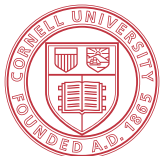


Accelerating Face Detection on Programmable SoC Using C-Based Synthesis

Nitish Srivastava, Steve Dai, Rajit Manohar, Zhiru Zhang

Computer Systems Laboratory
Electrical and Computer Engineering
Cornell University



Cornell University



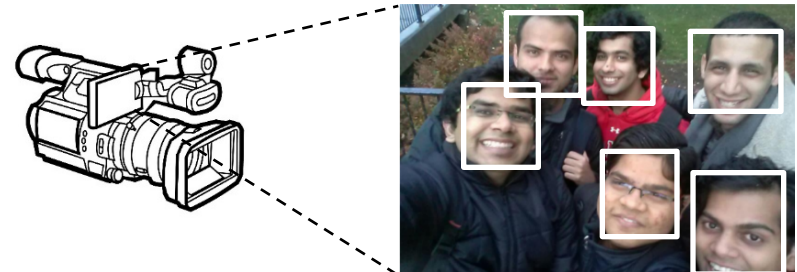
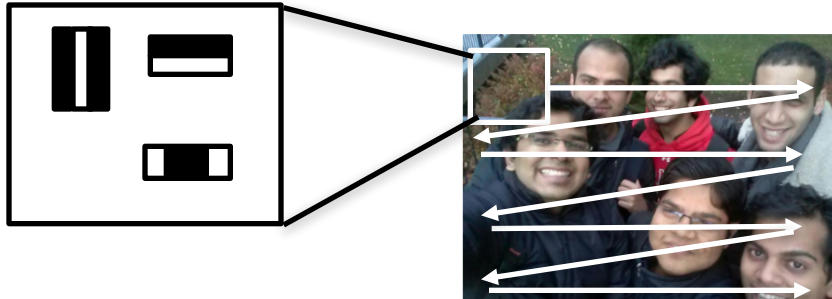
Summary

- ▶ **High Level Synthesis:** An emerging alternative to traditional register-transfer-level to improve design productivity
- ▶ **Context:** There is a lack of complex applications to benchmark FPGA high-level synthesis (HLS) tools
- ▶ **Case study:** Viola-Jones face detection
 - A **complex** and **reducible** benchmark
 - Has **realistic** performance constraint
 - Widely used in embedded applications (surveillance, photography, etc.)
- ▶ **Previous work:** To the best of our knowledge there is no existing open source RTL/HLS implementation
- ▶ **Our contributions**
 - Making a pure software code^[1] synthesizable and optimized for FPGAs
 - Real implementation/evaluation on FPSoC board
 - Open sourcing the design for the FPGA & HLS communities

[1] <https://sites.google.com/site/5kk73gpu2012/assignment/viola-jones-face-detection>

Viola-Jones: A Realistic and Reducible App

- ▶ It has **realistic** constraint
 - 30 fps for real-time image processing



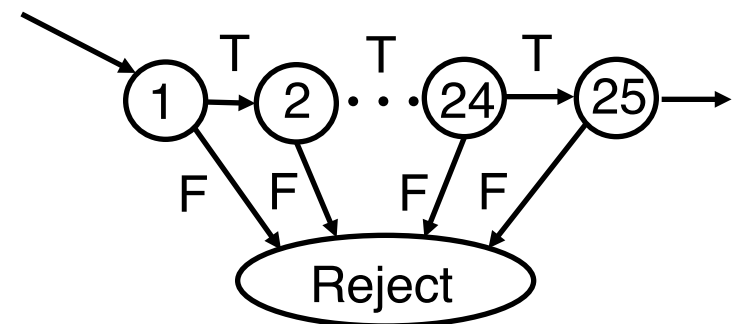
- ▶ It is **reducible**
 - composed of small kernel like

Integral Image Generation

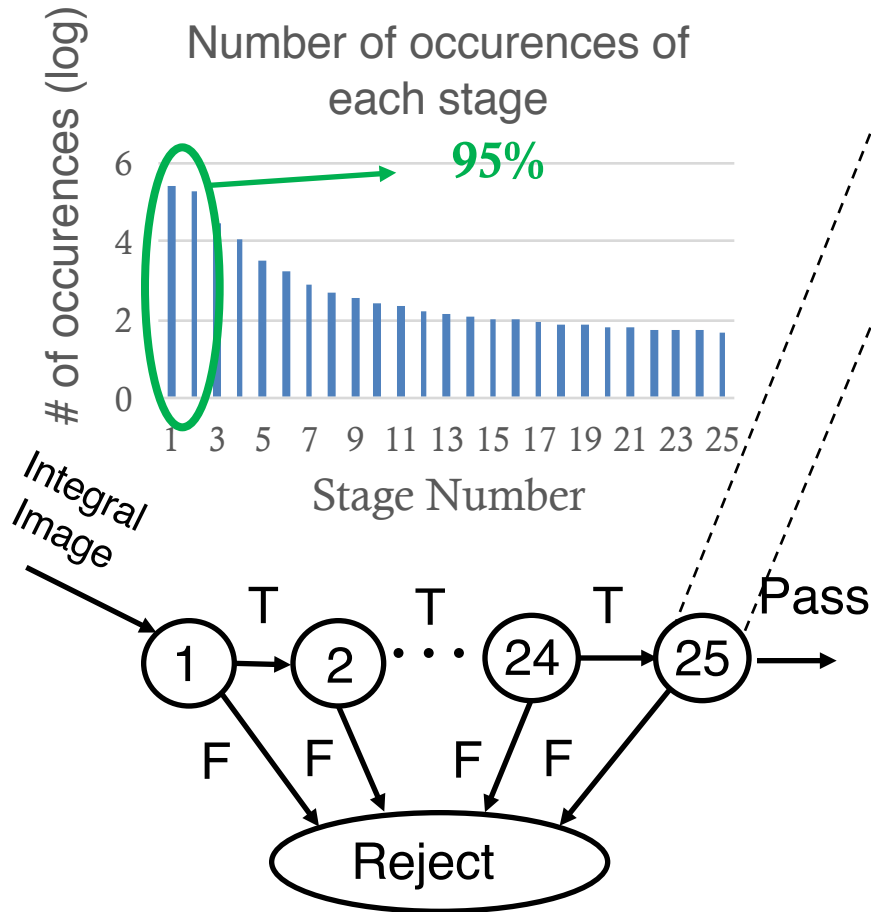
Image					Integral Image				
1	2	2	4	1	1	3	5	9	10
3	4	1	5	2	4	10	13	22	25
2	3	3	2	4	6	15	21	32	39
4	1	5	4	6	10	20	31	46	59
6	3	2	1	3	16	29	42	58	74

$1+5+3+2+5+4 = 20$
 $46 - 9 - 20 + 3 = 20$

Cascaded Classifier



Design Complexity in Parallel or Pipeline



2 X improvement !!

```

for ( i = 0; i < NUM_STAGES; i++ ) {
  stage_sum = 0;
  for ( j = 0; j < stages_array[i] ; j++ ) {
    // evaluate classifiers
    ....
  }
}
  
```

**Total 2,913 classifiers
- Can't unroll all**

```

for ( i = 0; i < 3; i++ ) {
  #pragma HLS unroll
  for ( j = 0; j < max(stages_array[0:2]) ; j++ ) {
    #pragma HLS unroll
    ....
  }
}
  
```

Parallelize first 3 stages

```

for ( i = 3; i < NUM_STAGES; i++ ) {
  stage_sum = 0;
  for ( j = 0; j < stages_array[i] ; j++ ) {
    #pragma HLS pipeline
    ....
  }
}
  
```

Pipeline rest of stages

Design Complexity in Memory Banking

Hand-coded
window and line buffers



Store integral image
window
buffer in discrete registers

```

for ( i = 0; i < 25; i++ ) {
  ...
  for ( j = 0; j < stages_array[i] ; j++ ) {
    ....
    c[0] = IntImg[r0.y][r0.x];
    c[1] = IntImg[r0.y][r0.x + r0.w];
    ....
    c[10] = IntImg[pt.y + r2.y + r2.h][pt.x + r2.x];
    c[11] = IntImg[pt.y + r2.y + r2.h][pt.x + r2.x + r2.w];
    ....
  }
}

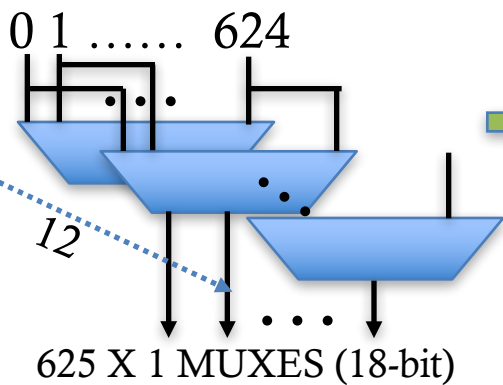
```

Memory addresses can't be
determined at compile time

Integral Image Banking

Synthesis
[FAILED TIMING]

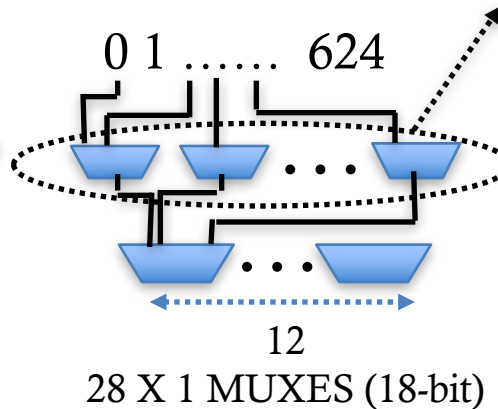
170K LUTs



28 18-bit MUXEs of
variable size
Max-size : 29 X 1

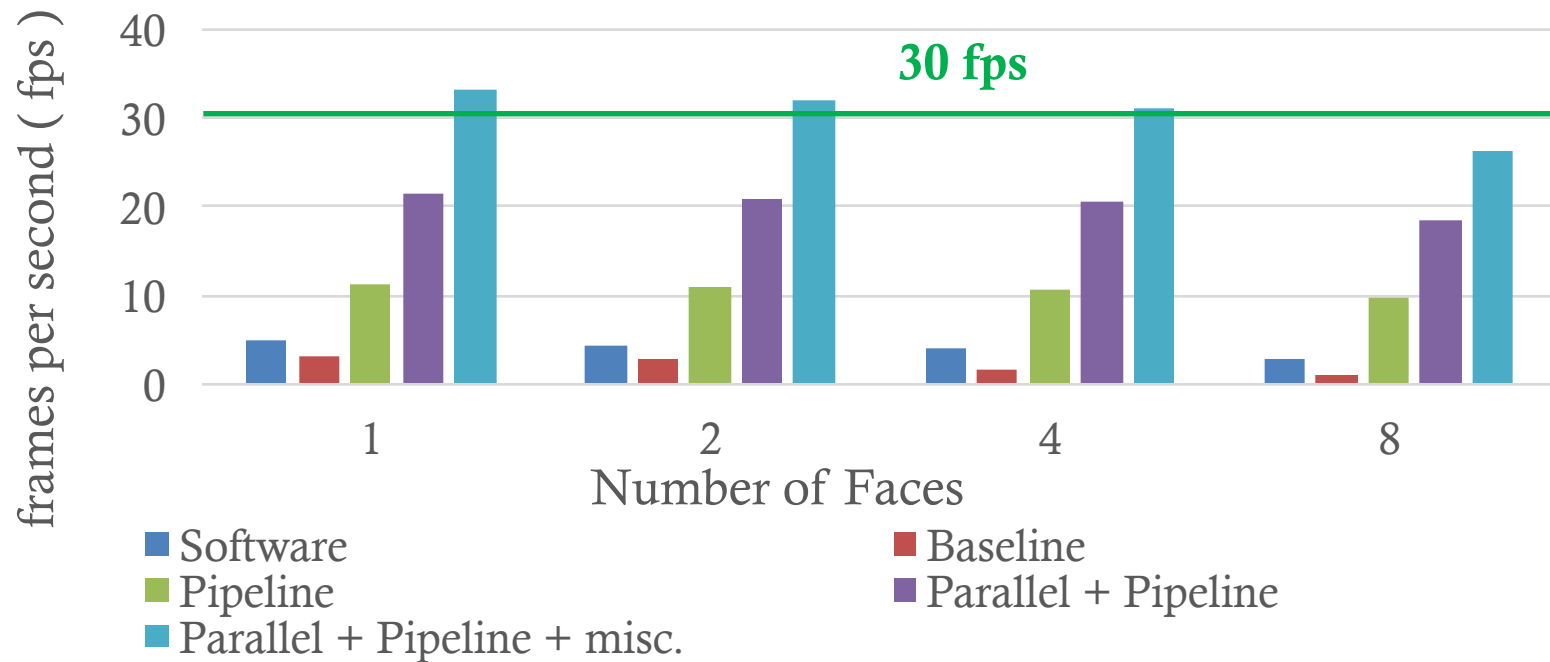
Synthesis
[TIMING MET]

16K LUTs



Implementation

<https://github.com/cornell-zhang/facedetect-fpga>



- ▶ ZC-706 board
 - Xilinx Zynq-7000 XC7Z045 FPGA
 - ARM Cortex-A9 CPU
- ▶ Xilinx SDSoC 2016.1
 - To generate data-motion network

Logic	Usage (%)
LUT	65,522 (29 %)
Registers	81,135 (19 %)
DSP48E	111 (12 %)
BRAM 18K	157 (29 %)