**FPGA Application Documentation**

**Introduction**

This FPGA application demonstrates a system designed to manage switches, LEDs, and BCD (seven-segment) displays. The system includes multiple FreeRTOS tasks for efficient multitasking and communication through queues, showcasing the integration of hardware control and real-time software.

**Objectives**

* Monitor and display the states of switches and LEDs.
* Use BCD encoding to drive seven-segment displays.
* Demonstrate real-time communication between tasks using FreeRTOS queues.

**System Architecture**

**Overview**

The system consists of:

* **Switch Input Monitoring**: Reads the states of switches connected to GPIO pins.
* **LED Control**: Updates LEDs based on switch states.
* **BCD Display**: Outputs numerical values to seven-segment displays.
* **UART Communication**: (Optional) Transmits combined data to a serial console for debugging.

**Components**

* **FPGA**: Xilinx XC7S50-CSG324A Spartan 7 FPGA.
* **Peripherals**:
  + GPIO for switches and LEDs.
  + GPIO for seven-segment displays.
  + UART for serial communication.

**Diagram**

+----------------+ +----------------+ +----------------+

| Switch GPIO | --> | LED Controller | --> | BCD Display |

+----------------+ +----------------+ +----------------+

\ \ \

\ \-----------------> UART

\--------------------------------> Task Scheduler

**Requirements**

**Hardware**

1. Xilinx XC7S50-CSG324A Spartan 7 FPGA
2. Switches (5 inputs)
3. LEDs (3 outputs)
4. Seven-segment displays (2 units)
5. UART console (optional)

**Software**

1. Vivado Design Suite
2. FreeRTOS kernel
3. Xilinx SDK

**Setup**

1. **Clone the Repository**:
2. git clone https://github.com/your-repo/fpga\_application.git
3. cd fpga\_application
4. **Open in Vivado**:
   * Open the Vivado Design Suite.
   * Import the project files.
5. **Generate Bitstream**:
   * Synthesize the design.
   * Implement the design.
   * Generate the bitstream.
6. **Load on FPGA**:
   * Use a JTAG cable to program the FPGA.
   * Confirm successful programming.

**Tasks and Processes**

**Process 1: Switch Monitoring**

* Reads the states of five switches connected to GPIO pins.
* Sends the switch states to a queue for other processes.

**Key Code**:

switchRawState = XGpio\_DiscreteRead(&SwitchGpio, 1);

for (int i = 0; i < 5; i++) {

switchStates\_local.switchState[i] = (switchRawState >> i) & 0x1;

}

xQueueSend(xQueue1, &switchStates\_local, portMAX\_DELAY);

**Process 2: LED Control**

* Reads switch states from the queue.
* Updates the LED states accordingly.

**Process 3: BCD Display Output**

* Receives numerical values and encodes them for the seven-segment displays.

**Process 4: UART Communication**

* Combines switch states, LED states, and display values.
* Transmits a formatted message to the UART console.

**Testing and Debugging**

1. **Functional Testing**:
   * Verify switch inputs reflect correctly in LED outputs.
   * Ensure correct numerical values are displayed on the seven-segment displays.
2. **Debugging**:
   * Use UART output for real-time monitoring.
   * Check for queue overflows or task starvation.

**Conclusion**

This FPGA application successfully demonstrates hardware and software integration for switch, LED, and display management. Potential extensions include:

* Adding more switches and LEDs.
* Expanding to multiple seven-segment displays.
* Incorporating advanced communication protocols.