

Efficient Usage of FPGA for Face Detection using Viola Jones

Abstract

Face detection using Viola Jones algorithm is a well-known technique. But Viola Jones method is computationally intensive and requires high performance computers to do face detection in real time. But for the applications which require compactness, portability and power optimization, this technique is not suitable. In this white paper, we describe an innovative FPGA based design which helped realize a 5 object classifier, in an economical embedded design, striking a perfect balance between cost and performance.

on an 8 core Xeon processor, 4GB RAM machine, into an under 300USD embedded solution. Using a high end FPGA was not desirable. This asked for a split in the design between hardware and software. Core computation and feature evaluation may be done in FPGA and post-processing may be handled in the software.

Overview

Object detection using Viola Jones is based on Adaboost learning algorithm using Haar features. The classifier is trained with several images of the object and the Haar features are extracted and arranged in a structured format, called the feature database. The feature database comprises of several cascaded stages with each making stricter requirements on the object candidate.

Detection algorithm looks for the specific Haar features of the object class in a rectangular section of the frame, called sub-window. When a Haar feature is identified, a sub-window is said to be qualified as an object candidate. All identified object candidates of a stage are passed to the subsequent stages and the process continues till the object is finally detected.

Now implementing all the stages in an economical FPGA was still not feasible. So going by the observation that more than 90% of the candidates get rejected in the first few stages, we compute only the initial three stages of sub-windows in FPGA. Only those windows which pass these stages are handed over to the software. This way the software has fewer windows to compute.

The feature database is extremely dynamic. It changes with training images and training parameters and is different for different classifiers. The detection procedure though more or less remains same for different classifiers (objects). So just by providing a different feature database we are able to detect different objects.

Design Objective

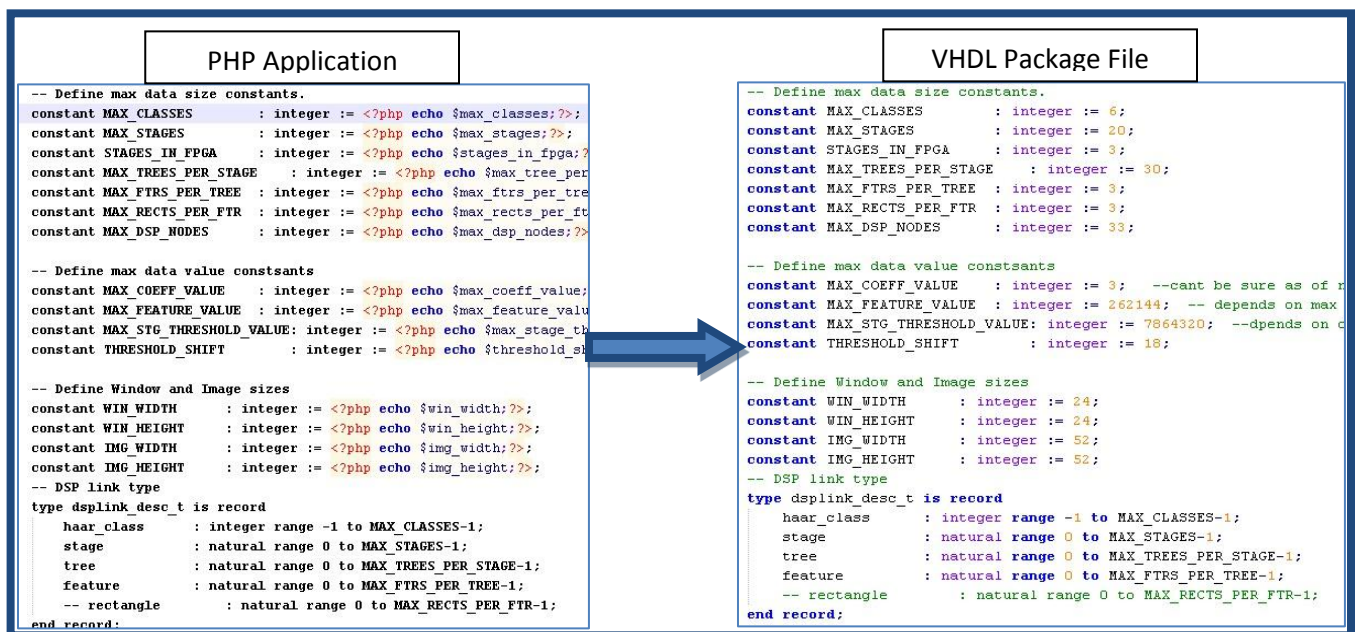
Real time object detection with processing speed close to 15-20fps translates into performance requirement of about a billion computations per sec for a single classifier implementation on a 512x512 image. Implementing the whole algorithm using an FPGA puts an enormous performance burden. Approach proposed by Jung Uk Cho, Shahnam Mirzaei, Jason Oberg and Ryan Kastner in "FPGA-Based Face Detection System Using Haar Classifiers", at International Symposium on Field Programmable Gate Arrays (FPGA), February 2009, used high-end Virtex series, Xilinx FPGA, for a single classifier implementation.

KritiKal's objective was to squeeze an existing, 5 object classifier solution, previously developed by KritiKal Solutions, and which ran

The feature database is characteristic of a particular classifier. It is the output of the classifier training and is arranged in an XML tree. Now many standard libraries are available for XML parsing in C, C++ and PHP. Writing a VHDL/ Verilog code to use feature database directly from an XML file, would have asked for an unnecessary effort and taken a toll on FPGA's real estate resource. Hard coding a particular feature database into a VHDL package file solves the purpose. But this would have meant maintaining different packages for different objects, where typical size of each feature database can go up to 1,00,000 lines of code for 5 stages. Such a code is totally un-manageable. Scalability and manageability are the key features of a successful design. So procedure to generate the synthesizable feature database had to be automated.

Approach: Generating VHDL code using PHP

PHP is a widely used general purpose scripting language that is especially suited for text processing. Normal text and PHP code can be easily intermixed in PHP. We used the template generation capability of PHP to write configurable VHDL package files. We wrote a PHP application which translated the XML file of feature database into a VHDL package. The template for our VHDL package is a mix of text which remains unchanged for all classifiers and the dynamic data extracted from XML files which is different across different classifiers. PHP code when run generates VHDL code file that is syntactically and semantically correct, with appropriate data inserted in them and becomes a synthesizable VHDL code. Using this application we were able to generate VHDL packages of about 1,00,000 lines instantly. This also provided us the flexibility to change the number of classifiers and use different training outputs easily. This would not have been possible by conventional methods. Following figure shows code snapshots of a PHP application and the corresponding output VHDL package file.



Keynotes:

With the processing distributed between processor and FPGA, we were able to utilize the resources optimally and efficiently. The solution requires a relatively cheaper FPGA and a less complex processor, without much compromise on speed or configurability.

This idea of using PHP to generate VHDL codes achieved a greater level of optimization at a little cost of field programmability, but retained the manageability and scalability of the design.