**A Distributed Optical Character Recognition System for IC Text Recognition**

1. **Executive Summary**

This report outlines the design and successful implementation of a distributed Optical Character Recognition (OCR) system aimed at Integrated Circuit (IC) text recognition. The system adopts a client-server architecture to intelligently distribute processing tasks between a PYNQ embedded board and a host PC. The PYNQ board handles real-time video capture, user-triggered image capture, and image preprocessing, while the computationally demanding OCR process is offloaded to a more powerful host PC running PaddleOCR. This approach achieves a balanced workload distribution, offering a responsive, real-time solution for high-accuracy text extraction from embedded systems or constrained edge devices.

1. **System Architecture**

The system comprises two principal components communicating via a TCP/IP socket connection, with each playing distinct roles:

* 1. **Client: PYNQ Board**

Data acquisition and preprocessing unit.

**Functionalities**:

* Initializes and streams live video using the onboard camera.
* Monitors BTN0 as a user-trigger mechanism.
* Captures frames upon user input.
* Applies a multi-stage image preprocessing pipeline to enhance OCR readiness.
* Serializes and transmits the processed frame to the server.
  1. **Server: Host PC**

Central processing and OCR analysis unit.

**Functionalities**:

* Initializes a TCP server socket to accept client connections.
* Deserializes the incoming image data.
* Feeds the image into the PaddleOCR engine for text extraction.
* Displays results with confidence scores in the terminal.
* Appends output to a persistent file ocr\_results.txt.

1. **Operational Workflow**

The system functions through the following sequenced operations:

* The server PaddleOCR\_PC.py is executed, initializing the PaddleOCR engine and listening on port 5002.
* The PaddleOCR\_PYNQ.ipynb notebook is run on the PYNQ board. And, it connects to the PC server using a predefined IP and port.
* The onboard camera provides a real-time video feed displayed in the Jupyter notebook interface.
* The user presses BTN0, prompting the PYNQ to capture and preprocess the current frame.
* The frame undergoes grayscale conversion, filtering, contrast normalization, and adaptive thresholding.
* The processed image is serialized via pickle and sent to the host server.
* The server reconstructs the image and applies PaddleOCR to extract text and confidence values.
* The recognized text is printed in the terminal and stored in ocr\_results.txt.
* The session continues until manually interrupted. A zero-byte signal is used to indicate connection closure.

1. **Setup and Execution Guide**
   1. **Prerequisites**

**Host PC:**

* Python 3.13.
* Libraries: paddleocr opencv-python numpy matplotlib (Install using pip install).

**PYNQ Board:**

* PYNQ development environment
* Required libraries: pynq, opencv-python, numpy
  1. **Configuration Steps:**

Ensure both devices share the same local network.

* Find PC’s local IPv4 address via ipconfig (Windows) or ifconfig/ip addr (Linux/macOS).
* Update the HOST variable in the PYNQ notebook:

HOST = '192.168.137.1’ # Replace with actual PC IP

* 1. **Execution Steps:**

1. Run PaddleOCR\_PC.py (Output: Listening on 5002...)
2. Open and execute all cells in the PYNQ Jupyter notebook.
3. View the live camera feed.
4. Press BTN0 to capture a frame.
5. OCR results are shown on the PC and saved to a .txt file.
6. **Analysis and Evaluation**
   1. **Strengths**

* **Efficient Task Offloading**: The system smartly separates real-time capture from OCR processing, utilizing each device’s strengths.
* **Advanced Preprocessing**: The preprocessing logic dynamically adjusts based on image statistics (contrast, variance, edge density).
* **Tactile Interface**: Physical button input offers a seamless user experience.
* **Well-Documented**: Clear setup and usage instructions in Readme.txt and the notebook.
  1. **Areas for Improvement**
* **Hardware Acceleration**: Offload preprocessing to FPGA logic blocks for real-time performance boosts.
* **Two-Way Communication**: Enable feedback to PYNQ for OCR result display or system status reporting.
* **Network Robustness**: Add socket-level error handling, timeouts, and retry mechanisms.
* **Graphical User Interface**: Enhance server-side usability with a simple GUI for image preview and result display.

1. **Conclusion**

This project stands as a robust and intelligent implementation of a distributed OCR system, effectively bridging edge computing with high-performance server-side processing. By integrating real-time image acquisition, adaptive preprocessing, and advanced OCR via PaddleOCR, it demonstrates both technical efficiency and usability. The system holds potential for a wide range of applications, especially in industrial inspection, PCB/IC text decoding, and field-deployed OCR scenarios. The modular design and clear workflow make it an excellent foundation for future enhancements, including FPGA acceleration, GUI integration, and broader text recognition applications.