Adoption of Abstract Power-Management Specification to FPGA-Based Design

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

***Abstract*—Modern system-on-chip designs are characterized by their high complexity. It results in an increased number of transistors in a single chip, size of which is continuously decreasing due to market requirements. These issues have negative impact on reliability of the system, mainly because of the overheating of the device due to its power requirements. The result is that designers have to implement power-management techniques to reduce power. Due to high complexity of the systems, the modern trend is to specify power management at the high abstraction levels. However, the existing low-power design methodologies mostly target the application-specific integrated circuit (ASIC) devices, because they offer more flexibility in adoption of various power-reduction techniques. In this paper, we target field-programmable gate array (FPGA) designs. We adopt the previously developed low-power systems design methodology to FPGA platforms, enabling power management at early design stages. A high degree of automation is involved in the adopted methodology, which ensures fast development of FPGA-based low-power systems. The utilized abstraction and automation is suitable especially for novice designers, such as students; therefore, the proposed methodology is useful in education. The experiments illustrate power-management overhead of the offered methodology as well as its usefulness for modern designs.**

***Keywords*—design; FPGA; low power; power management; power reduction; system level**

I. Introduction

The growing interest in the Internet of Things (IoT) causes common devices surrounding our everyday life to process information and to communicate. The field-programmable gate arrays (FPGAs) are adopted as IoT devices [1] due to their low manufacturing costs and low time-to-market of the implemented systems. However, many IoT applications require battery-operated devices or some sort of power harvesting. This is where FPGAs are not very efficient [2]. They have been primarily developed to increase offered functionality. Their low-power operation is targeted only in recent years. Over time, the FPGA devices have become highly integrated complex circuits, programmability of which comes at a cost of high area overhead [3]. It essentially contradicts with low-power operation.

Hardware designers have developed many power-reduction techniques to overcome the power-density problem and to prolong operation time on battery [4]. The most popular of these techniques include clock gating, power gating, scaling of voltages and operating frequencies. Some low-power FPGAs are available today, which use low-power techniques [2], [3]. However, the system that will be implemented (programmed) in this FPGA must be also designed to be low power. There are recommendations available for design or synthesis and place-and-route settings in [5], [6]. However, the power management is not utilized in the application design. The problem in FPGA-based design is that the designer is limited by the FPGA device itself. Therefore, the power-reduction techniques (such as power gating) must be enabled in the selected device. The designer should explore capabilities of the used FPGA device and implement a power management in the design which utilizes these capabilities.

A. Standard Power Management

Power management enables to control power requirements of system hardware components by switching them into various power states. Some components can be powered down, while other components can temporarily increase their performance. According to the low-power integrated circuits design and verification standard called UPF (Unified Power Format) [7], the power state corresponds to the supply voltage level at which the component operates. Many power-reduction techniques are adjusting the operation frequency of the component in addition to the supply voltage. Therefore, the power state is represented by a voltage-frequency pair in more general view. For more efficient power management, UPF utilizes the concept of power domains. A power domain contains the system components which always operate in the same power state. The power-management unit (i.e. the system component handling switching between power states) then controls the power states of the whole domains, not the individual components. Most of the modern low-power systems design methodologies are based on the UPF standard.

B. System-Level Approach

The UPF-based low-power design starts at the Register-Transfer Level (RTL). The RTL is no longer feasible as a design starting point, because of the system complexity. The international technology roadmap for semiconductors (ITRS) suggests adoption of the system level of abstraction (ESL) in the design to efficiently specify, implement, and verify the systems [8]. An FPGA is often used as a prototype platform for expensive custom hardware designs. Rapid prototyping in FPGA is based on high-level synthesis, which enables description of the system in highly abstract programming languages (e.g. C, C++, SystemC) and its automated low-level implementation (it corresponds to the ITRS suggestion).

The designed system is functionally verified in an FPGA before it is implemented as an application-specific integrated circuit (ASIC). However, many low-power design methodologies target ASIC technology [9]-[15], and they are simply not usable in FPGA due to used modelling approaches and physical limitations of the devices. The low-power ASIC design cannot be then properly verified in FPGA, which contradicts it primary usage.

C. *Paper Overview*

This paper describes an adoption of the existing system-level low-power design methodology [15] to the FPGA-based designs. It uses an abstract power-management specification, which is restricted by the target FPGA device capabilities. It can be then automatically synthesized to FPGA design, as well as a custom ASIC design. It enables to easily explore various power-management architectures and select the most suitable one, resulting in the lowest power consumption. Such a methodology also enables FPGA prototyping of low-power systems, which is essential to speed-up development time of ASIC-based systems.

The paper is organized as follows. The next section briefly introduces the selected design methodology. The modification, enabling its adoption in FPGA technology, is described in Section IV. Experimental results in Section V support usefulness of the adopted methodology. The paper is concluded with the discussion on the further work in this research area.

II. Overview of Abstract Power Management

Design of modern complex digital systems starts at the system level in order to deal with the system complexity and to increase the efficiency of system development. On the other hand, to avoid power problems in the system and to reduce energy consumption of battery-operated devices, the implementation of power management is unavoidable in the complex systems. The low-power systems design based on the UPF standard starts at the RTL, which is not suitable for complex systems for reasons stated above. And to start system design at the ESL and implement power management at the RTL results in even more problems. ESL power estimations become highly inaccurate and such an intrusion into the design at later stage implies a need for complex verification, which takes too much time.

Designers have developed multiple methods to implement power management from the beginning at the ESL, such as [9]-[14]. The existing methods have had many drawbacks, such as insufficient abstraction, missing automation, or inadequate verification. Therefore, we have developed a new method of abstract power-management specification and integrated it to the novel low-power systems design methodology [15]. The cornerstone of the methodology is the automation based on high-level synthesis [16], which is increasingly used in FPGA-based design. Thus, this methodology has potential for adoption in FPGA-based design process.

A. *Methodology Overview*

The described methodology is built around the design flow, shown in Fig. 1. The key idea lies in selecting several UPF concepts, which are suitable for system level

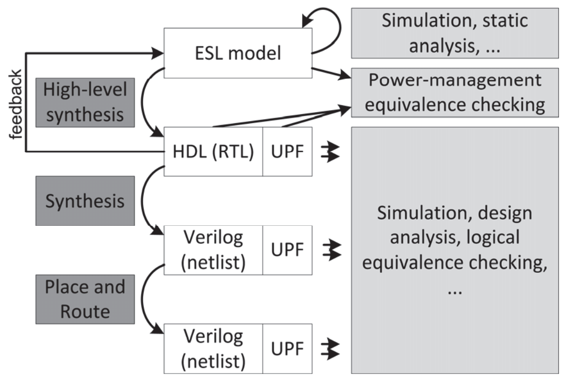
Simulation, static analysis, ...
ESL model
feedback
High-level synthesis
Power-management equivalence checking
HDL (RTL)
UPF
Synthesis
Verilog (netlist)
UPF
Simulation, design analysis, logical equivalence checking, ...
Place and Route
Verilog (netlist)
UPF

Figure 1. Overview of the low-power systems design flow

of abstraction. These include power domains, power states and their combinations in power domains (power modes), system components assignment to power domains, and switching between power states. The selected concepts are directly integrated into the ESL specification model in an abstract manner. Thus, the designer is able to specify both, the functionality and power management, in a single model. This is useful at the ESL, since the power management impacts the functionality – i.e. these aspects need to work together. The specification model is then refined during so-called abstraction-refinement process, which adds required details to the model. This process is complemented by various verification processes ensuring correct functionality and power-management specification.

When the model is sufficiently refined, it is automatically transformed into the RTL model during the high-level synthesis process. It extracts power-management information from the functional model and uses it to synthesize the power intent in UPF standard specification and also to synthesize the corresponding power-management unit (PMU) for generation of control signals for power-management elements (e.g. power switches, isolation and retention logic) in the synthesized UPF specification. Power-management high-level synthesis also generates corresponding assertions useable for functional verification of the synthesized PMU and for coverage measurement. The developed power-management equivalence checking ensures that the synthesized UPF specification at the RTL corresponds to the abstract power management at the ESL. At the RTL, the existing widely-used tools are used to analyze the design and to provide the trustworthy power information. This information is then used (if needed) for modification of abstract power management in order to obtain alternative power architecture. Thus, the designer can explore various power-management architectures and selects the suitable one, taking into account tradeoff between performance, power, and area of the system. After the RTL stage, the design flow continues according to the UPF standard. Thus, the existing methods and tools can be used later in the design process as it is nowadays.

B. *Abstract Power Management*

The UPF concepts imply that the system is split into multiple power domains, which group the components that always operate in the same power state. The abstract power-management specification is based on five predefined power states, which represent an intention to use specific power-reduction techniques in the design after high-level synthesis. These states include:

- *Hold* – operation of the components is stopped. It represents implementation of clock gating and operand isolation power-reduction techniques.
- *Off* – power supply of the components is cut off. It corresponds to the power-gating technique.
- *Off_ret* – state of the components is saved before the power supply is cut off. It represents power gating with state retention.
- *Diff_level* – the components operate at the voltage or frequency level different from the basic level of the system. This state represents voltage scaling, frequency scaling, or multiple voltages and frequencies in the system.
- *Normal* – the components operate at the basic voltage and frequency level of the system. No explicit architectural power-reduction technique is used when this state is active.

The power state of the whole system is represented by the power mode in which the system operates. A power mode represents a specific combination of power states in power domains. The dynamic nature of power management is implemented by switching between various power modes. The switching itself is specified in the functional part of the system specification, because it directly influences functionality of the system.

Based on the specified information, the high level synthesis is able to generate power-management elements, which actually change the states of the components. Also power supply ports and power nets are not explicitly specified in the abstract power management. This information is implicitly given by the power domains relations and architectural dependencies of the components. This is the main reason why the abstract power-management specification is multiple times simpler than the standard UPF specification. Experimental results in [15] showed that SystemC-based abstract power management specification is approximately five times less complex than UPF specification, in terms of the number of characters.

III. Adoption to FPGA-Based Design

The selected methodology has been designed for extension of the current low-power design flow based on the UPF standard. Therefore, it is essentially coupled with the ASIC technology. However, some of the methods used in the methodology can also be used in FPGA-based designs. Since the current FPGA flows do not work with UPF specification, the power-management aspects resulting in UPF are not usable. They mostly involve multiple voltages in the system. However, the PMU generated by the power-management synthesis process is fully usable in FPGA technology. Therefore, this methodology can simplify the adoption of architectural clock gating, operand isolation, or multiple clock domains in the system. The designer is thus restricted to the normal, hold, and diff_level power states in the abstract power-management specification. However, the isolation cells are not automatically implemented (UPF flow is not used), and therefore the designer needs to design the clock-gating and isolation elements manually. The use of other power-management techniques depends on the FPGA device – whether it supports the power-gating technique and how many voltages it offers. The control signals are correctly generated by the PMU; however, the designer has to use them to control specific FPGA elements. For the simulation purpose, the assertions can be used as well. Thus, the usage in the FPGA design flows is oriented towards simplifying the power-management strategy algorithm specification and automated generation of the PMU, not to the automated implementation of power-management supporting logic and nets.

The design flow resulting from the modified low-power systems design methodology is illustrated in Fig. 2. We propose that the system design starts from a crude specification at the ESL, similarly to [15]. The power-management concepts (such as power domains, power states, and power modes) are integrated into the functional specification model; thus, the designers use one language and one specification style. During the functional specification, the designer should split the system architecture into power domains in an empirical way. The next step is to assign to these domains some power states and create the most basic power modes (e.g. for a normal operation and for a power-saving operation). The functional specification represents the system functionality; thus, a dependency between the components can be deduced. The components that will probably be used at the same time should be in the same power domain. On the other hand, a component that will be used only occasionally should have its own domain.

The abstract power-management specification does not influence the system functionality, i.e. the designer does not need to worry that the system function is disrupted because of the wrong power-management specification. Moreover, the designer should use the proposed power-management static analysis to create a structurally correct and consistent specification.

At first, the designer should restrict abstract specification to the normal, hold, and diff_level states representing different frequency levels. The specification (both the functionality and power management) then goes through abstraction-refinement process. When the refined system-level model is executable, the best way to refine abstract power management is to simulate the design. Based on the simulation results, the designer can observe which components are active at the same time, how long some components are inactive, and so on. During the simulation, the preliminary power-mode changes can be observed, and thus the designer can notice what would be the state of the components. For example, if the power

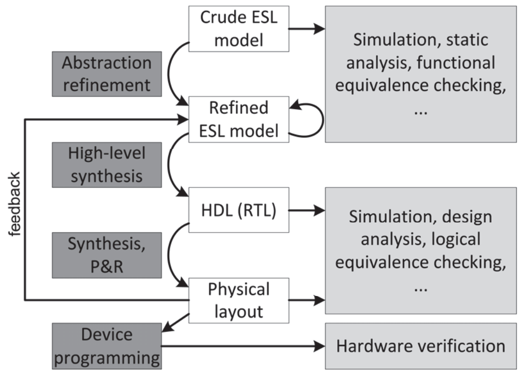

Figure 2. The proposed FPGA-based low-power systems design flow

mode informs that the state of some power domain is normal and the designer observes that the components in this domain are inactive, the designer should add some power-saving state (either hold or a lower-frequency diff_level state) to that power domain for such operation time.

At this stage, the designer should decide for a target FPGA device and analyze its capabilities. According to the available voltages, the performance levels (voltage-frequency pairs) should be specified and the diff_level states adjusted. Based on the availability of power-gating elements, the designer should analyze the possibility of usage of the off and off_ret abstract power states, which could replace the hold state in domains, components of which are inactive for longer periods of time. The reason is that the hold power state saves only dynamic power, while the off states also reduce the static power consumption (but produces higher delays).

After the system-level functional verification, the high-level synthesis usually takes place. During this process, the designer should firstly use the proposed power-management high-level synthesis, which extracts the power-management specification from the system-level model – in order to be synthesizable by a usual high-level synthesis tool (e.g. Vivado [17]). The power-management high-level synthesis process generates an RTL description of the PMU and informs the designer where the power-management supporting logic should be located (such as clock-gating, power-gating, or isolation elements). However, this supporting logic needs to be manually implemented by the designer, or if available, the existing elements in the FPGA device should be used and connected appropriately. During high-level synthesis, the automated static-analysis verification process informs the designer if some important aspects are missing in the abstract specification. The designer should then integrate the generated PMU into the top-level functional description of the system. The power-mode signal is driven by the power-mode register, generated by the functional high-level synthesizer. The clock signal is usually the basic clock in the system – it depends on the designer how fast the power-mode changes should be processed.

The next step is usually the RTL functional verification. To help alleviate the preparation overhead, the power-management high-level synthesis generates the assertions, monitoring the power-management control signals. Also, the designer should use the generated coverage assertions for a coverage-measurement purpose, and thus verify the design more robustly. After this step, the synthesis and implementation (including place and route process) take place. When the design is mapped to the FPGA device, the power can be analyzed with high accuracy. The usual FPGA synthesis tools directly offer this feature.

After the power analysis, the designer can modify the system-level specification and repeat the high-level synthesis. The designer can either modify both functionality and power management, or the power management only. The power-management high-level synthesis is much faster than the functional synthesis; therefore, it makes sense to explore various power architectures upon the synthesized system architecture. However, if the power-management strategy algorithm (power-modes changes in the functional design) or the power modes themselves are modified, the corresponding functional components have to be also resynthesized. In this way, the power-architecture exploration is achieved and the most suitable architecture should be used for the final device programming. The designer should carefully consider a trade-off between power, performance, and area.

IV. Experimental Results

Firstly, we show illustration of usage of the proposed methodology in a form of case study. Then, the area overhead of the adopted power management is estimated.

A. *Case Study*

To illustrate simple usage of the offered methodology, we use it on a case-study system. It represents a simple system-on-chip, consisting of one microprocessor *mu0* and two memories *ram0* (RTL description available in [18]). They are connected through a memory controller, which selects the memory with which the microprocessor will communicate based on the address at the address bus. We have described this system in SystemC code in order to mimic top-down design and to use the power-management high-level synthesis process. An abstract architecture view of this system is provided in Fig. 3.

Obvious thing, regarding the power management, is to set a low-power state to the memory that is not currently used by the microprocessor. Unfortunately, this is not something the existing synthesis tools could do automatically. Such an architectural power management needs to be created by the designer. For this purpose, we have used the proposed abstract power-management specification. We have divided the system into four power domains, one for each major component. Two power states have been assigned to each of these power domains, the active (normal) and inactive (hold) power state. We have specified two power modes, one for *M1* to be active and *M2* inactive, and the other one for vice versa activation. Power-mode switching is driven by the *MCU* decision about which memory is currently used. The early validation of the power management specification using the static analysis reported that the power domains for *CPU* and *MCU* are not necessary, because their states do not change in time (they are always in the normal power state). Therefore, only two explicit power domains remained in the specification, *PD_M1* and *PD_M2*. The resulted abstract power-management specification has taken only 237 characters.

The power-management high-level synthesis has generated the PMU, code-size of which has taken 2905 characters. For illustration, the synthesized power-management architecture of the system is provided in Fig. 4. In addition, the high-level synthesis has generated 10 coverage assertions controlling that each power mode, each power state and each transition between power states

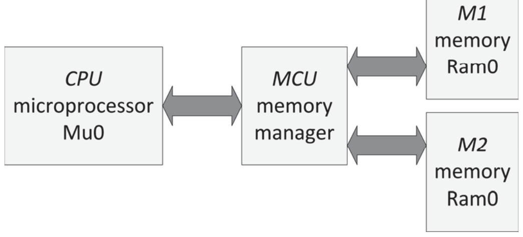

Figure 3. The case-study system architecture

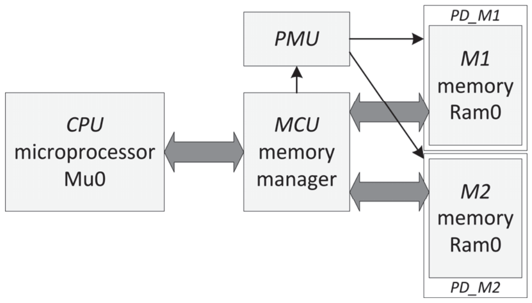

Figure 4. The case-study system power-management architecture

is exercised during RTL simulation. These assertions have taken 1172 characters of source code.

This case study has shown that even for such a simple power management (2 power domains, each with 2 power states and 1 component instance, 2 power modes), the proposed methodology simplifies the specification 17 times (237 vs. 4077 characters). However, the designer has to implement power-management elements, such as isolation logic, to proceed with RTL functional verification. Thus, the process is not yet fully automated.

For the area and power estimation, the tools integrated in the Xilinx ISE Design Suite 14.7 environment [19] have been used with default synthesis and implementation settings. As a target device, Spartan 3 xc3s1500l (target package fg320) has been selected, since we have this device available and the synthesized designs could be tested on the real device.

The estimation results are provided in Table 1. Interesting thing is that the area overhead is not high – utilization of some FPGA resources is even lower. Overall utilization seems to be more efficient in the power-managed design. Only the number of slice registers and the number of BUFGMUXs is higher. The power consumption is slightly reduced in the power managed design. Higher reduction is probable in more complex designs and in FPGAs in which power gating can be used.

B. *PMU Self-Management Area Overhead*

The utilized power-management high-level synthesis uses self-management of the PMU, which we have proposed in [20]. In this experiment, we have used FPGA synthesis to estimate area overhead of the integrated self-management of the synthesized PMU. Specifically, we target determination of the area overhead introduced by the modification of the power-state machine inside the PMU (introduction of additional elements to manage its own power consumption). It takes into account the area (utilization of FPGA resources), introduced by the comparison unit and the clock-gating logic.

We have pseudorandomly generated 15 samples of abstract power-management specification in SystemC, which have been then synthesized into the RTL form in order to get PMU designs. The parameters of these designs are provided in Table 2. The first column represents the identification number of the sample. The *APM* column represents the number of power modes in the abstract power-management specification. The *Power Domains* column represents the number of explicit power domains that are controlled by the PMU. The next two columns (*Power States* and *Instances*) contain average numbers of power states and instances in power domains. *PMU* represents the complexity of the generated PMU (in terms of the number of characters that are required for its description). The *GPM* column represents the number of power modes in the PMU (including the generated internal power modes). The number of control signals, driven by the PMU, is given in the last column (*Control Signals*). The samples are ordered according to the *PMU* column – it should correspond to the ascending order of PMU complexity, which scales from three thousand to half million characters; thus, these PMUs can be considered a representative set.

For easy comparison of the designs, we have used the equivalent-gates-count parameter offered by the Xilinx ISE Webpack 9.2i tool. In Table 3, the estimated area requirements for the PMU design without (Basic PMU design) and with (Self-Managed PMU design) the internal power management are provided. The results show that the area overhead can be as high as 50 % for simple PMU designs. However, for more complex PMU designs, the overhead is very small, even under 1 %. The area requirement of the self-management depends on the number of control signals, since they represent inputs of the comparison unit. The overhead is then indirectly proportional to the number of power modes, because it represents the complexity of the state and transition logic in the PMU design.

This experiment has shown that the proposed self-management inside very-complex power-management units has negligible area overhead. In small PMUs, a designer should carefully consider whether the gained power savings are worthy of the additional area requirements. However, as we have shown in the previous

TABLE II.
THE ESTIMATED POWER AND AREA DATA

Parameter	**Original Design**	**Power-Managed Design**	**Difference**
Number of Slice Flip Flops	50	53	6 %
Number of Slice Latches	0	2	200 %
Number of 4 input LUTs	260	230	-11.5 %
Number of occpied Slices	148	137	-7.5 %
Number of bonded IOBs	36	36	0 %
Number of BUFGMUXs	1	2	100 %
Average Fanout of Non-Clock Nets	4.22	4.22	0 %
Total power [mW]	150	148	-1.5 %

TABLE I.
THE PARAMETERS FOR AREA-OVERHEAD EVALUATION

#	APM	Power Domains	Power States	Instances	PMU	GPM	Control Signals
1	2	1	2	3	3300	4	3
2	3	2	2	2.5	4452	5	4
3	3	3	1.67	3	4619	5	3
4	3	4	2.25	3	7017	8	10
5	3	4	1.75	3	8658	12	7
6	3	4	2.25	3	10094	14	10
7	5	3	4	2	35205	46	13
8	10	3	3	2	63304	49	10
9	3	10	2.2	2	67691	65	21
10	5	5	3	5	107068	117	17
11	7	5	3	2	131478	101	17
12	7	4	3.5	2	142992	114	16
13	10	5	2.4	2	199866	106	13
14	5	10	2.1	3	214122	132	19
15	10	4	4	2	586303	346	18

TABLE III.
SELF-MANAGEMENT AREA-OVERHEAD ESTIMATION

#	Basic PMU Design	Self-Managed PMU Design	Area Overhead
1	57	76	33.33 %
2	116	153	31.90 %
3	57	85	49.12 %
4	284	342	20.42 %
5	631	702	11.25 %
6	944	1002	6.14 %
7	4167	4277	2.64 %
8	4693	4784	1.94 %
9	6346	6461	1.81 %
10	9896	9993	0.98 %
11	9866	9981	1.17 %
12	7550	7656	1.40 %
13	4740	4878	2.91 %
14	7271	7407	1.87 %
15	37501	37646	0.39 %

experiment, additional design elements could even result in more efficient utilization of FPGA resources. Thus, the designer should evaluate the power-area tradeoff on a case-by-case basis.

V. CONCLUSIONS AND FURTHER WORK

This paper is devoted to the adoption of the existing low-power systems design methodology to the FPGA platforms. It is oriented towards simplified adoption of power management in the system implemented in FPGA. The resulted methodology is useful for rapid design prototyping, utilizing a high degree of automation through so-called high-level synthesis. The adopted aspects of the methodology are usable for exploration of various power-management strategy algorithms, various power architectures, and corresponding power-management units. The experiments have shown the usefulness of the proposed methodology.

The further work can be oriented towards automated implementation of power-management logic, such as isolation elements and clock-gating logic. Then, the developed power-intent equivalence checking can be modified for the needs of the adopted design flow to verify synthesized RTL power management against ESL specification. Also, the adopted high abstraction of power-management specification enables further enhancement in the area of automated selection of abstract power states and modes of the system, as well as the automated segmentation of the system into power domains.

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 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).

REFERENCES

[1] W. Wong, "Low Power FPGA Targets Wearable Applications," *Electronic Design*, 2015, [online]. Available: http:// electronicdesign.com/fpgas/low-power-fpga-targets-wearable-applications. [Accessed 24 December 2015].

[2] J. Lamoureux and W. Luk, "An overview of low-power techniques for field-programmable gate arrays," in *NASA/ESA Conference on Adaptive Hardware and Systems*, IEEE, 2008, pp. 338-345.

[3] P. Singh and S.K. Vishvakarma, "Device/circuit/architectural techniques for ultra-low power FPGA design," *Microelectronics and Solid State Electronics*, vol. 2, no. A, pp. 1-15, 2013.

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

[5] G. Sutter and E. Boemo, "Experiments in low power FPGA design," *Latin American applied research*, vol. 37, no. 1, pp. 99-104, 2007.

[6] H. Belhadj, V. Aggrawal, A. Pradhan, and A. Zerrouki, *Power-aware FPGA design*. Actel, 2009. White paper.

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

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

[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] 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.

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

[12] 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.

[13] 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.

[14] 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.

[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] Xilinx Inc., Vivado Design Suite, [online]. Available: http://www. xilinx.com/products/design-tools/vivado.html. [Accessed 21 January 2016].

[18] A. Rogers, "Designing a Simple System-on-a-Chip in Under 60 Minutes with the mu0 Microprocessor and Xilinx Tools," 2003, [online]. Tutorial. Available: http://www.ece.uah.edu/~lacasa/ tutorials/mu0/mu0tutorial.html. [Accessed 7 May 2015].

[19] Xilinx Inc, ISE WebPACK Design Software, [online]. Available: http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html. [Accessed 27 March 2015].

[20] D. Macko and K. Jelemenská, "Self-managing power management unit," in *Proceedings of the 2014 IEEE 17th International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2014, pp. 159-162.