High Performance Connected Components Accelerator for Image Processing in the Edge

José L. Mira[1], Jesús Barba[1], Julián Caba[1], José A. de la Torre[1], Fernando Rincón[1], Soledad Escolar[1], and Juan Carlos López[1]

Technologies and Information Systems Department, School of Computer Science, Paseo de la Universidad, 4, Ciudad Real, 13071, Castilla La Mancha, Spain.

Abstract. In image processing, a connected components algorithm is a method used to identify and label the different objects or regions present in a digital image. This algorithm can be useful for a variety of image processing tasks, such as object recognition, image segmentation, and feature extraction. This work presents the implementation of a single-pass algorithm on an FPGA-based device suitable for high-performance edge computing vision applications, the Ultra96-V2 computing board. The design and implementation of the IP core have faced challenges using the AMD-Xilinx HLS workflow and tools, which require efficient and optimized use of resources, as well as the re-engineering of the algorithm to comply with the requirements imposed by the development framework. The performance of the proposed accelerator has been thoroughly analysed using the YACCLAB benchmarking framework against a high-end and a low-end CPU. The results show an expected loss in performance due to memory and clock frequency limitations. However, concerning energy efficiency, the hardware multicore architecture outperforms the software alternatives with an improvement between two and five times, depending on the size and complexity of the images.

Keywords: Connected Components · High Performance Edge Processing · Computer Vision · Field Programmable Gate Arrays.

1 Introduction

Connected components (CC) analysis is a crucial step of many applications in computer vision, consisting of assigning unique labels to each region of an image that has been previously segmented by applying a thresholding process to differentiate objects from the background. Next, features of each region, such as area, centre of gravity, bounding box, and pixel value, are extracted based on their labels with the goal to classify each region into one of multiple classes.

Time complexity and intensive memory usage are two of the main problems to be faced when it comes to the development of a valid solution for embedded devices with a limited amount or resources or constraint power budget. Indeed, this is the context for image processing in the edge, a computing paradigm that performs image processing tasks locally, near the source of the data, rather than

sending all the data to a centralized cloud or data center for processing. Image processing in the edge can also enable real-time or near-real-time analysis of image data, which can be critical in applications such as autonomous vehicles, surveillance systems, or medical imaging.

Despite the fact there have been multiple CC algorithms over time, only single-pass algorithms are actually suitable for FPGA-based implementations due to the reduction in memory bandwidth which represents the major bottleneck for such class of devices [11]. This class of CC algorithms are also suitable for stream image processing.

In the paper [7] the authors present a scalable architecture for efficient, parallel processing of connected components by optimising performance through the use of multiple processing units. However, it is important to note that the implementation of the algorithm was performed on a high-end FPGA, which may restrict its applicability on less capable devices. In addition, extensive testing with synthetic datasets was not carried out to obtain performance comparisons with other approaches. Therefore, further research is required to evaluate the effectiveness and scalability of the algorithm in different hardware configurations and conditions.

The paper [9] presents a novel hardware architecture for connected component labeling in embedded image processing systems. The proposed architecture achieves high processing speeds by effectively managing label collisions through an innovative design that combines collision resolution and DMA core configuration in parallel. However, the authors did not perform tests with synthetic images on their architecture.

In most works in the literature, only theoretical measures derived from architecture synthesis are taken into account, in this work, the modeling, design and implementation of the single-pass CC analysis algorithm originally proposed by Bailey et al. in [1] and later optimized in [8] is revisited and adapted to Xilinx-AMD Vitis design flow. The main goal is to evaluate the performance and energy efficiency of the FPGA implementation against state-of-the art software alternatives running on high-end and low-end processor platforms. It is out of the scope of this paper to compare our hardware solution in terms of resource usage (mainly BRAM memory) with other proposals due to the use of different FPGA architectures and synthesis tools. On top of this, the proposed HLS model of the core has been generalize to enable its benchmarking by means of YACCLAB [5] framework. The architecture allows multicore configurations, worst case label counter as well as resolutions beyond the common 640x640 resolution reported in the majority of the literature [10][8], which increases the demand of resources.

2 HW-CC accelerator

Although the algorithm proposed by Bailey et al. [1] is designed taking into account FPGA-based computing devices avoiding, for example, the need to store the whole image in BRAM memories, it still presents challenges when it comes to pack it as a fully functional core for FPA-SoCs architectures such as the Xilinx

ZynQ-UltraScale+. Also, limited performance and resource analysis have been made, limiting the validation to a specific size of input images and complexity. Therefore, a complete view of the actual development of a FPGA-based solutions is missing, preventing a fair benchmarking against software solutions. On top of this, facing the modeling of the architecture and logic of the algorithm using High-Level Synthesis (HLS) technology demands the re-engineering of the original design so as to make it compliant with the semantic and syntax constraints of HLS tools.

The main contribution of this work is the development of the HW-CC accelerator, a fully parameterized IP core ready-to-use in Xilinx's FPGA-SoC platform. Parameterization allows its use for a wide range of input images and scenarios (see 3) making it our solution flexible to adapt to different application and target platform requirements (memory availability, latency, etc.). This flexibility comes to a price, mainly due to the large memory needs for storing intermediate data when working with big images; for example, the *data table*, a data structure that holds statistics of the label regions, depends on the width (W) and the height (H) being $W/2 \times H/2$ the theoretical maximum. For this reason, in the proposed solution, mechanisms had to be developed to overcome this handicap, making an low-end FPGA-based platform suitable for CC algorithms.

With regard to the parameterisation mentioned above, this mainly deals with aspects related to the memory requirements of the system to guarantee its correct operation. Firstly, by parameterising the size of the image to be processed, the sizing of the necessary internal memory structures is crucially determined. This is of vital importance, as these structures are directly dependent on the size of the image. Another key aspect of the design is the possibility to parameterise the size of the cache system. This parameterisation is relevant in several aspects, such as the number of BRAMs used by this structure and the number of possible read or write misses that can occur. These factors, in turn, impact the performance of the algorithm. In addition, the system parameterisation also provides the option of using a one-way or two-way cache system, which may be of interest depending on the specific memory access patterns of certain types of images.

This architectural proposal represents an important contribution in addressing the challenges posed by the algorithm proposed above. By implementing an instance-based division of labour approach, a proper distribution of the workload in processing a complete image is achieved. This strategy allows harnessing the potential for parallelism inherent in processing image sections simultaneously, resulting in improved system performance and scalability. By splitting the overall task into smaller instances, greater flexibility is achieved by adapting to different applications and platform requirements. However, this division implies the development of mechanisms to maintain consistency in the algorithm's collection of statistics across instances.

Fig. 1 sketches a high-level view of the HW-CC architecture which have been modeled in Vitis HLS C/C++. Memory interface uses both AXI-Stream and AXI-Memory ports so the IP core can be easily integrated with the Processor Subsystem which runs the firmware for platform and peripheral configuration,

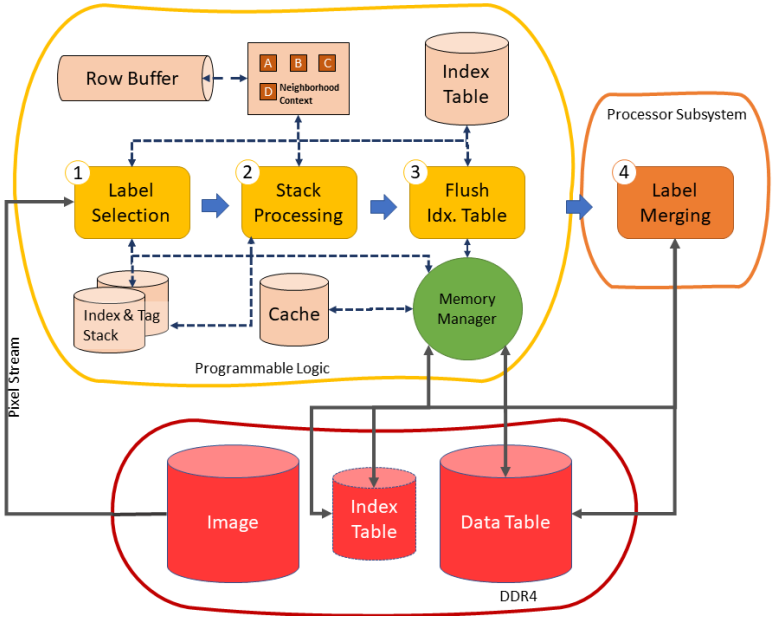

Fig. 1. Architecture of the HW-CC accelerator

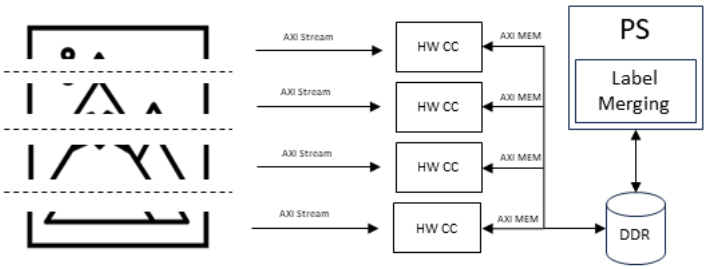

Fig. 2. High-level system composition

DMA transfer management and execution of the *Label Merging* step. Fig. 2 also shows the composition of the high-level system with the use of several instances of the accelerator working in parallel.

The HLS model defines the control and the memory elements, along with its mapping to FPGA resources, that realizes the algorithm in the most optimal way via the use of the appropriate pragmas and model parameters.

2.1 Memory Elements

- **Neigbourhood Context + Row Buffer**: These two elements are responsible for temporarily storing the information necessary to determine the labeling assignment for groups of pixels.
  On one hand, the row buffer temporarily stores the labels generated by the algorithm in the previous row, similar to a sliding window filter. The size of this row buffer in memory is directly determined by the number and size of the elements it will contain. In other words, the number of elements is directly related to the pixel width of the image, and the size of these elements coincides with the number of bits needed to encode the maximum possible number of labels. This, in turn, depends on the number of rows and columns in the image. This parameterization makes the size of this structure entirely customizable in the design.
  On the other hand, the neighborhood context is formed by the registers that spatially correspond to the pixels adjacent to the one being evaluated. These registers play a crucial role in selecting the value assigned to each pixel.
- **Index Table**: An array that holds the value of the tag to which a given index points. The size of this array can be parameterized in the model to fit the typical input image requirements. Another parameter of the model is the maximum number of labels, which determines the size of the word for the Index Table.
- **Index & Tag Stack**: This is a data structure used to merge two regions after each row is processed based on the values of their tags. In order to exploit the fact that index and tag information access is independent, it was decided to apply the pragma *ARRAY PARTITION* in order to allow parallel access to both values. The size of the stack is also a parameter of the model.
- **Memory Manager + Data Table + Cache**: The Data Table is the structure in which the statistics related to the labels are stored. In the proposed implementation the coordinates of the upper-left corner, width and height of each region is provided. The memory requirements of the Data Table would rapidly exhaust the available BRAMs in the FPGA fabric, specially for high resolutions. In order to make the HW-CC accelerator suitable for a variety of sizes and complexities, it has been decided to implement a simple cache system that allows the core handle large image sizes, without sacrificing the speed of access to the data. This module is also in charge of checking, reading and writing the blocks in the cache memory with respect to the on-board RAM when necessary. Given that during the algorithm's operation, it gradually assigns new labels to the new regions it passes through. In this process,

space is occupied in the Data Table corresponding to these new labels. At the same time as this process is taking place, the merging of several previously assigned tags is also taking place. In this way, access to the entries in the table is usually done in an alternating manner. When implementing the cache, this memory access pattern is a problem because there are areas of the image in which data table entries belonging to different data blocks are accessed consecutively. This situation results in a high number of cache misses, with the respective execution time spent in having to replace the data block multiple times. Based on these characteristics of the memory access patterns, it was decided to implement a 2-way cache system in order to avoid unnecessary replacement of data blocks.

2.2 Algorithm Stages

Label Selection (HW) This stage is performed once for each pixel of the image, where the Neighbourhood Context is evaluated to determine which label will be generated in a given pixel. The following situations may arise in this process:

- Generate a new label, whereby a new record has to be created in the data table. To achieve this, a counter is employed, starting with the value 1, and it increments each time this operation is performed. This counter is responsible for being assigned to the label. In this process, the row and column values in which this label is contained are also stored.
- Assign an existing tag, which requires accessing the data table to update the statistics related to that tag.In this process, the parameters of the row and column in which the label is contained, as well as its area, are updated.
- There are two possible tags in the Neighbourhood Context so they are stored in the Index & Tag Stack to be resolved later, the smaller of the two is assigned to the pixel and the statistics are updated in the data table.

Stack Processing (HW) This stage is performed for each row of the image that has completed the algorithm. It accesses the Index & Tag Stack to perform the merge process between tags, which accesses the index table to update the corresponding indexes. Thanks to the implementation of this structure, potential conflicts that may arise in the resolution of labels that have been merged when consumed from the row buffer in the next row are resolved. When merging these labels, the values related to the rows and columns in which the new merged label is contained, as well as its area, are also updated.

Flush Index Table (HW) Once the algorithm has finished traversing the image, the cached data blocks are updated in DDR. Subsequently, the entire set of information generated by the algorithm needs to be transferred from internal memory to the DDR memory of the SoC. This step is essential for the data to be readily available and integrated into an image analysis system.

Label Merging (SW) Once the FPGA component is finished, the processor goes through the Index Table and the Data Table contained in the RAM memory to perform the final merging process between the different regions in order to obtain the final number of labels and their statistics. This step was moved to the software realm due to be a control intensive tasks with irregular memory pattern access, an scenario where the processor performs better than the FPGA.

3 Results

In this section, we present the results of our performance evaluation of various connected components labeling algorithms using the YACCLAB [5] benchmarking framework. Our evaluation aimed to compare the speed and energy efficiency of the proposed HW-CC accelerator IP against different algorithms on a set of test images provided by YACCLAB. Software benchmarks were run on: (1) a 12th Gen Intel(R) Core(TM) i7-12700K (3.6GHz, 64GB DDR5) executing an Ubuntu 20.4 GNU/Linux distribution ; and (2) a Intel(R) Celeron(R) N5095 (2.0GHz, 8GB DDR4) executing an Ubuntu 22.4 GNU/Linux distribution. In both cases the test suite was compiled with GCC 11.3. By incorporating both a high-end workstation processor and a low-end CPU commonly present in commercial edge computing solutions, experimental validation can provide a better understanding of the proposed accelerator's capabilities.

The test suite comprised the out-of-the-box connected component algorithms that come with YACCLAB, namely: Scan Array-based with Union Find [12] (SAUF), Block-Based with Decision Tree [6] (BBDT), Pixel Prediction [4] (PRED), Directed Rooted Acyclic Graph [3] (DRAG) and Spaghetti Labelling [2].

The dataset used for experimental validation comprised a selection of the YACCLAB 2D subset which consists of binary images with labeled connected components. The dataset includes both synthetic images and real-world images from various sources. In this work, we test the HW-CC accelerator for edge image processing over the MIRflickr (25K images, average size and connected components: 0.17 megapixel and 495, respectively), Hamlet (104 images), Tobacco800 (1290 documents, 150-300 DPI and 1200x1600 to 2500x3200 resolutions), 3DPeS and synthetic random noise image subsets. A special note has to be made about the latter it allows to objectively test the scalability and efficacy of different algorithms. This subset provides ten images for each combination of size (32x32 up to 4096x4096 in steps of x2) and density (10% to 90%), resulting in a total of 720 images.

The Ultra96-v2 prototyping platform has been used to deploy and test the processing accelerator core. The Ultra96-v2 board is based on the Xilinx Zynq UltraScale+ MPSoC which integrates programmable logic and a four-core 64-bit Arm Cortex-A53 processor. The ZU3EG chip embedded in the platform is a low-end FPGA with modest capabilities and low power budget which makes it suitable for edge computing applications. Version 2021.1 of the Xilinx toolchain (Vitis HLS, Vivado and Vitis Unified Software Platform) has been used to model the HW-CC IP core, design and synthesize the platform and deploy the whole

solution. No operating system support is required for running the firmware which executes on baremetal.

First, we evaluated the performance of the proposed solution against the real-world subsets selected. Though the architecture of the HW-CC accelerator is flexible and allows multiple parameter configuration, for this experiment an aggressive strategy was proposed in order to obtain the maximum performance. Therefore, row buffer size was established to 4K words, label data memory to 64K and cache block memory to 1K. The number of accelerator cores instantiated in the FPGA fabric is four, which raises the utilization of BRAM resources to 83% of the total available in the device. Post-synthesis results showed that it was possible to configure a 175MHz clock to drive the logic. However, we establish a conservative approach in this regard in order to avoid unexpected on-board behaviour. Thus, a safe 150MHz clocking configuration was set.

Table 1. Average run-time test (ms) for real-image subset

	MIRFlickr	3DPes	Hamlet	Tobacco	Avg. Gain
HW-CC-150	1.71	2.73	19.77	36.92	-
Intel i7-12700K					
SAUF	0.44	0.61	4.96	7.65	**-4.29x**
BBDT	0.36	0.35	3.44	4.93	**-6.45x**
PRED	0.54	0.72	6.92	10.52	**-6.33x**
DRAG	0.36	0.35	3.44	4.91	**-6.46x**
Spaghetti	0.2	0.18	1.89	2.68	**-11.99x**
Intel Celeron N5095					
SAUF	1.46	2.76	22.19	37.33	**-1.01x**
BBDT	0.88	1.33	13.86	22.96	**-1.76x**
PRED	1.49	2.87	22.96	38.69	**+1.02x**
DRAG	0.87	1.33	13.86	23.03	**-1.76x**
Spaghetti	0.41	0.53	5.16	7.89	**-4.46x**

Table 1 presents the average execution times for all the targeted image subsets. The results show a significant loss in performance compared to the i7 processor, and a moderate decrease in performance in the case of the Celeron processor.

The most attractive feature of the Zynq UltraScale+ architecture for edge computing applications is its efficiency in terms of power consumption. Average energy intake for our HW-CC has been measured using the interface provided by the on-board *Platform Management Unit* resulting in 3.74W during test execution. In the case of the i7 processor, *powerstat* command line tool has been used to monitor the increment in power consumption through the RAPL (Running Average Power Limit) interface; an average of 31.4W was observed. The Celeron-based computing platform had, however, to used a external energy

consumption meter which reported a sustained increment of 10.3W during test execution.

Table 2. Gain in the average energy efficiency (J) for Real-Image Subset. HW-CC versus Intel i7 & Celeron.

	SAUF	BBDT	PRED	DRAG	Spaghetti
Intel i7	1.87	1.24	2.56	1.24	0.67
Intel Celeron	2.87	1.75	2.97	1.76	0.63

Table 2 shows the comparison of the efficiency, expressed as Joules, broke down by subset and algorithm. Average results expose a consistent cross-platform gain in all cases but the Spaghetti algorithm which outperform the rest due to its low execution time. It is important to note that all the algorithms evaluated achieve identical results in terms of region labelling. This observation demonstrates the consistency and accuracy of all the approaches analysed in this study in terms of their ability to identify and assign labels to the different regions present in the image.

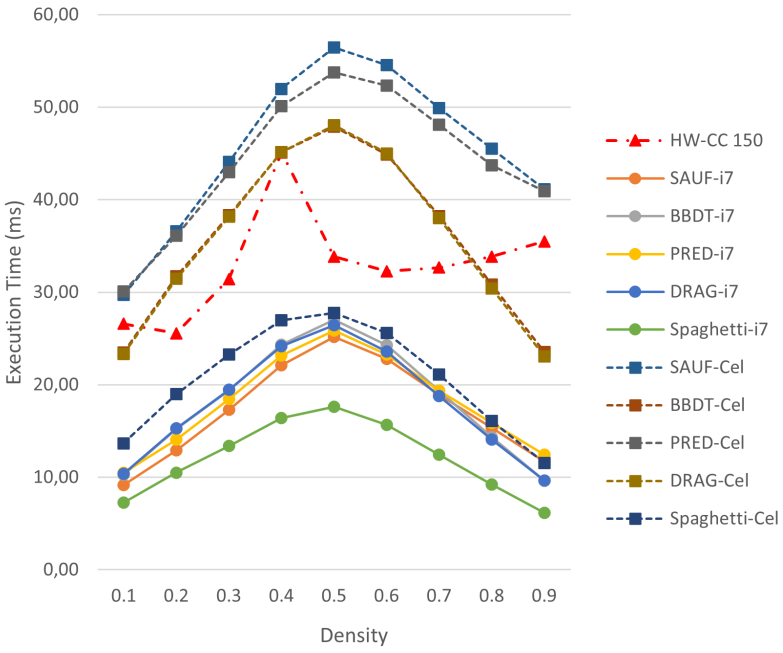

Fig. 3. Density test results.

In order to validate the performance and scalability of a solution, YACCLAB includes a set of random generated images that allows the execution of density and size tests. The density and size values mentioned above refer directly to the number of regions to be labelled in the image. These values are randomly generated and represent the granularity of the noise present in the image. The impact of this test is mainly evident in the total number of regions present in the image, as well as in the size of each region and the memory access pattern used by the algorithm. These aspects are important considerations as they influence the performance of the system. Figures 3 and 8 depict a comparison of run-times for each software algorithm and the 4-core 150MHz version of the proposed HW-CC accelerator. In the figure 7, the difference between images with different density factors can be observed.

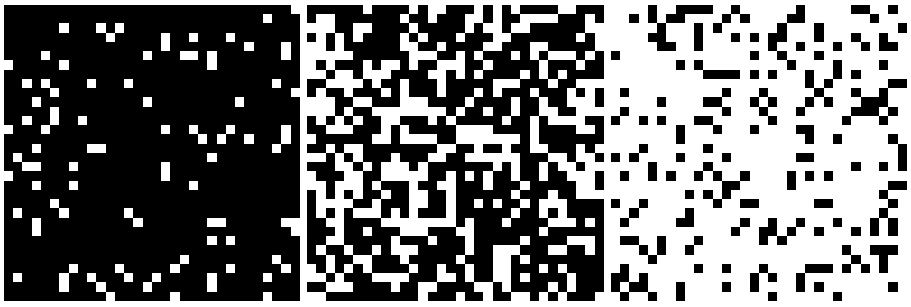

Fig. 4. 0.1 **Fig. 5.** 0.3 **Fig. 6.** 0.7

Fig. 7. Example of synthetic images in YACCLAB dataset with different intensities. It directly stresses the algorithm performance.

Average execution time is stable for HW-CC with slight higher values as as the density increases broken by a peak on 0.4 foreground density images as it can be seen in Fig. 3. Comparison to i7 processor shows a performance degradation of $\approx -2x$ for all algorithms but Spaghetti which reaches $-3.03x$ with a maximum loss of $5.76x$ for 0.9 density images. However, HW-CC is faster than the Celeron processor in all cases - SAUF (35.9%), BBDT (4.2%), PRED (32.6%), DRAG (3.4%) - but, again, Spaghetti ($1.72x$ slower in average with a maximum loss of $3.07x$ for 0.9 density images).

Regarding size tests, the performance of the proposed accelerator maintains the same pattern across processors and algorithms. As it can be seen in Fig. 8, HW-CC core maintains the linear dependency of execution time as its software counterparts. Average performance loss versus i7 processor is $-2.14x$. Taking into account that Intel i7 processor power consumption is approximately ten times the energy the FPGA needs, it results in a $\approx 5x$ improvement in terms of energy efficiency while maintaining a fair computing capability for a device to be deployed at the edge of the computing infrastructure. As to the Celeron

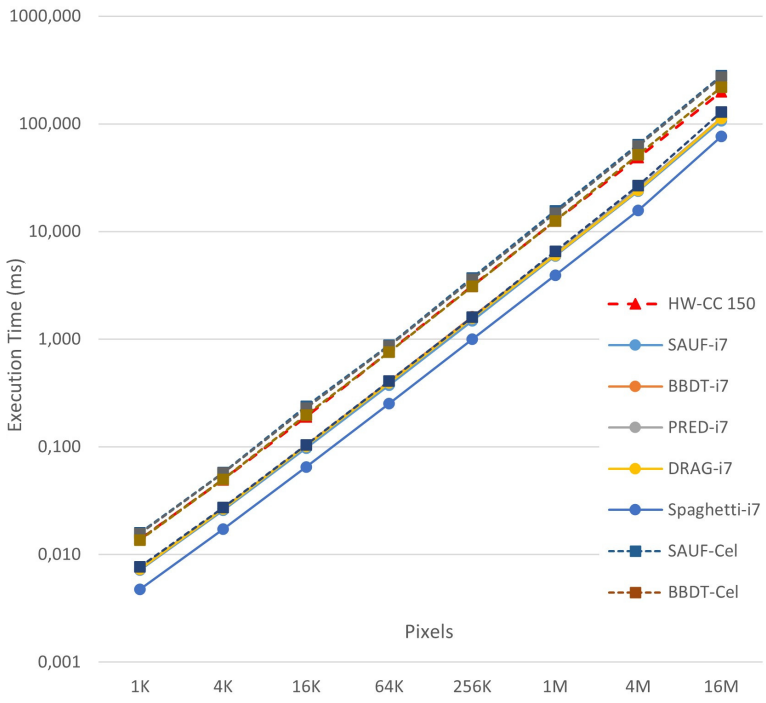

Fig. 8. Size test results.

CPU, average loss in performance is -1.09, highly penalized by the efficiency of Spaghetti algorithm ($1.82x$ better). In the other cases, HW-CC is 10% faster. Overall, our FPGA-based solution is ≈ 2.5 times more efficient than the Celeron computing platform.

4 Conclusions

In this work, the design and development of a hardware accelerator architecture for efficient analysis and labeling of components in images have been carried out, emphasizing the trade-off between computational power and energy consumption. Targeting embedded computing platforms deployed at the edge of information acquisition, processing, and transmission infrastructure (e.g., IoT platforms), the analysis of results considers not only performance but also the energy needs of the solution. Ultimately, the envisioned platforms for the final application involve a processing device based on an SoC-FPGA and powered by a battery, emphasizing the importance of energy efficiency.

Comparing with previous contributions in the state of the art is challenging as many works often lack necessary information, target FPGA devices from previous generations, and lack standardization in terms of conducted tests and utilized images, which directly impacts processing latency dependent on complexity and size. As a next step, we will work on developing a comparative framework and information compilation to objectively evaluate our solution in comparison to other state-of-the-art works.

Acknowledgment

This research has been funded by the European Union's H2020 programme under grant agreement no. 857159 (SHAPES project) and by the Spanish Ministry of Economy and Competitiveness (MINECO) under the TALENT project (PID2020-116417RB-C4, subproject 1) and the MIRATAR project (TED2021-132149B-C41).

References

1. Bailey, D., Johnston, C.: Single pass connected components analysis. Proceedings of Image and Vision Computing (01 2007)
2. Bolelli, F., Allegretti, S., Baraldi, L., Grana, C.: Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling. IEEE Transactions on Image Processing **29**, 1999–2012 (2020). https://doi.org/10.1109/TIP.2019.2946979
3. Bolelli, F., Baraldi, L., Cancilla, M., Grana, C.: Connected components labeling on drags. In: 2018 24th International Conference on Pattern Recognition (ICPR). pp. 121–126 (2018). https://doi.org/10.1109/ICPR.2018.8545505
4. Grana, C., Baraldi, L., Bolelli, F.: Optimized connected components labeling with pixel prediction. In: Blanc-Talon, J., Distante, C., Philips, W., Popescu, D., Scheunders, P. (eds.) Advanced Concepts for Intelligent Vision Systems. pp. 431–440. Springer International Publishing, Cham (2016)

5. Grana, C., Bolelli, F., Baraldi, L., Vezzani, R.: Yacclab - yet another connected components labeling benchmark. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 3109–3114 (2016). https://doi.org/10.1109/ICPR.2016.7900112
6. Grana, C., Borghesani, D., Cucchiara, R.: Optimized block-based connected components labeling with decision trees. IEEE Transactions on Image Processing **19**(6), 1596–1609 (2010). https://doi.org/10.1109/TIP.2010.2044963
7. Klaiber, M.J., Bailey, D.G., Simon, S.: A single-cycle parallel multi-slice connected components analysis hardware architecture. Journal of Real-Time Image Processing **16**(4), 1165–1175 (Aug 2019). https://doi.org/10.1007/s11554-016-0610-2, https://doi.org/10.1007/s11554-016-0610-2
8. Ma, N., Bailey, D.G., Johnston, C.T.: Optimised single pass connected components analysis. In: 2008 International Conference on Field-Programmable Technology. pp. 185–192 (2008). https://doi.org/10.1109/FPT.2008.4762382
9. Spagnolo, F., Frustaci, F., Perri, S., Corsonello, P.: An efficient connected component labeling architecture for embedded systems. Journal of Low Power Electronics and Applications **8**(1) (2018). https://doi.org/10.3390/jlpea8010007, https://www.mdpi.com/2079-9268/8/1/7
10. Spagnolo, F., Frustaci, F., Perri, S., Corsonello, P.: An efficient connected component labeling architecture for embedded systems. Journal of Low Power Electronics and Applications **8**(1) (2018). https://doi.org/10.3390/jlpea8010007, https://www.mdpi.com/2079-9268/8/1/7
11. Walczyk, R., Armitage, A., Binnie, T.D.: Comparative study on connected component labeling algorithms for embedded video processing systems. In: International Conference on Image Processing, Computer Vision, & Pattern Recognition (2010)
12. Wu, K., Otoo, E.J., Suzuki, K.: Optimizing two-pass connected-component labeling algorithms. Pattern Analysis and Applications **12**, 117–135 (2009)