J Real-Time Image Proc (2018) 15:13–30
https://doi.org/10.1007/s11554-016-0579-x

SPECIAL ISSUE PAPER

Hardware implementation and validation of a traffic road sign detection and identification system

Rihab Hmida[1] · Abdessalem Ben Abdelali[1] · Abdellatif Mtibaa[1]

Received: 9 September 2015 / Accepted: 10 March 2016 / Published online: 26 March 2016

Abstract Reconfigurability and parallel computing capability of field programmable gate array (FPGA) devices are highly exploited in real-time digital image and video processing applications. In this field, real-time traffic road signs detection systems present a huge interest since they help to assist drivers and decrease accidents. In this paper, we propose an FPGA-based hardware implementation of road signs detection and identification system. The proposed system can achieve real-time video constraints while assuring a high-level accuracy in terms of detection rate. The performance of the system in terms of processing latency was evaluated relatively to the reaction distance, the braking distance and the vehicle speed. The evaluation results show that our system can support real-time driving conditions until the speed of 110 km/h. To prove the validity of the proposed implementation, a hardware co-simulation strategy was applied. This is based on the use of Matlab/Xilinx system generator. A comparison of the co-simulation results shows the effectiveness of the developed architecture.

Keywords Road sign detection and identification · FPGA · Real time · Hardware implementation · Xilinx system generator · Hardware co-simulation

✉ Rihab Hmida
hmida.rihab@gmail.com

Abdessalem Ben Abdelali
Abdessalem.BenAbdelali@enim.rnu.tn

Abdellatif Mtibaa
Abdellatif.Mtibaa@enim.rnu.tn

[1] Laboratory of Electronics and Micro-electronics, University of Monastir, 5000 Monastir, Tunisia

1 Introduction

Passenger safety has become the primary interest of automobile factories, since the number of people killed in traffic accidents is tragically increasing. It begins with basic airbags till revolutionary motion sensors, cameras, and various computer-aided driving technologies [1, 2]. Driver assistance systems development is one of the most popular research subjects in autonomous navigation, and a lot of works are focusing on this topic. Most of the current systems embedded on today's vehicles are designed to help the driver to avoid accidents and dangerous situations and to ensure more comfort and energy efficiency. These systems present an electronic driving aid, like the anti-lock breaking system (ABS), the system trajectory control (ESP for 'Electronic Stability Program'), and the automatic parking cameras and mirrors.

A safe navigation requires that the vehicle be able to detect the road edges, obstacles, etc., while respecting the road signs. Sometimes, the drivers being not concentrated do not see the sign in the right time. Therefore, an automatic system for detection and recognition of road signs would be of great interest. This seems an easy task because panels follow a well-defined standard in terms of shapes, colors, sizes and positions in the road. But, the reality is different, due to various constraints related to the state of these signs: they can be partially hidden or not perfectly visible because of weather conditions, such as rain and fog or because of the shadow of trees, buildings, or other objects. All these problems can lead to a false detection of the road signs or confusion with similar objects. In literature, different algorithms have been proposed for road sign detection. In [3–5], color-based method was applied to detect and extract red road signs. Shape-based algorithms were applied in [6–9] using a large set of predefined templates to enforce the robustness of the system. In [10, 11], both image processing and machine learning algorithms were refined to improve the performance of the road sign recognition system.

Most applications of intelligent systems require high-speed operations to support real-time constraints. By performing these systems with only software implementation a satisfactory performances cannot be obtained, especially for complex and multi-techniques-based processing methods. Hardware implementation is necessary to ensure execution time acceleration and satisfy real-time constraints [2].

In this paper, we propose the hardware design of a road sign detection and identification system. The system design is divided into four steps: (1) a pre-processing stage to improve the image quality; (2) detection and extraction of the potential region of interest representing a road sign; (3) shape identification, (4) classification of the detected sign. The proposed method is based on a combination of the commonly adopted approaches in literature, while ensuring a compromise between accuracy and processing time.

The rest of the paper will be divided into five sections. In Sect. 2, some related works are presented. They include processing approaches in addition to implementation methods and tools used to embed the detection system into a hardware device. A discussion about the technical choices (advantages and drawbacks) is also presented in this section: color space choice, filter type choice, adaptive thresholding, and the proposed technique for shape identification. In Sect. 3, the road sign detection and classification algorithm will be presented in details. The algorithm starts with a sequence of some standard image processing. To reduce the light influence, we convert the RGB image to an YCrCb one. A median filter is used to minimize noises and then a segmentation step is performed to facilitate the differentiation between objects and the background. A new approach using some geometric characteristics will be exploited to determine the panel shape among predetermined forms. In Sect. 4, the hardware design phase is presented through the definition of a Platform of Test made within a specific dedicated library. The system is designed and synthesized for the Xilinx Virtex-5 FPGA. The simulation results and discussion of the obtained performances are presented in Sect. 5. We end the paper by a conclusion (Sect. 6).

2 Related works and the proposed system

It is difficult to compare the published works focusing on traffic signs because the majority of the studies consider the complete chain of detection, classification and tracking. Few of them deal only with the detection part, which is in fact the most important one. Even for the detection phase, some articles concentrate on a specific road sign category such as speed limit signs [12–14]. In literature, various techniques of image processing are proposed for all these steps and different techniques are used for the system implementation.

In [2], a hardware implementation was performed on a Xilinx Virtex-4 FPGA family. The described algorithm was adapted for hardware implementation using the Handel-C language. The acquired image is transformed into the HSV space, and then filtered with a median filter. Two neural networks are then used for the sign detection and recognition.

In [5], a parallel implementation was proposed for a complete system for sign recognition with map fusion, including localization and map matching, both on a multicore processor. The open multi-processing (OpenMP) on a graphics processing unit (GPU) was applied using compute unified device architecture (CUDA). Authors showed that the success of localization and map matching can be increased by employing a high number of particles and real-time performance can be achieved only by parallelization.

Other works used also a multicore processor or GPU to implement their systems such as in [15] where an implementation of a traffic sign detection system on a multicore processor is presented. In [12, 16, 17], researchers applied the idea of using a graphics processing unit (GPU) as an embedded co-processor for real-time detection of traffic signs.

In [18], the algorithm was written and validated in software using the "Java" programming language. Authors focused on solving problems of detecting and identifying objects in an image. The proposed system detects traffic signs in the video stream and tries to recognize them using a knowledge base. Detection is done in the HSB color space using fuzzy color segmentation. Afterward, detected segments are analyzed by a fuzzy rule-based system.

In [19], authors present the implementation of an embedded automotive system that permits to detect and recognize traffic signs within a video stream. Xilinx Embedded Development Kit (EDK) was used to enable the quick creation of an on-chip embedded processor (MicroBlaze) and user specific peripherals on a field programmable gate array (FPGA).

Many works such as those presented in [19, 20] concentrate on the methods more than on the hardware implementation. In fact, the transition to a hardware implementation is time-consuming and requires wide knowledge in electronics. In addition, at each simple modification in the model structure, the designer must go through all hardware implementation and simulation steps, which leads to a high cost and affects the time-to-market delays [21].

Several experiments of existent image processing techniques were realized to obtain an efficient road sign detection and identification system model. In Table 1, we summarize the main treatments and characteristics of some road sign detection methods existing in literature and the selected treatments used in our algorithm.

Two main aspects may be discussed, because they highly influence the detection rate: the choice of the color space and the method of shape detection and identification. For the color space choice, the input image has to be converted to an adequate color space, which permits to reduce the lighting effect. The HSV color space was used in many works such as those presented in [2, 10, 22, 23]. In [24], a color spaces comparison showed that HSV presents the best performance in automatic color-based segmentation/detection of road signs. However, transforming the RGB color space to HSV requires large computational time and it is not easy to define the threshold values for the H and S components, to extract red colored objects [2]. In [2], it was also shown that the detection rate of well-lit panels is about 90 % while it is about only 72 % for poorly lit and dark ones because of the non-adequacy of the threshold values. On another hand, the YCrCb color space has been the most widely used one [25–31] and it presents good results regardless the effects of lighting conditions.

Several studies were focused on shape-based detection algorithms such as "Hough transform" which is firstly known to detect circular forms and then it was extended to find most curves in an image to recognize regular features. "Hough transform" was exploited in [8, 32, 33] and appreciated for its good detection results. Unlikely, it was not as much valued by Barnes and Zelinky [32] since they show that this method is fast enough to work in real time only with circular signs. Also, in [34] authors show clearly that this technique necessitates a big computation time, since it contains nonlinear operations which are time-consuming. According to their results, "Hough transform" spend over than 15 s which present nearly 70 % of the detection time, and thus it is not adequate for real-time applications.

For the adopted sign detection technique, we tried to integrate, as much as possible, the key approaches applied in literature, while ensuring a compromise between accuracy and processing time. In fact, our main objective is to propose an accurate and real-time system. The following points summarize the main involved techniques and advantages of the proposed system:

- Lighting conditions consideration: the YCbCr color space will be used and depending on the lighting conditions (day/night/rainy or wet), an adaptive threshold will be defined to generate the binary image without loose on the road sign information.
- Use of a region of search (ROS): making panel detection in only a region of interest permits a significant reduction on the number of processed pixels. This can speed up the filtering, segmentation and shape identification processes.
- Shape identification before the classification step: a shape identification step will be added to make faster the panel classification. Indeed, the possible road sign will be correlated only with signs of the same shape. As previously discussed, various techniques of shape identification exist in literature. However, the main drawback of these ones is their complexity and cost either in terms of execution time or hardware resources. In this work, we propose a new algorithm that exploits linear operations and offers a good accuracy.
- Pipelined implementation: a parallel implementation of the system will be proposed.

Table 1 Taxonomy of road sign detection algorithm

	[2] (2013)	[22] (2013)	[5] (2012)	[28] (2008)	[48] (2009)	[3] (previous work)	Proposed algorithm
Color space	HSV	HSV	x	x	YCrCb	YCrCb	YCrCb
Segmentation	✔	✔	✔	✔	✔	Single threshold segmentation	Adaptive segmentation
Edge detector	x	Canny	x	Gradient	x	x	x
Filtering	Median filter (9 × 9)	Dilatation erosion	x	Median filter	Dilatation erosion	Dilatation erosion	Median filter (3 × 3)
ROS extraction	✔	x	x	x	x	x	✔
Shape identification	✔	✔	x		x	x	✔
ROI extraction	✔	x	✔	x	✔	✔	✔
Lighting conditions	✔	x	x	x	x	x	✔
Software programming	✔	x	✔	✔	✔	✔	✔
Hardware implementation	✔	✔	x	✔	x	x	✔

- Use of XSG tool: the use of the XSG tool has a big benefit in terms of conception time, since the same design will be used firstly for the software validation and then for the hardware system generation.

3 Description of the road sign detection and shape identification algorithm

In this section, we give a detailed presentation of the proposed technique for road sign detection, shape identification and classification.

3.1 Flowchart of the proposed system

The flowchart of the proposed system is illustrated in Fig. 1. This latter mainly expresses the processing mode of the proposed system regarding image acquisition and blocks synchronization. The flowchart input is an image sequence. A new image is sent when a road sign is not detected, its shape is not identified or it cannot be classified.

The first processing block (sign detection) includes morphological operations applied only on the ROS and a first test (possible sign?) to extract the selected ROS to be treated. This test indicates the presence or the absence of an object that could be a road sign. The second block (sign identification) includes the ROI extraction and the shape identification steps. When the shape is identified, a road sign recognition block will be used to classify the panel.

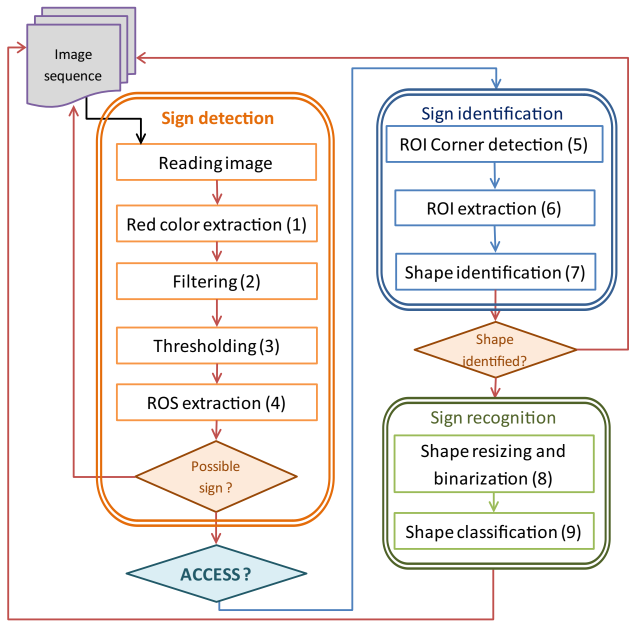


Fig. 1 Flowchart of the proposed road sign detection, shape identification and recognition

To reduce the execution time, we take profit from the independency of the detection phase and the identification and classification one to execute them in a parallel way. Indeed, the execution time of the detection and classification phases could be overlapped. When an image is processed by the sign identification and recognition blocks, the detection block treats the next one. To avoid overlapping data of both of current images, a second test (ACCESS) is added to command the activation of the sign identification block.

3.2 Color space conversion (step 1)

The input “RGB” image is converted to the “YCrCb” color space in which the luminance component is separated from the color components and thus the influence of luminance can be removed during the image processing. Y represents the luminance information and Cr and Cb are the color difference signals that represent the chrominance information. The conversion formula is expressed by Eq. (1) [3].

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & 0.081 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \quad (1)$$

3.3 Filtering (step 2)

Acquired images usually contain noise due to light, shadow, or dust. Filtering is incorporated to improve the visual quality of the captured image by reducing the intensity variations in the image while respecting the integrity of the scene [21]. The median filter, used in this work, is a non-linear filter particularly effective against noise in grayscale images. It is highly exploited in many works such as [2, 5, 35, 36]. The effect of the filter could be clearly seen after the binarization of the image (Fig. 2).

3.4 Thresholding (step 3)

There are two major objectives in segmenting an image. The first one is to highlight information that is relevant while removing irrelevant information based on prior knowledge of the application. The second objective is to reduce the amount of data to be stored and processed, and hence reduce computational cost [37]. Figure 2 shows the application of the segmentation step on an example of real road scenes and the effect of the median filter.

The threshold value determination is the most important issue [2, 38]. This value depends on the level of brightness

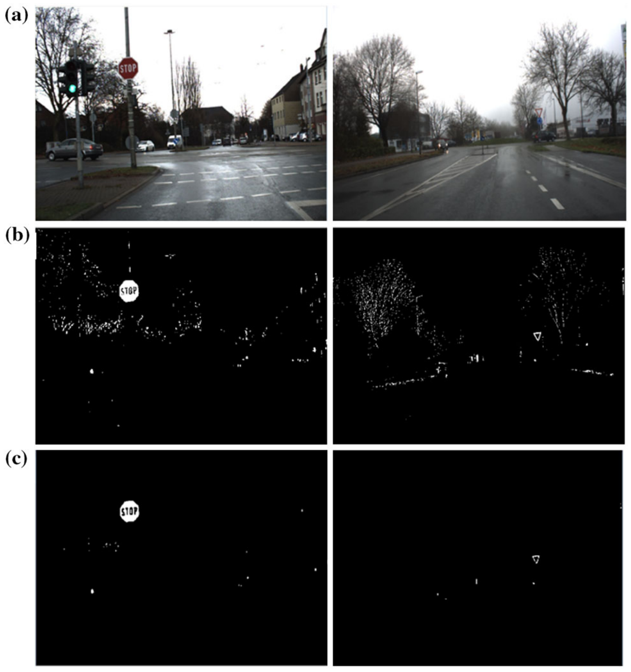


Fig. 2 Segmentation applied on RGB road sign images: **a** RGB road images, **b** binary images without filtering, **c** binary images with filtering

of the picture, i.e. on weather conditions during images capturing. A non-adequate threshold could generate a binary image containing; either the panel and other small objects around it or a partially hidden panel. This could, respectively, lead to false detection and false shape identification. To overcome this problem, we exploit an approach, which consists in detecting if the image was acquired on day, rain/fog, evening or night and we try to define threshold values for each of these image capturing conditions.

The threshold values were obtained based on a statistical study performed using a set of images for the different weather conditions. The margin of the Cr component was determined using the Cr histograms for each case of the considered weather conditions (Fig. 3). Also the brightness degree was evaluated for the same conditions. Four cases are summarized in Table 2.

Based on the brightness and the Cr margins, threshold values were deduced. For a high brightness, a high threshold value should be used to isolate the sign with minimum noise. For lower brightness, a lower threshold value should be used to grantee minimal loss on useful information (sign). Several experiments were re-done to ensure the efficiency of the made choice.

Fig. 3 Example of experimental tests to determine margins of Cr component in day and night

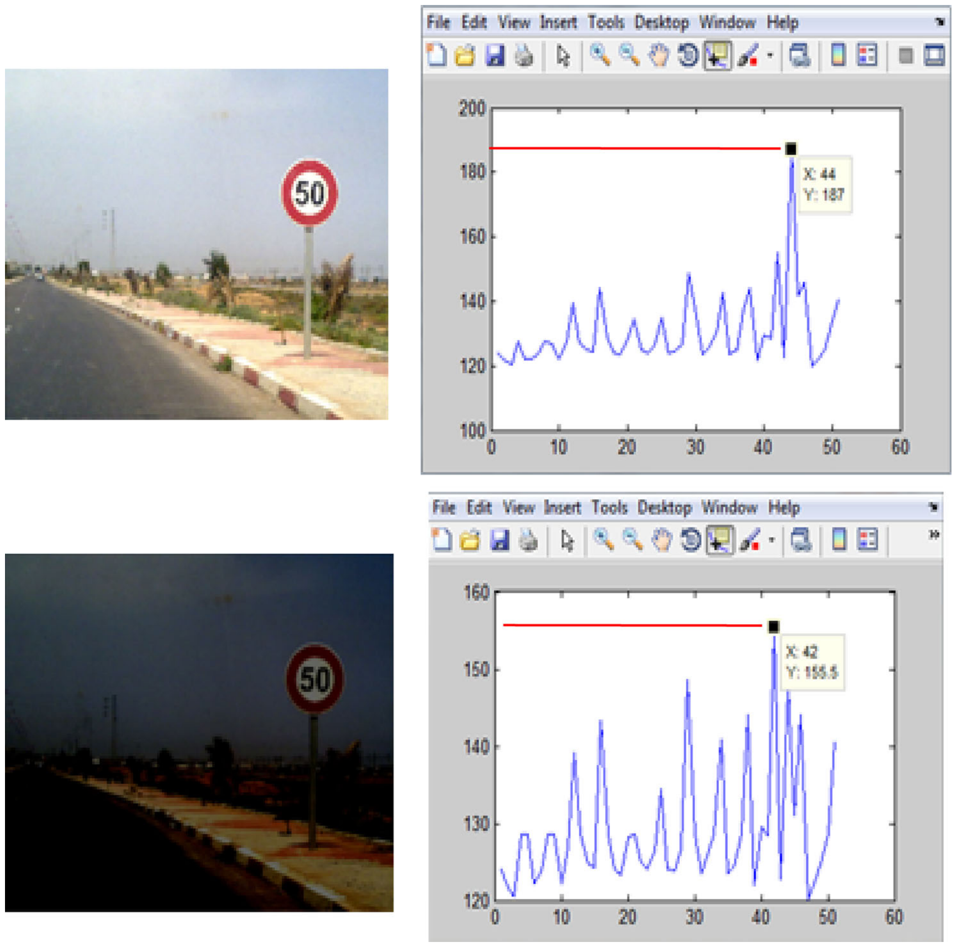


Table 2 Margin of different Cr components and brightness

	Cr_min	Cr_max	White_min	White_max
Day	140	195	500	765
Rain/wet	120	185	500	722
Evening	135	170	50	415
Night	120	160	0	300

3.5 ROS extraction and possible sign test (step 4)

The acquired images, extracted from the video stream, have a size of 640 × 480 pixels. To reduce the number of pixels to be covered, we define two regions of search which could mostly contain a road sign as shown in Fig. 4. The dimension of one region is 210 × 320 which leads to a data reduction of about 33 % in the preprocessing step as only the image portion containing the ROS is used and 78 % after the ROS extraction. In fact, only 22 % of pixels will be covered to find the road sign.

When reading the binary image pixels belonging to the ROS, accumulation of white pixels is checked to decide if there is a possible road sign in the ROS or not. This permits selecting and extracting the valid ROS between the two defined ones in the image (Fig. 4).

3.6 Panel detection and extraction (steps 5–6)

The next step is to localize, in the binary image ROS, an object which may be a road sign and to pick it out. The purpose here is to determine the coordinates of the four corners of the object ($X_{\min}$, $Y_{\min}$, $X_{\max}$, and $Y_{\max}$) to locate the object in the image as shown in Fig. 5.

Cartesian coordinates of each pixel belonging to the object will be then calculated using the following relation:

$$Y = \text{position}/210 \text{ and } X = \text{position} - (Y \times 210) \quad (2)$$

where "position" represents the pixel address.

A comparative step is then performed to extract the lowest values of X and Y ($X_{\min}$ and $Y_{\min}$) and the highest ones ($X_{\max}$ and $Y_{\max}$). Pixels, which belong to the area of interest, verifying the relation (3) are extracted and then saved in a separate memory buffer.

$$X_{\min} < X < X_{\max} \text{ and } Y_{\min} < Y < Y_{\max} \quad (3)$$

3.7 Shape identification (step 7)

For the shape identification, an approach based on corners detection is applied to identify the considered shapes relative to road signs. A search of a set of 4 × 4 matrix masks is performed in the region of search. According to the kind of mask found, we could conclude on the form of the road sign. In Fig. 6, we give the example of the considered panel shapes and in Fig. 7, we present the defined masks for these types of shapes: (a) a triangle up, (b) a triangle down, (c) a circle and (d) an octagon.

This step allows us to reduce the processing time in the recognition phase. In fact, we have just to use the signs which have the same shape for identification. Since inclination of the road sign panels is about ±17° relatively to the vertical line, we have also to define masks for rotated road signs as we do with the straight ones. Adding new masks for rotated panels will not influence on the runtime since that they will be applied simultaneously.

3.8 Shape classification (steps 8–9)

The shape classification involves three steps: binarization, resizing and recognition. To conserve the panel content, we proceed by a binarization of the green color component of the ROI as shown in Fig. 8.

The size of the ROI depends on the distance from which the image is captured. For a distance greater than 100 m, the ROI is not considered as a possible road sign since it is smaller than a possible sign whose minimum size is 15 × 15 pixels.

An algorithm of template matching is applied for classification. As a shape identification phase is used before this step, the classification is made easier and good matching results were obtained. Template matching consists in comparing a given templates stored in a database and the detected panel to find which is the most similar to it.

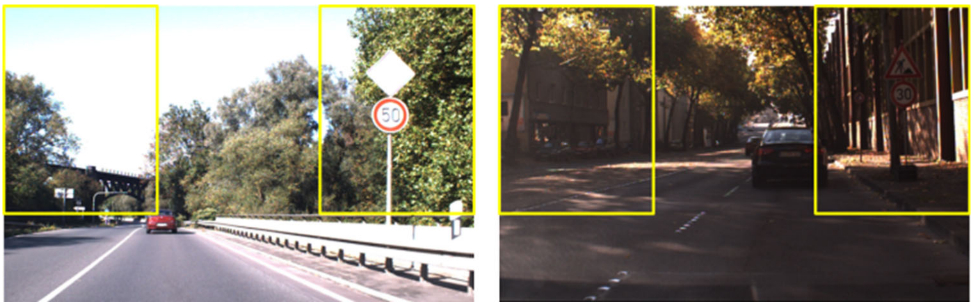


Fig. 4 Defined ROS in the original image

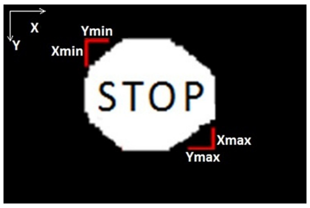


Fig. 5 Concept of panel localization

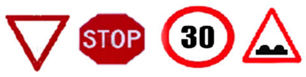


Fig. 6 Considered shapes

3.9 Complete workflow of the proposed algorithm

In Fig. 9, we present the workflow of the proposed method showing all the steps together with images. It gives the processing chain including the following steps: image filtering, segmentation, ROS extraction, ROI binarization, shape identification and road sign classification.

4 Hardware design of the proposed system

Classic FPGA implementing methodology consists of two main steps. In the first step, the algorithm is modeled and simulated using software tools such as Matlab/Simulink. The second step is dedicated to the hardware architecture design and the HDL description which is performed manually. This procedure is time-consuming, especially for those designers that are not familiar with the HDL hand coding process [39]. In this paper, we use a methodology based on the Xilinx System Generator (XSG), an integrator design environment (IDE) for FPGAs, which takes the abstraction level one step higher. XSG will be used for the automatic hardware system generation and also for software validation throughout the hardware co-simulation technique.

4.1 Xilinx system generator (XSG)

XSG provides the Simulink environment with a list of specific Xilinx building blocks which can be used to create designs optimized for Xilinx FPGA's with no knowledge of any HDL coding or FPGA's architecture [40]. The designer does not use blocks from the standard Simulink library but from a supplement library called "Xilinx Blockset". The blocks in XSG operate with Boolean values or fixed-point values, which represent the adequate formats for a hardware implementation. In contrast, Simulink works with double-precision floating point numbers [41].

The XSG environment allows direct mapping into hardware-description language (HDL), which eliminates the error-prone process of manually converting software language into HDL [42]. It provides a fast resource estimation system in order to take full advantage of the FPGA resources. All of the downstream FPGA implementation steps are automatically performed to generate an FPGA programming file. XSG has an integrated design flow, to move directly to the configuration bit file (*.bit) necessary for programming the FPGA [43]. Figure 10 presents the design flow of the XSG development tool.

XSG permits also hardware co-simulation [1], a simulation through hardware in the loop co-simulation, which gives many orders of simulation performance increase [44, 45]. It allows software and hardware simulation simultaneously with the same design formed by XSG blocks connected in cascade. It can perform one of the Simulink model subsystems on an FPGA board at a much more important sampling rate than the rest of the model [31].

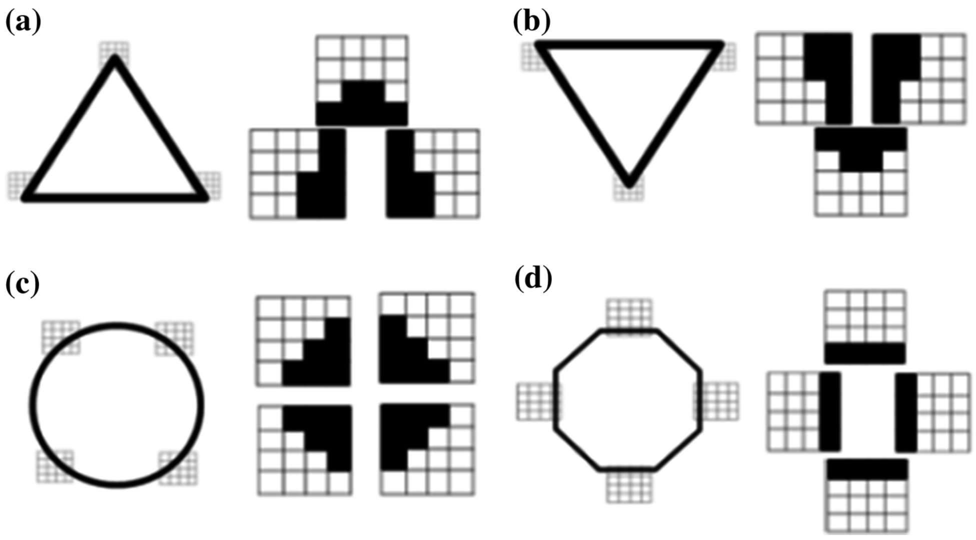


Fig. 7 Shape masks corresponding to the considered panels

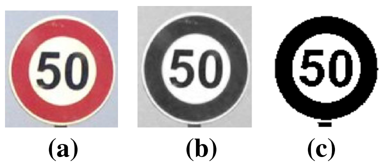


Fig. 8 Binarization principle: **a** RGB ROI, **b** green component, **c** binarised ROI

The hardware co-simulation technique permits to compare software and hardware results and to evaluate the degree of accuracy of the proposed implementation.

4.2 Xilinx proposed XSG design

In the defined platform of test, the acquisition and display of the input and output images is made in software. The fundamental scalar signal type in Simulink is double-precision floating point number and only the processing block is being made based on system generator blocks, which operate on Boolean and fixed-point values. Therefore, a need of an adaptation and an interfacing between both blocks is necessary. Fortunately, XSG offers a simple interfacing using the predefined "Gateway-In" and

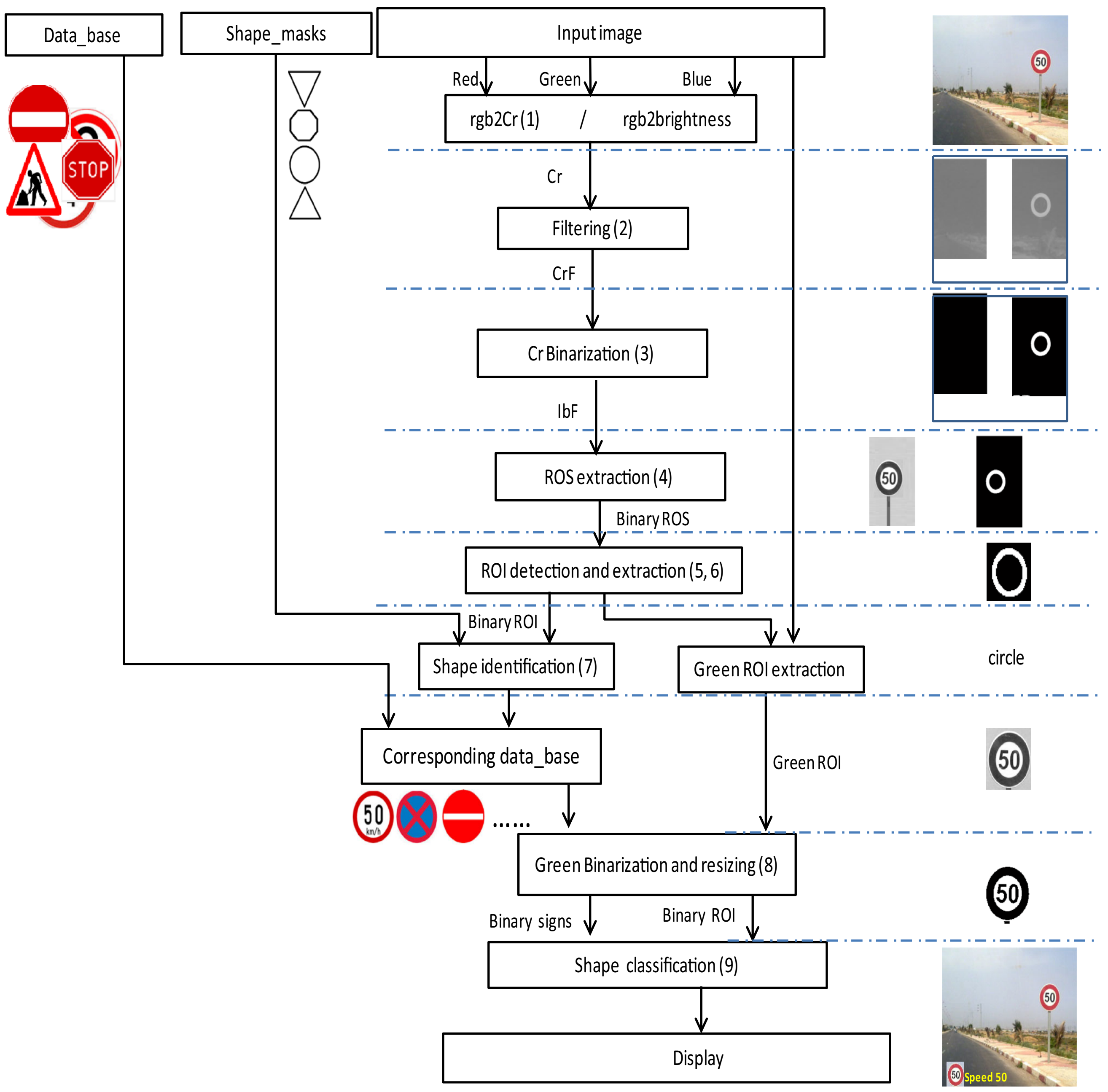


Fig. 9 Complete workflow of the proposed algorithm

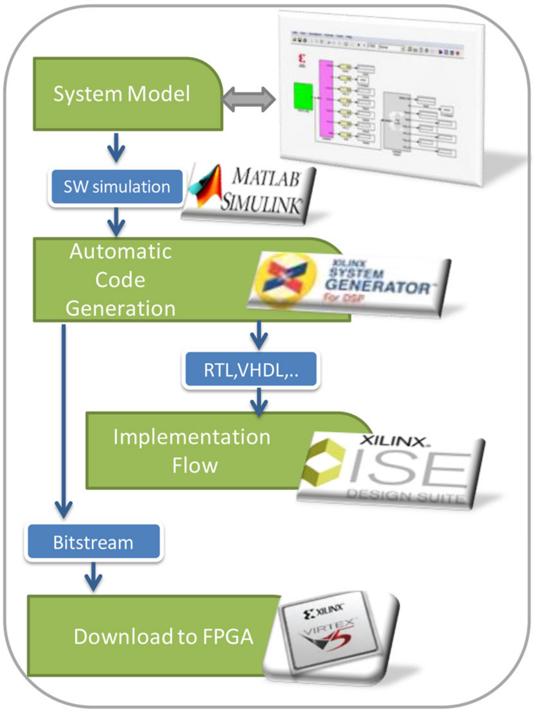


Fig. 10 Design flow with Xilinx system generator

"Gateway-Out" blocks provided by the Xilinx Blockset Library.

Figure 11 represents the global design with the different traffic sign detection system parts: image pre-processing and ROS extraction (part I), ROI extraction and shape identification (part II), and sign recognition (part III).

4.2.1 Pre-processing and ROS extraction blocks (part I)

The detailed architecture of the pre-processing part is shown in Fig. 12. It is constituted of 3 blocks. The first one is the color conversion block (Fig. 12a). It takes as inputs the values of the red, green and blue components of each pixel sent throughout the Gateway-In blocks.

The second block is the median filter, which consists in replacing the central pixel of each 3×3 square matrix of neighborhoods by the median value. We use a "3-Line Buffer" blockset provided by XSG library to extract the neighborhoods matrix. The extracted 3×3 matrix elements values are sorted using multiple comparison blocks, which are indicted as "filtre 22" in Fig. 12b. Each one permits to find out the minimum and the maximum value of two different inputs. Consecutive comparisons of the 3×3 matrix elements are made using the comparison blocks until having the median value.

The median filter is produced by sliding a 3×3 window over the input image and each time replacing the central pixel by the median value. The block indicated by a green box is called "structuring block". It permits to arrange data (pixels) in 3×3 matrix and to scan the image. The input data are introduced in serial format, and a new 3×3 pixels array is obtained in each clock cycle. Three rows are cached using shift registers and FIFO buffers. The size of each FIFO memory is:

$$\text{FIFO\_size} = \text{ROSwidth} - 2 \qquad (4)$$

where "ROSwidth" represents the ROS width.

Six of The 3×3 pixels of the current window are placed in registers and the 3 remaining pixels are directly taken from outputs of the three FIFOS. In the beginning, a number of clock cycles "NC" is necessary to obtain the first valid window (corresponding to the first window position). It is given by:

$$\text{NC} = (3 \times \text{width}) \qquad (5)$$

where "width" represents the image width.

NC corresponds to the necessary number of clock cycles to fill out all the FIFO memories and registers. After that, at every rising edge of clock, we obtain the next window.

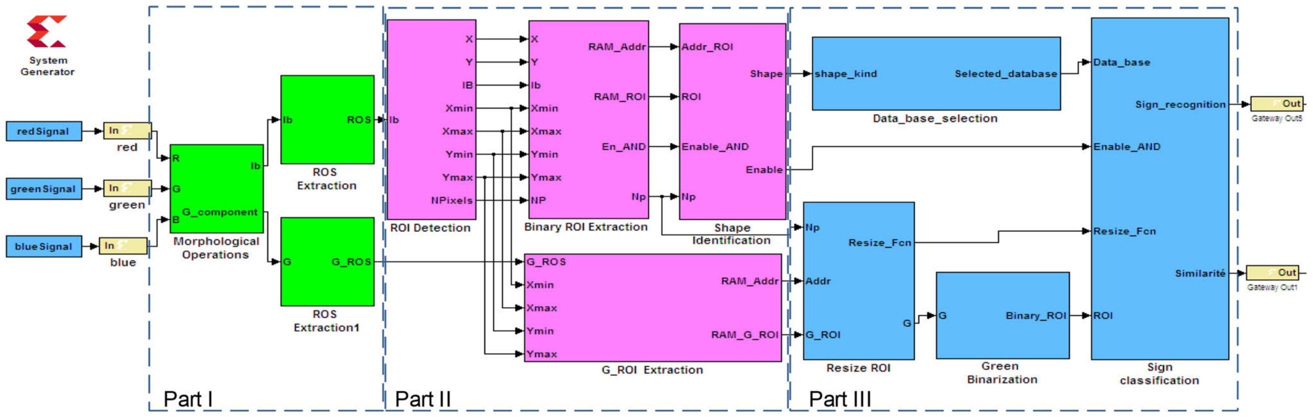


Fig. 11 XSG blocks for traffic sign detection

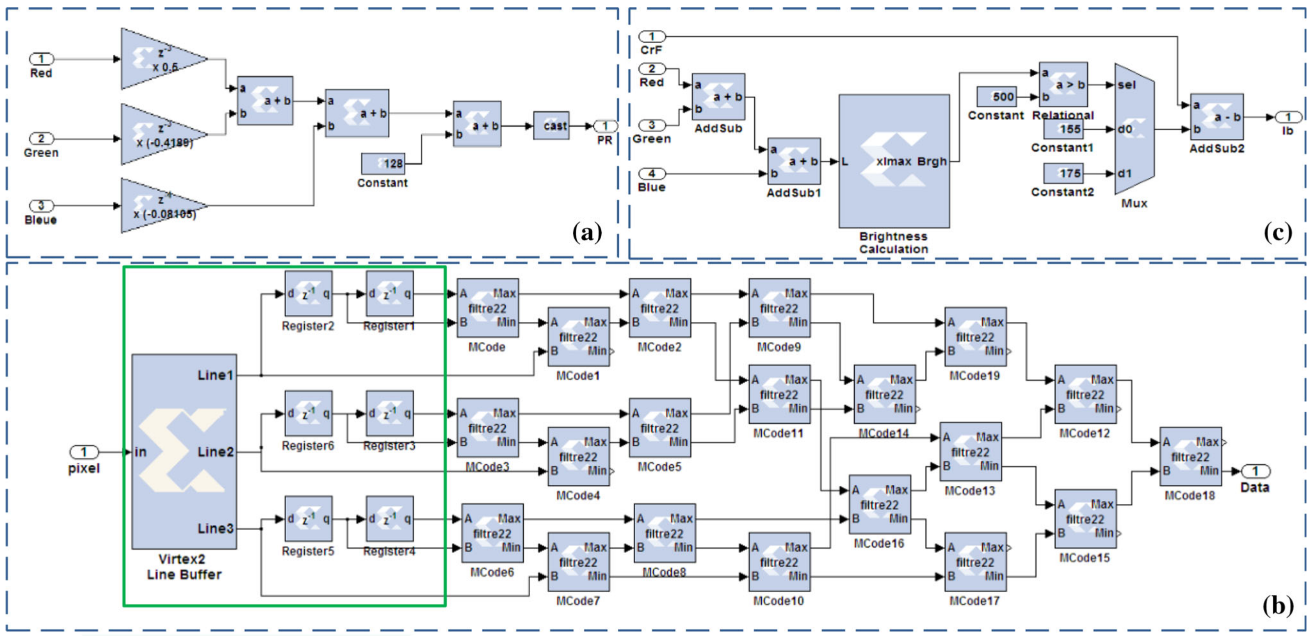


Fig. 12 Detailed architecture of the pre-processing part: **a** color conversion, **b** median filter, **c** adaptive segmentation

During the period of time corresponding to the NC number of cycles, the thresholding block (Fig. 12c) calculates the average value of the brightness using the input RGB pixels of the three first image lines. The obtained value will be used to select the threshold to be used in the binarization phase. Depending on the brightness value (> or <500), a logic signal is generated to be used as the select input of a MUX. The output of this MUX is one of the predefined threshold values (155 or 175), which will be used for comparison with the filtered image pixels.

After the image filtering and segmentation, we begin searching on each ROS (210 $\times$ 320) for an accumulation of white pixels surrounded with black ones that refer to the background. A road sign shape of 90 $\times$ 90 pixels is selected for an image captured 50 m away from the sign. The number of white pixels corresponding to a possible panel belongs to a margin which is set beforehand. An accumulator blockset is used to count the number of white neighborhood pixels to decide which ROS will be considered. If the accumulator output value exceeds the preset minimum value, we assume that the covered ROS could contain a possible sign. In this case, the sign detection and extraction steps will be launched.

The architecture of this block is represented in Fig. 13. It is mainly based on an accumulation block which counts the number of white pixels in a 5 $\times$ 5 neighborhood window. The obtained values for ROS1 (left side ROS) and ROS2 (right side ROS) are used to decide on which ROS to be extracted (the one containing a possible road sign).

4.2.2 *ROI extraction and shape identification architecture (part II)*

In Fig. 14, we present the hardware architecture of the ROI detection and extraction step, which contains five sub-blocks. The functioning of the ROI detection (block 1 until 3) and the ROI extraction (block 4 and 5) architectures is as follows. Block 1 is used to count the input ROS pixels. The output of this block ("Npixels") is used for two purposes:

- It will be introduced in block 2, which permits to calculate the coordinates (x and y) of the current pixel by dividing "Npixels" by the ROS width. These coordinates are used in block 3 to determine the four extreme corners ($X_{\min}$, $Y_{\min}$, $X_{\max}$ and $Y_{\max}$) that highlight the estimated road sign.
- It will also be used to activate block 4. This latter permits to calculate the (x, y) coordinates, which will be used to reread the ROS pixels for the ROI extraction. The Npixels value is compared to the number of pixels of the ROS ("NP-ROS"), and if it reaches the NP-ROS value then the (x, y) calculation block (block 4) of the ROI extraction module will be activated.

Block 3 is dedicated to determine the $X_{\max}$, $X_{\min}$, $Y_{\max}$, and $Y_{\min}$ of the ROI. When the binary input pixel "Ib" is equal to 1, the previous values of $X_{\max}$, $X_{\min}$, $Y_{\max}$, and $Y_{\min}$ are compared to the current x, y inputs to be updated. When "Ib" is equal to 0 these values are maintained. In this case, $X_{\max}$ and $Y_{\max}$ are compared to a very small value and $X_{\min}$ and $Y_{\min}$ are compared to a high value. This is done

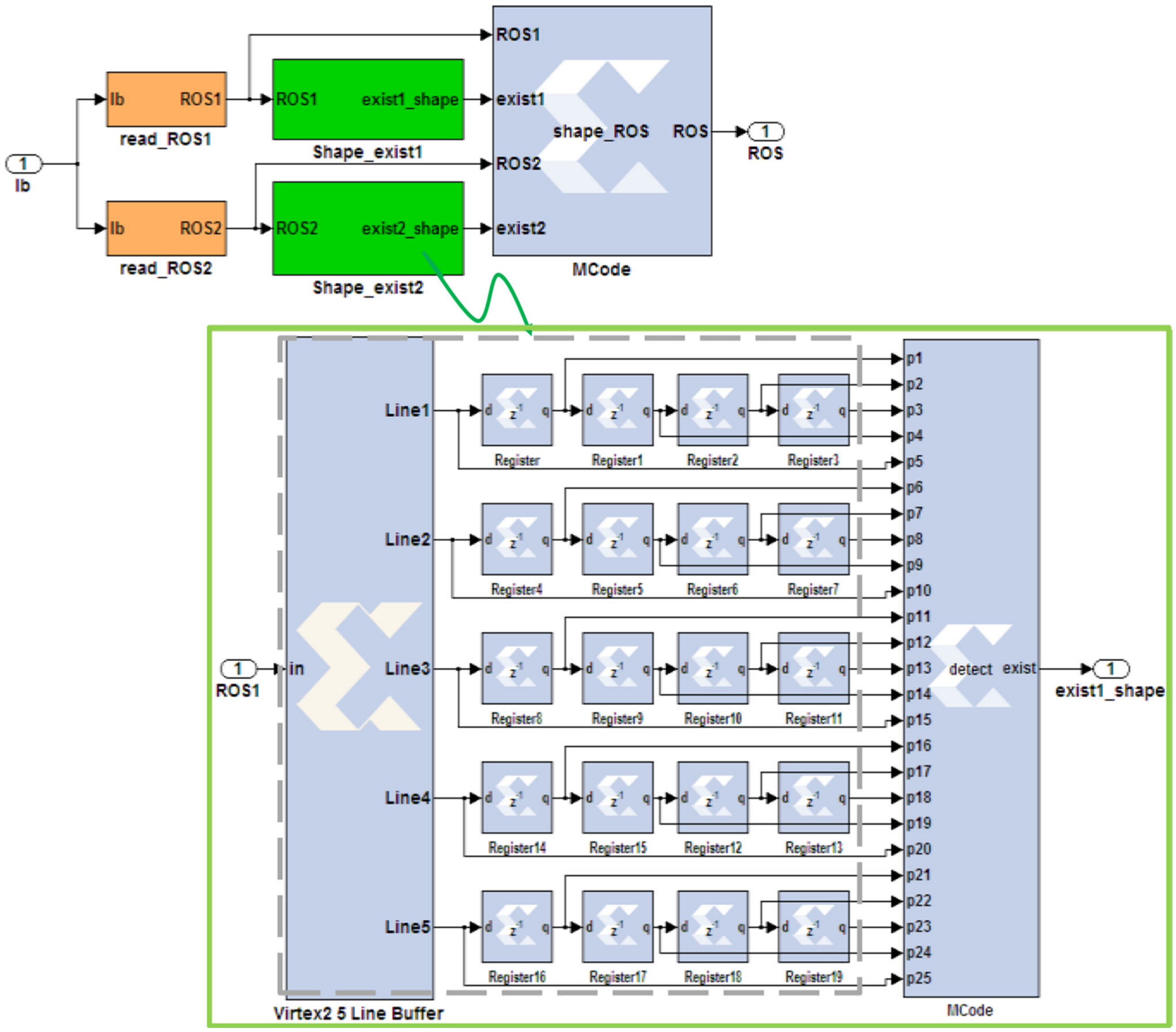


Fig. 13 Hardware architecture of ROS selection block

throughout multiplexers whose select inputs are related to the binary pixel input, their D0 inputs are related to the *x* or *y* coordinates and their D1 inputs are related to the previously mentioned small or high values.

Block 5 permits to extract the ROI from the binary ROS and to save it into a memory block. The *x*, *y* coordinates of the current pixel (generated by block 4) are each time compared to the $X_{\max}$, $X_{\min}$, $Y_{\max}$ and $Y_{\min}$ values to verify if the pixel belongs to the ROI. If it is the case, the enable input of the memory block and the address counter one are activated. An identical module to block 5 is used to extract the green color component ROI using the same $X_{\max}$, $X_{\min}$, $Y_{\max}$ and $Y_{\min}$. The green color component ROI is sent to the recognition block.

The architecture of the shape identification block is represented in Fig. 15. Four correlation blocks, relative to the 4 defined shapes masks (circle, triangle up, triangle down, and octagon) are applied in a parallel way. The final output of the identification block is the kind of the shape (0, 1, 2 or 3).

As illustrated in Fig. 7, each mask contains three or four 4×4 matrix of binary pixels. The circular mask, for example, is composed of four matrixes. The correlation block relative to this example is illustrated in Fig. 14. A 4×4 matrix is extracted on each clock cycle from the ROI and compared to all matrixes of the considered mask. For each mask matrix, we have the same architecture as detailed in Block (2) of Fig. 15. If the comparison result

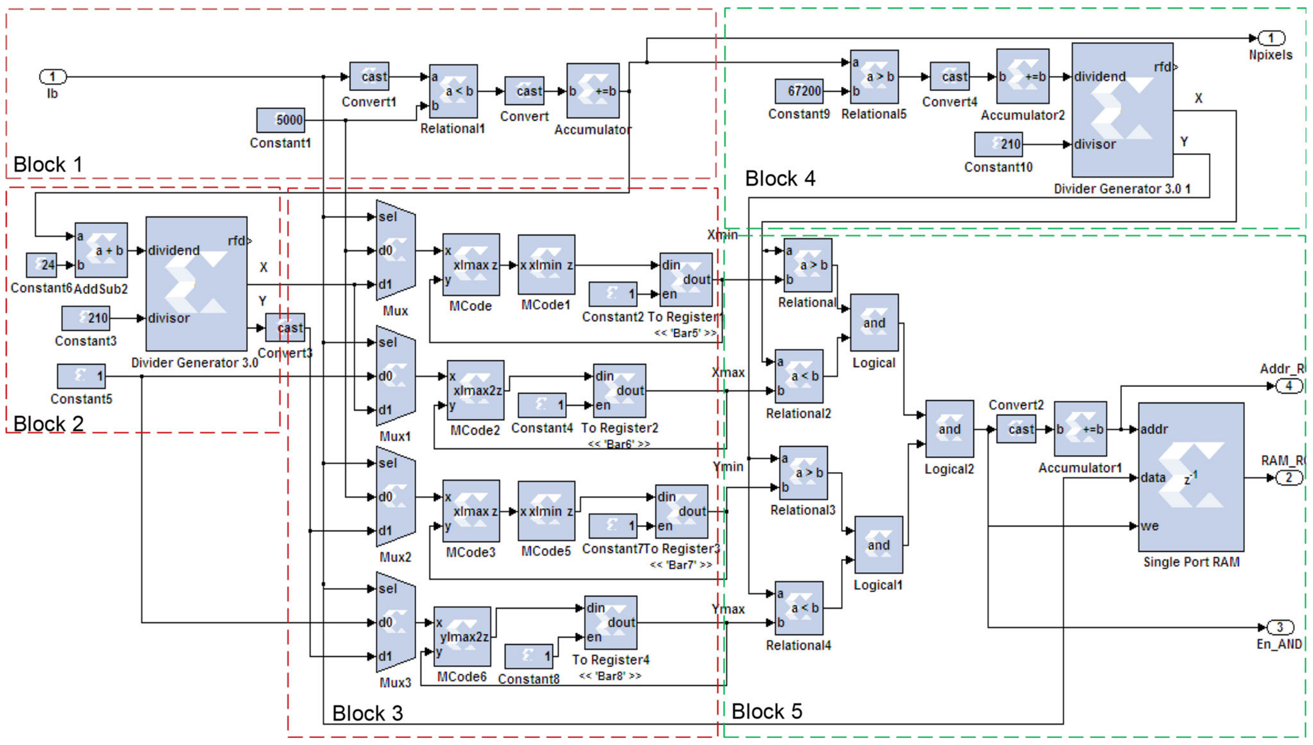


Fig. 14 Hardware architecture of ROI detection and extraction (part II)

with a given matrix of the circular mask is true, a logic signal (“circular_exist”) takes the “1” value. A sign shape is considered as circular if all the four masks are found in the ROI. The correlation block whose output is equal to 4 will define the shape kind.

4.2.3 *Shape recognition (part III)*

Figure 16 presents the hardware architecture of the recognition block. The output signal of the shape identification block (“shape-kind”) is connected to the “DB-selection” input, which is used to enable the corresponding database memory. This signal is also used as the select input of the database selection MUX whose output will be sent to the correlation block.

When the recognition block is enabled, a simultaneous reading of the ROI pixels and those of the database images will start. The outputs of these memories will be, each time, used by the correlation block to calculate the similarity between the ROI and the current image of the database. This is repeated *N* times, where *N* is the number of images in the selected database. During each correlation, a first accumulator is used to count the number of identical pixels and a second one is used to determine the number of the most similar sign. A comparison is, each time, made between the current image similarity rate and the previously saved one to conserve the number of the dataset road sign presenting the maximum similarity.

The counter used to address the ROI memory is reinitialized when the number of pixels reaches 15 × 15 (the database image and ROI size) and a reread process is launched. In this case, the first accumulator is reinitialized, the second one is updated, and a new correlation between the ROI and the next dataset image will start. The outputs of the recognition block are the dataset road sign image number and the similarity rate.

5 Implementation results and performance evaluation

In this paragraph, we begin by presenting the hardware implementation results of the developed system. Some examples of co-simulation results of the generated hardware block will be given. The efficiency of the proposed system is then discussed according to the detection rate and performance of the system implementation. A comparison with some existing works will be described.

5.1 Co-simulation results

Once the functioning of the entire system is approved by software simulation, its implementation on a Xilinx platform can be made. The configuration file is obtained automatically by following the necessary steps to convert the design into an FPGA synthesizable module.

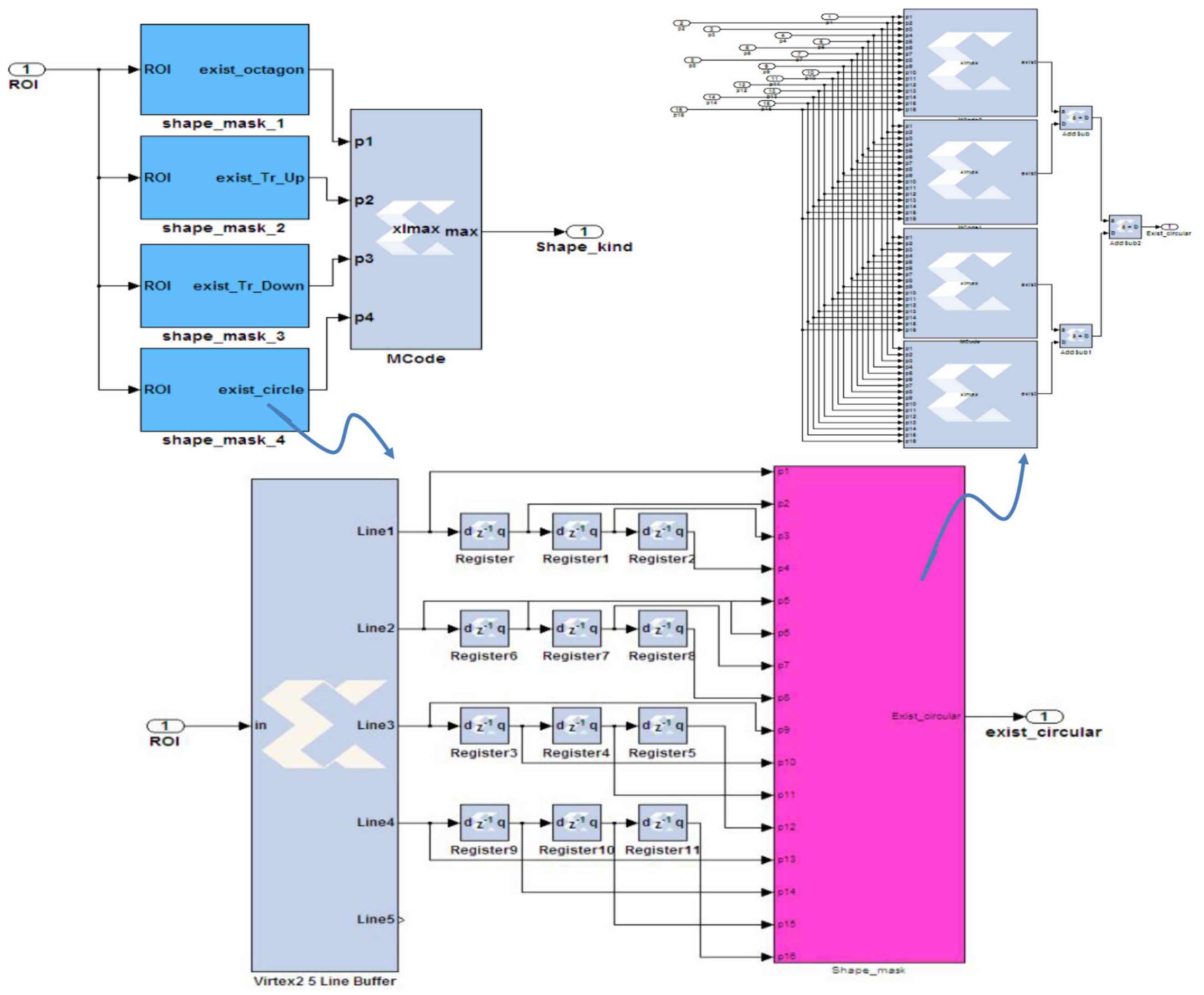


Fig. 15 Hardware architecture of the shape identification process (part II)

In this study, for experimental analysis, we considered a Tunisian road sign database and public European ones acquired in France [46] and Germany [47]. The Dataset also includes some recorded sequences, and a set of sign-free images, which can be used as negative training images.

In Fig. 17, we present some result examples which demonstrate good panel corners detection and a successful extraction of the area of interest. Sometimes the panel shape cannot be identified by the system due to various factors such as the incorrect orientation of the panel and its degradation. A road sign could be also unidentified by the detection system if it figures in a very little size in the acquired image. This problem could be dealt in post-acquiring image, since more the vehicle is closer to the road sign, bigger is the size of this panel in the image.

The hardware simulation of the obtained road sign detection hardware design is done throughout the JTAG interface and using a "System Generator" block. Settings of this block are defined to select the target device and a bitstream file is generated and loaded on the FPGA. In addition to the hardware block containing the bitstream file (Fig. 18), a VHDL or Verilog code is automatically generated. The target device selected for this work is the Virtex-5 FPGA of the ML507 platform.

5.2 Performances comparison and discussion

The proposed method of the traffic sign detection was applied on several images in various lighting conditions to evaluate the system's robustness. We divide the data base into three sets: circled shape-form road signs (set-1), octagon

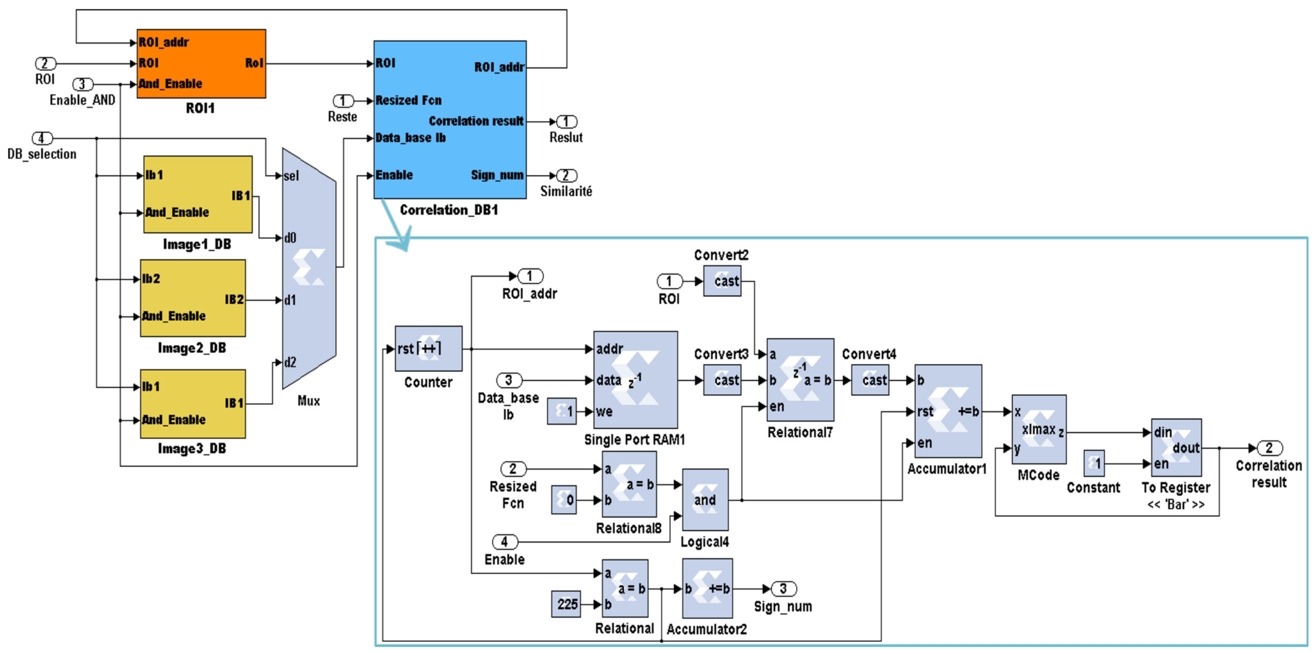


Fig. 16 Hardware architecture of the matching process (part III)

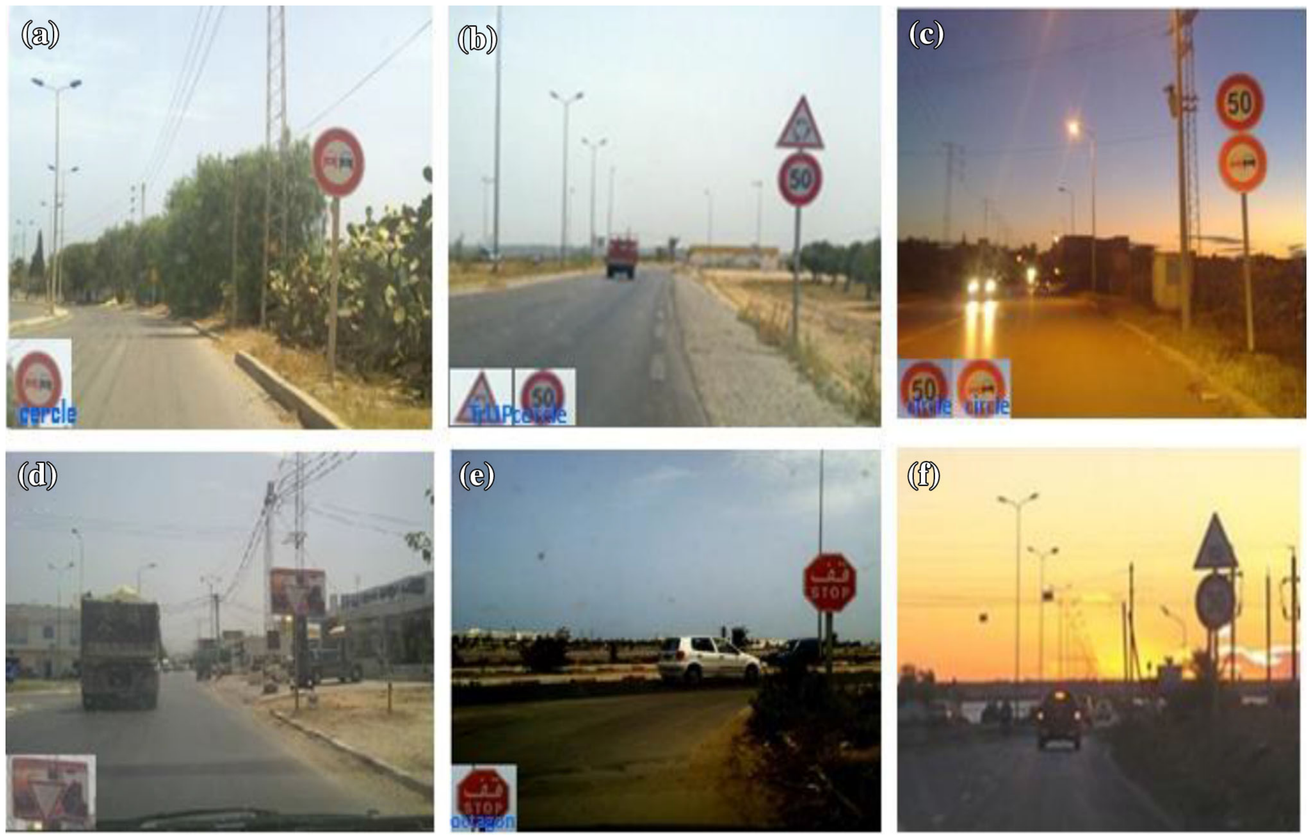


Fig. 17 Some experimental results of road sign scenes captured in different conditions. **a** Successful detection of a single road sign, **b** successful detection of double road signs, **c** successful detection of double road signs in night with well illuminated road, **d** successful detection of the ROI but the red background disturbs the identification of the shape, **e** successful detection and identification of the shape at sunrise, **f** fail detection of road signs in night due to very poor illuminated road

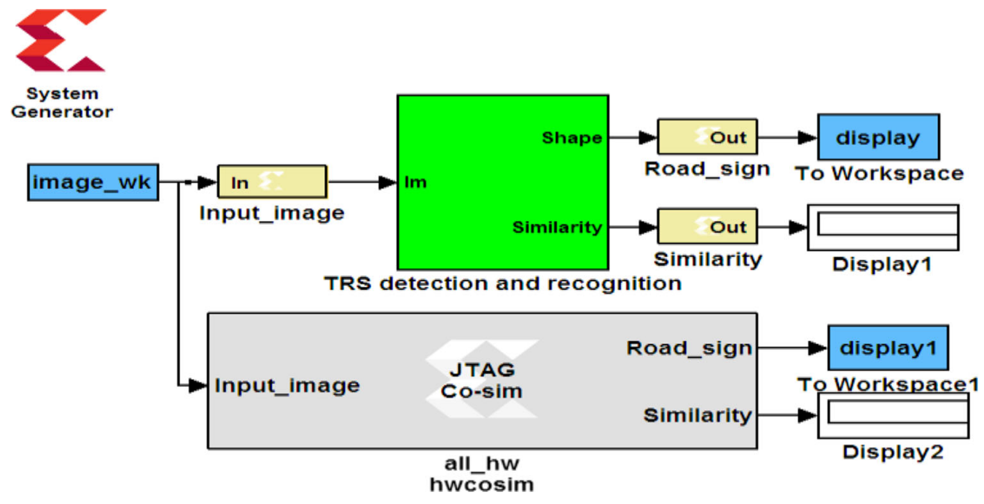
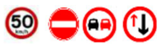


Fig. 18 Hardware block generation

Table 3 Performance of proposed method

Sign shape forms	TPR (sensitivity), %	FPR (specificity), %	Total
Set-1 	96.6	16.6	146
Set-2	90.9	25	126
Set-3	93.75	50	70

shape-form road signs (set-2) and triangle up or down shape-form (set-3). Receiver operating characteristic (ROC) curve is used to evaluate the accuracy of our system. Three sets of different road signs are regrouped and for each set, the true positive rate (TPR) and the false positive rate (FPR) were defined. The obtained results are depicted in Table 3.

The analysis is performed for each signs set. The experiments give different values of sensitivity and specificity as shown in the ROC curve of Fig. 19. The ROC curve of the three sets shows a good accuracy of the proposed algorithm since the area under curve (AUC) is between 0.85 and 0.89.

Even if a good accuracy is established, this is not enough to assess the efficiency of our method. In fact, our second purpose is to make the real-time implementation of the system successful on an FPGA device. Also the design time is a very important issue. The detection rate and the execution speed parameters should have acceptable values in the manner that the system can ensure a good detection rate while respecting real-time constraints. It is not acceptable to warn the driver later than the required time to make the necessary reaction. Morphological operations (color space conversion, filtering and thresholding) take

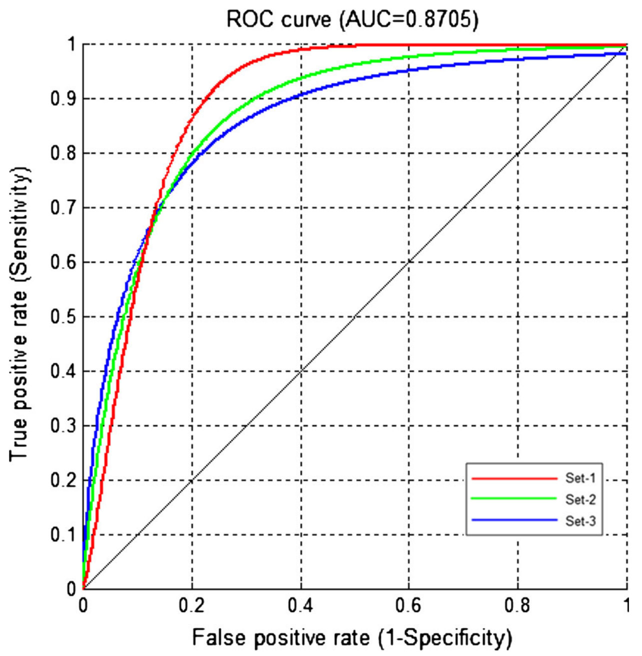


Fig. 19 ROC curve of all the set

27 ms and the identification phase takes 41 ms. The hole detection (Morphological operations and identification) and classification time is evaluated to 68 ms taking into consideration that these two stages are implemented in a pipelined way. These times were determined according to the hardware synthesis results obtained using a Virtex-5 FPGA platform running at a frequency of 86 MHz. In Table 4, a comparison with some references in terms of processing time and detection rate is presented.

The reliability and the effectiveness of the developed system will be proved throughout its evaluation for different real-time driving conditions. Depending on the application, real time may be defined very differently. Real-time system refers to a system that is able to process

Table 4 Performance comparison

References	Implementation platform	Clock speed	Detection speed (ms)	Detection rate (%)
[2]	Virtex-4	88 MHz	16	82
[5]	CPU&GPU	2.8 Ghz	250	90
[22]	Virtex-5	–	114	–
[34]	CYCLONE II DE2	1.83 GHZ	17.34	–
[49]	Virtex-5	100 MHz	70	90
Proposed algorithm	Virtex-5	86 MHz	68	91

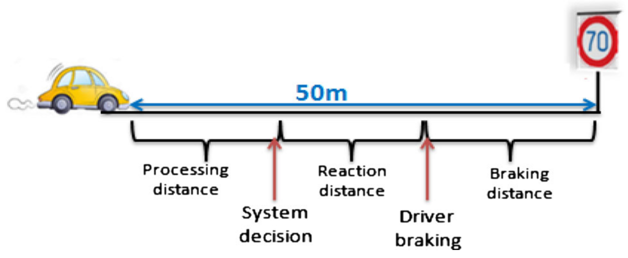


Fig. 20 A traffic sign system scenario

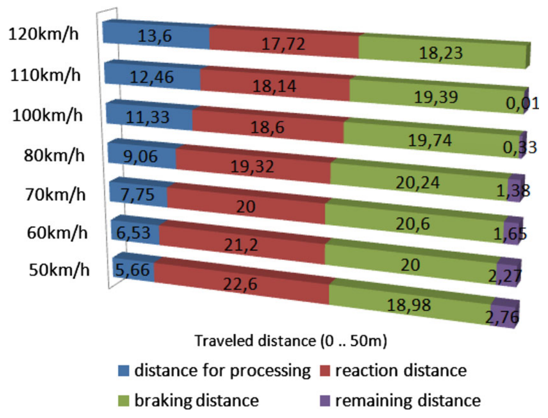


Fig. 21 Real-time driving distances measurement

the input data sufficiently rapid to be able to make the required actions in the right time [22]. To define the real-time conditions related to our system, let us suppose a vehicle moving on a city with different speeds (from 50 to 120 km/h). A distance of about 50 m from a traffic sign is estimated sufficient to get a clear panel image in a traffic scene. Two principle kinds of distances should be considered to evaluate the efficiency of our method. The first is the reaction distance (distance measured from the time the driver realizes the need to stop until he makes a reaction: braking) and the second is the braking distance (distance measured from the braking time until the time of stopping), as presented in Fig. 20.

In Fig. 21, we present some distance measurements in different real-time driving conditions. The distance needed to accomplish the processing step is measured with respect to the vehicle speed, as well as the reaction and braking distances.

For a vehicle moving at 110 km/h, which is the maximum speed permitted for many countries, the stopping distance (reaction and braking) is equal to 37.53 m. In the remaining distance (12.47 m), the system should process the input data and make decision. This distance is equivalent to approximately 408 ms. As the detection speed of the proposed system is 68 ms, for each acquired image, we can reach up to 6 acquisitions before the panel identification. By referring to experimental tests, this number of acquisitions can be sufficient to adequately take the final decision.

Other different real-time driving conditions are also described in Fig. 21 (50, 60, 70, 80, 100, and 110 km/h). For these conditions, the driver has a remaining distance to safely take the necessary decision (stop, speed increase, etc.). In the case of higher speeds, the same algorithm can be applied with a lower image acquisition rate (5, 4 or 3 images).

6 Conclusions

A system of traffic road sign detection and identification and its hardware implementation was presented in this work. For the adopted system, we tried to integrate the key approaches used in literature while ensuring a compromise between accuracy and processing time. The main advantages of the proposed processing method are; lighting conditions consideration, use of a region of search and use of a shape identification step before the classification. Also an algorithm based on linear operations was proposed for shape identification. It offers a good level of accuracy.

The performance of the proposed hardware implementation in terms of processing latency was evaluated relatively to the reaction distance, the braking distance and the vehicle speed. The evaluation results show that our system can support real-time driving conditions until the speed of 110 km/h. The XSG tool was used for the system development. The use of this tool has a big benefit in terms of conception time, since the same design was used firstly for the software validation and then for the hardware system generation.

References

1. Dembele, W.F., Kuhn, P.P.E., Ganley, R.T.: FPGA implementation of driver assistance camera algorithms. Technical report (2010)
2. Souani, C., Faiedh, H., Besbes, K.: Efficient algorithm for automatic road sign recognition and its hardware implementation. J. Real Time Image Proc. **9**(1), 79–93 (2014)
3. Hechri, A., Hmida, R., Mtibaa, A.: Robust road lanes and traffic signs recognition for driver assistance system. Int. J. Comput. Sci. Eng. **10**(1–2), 202–209 (2015)
4. Murphy-Chutorian, E., Trivedi, M.M.: N-tree disjoint-set forests for maximally stable extremal regions. In: BMVC, pp. 739–748, September 2006
5. Par, K., Tosum, O.: Real-time traffic sign recognition with map fusion on multicore/many-core architectures. J. Appl. Sci. **9**(2), 231–250 (2012). **(Acta Polytechnica Hungarica)**
6. De La Escalera, A., Moreno, L.E., Salichs, M.A., Armingol, J.M.: Road traffic sign detection and classification. IEEE Trans. Industr. Electron. **44**(6), 848–859 (1997)
7. Le, T.T., Tran, S.T., Mita, S., Nguyen, T.D.: Real time traffic sign detection using color and shape-based features. In: Nguyen, N.T., Le, M.T., Świątek, J. (eds.) Intelligent Information and Database Systems, pp. 268–278. Springer, Berlin (2010)
8. Loy, G., Barnes, N.: Fast shape-based road sign detection for a driver assistance system. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004) Proceedings, vol. 1, pp. 70–75 (2004)
9. Zadeh, M.M., Kasvand, T., Suen, C.Y.: Localization and recognition of traffic signs for automated vehicle control systems. In: Intelligent Systems and Advanced Manufacturing, pp. 272–282. International Society for Optics and Photonics (1998)
10. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. Neural Netw. **32**, 323–332 (2012)
11. Escalera, S., Baró, X., Pujol, O., Vitrià, J., Radeva, P.: Background on traffic sign detection and recognition. In: Traffic-Sign Recognition Systems, pp. 5–13. Springer, London (2011)
12. Glavtchev, V., Muyan-Ozcelik, P., Ota, J.M., Owens, J.D.: Feature-based speed limit sign detection using a graphics processing unit. In: IEEE Intelligent Vehicles Symposium (IV), pp. 195–200, June 2011
13. Rudorfer, M.: Design and implementation of a classification algorithm for speed limit traffic sign recognition. Thesis report, Department of Machine Tools and Factory Management Division of Industrial Automation, June 2014
14. Eichner, M.L., Breckon, T.P.: Integrated speed limit detection and recognition from real-time video. In: Proceeding on IEEE Intelligent Vehicle Symposium, The Netherlands, 2008
15. Han, Y., Oruklu, E.: Real-time traffic sign recognition based on Zynq FPGA and arm SOCS. International IEEE Conference on Electro/Information Technology, pp. 373–376, 2014
16. Muyan-Ozcelik, P. Glavtchev, V. Ota, J.M, Owens, J.D.: A template-based approach for real-time speed-limit sign recognition on an embedded system using GPU computing. In: Proceeding on 32nd DAGM Conference on Pattern Recognition, pp. 162–171, 2010
17. Ugolotti, R., Nashed Youssef, S.G., Cagnoni, S.: Real-time GPU based road sign detection and classification. Chapter Parallel Probl. Solving Nat. **1**, 153–162 (2012)
18. Dyczkowski, K. Gadecki, P., Kulakowski, A.: Traffic sign recognition system. In: World Conference on Soft Computing, San Francisco, USA, May 23–26, 2011
19. Fang, C.Y., Chen, S.W., Fuh, C.S.: Road-sign detection and tracking. IEEE Trans. Veh. Technol. **52**(5), 1329–1341 (2003)
20. Hechri, A., Mtibaa, A.: Robust road sign recognition system for autonomous mobile robot. Int. J. Comput. Sci. Eng. Syst. **6**(1), 19–29 (2012)
21. Prieto, M., Allen, A.: Using self-organizing maps in the detection and recognition of road signs. Image Vis. Comput. **27**, 673–683 (2009)
22. Waite, S., Oruklu, E.: FPGA-based traffic sign recognition for advanced driver assistance systems. J. Transp. Technol. **3**(1), 1–16 (2013)
23. Brkic, K.: An overview of traffic sign detection methods. Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing, vol. 3, p. 10000, Unska (2010)
24. Zakir, U.A. Leonce, N.J., Edirisinghe, E.A.: Road sign segmentation based on color spaces: a comparative study. In: Proceeding on the 11th IASTED International Conference on Computer Graphics and Imaging (CGIM), Innsbruck, Austria (2010)
25. Lai, C.: An efficient real-time traffic sign recognition system for intelligent vehicles with smart phones. In: Proceeding in International Conference on Technologies and Applications of Artificial Intelligence, Hsinchu, pp. 195–202 (2010)
26. Soendoro, D., Supriana, I.: Traffic sign recognition with color-based method shape-arc estimation and SVM. In: proceedings of International Conference on Electrical Engineering and Informatics, Bandung, pp. 1–6 (2011)
27. Yabuki, N., Matsuda, Y., Fukui, Y., Miki, S.: Region detection using color similarity. In: Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 98–101 (1999)
28. Schiekel, C.: A fast traffic sign recognition algorithm for gray value images. In: Computer Analysis of Images and Patterns. 8th International Conference, CAIP'99 Ljubljana, Slovenia, September 1–3, 1999 Proceedings. Springer, Berlin, Heidelberg (1999)
29. Ach, R. Luth, N., Techmer, A.: Real-time detection of traffic signs on a multi-core processor. In: Proceeding of the IEEE Intelligent Vehicles Symposium, pp. 307–312 (2008)
30. De la Escalera, A., Armingol, J., Pastor, J., Rodriguez, F.: Visual sign information extraction and identification by deformable models for intelligent vehicles. IEEE Trans. Intell. Transp. Syst. **5**(2), 57–68 (2004)
31. The MathWorks User's Guide. http://www.opal-rt.com
32. Barnes, N., Zelinsky, A.: Real-time radial symmetry for speed sign detection. In: Proceeding on IEEE Intelligent Vehicles Symposium, pp. 566–571, 2004
33. Antolovic, D.: Review of the Hough transform method, with an implementation of the fast Hough variant for line detection. Department of Computer Science, Indiana University, Indiana (2008)
34. Souki, M.A., Boussaid, L., Abid, M.: An embedded system for real-time traffic sign recognizing. In: 3rd International Design and Test Workshop, 2008. IDT 2008, pp. 273–276 (2008)
35. Adam, A., Ioannidis, C.: Automatic road-sign detection and classification based on support vector machines and hog descriptors. Int Soc Photogramm. Remote Sens. ISPRS Ann Photogramm. Remote Sens. Spatial Inform. Sci. **II-5**, 1–7 (2014)
36. Wali, S.B., Hannan, M.A., Abdullah, S., Hussain, A., Samad, S.A.: Shape matching and color segmentation based traffic sign detection system. Threshold **90**, 255 (2015)
37. Cao, T.P., Deng, G., Elton, D.: Grayscale image segmentation for real-time traffic sign recognition: the hardware point of view. In: Proceeding of SPIE Electronic Imaging, pp. 724405–724405. International Society for Optics and Photonics, February 2009

38. Kiran, C.G., Prabhu, L.V., Rahiman, V.A., Rajeev, K.: Support vector machine learning based traffic sign detection and shape classification using distance to borders and distance from center features. In: TENCON IEEE Region 10 Conference, pp. 1–6. IEEE, November 2008
39. Martín, P., Bueno, E., Rodríguez, F.J., Machado, O., Vuksanovic, B.: An FPGA-based approach to the automatic generation of VHDL code for industrial control systems applications: a case study of MSOGIs implementation. Math. Comput. Simul. **91**, 178–192 (2013)
40. Van Beeck, K., Heylen, F., Meel, J., Goedemé, T.: Comparative study of model-based hardware design tools. In: Proceedings of European Conference on the Use of Modern Electronics in ICT, ECUMICT, vol. 5, p. 2860, February 2010
41. Suthar, A.C., Vayada, M., Patel, C.B., Kulkarni, G.R.: Hardware software co-simulation for image processing applications. Int. J. Comput. Sci. Issues **9**(2), 560–562 (2012)
42. Wang, C.C., Shi, C., Brodersen, R.W., Marković, D.: An automated fixed-point optimization tool in MATLAB XSG/SynDSP environment. ISRN Signal Process. **2011**, 17 (2011). doi:10.5402/2011/414293
43. Moctezuma, J.C., Sanchez, S., Alvarez, R., Sánchez, A.: Architecture for filtering images using Xilinx system generator. In: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications, pp. 284–289. World Scientific and Engineering Academy and Society (WSEAS), January 2008
44. Karthigaikumar, P., Kirubavathy, K.J., Baskaran, K.: FPGA based audio watermarking—Covert communication. Microelectron. J. **42**(5), 778–784 (2011)
45. Toledo, A., Vicente-Chicote, C., Suardíaz, J., Cuenca, S.: Xilinx system generator based HW components for rapid prototyping of computer vision SW/HW systems. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) Pattern Recognition and Image Analysis, pp. 667–674. Springer, Berlin (2005)
46. Stereopolis database. (2015). http://www.itowns.fr/roadsign.php. Accessed 20 April 2015
47. German traffic sign recognition benchmark. http://benchmark.ini.rub.de. Accessed 20 April 2015
48. Miyata, S., Yanou, A., Nakamura, H., Takehara, S.: Road sign feature extraction and recognition using dynamic image processing. Int. J. Innov. Comput. Inform. Control **5**(11), 4105–4113 (2009)
49. IRMAK, H.: Real time traffic sign recognition system on FPGA. Thesis, Graduate School of Natural and Applied Sciences of Middle East Technical University (2010)

Rihab Hmida received her degree of Engineer in Electrical Engineering from the University of Monastir (ENIM), Tunisia, in 2010. In 2012, she received her MS degree in Electrical Engineering in the National School of Engineering of Monastir. She is currently preparing her PhD degree in the Electronics and Microelectronics Laboratory (EμE) at the Faculty of Sciences of Monastir. Her current research interests include camera systems modeling, mobile robots controlling and designing of underwater 3D scenes through real-time image and video processing.

Abdessalem Ben Abdelali received his degree in Electrical Engineering and his DEA in industrial informatics from the National School of Engineering of Sfax (ENIS), Tunisia, respectively, in 2001 and 2002. He received his PhD from ENIS and Burgundy University (BU), France, in 2007. Since 2008 he has been working as an Assistant Professor in digital embedded electronic at the High Institute of Informatics and Mathematics of Monastir (ISIMM). His current research interests include reconfigurable architectures and hardware implementation of image and video processing applications.

Abdellatif Mtibaa received his PhD degree in Electrical Engineering at the National School of Engineering of Tunis. Since 1990 he has been an Assistant Professor in Micro-Electronics and Hardware Design with Electrical Department at the National School of Engineering of Monastir. Since 2007, he has been a professor at the electrical engineering department at the ENIM. His research interests include high-level synthesis, rapid prototyping and reconfigurable architectures for real-time multimedia applications.