Nunez-Yanez, J. L., Hosseinabady, M., & Farhadi Beldachi, A. F. (2016). Energy Optimization in Commercial FPGAs with Voltage, Frequency and Logic Scaling. IEEE Transactions on Computers, 65(5), 1484-1493. DOI: 10.1109/TC.2015.2435771

Energy Optimization in Commercial FPGAs with Voltage, Frequency and Logic Scaling

Jose Luis Nunez-Yanez, Mohammad Hosseinabady, Arash Beldachi
Department of Electrical and Electronic Engineering
University of Bristol, UK.
E_mail : j.l.nunez-yanez@bristol.ac.uk, m.hosseinabady@bristol.ac.uk, arash.beldachi@bristol.ac.uk

Abstract— **This paper investigates the energy reduction possible in commercially available FPGAs configured to support voltage, frequency and logic scalability. Voltage and frequency scaling is based on in-situ detectors that allow the device to detect valid working voltage and frequency pairs at run-time while logic scalability is achieved with partial dynamic reconfiguration. The considered devices are FPGA-processor hybrids with independent power domains fabricated in 28 nm process nodes. The test case is based on a number of operational scenarios in which the FPGA side is loaded with a motion estimation core that can be configured with a variable number of execution units. The results demonstrate that voltage scalability reduces power by up to 60% compared with nominal voltage operation at the same frequency. The energy analysis show that the most energy efficiency core configuration depends on the performance requirements. A low performance scenario scenario shows that the serial core is more energy efficiency than the parallel core while the opposite is true under hard performance constraints. Finally, the paper considers power gating as an additional power knob possible in the device and investigates its feasibility, overheads and the scenarios when it could be deployed.**

***Index Terms*— FPGA, Power Gating, Energy Optimization, DVFS, AVS.**

I. INTRODUCTION

Energy and power efficiencies in Field Programmable Gate Arrays (FPGAs) have been estimated to be up to one order of magnitude worse than in ASICs [1] and this limits their applicability in energy constraint applications. Since FPGAs are fabricated using CMOS transistors power can be divided into two main categories, dynamic power and static power. Equ. 1 shows a simplified relation of power with voltage, frequency and capacitance.

$$\text{Power} = \alpha C V^2 F + g_1 V^3 \quad (1)$$

Where α = Activity factor < 1, C = Capacitance, V = Supply voltage, F = Frequency, g_1 = leakage factor. The first term of the equation represents dynamic power and the second static power. This equation shows how voltage affects both static and dynamic power. It is apparent that lowering the supply voltage in CMOS circuits reduces both dynamic and static power but it also has a cost in increased gate delay which in turns may cause the circuit to fail to satisfy timing constrains. As a result, voltage scaling is often combined with frequency scaling in order to compensate for the variation of circuit delay. An example of this is Dynamic Voltage and Frequency Scaling (DVFS) which is a technique that uses a number of pre-evaluated voltage and frequency operational points to scale power, energy and performance. With DVFS, margins for worst case process and environmental variability are still maintained since it operates in an open-loop configuration. However, worst case variability is rarely the case. For that reason, in Adaptive Voltage Scaling (AVS), run-time monitoring of performance variability in the silicon is used together with system characterization to influence the voltage and the frequency on the fly in a closed-loop configuration. The importance of this technology for future microprocessor design has been discussed in [2] that advocates for the need to consider also hardware customization at run-time to deliver the performance increases and the low power required over the next 20 years. In this work, we investigate a system built using standard FPGA devices that supports adaptive voltage scaling extended with logic reconfigurable at run-time that changes circuit capacitance. Logic scalability in this context is conceptually similar to technologies such as ARM big.LITTLE [3] that can use power gating to switch off processor cores of different complexity depending on workload demands, effectively changing capacitance. In the proposed approach the variation of the triplet formed by voltage, frequency and capacitance can happen at a finer granularity. The contributions of this work can be summarised as follows:

1. We present a power adaptive architecture that includes run-time support for voltage, frequency and logic scaling in commercial devices that combine a hardwired microprocessor and a FPGA fabric.
2. We investigate the best strategy to obtain power and energy savings with this power adaptive architecture using as a test case a reconfigurable motion estimation processor with a variable number of execution units.
3. We extend the framework with a power gating control knob that shuts down the FPGA fabric during idle periods and investigate its feasibility and overheads.

The rest of the paper is structured as follows. Section 2

describes related work. Section 3 presents the main features of the reconfigurable motion estimation processor used as a test case. Section 4 presents the power adaptive architecture that allows the device to dynamically regulate its voltage, frequency and logic complexity. Section 5 analyses the robustness of the approach and the conditions that must be met to enable stable operation. Section 6 presents and discusses the results focusing on power while Section 7 extends this analysis to energy. Section 8 investigates the feasibility of power gating the FPGA device and its interaction with the proposed scaling. Finally, Section 9 presents the conclusions and future work.

II. Related Work

In order to identify ways of reducing the power consumption in FPGAs, some research has focused on developing new FPGA architectures implementing multi-threshold voltage techniques, multi-Vdd techniques and power gating techniques [4-8]. These techniques are designed to be used in new architectures and devices. Other strategies have proposed modifying the map and place&route algorithms to provide power aware implementations [9-11]. Similarly main FPGA manufactures currently offer switches in their tools that optimize the synthesis and implementation runs for power in addition to performance or area. Recently FPGA manufactures have also started investigating the topic of voltage and frequency adaptation. Xilinx supports the possibility of using lower voltage levels to save power in their latest family implementing a type of static voltage scaling in [12]. The voltage identification *VID* bit available in Virtex-7 allows some devices to operate at 0.9 V instead of the nominal 1 V maintaining nominal performance. During testing, devices that can maintain nominal performance at 0.9 V are programmed with the voltage identification bit set to 1. A board capable of using this feature can read the voltage identification bit and if active can lower the supply to 0.9 V reducing power by around 30%. Altera offers a similar technology with the *SmartVoltage ID* bit and future devices will take this concept further with *Vcc PowerManager*. These devices can operate at either the standard Vcc voltage or a set lower voltage level by lowering the frequency. This feature can reduce total power by up to 40 percent and is suitable when maximum performance is not required [13] all the time. These techniques are open-loop in the sense that valid working points are defined at fabrication time and not detected at run-time as in this research. Run-time dynamic voltage scaling strategy for commercial FPGAs that aims to minimise power consumption for a giving task is presented in [14]. In this methodology, the voltage of the FPGA is controlled by a power supply that can vary the internal voltage of the FPGA. For a given task, the lowest supply voltage of operation is experimentally derived and at run-time, voltage is adjusted to operate at this critical point. A logic delay measurement circuit is used with an external computer as a feedback control input to adjust the internal voltage of the FPGA (VCCINT) at intervals of 200ms. With this approach, the authors demonstrate power savings from 4% to 54% from the VCCINT supply. The experiments are performed on the Xilinx Virtex 300E-8 device fabricated on a 180nm process technology. The logic delay measurement circuit (LDMC) is an essential part of the system because it is used to measure the device and environmental variation of the critical path of the functionality implemented in the FPGA and it is therefore used to characterise the effects of voltage scaling and provide feedback to the control system. This work is mainly presented as a proof of concept of the power saving capabilities of dynamic voltage scaling on readily available commercial FPGAs and therefore does not focus on efficient implementation strategies to deliver energy and overheads minimisation. A similar approach is also demonstrated in [15]. In this case a dynamic voltage scaling strategy is proposed to minimise energy consumption of an FPGA based processing element, by adjusting first the voltage, then searching for a suitable frequency at which to operate. Again, in this approach, first the critical path of the task under test is identified, then a logic delay measurement circuit is used to track the critical point of operation as voltage and frequency are scaled. Significant savings in power and energy are measured as voltage is scaled from its nominal value of 1.2V down to its limit of 0.9V. Beyond this point, the system fails. The experiments were carried out on a Xilinx ML402 evaluation board with a XC4VSX35-FF668-10C FPGA fabricated in a 90 nm process and energy savings of up to 60% are presented. The previously presented efforts are based on the deployment of delay lines calibrated according to the critical path of the main circuit. This calibration is cumbersome and it could lead to miss tracking due for, for example, the different locations of the delay line and the critical paths of the circuit having different temperature profiles. In-situ detectors located at the end of the critical paths remove the need for calibration. This technology has been demonstrated in custom processor designs such as those based around ARM Razor [16]. Razor allows timing errors to occur in the main circuit which are detected and corrected re-executing failed instructions. The lastest incarnation of Razor uses a highly optimized flip-flop structure able to detect late transitions that could lead to errors in the flip-flops located in the critical paths. The voltage supply is lower from a nominal voltage of 1.2V (0.13µm CMOS) for a processor design based on the Alpha microarchitecture observing approximately 33% reduction in energy dissipation with a constant error rate of 0.04%. The Razor technology requires changes in the microarchitecture of the processor and it cannot be easily applied to other non-processor based designs. Research targeting FPGAs presented in [17] uses a recalibration technique to remove the variable delays introduced by a detector that exhibits variable placement and routing. Our previous work has demonstrated the power and energy benefits of deploying voltage scaling using uncalibrated in-situ detectors in commercial FPGAs in [18]. In this paper we extend the work in [18] by evaluating the power and energy synergies possible by deploying voltage scaling with user designs that can be reconfigured at run-time with different levels of complexity implementing logic scalability. The proposed architecture is based on devices with a hardwired processor core and FPGA in independent power domains. This configuration allows the processor controls the run-time

adaption of frequency, voltage and logic in the FPGA fabric. This CPU-FPGA hybrid also offers the possibility of power gating the FPGA fabric and its interaction with voltage and frequency scaling and overall feasibility are investigated.

III. RECONFIGURABLE MOTION PROCESSOR TEST SYSTEM

A test system based around a reconfigurable motion estimation processor has been built to explore the effects of combining voltage, frequency and logic scalability in the same FPGA chip. The motion estimation microarchitecture is shown in Fig. 1. It is formed by a variable number of integer-pel execution units. Additional details for this processor core are available in [19] and here we only highlight important details. Each execution unit uses a 64-bit wide word and a deep pipeline with 11 stages to achieve a high throughput. All the accesses to reference and macroblock memory are done through 64-bit wide data buses and the SAD (Sum of Absolute Differences) engine also operates on 64-bit data in parallel. The memory is organized in 64-bit words and typically all accesses are unaligned, since they refer to macroblocks that start in any position inside this word. By performing 64-bit read accesses in parallel to two memory blocks, the desired 64-bits inside the two words can be selected inside the vector alignment unit. The maximum number of integer-pel execution units is only limited by the available resources in the FPGA technology selected for implementation. Each execution unit has its own copy of the motion estimation algorithm to run and processes 64-bits of data in parallel with the rest of the execution units. The motion estimation algorithms executed by the hardware are typically fast motion estimation algorithms with a base search pattern such as a diamond or hexagon search. Each integer-pel execution unit receives an incremented address for the point memory so each of them can compute the SAD for a different search point corresponding to the same pattern. This means that the optimal number of integer-pel execution units for a diamond search pattern is four, and for the hexagon pattern six. In algorithms which combine different search patterns, such as UMH (Uneven Multi Hexagon), a compromise can be found to optimize the hardware and software components. This illustrates the idea that the hardware configuration and the software motion estimation algorithm can be optimized together to generate different processors depending on the software algorithm to be deployed. Two different hardware configurations have been selected to demonstrate the effects of logic scalability corresponding to one and six execution units and named ME1 and ME6 respectively.

Table 1 summarizes the complexity of each of the considered hardware configurations in the ZYNQ device. The netlists for each of the hardware configurations are processed and 100 critical paths are protected with in-situ detection logic. The total number of slices where the logic is mapped increases approximately 5% compared the original design and the detectors do not affect the critical path of the main logic. The in-situ detectors are designed to map the logic tightly in the slice so the number of additional slices required is moderate. The additional logic effects in circuit delay are low and do not necessarily slow the design. Although in-situ detection logic adds slower flip-flops these are not part of the functional circuit.

.

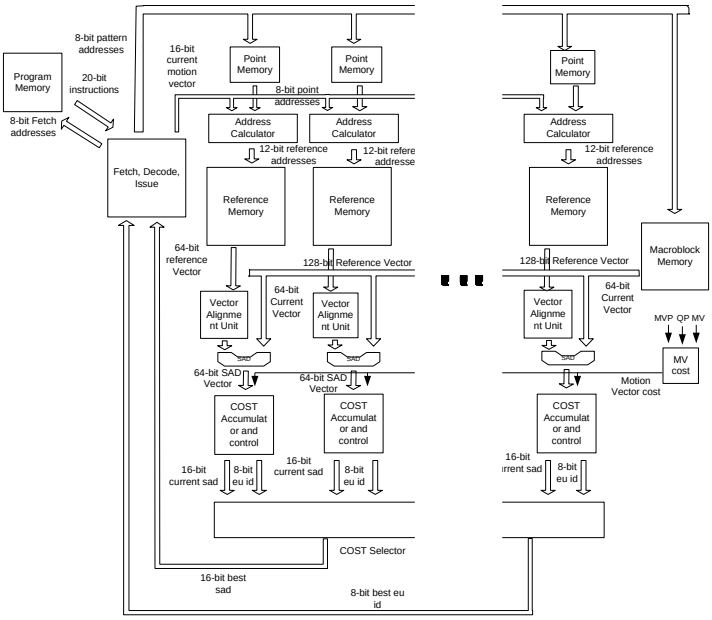

Fig. 1 Reconfigurable motion estimation processor

Table 1 Reference design resources original and elongate system (ZC7020 board/ ZYNQ 7020)

Resource	FF	LUT	BRAM36	BRAM18
ME1	1453	2302	8	1
ME6	3712	7300	28	3

IV. POWER ADAPTIVE SYSTEM ARCHITECTURE

The power adaptive controller is formed by two main IP blocks that correspond to the dynamic voltage scaler (DVS) and the dynamic frequency scaler (DFS) as shown in Fig. 2 . These two blocks can be instantiated independently and each one has its own AXI slave interface. This has certain advantages since it means that the technology can be used in different modes depending on the available features on the target board and device. The current prototype targets the ZC702 that implements the power manager bus (PMBUS) with access to all the power rails available for reading and writing as shown in Fig. 3. The presence of the PMBUS is required for the DVS unit to work. The DFS unit uses the MMCM (Mixed Mode Clock Managers) blocks to obtain different frequencies at run-time and it does not require other board level components. The following sections describe the features of the DVS and DFS units.

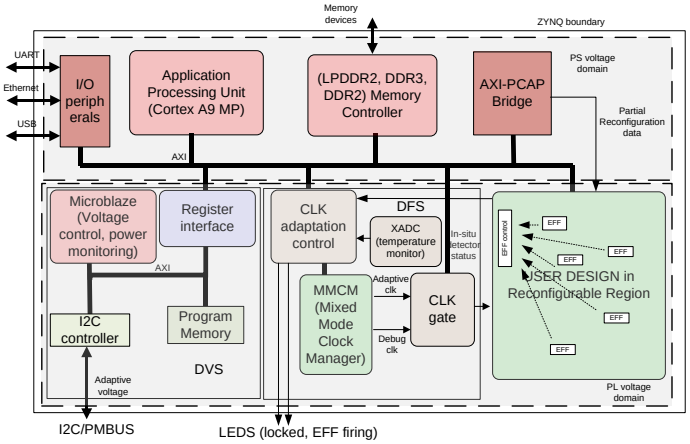

Fig. 2 Power adaptive system architecture

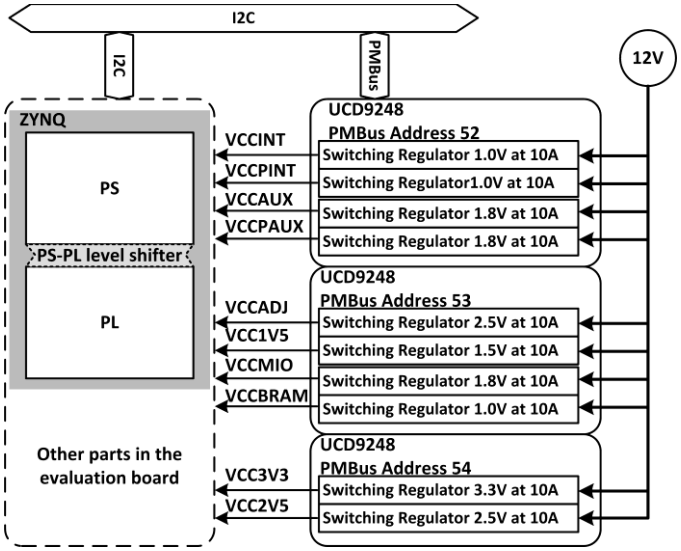

Fig. 3 Power rails in Zynq devices

A. *Dynamic Voltage Scaling*

As it can be seen in Fig. 2, the DVS unit has three main components which are a MicroBlaze processor (MB), a register file implemented using a Dual-Port RAM (DPRAM) and an IIC IP core. These components are connected to a local AXI bus. The DVS unit has full configuration and monitoring capabilities of the power rails connected to the power manager BUS. The DPRAM is used to receive the commands from the Cortex A9 processors. The commands control and record power, voltage values etc. The MB is responsible for the execution of the commands, communicating with the PMBUS via the IIC IP Core and writing the results to the DPRAM. In the ZC702 board the IIC IP Core is connected to the IIC Bus and accesses the PMBUS through a voltage shifter and an IIC 1-to-8 switch. The initialization code must set the 1-to-8 switch to the PMBUS channel before communication with the voltage regulators is possible. The initialization, configuration and monitoring code is written in C and compiled into an excutable *elf* file using the standard MB compiler. The *elf* is made part of the bitstream as a firmware and it is automatically stored in the program memory when the device is configured. The functionality of the DVS core is controlled with commands which are issued by Cortex A9 processor. A command has 32 bits and contains six parameters as it can be seen in Fig. 4. Action 1 and 0 are used to activate the core and signal task completion. The rest of the values indicate the type of operation (read/write), the target voltage regulator and the measurement type.

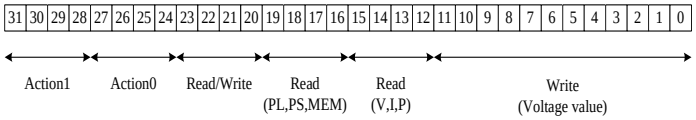

Fig. 4 Command parameters

B. *Dynamic Frequency Scaling*

The DFS unit receives the status information of the in-situ detectors embedded in the user design and uses that information to locate the maximum frequency that a particular voltage level can support automatically. More information about the internal structure of the in-situ detectors and how these are added to the original design are available in [19]. A ROM memory forms part of the DFS unit. This ROM contains configuration parameters for the Mixed Mode Clock Managers (MMCM) used to generate the clock for the user logic. The outputs obtained from this memory are written by the state machines part of the DFS unit using the dynamic reconfiguration port available in the MMCM blocks and new frequencies are generated at run-time. Once the MMCM has locked the clock is driven into the user logic. Once the frequency reaches a value that causes timing violations these are reported by the detectors and the state machine stops increasing the frequency until a different voltage is configured in the system. The DFS unit can also instantiate the system monitor IP block available in the FPGA device to monitor internal temperatures. This is advisable so that it is possible to react if internal core temperatures are excessive. Table 1 shows the complexity of the main blocks part of the DVS and DFS units.

Table 1 Logic resources (ZC7020 board)

Resource	FF	Utilization	LUT	Utilization
Microblaze processor (DVS)	972	0.9%	631	1.2%
I2C Controller (DVS)	343	0.3%	468	0.9%
Clock generation (DFS)	462	0.4%	683	1.2%
Total	1,777	1.6%	1,782	3.3%
Available	106,400		53,200	

C. *Dynamic Logic Scaling*

The Zynq family offers two ways to implement partial reconfiguration either using the ICAP interface controlled with logic implemented in the FPGA fabric and also the possibility

of using the PCAP interface controlled by the ARM processor. The PCAP interface is used during the standard boot sequence under Linux to load the full bitstream file to the PL fabric under processor control. The PCAP can also be used during application run-time to load the partial bit files stored in the file system as binary files. The PCAP interface features are equivalent to the ICAP operating at 100 MHz and offering a 32-bit wide bus with a maximum throughput of 400 Mbytes/second. The PCAP bridge is accessible through the AXI interconnect and a device Linux driver available developed by the company called *xdevcfg*. The driver uses a DMA mechanism to transfer the configuration bitstream to the PL at run-time. In this research we have selected the PCAP interface to save logic resources in the fabric and to simplify software development. Section VIII also uses the PCAP for power gating which is not possible with the ICAP. Configuration time scales largely linearly as the bitstream size grows with the number of reconfigurable frames. Fig. 5 shows the reconfigurable regions that have been defined to implement the motion estimation processors with one and six execution units. The utilization ratio is approximately 20% for the smaller configuration and 60% for the larger configuration in terms of logic resources. The amount of memory used in the region is 23% for the smaller configuration and 78% for the larger. Memory requirements are the limitation factor that determines the reconfigurable area size. The bitstream size for this region is 762752 Bytes or 44 frames. During synthesis of the motion estimation processor configurations used in the reconfigurable regions the synthesis tool automatically inserts a BUFG type buffer to the reset signal. This happens for the six core configuration and not for the single core configuration. An attribute is added to the source code to prevent BUFG to be added since these elements cannot be used in partial reconfigurable regions in the PlanAhead 14.4 version used in this work. Reconfiguration time for this region has been measured from *xdevcfg* driver call and the time it completes under the Linux Ubuntu OS is around 11 ms. Following Xilinx recommendations the motion estimation processors are held in reset during the configuration cycle to remove spurious transitions that could affect the processor side of the device or the static regions that contain the error detection logic.

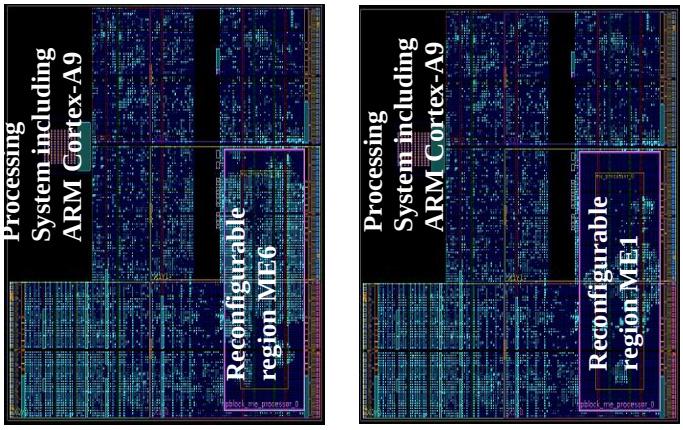

Fig. 5. Motion estimation (ME1 and ME6) layouts

V. Robustness Analysis of Detection Logic

The power adaptive architecture is designed to search for an optimal frequency for a given voltage value. In the test system the valid range of voltages extends from 0.7 V to 1 V. Frequencies are internally generated using the available MMCM (Mixed Mode Clock Manager) and its capability to reconfigure at run-time. The MMCM dynamic reconfiguration port enables the generation of changes in the clock frequency, phase and duty cycle on the fly. In this work only the clock frequency is varied. There are a number of registers in the MMCM that must be set correctly to control how frequencies are generated and a state machine is required to set the different registers correctly. The important registers in this work control the global clock divider that affects all the clock outputs in the MMCM (range 1 to 128), the individual clock divider for each of the clock outputs (range 1 to 128) and the clock multiplier that changes the voltage control oscillator (VCO) frequency in the MMCM (range 1 to 64).

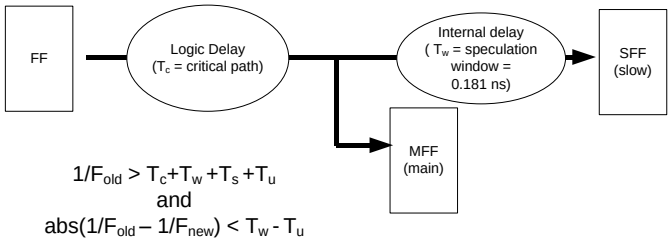

Fig. 6. Timing requirements

A problem exists if the instantaneous frequency change (in one single step) is such that both the slow flip-flop and the main flip-flop fail timing and the signal does not land inside the speculation window shown in Fig. 6. If this is the case the system will stop working. Fig. 6 shows the timing relations that must hold for the circuit to work. The first equation is the general timing equation and establishes that the clock period has to be large enough to accommodate the logic delay of the main circuit *(T_c)*, the speculation window *(Tw)*, the clock skew *(Ts)* and the clock uncertainty *(Tu)*. The second equation establishes that the change in the clock period between two successive frequencies has to be smaller than *Tw - Tu* since the clock uncertainty could potentially reduce the speculation window size. *Tw* is determined by the internal delays in the FPGA slice and calculated using the timing analysis tools to a value of 0.181 ns in the considered technology. *Tu* is also obtained from the post place&route timing report with a value of 0.035 ns. The tools originally introduce in [18] for Virtex-5 devices has been extended to use these values as input and calculate the clock frequency generation granularity required in the MMCM to obtain a safe circuit with the additional constraint of maintaining the VCO (Voltage Controlled Oscillator) part of the MMCM within the range allowed by the manufacturer for Zynq devices. The possible valid frequencies range from a minimum frequency of 22 MHz to a maximum frequency of 400 MHz. In total 448 different frequencies can be generated and the corresponding configuration values are stored in a read-only memory using device BRAMS. The CLK

generation logic reads these values from the BRAM and writes them to do MMCM in the correct sequence at run-time.

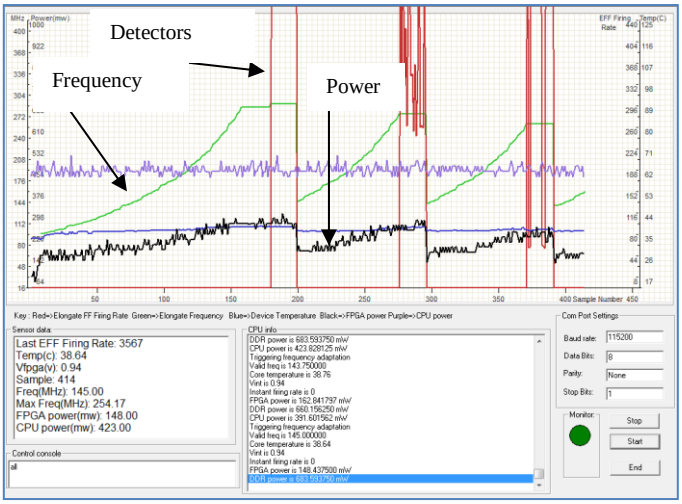

Fig. 7 Monitoring tool screen capture

VI. ME1 AND ME6 POWER ANALYSIS

The ARM processor executes a software daemon that reads the status of the in-situ detectors and writes commands to the DFS and DVS units. For these experiments the daemon monitors temperature, frequency, CPU power, FPGA power and detector state. This information is then sent through a USB-UART connection to an external monitoring tool used as a user interface. Fig. 7 shows a capture of the monitoring tool with different frequencies and voltages being generated and in-situ detector activity. Dynamic reconfiguration is controlled by the same daemon that initially loads the configuration under test. Partial reconfiguration can also be initiated from a shell with a command like "*cat me1.bin > /dev/xdevcfg*" . In the tests used to conduct the power analysis the software daemon starts by loading the ME1 or ME6 bitstreams as specified by the user. The FPGA core voltage is configured with commands written by the daemon to the DVS unit and then the DFS unit is configured by the daemon to search for the highest frequency possible for the given voltage. The DFS unit automatically detects this point and proceeds to inform the daemon. The daemon then restarts the process with a different voltage effectively sweeping the range of valid voltages. Notice that the user application runs in parallel activating the motion estimation processor continuously. This emulates how a real application such as a video codec will make use of a motion estimation accelerator implemented in hardware. The detectors embedded in the user application fire before timing violations affect the motion estimation data paths and control circuits. Fig. 8 shows the valid range of clock frequencies and voltages found by the daemon as it sweeps from nominal voltage of 1.0 V to a low voltage of 0.7 V. The figure shows that there is linear relation between frequency and voltage and, importantly, the detectors fire for ME1 at a frequency of 255 Mhz at 1 V (nominal voltage) which is much higher than the worst case frequency reported by the tools after timing analysis of 129 MHz. For the ME6 case the detectors fire at 166 MHz for the 1 V case which it also higher than the frequency reported by the tools at 90 MHz. Although additional factors such as operating conditions must be taken into account the difference between predicted and achieved performance suggests the existence of performance and power margins that could be exploited depending on workload by this AVS technique.

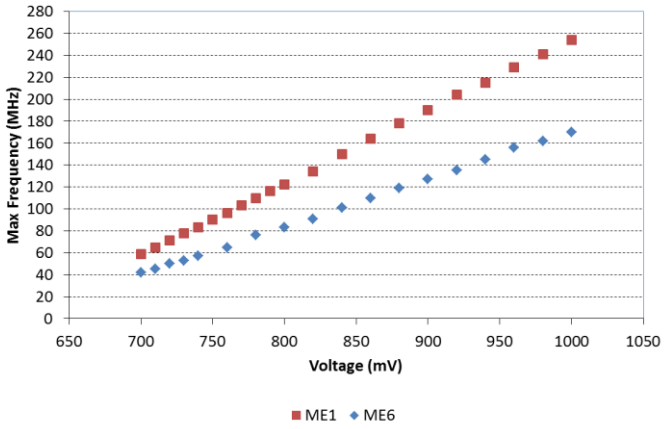

Fig. 8 Voltage and frequency.

In Fig. 9 the motion estimation processor is always active. The software daemon reduces the supply voltage via the PMBUS and the maximum frequency supported at each voltage level is auto-detected by the system. The obtained values define an optimal power profile that is compared with the nominal power profile. The nominal power line is based on a fixed nominal voltage of 1.0 V. The figure shows that for a given frequency value the optimal power line is up to 50% lower (from 285 mW to 137 mW at 50 Mhz) for the parallel ME6 processor and 63% lower for the serial ME1 processor (from 124 mW to 47 mW at 50 MHz). Fig. 10 shows the amount of power that is static for the motion estimation core. It can be seen that, for example, for the ME6 at 0.7 V static power is measured at 60 mW. This low voltage point corresponds to the minimum frequency of 50 MHz in Fig.8 in which total power is 137 mW so approximately 40% of power is static. For ME6 at 160 MHz total power is 493 mW in Fig. 9 and static power is 180 mW in Fig. 10 which represents a ratio of 36% static power. Overall static power is highly significant and has a slightly larger weight with low voltage configurations. Since static power is highly significant intuitively an idle core will be expensive from an energy point of view. This means that for example using a parallel core such as ME6 to complete the job fast and then idling until a new request is received could be a bad option compared with using ME1 and maintaining the core active during a longer time. Commercial FPGAs such as the Zynq devices considered in this work cannot power gate their fabric without losing the device configuration stored in SRAM memory so power gating states require a full reconfiguration cycle. Section 8 investigates the implications of power gating the device while Section 7 investigates the energy trade-offs of the proposed adaptive voltage and logic scaling technology.

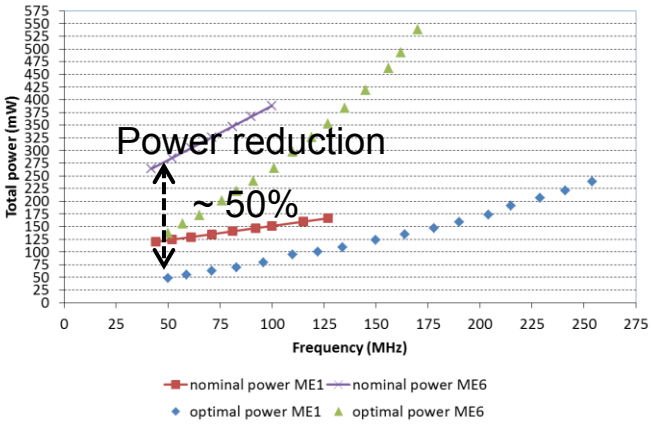

Fig. 9 Total power analysis.

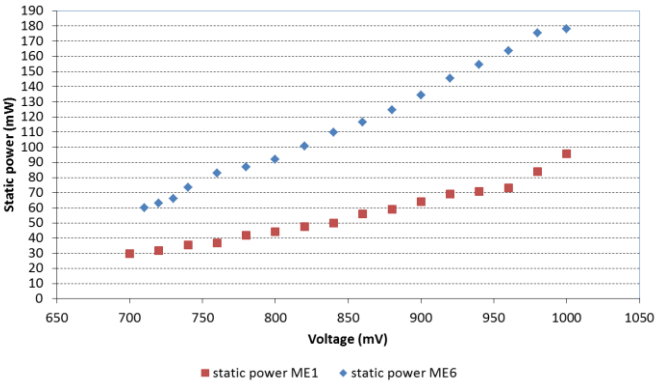

Fig. 10 Static power.

VII. ME1 AND ME6 ENERGY ANALYSIS

Fig. 11 shows the measurement strategy for the energy experiments. The total time T_{total} is fixed and determined by the time needed to obtain a predefined number of clock cycles of computation. T_{total} defines the time budget available to complete the task and it is used as a reference point. As voltage and frequency increase that amount of time the cores must be active to obtain the same number of cycles defined as T_{active} reduces. The time left from subtracting T_{total} and T_{active} is the idle time in which only static power remains. All the clocks driving the FPGA fabric are stopped by the processor at this point. For the energy analysis we consider two sample scenarios: the first scenario *A* assumes that the amount of time available to complete the task is large and defined such as the ME1 core at 50 MHz can just meet timing. This means that the ME1 at 50 MHz completes just in time while the rest of the configuration have idle time. We define this as the low performance scenario. The second scenario *B* assumes that the time to complete the task reduces by a factor of 4. We define this as the high performance scenario. This means that only ME1 configurations with a clock higher than 200 MHz can meet the timing. ME6 can execute the same algorithm approximately four times faster thanks to his additional execution units so all the ME6 configurations can meet timing. Notice that although ME6 has six executions units instead of one it is not six times faster due to overheads and the details fast motion estimation algorithm considered that starts checking the centre point before moving to a variable number of six-point hexagons that can be done in parallel in ME6. Checking the centre point takes the same amount of time with one or six execution units.

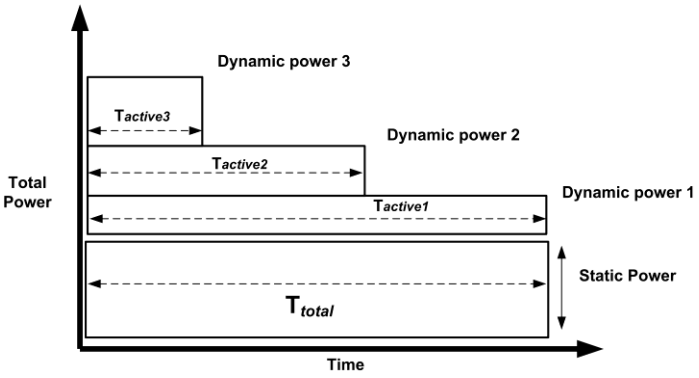

Fig. 11 Total energy calculations.

A. *Low performance scenario*

Fig. 12 shows the optimal energy analysis and compares it with the nominal energy obtained at a nominal voltage of 1 V. The nominal energy case remains constant for different frequencies since voltage is fixed at 1 V. Notice that T_{total} is fixed so static energy is constant and that T_{active} is changing because frequency is changing but the reduction of active time cancels with the increase in dynamic power so dynamic energy is also constant. The maximum performance point is the right most point in Fig. 12 in which the proposed approach approximately doubles the performance for the same amount of energy as the nominal case for both ME1 and ME6. The left most point of the figure represents the most energy efficient point in which the AVS points reduces energy by ~60% for both ME1 and ME6.

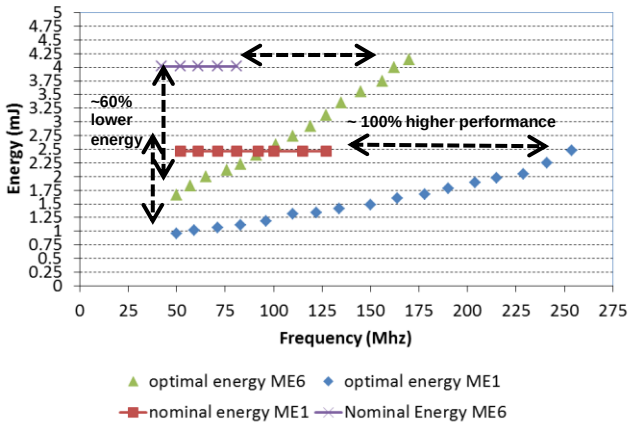

Fig. 12 Total energy calculations at low performance.

Overall, it is clear that all the voltage optimized configurations are much more energy efficient than the nominal cases. It is also clear than under this low performance requirement the slower configurations with ME1 are more energy efficiency than the parallel configurations with ME6. The optimal ME1 configuration is the slowest one at 50 MHz which requires only 0.7 V. The explanation is that the ME6 configuration completes the task earlier and then it incurs

higher static energy costs than ME1. Fig. 12 shows that ME6 at 50 MHz is more energy efficient than ME1 only when the frequency of operation of ME1 is higher than 175 MHz.

B. *High performance scenario*

The high performance scenario shown in Fig. 13 does not include the nominal case of ME1 because none of these configurations can meet timing. Only configurations with a frequency higher than 200 MHz are included for ME1. ME6 can meet timing with all its configurations and we can observe that overall energy requirements for the task decrease since the amount of time the core is idle is lower (lower static energy). In this case we can see that the slow ME6 configuration at 50 MHz is the most energy efficient. Comparing the two most energy efficient points for ME1 and ME6 the amount of energy reduces by approximately 25% with ME6.

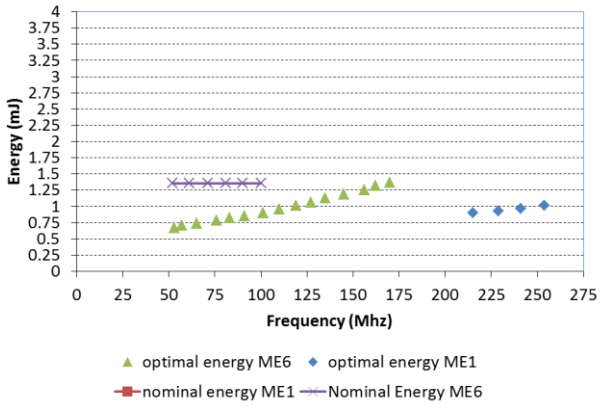

Fig. 13 Total energy calculations at high performance.

These experiments show that voltage and frequency scaling can be combined with logic scaling achieved with partial reconfiguration to obtain more energy efficiency execution. It is also clear than depending on the demands set on the accelerator different hardware configurations are more beneficial and it is not always the case that the more parallel hardware is the better choice due mainly to the additional static power that the parallel hardware needs. If the amount of time the device is in idle state is significant and the duty cycle of the core is low eliminating static power will be beneficial for both ME1 and ME6. This could be achieved power gating the reconfigurable logic followed by a full configuration cycle when the core is needed again. Section 8 investigates this additional power control knob. Power gating could be deployed when there is significant time slack even when the slowest hardware configuration is used.

VIII. Power Gating the FPGA Fabric

Power gating is possible because the PL can be shut down when it is in idle mode and this has been suggested by the manufacturer [20]. Unfortunately the current revision 1.0 of the ZC702 Zynq board shares some of the PL power rails with the PS so completely shutting down the PL affects the functionality of the PS which is not desirable. Taking into account this practical limitation of current hardware a number of experiments have been conducted to measure the overheads of PL shut down. The timing overhead required to turn off and turn on the FPGA has been measured to around to 3 *msec*. The PL must be fully reconfigured after turning on before it can be used again in the application. For this purpose, the PS can also utilise PCAP (Processor Configuration Access Port) interface to reconfigure the PL similarly as the partial reconfiguration used for logic scaling. To do this, first the PS activates and initialises the PCAP and then a DMA mechanism transfers the bitstream to the PL. The timing overhead for full reconfiguration of the 4,045,568 bytes present in the bitstream has been measured to about 34 *msec* in bare-metal and approximately 87 *msec* when the system is running the Linux OS. This configuration time is measured between the xdevcfg device driver call and the time when the FPGA DONE signal is asserted. This overhead consists of the PL initialization and bitstream transfer delays. The PCAP theoretical throughput is 400*MB/s* inferring than around 10 *msec* should be enough for a full configuration. However, the processor control method and AXI-PCAP bridge overheads reduce this theoretical throughput. Using the PL and the ICAP for partial-reconfiguration, the research in [21] and [22] have proposed partial reconfiguration management techniques which reach 382*MB/s* and 385*MB/s*, respectively. However, these techniques are not applicable in this case as they use the PL which is not available after it has been powered off. For low duty cycles shutting down the PL will improve the overall energy efficiency of the system as long as the system can remain in off state for a period estimated longer than 100 *msec*. In the next two sub-sections we investigate the energy effects of power gating the PL. To simplify the analysis we assume ideal power gating and voltage/frequency scaling so that neither of these two techniques involve additional costs in terms of power or time.

A. *Low performance scenario*

This experiment replicates the experiment of Fig. 12 but in this case it assumes that the core is power gated once the task completes. This means that the FPGA fabric is power gated from the moment defined by the end of T_{active} to T_{total}. We can see in Fig. 14 that the nominal energy scenarios that execute at 1 V benefit from running faster and as the clock rate increases energy reduces due to the power gating effect. The optimal energy scenarios (in which the voltage is adjusted to the minimum required to support the corresponding frequency) show a small change with frequency suggesting that the energy savings obtained by power gating (that increase with frequency) and the additional energy used due to the higher frequency and voltage during the active time cancel each other. The parallel configuration ME6 shows a slightly higher energy efficiency than ME1 at lower frequencies but this difference disappears at higher frequencies. The experiments indicates that the savings obtained by reducing voltage and frequency are slightly larger than those obtained by power gating and this means that ME6 is slightly more energy efficient at the low frequency. However this is not the case for ME1 which is largely constant.

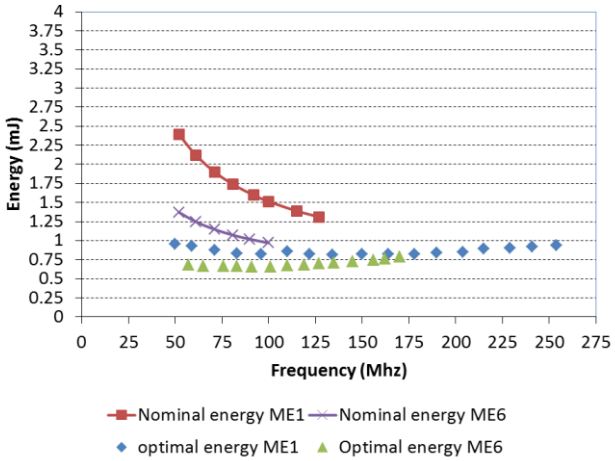

Fig. 14 Total energy calculations at low demand.

B. *High performance scenario*

The high performance scenario shows similar results in Fig. 15. The nominal configuration for ME1 is not present because it cannot reach the 200 MHz minimum frequency required. Otherwise the curves show a similar behaviour with the parallel core being more energy efficient with the slowest frequency.

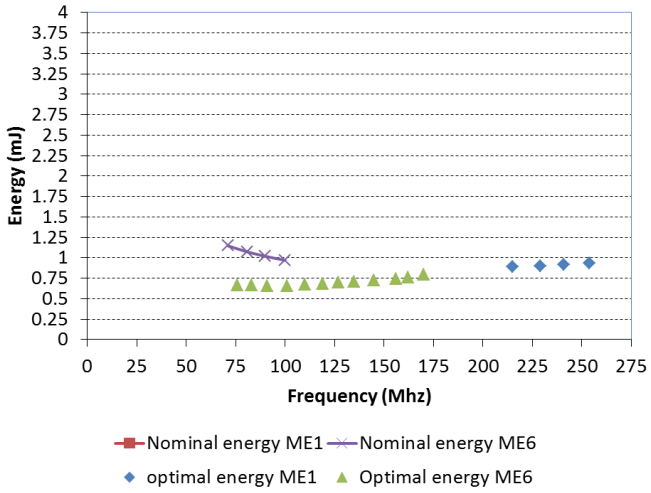

Fig. 15 Total energy calculations at high demand.

In the considered scenarios shown in Fig. 14 and Fig. 15 the power gating effect cancels most of the benefits obtained by running the core at low voltage and frequency. However the power gating costs are significantly higher than changing voltages and frequencies due to the costs of fully reconfiguring the device. Fig. 14 and Fig. 15 show that the nominal cases benefit from using higher frequencies and then power gating but for the optimal power configurations the advantage of running fast is not significant. On the other hand if the active cycle is very low and the core remains idle for longer than the estimated time of 100 *msec* then power gating can be used together with voltage and frequency scaling. In this case the core could be configured with a low voltage and frequency and then it could be power gated once the task has completed. This recommended strategy is summarized in Fig. 16 with T_{off} representing the power gated state.

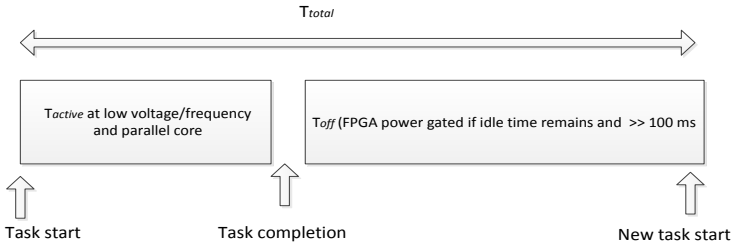

Fig. 16 Proposed combined energy optimization

IX. CONCLUSIONS

The considered Zynq devices offer a hybrid computing platform with a hardwire ARM dual-core Cortex A9 processor and a 28 nm FPGA fabric in different voltage domains. This configuration opens the possibility of having software daemons or the OS managing different power and performance configuration points in the FPGA fabric with voltage, frequency and logic scaling while power gating can also be included as a power control knob as long as the corresponding overheads are taking into account. The power adaptive architecture is designed to remove timing margins using in-situ timing detectors and includes two main components to control voltage and frequency: the DVS and DFS. The DVS exploits the presence of software programmable voltage regulators via the PMBUS protocol to change voltages at run time while the DFS uses the highly flexible mixed mode clock managers. The availability of the standard PMBUS means that a robust voltage control and monitoring loop can be created using only IP blocks without board modifications. The results show that the margins available make these chips a good platform for energy proportional computing with a reduction of energy of around 60% compared with a system running at the same frequency and nominal voltage. The energy studies also show that depending on the computational requirements placed on the FPGA fabric serial or parallel configurations can be more energy efficiency. A low computation scenario benefit from using a serial core at low voltage and low frequency while a high computation scenario benefits from using a parallel core at low voltage and low frequency. We also show that physical power gating of the FPGA fabric is an option but the overheads are significant. The experiments indicate that if power gating is available then the parallel configuration is the best choice with a low frequency/voltage followed by a power gated state once the task has completed and the idle time is sufficient to mask the costs of the full reconfiguration cycle. This option is slightly more energy efficient than running fast and power gating the device. In any case power gating can only be used in situations in which the FPGA fabric is not needed for a significant amount of time. Future work involves quantifying

with higher accuracy the thresholds in which the different techniques are more effective and validating the work with other acceleration cores that can be configured with different levels of complexity.

REFERENCES

1. I. Kuon, and J. Rose, "Measuring the gap between FPGAs and ASICs," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp.203–215, 2007.
2. S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM* vol.54, no.5, pp. 67-77, 2011.
3. Information available at http://www.arm.com/products/processors/technologies/biglittleprocessing.php
4. A. Rahman, S. Das, T. Tuan, and A. Rahut, "Heterogeneous routing architecture for low-power FPGA fabric," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 183–186, 2005.
5. J. Ryan, and B. Calhoun, "v," *IEEE Custom Integrated Circuits Conference* (CICC), pp.1–4, 2010.
6. F. Li, Y. Lin, and L. He, "Vdd programmability to reduce FPGA interconnect power," in *IEEE/ACM International Conference on Computer Aided Design* (ICCAD'04), pp.760-765, 2004.
7. F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics," in *Proceedings of 12th International Symposium on Field Programmable Gate Arrays*, (FPGA'04), pp.42-50, 2004.
8. A. Raham and V. Polavarapuv, "Evaluation of low-leakage design techniques for field programmable gate arrays," in *Proceedings of 12th International Symposium on Field Programmable Gate Arrays* (FPGA'04), pp.23–30, 2004.
9. J. Lamoureux, and S. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *International Conference on Computer Aided Design (ICCAD'03)* pp.701-708, 2003.
10. J. Lamoureux, and S. Wilton, "Clock-aware placement for FPGAs," in *Field Programmable Logic and Applications*, (FPL'07), pp.124 –131, 2007.
11. A. Gayasen, et al. "Reducing leakage energy in FPGAs using region constrained placement," in *Proceedings of the 12th International Symposium on Field Programmable Gate Arrays* (FPGA'04), pp.51-58, 2004.
12. Information available at http://www.xilinx.com/support/documentation/application_notes/xapp555-Lowering-Power-Using-VID-Bit.pdf
13. Information available at http://www.altera.com/literature/wp/wp-01200-power-performance-zettabyte-generation-10.pdf
14. C. Chow, L. Tsui, P. Leong, W. Luk, and S. Wilton, "Dynamic voltage scaling for commercial FPGAs," in *Proceedings of IEEE International Conference on Field-Programmable Technology*, pp.173-180, 2005.
15. N. Atukem, J. Nunez-Yanez, "Adaptive voltage scaling in a dynamically reconfigurable FPGA-based platform," *ACM Trans. Reconfigurable Technol. Syst*. Vol.5, no. 4, 2012.
16. S. Das, et al., "Razor II," *IEEE J. Solid-State Circuits*, pp. 32-48, Jan. 2009.
17. J. M. Levine, E. Stott, and P. Y. K. Cheung, "Dynamic voltage & frequency scaling with online slack measurement," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (FPGA'14), pp. 65-74, 2014.
18. J. Nunez-Yanez, "Adaptive voltage scaling with in-situ detectors in commercial FPGAs," Accepted for publication in *IEEE Transactions on Computers*.
19. J. L. Nunez-Yanez, et al. "Cogeneration of fast motion estimation processors and algorithms for advanced video coding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol.20, no.3, pp.437-448, 2012
20. Xilinx, "ZC702 evaluation board for the Zynq-7000 XC7Z020 all programmable SoC," *Xilinx, User Guide*, UG850 (v1.3), 2014.
21. K. Vipin, and S. A. Fahmy, "Zycap: Efficient partial reconfiguration management on the Xilinx ZYNQ," in *IEEE Embedded Systems Letters*, 2013.
22. A. Nabina and J. L. Nunez-Yanez, "Dynamic reconfiguration optimisation with streaming data decompression," in *Proceedings of the International Conference on Field Programmable Logic and Applications* (FPL), pp. 602 - 607, 2010.