FPL 2012
29-31 August
Oslo, Norway

FLOATING POINT HOG IMPLEMENTATION FOR REAL-TIME MULTIPLE OBJECT DETECTION

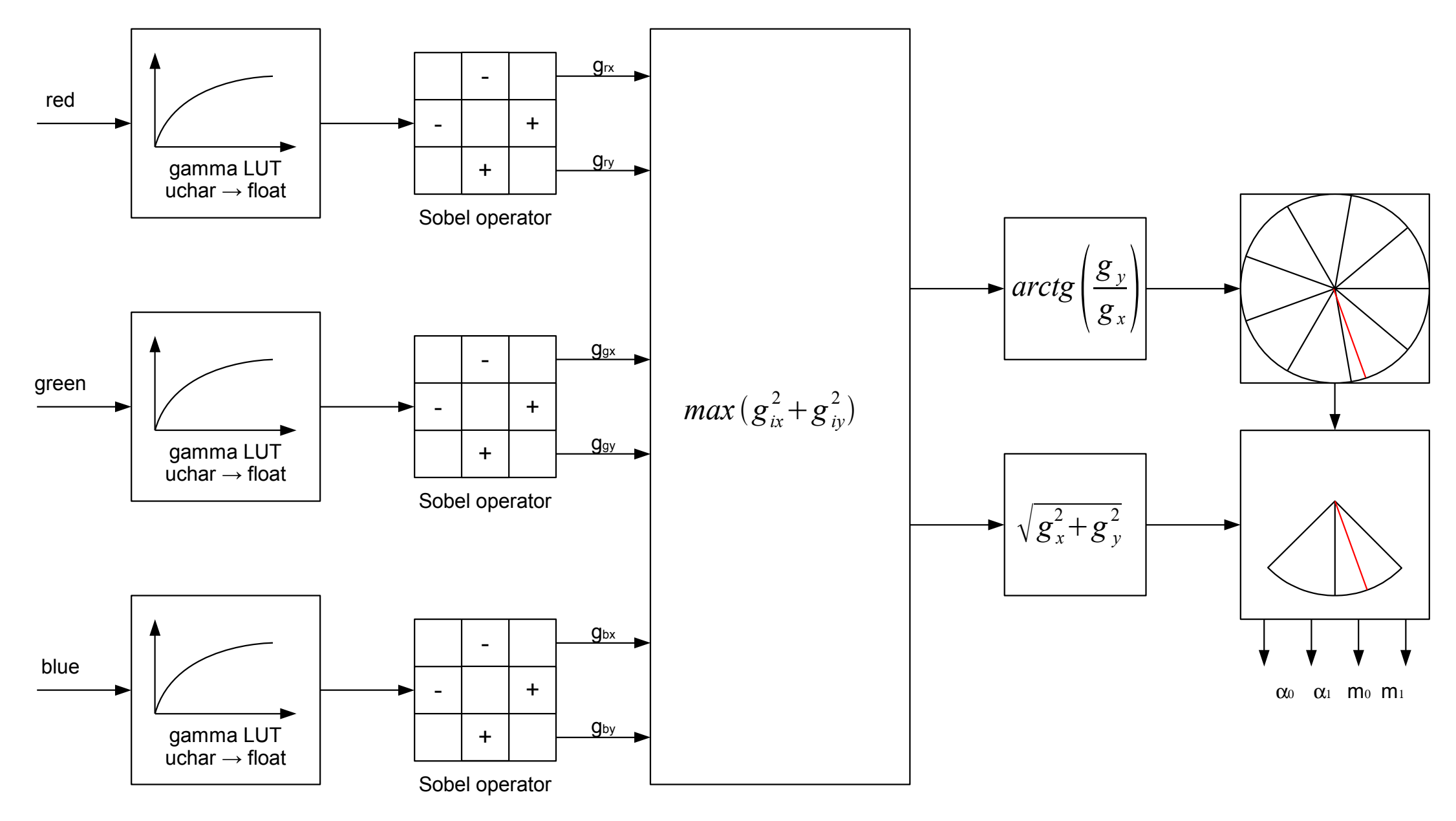
Mateusz Komorkiewicz, Maciej Kluczewski, Marek Gorgoń
AGH University of Science and Technology
al. A. Mickiewicza 30, 30-059 Kraków
email: { komorkie,kluczews,mago}@agh.edu.pl

1 INTRODUCTION

Object detection and localization in a video stream is an important requirement for almost all vision systems. In this work a design embedded into a reconfigurable device which is using the Histogram of Oriented Gradients for feature extraction and SVM classification for detecting multiple objects is presented. Superior accuracy is achieved by making all computations using single precision 32-bit floating point values in all stages of image processing. The resulting implementation is fully pipelined and there is no need for external memory. Finally a working system able to detect and localize three different classes of objects in color images with resolution 640x480 @ 60fps is presented with a computational performance above 9 GFLOPS.

2 HOG ALGORITHM

Histograms of Oriented Gradients [1] is a widely used algorithm for object detection (especially pedestrians). It uses histograms of oriented gradients for feature generation and a SVM (Support Vector Machine) classifier for classification. Unfortunately it has a quite high computational complexity and it is not possible to run it on CPU in real time. Available C++ implementation from OpenCV library was used as a reference for porting to FPGA. The first step is the gradient angle and magnitude computation. In the second stage the histograms of gradients are accumulated.

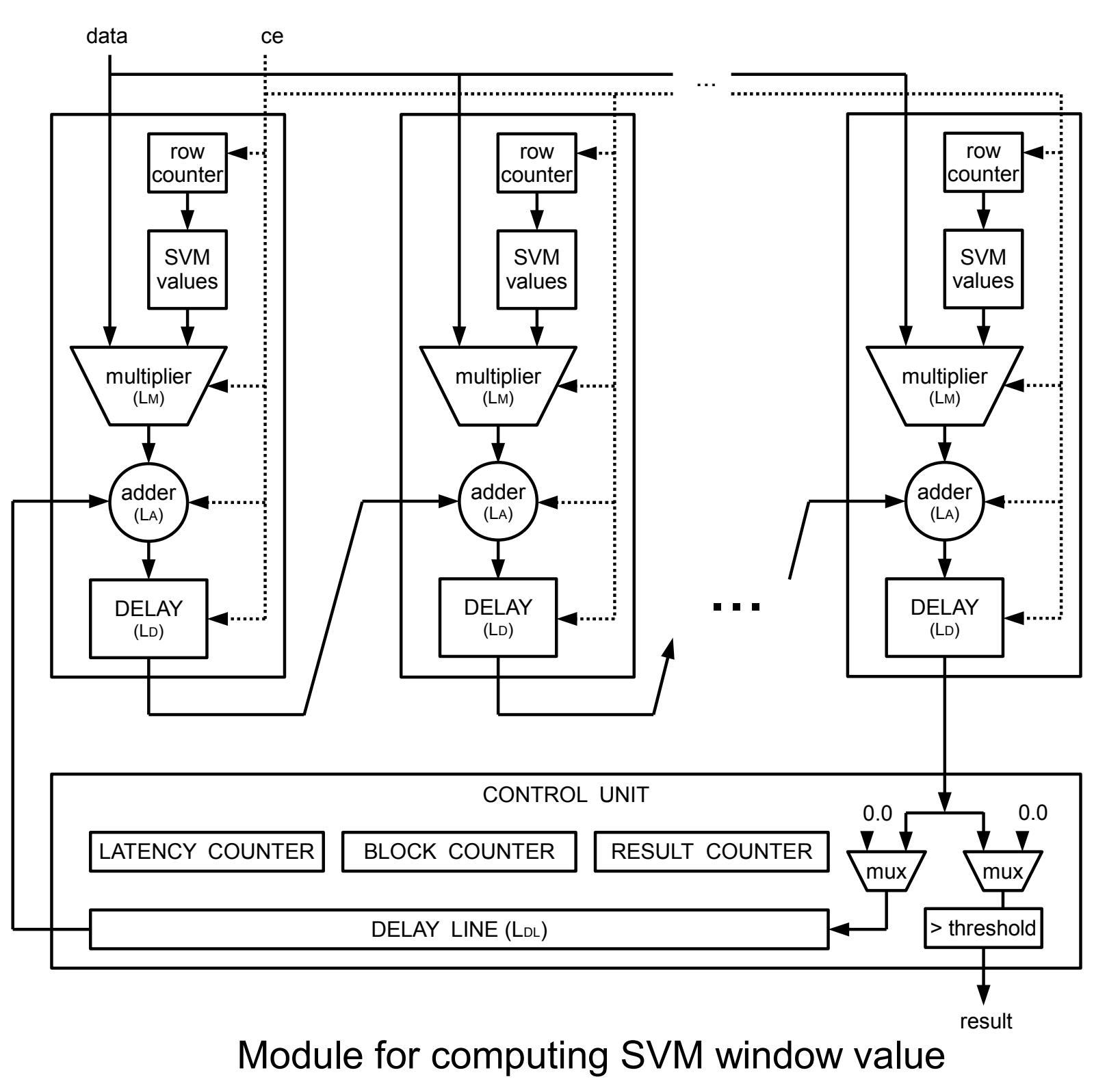


Module for gradient angle and magnitude computation

3 SVM CLASIFIER

The Support Vector Machines theory was presented in the work [2]. The SVM in this form is based on dividing the data into hyperplanes and is able to do a binary classification (negative/positive).

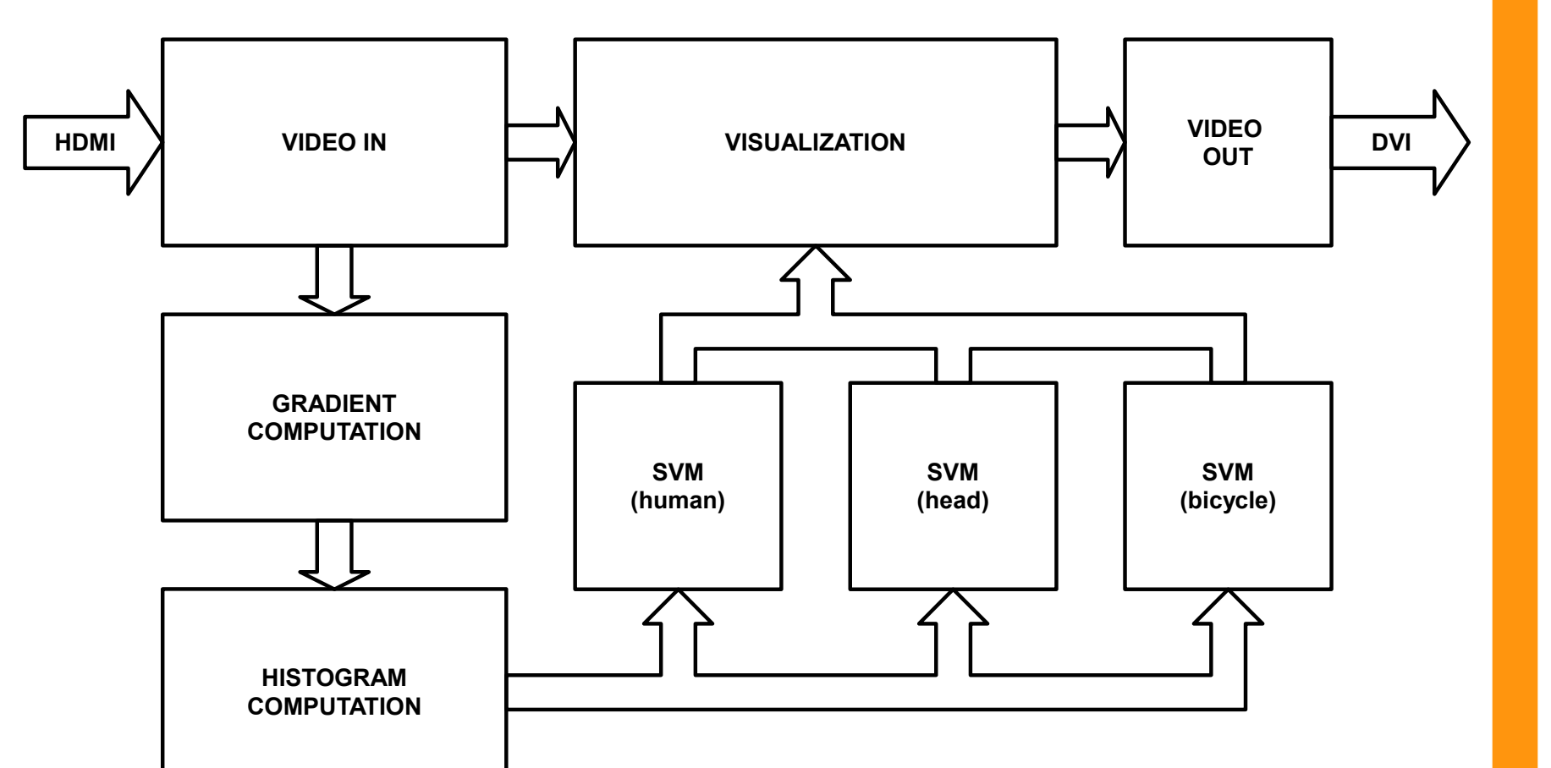
Since the previous research proved that implementing the SVM using fixed point arithmetic is reducing the detection rate, in our implementation we decided to use single precision floating point representation



Module for computing SVM window value

4 MULTIPLE OBJECT DETECTION

Because SVM does a binary classification, different instances has to be used for detecting multiple objects. The gradient and histogram computation modules are used to process the image, divide it into cells and blocks and compute the feature vector for each block. The data is then streamed to multiple instances of SVM classifiers. In the first block, a human detection is performed. In the second block heads are detected. In the third block bicycles are classified. When an object is detected in SVM, the information is sent to the visualisation module, which is used to draw the rectangles of a specified colour and size on the screen (to mark object type and location).



Detection system

5 FLOATING POINT & FPGA ?!

FPGA devices are not consider to be a good platform for implementing the floating point operations. It is quite an unjust opinion. The FPGA vendors are providing ready to use IP Cores for floating point arithmetic (addition, subtraction, multiplication, division and square root), implementing operations like:

$$v = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}$$

is possible. IP Cores are fast, the only tradeoff is longer latency and resource utilization.

Operation	Latency	Clock (MHz)	LUTs
ADD	1	77	414
ADD	12 (max)	516	447
MUL	1	104	842
MUL	8 (max)	516	668

Xilinx single precision floating point IP Cores

The number of resources is increasing very fast (newest Xilinx Virtex 7 device with more than 1.2 M LUTs) and it is possible to incorporate more than 1000 pairs of single precision floating point adder and multiplier (MAC) in a single chip.

6 RESOURCE UTILIZATION

Resource	Used	Available	Percentage
FF	75071	301440	24 %
LUT 6	113359	150720	75 %
SLICE	32428	37680	86 %
BRAM	119	416	28 %
DSP48	72	768	9 %

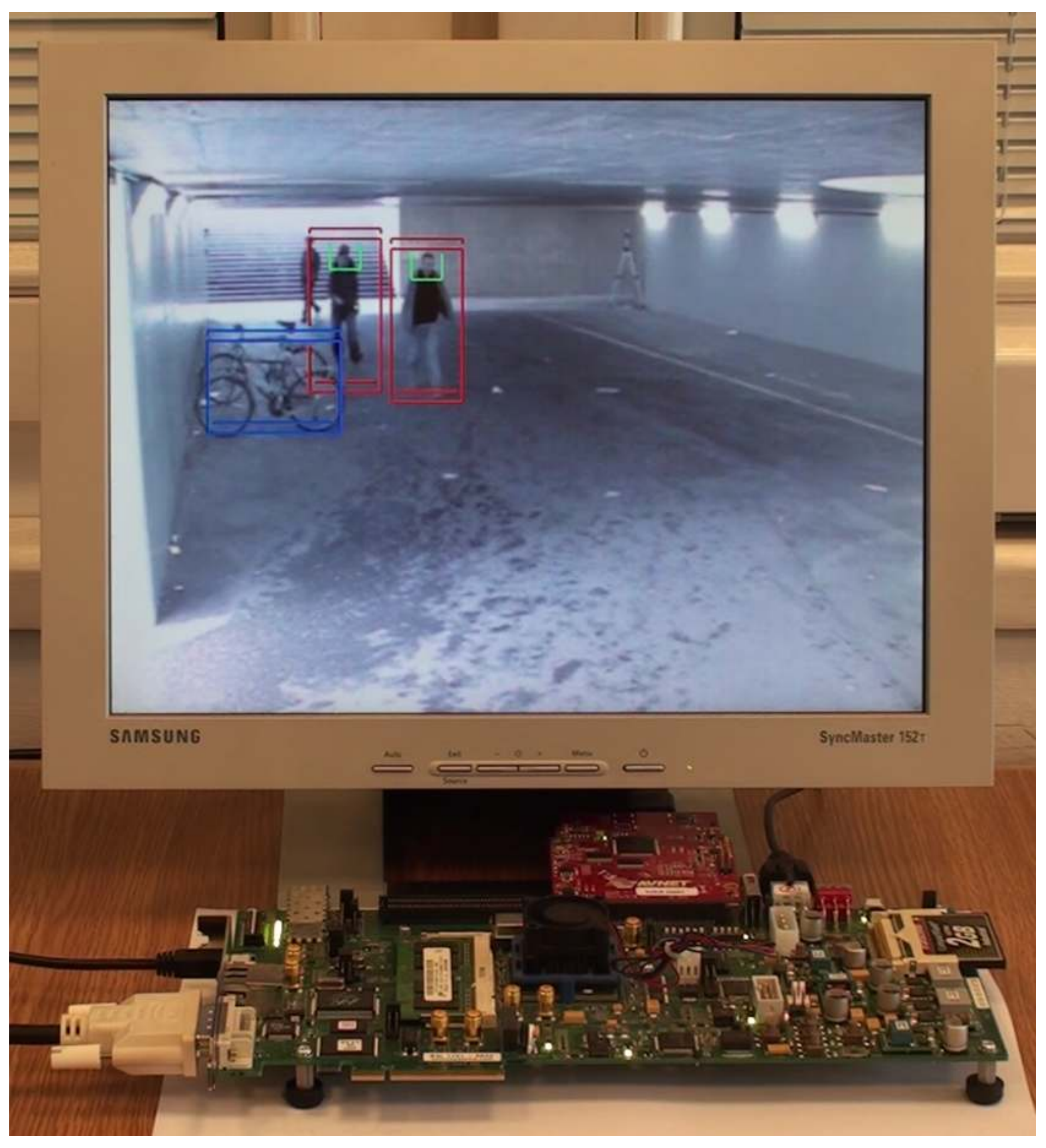
The design is using almost all slices of FPGA device, but it was our intent to test the limits and instantiate as many SVM classification modules as possible.

7 PERFORMANCE (9.47 GFLOPS)

Module	IP Cores	Clock	GFLOPS
Gradient	31	25 MHz	0.775
Histogram	77	25 MHz	1.925
SVM0	15	237 MHz	3.555
SVM1	7	50 MHz	0.350
SVM2	27	106 MHz	2.862
		Sum	9.47

The whole design is working with 25 MHz clock, with separate clock domains for SVM modules. The computational performance for all blocks reach the level of 9.47 GFLOPS.

8 WORKING SYSTEM (640x480 @ 60 fps)



Described implementation was tested using Xilinx ML605 board with Virtex 6 XC6VLX240T device and Avnet DVI I/O FMC expansion card for video input. Implemented system works with 60fps and process a single 640x480 frame in about 16.6 ms. For comparison, the original OpenCV implementation in C++ running on Core i7 2600 (3.4 GHz) process the same image in single scale with an average processing time of 50 ms when MMX and SSE instructions are used and 130 ms without them. The achieved speedup is 3.0x (with SIMD usage) or 7.8x (no SIMD)

9 REFERENCES

[1] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, June 2005, pp. 886 –893 vol. 1.
[2] V. N. Vapnik, The nature of statistical learning theory. New York, NY, USA: Springer-Verlag New York, Inc., 1995