Journal of Signal Processing Systems (2021) 93:481–498
https://doi.org/10.1007/s11265-021-01636-4

Real-Time FPGA Implementation of Parallel Connected Component Labelling for a 4K Video Stream

Marcin Kowalczyk[1] · **Piotr Ciarach**[1] · **Dominika Przewlocka-Rus**[1] · **Hubert Szolc**[1] · **Tomasz Kryjak**[1]

Received: 3 May 2020 / Revised: 1 December 2020 / Accepted: 4 January 2021 / Published online: 1 April 2021

Abstract
In this paper, a hardware implementation in reconfigurable logic of a single-pass connected component labelling (CCL) and connected component analysis (CCA) module is presented. The main novelty of the design is the support of a video stream in 2 and 4 pixel per clock format (2 and 4 ppc) and real-time processing of 4K/UHD video stream (3840 x 2160 pixels) at 60 frames per second. We discuss several approaches to the issue and present in detail the selected ones. The proposed module was verified in an exemplary application – skin colour areas segmentation – on the ZCU 102 and ZCU 104 evaluation boards equipped with Xilinx Zynq UltraScale+ MPSoC devices.

Keywords FPGA · Zynq UltraScale+ MPSoC · 4K · UHD · Real-time video processing · Connected component labelling (CCA) · Connected component analysis (CCA)

1 Introduction

Connected component labelling (CCL) and connected component analysis (CCA) are two operations often used in computer vision systems. The first one allows to assign an unique label to each connected group of pixels. Pixels belong to the same group if there is a path of adjacent pixels between them. Usually an 8-pixel, less frequent a 4-pixel neighbourhood is used. The second operation allows to calculate selected parameters of the detected objects. Most often these are: bounding box, area and centroid. Others are: number of pixels on the perimeter, major axis length, minor axis length and orientation (obtained using ellipse fitting), and other so-called shape coefficients. However, not all of them can be computed efficiently in a pipeline pixel processing system.

CCL and CCA are an intermediate step between image analysis and recognition. Their input is a binary image obtained after thresholding (also called binarization) or segmentation (e.g. of moving or foreground objects). The output is a list of detected objects (i.e. groups of connected pixels) and their features. The described approach is widely used in advanced video surveillance systems (AVSS), e.g. for abandoned luggage or prohibited zone violation detection. On this basis, simple classification can be implemented – for example, rejection of objects that are too small or of incorrect shape. In addition, using the bounding box to select a ROI (Region of Interest) can significantly reduce the computational complexity of the vision system (e.g. candidate preselection prior classification vs. a sliding window approach).

In recent years, we observe a dynamic development of vision sensors. Currently, the most common are three resolutions: High Definition (HD – 1280 × 720), Full High

The work presented in this paper was supported by the National Science Centre project no. 2016/23/D/ST6/01389 entitled "The development of computing resources organisation in latest generation of heterogeneous reconfigurable devices enabling real-time processing of UHD/4K video stream".

✉ Marcin Kowalczyk
kowalczyk@agh.edu.pl

Piotr Ciarach
piotrciarach@gmail.com

Dominika Przewlocka-Rus
dprze@agh.edu.pl

Hubert Szolc
szolc@agh.edu.pl

Tomasz Kryjak
tomasz.kryjak@agh.edu.pl

[1] Embedded Vision Systems Group, Computer Vision Laboratory, Department of Automatic Control and Robotics, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059, Krakow, Poland

Definition (FHD – 1920 × 1280) and recently Ultra High Definition (UHD, or 4K – 3840 × 2160). There are also 8K (7680 × 4320) and 16K (15360 × 8640) solutions, but due to the high cost they are currently not widely used. The analysis of high resolution images or video streams allows to improve the broadly understood performance of a vision system. For example, a 4K sensor would allow to detect objects further away from the camera, which is important in the case of advanced driver assistance systems (ADAS), advanced video surveillance systems (AVSS), as well as autonomous vehicles (cars, drones). In this case, small objects would be registered with more details (e.g. shape, colour or texture) and therefore classified more accurately (using a classical or deep convolutional neural network approach). Moreover, they would be detected earlier, which is essential for ADAS and self-driving cars (here also a very important parameter is the frame rate – the higher, the system can react more quickly assuming real-time processing with a low latency). In the case of AVSS, the use of high resolution cameras results in a larger field of view of a single device, which allows to limit their number within the considered surveillance system.

The above-mentioned benefits come at a price, as the resolution directly affects the amount of data to be processed or stored. An uncompressed 4K video stream i.e. 3840 × 2160 at 60 frames per second (fps) results in a data flow of 1424 MB/s. Its processing in real-time is quite a challenge and requires the use of a computing platform capable of providing the required computational power (depending on the algorithm). The designer can choose from the following solutions:

- general purpose processors (GPP),
- general purpose graphics processing units (GPGPU),
- application specific integrated circuits (ASICs),
- field programmable gate arrays (FPGAs),
- heterogeneous multiprocessor system on chip (MPSoC) – which are composed of an ARM processor system, reprogrammable logic and GPU (e.g. Zynq UltraScale+ from Xilinx).

It is worth emphasising that the energy efficiency and the ability to update the applied algorithm are essential in applications such as: ADAS, AVSS or autonomous vehicle perception systems. One possible solution that can meet all these requirements is the use of state-of-the-art FPGA or reconfigurable heterogeneous MPSoC devices – they allow 4K video stream processing in real-time, are relatively energy-efficient and can be reprogrammed many times.

The presented work is a continuation and extension of the scientific paper presented at the Applied Reconfigurable Computing (ARC) 2019 [3]. The main contributions are:

- discussion on the challenges of implementing the connected component labelling (CCL) and connected component analysis (CCA) algorithms for a pixel stream in 2, 4 and more pixels per clock cycle (ppc) format,
- implementation of two CCL/CCA modules working with 2ppc and 4ppc formats and their comparison,
- verification of the proposed modules in a skin colour segmentation vision system on a development board with Xilinx Zynq UltraScale+ MPSoC (Multiprocessor System on Chip) device.

According to the authors knowledge, this is the first FPGA implementation of CCL and CCA modules capable of processing a 4K video stream @ 60 fps in real-time verified on a development board (except for our mentioned previous work [3]).

The remainder of this paper is organised as follows. In Section 2 research related to CCL and CCA algorithms and their FPGA implementation is presented. In Section 3 the properties of a 4K video stream are discussed. Then, in Section 4 the discussion about different possibilities of implementing CCL/CCA for 4K video stream is presented. Finally, the proposed implementation of the CCL and CCA modules are described in Section 5. Their evaluation in a skin colour detection system is presented in Section 6. The paper ends with a conclusion and possible further research directions.

2 Previous Work

In this section, we first discuss general approaches to connected component labelling and analysis and later concentrate on real-time FPGA implementations.

2.1 CCL Solutions Overview

There are two main approaches to connected component labelling. The first one can be described as region growing based [4]. The binary input image is analysed line by line. When a pixel belonging to an object without a label is encountered, a new label is assigned and a neighbourhood search procedure is executed. Then the connected pixels are labelled recursively. It should be noted that this solution is not suitable for a single-pass implementation in a pipeline vision system, where pixels are processed "one-by-one" without frame buffering. In this scenario, there is practically no possibility of random access to image data (only in a small context). Moreover, the internal resources of today's FPGA are not sufficient to store a 4K image frame. Therefore, a recursive solution would have to be based on

an external RAM (usually dynamic RAM), which has a significant latency and increases the overall complexity of the system.

The second solution assumes linear/sequential image processing. These are so-called two-pass and single-pass algorithms. The two-pass solution by Rosenfeld and Platz [15] should be considered as "classic". It consists of two scans of the image and three stages. During the first scan, pixels are given temporary labels and possible conflicts (mergers), i.e. situations in which the same object has received two or more labels, are written to the equivalence table, which has a graph structure. For example, a conflict occurs for a U-shaped object. Within the first scan, two separate labels are given to the pixels and information about the connection appears when both arms are converging (Figure 1). Moreover, the image with temporary labels is stored. In the second stage, the equivalence table is analysed – the transitive closure of the graph is calculated. Next, the final labels are determined and assigned to particular pixels during the second image scan (temporary labels are used). It should be noted that the direct output of this method is a labelled colour image - every object has a different colour.

In the context of hardware implementation, the main issue with this solution is the buffering of the pre-labelled image. Again, due to the limited internal memory resources of FPGA devices, it is necessary to use external DRAM modules. These increases the complexity of the system, as a RAM controller is required and impacts energy efficiency.

There are also single-pass algorithms, that do not require buffering of a pre-labelled frame. Emerging conflicts are resolved "on-line" during the first scan. The implementation is more complex than in the two-pass case – this is shown in detail in Section 5. It should also be noted that the result of this operation is not an image with labels, but only a set of parameters of the detected objects (centroid, area, bounding box). However, in the vast majority of applications, this information is sufficient.

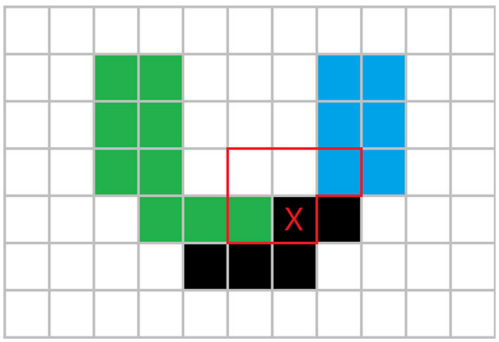

Figure 1 A "U" shaped object. The place where a conflict (merger) occurs is marked with a red cross (8-pixel neighbourhood).

2.2 CCL/CCA Implemented in FPGA

Due to the large practical significance of CCL and CCA in computer vision systems, a number of articles about the hardware implementation of these operations in FPGA have been published.

A classic two-pass approach implementation was described in the work [9] from 1995. The proposed system used 9 Xilinx XC4010 FPGA devices and processed up to 30 images with a resolution of 512 × 512 pixels per second.

A two-pass approach was also proposed in the paper [1] from 2010. It is distinguished by the analysis of a series of pixels appearing in a single row (so-called runs or series). The algorithm works in four steps:

- Conversion of pixels into series in the form (ID, EQ, s, e, r), where: ID - series identifier, EQ - assigned label, s - series start in the given line, e - end of the series in the given line, r - the number of the image line.
- The first run over series and creation of the equivalence table.
- Solving mergers/conflicts.
- Second pass over the image – assigning appropriate labels.

The solution has been implemented on the RC340 platform with a Xilinx Virtex 4 device. Real-time processing of a 640 × 480 @ 35 fps video stream was obtained.

In the case of hardware implementation in FPGAs, however, the single-pass approach seems to be the most attractive solution. It was first proposed by the team Ma, Bailey and Johnston in 2008 [2]. In this approach to CCL, only the last line of the image that has already received its labels is required during analysis, thus reducing the amount of buffered data (single line vs. frame). An equivalence table is also used to correctly handle any mergers of labels. In addition, the authors proposed a mechanism that protects against so-called merge chains, i.e. the occurrence of several mergers within the same object, in one line. For this purpose, a special stack was created, in which both labels involved in such a merger were stored. Then, during the horizontal blanking time in each line (the period in the video signal during which no pixels are transmitted), the equivalence table was updated with the stored mergers. The only disadvantage of this solution is the limitation of the maximum number of mergers present in the chain. The worst case requires the length of blanking time to be about 50 % of the length of the actual image line. Meanwhile, this value usually does not exceed 30 %. In real applications, however, the chance of the "worst case scenario" is very low, as usually some pre-processing operations like morphological or median filtering are applied. In addition,

the authors proposed a mechanism for recovering labels between successive image lines. The module has been implemented in the Handel-C language and verified on the RC300 board with the Xilinx Virtex II FPGA device. For 640×480 resolution, 100 frames per second were processed.

A development of the above presented idea is described in the work [6] from 2016. A module was added that protects against incorrect labelling in very specific cases when using the label reuse approach. The possibility of implementing the module on various FPGA devices: Virtex 6, Spartan 6 and Kintex 7 for different resolutions – also 4K and 8K was analysed. However, the obtained maximum operating frequency of the module does not allow real-time (i.e. 60 fps) operation for such large resolutions. For Virtex 6 and Kintex 7 real-time processing was achieved for 1920 × 1080 (pixel clock about 150 MHz).

A non-standard implementation of a CCL module was described in the work [5] from 2016. The equivalence table was omitted and a shift register of length equal to the entire image line was applied. This register kept $n + 1$ last assigned labels, where n – width of the image. To implement the required functionality, DRAM (Distributed RAM) memory resources are required. As a consequence, in the case of a merger event, all labels could be updated in the shift register at the same time. However, the solution has significant limitations. The maximum number of labels is 63 or 127. The rather complicated logic results in a small maximum clock frequency, which translates into the number of frames that can be processed in one second. For example, for 1920 × 1080 resolution and 127 labels: 37 fps (calculating only bounding box) or 28 fps (calculating the bounding box and centroid) were obtained. The Altera Cyclone IV device was used in the experiments.

Another interesting CCA hardware implementation was presented in the work [16] from 2017. The proposed system for extracting statistic information about objects, such as bounding boxes, accumulated values of horizontal and vertical coordinates, and the number of pixels in a region, was divided into four steps:

- Pixels form the binary image are received and buffered in order to process two (previous and current) rows simultaneously.
- Both rows are scanned using a 2 × 2 template and the statistical information about runs (consisting of continuous foreground pixels) are collected.
- Information from adjacent rows are merged and exported as completed connected regions.
- If the current row is the last row in the frame, runs are merged and the process is finished. If not – the current row is stored as the previous one and the above steps are repeated.

The proposed design was tested in simulation for different video stream resolutions. The greatest one was 2048 × 1536 and it resulted in a processing time about 53ms, which translates to around 19 fps. For smaller resolution, the authors obtained a higher fps rate.

In the paper [12] from 2018 authors extend the architecture presented by [2] to store component size and position while using internal memory. Authors reported that their system achieves 223 fps for 640 × 480 images and 94 fps for 1280 × 720 images with 128.07MHz clock on Virtex-5 FPGA.

In the work [11] the authors proposed a novel single-pass CCA algorithm to overcome the need to resolve label equivalence in the horizontal blanking period. In order to label, resolve equivalences and extract the object features in a single scan, the combination of linked list for representing equivalences and run-length encoding for labelling connected components techniques were proposed. The proposed system achieved impressive results in terms of memory efficiency compared to [16] and [6], while sustaining the real-time processing for 640 × 480 images. The architecture was implemented on Virtex II and Cyclone IV FPGAs.

In the article [10] the authors presented a CCL architecture suitable for implementation on a System-On-Chip, that exploits both the Programmable Logic and the Processing System with an ARM processor, as well as an external DDR memory for image storage. The main contribution of this paper is to overlap the label collisions resolutions and the DMA core configuration actions performed by the ARM processors. After processing the last pixel in the frame the resolution of all collisions starts and, at the same time, DMA sends an interrupt to the ARM cores with information that the labelled image is now ready in the external memory – by the time DMA is configured, the label collisions are resolved. For 640 × 480 pixels resolution, a rate of 700 fps was achieved on a Zynq AP-SOC 7045 chip.

In the paper [13] the same authors present a highly efficient hardware-oriented approach for CCA achieving, for 640×480 image resolution, a frame rate of 325.5 fps and occupying 760 LUTs and 787 FFs, with no BRAM usage, which exhibits high resource efficiency in comparison to other competitive CCA hardware implementations. The proposed solution simultaneously assigns provisional labels, manages equivalent labels and updates features data using auxiliary tables. The system was tested on a Xilinx Zynq XC7Z020 FPGA SoC.

The authors continue the problem of a parallel connected component labelling architecture for heterogeneous Systems-on-Chip in [14] to design the architecture complying with the fourth generation of the advanced extensible interface (AXI4), storing the intermediate and final outputs within an off-chip memory. Depending on the number of

labels, the design achieves from 411.3 (1024 labels) up to 594.2 (64 labels) fps for 640 × 480 image resolution. The architecture was also tested with higher resolutions – for $2K \times 2K$ image, depending on number of labels (1024 and 64), rate from 30.3 to 46.4 fps was achieved. Authors state that the proposed CCL approach can be implemented to work with 4K UHD resolution and sustain 60 fps rate, however it was not tested.

Summing up the review, it should be noted that it covers a period of 25 years (1995-2020). There was a dynamic development of technology during that time – both in vision sensors and FPGA devices. For example, in the work [9] from 1995, 9 Xilinx XC4010 chips were used and the image was processed with a resolution of 512 × 512 at 30 fps (clock frequency 10MHz), and in the most recent work, [14] from 2020, a single Zynq SoC was used for a stream with a resolution of $2K \times 2K$ and over 30 fps (clock frequency around 100MHz). Thus, the progress, understood as the possibility of processing a stream with higher resolution and fps, is firstly related to the use of newer generations of computing platforms. However, a direct analysis of this phenomenon and an attempt to compare (reduce to the "common denominator") solutions in terms of the use of different computing platforms is not simple and was not the aim of this article. Just like the optimisation of FPGA resource usage is not the main topic of our work.

The second factor enabling the aforementioned progress is the use of various algorithmic solutions, which consequently translate into the hardware architecture of the CCL module. The mentioned solutions can be divided into the following categories:

- two-pass – [9] (1995),
- two-pass with pixel "series" analysis – [1] (2010),
- single-pass – [2] (2008), [6] (2016), [13] (2019), [14] (2020),
- single-pass with shift register – [5] (2016),
- single-pass with advanced pixel "series" analysis – [16] (2017), [11] (2018),
- single-pass with post processing on an ARM core – [10] (2018).

Analysing the above list, it can be noticed that most of the works are based on the proposal from [2] and follow the single-pass approach. In each of the above-mentioned publications, the authors introduce some algorithmic improvements, which, apart from the use of newer generation equipment, allow to achieve better video stream processing parameters. Two-pass methods are not used nowadays. In addition, recently (2016-2020) authors have been looking for algorithmic improvements through the use of e.g. shift register and pixel series analysis. The aim of these works is, among others, the optimisation of resource utilisation, or the desire to eliminate calculations during horizontal and vertical blanking.

It should be also emphasised that, according to the authors' knowledge, no module capable of processing a 4K video stream with a frequency of 60 fps has been presented and tested in hardware so far (except for our previous work [3]). Moreover, no other work discusses the issue of 2 or 4 pixels per clock format for connected components labelling of a video stream in real-time.

3 4K Video Stream

A video stream in RGB format with a resolution of 3840× 2160 and 60 frames per second sent in 1 pixel (24 bits) per clock format (the so-called pixel clock) requires a pixel frequency of approx. 500 MHz. The so-called vertical and horizontal blanking fields present in the video signal increase these value to 600 MHz. This is the "limit" value for currently available reconfigurable systems (FPGA and reconfigurable SoC). Admittedly, selected components, such as block memories (BRAMs), hardware multipliers (DSPs) are, according to the Xilinx manufacturer's declaration, able to work with even higher frequencies (this depends, among others, on the version of the device – it's speed grade, supply voltage, type of operations). However, in practice, for more complex logic, achieving such frequencies can be very difficult, since the delay associated with the connection resources has also to be considered.

Due to the above-described 4K signal parameters and the limitations of the currently available reconfigurable devices, it is not possible to use the well known 1 pixel per clock scheme (1 ppc), which is the basis of the modules described in Section 2 and most of the works related to the real-time vision systems implemented in this technology. Therefore, for a 4K signal, 2 ppc or 4 ppc format is used. This allows to reduce the pixel clock frequency to 300 MHz and 150 MHz respectively. However, its use has quite significant implications for the way the CCL/CCA operation is implemented in a pipelined vision system – this is discussed in detail in Section 5. It is also worth mentioning that for the 2 or 4 ppc format it is necessary to multiply the used computing resources. In addition, contextual operations such as filtering or median require modification of a typical context generation scheme. This issue has been presented in our previous paper [8]. Finally, looking ahead for a 8K 7680 × 4320 signal, two solutions will be possible: a 4 ppc 600 MHz or 8 ppc 300 MHz format. Thus, works on the processing of video stream in vector format are fully justified.

4 Concept of the Proposed CCL/CCA Modules

In this section, we first discuss the main challenges and possible solutions in implementing a CCL/CCA module for a 4K video stream in 2 and 4 ppc format and then propose four approaches: simplified CCL/CCA for 2/4 ppc, CCL for 2 ppc, CCL/CCA for 4 ppc and CCL/CCA for pixel series in 2/4 ppc.

4.1 CCL/CCA in 4K – Challenges and Possible Solutions

The main challenge in the implementation of connected component labelling and analysis for a 4K stream is the need to process 2 or 4 pixels simultaneously. Difficulties arise from the fact that there are direct relationships between the pixels being analysed. This is a trade-off between clock frequency and label assignment logic complexity. For example, assuming no foreground pixels in the context of the 1001 vector (4ppc), it is necessary to assign two new labels, and for 1111 to propagate the same label to 4 pixels. These dependencies differentiate the implementation of CCL/CCA from other image processing operations, such as contextual filtering (median, Gauss, morphological) – where the use of the vector format: (1) increases the computing resources and (2) results in a more complex way of generating 2 or 4 contexts simultaneously.

Based on the analysis from Section 2, 4 single-pass approaches were selected for further consideration:

- CCL/CCA module with 4 ppc format, with simplifications,
- CCL/CCA module with 2 ppc format,
- CCL/CCA module with 4 ppc format,
- CCL/CCA module with 2/4 ppc format, with series analysis.

We have chosen these solutions because, in our opinion and based on our experience with FPGA image processing, they were the most promising in terms of real-time implementation in FPGA, the possibility to process 2, 4 or even more pixels in one clock cycle and scalability.

Firstly, we have focused on the 4 ppc format, because meeting timing constraints for a 150MHz clock is easier than for 300MHz. In addition, 4 ppc is more promising in the context of the future 8K format.

We considered how the extended context (4 pixels simultaneously) affects the CCL/CCA algorithm itself. It turned out that this configuration has pixel arrangements that were not present in the well known 1 ppc approach. This situation is presented in Figure 2.

In the depicted situation, the algorithm should be able to combine labels 2 and 3 with label 1. This involves the need to write appropriate values to the equivalence table (EQT).

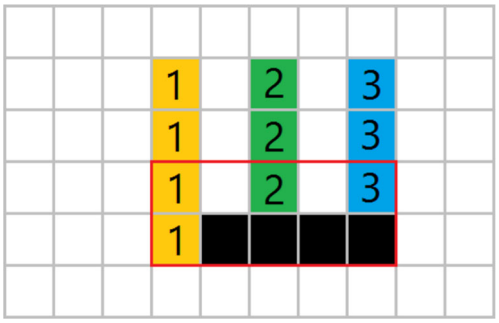

Figure 2 Exemplary situation with two merges in one clock cycle.

However, saving the value to two addresses in the memory in one clock cycle is not possible, while it is required to properly handle situations when a merged label (2 or 3 in this case) reappears in the context in the next clock cycles (in the same line). An additional challenge is to analyse the labels for all 4 pixels simultaneously.

4.2 Simplified CCL/CCA in 4ppc

Firstly, we proposed a solution to eliminate the above-described case (this was the main contribution of the ARC 2019 paper [3]). It is based on connecting two adjacent (binary) pixels using the "OR" operator, which eliminates the possibility of three mergers event. This is shown in Figure 3. The applied approach also reduces the necessary hardware resources required to implement the module.

It should be clearly stated that the presented solution reduces the horizontal resolution of the image. Combining two different pixels causes that the labelled image resolution is 1920×2160 instead of 3840×2160. This approach also results in joining objects that have a 1 pixel gap between them. Taking that into consideration, the proposed labelling algorithm does not process a "real" 4K image.

On the other hand, in a real vision system, after segmentation (binarization) and before labelling, the binary mask is usually filtered (post-processed) – using morphological operations or median. This results in similar to the above-mentioned merging effects of closely located pixels.[1]

4.3 CCL/CCA in 2ppc

Secondly, to avoid the effects described in the Section 4.2, we considered processing only 2 pixels in each clock cycle. This allows skipping pixel merging, but doubles the required operating frequency (300 MHz vs. 150 MHz). Therefore, the module should be carefully designed to avoid long critical paths. It is also necessary to prepare a whole new video track (pass-through, pipeline) in which all the system components would also operate in the 2 ppc scheme.

[1] In the case of erosion this gap would increase.

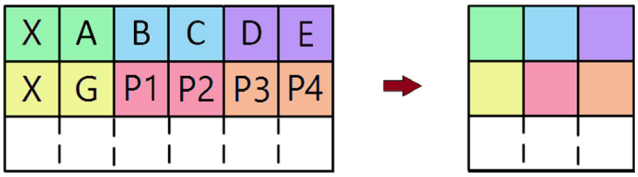

Figure 3 Illustration of the proposed concept. Pixels joined by the OR operator are marked with colours, P1–P4 - pixels to be labelled.

4.4 CCL/CCA in 4ppc

Next, we considered a method for 4K video resolution in 4 ppc format so that adjacent pixels are not combined. The proposed concept is based on two main assumptions. Firstly, labels must be assigned for all incoming pixels and all mergers (conflicts) must be determined within one clock cycle. However, the equivalence tables does not need to by updated in one cycle.

Secondly, if there is more than one merger in one data set, the video stream will be stopped and the next input will be buffered in a FIFO queue. For a 4 ppc format, a maximum of two mergers can occur in one clock cycle. Two clock cycles are needed to write the appropriate values into the equivalence tables – the stream has to be stopped for one clock cycle.

Using the AXI-Stream allows the receiving module to suspend the input stream using the *tready* signal. Such "pauses" can be later "worked off" e.g. in horizontal or vertical blanking areas, or even inside the line, if the stream frequency is slightly higher than the native frequency of the video stream for a given resolution.

A problem appears if the case of two merges in one clock cycle would appear very often. Then the system will not have enough time to process all data because of too many "pauses". However, it should be emphasised that the most problematic situation of two merges occurs quite rarely, especially if the binary image has been previously filtered (with the median or morphological operation). Therefore, a situation where this approach will cause synchronisation problems across the entire pipeline is highly unlikely.

However, in contrast to the 2 ppc or simplified 4 ppc formats, there are no easy rules on how to choose a new label value. Therefore, it is necessary to calculate new labels from left to right (like in the 1 ppc approach). Let's define the *MinNZ* operation as:

$$MinNZ(a,b) = \begin{cases} min(a,b) & \text{if } a \neq 0 \wedge b \neq 0 \\ a & \text{if } b = 0 \\ b & \text{if } a = 0 \end{cases} \quad (1)$$

For every foreground pixel, three $MinNZ$ operations are performed. First, for the pixels located on the upper left and left (A and D). Second, for the pixels located on the upper right and above (C and B). The third one is calculated with the results of the previous two operations. Visualisation of the context of the currently analysed pixel P is presented in Figure 4. In addition, the maximum value of the results of the first two operations $MinNZ$ is also calculated. If the analysed pixel does not belong to the background, then the result of the third operation $MinNZ$ is assigned as a new label of the considered pixel. Otherwise, 0 is assigned.

If the results of the first two operations $MinNZ$ are different from zeros and are not equal to each other, then a conflict (merger) is detected. Depending on the situation, more than one conflict can occur. If this is the first conflict detected (the corresponding flag equals 0), the flag indicating the occurrence of the first conflict is set and the labels: $[L_{max}, L_{min}]$ are saved – where L_{max} is the output of the previously calculated maximum and L_{min} is the output of the third operation $MinNZ$.

If the detected conflict is not the first one, then it is checked whether the current conflict is different from the previous one (in certain contexts such double conflict situations occur). If it is a different conflict, it must also be stored.

If a conflict is detected, the result of the second $MinNZ$ operation is also checked. If it is greater than the result of the first operation, the merge chain is detected and must be remembered in the same way as conflicts, however, it is not necessary to check if the same merger is detected a second time. If the result of the third operation $MinNZ$ is 0, then the pixel is given a new label.

The rest of the algorithm, i.e. handling of mergers and merge chains, works very similarly to the simplified solution for 4 ppc and is discussed in detail in Section 5. A critical element of the described solution for hardware implementation is the time needed to calculate labels for all four pixels, i.e. the operations described above.

Based on the above description, it is easy to tell that this element would be a bottleneck of the entire module. For this algorithm to work in real time, the critical path must have a delay less than 6.66 ns, which corresponds to a frequency equal to 150 MHz for 4K signal and 4 ppc format. It should be also noted that the above described algorithm has been implemented in Matlab and evaluated on several hundreds

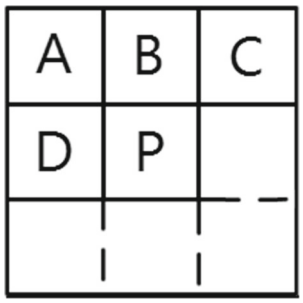

Figure 4 Context of the currently analysed pixel.

random binary images – in all cases the labelling results were identical to those returned by the `bwlabel` method.

4.5 CCL/CCA in 2/4ppc for Pixel Series data

The last considered approach is the use of pixel series analysis. It assumes that the CCL/CCA algorithm will not work directly on pixel values, but on the data that has been calculated based on them. It corresponds to series (runs) of adjacent pixels that represent a fragment of a continuous object within one line. Therefore, to represent them, a set of three numbers is sufficient: the label (ID), the beginning and the end of the given series. Depending on the implementation, it can be additionally supplemented with the line number in which the given series is located.

In this case, the algorithm can be divided into two parts: the first, which is responsible for the conversion of raw data into pixel series, and the second, which based on information about the series, will label the input image. The described approach was used, among others in the work [1]. It is worth noting that in this case it is not necessary to consider the effect of changing the label of one pixel on the labelling result of its nearest neighbour in a given row – all updates are done simultaneously on the whole group of pixels.

At the same time, obtaining these series, especially for the 4 ppc format, is highly problematic. Note that in one clock cycle even two series have to be processed at the same time. One of the possible solutions to this problem would be the use of a special switch, which would direct data to the appropriate processing modules based on the input vector structure. However, this is associated with an increase in the required hardware resources to handle all cases.

A more efficient solution to this problem is the conversion of the input image to a pixel series, combined with preliminary detection of mergers. This detection checks whether the currently detected series is connected to one of the series received in the previous line. This approach for the 1 ppc format was presented, among others, in papers [16] and [11]. In both cases, the authors suggest using a 2×2 context, shifted every clock cycle by 1 pixel. If such context is used, a finite number of pixels' combinations indicate the merging of two series within consecutive lines. When attempting to implement a similar solution for a 2/4 ppc stream, however, 2 or 4 contexts must be processed simultaneously. Moreover, they have to be considered in a proper order – according to the direction of image scanning. The very idea of detecting connected series, however, remains unchanged.

The series detected in this way and their initial mergers must then be stored for further processing after the row scan is completed. With a 1 ppc stream, this is done by simply writing the necessary values to the appropriate RAM addresses. In the case of 2/4 ppc streams, however, the possibility of a memory access conflict should be considered. It appears when processing more than one series in a clock cycle. Therefore, more sophisticated logic is needed to support communication with RAM.

Summing up – the advantage of using pixel series when working with a 2/4 ppc stream is that there is no problem of propagating label changes within one series. At the same time, however, this approach causes a number of other difficulties that seem to effectively balance the resulting profits.

4.6 Summary

At the conceptual stage, four solutions to CCL/CCA with 2/4 ppc format used in 4K video stream were considered. The first two, i.e. simplified 4 ppc and 2 ppc are based on the same approach – processing two pixels in parallel. Their hardware implementation has been described in detail in the following section. The third approach involves parallel processing of 4 pixels. The last solution, based on pixel series, is not straightforward to apply to the 2/4 ppc format. Therefore, additional research is required to explore all pros and cons.

5 The Implemented CCL/CCA Module

In this section the CCL/CCA module implemented in two variants is presented. One for the 4ppc format (as in [3]) and other for the 2ppc format.

An overview of the proposed algorithm is presented in Figure 5. First, the context is extracted from the incoming video stream. Then, if at least one of the considered pixels (P1-P4) belongs to an object, a label is assigned. Details of this process are depicted in Listing 1. Moreover, object parameters like area, bounding box or centroid are updated. Additionally, at the end of a line, if some labels were merged, the correspondence tables are updated, as well as object parameters.

The scheme of the proposed connected components labelling and analysis module for the 4K video stream is shown in Figure 6. The used colours indicate the differences to a typical 1 ppc single-pass approach – like in [2].

In the next subsections, each sub-module is discussed separately. All were described in Verilog hardware description language. For simulation and implementation, the Vivado 2018.2 tool was used.

5.1 Neighbourhood Analysis

The module assigns labels to subsequent pixels – two labels in parallel. Two pixels are merged with the use of OR operation only in case of 4 ppc format. This issue has been

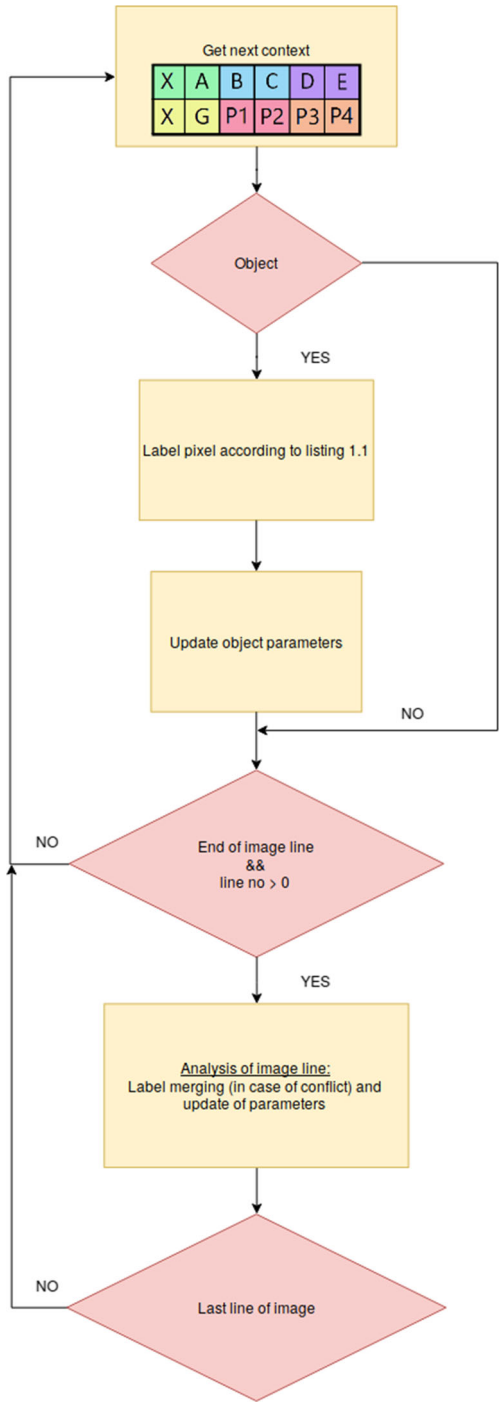

Figure 5 Overview of the implemented algorithm.

```
if P1 == 0 (background) then
    if P2 == 1 (objects) then
        if B == 0 and C == 0 then
            set new label (Case 1)
        else
            if B == 0 and C != 0 then
                set label C (Case 2)
            else
                set label B (Case 3)
 else
    if B != 0 and D != 0 and B != D then
            no label – ct. B, D (Case 4)
        else
            if B != 0 then
                if D == 0 or D == B then
                    set label B (Case 5)
 else
set label min(B, D) – ct. B, D (Case 6)
            if C == 0 then
                if A != 0 then
                    set label A (Case 7)
                else
                    if D != 0 then
                    set label D (Case 8)
                    else
                  set new label (Case 9)
            else
              if A == 0 and D == 0 then
                  set label C (Case 10)
                else
                    if A != 0 then
                        if A != C then
set label min(A, C) – ct. A, C (Case 11)
            else
                  set label A (Case 12)
                  else
                      if D != C then
set label min(C, D) – ct. C, D (Case 13)
                else
                    set label C (Case 14)
```

Listing 1 Label assignment procedure – pseudocode. ct. – conflict.

explained in Section 4.2. The information about labels in the neighbourhood is obtained from A, B, C and D registers (c.f. Figure 7a). If the group belongs to a new object, the stack of labels is used (see Section 5.5). Because of the 2 or 4 ppc format, the context generation scheme is different than e.g. in the work [2]. The value from register C is transferred to A, whereas B and C get values from the delay line (the values read from the delay lines are updated according the equivalence table in the EQT module). Register D stores the value that has been assigned to the pixel P2 in the previous clock cycle and register L the value assigned to the pixel P1. Multiplexers in front of registers A, B and C are used in the case of merging – the correct (merged) label is passed directly to all registers. Their usage eliminates the latency introduced by the equivalence tables implemented as a BRAM memory (its update takes at least one clock cycle). In result, the module informs about the given label and a possible merger event.

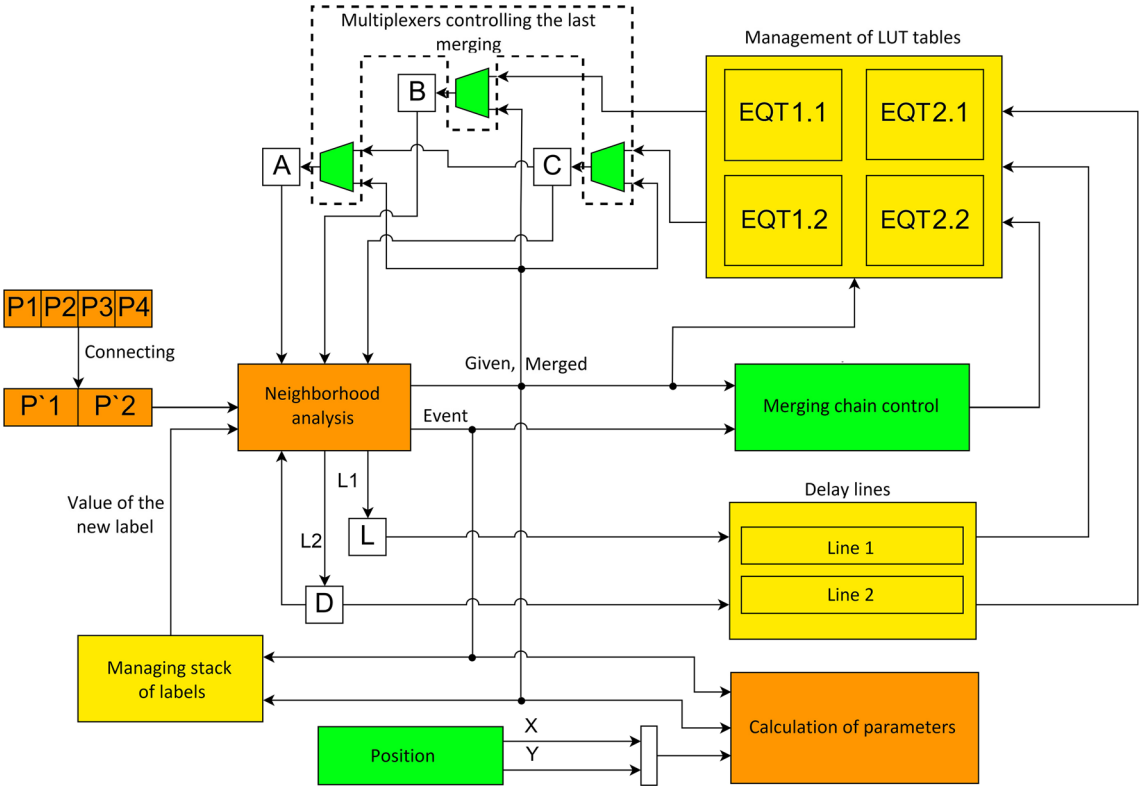

Figure 6 Scheme of the proposed 4K CCL/CCA module. Colours indicate the difference to a standard 1 ppc version: orange – large, yellow – medium, green – slight modifications.

The label assignment pseudocode is depicted in Listing 1. It is worth to emphasise the separation of operations into two cases – when P1 belongs to the background or to the object. In the first one, a separate sub-case is additionally handled when P2 also belongs to the background. Then a specific situation may occur, in which, despite the fact that both considered elements are equal to 0, a merger operation should be performed.

The listed cases are shown in Figure 7. The sub-image in Fig. 7a contains a reminder of the used symbols, and each next corresponds to the cases in Listing 1. White boxes are the background, blacks are the analysed groups of objects, labels are marked with orange and blue colours. In case when two values of one register are possible, the respective colours are placed in two halves of a given block – as it can be seen in Fig. 7d. The *C* register may have the same value as the *B* register or it may belong to the background and it will not change the result of the analysis. In addition, pixels coloured in grey are not taken into account in the analysis or are not yet known at this stage of the processing.

5.2 Delay Lines

Dual-port block memory resources (BRAM) were used to implement delay lines, as in the work [8]. The use of two module instances results from the processing of two elements in parallel.

5.3 Equivalence Tables

To allow the reuse of previously merged labels, it was decided to assign two equivalence tables for each group of two pixels. The applied solution is based on the work [2]. The first table contains equivalences of labels that have occurred at least once during the scan of the previous image line. Those are used for resolving the correct value of labels coming from delay lines. On the other hand, all assigned (new and already existing) labels during the analysis are stored in the second table. This information is required to reuse the merged labels that were not present in the previous line. At the end of each image row, the roles of the mentioned tables are swapped.

The first table is updated only in case of a merger event and when the label with a smaller value is on the left side of the considered pixel. The update of the second table is more complicated. It is executed in the case of: a merge, where the label with a smaller value is on the right site, in case of merger chain, new label and "continued label". The last two cases are handled by a temporary stack, where the labels are stored prior equivalence table update.

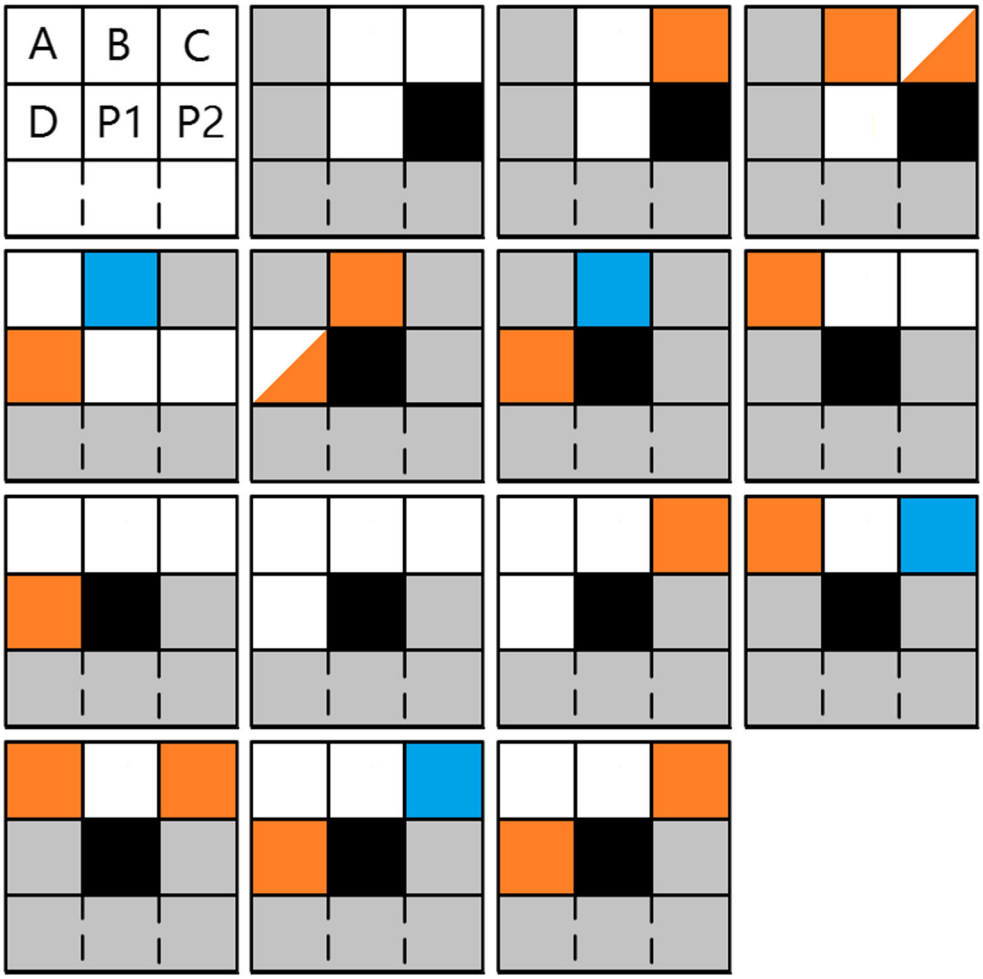

Figure 7 All the possible cases from Listing 1.

In a 1ppc approach, these labels are stored in the equivalence table when the current pixel belongs to the background. When 2 pixels (after the OR operation) are processed in parallel, a different approach is necessary, as one pixel could belong to the background and the second to another object. Therefore, the equivalence tables are updated when all considered pixels belong to the background. Two cases are possible. If in the previous context foreground pixels were present, the new label is stored. On the other hand, if the context was empty and some data was present on the stack, it is stored in the table. This concept is presented in Figure 8.

5.4 Merger Chain Control

The merger chain is a situation in which several mergers occur in one line for the same object – an example is illustrated in Figure 10. In the software version of the algorithm, the correct handling of this event is guaranteed by the used graph structure. In a pipelined hardware version, where the new label must be determined during one clock cycle, it is necessary to use an additional mechanism. This module uses a solution derived from the work [2].

Its basis is a stack storing the merged and given labels (only in case when the label with greater value is located to the left of the analysed group of pixels). Then, during the horizontal blanking period, three operations are performed: (1) retrieving from the stack, (2) reading the equivalent of the given label from the corresponding equivalence table, (3) writing this value into the equivalence table at the address of the merged label. In addition, in the second step, it is necessary to check whether the currently considered label is equal to the merged one from the previous stage (which protects against access conflict to the BRAM memory). Finally, it is worth noting that the use of an appropriate preprocessing (morphological operation, median filtering) allows to significantly reduce the probability of the occurrence of this type of pixel configurations.

5.5 Label Stack

In the described module, a mechanism for recovering unused (i.e. merged) labels was applied – a stack-based approach was used. The value of a new label (that is available) is present on

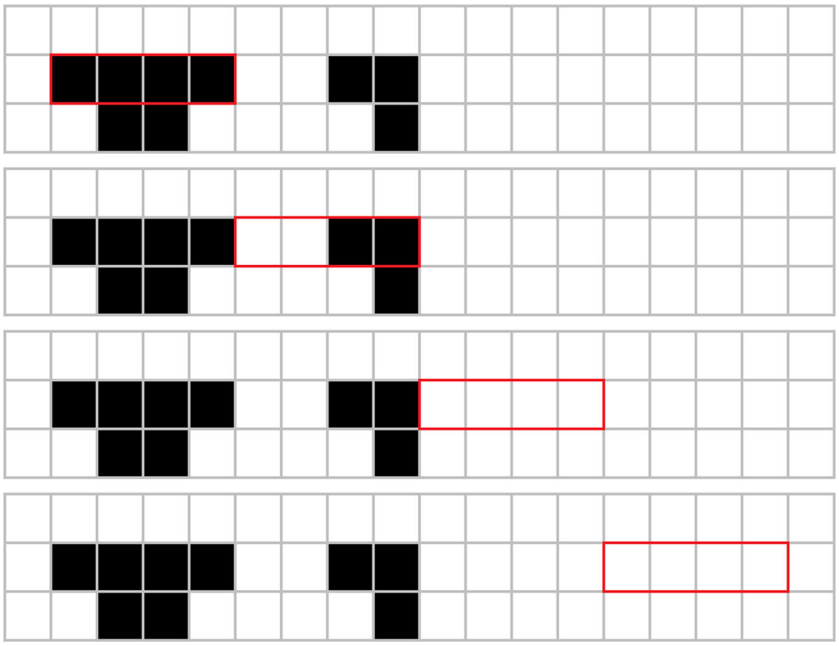

Figure 8 Operations performed when four consecutive background pixels are detected.

the top of the stack. Labels "recovered" during merging are placed on the stack at the end of the image line.

An important stage of the module's operation is restoring the stack to its initial state between consecutive frames. It is filled with successive numbers in the opposite order (the maximum number of labels is a parameter of the module). Due to the specificity of the 4K signal and the data transfer in the AXI Stream format (bus used on the ZCU 104 hardware platform), it was not possible to reset the entire stack (BRAM memory) during the vertical blanking period, as it was too short.[2] We therefore decided to implement a stack restoring mechanism that starts operation during the blanking period and when necessary, also continues to work in parallel with the processing of the next frame.

The concept is presented in Figure 9. The blue colour corresponds to an address, from where a new label will be read, while red corresponds to an address, where the stack is being reset. The first two reset steps are executed during the end of the previous frame. The reset is continued during the new frame, however it is stopped when a new label is required (step 4).

5.6 Calculation of Parameters

The single-pass connected components labelling module would be practically useless, if it was not integrated with the functionality allowing to compute object parameters. The calculation of bounding box coordinates, area (m_{00}) and the first order moments (m_{01}, m_{10}) was implemented. Based on the last two, the objects' centroids can be calculated.

The implementation uses a double buffering mechanism. In one data structure, the parameters of the objects from the previous frame are stored and available for further processing. In the second one, the current calculations are carried out. The switch is performed at the beginning of a new frame.

The key element of the module is merger handling. In the case of a 1 ppc stream, the single-pixel gap between two objects is used. It allows, after merger, to read the feature vector of the second object and save the integrated results to the data structure.

For the 2/4 ppc format, the application of the above described approach would require a gap of at least 2/4 pixels (cf. Subsection 5.3). Therefore, a solution based on a FIFO queue is used, where the values of merged labels are stored.

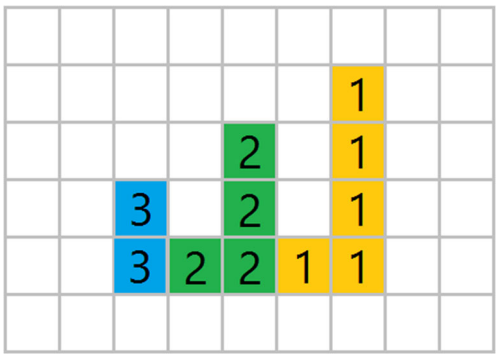

Figure 10 Sample merger chain: first labels 3 and 2, followed by 2 and 1.

[2] It should be noted that the blanking periods in the native format are usually much longer and the conversion to AXI Stream format "shortens" them. This is a result of the assumption that in AXI Stream only valid data is transferred.

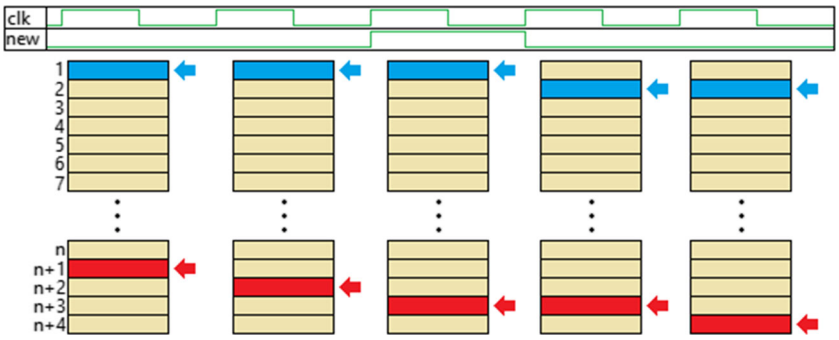

Figure 9 Label stack during reset and assignment of a new label.

The data integration itself occurs during the horizontal blanking period. In involves the following steps:

- data read from the FIFO (assigned and merged label),
- parameter read using the assigned label,
- parameter read using the merged label,
- fusing the parameters,
- resetting to zero the parameter memory at merged address,
- writing the fused parameters at the assigned address.

The whole process lasts four clock cycles, however pipelining can be introduced. The total latency of the module is given as:

$$L = \begin{cases} 2 + 2 \cdot m, \, form > 0 \\ 0, \, form = 0 \end{cases} \tag{2}$$

where: L – latency, m number of mergers in a single line.

It should be noted that in the considered system, the blanking period lasts about 60 clock cycles. For this value, the maximum number of "handled" mergers is 30 (this can be derived from Eq. 2). During analysis of the test sequences, we established that there are usually no more than 10 such events in a single line, which justifies the use of the described solution. However, if this would not be sufficient, the previously described image processing pipeline suspension could be applied.

6 Evaluation and Analysis

In this section, the proposed module is compared with the software "golden standard", the sample skin colour detection application is presented, the scalability of the solution is discussed, and finally a comparison with state-of–of-the art is provided.

6.1 Comparison with Software CCL/CCA

It should be clearly stated that the module described in Section 5 works in two modes: 2 ppc and 4 ppc. In the first one, it provides exactly the same results as a software CCL/CCA implementation. This was verified by comparing the output with the `bwlabel` function available in MATLAB. So in this case, due to the 2 ppc format, the algorithm was adapted to the FPGA architecture, but the operation itself was not modified.

In the 4 ppc mode, the module uses an additional OR operation, which is a modification to the standard CCL algorithm (more details in Section 4.2). To evaluate the impact of the introduced simplification we conducted a series of experiments. We compared the proposed pixel merging approach with ground truth data, as well as popular morphological operations with a 3×3 mask: erosion, dilation, opening, and closing, as their use also eliminates the possibility of triple merging (the main reason for using the OR approach). We used images from the PETS 2006 dataset (`http://www.cvg.reading.ac.uk/PETS2006/data.html`). Experiments showed that all simplifications/filtrations affect three key objects parameters: area, bounding box and centroid. The biggest changes were observed for small objects, but these are usually discarded from further analysis. The use of the proposed method for medium and large objects results in an area change of 4% and 2%, respectively. However, the error for the bounding box and centroid was in the range of ± 1 pixel. Such deterioration is fully acceptable in almost all vision systems.

6.2 Sample Application

One of the applications of the created CCL/CCA module is the segmentation of areas with a given colour, e.g. skin. It is a component of face detection and recognition systems that enables preselection of candidates and speeds up further image analysis.

The vision system consists of the so-called video pass-through, colour space conversions and segmentation module, CCL/CCA and visualisation of its results. All the components of the system were implemented in both 2 and 4 ppc formats. The source of the image is a computer graphics card or a 4K camera with HDMI output. Image processing is performed on the Xilinx ZCU 102 or ZCU 104 evaluation

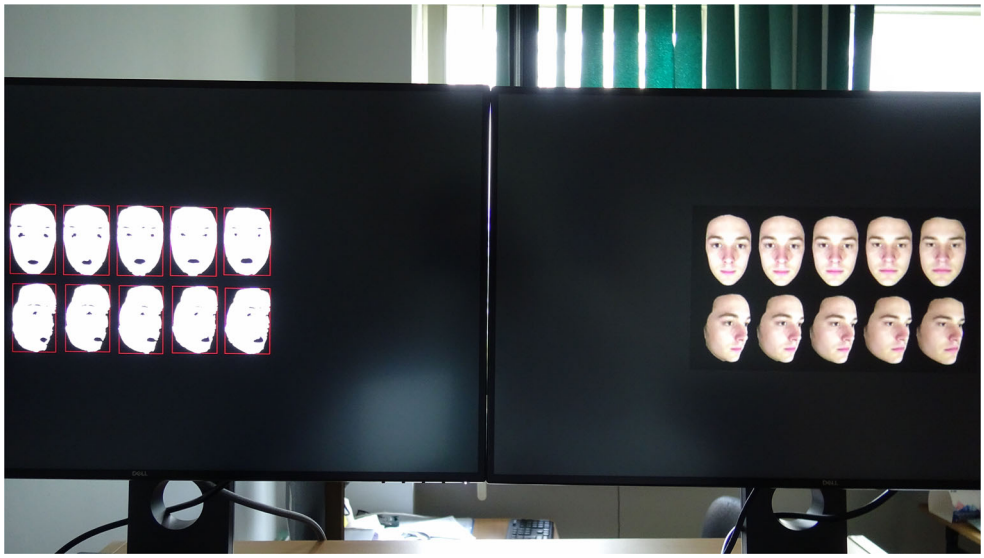

Figure 11 Working system. Right screen – input image. Left screen – CCL/CCA result.

boards with the Xilinx Zynq UltraScale+ MPSoC device. The results are displayed on a 4K monitor.

Data to the CCL module is sent with the use of AXI4-Stream bus. The modules receiving data from the HDMI input (HDMI 1.4/2.0 Receiver Subsystem and Video PHY Controller provided by Xilinx) are capable of transmitting data in 2 or 4 pixels in one clock cycle format. The AXI4-Stream bus consists of 5 signals:

- *tdata* – pixels' values,
- *tlast* – end of line (last pixel),
- *tready* – module ready to receive data,
- *tuser* – new frame,
- *tvalid* – module is ready to send data.

It should be also noted that for 4K video with 60 fps HDMI 2.0 input is required.

The input image in RGB format is subjected to a series of operations, which result in segmentation of skin colour areas – finally a binary image is obtained. To minimise the impact of variable lighting, different skin colour, occlusion and shadows, the frame is analysed in three colour spaces: RGB, YCbCr and HSV. Thus, the necessary conversions are carried out first.

Then, based on the algorithm described in the paper [7], the image is binarized by thresholding channels R, G, B, Y, Cb, Cr, H. The next step is the CCL/CCA described in the Section 5. Its result is a description of the detected objects in the form of bounding box coordinates and geometric moments m_{00}, m_{01} and m_{10}

In the final stage visualisation is carried out. First, it is checked whether the object has an area larger than a predefined threshold. If so, its parameters are saved to one of the K registers (K = 10 – it is also the maximum number of objects that can be displayed). Then, for each pixel in the video stream, it is checked whether it belongs to one of the predesignated bounding boxes. This is performed in K modules in parallel. In the last step, the OR operation is applied to the outputs from the modules – if a pixel belongs to at least one bounding box, its colour is changed (to red).

The working system is shown in Figure 11. The same output without any noticeable differences was achieved for both 2 and 4 ppc versions. The use of hardware resources is summarised in Tables 1 and 2. In Table 3 power consumption estimation obtained with the Xilinx XPower Analyzer tool for both solutions is compared. For the 4 ppc architecture a 150MHz frequency clock is used, while for 2ppc a 300MHz one. It should be noted that the proposed CCL/CCA module does not use many logic resources. Significant is only the BRAM utilisation (11 modules), as they are used for context generation and at different steps of the algorithm. Therefore, the module could be also used on a lower grade FPGA device. The only limiting factor is the possibility to receive a 4K video stream, which requires high-speed serial differential transceivers.

Table 1 Resource utilisation – 4 ppc format

Resource	Video pass-through	CCL/CCA	System
LUT	37397 (16.23%)	1712 (0.74%)	55040 (23.89%)
LUTRAM	3233 (3.18%)	21 (0.02%)	3983 (3.91%)
FF	43369 (9.41%)	796 (0.17%)	68998 (14.97%)
BRAM	6 (1.92%)	11 (3.53%)	17 (5.45%)
DSP	3 (0.17%)	0 (0.00%)	3 (0.17%)
BUFG	26 (4.78%)	1 (0.18%)	26 (4.78%)

Table 2 Resource utilisation – 2 ppc format

Resource	Video pass-through	CCL/CCA	System
LUT	30706 (13.33%)	1703 (0.74%)	40414 (17.54%)
LUTRAM	2950 (2.90%)	19 (0.02%)	3344 (3.29%)
FF	38183 (8.29%)	794 (0.17%)	51515 (11.18%)
BRAM	3 (0.96%)	11 (3.53%)	14 (4.49%)
DSP	3 (0.17%)	0 (0.00%)	3 (0.17%)
BUFG	25 (4.60%)	1 (0.18%)	24 (4.41%)

Analysing Tables 2 and 1 it can be noticed, that there is a significant difference in the resource usage for the whole system in the 4 and 2 ppc format. However, according to what was predicted, the resource utilisation for both CCL methods is roughly the same. Table 3 shows that in all cases the estimated power consumed by the Processing System (PS – i.e. ARM processors) is almost the same. This is not surprising, as the PS is only used to handle the parameters of the video passthorough and start its operation. The application is the same in every case. Furthermore, it is not unexpected, that the power consumption required to run the 4 ppc algorithm is about 21% lower than than for the 2 ppc version (taking into account only the dynamic power of Programmable Logic). Although less resources are used in the complete system (about 25% LUT and FF and 17% BRAMs), they need to work with double frequency.

6.3 Scalability of the Solution

The presented approach can be applied to any lower resolution video stream processing. In this case, using a parallel pixel mode (2 or 4 ppc) would result in a lower clock frequency of the vision system (including the CCL/CCA module). On the other hand, it would be problematic to apply the presented solution to higher resolution video streams, at least with the currently available programmable logic technology. Doubling the resolution, e.g. from Full HD to 4K, and 4K to 8K, results in quadrupling the number of pixels. Thus, in the case of an 8K video stream processing 8 pixels in 1 clock cycle at the clock frequency of 300 MHz would be required (8 ppc). 16K would require 32 pixels in one cycle at the same frequency. The presented algorithm would not be suitable for such formats (8 or 16 ppc), as the critical path for assigning labels would be too long. For even higher resolutions other solutions should be researched. One possibility is the approach with conversion to pixel series (runs) – although this solution for more pixels in one clock cycle is also not straightforward.

6.4 Comparison with Other Solutions

In Table 4 the modules discussed in Section 2, as well as the proposed one are summarised. The used "pixel merging" and other algorithmic advances allowed to obtain real-time processing for 4K @ 60 fps video stream in 4 ppc format. The system is also able to work with a 2ppc format (300 MHz clock). It should be emphasised that this performance is not a simple derivative of using a rather new device. To process this type of video stream, the well known modules had to be significantly redesigned. Moreover, the module could also be implemented in e.g. Virtex 7 series – here the main limitation is HDMI 2.0 format support (high-frequency differential input and output (for visualisation)). We did not compare the used logic resources for two main reasons. First, not all papers provide this information ([9], [2], [1]). Second, in all other cases the utilisation is rather low (one exception [5]). In our opinion, the main issue with real-time implementation of CCL/CCA modules is the design of a rather complicated control logic (label assignment, label merging, label re-use and parameter computation) and not resource optimisation (like for example in advanced image filtering – Vector Median Filter or Non-Local Means filter).

Table 3 Comparison of power consumption estimation (Xilinx XPower Analyzer)

Resource	Passthrough – 4 ppc	Passthrough – 2 ppc	System – 4 ppc	System – 2ppc
Total	4.623 W	4.761 W	5.043 W	5.370 W
PL Dynamic	0.606 W	0.691 W	1.022 W	1.295 W
PS Dynamic	2.698 W	2.698 W	2.698 W	2.698 W
PL Static	0.664 W	0.665 W	0.667 W	0.669 W
PS Static	0.100 W	0.100 W	0.100 W	0.101 W

Table 4 Comparison of the proposed solution with state of the art

Paper	Alg.	Features	Device / Clk. freq.	Real-time processing
[9] (1995)	2-pass	—	9 x Xilinx XC4010 10 MHz	512 x 512 @ 30 fps
[2] (2008)	1-pass	label re-use	Xilinx Virtex II Pro —	640 x 480 @ 100 fps
[1] (2010)	2-pass	pixel runs analysis	Xilinx Virtex 4 65 Mhz	640 x 480 @ 35 fps
[6] (2016)	1-pass	label re-use	Xilinx Virtex 6, Kintex 7 180 MHz, 150 MHz	1920 x 1080 @ 60 fps
[5] (2016)	1-pass	shift register	Altera Cyclone IV 58 - 90 MHz	1920 x 1080 @ 37 fps
[16] (2017)	1-pass	2x2 template for consecutive rows	simulation 100 MHz (synthesis)	2048 x 1536 @ 19 fps
[12] (2018)	1-pass	additional CCA statistics, internal memory	Xilinx Vertex-5 128.07 MHz	1280 x 720 @ 94 fps
[11] (2018)	1-pass	linked list, run-length encoding	Xilinx Virtex II 97.07 MHz	640 x 480 @ 30 fps
[10] (2018)	1-pass	SoC implementation, overlapping operations	Xilinx Zynq 7045 225 MHz	640 x 480 @ 700 fps
[13] (2019)	1-pass	auxiliary tables	Xilinx Zynq XC7Z020 140 MHz	640 x 480 @ 325.5 fps
[14] (2020)	1-pass	AXI4	Xilinx Zynq XC7Z020 97.3 MHz	2k x 2k @ 46.4 fps
Proposed (2020)	1-pass	2 and 4ppc support	Xilinx Zynq UltraScale+ 150 MHz and 300 MHz	**3840 x 2160** **@ 60 fps**

7 Summary

Implementing connected component labelling and analysis for 4K @ 60 fps video stream in real-time is a challenging task. Four possible approaches were discussed. Two of them were implemented and verified on the ZCU 102 and ZCU 104 development boards with a Xilinx Zynq UltraScale+ device, as a component of a skin-colour area segmentation application. The first one, working in the 4 ppc scheme, used the combining of two neighbouring pixels, which greatly simplified the required control logic. At the same time, the conducted experiments showed that this modification does not have a significant impact on the determined object parameters (bounding box, centroid, area). The second one, working in the 2 ppc scheme, processed the "full" 4K stream, however, used 21% more power. On the other hand, the resource utilisation was 25% and 17% lower for LUT/FF and BRAMs respectively.

The next step of this research is the implementation and verification in hardware of the concept described in Section 4.4. Also a more in-depth analysis of the series processing approach presented in 4.5 should be considered. An interesting option would be also a combination of both approaches. Ultimately, real-time processing of 8K video stream should be considered. This should be considered as rather challenging with the current technology, as it would require running the mentioned 4 ppc module at 600 MHz. The other possibility – 8 ppc – seems even more difficult, due to the complex label assignment logic and 4 possible mergers in one clock cycle. In addition, it was

considered to describe the presented algorithm in C/C++, use a HLS (High Level Synthesis) tool like Vivado HLS to generate the module, and then compare the results with the implementation in Verilog hardware description language.

References

1. Appiah, K., Hunter, A., Dickinson, P., Meng, H. (2010). Accelerated hardware video object segmentation: From foreground detection to connected components labelling. *Computer Vision and Image Understanding*, *114*(2), 1282–1291.
2. Ma, N., Bailey, D.G., Johnston, C.T. (2008). Optimised single pass connected components analysis, 2008 International Conference on Field-Programmable Technology 185-192.
3. Ciarach, P., Kowalczyk, M., Przewlocka, D., Kryjak, T. (2019). Real-Time FPGA implementation of connected component labelling for a 4K video stream. Applied Reconfigurable Computing, 165–180.
4. Haralick, R. (1981). Some neighborhood operations. Real Time Parallel Computing Image Analysis, 11–35.
5. Jeong, J.-W., Lee, G.-B., Lee, M.-J., Kim, J.-G. (2016). A Single-Pass connected component labeler without label merging period. *Journal of Signal Processing Systems*, *84*(2), 211–223.
6. Klaiber, M.J., Bailey, D.G., Baroud, Y.O., Simon, S.A. (2016). Resource-Efficient hardware architecture for connected component analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, *26*(7), 1334–1349.
7. Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C., Jatakia, J. (2016). Human Skin Detection Using RGB, HSV and YCbCr Color Models, International Conference on Communication and Signal Processing 2016.
8. Kowalczyk, M., Przewlocka, D., Kryjak, T. (2018). Real-time implementation of context image processing operations for 4K video stream in Zynq UltraScale+ MPSoc 2018 Conference on Design and Architectures for Signal and Image Processing.
9. Rachakonda, R.V., Athanas, P.M., Abbott, A.L. (1995). High-speed region detection and labeling using an FPGA-based custom computing platform. Field-Programmable Logic and Applications, 86–93.
10. Spagnolo, F., Frustaci, F., Perri, S., Corsonello, P. (2018). An efficient connected component labeling architecture for embedded systems journal of low power electronics and applications.
11. Tang, J.W., Shaikh-Husin, N., Ulah Sheikh, U., Marsono, M.N. (2018). A linked list run-length-based single-pass connected component analysis for real-time embedded hardware. *Journal of Real-Time Image Proceeding*, *15*, 197–215.
12. Tsai, T.-H., Ho, Y.-C., Tsai, C.-E. (2018). Implementation of real-time connected component labeling using FPGA, 2018 IEEE International Conference on Consumer Electronics-Taiwan, 1-2.
13. Spagnolo, F., Perri, S., Corsonello, P. (2019). An efficient Hardware-Oriented Single-Pass approach for connected component analysis, sensors 19.
14. Perri, S., Spagnolo, F., Corsonello (2020). A parallel connected component labeling architecture for heterogeneous systems-on-chip. Electronics 9.
15. Rosenfeld, A., & Pfaltz, J.L. (1966). Sequential operations in digital picture processing. *Journal of the ACM*, *13*(4), 471–494.
16. Zhao, C., Duan, G., Zheng, N.A. (2017). Hardware-Efficient method for extracting statistic information of connected component. *Journal Sign Processing System*, *88*, 55–65.

Marcin Kowalczyk I am a Research and Teaching Assistant at the Computer Vision Laboratory, Department of Automatic Control and Robotics, AGH University of Science and Technology in Krakow, Poland. My interests are focused on real-time vision systems for autonomous robots, with particular emphasis on object tracking. In my research I use heterogeneous computing platforms and high-level environments enabling the modelling of algorithms and designed systems.

Piotr Ciarach I am a former Automatic Control and Robotics student at the AGH University of Science and Technology in Krakow, Poland. My main areas of interest during my studies were embedded systems for video surveillance and autonomous vehicles using FPGA and Zynq System on Chip devices. In particular, I was involved in image processing algorithms implementation (connected components labelling, object segmentation and tracking).

Dominika Przewlocka-RusI am a Research and Teaching Assistant at the Computer Vision Laboratory, Department of Automatic Control and Robotics, AGH University of Science and Technology Krakow, Poland. My research is focused on hardware acceleration of neural networks in FPGA and Zynq System on Chip devices. I am interested in machine learning based vision systems (object detection and recognition, tracking) for autonomous vehicles, advanced driver assistance systems (ADAS), as well as advanced video surveillance systems.

Hubert Szolc I am a PhD student in the field of Automatic Control, Electronics and Electrical Engineering at the AGH University of Science and Technology in Cracow, Poland. The subject of my dissertation concerns the control of unmanned aerial vehicles (UAVs) based on embedded vision systems. In my research I collaborate with the Computer Vision Laboratory at the Department of Automatic Control and Robotics at AGH. I am interested in vision systems for UAVs, deep learning algorithms (especially deep reinforcement learning) and control theory.

Tomasz Kryjak I am an Assistant Professor at the Computer Vision Laboratory, Department of Automatic Control and Robotics, AGH University of Science and Technology Krakow, Poland. My research is focused on embedded vision systems implemented in FPGA and Zynq System on Chip devices (using hardware/software co-design). I am interested in advanced video surveillance systems (background modelling, foreground object segmentation, object recognition and tracking), as well as advanced driver assistance systems (ADAS), vision systems used in intelligent transportation systems (ITS) and vision system for autonomous vehicles (drones). I am a IEEE Senior Member, Steering Committee Member of the DASIP conference and TCP member of the ARC conference. I am the author and co-author of over 80 scientific papers.