PMHLS 2.0: An Automated Optimization of Power Management During High-Level Synthesis

Dominik Macko
Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava, Slovakia
dominik.macko@stuba.sk

***Abstract*—Design automation is very important in modern systems-on-chip development, complexity of which is ever growing. The most crucial issue in highly integrated systems is the increased power density and the corresponding temperature problems influencing reliability. Therefore, the power must be managed in such systems. Power management enables to implement various power-reduction techniques, such as power gating, multiple voltages, or voltage and frequency scaling. However, the automation of power-management design starts at the register-transfer level. Only the recent research begins to adopt power management at the system level of abstraction, which is increasingly used in the industry as a design starting point. In this paper, we propose an enhanced automation of the design process by using the optimized power-management high-level synthesis. This method transforms the system-level power-management specification to the traditionally used form at the register-transfer level. We have implemented this method to a tool called PMHLS, which automates the whole process. It uses optimization decisions to resolve some kinds of inconsistencies and thus makes the power management more efficient. This automation helps to reduce the number of human errors, potentially introduced by a designer during manual design. It also significantly speeds up the system development process. The benefits of the proposed method and the implemented design-automation tool are supported by the experimental results.**

Keywords*—*design automation; high-level synthesis; low power; power management; specification

I. Introduction

Due to the limited capacity of batteries in mobile devices, enhanced packaging or cooling problems, or just because of the reduction of energy consumption, the power is managed in almost every new chip design [1]. A system-on-chip (SoC) implemented in deep submicron CMOS (Complementary Metal-Oxide Semiconductor) technology suffers from the power-density problem, which influences its reliability. To reduce the SoC power consumption, one must understand the factors influencing the power. The static power (leakage) depends on the supply voltage, the threshold voltage for transistor switching, and the transistor size. The dynamic power depends on the switching activity, the clock frequency, the transistor and load capacitances, the supply voltage, and the short-circuit current [2].

The reduction of dynamic power can be achieved by reducing the clock frequency and switching activity. However, it influences the performance of the system. There are applications, in which not all parts of the system always need to operate at the full-performance capability. The spared power can be then saved to reduce the energy consumption, or redirected to other parts of the system to enhance their performance. This is called a dynamic power management. It also enables to reduce static power by temporal modification of voltage threshold or supply voltage, or by powering the unused SoC components down.

In highly integrated complex SoCs, the power management is quite difficult to design, and custom design approaches are not sufficiently efficient. Therefore, a systematic low-power design flow has been standardized in the industry under the no. IEEE Std 1801-2013 [3] (commonly known as UPF – Unified Power Format). UPF enables to specify power management at the RTL (Register-Transfer Level) as an extension to the functional HDL (Hardware Description Language) model. The low-level power management elements (such as power switches, level shifters, and isolation or retention cells), supply ports and supply nets can be specified and verified at the RTL design stage, and using the EDA (Electronic Design Automation) tools, they can be automatically implemented at lower abstraction levels. Modelling of the power management enables more accurate power analysis of the designed system at the RTL. Regarding the specification, UPF enables to divide the system architecture into so-called power domains. A power domain groups together components of the system, which always operate at the same supply-voltage level (i.e. power state). In the UPF, the power state is defined by a unique combination of control signals of aforementioned power-management elements belonging to one power domain. The designer specifies the possible voltages of each supply port or net and uses them to create a power-state table. It reflects allowed combinations of voltages in these ports and nets, what significantly helps the power-management verification.

To deal with the complexity of modern SoC designs, the International technology roadmap for semiconductors suggested the adoption of an abstract electronic system level (ESL) in the design process [4]. Therefore, the abstraction offered by the UPF is not sufficient and there are attempts to

This work was partially supported by the Slovak Scientific Grant Agency (VEGA 1/0616/14 and 1/0836/16) and the Slovak Research and Development Agency (APVV-15-0789).

This is an accepted version of the published paper:
D. Macko, "PMHLS 2.0: An automated optimization of power management during high-level synthesis," 2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Bremen, 2016, pp. 205-212.
doi: 10.1109/PATMOS.2016.7833689
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7833689&isnumber=7833417

extend this standard to the system level. However, there is usually missing a connection between the ESL and the RTL power-management specification, and thus the verification is somewhat difficult. We introduce this connection in a form of the power-management high-level synthesis process that we have previously developed [5]. In this paper, we show how the power management can be further optimized during this process to make it more efficient (regarding the area and power overhead). The result is that the designer does not need to keep all power-management aspects in mind during the abstract power-management specification and can focus on the system function.

This paper is organized as follows. In the next section, the related research works are described that target the system-level power management and high-level synthesis. Section III briefly describes the most commonly used power-reduction techniques for SoC designs. In Section IV, we introduce the principles of the previously developed abstract power-management specification [6]. Section V is devoted to the power-management high-level synthesis and the proposed optimization. And finally, before the conclusion, the benefits of our proposal are proved by the experimental results.

II. Related Work

There are various methods targeting different aspects and problems of system-level power management. The first group targets standard-based power-management modelling. The second group targets in-house modelling of power management aspects. The third group includes methods that enable modelling of SoC components' power consumption; however, they cannot model power management (switching between operating states with various power consumptions). The last analyzed method targets specification of one power-management technique and its automated implementation in the synthesized SoC model.

A. Standard-Based Methods

The ESL power-intent model used in [7] is based on the UPF standard concepts, and therefore the UPF specification can be automatically generated. It enables to use existing EDA tools for verification and power analysis. However, the used power-data model is strictly dependent on design reuse, and thus it is not suitable for top-down design approach. Moreover, this method uses similar amount of details for power-management specification at the ESL as the UPF specification; therefore, the specification is just translated into another format. The method [8] includes abstract specification of voltage relationships, TLM power states, operating conditions, and so on. It extends the power-domain UPF concept in such a way that each power domain is mapped to one clock domain. The disadvantages are missing automation towards lower abstraction levels and separated specification of SoC function and power intent. The PwARCH framework [9] augments an ESL model with abstract UPF concepts (e.g. power domains, power switches, power nets, or power-state table). The method requires power annotation to the model; thus, it is dependent on design reuse. It also uses separated specification of functional and power-management aspects, what is fairly unsuitable for the system level of abstraction.

B. Nonstandard Methods

The methods [10-12] also offer ESL power modelling, including the power management support. The developed power-intent specification approaches are not based on the standard concepts – i.e. they use in-house specifications, limiting the compatibility with the professional EDA tools. Nevertheless, these methods can be used for power-architecture exploration at the early design stages, and thus can help to make suitable power-management policy decisions. The analyzed methods do not take into account other important design parameters, such as area or performance. The used nonstandard concepts also complicate the verification of the equivalency between ESL and RTL power management.

C. Power-Estimation Methods

The existing ESL power-estimation methods [13-15] help to select the most power-efficient SoC architecture. The power consumptions of the components are required to be specified manually in the ESL model. Therefore, these approaches are also based on design reuse. In some cases, the default values can be used, but it would result in a less accurate power analysis. These methods do not support power-management specification. If the power management is introduced into the design only later, at the RTL, the ESL power estimation would not correspond to the actual power consumption. Therefore, these methods are unusable in power-managed SoC designs.

D. High-Level Synthesis Based Methods

A method has been developed [16], which focuses on the high-level synthesis to quickly obtain an RTL model for design exploration and more accurate power analysis. Clock gating specified in the ESL model is passed to the high-level synthesizer that automatically implements it in the RTL model. The disadvantage is that the location and activation of clock gating needs to be specified manually in the ESL model, what complicates the specification. Also, this method does not support other power-management techniques; therefore, the power can be wasted.

We have identified pros and cons of the existing methods and used this information for development of a new low-power systems design methodology [5, 6]. The methodology uses abstracted power-management specification based on UPF concepts to simplify its introduction (inspired by [8, 9]), offers high-level synthesis for trade-off among multiple parameters and more accurate RTL design analysis (inspired by [16]), and generates UPF standard specification at the RTL to maintain the compatibility with the existing verification tools (inspired by [7]). To clarify the developed extension of the standard UPF-based design flow, we illustrate the offered design flow in Fig. 1. The extensions are provided in dark-grey color and the parts further enhanced by the work described in this paper have a dashed-line border.

III. Power-Reduction Techniques

There are several power-reduction techniques that are commonly used in SoC design today. Some of them are described in more detail in [2] or [17]. However, since not all

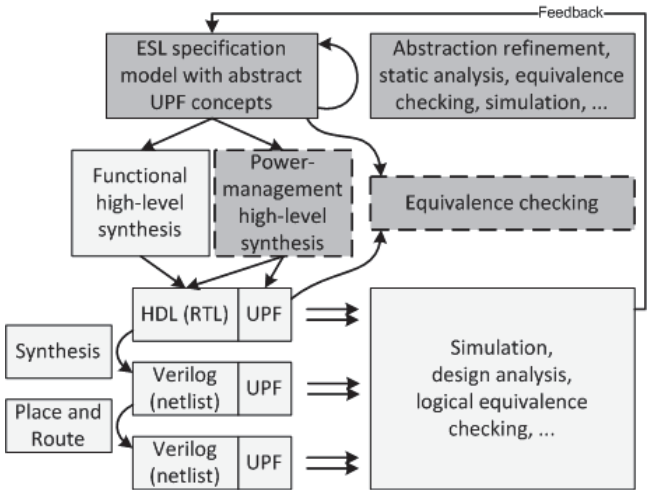

Fig. 1. An illustration of the developed low-power systems design flow.

of these techniques are applicable using the dynamic power management, we provide their overview with a brief discussion on their usage in power-managed SoC designs in Table I.

To apply these techniques in a SoC design, several rules must be followed, which require additional elements to be incorporated into the design. For example, when powering-down a SoC component, its output signals need to get isolated in order to prevent corruption of data in the powered components, which these signals are connected to. This is usually accomplished by some isolation cells added to the component boundary. These cells are implemented variously, using a simple AND gate or a more complicated latch-based elements. Also, the interconnections between components working at different voltage levels must be level-shifted in order to correctly exchange information. In some cases, these elements can be combined to operate as the isolators as well as the level shifters. Similarly to isolation, a clock signal can be stopped to apply the clock-gating technique. However, the designer should carefully consider timing impact when modifying the clock signal. When using multiple voltages for some component to increase or decrease its performance, a power switch is used to switch between different power-supply nets. This switch is also used to completely power-down the component. If the state of the component should not get lost during the power-down, some retention cells have to be used in its registers.

IV. ESL Power-Management Specification

The developed abstract power-management specification [6] is targeted towards a simple application of multiple power-reduction techniques in context of architectural dynamic power management. Similarly to UPF, it also utilizes power states, but in more intuitive way. There are no power-management elements in the abstract ESL specification; therefore, an abstract power state does not represent a combination of their control signals, but rather it is defined by a unique combination of the voltage and frequency values. Thus, it directly represents more like a performance level.

The power-reduction techniques that are applicable by dynamic power management can be specified by the predefined abstract power states (summarized in Table II). These states actually specify the support of which architectural power-reduction technique will be introduced at lower abstraction levels. The *OFF* and *OFF_RET* states specify the application of power gating without and with the state retention support. The *HOLD* state introduces the clock gating in combination with the operand isolation. The *DIFF_LEVEL* group of states enables the voltage and frequency scaling and the use of multiple voltages in the design. And finally, the *NORMAL* power state specifies that there will not be any explicit architectural power-reduction technique applied.

TABLE I. Overview of Power-Reduction Techniques

Technique	Description
Clock gating	It stops the clock signal, and thus prevents loading the same value in a register. From the architectural power-management perspective, it can be used to stop the operation of a synchronous SoC component. The unnecessary switching can be prevented in this way, and thus the dynamic power can be saved.
Power gating	It temporarily shuts-off the power from a SoC component when not currently needed. This technique can be used for reduction of the static power as well as the dynamic power. However, the time requirements for switching between on and off sates must be kept in mind.
Operand isolation	It isolates an unneeded combinational portion of the circuit, and thus reduces unnecessary switching activity, what saves the dynamic power. Applied as an architectural power-management technique, it can be used to isolate all input ports of an unused SoC component.
Voltage scaling	It enables to switch a SoC component between multiple voltage levels – e.g. a high-voltage level for the high performance and a low-voltage level for the energy saving. Usually, the voltage is scaled in the same way for all components grouped in a power domain based on the current performance requirements.
Frequency scaling	It enables to switch the clock frequency of a synchronous SoC component. When a component operates at a higher frequency, it processes the given task faster; however, it consumes more dynamic power. This technique usually works along with the voltage scaling to reduce power by decreasing the performance when the processed task is not time-critical.
Substrate biasing	It enables to temporarily increase the threshold voltage when the given SoC component is not used or a non-critical task is processed. It reduces the leakage current and thus reduces the static power. It is usually coupled with clock gating to reduce both the dynamic and static power of currently unused SoC component.
Multiple voltages, multiple thresholds, gate sizing, logic restructuring, pin swapping	These techniques are nowadays automatically implemented by a synthesis tool, selecting a suitable combination of library cells with various parameters to meet preset constraints. These techniques are used only during the design time and cannot change the SoC power consumption during the runtime.
Memory partitioning, bus segmentation, hardware acceleration	These techniques represent SoC component micro-architectural decisions. When a component is decomposed into smaller parts, they can be selectively turned into some power-saving states according the current requirements. On the other hand, the hardware acceleration is used to perform a given task more efficiently, what can eventually save the power.

TABLE II. ABSTRACT POWER STATES

Power State	Description
OFF	The components belonging to the power domain in this state are powered down.
OFF_RET	The same as OFF, but the component state is retained.
HOLD	The components belonging to the power domain in this state stop the operation, but stay powered.
DIFF_LEVEL#	The components belonging to the power domain in this state operate at the performance level different from the basic one; # represents an ordinal number enabling the specification of multiple states with different performance levels.
NORMAL	The components belonging to the power domain in this state operate at the basic performance level.

The SoC is usually set to a specific operating mode in order to process some task. This mode is called a power mode and it represents a combination of power states to which the power domains are set in this mode. It is easier for a designer to change just the power mode than to change the state of each power domain. Thus, the abstract power-management specification involves the specification of:

- *power domains* – the specification includes the name of the domain and a set of power states, in which the components of the domain can operate;
- *component assignments* – the specification includes assignments of SoC components to some already specified power domain;
- performance levels – the specification includes the definition of voltage and frequency values for each active power state (*NORMAL* or *DIFF_LEVEL*);
- power modes – the specification includes the name for the mode and a sequence of power states (one state for each power domain);
- policy – the specification involves a definition of how and when the switching between power modes occurs – it is incorporated into the functional specification.

For an illustration of SystemC abstract power-management specification, a code fragment is provided in Fig. 2. The architectural power management is specified in the top module

```
#include "systemc.h"
#include "pms.h"
SC_MODULE(soc){
 PowerDomain PD1,PD2;
 PowerMode PM1,PM2;
 soc_component C1,C2;
 ...
 SC_CTOR(System):C1("C1"), C2("C2"){
  PD1 = PD(OFF,NORMAL);
  PD2 = PD(NORMAL,DIFF_LEVEL(1));
  Set_Level(DIFF_LEVEL, 1.1 V, 100 MHz);
  Set_Level(NORMAL, 1 V, 50 MHz);
  PD1.AddComponent("C1");
  PD2.AddComponent("C2");
  PM1 = PM(OFF,NORMAL);
  PM2 = PM(NORMAL,DIFF_LEVEL(1));
  POWER_MODE = PM1;
  ...
 }
}
```

Fig. 2. A sample of power-management specificaiton in SystemC.

of the system. Power domains and power modes are declared in the declaration part of the model, and then defined in the functional part such as the constructor. There is a special variable, called *POWER_MODE,* which represents the current power mode. It is initialized to some defined power mode and used in the functional part of the specification (in this case not a constructor, but some SystemC process) to change the system power mode.

V. POWER-MANAGEMENT HIGH-LEVEL SYNTHESIS

Such a simplified power-management specification, along with architectural relations among components, provides a sufficient amount of details to implicitly deduce power intent and utilize the high-level synthesis process to generate a standard UPF specification [5]. There are some power-management rules that need to be followed by the high-level synthesis, which are briefly stated below and mapped to the abstract power states.

- Input signals of SoC components in the power domain that operates in the *HOLD* power state connected to a component outside of the domain have to be isolated.
- Signals connecting SoC components in different power domains that operates at different voltages (*DIFF_LEVEL* power states) have to be level-shifted.
- Signals connecting SoC components in different power domains that operates at different frequencies have to be synchronized.
- The synchronization signal (clock) of a SoC component in a powered-down domain (*OFF* or *OFF_RET* power states) has to be stopped.
- Signals of SoC components in a powered-down domain that are connected to a powered component outside of the domain have to be isolated.
- The power supply of SoC components in the power domain that is in the *OFF* or *OFF_RET* state has to be switched off.
- The power supply of SoC components of the power domain that operates in multiple power states with different voltages (*DIFF_LEVEL* or *NORMAL* power states) has to be switchable.

The benefit of high-level synthesis process automation is that the designer does not need to keep these rules in mind. The designer specifies only what should be accomplished, not how – it is implicit, based on these rules. The developed high-level synthesis process focused on power management consists of two parts. The first part involves the synthesis of power-management specification in the UPF standard form itself. The other part targets the synthesis of a power-management unit. It is a controller that drives control signals for the power-management elements located in the synthesized UPF – power switches, isolation cells, level shifters, and retention cells. Firstly, we introduce principles of the optimization of the power-management high-level synthesis process, and then, we describe the two parts of this process in more detail (along with the optimization of these parts).

A. Optimization Principles

The whole optimization process is based on the static analysis of the abstract power-management specification and of the structural dependencies among the SoC components. The analysis helps to locate power-management inconsistencies or redundant parts and the optimization process resolve them, if possible, based on some predefined priorities. For example, the specification of power modes has precedence over the specification of power states in power domains. It means that if some power state is specified in the power-mode definition for some power domain (deduced based on the state position in the sequence – the first state belongs to the firstly specified power domain), but it is not specified in the power-domain definition, the specification of power domain is considered wrong instead of the power mode. Another example of the stated priorities is in the state sequence in the power-mode definition. If there is a higher number of power states specified in a power mode than a number of power domains in the system, the states from the left have precedence – i.e. the states from the right exceeding the number of power domains are redundant.

These are examples of the resolved inconsistencies. Such an optimization helps only if the abstract power-management specification is wrong. It automatically corrects the specification, giving a designer the possibility to focus more on the functional specification and to not keep all power-management aspects in mind at the ESL. However, the optimization also focuses on the analysis of the architectural dependencies among the SoC components as well as relations among power domains. For example, if two power domains are always powered-off during the same period of time and are in some active state (although not necessary the same) in the remaining time, the isolation of signals among their components is not necessary. This optimization results in a more efficient UPF specification – sparing redundant power-management elements, which would require area and power.

B. UPF Specification Synthesis

This part of the proposed power-management high-level synthesis process extracts power intent from the ESL power-management specification and determines which power-management elements and power-distribution nets have to be synthesized in the UPF specification. In the following text, we briefly describe individual synthesis steps and how the optimization impacts the eventual specification.

1) Power Domains

The power domains are created as the first step in the UPF specification. These domains are already defined in the ESL specification; therefore, they are just rewritten to the UPF form, along with the assigned components. In the UPF, there is also a top domain created, which contains the components that are not explicitly assigned to any power domain in the ESL specification. Regarding the optimization, there is not much to do. The optimization influences the domain creation only if there is some inconsistency in the abstract specification. For example, if some power domain has no component assigned, it is considered redundant, and therefore it is not created in the UPF specification.

2) Power Distribution

After the domains are created, there is a need to create supply nets for them and connect them to supply ports. For each voltage level, determined from performance-level specifications, the individual power-supply port and supply net are created. If the power supply of a power domain has to be switchable (i.e. at the ESL, the power domain can operate in power states with at least two different voltages, including off state), a dedicated supply net is created. Otherwise, it reuses some top-domain supply net. In this step, the primary power and ground nets are assigned to each power domain. Optimization process determines whether all specified voltages are also used in the system – i.e. whether the power domain is set in some power mode to the power state corresponding to the voltage level and whether this mode is sometime activated. The support for unused power supplies is not created using the optimization.

3) Power-Management Elements

Afterwards, the power-management elements can be specified. For each power domain with a switchable power supply, a power switch is created. For each power domain that can be set in the *HOLD* power state, input isolation is created. Isolation of both, the inputs and outputs, is created for each power domain that can be powered down. Level shifters are created between power domains operating at different voltages. For each power domain that contains the *OFF_RET* power state, the retention is set. Similarly to the previous step, the optimization checks whether the power states are actually used in the system, and only then they are taken into account. It can reduce the number of control signals required by the power-management elements, or even reduce the number of these elements. Moreover, the isolation between domains is created only if they are not always active and inactive at the same time.

4) Power-State Table

The last step involves creation of the power-state table, containing the legal combinations of ports voltage states. However, before it is created, the voltage states for the created supply ports and the output ports of the created power switches have to be specified. The possible voltage states of these ports correspond to the associated supply nets, which represent the primary power net of some power domain. Abstract power states of such a power domain then determines the voltage states of the port. The specification of the power-state table is based on the ESL power modes; however, the states are specified for the ports instead of the power domains and UPF states represent only voltage states, the frequency is omitted. Moreover, there are additional voltage modes (analogously to voltage states) generated in the UPF power-state table. These additional states and modes will be discussed in more details in the power-management unit synthesis description (the next subsection). These additional modes are added to the power-state table only if the corresponding combination of voltage states is not yet specified in the table. The optimization influences the number of ports and their voltage states and the number of additional voltage modes, and thus it influences the complexness of the power-state table.

C. Power-Management Unit Synthesis

The power-management elements generated in the UPF specification contains the control signals that need to be activated and deactivated in a precise time to switch between power states. This is the task of the power-management unit. Since the abstract specification does not contain the power-management elements, there is no such unit in the ESL specification. Therefore, its functional description has to be completely generated in this part of the power-management high-level synthesis process.

The power-management policy algorithm is defined in the functional ESL specification using the *POWER_MODE* variable. The power-management specification high-level synthesis process precedes the functional synthesis. It modifies the type of the *POWER_MODE* variable to the enumerated type, enumerators of which are the identifiers of the specified power modes. In this way, the switching between power modes does not need to be modified – it is correctly modelled. An enumerator corresponding to the first specified power mode actually contains an unsigned integer value of 0, an enumerator of the next power mode contains 1, and so on. This is a functional description synthesizable by commonly used EDA tools supporting the high-level synthesis. The synthesized *POWER_MODE* variable drives the input of our power-management unit. This value represents the mode, into which the SoC should be switched. It is firstly processed by a power-mode determination component (see Fig. 3) that converts it to a vector representing a combination of control signals for the power-management elements in UPF. The encoded target power mode is then processed by the power-state machine, which is responsible for the correct transitions between power modes. It generates a sequence of intermediate power modes to safely reach the target power mode by following the power-management rules, summarized below.

- Before a SoC component is powered down, it has to stop its operation.
- Before a SoC component is powered down, isolation of its inputs and outputs has to be activated.
- Before a SoC component is powered down, its state retention (if required) has to be activated.
- Before the state of a SoC component is restored (if retained), the component has to be powered up.

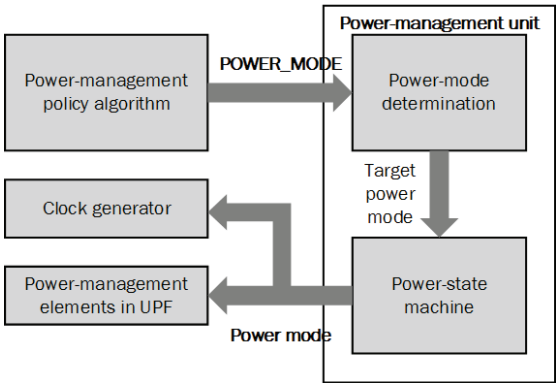

Fig. 3. The synthesized power-management unit overview.

- Before the isolation of a SoC component is deactivated, the component has to be powered up.
- Before the state of a SoC component is retained, the component isolation has to be activated.
- Before the isolation of a SoC component is deactivated, its state has to be restored (if retained).
- In case a SoC component changes from a high-performance state to a low-performance state, the frequency has to be changed before the voltage.
- In case a SoC component changes from a low-performance state to a high-performance state, the voltage has to be changed before the frequency.

Such a power-state machine implements the transitions between all possible power modes, including the intermediate modes. However, an intermediate power mode cannot be the target power mode – i.e. it cannot come as an input because the *POWER_MODE* variable cannot represent such a mode. The control signals generated by the power-management unit do not control only the power-management elements specified in UPF, but also the clock generator in order to enable the frequency scaling in SoC components.

The proposed optimization process greatly influences complexness of the power management unit, and thus it potentially reduces its area and power requirements. The key is the reduction of control signals for power-management elements, since these also represent the required state logic in the power-state machine, as well as the wiring required to transfer these signals. An important complexness reduction is achieved by optimization of the number of power states and power modes to only actually used ones. It influences the state logic as well as the transition logic complexness.

D. Equivalence Checking Extension

Because of the optimization of the abstract specification, the synthesized UPF does not directly correspond to the original ESL specification. However, the actual power intent achieved by the original and optimized specification is the same. Therefore, we have modified the equivalence checking procedure to verify the actual power intent, not the structural equivalency between power-management specifications at the ESL and the RTL. It means that the support of never-used power states and power modes does not need to coincide in order the specifications to be equivalent. The modified algorithm therefore firstly optimizes the original ESL specification and then checks its equivalence with the synthesized UPF specification. Thus, the extended power-management specifications equivalence checking consist of two procedures – checking of the structural equivalence (i.e. all specified aspects are checked) and checking of the power-intent equivalence (i.e. whether the eventual management of SoC components power is the same).

VI. Experimental Results

In order to experimentally evaluate the proposed optimization process, we have used a quantitative pseudorandom approach. Specifically, we have automatically

generated over ten thousand artificial abstract power-management specification samples in SystemC and used them to synthesize UPF specifications using both high-level synthesis algorithms, without (original, previously developed) and with the optimization (modified, proposed in this paper). The goal was to show the simplification when using the developed abstract-power management specification at the ESL compared to the standard UPF specification at the RTL. Of course, the goal was to also show the comparison of the UPF specifications synthesized without and with the optimization, in order to prove the benefits of the optimization proposed in this paper.

As the comparison parameter, we have selected the number of characters required for the specification (referred to as the specification complexity). The reason is that the UPF specification is based on the Tool Command Language (TCL), which has long inline commands, and therefore the comparison of statements would be unfair. Also, we could not leave the identifiers out from the comparison, because the abstract power-management is strongly identifier-based. One must realize that the complexity reduction is not achieved by a shorter description of some language constructs, but by the abstraction from the low-level details, such as the power-management elements and power-distribution network.

To use various complexities of specification samples, the pseudorandom generation has been based on the scaling of multiple parameters, including as the number of power states in power domains, the number of allowed system power modes, the number of power domains, the number of SoC components in power domains, and the number of connections among components.

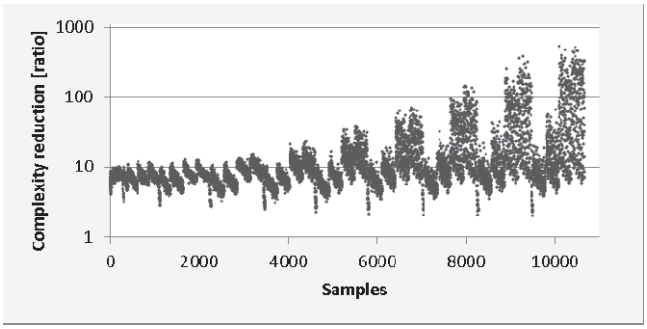

Fig. 4. A comparison of the SystemC abstract power-management specification and the UPF specification synthesized without the optimization.

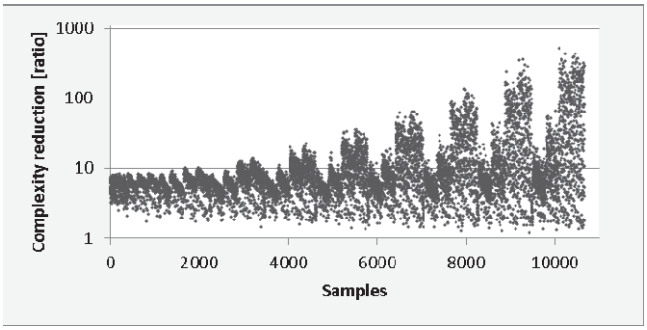

Fig. 5. A comparison of the SystemC abstract power-management specification and the UPF specification synthesized with the optimization.

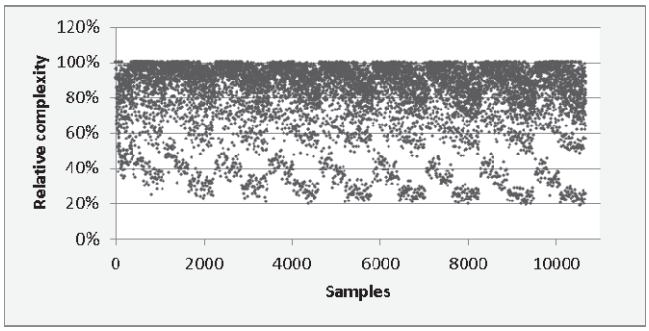

Fig. 6. The UPF specifications synthesized with the optimization compared to the UPF specifications synthesized without the optimization.

The results in Fig. 4 illustrate the complexity reduction when comparing the abstract SystemC power-management specification to the standard UPF specification synthesized without the optimization. The case comparing the abstract specification to the UPF synthesized with the proposed optimization is illustrated in Fig. 5. The vertical axis in the charts is provided in a logarithmic scale. The result of the experiment is that the SystemC power-management specification is approximately 16.8 times less complex in average than the UPF specification synthesized without the optimization and 14.3 times less complex in case of the optimization being used. Although, the comparison results show that the optimization brings the worse complexity reduction, we must realize that it synthesizes the optimized UPF specification (shorter).

To illustrate the benefit of the optimization in clear way, Fig. 6 provides the comparison of the UPF specifications synthesized with the optimization and without. The result of the experiment is that the optimization brings complexity reduction of the synthesized UPF specification approximately by 19% in average. It means that less power-management supporting logic will be used, but the power intent remains the same. Thus, the resulted power management is more efficient in terms of area and power savings achieved by not implementing the unnecessary elements. However, we must note that some of the used pseudorandomly generated samples contained inconsistencies, which could be resolved by the proposed optimization. For a correct and consistent abstract power-management specification, the UPF specifications synthesized without and with the optimization would not have as much difference. Nevertheless, the optimization fulfills its purpose, because a designer can focus on the functional part of the SoC specification. The power-management specification at the ESL does not need to be completely correct/consistent, in order the consistent UPF specification at the RTL to be synthesized.

As another experiment, we have used a professional EDA tool (Modelsim SE 10.2c) to verify the syntactical and semantical correctness of the synthesized UPF specifications as well as the synthesized power-management unit. Specifically, we have used UPF static analysis and power-aware simulation capabilities, offered by this tool. However, for this purpose, we have used only 15 selected samples with

various complexities, because it would take a lot of time to create test-benches for all the generated samples and to simulate them. For verification of the power-management unit, we have used the assertion-based verification – the assertions regarding the correct control sequences for power-management elements are automatically synthesized during the power-management high-level synthesis (this has been also developed in our previous work). All of these samples, synthesized to the optimized UPF, have successfully passed through the modified equivalence checking and have been successfully analyzed by the Modelsim power-aware static checks. The synthesized power-management units have been exercised during short simulations (each covering more than 80% of the power modes) and there was no assertion violated.

VII. Conclusions and Further Work

This paper is focused on the optimization of the power-management high-level synthesis process, which fills the gap between the abstract ESL power-management specification and the more-detailed RTL power management. The power-management high-level synthesis process analyzes the abstract power-management specification and automatically generates the standard well-supported UPF specification along with the functional description of the corresponding power-management unit.

The proposed optimization is able to automatically resolve some kinds of specification inconsistencies, potentially introduced by a designer at the specification stage. Using the developed abstract power-management static analysis and SoC components structural relations, the optimized power-management high-level synthesis process generates only the required power-management elements at the RTL. The proposed automated error-recovery simplifies the abstract power-management specification, because a designer does not need to keep all the power-management rules in mind. It is especially useful in the early design stages. The designer has more time to focus on the correct functional specification.

The experiments using a pseudorandom approach have proved the benefit of the developed optimization. The results have shown that the optimization can reduce the complexity of the synthesized UPF specification by not implementing the support for never-used abstract power states and power modes in the RTL model. The eventual power intent however remains the same. It has been verified by the modified equivalence checking that we have also proposed in this paper. Such an equivalence checking takes into account the optimization decisions. The specifications at the ESL and the RTL do not need to be structurally equivalent to have the equivalent power intent. The developed power-management high-level synthesis is intended to run alongside the functional high-level synthesis that is already used in the industry. Together, the proposed automated high-level synthesis significantly speeds-up low-power SoC development process and avoids many human errors, potentially introduced by the manual transition from the ESL to the RTL.

The further work in this area can be focused on automated dividing of SoC components into power domains and automated selection of suitable power states for power domains. It could bring us closer to a complete abstraction from power management at the ESL and its implicit introduction during the high-level synthesis. It could speed-up low power SoC development even more.

References

[1] M. Keating, D. Flynn, R. Aitken, A. Gibbons and K. Shi, Low Power Methodology: For System-on-Chip Design, Springer, 2007.

[2] Power Forward Initiative, A practical guide to low power design: User experience with CPF. Power Forward, 2012.

[3] IEEE standard for design and verification of low power integrated circuits. IEEE, 2013. IEEE Std 1801-2013.

[4] The international technology roadmap for semiconductors: Design. ITRS, 2011 edition, 2011.

[5] 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.

[6] 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 & Systems, 2015, pp. 259-262.

[7] 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), 2013, pp. 45-50.

[8] 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), 2014, pp. 112-119.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] M. Giammarini, M. Conti and S. Orcioni, "System-Level Energy Estimation with Powersim," in 18th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2011, pp. 723-726.

[14] D. Greaves and M. Yasin, "TLM POWER3: Power Estimation Methodology for SystemC TLM 2.0," Models, Methods, and Tools for Complex Chip Design, LNEE, vol. 265, pp. 53-68, 2014.

[15] F. Klein, R. Azevedo, L. Santos and G. Araujo, "SystemC-Based Power Evaluation with PowerSC," in Electronic System Level Design: An Open-Source Approach, S. Rigo, R. Azevedo and L. Santos, Eds., Springer, 2011, pp. 129-144. ISBN 978-1-4020-9939-7.

[16] S. Ahuja, High level power estimation and reduction techniques for power aware hardware design, Faculty of the Virginia Polytechnic Institute and State University, 2010. Dissertation thesis.

[17] Y. Shin, "Low-Power Circuits: A System-Level Perspective," in Energy-Aware System Design: Algorithms and Architectures, C.-M. Kyung and S. Yoo, Eds., Springer, 2011, pp. 17-46. ISBN 978-94-007-1678-0.