Logic design project

(251)

INTRODUCTION

1. OBJECTIVE

- Practice and improve the ability to use Verilog HDL hardware specification language to design digital logic circuits.
- Practice skills in analyzing and designing digital circuits using a hierarchical model approach: functions, modules, etc. Enhance creativity in designing products by applying digital systems.
- Practice research skills, self-study, gain more knowledge about related fields: techniques in common digital electronic circuits, application fields of digital systems, etc.

2. REQUIREMENTS

- Groups of 3.
- Each group is assigned one topic. Midterm report results after 6 weeks, report and demo content includes:
  - Ideas: Analyze the ideas that the group plans to implement to meet the topic's requirements.
  - System specification: Defines the specific requirements of the system.
  - Design: Block diagram, functions of blocks according to the group's ideas.
  - Implementation: Demo of what the group has done.
  - Difficulty and solution.
- Final report: Submit directly on BKEL after completing the demo for the supervisor. The report must include the following content:
  - **Chapter 1**: Introduction. Includes content about topic introduction, tools used, equipment used, and product functions.
  - **Chapter 2**: Background knowledge about the product, features applied to the product in practice.
  - **Chapter 3**: Design. Includes block diagrams, functional blocks, and functional descriptions of the blocks.

  - **Chapter 4**: Implementation. This includes how the design is implemented (Block diagrams, circuit structures, design models, etc.). Flow charts can be drawn to show the system's processing flow.
  - **Chapter 5**: Conclusion. Includes an overview of the results and summary of the team's work, future directions, challenges encountered, and a breakdown of work.
- For specific requirements in the topics, students can propose changes to improve ease of use, ease of feature expansion, suitability with group design, etc. Students discuss with the supervisor if there are any changes.

3. EVALUATION CRITERIA

About the product:

- Essential functions on the requirement.
- Simple user interface, ease of use,... (UX – User Experience).
- Advanced functions, additional suggestions from the team.

Technically:

- Reasonable division of functional blocks
- Good source code (does not violate hardware specifications, coding style, etc.)

4. REFERENCES

- Lecture slides.
- Practice exercises, instructional materials.
- Internet

Topic

Table of contents

1. List of project topics

1.1. Image Edge Detection using Sobel Operator

Requirements: Implement a Sobel operator in Verilog to detect edges in grayscale images. The design reads pixel data from memory, applies convolution masks, and writes the processed edge data back to memory.

- Input: Grayscale image stored in memory.
- Output: Edge map saved back to memory.
- Algorithm: Apply Sobel operator (horizontal + vertical masks) to compute gradient magnitude.

1.2. Image Smoothing with 2D Convolution (Gaussian Blur)

Requirements: Design a Gaussian blur module in Verilog to reduce image noise and detail. The system fetches image blocks from memory, performs weighted convolution, and saves the smoothed image.

- Input: Grayscale image stored in memory.
- Output: Smoothed (blurred) image written back to memory.
- Algorithm: Perform 2D Gaussian convolution using a sliding window (e.g., 3×3, 5×5).

1.3. Image Binarization using Otsu's Thresholding

Requirements: Build a Verilog module that performs automatic image thresholding. The system computes the histogram, determines the optimal threshold using Otsu's method, and generates a binary (black/white) image.

- Input: Grayscale image from memory.
- Output: Binary image (black/white) stored in memory.
- Algorithm: Compute histogram of pixel intensities, determine optimal threshold using Otsu's method, apply thresholding.

1.4. Image Compression using Run-Length Encoding (RLE)

Requirements: Implement an image compression accelerator in Verilog. The system scans the image from memory, encodes consecutive pixels of the same value, and stores the compressed data back into memory.

- Input: Binary image stored in memory.

- Output: Compressed data stream written into memory.
- Algorithm: Implement Run-Length Encoding for sequences of identical pixels.

1.5. Morphological Operations for Binary Images

Requirements: Design a Verilog system to perform image morphology (erosion and dilation) on binary images stored in memory. This is useful for noise removal and object extraction in image preprocessing.

- Input: Binary image from memory.
- Output: Processed image after erosion/dilation.
- Algorithm: Implement morphological erosion and dilation using a structuring element (3×3).

1.6. Image Rotation and Mirroring

Requirements: Design a hardware block to rotate an image by 90° (clockwise/counter-clockwise) and to mirror it horizontally or vertically. The design fetches pixels from memory, computes new coordinates, and stores the transformed image back.

- Input: Grayscale image stored in memory.
- Output: Rotated (90° CW/CCW) or mirrored image saved back to memory.
- Algorithm: Perform coordinate remapping for each pixel ($x,y \rightarrow x',y'$).

1.7. Median Filtering for Noise Removal

Requirements: Implement a median filter in Verilog to remove salt-and-pepper noise from grayscale images. The system reads a sliding window of pixels from memory, sorts them, and writes the median value back.

- Input: Noisy grayscale image from memory.
- Output: Denoised image written back to memory.
- Algorithm: Use a 3×3 median filter (sort window values, output median).

1.8. Image Histogram Equalization

Requirements: Develop a Verilog module that enhances image contrast using histogram equalization. The design computes the cumulative histogram, generates a lookup table, and applies it to all pixels.

- Input: Grayscale image from memory.

- Output: Contrast-enhanced image stored in memory.
- Algorithm: Compute histogram, cumulative distribution (CDF), and remap pixels.

1.9. Template Matching for Object Detection

Requirements: Build a Verilog accelerator that searches for a small template image inside a larger image. The system uses normalized cross-correlation or sum of absolute differences (SAD).

- Input: Large grayscale image + small template image stored in memory.
- Output: Coordinates of best match (x,y) displayed on 7-segment or memory.
- Algorithm: Use Sum of Absolute Differences (SAD) or cross-correlation to compare template against all positions.

1.10. Color Space Conversion

Requirements: Implement a color converter in Verilog to transform RGB images into grayscale or YCbCr format for further processing. The design reads pixel triplets, applies linear transformation, and stores the converted data.

- Input: RGB image from memory.
- Output: Converted grayscale or YCbCr image written in memory.
- Algorithm: Apply linear transform:
  - Grayscale = 0.3R + 0.59G + 0.11B
  - YCbCr conversion matrix.

2. Advanced Requirements (Bonus)

Use your knowledge of HDL Logic Design and related subjects to deploy on the Arty Z7 FPGA development kit or equivalent.

- Integrate with a simple RISC-V core such as PicoRV32 or Ibex,...
- Implement at least two cores of RISC-V
- Capture image data through a camera module or USB camera
- Output to UART or HDMI display
- Achieve high resolution (720p or higher) via real-time streaming processing.
- Support storing image frames in memory.
- Provide integration with an existing SoC available on the FPGA board.
- ASIC: build on ASAP7 PDK, using Yosys and openLane

3. References

About FPGA Image Processing:

- R. Gonzalez and R. Woods, *Digital Image Processing*, Pearson, 4th Ed., 2018.
- R. Woods, J. McAllister, G. Lightbody, and Y. Yi, *FPGA-based Implementation of Signal Processing Systems*, Wiley, 2017.
- OpenCV Documentation