

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/236135282>

# SystemC-AMS modeling and simulation of digitally controlled DC-DC converters

Conference Paper · February 2010

DOI: 10.1109/APEC.2010.5433676

CITATIONS

8

READS

316

5 authors, including:



[Matteo Agostinelli](#)

Infineon Technologies

40 PUBLICATIONS 748 CITATIONS

[SEE PROFILE](#)



[Robert Priewasser](#)

Alpen-Adria-Universität Klagenfurt

26 PUBLICATIONS 343 CITATIONS

[SEE PROFILE](#)



[Stefano Marsili](#)

Infineon Technologies

48 PUBLICATIONS 569 CITATIONS

[SEE PROFILE](#)



[Dietmar Straeussnigg](#)

Infineon Technologies

19 PUBLICATIONS 103 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Multi Rate Low Cost Filter Banks for Blocker Detection in LTE Systems [View project](#)



Short-Range Leakage Cancellation in FMCW Radar Transceiver MMICs [View project](#)

# SystemC-AMS modeling and simulation of digitally controlled DC-DC converters

Matteo Agostinelli, Robert Priewasser, Mario Huemer  
Networked and Embedded Systems – University of Klagenfurt  
9020 Klagenfurt, Austria  
email: Matteo.Agostinelli@uni-klu.ac.at

Stefano Marsili, Dietmar Straussnigg  
Infineon Technologies Austria AG  
9500 Villach, Austria  
email: Stefano.Marsili@infineon.com

**Abstract**—In this paper, an innovative method to model and simulate DC-DC converters with a digital or mixed-signal control loop is proposed using the SystemC-AMS hardware-description language. The proposed method was employed to model a specific test case, consisting of a Buck converter with a digital PID regulator. The reliability of the model was checked by comparing the results with MATLAB/Simulink simulations. The SystemC-AMS approach was found to be well suited to model the proposed system and very efficient from a computational point of view, since the simulation time can be strongly reduced with respect to other solutions (e.g. MATLAB/Simulink).

**Index Terms**—Modeling, SystemC, Analog and Mixed-Signal simulation and design, Digital control.

## I. INTRODUCTION

Digital control loops are increasingly adopted to achieve regulation of the output voltage of DC-DC converters for several reasons. The main advantages of digital solutions with respect to the analog counterparts are programmability, versatility and reduced power consumption. Moreover, reduced sensitivity to noise and analog variations and easiness of integration with other digital systems are further advantages of digital loops [1]–[3].

From the prototyping to the verification phase of such systems, modeling capabilities and computational efficiency are key aspects of the simulation environment. From the examination of several previous works, it can be inferred that one of the most popular modeling and simulation tools used in the recent past is the MATLAB/Simulink environment (see [2], [4], [5]). While this is a feasible approach, in this paper we present an alternative solution, the SystemC language [6] along with the analog and mixed-signal (AMS) extension [7]–[10], which was found to be better suited for the modeling of such systems and can also yield a substantial reduction of simulation time.

In order to identify advantages and properties of the proposed approach, a specific test case has been modeled and simulated in this work. A schematic representation of the system is reported in Fig. 1, in which its mixed-signal structure is noticeable. It consists of a Buck converter with a digital proportional-integral-derivative (PID) controller, a simple yet popular way to regulate DC-DC converters [11]. The converter's parameters have been chosen in order to reproduce a mobile application and have been taken from a real-world commercial product [12]. An additional control loop, consisting of a mixed-signal

proportional-integral-derivative (PID) controller, has also been simulated in order to highlight the mixed-signal modeling capabilities offered by SystemC-AMS.

## II. SYSTEMC-AMS

SystemC is a hardware-description language built on top of standard C++ and, as such, has some unique characteristics when compared to other hardware-modeling approaches, such as MATLAB/Simulink. A major difference between SystemC and MATLAB/Simulink lies in the fact that the SystemC code is compiled into an executable file, whereas in MATLAB/Simulink the source code is interpreted. Moreover, being an extension of C++, SystemC allows to re-use existing C/C++ code or to employ existing external C/C++ libraries. For instance, several libraries that support advanced numerical methods can be exploited. Furthermore, all the object-oriented features of C++, such as inheritance and polymorphism, can be effectively used to model hardware [13], due to the fact that all the building blocks (or *modules*) of a system are implemented as classes.

While SystemC was initially employed to model digital systems only, the SystemC-AMS extension offers the possibility to introduce system-level design and modeling of analog and mixed-signal systems by enabling the use of dedicated simulation kernels synchronized with the standard SystemC kernel [8], [10]. Due to this fact, the AMS extension of the SystemC language permits to model part of the system as a linear electrical network. Thus, the DC-DC converter model can be built by simply specifying the electrical netlist of the converter. On the contrary, a set of differential equations must be manually introduced in MATLAB in order to model the converter, assuming that no additional toolboxes (e.g. SimPowerSystem) are used. Hence, the refinement of the DC-DC converter model (e.g. the inclusion of the equivalent-series inductance of the capacitor) is easier in the SystemC-AMS implementation (by altering the netlist) with respect to the MATLAB one, where a new set of differential equations has to be solved.

As a further remark, the SystemC-AMS language can be effectively employed at different levels of design abstraction [8], and the most suitable description method can be adopted for a given module. On the other hand, system-level tools such as MATLAB/Simulink, which are commonly used to model

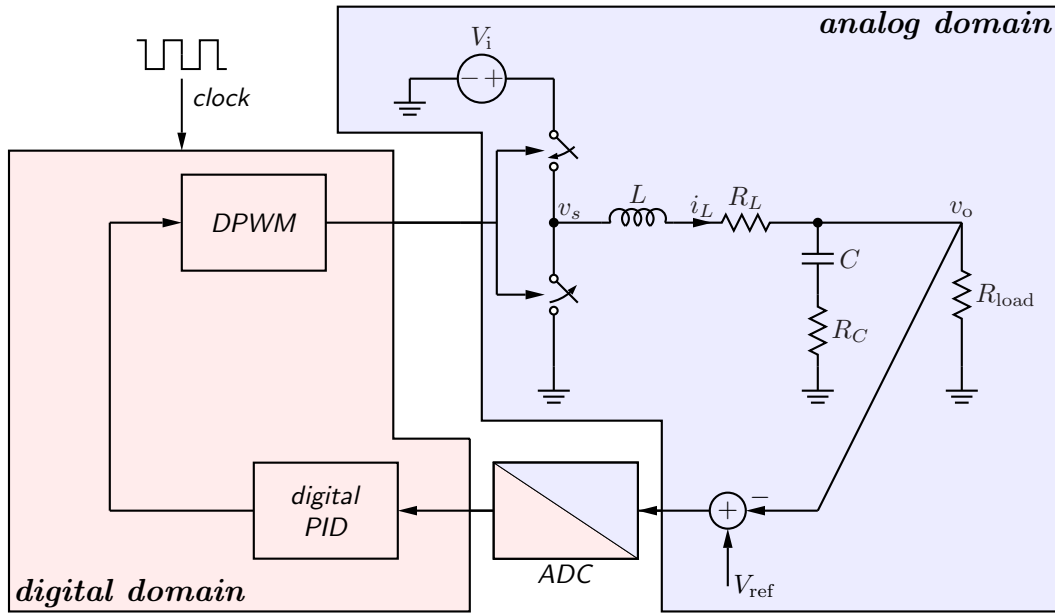


Fig. 1. Graphical representation of the system.

power converters, are capable of capturing continuous-time behavior but they do not target the design of AMS systems at an architecture-level.

### III. TEST CASE ANALYSIS: A BUCK CONVERTER WITH A DIGITAL PID REGULATOR

The test case presented in this paper is an AMS system, consisting of a Buck converter (represented in the analog domain) with a digital-domain PID voltage-mode controller (see Fig. 1). The system parameters have been taken from a real-world product [12] for mobile applications. Realistic values of the parasitics of the components of the converter are also included in the model. In this system, the analog-to-digital converter (ADC) and the digital pulse-width modulator (DPWM) represent the interfaces between the two domains. The ADC is a Double-Sampling Averaging ADC, thus two equidistant points per switching period are sampled and then the average is computed and passed to the PID regulator. The output of the PID block, i.e. the duty cycle, is then fed to the DPWM, which generates the signal used to drive the switches of the power-stage.

#### A. SystemC-AMS implementation

Some code excerpts are reported to illustrate briefly how to model the building blocks of the system in the SystemC-AMS language. In Lst. 1 an excerpt from the AveragingADC class declaration is shown. The AveragingADC model is defined such that the `sample_and_quantize()` function is executed on the positive edge of the `trigger` signal (see lines 11-12), which occurs twice every switching period. The implementation of this function is reported in lines 19-49 and it is briefly illustrated in the following. The average of the last two samples, taken from the continuous-time signal `in` which represents the error on the output voltage, is computed and

written on the `out` port, producing one value per switching period. The output port `out`, which is declared in line 5, is a standard SystemC port. The port `data_ready`, declared in line 6, is used for the synchronization with the PID block.

The code that implements the PID module is reported in Lst. 2. This module is implemented as a “conventional” SystemC class and its `update()` function is executed on every positive edge of the `data_ready` signal, as it can be seen on lines 15-16. It can be easily seen that this block implements the following discrete-time transfer function:

$$C(z) = g \frac{(z - z_1)(z - z_2)}{(z - p_1)(z - p_2)} \quad (1)$$

which is the discrete-time expression of a PID controller with an additional high-frequency pole  $p_2$ , in order to ensure the properness of the transfer function  $C(z)$  [11], [14].

On the other hand, the Buck model, which is reported in Lst. 3, is built by subclassing from the `DcDcConverter` base class, which in turn inherits from `sc_module`. The netlist of the circuit is provided by re-implementing the `architecture()` function. It is worth noting that it is possible to automatically generate the netlist from a schematic of the circuit. The definition of the high-side switch of the Buck converter is reported in Lst. 3. The connections to the electrical nodes (which are declared in line 6) are defined in line 18, while the control signal is assigned in line 19. In this case, the control signal is a standard SystemC boolean signal, as it can be seen in line 10, and it is generated by the DPWM block. Other parameters, such as the on- and off-resistances, are set in lines 20-21. A voltage source, that models the battery voltage, is then defined in lines 23-25. The voltage source is connected to the corresponding electrical nodes in line 24, while the magnitude of the voltage generated by this element

```

1  class AveragingADC: public sc_module {
2      public:
3          sca_sdf_in<double> in;
4          sc_in<bool> trigger;
5          sc_out<double> out;
6          sc_out<bool> data_ready;
7
8          void sample_and_quantize();
9
10         SC_CTOR(AveragingADC) {
11             SC_METHOD(sample_and_quantize);
12             sensitive << trigger.pos();
13         }
14
15         double vq; // Quantization step
16         double n_bits; // Number of bits
17     };
18
19     void AveragingADC::sample_and_quantize()
20     {
21         // read the sample
22         double sample = in.read();
23         // set first_sample flag
24         first_sample = !first_sample;
25
26         // quantization
27         double sample_quantized = floor(sample/vq);
28
29         // saturation
30         if (sample_quantized >= up_lim)
31             sample_quantized = up_lim;
32         if (sample_quantized < lo_lim)
33             sample_quantized = lo_lim;
34
35         // if it's the second sample,
36         // write the average on the output port
37         if (!first_sample) {
38             double average = 0.5*
39                 (0.5+sample_quantized + previous_sample)*vq;
40
41             out.write(average);
42             data_ready.write(true);
43         } else { // otherwise do nothing
44             data_ready.write(false);
45         }
46
47         // save sample
48         previous_sample = sample_quantized;
49     }

```

Listing 1. Code excerpt from AveragingADC class

is set by a signal `vin` in line 25. This signal can be generated in an appropriate testbench and could be used to simulate line jumps. The definition of the other elements of the converter is similar to the voltage source case reported in lines 23-25, thus it has been omitted.

Using a `DcDcConverter` base class is beneficial because portions of code can be shared among different DC-DC topologies (e.g. Buck, Boost, Buck-Boost) and can be included in the common base class, thus allowing code reuse. The base class can also be used to define a common interface for different DC-DC topologies, thus simplifying the substitution of a topology with another one.

### B. Comparison with MATLAB/Simulink models

The same system has been modeled in the MATLAB/Simulink environment in order to evaluate the accuracy of the SystemC-AMS implementation. The Buck converter has been modeled in MATLAB through its differential equations, discretized in time using the Euler method. An open-loop

```

1  class PID: public sc_module
2  {
3      public:
4          sc_in<double> in;
5          sc_in<bool> data_ready;
6          sc_out<double> out;
7
8          double pid_p1, pid_p2, pid_z1, pid_z2, pid_gain;
9
10         void update();
11
12         SC_CTOR(PID)
13         {
14             // read parameters from file (omitted)
15             SC_METHOD(update);
16             sensitive << data_ready.pos();
17         }
18
19     private:
20         double x1, x2;
21     };
22
23     void PID::update()
24     {
25         double error = in.read();
26         double v = error+x1; // intermediate variable
27         double w = v+x2; // intermediate variable
28         double output = pid_gain*w;
29
30         // compute next state
31         x1 = pid_p1*(x1 + error) - pid_z1*error;
32         x2 = pid_p2*(x2 + v) - pid_z2*v;
33
34         out.write(output);
35     }

```

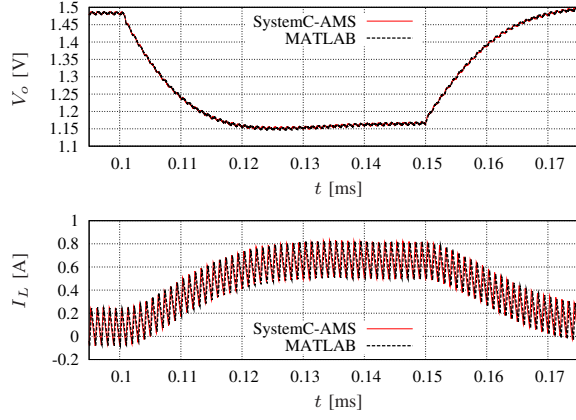
Listing 2. Code excerpt from PID class

```

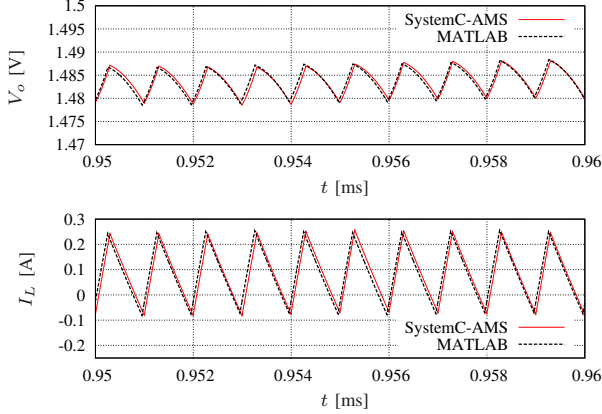
1  class Buck: public DcDcConverter {
2      public:
3          sca_sc_rswitch *ls, *hs; // Switches
4          sca_sdf2v *vbat; // Battery
5          sca_l *l; sca_c *c; // Coil and cap
6          sca_elec_node n1, n2; // Nodes
7          ...
8          sca_sdf_in<double> vin; /* Battery voltage
9                                   (inherited) */
10         sc_in<bool> pwm; /* PWM signal
11                             (inherited) */
12         ...
13     };
14
15     void Buck::architecture()
16     {
17         hs = new sca_sc_rswitch("high-side_sw");
18         hs->p(n1); hs->n(n2);
19         hs->ctrl(pwm);
20         hs->off_val = false;
21         hs->ron = p.Ronp; hs->roff = 1e12;
22
23         vbat = new sca_sdf2v("v_bat");
24         vbat->p(n1); vbat->n(gnd);
25         vbat->ctrl(vin);
26
27         ...
28     }

```

Listing 3. Code excerpt from Buck class



(a) Transient operation



(b) Steady-state operation

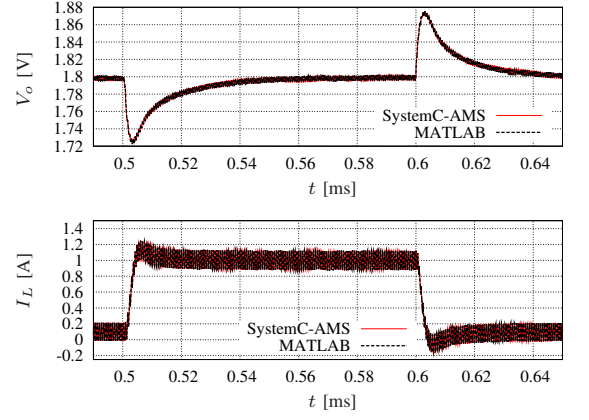
Fig. 2. Comparison of the output voltage and coil current waveforms obtained from the SystemC-AMS and MATLAB models of the open-loop system. A load jump (drop) is occurring at  $t = 0.1$  ms ( $t = 0.15$  ms).

TABLE I  
SIMULATION EXECUTION TIMES (IN SECONDS).

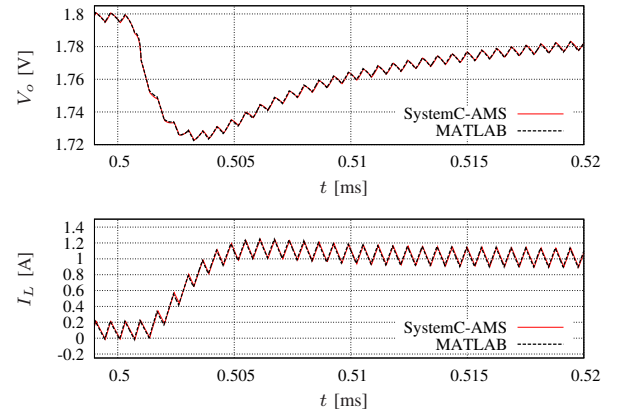
|                                     | open-loop | closed-loop |
|-------------------------------------|-----------|-------------|
| SystemC-AMS                         | 0.17      | 0.75        |
| MATLAB/Simulink                     | 1.04      | 7.04        |
| MATLAB/Simulink (Rapid Accelerator) | 2.11      | 2.37        |
| MATLAB/Simulink (SimPowerSystems)   | 1.64      | 7.05        |
| MATLAB/Simulink (PLECS)             | 1.08      | 7.01        |

simulation of the DC-DC converter has been run to check the agreement between the two different approaches. The results are plotted in Fig. 2, from which it can be concluded that the SystemC-AMS model is fully reliable. Then the closed-loop system, including the digital PID regulator, has been simulated in both environments and the corresponding results are reported in Fig. 3. Again, a perfect matching between the models can be observed.

The execution times of the simulations are compared in Tab. I in order to assess the speed performance of the different implementations. The SystemC-AMS implementation yields the best simulation times, due to the fact that the code is compiled into an executable file (however, the compilation



(a) Load jump and load drop



(b) Zoom around load jump instant

Fig. 3. Comparison of the output voltage and coil current waveforms obtained from the SystemC-AMS and MATLAB/Simulink models of the closed-loop system. A load jump (drop) with an amplitude of 1 A is occurring at  $t = 0.5$  ms ( $t = 0.6$  ms).

time has not been taken into account). The Rapid Accelerator mode of MATLAB/Simulink has also been introduced in order to provide a fair comparison with the SystemC case. In fact, when the Rapid Accelerator mode is enabled, a binary file is generated and executed by MATLAB, just as with the SystemC code. This is the reason why the gap between the SystemC-AMS and MATLAB simulation times is reduced. Instead, the Rapid Accelerator mode is not advantageous for the open-loop simulation because of the added overhead, i.e. the connection between the compiled binary file and MATLAB.

For the sake of completeness, two additional MATLAB/Simulink models were compared with the other solutions. The first one was built by using the SimPowerSystems toolbox, which allows the description of an electrical network by providing a schematic. In order to make a fair comparison, the option to use a discrete-time solver for the electrical network (with the same time-step) has been activated. The second one is instead based on the PLECS toolbox [15], a popular tool that can be used to simulate power electronics systems in the MATLAB environment. In this case, the discrete-time Buck

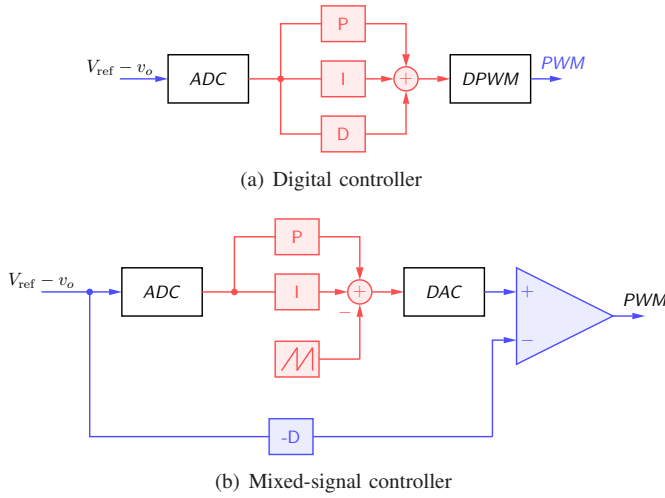


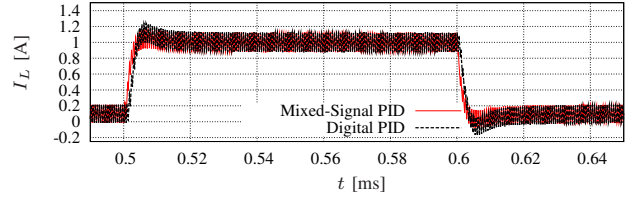
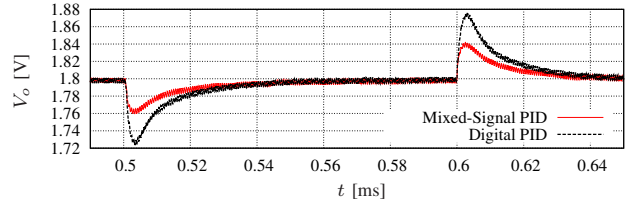
Fig. 4. Graphical representation of the (a) digital and (b) mixed-signal PID controller. Digital domain blocks (PI-controller, ramp generator) are painted in red, analog blocks (derivative action, comparator) in blue.

model has been replaced by a circuit representation, which is allowed by the toolbox. The resulting waveforms were found to be indistinguishable to the ones obtained by using the discretized differential equations in MATLAB for both models, thus they are not shown here. The execution times of the simulations are instead reported in Tab. I, from which it can be concluded that the SimPowerSystem and PLECS toolboxes exhibit a similar performance if compared to the MATLAB discrete-time model.

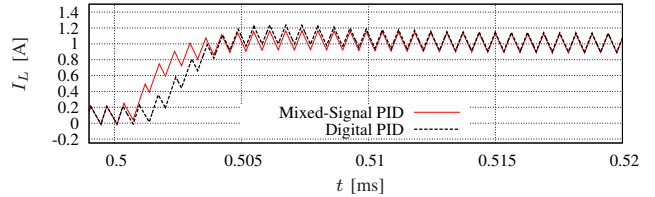
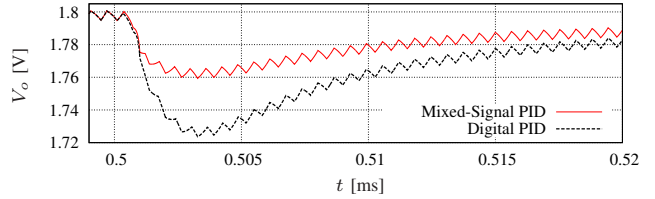
#### IV. MIXED-SIGNAL CONTROL LOOP

The proposed SystemC-AMS approach is particularly suited to model mixed-signal control loops, i.e. part of the controller is represented in the digital and part in the analog domain. The digital PID architecture that has been presented in Sec. III, which is schematically reproduced in Fig. 4(a), can be modified by moving some parts to the analog domain. The controller architecture that is evaluated in this section is a mixed-signal voltage-mode PID controller with analog derivative action [16], as reported in Fig. 4(b), where the DC-DC converter has not been drawn for simplicity.

As it can be seen from Fig. 4, the derivative action is transferred to the analog domain in order to allow a faster reaction to a line or load jump. In fact, the quantization and especially the delay introduced by the Analog-to-Digital converter do not affect the derivative part of the controller. The proportional and integral terms are instead maintained in the digital domain, by means of an ADC and a DAC. The ADC architecture has been maintained identical to the one used in the purely digital controller (as in Lst. 1). The digital PWM modulator has now been replaced by a (digital) ramp generator and an analog comparator. The system's parameters have been maintained identical to the digital PID case, while the controller coefficients have been slightly optimized to better exploit the new controller architecture.



(a) Load jump and load drop



(b) Zoom around load jump instant

Fig. 5. Comparison of the output voltage and coil current waveforms obtained from the mixed-signal and the digital control loops. A load jump (drop) with an amplitude of 1 A is occurring at  $t = 0.5$  ms ( $t = 0.6$  ms).

Simulation results are reported in Fig. 5, where the output voltage and inductor current waveforms are shown. It can be observed that, having shifted the derivative action to the analog domain, the mixed-signal controller is capable of a faster reaction to a load jump, thus reducing the under and overshoots. It is worth noting that the execution time of the simulations for the mixed-signal PID controller are comparable to the purely digital controller. In fact, the average execution time was found to be equal to 0.87 seconds.

#### V. CONCLUSIONS

In this paper, an innovative method to model and simulate DC-DC converters with a digital or mixed-signal control loop is proposed, using the SystemC-AMS language. The accuracy of the SystemC-AMS model has been evaluated by comparing the results with the ones obtained within the MATLAB/Simulink environment, and a satisfactory agreement between the different implementations has been proven. For completeness' sake, additional MATLAB/Simulink simulation tools, such as the SimPowerSystem and PLECS toolboxes, have been included in the comparison.



From the presented results, it can be concluded that the SystemC-AMS environment is capable of providing better performance in terms of simulation time and it proved to be very well-suited to model the chosen test case. An additional mixed-signal control loop, which is capable of improving the dynamic performance of the system, has also been successfully modelled and simulated with the proposed SystemC-AMS approach.

## REFERENCES

- [1] "Special issue on digital control in power electronics," *IEEE Trans. Power Electron.*, vol. 18, no. 1, pp. 293–503, Jan. 2003.
- [2] S. Saggini, M. Ghioni, and A. Geraci, "An innovative digital control architecture for low-voltage, high-current DC-DC converters with tight voltage regulation," *IEEE Trans. Power Electron.*, vol. 19, no. 1, pp. 210–218, Jan. 2004.
- [3] B. Patella, A. Prodic, A. Zirger, and D. Maksimović, "High-frequency digital PWM controller IC for DC-DC converters," *IEEE Trans. Power Electron.*, vol. 18, no. 1, pp. 438–446, Jan. 2003.
- [4] L. Corradini, S. Saggini, and P. Mattavelli, "Analysis of a high-bandwidth event-based digital controller for DC-DC converters," in *Proc. IEEE Power Electron. Specialists Conf.*, Jun. 2008, pp. 4578–4584.
- [5] J. Morroni, R. Zane, and D. Maksimović, "Design and implementation of an adaptive tuning system based on desired phase margin for digitally controlled DC-DC converters," *IEEE Trans. Power Electron.*, vol. 24, no. 2, pp. 559–564, Feb. 2009.
- [6] "Open SystemC initiative." [Online]. Available: <http://www.systemc.org>
- [7] K. Einwich, A. Vachoux, C. Grimm, and M. Bernasconi, "SystemC AMS extensions draft 1." [Online]. Available: <http://www.systemc-ams.org/>
- [8] C. Grimm, M. Bernasconi, A. Vachoux, and K. Einwich, "An introduction to modeling embedded Analog/Mixed-Signal systems using SystemC AMS extensions," Jun. 2008. [Online]. Available: <http://www.systemc-ams.org/>
- [9] A. Vachoux, C. Grimm, and K. Einwich, "Towards analog and mixed-signal SOC design with SystemC-AMS," in *Proc. IEEE Intl. Workshop on Electron. Design, Test and Applications*, Jan. 2004, pp. 97–102.
- [10] A. Vachoux, C. Grimm, and K. Einwich, "Extending SystemC to support mixed discrete-continuous system modeling and simulation," in *Proc. IEEE Intl. Symp. on Circuits and Systems*, vol. 5, May 2005, pp. 5166–5169.
- [11] R. Erickson and D. Maksimović, *Fundamentals of Power Electronics*, 2nd ed. Springer, 2001.
- [12] Linear Technologies, *LTC3404 Step-Down Regulator Datasheet*. [Online]. Available: <http://cds.linear.com/docs/Datasheet/3404fb.pdf>
- [13] L. Pomante, "Exploiting polymorphism in HW design: a case study in the ATM domain," in *Proc. of IEEE/ACM/IFIP Intl. Conf. on Hardware/software codesign and system synthesis*, 2004, pp. 81–85.
- [14] R. Priewasser, M. Agostinelli, S. Marsili, D. Straeusnigg, and M. Huemer, "Comparative study of linear and non-linear integrated control schemes applied to a buck converter for mobile applications," in *Proc. of 17th Austrian Workshop on Microelectronics (Austrochip)*, Oct. 2009, pp. 51 – 56.
- [15] Plexim GmbH, *PLECS toolbox*. [Online]. Available: <http://www.plexim.com/>
- [16] S. Saggini, P. Mattavelli, M. Ghioni, and M. Redaelli, "Mixed-signal voltage-mode control for DC-DC converters with inherent analog derivative action," *IEEE Trans. Power Electron.*, vol. 23, no. 3, pp. 1485–1493, May 2008.