop operands, Apply operator functionality, Push the result to the stack (i.e. '+' and '<=' are operators in Figure 2)

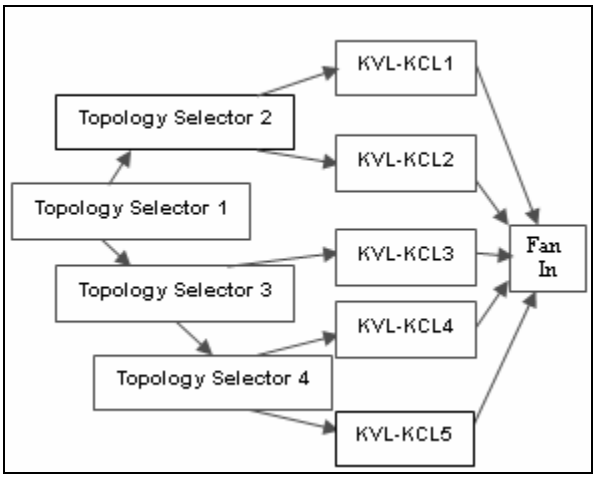

Figure 3: All topology equations in SIRE/M format with their selectors. Each selector is equivalent with one switch in circuit.

are needed for analog simulation. The first part is like the digital block. The second one represents the topology of the circuit. This topology is changed by variation in status of each analog switch in the circuit. So topology equations (KVLs and KCLs) should be changed based on the new topology. Figure 3 shows the general format of topology equations for all topologies of a circuit. Topology selectors are defined as shadows of the branches in the SIRE/M that are normally real switches in real circuits. Variations in values of branches (switches) change the values of topology selectors, so the new path of KVL and KCL equations is selected for analog simulation.

All the atomic operations during execution of a statement are simple and may be mapped to a single instruction on a processor. The implementation of SIRE/M model is a memory linked list model that is executed by a fast kernel interpreter.

As an example, consider the VHDL-AMS source code in Figure 4 and its SIRE/M is shown in Figure 5. The *First Digital* and *First Analog* components define the start of digital block and analog block. The *First Statement* components define the start of digital or analog sub-blocks. The *Branch Operator* (ABO) stands for the *simultaneous if statement* in the source code that is an analog switch model. The *Shadow Branch Operator* (Selector) is a *topology selector* which uses the value of the Branch Operator to select the correct KVL and KCL equations during the simulation time. For executing the concurrent statement, the digital kernel calls the first SIRE/M component of that concurrent statement or passes the control of simulation process to analog solver. The analog solver calls the first SIRE/M component of the analog block statements.

3. IMPLEMENTATION

The procedure of simulation consists of compilation phase, elaboration phase and mixed signal simulation.

```
ENTITY    ShowSIREM    IS    END ShowSIREM;
ARCHITECTURE example OF ShowSIREM IS
Terminal Ta, Tb : ELECTRICAL;
quantity Vsrc Across Isrc Through Ta ;
quantity Vr Across Ir Through Ta to Tb;
quantity Vc Across Ic Through Tb;
quantity Vl Across Il Through Ta;
constant R1: real := 25.0;
constant C1: real := 0.000001;
constant L1: real := 0.01;
Constant Vth: real := 2.5 ;
Signal Switch : Boolean ;
BEGIN
  Vr == Ir * R1;
  if NOW < 2.5 ms use
    Vsrc == 5.0;
  else
    Vl == L1 * Il'dot;
  end use;
  Ic == C1 * Vc'dot;
  Switch <= Vc'above(Vth)
END two;
```

Figure 4: The VHDL-AMS code of a simple example.

CHIRE/CE[2] is a memory representation of AIRE/CE[3] that its deficiencies are solved (for more information refer to [1][2][4]). This memory representation is an object oriented intermediate format and is designed to support VHDL-AMS. A VHDL-AMS source code is compiled by the analyzer and CHIRE/CE data structure is generated. After the CHIRE/CE generation, elaboration process [3] operates on CHIRE/CE. This process has two internal steps; CHIRE/CE general elaboration and simulation specific elaboration. In this step SIRE/M is generated from the CHIRE/CE representation. Mixed signal simulation kernel starts simulation [5] on the SIRE/M data structure and passed the results to the wave viewer for display.

4. SCHEDULING ALGORITHM

The synchronization unit is a basic unit in the mixed signal simulation. It synchronizes the simulation cycle of the analog solver and the digital kernel. Scheduling algorithm is the main part of synchronization unit. The mixed signal simulators usually use the canonical algorithm. We describe canonical and Liyi [7] algorithm and then introduce our new algorithm.

4.1. Canonical Scheduling Algorithm

The execution of a discrete-event process is instantaneous, meaning, time is not advanced during execution. The next active time is known before execution and is simply the time of the event time in the event queue. On the other hand, the execution of the analog solver advances the simulation time during execution. The start time and the end time of the execution are not the same. The analog solver divides duration time between the start and the end time into some slices for analog simulation. Only when the analog solver has solved a slice time, it is possible to say the analog solver advances at that time. Therefore the discrete-event process and the analog solver have two kinds of time. The mixed signal simulator needs to coordinate between the digital kernel and the analog solver, and synchronizes them during a simulation cycle precisely. It is called the canonical scheduling algorithm. To help the discussion of the synchronization algorithm, a graphical representation is used (see Figure 6). The digital time line can be thought as the representation of the execution of the digital kernel. The analog time line represents the execution of the analog solver. The time-points of analog generated by the analog solver are marked as dots on the time line and each of these dots could potentially result in events. The discrete-events are marked as square. The black square represents the executed events. The white square represents the events to be executed in the future. Assuming that the digital kernel and the analog solver are synchronized at Tc, the analog solver advances from Tc to Tn. When reaching Tn, the analog solver suspends, the digital kernel resumes as shown in Figure 6 (a). The events at Tn are executed. If a Q'Above (E) event occurs at the time interval [Tc; Tn], the analog solver suspends, the digital kernel must take it into account, and the corresponding sensitive process to the event will be executed. As a result the digital kernel and the solver are synchronized again as shown in Figure 6 (b).Figure 6 (c) shows in the execution of the break statement at time Tn, a discontinuity occurs; the analog solver must resume and recalculate an analog time-point. In fact, the digital events are the controllers.

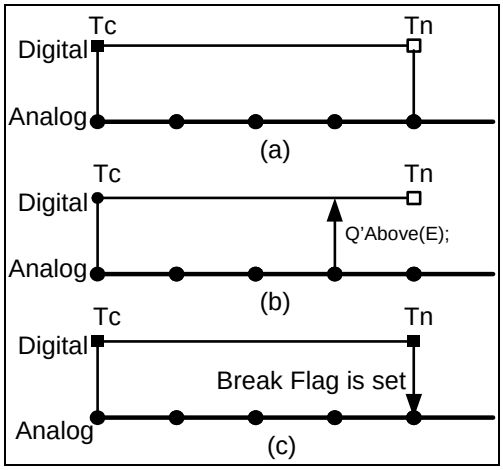

Figure 6: (a) No Q'Above (E) event occurs in the time interval [Tc; Tn]. (b) Q'Above (E) event occurs in the time interval [Tc; Tn]. (c) A discontinuity occurs at Tn;

[2] Compiled HDL Intermediate Representation with Extensibility/Common Environment

[3] AIRE/CE is a trademark of FTL systems

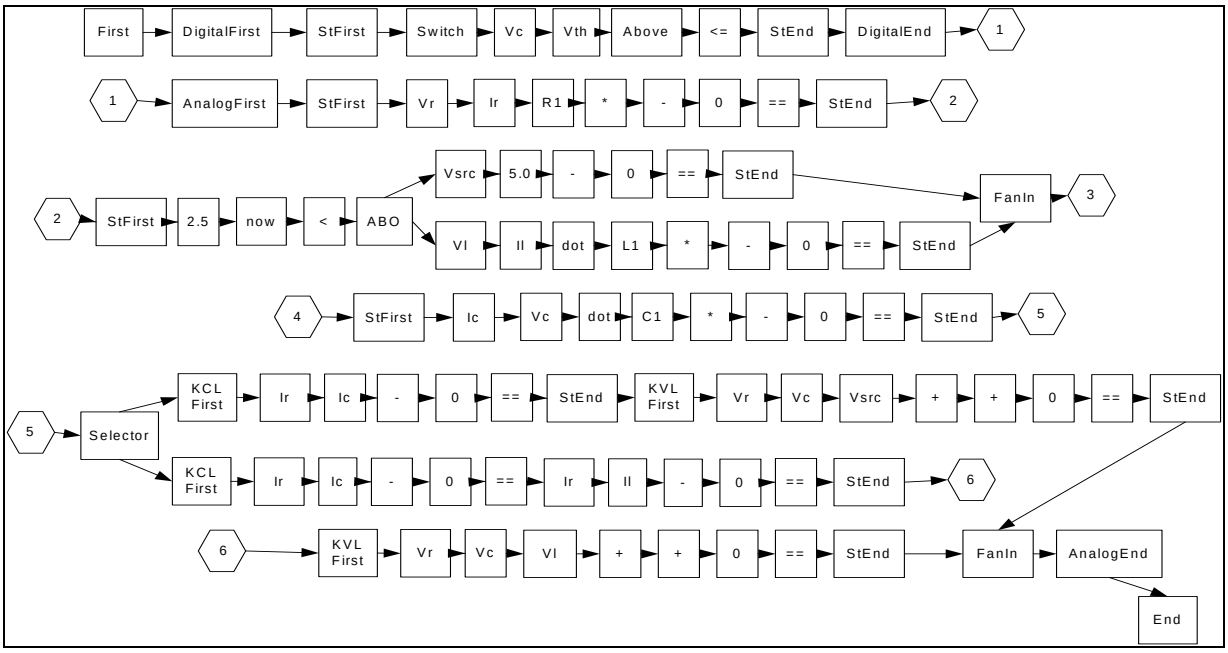

Figure 5: SIRE/M representation of the mixed signal circuit in Figure 4.

The analog solver shakes hands with the digital kernel at each digital event time. The analog solver must synchronize with each discrete-event time-point whenever it reaches the time-point.

4.2. Liyi Scheduling Algorithm

The difference between the canonical and Liyi algorithm [7] is the control to the time step when the analog solver reaches Tn This algorithm allows the analog solver to calculate a time-point beyond Tn as shown in Figure.7. If there is no Q`Above(E) event generated from Tc to Tn, the analog solver calculates a time-point beyond Tn, then it suspends. The simulation control is turned over to the digital kernel. The events at Tn are executed. If the values of quantities are referenced by processes, these values at Tn are calculated by interpolation using the time-points on the right and left of Tn. We need not invoke the analog solver to calculate a time-point at Tn. Therefore the computation cost can be reduced. Figure.7 (a) shows the case. If the two time points on each side of Tn are named as Ti and T_{i+1} respectively, Q represents the quantity values at each time-points, the linear interpolation value at Tn is:

$$Q_n = Q_i + \frac{T_n - T_i}{T_{i+1} - T_i}(Q_{i+1} - Q_i) \qquad (1)$$

It is marked as a rhombus on the analog time line. After the events at Tn are executed, if there is no break flag set, the analog solver resumes at T_{i+1}. Figure.7 (b) shows the case.

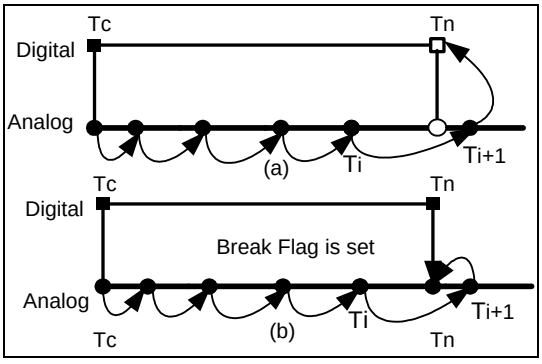

Figure 7 :(a) No Q`Above(E) event occurs in the time interval [Tc; Tn], the analog solver suspends at Ti+1. (b) A discontinuity occurs at Tn, the analog solver must do back tracking, the values at Ti+1 are discarded.

When the analog solver suspends beyond Tn, the digital kernel resumes at Tn. If the break statement is executed in some process, the break flag is set, the analog solver must do back tracking to calculate a time-point at Tn, and the values at T_{i+1} are discarded. In the next simulation cycle the analog solver starts from Tn to the next digital event. The analog solver needs back tracking only once.

4.3. New Scheduling Algorithm

The analog simulation consists of two main phases: the initial value calculation phase and the next value calculation phase. The initial value calculation phase is performed once

for the first time-point when the analog simulation starts. For the next time-points the next value calculation phase is entered, if some suspended time-points are eliminated, the initial value will not be calculated for these eliminated suspended time-points so the simulation time will be reduced. The events are partitioned into two sets. 1- The set of events that affect only the digital part. For these events there is no need to suspend the analog simulation because they do not change the status of the analog part of circuits. 2- The set of events that affect the analog part. The new scheduling algorithm works the same as the canonical algorithm for these events. According to the pseudo-code of Figure 8, if a quantity can potentially affect a signal driven by analog-digital interconnection element (e.g. above attribute in VHDL-AMS), and that signal can directly or indirectly change values of a simultaneous statement, there is a feedback from the quantity to the analog part of the circuit. Such quantities belong to QSetF, and all other quantities belong to QSetNf. These feedbacks will be named "ADA: Analog, Digital, Analog" feedbacks in this paper. These feedbacks are shown in Figure 9.

The new algorithm works identical to the canonical algorithm except two differences. The first difference is Liyi difference explained in previous section.

```
QSetF = { };// with ADA feedback

QSetNf = { };//without ADA feedback

For (i = 0; i < QNumber; i++)
{Qi = QList[i];
  If (Qi has an ADA feedback)
   QSetF = QSetF U {Qi};
  Else
   QSetNf = QSetNf U {Qi};
}
```

Figure 8: Pseudo-code of quantity partitioning

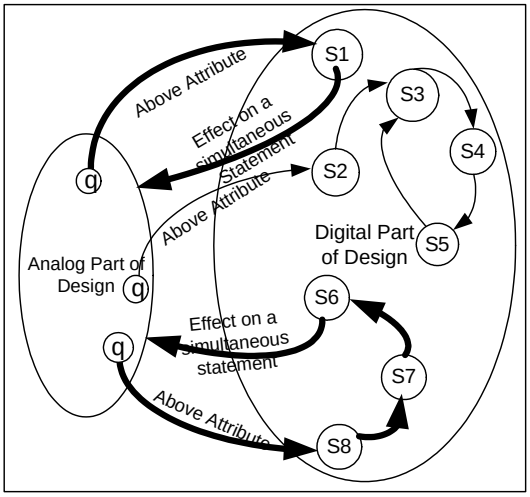

Figure 9:The bold lines show direct and indirect ADA feedbacks.

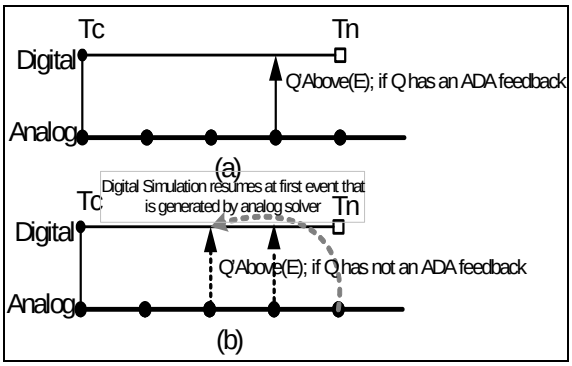

Figure 10: Q'Above (E) event occurs in the time interval [Tc; Tn]; (a) if Q has an ADA feedback, analog simulation will be suspended. (b) if Q has no ADA feedback, analog simulation will not be suspended;

The second difference is when the analog solver generates an event, it checks the event generated by a quantity If the quantity belongs to QSetF the analog solver suspends the simulation and the scheduler handles the events until that time, otherwise the analog solver can continue the analog simulation until it reaches the end time or another event generated by a quantity from QSetF. When the analog simulation is suspended, the current simulation time goes back to the first event that is not handled by the scheduler and the digital simulation kernel starts the simulation until it reaches the last analog simulation time as shown in Figure 10.

5. RESULTS AND CONCLUSIONS

We present two kinds of models. The first kind is analog to digital converter (ADC). This model has no ADA feedback. We choose an 8-bit and a 16-bit ADC. The input is a sine waveform. The simulation result of ADC8 is shown in Figure 11. The second kind is clock generator that has some ADA feedbacks. We choose a clock generator that has two interconnections, one with an ADA feedback and the other one without ADA feedback. And a clock generator that generates two outputs with two different periods and duty cycles. The part of VHDL-AMS codes of these examples are shown in Figure 12. The simulation result of ClockGen_1 is shown in Figure 13. The clock is generated based on capacitance voltage charged and discharged.

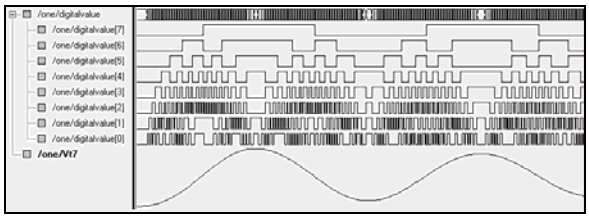

Figure 11: The simulation result of 8-bit ADC

```
library IEEE;
use ieee.electrical_systems.all;
entity adc_8 is end entity ;
architecture one of adc_8 is
terminal Ta, Trl, Tr2, Tb: ELECTRICAL;
quantity Vrl Across Irl Through Trl;
quantity Vsa Across Isa Through Ta ;
quantity Vsb Across Isb Through Tb ;
quantity Vla Across Ila Through Ta;
quantity Vlb Across Ilb Through Tb;
quantity Vca Across Ica Through Trl To Ta;
quantity Vcb Across Icb Through Trl To Tb;
quantity Vt7 : voltage;
constant Rl: real := 20.0;
constant Ca, Cb: real := 0.000001;
constant La, Lb: real := 0.1;
constant Cl: real := 0.000001;
signal digval : bit_vector(7 downto 0);
type bvector is array(natural range <>) of
boolean;
signal s:bvector(255 downto 0);
begin
-- signal generator
Vrl == Irl * Rl;
Ica == Ca * Vca'dot; Icb == Cb * Vcb'dot;
if NOW <= 2500000 ns use
Vsa == 5.0; Vsb == 5.0;
else
Vla == La * Ila'dot; Vlb == Lb * Ilb'dot;
end use;
--bias cicuit
Vt7 == (Vca + 5.0) * 25.6;
--sampling circuit
s(0) <= Vt7'above(0.0);
……1 to 254 sampling …….
s(255) <= Vt7'above(255.0);
--Encoder
process(s)
begin
if s(255) = true then digval <= "11111111";
……254 downto 1 encoding…….
elsif s(0) = true then digval<= "00000000";
end if;
end process;
end one;
```

```
library IEEE;
use ieee.electrical_systems.all;
entity clock_genl is end entity ;
architecture one of clock_genl is
Terminal Ta, Tb : ELECTRICAL;
quantity Vsrc Across Isrc Through Ta ;
quantity Vr Across Ir Through Ta to Tb;
quantity Vc Across Ic Through Tb;
constant Rl:real:=1000.0;
constant Cl:real:= 0.000001;
signal   sl, s2, st, f0, fl : boolean;
function Sw(f0,fl:boolean)return boolean
is begin   return f0 or not fl ;    end;
begin
Vr == Ir * Rl; Ic == Cl * Vc'dot;
if Sw(f0,fl) use Vsrc==5.0;else
Vsrc==0.0; end use;
sl <= Vc'above(3.0); st <= Vc'above(1.0);
s2 <= not st or sl;
f0 <= Vc'above(4.5); fl <= Vc'above(0.5);
end one;
```

Figure 12: The clock generator and an 8-bit ADC description

The simulation time of all examples is 1 sec. The comparison of Liyi algorithm with the new algorithm is shown in Table 2. The performance of the new algorithm depends on the number of interconnections between the analog and digital part of design and the number of ADA feedbacks. If all digital-analog interconnections of the circuit have an ADA feedback, the performance of both algorithms will be the same.

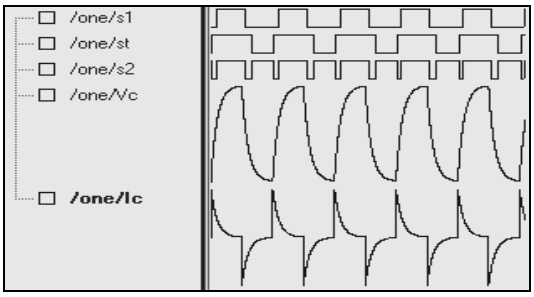

Figure 13: The simulation result of clock generator example

Table 2: The two scheduling algorithm and their performances

models	Liyi algorithm	New algorithm	Speed up
ADC8	140.72 s	100.61s	28.5%
ADC16	272.91 s	171.38 s	37.2%
ClkGen1	14.75 s	12.00 s	18.6%
ClkGen2	21.38 s	17.25 s	19.3%

6. REFRENCES

[1] "AIRE/CE: Advanced Intermediate Representation with Extensibility/Common Environment ver. 4.6," FTL Systems.

[2] A.M. Gharehbaghi, M.H. Reshadi, Zainalabedin Navabi "Intermediate Format Standardization Ambiguities, Deficiencies, Portability issues, Documentation and Improvements", Design Automation Conference 2000.

[3] Behnam Robatmili, Hamidreza Ghasemi, Dara Rahmati, Zainalabedin Navabi, "A Scalable Method for HDL Elaboration" IST 2003.

[4] Dara Rahmati, Abolfazl Salimi Zebardast, Mohammad H. Reshadi, Zainalabedin Navabi, "Handling Complex VHDL Semantics with an OO Intermediate Format," Canadian Conference on Electrical and Computer Engineering, May 2001.

[5] Dara Rahmati, Hamid reza Ghasemi, Behnam Robatmili, Zainalabedin Navabi, "A Hybrid Interpreted-Compiled Code VHDL Event Driven Simulator" IST 2003.

[6]IEEE Std 1076.1- 1999 IEEE Standard VHDL Analog and Mixed-Signal Extensions, 18 March 1999.

[7] XIAO Liyi, YE Yizheng, LI Bin" A New Synchronization Algorithm for VHDL-AMS Simulation". J. Comput. Sci & Tech Jan. 2002, Vol.17,No.1