Early-Stage Verification of Power-Management Specification in Low-Power Systems Design

Dominik Macko, Katarína Jelemenská, Pavel Čičák
Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava, Slovakia
dominik.macko@stuba.sk, katarina.jelemenska@stuba.sk, pavel.cicak@stuba.sk

***Abstract*—Power consumption becomes a dominant problem in current hardware-systems design. It is most commonly dealt with use of power-management techniques, such as clock gating, power gating, or voltage and frequency scaling. In modern complex systems, power-management adoption is difficult to achieve, and therefore new approaches to simplify power-managed systems design are evolving. We have also proposed such an approach, simplifying power-management specification at the system level of design abstraction. This paper describes the proposed verification approach, which can take place continuously, beginning at the early specification stage of the system development. It helps a designer to create correct and consistent specification of power management.**

Keywords—hardware design; low power; power management; specification; verification

I. Introduction

In current complex systems, advanced power-reduction techniques are usually applied using some additional specification form, such as UPF (Unified Power Format) [1]. It enables a designer to introduce power-management aspects to the functional design, usually modelled in some HDL (Hardware Description Language). It was intended for RTL (Register-Transfer Level) and lower-level models. Although UPF has significantly helped designers to develop low-power systems, it is not suitable for modern design processes starting at the system level of abstraction. Therefore, new methods and methodologies have been developed to extend low-power design to the system level, such as [2]-[8]. Based on the analysis of their strengths and weaknesses (e.g. insufficient abstraction, missing automation, separated specifications), we have proposed a novel methodology for low-power systems design. It is based on the specification of abstract power management at the system level, and on the application of high-level synthesis to obtain an RTL model [9]-[11].

This paper focuses on the verification of the introduced power-management aspects at early stages of the design. In Section II, the related work is analyzed. Section III provides a brief background of the power management design strategy. In Section IV, the proposed verification approach is described. The experimental evaluation of the contributions is provided in Section V. And finally, Section VI concludes the paper.

This work was partially supported by the Slovak Science Grant Agency (VEGA 1/0616/14) and Slovak University of Technology in Bratislava.

II. Related Work

Most of the existing methods and methodologies, oriented towards higher abstraction levels, target power-management introduction into the system design in an abstract manner. However, there are some which also target the verification problem accompanying the power-management specification. The verification methods and approaches used in the existing solutions are briefly analyzed in this section.

The methodology described in [2] is oriented towards the abstraction of several UPF concepts to the transaction level. It enables power-architecture exploration at the system level; however, the power information must be manually annotated to the model. The used power-aware verification is based on assume and guarantee assertions, which are able to report error notifications during simulation. The assertions are generated automatically and are hidden to the designer. The disadvantage is that this kind of verification checks only whether the interaction between components is correct. It does not provide any information about the completeness of the power-management specification. Since it depends on simulation-based verification, it is time consuming.

There are other methods [3]-[7], which are also focused on simulation-based verification only. Although the offered virtual prototyping speeds-up the simulation compared to the RTL functional verification, there are other aspects (such as specification completeness, consistency with functional model) which are not verified. The method used in [8] does not verify power management at the system level; rather it uses system-level simulation traces to analyze power at the RTL. This analysis involves functional verification of the model. In modern complex systems, the designers cannot rely on simulation to properly verify the design. The verification must be combined with formal methods to address those aspects which are not verifiable by simulation (e.g. completeness).

The method used in [12] uses a more formal approach. The lower-level inter-domain assertions regarding control-signal sequences are automatically translated into a form which enables their usage in the abstract architectural properties of the system. These properties can be then proved formally. The proposed method is indisputably useful; however, it still does not focus on a comprehensive verification of the introduced power-management aspects.

This is an accepted version of the published paper:
D. Macko, K. Jelemenská and P. Čičák, "Early-stage verification of power-management specification in low-power systems design," 2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Kosice, 2016, pp. 157-162.
doi: 10.1109/DDECS.2016.7482449
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7482449&isnumber=7482431

Based on the analysis of the existing methods, we have integrated into our low-power design methodology multiple verification steps, combining formal and informal approaches. Compared to the existing methods, our contribution is in three key methods, which are summarized below.

- *Power-management static analysis* – a unique method enabling verification of the consistency between power-management specification and functional specification at the system level of abstraction. Moreover, it enables verification of specification completeness, which helps the designer to refine the power management at early stages of the design.
- *Power-intent equivalence checking* – an original method for comparison between specifications at the system level and RTL. It is very useful to verify whether the power intent remains the same after the automated high-level synthesis process. The equivalence is checked formally and the process is fully automated; thus, this verification step is very fast.
- *Automated synthesis of power-control assertions* – it is not a completely original method (it has been inspired by [2] and [12]); however, we have integrated it into our high-level synthesis process in a unique form. The synthesized assertions monitor the power-management unit to behave functionally correctly. They are generated based on the abstract power-management specification only; thus, the designer is not required to specify any additional information.

III. Power Management in Specification

According to our previously published work [9]-[11], power-management specification at the system level is based on power modes, power domains, and power states. Power state is an operation state of some system component, defined by the operating frequency and supply voltage. Power domain is a collection of system components that always operate in the same power state. System power mode is a combination of power states in individual power domains. The basic principle of power management is to dynamically switch between power modes of the system according to the current power requirements. The goal is to reduce energy consumption, while successfully completing the given task.

A designer specifies the power management directly in the functional specification of the system. This specification contains the following.

- *Power domains* – The designer specifies the name for each domain and a set of power states, in which the internal components can operate. The allowed abstract power states are predefined (such as *normal*, *hold*, *off*; refer to [11] for more details).
- *Component assignments* – The designer assigns components to specific power domains. The components that always operate in the same power states are grouped into the same power domain.
- *Performance levels* – The designer specifies the voltage-frequency pair for each active power state.
- *Power modes* – The designer specifies the name for each power mode and a combination of power states (one power state for each power domain).
- *Management* – The designer specifies the switching between power modes directly in the functional specification.

Based on the specification at the system level (ESL), the UPF specification at the RTL is generated (see Fig. 1). Besides, the power-management unit (PMU), driving control signals for power-management elements in UPF, is automatically synthesized. Thus, the complex RTL power management is designed much faster using the proposed method.

IV. The Proposed Hybrid Verification Approach

After the power management is specified, the specification has to be verified for functional and structural correctness and completeness [13]. This verification step needs to be accomplished as soon as possible, enabling the designer to create a correct specification. Since the model needs not be executable at early phases of design, a formal approach is suitable for this kind of verification. We use compilers to check syntactical correctness of the specification combined with a static analysis revealing functional and structural inconsistencies of the abstract power-management specification.

The next step is to verify the correct functionality of the low-power system – this is usually accomplished by functional simulation. Since the abstract power-management specification does not model the effects on functionality, this verification step needs to be taken after the high-level synthesis process. At the RTL, the existing professional tools, such as Modelsim, can be used to simulate the synthesized UPF along with the functional design. As mentioned in [14], additional power-management elements are often a rich source of errors and must be thoroughly verified for all specified operating modes. One of the advantages of the proposed methodology lies in the automation. Since the power-management specification at the RTL is automatically generated, we are able to avoid many power-related human errors issued during the power-management insertion at such a later stage. The designer does not need to worry about specifying the low-level power-management logic, such as power switches, isolation, retention, or level shifters.

To assure the power intent is preserved during the high-level synthesis, the equivalence checking between the generated UPF specification and the abstract power management is a suitable verification approach. The

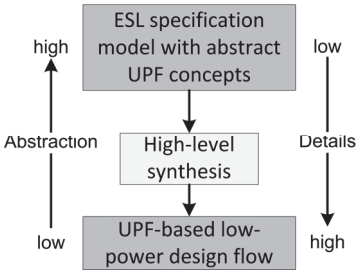

Fig. 1. Abstract view of the low-power design flow.

synthesized power-management unit is ideal for the assertion-based verification (ABV). The assertions can be generated for verification of control sequences as well as for functional-coverage measurement. The control-sequences assertions are automatically generated in a way similar to [12].

The overview of the proposed verification approach is shown in Fig. 2. The used combination of verification techniques enables the low-level power-related logic to be verified (both by simulation and formally) based only on the system-level abstract specification. This simplifies the complex verification process (especially the preparation and debugging steps).

A. Syntactic Checks

The syntax of power-management specifications is automatically verified by compilers at the compilation time – the predefined language keywords drive a designer to the correct specification. Moreover, the syntactical rules in HSSL (Hardware-Software Specification Language) [9] assure that if the power domains section is present in the system specification, then at least one power domain has to be specified. Also, each power domain has to have at least one power state specified. Syntactical rules ensure that only the predefined abstract power states are valid. For example, they can also ensure that at least one power mode is specified.

Such syntactic checks are not supported in SystemC-integrated power-management specification [10]. It relies on the designer's discipline to use the predefined power-management macros, ensuring correct specification. C++ modelling introduced to the extension library ensures that only the valid specified power mode is assigned to the variable representing the current power mode, and that the instances can be assigned only to the existing power domains. These issues are not revealed by the syntactic checks in HSSL. The mentioned unrevealed issues (in both the HSSL and SystemC) are checked-for in other forms of verification – during an execution time or through the static analysis.

B. Run-Time Checks

In SystemC extension library, the run-time checks reveal several errors that could not be revealed by a compiler. These checks are implemented in the traditional conditional manner. If the condition is true, the error message notifies the designer what is incorrectly specified. This verification form checks for the valid power states assigned to the power domains and to the power modes. In order to ensure that the designer does not cause the violation of this condition, the predefined power-state macros should be used. This kind of verification can also reveal the problem of specifying some state for some power domain more than once.

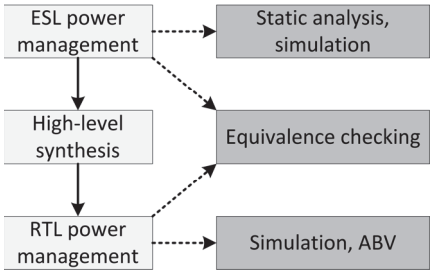

Fig. 2. Verification process overview.

C. Static Analysis

This form of verification is mainly intended to reveal the power-management inconsistencies that could not be revealed in the previous verification steps. It statically analyses the specification and checks the specified identifiers and assignments, related to the power-management specification. It is integrated into the high-level synthesis process but can also be used as a separate step to verify the specification in an early stage of development.

For example, the static analysis checks whether each power mode has the same number of power states as the number of power domains in the system. Also, it reveals the problem if some component instance is assigned to multiple power domains, or if some power domain does not contain any active power state. The static analysis also detects redundant parts of the specification (e.g. multiple power modes with the same combination of power states in power domains) and notifies the designer about their location. This verification step is a very helpful utility, driving the designer in early versions of the specification. The static analysis can be also used for verification of the power-management specification completeness. It detects missing required constructs, such as at least one active power state in a power domain, at least one power mode of the system, or missing power state in a power mode (i.e. power state is missing for some power domain in that power mode). This is very useful for the creation of early versions of power management. It significantly shortens debugging time.

An example of abstract power-management specification is illustrated in Fig. 3. The highlighted text represents the detected problems. The static analysis would detect that the *normal* power state of power domain named *power_domain1* is not used in any specified power mode (the power state of *power_domain1* corresponds to the first position of states in a power mode). The designer would be notified that the performance level for the *normal* state is not assigned. An error would be reported regarding the incorrect number of power states in *power_mode1*. The static analysis would detect that *power_mode3* is not used in the functional part of the specification (i.e. the module constructor or some process) – i.e. such a mode is redundant. Because of that, there would also

```
power_domain1 (off,normal,diff_level1);
power_domain2 (hold,normal,diff_level2);
diff_level1 (0.8 V, 5 MHz);
diff_level2 (1.2 V, 100 MHz);
system_component instance1(power_domain1);
system_component instance2(power_domain2);
power_mode1 (off,hold,normal);
power_mode2 (diff_level1,normal);
power_mode3 (diff_level1,diff_level2);
POWER_MODE = power_mode1;

...

if (control_condition)
  POWER_MODE = power_mode2;
else
  POWER_MODE = power_mode1;
```

Fig. 3. An example of static-analysis error detection.

be a notification that the *diff_level2* power state of *power_domain2* is not used.

D. Equivalence Checking

After the high-level synthesis, verification of power-management specification equivalency takes place. The power intent is extracted from the UPF and verified whether it corresponds to the abstract power-management specification at the ESL. Since the UPF contains only voltage aspects of the power management (frequency aspects are contained in the functional model), a common representation should be used for the comparison. Thus, even the system-level abstract power-management specification has to be translated into this common representation, in which the power states are represented by the corresponding voltage levels.

The common representation is formally expressed by a tuple *CR=(PDI,PDS,PM)*, where *PDI* is a finite set of tuples *(IDd, I)*, where *I* is a finite set of instances assigned to the power domain represented by an identifier *IDd*; *PDS* is a finite set of tuples *(IDd, VS)*, where *VS* is a finite set of voltage states enabled in the power domain represented by an identifier *IDd*; and *PM* is a finite set of tuples *(IDm, S)*, where *S* is a finite sequence of voltage states corresponding to power states representing the power mode with an identifier *IDm*.

When checking the equivalence between two specifications, we must distinguish between the power-management structural equivalence and the power-intent equivalence. The first one checks whether all specified aspects at the ESL are present in the generated UPF specification. The second one checks whether the power intent is preserved. The specifications do not need to be structurally equivalent to have the same power intent.

By an analysis of the UPF power-management specification, a quasi-reverse process to the high-level synthesis is used to extract the common representation. Firstly, the lists of power domains are compared. The ESL specification has to contain all UPF domains except for the top domain (implicitly added to the specification). To be structurally equivalent, the UPF specification also has to contain all domains in abstract specification. For the comparison purpose, the domains identifiers are used. The lists of assigned instances to each power domain have to be the same. In addition, the power states of power domains and the power modes with specific power-states combinations are compared. For the structural equivalency, the original abstract specification is used. For the power-intent equivalency, the abstract specification is modified using the optimization decisions (e.g. removal of redundant parts of the specification). This verification step produces error messages, which drives the designer to the source of an error, speeding-up the debugging process.

The equivalence-checking process can be explained using the example provided in Fig. 4. The figure contains two fragments of power-intent specifications at different abstraction levels. The fragment A) represents the specification at the system level of abstraction. Its transformation to the common representation is pretty straightforward. The power states in the specified power modes are translated to the voltage states based on the performance-level assignments. Thus, *power_mode1* specifies that the components in power_domain1 operate at the supply voltage of *1 V*. Similarly, *power_mode2* specifies that these components operate at the supply voltage of *1.2 V*. A more difficult task is to transform UPF specification to the common representation. Based on the specified state of the supply port connected to the primary supply net of *PD_top*, the voltage level of the *normal* state is determined. Other supply ports with specified states (such as *VDD_1_1*) imply the presence of multiple voltages in the design, and thus the presence of some *diff_level* state in the abstract specification. The specified states of the power-switch output port (*power_domain1_SW/vout*) serve as the basis for determination of other possible power states of the power domain, to which the switch belongs (i.e. drives its primary net). Based on the specified power modes in the power-state table of the UPF specification, the states of supply ports in the table are used to determine the used voltages for individual power domains in these power modes. In this way, we can observe that the *power_domain1_SW/vout* port is in *s_1_0* state in power mode of *power_mode1* and in *s_1_2* state in power mode of *power_mode2*. The port-state specification confirms that these states correspond to *1 V* and *1.2 V* supply voltages.

A)
```
power_domain1 (normal,diff_level1,…);
normal (1 V, 50 MHz);
diff_level1 (1.2 V, 100 MHz);
power_mode1 (normal,…);
power_mode2 (diff_level1,…);
```

B)
```
supply_on("/TB/DUT/VDD_1_0_port", 1.0);
supply_on("/TB/DUT/VDD_1_1_port", 1.2);
…
create_supply_port VDD_1_0_port -domain PD_top
create_supply_port VDD_1_2_port -domain PD_top
create_supply_net VDD_1_0_net -domain PD_top
create_supply_net VDD_1_2_net -domain PD_top
create_supply_net VDD_power_domain1_net\
 -domain power_domain1
…
connect_supply_net VDD_1_0_net\
 -ports { VDD_1_0_port }
connect_supply_net VDD_1_2_net\
 -ports { VDD_1_2_port }
set_domain_supply_net power_domain1\
 -primary_power_net VDD_power_domain1_net\
 -primary_ground_net VSS_0_0_net
…
create_power_switch power_domain1_SW\
 -domain power_domain1\
 -input_supply_port {vin_1_0 VDD_1_0_net}\
 -input_supply_port {vin_1_2 VDD_1_2_net}\
 -output_supply_port {vout VDD_power_domain1_net}
…
add_port_state VDD_1_0_port -state { s_1_0 1.0 }
add_port_state VDD_1_2_port -state { s_1_2 1.2 }
add_port_state power_domain1_SW/vout\
 -state { s_1_0 1.0 }
add_port_state power_domain1_SW/vout\
 -state { s_1_2 1.2 }
…
create_pst PST -supplies { VDD_1_0_port
VDD_1_1_port power_domain1_SW/vout }
add_pst_state power_mode1 -pst PST\
 -state { s_1_0 s_1_2 s_1_0 }
add_pst_state power_mode2 -pst PST\
 -state { s_1_0 s_1_2 s_1_2 }
…
```

Fig. 4. An example for checking the power-intent equivalence. A) Abstract power-management specification at the ESL. B) UPF power-intent specification at the RTL.

The example just provides a simple insight into the complex equivalence-checking process. There are other aspects, which are not shown due to space limitation in this paper. For example, the *off* port state represents the *off* or *off_ret* abstract power state. The isolation of power-domain inputs is used to determine the utilization of the *hold* state. If the inputs are isolated, but the port driving the primary-supply net of the domain does not have the *off* state specified, the *hold* state has to be specified for the corresponding power domain in the abstract specification. If there is the *off* state specified for such a port, the isolation does not imply the *hold* abstract power state. The presence of retention for some power domain implies the *off_ret* abstract power state to be specified in the ESL model. In this way, the common representation is extracted from the UPF specification and can be used for comparison to the ESL-extracted one.

E. Assertion-Based Verification

Based on the specified power management, assertions about lower-level control sequences can be generated. Firstly, the power states for each power domain are encoded using the control signals for power-management elements. The number of control signals for each domain depends on the number of used supply voltages, the number of clock frequencies, and the need for isolation and retention. The next step is to generate sequences that control the power-states transitions. These sequences include the intermediate power states, not specified at the system-level (such as isolation or retention states). Based on the specified power-state table, transitions between power states of power domains are determined and sequences generated. After the power-state encoding sequences and the power-state transition sequences are defined, these can be used to measure the coverage of the generated power management. An assertions-aware simulator counts if and how many times a property (represented by a sequence) has become true. Moreover, power-states encoding sequences are grouped together to define power-modes sequences. Thus, the designer knows how well the design has been tested in the simulation and what power modes yet need to be verified.

There is also a need to identify incorrect behavior of the power-management unit, generating the control signals. There are seven classes of illegal sequences, which reflect five categories described in [12] and additional two that we added (third and fourth). The properties representing these illegal sequences are asserted in the opposite way. If some of these properties evaluates true, an illegal sequence has occurred. The classes of illegal sequences include:

- illegal restoration before power-on,
- illegal power-off before retention,
- illegal de-isolation before power-on,
- illegal power-off before isolation,
- illegal retention before isolation,
- illegal de-isolation before restoration, and
- illegal performance-level transition.

Besides the assertions checking these illegal sequences, another three classes of control properties are asserted. One controlling that isolation is enabled during retention period, one controlling that isolation is enabled while powered-off, and the last controlling that retention is activated while powered-off. For now, we have used the generated assertions only in simulation-based verification; however, work of [12] gives us hope that they can also be used in formal property verification of the power-management unit.

V. Experimental Evaluation

We have evaluated the proposed verification approach by validating the synthesized power management using the existing industrial EDA (Electronic Design Automation) tool at the RTL. This evaluation checked whether our verification approach catches all errors, which would be otherwise detected at the RTL, in the specification phase of the design process.

In this evaluation, we have used power-aware static analysis offered by Modelsim SE 10.2c. We have created 15 samples of abstract power-management specification in SystemC and synthesized them into the UPF form. In order these samples to be representative we have used various complexity of power management (from under two thousand to over thirteen thousand characters in UPF specifications). Modelsim successfully validated the generated UPF specifications; thus, the proposed verification steps indeed drive a designer to the correct power-management specification. In addition to the static analysis, we have used the power-aware simulation capabilities offered by Modelsim to functionally verify the synthesized power management units. Also, the communication between the PMU and power-management elements in UPF is verified. For the simulation to take place we had to create test-benches, in which the power supplies were activated and the switching between various power modes occurred (using pseudorandom approach). These test-benches also contained the generated assertion (written in SystemVerilog Assertions language), used for checking the PMU functionality as well as for measuring the coverage during the simulation.

Data for the used samples are provided in Table I. The first column contains just the sample number for referencing. The ESL column represents the number of characters required for abstract power-management specification in SystemC. The UPF column contains the number of characters of the synthesized UPF specification. PMU represents the number of

TABLE I. Power-Management Verification Evaluation Samples

#	ESL	UPF	PMU	Assert	Directives	Coverage
1	313	1680	3300	3863	10	100 %
2	500	2706	4452	4749	17	100 %
3	642	2850	4619	3194	15	100 %
4	643	6035	35205	25912	103	98 %
5	751	4658	8658	9488	29	100 %
6	760	4854	7017	10340	34	100 %
7	813	5678	10094	13433	41	97.5 %
8	862	5557	63304	17779	81	100 %
9	953	8778	142992	52330	175	81.1 %
10	1051	9399	131478	45150	156	96.1 %
11	1090	11605	586303	100721	421	85.5 %
12	1275	7443	199866	48111	146	89 %
13	1324	9039	107068	43833	172	91.2 %
14	1402	13397	67691	50911	126	99.2 %
15	1939	12232	214122	91620	188	82.4 %

characters required for functional description of the synthesized power-management unit. The Assert column provides the number of characters of the generated assertion statements. The Directives column provides the number of explicit coverage directives in the generated assertions. The last column (Coverage) reflects the directive coverage reported by Modelsim.

To achieve full coverage for each sample, another stimuli generation approach should be used, such as directed testing. The goal was to show that the generated assertions can be directly used to verify the PMU and for coverage measurement, what has been proved. The number of characters of assertion statements can be really high; therefore, their automated synthesis spare significant amount of time, required for the manual verification preparation. Moreover, the human errors, possibly introduced by such a manual process, are completely avoided; thus, the debugging effort is reduced. The experiments have also shown that all the verification steps and proposed methods are scalable, since both the ESL static analysis and the equivalence checking after the high-level synthesis have taken only few seconds. The longest time is taken by the generation of the assertions. However, compared to the manual assertion creation, the proposed automated method can save days, or even weeks, of design time.

VI. CONCLUSIONS

The power-management specification and synthesis methods, which we have proposed in previous work, bring by themselves undeniable verification benefits. The abstraction from error-prone low-level details (e.g. power switches, isolation or retention) results into very concise and intuitive specification, avoiding many common power-management errors. This paper describes a way how a designer is guided to build a correct and complete abstract specification by automated means of the syntactical checks during the compilation, the run-time checks during the system-level simulation, and the original static analysis during specification creation and power-management synthesis. After the synthesis, the equivalence between the generated specification and the abstract power-management specification is verified by the novel equivalence checking approach. Besides these verification steps, the power-management synthesis generates the controlling assertions, checking the correct operation of the power-management unit. Also, the coverage assertions are automatically generated for a designer to track the power-modes verification progress.

In comparison to the analyzed existing approaches, we have proposed the most robust power-management verification approach. The early verification drives the designer to develop a complete and consistent power-management specification at the system level and the continuous verification steps during design flow ensure that the power intent remains preserved. Most of the verification steps are automated; thus, the preparation and debugging verification processes are significantly shortened.

The further work in this area can be oriented towards automated test-bench synthesis, specifically to creation of directed power-mode switching. It could provide a faster way to completely verify the synthesized power management at the RTL.

The possible next challenge is to properly generate power-management control sequences for asynchronous systems. The method [15] could help the PMU to know when the power state can be switched in the corresponding power domain. It could make power management to be more adaptive to the environment. However, the verification of such a PMU could be very difficult, since the generated assertions are strongly clock-based.

REFERENCES

[1] IEEE standard for design and verification of low power integrated circuits, IEEE Standard 1801-2013, May 2013.

[2] O. Mbarek, A. Pegatoquet, and M. Auguin, "Using unified power format standard concepts for power-aware design and verification of systems-on-chip at transaction level," IET Circuits, Devices & Systems, vol. 6, no. 5, pp. 287-296, 2012.

[3] J. Karmann and W. Ecker, "The semantic of the power intent format UPF: Consistent power modeling from system level to implementation," in 2013 23rd international workshop on power and timing modeling, optimization and simulation (PATMOS), pp. 45-50.

[4] F. Mischkalla and W. Mueller, "Advanced SoC virtual prototyping for system-level power planning and validation," in 2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), pp. 112-119.

[5] Y. Xu, R. Rosales, B. Wang, M. Streubühr, R. Hasholzner, C. Haubelt, and J. Teich, "A very fast and quasi-accurate power-state-based system-level power modeling methodology," in ARCS'12 Proceedings of the 25th international conference on architecture of computing systems, 2012, pp. 37-49.

[6] H. Lebreton and P. Vivet, "Power modeling in SystemC at transaction level, Application to a DVFS architecture," in IEEE Computer society annual symposium on VLSI, 2008, pp. 463-466.

[7] T. Bouhadiba, M. Moy, and F. Maraninchi, "System-level modeling of energy in TLM for early validation of power and thermal management," in DATE '13 Proceedings of the conference on design, automation and test in Europe, 2013, pp. 1609-1614.

[8] S. Ahuja, "High level power estimation and reduction techniques for power aware hardware design," Ph.D. dissertation, Faculty of the Virginia Polytechnic Institute and State University, 2010.

[9] D. Macko and K. Jelemenská, "Managing digital-system power at the system level," in IEEE Africon 2013 Sustainable Engineering for a Better Future, pp. 179-183.

[10] D. Macko, K. Jelemenská, and P. Čičák, "Power-management specification in SystemC," in 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 259-262.

[11] D. Macko, K. Jelemenská, and P. Čičák, "Power-management high-level synthesis," in The 23rd IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), 2015, pp. 63-68.

[12] A. Hazra, S. Goyal, P. Dasgupta and A. Pal, "Formal verification of architectural power intent," IEEE Transaction on very large scale integration (VLSI) systems, vol. 21, no. 1, pp. 78-91, January 2013.

[13] Cadence Design Systems, A Practical Guide to Low Power Design: User Experience with CPF, 2012.

[14] N. Khan, "Cosed-loop verification methodology for low-power SoC design," Special Technology Report – Low Power Design, no. 1, pp. 7-8, September 2008.

[15] L. Nagy and V. Stopjaková, "Current sensing completion detection in dual-rail asynchronous systems," in 2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 28-41.