Rapid Estimation of Power-Management Unit Overhead from System-Level Specification

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

***Abstract*—Power management becomes an integral part of hardware-systems design. In modern complex systems, the power-management design is not a simple task and it is quite difficult to evaluate whether the designed strategy is the best. In this paper, we propose a new method for overhead estimation of the required power-management unit, based on system-level abstract specification. It enables a designer to explore various power-management strategies in a short time and select the most suitable one. It is especially useful for ultra low-power systems, in which the power-management unit is a significant power consumer. The proposed method is simpler and faster than the existing approaches, and thus it speeds-up the low-power systems development process.**

I. Introduction

The power of highly integrated systems-on-chips (SoCs) has been a problem for some time. The concern is the overheating of the chip, caused by a high power density. This problem is dealt with by utilization of various power-reduction techniques during the system design. The most well-known techniques include clock gating, power gating (also known as power shut-off), multi-voltage design, or voltage and frequency scaling [1]. In complex systems, these techniques are usually applied using the so-called power management [2]. It enables to use various power states of the system components and to dynamically change these states. The goal is to redirect the power to more demanding components and to reduce the overall energy consumption.

To increase design productivity, the ITRS 2.0 [3] has identified as one of the challenges the system-level design automation. It also affects the power management. There are several new methods to model power management at the system level (ESL – electronic system level), such as [4]–[8]. However, the used abstraction is either too low, resulting in inefficient design exploration at the ESL, or it is too high to properly determine power and area overhead of the introduced power management. It is more crucial in the coming era of Internet of things (IoT), since these devices (mostly power-constrained) are especially sensitive to power overhead of a controller, strictly managing power of other chip components.

We focus in this paper on fast and simple determination of power-management controller overhead based on a specification in SystemC/PMS [9]. Based on our experiences in power-management high-level synthesis [8], we propose a new system-level method to predict power requirements of the controller at lower design-abstraction levels. This method can help to make proper decisions regarding the power-management strategy by shortening the time for determination of power-management controller overhead to a minimum. In the next section, the state-of-the-art in this area is analysed. Section III introduces power-management concepts, on which the proposed method is based. The new method itself is proposed in Section IV and supported by experimental results in Section V.

II. Related Work

Most of the existing system-level methods exploring power management are based on power-per-state model or power-per-transaction model [2], [6], [10]–[13]. Although they enable power estimation of system components under some power management policy at the ESL, these approaches require power characterization of the components from the lower-level power estimations (which take time). The power controller overhead is not taken into account in these approaches. This is also a downside of the COMPLEX [4] framework. It enables to generate an power-aware executable SystemC model based on a formal specification, which significantly contributes to design-automation area. There are other approaches that enable to model power controller, although manually [5], [7], [14]. Thus, its overhead can be taken into account while estimating power. However, in case of a change in power-management strategy, the controller must be remodelled and the simulation rerun, which takes time.

Based on the identified weaknesses of existing approaches (i.e. missing power-controller overhead and lack of automation), we have proposed a new method to eliminate them. As far as we know, none of the existing approaches takes automatically into account (or estimates) power overhead of the power controller itself at the ESL. In our method, we avoid its modelling at the ESL (which saves time), but predict its requirements for implementation. As we have mentioned in Section I, there are applications (e.g. IoT sensor devices) in which the power controller is a significant consumer of power (e.g. in standby mode). Thus, it is important to predict its power/area requirements for a certain power-management strategy.

This is an accepted version of the published paper:
D. Macko, "Rapid Estimation of Power-Management Unit Overhead from System-Level Specification," 2017 Euromicro Conference on Digital System Design (DSD), Vienna, 2017, pp. 9-13.
doi: 10.1109/DSD.2017.19
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8049761&isnumber=8049708

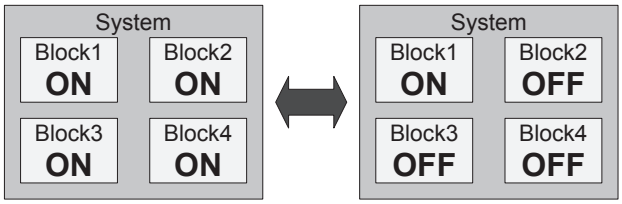

Fig. 1. Power-mode switching example.

III. Background

The power management enables to switch between various power modes of the system. A power mode is a combination of power states of all system blocks. A power state usually corresponds to supply voltage of the component and its operation frequency (thus, it can be also seen as a performance level). An example of switching between power modes is illustrated in Fig. 1. The left part contains a high-performance mode (i.e. all system blocks are operating) and the right part represents a low-power mode (i.e. three blocks are powered-down to save power).

At the RTL (register-transfer level), the power management is usually modelled using the UPF (Unified power format) [2] or CPF (Common power format) [15] specification alongside the HDL (hardware description language) functional model. Both power formats provide similar features; therefore, we further discuss only the standardized UPF. The UPF specification captures division of the system into so-called power domains. A single power domain contains multiple system blocks that operate in the same power state (it enables more efficient control). The UPF also enables to specify supply ports and nets and to connect them to power domains. There is also a special circuitry specified enabling to switch voltages and distribute them through supply nets - we call this circuitry the power-management elements. It includes power switches (enabling to cut-off supply voltage or to switch supply net), isolation logic (for isolation of signals entering or leaving the power domain), level shifters (adjusting voltage level of inter-domain logic signals), or retention cells (enabling to save the logic value during a power-down period). Therefore, the power state at the RTL is defined as a specific combination of values of signals, controlling the power-management elements in UPF. These signals are driven by a power controller in complex systems (i.e. PMU – power-management unit).

An example of a system with a dedicated PMU, controlling power of the blocks divided into power domains, is illustrated in Fig. 2. The PMU controls each power domain separately – i.e. the control signals for a domain are based on the power states, in which the blocks of the domain can operate. The number of control signals, as well as their type (isolation control, power switch control, retention control), varies for each domain. For example, the *PD1* power domain has three control signals (one for isolation and two for switching among three voltage levels), but the *PD2* power domain requires only two control signals (one for isolation and one to power down the *Block2* component).

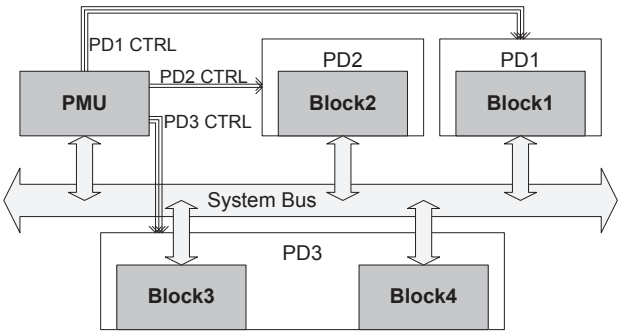

Fig. 2. A system view with a power-management unit.

The PMS library [9] simplifies the ESL power-management specification by abstracting away the low-level power-management details (e.g. supply nets, power-management elements). The introduced specification method also uses the already described concepts of power domains and power modes. However, since the power-management elements are not present in the SystemC/PMS-based specification, the power state cannot be defined by their control signals. The PMS uses five predefined power states, which imply application of various power-reduction techniques. The *NORMAL* power state represent basic operation of the system block with no power-reduction technique applied. The *HOLD* power state stops operation of the block, which will be implemented by clock gating and operand isolation. The *DIFF_LEVEL* power state adjusts voltage and frequency level of the block (there can be multiple such states specified, differentiated by a number appended in the identifier, e.g. DIFF_LEVEL(5)). Thus, it implies voltage and/or frequency scaling (also multi-voltage or multi-clock designs). The *OFF* power state powers-down the block; thus, the power-gating technique is implied. And finally, the *OFF_RET* power state also powers-down the block but its state is retained – i.e. power gating with state retention is implied. The clock-gating, operand-isolation and power-gating techniques require isolation of input/output signals of the block. The multiple voltages in the design imply a power switch and the multiple operation frequencies of a block imply a clock switch. Power gating also requires a power switch (switching between active and off voltage levels). Besides, power gating with state retention requires retention cells to be used inside the block. These implications are the basis of the new method, proposed in the next section.

IV. The Proposed Method

There is no PMU in the ESL power-management specification using SystemC/PMS [9]. Power-mode switching is specified directly in the functional model. However, it is important to estimate the future PMU complexity, because it can influence a designer's selection of power-management strategy. One way of doing so is a high-level synthesis (HLS) of PMU (either manual or automated), as illustrated in Fig. 3. It generates an RTL model of the PMU corresponding to the ESL power management. Such a PMU model can be then

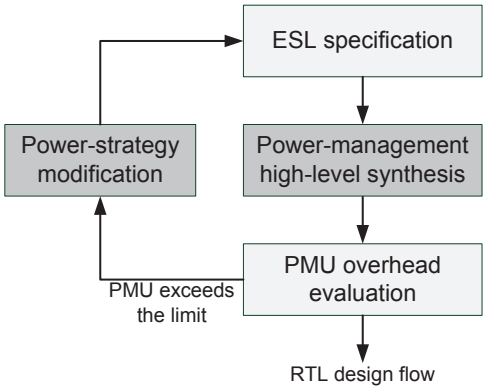

Fig. 3. PMU overhead determination using the high-level synthesis.

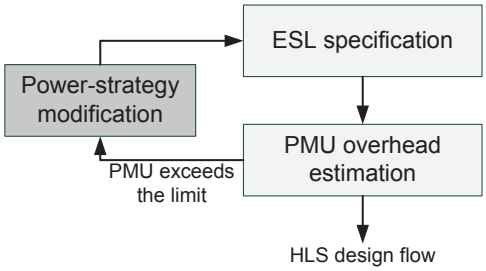

Fig. 4. PMU overhead determination using the proposed method.

synthesized to the gate level and its power and area can be estimated using the professional tools (e.g. Synopsys Power Compiler). However, this process is not very suitable for determination of PMU complexity, since it requires too many steps and uses too many tools. Exploration of various power-management strategies is thus complicated.

The proposed method (illustrated in Fig. 4) is based on a modified PMU high-level synthesis algorithm [8]. It analyses the ESL power-management specification and estimates the complexity of PMU that would be synthesized. The core is to evaluate the abstract power states in power domains, correlation between power domains (communication between their components), and power-states setting in power modes. The algorithm estimates which power-management elements are required (e.g. isolation logic, power switches, etc.), and thus how many control signals are required. It also estimates which intermediate power modes are required in order the power management to function correctly (e.g. the isolation must be activated before powering a component down).

The PMU is basically an application-specific finite-state machine. The power-state encoding corresponds to the control signals for power-management elements specified in UPF/CPF. The number of control signals is dependent on the abstract power states specified for each power domain in SystemC/PMS specification and their implication discussed in Section III. Two power domains can share the control signals if they are always activated/deactivated at the same time (i.e. they are simultaneously entering and leaving the same abstract power state). Otherwise, each power domain must have its own control signals. There are four types of required control signals:

- *Clock-switch control* – the number of such signals depends on a number of frequencies the domain can operate at (deduced based on the *NORMAL* and *DIFF_LEVEL* states).
- *Power-switch control* – the number of such signals depends on a number of voltage levels the domain can operate at, including cut-off voltage (deduced based on *NORMAL*, *DIFF_LEVEL*, *OFF* and *OFF_RET*).
- *Isolation control* – a single signal for a domain that can operate in at least one of the *HOLD*, *OFF*, and *OFF_RET* power states.
- *Retention control* – a single signal for a domain that can operate in the *OFF_RET* power state.

The number of required intermediate power modes is deduced based on the abstract power modes and allowed switching among them. For each power domain, it is analysed from which to which abstract power state it can be switched. If such a state change is not functionally correct at the RTL, a transient state is required. Then, each new unique combination of power states (including transient) in power domains represent an intermediate power mode.

Based on the estimated number of all control signals (for all power domains), we can predict a number of flip-flops to keep the PMU state and a number of input/output ports of the PMU. On the other hand, based on the estimated number of all power modes (including intermediate modes), we can predict a number of PMU states (representing power modes) and also a number of branches in the transition logics of the PMU. This information directly corresponds to the size of the PMU. Thus, the estimated parameters (the number of control signals and the number of all power modes) are used to determine the PMU complexity.

In order to derive some power estimation form this PMU complexity, the impact of the two parameters must be weighted. We have defined the estimated power (P_{PMU}) to be

$$P_{PMU} = x.CS + y.PM \quad (1)$$

where x is a weight representing impact of the number of control signals (CS) and y is a weight representing impact of the total number of required power modes (PM), including intermediate modes. Since the actual power consumption is highly dependent on the implementation technology, the weights of the parameters varies and must be calibrated before the estimation can be relevant.

The proposed method uses a really fast estimation algorithm, because it just uses static analysis of the ESL power management. Compared to the previous way of obtaining the power/area estimation, it provides significant speed-up. Using a single algorithm, we can obtain the PMU overhead estimation in few seconds directly form the ESL specification.

V. Experimental Results

For evaluation reasons, we have used Synopsys Power Compiler tool [16] to compute the power (*Power* in Table I)

TABLE I
ACHIEVED POWER ESTIMATION RESULTS

CS	PM	PMU VHDL	Power [μW]	Estimation	Δ [%]
3	5	4619	96,6	110,3	14
3	4	3300	114	107,8	-5
4	5	4452	167	142,9	-14
7	12	8658	260	258,2	-1
10	14	10094	355	361	2
10	8	7017	379	346	-9
10	49	63304	537	448,5	-16
13	46	35205	562	538,8	-4
13	106	199866	594	688,8	16
16	114	142992	705	806,6	14
19	132	214122	815	949,4	16
17	117	107068	873	846,7	-3
17	101	131478	891	806,7	-9
21	65	67691	961	847,1	-12
18	346	586303	1398	1451,8	4

of the experimental PMUs. The NanGate_15nm_OCL [17] technology library has been used in the experiments. The results are provided in Table I. The first two columns represents the estimated numbers of control signals (CS) and power modes (PM), based on the analysed system-level power-management specification. The next column (*PMU VHDL*) represents the number of characters required for description of the synthesized PMU model in VHDL. It is synthesized using the high-level synthesis method presented in [8]. It can help to imagine the complexity of the actual PMUs at the RTL. *Estimation* represents the estimated power values. For the estimation, we have stated $x = 32.6$ and $y = 2.5$ of (1). The last column reports a difference between the estimated and computed power values.

The results show that the selected parameters (the number of control signals and the number of power modes) can be indeed used for representation of the PMU complexity. Also, we can see that the estimated power roughly corresponds to the power reported by Power Compiler (inaccuracy up to 16%). It must be noted that the PMU is not present in the system-level power-management specification. There are no power-management control signals and the number of abstract power modes at the ESL is much lower than the eventual number of power modes at the RTL (since they also include intermediate power modes). Only relations among power domains and structural dependencies among the SystemC modules are used in the estimation. Thus, the proposed method is usable for estimation of future PMU power requirements and it represents a significant improvement in this area.

VI. CONCLUSION

The existing system-level power-management exploration methods lacks the automation required to quickly evaluate overhead of the specified power management, especially the power-management unit (PMU). We have proposed such a method, which uses SystemC/PMS specification [9] and is based on a modified power-management high-level synthesis algorithm. It is able to predict complexity of the PMU (it is not present in the specification), based on estimation of a number of required intermediate power modes and a number of required control signals for power-management elements implementing power management at lower abstraction levels.

The proposed method is especially useful for fast exploration of various power-management strategies at the system level. It complements the methods that can estimate power of system components under power management, but do not take into account the introduced PMU overhead. Furthermore, it can be used instead of methods that require high-level synthesis for this purpose, since it is much faster.

ACKNOWLEDGMENT

This paper was created with the support of the Ministry of Education, Science, Research and Sport of the Slovak Republic within the Research and Development Operational Programme for the project "University Science Park of STU Bratislava", ITMS 26240220084, co-funded by the European Regional Development Fund. This work was also supported by the Slovak Scientific Grant Agency (VEGA 1/0836/16) and the Slovak Research and Development Agency (APVV-15-0789).

REFERENCES

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

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

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

[4] K. Grüttner, P. A. Hartmann, K. Hylla, S. Rosinger, W. Nebel, F. Herrera, E. Villar, C. Brandolese, W. Fornaciari, G. Palermo *et al.*, "The COMPLEX reference framework for HW/SW co-design and power management supporting platform-based design-space exploration," *Microprocessors and Microsystems*, vol. 37, no. 8, pp. 966–980, 2013.

[5] 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)*. IEEE, 2016, pp. 227–228.

[6] K. Gagarski, M. Petrov, M. Moiseev, and I. Klotchkov, "Power specification, simulation and verification of SystemC designs," in *2016 IEEE East-West Design & Test Symposium (EWDTS)*. IEEE, 2016, pp. 1–4.

[7] A. Qamar, F. B. Muslim, J. Iqbal, and L. Lavagno, "LP-HLS: Automatic power-intent generation for high-level synthesis based hardware implementation flow," *Microprocessors and Microsystems*, vol. 50, pp. 26–38, 2017.

[8] 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)*. IEEE, 2015, pp. 63–68.

[9] ——, "Power-management specification in SystemC," in *Proceedings of the 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*. IEEE, 2015, pp. 259–262.

[10] D. Greaves and M. Yasin, "TLM POWER3: Power estimation methodology for SystemC TLM 2.0," in *Models, Methods, and Tools for Complex Chip Design: Selected Contributions from FDL 2012*, ser. Lecture Notes in Electrical Engineering, J. Haase, Ed. Springer International Publishing, 2014, vol. 265, pp. 53–68.

[11] 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*. San Jose, CA: EDA Consortium, 2013, pp. 1609–1614.

[12] 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)*. IEEE, 2013, pp. 45–50.

[13] 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)*. IEEE, 2014, pp. 112–119.

[14] Y. Xu, R. Rosales, B. Wang, M. Streubűhr, 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*, Berlin Heidelberg, 2012, pp. 37–49.

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

[16] Synopsys, "Power Compiler: Power optimization in Design Compiler," 2017. [Online]. Available: www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/power-compiler.html

[17] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, "Open cell library in 15nm FreePDK technology," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design (ISPD '15)*. New York, NY, USA: ACM, 2015, pp. 171–178.