# I2C Master Implementation for FPGA

This project implements an I2C master on an FPGA using Verilog. The I2C master communicates with a slave device, performing read and write operations. Below is a detailed description of the project, including the methods used, challenges faced, and solutions implemented.

## Project Overview

The I2C master is designed to:

- Initialize communication with an I2C slave device.

- Perform single-byte write and read operations.

- Handle errors such as arbitration loss and acknowledgment failures.

- Transmit data over the I2C bus (SCL and SDA lines).

The project consists of the following modules:

1. `i2c_msb` : Top-level module for the I2C master.

2. `i2c_master_byte_ctrl` : Low-level module for handling I2C protocol operations (start, stop, read, write).

3. Testbench : Simulates the I2C master and verifies its functionality.

## Methods

### 1. I2C Protocol Implementation

- The I2C protocol was implemented using a state machine to handle start, stop, read, and write operations.

- The `i2c_master_byte_ctrl` module manages bit-level communication, including clock generation, data shifting, and acknowledgment handling.

### 2. State Machine Design

- The state machine in the `i2c_msb` module controls the flow of operations:

  - IDLE : Waits for a start command.

  - WAIT_SLAVE_ADDR : Waits for acknowledgment after sending the slave address.

  - SEND_WR_DATA : Sends data to the slave.

  - WAIT_WR_DATA : Waits for acknowledgment after sending data.

  - SEND_RD_DATA : Reads data from the slave.

  - WAIT_RD_DATA : Waits for acknowledgment after reading data.

  - CLEANUP : Resets the state machine after completing an operation.

3. Testbench

- A testbench was created to simulate the I2C master and verify its functionality.
- The testbench performs the following tasks:
  - Initializes the I2C master.
  - Simulates write and read operations.
  - Checks for correct data transmission and acknowledgment.

**Challenges and Solutions**

1. Missing Module (`i2c_master_byte_ctrl`)

- Challenge : Vivado reported that the `i2c_master_byte_ctrl` module was not found during synthesis.
- Solution : The module was implemented and added to the project. The implementation included a state machine for handling I2C protocol operations.

2. Unspecified I/O Standards (`DRC NSTD-1`)

- Challenge : Vivado requires all I/O ports to have a defined I/O standard (e.g., `LVCMOS33`).
- Solution : An XDC file was created to specify the I/O standards for all ports.

3. Unconstrained Logical Ports (`DRC UCIO-1`)

- Challenge : Vivado requires all I/O ports to have a defined physical pin location.
- Solution : Pin constraints were added to the XDC file, mapping each port to a specific pin on the FPGA.

4. Missing CFGBVS and CONFIG_VOLTAGE (`DRC CFGBVS-1`)

- Challenge : Vivado requires the configuration bank voltage select (`CFGBVS`) and configuration voltage (`CONFIG_VOLTAGE`) properties to be set.
- Solution : The properties were added to the XDC file, specifying the voltage level for the configuration bank.

5. Simulation Errors

- Challenge : The testbench initially failed to simulate due to incorrect signal assignments.
- Solution : The testbench was debugged, and signal assignments were corrected to match the expected behavior of the I2C master

**How to Use the Project**

1. Setup

1. Clone the repository or download the project files.

2. Open the project in Vivado.

3. Add the following files to the project:

   - `i2c_msb.v` (top-level module)
   - `i2c_master_byte_ctrl.v` (I2C byte controller)
   - `i2c_msb_TB.v` (testbench)
   - `i2c_msb.xdc` (constraints file)

2. Simulation

1. Open the testbench (`i2c_msb_TB.v`).

2. Run the simulation to verify the functionality of the I2C master.

3. Synthesis and Implementation

1. Run synthesis and implementation in Vivado.

2. Generate the bitstream and program the FPGA.

4. Testing on Hardware

1. Connect the FPGA to an I2C slave device (e.g., EEPROM, sensor).

2. Verify that the I2C master can communicate with the slave device.