Turkish Journal of Electrical Engineering & Computer Sciences

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

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 3300 – 3318
© TÜBİTAK
doi:10.3906/elk-1608-94

# FPGA-based SOC for hardware implementation of a local histogram-based video shot detector

**Abdessalem BEN ABDELALI***, **Mohamed Nidhal KRIFA**, **Abdellatif MTIBAA**
Laboratory of Electronics and Microelectronics, University of Monastir, Monastir, Tunisia

**Received:** 10.08.2016 • **Accepted/Published Online:** 30.12.2016 • **Final Version:** 30.07.2017

**Abstract:** In this paper, we present a video application example and its implementation in a reconfigurable system-on-chip (SOC) platform. The proposed platform employs the benefits of field programmable gate array (FPGA) technology. A prototype based on a Xilinx Virtex-5 FPGA is developed. The application includes a video shot boundary detection module based on the local histogram (LH) technique. Diverse hardware module versions corresponding to different quantization levels and architectural solutions for an LH-based shot detection system are presented. The developed modules have different hardware resource occupations and can be used in a dynamic way to allow flexible management of the target hardware system. They also show high execution time efficiency and can reach an important bandwidth that can support the most recent high-resolution video formats with a high rate of frames per second. A complete SOC-based demonstration for video summarization based on the LH is also developed. It includes the LH extraction, shot boundary detection, and key frame visualization.

**Key words:** Hardware implementation, field programmable gate array, system on chip, partial dynamic reconfiguration, real time, local histogram, shot boundary detection, video summarization

## 1. Introduction

New generations of embedded video systems such as interactive TV and multimedia devices are characterized by the parallel application of a multitude of video processing techniques to assure a higher quality of service. The applied techniques have to be performed in real time and with high efficiency. In this case, the use of dedicated hardware systems with high processing power becomes a necessity. System flexibility and adaptability is equally important. Here field programmable gate array (FPGA) technology represents the most efficient solution. It offers a high level of flexibility while also preserving high processing power. FPGAs deliver application-specific integrated circuit (ASIC)-like density and performance, while their flexibility and operational characteristics offer distinct advantages over ASIC. Thanks to the innovative architectures of FPGAs, with embedded processors, memory blocks, and DSP blocks, more designers are resorting to these devices for system-on-chip (SOC) designs.

Video content analysis and indexing provide important services in new interactive TV and multimedia devices. They are applied in a wide range of advanced applications [1–3]. The main treatments developed in this field can be categorized into temporal, spatial, and spatiotemporal segmentations. These treatments involve a large variety of complex techniques and necessitate high processing power. Real-time video content analysis and its integration in embedded systems are also becoming a focus of research [2,4–6]. Video content analysis techniques are used in constrained applications [2,3] like real-time content delivery, interactive TV,

*Correspondence: abdessalem.benabdelali@enim.rnu.tn surveillance systems, security systems, and personal video recorders. In these cases, specialized tools for video analysis must work not only with stored data but also with live data broadcasted through high-speed means such as digital cable or data from telesurveillance systems. Moreover, in actual content description systems, video scene analysis is based on a set of features (color, texture, and motion). In this case, the system has to implement different descriptors relating to these different features and has to support multitasks and parallel treatments.

In this work, we are interested in video shot boundary detection, which is an essential task in video analysis and indexing applications [7,8]. It is utilized for video summarization, which gives users the opportunity to browse and select video documents of their choice for later viewing. Our objective is to design a real-time video summarization system based on the local histogram (LH) technique. A new and highly efficient hardware implementation of the LH-based shot detection system supporting real-time constraints of high definition (HD) video formats is proposed. The purpose of our work is also to realize a complete FPGA-based demonstration, including LH extraction, shot boundary detection, and key frame visualization.

An important aspect to be taken into consideration in our design is hardware system adaptability, which becomes possible thanks to the FPGA's dynamic partial reconfiguration (DPR) technique. We suggest various implementation solutions for the LH in order to offer different hardware modules with different characteristics in terms of hardware resource occupation. These modules can be selected and loaded on the FPGA depending on application requirements and hardware resource availability. A possible application scenario is illustrated in Figure 1.

Partial dynamic reconfigurable regions are defined in the FPGA to dynamically implement different hardware modules depending on the application and on the system requirements. The purpose here is to ensure a high level of flexibility and to essentially optimize the reserved hardware resources. For example, the LH can be applied in parallel with other video processing techniques to ensure the required services or it can be combined, depending on the video type, with other techniques for video summarization so as to ameliorate the detection quality. In these cases, a sufficiently large reconfigurable region must be reserved to be able to integrate the different simultaneous functionalities. Intelligent management of hardware resources is necessary to optimize the hardware area of the defined regions, to minimize resource wastage, and to meet the external requirements (such as the remaining energy). The possibility of selecting from different architectures with different types of resource occupation can be a solution to minimize the reserved resources. For example, when the LH module needs to be integrated with another one, necessitating a particular type of resources (block RAM (BRAM), digital signal processor (DSP) block, or configurable logic block (CLB)), we load the LH architecture with the minimum use of these type of resources. However, when dealing with energy consumption, for example, a different version of the LH (corresponding to a lower number of quantization levels or supporting lower resolution images) with fewer resources and lower energy consumption can be loaded.

The rest of this paper is organized as follows. In Section 2, we present the specification of the LH technique and its application for shot boundary detection. The different experimentation and evaluation results of the LH for shot boundary detection are given in this section. The suggested hardware architectures of the LH and its implementation results are provided in Section 3. In Section 4, the architecture of the proposed SOC integrating the LH hardware module for video summarization and its static and dynamic implementation are studied. The paper is concluded in Section 5.

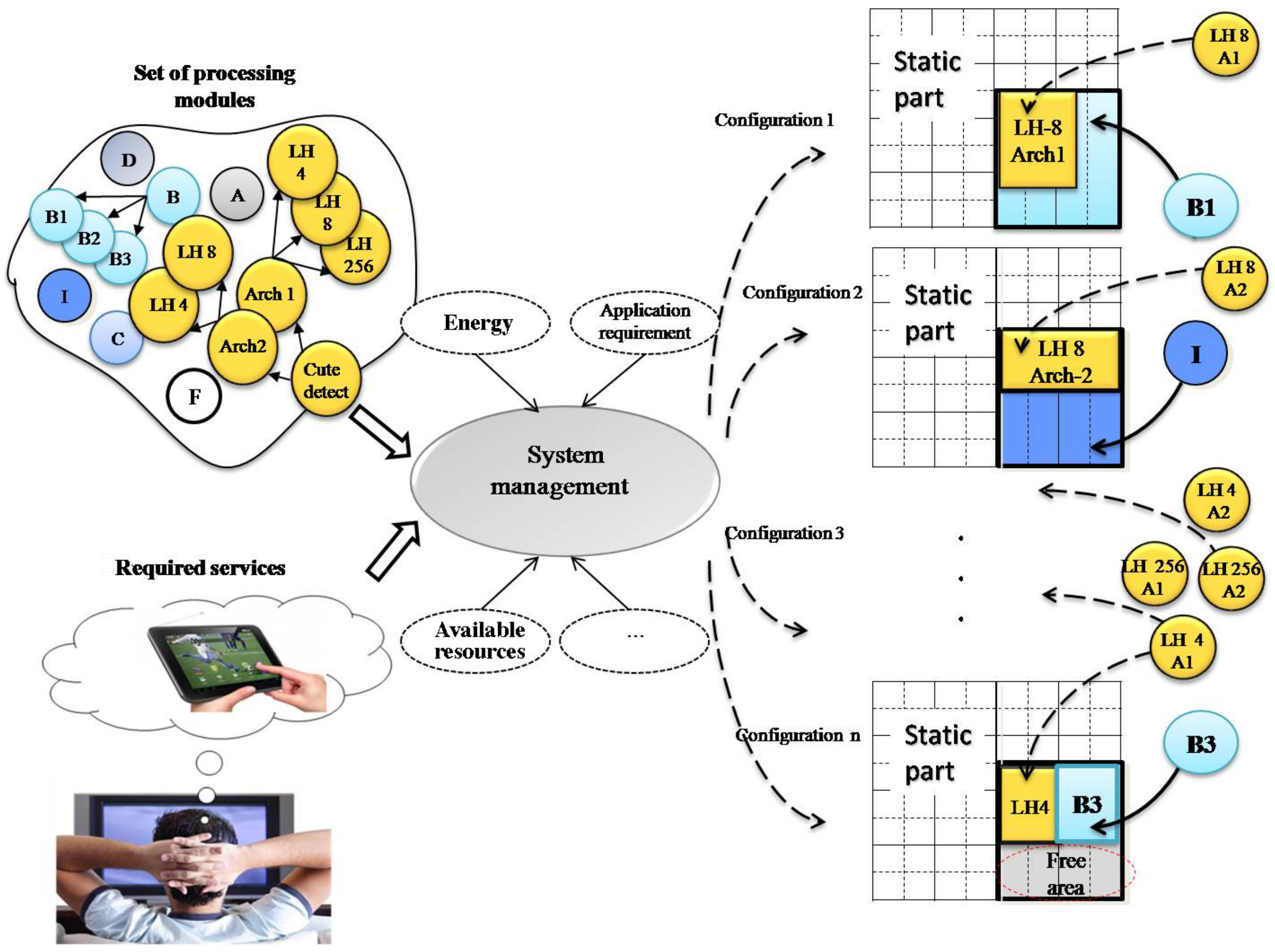

**Figure 1.** Dynamic application scenario of LH-based cut detector.

## 2. Local histogram for temporal video segmentation

Before any video content-based manipulation, an important step to be performed is the extraction of the hierarchical structure of the video sequence. The standard hierarchical video structure consists of frames, shots, and scenes. The most important video unit to be extracted is the shot. Shot boundary detection is then the main step in shot-based temporal video segmentation [8,9]. In a produced video, shots are separated by different types of transitions (or boundaries), either abrupt or progressive. Abrupt shot change, also called a cut, is the most used transition.

Short boundary detection is mainly based on computing the visual difference between consecutive frames using low-level image features. Several methods have been used to compute this difference, which expresses the visual discontinuity between consecutive video frames to indicate a possible shot transition. In this work, a method based on the LH is used for video shot cut detection. Even though the histogram is an old method, it is still one of the most commonly utilized and reliable methods in the literature [10–13]. The study presented in [7] in 2015 was proposed to help designers find the most popular features and techniques in shot boundary detection. Fifty research papers were studied from 1996 to 2014 to calculate the utilization popularity. This study demonstrated that the most popular feature to find the change in shot was the histogram for gray or colored images. Many recent studies have been based on the histogram technique [12,14,15]. In addition, the local feature can greatly increase the effectiveness of the histogram method [12,14,16–18], which makes it one of the most suitable methods.

### 2.1. LH application principle for temporal video segmentation

The process of shot boundary detection based on the adopted LH method is described in Figure 2. It includes the following steps: first, converting the image from the RGB color space to the grayscale and quantizing the obtained values; second, dividing the image into 4 × 4 blocks; third, calculating the local histogram of the current image; fourth, calculating the local histogram difference between image i and image i – 1, and, lastly, comparing the histogram difference to a threshold value and making a decision on cut detection.

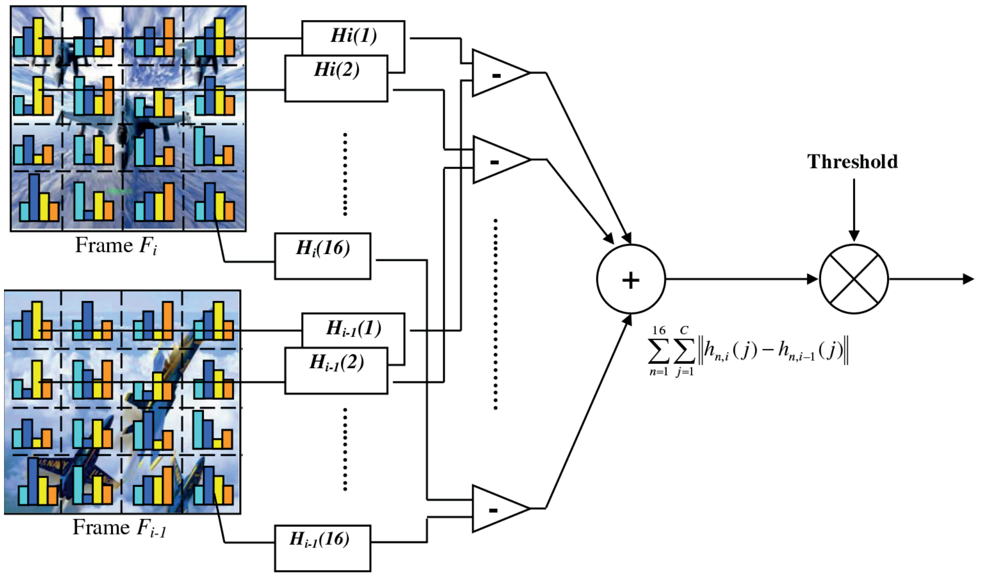

**Figure 2.** Block diagram of cut detection process based on LH technique.

The LH is applied directly on grayscale images with 256 gray levels and then after a quantization phase. A uniform quantization into 8 or 4 levels is applied. Table 1 gives the intensity ranges and the corresponding quantization values for 8 and 4 quantization levels. We show that the value of the quantized intensity corresponds to 3 or 2 mostly significant bits, respectively to 8 and 4 quantization levels.

Shot boundary detection is based on the color histogram difference as a measure of discontinuity. The difference between image i and image i – 1 is measured at two levels. The first one is the region level (the distance between two regions) and the second one is the image level (the measure of the overall similarity between two images with 16 regions). Each region in frame *Fi*, which is represented by $C$ dimensional vectors $(h_n(1),\ h_n(1),...,h_n(C))$, is compared to the corresponding one in frame $F_{i-1}$ by using the L1 distance. The difference measure between two consecutive frames is defined as:

$$FDM(F_i,F_{i-1})=\sum_{n=1}^{16}\|d(F_{n,i},F_{n,i-1})\|=\sum_{n=1}^{16}\sum_{j=1}^{C}\|h_{n,i}(j)-h_{n,i-1}(j)\|, \qquad (1)$$

where $F_{n,i}$ is the block (region) N$^\circ$n of frame $F_i$, $F_{n,i-1}$ is the block (region) N$^\circ$n of frame $F_{i-1}$, $h_{n,i}$ is the local histogram of the block N$^\circ$n of frame $F_i$, and $h_{n,i}$ is the local histogram of the block N$^\circ$n of frame $F_{i-1}$.

**Table 1.** Gray quantization.

| Intensity ranges (decimal) | Intensity ranges (decimal) | Quantized value (decimal) | Quantized value (decimal) |
|---|---|---|---|
| 4 quantization levels | | | |
| [0, 64[ | **00**000000, . . . , **00**111111 | 0 | 00 |
| [64, 128[ | **01**000000, . . . , **01**111111 | 1 | 01 |
| [128, 192[ | **10**000000, . . . , **10**111111 | 2 | 10 |
| [192, 256[ | **11**000000, . . . , **11**111111 | 3 | 11 |
| 8 quantization levels | | | |
| [0, 32[ | **000**00000, . . . , **000**11111 | 0 | 000 |
| [32, 64[ | **001**00000, . . . , **001**11111 | 1 | 001 |
| [64, 96[ | **010**00000, . . . , **010**11111 | 2 | 010 |
| [96, 128[ | **011**00000, . . . , **011**11111 | 3 | 011 |
| [128, 160[ | **100**00000, . . . , **100**11111 | 4 | 100 |
| [160, 192[ | **101**00000, . . . , **101**11111 | 5 | 101 |
| [192, 224[ | **110**00000, . . . , **110**11111 | 6 | 110 |
| [224, 256[ | **111**00000, . . . , **111**11111 | 7 | 111 |

### 2.2. Experimentation and evaluation results

To evaluate the LH-based method for shot boundary detection, we use a video corpus of 36 video sequences from six different types of videos: advertisements, cartoons, documentaries, movies, football, and news. The total duration (in seconds) and the number of shots corresponding to each video genre are as follows: advertisements: 120 s/100 shots, cartoons: 180 s/93 shots, documentaries: 420 s/111 shots, movies: 300 s/71 shots, football: 240 s/57 shots, and news: 480 s/124 shots. Precision, recall, and false positive (FP) measures are used to evaluate shot boundary detection performances.

$$\text{Precision} = \text{Correct}/\left(\text{Correct} + \text{False}\right) \tag{2}$$

$$FP = False/(Correct + False) \tag{3}$$

$$\text{Recall} = \text{Correct}/(\text{Correct} + \text{Missed}) \tag{4}$$

Here “correct”, “false”, and “missed” correspond respectively to the number of correctly reported transitions, false detections, and missed transitions.

The obtained results for the different evaluation tests are given in Table 2. This table represents the performances of the LH for different video sequence types and for different quantization levels. Figure 3a depicts an example of histogram distance results for 4 and 8 quantization levels. These results give an idea about the sensitivity of the LH to shot transitions. Figure 3b shows the recall and precision results compared to different existing methods based on the well-known MPEG 7 descriptors. The following descriptors are tested on the video corpus: edge histogram (EH), motion activity (MA), color structure descriptor (CSD), and color layout descriptor (CLD). The represented recall and precision curves demonstrate the superiority of the LH-based method in terms of detection rate with a satisfactory level of precision. This is true for the different LH-based method versions relative to the different applied quantization levels: 256 levels (LH-256), 8 levels (LH-8), and 4 levels (LH-4).

**Table 2.** Performance of LH for 256, 8, and 4 gray levels.

| | LH for 256 gray levels | | | LH for 8 quantization levels | | | LH for 4 quantization levels | | |
|---|---|---|---|---|---|---|---|---|---|
| Video genre | Precision | Recall | FP | Precision | Recall | FP | Precision | Recall | FP |
| News | 96% | 100% | 4% | 94% | 100% | 6% | 95% | 100% | 5% |
| Football | 98% | 97% | 2% | 96% | 96% | 4% | 83% | 96% | 7% |
| Documentaries | 83% | 89% | 17% | 79% | 91% | 21% | 70% | 91% | 30% |
| Movies | 89% | 95% | 11% | 85% | 95% | 15% | 84% | 97% | 6% |
| Advertisements | 70% | 68% | 30% | 69% | 69% | 31% | 67% | 73% | 33% |
| Cartoons | 78% | 92% | 22% | 76% | 93% | 24% | 76% | 81% | 24% |

## 3. Local histogram hardware architecture and implementation

### 3.1. Hardware architecture description

The global system architecture is given in Figure 4. It consists of the following blocks: color conversion (RGB to gray), region detection, address calculation, and histogram update. We can assume that the input image is initially stored in the memory, and an address counter is used for pixel reading.

The address calculation block, which permits addressing the histogram bin to be updated, comprises one adder and one multiplier. The histogram bin address corresponding to the current pixel is calculated as follows:

$$\text{Address} = (R_n \times C) + c_q \tag{5}$$

where $Rn$ is the current pixel region number, $C$ is the number of quantization levels, and $c_q$ is the quantization level of the current pixel.

The memory histogram (registers or memory) is constituted of ($C\times$ $16$) locations. Every memory block containing consecutive $C$ locations corresponds to a local histogram relative to one of the 16 regions of the image. The address of the histogram bin to be updated depends on the current pixel region number and quantization level. The address bus width $A_{bw}$ is defined as follows:

$$2^{A_{bw}} = C \times 16 \tag{6}$$

Memory cell width ($n$) depends on image size ($W \times H$). It is calculated as:

$$2^n \geq (W/4) \times (H/4) \tag{7}$$

Memory data width ($n$) and address bus width ($A_{bw}$) corresponding to the different image sizes and quantization levels ($C$) are as follows:

- Memory data width ($n$) only depends on the image size. $n$ = 13 bits, 14 bits, 15 bits, 17 bits, and 17 bits, respectively, for image sizes 320 × 240, 512 × 512, 640 × 480, 1280 × 1024, and 1920 × 1080.
- Address bus width ($A_{bw}$) only depends on the number of quantization levels. ($A_{bw}$) = 12 bits, 7 bits, and 6 bits, relatively, for 256, 8, and 4 quantization levels.

The color space conversion from RGB to gray is performed according to Eq. (**??**), and the architecture of the corresponding block is given in Figure 4.

$$Y = 0.257 \times R + 0.504 \times G + 0.098 \times B + 16 \tag{8}$$

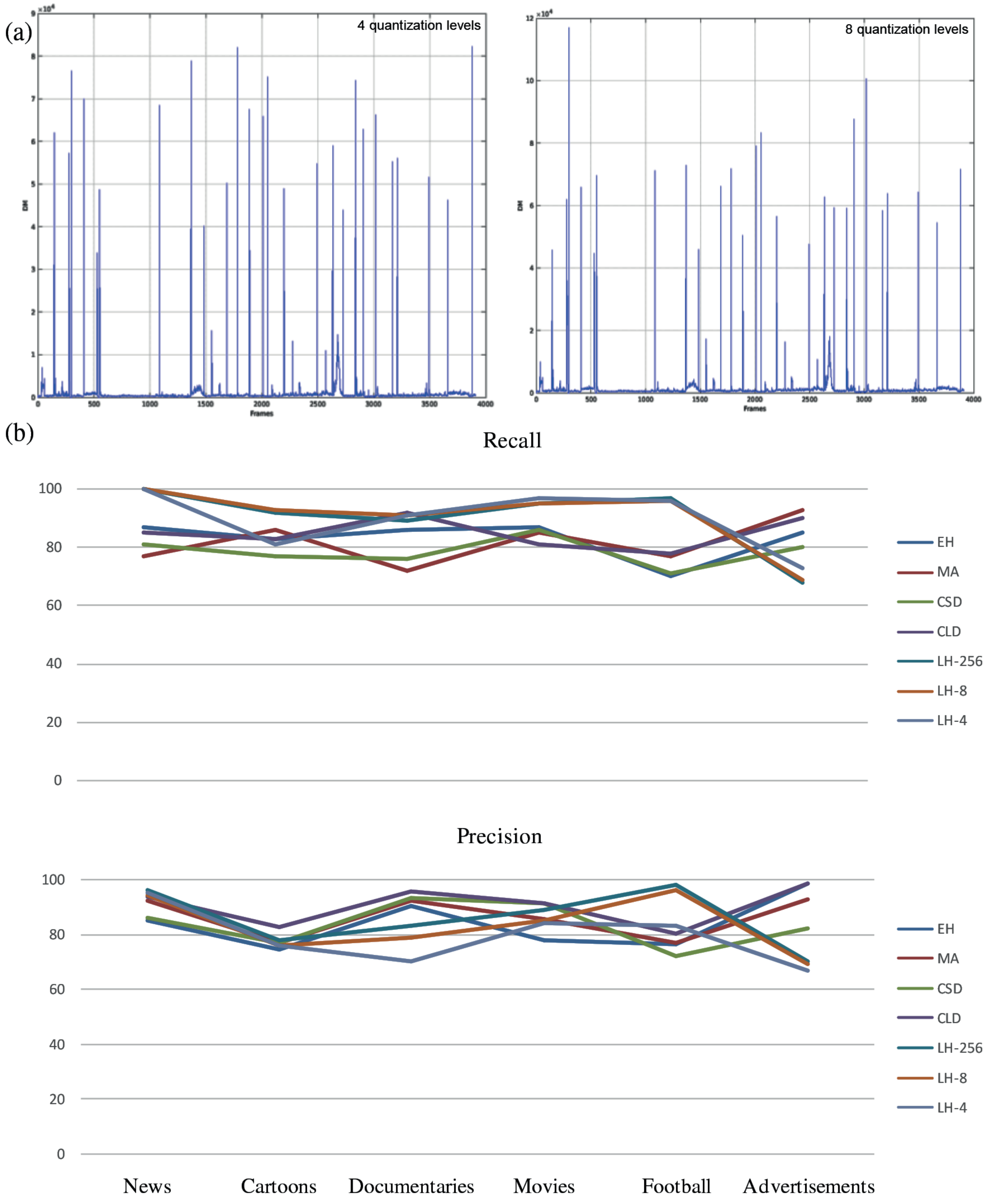

**Figure 3.** LH performances: a) LH sensibility, b) recall/precision compared to existent methods.

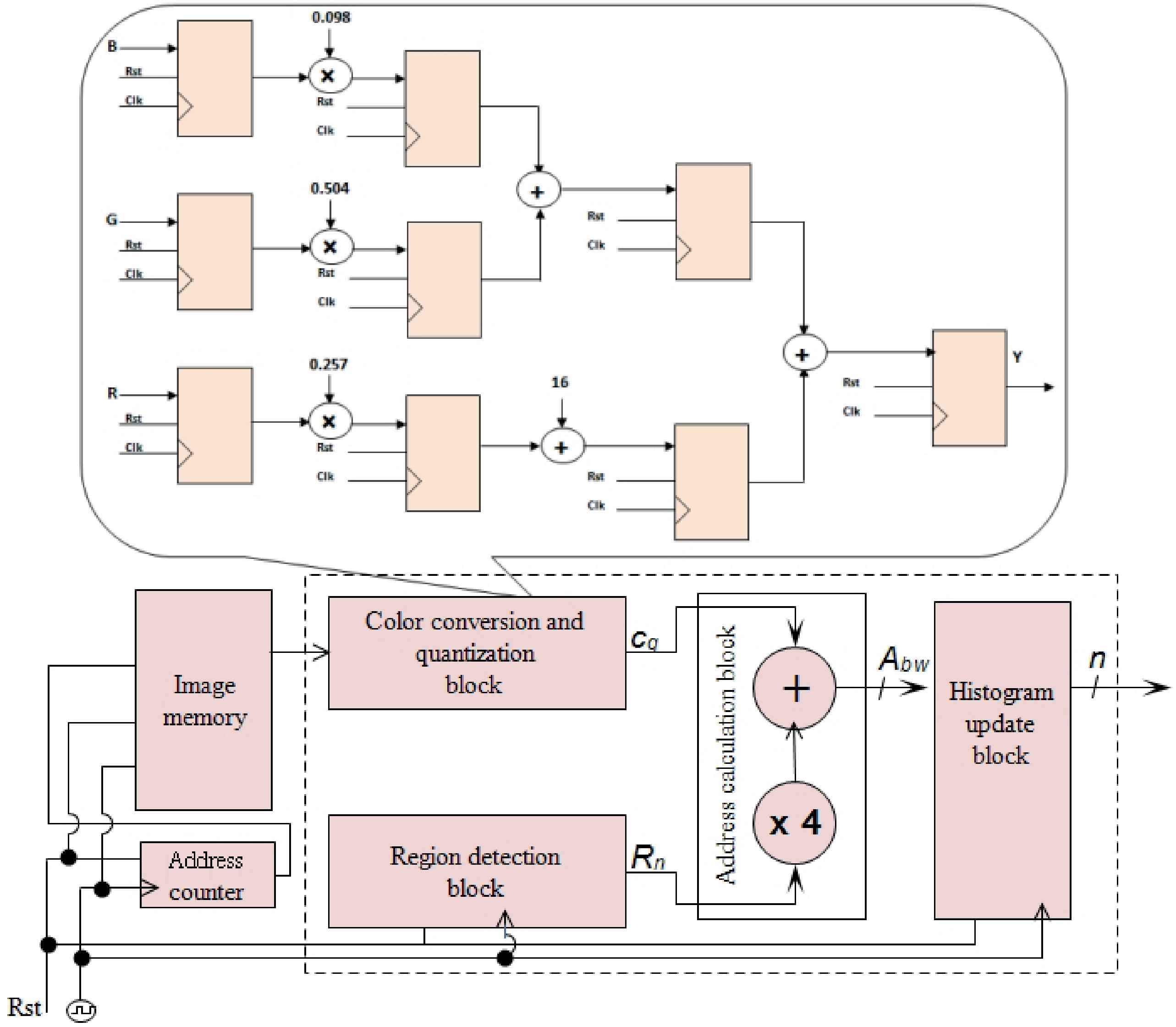

**Figure 4.** Global system architecture with color conversion block.

The architecture of the region detection block, which permits a determination of the current pixel image region number, is shown in Figure 5. The outputs $(i)$ and $(j)$ of the column and line counters (Figure 5) correspond to the column and line numbers of the current pixel. $(i)$ and $(j)$ are compared, respectively, to the $x_i$ ($x_0$, $x_1$, $x_2$, $x_3$) and $y_i$ ($y_0$, $y_1$, $y_2$, $y_3$) values to determine the pixel region number. $x_i$ and $y_i$ values correspond to the region limits. Enable (EN) input of the column counter is connected to the "start" input of the system, while the EN input of the line counter is connected to the output of the column counter. Each time the column counter reaches the value $x_3-1$ (the image length), the line counter is incremented.

The comparator blocks are used to determine the number of the region corresponding to the current pixel. For example, if $x_0 < i \leq x_1$ and $y_1 < j \leq y_2$, then the region number is 9. The output of each comparator block is 1 bit, which takes the 1 value if the corresponding condition is true and the 0 value if the condition is false. The outputs of the 16 blocks (16 bits) are fed to the input of a 16-to-4 encoder whose output gives the number of the image region corresponding to the current pixel.

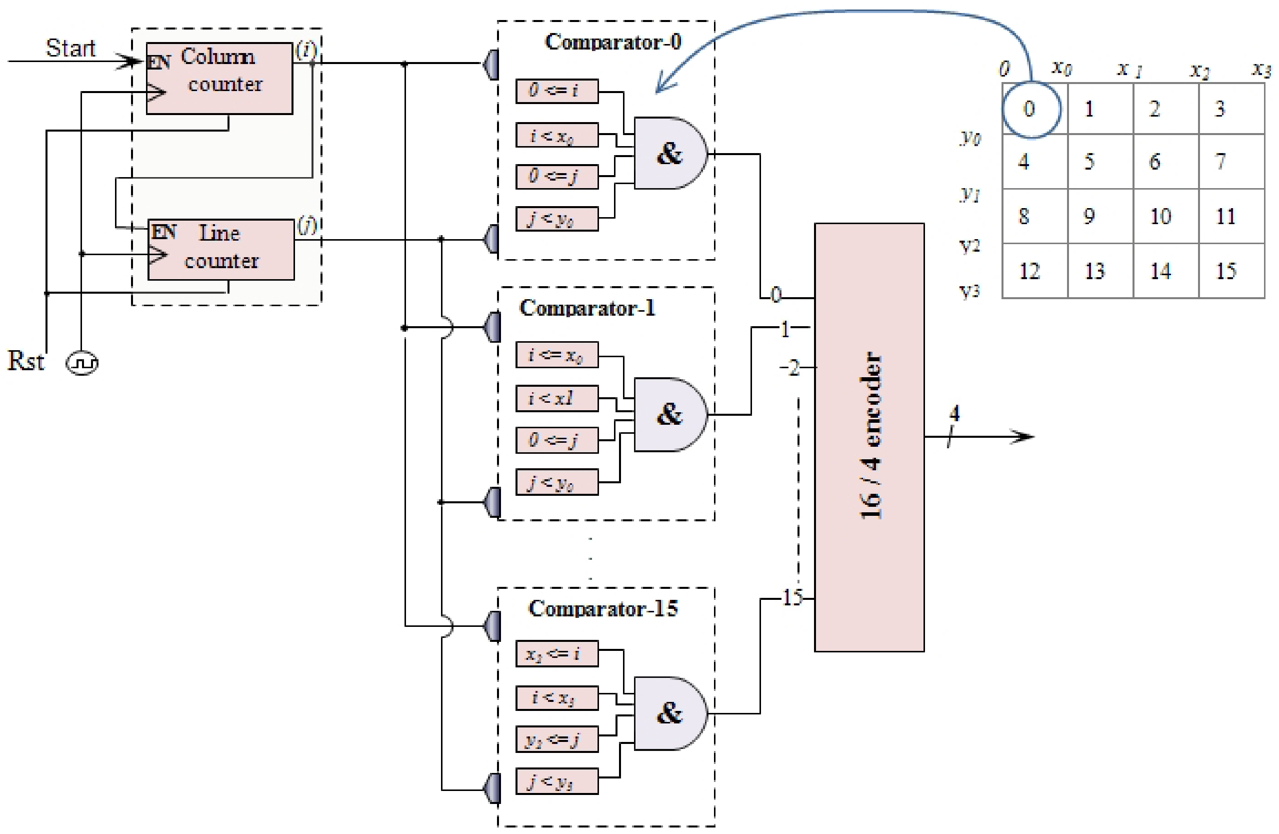

**Figure 5.** Hardware architecture of region detection module.

The histogram update block is composed of the histogram memory and the update mechanism. This block can be realized by using a memory block with ($C\times\ 16$) locations or a set of ($C\times\ 16$) register counters. To correctly realize the update phase, the value of the memory location specified by the input address must be read, incremented, and written in the same cycle. For a memory-based architecture, this can be assured by utilizing a dual-port memory. In the case of a register-based architecture, no synchronization mechanism is necessary, as asynchronous outputs corresponding to the selected location are immediately available to be incremented. However, the writing will take place at the next clock rising edge. The two implementation solutions of this block are illustrated in Figure 6: register-based and memory-based architectures. The represented register-based architecture corresponds to the example of 4 quantization levels.

For the memory-based architecture, the address calculation block is used to address the histogram memory locations. For the register-based architecture, the system directly takes the region number and the input pixel gray level as an input, and by a simple logic test, only the histogram bin counter corresponding to the actual image region will be activated. Each counter will be incremented if its input ("In") is at "1" logic and the EN signal is active. For example, counter N$^\circ$3 will be activated for quantization level 3 and region number 0. Counter N$^\circ$63 will be activated for quantization level 3 and region number 15. For this solution, the 16-to-4 encoder of the region detection block can also be eliminated, and the input "region N$^\circ$" and the associated AND gate will be replaced by the output bit of the corresponding comparator.

MUX-1 and MUX-2 of the memory-based architecture serve to select between the histogram accumulation mode and the distance calculation one. The first input of MUX-2 (address) corresponds to the address generated

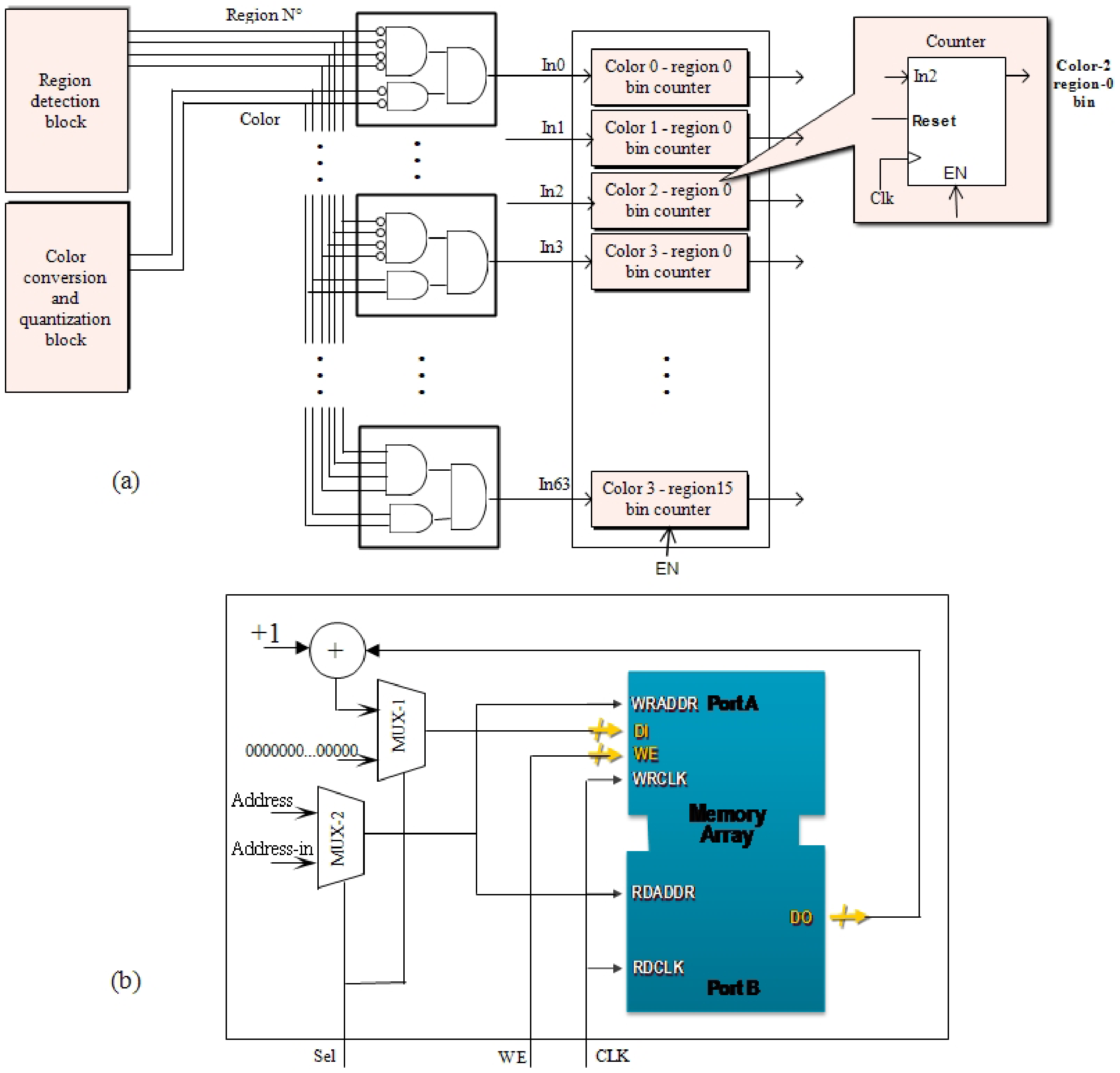

**Figure 6.** Histogram update block architecture: a) register-based, b) memory-based.

by the address calculation block to select the memory location to be incremented. The second one (address-in) corresponds to the address that will be generated by the "interhistogram distance" block to read the histogram bins' values. In the accumulation mode, MUX-1 takes the first input that corresponds to the incremented value of the corresponding histogram bin and the second one (zero value) in the distance calculation mode. In the calculation mode, each read histogram bin will be replaced by zero to initialize the histogram for the following frame. For the counter-based architecture, the "Reset" line is used (set to "1" for one cycle) to reinitialize the counter values. The signals "WE", "Sel", "EN", and "Reset" are generated by a simple finite state machine. "WE" or "EN" is activated when the first pixel value and region number are ready and the memory address is calculated. This depends on the color conversion and the region detection blocks' latency. The "Sel" signal is set to "1" logic at the end of the histogram calculation and to "0" at the end of the distance calculation.

### 3.2. Implementation results and comparison

The local histogram algorithm is synthesized using the Xilinx ISE tool for the XC5VLX50T FPGA. The proposed architectures are implemented for different quantization levels (4, 8, and 256) and different image sizes. By using the Xilinx timing analyzer for static timing analysis, the maximal frequency is determined for the different implemented architectures.

Tables 3 and 4 give the hardware resource occupation and frequency results for the counter-based architecture and the memory-based one, respectively. To conclude, the use of a lower number of quantization levels has a significant effect on the hardware resource occupation. For the counter-based architecture, the number of counters increases with the rise in the number of gray levels. In this case, the synthesis results for 256 gray levels are not represented, as the necessary number of registers exceeds the capacity of the utilized FPGA. For the memory-based architecture, more BRAMs are used when the number of quantization levels goes up.

**Table 3.** Hardware resource occupation for register-based architecture.

| Image size Image size | Quantization levels (C) | Slice registers | Slice LUTs | BRAM 18K | BRAM 36K | DSP48Es | Frequency |
|---|---|---|---|---|---|---|---|
| 320 × 240 | C = 4 | 957 | 1079 | 0 | 0 | 1 | 341 |
| | C = 8 | 1790 | 1979 | 0 | 0 | 1 | 341 |
| 512 × 512 | C = 4 | 1022 | 1144 | 0 | 0 | 1 | 318 |
| | C = 8 | 1919 | 2108 | 0 | 0 | 1 | 318 |
| 640 × 480 | C = 4 | 1086 | 1208 | 0 | 0 | 1 | 317 |
| | C = 8 | 2047 | 2236 | 0 | 0 | 1 | 317 |
| 1280 × 1024 | C = 4 | 1217 | 1333 | 0 | 0 | 1 | 307 |
| | C = 8 | 2306 | 2489 | 0 | 0 | 1 | 307 |
| 1920 × 1080 | C = 4 | 1217 | 1328 | 0 | 0 | 1 | 305 |
| | C = 8 | 2306 | 2484 | 0 | 0 | 1 | 305 |

**Table 4.** Hardware resource occupation for a RAM-based architecture.

| Image size Image size | Quantization levels (C) | Slice registers | Slice LUTs | BRAM 18K | BRAM 36K | DSP48Es | Frequency |
|---|---|---|---|---|---|---|---|
| 320 × 240 | C = 4 | 123 | 191 | 1 | 0 | 1 | 341 |
| | C = 8 | 124 | 192 | 1 | 0 | 1 | 341 |
| | C = 256 | 129 | 194 | 1 | 1 | 1 | 224 |
| 512 × 512 | C = 4 | 124 | 192 | 1 | 0 | 1 | 318 |
| | C = 8 | 125 | 193 | 1 | 0 | 1 | 318 |
| | C = 256 | 130 | 195 | 0 | 2 | 1 | 222 |
| 640 × 480 | C = 4 | 124 | 193 | 1 | 0 | 1 | 317 |
| | C = 8 | 125 | 194 | 1 | 0 | 1 | 317 |
| | C = 256 | 130 | 196 | 0 | 2 | 1 | 221 |
| 1280 × 1024 | C = 4 | 127 | 191 | 1 | 0 | 1 | 307 |
| | C = 8 | 128 | 192 | 1 | 0 | 1 | 307 |
| | C = 256 | 133 | 193 | 0 | 2 | 1 | 219 |
| 1920 × 1080 | C = 4 | 127 | 188 | 1 | 0 | 1 | 305 |
| | C = 8 | 128 | 189 | 1 | 0 | 1 | 305 |
| | C = 256 | 133 | 177 | 0 | 2 | 1 | 219 |

Table 5 represents the execution time (ms) for different image sizes and quantization levels. According to the architecture initiation period (1 cycle) and the obtained frequency results, our architectures can attain a bandwidth of at least 219 megapixels per second (Mpps) for the case of 256 gray levels. With this high throughput, our architecture can support resolutions from VGA (640 × 480) to UXGA (1600 × 1200) and all digital television formats including HDTV (1080i and 720p). For example, the 1080p/60 video format (1920 × 1080 pixels @ 60 Hz refresh) requires a throughput of 148.5 Mpps, which is notably lower than the throughput supported by our architectures.

**Table 5.** Execution time (ms) for different image sizes and quantization levels.

| Quantization levels (C) | Image size | | | | |
|---|---|---|---|---|---|
| | 320 × 240 | 512 × 512 | 640 × 480 | 1280 × 1024 | 1920 × 1080 |
| C = 4 | 0.225 | 0.824 | 0.969 | 4.269 | 6.799 |
| C = 8 | 0.225 | 0.824 | 0.969 | 4.269 | 6.799 |
| C = 256 | 0.343 | 1.181 | 1.390 | 5.985 | 9.468 |

We demonstrate that the number of regions and the use of more color components (R, G, and B) do not have an effect on execution time. In fact, the critical path remains the same, and the maximum frequency does not change. In terms of hardware resources, only the LH update block will be affected. For the three color components, three memory blocks will be used in parallel. The subdivision of the image to a bigger number of regions only affects the number and width of the LH memory locations.

A comparison with the existing hardware shot detection systems is provided in Table 6. The results indicate that our design ensures the best execution time and hardware resource occupation performances for a satisfactory level of accuracy.

**Table 6.** Comparison with existing hardware shot detectors.

| Ref | Method | Parameters | Device | Image format | Frequency (MHz) | Occupation |
|---|---|---|---|---|---|---|
| [10] | Hist L-1 + SVM | 5 bits gray | Virtex IV XC4VX35 | - | 251 | - |
| [11] | Local Hist L-1 | 2 bits gray | Virtex XSV800 | CIF 352 × 288 | 222 | 2 BRAM-4kb/112 FFs / 268 Luts |
| [4] | Color Structure Descriptor (CSD) | 8 bits HMMD | Virtex 5 XC5VLX50T | 1920 × 1080 | 186 | 4 BRAM-36K/6850 FFs / 8657 Luts |
| [4] | // | 3 bits HMMD | // | // | 210 | 4 BRAM-36K / 821 FFs / 973 Luts |
| Ours | Local Hist L-1 | 8 bits gray | // | // | 219 | 2 BRAM-36K / 133 FFs / 177 Luts |
| Ours | Local Hist L-1 | 3 bits gray | // | // | 305 | 1 BRAM-18K / 128 FFs / 189 Luts |

## 4. SOC for video summarization based on LH

### 4.1. System description

The functioning of the proposed system is depicted in Figure 7. Two memory areas are used to alternatively store two consecutive video frames ($F_i$ and $F_{i-1}$) to be able to assure a pipeline between video acquisition and the LH extraction process. The LH will be applied each time on the already stored frame. The LH vector will be transferred to register set 1 (dedicated to storing the LH of frame $F_i$) after each frame processing, and the distance calculation starts. In our design, we apply the example of 4 quantization levels. For this case, the shot detection step takes 65 clock cycles: one cycle for the vector transfer and 64 cycles for the distance calculation ($d(h_i,\ h_{i-1})$). The distance calculation is made by using one adder, one subtractor, and one register to store the result. When we finish the distance calculation, the LH vector stored in register set 1 will be transferred to register set 2 (dedicated to store the LH of frame $F_{i-1}$), which will be used for the next distance calculation.

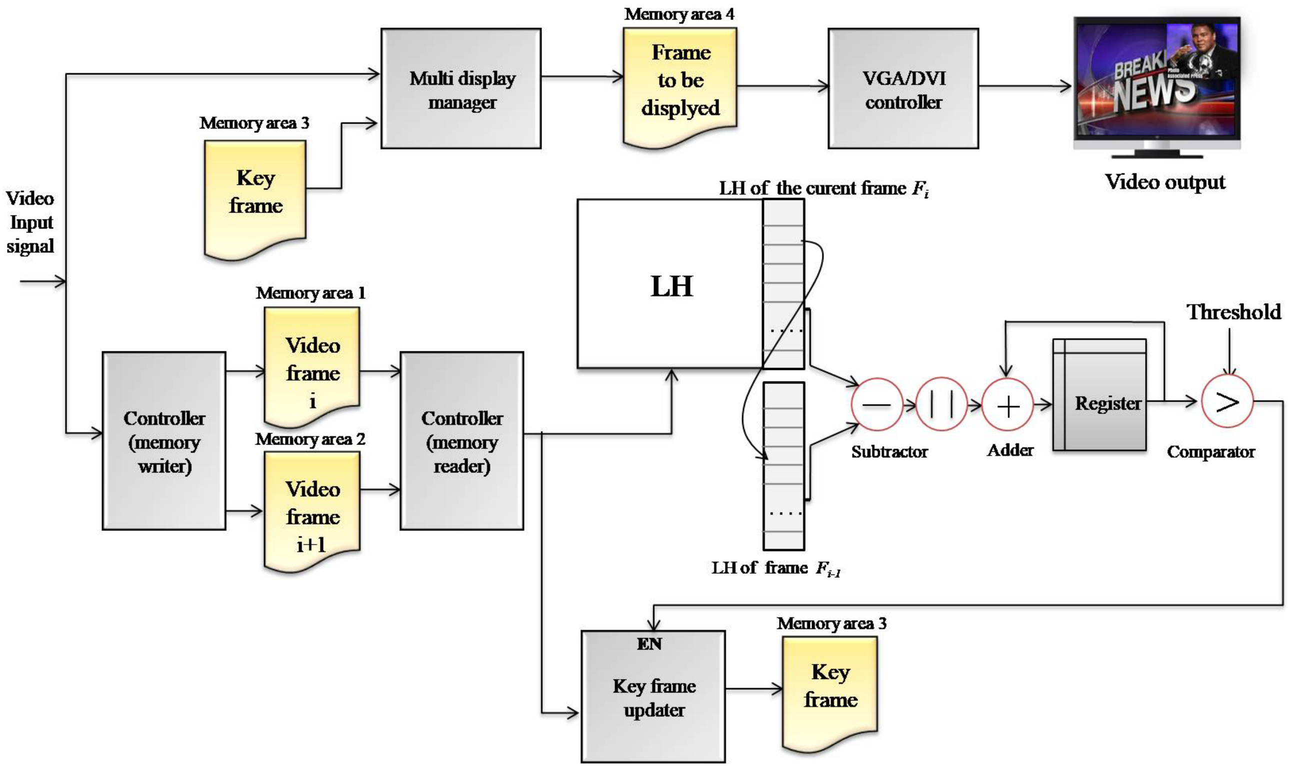

**Figure 7.** System description.

If $d(h_i,\ h_{i-1}) > \alpha$, where $\alpha$ is a predefined threshold, a scene change will be detected between frames $F_i$ and $F_{i-1}$, and the EN signal of the key frame updater block will be activated. In this case, the key frame updater block will store the coming frame (frame $F_{i+1}$) in the specified memory area dedicated to store the key frame (memory area 3). The multidisplay manager gives as an output a combined frame (current frame + key frame) that will be stored in memory area 4 to be displayed using the thin film transistor (TFT) controller.

### 4.2. Proposed SOC architecture and implementation results

The suggested SOC architecture (Figure 8) is mainly based on a MicroBlaze 32-bit microprocessor and the following important resources: embedded memory blocks, an external DDR SDRAM memory, a multiport memory controller (MPMC), an XPS TFT controller, and various other interfaces such as UART, System ACE Compact Flash, and XPS IIC. To these standard components, we add the LH core, the key frame updater, the multidisplay manager, and the memory access controller modules. We implement the entire platform on the Xilinx Virtex 5 ML505 development system.

The MPMC represents the central component in our design. It allows multiple buses to be connected to the same piece of memory through different ports. The ports are able to access the memory in a parallel way. A crucial MPMC feature is the ability to handle all clock domain crossing issues between system components.

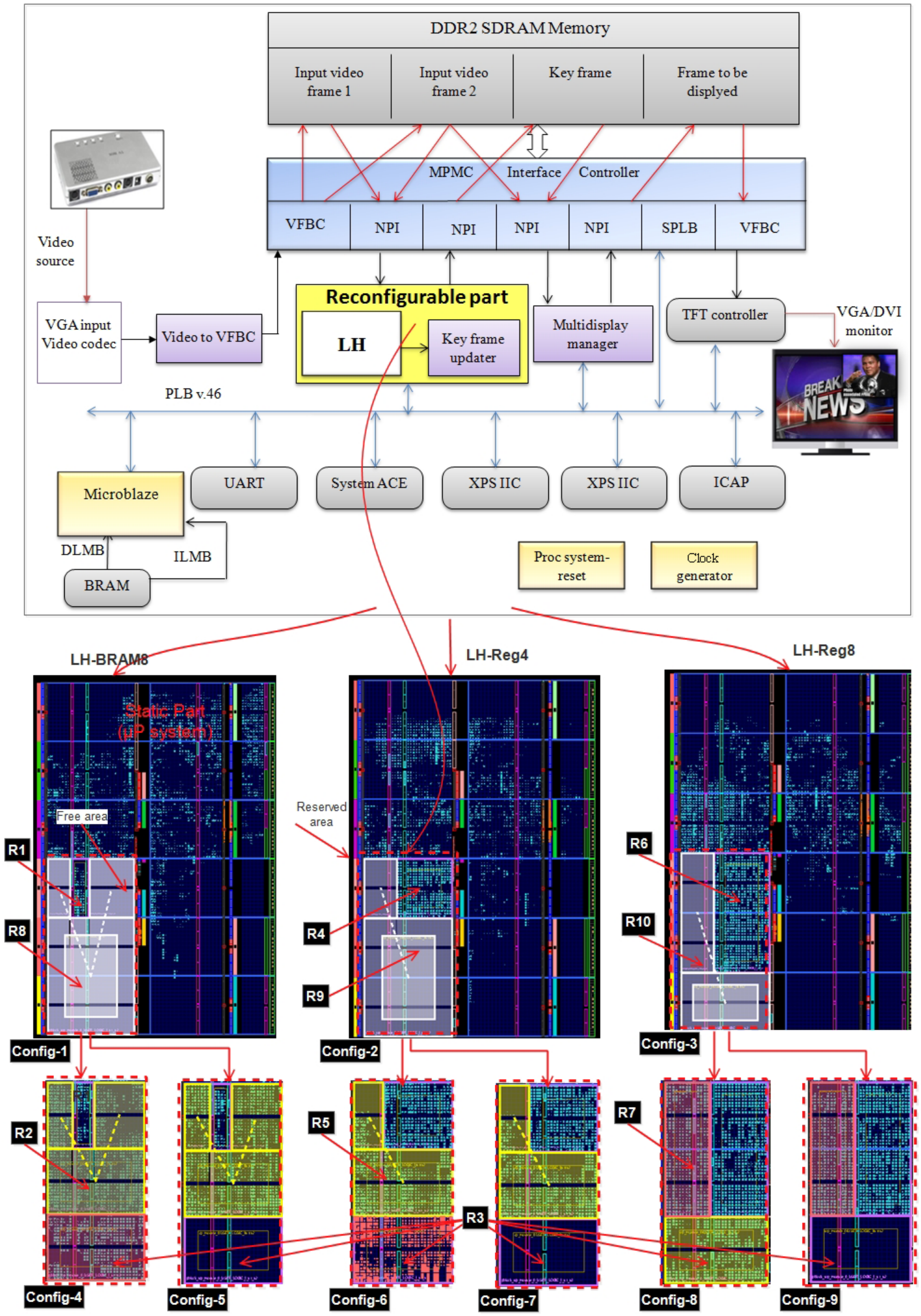

**Figure 8.** SOC architecture and implementation using DPR.

The MPMC consists of an eight-port, where each one can be chosen from a set of personality interface modules. The microprocessor is connected to the MPMC through the processor local bus and the native port interface (NPI). The video input and output modules are connected by using the video frame buffer controller interface. The LH, the key frame updater, and the multidisplay manager are connected to the MPMC through the NPI interface.

In Table 7, we present the synthesis results obtained using the Xilinx EDK tool. The hardware resources utilization before and after the integration of the LH core is presented. These results correspond to the static implementation of the system for a register-based LH module with 4 quantization levels and 640 $\times$ 480 image resolution. The system frequency is set to 100 MHz for the experiment. An implementation using the FPGA DPR technique is also performed to show the possibility and the benefit of a dynamic application of the developed LH hardware modules. The design is decomposed into a static part, which is composed of the processor system and a dynamically reconfigurable part to implement the LH-based shot detector. The system implementation results obtained using the PlanAhead tool for three different LH module versions are presented in Figure 8: a register-based LH module for 8 quantization levels (LH-Reg8), a register-based LH module for 4 quantization levels (LH-Reg4), and a BRAM-based LH module for 8 quantization levels (LH-BRAM8).

**Table 7.** SOC implementation results.

| Resources | Resource occupation | | Resource utilization | |
|---|---|---|---|---|
| | Before LH integration | After LH integration | Before LH integration | After LH integration |
| Registers | 7994 | 10,062 | 27% | 35% |
| LUTs | 7057 | 8903 | 24% | 31% |
| Memory blocks | 18 | 18 | 30% | 30% |

We define a reconfigurable region in the FPGA to be utilized to implement some specific treatments in a dynamic way. In addition to assuring a high level of flexibility, the main objective here is to save the occupied hardware resources. Intelligent use of the reconfigurable resources is necessary to be able to reach this objective. The number of available resources and the structure of the FPGA are very important factors to be taken into account. The FPGA is organized in frames, which are the smallest reconfigurable units. Three types of frames can be relatively distinguished from the FPGA resources: CLB, BRAM, and DSP frames. The CLB frames are the most available resources, and the BRAM frames are more required then the DSP frames. Each CLB frame in a Virtex-5 FPGA is equivalent to 160 look-up tables (LUTs) and 40 slices. A BRAM frame consists of 4 BRAM36 memory blocks, and a DSP frame is composed of 8 DSP48E blocks arranged vertically.

Table 8 provides the hardware resource occupation results of the reconfigurable region in terms of frames for the three LH module versions as set in the design of Figure 8. We define a partial reconfigurable region (PRR) composed of 42 CLB frames, 3 BRAM frames, and 3 DSP frames. The obtained results show the advantage of using different versions of the LH in terms of resource occupation. One of these solutions will be loaded depending on the application and system requirements: available and required hardware resources. For example, the LH-BRAM8 allows the liberating of the CLB frames, the LH-Reg8 permits the liberating of the BRAM frames, and the LH-Reg4 enables the liberating of BRAM frames with more free CLB ones.

The DPR offers the possibility of flexible management of the application by loading different modules and module versions for one of the following aims:

- Change the running modules: different applications or services.
- Change the running version of the module to adapt it to the application context and the system requirement: (i) change the module version to be able to combine it with different other ones without exceeding

**Table 8.** PRR management.

| Configuration | Resources occupation (frames): required/reserved | | | | | | Corresponding regions | Reconfiguration time (ms) |
|---|---|---|---|---|---|---|---|---|
| | CLB | | BRAM | | DSP | | | |
| Config-1: LH-BRAM8 | 2 | 2 | 1 | 1 | 1 | 1 | R1 | 0.11 ms |
| Config-2: LH-Reg4 | 12 | 12 | 0 | 0 | 1 | 1 | R4 | 0.15 ms |
| Config-3: LH-Reg8 | 16 | 16 | 0 | 0 | 1 | 1 | R6 | 0.29 ms |
| Config-4: LH-RAM8 + CSD32 + EH | 2 + 22 + 9 = 33 | 42 | 1 + 1 + 1 = 3 | 3 | 1 | 3 | R1 + R2 + R3 | 0.84 ms |
| Config-5: LH-Reg4 + CLD + CSD16 | 12 + 12 + 14 = 38 | 42 | 0 + 2 + 1 = 3 | 3 | 1 | 3 | R4 + R5 + R3 | 0.84 ms |
| Config-6: LH-Reg8 + CLD + EH | 16 + 12 + 9 = 37 | 42 | 0 + 2 + 1 = 3 | 3 | 1 | 3 | R6 + R7 + R3 | 0.84 ms |
| Config-7: LH-BRAM8 + CSD32 | 2 + 22 = 24 | 28 | 1 + 1 = 2 | 2 | 1 | 2 | R1 + R2 | 0.56 ms |
| Config-8: LH-Reg4 + CLD | 12 + 12 = 24 | 28 | 0 + 2 = 2 | 2 | 1 | 2 | R4 + R5 | 0.56 ms |
| Config-9: LH-Reg8 + EH | 16 + 9 = 25 | 28 | 0 + 1 = 1 | 2 | 1 | 2 | R6 + R7 | 0.56 ms |

the reserved reconfigurable area; (ii) change the module version or the applied combination to respond to external conditions such as energy consumption.

To clarify the advantages of the DPR technique and to show how the configurations can be managed for the presented design case, we consider the following application scenario example: in the same context of video shot boundary detection and video summarization, we consider the combination of the LH with other different descriptors: the EH, the CLD, and the CSD. The selected combination depends on the input video genre and system requirements (decreasing consumption, liberating resources, etc.). We present the application scenario and the corresponding PRR management method.

In a full functionality mode, a combination of three descriptors is applied ((LH + CSD + EH), (LH + CLD + CSD), or (LH + CLD + EH)). In this case, the selected version of the LH must be the one permitting introduction of the defined set of descriptors within the reserved PRR. The same reasoning is applied to the CSD as we use two different versions of this descriptor (CSD16 for 16 quantization levels and CSD32 for 32) [4]. For these three combinations, the entire reserved PRR is occupied. The corresponding configurations (Config-i) are as follows: Config-4 (LH-BRAM8 + CSD32 + EH), Config-5 (LH-Reg4 + CLD + CSD16), or Config-6 (LH-Reg8 + EH + CLD). To respond to the system requirements and the functioning conditions, a combination of two descriptors or the use of only one (the LH) can be then applied. In this case, only a part of the PRR is occupied and the rest of the PRR can be used for other treatments or left free to minimize dynamic energy consumption. The following configurations corresponding to the combinations of two descriptors are considered: Config-7 (LH-BRAM8 + CSD32), Config-8 (LH-Reg4 + CLD), or Config-9 (LH-Reg8 + EH).

For flexible management of the reconfigurable area during the transition from one functioning mode to another (3 descriptors to 2 and 2 to 1), the following partitioning technique (multiisland) can be applied: the PRR is partitioned into 3 regions for each three-descriptor combination case. As shown in Figure 8, different regions are defined (region R1 to region R10) to receive the descriptors for the different possible configurations. The reconfigured region or regions relative to the different possible configurations (Config-i) are given in Table 8. The different partial bit streams corresponding to the defined regions are generated.

The partial bit stream loading in the FPGA is managed by the microprocessor depending on the configuration (Config-i) to be applied. The transition from one combination to another one for the three-descriptor combination leads to a complete reconfiguration of the PRR (the three regions). To pass from a combination of three descriptors to two, of three to one, or of two to one, only the concerned regions are reconfigured. For example, in the transition from the (EH + CSD32 + LH-BRAM8) configuration to the (CSD32 + LH-BRAM8) one, only region R3 needs to be reconfigured. This region (R3) is a common region (same dimensions and location) for the different defined subdivisions of the PRR. When passing from the (CSD32 + LH-BRAM8) configuration to the (LH-BRAM8) one (Config-1 in Figure 8), only region R2 needs to be reconfigured. If the region needing to be reconfigured is needed for another treatment, it will be loaded using the corresponding bit stream. Otherwise, it will be configured using an empty module (the region consumes no dynamic energy).

The required area (to implement the descriptors) and the reserved area in terms of frames, and the regions corresponding to each PRR configuration, are presented in Table 8. The reconfiguration times relative to these configurations are also provided in this table. The internal configuration access port (ICAP) is utilized for the reconfiguration of the FPGA regions. Its frequency is set to 100 MHz. For the Xilinx Virtex-5 FPGAs, the ICAP features 32 bits, which can reach a theoretical maximum throughput of 400 KB/ms at 100 MHz.

## 5. Conclusion

The main objective of this paper is to design a real-time video summarization system based on the LH technique. After presentation of the LH specification, the application of this technique for video shot boundary detection was studied. The LH was tested for different grayscale quantization levels. The experimental results have shown that even with only four quantization levels, the technique presents reliable results. A new and highly efficient hardware implementation of the LH-based shot detection system supporting real-time constraints of HD video formats has been put forward. The system was evaluated for different possible architectural solutions and for different quantization levels and image sizes. We have also demonstrated that the number of color components and the number of image regions do not affect the obtained performances in terms of execution time. Comparison with the existing implementations of shot boundary detection systems has shown that the suggested system would present superior hardware performances with a satisfactory detection rate. At the end of the paper, SOC architecture integrating the LH-based shot boundary detection module has been introduced and a static implementation has been developed. An implementation using the partial dynamic reconfiguration technique was also discussed and realized. The presented results demonstrate the benefit of different hardware architectures of the LH for intelligent management of the system. This ensures a high level of flexibility and an optimization of hardware resource occupation.

## References

[1] Hu W, Xie N, Li L, Zeng X, Maybank S. A survey on visual content-based video indexing and retrieval. IEEE T Syst Man Cy C 2011; 41: 797-819.

[2] Kapela R, Świetlicka A, Rybarczyk A, Kolanowski K, O'Connor NE. Real-time event classification in field sport videos. Signal Process-Image 2015; 35: 35-45.

[3] Claus C, Stechele W. AutoVision-reconfigurable hardware acceleration for video-based driver assistance. In: Platzner M, Teich J, Wehn N, editors. Dynamically Reconfigurable Systems: Architectures, Design Methods and Applications. Dordrecht, the Netherlands: Springer; 2010. pp. 375-394.

[4] Ben Abdelali A, Hannachi M, Touil L, Mtibaa A. Adequation and hardware implementation of the color structure descriptor for real-time temporal video segmentation. Journal of Real-Time Image Processing 2014; 2014: 1-20.

[5] Kapela R, Śniatala P, Rybarczyk A. Real-time visual content description system based on MPEG-7 descriptors. Multimed Tools Appl 2011; 53: 119-150.

[6] Priyanka S, Jharna M, Santhosh Kumar L. Video shot detection on embedded system. International Journal of Advanced Research in Computer and Communication Engineering 2015; 4: 49-53.

[7] Pal G, Rudrapaul D, Acharjee S, Ray R, Chakraborty S, Dey N. Video shot boundary detection: a review. In: Satapathy CS, Govardhan A, Raju SK, Mandal KJ, editors. Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India CSI, Volume 2. Cham, Switzerland: Springer International Publishing; 2015. pp. 119-127.

[8] SenGupta A, Thounaojam DM, Singh KM, Roy S. Video shot boundary detection: a review. In: International Conference on Electrical, Computer and Communications Technologies; 2015; Coimbatore, India. New York, NY, USA: IEEE. pp. 1-6.

[9] Chirag V, Biswajit Kumar Y, Subhabrata S. Comprehensive survey on shot boundary detection techniques. International Journal of Computer Applications 2016; 140: 24-30.

[10] Hsu CF, Ku MK, Liu LY. Support vector machine FPGA implementation for video shot boundary detection application. In: International System on Chip Conference; 2009. pp. 239-242.

[11] Boussaid L, Mtibaa A, Abid M, Paindavoine M. A real-time shot cut detector: Hardware implementation. Comput Stand Inter 2007; 29: 335-342.

[12] Liu TR, Chan SC. Automatic shot boundary detection algorithm using structure-aware histogram metric. In: 19th International Conference on Digital Signal Processing; 2014. pp. 541-546.

[13] Thounaojam DM, Roy S, Singh KM. Video shot boundary detection using gray level cooccurrence matrix. Indian Journal of Science and Technology 2016; 9: 84338.

[14] Bendraou Y, Essannouni F, Aboutajdine D, Salam A. Video shot boundary detection method using histogram differences and local image descriptor. In: Second World Conference on Complex Systems; 2014. pp. 665-670.

[15] Apostolidis E, Mezaris V. Fast shot segmentation combining global and local visual descriptors. In: International Conference on Acoustics, Speech and Signal Processing; 2014. pp. 6583-6587.

[16] Imran H, Hussain A. Image segmentation using weighted average local histogram. International Journal of Computer Applications 2013; 64: 37-41.

[17] Priya GGL, Domnic S. Video cut detection using block based histogram differences in RGB color space. In: International Conference on Signal and Image Processing; 2010, pp. 29-33.

[18] Metty M, Sarifuddin M, Eri P, Djati K, Suryadi H. Content based image retrieval using local color histogram. International Journal of Engineering Research 2014; 3: 507-511.