

Synchronization of analogue and digital solvers in mixed-signal simulation on a SystemC platform

Tom J Kazmierski and Hessa Al Junaid
School of Electronics and Computer Science
University of Southampton
{tjk,hjaa01r}@ecs.soton.ac.uk

Abstract

This contribution proposes a synchronization technique for solvers able to handle analogue extensions to SystemC, for modelling of general, mixed-mode systems with digital and non-linear analogue behaviour. In order to comply with the SystemC simulation cycle semantics, we link the analogue kernel to the SystemC environment as a user module and synchronize it with the SystemC kernel via a lockstep synchronization algorithm. Operation of the extended, mixed-signal SystemC simulation platform is demonstrated using the practical example of a boost power converter, in which analogue behaviour interacts with a digital control loop. We hope that the result presented here might aid the recent efforts to standardize analogue extensions for SystemC.

1 Introduction

Several high level hardware description languages, such as VHDL and Verilog, have recently been extended to provide mixed-signal modelling capability and new standards for VHDL-AMS [CB99] and Verilog-AMS [Ver98] are now widely available. In the light of the growing popularity of mixed, analogue and digital ASICs, this is not surprising.

Naturally, with the advent of SystemC, several research publications highlighted the possibility to extend the standard, digital modelling capabilities of SystemC ([Ger00, Mue01, Pan01, ECN⁺01]) to the analogue domain. Bjornsen et al. [BHY01] described an efficient software framework for rapid behavioural modelling and simulation of analogue-to-digital converters based on SystemC. Their framework consists of three parts, the first is an analogue extension representing analogue signal classes, the second is a signal module library, which includes basic blocks required for A-D modelling, such as switched-capacitor integrators, operational amplifiers and a track-and-hold amplifier. The third part is a mixed-signal system test bench. Einwich *et al.* [ECN⁺01] presented a SystemC extension allowing an overall specification and verification of mixed-signal systems with linear analogue models. They consider an ADSL line driver with analogue linear filters as an example.

This contribution outlines the concept of a general, mixed-signal SystemC simulator comprising an analogue kernel with an underlying transient solver for non-linear, algebro-differential equations. We devote particular attention to the problem of synchronizing the analogue kernel with the digital one. Our synchronization technique is compliant with the definition of SystemC simulation cycle semantics [Mue01]. In order to comply with the SystemC simulation cycle semantics, we link the analogue kernel to the SystemC environment as a user module and synchronize it with the SystemC kernel via a lockstep synchronization algorithm. Operation of the extended, mixed-signal SystemC simulation platform is demonstrated using practical example of a boost power converter, in which analogue behaviour tightly interacts with a digital control

loop. The results presented here might aid the current efforts to standardize analogue extensions for SystemC [VGE03].

2 Solver synchronization

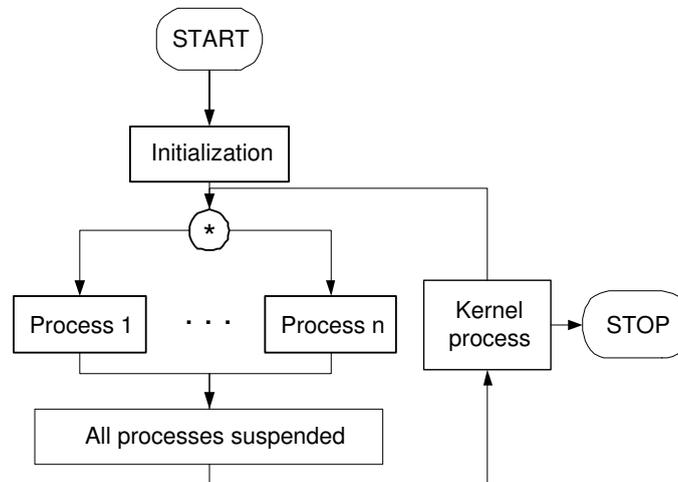


Figure 1: SystemC simulation cycle [Mue01].

Like in the case of most high-level hardware description languages, a SystemC model consists of a hierarchical network of parallel processes which exchange messages under the control of the simulation kernel process concurrently update the values of signals and variables. Signal assignment statements do not affect the target signals immediately, but the new values become effective in the next simulation cycle. The kernel process resumes when all the user defined processes become suspended either by executing a wait statement or upon reaching the last process statement. On resumption, the kernel updates the signals and variable and suspends again while the user processes resume. If the time of the next earliest event t_n is equal to the current simulation time t_c , the user processes execute a delta cycle.

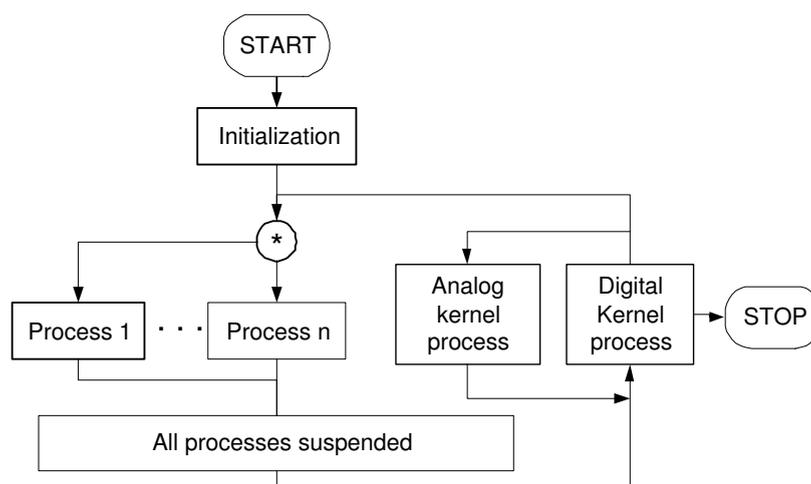


Figure 2: Simulation cycle of a SystemC system with an analog extension.

To comply with the SystemC execution semantics, our mixed-signal SystemC simulator comprises an analogue kernel which drives user defined analogue modules, as shown in figure 2.

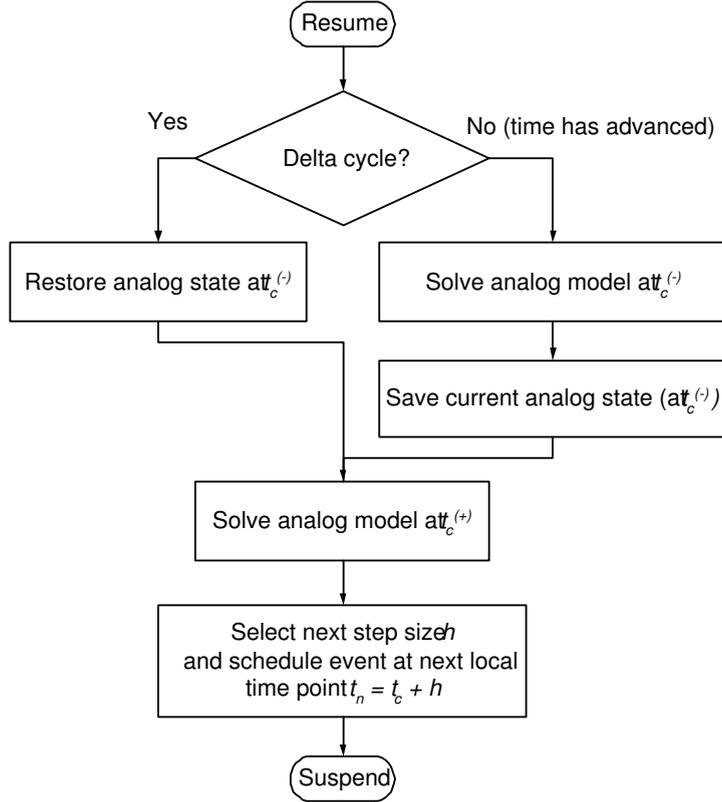


Figure 3: Analog kernel’s simulation cycle.

This scenario requires the analogue solver to be able to handle delta cycles in a manner similar to that of digital processes, namely, the state of the analogue solver may not be updated until after the SystemC kernel advances the simulation time ahead of the current simulation time t_c . In other words, the underlying analogue solver not only must be able to execute delta cycles with the step size $h = 0$, but also must have a capability to backtrack to the state just before t_c ($t_c^{(-)}$), should new events at t_c change the analogue stimuli. The former requirement would be difficult to satisfy with most standard, SPICE-like simulators since they require a minimum step size greater than zero. Moreover, small step sizes can cause large round off errors and lead to inaccurate results. The analogue model proposed in section 3 can be solved with arbitrary step size, including $h = 0$. If necessary, backtracking can be achieved by saving the analogue state at $t_c^{(-)}$ when required. The analogue kernel repeatedly executes the simulation cycle shown in figure 2, which might involve delta cycles and backtracking.

Analogue simulators do not use events but instead employ an entirely different approach to time step control. The time step may be continually adjusted to minimise the errors caused by the numerical integration method used to solve differential equations in the circuit model. In order to minimise the error in the numerical methods, it may be necessary to repeatedly reject the circuit solution at a time point and to cut the time step. It is therefore necessary for the analogue kernel in a SystemC environment not to advance past the current simulation time t_c lest a delta cycle occurs and re-evaluation of the current step is necessary.

There are essentially two algorithms for the synchronisation of two or more solvers running concurrently [ZGM⁺95]. One, optimistic approach allows each simulator to progress in time until it runs out of internal events (internal activity has ceased) and then suspend. The alternative approach, adopted here, is the lockstep algorithm. The analogue kernel advances until the current simulation time and, before suspending, schedules and event at the time equal to the current simulation time plus the next selected step size. This approach ensures that the SystemC

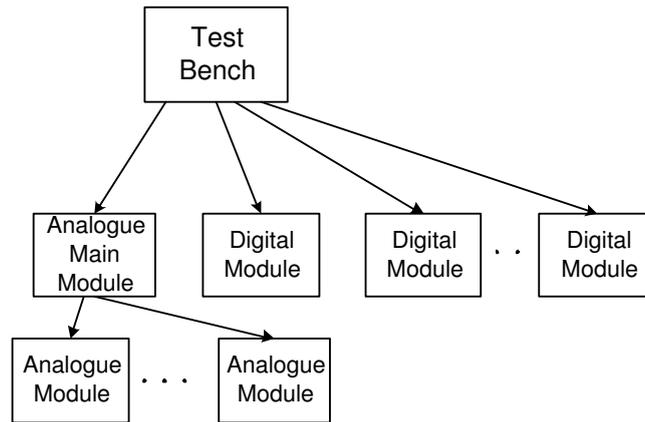


Figure 4: Analogue modules in a mixed-signal SystemC module.

kernel will make a step in time no larger than the analogue kernel's step size. Since the analogue kernel runs as a user process and is controlled by the SystemC kernel, no synchronization deadlock may happen. The only causes for deadlock can arise due to a failure to converge in the analogue solver or due to unresolvable delta cycles.

Global nets and synchronization. The signals that pass between the digital and analogue domains are carried by global nets. The solution method adopted here allows each global net to be able to connect two partitions (the digital and analogue simulator), one of which owns the net. This means that a general global net is divided into two smaller terminal nets, implemented as SystemC ports. The synchronization approach adopted in this paper is based on the lock-step principle to ensure that no results are thrown away and there is no need for backtracking. Most existing digital solvers cannot backtrack and therefore no fundamental changes are required if a mixed-signal system is integrated to the SystemC kernel. Our lock-step synchronisation algorithm has been implemented as a modification to the digital kernel and can be described in the form of pseudo-code as follows:

```

time <- 0;
initialise both the analogue and digital kernel;
while (time <= end_time) do

    while (immediate notifications are pending) do
        execute the analogue kernel
        distribute notifications generated by the analogue kernel on global nets

        while (there are active processes) do
            run a selected process
        end while (there are active processes)

        update signals
    end while (immediate notifications are pending) // delta cycle loop

    advance time to the next timed notification
end while (time<= end_time)
  
```

In most cases, delta cycles caused by zero-delay paths are eventually resolved but, in general, sharing zero-delay paths between the two kernels should be avoided. If the maximum allowed number of delta cycles is exceeded, the algorithm treats this situations as deadlock and stops.

3 Analogue Model

The description of the analogue model relies on C++ classes that provide behaviour for circuit nodes and primitive components. The component classes contain virtual methods that are invoked at matrix build time each time the analogue solver requires to update the Jacobian according to modified nodal analysis stamps. The top analogue module must run a method named BuildCircuit() which contains the analogue circuit structure.

```
void analog::BuildCircuit(void){ // -----  
    ...  
    node v1("v1"),v2("v2"),vc("vc"),vout("vout"),vf("vf");  
    node vramp("vramp");  
  
    voltageS    E("E",v1,0,1.5);  
    inductor    l1("l1",v1,v2,10e03,5.0); // 10mH,5ohm  
    resistor    ro("ro",vout,0,500.0);  
    resistor    r1("r1",vout,vf,23e3);  
    resistor    r2("r2",vf,0,10e3);  
    MOS0        M1("M1",v2,vc,0);  
    diode0      D1("D1",v2,vout);  
    capacitor    Co("Co",Vout,0,1e-3);  
    voltage_ramp vr("vr",vramp,0,1e-6);  
    ...  
} // analog::BuildCircuit() -----
```

The analogue kernel invokes BuildCircuit() once prior to the simulation. Additional user-defined methods represent interfacing components that provide functionality for synchronization between SystemC ports and their corresponding values in the analogue solver. The interfaces are bound at the build stage.

```
sc_signal<bool> ASig;  
sc_signal<double> FSig;  
sc_signal<bool> S1;  
sc_signal<double>S2;  
sc_clock TestClk("TestClock", 0.1, SC_US,0.5);//label, period,duty cycle  
    ...  
analog analog1("analog1");  
digital digital1("digital1");  
  
analog1.Vcontrol(S1);  
analog1.Vout(S2);  
analog1.clock1(ASig);  
digital1.Vout(S2);  
digital1.Vcontrol(S1);  
  
stim stim1("stimulus");  
stim1.A(ASig);  
stim1.Clk(TestClk);  
  
mon monitor1("monitor");  
monitor1.A(ASig);  
monitor1.F(FSig);  
monitor1.Clk(TestClk);  
    ...
```

4 Example

The analogue extensions added to the language contain classes to handle electrical nodes and primitive analogue components with user defined behaviour necessary to build and solve the analogue equation set. A boost 1.5V to 3.3V power converter has been used as an example of a non-trivial analogue and digital interaction. Systems of this kind usually put standard, SPICE-like simulators into difficulties because of the disparate time scales of their transients. In the case of a switched-mode power supply, the analogue transient in the output circuit is four to five orders of magnitude slower than the fast switching waveforms in the digital controller. As a typical simulation in a system of this kind might require a hundred million time points, excessive CPU times often occur when the entire system is modelled on the circuit level. The capacity of SystemC to enable system-level, mixed-signal modelling can vastly reduce simulation times where concepts need to be verified quickly and detailed, circuit-level modelling is not required.

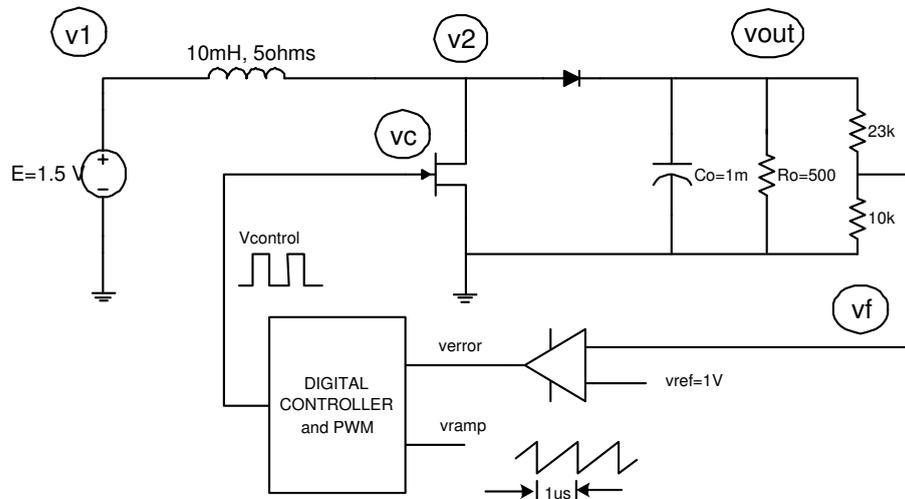


Figure 5: Boost 1.5V to 3.3V switch-mode power supply with digital control.

The system's block structure is presented in figure 4. The controller's digital behaviour in the example is modelled as a standard SystemC *SC_MODULE*. The testbench instantiates both the analogue and digital module and provides global nets. Sample simulation results at steady-state are presented in figure 4.

5 Conclusion

An extension to SystemC has been developed to simulate general, mixed-mode systems with digital and non-linear analogue behaviour. The analogue solver is coupled to the SystemC kernel via a lockstep synchronization algorithm, whose principle has been demonstrated using the practical example of a mixed-signal boost power converter, in which analogue behaviour interacts with a digital control loop. The results might be useful in future efforts to standardize analogue extensions for SystemC.

References

- [BHY01] T.E. Bonnerund, B. Hernes, and T. Ytterdal. A mixed-signal, functional level simulation framework based on systemc for soc applications. *Proc. Custom IC Conf., CICC'2001, San Diego, California*, pages 541–545, May 2001.

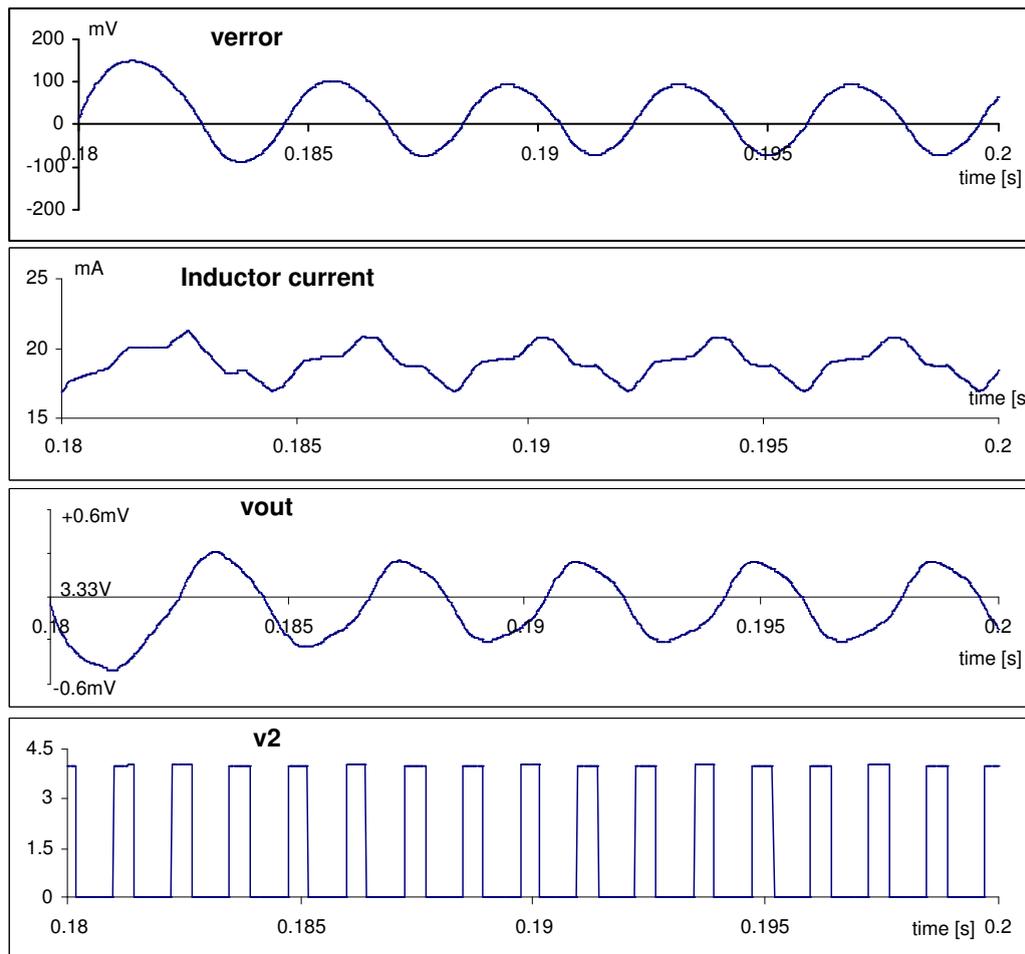


Figure 6: Simulation results for a 20ms time window in steady-state of the boost switch-mode power supply working in continuous mode.

- [CB99] E. Christen and K. Bakalar. Vhdl-ams - a hardware description language for analog and mixed-signal application. *IEEE Trans CAS II*, 46(10):1263–1272, October 1999.
- [ECN⁺01] K. Einwich, C. Clauss, G. Noessing, P. Schwarz, and H. Zojer. SystemC extensions for mixed-signal system design. *Proc. FDL'2001, Lyon, France*, September 2001.
- [Ger00] J. Gerlach. System level design using the systemc modeling platform. *Proc. SDL 2000*, 2000.
- [Mue01] W. Mueller. The simulation semantics of systemC. *Proc. DATE 2001*, 2001.
- [Pan01] P. Panda. Systemc - a modeling platform supporting multiple design abstractions. *Proc. ISSS 2001, Montreal, Canada*, 2001.
- [Ver98] *Verilog AMS Language Reference Manual, Accelera*. August 1998.
- [VGE03] A. Vachoux, C. Grimm, and K. Einwich. Systemc-ams requirements, design objective and rationale. *Proc. DATE'2003, Munich, Germany*, pages 388–393, March 2003.
- [ZGM⁺95] M. Zwolinski, C. Garagate, Z. Mrcarica, T.J. Kazmierski, and A.D. Brown. The anatomy of the simulation backplane. *IEE Proceedings on Computers and Digital Techniques*, 142(6):377–385, November 1995.