HardwareX 1 (2017) 68–75

Contents lists available at ScienceDirect

HardwareX

journal homepage: www.elsevier.com/locate/ohx

Hardware Article

# An efficient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm

Peter Irgens, Curtis Bader, Theresa Lé, Devansh Saxena, Cristinel Ababei *

*Dept. of Electrical and Computer Engineering, Marquette University, Milwaukee, WI, USA*

ARTICLE INFO

*Article history:*
Received 12 September 2016
Received in revised form 17 March 2017
Accepted 25 March 2017

*Keywords:*
Face detection
Viola-Jones algorithm
Field programmable gate arrays
Parallelization
Open source

ABSTRACT

We present an field programmable gate arrays (FPGA) based implementation of the popular Viola-Jones face detection algorithm, which is an essential building block in many applications such as video surveillance and tracking. Our implementation is a complete system level hardware design described in a hardware description language and validated on the affordable DE2-115 evaluation board. Our primary objective is to study the achievable performance with a low-end FPGA chip based implementation. In addition, we release to the public domain the entire project. We hope that this will enable other researchers to easily replicate and compare their results to ours and that it will encourage and facilitate further research and educational ideas in the areas of image processing, computer vision, and advanced digital design and FPGA prototyping.



## Introduction

Field programmable gate arrays (FPGAs) have become extremely popular in virtually all application domains. An example of such an application domain is computer vision, where one often finds object detection and tracking as basic techniques that are used to create more complex systems. The realtime performance of such systems crucially depends on highly efficient and cost effective implementations of those basic techniques. For example, in systems that deal with airport security where one may be interested in object or activity recognition and tracking, face detection is a crucial technique.

One of the most popular face detection algorithms for realtime applications is the Viola-Jones (VJ) algorithm [1]. While other variations of this algorithm have been proposed [2] in this paper, we present a complete hardware implementation of the Viola-Jones face detection algorithm on a low-end FPGA chip. We focus on the Viola-Jones face detection algorithm due to its popularity and efficiency and because it underlies a lot of other face detection algorithms. Our hardware implementation is described entirely in a hardware description language (HDL). We compare our HDL implementation to software based executed on general purpose processors or CPUs. The hardware FPGA based implementation offers a lower performance measured as frames per second (fps) compared to the software CPU-alone implementations for an image size of $320 \times 240$ pixels. However, it represents a good solution from a performance-power-price point of view. In addition, the FPGA based implementation has the potential to improve performance if deployed with greater parallelism and especially for larger image sizes on more complex but also more expensive FPGA chips. As such, we release the FPGA based implementation to the public domain.

* Corresponding author.
*E-mail address:* cristinel.ababei@marquette.edu (C. Ababei).

http://dx.doi.org/10.1016/j.ohx.2017.03.002
2468-0672/

While there are previous studies that reported FPGA based implementations of the Viola-Jones algorithm or of portions of it [3–5], to the best of our knowledge, our contribution is the first such implementation that is fully disclosed and made publicly available. This ensures that our results can be replicated and hopefully will encourage further research and educational projects in the areas of image processing, computer vision, and advanced HDL design and FPGAs.

## Background on Viola-Jones face detection algorithm

In this section, we present a high level description of the Viola-Jones face detection algorithm. This description is kept to a minimum and is necessary as it will help us better understand how to port and describe in HDL key tasks for the FPGA based implementation. For more details on this algorithm, please see [1]. The pseudocode description of the Viola-Jones algorithm is presented in Fig. 1.

In a face detection algorithm, we must use an accurate numerical description such that it sets human faces apart from other objects in a given image. Such characteristics can be extracted with a committee algorithm called Adaboost [6]. Such a committee can be created with weak classifiers to form a strong classifier by employing a voting mechanism. The Viola-Jones algorithm uses Haar-like *rectangle features* to construct classifiers. A Haar-like rectangle feature is a scalar product between the image and some Haar-like *pattern*. An example of a Haar-like pattern is shown in Fig. 2.

A crucial element of the Viola-Jones algorithm is a technique to compute rectangle features very rapidly [1]. This technique uses an intermediate representation for the image, the so called *integral image*.

The integral image at location $(x, y)$ contains the sum of the pixels above and to the left of $(x, y)$.

Instead of summing up all the pixels inside a rectangular window, this technique mirrors the use of cumulative distribution functions. Using the integral image any rectangular sum can be computed in four array references as shown in Fig. 2.

The Viola-Jones algorithm uses so called *cascade classifiers*. A cascade classifier is constructed as a sequence of stages. At each stage a list of filters are applied to the area within the sliding sub-window. An example of such a multistage filter with 25 stages is shown in Fig. 3. Each time the sliding sub-window shifts (typically pixel by pixel, but it can be more pixels at a time to further speed things up), the new region within the sliding sub-window is processed through the cascade classifier stage-by-stage. At each stage, the rectangle feature is evaluated and the weak classifier is computed. Then, a threshold check is used to see if the region is rejected as a face candidate or if it needs to continue to be processed in the next stage.

To be able to detect faces of different sizes, the algorithm works with a pyramid of scaled images. This allows sweeping using the same set of Haar-like patterns different scaled versions of the initial image. Thus, sliding sub-windows will sweep each of the images from the pyramid. When the outer-most for loop in the pseudocode description from Fig. 1 finishes its execution, the Viola-Jones algorithm would have found and marked with rectangle indicators all faces present in the original test image as well as in the scaled versions of the image.

**Algorithm**: Viola-Jones Face Detection Algorithm

```
1: Input: original test image
2: Output: image with face indicators as rectangles
3: for i ← 1 to num of scales in pyramid of images do
4:     Downsample image to create image_i
5:     Compute integral image, image_ii
6:     for j ← 1 to num of shift steps of sub-window do
7:         for k ← 1 to num of stages in cascade classifier do
8:             for l ← 1 to num of filters of stage k do
9:                 Filter detection sub-window
10:                Accumulate filter outputs
11:            end for
12:            if accumulation fails per-stage threshold then
13:                Reject sub-window as face
14:                Break this k for loop
15:            end if
16:        end for
17:        if sub-window passed all per-stage checks then
18:            Accept this sub-window as a face
19:        end if
20:    end for
21: end for
```

**Fig. 1.** Pseudocode description of the popular Viola-Jones face detection algorithm.

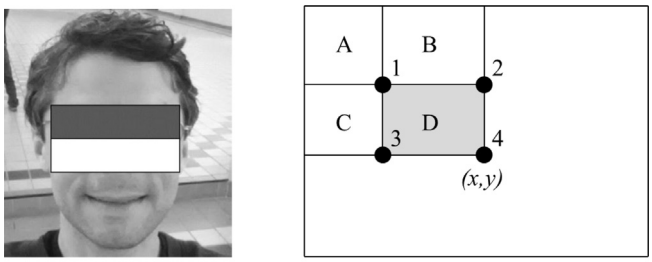


**Fig. 2.** (a) Haar-like rectangle feature is calculated only by using the pixels inside the Haar-like pattern (i.e., black and white rectangles). This example pattern illustrates that the area covering the eyes is usually darker than the area just above the cheeks. (b) The sum of the pixels inside rectangle D can be computed with four array references on values of the so called integral image: 4 + 1 – (2 + 3).

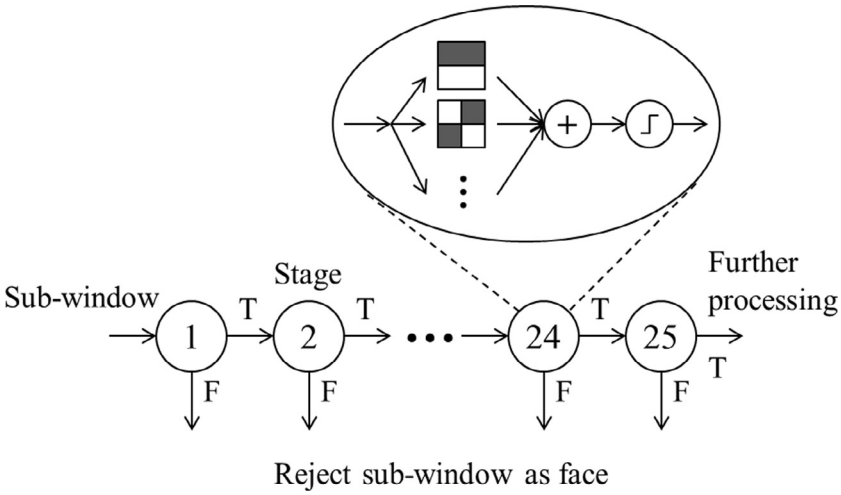


**Fig. 3.** Illustration of a cascade classier with 25 stages as a decision tree, where at each node a threshold check is done to decide if the sub-window is rejected as no face (False, F) or if it is passed for further processing to the next stage (True, T), which means that the sub-windows still has chances to contain a face.

## Hardware description

Upon careful study of the actual algorithm described in Fig. 1, we conclude that two of the most important aspects of the Viola-Jones algorithm are:

(1) The cascade classifier described in Fig. 3 must be applied to each sub-window during the scanning process of a given image from the pyramid of scaled images. Thus, it is desirable to implement this classifier in HDL as a *kernel*, which is then instantiated or deployed as many times as possible in order to perform multiple sub-window evaluations concurrently.
(2) The same algorithm logic is essentially applied in exactly the same way to each of the images from the pyramid of scaled images. Hence, if the target FPGA chip is large enough, we could run the same top-level Viola-Jones implementation on multiple images from the pyramid at the same time.

The simplified block diagram of the structural description in HDL of the entire Viola-Jones algorithm is shown in Fig. 4. To keep the block diagram readable, many details related to the control signals are omitted. Such details can be found and better understood by studying the HDL source code, which we make publicly available. Note that our implementation captures the entire logic of the algorithm (as described in Fig. 1) and thus it could be referred to as a complete baremetal implementation. That is, there is no connection to a host computer, no portion of the VJ algorithm is done on any computer; everything is inside the FPGA.

In our experimental setup for testing and validation, we use the OV7670 CMOS camera module [7] for capturing realtime video. The video is at the same time displayed on a generic VGA monitor connected to the DE2-115 FPGA evaluation board

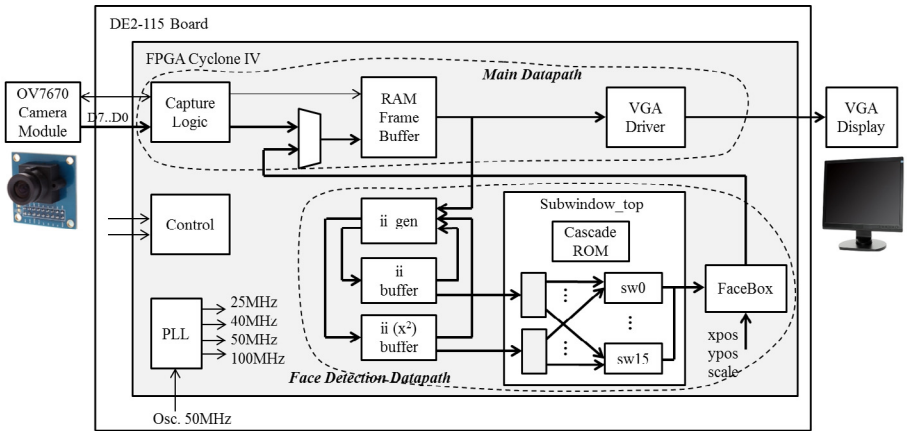


**Fig. 4.** Block diagram of the top-level HDL description of the design entity that implements the VJ algorithm.

[8]. The face detection portion of our design draws red squares, overlaid on the video frames themselves, to indicate faces detected in each frame of the video.

The face detection algorithm operates in parallel with the main datapath of the system shown in Fig. 4. The main datapath includes the *Capture Logic* block, which retrieves the video stream coming out of the OV7670 CMOS camera module and buffers it inside the random access memory (RAM) *Frame Buffer*. This buffer is implemented using existing embedded RAM memory blocks inside the Cyclone IV FPGA, which is the target FPGA chip in our hardware implementation. These buffers can be written and read asynchronously at different write and read clocks. Each video frame is used to drive, via the *video graphics array (VGA) Driver*, the VGA display monitor connected to the DE2-115 evaluation board.

By default, the video stream is captured from the camera module at a rate of 30 frames per second (fps) and a frame size of $320 \times 240$ pixels.

The face detection portion of the system is implemented by the following component blocks in Fig. 4: *ii-gen, ii buffer, ii* $(x)^2$ *buffer, Subwindow-top,* and *FaceBox*. The *ii-gen* block is responsible for the linear scaling and integral image generation. While the *Frame Buffer* stores an entire frame of $320 \times 240$ pixels, the *ii buffer* and *ii* $(x)^2$ *buffer* store only $39 \times 59$ pixels, which are dynamically updated on the fly during the sub-window sweeping process that covers the entire image. We do this because there is no need to compute and store the integral images for all $320 \times 240$ pixels. In addition, the Cyclone IV target FPGA chip does not have enough embedded RAM blocks to allow us to implement three full size $320 \times 240$ buffers.

The *Subwindow-top* block from Fig. 4 is the most important component of the *face detection datapath*. In our current implementation, it contains 16 copies (indicated as sw0, sw1, . . ., sw15) of the cascade classifier kernel applied to 16 sub-windows in parallel. In other words, in this implementation we run concurrently 16 classifiers whose outputs indicate whether any of the 16 sub-windows contains a face or not. For each classifier that detects a face, the *FaceBox* block then draws the square indicators by updating the appropriate pixels (i.e., changes their color to red) directly inside the *Frame buffer*, which is displayed on the VGA monitor. The red square indicators are drawn by the *FaceBox* block at the correct coordinates (*xpos, ypos*) with the appropriate scaling factor, which corresponds to the currently processed image from the pyramid of scaled images of a given frame.

In our current implementation, we use 16 classifiers because we are limited by the available resources, including embedded multipliers, available on the Cyclone IV FPGA chip. The number of classifier instances can be easily extended to further increase the hardware parallelism if more complex FPGA chips are utilized. The *Control* block from Fig. 4 is responsible with the generation of all necessary control signals that orchestrate the correct operation of both datapaths, i.e., the main video buffering datapath and the face detection datapath, respectively. The *phase locked loop (PLL)* block uses one of the embedded PLL cores inside the Cyclone IV FPGA chip to generate all the necessary clock signals.

## Experimental results

### *Connections*

The experimental setup requires connections that must be created as shown in Fig. 5. The setup includes the DE2-115 FPGA board that has connected the OV7670 camera to it as well as a regular monitor. The board itself comes with its own power adapter and the board has a power switch. The connection between the board and the monitor is made through a standard video graphics array (VGA) cable, which comes with by default with any monitor. The connection between the camera module and the board is done using regular jumper wires. The jumper wires connect the pins of the camera (shown at the bottom, left in Fig. 5) to general purpose input/output (GPIO) pins on the board (shown to the right in Fig. 5). The one-to-one connections between these pins are listed in Table 1 below.

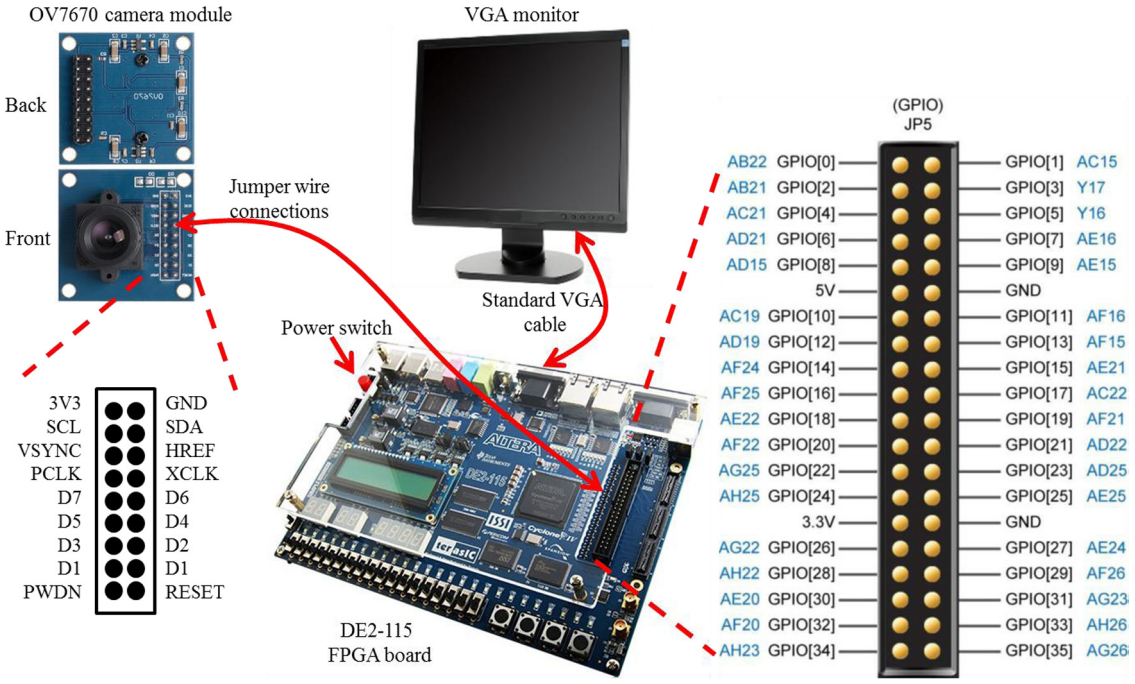


**Fig. 5.** Connections between board and camera module and display monitor.

**Table 1**
One-to-one connections between camera and board pins.

| Camera pin | Board pin | Camera pin | Board pin | Camera pin | Board pin | Camera pin | Board pin |
|---|---|---|---|---|---|---|---|
| 3V3 | 3.3V | VSYNC | AF25 | D7 | AE16 | D3 | Y17 |
| GND | GND | HREF | AC22 | D6 | AD21 | D2 | AB21 |
| SCL | AF24 | PCLK | AC19 | D5 | Y16 | D1 | AC15 |
| SDA | AE21 | XCLK | AF16 | D4 | AC21 | D0 | AB22 |
| RESET | AF15 | PWDN | AD19 | | | | |

*Hardware design*

The entire design from Fig. 4 was coded in the very high speed integrated circuit (VHSIC) hardware description language (VHDL), synthesized, placed, and routed with Quartus II Web Edition 15.1 tool [9]. These design steps are typical in the design of digital circuits where the specification is done in VHDL. Detailed tutorials on how to accomplish that can be found on the tool's website [9]. The design was tested and verified on the DE2-115 development board [8], which uses Altera's low-end Cyclone IV FPGA chip [10].

Our actual experimental setup is shown in Fig. 6. It includes the DE2-115 evaluation board with the low-end OV7670 camera module attached to the board, which also has connected to the VGA port a regular computer monitor. In our

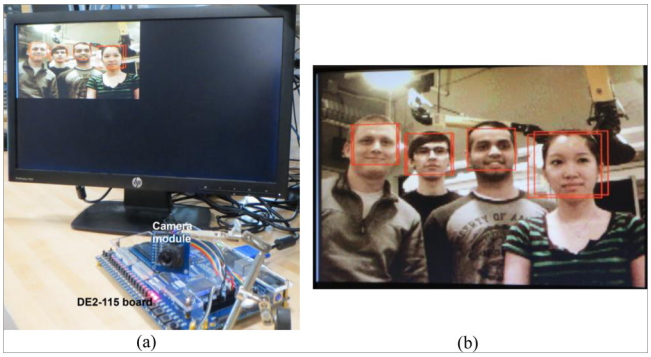


**Fig. 6.** (a) The experimental setup of the face detection design. Images are captured by the camera module and displayed on the VGA display as 320 × 240 pixel images. (b) Close-up of the display monitor.

**Table 2**
Summary of the report from the Quartus II tool.

| Item | Report |
|---|---|
| Family | Cyclone IV E |
| Device | EP4CE115F29C7′ |
| Total logic elements | 33,327/114,480 (29%) |
| Embedded multipliers | 369/532 (69%) |
| Total memory bits | 2,175,501/3,981,312 (55%) |
| Total pins | 56/529 (11%) |

**Table 3**
Comparison of performance achieved by the hardware FPGA design with software solutions.

| Test | Software GPU program | Software C++ computer program | Hardware FPGA design |
|---|---|---|---|
| Image 320 × 240, 1 face | NA | 7.76 fps | 4.4 fps |
| Image 320 × 240, 6 faces | NA | 6.43 fps | 4.4 fps |
| Image 320 × 240, 10 faces | NA | 6.41 fps | 4.4 fps |
| Image 640 × 480 (unknown number of faces) | 12.2 fps [11] | NA | NA |
| Image 1280 × 1024 (unknown number of faces) | 5.02 fps [12] | NA | NA |

experiments, we worked with an image size of 320 × 240 pixels and a pyramid of scaled images containing four images, which turned out to provide good performance in practice.

Key data of the summary report provided by Quartus II tool are presented in Table 2. Note the overall low resource (i.e., logic elements) utilization, which is 29%. While we have plenty of logic elements left to implement more than 16 kernel classifiers, we used the majority of the integrated embedded multipliers. Also note that the design uses 55% of embedded block memory bits. This rather high memory utilization is due to the buffers, with the *Frame Buffer* taking the lion share because it stores the pixel data for an entire frame of 320 × 240 pixels. This is the reason for which we do not work with larger image sizes; we do not have sufficient RAM memory inside the FPGA to store more pixels. For example, if we wanted to work with a frame size of 640 × 320 pixels, we would need four times as much memory bits, but we already used more than half of all that are available when working with a frame size of 320 × 240 pixels.

The final design has a performance of 30 fps in the default mode of operation where the video stream from the camera module is directly displayed onto the monitor. In the face detection mode, the performance is 4.4 fps, irrespective of the number of faces in the video frame, for tests that we performed with up to ten faces in the video frame.

*Comparison to a software implementation*

To get further insight into the performance of the FPGA based implementation, we compared it to a software implementation, which we ran on a computer (Intel Quad-core i7-2600 processor, 3.4 GHz, 6 GB of DRAM memory, and Linux Ubuntu 14.04 operating system). This software implementation is a computer program written in the C++ programming language. The program is a direct implementation of the Viola-Jones algorithm as described in Fig. 1. The comparison between the software computer program and the FPGA hardware implementation is shown in Table 3, where the performance is reported as frames per second (fps), which is better when it is higher. For reference only, we also include performance numbers achieved with graphics processing unit (GPU) based implementations reported in previous work, and discussed more in the next section.

The first test image contains just one face, the second test contains 6 faces, and the third test contains 10 faces. We look at images with different numbers of faces because we expect that images with more faces to require longer computational runtimes. That is because the cascade classifier from Fig. 3 needs to go more times (proportional to the number of faces that need to be detected) through the evaluation of each of the 25 stages. We note that as the number of faces in the test image increases, the performance of the software program degrades (i.e., the number of fps decreases) while the performance of the FPGA design remains stable. This indicates that the proposed FPGA hardware implementation has the potential to provide a bigger acceleration impact for large images containing many faces. That is because the scalability of the FPGA based implementation can be improved by increased hardware parallelism, i.e., by instantiating a larger number of kernel classifiers. However, we currently do not have data to support that because the low-end FPGA chip that we use does not have available more embedded memory bits, as discussed in the previous section. We would need to buy a more expensive FPGA for that.

## Discussion

First, we would like to emphasize that the target audience for the proposed design includes educators, researchers, and industry practitioners who have the minimum necessary background on digital logic design and HDL programming. In this context, the proposed face detection design can be used to construct laboratories that deal with image processing concepts

and can use this design as a platform to showcase the face detection algorithm or to verify new ideas. The design can be extended or integrated in larger systems for research purposes on computer vision topics as well. Practitioners can use it for practical applications too. However, as already mentioned, some minimum background on digital design, HDL programming, and FPGA tools is necessary. This design is not intended for anyone in the general public.

Finally, we would like to make a couple of observations that list the main takeaway's about our design:

- The presented FPGA hardware implementation of the Viola-Jones face detection algorithm achieves a performance of 4.4 fps for image sizes of 320 × 240 pixels. When the image under test contains a relatively small number of faces, the software implementation runs faster. However, that requires a general purpose computer, which is significantly more expensive than the FPGA board. When the image under test contains a large number of faces the performance of the FPGA design becomes comparable to that of the software implementation. In addition, we project that for larger image sizes and larger number of faces, the FPGA design can potentially offer better performance because the FPGA design can execute in parallel more computations by exploiting the hardware parallelism offered by the FPGAs. However, this is only a projection because we do not have a bigger (and more expensive) FPGA board to verify that. While we do not have experimental results to verify this prediction, we observe that the performance of the FPGA implementation is relatively stable while that of the software implementation degrades with the increase in image size as well as with the increase in the number of faces in the image.
- We would like to mention that yet another approach to implement the Viola-Jones algorithm is to use the recently popular graphics processing units (GPUs). GPUs have become increasingly popular and can be utilized for general purpose computations as well. For example, the study in [11] presented such a GPU implementation and reported a performance of 15.2 fps using a system with 4 T GPUs. However, the price of one single Tesla GPU card is more than USD 2000.
- While, generally, FPGA based solutions offer better face detection performance, it is still unclear about which solution, GPU based or FPGA based, typically offers a better performance-price solution point.
- For example, the study in [11] makes a case for GPU based solutions because the price of low-end GPUs (such as GeForce GTX285) is in the USD 300–500 price range (but that does not include the price of the computer that is needed to host the GPU card), while the price of a Virtex-5 FPGA evaluation board may vary within USD 995–3995 price range. However, the price of the DE2-115 board that we use in this paper is about USD 600 (or USD 300 as academic price). Hence, it is just difficult to say which solution is better.
- However, from a power consumption point of view, GPUs consume tens of times more power than FPGAs. This makes for the FPGAs to be a more desirable solution especially in battery operated systems.

## Specifications table, design files, and bill of materials

*Specifications table*

| | |
|---|---|
| Hardware name | FPGA based implementation of Viola-Jones face detection algorithm |
| Subject area | • Engineering and material science<br>• Educational tools and open source alternatives to existing infrastructure |
| Hardware type | • Imaging tools<br>• Electrical engineering and computer science |
| Open source license | Creative Commons Attribution 4.0 (CC-BY) |
| Cost of hardware | USD 310 |
| Source file repository | http://dejazzer.com/hardware.html (see project Face Detection on FPGA) |

*Design files summary*

| Design file name | File type | Open source license | Location of the file |
|---|---|---|---|
| Multiple hardware description VHDL source files | VHDL description of all the components from Fig. 4 | CC-BY 4.0 | http://dejazzer.com/hardware.html (see project Face Detection on FPGA)https://github.com/eigenpi/Face-Detection-on-FPGA(back-up copy of entire project) |
| E50FinalReport.pdf | Detailed description of design and experiments | CC-BY 4.0 | http://www.dejazzer.com/eigenpi/facedetection/E50FinalReport.pdf |
| Youtube video | Video demonstration | CC-BY 4.0 | https://youtu.be/aj4FEovXVXM |

The VHDL source files contain abundant code comments to aid in understanding the overall design. There are in total 43 VHDL source files that make up the complete description of the entire project. However, the main archive (available at

http://dejazzer.com/hardware.html) and its backup copy at Github contain the complete Quartus II project directory for convenience or reuse. In this way, the user only needs to only open the project with the Quartus II tool and program the FPGA board directly without the need to create a new Quartus project from scratch to which the VHDL files could be added.

Note that the user needs to download and install first the Quartus II tool, which is free and available here: https://www.altera.com/downloads/download-center.html

Basic knowledge of VHDL and digital design as well skills to use the Quartus II tool are required. Therefore, this project is not intended for the wide general audience but rather for educators, researchers, and practitioners in the field of digital design, computer vision, and FPGA prototyping.

The report file E50FinalReport.pdf provides further design details as well as instructions on how to build the experimental setup.

*Bill of materials*

| Designator | Component | Number | Cost per unit – currency | Total cost | Source of materials | Material type |
|---|---|---|---|---|---|---|
| Board | DE2-115 FPGA board | 1 | USD 300 (academic) USD 600 (regular) | USD 300 | www.terasic.com | Printed circuit board (PCB), semiconductor |
| Camera | OV7670 camera module | 1 | USD 10 | USD 10 | www.amazon.com | PCB, semiconductor |
| Wires | Jumper wires pack of 20, female to female | 1 | USD | USD 6 | www.amazon.com | Metal, plastic |

## Conclusions

We presented a new FPGA based hardware implementation of the popular Viola-Jones face detection algorithm. The implementation was verified on the affordable DE2-115 evaluation board. We found that for an image size of 320 × 240 pixels the hardware FPGA based implementation is providing a performance of 4.4 frames per second. We expect that the benefits of the FPGA based implementation to be even larger for larger image sizes. As another contribution, we release the complete implementation to the public domain. We hope that this will enable easy replication and comparison of results, and more importantly, will encourage further research and educational ideas in the area of hardware acceleration with application in computer vision and related topics.

## References

[1] P.A. Viola, M.J. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR), 2001.
[2] S. Zafeiriou, C. Zhang, Z. Zhang, A survey on face detection in the wild: past, present and future, J. Comput. Vision Image Underst. (2015).
[3] J. Cho, B. Benson, S. Mirzaei, R. Kastner, Parallelized architecture of multiple classifiers for face detection, in: IEEE Int. Conf. on Application-specific Systems, Architectures and Processors (ASAP), 2009.
[4] A. Acasandrei, A. Barriga, Design methodology for face detection acceleration, in: IEEE Conf. of the Industrial Electronics Society (IECON), 2013.
[5] V. Suse, D. Ionescu, A real-time reconfigurable architecture for face detection, in: Int. Conf. on ReConFigurable Computing and FPGAs (ReConFig), 2015.
[6] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (SS971504) (1997) 119–139.
[7] OV7670, Camera module with OmniVision CMOS sensor, Datasheet, 2015. Available on-line at <http://www.cutedigi.com/pub/sensor/Imaging/OV7670-Datasheet.pdf>.
[8] DE2-115, Development and education board, 2016. Available on-line at <http://www.altera.com/education/univ/materials/boards/de2-115/unv-de2-115-board.html>.
[9] Quartus II Web Edition Software 15.1 tool, 2016. Available on-line at <http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html>.
[10] Cyclone IV FPGA family: lowest cost, lowest power, integrated transceivers, 2016. Available on-line at <http://www.altera.com/devices/fpga/cyclone-iv/cyiv-index.jsp>.
[11] D. Hefenbrock, J. Oberg, N. Thanh, R. Kastner, S. Baden, Accelerating Viola-Jones face detection to FPGA-level using GPUs, in: IEEE Int. Symposium on Field-Programmable Custom Computing Machines, 2010.
[12] J. Kong, Y. Deng, GPU accelerated face detection, in: Int. Conf. on Intelligent Control and Information Processing, 2010.