Run-Time Power Gating in Hybrid ARM-FPGA Devices

Mohammad Hosseinabady and Jose Luis Nunez-Yanez
Department of Electrical and Electronic Engineering University of Bristol, UK.
Email: {m.hosseinabady, j.l.nunez-yanez}@bristol.ac.uk

***Abstract*—Energy proportional computing (EPC) enables the allocation of energy to tasks depending on computational demands. Computing at full speed and then dynamically turning off modules when they are not required for a period of time can be used to obtain EPC and it is an alternative to voltage scaling techniques in which the computation is slowed down. This paper investigates the viability of physical power gating FPGA devices that incorporate a hardened processor in a different power domain. The run-time power gating approach is applied to Xilinx ZYNQ devices that incorporate a hardened Cortex A9 multi-processor. The paper demonstrates that power down followed by a full reconfiguration can be controlled by the embedded processor autonomously. The results show that the minimum time that the FPGA fabric must remain in power-off state for the technique to be energy efficient is in the order of milliseconds and up to 96% power reduction occurs when the fabric voltage is lowered below critical level. These results take into account the overheads of controlling the programmable voltage regulators interfaced to the FPGA and the overhead of the reconfiguration needed when the device must be returned to the active state.**

Keywords*—Energy Proportional Computing, Power Gating, ZYNQ, FPGA

I. Introduction

Energy consumption is one of the main constraints in the new area of embedded and mobile multi/many-core heterogeneous systems [1]. Energy proportional computing (EPC) has emerged as a solution to restrict the energy to the exact amount required by a respective application [2]. The dynamic behaviour of EPC requires that the target system provides fine grained software and hardware reconfiguration features. Modern FPGAs (Field Programmable Gate Arrays) offer partial and dynamic reconfiguration and recent research has shown that they are also suitable for dynamic voltage and frequency scaling [3]. This makes them a suitable candidate to realise the underlying hardware required to implement an energy proportional computing platform.

One of the effective techniques in EPC is the power gating of unused modules [4][5] in which the modules that are not used for a certain amount of time are shut down and then turned on whenever they are required. This technique reduces the energy consumption caused by leakage and clock activity while the module is not required in a system.

This paper investigates a software-controlled power gating technique applied to the Zynq-7000 All Programmable SoC platform [6]. The Zynq-7000 consists of a dual-core ARM Cortex-A9 processor as the processing system (PS) and a Xilinx 7 series FPGA as the programmable logic (PL). The PL and PS are in two different power domains supplied by programmable voltage regulators. This enables the PS to control the PL power rails programmatically. In this investigation, run-time software-controlled PL power gating is used to reduce the energy consumption in an application containing idle modules. In this technique, when the hardware design in the PL is in its idle mode or it is not required any more, the PL power can be reduced by clearing the PL configuration data and reducing its power supplies. In order to reuse the PL, its power supplies should be increased to nominal levels and it should be reconfigured again. The usage of this technique is twofold: shutdown the PL to reduce power and replace a hardware module with another which increases the hardware utilization and delivers EPC. The experimental results show that for a large design in the PL which contains five MicroBlaze soft processor cores [7] to save energy the module idle time should be longer than $42.58msec$. In this case, the idle module power reduces by the factor of 30.43. For the other smaller designs the idle module power reduces by the factor between 2.56 and 17.05.

The rest of this paper is organized as follows. The next section reviews the previous work on power-gating in FPGAs. The motivations and contributions of this paper are explained in Section 3. Section 4 discusses the details of the proposed technique and its accuracy. Experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

II. Previous Work

Power gating in which some parts or the design are shut down for a period has been proposed as a low-power technique in the FPGA area especially for reducing leakage power. Researchers have proposed techniques to implement power gating in different levels of design granularity including transistor level, gate level, look-up table level, and module level.

The basic idea for power gating is adding a switch to the design in order to disconnect the power supply from the circuit. At the transistor level, a single transistor called *sleep transistor* can realise this switch as shown in Fig. 1. The SLEEP input in Fig. 1 controls the sleep transistor. When the sleep transistor is off the leakage power is limited to that of the sleep transistors which is negligible. Considering this sleep transistor, researchers have proposed different power gating techniques for FPGA architectures. Note that in the FPGA area, as well as the performance overhead caused by sleep transistors, adding these transistors requires modification to the underlying FPGA fabric. Therefore, researchers usually use simulation techniques along with synthetic applications to evaluate their proposed techniques. In contrast to the logic design techniques reviewed in this section, the technique proposed in this paper can be applied to commercially available FPGAs and controls the FPGA power lines directly by adjusting the voltage levels at the output of the programmable voltage regulators which provide power to the fabric.

Statically control of the sleep transistor using FPGA configuration data is presented in [8]. This technique utilises a

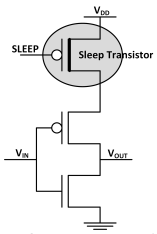


Fig. 1: Basic idea of power gating in logic design

new design methodology by grouping the design into clusters based on temporal locality and then turn-off/on clusters by configuration data.

Dynamically controlled power gating is presented in [9] in which the SLEEP signals are connected to general-purpose routing fabrics in the FPGA and are controlled by separate circuits or circuits part of the FPGA itself. This technique, can switch off a specific part of the design or a logic cluster in FPGA. This technique is evaluated by HSPICE simulation using an application model. In contrast, the technique proposed in this paper is applied to a commercial FPGA and can be used by all designers.

Ishihara et al. [10] propose a look-up table level power gating technique in which the power of the look-up tables with fine granularity can be controlled by a sleep transistor. This techniques relies on a new FPGA architecture and it is not applicable to commercially available FPGAs.

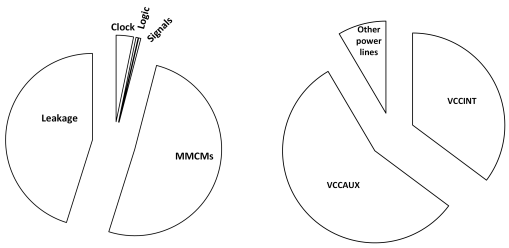
FPGA manufactures have used different static power gating at the module level to shut down unused PLLs, DCMs, I/Os during the FPGA configuration. For example, as block RAMs are the source of the 30% of total leakage power in an FPGA [11], Xilinx provides static power gating techniques for block RAMs in $28nm$ 7-series devices. Therefore, only the block RAMs used by a design cause leakage.

III. Motivations and Contributions

Using a simple design, this section explains the motivations and contributions behind this research. Before delving into the details, the next subsection describes the underlying platform considered in this research.

A. Zynq-7000 platform description

The ZC702 [6] evaluation board is used to perform this research. The board utilises a Zynq-7000 SoC consisting of a dual-core ARM Cortex-A9 processor as the processing system (PS) and a Xilinx 7-series FPGA as the programmable logic (PL). The PS and PL power domains are completely independent [6]. Each PS and PL has six different power pins to provide power for their different parts. As this paper only focuses on PL power gating, the power lines for PL are mentioned here. Interested readers can refer to [6] for more details on these power pins. V_{CCINT}, V_{CCAUX}, $V_{CCO\_\#}$, V_{CC_BATT}, V_{CCBRAM}, and $V_{CCAUX\_IO\_G\#}$ are the PL power pins [6]. The two PL power lines that are considered in this paper are V_{CCINT} and V_{CCAUX}. Whereas V_{CCINT} provides the power for the internal core logic, the V_{CCAUX} provides the power for auxiliary logic such as I/O buffer pre-drivers, along with Mixed-Mode Clock Managers (MMCMs) and PLLs.



(a) Xilinx XPower Analyzer (b) Real power measurement on ZC702

Fig. 2: Power consumption

B. Motivations

A simple motivation example is considered in this subsection. The example consists of a MicroBlaze soft processor core configured in the PL and the ARM processor available in the PS. The MicroBlaze runs two different programs, separately. The first one just prints messages on the console and the second one performs a 32×32 floating point matrix multiplication. Table I shows the PL resources used by this design.

TABLE I: PL resources used by the MicroBlaze example

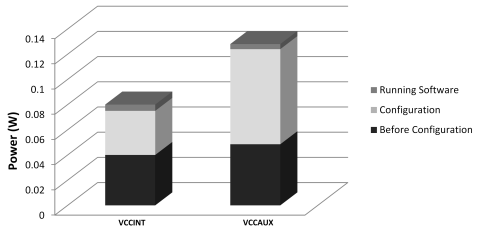
Slice LUTs	Slice Register	MMCM	DSP48E	RAM36E
1725	1525	1	3	4

Fig. 2(a) shows the contribution of each part of the PL design in the consumed power obtained by Xilinx XPower Analyser [12] without applying any input patterns. As it can be seen, the MMCM module is the most power intensive due to clock activity. Leakage is the next important part in the consumed power. Logic and signals show a negligible contribution in the power consumption. Note that, leakage power is consumed as long as the design exist in the PL, whether the design is used or is in the idle mode. A real measurement for the same design on the ZC702 board, depicted in Fig. 2(b), shows that V_{CCINT} and V_{CCAUX} are the two power rails that provide most of the power consumed by the PL. Since V_{CCINT} provides the voltage for logic circuits in the PL then it mainly represents the leakage power due to lack of input activity in this measurement. In addition, as V_{CCAUX} provides the voltage for MMCM, it represents the power consumption caused by the clock activity as well as the leakage power in the related logic.

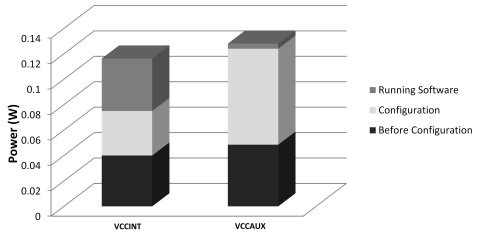
Considering the two aforementioned programs, Fig. 3 shows the contribution of consumed power in the PL for three different phases: PL turn-on without configuration (i.e., Before configuration), PL with configuration (i.e., Configuration), and PL used by the application (i.e., Running software). As it can be seen, the PL power consumption after configuration is a considerable portion of power even if PL does not run any program yet and PL is in the idle mode. Therefore, turning-off the PL for designs in which PL is in idle model for a long time can reduce the total power consumed by the corresponding application. This motivates our research to investigate power savings with PL power-off in commercial FPGAs. Note that, although using the clock gating technique can reduce the power consumption due to clock activity in the MMCM, PL power gating technique reduces both leakage and clock activity power consumption.

C. Contributions

The main contribution of this paper, compared to the previous work, is to investigate software-controlled run-time power-



(a) MicroBlaze processor in PL printing messages on terminal



(b) MicroBlaze processor in PL running floating point matrix multiplication

Fig. 3: MicroBlaze power consumption

gating in a system formed by and FPGA (i.e., PL) and ARM processor (i.e., PS) subsystems. For this purpose, considering the ZC702 board, this paper:

- provides a group of functions for turning off, turning on and reconfigure the PL at run-time
- measures the time and energy overhead associated with PL power gating
- identifies the possible utilisation scenarios when power gating is a viable option

The software-controlled FPGA power gating can be used in two main scenarios. In the first scenario the PL can be turned off to save energy. In the second scenario, the configuration in the PL can be replaced with more efficient design in respect to the workload at run-time to increase the application throughput and obtain energy consumption proportional to computing.

D. Problem formulation

This subsection formulates the contributions of this research in which the PL is shut down when its design is in idle mode and it is powered off/on and reconfigured by the PS. Fig. 4 shows the PL power gating timeline considered in this research. This timeline shows three periods for turning off the PL and three periods for turning on the PL. These periods are explained below. As the PL loses its configuration after turning off, the state of the design should be saved to be used after resumption. The time and power associate to this period are denoted by t_{ss} and P_{ss}. To turn off the PL, the signalling between PS and PL should be terminated and the voltage levels of the corresponding power supplies should be reduced. The t_{tfpl} and P_{tfpl} represent the time and power associated to this period, respectively. After turning off the PL, there is not any signalling between PS and PL and PL power consumption is almost zero. The time and power associated to this period are denoted by t_{pltf} and P_{pltf}. If we want to reuse the PL, its power supplies should be increased to the nominal values. The t_{tnpl} and P_{tnpl} determine the time and power associated to this period, respectively. The next period (after turning on the PL) is reconfiguring the PL and associated time and power are denoted by t_{reconf} and P_{reconf}. Finally, the last period is to restore the previous state and the associated time and power are denoted by t_{rs} and P_{rs}.

The total power consumption during these periods is

$$P_{powerGating} = P_{ss} + P_{tfpl} + P_{pltf} + P_{tnpl} + P_{reconf} + P_{rs} \quad (1)$$

For simplicity, a constant average power is considered for each period. With this assumption, the total energy during this periods equals to

$$E_{powerGating} = t_{ss}.P_{ss} + t_{tfpl}.P_{tfpl} + t_{pltf}.P_{pltf} + t_{tnpl}.P_{tnpl} + t_{reconf}.P_{reconf} + t_{rs}.P_{rs} \quad (2)$$

To save energy using the PL turn-off technique, this energy should be less than the energy consumed by the PL in the idle state, i.e.,

$$E_{powerGating} < E_{plidle} \quad (3)$$

in which E_{plidle} denotes the energy consumed by PL when its design is idle, without considering the power gating technique. This energy is equal to the multiplication of t_{plidle} and P_{plidle} which denote the duration of PL in idle model and its corresponding average power consumption, respectively.

By substituting the $E_{powerGating}$ and E_{plidle} in Equ. 3 with the corresponding expression, Equ. 4 can be obtained.

$$t_{ss}.P_{ss} + t_{tfpl}.P_{tfpl} + t_{pltf}.P_{pltf} + t_{tnpl}.P_{tnpl} + t_{reconf}.P_{reconf} + t_{rs}.P_{rs} < t_{plidle}.P_{plidle} \quad (4)$$

In addition, to use the PL turn-off technique the PL idle time should greater than the technique timing overhead, therefore:

$$t_{plidle} > t_{ss} + t_{tfpl} + t_{tnpl} + t_{reconf} + t_{rs} \quad (5)$$

If we assume P_{pltf} is very small and negligible then, to save energy, PL idle time should satisfy Equ. 6, using Equ. 4

$$t_{plidle} > (t_{ss}.P_{ss} + t_{tfpl}.P_{tfpl} + t_{tnpl}.P_{tnpl} + t_{reconf}.P_{reconf} + t_{rs}.P_{rs})/P_{plidle} \quad (6)$$

The minimum PL idle time for which PL power-off can save energy is called *power-off-efficiency time* in this paper and should satisfy Equs. 5 and 6.

IV. PL Power Gating Framework

This section explains the PL power gating framework and its accuracy.

A. Framework setup

A framework has been developed to implement the PL power gating technique and power measurement in the ZC702 evaluation board. Fig. 5 shows an overview of this framework. It consists of three components: PS, PL, and UCD9248 which is a digital PWM controller. The ZC702 board utilises three UCD9248 digital PWM controllers to control PL and PS voltages and monitor energy consumption in different parts of the system. This controller supports a wide range of PMBus [13] commands including voltage/current

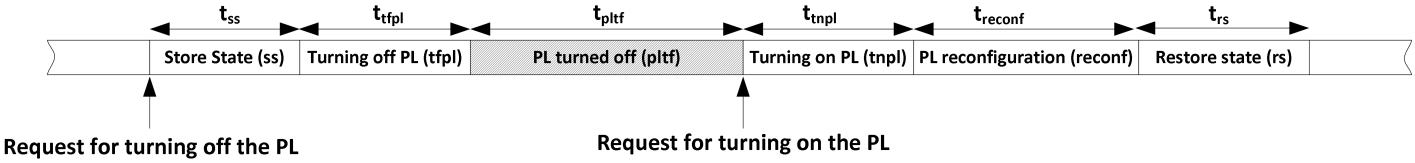


Fig. 4: PL power-gating timeline

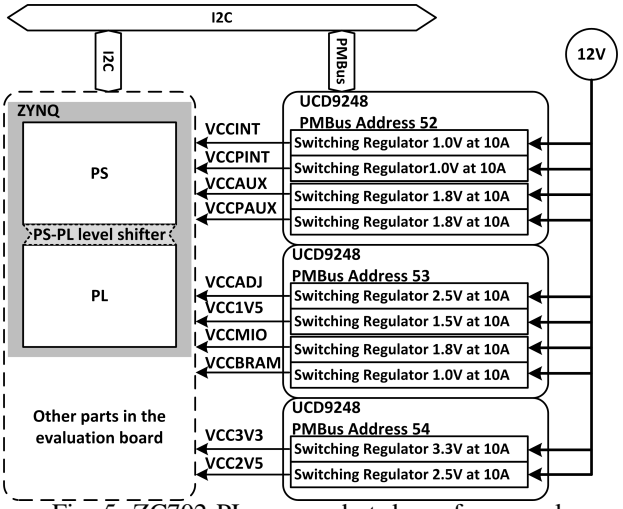


Fig. 5: ZC702 PL power shut down framework

monitoring and adjustment. The three UCD2948 available in the ZC702 board provides ten power rails supplying the PS and PL parts. The output voltage at a rail can be adjusted by PMBus commands via determining a few parameters in the control circuit including $VOUT\_MARGIN\_HIGH$, $VOUT\_MARGIN\_LOW$, $VOUT\_COMMAND$, $VOUT\_MAX$, and $VOUT\_CALL\_OFFSET$ [13]. UCD9248 employs different protection mechanisms to prevent the circuits from getting damaged. To adjust the output voltage at a rail, the voltage protection levels should also be modified. The protection level parameters include $VOUT\_OV\_FAULT\_LIMIT$, $VOUT\_OV\_WARN\_LIMIT$, $POWER\_GOOD\_ON$, $VOUT\_UV\_WARN\_LIMIT$, $VOUT\_UV\_FAULT\_LIMIT$, $POWER\_GOOD\_OFF$, and $IOUT\_OC\_LV\_FAULT\_LIMIT$. For detail information about each parameter and the corresponding PMBus command, interested readers can consult [13]. The PS component runs the software part which performs and controls the PL power gating. This software consists of three parts: turn-off PL, turn-on PL and PL reconfiguration. In addition, the software parts can monitor voltage and current at each power rail. The software part employs three groups of functions including energy measurement, power gating, and PL reconfiguration functions. Using the PMBus commands, the first group of functions read the current and voltage on each output power rail provided by the UCD9248. The details of these commands can be found in [14]. The second group of functions adjust the voltage levels at the output of UCD9248 which are used to turn-off and turn-on PL. These functions use the PMBus commands and UDC9248 driver provided by Xilinx. Algorithm 1 contains the PMBus commands for reducing the V_{CCINT} power rail connected to PL. The algorithm consists of three steps: *pre-adjustment*, *adjustment*, and *post-adjustment*. Lines 3-8 modify the lower voltage level protections, Line 10 sends the PMBus command to change the voltage level at the V_{CCINT}. Finally, Lines 12-17 modify the upper voltage level protections.

Algorithm 1: Turn off VCCINT

```
1  Reduce-VCCINT() {
2    //Step 1: pre-adjustment
3    Modify POWER_GOOD_OFF,
4           POWER_GOOD_ON,
5           IOUT_OC_LV_FAULT_LIMIT,
6           VOUT_UV_FAULT_LIMIT,
7           VOUT_UV_WARN_LIMIT,
8           VOUT_MARGINE_LOW
9    //Step 2: adjustment
10   Reduce VCCINT voltage using VOUT_COMMAND
11   //Step 3: post-adjustment
12   Modify VOUT_MARGIN_HIGH,
13          VOUT_MAX,
14          VOUT_OV_FAULT_LIMIT,
15          VOUT_OV_WARN_LIMIT,
16          VOUT_MARGIN_HIGH,
17 }
```

Reconfiguration functions in the software part use the *xdevcfg* driver through Processor Configuration Access Port (PCAP) interface which is 32 bits wide and clocked at 100 MHz and provides 400 MB/s download via non-secure PL configuration mechanism[6]. These functions use a Direct Memory Access (DMA) mechanism to transfer the configuration bitstream to the PL part at run-time.

B. Framework Accuracy

PL shut-down is an option considered by the manufacture to save power [16]. According to the Zynq-7000 technical reference [16], the sequence for turning-off the PL consists of three steps: stop using signals between PS and PL, disable PS-PL level shifter, and shut-down PL. To get the biggest benefit of PL power gating, all PL power supplies (i.e., V_{CCINT}, V_{CCBRAM}, V_{CCAUX}, V_{CCAUX_IO}, and V_{CCO}) should be turned off in the correct order. However, the ZC702 board has a deficiency in revision 1.0 that uses V_{CCAUX} to supply power to a clock generator for the PS therefore, turning off the V_{CCAUX} freezes the PS which is not desired. Hence, this research, instead of completely shutting down the PL, reduces the PL V_{CCINT} voltage to a level in which the PL loses its configuration and significantly lower PL energy. As the V_{CCINT} goes below $VDRINT$ [17](i.e., data retention voltage) the PL configuration is lost. By reducing the V_{CCINT} voltage, Fig. 6 shows the changes in power consumption in the PL provided by V_{CCINT} and V_{CCAUX} for the motivation example. As it is expected reducing the V_{CCINT} reduces the corresponding power consumption. Fig. 6 shows that at the V_{CCINT} voltage of around $0.55V$, the PL loses its configuration such that V_{CCAUX} does not deliver power to the circuits, especially to MMCMs.

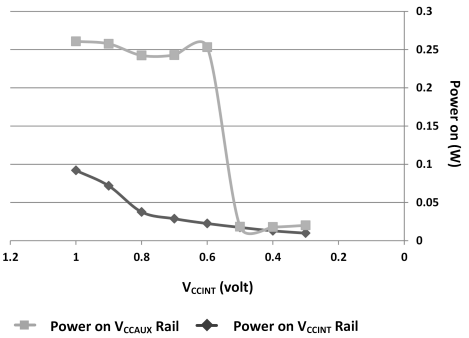


Fig. 6: Power versus V_{CCINT} voltage on ZC702

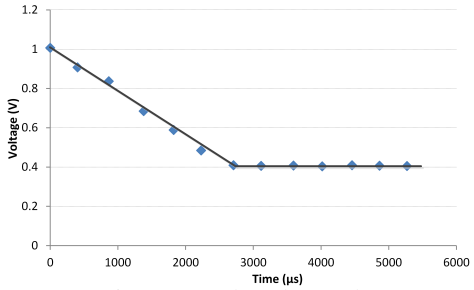


Fig. 7: PL shut-down slop

This results in a sharp drop in V_{CCAUX} power. The voltage level $0.4V$ is investigated as a critical level to turn off the PL. At this voltage level the ZC702 board shows a stable behaviour such that it can be turned on without instability in other parts of the system.

Fig. 7 shows the V_{CCINT} voltage reduction time it takes to reach $0.4v$. As it can be seen it takes $2.73msec$ to reduce this voltage. As shown in Algorithm 1, adjusting the V_{CCINT} voltage consists of three steps: pre-adjustment, adjustment and post-adjustment. The timing delay for changing the V_{CCINT} comprises of pre-adjustment and adjustment execution times as well as the time required to change the voltage from $1v$ to $0.4v$. This delay can be considered as the timing overhead for turning off the PL.

$$T_{tfpl} = T_{pre-adjustment} + T_{adjustment} + T_{V_{CCINT}(1to0.4)} \quad (7)$$

These times, shown in Equs. 8-10 are measured for the motivation example. Note that these times are not dependant on the design in the PL, becuase they are defined by the UCD9248.

$$T_{pre-adjustment} = 1.9msec \quad (8)$$

$$T_{adjustment} = 205usec \quad (9)$$

$$T_{V_{CCINT}(1to0.4)} = 2.73msec \quad (10)$$

An algorithm similar to Algorithm 1 can be used to turn-on the PL. Therefore,

$$T_{tnpl} = T_{pre-adjustment} + T_{adjustment} + T_{V_{CCINT}(0.4to1)} \quad (11)$$

After turning on the PL, the PS uses PCAP to reconfigure the PL. Before reconfiguration, PS consumes the power corresponding to the running OS and software applications. During reconfiguration, which lasts about $34.33msec$, the PS power consumption increases by an average of $20mW$. The configuration time consists of PL initialization and DMA transfer delay.

V. Experimental Results

This section explains experimental results obtained by applying the proposed technique to four different designs in the PL. Table II shows statistics of FPGA resources used by these designs. The smallest design is an AXI-timer which is a simple 32-bit counter connected to PS via AXI bus interface. The second design is a motion estimation processor [18]. The third design consists of a MicroBlaze in the PL. Finally, the last and largest design contains five MicroBlazes. Note that, we have assumed that these four designs do not require to keep their states during PL turn-off and the state saving/recovery timing overhead is zero.

TABLE II: PL resources used by different designs

	AXI-Timer	Motion Estimation	One-MicroBlaze	Five-MicroBlaze
Slice LUTs	936	18351	1725	12708
Slice Register	890	14484	1525	30762
MMCM	0	1	1	1
DSP48E	0	0	3	5
RAM36E	0	43	4	135

Table III shows the timing overhead for turning off, turning on, and reconfiguring the PL. As mentioned in Subsection IV-B, the tuning off/on timing is determined by the UCD9248 and are independent of the design in the PL. In addition, the PL reconfiguration time depends on the bitstream file size. Since for each FPGA the size of the bitstream file is constant, and for the Zynq 7020 FPGA used in this research the PL configuration file used by PCAP is $4,045,564\,bytes$ then the configuration time is about $34.33msec$. This configuration time, which consists of PL initialization timing overhead and bitstream transferring delay is measured between the PL initialization call and the time when the FPGA DONE signal is asserted [6]. Note that the PCAP theoretical throughput is 400 MB/s resulting in $10msec$ delay for transferring $4,045,568\,bytes$. Considering the configuration software overhead the throughput is lower than this maximum value. Kohn [19] has reported $32msec$ and $44msec$ for only DevC DMA transfer delay in standalong and Linux environments, respectively. Vipin and Fahmy [20] has proposed a partial reconfiguration management technique called ZyCAP that reaches 382 MB/s throughput for PL configuration. Another partial reconfiguration management technique proposed by [21] reaches 385 MB/s throughput for PL partial reconfiguration. However, these work have used the PL itself to achieve this throughput which makes that suitable for partial reconfiguration not PL complete configuration.

TABLE III: Timing overheads in $msec$

Period	Time ($msec$)
Turn-off PL	4.84
Turn-on PL	4.84
Reconfiguration PL	34.33

By reducing V_{CCINT} to $0.4V$, the PL loses its configuration and PL power consumption reduces to $20.7mW$. Table IV shows the PL idle power consumption, PL turn-off power consumption and power reduction for different design examples. As it is expected, the power reduction of the PL shut down is high, however, to make this technique efficient in an EPC scenario, the PL shut down period should be longer than a minimum value which is called *turn-off-efficiency time* as mentioned in Subsection III-D. Table V shows the lower level times determined by Equs. 5 and 6 for the four designs. As it can be seen, the turn-

off-efficiency times are determined by the timing overhead of the proposed technique and are $42.58msec$. Table VI shows the percentage of the energy reduction during the idle time considering the turn-off-efficiency time. Note that, the proposed technique shows much more energy reduction if the PL idle time is much larger than the turn-off-efficiency time.

TABLE IV: Power consumption comparison

	AXI-Timer	Motion Estimation	One-Microblaze	Five-Microblaze
Power of the PL in idle mode (W)	0.053	0.15	0.353	0.63
Power of the PL shut down (W)	0.0207	0.0207	0.0207	0.0207
Power reduction	60.9%	86.2%	94.1%	96.7%

TABLE V: Turn-off-efficiency time in $msec$

	AXI-Timer	Motion Estimation	One-Microblaze	Five-Microblaze
Lower level time by Equ. 6	18.6	6.5	2.7	1.5
Lower level time by Equ. 5	42.58	42.58	42.58	42.58
Turn-off-efficiency time	42.58	42.58	42.58	42.58

TABLE VI: Energy reduction in $\%$

AXI-Timer	Motion Estimation	One-Microblaze	Five-Microblaze
56.4	84.59	93.45	96.33

As it can be seen, the FPGA reconfiguration time is the dominant component in the turn-off-efficiency time parameter. Therefore, as a rule of thumb, this reconfiguration time can be considered as the timing overhead of PL turn-off for the ZYNQ platform. In addition, the efficiency of this technique depends on the power consumption of the PL during its idle period. Note that, this power can also be reduced by other techniques such as voltage/frequency scaling or clock gating.

VI. Conclusions and Future Work

This paper has demonstrated a run-time software-controlled power gating technique applied to the Xilinx ZYNQ embedded system platform to reduce FPGA power consumption when idle states can be identified during system operation. The Xilinx ZYNQ platform consists of two parts: a dual-core ARM processor and an FPGA. In the proposed technique, the ARM processor controls the FPGA power supplies via the programmable voltage regulator available in the ZC702 evaluation board. The ARM processor also reconfigures the FPGA after turning on the corresponding power supplies.The experimental results show that the overhead of this technique is in the order of milliseconds which makes it suitable to be used in energy proportional computing techniques. As the PL configuration overhead is the main bottleneck in the proposed technique, one of the future goal of this research is proposing new architectures and strategies to reduce this overhead. In addition, future work will compare the proposed technique with the voltage scaling technique proposed in [3].

Acknowledgement

The authors would like to thank the reviewers for their valuable comments and especially our colleague Dr. Arash Farhadi, whose suggestions helped us to do this research. This research is a part of the ENPOWER project sponsored by EPSRC and done in collaboration with Queen's University Belfast.

References

[1] ITRS, "International technology roadmap for semiconductors: System drivers," Tech. Rep., 2011 EDITION. [Online]. Available: http://www.itrs.net/Links/2011ITRS/2011Chapters/2011SysDrivers.pdf

[2] K. W. Cameron, "The challenges of energy-proportional computing," *IEEE Computer*, vol. 5, no. 4, pp. 82–83, 2010.

[3] J. L. Nunez-Yanez, "Adaptive voltage scaling with in-situ detectors in commercial FPGAs," *IEEE Transactions on Computers*, 2013. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TC.2013.73

[4] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 320–331. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485950

[5] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of datacenter workloads using per-core power gating," *IEEE Comput. Archit. Lett.*, vol. 8, no. 2, pp. 48–51, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1109/L-CA.2009.46

[6] Xilinx, "Zynq-7000 all programmable SoC," Xilinx, Technical Reference Manual, UG585 (v1.6.1), 2013.

[7] ——. (2014) Microblaze soft processor core. [Online]. Available: http://www.xilinx.com/tools/microblaze.htm

[8] R. P. Bharadwaj, R. Konar, P. T. Balsara, and D. Bhatia, "Exploiting temporal idleness to reduce leakage power in programmable architectures," in *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '05. New York, NY, USA: ACM, 2005, pp. 651–656. [Online]. Available: http://doi.acm.org/10.1145/1120725.1120985

[9] A. A. M. Bsoul and S. J. E. Wilton, "An FPGA architecture supporting dynamically controlled power gating," in *International Conference on Field-Programmable Technology*, ser. FPT'10, 2010, pp. 1–8.

[10] 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.

[11] M. K. J. Hussein and M. Hart, "Lowering power at 28 nm with Xilinx 7 series devices," Xilinx, White paper, WP389 (v1.2), 2013.

[12] Xilinx, "Xpower estimator user guide," User Guide, UG440 (v2013.4), 2013.

[13] SMIF, "PMBus power system management protocol specification part II-command language," System Management Interface Forum, Protocol Specification, 2007.

[14] T. Instrument, "UCD92xx digital PWM system controller PMBus command reference," Texas Instrument, User's Guide SLUU337, 2008.

[15] Xilinx. (2014) Xilinx linux kernel source. [Online]. Available: https://github.com/Xilinx/linux-xlnx

[16] ——, "ZC702 evaluation board for the Zynq-7000 XC7Z020 all programmable SoC," Xilinx, User Guide, UG850 (v1.2), 2013.

[17] ——, "Zynq-7000 All Programmable SoC(Z-7010, Z-7015, and Z-7020):DC and AC Switching Characteristics," Xilinx, Product Specification DS187 (v1.11), 2014.

[18] J. L. Nunez-Yanez, A. Nabina, E. Hung, and G. Vafiadis, "Cogeneration of fast motion estimation processors and algorithms for advanced video coding." *IEEE Trans. VLSI Syst.*, vol. 20, no. 3, pp. 437–448, 2012.

[19] C. Kohn, "Partial reconfiguration of a hardware accelerator on zynq-7000 all programmable soc devices," Xilinx, XAPP1159 (v1.0), 2013.

[20] V. K. and F. S. A., "Zycap: Efcient partial reconguration management on the xilinx zynq," in *IEEE Embedded Systems Letters*, 2013. [Online]. Available: http://dx.doi.org/10.1109/LES.2014.2314390

[21] A. Nabina and J. L. Nez-Yez, "Dynamic reconfiguration optimisation with streaming data decompression." in *FPL*. IEEE, 2010, pp. 602–607. [Online]. Available: http://dblp.uni-trier.de/db/conf/fpl/fpl2010.html#NabinaN10