Rosli, F.; Elhossini, A.; Juurlink, B.

Real-Time Vision System for License Plate Detection and Recognition on FPGA

Rosli, F.; Elhossini, A.; Juurlink, B. (2015). Real-Time Vision System for License Plate Detection and Recognition on FPGA. PARS: Parallel-Algorithmen, -Rechnerstrukturen und -Systemsoftware, 32(1), pp. 69-79. https://hdl.handle.net/20.500.12116/1930

Real-Time Vision System for License Plate Detection and Recognition on FPGA

Faird Rosli , Ahmed Elhossini, Ben Juurlink

Embedded Systems Architectures (AES)
Technical University of Berlin
Einsteinufer 17
D-10587, Berlin, Germany
{mohd.f.mohdrosli1,ahmed.elhossini,b.juurlink}@tu-berlin.de

Abstract: Rapid development of the Field Programmable Gate Array (FPGA) offers an alternative way to provide acceleration for computationally intensive tasks such as digital signal and image processing. Its ability to perform parallel processing shows the potential in implementing a high speed vision system. Out of numerous applications of computer vision, this paper focuses on the hardware implementation of one that is commercially known as Automatic Number Plate Recognition (ANPR).Morphological operations and Optical Character Recognition (OCR) algorithms have been implemented on a Xilinx Zynq-7000 All-Programmable SoC to realize the functions of an ANPR system. Test results have shown that the designed and implemented processing pipeline that consumed 63 % of the logic resources is capable of delivering the results with relatively low error rate. Most importantly, the computation time satisfies the real-time requirement for many ANPR applications.

1 Introduction

In recent years, the significant evolution of computer vision can be seen as it is making its way into an increasing number of application domains. The research and development in the field of computer or machine vision has defined methods for processing and analyzing images from real world to provide human capabilities of understanding images to machines and robots. There is a strong and growing demand for computer vision systems in the automotive domain. Intelligent cars that are available in the market nowadays are equipped with various camera-based driver assistance systems such as lane detection, night view assist, pedestrian detection and traffic sign recognition [Gr13].

These applications are required to work reliably in a large range of lighting and climatic conditions and to process a very high frame rate video signal in real-time. Besides those above-mentioned applications, the Automatic Number Plate Recognition (ANPR) is also one of the computer vision applications that is widely used in the automotive domain. However, it is better known as a surveillance technology rather than a driver assistance system. Vehicles are usually identified by their registration number which are easily readable by humans. But for machines, a plate number is a grey

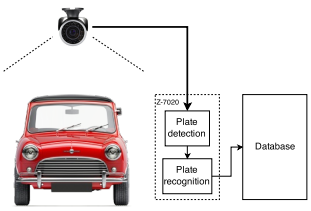


Figure 1: System Configuration for ANPR on FPGA.

image defined as a mathematical function that represents light intensity at a certain point within an image [Ma07].

The main objective of this paper is to develop a framework of embedded components for custom computer vision applications that run on a reconfigurable hardware and operate in real-time. The proposed framework was customized for an application in the automotive domain. The ANPR application is implemented on FPGA as illustrated in Figure 1 which is suitable for a wide range of applications. The application is constructed using basic computer vision and image processing algorithms. Various hardware cores are developed for each algorithm that later can simply be reused by other vision applications.

The paper is organized as follows: Section 2 gives overview of the related work to this paper. Section 3 describes the proposed methods for license plate detection and recognition. Section 4 reveals the architecture of each processing block. Section 5 presents the testing procedure and analysis of the implemented license plate detection and recognition system. Finally, Section 6 summarizes the whole work that is done so far and concludes the work.

2 Related Work

In this paper we investigate employing FPGA to implement a framework for computer vision and specifically for automatic license plate recognition. The most common approaches applied to detect the license plate within an image are the combination of edge detection and binary morphology as explained in [SGD06]. The detection rate with this method is highly affected by the quality of the image. They are based on the assumption that car plates have strong edges that will survive strong filtering. Unfortunately, this is not very effective for urban environment. License plate detection and segmentation algorithm with histogram projection as proposed in [As13] and [Hs09] is found to be challenging especially when the image contains a lot of details in the background. Defining the threshold value require a few steps of mathematical calculation and the process of finding the peak may introduce some delay in the processing. A simpler approach that produces satisfactory result is by using greyscale morphology that is already used in [Iw] and [Od]. Optical character recognition (OCR), that is normally used to translate images of handwritten text into machine encoded text, is applicable for the recognition of license plate characters. The work in [ZDF10] divides a handwritten character into several rectangular zones to obtain a 13-element feature vector. Each element represents the number of foreground pixel in each defined zone. In [SDR12] a similar method is used to divide a 32 x 32 pixel character image into 16 zones. The authors have proposed eight directional distribution features which are calculated for each zone. Three different classification methods, which are artificial neural networks (ANN), support vector machine (SVM) and k-nearest-neighbour (KNN) are used to recognise the characters.

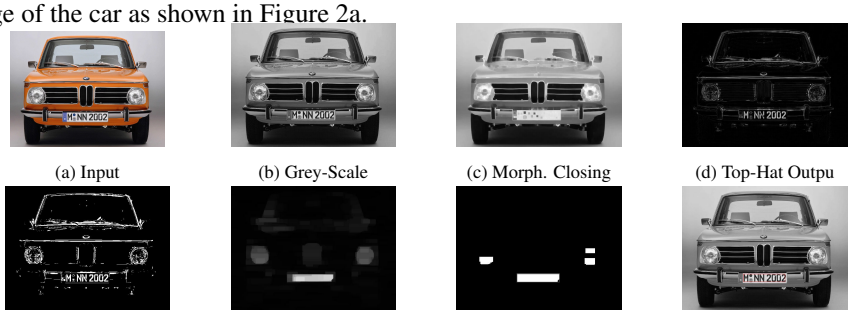
Some of the methods and algorithms described above can be combined and implemented on embedded hardware platform. For example, license plate detection and recognition on an embedded DSP platform was introduced in [ALB07]. The total processing time, beginning from image acquisition until character classification requires 41.35 ms. Another implementation of plate localisation on Xilinx Virtex-4 was described in [ZBR11] and is capable of processing one image in 3.8 ms and it consumes less than 30% of the on-chip resources. No recognition stage was presented in that work. A complete ANPR system is implemented on FPGA in [Je13]. The system utilizes 80 % of the Xilinx Virtex-4 LX40 re-

sources and is capable of processing a standard definition image (640 x 480) in less than 10 ms. Algorithms for license plate detection are also developed in [TKA07] using Handel-C and then translated into Verilog HDL with Celoxca's DK4 to implement on Virtex II Pro. Performance comparison between the software and the hardware implementation is stated at the end of the paper. The resource utilization for their hardware implementation is about 2 to 3 times more than the work in [ZBR11] and yet requires longer processing time for an input image with a smaller resolution. However,their results show that the same algorithms execute 4 times faster with the hardware than the software.

In this paper we adopted some of the listed methods and algorithms to design and implement a pipelined processing system for computer vision on a reconfigurable hardware. The proposed work presents hardware modules for various morphology based filters, along with connected component analysis and k-means classifier. These components are arranged in a single pipeline for ANPR. These components are designed to be easily used in various computer vision applications.

3 Proposed Algorithm

In this section we propose an algorithm for ANPR. The input to the algorithms is a frontal image of the car as shown in Figure 2a.



(a) Input (b) Grey-Scale (c) Morph. Closing (d) Top-Hat Outpu

(e) Binary Output (f) Morph. Cleaning (g) Cleaned Output (h) Bounding Box

Figure 2: Output of Various Processing Stages (Actual Output of the Hardware System)

3.1 License Plate Detection

3.1.1 Top-Hat Filtering

The input image is converted to grey-scale (Figure 2b). Several stages of mathematical morphology are chosen to locate the license plate of the vehicle in the image. Firstly, the closing operation with a structuring element of 7 x 7 pixel is applied to erase the characters of the license plate (Figure 2c). When image subtraction is performed between the resulting image and the initial grey-scale image, an image as shown in Figure 2d is obtained. This operation is known as black Top-Hat morphology. It returns an image containing objects or elements that are smaller than the structuring element and darker than their surroundings. Since European license plates mostly have white background and the characters are black in colour, they will remain as foreground objects in the output image. A binary image (Figure 2e) is obtained from the Top-Hat image by using thresholding.

3.1.2 Background Cleaning and Plate Segmentation

Grey-scale morphology is applied to the top-hat image to find the region that contains the vehicle license plate. The license plate location can be detected roughly with a closing

Figure 3: Bounding Box for Each Character.

Figure 4: A letter ”K” is divided into 8 zones.

operation. The structuring element for this operation has a size of 1 x 45 pixel. The length is chosen such that it has at least twice the length of a license plate character. After that, unwanted elements that does not belong to the license plate area are removed by using morphological opening with a rectangular structuring element 15 x 25 pixel (Figure 2f). The resulting image contains several light areas, and the area of the license plate appears to be lighter than others. Thresholding is applied to produce a binary image that maintains the license plate area and suppresses the darker regions (Figure 2g). Finally, binary morphology dilation with a structuring element of 5 x 5 pixel is applied to enhance the edges of the binary image. Plate segmentation is performed by applying the connected component labelling algorithm to distinguish these regions. A bounding box algorithm is applied to find the rectangular boundary that encloses each region.

3.2 License Plate Recognition

3.2.1 Character Segmentation

The process of finding characters on a license plate is actually the same as finding the license plate in the input image. Connected component labelling is applied to give a label for each character and a bounding box that encloses each character is defined so that it can be segmented from the image and sent to the recognition (classifier) unit. The bounding box that encloses a detected object must have a minimum width and height to distinguish between characters and noise. The result of character segmentation is shown in Figure 3.

3.2.2 Feature extraction

An image of a license plate character is partitioned into 8 zones (Figure 4) and the number of foreground pixels in each zone (pixel density) is calculated. Thus, an image can have a feature vector of at least 8 elements. Additionally, several types of edges of each extracted character can also be determined to produce a feature vector of the character. There are 14 different types of edges (Figure 5) and the number of occurrences of each type is calculated for each zone. Combined with the pixel density of each zone a total of 72 elements in the features vector of each character is calculated.

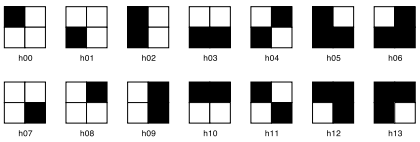


Figure 5: Edge Types in Feature Extraction

3.2.3 Character Classification

The detected characters are classified using a simplified k-means clustering algorithm. For each alphabetical character, at least 5 sample images are taken and a feature vector is extracted from each image. A mean vector for a character is determined by using the extracted feature vectors. Each mean vector is stored in a database which is used for the classification task. There are 36 types or classes of characters that the designed vision system must be able to recognize (26 alphabetical characters + 10 numerical digits).

4 Implementation

In this section we present the details of the proposed architecture. Each component is modelled using VHDL and simulated using Xilinx ISE development tools from Xilinx. The target device is Xilinx Artix-7 architecture. The design is assumed to stream the image data pixel by pixel in every clock cycle which makes it suitable for streaming data from various image sources such as cameras.

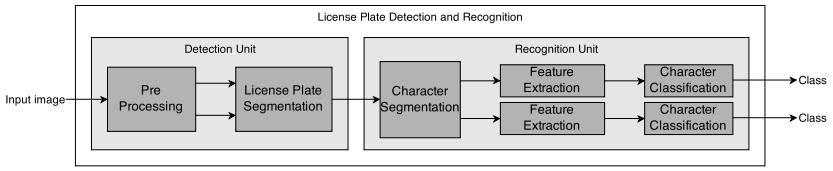


Figure 6: The Architecture of the Processing Pipeline

4.1 Complete Processing Pipeline

The complete processing pipeline shown in Figure 6 is composed of two main parts. The first part is the detection unit, which is composed of the pre-processing and the license plate segmentation units. It receives a stream of input images in RGB format and produces a binary image of a license plate. The second part is the recognition unit that receives the binary image and performs character segmentation in its first processing stage. Two license plate character images are segmented simultaneously. Therefore parallel execution of character classification is possible.

4.2 License Plate Detection Unit

In this stage the position of the license plate is detected using a series of morphological operations and connected component labelling as explained earlier. Two main units in this stage: the pre-processing unit, and segmentation unit.

4.2.1 Image Pre-processing Unit

During the pre-processing stage (Figure 7), the grey converter unit prepares a grey-scale image to the black Top-Hat unit by converting the colour space of the RGB image. The background cleaning and thresholding of the resulting image of black Top-Hat morphology are executed in parallel. This processing unit will produce two different output images similar to that are shown in Figure 2e and Figure 2g.

Morphological Filter: The architecture for grey-scale morphology is based on the design in [Ba11]. For a rectangular structuring element, efficient separable implementations are applicable as shown in Figure 8. The implementation of this architecture allows users to select between various morphological operation and structuring elements sizes using a configuration word. Additionally, the length of the row buffer, which is implemented using block RAM can also be adjusted via the configuration word. To implement opening and closing, the proposed architecture must be duplicated and ordered accordingly.

Top-Hat Filtering: The Top-Hat filter is performed by subtracting the grey-scale image and morphological image . The architecture of a separable morphological filter with structuring element 7 x 7, explained in the previous section, is instantiated twice here. One is used as dilation and the other as erosion. The architecture for the FPGA implementation of the Top-Hat transform is shown in Figure 9.

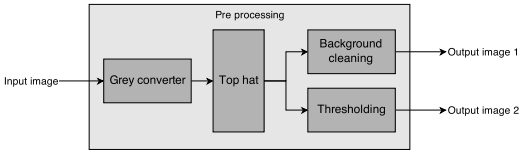


Figure 7: Image Pre-processing Unit

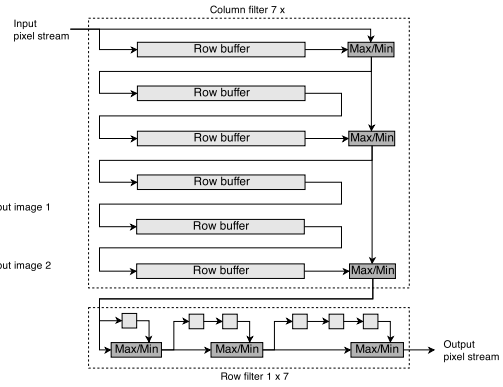


Figure 8: 7x7 Configuration of the Morphological Filter Unit

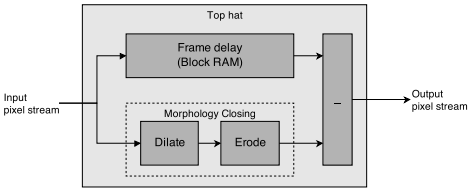


Figure 9: Top-Hat Morphological Filter.

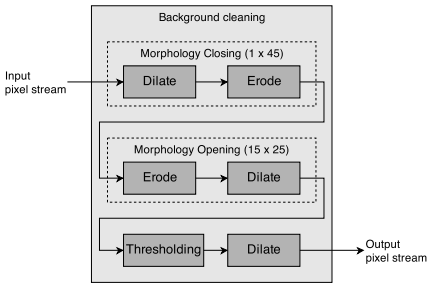


Figure 10: Background Cleaning Unit

Background Cleaning: The background cleaning consists of two morphological operations, which are opening and closing. The morphological closing is done with a horizontal filter, which has a structuring element of 1 x 45, whereas the opening is done with a normal rectangular structuring element of 15 x 25. Additionally, it also contains a global thresholding unit and a binary morphology, which is used to dilate the resulting binary image from the previous processing stages. The processing pipeline for the background cleaning is shown in Figure 10. The architecture shown in Figure 8 is employed in all stages.

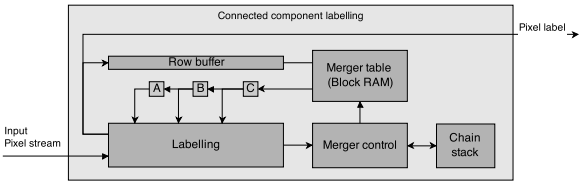


Figure 11: Connected Component Labelling Unit

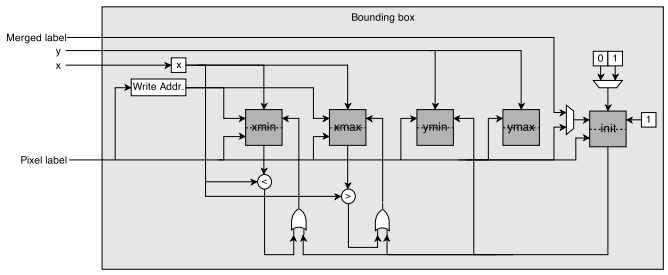


Figure 12: The Bounding Box Module

4.2.2 License Plate Segmentation Unit

Connected Component Labelling: Connected component labelling is a commonly used algorithm segment images based on the neighbourhood of pixels. Connected component labelling gives a label to every group of pixels that are neighbour to each other. The algorithm requires at least two passes to process pixels. A modification for connected component labelling is introduced in [MBJ08] that provides a single pass algorithm that eliminates the need of frame buffering and significantly reducing the latency. A simplified block diagram of the developed unit based on the single pass connected component labelling is shown in Figure 11. The neighbourhood context within a window of the size 2x2 provides the labels of the adjacent pixels to the current pixel. Unlike the architectures proposed in [MBJ08] and [JB08], the neighbourhood pixel labels are stored in registers A, B and C. These are shifted with every clock cycle as the window is scanned across the image. The merger control block updates the merger table when two objects are merged and handles new labels.

Bounding Box: Figure 12 shows the implementation of the bounding box processor for a binary image that contains multiple labels. Compared to other segmentation algorithms such as watershed and Hough transform, it provides low processing and computation cost. When the first pixel of an object is detected, the coordinates of that pixel is loaded into x_{min}, x_{max}, y_{min} and y_{max}. The y-coordinate of the following object pixel is stored into the y_{max}. The current x-coordinate is compared with x_{min} and x_{max}. At the end of the frame, the four registers indicate the extent of the object pixels within the image [Ba11].

4.3 License Plate Recognition Unit

Characters in the license plate are segmented using the same method used in plate segmentation. The image of each character is processed by several modules to extract features that will be used for character classification.

4.3.1 Feature Extraction Unit

Zoning Unit: The character sub-image is divided into eight zones as shown in Figure 4. Zoning of a character image is possible when its width and length are known. These parameters are calculated during the segmentation of a character using bounding box. The borders that define the zones can be calculated by dividing the width and height by 2 and 4 respectively.

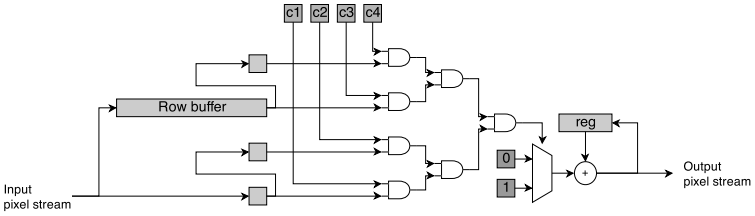


Figure 13: Edge Matching Unit

Edge Matching Unit: As explained earlier, there are 14 types of edges that an image can have. Each type is detected using the circuit shown in Figure 13. The coefficient of each field is initialized in registers c1 to c4 via a configuration word. The row buffer is used to store the previous row of the image. The result of the comparison will switch the multiplexer that selects the operand for the adder. If an edge match occurs, the register that stores the number of detected edges will increment. It will be reset when a new character image is received.

Feature Extraction Unit: As explained in section 4.3.1, a character image is divided into eight zones. Therefore, feature extraction is done for each zone instead of the whole image. The hardware architecture for feature extraction of a character image is implemented according to Figure 14. The zone selector acts like a demultiplexer that receives a zone number from the zoning unit and enables a corresponding zone. Each zone will produce a feature vector that contains 9 elements. Therefore, the architecture will produce a total number of 72 feature elements for a character.

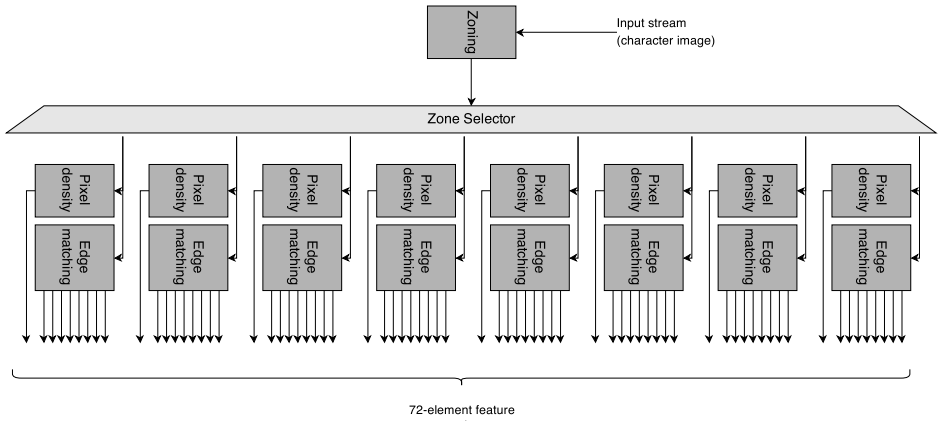


Figure 14: Feature Extraction Module.

4.3.2 Classification Unit

The classification algorithm requires a database that stores the mean vector for each license plate character. Entries of this database are stored in array of registers. The classification unit includes 36 arrays for each class. The absolute error between a feature vector and each mean vector is determined by doing element-wise subtraction of the feature vectors. The

sum of absolute error is calculated by simply adding up all the elements in the error vector. Pairwise classification algorithm is used in this case, where two errors are compared at the same time. The pairwise classification unit is implemented as shown in Figure 15.

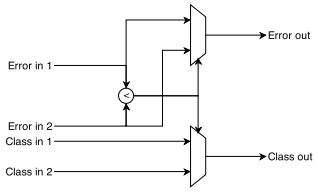


Figure 15: Pairwise Classification Unit

5 Results

The implemented processing pipeline is tested with several images containing frontal view of cars. A test-bench that is created that reads an image and send it to the processing pipeline. The test-bench simulates a video stream, with standard video frame rate that can be adjusted in the test-bench. The pipeline will process the video stream and produce the output image of each processing stage. These images will be written as a bitmap file by the test bench for debugging purposes. The license plate characters are given out at the other end of the pipeline. Xilinx ISim simulator 14.5 was used to perform all types of simulations, both behavioural and timing as well as synthesising the design.

	Execution Time (ms)			Accuracy		
Processing stage	Proposed Sys.	FPGA Sys.[Je13]	DSP Sys. [Je13]	Proposed Sys.	FPGA Sys.[Je13]	DSP Sys. [Je13]
Platform- Resolution	Zynq 7020	Virtex 4 LX 40	ARM-DSP-Soc	640 x 480	640 x 480	1920 x 1080
Pre-processing	6.012	4.7				
License plate detection	0.12	0.11		90%	97%	97%
Character segmentation	0.0203	1.4				
Character recognition	0.0204	0.7		83.3%	97%	97%
Total	6.1727	6.91	71.35			

Table 1: Performance and accuracy results

5.1 Detection, Recognition, and Performance Results

If the FPGA is set to operate at maximum clock frequency to process an image with VGA resolution, license plate detection can be accomplished in less than 10 ms. This should satisfy real-time processing requirement of any ANPR application, especially for the detection of fast moving cars on a highway. The processing pipeline manages to achieve a plate detection rate of 90 % and recognition rate of 83 % as shown in Table 1. The pre-processing stages (morphology operations plus the connected component analysis) consumes 97% of the total time required by the system to process a single frame. The proposed architecture outperform the FPGA architecture presented in [Je13] in terms of processing time per-frame (Table 1). In addition, the architecture proposed in this paper is fully pipelined. This means that we can achieve higher throughput for a video stream. The DSP implementation presented in [Je13] is operating on Full-HD resolution which will require more time to process (71.35ms). However, the proposed architecture can be simply modified for Full-HD resolution, with minimal effect on the performance. Another advantage of the presented work is that it does not employ any off-chip resource which is not the case in [Je13]. The work presented in [Je13] presented a higher accuracy. This is due to the fact that ANN is employed for the classification stage. The simplified K-means algorithm employed in this paper requires larger training database which can be used to increase the accuracy of the system.

Device utilization summary			(xc7z020-1-clg484)	
Number of Slice Registers:	8456	out of	106400	7%
Number of Slice LUTs:	33975	out of	53200	63%
Number used as Memory:	88	out of	17400	0%
Specific Feature Utilization				
Number of Block RAM/FIFO:	127	out of	140	90%
Number of BUFG/BUFGCTRL/BUFHCEs	4	out of	104	3%
Number of DSP48E1s:	3	out of	220	1%

Table 2: Device utilization summary of the processing pipeline.

5.2 Resource Utilization and Implementation Results

A detailed logic utilization is listed in Table 2. The processing pipeline has a maximum operating frequency of 57.823 MHz. Multipliers are only used for translating pixel coordinate memory addresses. Block RAMs are mostly used as row and frame buffers for morphological filtering, license plate and character image segmentation. Figure 16 summarizes the resource allocation to every stage of the processing pipeline. The classifier takes almost half of the on-chip resources due to the large adder trees used to calculate the sum of absolute errors. Other components that mainly use block RAM for their task such as plate detection and character segmentation requires only 2 %. Compared to the work presented in [Je13], 80% of the Virtex-4LX 4M gate FPGA were consumed to build complete ANPR system. The Zynq 7020 chip employed in this paper has smaller size and only 63% of the resources were consumed which allows more customization of the pipeline.

6 Conclusions

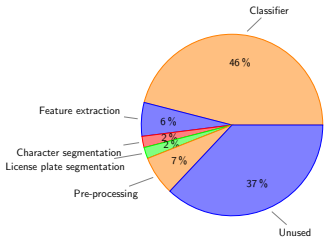


Figure 16: Resource Allocation for Each processing Stage.

In this paper, a processing pipeline for the license plate detection and recognition system has been designed and implemented on an FPGA. It consists of two main parts which are assigned for license plate localization and license plate character recognition respectively. Most of the on-chip resources are allocated to the license plate recognition part to make it capable of processing multiple characters in parallel. The pipeline has been designed to be highly reconfigurable so that it can be ported to another FPGA device that offers more logic resource. According on the test results, the morphological approach is proven to be very effective for the license plate detection task with accuracy up to 90%. The classification with k-means clustering also proves to be reliable with accuracy up to 83%. Parallel execution of the recognition unit reduces the computation time which allows the proposed architecture to process a single frame in approximately 10 ms. The pipelined operation of the system can be used to hide this latency and increase the frame rate. The processing pipeline can be customized to improve the flexibility of the vision system and to support other applications.

References

[ALB07] Arth, Clemens; Limberger, F.; Bischof, H.: Real-Time License Plate Recognition on an Embedded DSP-Platform. In: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on. pp. 1–8, June 2007.

[As13] Ashourian, M.; Daneshmandpoura, N.; Tehrania, O. Sharifi; Moallem, P.: Real Time Implementation of a License Plate Location Recognition System Based on Adaptive Morphology. International Journal of Engineering, 2013.

[Ba11] Bailey, Donald G.: Design for Embedded Image Processing on FPGAs. "John Wiley and Sons (Asia) Pte Ltd", 2011.

[Gr13] Grimm, Michael: Camera-based driver assistance systems. Advanced Optical Technologies, 2013.

[Hs09] Hsieh, Ching-Tang; Chang, Liang-Chun; Hung, Kuo-Ming; Huang., Hsieh-Chang: A real-time mobile vehicle license plate detection and recognition for vehicle monitoring and management. In: Pervasive Computing (JCPC), 2009 Joint Conferences on. pp. 197–202, Dec 2009.

[Iw] Iwanowski, Marcin: Automatic car number plate detection using morphological image processing. PhD thesis, Warsaw University of Technology, Institute of Control and Industrial Electronics EC Joint Research Centre, Institute of Environment and Sustainability.

[JB08] Johnston, Christopher T.; Baily, Donald G.: FPGA implementation of a Single Pass Connected Components Algorithm. In: 4th IEEE International Symposium on Electronic Design, Test and Applications. 2008.

[Je13] Jeffrey, Zoe; Zhai, Xiaojun; Bensaali, Faycal; Sotudeh, Reza; Ariyaeeinia, Aladdin: Automatic Number Plate Recognition System on an ARM-DSP and FPGA Heterogeneous SoC Platforms. In: Poster session presented at Hot Chips: A Symposium on High Performance Chips,HC25 ,Stanford, Palo Alto, United States. 2013.

[Ma07] Martinsky, Ondrej: Algorithmic and Mathematical Principles of Automatic Number Plate Recognition Systems. Master's thesis, Brno University of Technology, 2007.

[MBJ08] Ma, Ni; Bailey, D.G.; Johnston, C.T.: Optimised single pass connected components analysis. In: ICECE Technology, 2008. FPT 2008. International Conference on. pp. 185–192, Dec 2008.

[Od] Odone, Francesca: Experiments on a License Plate Recognition System. PhD thesis, DISI, Università degli Studi di Genova.

[SDR12] Siddharth, Kartar Singh; Dhir, Renu; Rani, Rajneesh: Comparative Recognition of Handwritten Gurmukhi Numerals Using Different Feature Sets and Classifiers. International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012), 2012.

[SGD06] Shapiro, Vladimir; Gluhchev, Georgi; Dimov, Dimo: Towards a Multinational Car License Plate Recognition System. Machine Vision and Applications, 17(3):173–183, 2006.

[TKA07] T. Kanamori, H. Amano, M. Arai; Ajioka., Y.: A High Speed License Plate Recognition System on an FPGA. In: Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on. pp. 554–557, Aug 2007.

[ZBR11] Zhai, X.; Bensaali, F.; Ramalingam, S.: Real-time license plate localisation on FPGA. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on. pp. 14–19, June 2011.

[ZDF10] Zhong, Chongliang; Ding, Yalin; Fu, Jinbao: Handwritten Character Recognition Based on 13-point Feature of Skeleton and Self-Organizing Competition Network. In: Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on. volume 2, pp. 414–417, May 2010.