Integrating a Virtual Platform Framework for Smart Devices

Valerio Guarnieri, Francesco Stefanni, Franco Fummi
EDALab s.r.l. – Strada le Grazie 15 – 37134 Verona, Italy
{name.surname}@edalab.it

Michelangelo Grosso, Davide Lena
ST-POLITO s.c.ar.l. – Corso Duca degli Abruzzi 24 – 10129 Torino, Italy
{name.surname}@st-polito.com

Angelo Ciccazzo, Giuliana Gangemi, Salvatore Rinaudo
STMicroelectronics s.r.l. – Via Franco Gorgone 39/41 – 95121 Catania, Italy
{name.surname}@st.com

***Abstract*—Smart system design flow is nowadays affected by a number of issues. The descriptions of the components of a smart system are extremely heterogeneous and written in a variety of different languages. Additionally, design levels for such components are too low to provide a global view of the whole system, and analog and mixed-signal components are difficult to integrate into higher-level systems. HIFSuite aims at solving these issues by allowing to translate and abstract heterogeneous descriptions of components into a homogeneous C++ description, thus enabling their integration into virtual platforms.**

Keywords—virtual platform integration; smart system; design; simulation

I. INTRODUCTION

Figure 1 shows the typical components of a smart system (blue boxes), and the related tools and languages used for their representation in a CAD environment (green boxes). Three issues are evident:

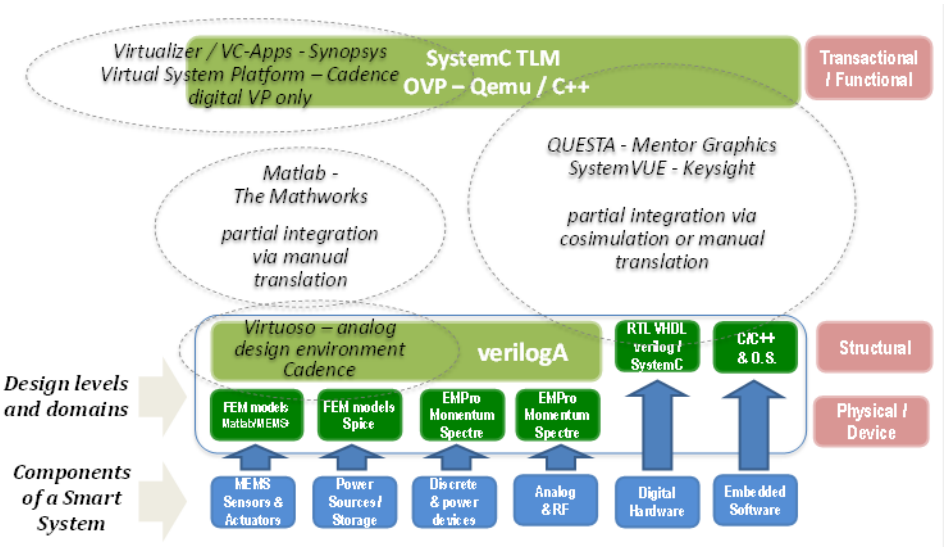

Figure 1. The state of the art heterogeneous and non-integrated modeling/design/simulation platform for smart devices.

1. The representations for the various components are extremely heterogeneous and their design is based on a variety of specialized description languages. The state of the art to deal with such heterogeneity is to employ co-simulation [4][5], which provides poor simulation performance, or manual translation, which is time-consuming and error-prone, without incorporating extra-functional behaviors.
2. Design levels for the heterogeneous components are typically too low to provide a global view of the entire smart system. In fact, design levels are usually physical/device or structural at most, while a global

view of the smart device should be at least functional or transactional. However, this global view is fundamental to check the entire functionality of the system, especially in the case of extra-functional properties. The state of the art solution for this problem is the manual abstraction of the structural and physical models into a corresponding functional description, which is a time-consuming and error-prone process.

3. Analog and mixed-signal components are difficult to integrate into higher-level systems, as they are often very sensitive against environmental influences, statistical process shifts, degradation and stress effects. However, such components are absolutely required to build up sensing and communication interfaces for smart systems. They mostly have to handle the trade-off between circuit performance, specifications and mismatch. For this reason, effective and efficient circuit analysis and automated optimization algorithms are required to handle such multi-dimensional circuit sizing and verification problems with reduced time and effort. Furthermore, intelligent modeling tools are needed to seamlessly integrate circuits within systems under full consideration of these factors. The state of the art solution for this problem relies on manual sizing and optimization through incremental changes on the design, iterative re-simulations and basic verification methods such as brute-force Monte-Carlo with extreme high-effort.

II. TESTCASES

Two testcases provided by ST will be used as demonstrators. These devices are working prototypes, but not marketed products yet, developed in the context of the SMAC FP7 European project [6]. They include:

1. A Modular Sensor Node for the monitoring of human gesture or movement;
2. An Enhanced LED Driver Engine for the implementation of smart lighting modules.

These devices are currently designed with well-defined design flows based on commercial tools that do not provide the necessary quality (i.e., coverage of domains, time-to-market, consideration of non-functional metrics). They provide the typical EDA challenges in the context of system-level design, MEMS-design and analog-mixed signal-design. The newly developed tools will use these testcases as relevant, product-strength, industrial benchmarks to validate their new features.

A. Modular Sensor Node

The first demonstrator is a modular sensor node consisting of a microcontroller, one (or more) sensor, a wireless transceiver and a power source. Depending on the target application, specific sensors can be added to the system, and the other components are selected according to the required computational power (e.g., from ultra-low-power microcontrollers such as STM32Lx to high-performance devices with FPU and DSP such as STM32F4x) and data transmission rate. The system-level model can be used in this case for evaluating functional performance under realistic workloads in terms of accuracy of data sensing and elaboration, achievable data transmission rate, and energy consumption.

Once a specific application is identified, the target system has to be defined and implemented, by using available modules and possibly by designing ad-hoc components. Hardware and software design proceeds in parallel with the development of suitable models, which need to be refined in a top-down approach leading to engineering and industrialization, and abstracted in a bottom-up manner to allow for system-level performance evaluation and optimization.

B. Enhanced LED Driver Engine

The main components of the demonstrator Enhanced LED Driver Engine are microcontroller and power conversion units, wireless transceivers, an optional Power Line Communication (PLC) modem and different kinds of smart sensors and actuators. Furthermore, the demonstrator also contains the firmware for the management of LED, power conversion control and sensors/actuators. The addressed design represents a fully controllable light system, embedding light changing and adaptive features, complex wired or RF communication, sensors and actuators. Thanks to the communication channels, a full remote control of the light engine and the acquisition of the information collected by the adopted sensors is possible.

Having the possibility to adopt a full-system approach, it is expected to have a better understanding of complex sub-systems interactions, which are not easy to be addressed when each sub-module is developed independently. In that sense, several examples can be proposed to highlight the advantages achievable by the usage of a full system simulation/co-simulation environment, but the most relevant are:

- The possibility of deeply investigating the MCU capabilities and peripherals usage. A MCU solution is required to sustain the appropriate workloads in managing LED engine, sensor data acquisition and network communication stacks. This allows to early address performance issues, firmware optimization or to switch to the adoption of dedicated peripherals in case of heavy duties.
- Different power modes, each related to the provided functionalities (LED engine ON/OFF, stand-by modes, sensor acquisition, network activities, etc.), need to be optimized for efficiency in the most relevant configurations. Typically, each adopted sub-system is equipped with its own control and data interfaces, so several heterogeneous protocols and physical communication layers are involved. In order to grant the correct interoperability of each module, considering the possible interferences and incompatibilities, a fully described system is required.

It is easy to understand that many possible interactions have to be evaluated and obviously the possibility to manage most of them at simulation level will allow to anticipate issues during design phase, thus reducing development costs.

III. INTEGRATION TECHNOLOGIES BY HIFSUITE

HIFSuite [2][3] is a set of tools for the automatic conversion and abstraction of HDL descriptions. It allows to translate heterogeneous descriptions of components into SystemC and to abstract them at TLM or into C++ descriptions, thus enabling their integration into virtual platforms. Additionally, the obtained descriptions are optimized for simulation performance.

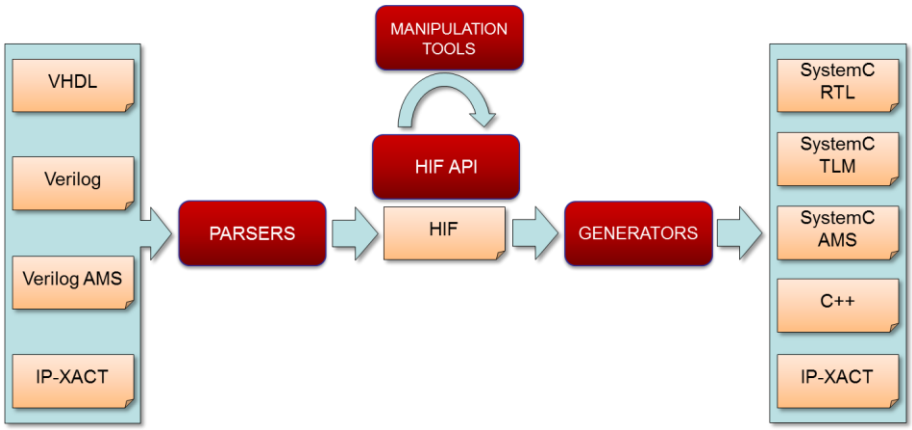

Figure 2. HIFSuite overview.

Figure 2 provides a visual overview of HIFSuite. Parsers are used to convert the original HDL description into the Heterogeneous Intermediate Format (HIF), which is used as internal representation by HIFSuite tools. Generators are used to produce the output SystemC AMS/RTL/TLM or C++ description. Manipulation tools are built upon the HIF core API to manipulate HIF descriptions. The Automatic Abstraction Tool (A^2T) abstracts RTL descriptions into corresponding TLM-2.0 or C++ descriptions, while DDT abstracts original HDL data types into C++ built-in data types to greatly improve simulation performance of the description.

HIFSuite allows to integrate heterogeneous components into a virtual platform by creating a homogeneous simulation platform. Such a platform provides much better simulation performance with respect to the time-consuming co-simulation approach. The SMAC [6] Open Source Test Case (OSTC) showcases the use of HIFSuite to obtain a virtual platform from heterogeneous components.

Figure 3 illustrates the steps composing the flow to generate a transactional-level homogeneous description of a smart system through HIFSuite:

1. Generate the top-level description from the automatically generated IP-XACT interfaces of heterogeneous components. hif2ipxact allows to obtain the IP-XACT descriptions of the heterogeneous component, which can then be integrated to specify their interconnections within the whole system by using an IP-XACT visual editor (e.g., Kactus 2 [7]). Finally, ipxact2hif and hif2sc can be used to generate the top-level description of the system, in terms of instantiation of components and their port bindings.
2. Convert heterogeneous descriptions of components into a homogeneous SystemC AMS/RTL platform. The whole system is now described homogeneously, but simulation will be affected by the poor performance of SystemC and SystemC-AMS simulation kernels.
3. Abstract components to C++ descriptions to improve simulation performance. DDT and A^{2}T are employed to abstract RTL descriptions into highly efficient C++ descriptions, so that simulation can be performed by an optimized automatically generated process scheduler.

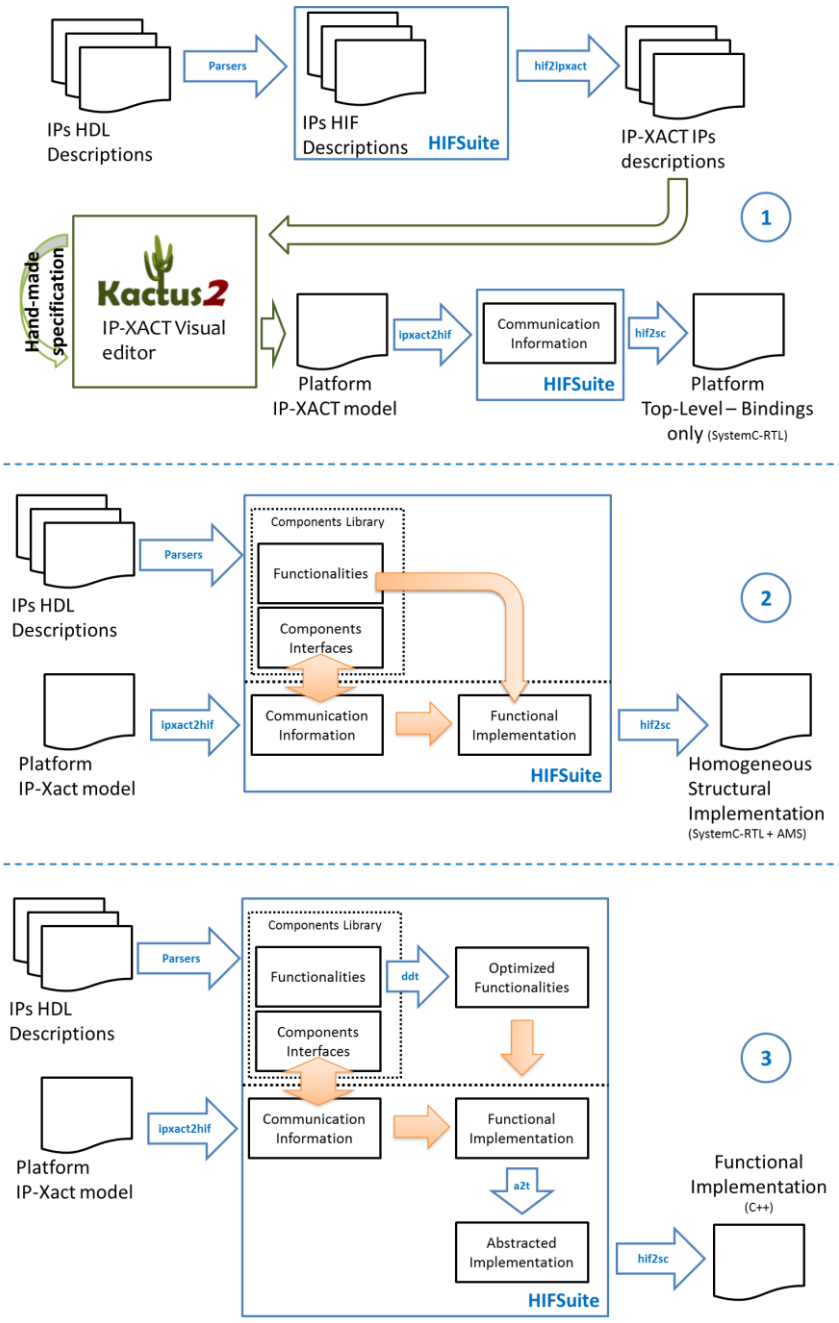

Figure 3: Flow to generate a transactional-level homogeneous description of a smart system through HIFSuite.

IV. APPLICATION TO TESTCASES

The proposed virtual platform framework aims at supporting system integrators and application developers. "Smart system" design space exploration and application development can largely benefit from realistic and fast system-level models including both hardware and software components, which enable the evaluation of system performance and the optimization of application algorithms before a prototype is available.

The following paragraphs describe how the activities related to the development of the demonstrators is mapped to the proposed virtual platform framework for smart system devices.

Figure 4 represents the design domains involved in the design of a MEMS-based sensor nodes (including, e.g., an accelerometer). The "sensor" is actually composed of quite different parts in the same package, i.e., the MEMS transducer, converting the physical stimuli in an electrical quantity such as voltage, capacitance ratio or frequency, an analog front-end applying the required amplification and filtering, and increasingly often an analog-to-digital converter (ADC) and a digital interface. This sub-system is then connected to a microcontroller that handles computation and data transfers and to a radiofrequency transceiver.

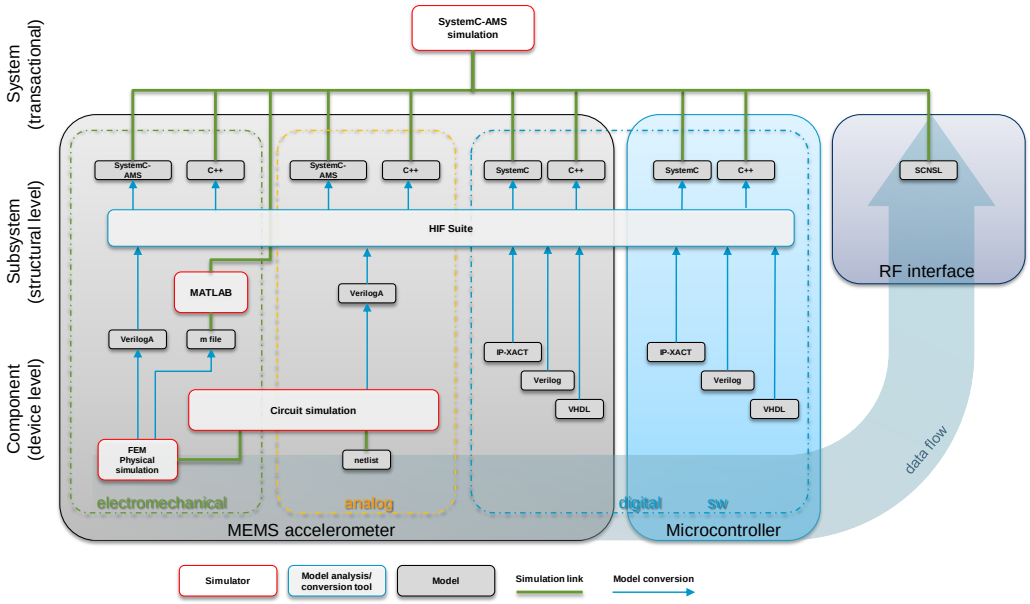

Figure 4. Mapping of a MEMS-based sensor node on the proposed platform.

Depending on the component and involved design domain, on the design phase and on the feature and performance to be evaluated, different models and tools are required to develop the components and the final system. In this case, the electromechanical parts are expressed as MATLAB or VerilogA descriptions. They can be directly used in co-simulation with the other system parts, or SystemC-AMS models can be derived through abstraction to be employed in system-level simulation.

The analog electronics flow starts with the definition of an ideal schematic entry from where a netlist is obtained. From there, the development of the layout is mostly a manual task. The netlist is then back-annotated to provide a precise simulation model for validation and optimization, which, in this case, are described as VerilogA models. HIFsuite can then be employed for generating equivalent SystemC-AMS descriptions.

The digital electronics flow is based on Register-Transfer-Level (RTL) descriptions that are synthesized and converted to netlists and layouts, or can be used for system-level simulation. The complexity of the digital part (e.g., when a microprocessor is involved) can create bottlenecks in system-level simulation. The abstraction to SystemC TLM or C++ performed by HIF Suite greatly mitigates the problem.

Finally, the networking layer, operated by the microcontroller and the RF interface, is modeled at high level relying on the SystemC Network Simulation Library (SCNSL) [1].

For the second demonstrator, the design domains span from digital, to analog, power and RF components (Figure 5). Similarly as in the former case study, the design activity fits within the proposed platform and

especially leverages on the analog electronics flow. The target is to be able to extract suitable high-level models for the system sub-blocks, in order to approach an efficient, reliable and fast system-level simulation.

In addition, to evaluate the system-of-systems performance, a network-level model is required, supporting a number of smart LED driver modules and a set of the possible connected Internet-of-things devices. This high-level model is mapped into the SCNSL at the transactional level.

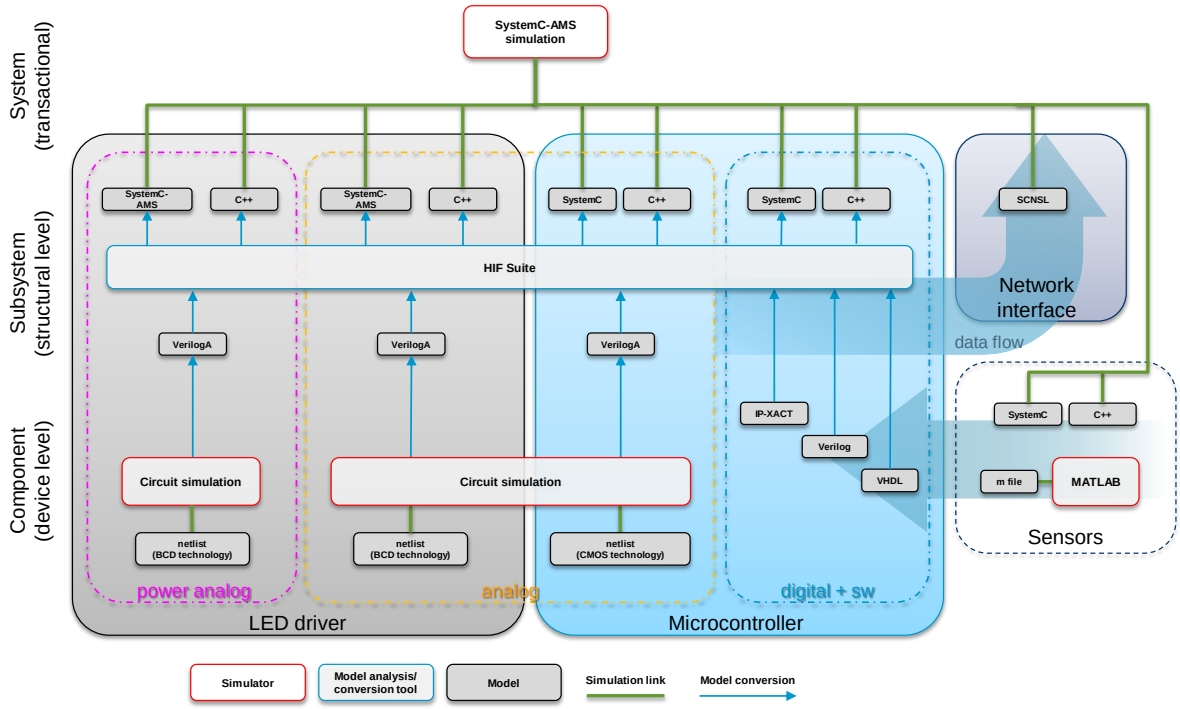

Figure 5. Mapping of an enhanced LED driver engine on the proposed platform.

V. ABSTRACTION RESULTS

A^{2}T and DDT tools in HIFSuite contribute to enhance simulation speed by providing three different kinds of abstraction:

- Process scheduler: the classical HDL process scheduler is replaced with a more performing process scheduler.
- Data types: the original HDL data types are replaced with C++ built-in data types.
- Interface: the original RTL interface is replaced to be compliant with the desired level of abstraction (i.e., TLM or C++).

A^{2}T is responsible for the abstraction of the process scheduler and the interface, while DDT carries out the abstraction of the data types.

Table 1 reports the simulation times for three different versions of each design. Column *SystemC RTL* displays the simulation time for the SystemC RTL description generated by HIFSuite. Column *ModelSim* indicates the simulation time for the original VHDL/Verilog description by ModelSim. Column *Abstracted C++* reports the simulation time for the Abstracted C++ description generated by A^{2}T and DDT. Finally, the last two columns provide the speedup of the abstracted C++ description with respect to the SystemC RTL description and the original VHDL/Verilog description, respectively.

As expected, SystemC RTL simulation is on average one order of magnitude slower than the simulation of the original VHDL/Verilog description by ModelSim. Conversely, the abstracted C++ description provides a speedup of up to three orders of magnitude with respect to the SystemC RTL simulation (Figure 6), and up to two orders of magnitude with respect to the original VHDL/Verilog simulation by ModelSim (Figure 7).

Table 1. Simulation times for the SystemC RTL, the original VHDL/Verilog, and the abstracted C++ descriptions.

Design	SystemC RTL (s)	ModelSim (s)	Abstracted C++ (s)	Speedup vs SystemC RTL (x)	Speedup vs ModelSim (x)
Camellia	26,974.8	1,074.7	3.4	7,933.8	316.1
DES56	7,112.2	790.4	4.3	1654.0	183.8
AES	850.9	67.5	7.1	119.8	9.5
SHA256	4,682.5	152.4	3.4	1,377.2	44.8
SHA512	6,302.1	175.6	5.0	1,260.4	35.1
XTEA	975.2	170.9	3.4	286.8	50.3

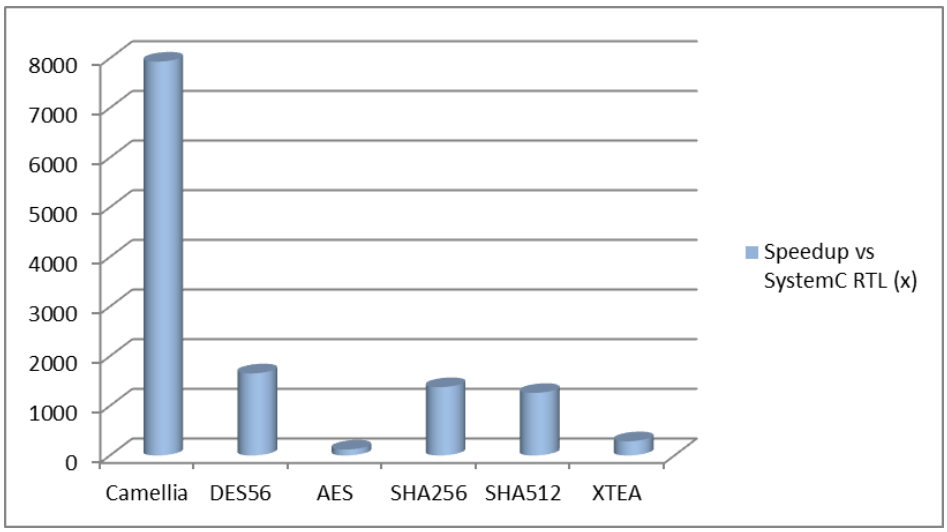

Figura 6: speedup of abstracted C++ descriptions with respect to SystemC RTL simulation.

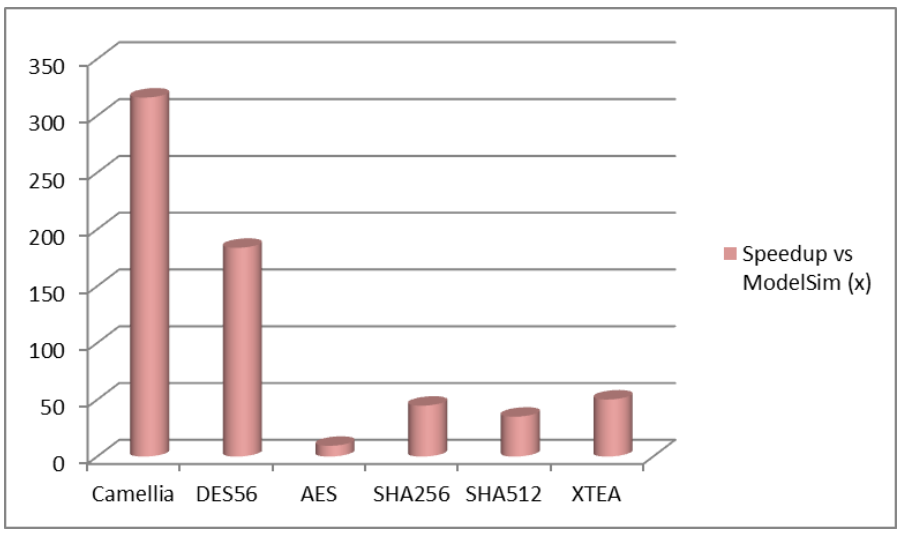

Figure 7: speedup of abstracted C++ description with respect to ModelSim simulation.

VI. CONCLUSIONS

HIFSuite assists and facilitates designers of smart devices by allowing to translate heterogeneous descriptions of components into a homogeneous SystemC description, thus removing the need for time-consuming co-simulation. Such homogeneous description can be then abstracted to a corresponding C++ description to further enhance simulation performance.

The paper described how HIFSuite can be applied to the heterogeneous components of two complex smart systems to generate homogeneous transactional-level SystemC virtual platforms.

Additionally, designers can reuse already existing IPs and integrate them into virtual platforms for design space exploration, design validation and performance evaluation.

VII. RELEVANCE

- Reduction of time to market: the use of HIFSuite tools allows designers to accelerate design space exploration, design validation and performance evaluation with respect to what is achieved with traditional techniques. Smart system integrators can easily work with realistic yet relatively fast models of components and systems deriving from detailed representations, thus avoiding (or at least reducing) possibly expensive iterations of prototyping and experimental characterization.
- Reuse of already existing IPs: HIFSuite allows designers to reuse existing IPs and to integrate them into virtual platforms.
- Fast simulation: HIFSuite generates abstracted descriptions which enhance simulation performance and prevent from having to resort to time-consuming co-simulation.

REFERENCES

[1] F. Fummi, D. Quaglia, F. Stefanni, "A SystemC-based framework for modeling and simulation of networked embedded systems," Proceedings of IEEE Forum on Specification, Verification and Design Languages (FDL) 2008, pp. 49-54.

[2] N. Bombieri, M. Ferrari, F. Fummi, G. Di Guglielmo, G. Pravadelli, F. Stefanni, A. Venturelli, “HIFSuite: Tools for HDL Code Conversion and Manipulation”, EURASIP Journal on Embedded Systems, 2010.

[3] HIFSuite website: www.hifsuite.com

[4] M. Bombana, F. Bruschi, "SystemC-VHDL co-simulation and synthesis in the HW domain", Proceedings of Design, Automation and Test in Europe Conference and Exhibition 2003, pp. 101-105.

[5] R. Maciel, B. Albertini, S. Rigo, G. Araujo, R. Azevedo, "A Methodology and Toolset to Enable SystemC and VHDL Co-simulation," IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2007, pp. 351-356.

[6] SMAC (Smart components and smart systems integration) FP7 European Project website: http://www.fp7-smac.org/.

[7] Kactus2 website: http://funbase.cs.tut.fi/.