2014 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

Simulation and Debug of Mixed Signal Virtual Platforms for Hardware-Software co-development

Vincent Motel, Cadence Design Systems, Inc. (*vmotel@cadence.com*)

Alexandre Roybier, Cadence Design Systems, Inc. (*aroybier@cadence.com*)

Serge Imbert, Cadence Design Systems, Inc. (*sergeim@cadence.com*)

***Abstract*—Virtual platforms are often used for high level architecture exploration and hardware-software interactions verification. AMS domain at system level is just emerging. It adds more realistic behavior that can be used to correctly check the hardware-software co-development. SystemC with its TLM extension is a natural choice for the HW/SW digital simulation part. For AMS, Real Number Modeling and Timed Data Flow are a good choice to reach the required simulation speed. The Cadence Incisive Enterprise Simulator supports all these languages and is also able to provide debugger and profiler to understand, debug and optimize the performance of the mixed simulation. Based on this, we present the experiments made with a virtual platform simulating a CPU with its peripherals including an analog sensor and its physical environment.**

Keywords—Virtual Platform; Analog Mixed-Signal; Real Number Modeling; SystemC-AMS; Timed Data Flow

I. INTRODUCTION

The modern Systems-on-Chip (SoC) have reached an unprecedented level of complexity, especially in terms of software content. Verification of a SoC before fabrication and development of the embedded software is truly challenging and relies more and more on virtual platforms.

In this paper, we discuss the support of the Analog-Mixed Signal domain (AMS) in the SoC virtual platforms. In section 2, we explain why virtual platforms need to be extended to support the AMS domain. In section 3, we discuss the possibilities to model the AMS in a virtual platform, based on the available languages and abstraction levels. An example of mixed signal virtual platform for a magnetic sensor application is presented in section 4. In section 5, we review the tool capabilities required to make an efficient use of AMS in virtual platforms. In conclusion, we summarize the lesson learned on our reference example, and consider the challenges faced by the system-level AMS.

II. ON THE NEED FOR MIXED SIGNAL VIRTUAL PLATFORMS

Development of virtual platform is related to the increase of software content in SoC, in several dimensions.

- The scale: the increasing costs of hardware creation has pushed for less application-specific hardware, more generic and more programmable. In addition, the performance requirements of modern products cannot be met by single-core architectures anymore. Processors have been bound in their maximum performance - maximum operating frequencies have not increased significantly since 10 years. Hence multi-core and even many-core architectures have become ubiquitous. This has driven up the sheer volume of software code required for each project, and the effort to produce it.

- The complexity: in those architectures, the software has complex interactions with the hardware and with the environment outside the SoC. The software stack becomes also more and more complex, as a result of the increase in scale (e.g. OS running on multi-core systems, with advanced virtualization and security modes).

- The quality requirements: the cost of SoC fabrication imposes not only a zero tolerance for bugs in hardware, but also high quality standards for the software too, because it is very related to the hardware and expected to be fully functional very soon after the availability of the silicon parts. And in many cases it is very expensive to fix the embedded software once the SoC is in production [1].

- Time to market and bound resources impose strong constraints too, forcing the SoC team to constantly improve their methodologies.

As the hardware and software development cannot be separated, prototypes of the hardware are required before the SoC fabrication, to develop and validate the software, and verify its interactions with hardware. A solution is to make hardware prototypes of the SoC, typically based on FPGAs. This approach becomes less attractive because of the time and cost of the hardware prototype development - a parallel project by itself -, expensive in terms of hardware prototype duplication and usually available late in the product development cycle, bringing less value in terms of time to market.

Virtual platforms offer an alternative to hardware prototypes [2]. Software models of the key components of a SoC are combined to form an executable system. They are faster to create because they are more abstract and only functional (no need to write synthesizable models), and inexpensive to duplicate. The models must have enough functionality to execute the code correctly and should be raised to a level of abstraction that provides the performance necessary for extensive testing. In addition, virtual platforms can be used for architecture exploration and system validation.

As SoC are very commonly interfaced to the analog world, software depending on analog behavior already exists today. However, the development, debug and validation of such software require stimuli which are not easily produced without expert knowledge, and that may require a hardware prototype or even the real device. That has an impact on costs and delays, and that limits the possible exploration of use cases at the early stages of the project.

By adding the AMS domain in the virtual platform, we can make another big step. It can help to develop more complex or more ambitious software for a smart and efficient control of the analog parts, and it can contribute to reduce the time and cost of development.

The need for an early analog model takes even more importance if software-controlled analog is involved, with feedback loops in place between the controlling software and the analog part, through digital logic. That can be found in sensitivity adaptation, or in aggressive power optimizations.

In the AMS domain too, architecture exploration can be required: less expensive analog sensors but strong post processing algorithms, power consumption and tradeoff for the cost and performance.

But all these algorithms or applications, sometimes on top of operating systems, require execution of billions of cycles. To reach an acceptable execution time for realistic scenarios, performance of simulation models requires close consideration. Digital and analog sections need to have close range of execution speed. This is discussed more in depth in the following section.

III. MODELING ASPECTS

Several languages are available to model analog behavior, digital functionality, operating system and application software. But to create and run such models efficiently in the same simulation, the abstraction level is the key aspect.

To simulate the HW/SW digital part, the SystemC language with its TLM extensions [3], using the loosely-timed coding style (LT), is the usual choice because of its good modeling capabilities with reduced effort, and fast execution. When the virtual platform includes an analog part, it is more difficult to select the optimal language and abstraction level.

As a general guideline, it is important to keep in mind that the software usually runs in the range of about 100 MHz (10x faster or slower). So in rare case when the analog time scale is very slow, we can still use an electrical representation without killing the performance of the virtual platform. As an example, a wind turbine could be modeled with an electrical discipline while keeping a good performance in term of speed because of its slow timescale. On the contrary, the same level of details for an RF front-end has no chance to present an acceptable simulation performance.

To create a model of an analog block with classical AMS languages, there are 2 main possibilities.

The first one is to use the electrical discipline of languages like Verilog-AMS [4] or VHDL-AMS [5]. In this case, the analog solver based on the physical Kirchhoff laws -voltage and current- will be used to solve differential equations. It is a conservative representation.

The second possibility is to use the digital engine only, by using analog Real Number Modeling (RNM) [6], based for example on wreal types of Verilog-AMS, or real types of SystemVerilog [7]. With this approach we don't use any more an analog solver which is too much time-consuming. We remove CPU-intensive computation from analog environment and use only a digital engine. As a consequence that modeling level is discrete in time, but still continuous in value.

For example, author of [8] reports that the simulation of the 16,384 states of a 14bit ADC-DAC pair took 3 days on a conventional mixed-signal transistor level simulation, and was reduced to 3 seconds with a model of the analog section using RNM.

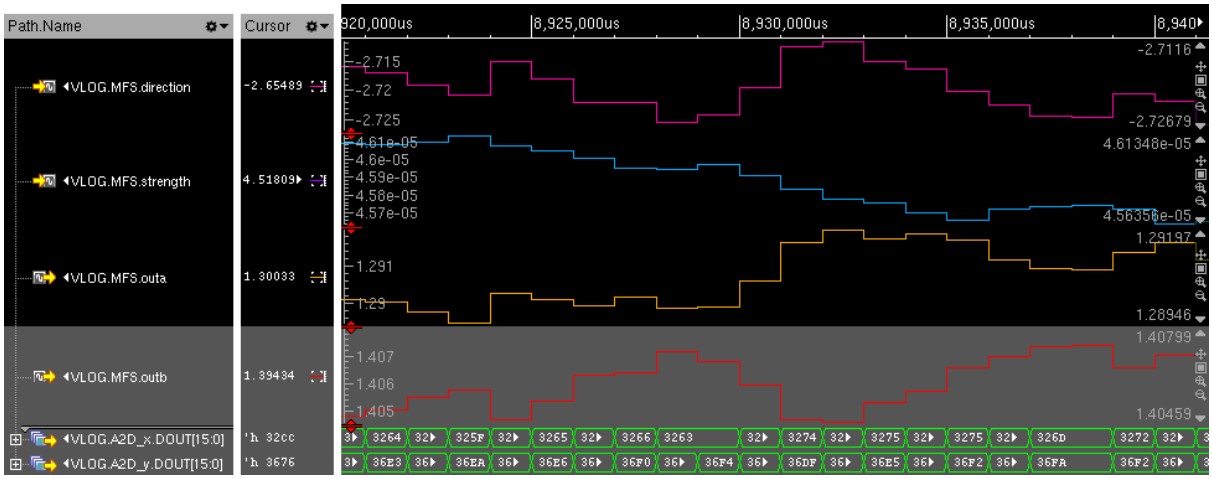

Figure 1. Waveforms of the RNM simulation of a sensor.

Figure 1 shows a screenshot of the simulation of the magnetic sensor described in section 4, based on wreal types from Verilog-AMS. We can note that inputs "direction", "strength", and outputs "outa", "outb" have values without unit, because they are not an electrical signal but signals with a real value handled by the digital engine of the simulator (Cadence Incisive Enterprise Simulator). Then "A2D_x.DOUT" and "A2D_y.DOUT" are the digital outputs from the ADC modules, they are the result of the "outa" and "outb" conversion.

One limitation of RNM is that it is non-conservative and the impedance interactions are not taken into account. It is a term of the performance/accuracy trade-off at that abstraction level. But a related advantage is that we avoid any convergence issues that happen because of discontinuities in analog models.

It is still possible and important to keep many aspects of the analog behavior in RNM, without renouncing to the speed gains. Power voltage values can be used in the models, so that some power errors can be detected in multi-power designs. Pin-level compatibility can be preserved for some components, which allows an easier replacement with more detailed models. A typical example is the Voltage-Controlled Oscillators (VCO), whose output frequency can be actually variable, and controlled by only one pin.

Another emerging language providing interesting capabilities is SystemC-AMS [9]. Similarly, it proposes several models of computation: Electrical Linear Networks (ELN) and Linear Signal Flow (LSF), both modeling continuous time and relying on a differential equation solver, and Timed Data Flow (TDF).

Like the electrical discipline of Verilog-AMS and VHDL-AMS, in most applications ELN and LSF are likely to be too computation-intensive for the speed requirement of virtual platforms. But TDF is very fast [10], even faster than the equivalent event-driven simulation, since all the computations are pre-scheduled at elaboration. Authors of [11] have shown that replacing ELN with TDF and doing further optimization of replicated structures and time step adjustment can lead to speed gains in the range of 100000x, while preserving an acceptable accuracy.

Synchronization between TDF and SystemC event-driven simulation is made by converter ports, which allow the transfer of the values to/from sc_signal in the digital part. To avoid the slowdown of TDF by frequent synchronization with the discrete event kernel, work in [12] suggests an interesting approach for the direct

connection of SystemC-AMS TDF to TLM 2.0, using temporal decoupling to preserve a high simulation speed. Recently, some limitations of TDF related to fixed time-step scheduling have been removed by the introduction of Dynamic Time Data Flow in SystemC-AMS 2.0 [13], further enhancing the possibilities for the analog models to reach the speed of digital virtual platforms.

As TDF describes only the time-domain behavior of continuous values at fixed time steps, the modeling effort is low, and furthermore, it can be used beyond the electrical analog domain to model the surrounding multi-physics environment [14].

Overall, the performance gain required for virtual platforms comes from the abstraction of several aspects of the analog circuitry and behavior : structural details are replaced by functional blocks, conservative behavior of electrical nodes can be ignored, continuous time is replaced by discrete time to avoid the need for differential equation solver, electrical values are not all represented (although is it good to know which values are represented, with relevant unit and absolute values), and in TDF, the signal flow direction is pre-determined, as well as the scheduling of the computations.

IV. AN EXAMPLE OF MIXED SIGNAL VIRTUAL PLATFORM

A. *System description*

To illustrate the various aspects of mixed signal virtual platform development and usage, we created an example based on a magnetic sensor and a software compass application with graphical display. The example is simple, yet sufficiently complete to illustrate the combination of domains and languages, the modeling possibilities and the tool support.

The center of the analog part is a two-axis magnetic sensor. It converts each direction of a low magnetic field of in a voltage value. It is followed by a pair of 16-bit analog-to-digital converters (ADC), and a digital sensor interface which makes the interface with the processor bus, with value registers and a capability to send interrupts when a programmable deviation is detected.

That AMS sub-system is integrated in two different systems: a minimal system made of an ARM Cortex processor, a LCD controller and a timer (Figure 2); and a complete SoC, the Zynq-7000 platform of Xilinx, with a dual-core ARM Cortex processor and all the peripheral required to boot a Linux OS [15].

This system is sensitive to the earth's magnetic field, whose perceived strength and direction depends on the physical position of the sensor, and the magnetic perturbations of the environment.

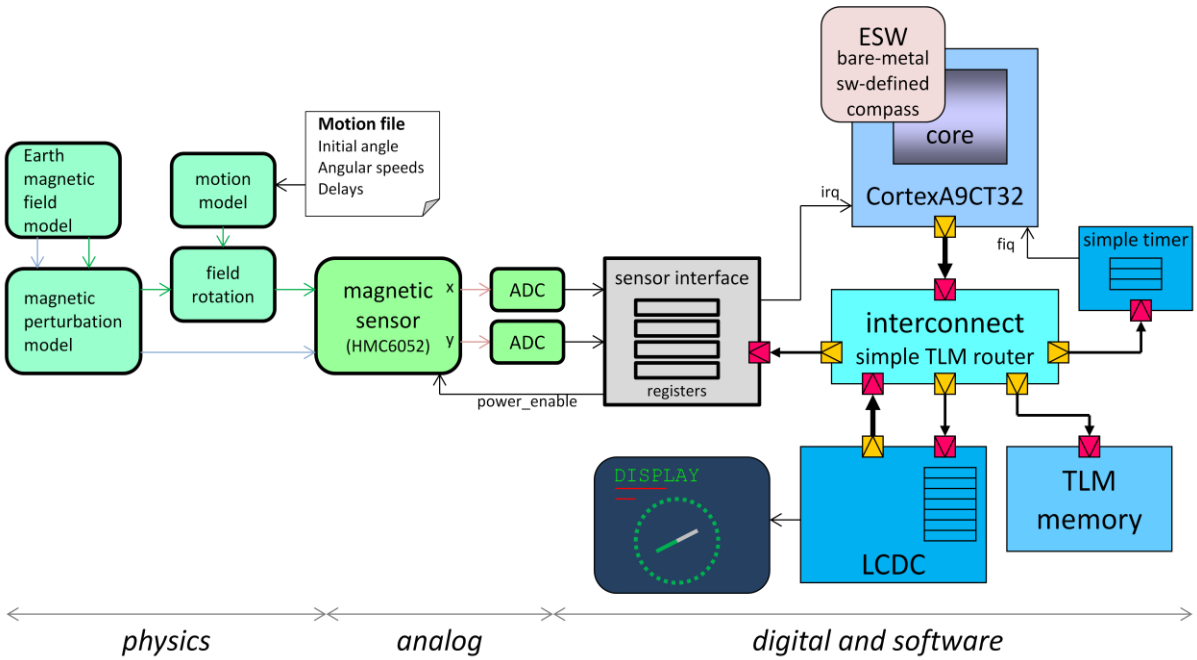

Figure 2. Example AMS virtual platform.

B. System modeling

The digital components of the virtual platform are classically modeled by SystemC modules, TLM 2.0 sockets and tlm_wire ports [16], and a fast processor model which is instruction-accurate.

The physical domain is represented by a pair of values of the magnetic field: the strength in tesla and the direction in radian. Although it could be possible to make a high level model without paying attention to the units and the absolute values, we found it was useful to take them into account from the beginning. That does not require a lot of extra effort, and it allows the early estimation of possible excursion range of the variables in the hardware models and in the software.

The chain of models starts with the earth magnetic field models, which simply outputs two constants for the magnetic field strength and direction (e.g. 50000 nT and -9°). It is followed by a motion model, which reads an input "motion file" in text format, defining the direction of the sensor's physical position: the initial angle in degrees, the angular in degrees/second, and some delays in milliseconds. The motion model integrates angular speed to determine the current angle. That angle is applied to the magnetic field by a field rotation model, simply adding the angle to the direction of the magnetic field. Another module aims at modeling some magnetic perturbations and noise from the environment.

The field is then applied to a model of the sensor. It computes the x and y values in volts, based on configurable "zero field output" and sensitivity values. Those output signals are digitalized by a pair of analog-to-digital converters (ADC). The x and y values are carried by 16bit digital signals, to the inputs of a digital sensor interface module, which is a peripheral of the processor.

That module typically stands at the boundaries of three domains: the AMS sub-system (outputs from ADC), the digital hardware (connection to other digital component of the virtual platform by TLM 2.0 target socket and tlm_wire output), and the software (its data and control registers are read and written by software, and the module can raise interrupts through interrupt request output).

We implemented the AMS part twice in two different languages, for comparison: SystemC-AMS model at TDF level made of one dataflow cluster whose time step is set in the motion model; and Verilog-AMS wreal model (RNM), also driven by the events generated by the motion model. Only the digital engine of the simulator is used. Both models are strictly equivalent: they do the same computations at the same frequency, from the same inputs. To experiment with the simulation performance aspects, time step value can be configured.

V. TOOL SUPPORT

To enable a wider adoption of System-Level AMS modeling, tool support is a key factor. Not only the tool must support the simulation of appropriate modeling level and languages, but it should provide capabilities to make the users more productive. This section discusses how this is possible with the Cadence Incisive Enterprise Simulator [17].

A. Support of AMS languages

The Cadence Incisive Enterprise Simulator supports natively, among other languages, VHDL-AMS, Verilog-AMS and SystemVerilog. To simulate SystemC-AMS models, we added the SystemC-AMS proof-of-concept simulator from Fraunhofer IIS [18], which is an extension of the SystemC kernel. As it is open-source, it can be recompiled once for the SystemC implementation of Incisive, and linked with the simulator and the compiled models of each simulation. That makes a simple and effective use model, and provides the entire implementation of SystemC-AMS.

All languages are then executed in the same simulator, in a single process, which is more efficient than a co-simulation based on several simulators running in parallel, with an overhead and additional complexity due to communication between simulators.

B. Processor models

In order to simulate a significant amount of software, virtual platforms need to use fast processor models, which are a trade-off between speed and accuracy. Typical models are instruction-accurate but not cycle-accurate,

although other possibilities exist. They are usually wrapped in SystemC modules with TLM interfaces. That approach scales well to the efficient simulation of multi-processors systems [19].

C. Visualization, debug and analysis

The Cadence Incisive Enterprise Simulator provides numerous debug and visualization possibilities, in multiple languages at different levels.

- For all the languages we can see a structural view of the design: hierarchy browser, and schematic tracer.

- Specifically for the analog part, the main capability is waveforms tracing, with a complete toolbox dedicated to analog waveforms. For the SystemC-AMS extensions, an instrumentation of the AMS kernel with existing simulator API enables full observability of the analog values, including waveforms recording.

- For the SystemC/TLM part, debug is available at several levels: at C++ source level, at the SystemC semantic level (where the tool is aware of the SystemC processes and the specific SystemC objects), and at TLM level (tools is aware of the TLM 2.0 transactions and has an API to get knowledge of custom extensions).

- For the embedded software, a complete software debugger (with source level symbolic debug, breakpoints and stepping) is available for control of the processor models. If the software under development is an application or driver running on top of an OS such as Linux, OS-aware debug capabilities raise the debugging at that level.

- Hardware-Software interface structures such as registers (organized in banks/registers/fields), interrupt signals and memory maps are of special interest in virtual platforms, then dedicated debug features are available.

It is especially useful to be able to combine all those capabilities at the same time in the same tool environment. To illustrate this, on the magnetic sensor example, it is possible to trace the analog waveforms, display the transactions from the processors to the sensor interface, and run step by step in the compass software at the same time (Figure 3). That makes possible the debug of system level scenarios, such as the refinement of the software calibration algorithm, which depends heavily on its analog inputs.

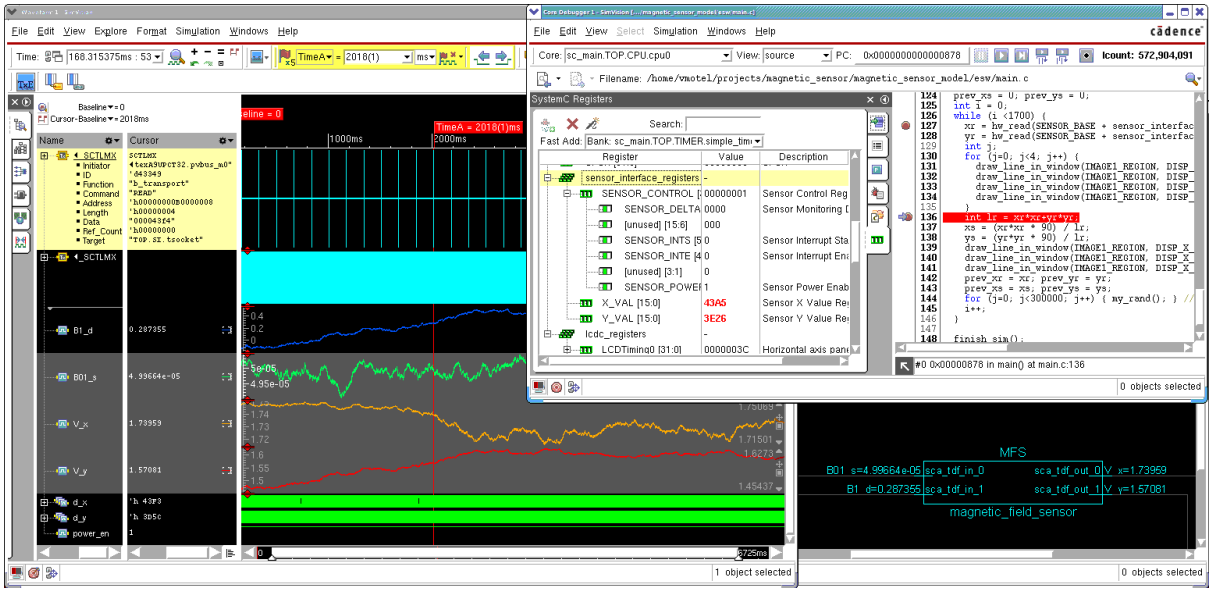

Figure 3. Debug of an AMS virtual platform in multiple domains.

D. Profiling and simulation performance analysis

As discussed in section 3, the simulation performance is a key enabler of the execution of realistic scenarios with virtual platforms, and the high performance first comes from the selection of the appropriate modeling level for the application. To optimize further the simulation speed and to chase performance bugs, an accurate multi-language profiler is instrumental. It should present the profiling data at multiple levels, corresponding to the levels where the models can be optimized. At high level, tool can count processes activation and transactions, to

check that the models run at the expected behavioral granularity. Those counts are not necessarily related to the time spent in the processes or transactions. At a lower level, it can report of the actual time spent in processes and functions. To get the best understanding from the reported figures, the profiler consolidates them by instance, by module type, or by category. For SystemC, some awareness of the language semantics is required to make the relation between code execution and its context (e.g. a function executed by a specific SystemC thread of a specific sc_module).

We used the simulator's profiler on both implementations of the example magnetic sensor design (SystemC/TLM + SystemC-AMS and SystemC/TLM + Verilog-AMS), with values of the AMS time step varying from 1 ms to 1 µs (10^3 to 10^6 samples/s). The first thing to consider is the total simulation time for a given use case, then the split of that time in AMS vs. digital models (Figure 4).

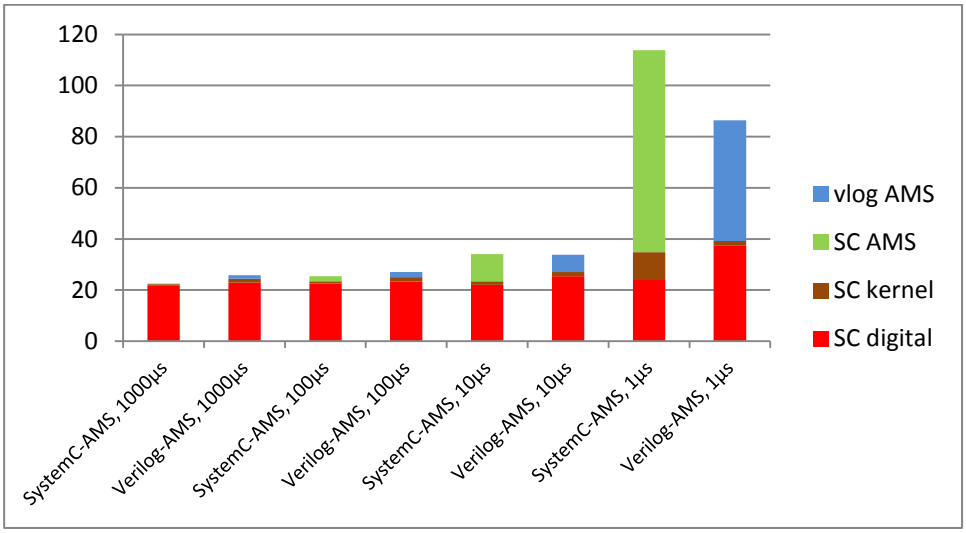

Figure 4. Simulation time, in seconds, for 30s of simulated time.

It is interesting to observe that the SystemC-AMS model is the fastest for large time steps (fewer samples), while the Verilog-AMS model is faster for smaller time steps (higher number of samples, more time spent in the analog part). This reveals that the highly optimized Verilog engine, although it is event-driven, tends to be faster than pre-scheduled SystemC-AMS TDF on intensive analog computations. On the other hand, when the communication with the digital SystemC has more weight, the SystemC-AMS has a slight advantage.

The profiler allows a closer examination of the performance of a simulation. For example, Table 1 shows the time spent in each module in the case of the Verilog-AMS implementation, with a 10 µs time step.

Table I. Time spent in each module, for 30 s of simulated time with a 10 µs time step

Language	Module	CPU Usage (seconds)	CPU Usage (%)
SystemC	LCD_controller	12.38	36.7
	processor_model	11.10	32.9
	other	1.89	5.6
	SystemC kernel	1.73	5.1
Verilog-AMS	magnetic_perturbation	1.17	3.5
	magnetic_field_sensor	1.01	3.0
	motion_model	0.40	1.2
	adc	0.36	1.1
	field_rotation	0.09	0.3
	verilog engine	3.62	10.7
Total		34.00	100.0

This can be further refined to the process and function levels.

E. Using the profiling results

These experiments point out that running multi-language profiling is important to check the assumptions on relative time spent in analog vs. digital parts, and check that modeling level are appropriate for the application. When the simulation time is largely dominated by the analog models, the developer can consider speeding-up the analog part, by a shift in the abstraction level or by an optimization of the level used (e.g. from TDF to Dynamic TDF). When the simulation time is largely dominated by the digital models, there is possibly an opportunity to add more accuracy in the analog models, without sacrificing the overall execution speed. When similar times are spent on both analog and digital, more detailed analysis is required to make incremental optimizations.

Furthermore, profiling results can be used to evaluate the scalability of the models. For example on the magnetic sensor platform, the profiling results suggested that the default time step of 1 ms, realistic for the application, was compatible with the Linux boot on the Zynq virtual platform. That was confirmed by the experience: adding the AMS model to the platform had a negligible impact on the boot time, and the compass software, once transformed into a Linux application, can run on the Zynq virtual platform at full speed.

VI. CONCLUSION

In this paper, we discussed how the extension of the virtual platforms with the AMS domain can bring valuable benefits for the development of embedded software, architecture exploration and system validation. We explained how to make it a reality, by using the AMS languages available today at the appropriate abstraction level, and by leveraging existing tool capabilities such as multi-level debug and advanced multi-language profiling. Tools can certainly be enhanced to offer more specific features for that level of modeling.

The approach was explained and demonstrated on a sensor example which is simple but representative of some questions that awaits the developers of AMS models for virtual platforms.

There are of course some challenges ahead on the road to ubiquitous system-level AMS modeling. The availability of the models of complex analog IPs is likely to be the first obstacle. The digital-only virtual platforms have faced that situation for years, but the relative simplicity of the abstracted models and the overall benefit they brought to the design flow showed that the development of those models is a good investment.

REFERENCES

[1] C. Ebert, C.Jones, “Embedded Software: Facts, Figures, and Future”, in Computer, vol. 42, issue 4, 2009, pp. 42-52

[2] P. Avss, S. Prasant, R. Jain, “Virtual prototyping increases productivity - A case study”, VLSI-DAT, 2009.

[3] “IEEE Standard for Standard SystemC Language Reference Manual”, IEEE Computer Society, 2012

[4] “Verilog AMS 2.4” http://www.accellera.org/downloads/standards/v-ams

[5] “1076.1-2007 - IEEE Standard VHDL Analog and Mixed-Signal Extensions”, IEEE Computer Society, 2012

[6] “Real-value or wreal modelling”, http://www.techdesignforums.com/practice/guides/real-value-wreal-modelling/

[7] “IEEE Std 1800-2012: IEEE Standard for SystemVerilog - Unified Hardware Design, Specification, and Verification Language”

[8] P. Hardee, “Real datatypes and tools enable fast mixed-signal simulation”, http://www.techdesignforums.com/practice/technique/real-wreal-datatypes-fast-mixed-signal-simulation/, 2014

[9] “About the SystemC AMS extensions”, http://www.accellera.org/community/systemc/about_systemc_ams/

[10] M. Barnasconi, “SystemC AMS Extensions: Solving the Need for Speed”, DAC Knowledge center, 2010

[11] F. Cenni, S. Scotti, E. Simeu “A SystemC AMS/TLM platform for CMOS video sensor”, DASIP, 2011

[12] M. Damm, C. Grimm, J. Haas, A. Herrholz, W. Nebel, “Connecting SystemC-AMS models with OSCI TLM 2.0 models using temporal decoupling”, Forum on Specification, Verification and Design Languages. FDL 2008

[13] M. Barnasconi, K. Einwich, C. Grimm, T. Maehne, A. Vachoux “Whitepaper: Advancing the SystemC Analog/Mixed-Signal (AMS) Extensions”, http://www.accellera.org/resources/articles/amsdynamictdf, 2011

[14] S. Scotti, “Multi Physical Domain Applications challenges: Design Flow integration”, ESCUG Grenoble, 2013

[15] “Zynq™-7000 SoC Extensible Virtual Platform”, http://www.xilinx.com/products/zynq-7000/extensible-virtual-platform.htm

[16] S. Swan, J. Cornet, “Beyond TLM 2.0: New Virtual Platform Standards Proposals from ST and Cadence”, NASCUG at DAC, 2012

[17] “Incisive Enterprise Simulator”, http://www.cadence.com/products/sd/enterprise_simulator/pages/default.aspx

[18] “SystemC AMS PoC 2.0 Beta Version”, Fraunhofer IIS, 2013

[19] Z. Wang, D. Zhang, X. Yu, Z. Yu , X. Zeng, “ A fast multi-core virtual platform and its application on software development”, IEEE 10th International Conference on ASIC (ASICON),2013