Automated Integration of Dynamic Power Management into FPGA-Based Design

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

***Abstract*—A low power or energy efficient hardware operation is nowadays gaining attention. It is especially true for battery-operated or energy-harvesting devices, such as most of the Internet of Things end nodes. For specific applications with rather limited market, the FPGAs are very good alternative. However, evolution of these devices is focused on high-level programming, giving application designers space to focus on application function rather than to be concerned about its low-level implementation on FPGA device – it is handled by automation tools. Thus, new FPGA-application designers are nowadays not very familiar with hardware aspects and it is difficult for them to apply power-reduction techniques in order to create an energy-efficient system. This paper is focused on automation of power-management integration into the FPGA-application design based on abstract specification, which is easy-to-use even for unfamiliar designers. It simplifies and speeds-up the low-power and energy-efficient FPGA-application design process. Moreover, the automation prevents many human-errors and thus it also alleviates the verification process. Experimental results indicate that the proposed power-management scheme is working correctly and it can be automatically generated.**

***Index Terms*—CAD, design automation, FPGA design, low power, power management**

I. Introduction

The popularity of the FPGA (Field-Programmable Gate Array) technology is mainly increasing due to market pressure on the manufacturers to reduce cost and time-to-market of their products. FPGAs enable to program the pre-manufactured device to perform some application-specific function. To support various applications, modern FPGAs are very complex circuits, often integrating several specialized blocks, such as multipliers, memories, or DSPs. The higher complexity however comes at a cost of increased power consumption. There are many applications (e.g. Internet of Things applications [1]) that require low-power operation, since the devices are powered by batteries or harvesting the energy from the environment. Therefore, the power in FPGA designs must be reduced in such cases. There exist many power-reduction techniques for hardware designs, such as those summarized in Table I. However usage of individual techniques is limited by their support in a specific FPGA device. For example, the voltage islands power-reduction techniques can be used only if the FPGA device supports multiple voltages to be used in the application design – i.e. it is given by the hardware architecture of the device itself. Most of these techniques can be activated and deactivated by the application according to current needs, enabling to temporarily disable unneeded parts of the system and thus save the energy. FPGA-based design nowadays focus on system-level abstraction programming, which enables a designer to focus on system function rather than the technological aspects. However, application of power-reduction techniques is quite complicated in this context. Therefore, this work targets abstract system-level power-management specification that is automatically incorporated into FPGA application design during the synthesis process. This simplifies the low-power FPGA-applications design process and enables designers unfamiliar with power-reduction techniques to create energy-efficient applications.

TABLE I
Popular Power-Reduction Techniques Overview

Technique	Description
Clock gating	Temporarily disables the synchronization signal of a synchronous module and thus prevents its operation.
Operand isolation	Reduces switching activity of a currently unneeded combinational part of the module.
Voltage scaling	Enables to switch the supply voltage of the module to a higher or lower level.
Frequency scaling	Enables to operate the module at multiple operation-speed levels.
Power gating	Temporarily disables the supply voltage of the currently unneeded module.
Clock islands	Splits the system into multiple domains, operating at different fixed frequencies.
Voltage islands	Splits the system into multiple domains, supplied by different fixed voltages.

Rest of the paper is organized as follows. The next section includes an analysis of the previous related works in this research area. Section III describes the proposed method for automated insertion of dynamic power management into FPGA-based design according to the abstract specification in SystemC/PMS (introduced in [2]). In Section IV, the experimental results are presented, which support usefulness of the proposed method. And the last section contains conclusions of this work.

II. Related Works

The method proposed in [3] targets power reduction of the FPGA interconnect by intervening the place-and-route pro-

This is an accepted version of the paper:
M. Škuta and D. Macko, "Automated integration of dynamic power management into FPGA-based design," in 2019 IEEE 22th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2019, pp. 151-154.
doi: 10.1109/DDECS.2019.8724635
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8724635

cess. The FPGA-based application cannot optimize the power consumption during runtime. In [4], the authors have used the autonomous fine-grained power gating for look-up-tables in an asynchronous FPGA, which offers relatively small overhead. The authors of [5] have proposed "mega cells" to reduce FPGA power consumption, which suffer by a rather higher overhead and limited scalability. These methods are focused on the FPGA-chip architecture and they cannot be used to manage power dynamically by the application itself. This is targeted in [6]; however, the proposed method still represents a modification of the FPGA architecture and cannot be used on commercially available FPGAs. The method used in [7] is focused on the Xilinx ZYNQ device, in which the software-driven dynamic power gating was utilized. Such a method cannot be used on a device without an embedded processor. Another method, presented in [8], focuses on dynamic-power optimization of FPGA-based designs at the system abstraction level, using algorithm partitioning and clock domains. It however cannot be used for dynamic power management.

The approach proposed in [9] offers fast evaluation of low-power systems prototypes on an FPGA platform. It uses performance monitoring to report activity of the system, which is then used to estimate its energy consumption. In [10], the FPGA-based education system has been used to demonstrate the impact of various designs and technologies on static and dynamic power dissipation. These approaches can be used for power evaluation of a given application; however, they do not support dynamic power management.

We have already proposed an idea of automating power management for FPGA-based designs in our previous work [11]. This work was targeted mostly towards automated generation of the power-management unit based on an abstract specification. The work however lacks automated implementation of power-management support logic (e.g. clock-gating and isolation cells). Moreover, it is focused on frequency scaling, and thus power-reduction techniques adjusting voltages are omitted.

III. A New Dynamic Power Management Automation Method

The proposed automation method is a continuation of the idea from our previous work [11]. Similarly, we are processing system-level specification of power management in SystemC/PMS [2]. Our method will firstly extract the power-management specification from the system-level model. This way can the system model be synthesizable by usual high-level synthesis tools. The difference of the newly proposed method and the previous one is that the new method also supports generation of power-management logic (clock-gating logic and isolation/synchronization elements) and automatically integrates the generated power-management unit and power-management elements into the system functional design.

The proposed method will be implemented into an automation tool, main components of which are shown in Fig. 1. To make the tool flexible and usable for others, it consists of separate components with their own functions (i.e. a modular

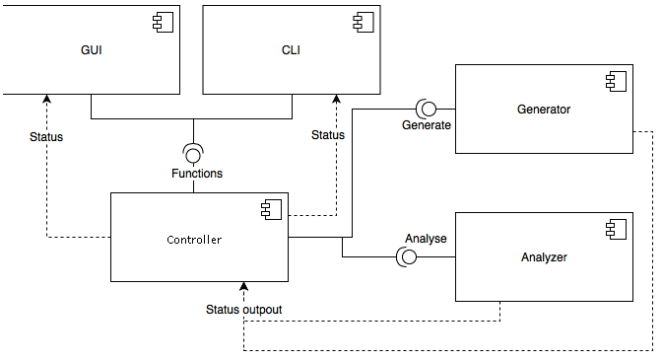

Fig. 1. Components of the automation tool.

design is followed). The main function of the controller component is to glue together functions from other components (mainly Generator and Analyzer) in a meaningful way.

The Analyzer's job is to read the specification written in SystemC/PMS and create an intermediate data structure holding all necessary information in a simple and easy-to-use format. The proposed format of this data structure is provided in Fig. 3. We have chosen JSON as our main data format because working with it is simple (from human and from a machine perspective). The first step of Analyzer is to find all performance levels specified in the code. In SystemC/PMS, the performance levels are specified by the used frequency and voltage. Some levels (HOLD_OFF, OFF_RET) are automatically created for a designer by PMS, so we need to count with them. The next step is to find all power domains and components belonging to them. Power domains are used to split the system into multiple domains operating at different levels. The designer can choose which levels will be used by the specific power domain. The subsequent step is to index all the signals in the specification. We also need to mark the component from which the signal is created and also mark all components to which the signal is connected. This information will be very helpful for creating isolation elements. The last step is to find all power modes, which specify the selected levels in all power domains.

The most important component is Generator, which will generate both, the power-management unit and the support logic. The input for the generator will be the previously mentioned JSON data structure and a generator configuration file (see Fig. 2). In the configuration file, the designer chooses

```
{
      "name":"iCE40Template",
      "pmu_generation":"levels/levels+power_modes/power_modes",
      "clock_buffers":"8",
      "reconfiguration":"4",
      "main_freq":"12",
      "max_freq":"48",
      "use_pll": true,
      "divide_pll": false
}
```

Fig. 2. Illustration of configuration file contents.

```
{
      "name":"FromSC-Module",
      "levels": [
            {"name": "normal",
            "freq": "12",
            "voltage": "1",
            "used": true},
            {"name": "hold",
            "freq": "0",
            "voltage": "1",
            "used": true},
            {"name": "off",
            "freq": "0",
            "voltage": "0",
            "used": true},
            {"name": "diff_level_1",
            "freq": "33",
            "voltage": "1",
            "used": false}
      ],
      "power_domains": [
            {"name": "PD1",
            "components": ["counter","filter", "filter1"],
            "levels": ["normal,hold,off"]},
            {"name": "PD2",
            "components": ["divider","filter2"],
            "levels": ["normal,off"]}
      ],
      "signals": [
            {"name" : "cpu_dout",
            "used_in" : [
                  {"name": "PD1",
                  "components": ["filter", "filter1"]},
                  {"name": "PD2",
                  "components": ["filter2"]}
            ]}
      ],
      "components": [
            {"name" : "counter",
            "assigned_to" : "PD1"},
            {"name" : "divider",
            "assigned_to" : "PD2"}
      ],
      "power_modes": [
                  {"name" : "on_mode",
                  "PD1" : "normal",
                  "PD2" : "normal"},
                  {"name" : "off_mode",
                  "PD1" : "normal",
                  "PD2" : "off"}
            ]
}
```


Fig. 3. An internal structure in the JSON format.

how the RTL (Register-Transfer Level) model will be generated. In the current proposal, the designer can specify the maximum number of clock domains, a maximum or minimum frequency, or the use of specific modules in FPGA (like a PLL). We expect that in the future the designer will have more options to choose in order to generate more specific power management that fulfils particular goals. We are also evaluating the possibility to directly change the RTL code of some generated RTL components (e.g. isolation elements). The RTL code of power-management components is more closely described in the next section. Remaining components are used just for controlling the main Controller component via the command line or via the graphical user interface.

IV. Experimental Results

To test the proposed method, we needed to select a suitable FPGA device and create an FPGA application (i.e. a system design to be running on the FPGA device), where it will be possible to apply power-reduction techniques.

As a target FPGA device, we have selected the Lattice iCEstick Evaluation Kit [12], because of its simple design, cheap price, and a great support even by open-source tools. The disadvantage of this simple FPGA is that it supports just one voltage level for the application logic. Therefore, we have focused on applying the clock gating and frequency scaling power-reduction techniques.

We designed a simple case-study system, consisting of an open-source 8-bit microprocessor (CPU) with the UART (Universal Asynchronous Receiver-Transmitter) interface [13]. Then, we split this design into two clock domains. The microprocessor with a memory in one clock domain and the UART interface in the second. We also manually created a power-management unit (PMU) and all support logic like Synchronizers, which can cross signals between different clock domains. A simple scheme (showing just the key information) of this system is illustrated in Fig 4. The synchronizers are applied on signals and buses that are interconnecting the two clock domains. A one extra synchronizer is used between the CPU and the PMU to control the PMU by the CPU. In this case study, we designed the PMU capable of handling three clock domains. The PMU was able to change the frequency of each clock domain to the three predefined frequencies (specified before the synthesis of the design to the FPGA device). To change the frequency, the PMU uses an 8-bit bus and a ”set” signal controlled by the CPU. The four upper bits select the power domain and the four lower bits select one of the predefined frequencies. This simple design is just for

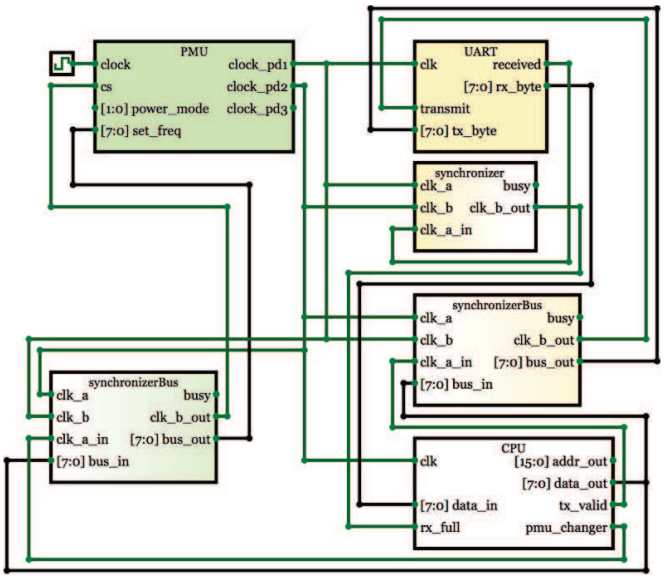

Fig. 4. The implemented experimental system.

TABLE II
FREQUENCY-SCALING MEASUREMENTS DATA

Frequency [kHz]	Current [mA]
0	122
1.2	124
12000	127
48000	130

an illustration purpose, the final automatically-generated PMU will handle more than three frequencies.

The results of the case-study are looking promising. We created a simple assembly program for the used microprocessor that is writing specific text strings over the UART. By pressing a special key, the user can change the frequency of the *clock_pd2* domain, to which the CPU belongs. A speed of the program execution has visibly changed after changing the frequency, and therefore, we can conclude that the CPU is working at different frequencies. We have also measured the current consumption (directly related to the power consumption) of the FPGA board, and there were small but clearly some variations. The results from this measurement are provided in Table II. The first column represents the selected clock frequency and the second column represents the current consumption under the selected clock frequency. Even when the clock signal was stopped, the current of 122mA was measured. Thus, this value relates to the static power consumption, which is dominant in the used FPGA device. The small increase in the measured current while scaling the frequency up shows that the frequency scaling influences the dynamic power consumption.

This experiment has proved that the designed power management is usable (i.e. it can scale the frequency of multiple clock domains), it is correctly working (the synchronizers avoid metastability issues), and thus it can be used in the proposed low-power FPGA-applications design-automation method.

V. CONCLUSIONS

The work presented in this paper was targeted towards low-power applications design based on programmable industrial FPGA platforms. Specifically, the aim was to automate (and thus simplify) usage of some power-reduction techniques that are usable in FPGA designs, but are rather difficult to apply for inexperienced designers. The result of our work is a tool that is able to analyze abstract model described in SystemC/PMS (with simple specification of power intent) and automatically generates the required components (e.g. power-management unit or synchronizers) into the more-detailed Verilog model. Since the complicated lower-level power intent is generated automatically, the proposed method speeds-up the design process and minimizes the possibilities of introduction of human errors. The method has been verified and validated in the experimental system, which has shown that it works correctly and that it helps to improve the energy efficiency of the designed FPGA application.

The further work will be targeted towards integration of other power-reduction techniques, such as power gating or dynamic reconfiguration. We will also test the method on another industrial FPGA device, which will support even the voltage-scaling technique.

ACKNOWLEDGMENT

This work was supported by 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. The work was also partially supported by the Slovak Research and Development Agency (APVV-15-0789) and the Slovak Cultural and Educational Grant Agency (KEGA 011STU-4/2017).

REFERENCES

[1] T. Gomes, S. Pinto, T. Gomes, A. Tavares, and J. Cabral, "Towards an FPGA-based edge device for the Internet of Things," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sep. 2015, pp. 1–4.

[2] D. Macko, K. Jelemenská, and P. Čičák, "Simplifying low-power SoC top-down design using the system-level abstraction and the increased automation," *Integration*, vol. 63, pp. 101–114, 2018.

[3] S. Huda and J. H. Anderson, "Power optimization of FPGA interconnect via circuit and CAD techniques," in *Proceedings of the 2016 on International Symposium on Physical Design*. ACM, 2016, pp. 123–130.

[4] S. Ishihara, M. Hariyama, and M. Kameyama, "A low-power FPGA based on autonomous fine-grain power gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 8, pp. 1394–1406, 2011.

[5] A. Ahari, B. Khaleghi, Z. Ebrahimi, H. Asadi, and M. B. Tahoori, "Towards dark silicon era in FPGAs using complementary hard logic design," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2014, pp. 518–523.

[6] A. A. M. Bsoul, S. J. E. Wilton, K. H. Tsoi, and W. Luk, "An FPGA architecture and CAD flow supporting dynamically controlled power gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 178–191, Jan 2016.

[7] M. Hosseinabady and J. L. Nunez-Yanez, "Run-time power gating in hybrid ARM-FPGA devices," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2014, pp. 512–517.

[8] P. P. Czapski and A. Śluzek, "System-level approaches to power efficiency in FPGA-based designs (data reduction algorithms case study)," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 5, pp. 49–59, 2011.

[9] G. Patrigeon, P. Benoit, and L. Torres, "FPGA-based platform for fast accurate evaluation of ultra low power SoC," in *2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 2018, pp. 123–128.

[10] A. Schwandt and M. Winzker, "Modular evaluation system for low-power applications: Educating undergraduate students in advanced digital design," in *24th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2018, pp. 364–367.

[11] D. Macko, "Adoption of abstract power-management specification to FPGA-based design," in *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE, 2016, pp. 199–204.

[12] Lattice Semiconductor. (2019) iCEstick evaluation kit: Rapid development for affordable innovation. [Online]. Available: https://www.latticesemi.com/icestick

[13] M. Škuta. (2019) iCE40HX1K demos. [Online]. Available: https://github.com/mintos5/iCE40HX1K-demos