Power-Aware Wrappers for Transaction-Level Virtual Prototypes: a Black Box Based Approach

Ons Mbarek, Alain Pegatoquet, Michel Auguin
LEAT, University of Nice Sophia Antipolis-CNRS
Bâtiment Forum, Campus Sophia@Tech
930-Route des Colles, Sophia Antipolis 06903, France.
{mbarek, pegatoquet, auguin}@unice.fr

Houssem Eddine Fathallah
LEAT, University of Nice Sophia Antipolis-CNRS
Bâtiment Forum, Campus Sophia@Tech
930-Route des Colles, Sophia Antipolis 06903, France.
{fathallah}@etu.unice.fr

***Abstract*—Low power design and verification at the Electronic System Level (ESL) have recently emerged as a challenging research field. This work[1] presents a reliable solution to add such capabilities to Transaction-Level virtual prototypes composed of black-box hardware Intellectual Properties (IPs). This solution relies on a wrapper-based approach in which power intent specification and verification are added as separate layers of the IPs functional ones. Using Synopsys's Innovator™ virtual prototyping toolset, our approach has been validated with an audio application TL platform. Results show our approach benefits to enable different power intent alternatives exploration with low simulation speed overhead and reduced modeling effort.**

Keywords-Intellectual Properties (IPs), low power design and verification, virtual prototyping (VP), Transaction Level Models (TLM), Power Domain (PD), power intent, System-on-Chip (SoC), Design-by-Contract (DbC), Unified Power Format (UPF) standard.

I. INTRODUCTION & RELATED WORK

One of the key Electronic System Level (ESL) design methodologies is Transaction-Level virtual prototyping (VP) solutions. These solutions such as the Synopsys Innovator™ tool [1], the Mentor Graphics Vista™ solution [2] or the Imperas Open Virtual Platforms (OVP™) environment [3], rely on building software models of embedded systems through assembling SystemC Transaction-Level (TL) [4] Intellectual Property (IP) cores. Depending on the used virtual prototyping tool, these cores can be either white-box IPs with accessible source codes or black-box ones already pre-designed, pre-compiled and pre-verified. For instance, unlike OVP™ which provides an open source library composed of white-box IP models, an Innovator™ VP is usually composed of black-box IPs taken from the Synopsys DesignWare System-Level Library (DWSLL) [5].

Power-aware TL virtual prototyping have recently gained great interest. At this level, rapid exploration of architecture design alternatives can be made. So, greatest and costless power reduction can be achieved.

On the one side, EDA vendors, industrials and academics have supplied tools and methodologies that ease system-level power modeling, analysis and optimization. Most of them focus on enabling power estimation and analysis inside existing VP tools using annotation-based approaches [6] [7]. Authors in [8] [9] [10] [11] have proposed approaches for source code instrumentation targeting only white-box types of IPs. So far, validation of efficient and well-structured power management strategies according to a specific low power design has not been targeted by state of the art approaches and existing VP tools. This is due to the lack of system-level *power intent* specification support [12]. *Power intent* refers to the design's power domains partitions, supply network distribution, retention strategies and system power modes. Among additional common shortcomings of these mentioned ad hoc approaches are the lack of modularity and the clear separation of power and functional concerns. On the other side, some EDA tool vendors have come up with automation solutions for low power design, verification and exploration. Most of these solutions [14] [15] are built on support for the recent IEEE-1801™ Unified Power Format (UPF) industry-standard [13]. This standard enables power intent specification and verification along the design flow from Register Transfer Level (RTL) to signoff.

In our previous work [16], we have proposed a methodology to add power intent and management features to Transaction-Level functional models based on abstract UPF standard concepts and check relevant power-aware properties. An efficient approach to instrument white-box TL IPs with these capabilities according to this methodology has been also proposed. However, this approach cannot be applied to black-box TL IPs due to some modeling constraints. An approach to handle the particular case of black-box IPs while applying the same methodology is hence required.

The work shown in this paper contributes to define a modular approach that applies our methodology to add low power design, management and verification features to TL platforms composed of black-box IPs. In this approach, the IEEE 1801 (UPF) standard [13] has been used as a support. Separation of power and functional concerns is achieved by wrapping power-aware features on top of black-box IPs.

In the sequel, section 2 briefly describes the main goals and stages of our power-aware TL design methodology. Section 3 introduces some constraints to apply this methodology on black-box types of virtual platforms. Section 4 explains our wrapper-based approach. In the fifth

[1] This work is supported by the French National Research Agency (ANR) project HELP bearing reference ANR-09-SEGI-006

section, we evaluate our approach on an audio system virtual prototype using the Synopsys Innovator VP tool. Section 6 closes with conclusions and future works.

II. A POWER-AWARE TRANSACTION-LEVEL DESIGN METHODOLOGY

Starting from a functional and timed Transaction-Level model of a SoC, our methodology mainly aims at adding power management capabilities to this model. First, at the **power intent specification stage,** the designer starts by specifying a power intent alternative for the TL-model using the main concepts of the IEEE-1801 UPF standard [13]. Among these concepts are power domains partitioning, power distribution (supply networks and power switches), system power modes and legal transitions among them. A system power mode is a combination of power domains states, where each power domain state is defined by its primary supply nets states. The PST, a central concept in the UPF standard, summarizes the system power management policy. This stage is based on the software flow analysis in order to determine power intent alternatives. An example of a power intent alternative is depicted by Fig. 4 (Alternative (a)) and Fig. 5.

Then, at the **PMU modeling stage**, the Power Management Unit (PMU) is modeled to manage the different system power modes at runtime. It dynamically sets the requested system power mode through adequate adjustments of the power architecture status (e.g. supply nets' voltages and power switches states) as defined in the PST. Our PMU model includes a Domain Power Controller (DPC) for each power-gated domain in charge of changing the domain power state between sleep and wake-up. It also includes a Power Manager (PM) module used to set request DPCs to change their power domains states according to the requested system power mode. Requests to the PMU for setting a specific power mode consist in power control transactions (PCTr) added to the embedded application.

Finally, at the **full power-managed simulation stage**, the power-managed behavior of the TL-model is simulated and power equations are recursively updated as soon as a change in a domain state is detected. Power values of each IP are taken either from datasheets or low level simulations.

A **power-aware and contract-based verification stage** occupies an orthogonal position. By running the simulation after each previously mentioned stage, the power-aware verification process is activated to check specific coherence properties between added power features and existing functional ones. Each of these properties belongs to a contract class. Four classes of contracts have been identified. Each class gathers all possible precondition (assume) and postcondition (guarantee) properties between two specific types of components**. Class 1 contracts** checks correctness of the power architecture structure. **Class 2 contracts** checks the correct functionality of the PMU. **Class 3 contracts** checks coherence between the functional design and the power objects. Noting an activity in a powered-down domain is an example of a violated class 3 contract. **Class 4 contracts** checks coherence between the functional design and the PMU activity. More details on our contracts and methodology can be found in [16].

III. TRANSACTION-LEVEL BLACK-BOX IP MODELS: FEATURES & CONSTRAINTS

In this section, we explain how specific features of a TL black-box IP constrain the application of our methodology on a TL platform with black-box IPs.

A. TL Black-Box IP Main Features

The black-box basic feature is the limited observability of internal state changes of IP cores. However, without looking inside a black-box IP, the developer can in most cases determine its behavior. Actually, most black-box IP cores are software-configurable and their operational status can be determined through capturing and analyzing exchanged transactions at their interfaces. For instance, read or write transactions to memory-mapped control and status registers (CSRs) may give information about current operations of this IP. In addition, IP vendors offer minimum information, not only about the IP interface signals, but also about memory-mapped registers of each IP (e.g. description of their offset and bit fields' access). This kind of information is mandatory for the embedded software developer to correctly configure and use a black-box IP. In particular, only memory-mapped registers of a black-box IP are usually public in virtual prototyping tools so as to facilitate the debug of a packaged and distributed IP.

B. Constraints on Power Intent Specification & Simulation

Power-aware behavior may alter the IP initial functionality. We believe that the specified state retention strategy may alter the IP functionality if not well chosen. In this paper, the retention-register approach based on replacing a standard register with a retention register is used. In a retention register, state is locally preserved during power-down and restored at power-up [12]. All non-retained registers must be initialized during power-down, so that they power up in the reset condition. A block state can be fully retained (i.e. all its registers are replaced with retention ones). However, partially retaining the IP state reduces area penalty and is almost efficient [12]. At Transaction-Level of modeling, simulation of partial state retention requires only resetting non-retained registers of an IP during its power-down while states of retention registers remain untouched. In addition, only memory-mapped registers represent possible candidates of non-retained registers in a black-box IP. This is due to the public access given only for this type of registers. So, resetting these registers from outside the black-box IP while powering down is possible. Remaining registers such as internal memories and buffers are usually made private with no access from outside the black-box IP. So their state cannot be changed.

C. Constraints on Power-Aware Contract-Checking

The major constraint related to the power-aware verification stage is that power-aware assertions cannot be embedded into a black-box IP source code. In the TL white-box IP case, atomic operations (i.e. non-interruptible) can be surrounded by class 3 and 4 checks included in the IP source code.

As an example of class 3 precondition properties, an IP operation can only be performed if a specific register state has been retained during the last power-down (P_1). In the black-box version, two constraints can be faced for (P_1) checking. First, this check is only possible if this operation can be accessed from outside the IP. Otherwise, if the beginning of this operation can be identified through capturing transactions to a specific memory-mapped register at the IP interface, (P_1) can be placed before receiving such transactions. Second, states of only memory-mapped registers can be checked from outside the black-box IP. These constraints limit the number of power-aware properties that can be verified in case of a black-box IP and impose a particular checking method.

So, how our methodology can be applied on TL platforms including black-box IPs while taking these constraints into account?

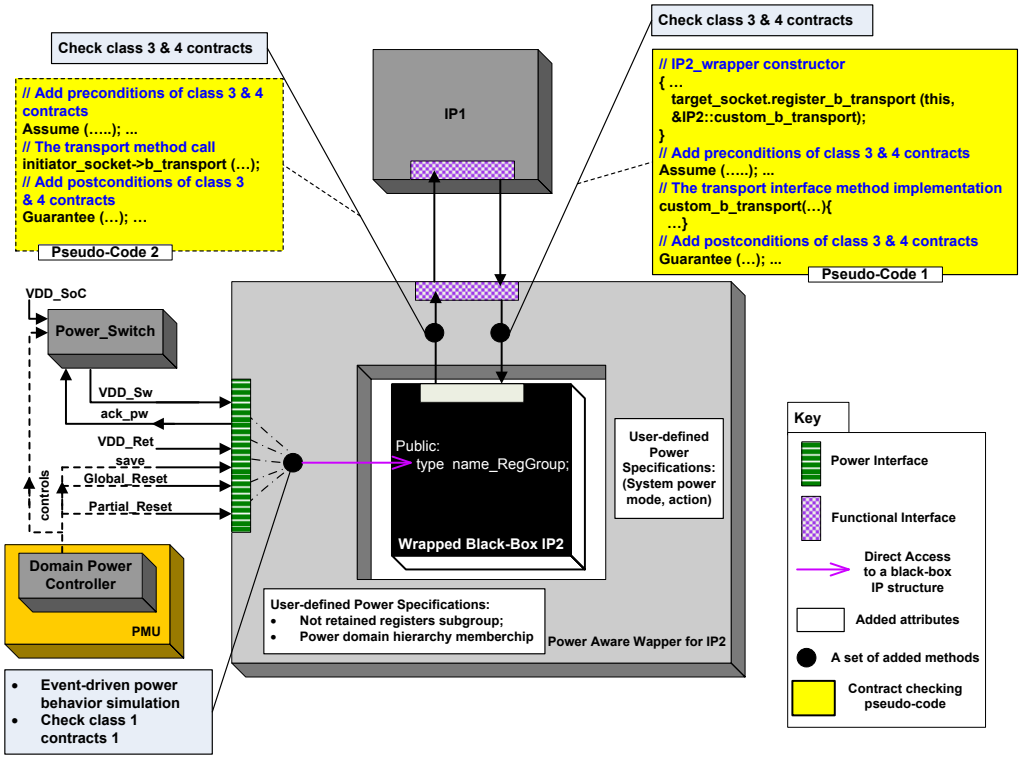


Figure 1. Structure and behavior of a slave/master IP's power-aware wrapper

IV. A Power-Aware Wrapper-Based Approach

The proposed approach consists in encasing each black-box IP of a platform in a power-aware wrapper. By using this approach required power-aware features are not hardcoded into the IP component but are rather layered on top of it. Hence, power and functional concerns of an IP are separated. In the following, the main features of this specialized layer are presented.

A. Power-Aware Wrapper Features

A power-aware wrapper has two main features: the first is to specify power intent for the wrapped IP. The second consists in checking the relevant power contracts properties. Fig. 1 depicts the general structure and features of a power-aware wrapper.

1) Power Intent Specification and Simulation

A power-aware wrapper includes power intent, as well as mechanisms for simulating power-aware behavior of the black-box IP. It provides a power interface that connects the wrapped IP to the power management unit (PMU) as illustrates Fig. 1. It also allows modifying the internal state of the wrapped IP as soon as changes occur on the power interface.

A power interface contains at least a voltage signal (e.g. VDD_Sw on Fig. 1) representing the IP primary power net. It can also include a retention voltage signal (e.g. VDD_Ret on Fig. 1) which supplies retention registers of the wrapped IP during power-down. This interface gathers also control signals handled by the power controller of the wrapped IP. For instance, in case of applying a partial retention strategy, retention registers are saved on power-down (e.g. when the save control signal on Fig. 1 is asserted) and restored on power-on. However, non-retained registers are initialized on power-down. As depicts Fig. 1, simulation of this behavior is done by only resetting the non-retained registers once a partial reset signal is received. Remaining registers that must be retained are not touched. For that, definition of the non-retained registers and their characteristics such as their default value, offset and bit fields access inside the wrapper code is required. As we merely suppose having direct access to memory-mapped registers of the black-box IP, each of these defined registers points to its corresponding register in the wrapped IP. In this way, they are effectively changed to their reset value inside the encapsulated IP code on power-down.

A power-aware wrapper also ensures event-driven power-aware behavior simulation and estimation. For that, we have added methods into the wrapper's code as depicts Fig. 1. These methods are called when changes occur on the power interface. Each method is in charge of handling input signals

(e.g. VDD_Sw) as well as power-down and power-on sequencing by notifying specific events.

2) Power-Aware Contract Checking

Our proposed power-aware wrapper plays the role of a "checking" wrapper. Checking concerns only the four-class power-aware contracts. As a solution for power-aware contracts checking constraints earlier explained, we have duplicated the functional interface of a black-box IP within the wrapper. The goal is to capture the beginning and the end of some IP operations and surround them with assume and guarantee assertions. Assume assertions are used to check the precondition part of a contract. However, guarantee assertions are used to verify the post-condition part of a contract.

As depicts Fig. 1, a power-aware wrapper provides a functional interface which is similar to the one used by the black-box IP. Semantically, they differ on how they behave when either a precondition or a postcondition of an invoked operation is violated. A two-way checking wrapper has been modeled: it reports both its client and wrapped IP interface violations. Clients represent IP blocks communicating with the black-box IP through invoking its public operations.

In the TLM context, the wrapper functional interface mainly consists of TLM ports and interrupts which allow the wrapped IP blocks to communicate with other blocks of the platform [4]. So, before conveying relevant transactions to their destination, the wrapper is designed to intercept them at its functional interface and check appropriate power properties. For that, it implements contracted interface method calls inside the wrapper. Again, in the TLM context, communication can only be established through calls to the TLM transport interface methods (b_transport() or nb_(fw/bw)_transport() methods) [4]. Clients may hence represent slaves for the layered black-box IP. In this case, contract-checking code must be placed around the call to transport interface methods inside the wrapper as illustrated by the pseudo-code 2 in Fig. 1. However, clients may also represent masters for the layered IP. In this case, power contract-checking code must be placed around the transport interface methods implementation inside the wrapper as illustrated by the pseudo-code 1 in Fig. 1. It is worth mentioning that only class 3 and 4 contracts are checked at this level. For instance, when IP1 communicates with IP2 through a transport interface method call, the IP2 power domain must already be powered-on. This is an example of a class 3 precondition that must be checked using an assume assertion at the wrapper functional interface, before entering the transport method implementation in IP2.

B. A Base Wrapper Class for Modularity and Reuse Support

Our wrapper-based approach can be applied whatever the IP functional behavior. Therefore, it clearly separates functional and power concerns of each IP. We have defined a base *Wrapper* class with a generic structure and behavior of a power-aware wrapper. Each IP wrapper is hence a subclass of the *Wrapper* class. When created, it points to the IP to wrap and a list of enabled preferences, denoted *PwPrefs* is defined. This list indicates the basic options

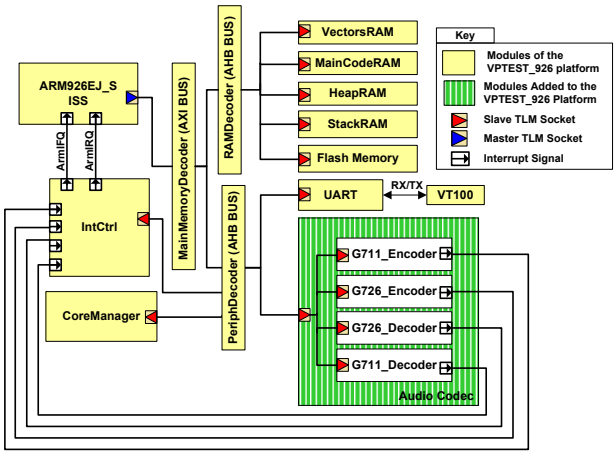


Figure 2. The audio virtual platform block diagram

enabled inside a power-aware wrapper. Examples of these options are the enabling of preconditions or/and postconditions checking, the creation of power-aware wrappers and the use of partial or full retention strategies. The *PwPrefs* mechanism allows selective enabling of the wrapper's capabilities without editing its source code. As a consequence, the refinement and reuse of an IP power-aware wrapper to explore different power intent alternatives for a given virtual platform is made simpler.

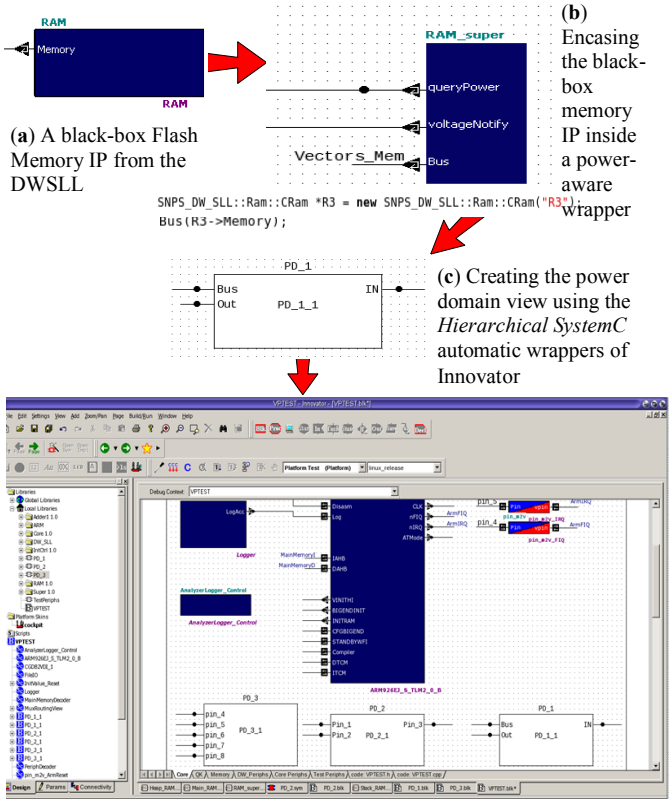
V. APPLICATION TO AN AUDIO SYSTEM VIRTUAL PROTOTYPE

In this section, we demonstrate how our power-aware methodology can be easily integrated into an existing virtual prototyping tool. As the Synopsys's Innovator™ tool offers black-box types of virtual prototypes, we have used it to also validate our wrapper-based approach.

A. A Transaction-Level Virtual Platform for Audio Codec System

An existing software virtual prototype in the Synopsys DesignWare® System-Level Library (DWSLL), named "Timed_926" has been chosen as a starting point for building an audio application. As depicts Fig. 2, the "Timed_926" platform is an approximately-timed (AT) [4] TL platform based on an Instruction-Set Simulator (ISS) for the ARM926EJ-S processor and incorporating black-box TL IP models from the DWSLL. A detailed description of memory-mapped registers, as well as interfaces of each block is given. Each block can be configured through editing the ARM embedded software.

The audio virtual platform has been built on top of the "Timed_926". It models a voice messaging system which mimics a phone answering machine. As illustrates Fig. 2, an audio encoder/decoder hardware accelerator based on the G.711 (Pulse Code Modulation (PCM)) and the G.726 (Adaptive Differential Pulse Code Modulation (ADPCM)) speech codecs ITU-T standards [17] has been added. This accelerator is composed of four TLM sub-modules. On the one side, the G.711 encoder module creates a 64 kbit/s bitstream from an analog signal sampled at 8 khz. The G.711



(**a**) A black-box Flash Memory IP from the DWSLL

(**b**) Encasing the black-box memory IP inside a power-aware wrapper

(**c**) Creating the power domain view using the *Hierarchical SystemC* automatic wrappers of Innovator

(**d**) Integrating the power domains views in the considered platform

Figure 3. Developing power wrappers using the Innovator tool

decoder does the opposite. On the other side, the G.726 encoder encodes into 5, 4, 3 or 2 bits per sample the 64kbit/s bitstream. The G.726 decoder implements the reverse procedure. These modules, created with the Innovator's Component Creator tool, are included in the Synopsys DesignWare® System Level Library (DWSLL) for easy reuse. The ARM embedded application has been enriched with different application scenarios: record a voice message and play a recorded message or an incoming one.

B. *Power-Aware Wrappers Development*

Using the Synopsys's Platform Analyzer tool, activity profiles of each hardware component for each application scenario can be captured and analyzed in order to determine power intent alternatives. Then, each power intent alternative is elaborated by **(1)** placing the power switch modules, **(2)** creating and **(3)** parameterizing IP power-aware wrappers **(4),** implementing the power state table (PST) header file, **(5)** implementing the power management unit by adding to it **(6)** the required domain power controllers, **(7)** enriching the embedded application with power control transactions according to the defined PST and finally, **(8)** creating the power domains view using the *Hierarchical SystemC* Innovator wrapping capability.

For each IP in the virtual platform, its power-aware wrapped version (consisting in the IP itself encased in its power-aware wrapper) is created only once using the Component Creator tool and instantiated henceforth from the DWSLL whenever needed. The last step (step (8)) serves only to group wrapped IPs belonging to the same power domain in order to better structure the low power design. For example, Fig. 3 shows steps required to build the flash memory power domain (PD_1) starting by adding a power-aware wrapper to the DWSLL IP and ending with integrating PD_1 into the initial virtual platform. As a consequence, evaluating a new power intent alternative requires redoing only steps (3), (4), (5), (7) and (8). It is also worth mentioning that a power switch IP, as well as a generic domain power controller IP are created only once using the Component Creator tool. They are afterwards instantiated from the DWSLL whenever needed.

C. *Experimental Results*

Fig. 4 depicts power domains partitioning and supply network of the different power intent alternatives applied to the audio system VP. Only the power state table of alternative (a) is given in Fig. 5. In alternative (c), the four audio sub-modules belong to the same power-gated domain. In alternative (a), similarly to decoder sub-modules, encoder

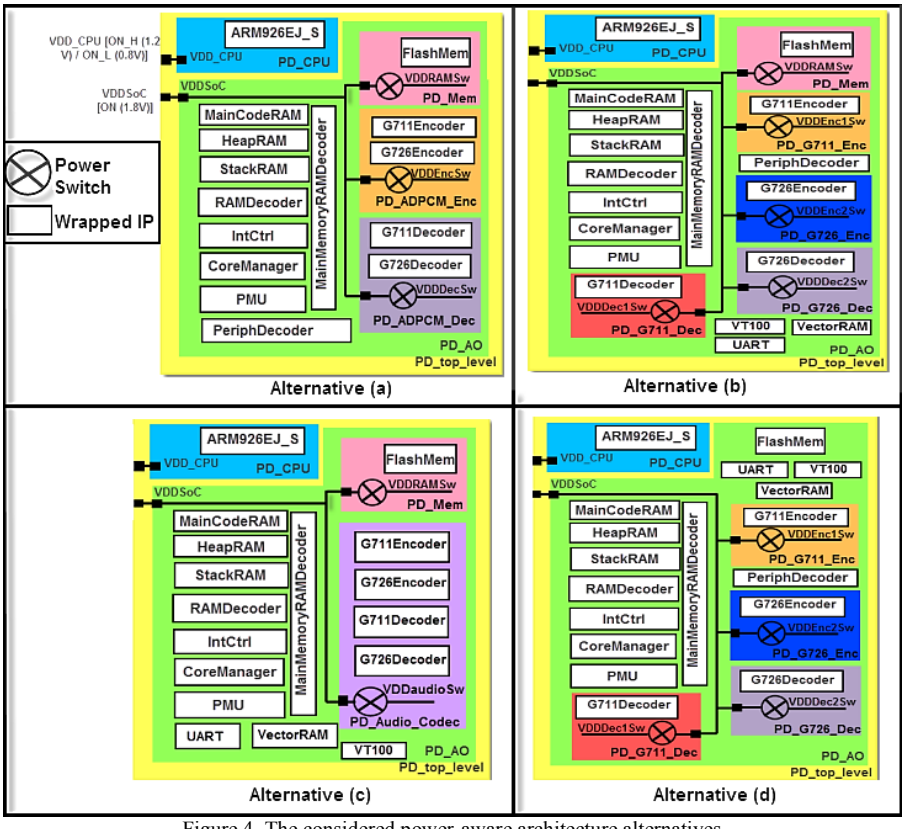


Figure 4. The considered power-aware architecture alternatives

	VDD_CPU	VDD_SoC	VDDRAMSw	VDDEncSw	VDDDecSw
all_off	OFF	OFF	OFF	OFF	OFF
init	ON_H	ON	ON	OFF	OFF
config_ADPCM	ON_H	ON	OFF	ON	ON
record	ON_L	ON	OFF	ON	OFF
play	ON_L	ON	OFF	OFF	ON
transfer_record	ON_H	ON	ON	ON	OFF
transfer_play	ON_H	ON	ON	OFF	ON
idle	ON_L	ON	OFF	OFF	OFF

Figure 5. Power state table for alternative (a)

TABLE I. Energy savings for the different power intent alternatives according to the **Play & Record** software scenario

Power Intent (PI)	Number of Power Switch State Transitions					Time Switching Penality = 200 ns	
	Flash Mem	G711. Enc	G.726 Enc	G.711 Dec	G.726 Dec	Energy Value (J)	Energy Savings (%)
PI (a)	26373	6		6		7,23	20,81
PI (b)	26373	3	3	3	3	6,32	30,8
PI (c)	26373	2				9,03	1,1
PI (d)	0	3	3	3	3	4,3	53
Without Power Intent	Energy = 9,13 (J)						

sub-modules are gathered in a single power-gated domain. In alternative (b) and (d), each audio sub-module belongs to a separate power-gated domain. Unlike the other alternatives, the flash memory IP-block in alternative (d) belongs to an always-on power domain.

TABLE I. shows results obtained for a Play & Record scenario. Note that alternative (d) is the most energy-efficient one since it provides up to 53% of energy savings compared to the non-partitioned initial platform. Note also that power intent alternative (d) achieves up to 32% of energy savings compared to alternative (b). This is due to the considerable power penalties caused by the frequent transitions of the Flash IP in alternative (b) from power-off to power-on (up to 26373 transitions on TABLE I). It is also worth mentioning that using power-aware wrappers adds a negligible amount of simulation run-time overhead. For instance, simulation speed for alternative (d) is only 0.02% less than the initial behavioral platform. Moreover, in order to evaluate new power intent alternative, redesigning and rebuilding power-aware wrappers and power management blocks is only a matter of hours.

VI. CONCLUSIONS

We have presented a wrapper-based approach as a solution to apply a well-structured methodology for adding power intent, management and verification features to black-box IPs of a virtual platform. Modularity and reuse of this approach can be achieved using our guidelines for modeling structure and behavior of a Transaction-Level *power-aware wrapper*. The efficiency of our wrapper-based approach in terms of enabling fast exploration of different power intent alternatives with reduced modeling effort has also been proved. Our approach can also support the Common Power Format (CPF) industry standard [19]. Managing retention of a black-box IP internal registers from outside the IP represents a serious limitation for power intent alternatives specification for a black-box IP. To address this issue, IP-XACT [18] standard could be extended to best provide constraints on an IP block power intent.

ACKNOWLEDGMENT

We gratefully acknowledge the Synopsys's senior software engineer, Denis Paterson, for providing us technical support and recommendations to use the Innovator's toolset.

REFERENCES

[1] Innovator: data sheet, http://www.synopsys.com/Tools/SLD/VirtualPlatforms/Pages/Innovator.aspx.
[2] Vista data sheet, http://www.mentor.com/esl/vista/upload/vista-architect-ds.pdf
[3] OVP website, http://www.ovpworld.org/
[4] Open SystemC initiative. SystemC Transaction Level Modeling Library 2.1.0, 2009. http://www.systemc.org.
[5] Synopsys DWSLL, http://synopsys.mediaroom.com/index.php?s=43&item=490
[6] Y. Veller, and S. Matalon, "Why you should optimize power at the electronic system level", Mentor Graphics Datasheets.
[7] *M. Willems, and S. Tennent,* "Accelerating the Development of TLM-2.0 Models Using Model Authoring Kits (MAKs)", *Synopsys Inc.*
[8] N. Dhanwada, R. A. Bergamaschi, W. W. Dungan, I. Nair, P. Gramann, W. E. Dougherty, and I. Lin, "Transaction-level modeling for architectural and power analysis of PowerPC and CoreConnect-based systems", Design Automation for Embedded Systems, Vol. 10, No. 2-3, September 2005, pp. 105-125.
[9] I. Lee, H. Kim, P. Yang, S. Yoo, E.-Y. Chung, K-M. Choi, J-T. Kong, and S-K. Eo, "PowerViP: Soc Power Estimation Framework at Transaction Level", Proc. Of the 11th Asia and South Pacific Design Automation Conference (ASP-DAC), Japan, 2006, pp. 551–558.
[10] R. Ben Atitallah, S. Niar, and J. L. Dekeyser, "MPSOC Power Estimation Framework at Transaction Level Modeling", Proc. of the 19th International Conference on Microelectronics (ICM), Egypt, 2007, pp. 245–248.
[11] H. Lebreton, P. Vivet, "Power Modeling in SystemC at Transaction Level, Application to a DVFS Architecture", Proc. of the 2008 IEEE Computer Society Annual Symposium on VLSI, France, 2008, pp. 463–466.
[12] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, "Low Power Methodology Manual: for System-on-Chip Design, integrated circuits and systems", Springer, 2007.
[13] Unified Power Format (UPF 2.0) Standard: IEEE standard for design and verification of low power integrated circuits. IEEE 1801TM, March 2009.
[14] Cristal Bulb, magillem site : http://www.magillem.com/eda/index.php?option=com_content&view=article&id=136&Itemid=119
[15] Docea's Aceplorer: http://www.doceapower.com/products-services/aceplorer.html
[16] O. Mbarek, A. Pegatoguet, and M. Auguin, "A methodology for power-aware transaction-level models of systems-on-chip using UPF standard concepts", Proc. of the 21st international conference on integrated circuit and system design: power and timing modeling, optimization, and simulation (PATMOS), Madrid, September 2011, pp. 226-236.
[17] ITU-T website: http://www.itu.int/itu-t/recommendations/rec.aspx?id=10651
[18] IEEE standard for IP-XACT standard structure for packaging, integrating, and reusing IP within tool flow. IEEE Std 1683TM, 2009.
[19] S. I. Initiative, Common Power Format (CPF) 1.1 Specification. Silicon Integration Initiative (Si2), Inc., 2008.