Hardware implementation of connected component labelling
Outline

• Overview
• Challenges/Considerations
• Objective description
• Component description
Overview

- A connected component labelling algorithm implemented in the FPGA, communicating through a device driver with the kernel to read and write the image pixel information.
- The output is a connected-component labeled image
Connected Component Labeling

- **Connected components.** - a region in a image within which there is an adjacency path between every pair of vertices

- **Connected-component labeling (CCL).** - algorithmic application to assign labels to connected components based on some heuristic

- In our case the heuristic we used is 6 adjacency which is water tight
Applications

• Includes many automated image analysis applications:
  • OCR
  • Medical image analysis
  • Pathogen detection in image processing
  • Object detection and localization
  • Pattern/Gesture/Facial recognition
Technique

• On the first pass:
  • Iterate through each element of the data by column, then by row (Raster Scanning)
  • If the element is not the background
  • Get the neighboring elements of the current element
  • If there are no neighbors, uniquely label the current element and continue
  • Otherwise, find the neighbor with the smallest label and assign it to the current element
  • Store the equivalence between neighboring labels

• On the second pass:
  • Iterate through each element of the data by column, then by row
  • If the element is not the background
  • Relabel the element with the lowest equivalent label
Algorithm

• Neighborhood & label resolution

<table>
<thead>
<tr>
<th>Top Left</th>
<th>Top</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>Current</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>o</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>L(New)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L(TL)</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>L(TL)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>o</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>L(CL)</td>
<td>L(CL)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>o</th>
<th>L(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>o</th>
<th>L(T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L(CL)</td>
<td>L(C)</td>
</tr>
</tbody>
</table>

**Equivalence**

L(T) \(!=\) L(CL) \(\Rightarrow\) LUT[Larger label] = Smaller label
L(C) = Smaller label
Proposed solution

• For the implementation of a CCL module and considering the previously mentioned limitations:
  • Implement a software code that decomposes the input image into pixel row pieces of information
  • Implement a device driver that writes and reads the pixel information to the FPGA through the Avalon MM bus
  • Implement a circular buffer that holds the pixel row information and its top neighbor for a sufficient amount of time to label it
  • Implement a set of modules that threshold the pixels, check neighbors, assign labels and update a label lookup table
Architecture/Block diagram
Main functionality

• Pixel information is obtained in the kernel
• The image row pixel information is sent to the FPGA and stored
• The first stage of labeling of the processing is conducted on each row and read back. The procedure is repeated until the full image is processed
• The kernel will initiate a second transfer of the image for label resolving with the pixels being written and read to and from the FPGA
Block/Module

- Kernel C Code
- Device Driver
- Avalon MM bus
- Circular buffer
- RAM
- Connected Component labeler
Avalon MM Bus

- Memory mapped interfaces
- Master (Device Driver) – Slave communication
- Implemented in Qsys
Circular Buffer

• Accounts for the limitations in FPGA memory size
• Handles pixel information writing and reading for 3 image rows to and from the device driver one row at a time
• Handles signaling for CCL modules: active and top row
• As rows are being processed, the software will be told to read out the labels and clear the circular buffer for new row pixels to be written into
RAM

• Dual-port, single-clock, 2048 byte-size memory locations

• Port A data and output bus is 4 byte wide to handle read/write operations from the device driver

• Port B data and output bus is 1 byte which operates with CCL module
Challenges/Considerations

- Limited memory capacity in the FPGA. - can not store a full image in it at once
- Limited Avalon bus width. - only 4 bytes can be sent through the bus at a time
- Architecting and implementing a multi-stage sequential algorithm on a FPGA
- Identifying source of errors: ex. Little Endian format used for RAM, Additional delay introduced in the read cycle because of the use of the dual port RAM
References

- http://ps4daily.com/2013/04/playstation-4-developers-didnt-know-about-8-gb-ram-until-reveal/