IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 23, 2019, accepted December 5, 2019, date of publication December 12, 2019,
date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2959084

Parallelism Optimized Architecture on FPGA for Real-Time Traffic Light Detection

XUE-HUA WU[1], RENJIE HU[1], AND YU-QING BAO[2]
[1]School of Electrical Engineering, Southeast University, Nanjing 210096, China
[2]School of Electrical and Automation Engineering, Nanjing Normal University, Nanjing 210023, China

Corresponding author: Renjie Hu (hurenjie@seu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 51707099.

ABSTRACT In this paper, a portable assistance system is designed to help the visually impaired to detect the traffic light. The designed system is realized on the basis of the AdaBoost algorithm, which is fast and robust in object detections. In order to accelerate the AdaBoost-based approach, a flexible parallel architecture is implemented on the field-programmable gate array (FPGA) platform. The architecture is designed utilizing the parallelism of computations in the AdaBoost-based detection. The computations of the window integral image are implemented in parallel, and the confidences of the weak classifiers are calculated in parallel. The parameters of the weak classifiers are trained by the AdaBoost algorithm with multi-layer features in the MATLAB software, and then are configured on the FPGA platform via the Vivado design suite before the detection process. The parallelism optimized architecture is implemented on an Artix-7 FPGA at 200 MHZ. Experiments show that it can detect the traffic light in videos with a rate of 30 frames per second (fps).

INDEX TERMS AdaBoost algorithm, field-programmable gate array, integral image, parallel architecture, traffic light detection.

I. INTRODUCTION

Many people in the world suffer from impaired vision. The visually impaired are limited in the independent mobility in urban roads, which is mainly because they cannot catch the information for safety in the heavy traffic [1]. Since the traffic light provides the crucial information to go and pause, it is urgent to develop a cheap and portable assistance system for real-time traffic light detection [2]. In this paper, a real-time detection system is designed on the FPGA platform to detect the nearby traffic light. Our research is motivated by the popularity of cameras and the developments of hardware technology. The portable computation hardware equipped with small cameras provides the possibilities of mobile vision for real-time traffic light detection.

The task of the real-time detection is to determine the position of the traffic light in each frame of the video. In general, the detection approaches consists of two categories: 1) model-based, 2) learning-based [3]. In the model-based approaches, heuristic knowledge of the traffic light (such as shape, colour) is used to design a filtering scheme, which relies on the experience of researchers. The model-based approaches are simple and comprehensible. In [4], the colour distribution of the traffic light is analyzed to build a colour-based model. In [5], the Hough transformation is adopted to detect the circular edge of the traffic light in a shape-based model. In [6], properties of shape, colour arrangement, circuitry, background, design and installation are utilized to build a model of the traffic light. Then the model is implemented on a mobile phone for test with videos about 5 to 10 fps. Another approach on the phone is presented in [7], in which a filtering scheme is designed to detect the traffic light with a robust image acquisition method under particular exposure conditions. In the learning-based approaches, features of the traffic light are learnt by algorithms to construct a detector without manual intervention. In [8], the AdaBoost algorithm with the aggregate channel features is adopted to detect the traffic light. In [9], a CPU-GPU fusion-based approach of traffic light detection is implemented on a smart phone platform. In their prototype, a kernel extreme learning machine is adopted to detect the traffic light. The detection rate of the prototype is about 20 fps. In [10], the adaptive background suppression filter is adopted for coarse detection, and then the cascaded linear support vector machine with features (histograms of oriented gradients and the local colour histograms) is used for fine detection. The similar approaches

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

TABLE 1. Summary of the existing methods of the traffic light detection.

	Roters [6]	Mascetti [7]	Philipsen [8]	Liu [9]	Shi [10]	Li [12]	Wang [13]
Resource	Video	Image with proper exposure	Video	Video	Video	Video	Dual-channel video
Frame/Image size	640×480	1413×1884	1280×960	1280×720	1000×1000	1280×960	111×111 (ROI)
Colour	RGB	HSV	CIE-LUV	HSL	RGB	CIE-LUV, CIE-Lab	RGB
Features	–	–	Aggregate channel features	HOG-LBP features	HOG, local colour histograms	Aggregate channel features	–
Candidates extraction	–	–	Upper 1280×580 part of the frame	Ellipsoid geometry threshold model	Adaptive background suppression filter	Prior characteristic analysis	Salience map filtering
Algorithm	Filtering scheme	Filtering scheme	AdaBoost	Kernel extreme learning machine	Support vector machine	Boosting (baseline)	Deep learning
Postprocessing	Temporal analysis	–	–	Spatial-temporal analysis	–	Inter-frame correlation analysis	Temporal trajectory analysis
Platform	ARM	ARM	Not provided	CPU & GPU	CPU	CPU	CPU & GPU
Clock frequency	330 MHz	Not provided	Not provided	CPU 2.3 GHz; GPU 450 MHz	Not provided	2.6 GHz	2.5GHz
Rate	5-10 fps	Several times a second	1.275 seconds per frame	20 fps	15fps	0.081 seconds	30-40 fps

are found in references [11]–[13]. In [11], the background filter is used to generate the candidate regions in complex background, and then the cascaded classifier trained by the AdaBoost algorithm is used to detect the traffic light. In [12], the traffic light is recognized in images by a succession of basic steps, including a process by prior characterisc analysis, aggregate channel feature learning and inter-frame correlation analysis. In [13], a high dynamic range camera is set to capture consecutive low and high exposure frames. In their prototype, the traffic light candidates are detected in low exposure frames and then classified using a deep learning algorithm in high exposure frames. The summary of the existing methods of the traffic light detection is listed in Table 1.

The learning-based approaches are widely used due to their accuracy and robustness. Though each of the supervised learning algorithms has a good performance, it is difficult to select the best algorithm according to the no free lunch (NFL) theorems in supervised learning [14]. However, a breakthrough is made by considering the goal of developing a cheap and portable assistance system. The detection rate of the AdaBoost algorithm is faster than that of SVM, although it needs more time in the training process [15]. The AdaBoost algorithm is a classic ensemble method, in which multiple decisions are synthesized to reduce the risk of making a serious poor decision [16]. The synthesis of decisions provides the possibility of ensembles of new decisions to update the original system [17]. There are several prototype systems implemented on FPGA for the AdaBoost-based object detections [18]. In the approach based on the deep learning, hierarchical features are computed intensively instead of the easy-compute hand-crafted features [19]. The level of hierarchical features increases with the depth of the deep network [20]. Since the representations of feature maps and weights are floating-point operands, the approach is usually implemented on GPU which results in high-precision and excessive memory [21]. Recently the deep-learning-based approaches are optimized to run on FPGA [22].

FPGA is flexible in configuration and consumes low power. It can reconfigure the compute-intensive calculations for acceleration [23]. The reconfigurable computing on FPGA is applied to design the accelerator of AlexNet [24], VGG-16 [25], and optimized LeNet-5 [26]. The optimized architectures change with different networks. The novel components in networks bring a risk that the developed system needs to be redesigned for a new generation [21]. In addition, there are challenges of network compression, data quantization from floating-point to fixed-point, nonlinear function representation, frequent memory accesses to develop the networks on FPGA [24]–[26]. In contrast, the implementation of the AdaBoost-based detection on FPGA is simple in structure and easy to upgrade. In the structure, the sum of confidences of weak classifiers is compared with a threshold to get the detection result [27], [28].

The AdaBoost-based detection on FPGA is capable to accelerate in videos of higher resolution and frame rates with progress of cameras [29]–[31]. Several specialized hardware architectures are reported to reconfigure the intensive computations on FPGA within limited power and size [32]–[38]. In [32], a partially parallel architecture is proposed for the face detection using the AdaBoost algorithm with Haar-like features. The proposed architecture is implemented on a Virtex-5 FPGA with the detection rate of 30 fps on VGA videos. In the architecture, the integral image of the window is calculated by the one-pass computation, and the weak classifiers in the early stages are implemented in parallel. In [33], a real-time face detection system is implemented on a Cyclone II FPGA. In the approach, it is found that the candidate regions can pass the similar number of weak classifiers at different scales. So the performances of the candidate regions at the smaller scale are used to skip unlikely

TABLE 2. Summary of the existing hardware architectures for the AdaBoost-Based object detection.

	Hiromoto [32]	Yang [33]	Kyrkou [34]	Zhou [35]	Jin [37]
Target	Face	Face	Face	Face	Face
Image size	640 × 480	Not provided	320 × 240	1024 × 1024	640 × 480
Colour	Gray	Gray	Gray	Gray	Gray
Features	Haar-like features	Haar-like features	Haar-like features	Haar-like features	LBP features
Size of window detector	24 × 24	20 × 20	24 × 24	24 × 24	19 × 19
Size of integral image	25 × 25	Frame size	80 × 60	24 × 24	No
Squared integral image	Yes	Yes	Yes	Yes	No

windows at the larger scale. The work [34] presents a flexible parallel architecture based on a massively parallel systolic computation of the classification engine. The architecture is implemented on a Virtex II pro FPGA. In the architecture, the systolic array implementation is used to boost the parallel computation of the classifiers and parallelize the calculation of the window integral image. In some architectures, the current window integral image can be generated in one clock cycle, due to the correlation between the integral images of the adjacent windows [31], [35]. In the work [36], a heterogeneous system integrated by the ARM and FPGA platform is proposed to accelerate the AdaBoost-based detector. The detector is faster than that on a single ARM platform. In the work [37], [38], the local binary pattern (LBP) transform is adopted in the preprocessing step to minimize the effect of different lighting conditions. The summary of the existing hardware architectures for the AdaBoost-based detection is listed in Table 2. The performances of the mentioned architectures are verified in their applications. However, the parallelism of the computations is not fully exploited. In this paper, we propose a parallelism optimized architecture on FPGA to detect the traffic light. The contributions of this paper are mainly summarized as following:

- The computation of the window integral image is fully parallel designed, and all of the weak classifiers are implemented in parallel.
- The parameters of the weak classifiers are trained by the AdaBoost algorithm with multi-layer features.

The remaining of this paper is organized as follows: In Section II, concepts of the AdaBoost-based traffic light detection with multi-layer features is introduced. In Section III, the framework of the designed system is described in details. The experiment results are shown in Section IV. In Section V, the conclusions are given and the future work is discussed.

II. ADABOOST-BASED TRAFFIC LIGHT DETECTION WITH MULTI-LAYER FEATURES

In this section, concepts of the AdaBoost-based traffic light detection with multi-layer features are introduced. The AdaBoost algorithm is short for the adaptive boosting algorithm, which adjusts adaptively to the errors of weak classifiers [39]. The AdaBoost algorithm is popular after its successful application in the face detection. In the application, Haar-like features are used to describe the information of the face, and the AdaBoost algorithm is adopted to train

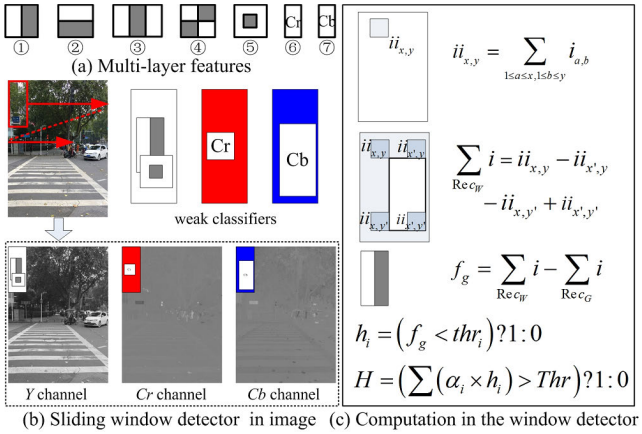

FIGURE 1. Concepts of the AdaBoost-based traffic light detection with multi-layer features. (a) Multi-layer features, (b) Sliding window detector in image, (c) Computation in the window detector.

weak classifiers on the features. The Haar-like features are image representations of the Haar wavelet function, which is efficient to capture the differences along the horizontal, vertical and diagonal directions [40]. The traditional Haar-like features are crafted in the grayscale image and are limited in the colour representation. In order to describe the colour information, additional features are designed in the chrominance layers. The multi-layer features are good at extracting the shape and colour information of the traffic light in natural scenes [41].

The concepts of the AdaBoost-based traffic light detection with multi-layer features are shown in Fig.1. The multi-layer features consist of seven types. The features ①-⑤ are luminance-layer features, which are implemented in the *Y* channel. The features ⑥-⑦ are chrominance-layer features, which are implemented in the *Cr*, *Cb* channel respectively. Since the YCrCb colour space is an opponent luma-chroma space, the quantization of the chrominance is not affected by the luminance. As a result, the chrominance-layer features are able to accurately extract the colour information under different lighting conditions. The multi-layer features are composed of rectangles. The features change with rectangles of different number, position or size. There are 89925 features in a window of 30 × 10. The value of each feature is calculated by sum of the intensities in the white rectangle subtracting that in the gray rectangle (or 0 if there is no gray rectangle),

$$f_g = \sum_{Rec_W} i - \sum_{Rec_G} i. \tag{1}$$

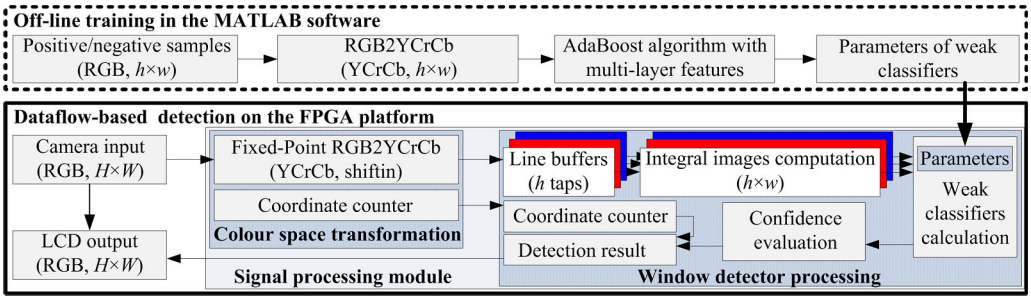

FIGURE 2. Framework of the designed system. In the dataflow-based detection on the FPGA platform, videos captured by the camera are processed in the signal processing module, and the detection result is displayed on the LCD. The parameters of the weak classifiers configured on FPGA are trained by the AdaBoost algorithm with multi-layer features off-line.

where f_g represents the value of feature, and i represents the intensity of pixel. The pixel sum within each rectangle can be quickly calculated with four values in the integral image,

$$\sum_{Rec_W} i = ii_{x,y} - ii_{x',y} - ii_{x,y'} + ii_{x',y'}. \tag{2}$$

where $ii_{x,y}$ represents the value at the point (x, y) in the integral image. In computations of the integral image in each channel, the value represents the sum of intensities on the upper left of the point, that is,

$$ii_{x,y} = \sum_{1\le a\le x, 1\le b\le y} i_{a,b}. \tag{3}$$

Based on the easy-use features, the AdaBoost algorithm is adopted to train the parameters of weak classifiers in iterations. During each iteration, the most distinguishing feature is selected to train a weak classifier on samples of updated distribution. The function of the weak classifier is as below,

$$h_i = (f_g < thr_i)?1 : 0. \tag{4}$$

where f_g represents the feature value, thr_i represents the threshold of the weak classifier. The threshold can be quickly trained by sorting feature values [27]. The feature value with the minimal error can be taken as the threshold of the weak classifier. After the parameters of the weak classifier are determined, the weight of the weak classifier is calculated using its error as below,

$$\alpha_i = \log \frac{1-\varepsilon_i}{\varepsilon_i}. \tag{5}$$

In addition, the distribution of samples are updated with new sample weights which are calculated as below,

$$w_{i+1,j} = w_{i,j} exp(-\alpha_i h_i(x_j) y_j). \tag{6}$$

In this equation, $w_{i,j}$ is the weight of sample x_j in i-th iteration, α_i is the weight of the i-th weak classifier, $h_i(x_j)$ is the label given by the weak classifier, and y_j is the original label of sample x_j. After all of the iterations are finished, the weak classifiers are synthesized to construct a strong classifier as below,

$$H = (\sum(\alpha_i \times h_i) > Thr)?1 : 0. \tag{7}$$

where α_i and h_i represents the weight and value of the weak classifier respectively, Thr represents the threshold of the strong classifier. The traffic light is detected when the weighted sum of the weak classifiers is greater than the threshold of the strong classifier.

In order to detect the traffic light in uncertain positions, the window detector slides across the image with a spacing stride of one. The number of the sub-windows with different positions is calculated as below,

$$N = \frac{W-w+1}{s} \times \frac{H-h+1}{s}. \tag{8}$$

where s represents the spacing stride, W, H represents the width, height of the image respectively, and w, h represents the width, height of the window detector respectively. The size of the window detector is independent of the size of the image.

III. THE DESIGNED SYSTEM

In this section, the designed system is introduced. The framework of the designed system is shown in Fig. 2. In the dataflow-based detection process, the video of the natural traffic scene is captured by a camera in the RGB mode. The size of each frame in the video is $H \times W$. The data in each frame are processed in the signal processing module. This module consists of two blocks. In the block of colour space transformation, the pixel in the frame is transformed into the YCrCb colour space in sequence, and the coordinate of the pixel is counted to record the position. In the block of window processing, the pixels of each channel are stored in a line buffer with h taps. Then the integral images of the window of $h \times w$ are calculated with the taps outputs. Based on three integral images, the confidences of weak classifiers are evaluated in order to calculate the detection result. The parameters of the weak classifiers are previously trained by the AdaBoost algorithm with multi-layer features in the MATLAB software

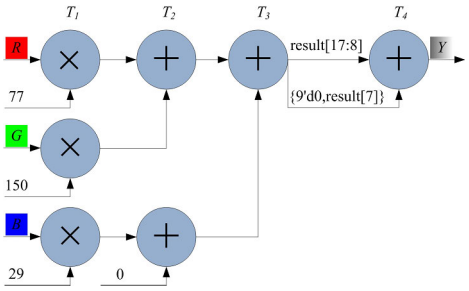

FIGURE 3. Illustration of the fixed-point arithmetic transformation of Y channel. The transformation costs four clock cycles per pixel.

off-line. The position of the detection result is provided by the coordinate counter. For visual display, the detection result is displayed on an LCD in the RGB mode. The size of the LCD equals to the frame size.

A. PARAMETERS ACQUISITION AND CONFIGURATION

The parameters of weak classifiers are trained in MATLAB before the configuration on FPGA. In the training, the positive and negative samples are adjusted to the size of $h \times w$ firstly. Then the primary RGB data in each sample are converted to luminance data and colour-difference data in the YCrCb colour space. The luminance data Y is helpful to minimize the interference of the illumination, and the colour-difference data Cr and Cb are efficient to indicate the presence of the colour [42]. The conversion is formulated as follows [43]:

$$Y = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B. \quad (9)$$

$$Cr = 0.5 \times R - 0.4187 \times G - 0.0813 \times B + 128. \quad (10)$$

$$Cb = -0.1687 \times R - 0.3313 \times G + 0.5 \times B + 128. \quad (11)$$

where R, G, B is the pixel value in red, green, and blue channel respectively, Y represents the luminance data, Cr represents the colour-difference data of $(R - Y)$, Cb represents the colour-difference data of $(B-Y)$. After the colour space transformation is finished, the AdaBoost algorithm with multi-layer features is adopted to determine the parameters of the weak classifiers. The concepts of the AdaBoost-based traffic light detection with multi-layer features are described in Section II. Then the parameters of the weak classifiers are stored in the memory of FPGA for computations.

B. FIXED-POINT RGB2YCRCB

After realizing the parallelism optimized architecture on the FPGA platform, the dataflow-based detection can be carried out. The dataflow of primary RGB data is delivered to the colour space transformation block in the form of one pixel after another. The fixed-point arithmetic transformation is adopted since the fixed-point computation is faster than the floating-point computation [44], [45]. In the transformation, the division of divisor 256 is realized by truncating the last eight bits of the data value, which is shown in Fig. 3. The fixed-point arithmetic transformation costs four clock cycles per pixel.

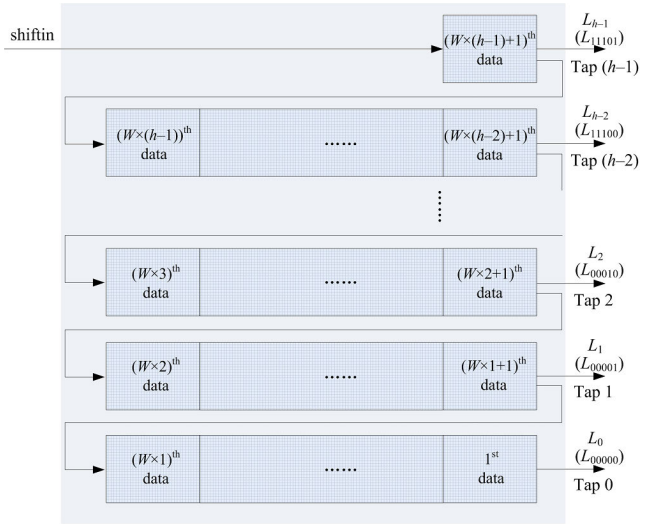

FIGURE 4. Illustration of the dataflow in the line buffer. The line buffer is composed of one input, $W \times (h-1)+1$ shift registers, and h outputs. It can output the data in the same column of h adjacent rows simultaneously.

C. LINE BUFFERS

There are three line buffers in the block of window processing. The data in the Y channel, Cr channel, and Cb channel are stored respectively in each line buffer one pixel after another. The line buffer is constructed by registers. As shown in Fig. 4, the registers are connected in a serial chain. During every clock cycle, the data of the new pixel is shifted in the line buffer. At the same time, each register shifts its storage to the right register and stores the data from the left register. Each line buffer can store $W \times (h-1)+1$ data in total. Starting with the first data, the data at each interval W is tapped out respectively. Hence, the line buffer can output h data in the same column of adjacent rows simultaneously per clock cycle. These outputs construct each column in a window of $h \times w$.

D. INTEGRAL IMAGE COMPUTATION

The outputs of the line buffer are utilized to calculate the integral image of the window. The integral image calculation is intensive, and it needs frequent access to memory [46]–[48]. Herein numerous registers are used to store data for efficiency calculations. The integral image calculation consists of two stages.

On the first stage, the data at the same column are accumulated in an extension of the application of the balanced binary tree in the all-prefix-sums operation [49]–[51]. The tree technique has two advantages: 1) the partial sums can be calculated in parallel; 2) the partial sums can be reused

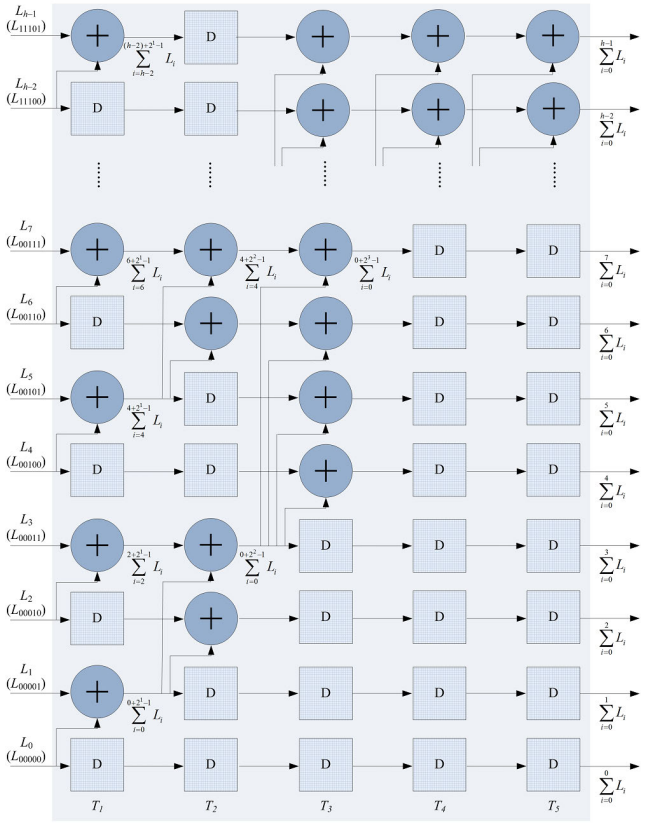

FIGURE 5. Parallel computation of the column accumulative sum in the calculation of the window integral image. In the diagram, D is the register, which represents data delay. T_i represents the i-th clock cycle.

to generate the desired sum. It takes at least $\lceil \log_2 n \rceil$ clock cycles to calculate the column accumulative sum at the n-th row. For instance, the number n can be expressed in binary form as $5'bx_4x_3x_2x_1x_0$, where the value of each bit is zero or one. The calculation of the column accumulative sum is shown in Fig. 5. The calculation takes five clock cycles. In the first clock cycle, the partial sums over regions of two adjacent rows are calculated in parallel. In the second clock cycle, the partial sums over regions of four adjacent rows are generated with the sums calculated in the first clock cycle. In the third clock cycle, the partial sums over regions of eight adjacent rows are generated with the sums calculated in the previous clock cycle. The partial sums over regions of sixteen, thirty-two adjacent rows are similarly generated in the fourth, fifth clock cycle respectively. The calculation is formulated as below,

$$\begin{aligned}\sum_{i=0}^{n} L_i = x_4 \times \sum_{i4=n4}^{n4+2^4-1} L_{i4} + x_3 \times \sum_{i3=n3}^{n3+2^3-1} L_{i3} \\ + x_2 \times \sum_{i2=n2}^{n2+2^2-1} L_{i2} + x_1 \times \sum_{i1=n1}^{n1+2^1-1} L_{i1} \\ + x_0 \times L_{n0} + L_n. \end{aligned} \tag{15}$$

In the equation, L_i represents the output in the i-th row of the line buffer, and $n - n4$ represent the number of the output.

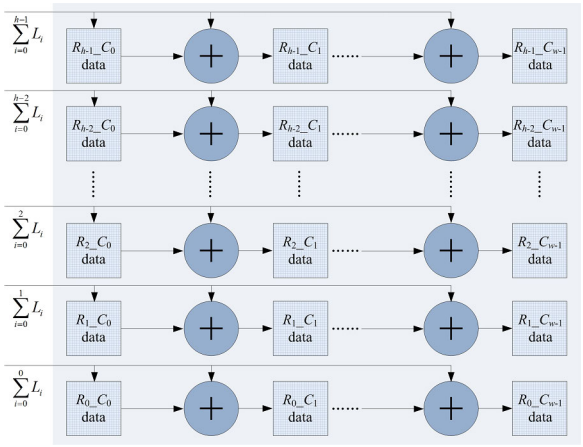

FIGURE 6. Refreshment of the window integral image by the addition in each row. The column accumulative sums are accumulated in each row per clock cycle.

Specifically, each number of the output can be expressed as below,

$$\begin{aligned} n &= 5'bx_4x_3x_2x_1x_0, \quad n0 = 5'bx_4x_3x_2x_10, \\ n1 &= 5'bx_4x_3x_200, \quad n2 = 5'bx_4x_3000, \\ n3 &= 5'bx_40000, \quad n4 = 5'b00000. \end{aligned} \tag{16}$$

On the second stage, the column accumulative sums are accumulated in each row, which is shown in Fig. 6. In every clock cycle, registers which store values of the first column in the integral image are refreshed by the new column accumulative sums. At the same time, the remaining registers are refreshed by the sum of its left value and the new column accumulative sum, respectively. The row accumulative sums are calculated as below,

$$\begin{aligned} R_n\_C_0 &= \sum_{i=0}^{n} L_i, \\ R_n\_C_{m+1} &= R_n\_C_m + \sum_{i=0}^{n} L_i. \end{aligned} \tag{17}$$

where $R_n\_C_m$ represents the data at the position (n, m) in the integral image of the window. The integral image of each channel in the YCrCb colour space is calculated simultaneously.

E. WEAK CLASSIFIERS CALCULATION

The integral images are stored in registers, where all integral pixel values are available to access synchronously. On the integral image of each channel, the computations of weak classifiers are implemented in parallel. These weak classifiers are previously configured on FPGA via the Vivado design suite.

The configuration of each weak classifier changes with the selected feature. In general, there are seven types of configuration according to the category of the multi-layer features. The most complex feature consists of four adjacent rectangles. To calculate the feature value, nine registers in

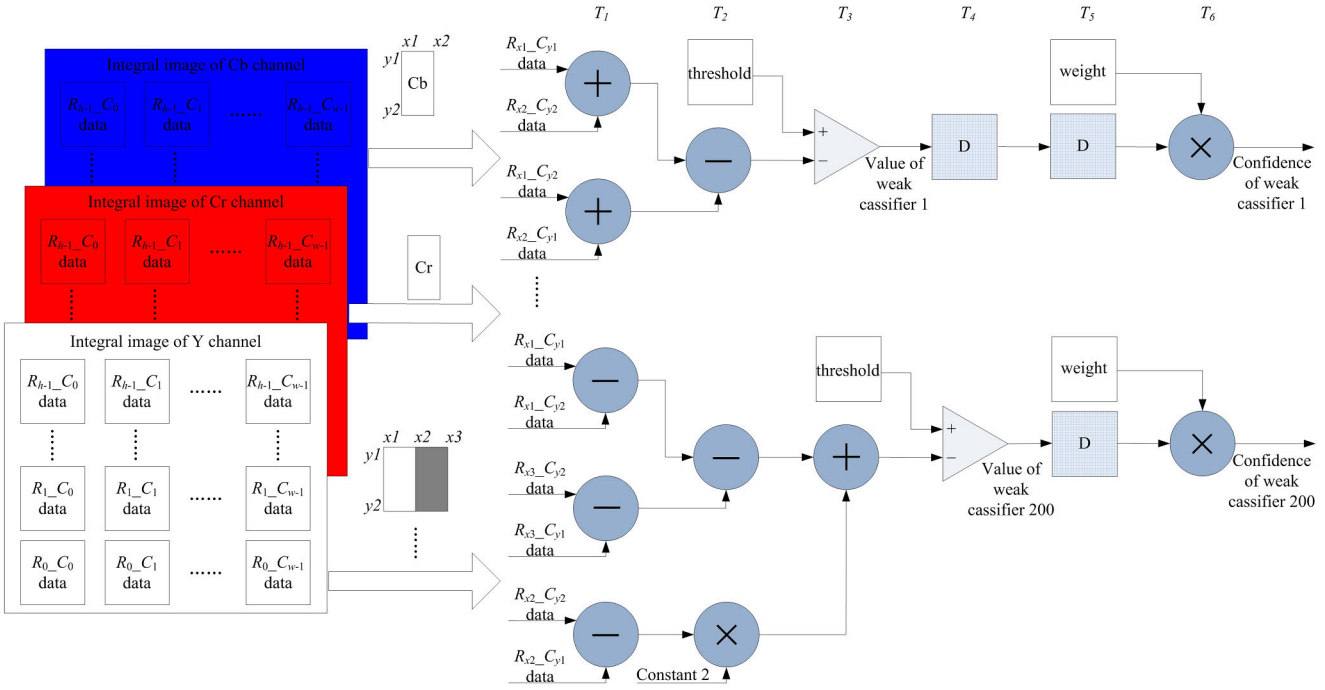

FIGURE 7. Parallel implementation of the weak classifiers. In each weak classifier, the value of the selected feature is quickly calculated on the integral image of *Y*, *Cr*, and *Cb* channel respectively. The computations of 200 weak classifiers are implemented in parallel. It takes six clock cycles to calculate the confidence of each weak classifier.

the integral image of *Y* channel are accessed synchronously. It takes five clock cycles to calculate the confidence of the corresponding weak classifier [31]. The calculation of feature with fewer rectangles is simpler and faster. In order to realize the parallelism of all weak classifiers, several registers are used to store the values of weak classifiers with fewer calculation steps. As shown in Fig. 7, the value of the chrominance-layer feature which consists of one rectangle can be calculated in two clock cycles. Firstly, the data in four registers in the integral image of *Cb* channel are read into two independent adders for addition operation. Secondly, the two sums are subtracted to calculate the feature value. The value of the weak classifier is true if the feature value is smaller than the threshold. Then the value of the weak classifier is stored in registers with the latency of two clock cycles. The similar calculation is implemented on the two-rectangle feature, and the value of the weak classifier is stored with one clock cycle delay. After getting the values of all weak classifiers, in the sixth clock cycle, the confidence of each weak classifier is calculated by multiplying the value by weight of the weak classifier.

These confidences are evaluated to calculate the detection result. The window is considered to contain the traffic light when the sum of the confidences is higher than the threshold tuned by off-line training.

IV. EXPERIMENT RESULTS

In order to test the performance of the designed system, the off-line training is carried out in the MATLAB software,

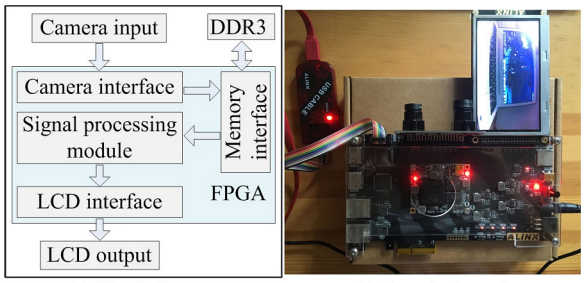

FIGURE 8. Experimental platform. (a) Block diagram, (b) Practical result. The video of the practical result is provided in supplemental materials.

and the hardware architecture is implemented on an FPGA platform for experiments.

The off-line training is realized by programming on the basis of GML AdaBoost Matlab Toolbox [52]. The configuration of weak classifiers is obtained after running the program. It takes several hours to train the weak classifiers. The number of weak classifiers is 200. In the training process, 234 positive samples of the traffic light and 468 negative samples of the surroundings are used to tune the parameters of 200 weak classifiers. These samples are extracted in natural scene images. The size of each sample is 30×10, which is the same with the window detector. The configuration of weak classifiers is verified since its high detection accuracy on another 234 positive and negative samples [41].

The configuration of weak classifiers is coded in Verilog on an FPGA platform. The experimental platform is shown in Fig. 8. The FPGA platform is equipped with the camera,

TABLE 3. Summary of resources utilization in the signal processing module and time complexity of the window integral image computation.

	Hiromoto's method [32]	Kyrkou's method [34]	Designed system
Slice LUTs	41038	136238	37116
Slice Registers	28715	203700	33267
F7 Muxes	156	0	0
F8 Muxes	51	0	0
Time complexity	$O(hw)$	$O(2w+h)$	$O(\lceil \log_2 h \rceil + w)$
Latency ($w=10$,$h=30$)	$\geq$ 300 clock cycles	$=50$ clock cycles	≤ 15 clock cycles

w, h represents the width, height of the window respectively.

external memory and LCD. The CMOS camera can capture videos in the RGB mode at the rate of 30 fps. The size of the video graphics array is 640 × 480. Pixels in the array are cached in the external memory. The external memory is based on DDR3 SDRAM, which is controlled by the memory interface generator in the FPGA. The cache is read for the signal processing. The processing result is displayed in the RGB mode on the LCD of size 272 × 480. The display rate is 30 fps. Please note that because the detection rate is limited only by the hardware clock speed of the signal processing module, the designed system is suitable for the detection on high-resolution and high-rate videos [30].

The parallelism optimized architecture is designed in Verilog codes on the FPGA platform. For contrast study, two window integral image calculation methods, Hiromoto's method [32] and Kyrkou's method [34] are implemented in the signal processing module for comparison. The summary of resources utilization in the signal processing module and time complexity of the window integral image computation is listed in Table 3. The calculation of the window integral image in the designed system has the minimal time complexity. The designed system takes 15 clock cycles to calculate the first integral image of window of size 30 × 10, and then it can generate the integral image of the current window per clock cycle. The designed system utilizes obviously fewer look-up-tables (LUTs) than the two earlier methods, due to the parallel architecture by which few memory blocks are needed. The registers utilized by the designed system are dramatically fewer than Kyrkous's method but slightly more than Hiromoto's method. The more consumed registers are mainly used to store the partial sums in the parallel computation. In the implementation of Hiromoto's method, 156 F7 Muxes and 51 F8 Muxes are used to realize several logic commands, which are realized by LUTs in the other two methods [53]. The result of the designed system is in accordance with the two earlier methods, although the architectures of the three methods are different. The parallelism optimized architecture is implemented on the XC7A100TFGG484-2 chip in Artix-7 series of Xilinx. The chip operates at 200 MHZ. The device utilization of summary report is listed in Table 4. The designed system utilizes 65.49%, 60.92% of the available LUT, LUTRAM resources respectively. The devices of the chip are mainly utilized in the signal processing module, which has three line buffers and three integral images. A line buffer consists of 13921 registers of 8 bits, and an integral image consists of 300 registers of 17 bits. In the line buffer, each data occupies 8 bits (1 byte), in order to represent the maximal intensity of 255. Since the maximal sum of these data is 76500 (255 × 30 × 10), the bit depth of the data in the integral image is set to 17. The bit depth is proportional to the size of the window detector.

TABLE 4. Device utilization of the designed system.

Resource	Utilization	Available	Utilization %
LUT	41523	63400	65.49
LUTRAM	11574	19000	60.92
FF	39682	126800	31.29
BRAM	8	135	5.93
IO	132	285	46.32
BUFG	8	32	25.00
MMCM	2	6	33.33
PLL	1	6	16.67

TABLE 5. Summary of indexes of the designed system on embedded platforms.

Frame size	640 × 480
Colour	YCrCb
Features	Multi-layer features
Algorithm	AdaBoost
Platform	FPGA
Clock frequency	200 MHz
Rate	30 fps

The indexes of the designed system are summarized in Table 5. Comparing with the indexes of the existing works, the advantages and novelties of the designed system are summarized as following:

- The designed system realizes FPGA-based design of the traffic light detection, whereas the existing works are implemented on CPU or GPU.
- The designed system adopts the AdaBoost-based machine learning scheme, whereas [6], [7] is based on the filtering scheme.
- The designed system uses multi-layer features which consist of the luminance-layer features and the chrominance-layer features, whereas [9] only adopts the luminance-layer features.
- The designed system results in higher detection rate (30 fps) over other methods.

V. CONCLUSION

In this paper, a parallelism optimized hardware architecture is designed on FPGA for real-time traffic light detection. The practical results show that the designed system can detect the traffic light on videos of 640 × 480 at 30 fps. In addition, the designed system is also theoretically suitable for the detection on videos of high resolution and high rate.

The designed system is efficient, inexpensive, and the hardware architecture is designed in Verilog codes. Therefore, it can be easily transplanted to the other FPGA platforms. In addition, the parallel computations of the

integral image can also speed up the binaryzation using the adaptive thresholding method. The future work may be improving the designed system by introducing more types of weak classifiers, in order to achieve a higher detection rate.

ACKNOWLEDGMENT

The authors want to thank the editors and reviewers for their constructive suggestions.

REFERENCES

[1] M. A. Hersh and M. A. Johnson, Eds., *Assistive Technology for Visually Impaired and Blind People*. London, U.K.: Springer, 2008, pp. 173–174.

[2] A. Almagambetov, S. Velipasalar, and A. Baitassova, "Mobile standards-based traffic light detection in assistive devices for individuals with color-vision deficiency," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1305–1320, Jun. 2015.

[3] M. B. Jensen, M. P. Philipsen, M. Trivedi, T. Møgelmose, and T. Moeslund, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 1800–1815, Jul. 2016.

[4] J. Aranda and P. Mares, *Visual System to Help Blind People to Cross the Street*. Berlin, Germany: Springer, 2004, pp. 454–461.

[5] M. Omachi and S. Omachi, "Traffic light detection with color and edge information," in *Proc. IEEE Int. Conf. Comput. Sci. Inf. Technol.* Beijing, China, Aug. 2009, pp. 284–287.

[6] J. Roters, X. Jiang, and K. Rothaus, "Recognition of traffic lights in live video streams on mobile devices," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 10, pp. 1497–1511, Oct. 2011.

[7] S. Mascetti, D. Ahmetovic, A. Gerino, C. Bernareggi, M. Busso, and A. Rizzi, "Robust traffic lights detection on mobile devices for pedestrians with visual impairment," *Comput. Vis. Image Understand.*, vol. 148, pp. 123–135, Jul. 2016.

[8] M. P. Philipsen, M. B. Jensen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Traffic light detection: A learning algorithm and evaluations on challenging dataset," in *Proc. 18th IEEE Intell. Transp. Syst. Conf.*, Sep. 2015, pp. 2341–2345.

[9] W. Liu, S. Li, J. Lv, B. Yu, T. Zhou, H. Yuan, and H. Zhao, "Real-time traffic light recognition based on smartphone platforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 5, pp. 1118–1131, May 2017.

[10] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 690–700, Mar. 2016.

[11] X.-H. Wu, R. J. Hu, and Y.-Q. Bao, "Fast vision-based pedestrian traffic light detection," in *Proc. IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, Miami, FL, USA, Apr. 2018, pp. 214–215.

[12] X. Li, H. Ma, X. Wang, and X. Zhang, "Traffic light recognition for complex scene with fusion detections," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 199–208, Jan. 2018.

[13] J.-G. Wang and L.-B. Zhou, "Traffic light recognition with high dynamic range imaging and deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1341–1352, Apr. 2019.

[14] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[15] X. Wen, L. Shao, W. Fang, and Y. Xue, "Efficient feature selection and classification for vehicle detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 3, pp. 508–517, Mar. 2015.

[16] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Syst., Man, C, Appl. Rev.*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[17] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Sep. 2006.

[18] K. Xu, Y. Zheng, F. Zhang, Z. Jiang, Y. Qi, H. Chen, and J. Zhu, "An energy efficient adaboost cascade method for long-term seizure detection in portable neurostimulators," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 11, pp. 2274–2283, Nov. 2019, doi: 10.1109/TNSRE.2019.2947426.

[19] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*. Cham, Switzerland: Springer, 2014, pp. 818–833.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[21] S. I. Venieris, A. Kouris, and C.-S. Bouganis, "Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–39, 2018.

[22] G. Lacey, G. W. Taylor, and S. Areibi, "Deep learning on FPGAs: Past, present, and future," Feb. 2016, *arXiv:1602.04283*. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2016arXiv160204283L

[23] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, 2002.

[24] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, New York, NY, USA, 2015, pp. 161–170.

[25] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays* New York, NY, USA, 2017, pp. 45–54.

[26] M. Peemen, A. A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Asheville, NC, USA, Oct. 2013, pp. 13–19.

[27] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[28] F. Wang, Z. Li, F. He, R. Wang, W. Yu, and F. Nie, "Feature learning viewpoint of adaboost and a new algorithm," *IEEE Access*, vol. 7, pp. 149890–149899, 2019.

[29] Y. Wei, X. Bing, and C. Chareonsak, "FPGA implementation of AdaBoost algorithm for detection of face biometrics," in *Proc. IEEE Int. Workshop Biomed. Circuits Syst.*, Singapore, Dec. 2004, pp. S1/6–17.

[30] H.-C. Lai, M. Savvides, and T. Chen, "Proposed FPGA hardware architecture for high frame rate (≫ 100 fps) face detection using feature cascade classifiers," in *Proc. 1st IEEE Int. Conf. Biometrics, Theory, Appl., Syst.*, Crystal City, VA, USA, Sep. 2007, pp. 1–6.

[31] J. Cho, B. Benson, S. Mirzaei, and R. Kastner, "Parallelized architecture of multiple classifiers for face detection," in *Proc. 20th IEEE Int. Conf. Appl.-Specific Syst., Architectures Processors*, Boston, MA, USA, Jul. 2009, pp. 75–82.

[32] M. Hiromoto, H. Sugano, and R. Miyamoto, "Partially parallel architecture for AdaBoost-based detection with Haar-like features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 1, pp. 41–52, Jan. 2009.

[33] M. Yang, J. Crenshaw, B. Augustine, R. Mareachen, and Y. Wu, "AdaBoost-based face detection for embedded systems," *Comput. Vis. Image Understand.*, vol. 114, no. 11, pp. 1116–1125, 2010.

[34] C. Kyrkou and T. Theocharides, "A flexible parallel hardware architecture for AdaBoost-based real-time object detection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 6, pp. 1034–1047, Jun. 2011.

[35] W. Zhou, H. Wu, and X. Zeng, "A low cost architecture for high performance face detection," *Microprocessors Microsyst.*, vol. 39, no. 6, pp. 339–347, 2015.

[36] Z. Xu, R. Shi, Z. Sun, Y. Li, Y. Zhao, and C. Wu, "A heterogeneous system for real-time detection with AdaBoost," in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun., IEEE 14th Int. Conf. Smart City, IEEE 2nd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Sydney, NSW, Australia, Dec. 2016, pp. 839–843.

[37] S. Jin, D. Kim, T. T. Nguyen, D. Kim, M. Kim, and J. W. Jeon, "Design and implementation of a pipelined datapath for high-speed face detection using FPGA," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 158–167, Feb. 2012.

[38] S.-S. Lee, S.-J. Jang, J. Kim, and B. Choi, "A hardware architecture of face detection for human-robot interaction and its implementation," in *Proc. IEEE Int. Conf. Consum. Electron.-Asia (ICCEAsia)*, Seoul South Korea, Oct. 2016, pp. 1–2.

[39] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Proc. Int. Conf. Comput. Learn. Theory*, 1995, pp. 23–37.

[40] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 1997, pp. 193–199.

[41] X.-H. Wu, R. Hu, and Y.-Q. Bao, "Pedestrian traffic light detection in complex scene using adaboost with multi-layer features," *J. Eng. Res.*, vol. 6, no. 3, pp. 34–53, Sep. 2018.

[42] S. V. Dharan, M. Khalil-Hani, and N. Shaikh-Husin, "Hardware acceleration of a face detection system on FPGA," in *Proc. IEEE Student Conf. Res. Develop. (SCOReD),* Kuala Lumpur, Malaysia, Dec. 2015, pp. 283–288.

[43] Recommendation ITU-R BT.6017. 2011. Studio Encoding Parameters of Digital Television for Standard vol. 4, p. 3and Wide-screen 16:9 Aspect Ratios. http://www.itu.int/rec/R-REC-BT.601-7-201103-I/en
[44] B. Ahirwal, M. Khadtare, and R. Mehta, "FPGA based system for color space transformation RGB to YIQ and YCbCr," in *Proc. Int. Conf. Intell. Adv.Syst.*, Kuala Lumpur, Malaysia, Nov. 2007, pp. 1345–1349.
[45] X. Zhang, X. Li, W. Yang, and R. Li, "FPGA-based color space conversion system design and implementation," in *Proc. IEEE 7th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, New York, NY, USA, Oct. 2016, pp. 1–4.
[46] P. Ouyang, S. Yin, Y. Zhang, L. Liu, and S. Wei, "A fast integral image computing hardware architecture with high power and area efficiency," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 1, pp. 75–79, Jan. 2015.
[47] D. Puchala and K. Stokfiszewski, "GPU accelerated image binarization based on first and second order integral images," in *Proc. IEEE 13th Int. Sci. Tech. Conf. Comput. Sci. Inf. Technol. (CSIT)*, Lviv, Ukraine, Sep. 2018, pp. 404–407.
[48] F. Spagnolo, P. Corsonello, and S. Perri, "Efficient architecture for integral image computation on heterogeneous FPGAs," in *Proc. 15th Conf. Ph.D Res. Microelectron. Electron. (PRIME)*, Lausanne, Switzerland, Jul. 2019, pp. 229–232.
[49] R. L. Ladner and M. J. Fisher, "Parallel prefix computations," *J. Assoc. Comput. Mach.*, vol. 27, no. 4, pp. 831–838, 1980.
[50] G. Blelloch, "Prefix sums and their applications," in *Synthesis of Parallel Algorithms*, 1st ed. San Francisco, CA, USA: Morgan Kauffman, 1993, ch. 1, pp. 35–60.
[51] K. Abdelouahab, M. Pelcat, and F. Berry, "The challenge of multi-operand adders in CNNs on FPGAs: How not to solve it!" in *Proc. 18th Int. Conf. Embedded Comput. Syst., Architectures, Modeling, Simulation*, Pythagorion, Greece, 2018, pp. 157–160.
[52] A. Vezhnevets and V. Vezhnevets, "Modest AdaBoost-teaching adaBoost to generalize better," *Graphicon*, vol. 12, no. 5, pp. 987–997, 2005.
[53] *UltraFast Design Methodology Guide for the Vivado Design Suite*. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug949-vivado-design-methodology.pdf

XUE-HUA WU received the B.S. degree in electrical engineering and the M.S. degree in power system and its automation from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2009 and 2012, respectively. She is currently pursuing the Ph.D. degree in electrical engineering with Southeast University (SEU), Nanjing. Her research interests include image processing, machine learning, and object detection.

RENJIE HU received the B.S., M.S., and Ph.D. degrees in electrical engineering from Southeast University (SEU), Nanjing, China, in 1985, 1994, and 2002, respectively.

He is currently a Professor with Southeast University and serves as the Chief of the Electrical and Electronic Experiment Center. His research interests include electrical detection, intelligent instrument, and power electronics.

YU-QING BAO was born in Zhenjiang, China, in 1987. He received the Ph.D. degree from Southeast University (SEU), Nanjing, China, in March 2016.

He is currently an Associate Professor with Nanjing Normal University (NJNU). His research interests include smart city, intelligent instrument, power system operation and scheduling, power demand side management, and the frequency control of the power systems.