Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 3405 – 3418
© TÜBİTAK
doi:10.3906/elk-1408-84

Design of low power system on programmable chip for video zoom-in processing

Lamjed TOUIL*, **Lilia KECHICHE**, **Bouraoui OUNI**
Electronic and Microelectronic Laboratory, Faculty of Sciences at Monastir, University of Monastir, Monastir, Tunisia

Received: 13.08.2014 • **Accepted/Published Online:** 09.04.2015 • **Final Version:** 20.06.2016

Abstract: Today, power consumption presents a critical issue when designing embedded systems. It has become a key factor in a product's success when being marketed, especially for mobile systems. In this paper, we have developed a typical design for video zoom-in processing with low power consumption. The proposed design has been implemented on the Xilinx Virtex-5 with the integration of different important resources like embedded processor, embedded memory, MPMC, DDR, and various interfaces. The design results show a significant gain in power at the simulation level as well as the physical one. In addition, the proposed design provides an important gain in power without degrading other parameters such as frequency, area, and others.

Key words: Video processing, power optimization, SOPC, video zoom-in processing

1. Introduction

Today electronic devices, such as mobile phones, are being used extensively to provide users with various multimedia applications like video streaming and encoding. These devices face fundamental challenges related to energy constraints as a result of their modest sizes, weights, and energy supply. One of the principle factors that aggravate energy problems is battery technology, which is not progressing at the same rate as the power consumed by different hardware applications. In addition, for digital circuits, video processing applications are known to be the biggest energy consumers as shown in Table 1 [1]. This huge consumption is a result of intensive use of bandwidth and memories. This fact pushed designers to search for new methods to grant low power consumption for video processing applications. Many solutions have been proposed as an attempt to minimize both static and dynamic energy for video applications [2–4] and, although having reached a remarkable percentage, the obtained results remain insufficient and need more improvements. Therefore, power consumption should be added as a major constraint to be optimized. In this paper, we have focused on the development of a new design method that aims to build a low power SOPC for one of the widely used video processing applications, video zoom-in processing. In the literature many works have proposed designs for video zoom-in architecture [5–7]. In [5] the authors presented an end-to-end real-time system to interactively zoom and pan into high-resolution panoramic videos. In [6] the authors have presented a spatial-random-access-enabled video compression that encodes the content such that arbitrary rolls corresponding to different zoom factors can be extracted from the compressed bit stream. In [7], the authors presented an end-to-end real-time system with interactive zoom and panning. None of the presented architectures deal with power consumption minimization. In this work we propose a general purpose architecture for video zoom-in with a minimization of dynamic power.

*Correspondence: lamjedtl@yahoo.fr

Table 1. Power consumption for different parts of the mobile phone.

Technology		Action	Power [mW]	Energy [J]
Wireless data	Bluetooth	BT off	12	
		BT on	915	
		BT connected and idle	67	
		BT sending	432	
	Wi-Fi IEEE 802.11 (infrastructure mode)	In connection	868	8.2
		In disconnection	135	0.4
		Idle	58	
		Idle in power save mode	26	
		Downloading @ 4.5 Mbps	1450	
	Wi-Fi IEEE 802.11 (ad hoc)	Sending @ 700 kb/s	1629	
		Receiving	1375	
		Idle	979	
Miscellaneous	CPU usage	2%	55	
		25%	310	
		50%	462	
		75%	561	
		100%	612	
	Mobile TV	Watching mobile TV with DVB-H	790	
		Black background 100%	259.65	
		White background 60%	254.16	
		White background 100%	527.05	
		Screen saver mode	13.86	
	Memory	Saving 1 Mb on drive C	587.7	56.2
		Saving 1 Mb on drive E	612.8	36.4
		Saving 1 Mb on drive D	560.0	2.2
Mobile	Voice	Making a voice call (3G)	1265.7	
		Receiving a voice call (3G)	1224.3	
		Idle (3G)	25.3	
		Making a voice call (2G)	683.6	
		Receiving a voice call (2G)	6.12.7	
	Video	Making a video call (3G)	2210	
		Receiving a video call (3G)	2145	
		300 bytes length (2G)		3.15
		300 bytes length (3G)		4.22

2. Interactive video zoom-in architecture

A digital zoom is defined as a transformation subjected to a discrete image I1 at a defined precision (p), which produces a discrete image I2 at a better precision.

Zoom-in is needed to be applied when someone needs to focus on a specific detail in the video or get information that is not clear with the current image size. It allows one to take a deeper look at a particular detail by zooming in on a chosen region called the region of interest. This functionality can be referred to as the interactive region of interest (IRoI) [8]. For example, in Figure 1, we apply a zoom-in to the video (Figure 1a) to verify the nature of the fish taken by the bird (Figure 1b).

One of the concerns of image zoom-in is the preservation of the image quality as much as possible. In reality, many zoom-in methods exist in the literature, such as nearest neighbor, bilinear, and cubic zoom [8–11]. In the nearest neighbor algorithm, the nearest pixel of the original image is sampled. It is considered to be a very simple and fast method, and in addition, it does not add any artificial data to the output. The major drawback of this method is the block artifact problem, which is visually noticeable. In the bilinear method [8], the output pixel changes its value linearly according to the sampling position (Figure 2), and the pixels surrounding the needed data are used to make an approximation of the missing value. This method produces a smoother output at the expense of high computational complexity.

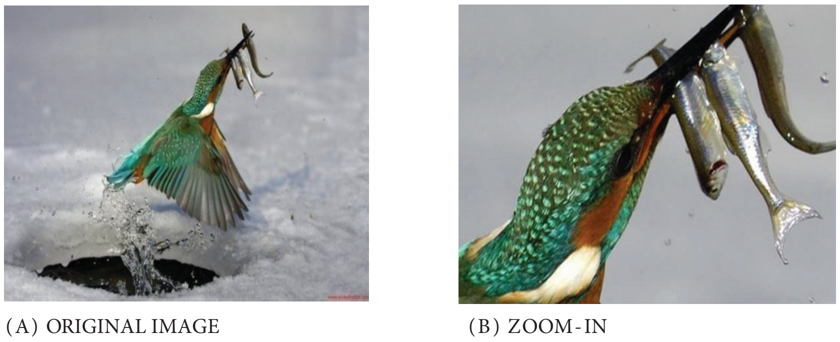


Figure 1. Display effect of zoom-in function: a) original image, b) zoom-in.

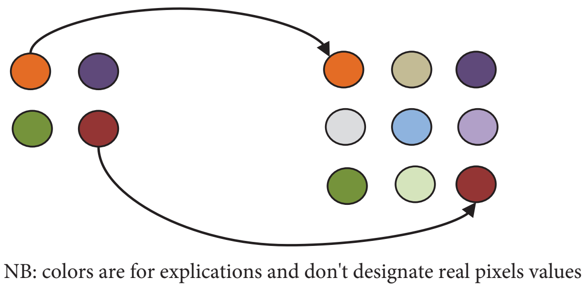

NB: colors are for explications and don't designate real pixels values

Figure 2. Bilinear interpolation.

In this work, we propose an interactive video architecture with a zoom function. This kind of architecture allows the user to quickly and easily zoom in on any video detail. In addition, it also allows him to apply other interesting effects such as adding audio, titles, and transitions.

Figure 3 gives the top level description of a general interactive zoom-in architecture. The zoom function is implemented using a bilinear algorithm [8].

The proposed hardware system is an embedded system based on the MicroBlaze 32-bit microprocessor. The high-level hardware design components are shown in Figure. 3.

The design includes different resources: the MicroBlaze processor core, video DAC, multiport memory controller (MPMC), XPS thin film transistor (TFT) controller, and some interfaces like a USB controller, system ACE compact flash, and XPS IIC. In addition to these standard components, we added user IPs like the zoom-in and clock controller IP cores.

The MicroBlaze processor is used to set up the system and configure the analog to digital conversion (ADC) board. This board is used to capture the phase alternate line (PAL) signal and digitize it into CCIR 601/656 format.

The ADV7183 is one of the main components of the ADC board. It is an integrated video decoder that converts a standard analog baseband television signal compatible with NTSC or PAL standards into 4:2:2 component video data. MicroBlaze uses the I2C bus to configure the ADV (http://www.digilentinc.com).

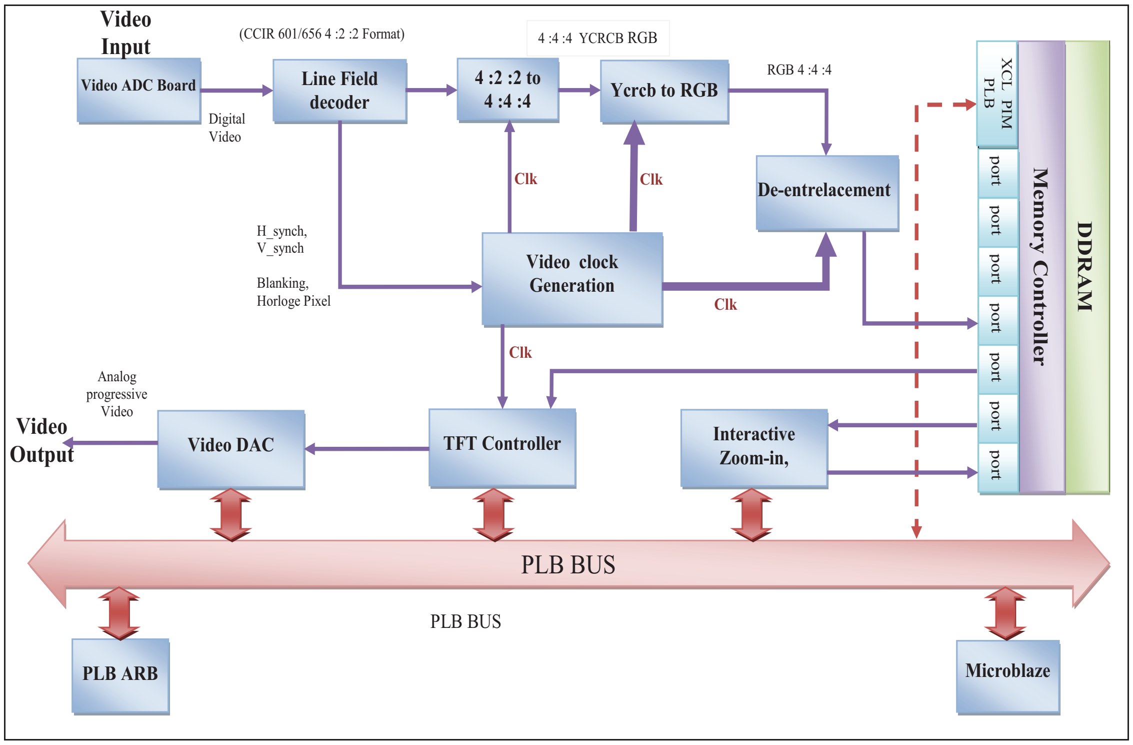


Figure 3. Video zoom-in architecture.

The particular format used by the ADV7183 is compatible with the ITU-601/656 video standard. It is a modified YUV format that uses the YCrCb color space. The purpose of using this color space is to separate the data that arrive over the pixel bus in YUV format. In our design we need to convert the image data into RGB. The YUV format consists of three basic components: Y, which represents the luminance or brightness of a pixel; U, which represents the color difference between blue and yellow; and V, which represents the color difference between red and cyan. Video decoding operates in a clock domain of 27 MHz. However, the custom hardware runs at 100 MHz. This domain crossing is handled by the use of an asynchronous double line buffer. The components included for decoding are 'line decoder', '4:2:2 to 4:4:4 format conversion', 'YCrCb to RGB color space conversion', 'timing generation', and 'deinterlacing'. These modules perform the task of converting the data from composite format to RGB and also keep track of the VSynch and HSynch to determine a new frame and a new line.

The acquisition and display parts of the architecture were presented in [12] and improved in the ECE 532 design project for real-time color replacement in 2009 (http://www.eecg.toronto.edu/∼pc/courses/432/2009/projects/colorreplacement.pdf).

The input image coming from the camera is first stored in the DDRAM memory through the video-in module after some standard video conversions (YCrCb to RGB, 4:2:2 to 4:4:4, etc.). The zoom-in algorithm is completely based on PAL video signal format.

The central component in our design is the MPMC. The processor local bus (PLB) TFT controller is a hardware display controller, which is capable of showing up to 256 colors. The PLB TFT controller accesses the memory through a VFBS MPMC port (www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf). The MPMC is relatively new in the Xilinx IP library, starting with version 10.1. It allows multiple buses to be connected to the same piece of memory through different ports, and the ports are able to access the memory in parallel. We used 5 ports from the 8 supported by the MPMC (v6.03.a). The DDR memory runs at 133 MHz, and therefore the MPMC can handle all clock domain crossing issues between system components. To design the system we have used high level tools, which are:

- Xilinx Embedded Development Kit (EDK), which is an integrated development environment to design embedded processing systems and that contains the Xilinx Platform Studio (XPS), which is a graphical IDE that supports the development of hardware platforms.
- We have used the Memory Interface Generator (MIG), which is a software tool used to generate memory controllers and to simplify the memory interface design process.

The use of the Xilinx EDK tool, the Xilinx Integration Software Environment, and the MIG makes the integration of IP cores easier and more flexible.

Figure 4 shows the hardware implementation of the zoom-in function. The FIFO located at the beginning of the module aims to facilitate the synchronization. Each two pixels of the same line contribute to the creation of a new pixel, and each two lines contribute to the creation of a new line. The system is controlled by a finite state machine. The same principle is applied to the chrominance data.

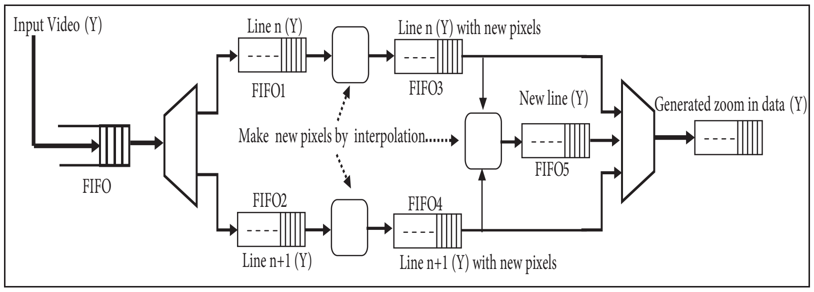

Input Video (Y)
FIFO
Line n (Y)
FIFO1
Line n (Y) with new pixels
FIFO3
Make new pixels by interpolation
New line (Y)
FIFO5
Generated zoom in data (Y)
FIFO2
Line n+1 (Y)
FIFO4
Line n+1 (Y) with new pixels

Figure 4. Hardware architecture for video zoom-in.

3. Methodology and proposed approach

We used platform-based design (PBD) methodology combined with a bus-based approach (BBA). The PBD uses an existing base of components and architectures to decrease design time and uses IP blocks to build system architectures. The PBD approach separates the HW/SW partition into behavior approach, architecture approach, and mapping. This method supports IP reuse and derivative design.

For the BBA methodology, components are designed around an on-chip bus architecture. This approach needs the interface of the concerned bus protocol to be integrated on the entire system, which leads to the creation of a logic of communication. The bus behaves as a generic component responsible for interfacing blocks and as a bridge if there is more than one bus in the system architecture. Combining both PBD and BBA approaches makes the platform more flexible for the addition of external blocks. Hence, it is possible to add other blocks without worrying about their interfacing and integration.

In this paper we propose a low power video zoom-in architecture as shown in Figure 5. This architecture aims to minimize the consumed power as it has become a big concern, especially for mobile devices that have limited energy supplies. This new approach is applied to the YCrCb to RGB block, which is a very commonly used block and a very necessary module to work with in zoom-in architecture; therefore, we have chosen to minimize its dynamic power. This leads to a significant gain in it for the whole architecture, as shown in Section 4.

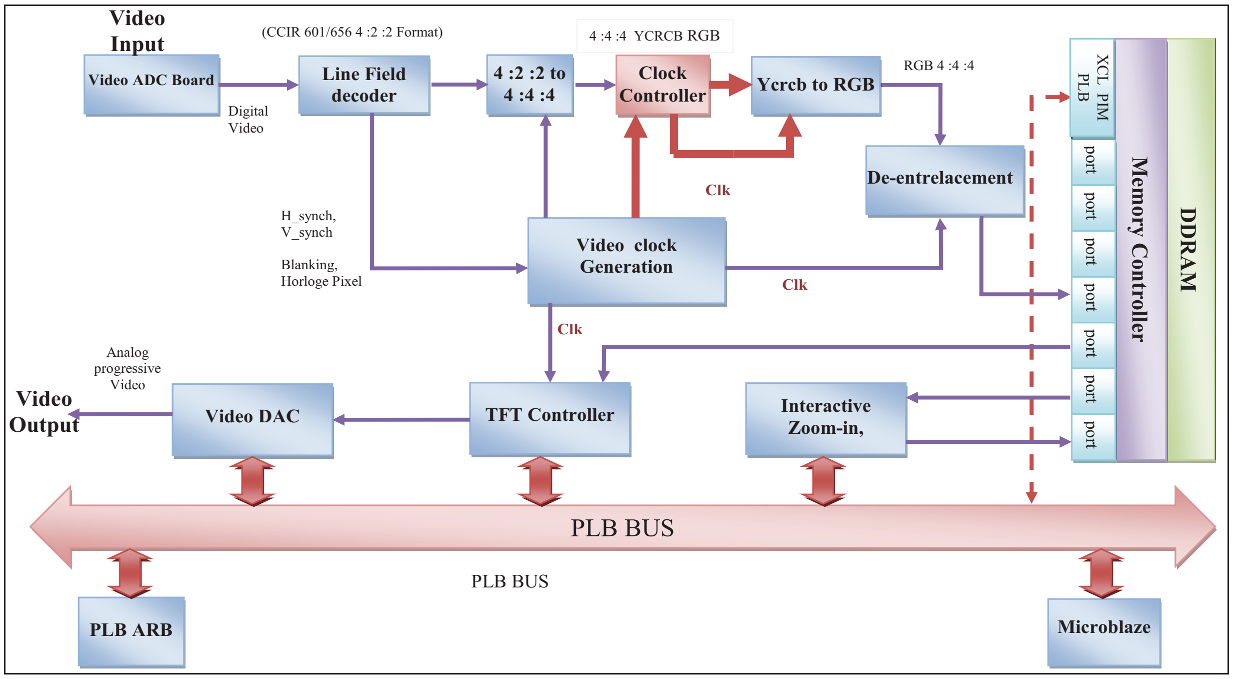


Figure 5. Video zoom-in architecture.

When studied separately, one of the main sources that increase the dynamic power of the YCrCb to RGB block is its clock. At each clock edge, this block receives input data from other components and performs its task to produce output data. The input data received are pixels coming from a single video frame that have been subjected to anterior treatments. If the current input is similar to the previous input, each block produces the same output data.

One may wonder why one would let the YCrCb to RGB block run to produce the same outputs. That would ultimately lead to a considerable waste of dynamic power. Thus, it is more suitable to disable the YCrCb to RGB block if the current input is equal to its previous input. One solution for disabling this block is preventing operation of its clock. Thus, in this paper, we introduce a new design method that disables the clock of the YCrCb to RGB block when its current input data are similar to the previous input data. Our method is based on adding a new block that we have called the 'clock controller' to the YCrCb to RGB block as shown in Figure 5. In the new video zoom-in architecture, the clock of the YCrCb to RGB block is handled by the proposed block instead of by the system clock.

3.1. YCrCb to RGB block

A color space is a method by which we can visualize the color. Humans may specify a color by its attributes of brightness, hue, and colorfulness. A color space is a mathematical representation of a set of colors [13]. The most used color models are:

1. RGB (in computer graphics);

2. YIQ, YUV, or YCrCb (in video systems);

3. CMYK (in color printing).

In the YCbCr space, Y presents the luma component; Cb and Cr present the chroma components. The YCbCr color space is a modified version of the YUV color space. Y is defined to have a range of 16–235; Cb and Cr are defined to have a range of 16–240 [14]. The reason for using YCrCb signals is that the human eye is less sensitive to chrominance than luminance. Compression algorithms can take advantage of this phenomenon and subsample the values of Cb and Cr without significant visual degradation of the original color signal. The RGB color space is generally adapted in computer graphics and it is one of most widely used color spaces for processing and storing digital image data. R, G, and B are three primary additive colors. The basic equations to convert between YCbCr and RGB are:

$$R = 1.164(Y - 16) + 1.596(Cr - 128) \tag{1}$$

$$G = 1.164(Y - 16) - 0.813(Cr - 128) - 0.391(Cb - 128) \tag{2}$$

$$B = 1.164(Y - 16) + 2.018(Cb - 128) \tag{3}$$

Based on the values of Y, Cr, and Cb, at each clock edge, the YCrCb to RGB block provides the values of R, G, and B.

3.2. Clock controller

The main aim of the clock controller block is the following:

At each clock edge:

(i) It verifies if the input data of the YCrCb to RGB block (values of Y, Cr, Cb) at the current clock period T_n are similar to the input data at the previous clock periods T_{n-1}.

(ii) If (1) is verified, the clock controller disables the clock signal of the YCrCb to RGB block.

The clock controller presents the main idea of this work. At each clock event, the YCrCb to RGB block receives a pixel (in the form of a binary code) to perform the predefined treatment and give the desired result. Thus, for a given image, if several successive pixels are similar, the input data of the YCrCb to RGB block will remain unchangeable and the output will be so. In reality, most images contain several similar successive pixels at different rates. For example, in Figure 6, all pixels inside the dashed area are similar. Therefore, during the processing time of the ith image, the input data of the YCrCb to RGB block will be unchangeable during a large part of the image processing time. Hence, (1) and (2) are also verified during a large part of the image processing time. As a result, according to our design, the YCrCb to RGB block is almost disabled during image processing time. This leads to important gain in dynamic power consumed by the global architecture.

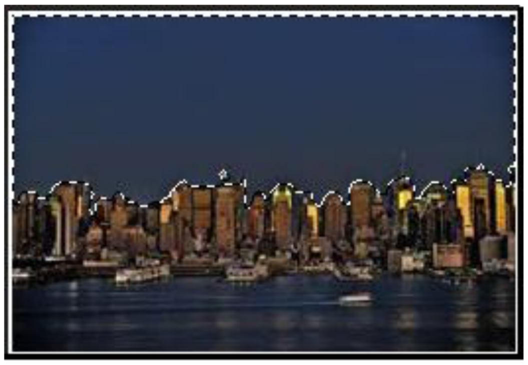


Figure 6. Example of an image with similar successive pixels.

3.3. Clock controller design

As mentioned in Section 3.2, the first aim of the clock controller is to verify if the current input data of the YCrCb to RGB block are similar to its previous input data. Thus, we need to memorize the previous input data and more precisely the values of Y, Cr, and Cb attributes. To meet this aim, we have proposed three registers for each datum as show in Figure 7. To compare current input data to previous data, we have added three XOR logic gates and, finally, to prevent the clock we have added an AND logic gate.

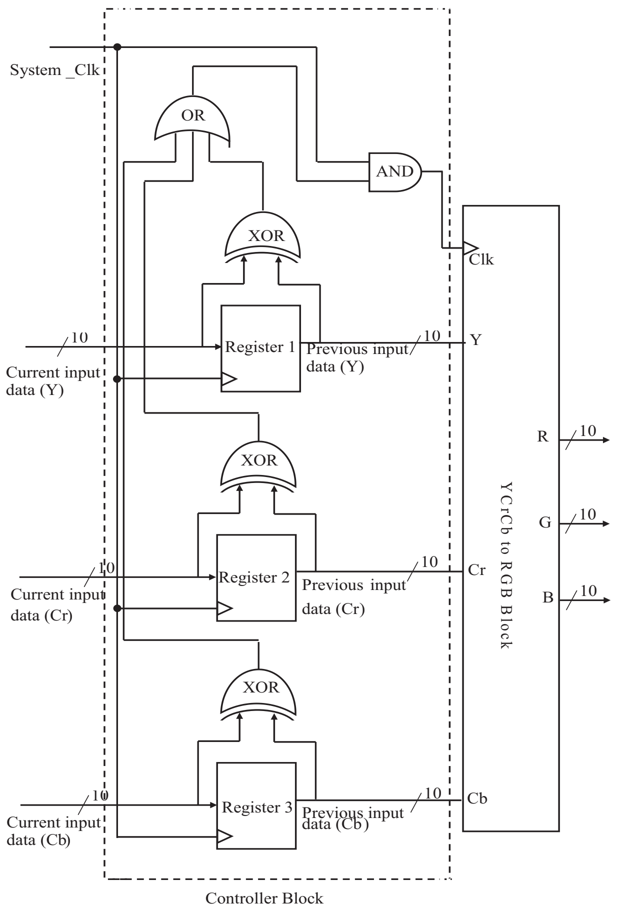


Figure 7. Clock controller block.

We can easily verify that Figure 7 meets the requirements (1 and 2) given in Section 2.1.

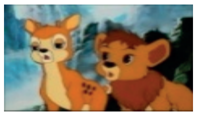
4. Experimental results

In this part, we have applied the proposed approach shown in Figure 5 to some videos (Table 2) chosen from different categories (documentary, cartoon, news) and we have compared the obtained results with those obtained using the classical architecture (Figure 3). In this comparison, we have used two FPGA platforms (the Xilinx Xup Virtex 2Pro and the Xilinx ML505 development systems) and we have focused on three main fields, which are power, maximum frequency, and resources. To meet our aim, we have used the following:

- PC HP based on Intel Core 2 as a processor running at 2.5 GHZ.
- Windows 7 as an operating system.

- Xilinx ISE (version 13.4) as a synthesis and simulation tool.
- Xilinx (Xup Virtex 2 Pro) and ML 505 Virtex-5.
- Video decoder based on ADV7183.

Table 2. Shots from used videos.

Video 1	Video 2	Video 3
		

4.1. Design results at simulation level

To evaluate the proposed method for power optimization, we have used a video corpus of 3 video sequences from three different types: documentaries, cartoons, and news. Table 2 shows an example of frames from the video sequences employed in the evaluation phase.

4.1.1. Without clock controller block

Let us consider the component shown in Figure 8. We send a set of random data as input to the YCrCb to RGB block and we note the simulation behavior of the YCrCb to RGB block.

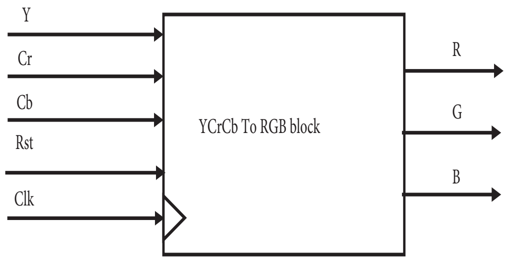


Figure 8. Clock controller block.

The Mentor Graphics ModelSim tool was used to run the simulations for functional validation. Figure 9 shows the simulation results of the YCrCb to RGB block. Based on this figure, we can state the following:

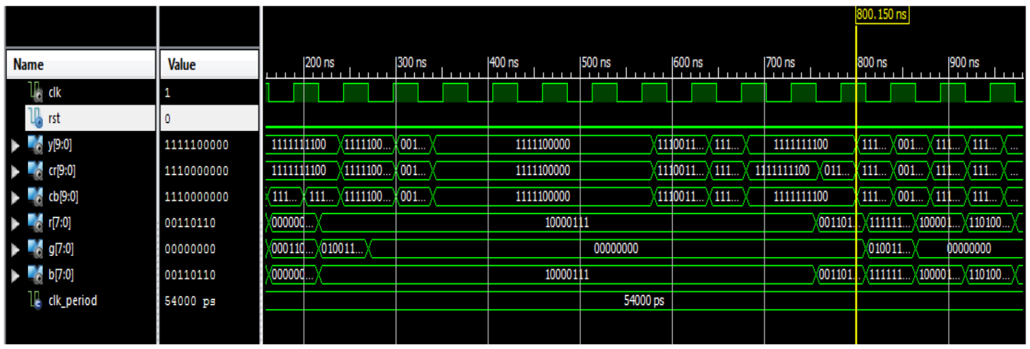


Figure 9. Timing diagram of simulation by classical design.

- The YCrCb to RGB block runs correctly; we always have Eqs. (1), (2), and (4) verified.
- Now let us calculate the dynamic power Pd in the interval of time T = [0 ns, 800 ns]. Knowing that the dynamic power Pd depends on the logic used by the design and the frequency [15], it can be written as:

$$Pd = A \times C \times V^2 \times F \tag{4}$$

where:

- A is the percentage of active logic gates, which are charged dynamically.
- C is the total capacitance load.
- V is the supply voltage and F is the execution frequency.

In our example:

$$F = F1 = 27 \times 10^6 Hz \tag{5}$$

$$\Rightarrow Pd1 = A \times C \times V^2 \times F1$$

$$= A \times C \times V^2 \times (27 \times 10^6 Hz) \tag{6}$$

4.1.2. With clock controller block

Let us considered the architecture shown in Figure 10. We send a set of random data; let us show the simulation behavior of the YCrCb to RGB block.

Figure 11 shows the simulation result of the YCrCb to RGB block, based on which we can state the following:

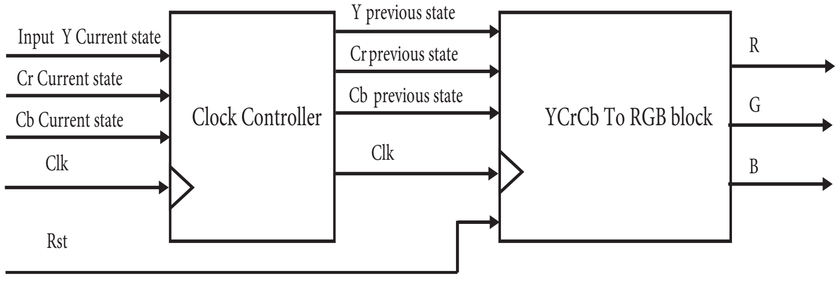


Figure 10. Clock controller and YCrCb to RGB block.

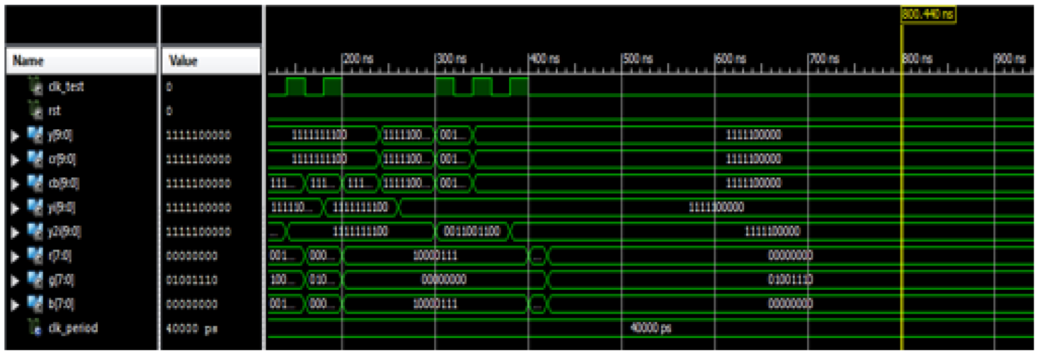


Figure 11. Timing diagram of simulation by the proposed design.

- The YCrCb to RGB block runs correctly; we always have Eqs. (1), (2), and (3) verified.
- The clock is not periodic.

Now let us calculate the dynamic power Pd in the interval of time T = [0 ns, 800 ns]. Since the clock is not periodic, we have considered its average period:

$$< T_{clock} >= T/N \tag{7}$$

where:

T: the interval of time [0 ns, 800 ns];

N: the number of rising edges of the clock in T.

In our example:

$$F = F1 = 7 \times 10^6 Hz \tag{8}$$

$$Pd1 = A \times C \times V^2 \times F1 = A \times C \times V^2 \times (7 \times 10^6 Hz) \tag{9}$$

Based on Eqs. (6) and (9), the gain in power is:

$$G = 100(Pd1 - P2/Pd1) = 72\% \tag{10}$$

Based on Eq. (10), for this simple example, our approach provides a gain, calculated at simulation level, of 72%.

4.2. Design result at physical level

In this part, we have implemented both the proposed approach and the classical one and we have compared them in terms of dynamic power, maximum frequency, and resources.

4.2.1. Comparison in terms of dynamic power

In this part, we have used the Xpower tool. Our clock system is running at 27 MHz.

Based on Table 3, we conclude that our approach managed to lower the level of consumed energy, attaining a decrease of about:

Table 3. Design results.

Platform	Xup Virtex2 Pro		
	Classical design	Our design	Gain %
Video 1	5.98 mW	3.92 mW	34.44
Video 2	6.44 mW	4.32 mW	32.91
Video 3	7.02 mW	4.48 mW	36.18
Platform	ML 505 Virtex5		
Video 1	5.87 mW	3.69 mW	37.13
Video 2	6.23 mW	4.05 mW	35.00
Video 3	6.89 mW	4.23 mW	38.6

- 35% compared to the classical design for the Xup Virtex 2 Pro board,
- 36% compared to the classical design for the ML 505 Virtex 5 board.

This table compares the hardware resource utilization before and after the integration of the clock controller module.

Figure 12 shows the difference of power minimization on two different platforms. The newer platform has less consumed power. This can be explained by the power-optimization solutions that are added to the design during the synthesis process.

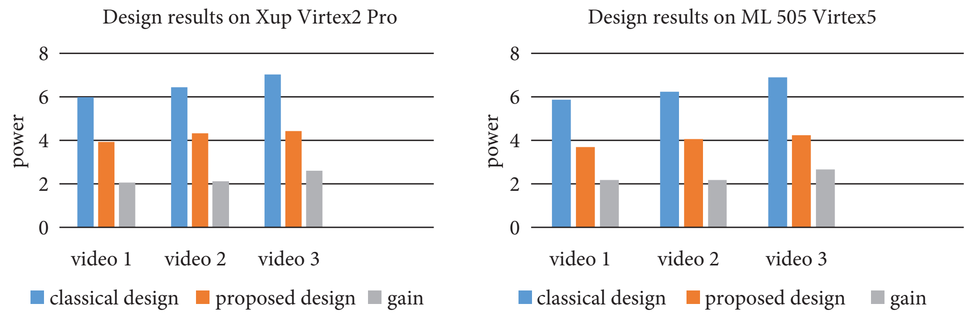


Figure 12. Power consumption with and without the proposed system compared on two different platforms.

4.2.2. Comparison in terms of maximum frequency

Often, optimizing one design parameter may lead to a strong degradation of other design parameters. For this reason, we add a comparison in terms of FPGA resources and in terms of maximum frequency between our design and the classical one. To improve the performance of the designed circuit, designers often choose to use an external clock, not the one used by the electronic board. A design is classified as better if it provides circuits that support the highest maximum frequency value, which is automatically calculated by ISE. Table 4 refers to the maximum frequency values.

Table 4. Design results.

	Our design, Xup	Classical design	Loss %
Xup Virtex 2 Pro	174.564 MHz	234.258 MHz	25.48
ML 505 Virtex 5	198.689 MHz	264.271 MHz	24.2

Table 4 show a loss by:

→ 25% compared to the classical method for the Xup Virtex 2 Pro board,

→ 24% compared to the classical method for the for the ML 505 Virtex 5 board.

The loss in maximum frequency has no effect on the system's behavior. In our case we do not need to speed up the system's clock more than 27 MHz because its frequency depends on the video decoder, which runs at 27 MHz.

4.2.3. Comparison in terms of FPGA resources

Synthesis results obtained using the EDK tool are represented in Table 5.

Table 5. Resource design results.

Logic utilization	Our design	Classical design	Loss %
SLICES	72	59	18
Flip-flops	37	33	11

The design results of Table 5 show that the loss in slices and in flip-flops is less than 18% and 10%, respectively. This loss is considered as negligible compared to the gain in power consumption.

According to these results, the system can support additional image and video treatments. This can be very useful for multiple and parallel video processing applications.

5. Conclusion

In this paper, we have shown that existing video zoom-in architectures still lack effective methods for the optimization of power dissipation through restricting the optimization goal to the performance in terms of image quality. To overcome this lack, we have presented in this paper a new design for low power zoom-in architecture. Our architecture has been implemented and compared with the existing ones. Design results show a significant gain in dynamic power without degrading other design parameters such as area or maximum frequency. In conclusion, our approach can be very interesting in the field of video processing design and can be applied to more consuming blocks.

References

[1] Perrucci GP, Fitzek P, Widmer J. Survey on energy consumption entities on the smartphone platform. In: IEEE 2011 Vehicular Technology Conference; 15–18 May 2011. New York, NY, USA: IEEE. pp. 1-6.

[2] Cao Z, Foo B, He L, van der Schaar M. Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications. IEEE T Circuits-I 2010; 57: 681-690.

[3] Mastronarde N, Kanoun K, Atienza D, Frossard P, van der Schaar M. Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems. IEEE T Multimedia 2013; 15: 268-278.

[4] Nguyen D, Davare A, Orshansky M, Chinnery D, Thompson B, Keutzer K. Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization. In: IEEE 2003 Low Power Electronics and Design Conference; 25–27 August 2003; Seoul, Korea. New York, NY, USA: IEEE. pp. 158-163.

[5] Mavlankar A, Girod B. Video streaming with interactive pan/tilt/zoom in high-quality visual experience. In: Mrak M, Grgic M, Kunt M, editors. High-Quality Visual Experience. Berlin, Germany: Springer, pp. 431-455.

[6] Gaddam VR, Langseth R, Ljodal S, Gurdjos P, Charvillat V, Griwodz C, Halvorsen P. Interactive zoom and panning from live panoramic video. In: ACM 2014 Network and Operating System Support on Digital Audio and Video Workshop; 19–21 March 2014. New York, NY, USA: ACM. pp. 19-24.

[7] Gaddam VR. Be your own cameraman: real-time support for zooming and panning into stored and live panoramic video In: ACM 2014 Multimedia Systems Conference; 19–21 March 2014; Singapore. New York, NY, USA: ACM. pp. 168-171.

[8] Wang F, Wu B, Zhang H, Liu H. Image zoom method based on bandelet transform modified bilinear interpolation. In: IEEE 2001 International Computational and Information Sciences Conference; 21–23 October 2011; Chengdu, China. New York, NY, USA: IEEE. pp. 121-124.

[9] Jain S, Suresh D. A new jpeg image scaling algorithm based on the area pixel model. International Journal of Engineering and Science 2013; 2: 46-56.

[10] Kim CH, Seong SM, Lee JA, Kim LS. Winscale: an image-scaling algorithm using an area pixel model. IEEE T Circ Syst Vid 2003; 13: 549-553.

[11] Danahy E, Agaian SS, Panetta KP. Algorithms for the resizing of binary and grayscale images using a logical image processing. Proc SPIE 2007; 6497: 306-316.

[12] Li J, He H, Man H, Desai S. A general-purpose FPGA-based reconfigurable platform for video and image processing. In: Advances in Neural Networks; 26–29 May 2009; Wuhan, China. Berlin, Germany: Springer. pp 299-309.

[13] Sima M, Vassiliadis S, Cotofana S, van Eijndhoven JTJ. Color space conversion for MPEG decoding on FPGA-augmented TriMedia processor. In: IEEE 2003 Application-Specific Systems, Architectures and Processors Conference; 24–26 June 2003; The Hague, the Netherlands. New York, NY, USA: IEEE. pp. 250-259.

[14] Hsu RL, Abdel-Mottaleb M, Jain AK. Face detection in color images. IEEE T Pattern Anal 2002; 5: 696-707.

[15] Wang L, Khan SU, Chen D, Kolodziej J, Ranjan R, Zu CZ, Zomaya A. Energy-aware parallel task scheduling in a cluster. Future Gener Comp Sy 2013; 29: 1661-1670.