# A Power-Aware Real-Time System for Multi-Video Treatment on FPGA with Dynamic Partial Reconfiguration and Voltage Scaling

Lilia Kechiche
Department of Science and Technology
Taif University
Saudi Arabia
l.kechiche@gmail.com

Lamjed Touil
Laboratory of Electronics and Microelectronics
University of Monastir
Monastir, Tunisia
lamjedtl@yahoo.fr

Mehdi Jemai
Laboratory of Electronics and Microelectronics
University of Monastir
Monastir, Tunisia
jmehdie@gmail.com

Bouraoui Ouni
Networked Objects Control and Communications
Systems Lab, University of Sousse
Sousse, Tunisia
ouni_bouraoui@yahoo.fr

*Received: 21 May 2022 | Revised: 10 June 2022 | Accepted: 13 June 2022*

**Abstract-As the energy consumption is an evaluating factor for System-On-Chip (SOC) design, this paper presents a power-aware architecture for a real-time multi-video system on FPGA. This architecture aims to optimize power consumption for a multi-video system on ARM-based architectures. The proposed architecture uses dynamic reconfiguration and voltage scaling to create a power-aware system for real-time multi-video processing with minimal power dissipation. Dynamic partial reconfiguration was used to optimize the utilization of resources and reduce dynamic power consumption. Voltage scaling was also used to optimize dynamic power consumption, by configuring the blocks to use the minimum necessary voltage for normal operating conditions. The proposed architecture focused on the Zynq platform. The results showed power savings of up to 70% concerning performance and real-time constraints.**

***Keywords-power consumption; Zynq; ARM A9; dynamic partial reconfiguration; voltage scaling***

## I. INTRODUCTION

Embedded real-time video applications are widely spread in many systems and have important applications in various domains such as segmentation [1, 2], object tracking [3], visual detection and matching [4], motion estimation [5], etc. These systems are generally executed in an embedded environment and are subjected to many constraints such as power consumption, time, and resources. As the market requires these systems to have high performance at a low cost, designers have to propose new architectures to meet different requirements. Many dedicated technologies and methods have been proposed to develop and implement high-quality real-time applications and optimized systems. The proposed technologies range from specific processors like General Purpose Processors (GPPs), Graphics Processing Units (GPUs), and Digital Signal Processors (DSPs) to parallel architectures like Application-Specific Integrated Circuits (ASICs) or even programmable logic devices (FPGAs). Today, FPGAs are being increasingly used to build complex video processing applications. They provide real-time performance that is difficult to achieve with GPP or DSP [6-8] while limiting the extensive design work required for ASICs. Furthermore, FPGAs provide the ability to implement highly parallel architectures due to the huge number of programmable logic available on the chip [9].

One of the major problems with FPGA implementations compared to ASIC solutions is power consumption, which is a limiting factor [10]. Therefore, more efforts are spent to propose a design with low-power dissipation. Since FPGAs are CMOS-transistors, power consumption can be divided into two main types: static power and dynamic power. Static power is dissipated when the circuit is in a quiescent state caused by leakage currents of the CMOS transistor. These currents are the sub-threshold leakage current, the gate leakage current, and the junction leakage current [11]. Dynamic power consumption is given by:

$$P_d = \alpha . C . V^2 . f \quad (1)$$

where $C$ is the capacitance, $\alpha$ is the charging rate depending on clock frequency, $V$ is voltage supply, and $f$ is the clock frequency. This power is highly related to technology and has become a concern with modern FPGAs implemented in 24nm [12-14]. To reduce power consumption, the principal causes of power dissipation have to be investigated during all steps of the design process, from the algorithmic level down to the

Corresponding author: Lilia Kechiche

transistor level, considering the latest low-power technology methods at all the design levels.

This paper proposes a power-aware hardware architecture for real-time multi-video processing, using dynamic reconfiguration and voltage scaling to optimize the power consumption of a multi-video system. This method was applied to a high-performance video system of 1920×1080 pixels at 60fps.

## II. FPGA POWER REDUCTION

Most modern FPGA boards are computing platforms that include programmable hardware elements, memory resources, configurable I/O, embedded processors, and even embedded operating systems. Hardware (HW) and software (SW) functionalities allow the combination of hardware's performance and software's flexibility. This combination of performance and flexibility comes at the cost of high power consumption, which is the main limit of FPGA platforms. To overcome this limit, new methods to optimize power consumption are investigated. The proposed methods explore all abstraction levels of the design process, from the system level to the circuit level and the technological level. Some works used HW/SW partitioning to propose optimal systems with low power consumption as power-aware decisions at a very early stage of the design process [15-17]. HW/SW partitioning is the problem of assigning application tasks to the existing computational cores under defined constraints such as area and power. It is formalized as an optimization problem aiming to minimize an objective function under defined constraints. In [15], an algorithm was proposed for HW/SW partitioning to find the best tradeoff between power and latency, modeling the application as a data flow graph and computing the latency and power consumption for every proposed partitioning. This algorithm performed a heuristic search for the best solution that respected the defined constraints. In [17], a data flow graph based on the Bee Colony Algorithm was proposed to solve the optimization problem of HW/SW partitioning under time and power constraints. The heuristic algorithm treated the optimization problem as NP-Hard, and the exact resolution may take a much longer time. To adjust the constraints according to the user's requirements, weighting coefficients were added to the constraints to specify which of the two conflicting terms is more important for the final partitioning result.

Other studies examined methods at the architecture level, like Dynamic Voltage Scaling (DVS) and Dynamic Frequency Scaling (DFS). DVS, DFS, or even Dynamic Voltage and Frequency Scaling (DVFS) were first proposed to reduce the power consumption of microprocessors [18-20] and, as they were successful, they were generalized and used on FPGAs [21-22]. In [18], a method for static timing analysis in dynamic scheduling schemes was proposed. A safe timing analysis was proposed for systems with off-chip memories where memory latency did not scale with processor frequency. This method, called 'frequency-aware', replaced the Worst-Case Execution Time (WCET) [23] obtained by static-timing analysis. It expressed WCET bounds with frequency-sensitive parameters, where cycles were interpreted in terms of processor frequency and memory accesses were expressed in terms of the memory latency overhead. The new proposed WCET was determined on-the-fly for a given frequency.

The problem of real-time systems with time-critical applications was addressed in [19]. A new algorithm was proposed for DFS, applied directly to the scheduler to modify task management. At first, a static frequency was assigned to every task, which was the lowest possible frequency that allowed the scheduler to meet the deadlines for a given task set. If a task was completed before the worst-case specification, the frequency was re-computed using the latest information. The new value was used until the release of the task for a future invocation. Therefore, the utilization was recomputed at every new scheduling time using the real computed time for accomplished tasks and the specified worst-case for the others. This method allowed a 20-40% power reduction. A DVFS technique for non-real-time applications was proposed in [20], where the main idea was to lower the CPU frequency when accessing off-chip peripherals. The temporal distribution of on- and off-chip workloads was computed with different scenarios to determine the CPU frequency during idle periods. The energy savings were up to 70%, with a variable performance penalty depending on the saving value.

Two methods for DVS on commercial FPGAs were presented in [21, 22]. In [21], a circuit was used to measure the logic delay, which would be used by the voltage controller block to dynamically adjust the supply voltage using a closed loop. The voltage controller was an external module implemented on a PC. Experimental results using a Xilinx Virtex XCV300E FPGA were presented, and power savings were 4-54%. Although the study noted that implementing the voltage controller as an internal module is a way for better resource utilization, giving results and statistics based on an external module is disputable since it may lack precision and can cause delay violations or more resource consumption.

## III. REAL-TIME VIDEO PROCESSING AND ARCHITECTURE

### *A. Real-Time Video Processing*

Real-time video processing plays a key role in industrial systems and is expanded to many fields. Real-time video processing systems process large amounts of image data in a short time. The purpose varies from a simple display to the extraction of useful information for intelligent scene analysis. Digital videos are data-intensive and resource-demanding multidimensional signals, as they need an important amount of resources for computations and memory operations. A typical video system consists of a video source, internal processing, and destination, as shown in Figure 1. The video processing module is traditionally classified into three levels: low, intermediate, and high. Each level differs from the others in input/output format and processing type. For example, the low-level takes an image and produces an output image, while the high-level takes image attributes as input and makes high-level interpretation to produce a knowledge-based control as output.

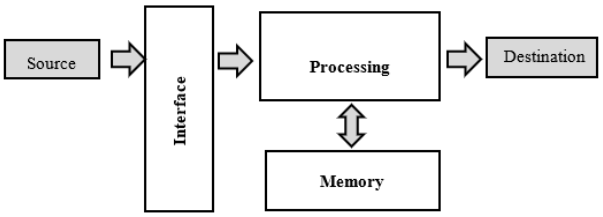


Fig. 1. Video processing system.

A common concern in real-time video processing systems is how to deal with large amounts of data. This pushes designers to search for a suitable architecture that guarantees the required performance with the worst-case latency to ensure no frame drops. This remains a challenging task, especially for embedded systems that have limited resources and power supply. The architecture has to allow parallel treatments, as it is hard to perform such a large data treatment serially. The best solution often comes with a combination of hardware and software approaches. The hardware offers high performance using parallelism, and the software guarantees flexibility. Fortunately, modern FPGA devices are characterized by sufficient logic resources and high operating frequencies for video processing.

### *B. Hardware and Software Architecture for Real-Time Multi-Video Processing*

The design of a real-time system for multi-video processing is a demanding task. In addition to the constraints discussed above, constraints such as available resources and memory access management are added to the problem. The design of multi-video architecture needs to answer the following demands:

- Achieve the performance required by different parallel and communicating blocks. Blocks concerning video processing systems are subjected to timing constraints for high-bandwidth and data-intensive operations with the need to access the memory at the video rate.
- Extend it when needed. The extension is the ability to add new video processing chains or blocks or the limit of available resources.
- Obtain an optimal use of available resources. System partitioning between hard and soft resources allows getting the high performance of HW and the flexibility of SW.

In addition to the previous requirements, which are in direct relation to the system and platform, power consumption is another constraint that has to be considered, as battery life has become the main factor in the evaluation of embedded systems. The choice of the appropriate target platform is very essential for real-time systems. Of the existing FPGA platforms, many are appropriate for video processing with real-time constraints, as they can work at a frequency that can exceed 150MHz, and as a result, can support HD resolutions. In addition, FPGA vendors have incorporated various hardware and software IP cores that can be used for different functions needed in video processing, like timing generation and RGB conversions.

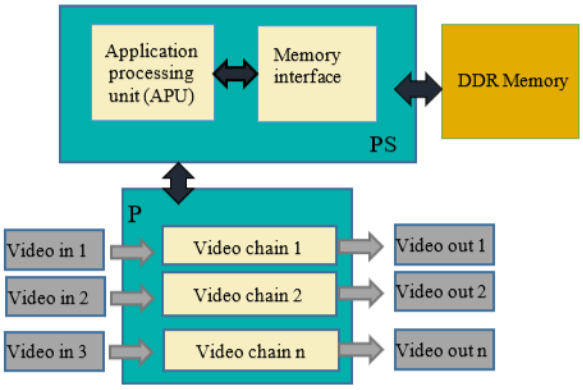


Fig. 2. General architecture for multi-video processing on FPGA.

Figure 2 shows the general architecture for multi-video processing using HW and SW available in the FPGA. The HW or Programmable Logic (PL) is used to implement video-related blocks like video-in, video processing sub-systems, and video-out. The SW or Processing System (PS) contains the Application Processing Unit (APU), which is responsible for system-level control registers, DMA controllers, and the Accelerator Coherency Port (ACP). The memory interface allows the PS and PL blocks to access the memory. The utilization of PS and PL to implement video processing modules is performed according to the available resources, performance constraints, power constraints, and other issues like flexibility and time to market.

This study used a Zynq ZC 702 based Xilinx evaluation kit as a target platform. This kit includes SW, HW, and IP components that facilitate the development of custom video applications. The APU contains two ARM Cortex-A9 processors sharing a 512KB L2 cache, which can be used for dual-core or single-core devices. Each processor is a low-power, high-performance core with a 32KB L1 cache for instruction and data. The AXI protocol [24] defines 3 types of interfaces: AXI4 for high-performance memory-mapped requirements, AXI4-Lite for low-throughput memory-mapped communications, and AXI4-Stream for high-speed streaming data. These application domains make the protocol indispensable for every real-time video system. The AXI interconnect, AXI3, and AXI Video DMA IP cores can form the basis of video systems capable of handling video frame buffers and giving access to a shared DDR3 SDRAM. This design utilized the AXI4, AXI3, AXI4-Lite, and AXI4-Stream interfaces. The Accelerator Coherency Port (ACP) is used to communicate the PL with the APU, as it is an AXI interface that allows the PL to implement an AXI master to access the L2 [25-27]. The AXI Video Direct Memory Access (VDMA) core implements a video-optimized direct memory access engine with a frame buffer. The AXI VDMA core transfers video data to and from memory under dynamic software control. Figure 3 shows the block diagram of the implemented design with the interface connection. The processor can access the PL using the AXI3 master General-Purpose (GP) 32-bit interfaces. The Zynq ZC702 has 4 AXI_HP interfaces, which allow PL to access DDR memory through high-bandwidth data path bus masters.

## IV. DYNAMIC PARTIAL RECONFIGURATION FOR POWER REDUCTION

The Dynamic Partial Reconfiguration (DPR) offers a way to optimize the dynamic power consumption by enabling the usage of temporarily shared resources on the FPGA among different processing functions in a time-multiplexed manner. Hence, DPR enables better resource utilization and efficiency. The DPR results in a partial bitstream loaded onto the FPGA, targeting a fixed part of its area. Figure 3 shows the general architecture of multi-video processing. Fixed areas are predefined during HW design and are called the Reconfiguration Partition (RP).

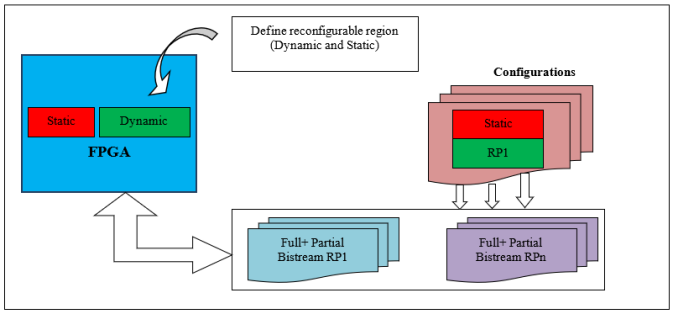


Fig. 3. General DPR architecture for multi-video processing.

## V. DYNAMIC PARTIAL RECONFIGURATION FOR REAL-TIME MULTI-VIDEO PROCESSING

Real-time multi-video processing is a demanding task in terms of performance, resource utilization, and power consumption. The target architecture was composed of the PS part, which allows communication with the DR, and the PL parts where the video-related blocks are implemented. A multi-video architecture is defined as a system with *n* video inputs, where *n* is limited by the resource constraints of the target platform. In this case, *n*=4 since the target Zynq platform has 4 HP ports that allow high performance communication with DDR, as shown in Figure 4.

A DPR-based architecture was proposed to optimize resources and power. DPR, also known as active partial reconfiguration, allows changing a part of the device while other blocks are running. While the FPGA is executing different blocks, the partial data will be sent to be configured. There are two ways for DPR [28], known as Difference Based Partial Reconfiguration (DBPR) and Module Based Partial Reconfiguration (MBPR). DBPR is used when a small change is made to the design. It is especially useful when changing Look-Up Table (LUT) equations or dedicated memory block content. MBPR uses modular design concepts to reconfigure large logic blocks. MBPR was used to implement the DPR on the target platform, defining two different parts: static modules and dynamic or Reconfigurable Modules (RMs). The static module is the part of the design that remains in operation during the PR process. The dynamic modules are the parts of the design that can be swapped in and out of the device on the fly, where multiple RMs can be defined for a specific region.

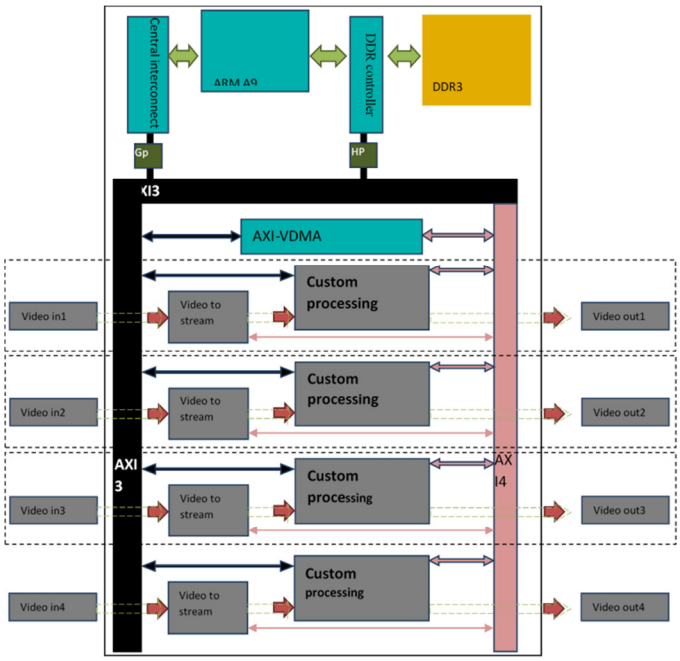


Fig. 4. Real-time multi-video processing.

Figure 5 shows the proposed architecture with both static and dynamic parts. PS and the multi-display present the static part while the dynamic part contains the PL.

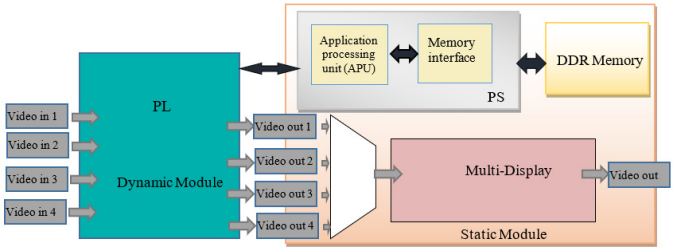


Fig. 5. The proposed architecture with RDP.

## VI. VOLTAGE SCALING

DVS is the process of varying the voltage of a target block at run-time. As voltage is directly related to power consumption [1], reducing it allows for a quadric reduction of power consumption. The following definitions must be considered to use DVS in a target FPGA [29]:

- The processing strength of a device is the ability and degree of variation in the attributes of the integrated transistors. There are three classes: weak, nominal, and strong. A weak device can operate with the lowest acceptable frequency at nominal voltages. Strong devices can run at faster frequencies than required at nominal voltages and can function at voltages lower than nominal at the minimum specified frequency.
- The voltage domain is defined as the group of modules sharing the same power supply voltage for the core logic of each device.

- The operating performance point is the voltage required for every device to operate at the desired processor clocking frequency. For example, for the processing system DDR I/O supply voltage, the minimum value is 1.14V and the maximum is 1.89V [30].
- The critical path of a system is defined as the longest path to achieve the execution of a program from the source to the sink of a data flow graph. This information is used to track the critical tasks when scaling voltage and frequency to ensure the performance of the system.

To implement DVS on a specific platform for a defined system, information about the operating voltage of each block must be accessible at runtime. This information can be obtained in two different ways:

- Dynamic analysis of the behavior of the system at runtime requires real-time access to the working blocks on the PL side. This can be done with additional modules implemented in PL or PS. This method requires extra resources to gather information at run-time.
- Static analysis of the system behavior using different test scenarios. The obtained test results can be used with the information provided by the manufacturer to make design decisions for frequency and voltage scaling.

This study followed the second method to perform DVS and DFS at runtime. Let $V_{min\_x}$ be the minimum voltage required for block $x$ to operate in normal conditions and under which the system fails. These values are a characteristic of hardware blocks, fixed at the time of device manufacturing. Let $V_{op\_x}$ be the operating voltage of block $x$. The algorithm used to scale voltage is shown in Figure 6, where st is a float representing the step of incrementing and decrementing the voltage.

```
Inputs : Vmin_x, , Vop_x, st;
Begin
Identify voltage bounds for the target block;
Do
    Vop = Vop - st ;
    Test performance;
    if test failed
        Vop = Vop + st;
        Break;
     End if
While (Vop > Vmin_x)
End
```

Fig. 6. Voltage scaling algorithm.

In the ZC702 board, power is supplied to the components through several independent rails using programmable power regulators (UCD9248) and a Power Management Bus (PMBus) compliant system controller from Texas Instruments. PMBus is an open standard protocol that defines communication with power converters and allows to write and read power, current, and voltage information. The processors can access the PMBus using a 1-to-8 I2C switch. Zynq-ZC7020 is divided into several power domains, as shown in Figure 7. These power domains are generated by 6 regulators that generate different voltages required by the Zynq and the onboard components. The supplied voltage and currents are continuously measured and monitored by three UCD9248 power controllers available on the ZC702 board.

In this work, scaling the voltage of hard blocks was performed using the PMBus protocol, which simplifies communication with power converters as it allows the device's SW or HW to access a manageable power supply [31, 32]. Optimal operating voltages were calculated using the algorithm shown in Figure 6. Then, the system was configured only one time with these values at every start. The configuration and implementation of the proposed algorithm were carried out by the processor through soft code.

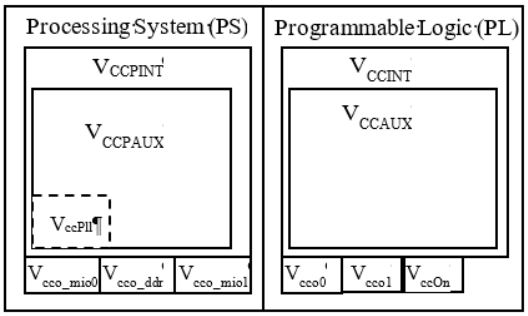


Fig. 7. Zynq power domains [31].

## VII. IMPLEMENTATION AND RESULTS

The proposed architecture was prototyped on the Zynq ZC702 evaluation board, which provides a hardware environment for developing and evaluating designs targeting the Zynq XC7Z020 device. ZC702 includes 1GB DDR3 component memory, 128MB Quad SPI flash memory, USB 2.0, Secure Digital (SD) connector, HDMI codec, I2C bus, 2 UART interfaces, and other features. This section provides experimental results and shows the benefits of the proposed design in power reduction. A performance comparison was also held to verify the performance of the whole system.

### *A. Hardware and Software Architecture*

The target system was designed using the Vivado Design Suite. This design suite can accelerate design implementation with place and route tools that analytically optimize for multiple and concurrent design metrics, such as timing and power. Vivado gives the ability to analyze the design at each design stage, allowing for modifications in the design process. It also provides timing and power estimations after synthesis, placement, or routing. The following blocks were used to implement the hardware architecture:

- The processing system was used to initialize blocks and master memory access. The frequency and voltage scaling modules were also implemented in the processing system.
- The AXI interconnects were responsible for handling information to and from the processing system.
- The AXI performance monitor was used for different statistics with the AXI protocol interfaces. It was used for transactions, external system events, and performance measurement for AXI4, AXI3, AXI4-Stream, and AXI4-Lite interfaces and captured real-time performance metrics

for throughput and latency. The performance monitor IP was used to perform real-time profiling using the SDK.

- The video pipeline contains Video Direct Memory Access (VDMA), which is a soft IP core that provides high bandwidth for direct access to the memory using AXI4-Stream video peripherals. The AXI4-Lite slave interface was used to perform initialization, registers, and status.
- The video-related blocks are standard blocks used in a video chain to transfer the video stream after the necessary conversions. The RGB to YCrCb module converts the design pixels from the RGB encoding used by the AXI4-Stream to the 16-bit YUV 4:4:4 signal format. The Chroma Resampler block had the output of the RGB to YCrCb as input. Its role was to convert the input signals to YUV 4:2:2 required by the HDMI output.

The software part running on the PS was built using the Xilinx Software Development Kit (SDK). The software part running on the ARM processor with a standalone operating system had the role of hardware control.

### *B. Partial Dynamic Reconfiguration Results*

Figure 8 shows the resource results without (a) and with PDR (b). Figure 9 shows the occupation of the resources on the target platform without (a) and with DPR (b). The figures show an improvement in the used resources, which lowered power consumption.

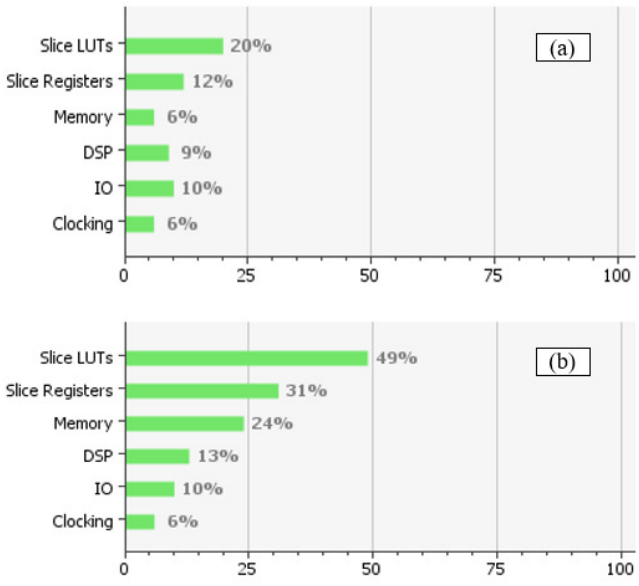


Fig. 8. Resource utilization.

### *C. Voltage Scaling Results*

The voltage scaling method was implemented as software code running on the processor. Communication with the voltage rails was performed using the I2C bus. To avoid unnecessary additional resources, the optimal values were computed using excessive test scenarios. This analysis defined the optimal values that would be used by the system without the need to collect them at run-time using additional resources. The DVS method requires knowledge of the voltage limits of different blocks. Table II shows the maximum and minimum recommended operating voltage values for some blocks of the ZC702 device [30].

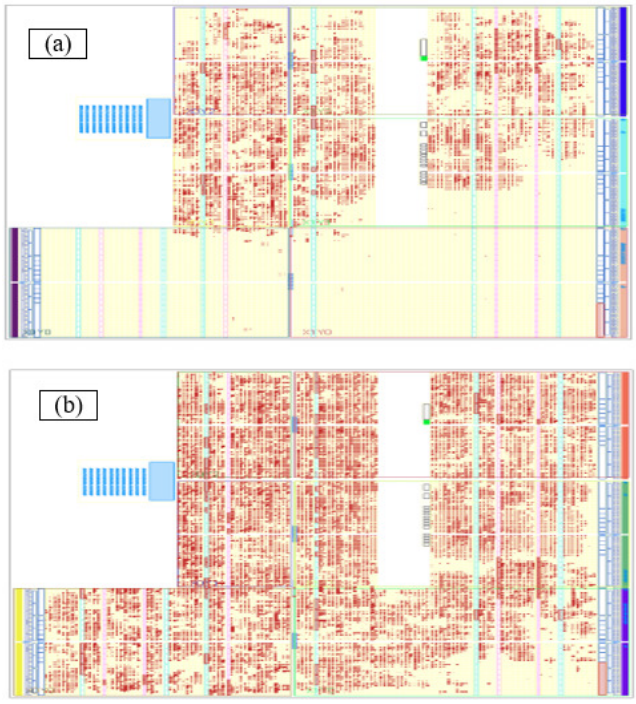


Fig. 9. Resource occupation.

TABLE I. VOLTAGE RECOMMENDED VALUES FOR SOME BLOCKS OF THE TARGET ZYNQ ZC702 DEVICE

| Blocks | Description | Minimum operating voltage (V) | Typical operating voltage (V) | Maximum operating voltage (V) |
|---|---|---|---|---|
| vccpint | PS internal supply voltage | 0.95 | 1 | 1.05 |
| vccint | PL internal supply voltage | 0.95 | 1 | 1.05 |
| vccpaux | PS auxiliary supply voltage | 1.71 | 1.80 | 1.89 |
| vccaux | PL auxiliary supply voltage | 1.71 | 1.80 | 1.89 |
| vccbram | PL block RAM supply voltage | 0.95 | 1.00 | 1.05 |

Although voltage can reach more inferior values than those indicated in the safety bounds, for example, Vccpint minimal value can reach 0.5V [30], the safe functioning of the device outside the indicated bounds is not tested. It is indicated that exposure to maximum values for extended periods could affect the device's reliability [30]. Figure 10 shows the voltage scaling results of some Vcc rails. Every figure shows the power margin for every chosen value. For example, for the Vccint rail, when the minimum operating value is chosen (0.95), the power consumption of this block varies between 22 and 31mW.

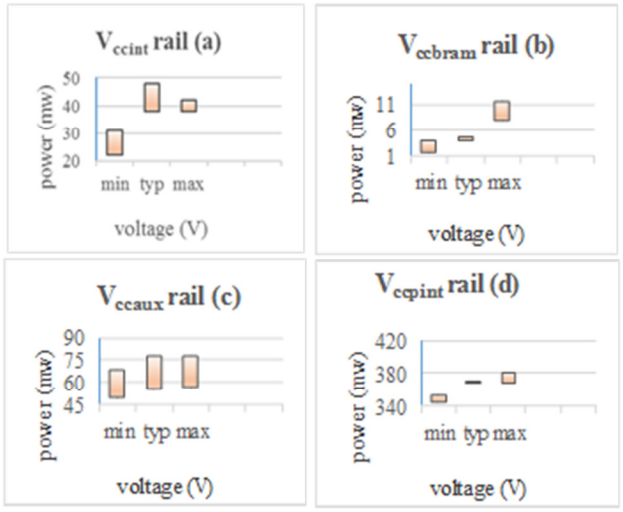


Fig. 10. Voltage scaling results on different rails.

Figure 11 shows the time needed by the rail to reach the target voltage value. This figure shows the case of two rails with different typical voltage values. For both rails, the typical functioning voltage was attended after 5ms.

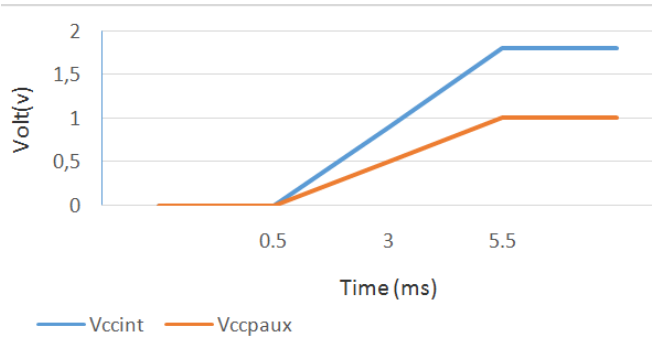


Fig. 11. Voltage configuration time.

### *D. Frequency Scaling and Design Performance*

Frequency scaling is the concept of varying the operating frequencies of blocks at run-time according to the target architecture. There are two frequencies: the frequency of the PL blocks and the frequency of the PS. An analysis of the functioning of the system was performed to study the possibility of applying frequency scaling, using the System Performance Analysis (SPA) toolbox. This toolbox was used to explore the performance of HW and SW at an early stage in the target system. Observing system performance in critical stages helped make the right optimization decisions and refine system performance without degradation. The SPA software metrics included CPU utilization, instructions per cycle, L1 data cache access, miss rate, write and read stall cycles per instruction, and other metrics. The hardware metrics were available using the AXI performance monitor core, designed to measure the real-time performance of the connected AXI interfaces, including AXI read and write transactions, throughput, bandwidth, and others. This work used CPU utilization and AXI read latency and throughput metrics.

Figure 12 shows the CPU utilization rate of the proposed real-time architecture. Only one core was used. The graph shows that the utilization ranged between 91% and 100%. These results show that applying DFS on one processor executing a real-time video system does not allow a noticeable optimization, as the CPU utilization is usually at its maximum level.

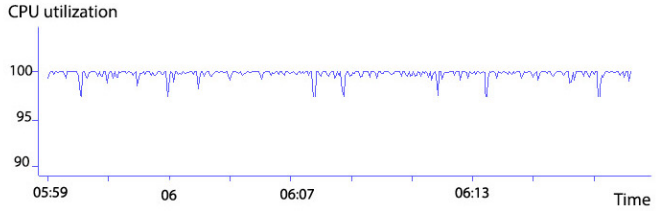


Fig. 12. CPU utilization.

Figure 13 shows the read transactions and latency cycles of the ports HP0 (Slot 0) and GP0 (Slot 1).

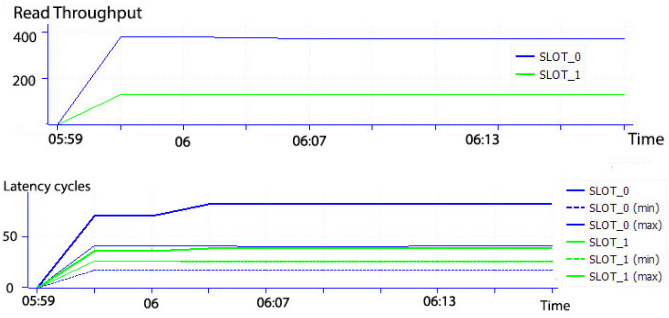


Fig. 13. Performance metrics at runtime.

### *E. Comparison with Other Works*

To the best of our knowledge, this is the first work to bring together DPR and voltage scaling. This section limits the discussion and comparison with works that used frequency and voltage scaling, as these two methods allow the reduction of the biggest part of power consumption (60%). Voltage and frequency scaling of commercial FPGAs was implemented in several studies using different methods. The proposed method was compared with two existing main methods that focused on the frequency and voltage of commercial 28nm FPGAs. The first study used voltage and frequency scaling [34], while in [35], voltage, frequency, and logic scaling were used to optimize power consumption. The voltage scaling unit was identical to the one used in [34]. The frequency-scaling unit used a ROM containing the configuration parameters used by the MMCM to generate the clock for the user logic. The frequency decreased continually at run-time and stopped when a value was detected that could cause timing violations. Compared to those studies, this study used voltage scaling in addition to the DPR. Authors in [34, 35] collected information like frequency and voltage of functioning blocks at run-time and scaled them accordingly. Collecting information at run-time results in extra resources and additional complexity. These works applied the proposed

methods to Power Consuming and Speed Testing Modules (PCASTMs). These PCASTMs were proposed with different numbers of modules occupyινγ different percentages of the device. For voltage scaling, the voltage values were tested under the recommended values in the datasheet. Testing with values below the recommended minimum can give more power savings of up to 60% [36], but with doubtful safety for long-term function with real-time constraints.

This study exploited the fact that since the different execution scenarios are known in advance, the collection of information was performed at the design level using tests and timing analysis to determine the optimal operating points. Implementing the proposed method requires only an I2C IP core and is of low complexity. Additionally, scaling the voltage many times at runtime is not useful as it results in power and heat dissipation while accessing the power rails. It is more efficient to configure the device blocks with the optimal voltage at the start. Table III shows a comparison between [34, 35] and this study. According to the results obtained, the proposed method presents up to 70% power savings with an improvement of 5% compared to the other studies.

TABLE II. COMPARISON WITH OTHER WORKS

| | **Additional resources** | **Frequency scaling method** | **Voltage scaling method** | **Test method** | **Power achievements** |
|---|---|---|---|---|---|
| **[34]** | Microblaze, Picoblaze, I2C, IP core, Dual-port RAM | I2C communication with Si570 oscillator | I2C communication with PMBus | Random functions with different sizes | Up to 64.98%, (using voltage and frequency scaling) |
| **[35]** | Microblaze, I2C, IP core Dual-port RAM ROM | Configuration of MMCM, ROM | I2C communication with PMBus | Random functions with different sizes | Up to 60% (using frequency, voltage, and logic scaling) |
| **This study** | I2C IP core | none | Soft configuration of PMBus | Real-time video application with real-time constraints | Up to 70% (voltage scaling and PDR) |

## VIII. CONCLUSION

The design of an optimized system is very challenging due to the complexity of a SOC design. Power consumption has become a struggle with the increasing consumers' demands and the saturation due to Moore's law. Designers have to find new solutions to reduce power consumption. This study proposed a design method that brings together partial reconfiguration and voltage scaling. These methods were used to design a real-time multi-video processing system, implemented on a Zynq ZC702 board with an ARM 9 target processor. The proposed method ισ general and can be applied to other real-time systems. Compared to existing works, the proposed design allowed up to 70% power savings with minimal additional resources. Future work should examine the application of other methods like clock gating and their integration in the design of real-time video systems.

## REFERENCES


[1] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards Real-Time Multi-Object Tracking," in *Computer Vision – ECCV 2020*, 2020, pp. 107–122, https://doi.org/10.1007/978-3-030-58621-8_7.

[2] A. N. Saeed, "A Machine Learning based Approach for Segmenting Retinal Nerve Images using Artificial Neural Networks," Engineering, Technology & Applied Science Research, vol. 10, no. 4, pp. 5986–5991, Aug. 2020, https://doi.org/10.48084/etasr.3666.

[3] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards Real-Time Multi-Object Tracking," in *Computer Vision – ECCV 2020*, 2020, pp. 107–122, https://doi.org/10.1007/978-3-030-58621-8_7.

[4] S. Banerjee, A. Bandyopadhyay, A. Mukherjee, A. Das, and R. Bag, "Random Valued Impulse Noise Removal Using Region Based Detection Approach," *Engineering, Technology & Applied Science Research*, vol. 7, no. 6, pp. 2288–2292, Dec. 2017, https://doi.org/10.48084/etasr.1609.

[5] I. Usman, "An Efficient Depth Estimation Technique Using 3-Trait Luminance Profiling," *Engineering, Technology & Applied Science Research*, vol. 9, no. 4, pp. 4428–4432, Aug. 2019, https://doi.org/10.48084/etasr.2857.

[6] 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*, Monterey, CA, USA, Oct. 2012, pp. 47–56, https://doi.org/10.1145/2145694.2145704.

[7] K. Pauwels, M. Tomasi, J. Diaz Alonso, E. Ros, and M. M. Van Hulle, "A Comparison of FPGA and GPU for Real-Time Phase-Based Optical Flow, Stereo, and Local Image Features," *IEEE Transactions on Computers*, vol. 61, no. 7, pp. 999–1012, Jul. 2012, https://doi.org/10.1109/TC.2011.120.

[8] G. Mingas and C.-S. Bouganis, "Population-Based MCMC on Multi-Core CPUs, GPUs and FPGAs," *IEEE Transactions on Computers*, vol. 65, no. 4, pp. 1283–1296, Apr. 2016, https://doi.org/10.1109/TC.2015.2439256.

[9] M. Baklouti, Y. Aydi, Ph. Marquet, J. L. Dekeyser, and M. Abid, "Scalable mpNoC for massively parallel systems – Design and implementation on FPGA," *Journal of Systems Architecture*, vol. 56, no. 7, pp. 278–292, Jul. 2010, https://doi.org/10.1016/j.sysarc.2010.04.001.

[10] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, Oct. 2007, https://doi.org/10.1109/TCAD.2006.884574.

[11] Altera Corporation, "40-nm FPGAs: Architecture and Performance Comparison," White Paper ver. 1.0, Dec. 2008.

[12] A. Kumar and M. Anis, "An analytical state dependent leakage power model for FPGAs," in *Proceedings of the Design Automation & Test in Europe Conference*, Munich, Germany, Mar. 2006, https://doi.org/10.1109/DATE.2006.243995.

[13] T. Tuan and B. Lai, "Leakage power analysis of a 90nm FPGA," in *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, San Jose, CA, USA, Sep. 2003, pp. 57–60, https://doi.org/10.1109/CICC.2003.1249359.

[14] J. Hussein, M. Klein, and M. Hart, "Lowering Power at 28 nm with Xilinx 7 Series FPGAs," Xilinx, White Paper WP389 (v1.1), Jun. 2011.

[15] S. Ben Haj Hassine, M. Jemai, and B. Ouni, "Power and Execution Time Optimization through Hardware Software Partitioning Algorithm for Core Based Embedded System," *Journal of Optimization*, vol. 2017, Feb. 2017, Art. no. e8624021, https://doi.org/10.1155/2017/8624021.

[16] H. Han, W. Liu, J. Wu, and G. Jiang, "Efficient Algorithm for Hardware/Software Partitioning and Scheduling on MPSoC," *Journal of Computers*, vol. 8, no. 1, pp. 61–68, Jan. 2013, https://doi.org/10.4304/jcp.8.1.61-68.

[17] L. Kechiche, L. Touil, and B. Ouni, "High-level optimised systems design using hardware-software partitioning," *International Journal of Advanced Intelligence Paradigms*, vol. 13, no. 3–4, pp. 346–367, Jan. 2019, https://doi.org/10.1504/IJAIP.2019.101984.

[18] K. Seth, A. Anantaraman, F. Mueller, and E. Rotenberg, "FAST: Frequency-aware static timing analysis," *ACM Transactions on Embedded Computing Systems*, vol. 5, no. 1, pp. 200–224, Oct. 2006, https://doi.org/10.1145/1132357.1132364.

[19] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, Alberta, Canada, Jul. 2001, pp. 89–102, https://doi.org/10.1145/502034.502044.

[20] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 18–28, Jan. 2005, https://doi.org/10.1109/TCAD.2004.839485.

[21] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Dynamic voltage scaling for commercial FPGAs," in *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005.*, Singapore, Sep. 2005, pp. 173–180, https://doi.org/10.1109/FPT.2005.1568543.

[22] J. L. Nunez-Yanez, "Adaptive Voltage Scaling with In-Situ Detectors in Commercial FPGAs," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 45–53, Jan. 2015, https://doi.org/10.1109/TC.2014.2365963.

[23] R. Wilhelm *et al.*, "The worst-case execution-time problem overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, pp. 1–53, May 2008, Art. no. 56, https://doi.org/10.1145/1347375.1347389.

[24] Xilinx, "AXI Reference Guide," Xilinx, UG761 (v14.3), Nov. 2012.

[25] Xilinx, "ZC702 Evaluation Board for the Zynq-7000 XC7Z020 SoC User Guide," Xilinx, UG850 (v1.7), 2019.

[26] Xilinx, "Zynq-7000 SoC Technical Reference Manual," Xilinx, UG585 (v1.11), Sep. 2016.

[27] J. Lucero and Y. Arbel, "Designing High-Performance Video Systems with the Zynq-7000 All Programmable SoC," Xilinx, XAPP792 (v1.0.1), Oct. 2012.

[28] W. Lie and W. Feng-yan, "Dynamic Partial Reconfiguration in FPGAs," in *2009 Third International Symposium on Intelligent Information Technology Application*, Nanchang, China, Aug. 2009, vol. 2, pp. 445–448, https://doi.org/10.1109/IITA.2009.334.

[29] F. Dehmelt, "Adaptive (Dynamic) Voltage (Frequency) Scaling—Motivation and Implementation," Texas Instruments, Dallas, TX, USA, Application Report SLVA646, Mar. 2014.

[30] Xilinx, "Zynq-7000 SoC (Z-7007S, Z-7012S, Z-7014S, Z-7010, Z-7015, and Z-7020): DC and AC Switching Characteristics," Xilinx, Product Specification DS187 (v1.21), Dec. 2020.

[31] A. F. Beldachi and J. L. Nunez-Yanez, "Accurate power control and monitoring in ZYNQ boards," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Munich, Germany, Sep. 2014, pp. 1–4, https://doi.org/10.1109/FPL.2014.6927415.

[32] J. Nunez-Yanez, "Adaptive voltage scaling in a heterogeneous FPGA device with memory and logic in-situ detectors," *Microprocessors and Microsystems*, vol. 51, pp. 227–238, Jun. 2017, https://doi.org/10.1016/j.micpro.2017.04.021.

[33] Xilinx, "Zynq-7000 SoC PCB Design Guide," Xilinx, UG933 (v.1.12), 2019.

[34] A. F. Beldachi and J. L. Nunez-Yanez, "Run-time power and performance scaling in 28 nm FPGAs," *IET Computers & Digital Techniques*, vol. 8, no. 4, pp. 178–186, 2014, https://doi.org/10.1049/iet-cdt.2013.0117.

[35] J. Luis Nunez-Yanez, M. Hosseinabady, and A. Beldachi, "Energy Optimization in Commercial FPGAs with Voltage, Frequency and Logic Scaling," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1484–1493, Feb. 2016, https://doi.org/10.1109/TC.2015.2435771.

[36] M. Hosseinabady and J. L. Nunez-Yanez, "Run-time power gating in hybrid ARM-FPGA devices," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Munich, Germany, Sep. 2014, pp. 1–6, https://doi.org/10.1109/FPL.2014.6927503.