Contribution to Automated Generating of System Power-Management Specification

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

***Abstract*—Nowadays, the electronic system design is constrained by several factors. One of them is the system power consumption, since many devices have limited energy source (e.g. Internet of Things sensor devices) – they run on batteries or are powered by energy harvesting from environment. The so-called power management is usually adopted during system design to minimize power consumption of the system under development, which enables to apply very popular power-reduction techniques, such as clock gating, power gating, or voltage and frequency scaling. The specification of power management is not an easy task, and therefore in our previous work, we have proposed a simplification method by increasing abstraction and automation of the design process. In this paper, we are taking the design automation one more step forward by proposal of a method that enables automated specification of system power management using the system architecture and abstract simulation results. The automatically generated power management can reduce the power by tens of percent as showed by the experiments.**

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

I. Introduction

The continuously increasing power density in modern complex Systems-on-Chips (SoCs) causes packaging and cooling problems and can potentially damage the device [1]. Therefore, the power reduction is one of the key issues during the system development. Another point of view is that many mobile devices (e.g. smart phones, laptops, or wearable electronics) and Internet-of-Things (IoT) sensor devices [2] (e.g. video camera, or humidity and temperature monitoring device) are powered by batteries or by limited amount of energy harvested from the system environment. For such devices, energy efficiency is very important and reduction of energy consumption can prolong lifetime of the device, reduce maintenance costs, or can enable new features and more functions of the device.

Therefore, in any modern electronic system, the power must be properly managed. The power management as a concept enables to switch the system operating modes according to current needs – i.e. not used components are powered down, or at least switched to some low-power state to save the available energy. This is accomplished by application of the power-reduction techniques, such as clock gating, power gating, or voltage and frequency scaling [1]. The supporting logics and structures behind these techniques and their introduction to the system design are rather complicated, which prolongs the development time. Therefore, the design-automation techniques are evolving also in this sphere.

The International technology roadmap for semiconductors [3] suggested the adoption of an abstract electronic system level (ESL) in the design process to cope with the increasing design complexity and to improve the design productivity. The ESL slowly becomes the design starting point in the industry and there are methods developed to raise the abstraction level for adoption of power management into the design. In our previous work, we have also proposed such a method in [4], which was targeted to system-level specification of abstract power management for a SoC design. We have also integrated automated power-management high-level synthesis [5] to the design process to speed-up the system development. And finally, we have enhanced power-management verification at early design stages [6], which also contributes to development process speed-up.

In this paper, we are going one more step forward in automated power-management specification. We have proposed a method that takes a system-level functional model (i.e. a SystemC algorithmic model) as an input and automatically provides the abstract power-management specification for this model (in SystemC/PMS). It is especially useful for designers who are not familiar with power-reduction techniques to design an energy-efficient low-power system.

The paper is organized as follows. In the next section, the existing works related to this problem area are analyzed. The Section 3 is focused on description of the proposed automation method for power-management introduction. We provide experimental results in Section 4, and Section 5 concludes the paper.

II. Related Works

There exist several works automating the transformation from a more abstract specification of power intent to a more detailed implementation. For example, professional tools (e.g. Power Compiler [7] or Genus Synthesis Solution [8]) commonly accept a power-intent specification in the standardized UPF (Unified Power Format) [9] or widely spread

This is an accepted version of the published paper:
D. Macko, "Contribution to automated generating of system power-management specification," in 2018 IEEE 21th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2018, pp. 27-32.
doi: 10.1109/DDECS.2018.00012
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8410501

CPF (Common Power Format) [10] at the RTL (Register-Transfer Level) abstraction, and automate its transition to lower levels. However, the specification of power intent at RTL is complex and it must itself be automated to increase productivity.

Using a system-level specification and automated UPF/CPF synthesis is obvious way to deal with the complexity. Such an approach was used in [11]. It proposed a SCPower extension for SystemC modelling, which enables to specify power intent directly in SystemC design, perform power-aware verification, and generate the corresponding UPF file. However, it uses the RTL power-intent abstraction level, and is focused more on enabling power-aware verification of SystemC designs. Therefore, from a specification perspective it represents just another way of annotating the required information. Thus, the supported UPF generation represents rewriting of that information from one form to another. Another approach [12] proposed a LP-HLS methodology. It enables to specify power intent regarding the usage of power gating technique in SystemC model and generates the corresponding CPF specification at RTL. It offers a simplified specification with respect to [11], but it enables a limited number of power-reduction techniques (power gating, and perhaps operand isolation). The used abstraction is also not very suitable for system level. The methodology requires also specification of used cells and isolation rules, or specification of isolation activating and power-gating activating signals. Thus, it also seems more like a rewriting procedure than high-level synthesis of CPF specification. In our previous work [5], we have incorporated even more abstraction, in which all the control signals and power-management low-level elements are synthesized automatically. The proposed PMHLS method not only synthesizes specification in UPF, but also description of the corresponding power-management unit in RTL functional model. Beside the power-gating and isolation techniques, it supports also voltage and frequency scaling, clock gating, and multi-voltage designs.

Based on the current state-of-the-art, we focused our work, presented in this paper, to automate the power-management specification even more. The goal is to automatically specify as much power-related information as possible to alleviate design aspects on which a designer must focus at early design stages.

III. A New Power-Management Specification Automation Method

In order to understand what aspects can be automatically specified for proper introduction of power management into the system design, we firstly present the underlying specification model. If we selected the standard power intent specification in UPF, the automation would be very complicated. The reason is that it was primarily developed for RTL and lower abstraction levels. The result is that a designer must specify low-level details regarding the power management, such as power supply networks, power switches, or isolation and retention cells. Such information is unsuitable for early design stages, where the designer must primarily focus on system functional aspects. And introduction of the power management at later design stages makes the system verification unbearable. Therefore, we have chosen the abstract power-management specification model, proposed in our previous work [4]. This model is briefly described in the following subsection.

A. The Underlying Specification Model

The abstract power-management specification model is based on five key aspects: power domains with power states, power modes, component assignments, performance levels, and policy.

1) Power domains

A power domain is a collection of system components that operate in the identical power states during the system lifetime. The specification of a power domain includes its identifier and a set of power states, in which the domain components can operate. There are five predefined abstract power states, summarized in Table 1.

2) Power modes

A power mode of the system is a specific combination of power states in power domains (i.e. one power state per domain). The specification of a system power mode consists of its identifier and a sequence of power states. The order of specified power states is important, because the first power state is the state of the firstly specified power domain, the second state relates to the second domain, and so on. The number of states in a power-mode specification has to match the number of power domains in the system.

3) Component assignments

The system components must be explicitly assigned to the specified power domains in order the system be split into parts that can be powered separately (i.e. operate in different power states). The specification connects identifier of a power domain with an identifier of a component. There can be multiple components in a power domain, but one component cannot be assigned to multiple domains.

4) Performance levels

A performance level represents a specific combination of voltage and frequency values. It must be explicitly specified for active power states (NORMAL and DIFF_LEVEL), since it

TABLE I. The Predefined Abstract Power States

Power State	Overview
OFF, OFF_RET	A component is not powered in order to save energy. In case of OFF_RET, the state of memory elements of the component is retained. These power states represent application of the power-gating technique.
HOLD	A component is powered, but it stops operation in order to reduce dynamic power consumption. It represents application of the clock gating and operand isolation techniques.
DIFF_LEVEL	A component operates at the performance level higher or lower than the basic performance level of the system (i.e. either the voltage or operation frequency differs). Such a power state enables to use multiple fixed voltages or frequencies in the system. It also enables application of dynamic voltage and frequency scaling power-reduction techniques.
NORMAL	A component operate at the basic performance level of the system (i.e. it uses the primary voltage and clock supplies). Meaning that no power-reduction technique is explicitly applied in this power state.

cannot be deduced. HOLD has the frequency nulled and the voltage is the lowest voltage level specified in the domain. OFF and OFF_RET have both parameters nulled. The specification includes the power state, the voltage value and the frequency value.

5) Power-managment policy

The power-management policy represents the specification of a control function, which defines how and when the power mode of the system is switched. It is the key aspect, which enables dynamic power management during the system runtime. It can switch the system into some energy-saving power mode or it can redirect the power from currently unneeded components to others, what can speed-up the operation. Since the power-management policy is part of the functional specification, there is no predefined way of specification of this aspect – i.e. usual software or hardware functional design can be used.

B. The Specification-Automation Method Requirements

Based on the analysis of related works and using of the previously described underlying specification model, we have stated the following requirements for the method.

- Analysis of the system model – the method must be able to parse the functional specification of the modelled system and analyze structural dependencies among the components.
- Analysis of the simulation results – in order to make proper decisions regarding the power-management specification, the method must be able to analyze the runtime behavior of the system.
- Partitioning of the system into power domains – based on analysis of the components activity, the method should group the components that are active during the same time periods into the same power domains.
- Selection of power states for power domains – based on the analysis of the system behavior, the method must specify suitable power states, in which the components can operate.
- Selection of allowed power modes – the method must be able to provide a list of actually required power modes, in order to minimize the generation of unneeded power-management elements at later design stages.
- Generation of the power-management specification – the method must automatically generate the source code specifying the mentioned aspects.
- Integration into the system model – the method should integrate the generated specification directly into the system functional model.
- Verification of the generated specification – the method should provide means to check whether the specification is complete and consistent with the functional model.

The proposed method fulfilling these requirements is more closely described in the following subsection.

C. The Proposed Method

As one can notice from the stated requirements for the method, it consists from multiple parts: analysis part, specification part, generation part, and verification part. The method overview is illustrated in Fig. 1.

1) System analysis

This part of the method is focused on the first two requirements regarding analysis of the system model and system-level simulation results. An input of the analysis part is a SystemC model of the system and simulation results in the VCD (Value Change Dump) form. The goal is to parse the model source files and to obtain a hierarchy of SystemC modules (representing system components definitions), their identifiers, input/output and state variables, identify clock variables for each component, identify a top module (representing the whole system), and identify vector-variables representing interconnections among the components. The next step is to analyze the VCD file and to obtain time intervals, in which the system components are active and in which they are just waiting. A component is considered active, if its variables are switching values during that time interval. All the analyzed objects and data are then filled into the prepared structures and passed to the specification part of the proposed method.

2) Power-related specification

This part covers the next three requirements regarding specification of power domains, power states, and power modes. Using the analyzed information about time intervals of components activity, the components that are active in the

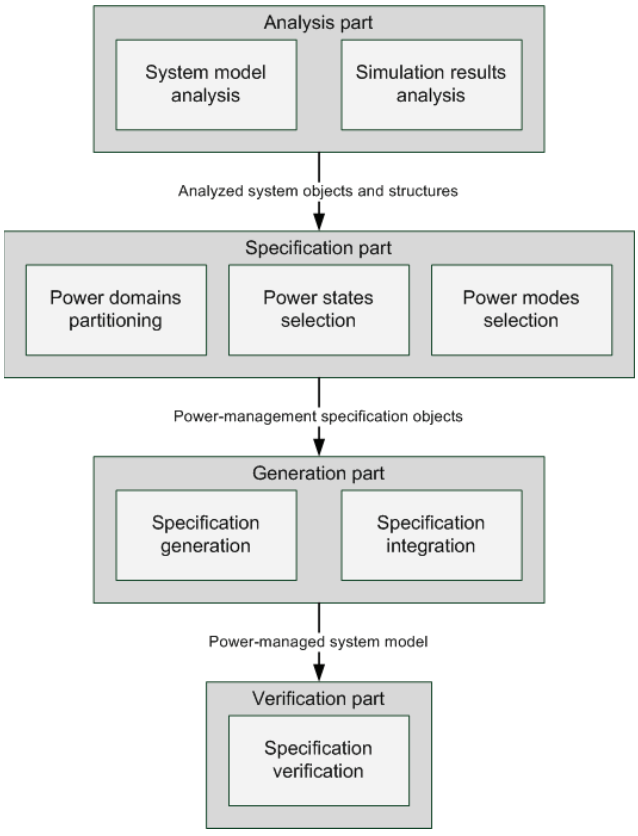

Fig. 1. The proposed method overview.

same time intervals and inactive in the others are assigned into the same power domain. The result is that these components are managed together, and therefore the power management is more efficient.

Then, the power states are specified for power domains. For inactive periods, a power domain must contain some passive power state (HOLD, OFF, OFF_RET). Which of the passive states is the most suitable depends on length of time intervals. Since both off power states represent the power-gating technique application, it takes time to power down and power up the components. Therefore, there is a threshold value representing a limit, above which it is beneficial to power down the domain and under which the HOLD power state is used (just isolating the input variables is quick enough). This limit depends on the implementation technology, thus must be set by a designer. If the components of a power domain contain state variables (i.e. memory elements), the OFF_RET state is used instead of OFF. Usually, there are multiple inactive time intervals with various lengths. Thus, both HOLD and OFF (or OFF_RET) power states can be specified for a power domain.

Then, a decision has to be made about using NORMAL or some DIFF_LEVEL power state for active time intervals. A default active state is NORMAL. The DIFF_LEVEL is a group of power states, where each of them operates at a different voltage and/or frequency level. The number of these states has to be provided by a designer, along with their voltage and frequency parameters (depending on what voltages and frequencies the implementation technology will support). For selections of a DIFF_LEVEL power state, inter-domain communications are analyzed. When a situation is found that some domain is active and another connected domain is waiting for its computation to finish, the power of the first domain can be increased to shorten the operation time. On the other hand, the power of the waiting domain can be lowered to reduce power consumption. In such cases, DIFF_LEVEL power states are specified for these power domains.

When the allowed power states are specified for all power domains, the power modes can be specified. Such a specification includes all occurring combinations of power states in power domains during the simulation time. These represent the allowed power modes of the system, and the system cannot reach other modes.

3) Generation of the specification

The generation part of the proposed method is focused on generation and integration requirements. The goal is to incorporate the power-related specification source code into the original SystemC model in a compatible way. As an output, we have proposed using the SystemC/PMS library [4]. The power domains and power modes are instantiated in the declaration part of the identified top SystemC module. The power states of the power domains, the components assignment, the performance-level definitions, and the power-modes specification are included in the constructor of this module. The only part of the abstract power-management specification that was not automatically generated is the power-management policy. It is dependent on the system function and a designer must design that. Thus, the proposed automation includes the underlying support for system power management, but the switching between power modes is not automated. For illustration, an example of a part of system description in SystemC, which is an input to the proposed method, is provided in Fig. 2a. A part of the enriched system model with a generated power-management specification is illustrated in Fig. 2b (the highlighted text represents the added power-related specification aspects).

4) Power-management verification

This part includes the last requirement stated for the proposed method. The verification is focused on the enriched system model that includes functional specification of the system as well as the power-management specification. To check consistency of the power-management specification with the functional aspects, we propose to use the previously developed power-management static analysis [6]. It formally verifies the consistency and also the completeness of the abstract power management specified by SystemC/PMS. Thus, a designer is ensured that the specification is correct and can proceed with the high-level synthesis process.

a)

```
SC_MODULE(system1){
  ... //omitted source code
  mu0 CPU;
  ram0 MEM1, MEM2;
  ... //omitted source code
  SC_CTOR(system0): CPU("CPU"), MEM1("MEM1"),
MEM2("MEM2")
    {
      CPU(clk,Abus,Dbus,reset,MEMrq,Rnw);
      MEM1(clk,Rnw,MEMrq1,Abus,Dbus);
      MEM2(clk,Rnw,MEMrq2,Abus,Dbus);
      SC_METHOD(select);
        sensitive << Abus;
    }
};
```

b)

```
SC_MODULE(system1){
  PowerDomain PD1, PD2;
  PowerMode PM1, PM2, PM3;
  ... //omitted source code
  mu0 CPU;
  ram0 MEM1, MEM2;
  ... //omitted source code
  SC_CTOR(system0): CPU("CPU"), MEM1("MEM1"),
MEM2("MEM2")
    {
      CPU(clk,Abus,Dbus,reset,MEMrq,Rnw);
      MEM1(clk,Rnw,MEMrq1,Abus,Dbus);
      MEM2(clk,Rnw,MEMrq2,Abus,Dbus);
      PD1 = PD(NORMAL);
      PD2 = PD(HOLD, NORMAL, DIFF_LEVEL(1));
      PD1.AddComponent("CPU");
      PD2.AddComponent("MEM1");
      PD2.AddComponent("MEM2");
      PM1 = PM(NORMAL, HOLD);
      PM2 = PM(NORMAL, NORMAL);
      PM3 = PM(NORMAL, DIFF_LEVEL(1));
      POWER_MODE = PM1;
      SetLevel(NORMAL, 0.8V, 50MHz);
      SetLevel(DIFF_LEVEL(1), 0.72V, 30.0MHz);
      SC_METHOD(select);
        sensitive << Abus;
    }
};
```

Fig. 2. An example of partial input (a) and output (b) SystemC files.

IV. Experimental Results

Two experiments were conducted. The first one was oriented towards synthesis of the enriched ESL system model to the RTL model and estimation of its power consumption using a professional tool. We have used two simple models: a system consisting of one microprocessor communicating with two memories (System1), and a system consisting of one microprocessor and one memory (System2). The verified SystemC/PMS models were synthesized using the power-management high-level synthesis method, proposed in [5], which generated UPF power-intent specification equivalent to the abstract specification compatible with functional RTL model of the system. For estimation purpose, we have used Synopsys Power Compiler [7] and the NanGate_15nm_OCL [13] technology library. Since the power-mode switching is not automatically generated using the proposed method, default switching activity was used by the tool. The results of the experiment, provided in Table 2 (the values represent average total power consumption reported by Power Compiler), show that the automatically generated power-management specification is beneficial regarding reduction of the system power consumption.

Since the power-management switching was not modelled in the first experiment and the results might be misleading, we have conducted another one. Just for the experimental purpose, we have automatically generated also power-mode switching for System2 design by using time points from the analysis of ESL simulation results. We have functionally described switching of the power modes using the typical if-else language construct. If the current simulation time reached a specific time point, the power mode was switched to another one based on information that was used for selection of power states. The syntactical and semantical correctness of the enriched model was also verified by functional simulation in Modelsim. It was however difficult to synthesize an RTL model form such a specification; therefore, we have used ESL power estimation using the PESL [14] tool instead. This tool estimates the dynamic energy consumption of a power-managed system in SystemC/PMS using a modified Hamming-distance computation. For static power comparison, we have used the tool proposed by [15]. It estimates the component size using its SystemC description, which roughly represents its static power, and uses the power-state-based coefficients to compute static energy of the power-managed system. Because of the abstraction, these two tools cannot be used for estimation of absolute values for energy consumption, but they can be used for a relative comparison of two alternative ESL designs.

TABLE II. RTL Power-Comparison of Original and Power-Managed System using Power Compiler

Design	Original	Power managed	Difference
System1	27.001mW	2.085mW	-92.28%
System2	63µW	46.5µW	-26.19%

TABLE III. ESL Power-Comparison of Original and Power-Managed System

System2 design	Original	Power managed	Difference
Dynamic energy	80314	57380	-28.56%
Static energy	1015.8	969.3	-4.58%

In our case, we have used these tools to compare energy consumption of a system without and with automatically generated power-management specification. The results are provided in Table 3. The values in the table are without units (due to abstract computation they cannot refer to Watts or Joules). The values for dynamic energy represent summed Hamming distances of variables bit-vectors during the simulation time, taking into account the specified power states for specific time intervals. Similarly, the values for static energy represent summed static energy consumptions during time intervals adjusted according to specified power states in those intervals. In this experiment, both tools confirmed that the automatically generated power-management specification indeed reduces the power consumption.

Just for clarification, the proposed automated generation of power-management specification is based on static analysis of system model and simulation results. Therefore, there is only very small time-overhead of using the proposed method. In our experiments, it took few seconds to analyze the model and generate the appropriate power-management specification. However, it depends on the size of the model and the size of simulation results (which could be very long). But compared to manual specification of power management, which could be erroneous and require debugging (hours or even days of effort), the proposed method provides undeniable benefits.

V. Conclusions

In this paper, we have proposed a method for automation of power-management specification. The input of the method is a system functional model in SystemC and ESL simulation results in VCD. The output is an enriched system model, which includes the power-management specification using SystemC/PMS. The method minimizes designer's input regarding power aspects, while all the decisions are made automatically (such as division of the system into power domains, selection of suitable power states for power domains, and selection of allowed power modes of the system). The experimental results supported benefits of the proposed method. The automatically generated power-management specification reduced the power significantly.

The proposed method is usable for automated application of power-reduction techniques by implementing power management. Although the specified power management might not be optimal, it is especially beneficial for designers not familiar with low-power systems design. They do not even need to possess knowledge of aspects required for power-management specification (in UPF or SystemC/PMS). It is also useful for designers to automatically obtain the first power-management alternative, which can be manually modified to further optimize the power management.

Limitation of the proposed method lies in dependency on ESL simulation results. The simulation must be representative in order to obtain well-balanced power management. Thus, it is currently suitable especially for systems that execute the same function during their lifetime. Such a method is not suitable for general-purpose devices, which function is not known beforehand. This opens space for further enhancement of the proposed method in future work.

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), the Slovak Research and Development Agency (APVV-15-0789), and the Slovak Cultural and Educational Grant Agency (KEGA 011STU-4/2017).

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] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang and V. C. M. Leung, "Lightweight management of resource-constrained sensor devices in Internet of Things," IEEE Internet of Things Journal, vol. 2, no. 5, pp. 402-411, Oct. 2015.

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

[4] 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.

[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, "Verification of power-management specification at early stages of power-constrained systems design," Journal of Circuits, Systems, and Computers, vol. 26, no. 8, 1740002, Aug. 2017.

[7] Synopsys, "Power Compiler: Power optimization in Design Compiler," 2014. [Online]. Available: www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/power-compiler.html

[8] Cadence Design Systems, "Genus Synthesis Solution: Massively parallel RTL synthesis and physical synthesis," 2015. [Online]. Available: https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus-synthesis-solution-ds.pdf

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

[10] S. Carver, A. Mathur, L. Sharma, P. Subbarao, S. Urish and Q. Wang, "Low-power design using the Si2 common power format," IEEE Design & Test of Computers, vol. 29, no. 2, pp. 62-70, April 2012.

[11] 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), 2016, pp. 1–4.

[12] 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.

[13] 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 International Symposium on Physical Design (ISPD '15), ACM, 2015, pp. 171–178.

[14] J. Erdelyi, "Estimation of power-management effect on dynamic power consumption," in Proceedings of the 13th Student Research Conference in Informatics and Information Technologies, 2017, pp. 218-221.

[15] T. Rychvalský, Estimation of Power-Management Effect on Static Power Consumption, FIIT STU Bratislava, 2017. Master thesis.