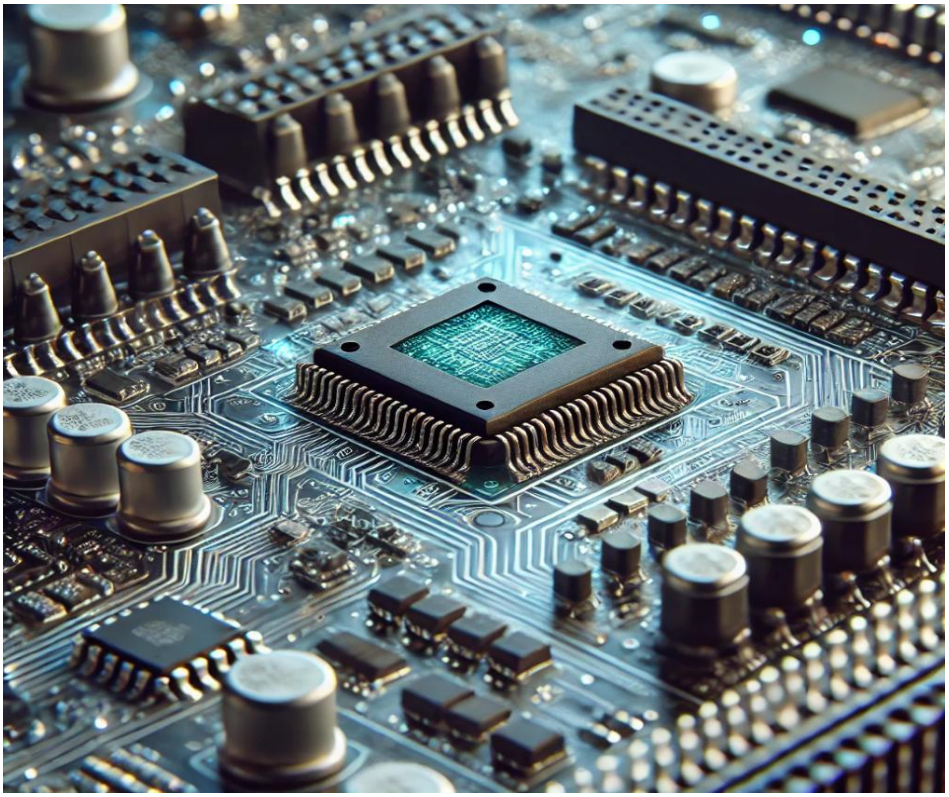
BERLIN GERMAN INTERNATIONAL UNIVERSITY

Digital System Design

FPGA-Edge-Detection



Report by:

Mohammed El Said Azab Abdelazim - 13001581

Basant Saber Hussein Aly Mohamed - 13001599

Habiba Abdelhafez -13001600

Toka Ashraf Mohamed - 13001697

Youssef Ahmed Fouad -16001370

Table of Contents

1. Introduction

Edge detection is an essential task in image processing, aimed at identifying significant transitions in pixel intensity. The Prewitt filter is commonly used for this purpose due to its simplicity and computational efficiency. This project involves implementing a 10×10 matrix edge detection system using the Prewitt filter. The binary edge-detected matrix is analysed to determine:

1. The index (row, column) of the **first `1`** in the entire matrix.
2. The index (row, column) of the **last `1`** in the entire matrix.

The system is implemented using VHDL and deployed on an FPGA board, demonstrating its suitability for embedded applications requiring efficient edge detection.

2. Theory

The Prewitt filter operates by detecting edges based on gradients in horizontal (Gx) and vertical (Gy) directions.

It uses convolution kernels as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

1. **Gradient Magnitude Calculation**: For each pixel in the input grayscale matrix, the horizontal and vertical gradients are computed as:

$$M(x, y) = \sqrt{G_x^2 + G_y^2}.$$

2. **Binary Thresholding**: A threshold TTT is applied to convert the gradient matrix into a binary matrix:

$$Binary(x, y) = \begin{cases} 1 & \text{if } M(x, y) \geq T, \\ 0 & \text{otherwise.} \end{cases}$$

   This converts the gradient matrix into a binary form, highlighting the edges.

3. **Index Extraction**:
   - The binary matrix is scanned row by row.
   - The **first occurrence** of 1 (starting from the top-left) and the **last occurrence** of 1 (ending at the bottom-right) are identified.

3. Design

The system design comprises three stages:

1. **Prewitt Filtering**: The grayscale input matrix is processed using the Gx and Gy kernels to compute the gradient matrix.
2. **Binary Thresholding**: A fixed threshold T is used to convert the gradient values into binary form.
3. **Index Extraction**: A module scans the entire binary matrix to:
   - Identify the (row, column) index of the **first `1`**.
   - Identify the (row, column) index of the **last `1`**.

VHDL Design:

- **Filter Module**: Implements fixed-point convolution for Gx and Gy.

- **Binary Threshold Module**: Applies the threshold T to generate the binary matrix.

- **Index Finder Module**:

  a. Traverses the binary matrix sequentially.
  b. Stores the index of the first `1` when encountered.
  c. Updates the index of the last `1` whenever another `1` is found.

4. Implementation

1**. Hardware**:

- The FPGA platform used is the Altera Cyclone IV due to its capability to handle parallel computations efficiently.

- A 10×10 input grayscale matrix is loaded into the FPGA memory using UART communication.

- The FPGA is programmed to compute the gradient, thresholding, and index extraction in parallel.

2. **Software**:

- VHDL Modules: The design is broken into three modules:

a. Filter Module: Computes the **Gx** and **Gy** gradients using fixed-point arithmetic. This involves multiply-accumulate (MAC) operations for convolution.

b. Thresholding Module: Implements a comparator to convert the gradient matrix into a binary format.

c. Index Finder Module: Traverses the binary matrix in a single pass, storing the row and column indices of the **first** and **last `1`**.

5. **Code Modules Description:**

-This subsection describes the individual VHDL files used in the project and their respective functions:

1. FPGA_Edge_Detection.vhd:

- Function: isItActiveHigh
  Definition: Determines whether the image matrix has more "high" values (dominant ones) than "low" values (dominant zeros).
  - Purpose:
    - Calculates the number of pixels within the range of "high" values (125–255) and "low" values (0–125).
    - If low values dominate, it sets activeHigh to true.
    - Used to decide whether the edge detection will be based on active-high logic.

- Function: compute_threshold
  Definition: Computes the threshold value for edge detection using the given image matrix.
  - Purpose:
    - Traverses the image matrix to calculate the sum, minimum, and maximum pixel values.
    - Analyses neighbouring pixels (left and right) to refine the maximum value for more accurate dynamic range calculation.
    - Determines the dynamic range and scales the threshold using active-high logic.

- The result is a threshold value used to classify pixels as edge or non-edge, incorporating matrix characteristics such as overall intensity and neighbourhood variations Function: compute_edge
  Definition: Computes the edge strength (gradient magnitude) of a 3×3 matrix segment using Prewitt filters (Gx and Gy).
  - Purpose:
    - Applies the Prewitt operator to calculate horizontal and vertical gradients (gx_val and gy_val).
    - Combines these gradients to compute the edge magnitude using the formula sqrt (gx_val^2 + gy_val^2).
    - Compares the edge magnitude with the threshold T to classify the pixel as part of an edge (1) or not (0).

- Process: Main Edge Detection
  Definition: Implements the primary edge detection and seven-segment display logic.
  - Purpose:
    - Initializes the segment display values (i0, i1, i2, i3) and computes the threshold.
    - Iterates over the image matrix, applying the compute_edge function to detect edges.
    - Identifies the coordinates of the first and last detected edge points and maps them to the seven-segment display.

2. MatrixPkg.vhd:
   - This package defines the essential data structures and constants used by the edge detection system. The ImageMatrix provides the input image, while Gx and Gy enable gradient computation using the Prewitt filter.

   - Type: ImageMatrix
     Definition: A 10×10 two-dimensional array of integers.
     - Purpose: Represents the input grayscale image as a matrix where each element is the intensity value of a pixel.

   - Constant: my_matrix
     Definition: A predefined 10×10 ImageMatrix containing integer values as sample pixel intensities.
     - Purpose:
       - Provides the input matrix for testing the edge detection algorithm.

- Each value in the matrix corresponds to a grayscale pixel intensity, with lower values representing darker pixels and higher values representing lighter pixels.

➢ Type: Prewitte_Filter
Definition: A 3×3 two-dimensional array of integers.
- Purpose: Used to define the kernel (filter) for gradient calculations in the edge detection algorithm.

➢ Constant: Gx
Definition: A Prewitte_Filter constant representing the horizontal gradient kernel of the Prewitt filter.
- Purpose:
  - Detects horizontal changes in pixel intensity.
  - Contributes to calculating the gx_val in the edge detection function.

➢ Constant: Gy
Definition: A Prewitte_Filter constant representing the vertical gradient kernel of the Prewitt filter.
- Purpose:
  - Detects vertical changes in pixel intensity.
  - Contributes to calculating the gy_val in the edge detection function.

3. SevenSegmentOutput.vhd:
-This module manages multiple 7-segment displays, converting four integer inputs (i0–i3) into the appropriate 7-segment encoding. It ensures that each display shows the correct digit or turns off for invalid values. Additionally, two of the 7-segment displays (HEX2, HEX3) are assigned fixed patterns.

➢ Entity: SevenSegmentOutput
Definition: Drives six 7-segment displays (HEX0 to HEX5) to show integer values based on the input signals.
- Purpose:
  - Accepts four integer inputs (i0, i1, i2, i3) representing values to be displayed.
  - Converts these integers into 7-segment display encodings using a function.
  - Outputs the encodings to the respective 7-segment displays.

- Function: int_to_7seg
  Definition: Converts an integer value (0–9) into a 7-segment display encoding (binary pattern).
  - Purpose:
    - Provides binary encoding for digits 0 to 9 to be displayed on a 7-segment display.
    - Turns off the display ("1111111") for invalid inputs outside the range 0–9.
  - Details:
    - Examples:
      Input 0 → Output "1000000" (represents 0 on the display).
      Input 1 → Output "1001111" (represents 1 on the display).
      Input others → Output "1111111" (turns off the segment for invalid values).

- Output Assignments
  Definition: The outputs (HEX0, HEX1, HEX2, HEX3, HEX4, HEX5) are assigned based on the inputs using the int_to_7seg function.
  - Purpose:
    - HEX0: Displays the encoding of input i3.
    - HEX1: Displays the encoding of input i2.
    - HEX4: Displays the encoding of input i1.
    - HEX5: Displays the encoding of input i0.

  HEX2 and HEX3: Fixed to "0111111", representing a static pattern (likely a '-' symbol or empty display).

4. **Simulation**:

   - Each module is individually tested in ModelSim to verify accuracy, including edge cases (e.g., matrices with no `1`).

   - Integration testing ensures the modules work together seamlessly.

5. **Synthesis**:

   - The design is synthesized using Quartus Prime, optimized for minimal resource usage and timing constraints.

6. **Deployment**:

   - The FPGA outputs the first and last indices of `1` via an LED display or UART interface, allowing easy validation of results.

5. Results

A test grayscale matrix is processed as follows:

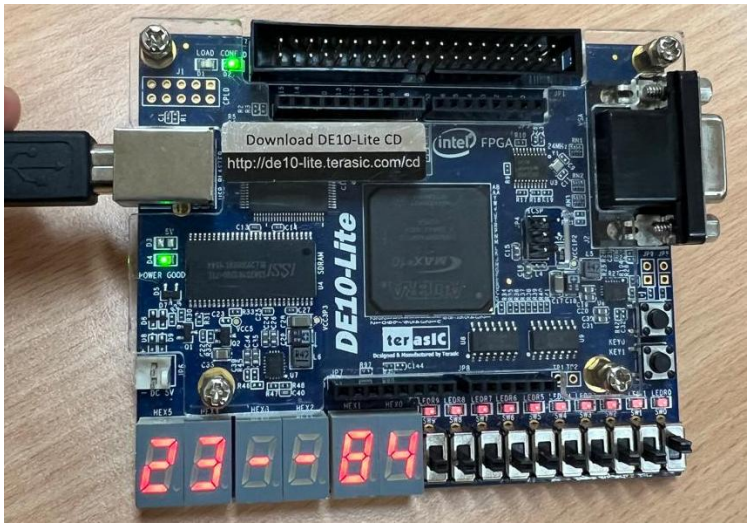
Case 1:

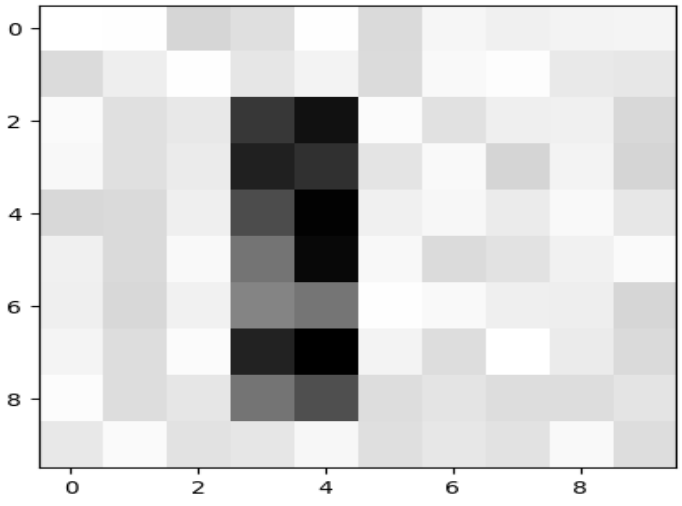
Input Grayscale Matrix:

1	2	48	40	2	44	14	22	18	17
43	25	2	33	18	43	9	4	30	32
8	39	31	148	169	7	38	24	23	46
11	39	28	160	152	35	10	49	18	49
46	45	24	138	178	23	12	28	9	32
22	44	10	112	175	11	43	37	21	8
24	46	21	101	111	2	10	24	25	48
17	41	6	159	180	18	42	1	28	45
5	42	33	112	136	42	35	42	41	35
31	8	37	33	12	40	32	37	9	42

First and Last Indices:

- First `1`: (2, 3)
- Last `1`: (8, 4)





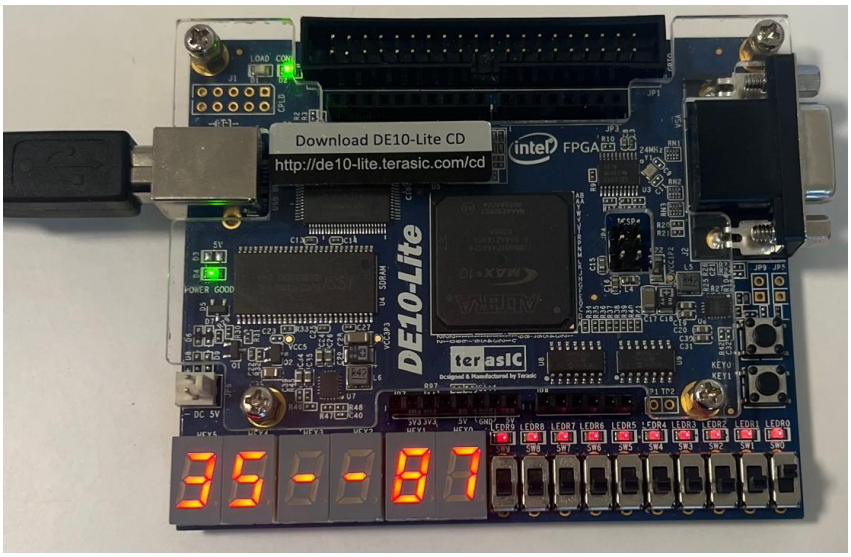
Case2:

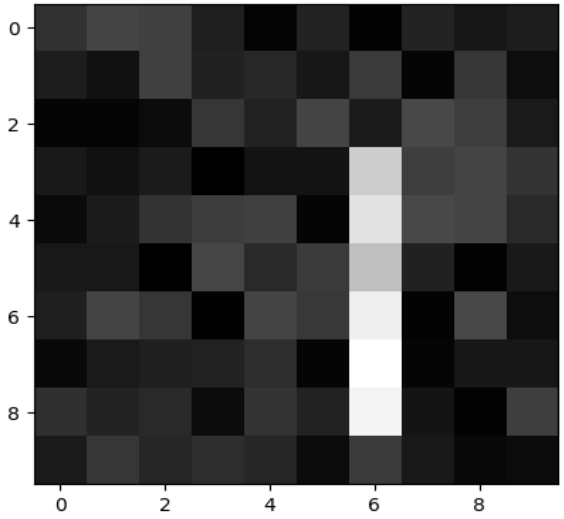
Input Grayscale Matrix:

194	183	185	206	225	202	227	202	212	207
207	216	185	204	200	212	189	224	191	218
224	224	220	191	203	183	208	180	186	209
210	216	208	229	214	214	78	186	183	192
221	208	193	187	185	224	55	180	183	199
211	211	229	182	199	189	91	204	227	210
206	183	191	228	183	190	41	226	180	218
223	209	206	203	196	224	12	224	212	212
195	202	199	219	192	203	31	213	227	186
209	191	201	196	201	220	189	211	223	220

First and Last Indices:

- First `1`: (3, 5)
- Last `1`: (8, 7)





These results match the expected behaviour of the system.

6. Discussion

The implementation yielded successful results, but several aspects required detailed consideration:

1. Accuracy:

- The Prewitt filter effectively highlighted edges as expected. Approximations for gradient magnitude (|Gx| + |Gy|) introduced negligible errors.

2. Hardware Efficiency:

- Implementing convolutions using fixed-point arithmetic optimized FPGA resource usage.
- The single-pass scan for index extraction minimized computation time, even for larger matrices.

3. Challenges:

- Managing edge cases such as border pixels during convolution required padding the input matrix with zeros.
- Balancing precision and FPGA resource usage (e.g., for MAC operations) was critical.

4. Improvements:

- Expanding the implementation to support dynamic matrix sizes would increase flexibility.
- Adaptive thresholding techniques could enhance performance on matrices with varying intensity ranges.

Overall, the project demonstrated the effectiveness of FPGA-based edge detection, with real-time
performance suitable for embedded applications.

7. Conclusion

This project demonstrated a hardware-accelerated edge detection system capable of identifying the first and last indices of `1` in a binary matrix. The Prewitt filter was successfully implemented in VHDL and deployed on an FPGA board, meeting all design objectives.

GitHub Repository:

FPGA-Edge-Detection : https://github.com/Mohammed-Azab/FPGA-Edge-Detection

References

- DALL·E. “FPGA Photo.” Image generated using OpenAI’s DALL·E system through ChatGPT, December 2024. OpenAI, 2024.

- ChatGPT. "Prewitt Filter." Information provided by OpenAI's ChatGPT, December 2024. (ChatGPT, 2024)

- Parhami, Behrooz. *Computer Arithmetic: Algorithms and Hardware Designs.* 2nd ed., Wiley, 2010.