Power-Efficient Power-Management Logic

Dominik Macko, Katarína Jelemenská, Pavel Čičák
Faculty of Informatics and Information Technologies
Slovak University of Technology
Bratislava, Slovakia
dominik.macko@stuba.sk, katarina.jelemenska@stuba.sk, pavel.cicak@stuba.sk

***Abstract*—Since the power consumption has become the key aspect in almost every digital-system design, many advanced power-reduction techniques have been developed to minimize power. The most popular strategy to apply these techniques is adoption of so-called power management. The control signals for the power-management elements are generated by power-management logic. When centralized in one system block, it is referred to as power-management unit (PMU). PMU should also be targeted by power-efficient design techniques. This paper shows the value of power-management inside the PMU, reducing the overall system power consumption. We show that the power-state machines in the PMU can be designed in such a way that only necessary components stay active in the sleep mode. Experimental results illustrate that approximately 30% of power-state machines power can be saved.**

Keywords—low power; power control; power management; power reduction; power-state machine

I. INTRODUCTION

Generally, the power in CMOS (Complementary Metal-Oxide Semiconductor) technology consists of two components, a leakage and dynamic power. The first one depends on the supply voltage, the switching threshold voltage, and the transistor size. The dynamic power depends on the switching activity, clock frequency, capacitance, supply voltage, and short-circuit current [1].

The power consumption is one of the key concerns in modern digital system designs. Whether the hardware designers want to maximize the battery life of the mobile devices or to minimize the system operating cost, they need to make power-efficient products. Therefore, the use of power management in modern hardware designs is a common practice.

With emerging market of so-called Internet of Things (IoT) [2], battery-operated ultra-low-power electronic devices surround our everyday life. IoT connects devices in our intelligent households (e.g. security wireless sensors), our wearable electronics (e.g. smartwatches or smartglasses), or even human-body-implanted healthcare devices. It is not rare in such devices that they spend most of their lifetime in a sleep mode, just waiting for some event to occur. Thus, the power consumption is converging towards the sleep-mode power. Important aspect concerning the power-management unit (PMU) is that it stays active even in the sleep mode, thus consuming significant power.

This paper is focused towards the evaluation of novel power-management design strategy, proposed in [3]. The paper is organized as follows. In Section II, the background information regarding the power management is given. Section III describes the related work done in this area. Next section introduces novel power-management logic architecture. And before conclusion, the experimental results are shown.

II. POWER MANAGEMENT

Power consumption has been an issue in hardware design for a long time. Therefore, many power-management techniques have already been developed, such as clock-gating for a dynamic power reduction, or power-gating for a leakage power reduction (see Table I).

TABLE I. POWER-MANAGEMENT TECHNIQUES

Technique	Description
Clock gating	Disables clock tree part not in use. Synchronous block stops its operation.
Operand isolation	Prevents switching of inactive datapath element.
Substrate biasing	Dynamically bias the substrate or the appropriate well in order to raise transistor voltage threshold in inactive mode, thereby reduce leakage.
Dynamic voltage scaling	Different blocks are operated at variable supply voltages. Uses look-up tables to adjust voltage on-the-fly to satisfy varying performance requirements.
Adaptive voltage scaling	Different blocks are operated at variable supply voltages. The block voltage is automatically adjusted on-the-fly based on performance requirements.
Frequency scaling	Frequency of the block is dynamically adjusted. Works alongside with voltage scaling.
Power gating	Turns off supply voltage to blocks not in use. Significantly reduces the leakage. Block outputs float and need to be isolated when connected to active block.
State retention power gating	Stores the system state prior to power-down. Avoids complete reset at power-up, which reduces delay and power consumption.

Some of these techniques are straightforward (e.g. clock gating or operand isolation), others are difficult to adopt. To alleviate this difficulty, the standard for design and verification of low-power integrated circuits was developed (commonly known as UPF – Unified Power Format) [4]. This standard simplifies the adoption of power-management strategy in the design process. It contains the constructs to specify power-management elements (e.g. power switches, isolation cells, level shifters) at the early stage of the design process, when mostly the HDL (Hardware Description Language) modeling is

This is an accepted version of the published paper:
D. Macko, K. Jelemenská and P. Cicák, "Power-efficient power-management logic," 2014 24th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Palma de Mallorca, 2014, pp. 105-111.
doi: 10.1109/PATMOS.2014.6951881
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6951881&isnumber=6951857

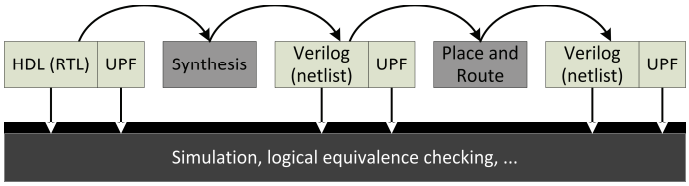

Fig. 1. UPF-based low-power design flow.

used. In such way, the functional HDL model along with the UPF power management can be verified.

Fig. 1 shows a typical flow of the low-power design when using UPF. In the figure, the HDL (RTL) item represents a HDL functional model at the register-transfer level. Verilog (netlist) represents a model after synthesis and after place-and-route process. As the figure shows, power-management in UPF is separated from the main functional model. UPF is preserved throughout the whole design flow and is participating in the verification process (simulation or equivalence checking) – EDA (Electronic Design Automation) tools load HDL files along with UPF files and verify them together.

The key UPF concept is to provide the means for dividing the system into so-called power domains. Power domain is a collection of blocks that always operate at the same supply voltage level. UPF allows the designer to specify which blocks are grouped into power domain, what voltage levels the power domain can operate at, what the power-down condition for each power domain is, where the isolation cells and level shifters should be used, and so on.

A. Power-management unit

UPF assumes the control signals for the power-management elements to be generated by the functional model. Usually, this power-management logic is grouped into separated system block, called power-management unit (PMU). It is responsible for determination of the suitable system power mode (combination of power states of the power domains) and handles the transition to that power mode.

In Fig. 2, a typical architecture with PMU is shown. PMU contains several modules, such as power-mode determination (PMD) and power-state machines (PSMs). Based on the power mode determined by PMD, the PSMs process the transition to the new power states for individual power domains (PDs). We may notice that the block 3 and block 4 are grouped together into the power domain PD3. Thus, there is the redundant PSM omitted.

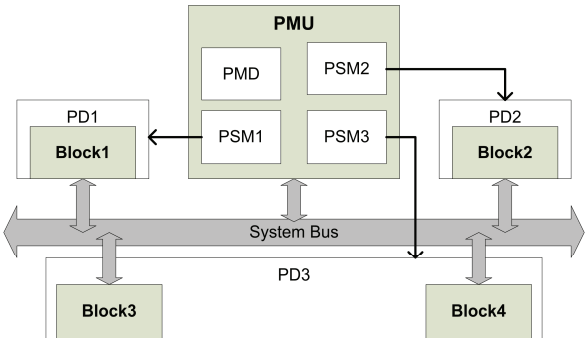

Fig. 2. Power management unit controlling the power domains.

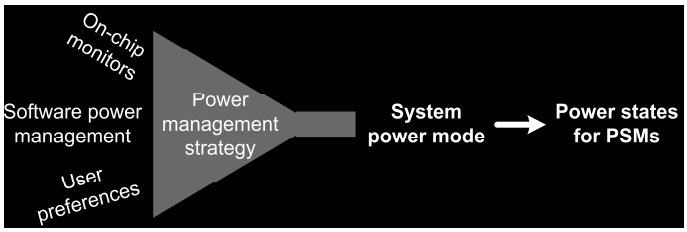

Fig. 3. System power mode determination.

Fig. 3 shows how the system power mode is determined. The power-management-strategy algorithm takes available information into account and determines the suitable power mode. This power mode represents the combination of the power states for individual power domains. Thus the new power states are passed to the PSMs that handle the transitions. Based on the current state of PSM, the control signal values for power-management elements are generated.

III. RELATED WORK

In [5], a method for power management inside a SoC to satisfy the corresponding application with low power consumption is described. The software algorithm determines the system operating mode needed for a given application. The switch of this mode is controlled by the operating mode switch finite state machine. The method uses the dynamic voltage and frequency scaling technique, thus each operating mode corresponds to a set of frequency and voltage values. The experimental results showed a significant reduction of power consumption. This approach does not use the power domain concept, as offered by UPF. Thus, even the system blocks that are always in the same power state have the dedicated power management elements and control signals. These are driven by additional control elements of the PMU, introducing unnecessary overhead.

When the power domains are used (as suggested by UPF standard), the PSMs are generated not for the system blocks, but for the power domains. It eliminates the unnecessary duplicates of the same state machine and sometimes reduces even the complexity of power-management elements. Such an approach is commonly used today, and we can find it for example in [6, 7]. Although the number of state machines is reduced, the unnecessary power dissipation inside the PMU still occurs.

Another approach reducing the PMU power dissipation is described in [8]. It is focused on increasing the power-efficiency of the active-inactive state transitions.

In [9], the clock-gating technique was introduced into finite-state machine design. The method uses self loop in Moore state machine to disable the clock signal, saving the dynamic power.

The method described in [10] combines the previous approach with the finite-state machine decomposition. Based on the state-transition graph, the method divides the machine into two parts. The clock-gating technique is then used to stop the clock in the currently-unused part of the machine.

These approaches were further extended by implementation of power-gating technique in [11, 12]. These extensions use locally-extracted clock-enable signals and use them as sleep signals for the power gating.

The method combining the machine decomposition and state-encoding alteration with power gating was reported in [13]. This method uses genetic algorithm for the state-transition graph decomposition into two parts. Similarly to [10], when one part of the transition logic is not needed, it is powered-down. Sometimes both parts have to be active for the next-state transition, thus the power is excessively consumed. Therefore, this method is suitable when the probability of the machine boundary states is low (boundary state in the state-transition graph point of view).

As stated in Section I, many IoT (Internet of Things) devices spend most of the operation time in the sleep mode. Thus, the power mode transition is very rare in general, and therefore the leakage power inside the PMU is dominant. To reduce power dissipation, we target the PSM design.

IV. Power-State Machine Design

The power-state machine (PSM) is an application-specific kind of finite-state machine (FSM). In PSM, the inputs are the subset of the outputs. It means that the change of the inputs values triggers transitions through several states, each producing different outputs values.

A finite-state machine usually consists of the transition logic (combinational circuit), the state logic (sequential circuit), and the output logic (combinational circuit). In general, there are two types of FSM, the Moore and the Mealy type. Mealy-machine output values depend on the current state of the machine and on the input values (Fig. 4a). Since, the PSM needs to generate a synchronous sequence of state transitions (along with output values changes) based on one input change, the Mealy type FSM is not usable for PSM design. The output values of the Moore machine depend on the current state only. This type of FSM is shown in Fig. 4b. Regarding the PSM, the states of FSM should directly correspond to the control signals for the power management elements, i.e. the output logic is not present (Fig. 4c). Such a special kind of Moore FSM is called Medvedev machine [14]. It is beneficial to have no delays among the power-management control signals, because it prevents the hazards. Also considering the design for testability, direct observability and controllability of outputs via scan-chains is welcome for PSM design.

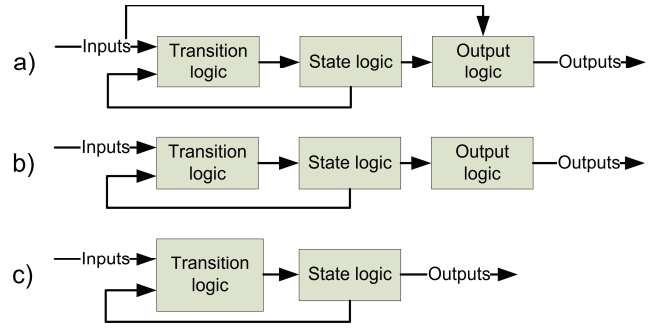

Fig. 4. FSM components – a) Mealy machine; b) traditional Moore machine; c) Medvedev-type Moore machine.

The realization of PSM as a Medvedev machine has more power-saving benefits. In such a manner, the flip-flops saving the machine state can be continuously powered and thus be able to retain the control signals. The transition logic can be powered-down when the power state does not need to be changed (idle time – from PSM point of view, not the system as such). Although the transition logic is powered-down, the isolation between transition logic and state logic is not needed because of the integrated controlling mechanism of the flip-flops. This mechanism states that the data input has to be active only at the active clock edge. Moreover, the clock is stopped during the idle time using clock-gating technique.

Even though the number of flip-flops representing the state is sometimes not optimal using Medvedev machine, this PSM design strategy showed itself beneficial regarding the power consumption.

A. Idle-time determination control logic

There are several options to power some circuit part down. When we do not want the transition logic power-up/down cycle to be controlled by yet another state machine, we can locally generate the control signals. The power switch sleep signal is generated by comparing the current power state at the PSM outputs with the target power state at the PSM inputs. When the power states are the same, the sleep signal is activated, powering the transition logic down. When the power state needs to be changed (i.e. the target power state differs from the current power state), the sleep signal is deactivated, the transition logic powers-up and the sequence of power-state transitions is handled in order to achieve the target state. The comparison logic simultaneously generates the clock-gating enable signal for the state logic – it guarantees that the state is not changing when the transition logic is powered-down.

The proposed PSM modified architecture is shown in Fig. 5. The power-management elements of transition logic consist of the power switches. They allow the logic gates to be powered-down. The power-management elements of the state logic contain only the clock-gating logic. It allows the clock signal to be stopped and thus the state value is not updated during the idle time.

Depending on the realization there can be an overhead of a few clock cycles to wake up the transition logic. It means that the control signals for the power-management elements are latish. When considering the long period of time between power state changes, such an overhead is negligible.

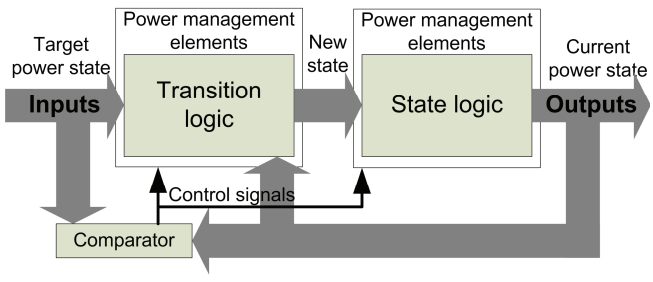

Fig. 5. The proposed PSM architecture.

V. EXPERIMENTAL RESULTS

The primary goal of the experiments was to estimate the power-efficiency of the proposed self-management inside the power-state machines.

A. Experimental design

We have processed the experiments on the single PSM design controlling one power domain. We assume the power domain needs to be managed for changing between the four power states (i.e. *off*, *low-voltage*, *normal*, and *high-voltage*). From each power state it is possible to transit to any of the other states. In order to prevent floating signal values, the inputs and outputs of this power domain have to be isolated before switching-off the power. Therefore, a new state controlling the isolation is added. However, this power state is transparent to the power-mode determination module – i.e. it cannot be determined as a target power state. It is just an intermediate state in order to correctly reach some other state. These five power states are represented by three control signals – two for controlling the power switch (switching four supply-voltage levels) and one for controlling the isolation.

B. Experimental setting

Since the power has also the dynamic contributing element, the experiments consist of six test-cases for simulation, each with different set of parameters. These parameters are the clock frequency and the toggle-rate of the target state (PSM input). The simulation test-cases are summarized in the Table II. The first column refers to the test-case number. f(CLK) stands for the frequency of clock signal. Simulation time reports the actual runtime of the simulation test-case. TR_{TS} represents the toggle-rate of the target state signal and is expressed by the number of target-state changes (toggles) per clock period (clock cycle). Pseudorandom constraint reflects the intended value set in the test-bench. The column Generated represents the average toggle-rate of the target state for particular test-case as was randomly generated. Since the generator has randomly selected the target state from the set of four, there was one quarter chance to select the current power state. Therefore actual target-state changes are slightly different, shown in the last column.

TABLE II. SIMULATION TEST-CASES DESCRIPTION

Test-case #	f(CLK) [MHz]	Simulation time	TR_{TS} [toggles/T_{CLK}]		
			Pseudorandom constraint	*Generated*	*Actual*
1	50	5 µs	0.4	0.408	0.288
2	50	5 µs	0.133	0.148	0.116
3	50	5 µs	0.04	0.048	0.04
4	0.05	10 s	0.000004	0.000012	0.00001
5	500	5 µs	0.04	0.0408	0.0296
6	5	5 µs	4	4.08	3.2

The first three test-cases have the same clock frequency, but the toggle rate is decreasing respectively. These test-cases are used for illustration of the dependency of PSM power consumption on the number of power-state changes. The fourth test-case reflect the situation, when the toggle-rate per clock-period is very low – there were 5 actual toggles of the target state input during 10 seconds simulation runtime. Moreover, the clock frequency was lowered to 50 kHz in order to simulate more realistic situation (the real PMUs operate commonly at the real-time clocks – usually 32.768 kHz). This test-case is used for determination of the self-management impact on the PSM power consumption in the systems with very rare power-state changes. The last two test-cases represent the simulations with the same number of target state changes as the first one, but with the different clock frequency. These three test-cases (1, 5, and 6) are used for illustration of the dependency of PSM power consumption on the PSM clock frequency.

The power-aware simulation along with the estimation of power consumption was realized in the professional commercial EDA tools. We have used free-available 45 nm technology library NangateOpenCellLibrary revision 1.0 [15].

C. Experiment 1: Self-management power impact

This section describes the experimental results for determination of the impact of modified PSM on the power consumption. In the experiment, we compare the estimated power consumption of the original PSM architecture (without the integrated power management; referred to as *basic PSM*) to the modified PSM architecture (with the integrated power management; referred to as *self-managed PSM*). The PSM in this experiment was realized as the Medvedev-type machine.

In Table III, the power estimations for basic and for self-managed Medvedev-type PSMs are compared. We determine the benefit of the self-management integration into the PSM by comparing the leakage, dynamic and total power for individual test-cases.

TABLE III. SELF-MANAGEMENT POWER-IMPACT OF MEDVEDEV PSM

Test-case #	Basic PSM			Self-managed PSM			Power saving
	Leakage [nW]	*Dynamic [nW]*	*Total [nW]*	*Leakage [nW]*	*Dynamic [nW]*	*Total [nW]*	
1	792.22	1884.32	2676.54	571.16	2030.11	2601.27	3%
2	789.05	1145.03	1934.09	354.78	1013.16	1367.94	29%
3	778.92	1008.08	1787	282.83	749.28	1032.11	42%
4	798.96	0.73	799.7	222.25	345.75	567.99	29%
5	793.06	8876.35	9669.41	259.88	3688.89	3948.77	59%
6	790.62	779.52	1570.13	1056.84	1664.94	2721.78	-73%

We may notice that the power saving of the modified PSM scales from 3 to 59 %, and in the sixth test-case the power consumption was actually significantly increased (by 73%). This is actually an expected fact, because the sixth test-case was unrealistic one, when there was 3.2 target state changes per clock cycle in average (see TR_{TS} in Table II). In this case the PSM was almost always active, thus it could not benefit from integrated power-gating to save the leakage power. The power saving increases with decreasing TR_{TS} parameter and starts to decrease with very rare target state changes (test-case 4). In the fourth test-case the added components start to consume significant portion of power compared to the rest of PSM (even then there is a power saving of 29 %).

More illustrating comparison is given in Fig. 6. The huge difference in power consumption between test-case 4 and 5 is mostly due to the clock frequency. In the fifth test-case there is

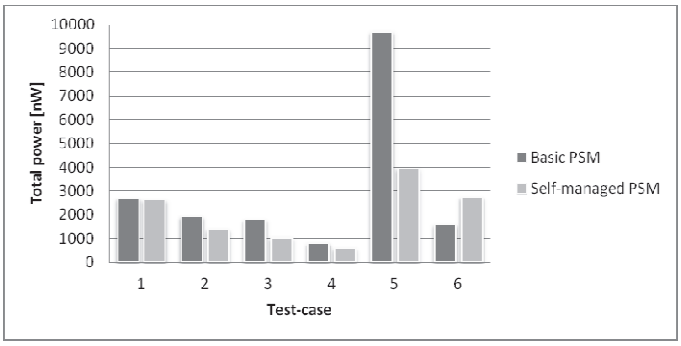

Fig. 6. Power consumption of basic and self-managed Medvedev-type PSM.

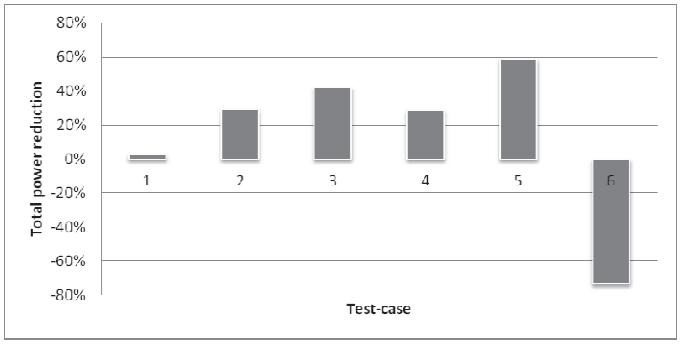

Fig. 7. Power-reduction of self-managed Medvedev-type PSM.

10000 times higher frequency then in the fourth, as is shown in Table II. The relative comparison between basic and self-managed PSM is given in Fig. 7. The values represent the power reduction of self-managed PSM compared to the basic PSM.

The most power reduction was determined for high-frequency test-cases. The dependency of the PSM power consumption on the clock frequency is shown in Fig. 8. It is based on the results obtained for the sixth, first, and fifth test-cases (the same number of toggles, only the frequency is scaling). As the figure shows, the PSM power consumption increases with the increasing clock frequency. The basic PSM is more dependent on clock-frequency than the self-managed PSM – the power consumption grows much faster. Therefore, the use of self-management has more power-saving benefit for high-frequency PSMs. The main reason is the use of clock-gating technique that stops the machine to update its state when it is not necessary (the same next state). The power increase introduced by the self-management in the sixth test-case (5 MHz in the figure) was clarified before.

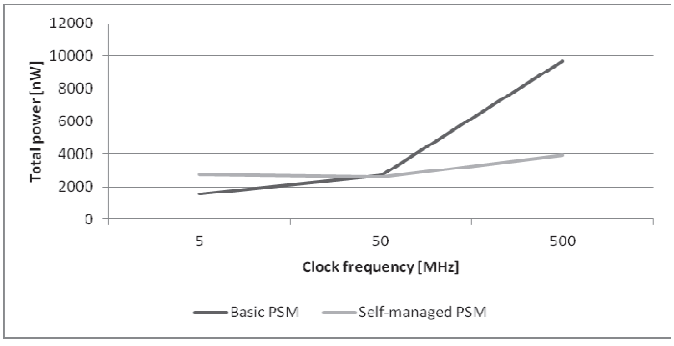

Fig. 8. PSM power consumption dependency on the clock frequency.

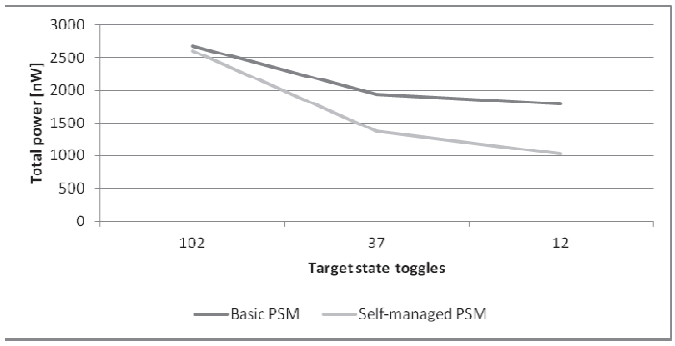

Fig. 9. PSM power consumption dependency on the number of target-state changes.

Fig. 9 shows the dependency of the PSM power consumption on the number of target-state toggles. The first three test-cases from Table II were used in the figure (the same clock frequency, the number of toggles decreases). The results show that with the decreasing number of toggles the power consumption decreases. Moreover, as the figure shows, the power consumption of the self-managed PSM is decreasing faster than the basic PSM power consumption, thus the power-saving increases.

The power-reduction in the first test-case (3 %) is very low and for such case it would not be worthy of additional area requirements. This test-case can be considered boundary case, for which the modified PSM architecture is beneficial regarding the power consumption. In Table II, the TR_{TS} for the first test-case is shown to be 0.288. It means that the target state changes approximately each third clock cycle. The transition logic is activated for one or more clock cycles, depending on the sequence needed to correctly reach the target state from the current power state. Therefore we may assume, that in this case the PSM was active (transition logic powered-up) more than 30 % of the simulation time.

1) Experiment summary and outcomes

The experiment was successful. It has proven the undeniable benefits of the modified PSM architecture that can manage its own power consumption. The power consumption of PSM was reduced for 5 test-cases (32 % power reduction in average); only the unrealistic test-case showed the power increase. We showed that for systems with very rare power state changes (many IoT devices) it can save approximately 30 % of power, and for high-frequency PSM even more. Therefore, the self-management integration into the PSM architecture significantly reduces the PSM power consumption.

The results showed that the self-management power-efficiency grows with decreasing number of target-state changes and with increasing clock frequency. As the boundary case for the self-managed PSM to reduce power we have determined the situation when the power state needs to be changed 30 % of the time. It means that the PSM has to be at least 70 % of the time inactive.

D. Experiment 2: Self-managed Moore-type PSM evaluation

This section describes the experimental results for determination of the power impact of self-management in the traditional Moore-type PSM. The goal is to show that the self-

management benefits (regarding the power consumption) are not limited to the Medvedev-type power-state machines. We also compare these two PSM types to see which one is more power-efficient for this experimental design (described in Subsection A).

In Table IV the power estimations for basic and for self-managed Moore PSMs are compared. The same comparison is shown as in the previous experiment.

TABLE IV. SELF-MANAGEMENT POWER-IMPACT OF TRADITIONAL MOORE-TYPE PSM

Test-case #	Basic PSM			Self-managed PSM			Power saving
	Leakage [nW]	*Dynamic [nW]*	*Total [nW]*	*Leakage [nW]*	*Dynamic [nW]*	*Total [nW]*	
1	1385.84	3448.23	4834.07	721.36	2608.49	3329.85	31%
2	1367.19	1887.56	3254.74	606.16	1575.49	2181.69	33%
3	1372.22	1248.85	2621.07	546.31	964.07	1510.38	42%
4	1287.37	1.03	1288.4	469.9	158.19	628.09	51%
5	1381.02	11148.74	12529.76	536.31	4718.18	5254.49	58%
6	1389.25	1458.71	2847.95	1313.34	1105.25	2418.6	15%

The reported power saving is scaling from 15 to 58 %. The lowest power saving is in the sixth test-case, when the PSM machine was almost always active. The transition logic is quite complex in this design using the Moore FSM type, and therefore even the short periods of idle time reduce the power consumption. The highest power saving is shown in the fifth test-case, with the highest clock frequency.

More illustrating comparison is given in Fig. 10 (absolute values) and Fig. 11 (relative comparison). This comparison clearly shows the benefits of using the self-management inside the designed PSM realized even as the Moore machine.

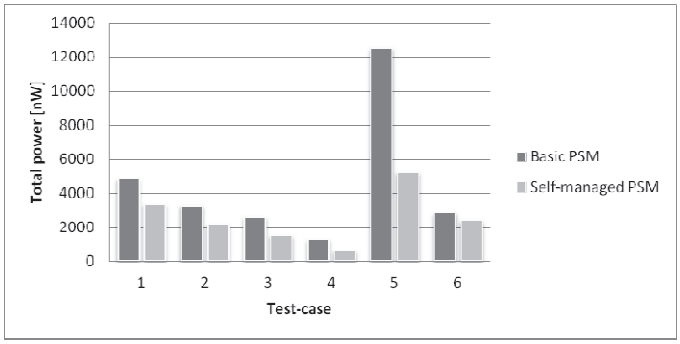

Fig. 10. Power consumption of basic and self-managed Moore-type PSM.

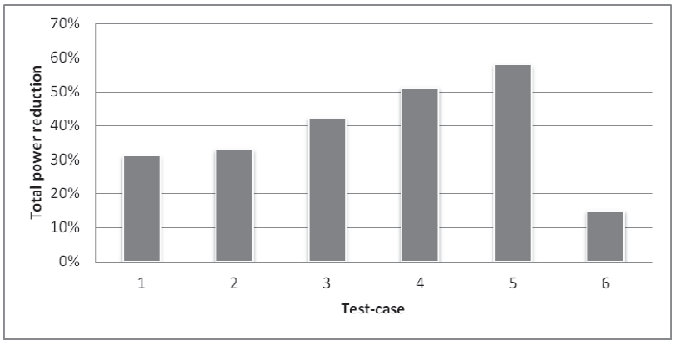

Fig. 11. Power-reduction of self-managed Moore-type PSM.

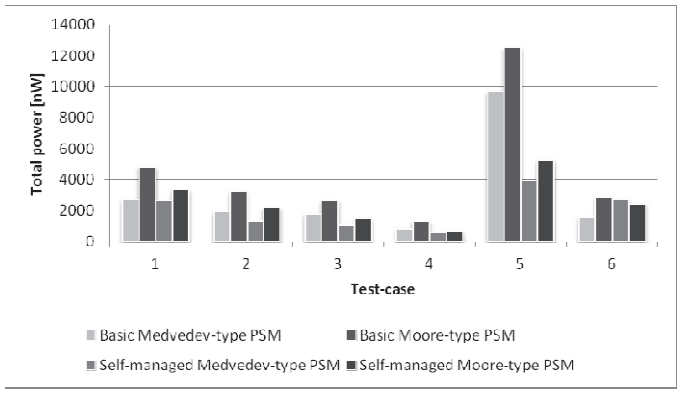

Fig. 12. Power consumption comparison of Medvedev-type and traditional Moore-type PSM.

The results of power-consumption comparison between Medvedev-type and traditional Moore-type PSM are shown in Fig. 12. We have compared the basic architectures and the self-managed PSM architectures. The comparison results show that Medvedev-type machine is more power-efficient for this experimental PSM design than the traditional Moore machine (with the output logic).

The relative power comparison between these two types of machines is clearly shown in Fig. 13. The figure shows the power reduction when using Medvedev-type PSM relatively to the traditional Moore-type PSM. When using Medvedev machine, the power consumption is reduced in average by 37 % for basic PSM architecture and by 19 % for self-managed PSM. Only in the sixth test-case for self-managed PSM, the traditional Moore-type machine is more power-efficient. As stated before, this is an unrealistic test-case in which the target state toggles more often than clock signal (see Table II).

In Fig. 14, the power reduction of self-managed Medvedev-type PSM is compared to the basic traditional Moore-type PSM. The results can show us the power-reduction benefit of changing the machine type from traditional Moore to Medvedev and followed by the integration of self-management. The results show that the power consumption was reduced by more than 40 % for all test-cases besides the last one. In the sixth test-case, the power was reduced only by 4 %. In Fig. 7 we have shown that for this test-case the self-managed Medvedev-type PSM increases the power consumption. For this test-case, the most power-efficient architecture is the basic Medvedev-type PSM, as shown in Fig. 12.

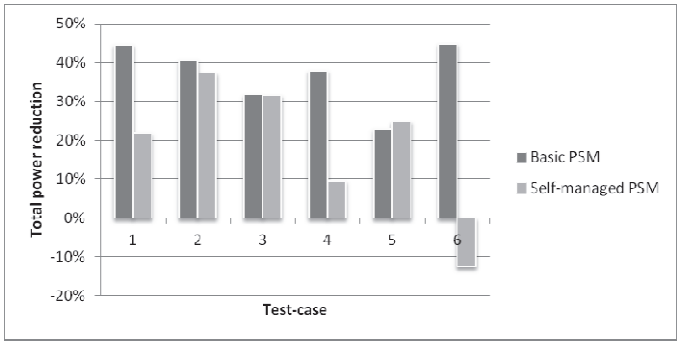

Fig. 13. Total power reduction of Medvedev-type PSM in comparison to traditional Moore-type PSM.

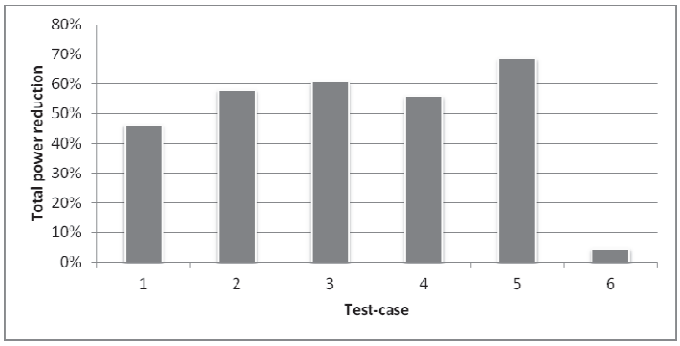

Fig. 14. Total power reduction of self-managed Medvedev-type PSM in comparison to basic traditional Moore-type PSM.

1) Experiment summary and outcomes

This experiment showed that self-management integration into the traditional Moore PSM reduces the power consumption even in the case, when the target state changes 3.2 times per clock cycle in average. It means that the power-reduction achieved during the short idle time periods compensates the additional control circuitry (comparator, clock and power gating logic) power consumption. The self-management inside the traditional Moore FSM for this experimental PSM design saves 38 % of the power in average (including the sixth test-case). It supports the conclusion from the previous experiment that the self-management integration into PSM architecture has a significant positive impact on the power reduction.

The comparison of the results obtained for Medvedev and traditional Moore PSMs showed that for the selected experimental PSM design the self-managed Medvedev-type PSM has the lowest power consumption. The usage of Medvedev-type PSM instead of Moore-type resulted in significant power reduction. However, the used PSM switches between 5 power states, meaning that there have to be at least 3 flip-flops to save the machine state. Therefore, the Moore-type machine could not benefit from the state-elements reduction. Thus, for other PSM designs, the Moore-type machine with simple output logic might be more power-efficient. Therefore, to obtain more illustrating results comparing these two FSM types, the more-complex PSM should be designed.

VI. Conclusions

Power-state machines (PSMs) of the power-management unit are handling the transitions to determined power states. These power states represent the values of control signals for the power-management elements (e.g. isolation cells, retention cells, or power switches) in the advanced power-reduction techniques (e.g. power gating or dynamic voltage scaling). We have proposed and evaluated novel self-managed PSM architecture allowing the management of its own power consumption. We have used the Medvedev machine to avoid timing hazards of the control signals and integrated both the clock and power gating power-management techniques for the power reduction. In the experiments we have shown that the self-management integration into the PSM can reduce up to 58 % of the PSM power consumption. In case the hazards would not be an issue, we compared this architecture to the self-managed traditional Moore machine (with the output logic). The results showed that for the selected PSM design the Medvedev machine was more power efficient (by 19 % in average). However, the Moore machine could not benefit from the state-elements reduction in the selected experimental design. Therefore, the further work will include the experiments for more complicated PSM design.

The self-management inside the PSM has undeniable benefits, but to be worthy the additional area requirements, the PSM has to be at least 70 % of the time inactive. It means that the power mode of the device cannot be changed very often. Moreover, considering the whole system power, the power of the PSM is negligible. Thus the proposed architecture is suitable for the systems in which the power consumption is converging towards the sleep-mode power (many Internet of Things devices). The experiments showed that for such systems, the novel PSM architecture can save approximately 30 % of the total power (leakage + dynamic power).

Acknowledgment

This work was partially supported by the Slovak Science Grant Agency (VEGA 1/1008/12 and VEGA 1/0616/14), Slovak University of Technology ("ANSNS) and COST Action IC 1103 MEDIAN.

References

[1] Cadence Design Systems, A practical guide to low power design: User experience with CPF, 2012. http://www.si2.org/?page=1061

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] D. Macko and K. Jelemenská, "Self-Managing Power Management Unit," in DDECS, IEEE, 2014, pp. 159-162.

[4] IEEE Standard for Design and Verification of Low Power Integrated Circuits, IEEE, 2013, (IEEE Std 1801-2013).

[5] D. Sun, S. Xu, W. Sun, S. Lu, and L. Shi, "Low power design for SoC with power management unit," in ASICON, IEEE, 2011, pp. 719-722.

[6] T. Coulot, T. Souvignet, S. Trochut et al., "Fully integrated power management unit (PMU) using NMOS Low Dropout regulators," in EUROCON, IEEE, 2013, pp. 1445-1452.

[7] H. Unterassinger, M. Dielacher, M. Flatscher et al., "A power management unit for ultra-low power wireless sensor networks," in AFRICON, IEEE, 2011.

[8] M. Alioto, E. Consoli, and J.M. Rabaey, "'EChO' Reconfigurable Power Management Unit for Energy Reduction in Sleep-Active Transitions," IEEE Journal of Solid-State Circuits, vol. 48, no. 8, pp. 1921-1932, 2013.

[9] L. Benini, P. Siegel, and G. De Micheli, "Automatic Synthesis of Low-Power Gated-Clock Finite-State Machines," IEEE Transactions on Computer-Aided Design, vol. 15, no. 6, 630–643, 1996.

[10] J.C. Monteiro and A.L. Oliveira, "Finite State Machine Decomposition For Low Power," in DAC, IEEE, 1998, pp. 758–763.

[11] K. Usami and H. Yoshioka, "A Scheme to Reduce Active Leakage Power by Detecting State Transitions," in MWSCAS, IEEE, 2004, pp. I-493-I-496.

[12] K. Usami and N. Ohkubo, "A Design Approach for Fine-grained Run-Time Power Gating using Locally Extracted Sleep Signals," in ICCD, IEEE, 2006, pp. 155-161.

[13] S.N. Pradhan, M.T. Kumar, and S. Chattopadhyay, "Low power finite state machine synthesis using power-gating," Integration, the VLSI Journal, vol. 44, no. 3, pp. 175-184, 2011.

[14] H. Kaeslin, Digital Integrated Circuit Design: From VLSI Architectures to CMOS fabrication. Cambridge University Press, 2008.

[15] Nangate, "Open Cell Library," Online, January 2014. http://www.nangate.com/?page_id=22