ABSTRACT - This paper proposes the hardware architecture of face detection FPGA hardware system using the AdaBoost algorithm. The proposed structure of face detection hardware system is possible to work in 30 frames per second and in real time. And the AdaBoost algorithm is adopted to learn and generate the characteristics of the face data by MATLAB, and finally detected the face using this data. This paper describes the face detection hardware structure composed of image scaler, integral image extraction, face comparing, memory interface, data grouper and detected result display. The proposed circuit is so designed to process one point in one cycle that the proposed design can process full HD (1920x1080) image at 70MHz, which is approximate 2316087 x 30 cycle.

General Terms: Pattern Recognition, Security, Algorithms et. al.

Keywords: FPGA, Adaboost, Real time, face detection

1. INTRODUCTION

Recently, research on biometric technologies that can perform personal authentication, information protection, and identity verification using the human body such as face, fingerprint, and iris has been actively conducted. Object recognition in video sequence or images is an important field in computer vision, image processing applications, security, bioinformatics, and artificial intelligence.

Object detection involves extracting information from an image (or a sequence of) frames, processing the information, and locating specific objects and images in the information. This process is computationally intensive, and there have been several attempts to implement object detection algorithms in hardware, especially in embedded and real-time systems [1-6]. Most of the proposed works are designed for FPGA target implementations. Until now, many hardware design methods of adaboost algorithm using FPGA have been proposed. The method of proposing the face recognition hardware using the Adaboost method includes [7-13].

In this paper, face recognition system using FPGA board is proposed based on the Adaboost algorithm by Viola and Jones [14]. For the implementation of hardware, HAAR-like feature used for face detection, an integral image designed for speed improvement, and a method of boosting from a weak classifier to a strong classifier were used. Through comparison with the existing hardware structure [7][9][13], the advantages of this hardware structure will be verified.

2. THE TRAINING USING ADABOOST ALGORITHM

In this paper, we used the HAAR-like feature that detects and compares face regions in images using Viola and Jones's Adaboost algorithm. Figure 1 shows the feature vectors obtained through learning the Adaboost algorithm of Viola and Jones and the feature vectors used in this paper. The Adaboost algorithm needs to generate feature vectors for the pre-processing process. As shown in figure 1, for a 20x20 image, 17,100 vectors were created for Type 0 and Type 1 patterns, 7600 vectors were created for Type2 and Type3 patterns, and 8100 vectors were created for Type4. For training, training was performed using MATLAB®, and the MIT CBCL Face Data Set was used. The Face data set consists of 6,977 cropped images (2,429 faces and 4,548 non-faces).

The total number of feature vectors used in this paper is 57,500, of which only 200 feature vectors are used for face recognition.

The weights of the classifier are modified while repeating step by step. At each step, learning proceeds in such a way that only the features that play a decisive role in detecting faces from among the feature sets are left and the rest are removed [16]. Therefore, this module is designed to receive the data output from the integral image processing process and directly compare the current data and the trained data value to determine whether it is a human face or not.
3. HARDWARE ARCHITECTURE

With respect to the YCbCr signal of the image input from the camera, the VIM module detects only the Y signal, which is a black and white signal, and stores it in the first memory (org_sram) to input data to the module for face recognition.

Figure 2 shows the process of performing the image size reduction section and integral image extraction section on the stored data, and outputting the facial region area on the LCD screen of the Vertex5 LX330 board through the Adaboost classifier.

In this paper, since the image size is reduced to 0.75 times that of the previous image, the reduction process is repeated several times depending on the size of the screen to detect the face area.

In order to apply the Adaboost algorithm, it is necessary to calculate the integral image. Therefore, in the data flow, the image reduction process and the integral image extraction process are simultaneously performed on the first frame start signal, and the face region extraction, image reduction and integral image extraction processes are repeatedly performed from the second frame start signal. Since the pipeline structure was applied, time wasted in the data processing process was minimized and an efficient hardware structure was constructed. Figure 3 shows the pipeline flow diagram of hardware for face recognition.

3.1 Integral image calculation module

Many image processing filters perform window operations, and as the window increases, the time complexity increases accordingly. One of the methods to reduce the time complexity is the integral image processing method. When the average filter is used in the integral image processing, a window of a certain size is set in each pixel, and a new pixel value is set by taking the average value of the pixels in the window. If the integral image processing method is used, the sum of the pixel values for a rectangle of any size can be calculated within a certain time regardless of the size.

In addition, the process for detecting a specific object in the image has the advantage that it can quickly obtain the Haar-like feature value in the next pipeline after converting the data value of the original image into an integral image. Figure 4 shows the data flow of the integral image processing designed in this paper and the operation to obtain the pixel value of region D.

3.2 Image resizing

In the image resizing process, as shown in Figure 4, the input image data is resized in units of 4 pixels. The first pixel value to be resized is the sum of the first pixel value before resizing multiplied by 3/4 and the second pixel value multiplied by 1/4. The second pixel value to be resized is multiplied by 1/2 and added to the second and third pixel values before resizing.

The third pixel value to be resized was used as the sum by multiplying the third pixel value by 1/4 and multiplying the fourth pixel value by 3/4. A 4x4 image was reduced to 3x3 by performing a resizing process for each pixel value of the image. As shown in Figure 5, if the size of the original image is 480x272, the image after the image resizing process has a size of 360x204.
Table 1. Cycle calculation for 480x272 image

| Step | Width-20/2 | Y inspection point | First row processing cycle(s) | Row change Cycle (B) | total cycle |
|------|------------|--------------------|-----------------------------|----------------------|------------|
| 1    | 1920*1080  | 950                | 108070                      | 100530               | 3033150    |
| 2    | 1440*810   | 710                | 13900                       | 15094                | 572590     |
| 3    | 1080*607   | 530                | 10070                       | 130580               | 320650     |
| 4    | 810*455    | 395                | 7505                        | 171345               | 1789365    |
| 5    | 607*341    | 294                | 5386                        | 94080                | 996665     |
| 6    | 455*255    | 218                | 4142                        | 51012                | 591556     |
| 7    | 343*191    | 161                | 27370                       | 80429                | 777649     |
| 8    | 255*141    | 118                | 2143                        | 14396                | 166516     |
| 9    | 191*107    | 84                 | 1634                        | 7986                 | 8030       |
| 10   | 143*80     | 62                 | 1178                        | 1596                 | 4774       |
| 11   | 107*60     | 44                 | 836                         | 1672                 | 2908       |
| 12   | 80*45      | 30                 | 570                         | 720                  | 1290       |
| 13   | 60*31      | 20                 | 380                         | 240                  | 620        |
| 14   | 45*24      | 13                 | 247                         | 36                   | 273        |
| Sum of Cycles/frame | 14400 | 185204 | 532666 |

Table 2. Cycle calculation for 1920x1080 image

| Step | Width-20/2 | Y inspection point | First row processing cycle(s) | Row change Cycle (B) | total cycle |
|------|------------|--------------------|-----------------------------|----------------------|------------|
| 1    | 1920*1080  | 950                | 108070                      | 100530               | 3033150    |
| 2    | 1440*810   | 710                | 13900                       | 15094                | 572590     |
| 3    | 1080*607   | 530                | 10070                       | 130580               | 320650     |
| 4    | 810*455    | 395                | 7505                        | 171345               | 1789365    |
| 5    | 607*341    | 294                | 5386                        | 94080                | 996665     |
| 6    | 455*255    | 218                | 4142                        | 51012                | 591556     |
| 7    | 343*191    | 161                | 27370                       | 80429                | 777649     |
| 8    | 255*141    | 118                | 2143                        | 14396                | 166516     |
| 9    | 191*107    | 84                 | 1634                        | 7986                 | 8030       |
| 10   | 143*80     | 62                 | 1178                        | 1596                 | 4774       |
| 11   | 107*60     | 44                 | 836                         | 1672                 | 2908       |
| 12   | 80*45      | 30                 | 570                         | 720                  | 1290       |
| 13   | 60*31      | 20                 | 380                         | 240                  | 620        |
| 14   | 45*24      | 13                 | 247                         | 36                   | 273        |
| Sum of Cycles/frame | 68080 | 2247088 | 2316087 |

In Table 2, in the same way as Table 1, the number of cycles for full HD is $2,316,087 \times 30 = 69,482,610$, indicating that it operates at approximately 70MHz and real-time processing is possible.

3.4 Face detection module

The face detection method using the Adaboost algorithm is learned using a feature set extracted from a learning image set composed of face and non-face images. The learning process is performed through repeated calculations, and consists of the steps of feature selection, error rate calculation corresponding to the observed value, and weight updating.

The classification ability is strengthened at each iteration because weak classifiers are combined to form a strong classifier through stages.
By repeating the process of increasing the weight of the misclassified training image and decreasing the weight of the correctly classified training image, the HAAR-like feature with the minimum error rate is selected.

\[
h_n(x) = \begin{cases} 
1 & \alpha_1 h_1(x) + \ldots + \alpha_n h_n(x) \geq \frac{1}{2} (\alpha_1 + \ldots + \alpha_n) \\
0 & \text{otherwise}
\end{cases}
\]  

Equation 3.1 is a strong classifier, meaning that it is classified by linearly combining n features. The weight of the classifier is modified while repeating step by step. At each step, learning proceeds in such a way that only the features that play a decisive role in detecting faces from among the feature sets are left and the rest are removed.

\[
\alpha_i = \log \left( \frac{1}{\beta_i} \right), \beta_i = \frac{\varepsilon_i}{1-\varepsilon_i}
\]

3.5 Reducing bit of integral image

In Figure 8, as the critical path of the adder tree used in the face region detection module using Equation (3.1) increases, F/F is used to reduce the critical path of the adder tree. Through this, it was possible to reduce the delay of the critical path used in the face region detection module to 5ns.

3.6 Removing Overlap

The Overlap module is designed to make a simple expression by treating two or more rectangles as one rectangle when they are located inside the same position, and is performed as shown in Figure 9. The position of the inner rectangle is found with the x-position and y-position values output from the face recognition controller module. In Figure 9, in order to know the inclusion relationship of rectangular boxes such as (1), (2), (3), it is designed to store up to 20 coordinates output from the face recognition controller and find the inner rectangle.

The result obtained by applying the overlap removal module is output on the screen of the target board, and Figure 10 shows the actual hardware output screen.

4. EXPERIMENTAL RESULTS

This paper verified hardware by using MODELSIM and comparing the results from MATLAB, to the output data values of each module, the integral image part, the image reduction part, and the last face region detection part as shown in Figure 11 using MODELSIM.

Table 3 shows the results of comparative analysis of this paper with other systems. In this paper, a 480x272 (VGA) image and a clock frequency of 24 MHz were used for test. Other studies aim to process 360x240 images and show low frame rate performance. Compared to other researches, it was confirmed that it operates at a high frame rate and low frequency clock for a large screen size.
In this paper, we designed a face recognition hardware system for DVR system that detects a human face from an image and outputs it on the screen. The hardware designed by creating a reference vector using MATLAB was verified using MODELSIM, and the synthesis was performed by FPGA verification using Xilinx ISE Design Suite and Synopsys’ Design Compiler in 0.18um process, 35% of Vertex5 LX330 Slice LUT 74,757 was used. It is designed to operate at an operating frequency of 24MHz, so it is possible to process 112 frames per second for VGA-level images and 30 frames per second in the case of 70MHz clock system for full HD images.

6. ACKNOWLEDGMENTS
This research is based on the support of 2021 Woosong University Academic Research Funding.

REFERENCES
[1] Jeong-Won Ryu, Jae-Hong Lee. (2021). "An Efficient Vehicle License Plate Recognition System Based on Embedded Systems", Journal of Next-generation Convergence Technology Association, Vol.5, No.1, pp. 22-27, doi: 10.33097/JNICTA.2021.05.01.22

[2] Young-Jung Yu, Sang-Ho Moon, Sang-Jin Sim, Seong-Ho Park. (2020). "Recognition of License Plate Number for Web Camera Input using Deep Learning Technique", Journal of Next-generation Convergence Technology Association, Vol.4, No.6, pp. 565-572, doi: 10.33097/JNICTA.2020.04.06.565

[3] P. Peng, Y. Zhang, Y. Wu and H. Zhang. (2018). "An Effective Fault Diagnosis Approach Based On Gentle AdaBoost and AdaBoost.MH", 2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEM), pp. 8-12, doi: 10.1109/AUTEEM.2018.8720764

[4] Se Hun Park, Yeong Geon Seo. (2021). "Hand Area Detection based on Color using Cheap Webcam", Journal of Next-generation Convergence Technology Association, Vol.5, No.1, pp. 5-13, doi:10.33097/JNICTA.2021.05.01.5

[5] S. Saurav, S. Saini and S. Singh. (2018). "FPGA Based Implementation of Linear SVM for Facial Expression Classification", 2018 International Conference on Advances in Computing, Communications and Informatics, pp. 766-773, doi: 10.1109/ICACCI.2018.8554645.

[6] Sarabpreet Kaur, Jyoti Patel (2018), A Robust Image Mosaicing Technique Using Frequency Domain. IJEER 6(1), 1-8. DOI:10.37391/ijeer.060101. http://ijeer.forexjournal.co.in/archive/volume-6/ijeer-060101.php

[7] S. Wu and H. Nagahashi. (2014). "Parameterized AdaBoost: Introducing a Parameter to Speed Up the Training of Real AdaBoost", in IEEE Signal Processing Letters, vol. 21, no. 6, pp. 687-691, doi: 10.1109/LSP.2014.2313570

[8] C. Kyrkou and T. Theocharides. (2011), "A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection", in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 19, no. 6, pp. 1034-1047, doi: 10.1109/TVLSI.2010.2048224.

[9] Felix Noel Sitorus; Yustina Manihuruk; Good Fried Panggabean. (2019). "Implementation of AdaBoost Algorithm for Object Detection Using FPGA", International Conference of Computer Science and Information Technology, DOI:10.1109/ICSNIKOM4875.2019.9111611

[10] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner. (2009). "FPGA-Based Face Detection System Using Haar Classifiers", FPGA 09: Proceedings of the ACM/SIGDA International symposium on Field programmable gate arrays, Feb. Pages 103-112 doi:10.1145/1508128.1508144

[11] M. Ibarra-Manzano and D. Almanza-Ojeda. (2011). "Design and Optimization of Real-Time Boosting for Image Interpretation Based on FPGA Architecture", 2011 IEEE Electronics, Robotics and Automotive Mechanics Conference, pp. 167-172, doi: 10.1109/CERMA.2011.33.

[12] S. V. Chakrasiil and S. Kuthale. (2016). "Optimized face detection on FPGA, International Conference on Circuits, Controls", Communications and Computing, pp. 1-6, doi: 10.1109/CICMCA.2016.8053269.

[13] Aatray Kumar Singh, Ashish Nanaware, Ashish Nanaware (2016), Vehicle Simulator: Virtual Trip Runner (VTR)/ Real Time Recorder (RTR). IJEER 4(1), 33-36. http://ijeer.forexjournal.co.in/archive/volume-4/ijeer-040107.php

[14] Mitali, Deepak Bhati (2016), Review Paper on Placement Algorithms. IJEER 4(2), 49-52. DOI: 10.37391/JNCTA.2016.04.0202. http://ijeer.forexjournal.co.in/archive/volume-4/ijeer-040202.php

[15] Yu Wei, Xiong Bing, and Charayaphay Charoenakon. (2004). "FPGA Implementation of AdaBoost Algorithm for Detection of Face Biometrics", Biomedical Circuits and Systems, 2004 IEEE International Workshop on, DOI: 10.1109/BIOCAS.2004.1454161

[16] K. -C. Chang and C.-P. Fan. (2019). "Cost-Efficient AdaBoost-based Face Detection with FPGA Hardware Accelerator," 2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), 2019, pp. 1-2, doi: 10.1109/ICCE-TW.46550.2019.8991862.

[17] Viola, P., Jones, M.J. Robust. (2004). "Real-Time Face Detection" International Journal of Computer Vision,57, 137–54. DOI:10.1023/B:VISI.0000013087.49260.fh

[18] Su-Ilun LEE, Yong-jin JEONG (2014). "A New Integral Image Structure for Memory Size Reduction", IEICE TRANSACTIONS on Information and Systems Vol.97-D, No.4, pp.998-1000, DOI: 10.1587/transinf.97.D.998

[19] Venkata Naresh Mandhala, Debnath Bhattacharyya, Tai-hoon Kim. (2016). "Face Detection using Image Morphology – A Review", International Journal of Security and Its Applications, NADIA, ISSN: 1738-9976 (Print); 2207-9629 (Online), vol.10, no.4, pp. 89-94, http://dx.doi.org/10.14257/ijisia.2016.10.4.10.

[20] Pradeep S (2014), Design and FPGA Implementation of Image Compression Based Fuzzy Technique. IJEER 2(2), 1-4. 10.37391/IJEER.020201 http://ijeer.forexjournal.co.in/archive/volume-2/ijeer-020201.php