Rapid Power-Management Exploration Using Post-Processing of the System-Level Simulation Results

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

***Abstract*—Managing the power in highly-integrated systems on chips becomes inevitable in modern designs. Complex systems require complex power management, and it is always difficult to determine whether the designed power management is the most efficient. In our previous work, we have proposed a simplified power-management specification method at the system level of abstraction. In this paper, we propose a system-level power-management evaluation approach that enables a designer to explore various power-management designs in a short time and select the best. The proposed exploration method is easy-to-use and can be used to speed-up the low-power systems development.**

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

I. Introduction

One of the challenges in electronic-systems design is to increasingly use the system-level abstraction (ESL – Electronic System Level) and more automation to improve design productivity [1]. It is especially important in the design process of low-power systems, such as Internet of Things (IoT) devices, which become more and more complex. A dynamic power management is usually used to apply various power-reduction techniques, such as voltage and frequency scaling, power and clock gating, or voltage and clock islands [2]. From a system point of view, power management enables to switch among several power modes of the system, according to the current needs to efficiently perform a task.

There are many works addressing the system power estimation under a specific power-management scheme at the ESL, such as [3-13]. However, the used abstraction is either insufficient, making the ESL specification unnecessarily complicated. Or the used power-management modelling approach is not based on the standard concepts (see Section 2) used at lower abstraction levels, which makes checking of the equivalence between higher and lower level models difficult and automated transition between these levels is quite impossible. Moreover, the existing methods usually rely on power data based on design reuse, which is not usable for a top-down design of custom hardware. Strictly power-managed IoT devices are especially sensitive to overhead of the introduced power controller (managing power of other components of the system). This overhead is rarely taken into account in existing methods, and if it is, its modelling is mostly manual.

Therefore, there is a need for an approach that simplifies the ESL specification of power management and quickly estimates its impact on the system power consumption. This is what we address in this paper. We propose a new ESL power-estimation method that takes into account the specified power management. To increase the estimation accuracy, the method automatically predicts power overhead of the required power controller, which is not modelled at the ESL. The method does not influence simulation performance and, in some cases, it does not even need re-simulation of the modified model. This, along with the automation of the method itself, makes the exploration of various power-management schemes fast.

The rest of the paper is organized as follows. Section II introduces standard basics of power management and related works in the area of ESL power-management exploration. The new method is proposed in Section III, followed by experimental evaluation in Section IV, and Section V concludes the paper.

II. Background and Related Work

The standard way to model and verify power management is to use UPF (Unified Power Format) [9] or CPF (Common Power Format) [14] specification alongside the traditional HDL (Hardware Description Language) functional design at the RTL (Register-Transfer Level) and lower abstraction levels. UPF enables to capture power intent, specifying power-supply nets and supply ports, connecting them to power domains (grouping multiple components for more efficient power management). These domains can also contain so-called power-management elements, such as power switches, isolation and retention logic, or level shifters. Thus, the power state of a power domain in UPF context is defined as a combination of control signals activating and deactivating power-management elements assigned to the power domain. In more complex systems, these control signals are driven by a dedicated unit representing power controller (referred to as PMU – Power Management Unit). A power mode of the system (set by the PMU) is then a combination of power states of all power domains (one state per domain).

Modification of power intent in UPF is not an easy task. A designer must keep the system context in mind and change precise portions of the specification. For example, in order to give some component ability to power it down, a designer has to assign it to a power domain (new or an existing one), modify

This is an accepted version of the published paper:
D. Macko, "Rapid power-management exploration using post-processing of the system-level simulation results," 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, 2017, pp. 108-113.
doi: 10.1109/PATMOS.2017.8106965
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8106965&isnumber=8106944

the primary supply net of the domain (if necessary), add a power switch to the domain or add an off state to the existing power switch in the domain, modify the control signals of the switch (if necessary), set isolation at the domain boundaries, and so on. Moreover, the PMU design has to be modified to incorporate the new power state of the power domain. Another downside of RTL modelling is that clock signals (i.e. operation frequency of the components) are modelled in HDL. Thus, some power-reduction techniques are applied using UPF and some using functional modelling in HDL, what increases the complexity.

Due to these reasons, RTL modelling is not convenient for power-management exploration and more abstraction must be used in the process. We have analyzed several ESL-based approaches targeting this issue [3-13]. All of the analyzed methods require too much manual effort for exploration of various power-management strategies at the system level. To obtain power data of the components, these methods usually depends on design-reuse concept, which is not suitable for top-down design approach addressed in this paper. Moreover, there exist low-power systems in which the power-management unit (PMU) represents a significant area/power overhead, because other components of the system are powered down (many IoT devices). Therefore, the PMU (i.e. power controller) overhead must be taken into account when exploring power-management alternatives. Based on the downsides of the existing methods, we have formulated requirements for a new power-management exploration method, which also represent our key contributions summarized below.

- *Easy modification of power-management strategy* – by using a sufficiently abstract power-management specification method, we have made the modification of power management clear, simple, and intuitive.
- *Minimizing simulation-performance overhead* – since the power-management effect is not modelled at the system level, ESL simulation performance remains intact.
- *Fast estimation of system energy consumption* – due to using simulation post-processing for power evaluation, this method benefits from high degree of possible parallelism.
- *Evaluation of PMU overhead* – although the power controller is not specified at the system level due to the abstraction, we have proposed a method to estimate its overhead at the ESL.

III. Power-Management Exploration

The proposed power-management exploration method works as illustrated in Fig. 1. We assume that a designer has already specified the system function in SystemC and that the specification is executable. The designer then empirically specifies the first power-management alternative using the SystemC/PMS library [15]. Afterwards, the ESL simulation is performed, which not only verifies the system function, but also monitors the switching among power modes of the system. The switching activity of the system is saved into a VCD (Value Change Dump) file. Then, the post-processing of the VCD data is performed. The post-processing includes an estimation of energy consumption of the system, while taking into account the specified power management. By the analysis of the ESL power management, the power/area requirements of the corresponding PMU are also estimated (PMU will be included in the RTL model). The designer then modifies the power-management specification and executes the simulation again if necessary (i.e. a number of power modes has changed or power-mode switching has been modified). The simulation post-processing estimates the power consumption of the system with modified power management and estimates the requirements of the new PMU.

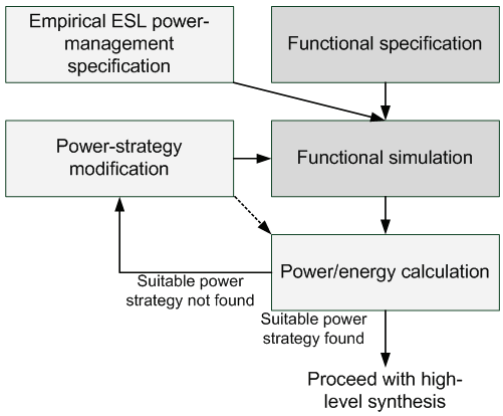

Fig. 1. The proposed power-management exploration method overview.

This procedure can be repeated as many times as the time allows. The designer then compares the estimated results and selects the most suitable power-management alternative for the designed system. In the following subsections, individual parts of the proposed power-management exploration method will be described in more details.

A. ESL Power-Management Specification

The specification of power management at the ESL is really simple. There are only few things that need to be specified: assignments of power states and system components to power domains, power states of each power domain in individual power modes, definition of voltage and frequency values for each active power state, and switching among power modes.

The power states that can be assigned to a power domain can be divided into the following two groups.

- Active – the components in the domain are operating.
  - *Normal* – basic voltage and frequency levels.
  - *Diff_level* – alternative voltage and frequency levels.
- Passive – operation of the components in the domain is stopped.
  - *Hold* – the components remain powered.
  - *Off* – the components are powered down.
  - *Off_ret* – values of memory elements in the components are saved while the components are powered down.

Using the SystemC/PMS specification, an example of specification of power states for a power domain is illustrated

below. *Power_domain1* is a name of the power domain, *PD* is a PMS macro to create an instance of power domain, and the power states *Normal* and *Off* can be used in the power domain.

```
Power_domain1 = PD(NORMAL, OFF);
```

The components are assigned to a power domain using the *AddComponent* PMS method, in which the only argument represents an instance name of the component. An example is provided below.

```
Power_domain1.AddComponent("CPU");
```

Since *Normal* is an active power state, the voltage and frequency values have to be defined (we say that a performance level is assigned to this state). A code example is shown below, where *Set_Level* is a PMS macro in which the first argument represents a power state, the second argument represents its supply voltage, and the last argument refers to the operation frequency.

```
Set_Level(NORMAL, 1V, 50MHz);
```

A power mode is specified as a combination of power states in individual power domains (i.e. the number of power states must correspond to the number of power domains in the system). The specification code example is provided below.

```
Power_mode1 = PM(NORMAL, HOLD, NORMAL);
```

Where *Power_mode1* is a name of the power mode, *PM* is a PMS macro to instantiate the power mode, and the states correspond to power domains according to the order in the specification. It means that the first state belongs to the firstly specified power domain, the *Hold* state belongs to the second domain, and the last power state corresponds to the third power domain specified in the system.

Such a specification is very intuitive and quite understandable not only for a hardware designer, but also for a system architect or an embedded-software developer. Since this specification is really simple, it is easy to modify it (even several times), and thus explore multiple alternatives.

B. Estimation of Energy Consumption

The proposed specification does not model effects of power management on the system. It has both advantages and disadvantages. One advantage is that the simulation performance is not affected – simulation performance is one of the key benefits of the system-level abstraction. The other advantage is that a designer does not need to worry about the system function when specifying power management. However, the problem lies in estimating the impact of the specified power management.

Our energy-estimation method is based on power factors, which are specific for the wide-spread CMOS (Complementary Metal-Oxide Semiconductor) technology. The factors influencing power are divided into two groups: static and dynamic. The static factor represents power required for a component to be powered. The dynamic factor represents power required to switch internal state of the component. At the ESL abstraction, these factors can be roughly interpreted as the size of the component and the activity of the component. An overview of the proposed estimation method is illustrated in Fig. 2. Explanation of individual parts of the method, along with formal expressions for computation, follows.

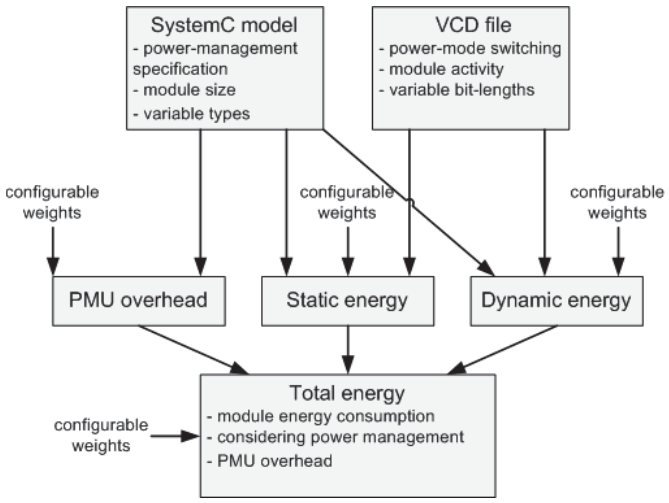

Fig. 2. An overview of the proposed energy-estimation method.

The accurate size of a system/component cannot be determined without the technology to be used for implementation and without the synthesis. However, to explore multiple power-management alternatives, we do not need highly accurate numbers. We use a fast analysis of the SystemC model and estimate the size (i.e. static power factor) based on the description of modules. We take into account a number of characters required for description combined with a number of lines of each SystemC module (Eq. 1).

$$CP_S = w_c.N_c + w_r.N_r \tag{1}$$

In Eq. 1, CP_S is static power of a component representing description size, computed as weighted values of the number of characters ($w_c.N_c$) and the number of rows ($w_r.N_r$) of the description.

Since this estimation cannot ensure accurate values (due to wide variety of coding styles and the fact that a larger description not always implies a larger circuit), these data can be set by a designer. They are usually obtained from previous implementations of the used components. However, what we have accomplished in contrast to the existing methods is that we use this static power characterization in combination with the analyzed power management and estimate the system static energy consumption under specific power-management strategy. The method determines power-mode switching from the VCD file, where the time and power-mode number are monitored. Based on the analysis of power management, the static power value of the component is multiplied by the time operating in a specific power state and a coefficient corresponding to that power state. We thus obtain the static energy consumed by the component during the simulated time as a sum of these values for each power state. Formally, it is expressed by

$$CE_S = CP_S . \sum_{PS} (sc_{PS} . \Delta t_{PS}) \quad (2)$$

where CE_S is static energy consumed by a component, CP_S is its static power (description size, or the value inserted by a designer), sc_{PS} is a coefficient influencing static power consumption in the power state *PS* and Δt_{PS} is a time interval in which the component operated in the power state *PS*.

Activity of the component is saved in the VCD file, resulted from the system simulation. In order to use it for determination of dynamic power, usually the Hamming distance is used. It refers to a number of different bit values between two vectors. In our case, it can be used as a number of bits (of SystemC module variables) that have changed between two subsequent simulation times in VCD. Since different types of variables have different impact on dynamic power, their contribution is weighted based on the type (input/output port, signal, internal variable). A sum of the weighted contributions (i.e. a number of bit-changes) of all variables between two time points represents the activity of the component during that time interval. The time intervals are created based on the time points, in which the power mode of the system was switched. For each interval, the dynamic energy consumption is computed by a function of the component activity, corresponding to the power state in that interval. For example, in the *Hold* power state, the activity is not taken into account; on the other hand, in the *Off* power state, only the activity required to power down and up the component is reflected. Formally, the dynamic energy consumption of a component (CE_D) is expressed by

$$CE_D = \sum_{\Delta t} \sum_{i} [w_i . f_{PS}(a_i)] \quad (3)$$

where w_i is a weight of variable *i* based on its type, f_{PS} is a function that modifies the activity contribution a_i of variable *i* based on the power state in time interval Δt.

The static energy E_S consumed by the system is a sum of consumptions of all its components *j* (Eq. 4). Similarly, a sum of dynamic energy consumptions of all components *j* during whole simulation time represents the dynamic energy consumed by the system E_D (Eq. 5).

$$E_S = \sum_{j} CE_{S_j} \quad (4)$$

$$E_D = \sum_{j} CE_{D_j} \quad (5)$$

It is recommended to use the proposed estimations of static and dynamic energy consumption separately, because there is no direct correlation between the obtained values. However, if a designer needs to use only one number to compare it to another design, we have introduced weights of static and dynamic factors to differentiate impact of these factors on total energy consumption (also usable for various implementation technologies). The total energy consumption (E) is then

$$E = w_S . E_S + w_D . E_D + w_{PMU} . C_{PMU} . t \quad (6)$$

where w_S is a weight of the static factor and w_D of the dynamic factor, w_{PMU} is a weight of PMU overhead C_{PMU} (explained in the next subsection) and *t* is the simulation time.

What is interesting and important regarding the proposed method is that the computation does not always need to be performed completely. The method is oriented towards fast exploration of multiple power-management alternatives, what enabled its high speed. If the power-mode switching is not modified, the simulation does not need to be rerun. The consumed energy is recomputed based on the new power states in the power modes. Therefore, it is recommended to explore partitioning of the system to power domains and various power states before modification of the power-mode switching or changing the number of power modes. Obtaining the energy consumption of new power-management strategy without new simulation is something not found in any existing method.

C. Estimation of PMU Overhead

In the used SystemC/PMS specification of the system, the PMU is not specified (i.e. it is abstracted away at the ESL). Switching of power modes is part of the functional specification. As we have already mentioned, the PMU can become a significant power consumer in some ultra-low-power systems. Since the required PMU can influence the selection of a power-management alternative, it is important to predict/estimate its complexity. For a higher accuracy, the high-level synthesis of PMU can be used [16]. Such a synthesis process automatically generates an RTL model of the PMU that corresponds to the ESL specification of power management. Based on the synthesized RTL model, the professional tools, such as Power Compiler [17] or Joules RTL Power Solution [18], can estimate its power and area requirements. Although such estimation is quite accurate, this process is not very suitable for power-management exploration. To obtain the requirements of each PMU alternative, a designer needs to always synthesize the power management, perform the design analysis, and report back the obtained data. When we take into account also preparation time, this process is just too long (minutes, or even hours).

For a complexity comparison of two alternative PMU designs, we do not need precise power values. The estimation can be achieved in matter of seconds directly from the ESL model. In the proposed method, we have used a modified PMU high-level synthesis algorithm, which analyzes the ESL power-management specification and estimates the PMU complexity (which would be otherwise synthesized). The method statically analyzes the ESL power states that can be used in power domains, structural dependencies among the components in different power domains, and setting of power states of the domains in each power mode. Based on the analyzed data, the method then estimates the required power-management elements, and thus we can determine the number of required control signals. Moreover, the required intermediate power modes for the functionally correct power management are determined. For example, if a power domain can operate in the *Normal* and *Off* power states, an intermediate state is needed to

activate the isolation prior to powering the domain down. These are not explicitly specified at the ESL.

The PMU is an application-specific Medvedev-type finite-state machine, in which the state encoding directly corresponds to the control signals (in order to avoid synchronization issues). Based on the estimated information, we can predict a number of states in the PMU, a number of flip-flops to keep the state, a number of input/output ports, and also a number of branches in the transition logics of the PMU. Such information directly corresponds to the size of the PMU, and thus we use the number of control signals and the number of power modes to estimate its complexity. Since the actual area/power requirements depend on the implementation technology, we use weights of the used parameters to calibrate the estimation. Thus, the estimated PMU complexity C_{PMU} is expressed as

$$C_{PMU} = w_{CS}.CS + w_{PM}.PM \quad (7)$$

where w_{CS} is a weight influencing impact of the number of power-management control signals CS and w_{PM} is a weight of the number of power modes PM.

The proposed method uses a single algorithm to predict PMU overhead directly from ESL model. Although it is less accurate than the approach using high-level synthesis and RTL estimation, it is easy to use, faster (results in few seconds), and does not require any preparation steps.

IV. Experimental Results

We have divided the experimental results into three groups. In the first part, we show the power-management exploration capabilities of the proposed method. The second part focuses on the accuracy of the estimated data, compared to an RTL method. The last part shows that the proposed estimation method for PMU overhead is relevant and useful.

A. ESL Power-Management Exploration

The exploration capabilities of the proposed method are illustrated on a simple system design. This system consists of one microprocessor component and two memories. We have estimated energy of such a system without power management. Then, we have specified five power-management alternatives (various combinations of power states in the system power modes) and estimated the energy consumed by this system under specific power management during the same simulation task. The results of this experiment are provided in Table I. The first column represents a power-management alternative upon the same system design (alternative *A* is without the power management). The next columns represent the estimated static energy, dynamic energy, PMU complexity, and energy of the whole system. The last column represents a comparison of energy values of the alternatives, compared to the alternative *A* (i.e. without power management).

TABLE II. The Power-Management Exploration Results

Alternative	E_S	E_D	C_{PMU}	E	E/E_A
A	37824	818963	0	256502,5	100,0%
B	35263	522641	233.1	242996,8	94,7%
C	30788	522641	355.8	217590,3	84,8%
D	45725	522641	107.8	307240,8	119,8%
E	42278	534308	107.8	284987,0	111,1%
F	34083	675620	215.6	236790,6	92,3%

The results show, that the alternative *C* is the most energy-efficient. This experiment has proved that the proposed method can be used to evaluate various power-management alternatives and select the most efficient one. However, the estimated values can be used only for comparison reasons, not for verification of the power budget (too high abstraction is used for such a case).

B. Comparison of ESL and RTL Energy Estimation

Since the estimated energy values are not provided in Joules (or Watts), it is difficult to compare them to the values obtained from a professional tool. Therefore, we have compared the accuracy of a ratio between the estimated energy of a design with power management and without power management. This ratio is compared to a ratio of the RTL estimations (with and without power management) computed by Synopsys Power Compiler using the NanGate_15nm_OCL technology library. Two different designs have been used for comparison. The results are provided in Table II. The first column represents a factor, ratio of which was compared. The ESL columns represent ratios based on the ESL estimations (the proposed method was used). Based on the statistical analyzes (i.e. calibration), the weights of the factors were set to w_S=6.5, w_D=0.013, and w_{PMU}=1000. The RTL1 and RTL2 columns represent ratios based on the RTL estimations. The Δ columns represent difference between ESL and RTL ratios.

The results show that the proposed ESL energy-estimation method is accurate enough (inaccuracy up to 22%) regarding the comparison of multiple design alternatives with different power management strategies. It cannot be used to estimate absolute values of energy consumption, but it is useful for power-management exploration.

C. Evaluation of PMU Overhead Estimation

To evaluate the PMU requirements estimation capabilities of the proposed method, we have specified 15 different power-management alternatives in SystemC/PMS. The proposed algorithm then computed *CS* and *PM* parameters of these alternatives to predict the number of control signals and the number of power modes. The experimental data are provided in Table III. The first column represents just an index of a power-management alternative. The next two columns represent the estimated parameters. Based on the calibration, the weights of the parameters have been set to w_{CS}=32.6 and w_{PM}=2.5. The estimated complexity C_{PMU} was afterwards compared to the synthesized and estimated power of the PMU at the RTL (Power Compiler and NanGate_15nm_OCL technology have been used). This comparison is illustrated in the last column of the table (Δ).

TABLE I. A Comparison of ESL and RTL Estimation Results

Factor	ESL1	RTL1	Δ	ESL2	RTL2	Δ
static	1.08	1.1	-1.8%	0.68	0.87	-21.8%
dynamic	0.067	0.06	+11.7%	0.083	0.09	-7.8%
total	0.87	0.82	+6%	0.57	0.68	-16.7%

TABLE III. PMU OVERHEAD EVALUATION DATA

#	CS	PM	C_{PMU}	RTL power [uW]	Δ
1	3	5	110,3	96,6	+14%
2	3	4	107,8	114	-5%
3	4	5	142,9	167	-14%
4	7	12	258,2	260	-1%
5	10	14	361	355	+2%
6	10	8	346	379	-9%
7	10	49	448,5	537	-16%
8	13	46	538,8	562	-4%
9	13	106	688,8	594	+16%
10	16	114	806,6	705	+14%
11	19	132	949,4	815	+16%
12	17	117	846,7	873	-3%
13	17	101	806,7	891	-9%
14	21	65	847,1	961	-12%
15	18	346	1451,8	1398	4%

The results indicate that the estimated number of control signals and the number of power modes can be indeed used to represent PMU complexity at the ESL, since the estimated complexity corresponds to obtained power values from the RTL analysis with inaccuracy of 16% and lower. This is an improvement in this area, because the proposed method can be used to estimate future requirements of the PMU not specified at the ESL.

V. CONCLUSION

We have proposed a new power-management exploration approach that enables easy "what-if" analysis of multiple alternatives. Really easy and intuitive modification of ESL power-management specification makes it usable not only for hardware designers, but also for system architects and embedded software developers. The fast energy-estimation method does not influence simulation performance and, in some cases, it does not even require re-simulation of the system with an alternative power management. A unique method for prediction of future power-management unit requirements makes the exploration more accurate, especially for systems in which the power controller is a significant power consumer (e.g. some ultra-low-power systems spending most of the lifetime in a sleep mode).

The experiments showed that the proposed methods are useful for power-management exploration. We have not properly evaluated time, which the proposed exploration approach can save, because it is subjective, dependent on designer's experience. We can estimate that the modification of power management is faster, because of the high abstraction used for specification. ESL simulation is faster than using the existing methods that affect its performance. The energy-estimation itself is perfect for parallel computation; thus, the results are obtained in seconds.

ACKNOWLEDGMENT

This work was partially supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic (ITMS 26240220084), the Slovak Scientific Grant Agency (VEGA 1/0836/16), and the Slovak Research and Development Agency (APVV-15-0789).

REFERENCES

[1] ITRS 2.0, "International technology roadmap for semiconductors 2.0," 2015. [Online]. Available: www.semiconductors.org/main/2015_international_technology_roadmap_for_semiconductors_itrs/

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

[3] K. Grüttner et al., "COMPLEX: Codesign and power management in platform-based design space exploration," in 2012 15th Euromicro Conference on Digital System Design, 2012, pp. 349–358.

[4] Y. Xu et al., "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.

[5] 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.

[6] 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.

[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] IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems, IEEE Standard 1801-2015, Dec. 2015.

[10] T. Cahayag, Ed., "Synopsys Platform Architect MCO Delivers Industry's First Power-Aware Architecture Analysis Tool Supporting IEEE 1801-2015 UPF 3.0," Synopsys, Jan. 2016. [Online]. Available: http://news.synopsys.com/2016-01-25-Synopsys-Platform-Architect-MCO-Delivers-Industrys-First-Power-Aware-Architecture-Analysis-Tool-Supporting-IEEE-1801-2015-UPF-3-0

[11] 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 Circ. Device. Syst., vol. 6, pp. 287–296, Sept. 2012.

[12] H. Affes, M. Auguin, F. Verdier, and A. Pegatoquet, "A methodology for inserting clock-management strategies in transaction-level models of system-on-chips," in 2015 Forum on Specification and Design Languages (FDL) , 2015, pp. 1–7.

[13] H. Affes, A.B. Ameur, M. Auguin, F. Verdier, and C. Barnes, "An ESL framework for low power architecture design space exploration," in 2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2016, pp. 227–228.

[14] A Practical Guide to Low Power Design: User Experience with CPF. Cadence Design Systems, 2012. [Online]. Available: www.si2.org/?page=1061

[15] 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, 2015, pp. 259-262.

[16] 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.

[17] Power Compiler: Power Optimization in Design Compiler. Synopsys, 2014. [Online]. Available: https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/power-compiler-ds.pdf

[18] Joules RTL Power Solution: Unified power calculator for accurate RTL power and signoff-quality gate power. Cadence Design Systems, 2016. [Online]. Available: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/joules-rtl-power-solution-ds.pdf