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
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

<table>
<thead>
<tr>
<th>Image</th>
<th>Integral Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 2 4 1</td>
<td></td>
</tr>
<tr>
<td>3 4 1 5 2</td>
<td></td>
</tr>
<tr>
<td>2 3 3 2 4</td>
<td></td>
</tr>
<tr>
<td>4 1 5 4 6</td>
<td></td>
</tr>
<tr>
<td>6 3 2 1 3</td>
<td></td>
</tr>
</tbody>
</table>

\[
1 + 5 + 3 + 2 + 5 + 4 = 20
\]

\[
46 - 9 - 20 + 3 = 20
\]
Design Complexity in Parallel or Pipeline

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

for (i = 3; 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

Parallelize first 3 stages

Pipeline rest of stages

2 X improvement!!

Number of occurences of each stage

95%

Number of occurences (log)

Stage Number

1 3 5 7 9 11 13 15 17 19 21 23 25

T T T F F

F F F F T

Reject
Design Complexity in Memory Banking

Hand-coded window and line buffers

Store integral image window buffer in discrete registers

Integral Image Banking

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

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

Synthesis [ TIMING MET ]
16K LUTs

Synthesis [ FAILED TIMING ]
170K LUTs

Integral Image Banking

Store integral image window buffer in discrete registers
Implementation

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

$$\text{frames per second (fps)}$$

30 fps

Number of Faces

<table>
<thead>
<tr>
<th>Logic</th>
<th>Usage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT</td>
<td>65,522 (29 %)</td>
</tr>
<tr>
<td>Registers</td>
<td>81,135 (19 %)</td>
</tr>
<tr>
<td>DSP48E</td>
<td>111 (12 %)</td>
</tr>
<tr>
<td>BRAM 18K</td>
<td>157 (29 %)</td>
</tr>
</tbody>
</table>

Software Pipeline Parallel + Pipeline Parallel + Pipeline + misc.

Software Pipeline Parallel + Pipeline Baseline

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