Face-Detection on FPGA

Review

Lakshita Singhal, Prashant Pandey, Chaitanya Tejaswi,
Archisman Dey, Manchikatla Navya Sri

.

CS577: C-Based VLSI Design

Indian Institute of Technology, Guwahati

April 24, 2022

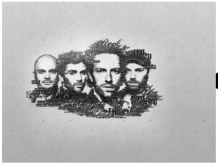
Reference Work (1)

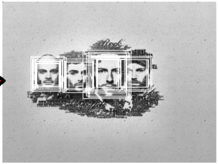
Accelerating Face Detection on Programmable SoC Using C-Based Synthesis

Nitish Srivastava Steve Dai Rajit Manohar Zhiru Zhang
School of Electrical and Computer Engineering, Cornell University, Ithaca, NY
{nks45, hd273, rm92, zhiruz}@cornell.edu

- Hardware-optimized Face Detection (Viola-Jones algorithm) implementation on PSoC using HLS-C.
- Frame Rate of 30 fps for Hardware implementation.
- Classification for 1, 2, 4, 8 faces.
- Original Work: Software (ARM Cortex A9), Hardware (Artix7 FPGA). [*]
- Our Work: Validation of Hardware Implementation.

# of faces	Software classifier	Hardware classifier
1	206 ms 4.8 fps	30 ms 33.4 fps
2	232 ms 4.3 fps	31 ms 32.1 fps
4	250 ms 4.0 fps	32 ms 31.3 fps
8	371 ms 2.7 fps	38 ms 26.3 fps





How did they do it?

Implementations

- baseline: Baseline implementation which replaces all the non-synthesizable constructs.
- pipelined: All Classifiers are pipelined.
- parallel-pipelined : Classifiers in the first 3 stages are parallelized, the next 22 stages are pipelined.
- main: All the above optimizations + sqrt.

Image Downsampling

Haar Features

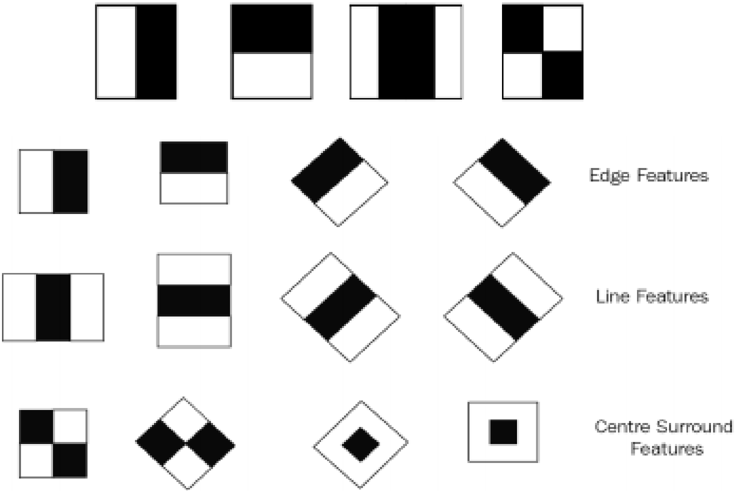
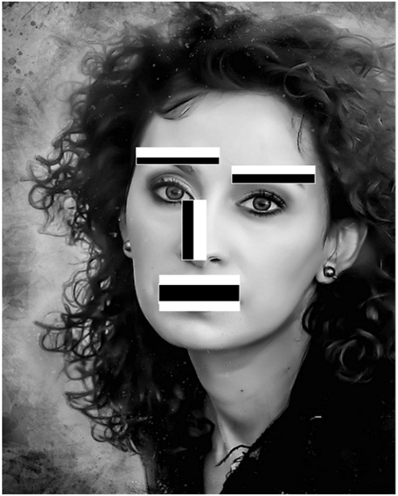
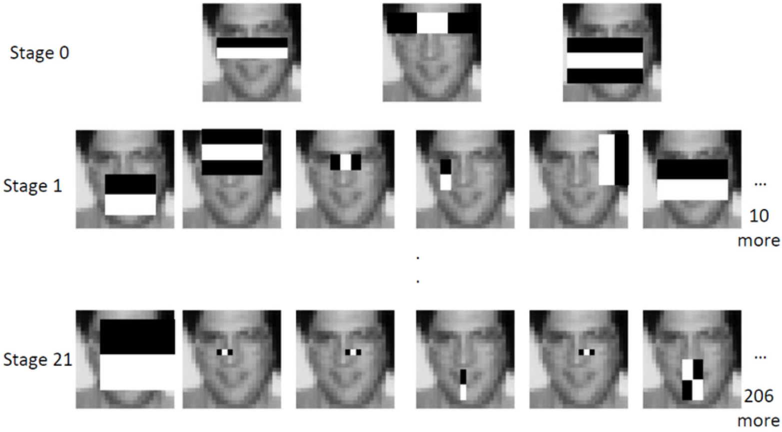


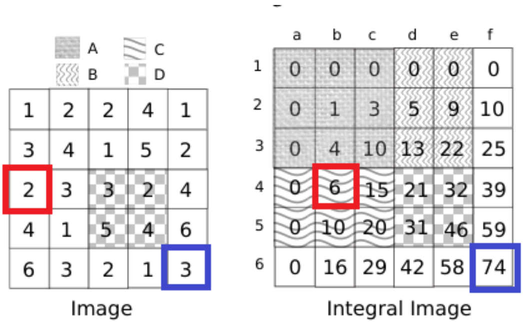
Figure: 2-3-4 Rectangle Features

Haar Features: Example




Stage 0
Stage 1
...
10 more
Stage 21
...
206 more

Integral Images



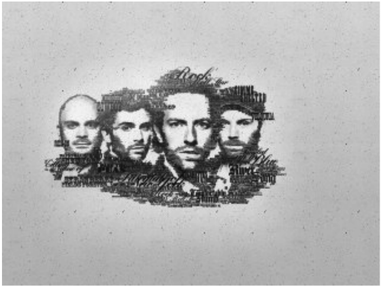
- The value at any location (x, y) of the integral image is the sum of the image pixel value above and to the left of the location (x, y).
- An IntegralImage pixel represents the sum of pixel values before it.

Viola-Jones Algorithm (2) (3)

1. Haar Feature Selection
2. Creating an Integral Image
3. Adaboost Training
4. Cascading Classifiers

Setup

- PC: sends image to FPGA in pgm format (pixel=8bit), where hardware implementation takes place.
- FPGA: detects all possible faces; returns coordinates of rectangles.
- PC: prints rectangles on image.



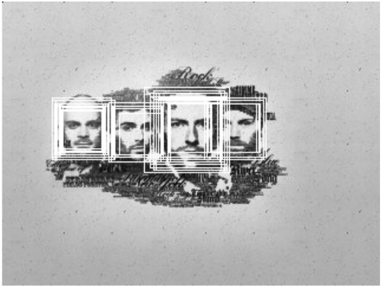


Image Scaler

- DownSamples the image ($f = 1.2$) using linear interpolation.
- Implementation: 2 nested loops (for image height, width). Inner loop has shift, multiply, assignment operations.

Integral Image Generator

- Takes downsampled image, makes Integral Image, which is stored into BRAM.
- Implementation: 2 nested loops (for image height, width). Inner loop updates pixel values by accumulating values to left/top. 2 nested loops (for image height, width). Shifts origin of window by one pixel after every iteration.

Cascade Classifier

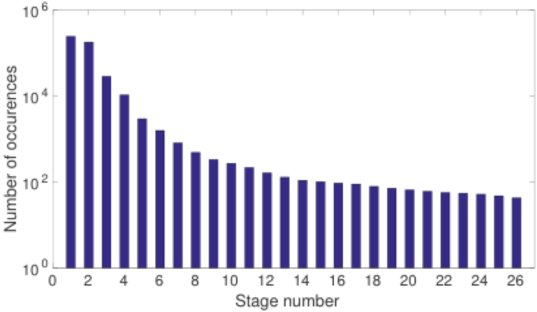
- Takes integral image & window origin coordinate, runs 25 stages of pre-trained Haar classifiers.
- Implementation: 2 nested loop (for no. of stages, no. of classifiers in each stage).

What to optimize?

- Nested loop - specially in Cascade Classifier!
- Fix WindowSize - 25x25 (height/width) so design is synthesizable.

Pipelining/Parallelization

- Nested loops in Cascade Classifier can be unrolled, and all 52 individual classifiers are run in parallel. (high hardware usage)
- Or, Pipeline only the inner loop ⇒ only single classifier whose parameters change every cycle.
- From fig, majority of windows pass through 1-3 stages, so, parallelize first three stages (9,16,27 classifiers respectively), and pipeline the rest.
- Coordinates obtained by classifiers of 1-3 stages are stored in BRAM.



Integral Image Windowing

- Coordinate fetching time - 12 cycles.
- Each classifier last for 1 cycle.
- Solution: Sub-Windowing - storing in registers - line buffers.
- Instead of producing Integral Image at once, produce windows of it, only needed by classfier. This addresses BRAM resource constraint.

Integral Image Banking

- To read 12 coords - 625$\times$1 MUX required - 170k LUTs.
- Instead, leveraged into 28 banks, such that any of the 12 coords don't lie in the same bank.

SquareRoot Approximation

- For eliminating lighting effects, normalization is required. Requires calculation of mean and standard deviation.
- Limiting Factor: 16 cycles.
- SquareRoot Calculation: SquareRoot of upper & lower halfs and left shifting the first result by 8, and adding them.
- Variance: Sum of MSB 10-bits left-shifted by 16, and lower 16 bits.

Test

Baseline Model

```
#pragma HLS inline off

#pragma HLS inline

#pragma HLS array_partition variable=coord complete dim=0

#pragma HLS pipeline

#pragma HLS unroll
```

Pipelined Model

```
#pragma HLS inline

#pragma HLS interface ap_stable port=aa

#pragma HLS array_partition variable=< > complete dim=<0, 1>

#pragma HLS unroll

#pragma HLS inline off

#pragma HLS pipeline
```

Parallel and Pipelined Model

```
#pragma HLS array_partition variable=< > complete dim=0

#pragma HLS inline

#pragma HLS inline off

#pragma HLS unroll

#pragma HLS inline

#pragma HLS pipeline
```

Baseline Model - Without pragmas

`imageScalar` module has latency 154111 ns.
viola$_f$*eaturecascade*

Utilization Estimates: 690 BRAM_18K, 35 DSP48E, 7834 flip-flops and 15193 LUTs.

Baseline Model - With pragmas

`imageScalar` module has latency 76834 ns.

Utilization Estimates: 690 BRAM_18K, 38 DSP48E, 8179 flip-flops and 16283 LUTs.

Pipelined Model - Without pragmas

`imageScalar` module has latency 154111 ns.

Utilization Estimates: 194 BRAM_18K, 38 DSP48E, 10563 flip-flops and 18275 LUTs.

Pipelined Model - With pragmas

`imageScalar` module has latency 76834 ns.

Utilization Estimates: 214 BRAM_18K, 45 DSP48E, 54434 flip-flops and 50528 LUTs.

Parallel and Pipelined Model - Without pragmas

`imageScalar` module has latency 154111 ns.

Utilization Estimates: 194 BRAM_18K, 76 DSP48E, 16694 flip-flops and 27389 LUTs.

Parallel and Pipelined Model - With pragmas

`imageScalar` module has latency 76834 ns.

Utilization Estimates: 214 BRAM_18K, 83 DSP48E, 56590 flip-flops and 58149 LUTs.

References

[1] N. K. Srivastava, S. Dai, R. Manohar, and Z. Zhang, “Accelerating face detection on programmable soc using c-based synthesis,” in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA ’17, (New York, NY, USA), p. 195–200, Association for Computing Machinery, 2017.

[2] P. Viola and M. Jones, “Robust real-time face detection,” in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, pp. 747–747, 2001.

[3] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

Thank You!