JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

REAL TIME FACE DETECTION USING FPGA

[1]SUJATHA H, [2]PALLAVI B
[1]Lecturer, [2] Lecturer
[1]Department of Electronics and Communication Engineering, [2]Department of Computer Science and Engineering
[1] Government Polytechnic Nagamangala, Mandya, India, [2]Government Polytechnic Nagamangala, Mandya, India

Abstract: The main aim of our work is to design and implementation of a physically feasible hardware system to accelerate the processing speed of the operations required for real time face detection. Therefore, it has resulted in the development of a real-time face detection system employing an FPGA implemented system designed by Verilog HDL.In this project we present hardware architecture for real-time face detection system. We propose hardware design techniques to accelerate the processing speed of face detection. The face detection system generates an integral image window to perform a Haar feature classification during one clock cycle. And then it performs classification operations in parallel using Haar classifiers to detect a face in the image sequence.

***Index Terms* –FACE DETECTION, FPGA, GPIO, PUSHBUTTONS**

I. INTRODUCTION

Computer vision is one of the fields which have seen increasing number of applications in the recent years in various domains like biomedical imaging, surveillance systems, interactive systems like gesture recognition, gaming etc. Detection of human faces is one of the key elements in the applications of computer vision in the above-mentioned domains. Implementation of this face detection algorithm has been achieved through various methods like genetic programming, neural network-based methods, skin detection, color and texture detection etc. However, these methods break down easily because of the complexity of the real world. Genetic programming and neural-network based methods involve loads of computations and are hence better suited for software implementations [1].

Skin, color and texture detection methods rely heavily on the lighting conditions and hence do not yield good results under poor lighting conditions. The method proposed by Viola and Jones is probably the most popular and the most robust mechanism for real-time face detection in hardware. It employs an "integral image" representation to detect features quickly and efficiently. With the help of a learning algorithm based on AdaBoost which creates a strong classifier using a set of weak classifiers and a cascade of these classifiers it quickly eliminates non-face portions of the image [2].

Real-time face detection has become an increasingly crucial application in various fields, ranging from surveillance and security to human-computer interaction and augmented reality. To achieve real-time performance and high accuracy, many researchers and developers are turning to Field-Programmable Gate Arrays (FPGAs) as a hardware platform. FPGAs offer the advantage of hardware-level parallelism and configurability, making them well-suited for computationally intensive tasks like face detection. It explores the development of a highly efficient and low-latency system for detecting faces in real-time video streams. By leveraging the FPGA's reconfigurable nature, we can design custom hardware accelerators that optimize face detection algorithms, such as Haar cascades or deep learning-based approaches like Single Shot Multibox Detector (SSD) or You Only Look Once (YOLO)[3].

The proposed system intends to surpass the performance limitations of traditional software-based face detection methods, enabling real-time processing of high-definition video feeds at frame rates of 30 frames per second or higher. Furthermore, by utilizing FPGA's power efficiency, we can build energy-efficient face detection solutions for embedded devices or resource-constrained environments, where traditional CPU or GPU-based solutions may not be viable due to their high-power consumption.

In this project, we will explore the implementation of the face detection algorithm on an FPGA platform, considering factors like memory management, pipelining, and parallel processing techniques to achieve maximum throughput. Additionally, we will investigate how to interface the FPGA with cameras and displays to create a complete real-time face detection system. By the end of this research, we aim to demonstrate a robust and high-performance FPGA-based face detection system that can find applications in a wide range of domains, including security, robotics, and human-computer interaction.

II. RELATED WORK

Almost all of the available literatures on real-time face detection are theoretical or describe a software implementation. Only a few have addressed hardware design and implementation of real time face detection [4]. Theocharides presented the implementation of neural network-based face detection in an ASIC to accelerate processing speed. However, VLSI technology requires a large amount of development time and cost. Also, it is difficult to change design [5]. McCready designed and implemented face detection for the Transmogrifier-2 configurable hardware system. This implementation utilized nine FPGA boards [6]. Sadri implemented neural network-based face detection on the Virtex-II Pro FPGA [7]. Skin color filtering and edge detection are used to reduce the processing time. However, some operations are implemented on hardcore PowerPC processor with embedded software.

Wei et presented FPGA implementation for face detection using scaling input images and fixed-point expressions. However, the image size is too small (120×120 pixels) to be practical and only some parts of classifier cascade are actually implemented. A low-cost detection system was implemented using Cyclone II FPGA by Yang [8]. The frame rate of this system is 13 fps with low detection rate of about 75%. Nair implemented an embedded system for human detection on an FPGA. It can process the images at speeds of 2.5 fps with about 300 pixels images. Gao et al. presented an approach to use an FPGA to accelerate the Haar feature classifier-based face detection. They re-trained the Haar classifier with 16 classifiers per stage [9]. However, only classifiers are implemented in the FPGA. The integral image generation and detected face display are processed in a host microprocessor. Also, the largest Virtex-5 FPGA was used for the implementation because the design size is too large.

Hiromoto et al implemented real-time object detection based on the AdaBoost algorithm. They proposed hybrid architecture of a parallel processing module for the former stages and a sequential processing module for the subsequent stages in the cascade. Since the parallel processing module and the sequential processing module are divided after evaluating a processing time with fixed Haar feature data, it should be designed and implemented again in order to apply new Haar feature data.

III. PROBLEM STATEMENT

The problem statement for real-time face detection using FPGA is to design and implement an efficient and low-latency face detection system that can accurately detect human faces in real-time video streams. The goal is to surpass the limitations of traditional software-based approaches and leverage the hardware parallelism and reconfigurability of FPGAs to achieve high-performance face detection. The system must be capable of processing high-definition video feeds at frame rates of 30 frames per second or higher while maintaining a high level of accuracy in detecting faces under varying environmental conditions, lighting conditions, and facial orientations. It should be optimized for power efficiency to enable deployment in resource-constrained environments and embedded devices. The FPGA-based face detection system should employ a suitable face detection algorithm, such as Haar cascades, Single Shot Multibox Detector (SSD), or You Only Look Once (YOLO), and tailor it for hardware implementation to maximize throughput and minimize latency. Efficient memory management and data flow strategies should be employed to handle the limited on-chip memory resources and ensure smooth processing of video frames. Furthermore, the system needs to be integrated seamlessly with the camera module to capture real-time video frames and the display module to present the detected faces or processed frames. The performance of the FPGA-based system will be evaluated in terms of accuracy, speed, and power efficiency, and comparisons will be made with software-based approaches to demonstrate the advantages of using FPGA for real-time face detection.

IV. PROPOSED METHODOLOGY

The proposed methodology for real-time face detection using FPGA involves a multi-step approach that harnesses the FPGA's hardware parallelism and reconfigurable nature to achieve efficient and low-latency face detection. The first step is to choose an appropriate face detection algorithm that suits the FPGA architecture. Algorithms like Haar cascades, Single Shot Multibox Detector (SSD), or You Only Look Once (YOLO) are popular choices. Once the algorithm is selected, we need to optimize it for hardware implementation, considering factors like data representation, memory requirements, and computation complexity. The core of the methodology involves designing custom hardware accelerators for the selected face detection algorithm. These accelerators are tailored to leverage the FPGA's parallel processing capabilities efficiently. The algorithm's critical components, such as convolutional layers or feature extraction stages, are mapped to dedicated hardware modules to enable simultaneous processing of multiple data streams in parallel. FPGA resources and memory are limited, so an efficient memory management scheme is crucial. We need to design optimized data flow to minimize data transfer and maximize on-chip memory utilization. Techniques like data buffering, caching, and pipelining will be employed to minimize latency and achieve real-time performance. To complete the real-time face detection system, the FPGA must be interfaced with the camera module to receive video frames and the display module to output the processed frames. FPGA's I/O capabilities will be used to handle these interfaces efficiently, ensuring a seamless integration of the FPGA-based face detection system with the overall video processing pipeline. Finally, the proposed methodology will be tested and evaluated extensively to measure its accuracy, speed, and power efficiency. We will compare the FPGA-based implementation with software-based approaches to highlight the advantages of using FPGA for real-time face detection. Performance benchmarks will be established to demonstrate the system's capabilities in handling different video resolutions and frame rates.

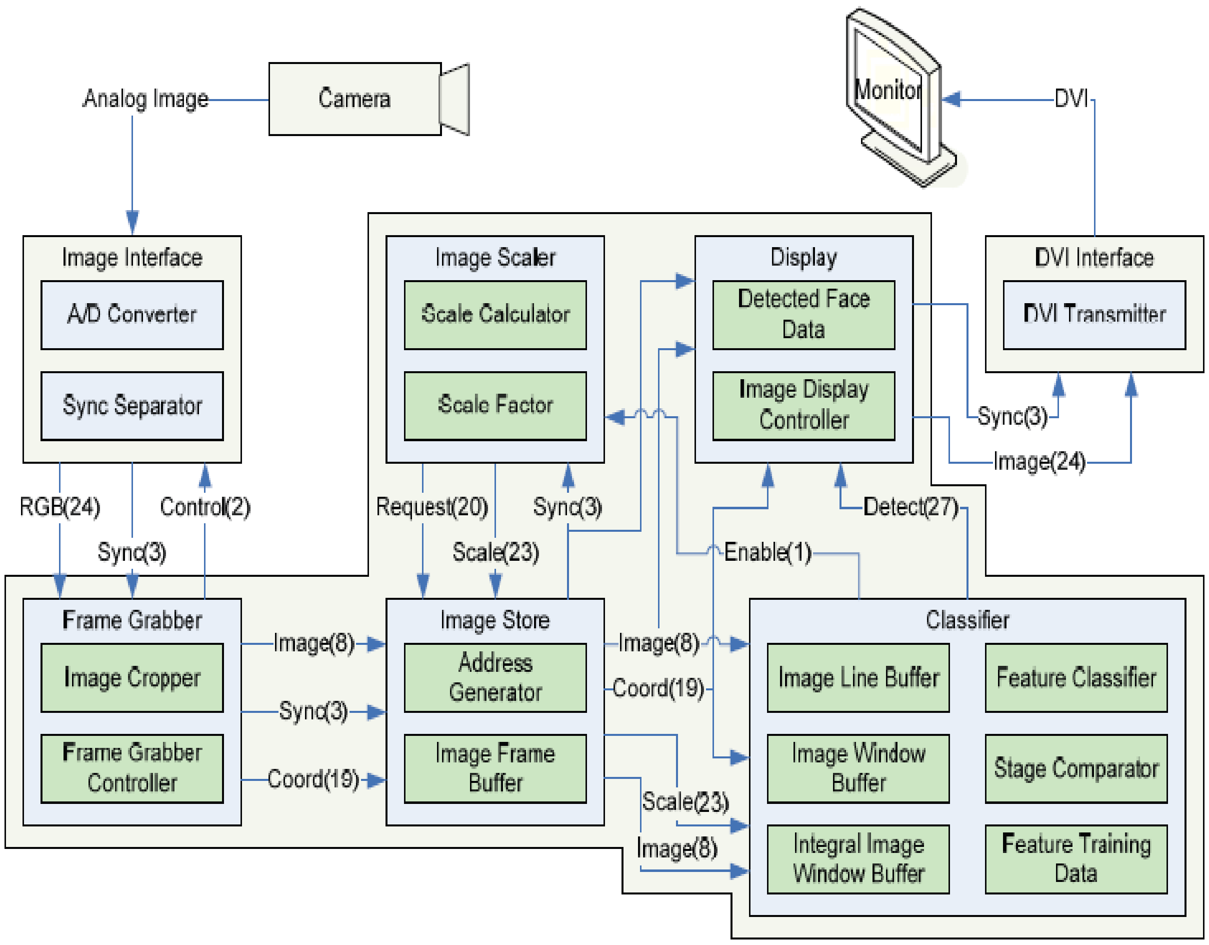


Figure 1. Block diagram of proposed face detection system.

The image interface and DVI interface are implemented using ASIC custom chips with the FPGA board. The others are designed using Verilog HDL and implemented in an FPGA in order to perform face detection in real-time. The input to the system is taken either from the webcam or we can give the input as a video format. It accepts the data from the Camera output which is in the Analog form. It converts the DVI format into the respective RGB components each of the 8 bits respectively. It also produces the Horizontal and Vertical component along with Reset and clock and sync output. It is the ASIC using AD9980.

The Frame Grabber module consists of two sub modules namely. The Frame Grabber Controller generates control signals required to control the A/D converter and the sync separator sub-modules within the Image Interface module. The Frame Grabber Controller uses a Pico Blaze processor to generate these control signals. The image cropper crops the images based on the sync signals. These image data and sync signals are also used by the other modules in the design. The Image Store module stores the incoming images in the Image Frame Buffer. Block RAMs are used within the Image Frame Buffer to store the images frame by frame. The Address Generator generates the appropriate address for storing and accessing the stored images. Based on the current value of the scale factor, the image store module forwards the image stored to the classifier module on a pixel basis. The image scaler module scales down the image according to the scale factor used during the training. Scaling down the images helps to detect faces of different sizes. Whenever an image pixel value is requested by the classifier module, the image scaler passes the request along with the scale factor value to the image store module. The image store module uses this information to provide the appropriate pixel data to the classifier module. The classifier module is the heart of the face detection system where most of the computation for the face detection occurs.

Feature classifier module keeps track of the classifier to be run on the current image window. The Stage Comparator sub-module compares the output of each stage with the threshold value obtained during training to determine whether the current stage is a pass or a failure. The data obtained during training and needed for face detection is stored in the Feature Training Data module. The architecture used for the generation of the integral image of a window is as shown in Figure 1. The Classifier module generates a request to the Scaler module for a new pixel of data when it is ready to accept the new pixel. The Classifier module generates a request to the Scaler module for a new pixel of data when it is ready to accept the new pixel. The Image Line Buffer stores the values of each incoming pixel in dual port Block RAMs. The X co-ordinate of the pixel is used as the address to store the pixel value in the appropriate line buffer. When a new pixel arrives, all the previously stored values are shifted up as shown. The data in the Integral Image Window Buffer is then used to calculate the area of the rectangles in the Haar features. Area calculation requires four buffer accesses corresponding to the four vertices of the rectangle. After running a particular feature on an image, the computed value is compared against the feature threshold. If the computed value is higher than the feature threshold, there is more likelihood of a face being present and hence a larger value, namely the right threshold value is added to the existing sum for that particular stage. If the computed value is lower than the feature threshold, there is less likelihood of a face being present and hence a smaller value, namely the left threshold value is added to the existing sum for that particular stage. Once the calculation of all features in a stage is complete, the calculated value is compared with the stage threshold. Accumulation of larger right values indicates more likelihood of a face being present and ensures that the particular stage threshold is passed. If the value is greater than the stage threshold, the successive stages of the face detection are run on the image window. If the value is lesser than the threshold, the image window is dropped and the next image window is used for running the face detection algorithm. During each clock cycle, the integral pixel values of Haar classifier from the integral image window buffer and the parameters of Haar classifier from the Haar feature BRAMs are fed to calculate the result of classification continuously. The latency for the first Haar classifier is five clock cycles. The Display module generates the appropriate signals required for the display of the image on the monitor. The detected face data sub-module draws a box around the area of the image which is detected to contain a face.

V. FACE DETECTION ALGORITHM

The face detection algorithm proposed by Viola and Jones is used as the basis of our design. The face detection algorithm looks for specific Haar features of a human face. When one of these features is found, the algorithm allows the face candidate to pass to the next stage of detection. A face candidate is a rectangular section of the original image called a sub-window. Generally, these sub-windows have a fixed size (typically 24×24 pixels). This sub-window is often scaled in order to obtain a variety of different size faces. The algorithm scans the entire image with this window and denotes each respective section a face candidate. Based on a machine learning algorithm called AdaBoost, this algorithm uses clever techniques like the Integral Image, Cascade structure. It is used to reduce the complexity in computation and quickly eliminate the non-face portions of the image. The algorithm uses an integral image in order to process Haar features of a face candidate in constant time. It uses a cascade of stages which is used to eliminate non-face candidates quickly. Each stage consists of many different Haar features. Each feature is classified by a Haar feature classifier. The Haar feature classifiers generate an output which can then be provided to the stage comparator. The stage comparator sums the outputs of the Haar feature classifiers and compares this value with a stage threshold to determine if the stage should be passed.

VI. HARDWERE DESCRIPTION

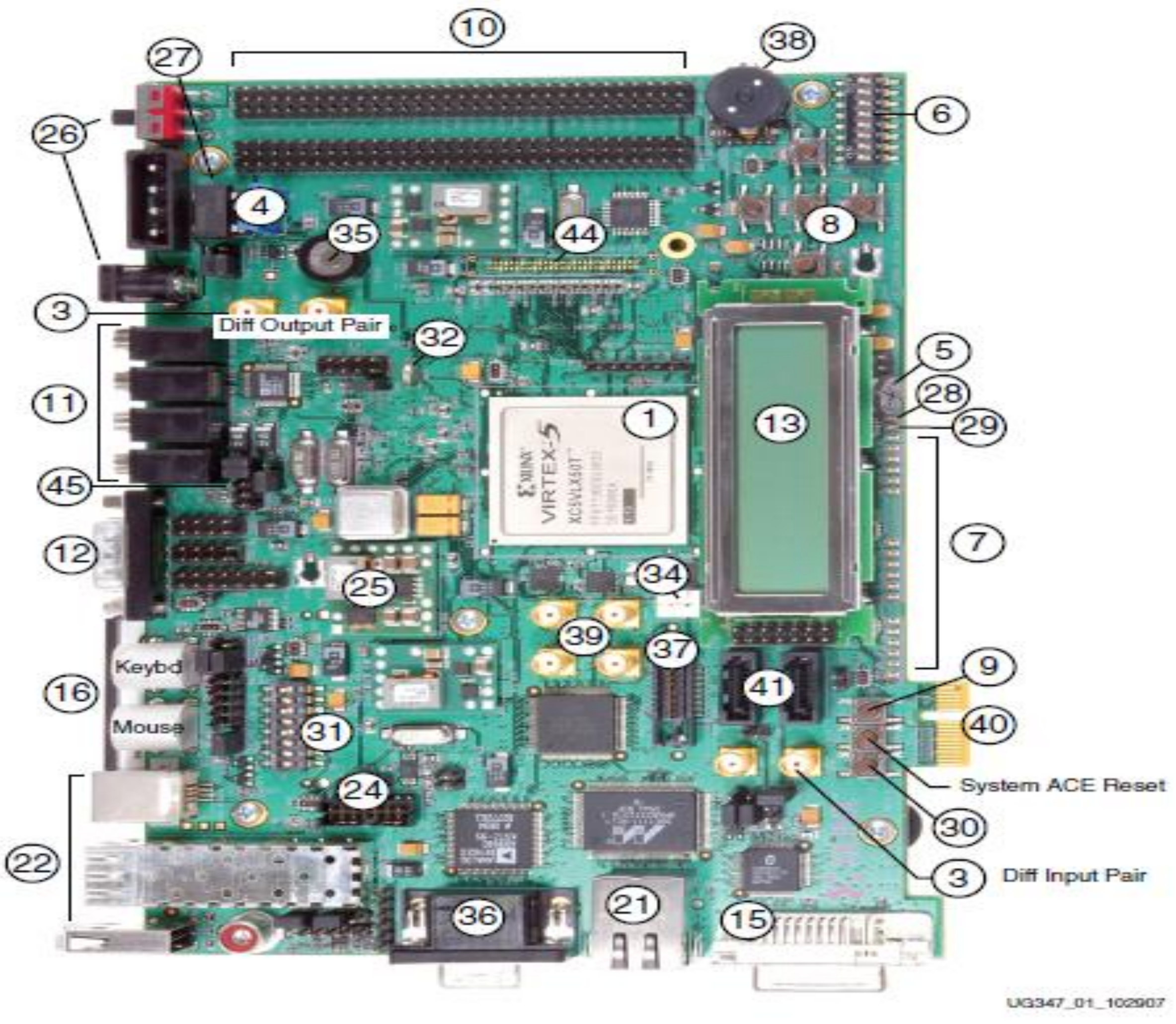


Figure 2. VIRTEX-5 FPGA

A Xilinx Virtex-5 FPGA is installed on the board. The board supports configuration in all modes JTAG, Master Serial, Slave Serial, Master Select MAP Slave, Byte-wide Peripheral Interface (BPI) Up, BPI Down, SPI modes.

6.1 GPIO DIP Switches

Eight general-purpose (active-High) DIP switches are connected to the user I/O pins of the FPGA. Table 1 summarizes these connections.

SW4	FPGA Pin
GPIO_DIP_SW1	U25
GPIO_DIP_SW2	AG27
GPIO_DIP_SW3	AF25
GPIO_DIP_SW4	AF26
GPIO_DIP_SW5	AE27
GPIO_DIP_SW6	AE26
GPIO_DIP_SW7	AC25
GPIO_DIP_SW8	AC24

7.1 User Pushbuttons

Five active-High user pushbuttons are available for general purpose usage and are arranged in a North-East-South-West-Center orientation.

Table 2 summarizes the user pushbutton connections.

Reference Designator	Label/Definition	FPGA Pin
SW10	N (GPIO North)	U8
SW11	S (GPIO South)	V8
SW12	E (GPIO East)	AK7
SW13	W (GPIO West)	AJ7
SW14	C (GPIO Center)	AJ6

8.1 16-Character x 2-Line LCD

The ML50x board has a 16-character x 2-line LCD (Tianma TM162VBA6) on the board to display text information. Potentiometer R87 adjusts the contrast of the LCD. The data interface to the LCD is connected to the FPGA to support 4-bit mode only. The CPLD is used to shift the voltage level between the FPGA and the LCD. The LCD module has a connector that allows the LCD to be removed from the board to access to the components below it.

9.1 DVI Connector

A DVI connector (P7) is present on the board to support an external video monitor. The DVI circuitry utilizes a Chrontel CH7301C capable of 1600 X 1200 resolutions with 24-bit color. The video interface chip drives both the digital and analog signals to the DVI connector. A DVI monitor can be connected to the board directly. A VGA monitor can also be connected to the board using the supplied DVI-to-VGA adaptor. The Chrontel CH7301C is controlled by way of the video IIC bus.

10.1JTAG Configuration Port

The JTAG configuration port for the board (J1) allows for device programming and FPGA debug. The JTAG port supports the Xilinx Parallel Cable III, Parallel Cable IV, or Platform USB cable products. Third-party configuration products might also be available. The JTAG chain can also be extended to an expansion board by setting jumper J21 accordingly.

11.1VGA Input Video Codec

The DB15HD connector (P8) on the board supports connectivity to an external VGA

Source. The VGA input codec circuitry utilizes an Analog Devices AD9980 device (U19). The AD9980 is an 8-bit 95 MSPS interface optimized for capturing YPbPr video and RGB graphics signals. Its 95 MSPS encode rate supports HDTV video modes and graphics resolutions up to XGA (1024 × 768 at 85 Hz). The Analog Devices AD9980 device is controlled by way of the Video IIC bus.

VII. RESULTS AND DISCUSSION

After the simulation of our face detection progamme. The RTL Schematic view of the top-level block is as shown below.

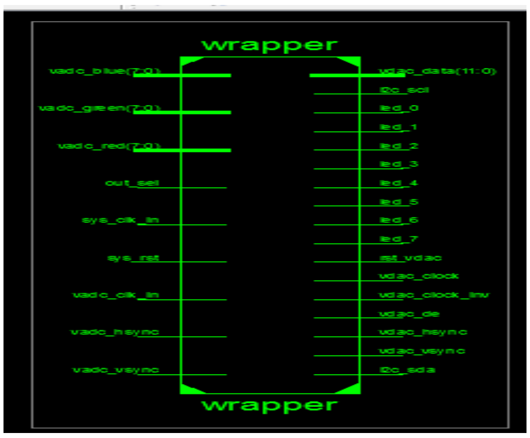


Figure 3. Top level block

In the above top-level model, the inputs are from the image interface (AD9980) and the output goes to the DVI transmitter as shown in block diagram fig. The inputs to the FPGA system are 8 bits of Red(R), Blue(B), Green(G), Horizontal sync, Vertical sync, Clock, Reset, Control signals. The top-level model can be further divided into 4 important sub-blocks they are I2C BLOCK, DCM system, Face detection, Camera Decode.

7.1 I2C BLOCK

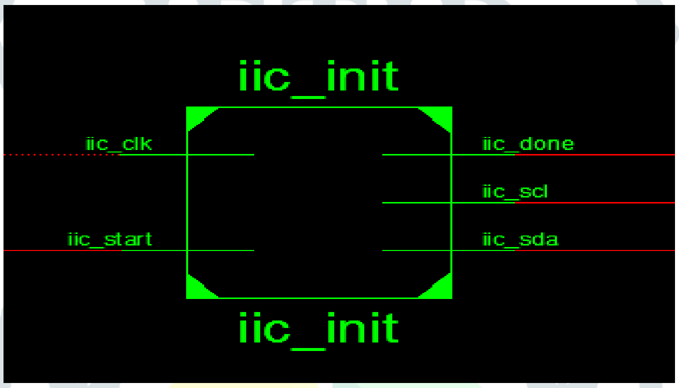


Figure 4. I2C BLOCK

The input for the I2C Block will be I2c clock and I2c start. The I2c is a 2 wired bus used for the serial data transfer of 8 bits in the bidirectional and it supports the master-slave relationship with multi-master option also. The 2 wires are namely Serial Data Line (SDA) and Serial Clock Line (SDL). The I2C clock input in the system is from the CLK output of the DCM_SYSTEM in the system. The I2C Start input in the system is from the output of Start_a_imp which is present internal to the main block. The output of the I2C block is iic_sda, iic_scl, iic_done. The internal of i2c contain another 2 sub blocks namely iic_init_7301, iic_init_9980.Internal of iic block is shown in figure 5.

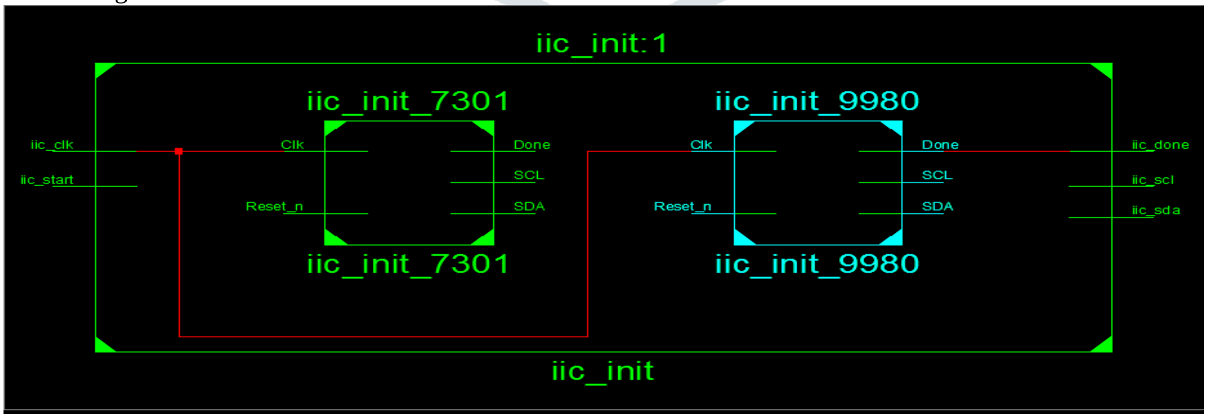


Figure 5. Internal of I2C Block

7.2 DCM SYSTEM

There is 1 clock source that produces a clock frequency of about 100 MHz This same signal is being used for Processing Logic, to Drive VGA.So we are generating the Digital Clock Manager from the IP core taken from XILINX IP core library called Core Generator. In our design we take 2 type of DCM system -DCM_SYS – It is used to produce the clock frequency suitable for the system operation.DCM_VGA – It is used to produce a clock frequency suitable to drive the VGA system.

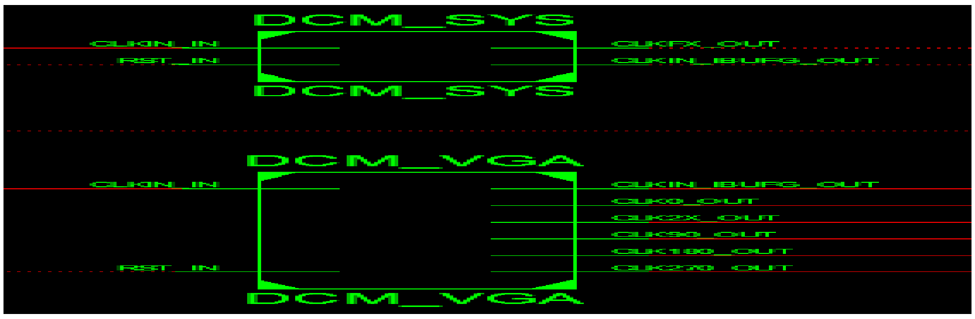
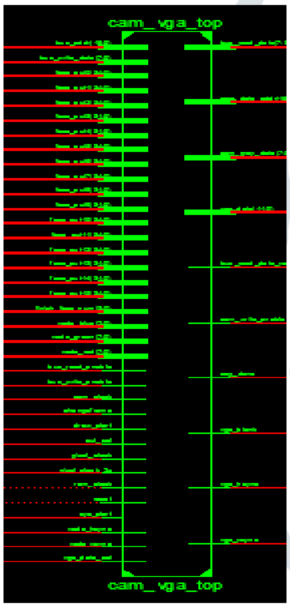


Figure 6. DCM system

The above figure shows the DCM system both has a common input of Clkin_IN, RST_IN. The output of DCM system is always the clock each clock generated will have the different frequency. Depending upon the requirement particular clock is taken.

7.3 CAM_VGA_TOP

The most important block in the system. This block is actually interconnected with the face detection block the output of the face detection block is cascaded with this block. The input to the CAM_VGA_TOP is from the entire Face detection block output and the remaining input are from the DCM _SYS as the different clock frequency and importantly the RGB component each of 8 bits obtained from image interface is given as the input to the system. The important output obtained is a 12-bit vga_data which is used to drive the DVI Transmitter and there will be a one Horizontal and Vertical sync. The figure 7 shows CAM_VGA_TOP block with its input and output and the figure 8 shows the interconnection between face detection block and VGA_VGA_TOP.



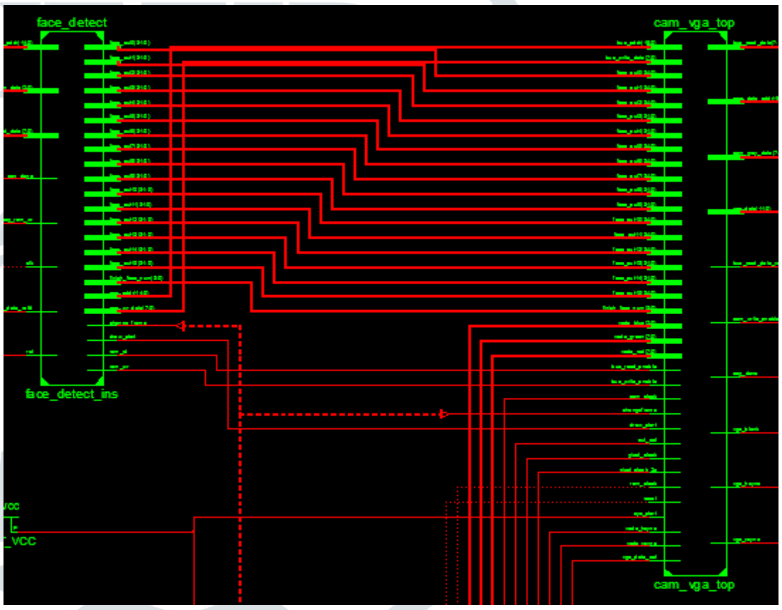


Figure 7 Figure 8

7.4 Face Detection

It is a block that actually interconnected with the CAM_VGA_ TOP the output from the cam is driven to the Face detection block. There is also a reset button as an input and the clock out of the DCM_system is given as one of the inputs to the face detection block. This block internally contains Bram and the output of the Face detection block is Face_out each of 32 bits and it is totally 16 in number. The block of Face Detection is shown in figure 6.6 below.

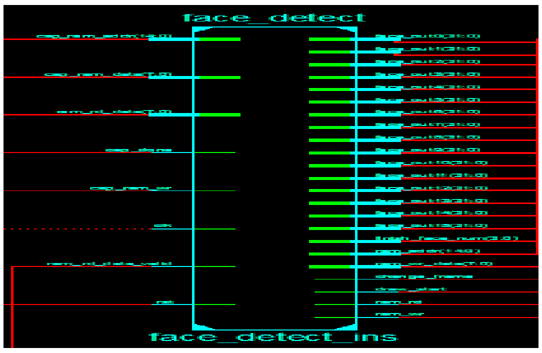


Figure 9. Face detection block

VIII.CONCLUSION

In our work the face detection is based on the AdaBoost algorithm using Haar features. The scaling image technique is used instead of the scaling sub-window, and the integral image window is generated instead of the integral image contains whole image during one clock cycle. The face detection is implemented on top of a High-Density FPGA VIRTEX-5. This face detection creates a sub window of fixed size of (24×24 pixels). We use a resolution of (600×800) for our project. Finally, as engineers we are committed to create innovative solutions for the use of people. Hence, we have designed an A Real Time Face Detection system using an FPGA which is useful in Surveillance, biometrics etc. We have designed for the Resolution of (600*800) for a VGA so it can be generalized for any resolution in further. Further Enhancement for Multiple Face component with better resolution. It can be implemented on more enhanced FPGA for high speed and low power of the system.

IX. Acknowledgment

The authors will like to express sincere thanks and appreciation to his esteemed mentor and supervisor for their ongoing inspiration, technical support, guidance, encouragement, and insightful suggestions that enabled him to undertake this research work.

References

1. P. Viola and M. Jones, “Robust real-time face detection,” Computer Vision, IEEE International Conference on, vol. 2, p. 747, 2001.

2.MA. Berbar, H.M. Kelash, and A.A. Kandeel, "Faces and facial features detection in color images," Geometric Modeling and Imaging-New Trends, pp. 209-214, 1993.

3.L. He, Y. Chao and K. Suzuki, "A run-based two-scan labeling algorithm," IEEE Transactions on Image Processing, vol. 17, no. 5, pp. 749-756, 2008. (Pubitemid 351597165)

4.L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," Pattern Recognition, vol. 42, pp. 1977-1987, 2009.

5.T. Wark, "An approach to statistical lip modelling for speaker identification via chromatic feature extraction," Fourteenth International Conference on Pattern Recognition, vol. 3, pp. 123-125, 1998.

6.Ojala, T., Pietikäinen, M., Harwood, D. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions," Proc. 12th International Conference on Pattern Recognition. Vol. I (1994) 582-585.

7.A. Hadid, "The local binary pattern and its applications to face analysis," First Workshops on Image Processing Theory, Toolsand Applications, pp. 28-36, 2008.

8.R. McCready "Real-time face detection on a configurable hardware system," in Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing, 2000, pp. 157-162.

9.D. Nguyen, P. Aarabi, D. Halupka, and A. Sheikholeslami, "Real-time face detection and lip feature extraction using field-programmable gate arrays," IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics, vol. 36, 2006.

10.N. Farrugia, F Mamalet, S. Roux, F. Yang, and M. Paindvoine, "Fast and robust face detection on a parallel optimized architecture implemented on FPGA," IEEE Transactions on Circuits Systems. Video Technology, vol. 19, no. 4, pp. 597-602, 2009.
11.K.-T. Hu, Y-T. Pai, S.-J. Ruan, and Edwin Naroska, "Ahardware efficient color segmentation algorithm for face detection," IEEE Asia Pacific Conference on Circuits and Systems, pp. 688-691, 2010.
12.S.-W. Yang, M.-H. Sheu, H.-H. Wu, H.-E. Chien, P.-K. Weng, and Y-Y Wu, "VLSI architecture design for a fast parallel label assignment in binary image," IEEE International Symposium on Circuits and Systems, vol. 3, pp. 2393-2396, 2005.
[