Full Name: Mitta Veera Sushanth Reddy

Email:sushanthreddy635@gmail.com | Phone: +91 6281093293

LinkedIn: linkedin.com/in/veerasushanthreddymitta | Location: Kadapa, Andhra pradesh

# I2C Address Translator Project

## Objective

project implements an I²C address translator in Verilog that remaps physical I²C address 0x50 to virtual address 0x60, enabling protocol-level address resolution without hardware modification.

## Introduction

In digital design, particularly when working with I²C protocol models in Verilog, it is sometimes necessary to simulate or translate communication between masters and slaves that share the same physical address. While real-world I²C buses cannot support multiple slaves with identical addresses without contention, a protocol-level translation layer can be implemented in RTL to enable address remapping for verification, virtualization, or system integration purposes.

This project implements a simple I²C address translator in synthesizable Verilog. The design captures an incoming I²C transaction, remaps a predefined slave address (e.g., 0Xa0) to a virtual address (e.g., 0Xc0), and reissues the transaction using an internal I²C master. The system consists of three core blocks: a passive address monitor (i2c_slave_simple), a combinational address mapper (addr_mapper), and a basic I2C transmitter (i2c_master_simple), all integrated under the top-level module i2c_addr_translator_top.

This implementation is intended for simulation and FPGA-based protocol adaptation, demonstrating how address conflicts can be resolved at the logic level without altering physical devices or requiring external hardware.

## Architecture Overview

I designed an I²C Address Translator that sits between the master and slave device on the bus. It mainly consists of three blocks:

1. I2C Slave Module – receives data and address from the master.

2. Address Mapper – checks if the incoming slave address matches a specific one and changes it.

3. I2C Master Module – sends the modified address to the actual slave device.

All these modules are connected inside the top module (i2c_addr_translator_top). The design runs on FPGA and modifies only the address byte, not the data.

## FSM / Logic Explanation

I didn’t build a big custom FSM for this design. The slave and master modules already include small state machines.

• The slave module detects START condition, reads 8 address bits, and outputs them with a valid flag.
• The master module has four states: IDLE → START → SHIFT → STOP. In the SHIFT state, it toggles the SCL line and shifts each bit out, then generates a STOP condition.
• The address mapper is purely combinational—it just compares and replaces the address when needed.

During simulation in Vivado, when I sent 0xA0, the output became 0xC0, confirming the address translation worked correctly.

## Address Translation Working

The translation happens inside the addr_mapper.v file. It checks the 7-bit address part (bits [7:1]) of the incoming byte and replaces it if it matches the predefined address.

Example:
Input: 8'hA0 (binary 1010_0000) → address = 101_0000 (0x50).
If it matches 0x50, it is replaced with 0x60 → output becomes 8'hC0.
The R/W bit (LSB) stays unchanged using {SLAVE_ADDR_OUT, addr_in[0]}.

## Design Challenges Faced

While working on this project, I faced a few challenges and learned how to fix them:

• Understanding I2C timing and how START/STOP conditions are formed.

• Confirming that data is transferred MSB first as per I2C standard.

• Remembering to compare only bits [7:1] of the address, since bit[0] is R/W.

• Ensuring that simulation used delays only in testbench and RTL was fully synthesizable.

## Tools Used

• Verilog HDL
• Xilinx Vivado 2022.1
• Synopsys VCS (for simulation and debugging)
• Verdi (for waveform and signal analysis)
• FPGA: Artix-7
• Simulation: Behavioral + Waveform analysis

## Results and Verification

Simulation results in Vivado showed that when address 0xA0 was sent by the master, the translator changed it to 0xC0 before reaching the slave. This confirmed the proper working of the address mapping logic.

https://drive.google.com/file/d/11gRBKaB8LihQG2QbWBIpEGIYCdjaSWmj/view?

https://drive.google.com/file/d/1oq80wwFf4U6MvfYBjZkDAQPgW2qzeppa/view?usp=drive_link

https://drive.google.com/file/d/1ppVcs6Wwf_AaVqC2qyNJjpv9cZ0nuF7q/view?usp=drive_link

https://drive.google.com/file/d/1N54o3sjclr4v44gw3GPv_BDDM59Tmluo/view?usp=drive_link

https://drive.google.com/file/d/1Tia52orylfyPcVsnuSR5IrJPszNtKpI7/view?usp=drive_link

https://drive.google.com/file/d/1n30jjvUGQYhWj3cNt8o2r9X-t51kb4d-/view