Using Hardware Parallelism for Reducing Power Consumption in Video Streaming Applications

Karim M. A. Ali*‡, Rabie Ben Atitallah*‡, Nizar Fakhfakh§ and Jean-Luc Dekeyser†‡
‡DreamPal team, INRIA Lille-Nord-Europe, France
*LAMIH, University of Valenciennes, France, Email: {karim.ali, rabie.benatitallah}@univ-valenciennes.fr
†CRIStAL, University of Lille1, France, Email: jean-luc.dekeyser@univ-lille1.fr
§NAVYA Company, France, Email: nizar.fakhfakh@navya-technology.com

***Abstract*—Reconfigurable technology fits for real-time video streaming applications. It is considered as a promising solution due to the offered performance per watt compared to other technologies. Since FPGA evolved, several techniques at different design levels starting from the circuit-level up to the system-level were proposed to reduce the power consumption of the FPGA devices.**

In this paper, we present a flexible parallel hardware-based architecture in conjunction with frequency scaling as a technique for reducing power consumption in video streaming applications. In this work, we derived equations to ease the calculation for the level of parallelism and the maximum depth for the FIFOs used for clock domain crossing. Accordingly, a design space was formed including all the design alternatives for the application. The preferable design alternative is selected in aware of how much hardware it costs and what power reduction goal it can satisfy. We used Xilinx Zynq ZC706 evaluation board to implement two video streaming applications: Video downscaler (1:16) and AES encryption algorithm to verify our approach. The experimental results showed up to 19.6% power reduction for the video downscaler and up to 5.4% for the AES encryption.

***Index Terms*—FPGA, Reconfigurable architecture, Power consumption reduction, Parallel architecture, Video streaming applications, Zynq platform.**

I. Introduction

There is a growing demand for video streaming-based embedded systems in several industrial domains such as automotive and surveillance systems. These embedded systems require a total management of the used hardware resources, the delivered performance and the consumed power. Indeed, these systems are responsible for collision avoidance, driver assistance, target tracking, motion detection, path planning or for navigation among the others. In all these applications, parallel data acquisition and processing in real-time drives the need for high computation rates while carrying-out intensive signal processing.

Recently, the ITRS [12] and HiPEAC [6] roadmap promote that *power defines performance* and *power is the wall*. To overcome this obstacle, a new era, in which parallelism dominates the cutting-edge of embedded architecture appeared [10]. As a result, the whole computing domain is being forced to switch from a focus on performance-centric sequential computation to energy-efficient parallel computation. This switch is driven by the energy efficiency of using many slower parallel processors instead of a single high-speed one [6]. This has led to the design of Multiprocessor System-on-Chip (MPSoC) that integrates multiple cores or processors on a single die [19]. As an example of commercial platforms based on such architecture, we quote the NVIDIA Tegra [20] processor which integrates a quad-core ARM Cortex A15. Kalray Incorporation proposes a Multi-Purpose Processor Array (MPPA) that integrates up to 256 processors onto a single silicon chip through a high bandwidth Network on Chip [1]. Unfortunately, these trends are adequate only for a given range of applications particularly in systematic signal processing domain due to the general purpose processor used in these architectures. This was not enough for other applications such as video streaming where more performance and energy-efficient systems are required.

FPGA reconfigurable circuits have emerged in parallel as a privileged target platform to implement intensive signal processing applications. In fact, FPGAs have the benefits of being high speed and adaptable to the application constraints at a reduced performance per watt if compared to the General Purpose Processors (GPP) [9]. Furthermore, today FPGA technology enables us to implement massively parallel architectures due to the huge number of programmable logic fabrics available on the chip. In such architecture, with the management of the parallelism intrinsic in the application, the system designers will have several design choices such as sequential software, parallel software, hardware/software, parallel hardware or even dynamic hardware to implement their systems. The adequate choice will depend mainly on the application requirements in terms of performance and energy consumption.

In this work, we will invest in research and development of parallel hardware-based architecture for video streaming-based embedded systems guided with a power-aware design criteria. Mainly, we target reconfigurable technology to propose a flexible parallel system where the designers can adapt the parallelism level according to the available resources in order to control the overall system power consumption.

Furthermore, we will formulate the equations needed to calculate the level of parallelism and the depth of the used FIFOs. This work is considered as a first step towards a parallel and dynamically reconfigurable architectures. Such embedded systems will be able to adapt their functioning mode at run-time according to the available resources to provide deterministic timing guarantees, energy efficiency or a certain

Quality-of-Service.

The rest of the paper is organized as follows. Section II describes the current practices used in hardware design for reducing the power consumption. Section III describes the video processing system architecture. Section IV formulates the equations to calculate the level of parallelism and the depth of the used FIFOs. In section V, we will show the results obtained during our experiments and finally, section VI concludes the paper and draws our future works.

II. Related works

Several research efforts have been devoted to reduce the power consumption for reconfigurable technology at different design steps starting from the circuit-level up to the system-level.

At the circuit level, the number of transistors double with the reduction of the transistor size. Unfortunately, the static power consumption increases as well due to the diminish of the gate dielectric layer. The ITRS 2002 roadmap [16] mentioned that by the year 2005, the grand challenges were that the static power would increase to be equal to the dynamic power consumption. Consequently, the need for a gate with high K dielectric material would be a must for low power logic design. In 2010, Xilinx announced the arrive of the 28nm FPGA devices with up to 50% power reduction than the previous 40nm FPGA devices. The reason behind this reduction arose from the replacement of the Poly/SiON gate in the 40nm technology by the HKMG gate in the 28nm technology [21].

Three sources contribute to the CMOS node total power consumption. They are dynamic power (P_{dynamic}), leakage power (P_{leak}) and short circuit power (P_{SC}). P_{leak} is directly proportional to the supply voltage while P_{dynamic} is squarely proportional to it [14]. Therefore, scaling the input supply voltage will reduce the total consumed power. Dynamic Voltage Scaling unit (DVS) was suggested in [5] to scale the input voltage at run-time by configuring the power controller chip UCD92xx using the PMBus commands.

At the gate level, the clock network is responsible for delivering the clock signal to every single logic block. It divides the FPGA chip into a number of clock regions controlled by an enable signal. In [11], four clock gating techniques were considered. The results showed up to 50% reduction in the clock power with an overall power reduction reached to 6.2%-7.7%.

Some power reduction techniques can be applied during the design flow. For example, the authors in [18] added timing and placement constraints during the PAR phase for dynamic power reduction. While the authors in [13] showed that the selected synthesis and implementation options offered by the synthesis tool can affect the power consumption of the final implemented design.

At the architecture level, authors in [3] presented how splitting the data stream into parallel processing pipelines can reduce the power consumption in contrast to the traditional spatial pipeline processing technique. In our work, we will go further in this idea by considering video streaming applications of coloured 1080p60 HD video input stream. These applications will be processed using parallel hardware-based architecture in conjunction with frequency scaling. The chosen level of parallelism with a certain clock frequency scaling will offer several design choices leading to different trade-offs in terms of hardware cost and power consumption.

III. Video processing system architecture

Fig. 1 shows the video processing system architecture used in our research. It consists of VITA-2000 color image sensor [15] configured for high definition frame resolution 1080p60. It is coupled to Xilinx Zynq-7000 All Programmable SoC ZC706 evaluation kit [24] through an Avent IMAGEON FMC card [4]. The VITA-2000 is a CMOS image sensor [8] which captures the pixels in a monochrome nature of size 10-bit for each pixel. To generate an RGB color image, the Color Filter Array (CFA) is used to restore the other missing two colors based on the neighbouring pixels [22]. Some other filters such as (gamma, noise, edge enhancement, ...) can be also added to improve the quality of the input image. A Video Timing Controller (VTC) is connected for detecting/generating the video timing signals at both ends of the video processing channel. Normally the video stream is accompanied with video timing signals: (i) the vertical blanking (*vblank*) to mark the start of the frame, (ii) the horizontal blanking (*hblank*) to indicate the start of a line in the frame and (iii) the active video signal to show the periods of valid pixels within the frame (for simplicity they are gathered and named as *ctrl* signal in Fig. 1).

The proposed pixel distribution architecture in [2] is used to distribute the input pixel stream for parallel video processing. As depicted in Fig. 1, there are three processing channel one for each color component (red, green and blue). The role of the pixel distributor is to distribute the input pixel stream in the form of macro-blocks of size HxV, where H is the horizontal size and V is the vertical one. The pixel distributor stores the pixels in its internal buffer during the first (V-1) rows of the macro-block (i.e. *idle time*) while during the last row, it starts to distribute the pixels in the form of macro-blocks with the *valid* signal assigned high with each block (i.e. *distributing_time*) as shown in Fig. 2.

The parallel Processing Elements (PEs) are operating at clock frequency CLK2 which is slower than the one (CLK1) used by the other part of the system. Therefore, a FIFO is required to store the macro-blocks during their transfer from one clock domain to another. FIFO is typically implemented using a dual-port RAM where we have two input clock frequencies: *clk_wr* for writing and *clk_rd* for reading. The block named *FIFO_DeMux* has two roles: (i) to store the macro-blocks when they are transferred from clock domain CLK1 to clock domain CLK2. (ii) to distribute the macro-blocks among the processing elements (PE_1, PE_2, PE_3 ,....., PE_n). Multiplexers are used to gather the processed data from the parallel PEs; then they are later written to the pixel collector. When the pixel collector have enough pixels, it starts streaming them to the RGB-to-YCbCr422 block. RGB-to-YCbCr422 converts

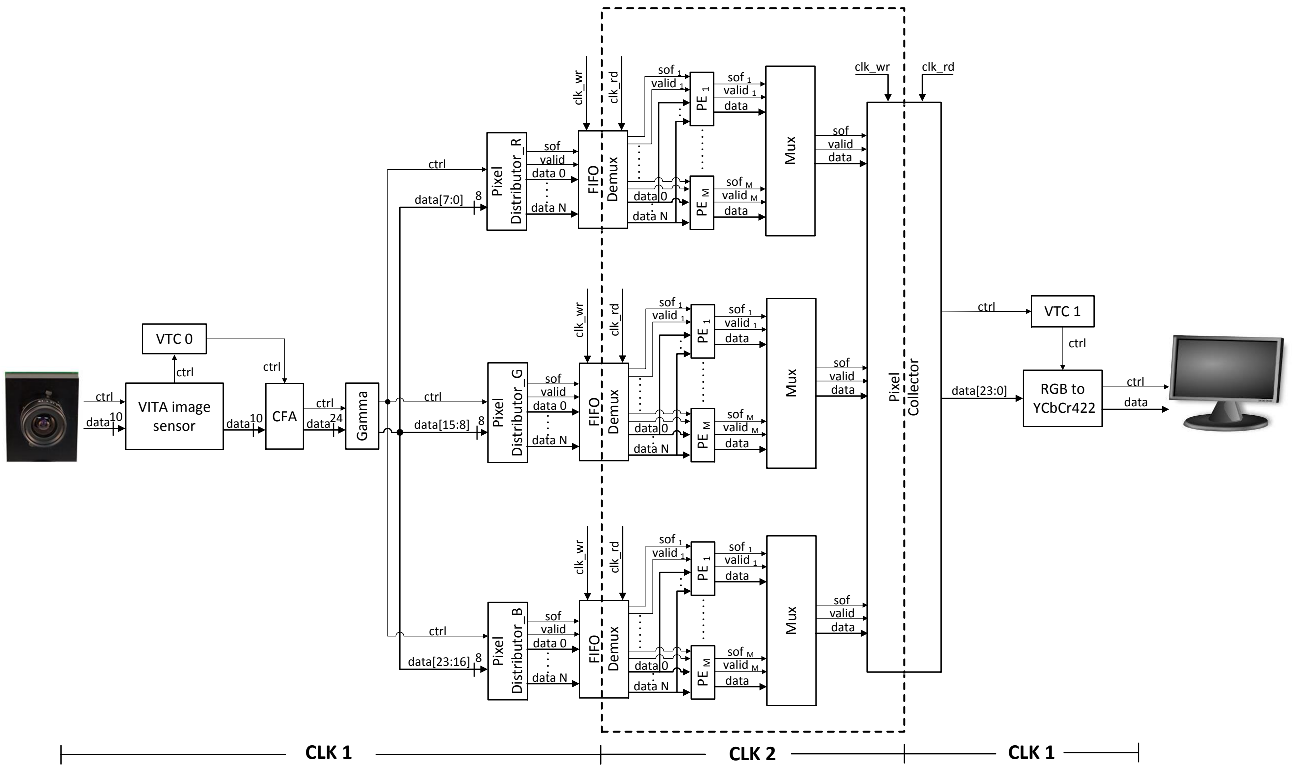

Fig. 1: The video processing architecture

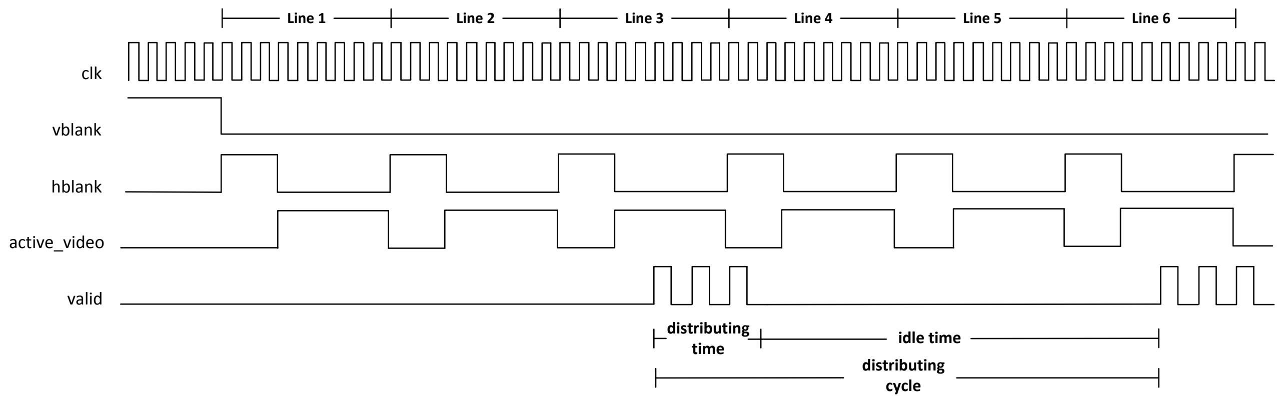

Fig. 2: The *valid* signal during the *distributing_cycle* for macro-blocks of horizontal size = 2 and vertical size = 3

the pixels to the YCbCr 4:2:2 format ready to be streamed to the HD monitor according to the HDMI specifications. The communication between the blocks is done through the signals named *valid* and *sof*. The *valid* signal is asserted high when there are available data at the output port, while the *sof* signal is flagged only if this data represent the start of the frame.

IV. Level of parallelism and FIFO depth Calculations

A. Level of parallelism

If the distributor sends data to the FIFO at a rate faster than the receiving side can handle, then the depth of the FIFO will grow indefinitely. As shown in Fig. 2, to bound the maximum depth of the FIFO, the macro-blocks produced during the *distributing_time* should be processed within the time of one *distributing_cycle* otherwise the maximum FIFO depth will grow up.

Taking this constraint into consideration, we can calculate the maximum computation delay (*max_comp_delay*) available for each processing element as following:

$$\begin{aligned} &\mathit{max\_comp\_delay} \\ &= \frac{\mathit{distributing\_cycle} * \mathit{N\_PE}}{\mathit{N\_mblocks} * \mathit{rd\_clk}} \\ &= \frac{V * \mathit{line\_period} * \mathit{wr\_clk} * \mathit{N\_PE}}{\mathit{N\_mblocks} * \mathit{rd\_clk}} \end{aligned} \tag{1}$$

Where V is the vertical dimension of the macro-block, *line_period* is the time required to stream one line of pixels in the horizontal direction, *distributing_cycle* is the time required to stream V lines of pixels, *N_PE* is the number of parallel processing elements, *N_mblocks* is the number of macro-blocks per *distributing_cycle*, *wr_clk* is the clock period for FIFO writing clock (CLK1) and *rd_clk* is the clock period for FIFO reading clock (CLK2).

From the same equation, by fixing the computation delay (*comp_delay*), then we can calculate the required level of parallelism (i.e. *N_PE*) to be:

$$\begin{aligned} &\text{Level of Parallelism} \\ &= \frac{\mathit{comp\_delay} * \mathit{N\_mblocks} * \mathit{rd\_clk}}{V * \mathit{line\_period} * \mathit{wr\_clk}} \\ &= \frac{\mathit{comp\_delay} * \mathit{N\_mblocks} * \text{CLK2}}{V * \mathit{line\_period} * \text{CLK1}} \end{aligned} \tag{2}$$

B. FIFO Depth

Since we can not simultaneously read and write at the same FIFO position; therefore, a constant value equal to 2 will be added to guarantee a minimum non-zero FIFO depth. At every clock *rd_clk*, one PE can be activated, so to calculate the maximum FIFO depth, we will have two cases according to how much slower is *rd_clk* than *wr_clk*.

*1) When not all PEs are yet activated by the end of the distributing_time (i.e. N_PE * rd_clk $>$ distributing_time):*

$$\begin{aligned} &\text{FIFO depth} \\ &= \mathit{N\_mblocks} - \mathit{N\_act\_PE} + 2 \\ &= \mathit{N\_mblocks} - \frac{\mathit{distributing\_time}}{\mathit{rd\_clk}} + 2 \\ &= \mathit{N\_mblocks} - \frac{\mathit{N\_pixels\_line} * \mathit{wr\_clk}}{\mathit{rd\_clk}} + 2 \end{aligned} \tag{3}$$

Where *N_mblocks* is the number of macro-blocks per *distributing_cycle*, *N_act_PE* is the number of active processing elements by the end of the *distributing_time* and *N_pixels_line* is the number of pixels per *line_period*.

*2) When all PEs are activated at least once during the distributing_time (i.e. N_PE * rd_clk $\leq$ distributing_time):*

$$\begin{aligned} &\text{FIFO depth} \\ &= \mathit{N\_mblocks} - \frac{\mathit{distributing\_time} * \mathit{N\_PE}}{\mathit{rd\_clk} * \mathit{comp\_delay}} + 2 \\ &= \mathit{N\_mblocks} - \frac{\mathit{N\_pixels\_line} * \mathit{wr\_clk} * \mathit{N\_PE}}{\mathit{rd\_clk} * \mathit{comp\_delay}} + 2 \end{aligned} \tag{4}$$

where *comp_delay* is the number of clock cycles required by PE to process one macro-block.

V. Experimental Results

In this section, we will discuss the implementation of two different applications: video downscaler (1:16) and AES encryption algorithm. By applying the equations obtained in the previous section, we were able to obtain different design alternatives varying in the depth of the FIFO and in the level of parallelism. For each design alternative, the power was estimated by Xilinx XPower Analyzer and measured using TI Fusion Digital Power Designer. The preferable design is then selected based on the percentage decrease in power compared to the hardware cost needed to implement this solution.

A. Design Points

For video downscaler (1:16) application, an HD frame of size 1920x1080 was scaled down to one sixteenth of its size to be 480x270. The application was synthesized using the parallel video processing architecture depicted in Fig 1 over the Zynq XC7Z045-FFG900 platform. The image sensor was configured for 60 frame/sec such that CLK1=148.5 MHz while CLK2 was a divisor of CLK1 according to the selected design point. In this application, the pixel distributor distributed the HD frame in the form of macro-blocks of size 4x4 while the PE is a video downscaler IP with a computation delay equal to 4 clock cycles.

For the AES encryption application, the HD frame was encrypted through a non-pipelined 128-bit AES encryption IP of computation delay equal to 12 clock cycles. We have chosen the Electronic Codebook cipher mode (ECB) since it is the simplest AES encryption mode [7]. The plaintext in the ECB mode is separately encrypted using the same 128-bit cipher key.

Table I listed a set of different design points. These points could be obtained using equation (2) by either varying the level of parallelism or the operating frequency CLK2. For both applications, the design point D1 is considered as the reference design point because it has the minimum required level of parallelism as well as it operates at the same clock frequency (i.e. CLK1 = CLK2 = 148.5 MHz).

B. Synthesis Results

The selected strategy for synthesis and implementation can affect the power consumption of the implemented design [13]. Taking this in consideration, it is worth to mention

Design point	Level of parallelism	CLK1 (MHz)	CLK2 (MHz)	FIFO depth
Video Downscaler (1:16) Application				
D1	1	148.5	148.5	0
D2	1	148.5	74.25	242
D3	1	148.5	37.125	362
D4	2	148.5	74.25	2
D5	2	148.5	37.125	242
D6	2	148.5	18.5625	362
D7	4	148.5	37.125	2
D8	4	148.5	18.5625	242
D9	4	148.5	9.28125	362
AES Encryption Application				
D1	3	148.5	148.5	0
D2	3	148.5	74.25	242
D3	3	148.5	37.125	362
D4	6	148.5	74.25	2
D5	6	148.5	37.125	242
D6	6	148.5	18.5625	362
D7	12	148.5	37.125	2
D8	12	148.5	18.5625	242
D9	12	148.5	9.28125	362

TABLE I: The design points for video downscler (1:16) and AES encryption applications

Design point	Occupied Slices	Slice Reg	Slice LUT	LUTRAM	BRAM18	BRAM36	DSP48E1
Video Downscaler (1:16) Application							
Base	8860	17273	17046	1168	29	114	16
D1	183	573	342	0	0	0	0
D2	1125	3537	2645	0	0	6	0
D3	1799	4989	4154	0	0	6	0
D4	613	1665	1034	264	0	0	0
D5	1042	3471	2830	0	0	6	0
D6	1508	4557	3767	0	0	6	0
D7	1004	2889	1797	264	0	0	0
D8	1612	5406	4026	0	0	6	0
D9	2165	6849	5005	0	0	6	0
AES Encryption Application							
Base	8873	17376	17027	1168	77	18	16
D1	5660	7518	14645	0	0	0	0
D2	6378	10482	17113	0	0	6	0
D3	7157	11934	18435	0	0	6	0
D4	11564	16635	30147	264	0	0	0
D5	12643	19164	32025	0	0	6	0
D6	12998	20610	33149	0	0	6	0
D7	21881	32451	59936	264	0	0	0
D8	22539	34986	62033	0	0	6	0
D9	23534	36429	62929	0	0	6	0

TABLE II: The Synthesis results for each design point for both video downscaler (1:16) and AES encryption

our selected options for synthesis and implementation during our experiments. PlanAhead 14.3 tool was used during the design process. For both applications, *PlanAhead Defaults* was used as a synthesis strategy while the implementation strategy was as following: (i) For video downscaler, we used *ISE Defaults* for all except for D8 and D9, it was *ParHighEffort* to meet the timing constraints. (ii) For AES encryption, we used *ParHighEffort* strategy except for D2, *MapTiming* was used to avoid timing constraints violation.

Table II shows the hardware cost for each design point. For each application, the row named *base* represents the required resources for implementing the basic blocks which exist in every single design point like VITA image sensor, VTC, CFA, GAMMA, pixel distributors or pixel collector. While the row named after each design point represents the needed resources for implementing that specified design. Therefore; the total resources used for realizing a single design point is equal to the sum of the *base* row and the row representing that design point. For example, the total design cost for D1 for video downscaler is: Occupied Slices = 9043, Slice Reg = 17846 and Slice LUT = 17388.

Design point	Video Downscaler (1:16) Application		AES Encryption Application	
	Measured Power (in mW)	Percentage power decrease (%)	Measured Power (in mW)	Percentage power decrease (%)
D1	1288.95	0	1038.36	0
D2	1212.94	5.9	1046.87	-0.82
D3	1116.93	13.35	1020.26	1.74
D4	1126.36	12.61	1023.54	1.43
D5	1111.96	13.73	1005.16	3.2
D6	1067.65	17.17	991.15	4.55
D7	1059.1	17.83	989.26	4.73
D8	1055.32	18.13	992.04	4.46
D9	1036.56	19.58	982.36	5.39

TABLE III: The measured power for different design points for video downscaler and AES encryption

From the synthesis results, we can get some observations that will later help us to understand how the power is consumed in the system. (i) It is obvious that the used BRAMs for video downscaler application was more than that used for AES application. This occurred because video downscaler needs to store more pixels before start streaming the video frames. (ii) The required level of parallelism for AES application is higher than that needed for video downscaler as mentioned in Table I. Consequently, the total used logic for AES application will be greater than that used for video downscaler.

C. Power Analysis

The power consumption for each design point was estimated using XPower Analyzer [23] to understand how the power was consumed by the different hardware resources. The power was also measured for verification through the power controller UCD90120A mounted on the evaluation board using Fusion Digital Power Designer [17]. During our experiments, we considered the slice register number as the cost function to implement a certain design choice. For sure, we can choose any other hardware resource as the cost or we can even have multiple factors in the cost function (for example, the summation of both register and LUT number as the cost function).

In Fig. 3, the estimated and measured power for video downscaler application was plotted against the number of slice register required for each design point. Experimentally, the power consumption decreased from 1.29 W for D1 to be 1.04 W at D9 with a percentage power reduction equal to 19.6%. According to the available register resources, the designer can

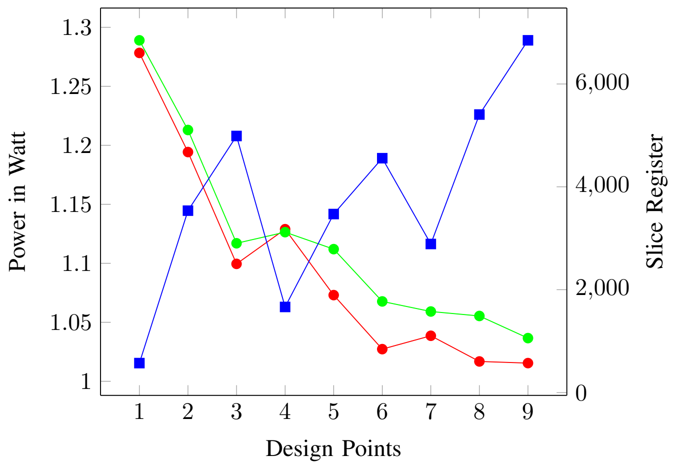

Fig. 3: The trade off between the estimated power —●—, the measured power —●— and the slice register cost —■— for each design point for video downscaler

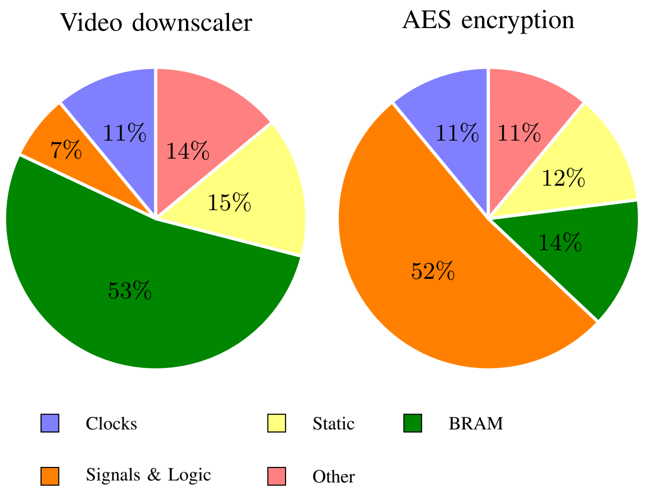

Fig. 5: The power consumed by different resources to implement the reference design D1 for both video downscaler and AES encryption

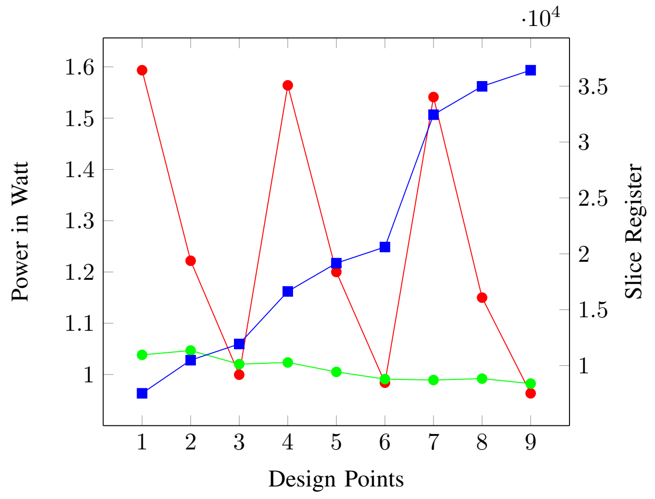

Fig. 4: The trade off between the estimated power —●—, the measured power —●— and the slice register cost —■— for each design point for AES encryption

select which design alternative to use and what percentage decrease in power to gain as shown in Table II and Table III. For example, the percentage power reduction for D7 was 17.8% at register cost = 2889 and for D6 was 17.1% at register cost = 4557 so D7 is always better than D6 since it achieved more power reduction at lower register cost. Also, we can consider D7 as a design choice better than other points like D8 or D9 because the percentage decrease in power between these points and D7 is not so significant (0.3% for D8 and 1.7% for D9) if compared to the percentage increase in the register cost (87% for D8 and 137% for D9).

For AES encryption application, Fig. 4 depicts the estimated and measured power versus the slice register cost for different design points. From the experimental measurements, the percentage decrease in power compared to that for the reference design was in the range of -0.8% up to 5.4% as reported in Table III. One reason for having such power increase at D2 is because that the used implementation strategy was changed to satisfy the timing constraints. It relies on the designer decision either to profit from the maximum possible power reduction of 5.4% at register cost = 36429 or to stay at some moderate hardware cost like at D6 with register cost = 20610 and power reduction of 4.5%.

Fig. 5 depicts the power estimations for the reference design D1 for both applications. When we look deep into how the power consumption is distributed between the different hardware resources; then, we can easily deduce that the big fraction came from the BRAM in the case of video downscaler while it came from the Signals & Logic for AES application. This can help us to explain why the maximum possible power reduction was large for video downscaler (19.6%) and it was small for AES encryprtion (5.4%): (i) For video downscaler, the large portion of the used BRAM were counted from the base design resources and the large fraction of the power was consumed by the BRAM as well. The total system power consumption was decreased when CLK2 was scaled over the BRAMs. Table I showed that scaling down CLK2 was accompanied by an increase in the level of parallelism as well as the depth of FIFOs and consequently the used hardware resources increased. But fortunately, the achieved power reduction was not too much affected by the power consumption arose from that added logic and thus we obtained a percentage decrease reached up to 19.6%. (ii) For the AES encryption application, the number of the used BRAM was not too much compared to the used logic, so the big portion of the consumed power was due to the used logic. Accordingly, as the level of parallelism increased, the used logic increased as well. Unfortunately, scaling CLK2 in this case was not enough to compensate the increase in the power consumption due to the added logic and to show in return a significant decrease in the total power consumption. Therefore, although D1, D4 and D7 operate at different clock frequencies equal to 148.5 MHz, 74.25 MHz and 37.125 MHz respectively, they reported a small percentage decrease in power reduction because of the added logic due to the increase in the level of parallelism.

It is notable that the percentage error between the estimated and measured power was small for the video downscaler while it was large for the AES encryption. This behaviour from XPower Analyzer can be explained in the highlight of Fig. 5. For video downscaler application, the power consumption was dominated by the BRAM while it was dominated by the Signals & Logic for AES application. If we suppose that XPower Analyzer can assume better activity rates for BRAMs than that assumed for Flip-Flops; therefore, the power estimations for video downscaler will be more close to the real measurements than that in the case of AES application.

D. Performance

To satisfy the timing condition of 60 frame/sec, the output video channel was constrained to clock frequency CLK1 = 148.5 MHz. We also limited the maximum depth of the FIFOs by processing the produced macro-blocks within their distributing cycle as mentioned before in section IV-B. According to these constraints, not every pair (level of parallelism, scaled frequency CLK2) could suite as a design point for our application. As a result for that, regardless what level of parallelism is applied or what value for CLK2 is chosen, the performance was kept constant at 60 frame/sec for all design points.

VI. Conclusion

In this paper, we presented a parallel hardware-based architecture in conjunction with frequency scaling to reduce power consumption for video streaming applications. Firstly, the equations required to calculate the level of parallelism and the depth of the FIFOs were derived. With the help of these equations, a design space including all the possible design alternatives was obtained. Two video processing applications: video downscaler (1:16) and AES encryption algorithm were implemented to verify our approach. The results for the measured power showed up to 19.6% power reduction for video downscaler and up to 5.4% for AES application. Finally, the designer is free to choose whichever design alternative to use based on the tradeoff between the hardware cost and the defined goal for power consumption.

As a future work, we will get benefit from this parallel architecture to introduce a dynamically reconfigurable embedded system. This system will be able to adjust its functioning mode at runtime to satisfy a certain power consumption goal according to the available hardware resources.

References

[1] MPPA MANYCORE, Multi-Purpose Processor Array. http://www.kalrayinc.com.

[2] K. M. A. Ali, R. Ben Atitallah, S. Hanafi, and J.-L. Dekeyser. A Generic Pixel Distribution Architecture for Parallel Video Processing. In *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, pages 1–8, Dec 2014.

[3] W. Atabany and P. Degenaar. Parallelism to reduce power consumption on FPGA spatiotemporal image processing. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1476–1479. IEEE, 2008.

[4] Avent. *FMC-IMAGEON EDK Reference Design Tutorial*, September 2012.

[5] A. Beldachi and J. Nunez-Yanez. Run-time power and performance scaling in 28 nm FPGAs. *Computers Digital Techniques, IET*, 8(4):178–186, July 2014.

[6] M. Duranton, D. Black-Schaffer, K. De Bosschere, and J. Maebe. *The HIPEAC vision for advanced computing in horizon 2020*. HiPEAC network of excellence, 2013.

[7] M. J. Dworkin. SP 800-38A 2001 Edition. Recommendation for Block Cipher Modes of Operation: Methods and Techniques. Technical report, Gaithersburg, MD, United States, 2001.

[8] E. Fossum. CMOS Image Sensors: electronic camera on a chip. In *Electron Devices Meeting, 1995. IEDM '95., International*, pages 17–25, Dec 1995.

[9] J. Fowers, G. Brown, P. Cooke, and G. Stitt. A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-window Applications. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '12, pages 47–56, New York, NY, USA, 2012. ACM.

[10] S. Fuller and L. Millett. Computing performance: Game over or next level? *Computer*, 44(1):31–38, Jan 2011.

[11] S. Huda, M. Mallick, and J. Anderson. Clock gating architectures for FPGA power reduction. In *Field Programmable Logic and Applications (FPL), 2009 International Conference on*, pages 112–118, Aug 2009.

[12] A. B. Kahng. The ITRS design technology and system drivers roadmap: Process and status. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 34:1–34:6, New York, NY, USA, 2013. ACM.

[13] D. Meidanis, K. Georgopoulos, and I. Papaefstathiou. FPGA power consumption measurements and estimations under different implementation parameters. In *Field-Programmable Technology (FPT), 2011 International Conference on*, pages 1–6, Dec 2011.

[14] W. Nebel and J. P. Mermet, editors. *Low Power Design in Deep Submicron Electronics*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[15] ON semiconductor. *VITA 2000 2.3 Megapixel 92 FPS Global Shutter CMOS Image Sensor*, June 2013.

[16] Semiconductor Industry Association. *International Technology Roadmap for Semiconductors (ITRS)*, 2002 Update.

[17] Texas Instruments. *Fusion Digital Power Designer GUI for Isolated Power Applications*, June 2014.

[18] L. Wang, M. French, A. Davoodi, and D. Agarwal. FPGA Dynamic Power Minimization Through Placement and Routing Constraints. *EURASIP J. Embedded Syst.*, 2006(1):7–7, Jan. 2006.

[19] W. Wolf, A. Jerraya, and G. Martin. Multiprocessor System-on-Chip (MPSoC) Technology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(10):1701–1713, Oct 2008.

[20] X. Wu and P. Gopalan. *NVIDIA Tegra 4 Family GPU Architecture*, Whitepaper v1.0, February, 2013.

[21] X. Wu and P. Gopalan. *Xilinx Next Generation 28 nm FPGA Technology Overview*, WP312 (v1.1.1) July 23, 2013.

[22] Xilinx. *LogiCORE IP Color Filter Array Interpolation v3.0*, December 2010.

[23] Xilinx. *Power Methodology Guide*, April 2013.

[24] Xilinx. *ZC706 Evaluation Board for the Zynq-7000 XC7Z045 All Programmable SoC User Guide*, July 2013.