Canny Edge Detector
Verification Plan

Team Members:

Oluwatosin Adeosun, Sukhyun Hong, Eric Nielsen, Jiyuan Zhao

Design Specific Success Criteria

1. Demonstrate by simulation of a Verilog test bench that Gaussian block of the design performs the correct Gaussian convolution algorithm on the original image pixel data.
   (1pt)
2. Demonstrate by simulation of a Verilog test bench that the Gradient/Magnitude block of the design performs the correct gradient convolution algorithm and calculates the correct magnitudes based on the data provided by the Gaussian block.
   (1pt)
3. Demonstrate by simulation of a Verilog test bench that the Suppression block of the design correctly calculates pixel angles, compares the angles of neighboring pixels, passes out the appropriate values based on the data provided by the Gradient/Magnitude block.
   (1pts)
4. Demonstrate by simulation of a Verilog test bench that the Hysteresis block is able to accurately determine if pixels are/are not part of en edge using the defined thresholds and based on the data provided by the Suppression block.
   (1pts)
5. Demonstrate by simulation of a Verilog test bench that the complete design is able to produce an output bitmap image with the edges in the original image accurately detected.
   (4pts)

Verification Plan Summary

What to Verify	Design Module(s) involved	Verification Procedure Summary	DSSC(s) Proved	Use in Final Demo	Comments
Correct Gaussian blur algorithm achieved	Gaussian Block, 9x9 Buffer, 7x7 Buffer	Compare Gaussian results to alternate implementation	DSSC 1	Only if can't show using top level	Use MATLAB code to do Gaussian blur on identical input data and compare results
Correct gradient calculation algorithm achieved	Gradient/ Magnitude Block, 7x7 Buffer, 5x5 Buffer	Compare gradient results to alternate implementation	DSSC 2	Only if can't show using top level	Use MATLAB code to do gradient calculation on identical input data and compare results
Correct magnitude calculation algorithm achieved	Gradient/ Magnitude Block, 7x7 Buffer, 5x5 Buffer	Compare magnitude results to an alternate implementation	DSSC 2	Only if can't show using top level	Use MATLAB code to do magnitude calculation on identical input data and compare results
Correct suppression (edge-thinning) algorithm achieved	Suppression Block, 5x5 Buffer, 3x3 Buffer	Compare suppression results to alternate implementation	DSSC 3	Only if can't show using top level	Use Java code to do non-maximal suppression on identical input data and compare results
Correct hysteresis algorithm achieved	Hysteresis Block, 3x3 Buffer	Compare hysteresis results to alternate implementation	DSSC 4	Only if can't show using top level	Use MATLAB code to do hysteresis thresholding on identical input data and compare results
Correct overall functionality achieved	Top Level	Compare output image to image produced by alternate canny edge detection implementation	DSSC 5	Yes	Use Python script to send in bitmap image to SRAM and to turn output back into a bitmap image

Detailed Verification Test Breakouts

Correct Gaussian Blur Algorithm Achieved

- Highest Level of Design Module(s) involved:
  - Gaussian Block
- Test bench Expectations/Requirements:
  - Input data stored into 9x9 buffer
  - Initial Gaussian convolution performed on right most 3 columns (3x9 block)
  - On each clock cycle, data in the buffer is shifted by column
  - Also on each clock cycle, Gaussian convolution is performed on new data in the right most 3 columns
  - Output is 7 values (corresponding to middle 7 pixels in 3x9 block)
- No external or premade references are needed
- No pre/post processing is needed
- Main Verification Test Steps:
  1. Generate Gaussian output values for 9 shifts/calculations
  2. Capture these output values
  3. Generate Gaussian output values for 9 identical 3x9 blocks using MATLAB code
  4. Check that ASIC output values match MATLAB output values

Correct Gradient Calculation Algorithm Achieved

- Highest Level of Design Module(s) involved:
  - Gradient/Magnitude Block
- Test bench Expectations/Requirements:
  - Input data stored into 7x7 buffer
  - Initial gradient convolution performed on right most 3 columns (3x7 block)
  - On each clock cycle, data in the buffer is shifted by column
  - Also on each clock cycle, gradient convolution is performed on new data in the right most 3 columns
  - Output is 5 values for Gx and 5 values of Gy (corresponding to middle 5 pixels in 3x7 block)
- No external or premade references are needed
- No pre/post processing is needed
- Main Verification Test Steps:
  - Generate gradient output values (Gx’s and Gy’s) for 7 shifts/calculations
  - Capture these output values
  - Generate gradient output values (Gx’s and Gy’s) for 7 identical 3x7 blocks using MATLAB code
  - Check that ASIC output values match MATLAB output values

Correct Magnitude Calculation Algorithm Achieved

- Highest Level of Design Module(s) involved:
    - Gradient/Magnitude Block
- Test bench Expectations/Requirements:
    - Input data stored into 7x7 buffer
    - Initial magnitude calculations performed on right most 3 columns (3x7 block)
    - On each clock cycle, data in the buffer is shifted by column
    - Also on each clock cycle, magnitude calculations are performed on new data in the right most 3 columns
    - Output is 5 values for Gmag (corresponding to middle 5 pixels in 3x7 block)
- No external or premade references are needed
- No pre/post processing is needed
- Main Verification Test Steps:
    - Generate magnitude output values (Gmag) for 7 shifts/calculations
    - Capture these output values
    - Generate magnitude output values (Gmag) for 7 identical 3x7 blocks using MATLAB code
    - Check that ASIC output values match MATLAB output values

Correct Suppression Calculation Algorithm Achieved

- Highest Level of Design Module(s) involved:
    - Suppression Block
- Test bench Expectations/Requirements:
    - Input data stored into three 5x5 buffers
    - Initial suppression calculations performed on right most 3 columns (3x5 block) of the Gmag buffer, using angles determined by Gx and Gy buffer values
    - On each clock cycle, data in the buffers is shifted by column
    - Also on each clock cycle, suppression calculations are performed on new data in the right most 3 columns
    - Output is 3 values (corresponding to middle 3 pixels in 3x5 block of Gmag buffer)
- No external or premade references are needed
- No pre/post processing is needed
- Main Verification Test Steps:
    - Generate suppression output values for 5 shifts/calculations
    - Capture these output values
    - Generate magnitude output values for 5 identical 3x5 blocks using MATLAB code
    - Check that ASIC output values match MATLAB output values

Correct Hysteresis Algorithm Achieved

- Highest Level of Design Module(s) involved:
  - Hysteresis Block
- Test bench Expectations/Requirements:
  - Input data stored into 3x3 buffer
  - Hysteresis calculation performed on values in the buffer
  - On each clock cycle, data in the buffer is shifted by column
  - Also on each clock cycle, hysteresis calculation is performed on the new set of data in the buffer
  - Output is 1 values (corresponding to middle 1 pixel 3x3 buffer)
- No external or premade references are needed
- No pre/post processing is needed
- Main Verification Test Steps:
  - Generate hysteresis output value for 6 shifts/calculations
  - Capture these output values
  - Generate hysteresis output values for 6 identical 3x3 blocks using MATLAB code
  - Check that ASIC output values match MATLAB output values

Correct Overall Functionality Achieved

- Highest Level of Design Module(s) involved:
  - Total Design/Chip
- Test bench Expectations/Requirements:
  - Use Python script to store input bitmap image data into the off-chip SRAM
  - Send start signal to the chip
  - Wait for the system to complete calculations on all pixel data
  - Use Python script to convert output image data in SRAM into bitmap image
- Use http://matlabserver.cs.rug.nl/cgi-bin/matweb.exe as reference for Canny Edge correctness
- Python scripting pre/post processing is needed
- Main Verification Test Steps:
  1. Use chip to perform Canny Edge detection algorithm on input image
  2. Check output bitmap image with above gold reference model
  3. Repeat steps 1-2 for multiple input images