# I2C Communication and Address Translator

## 1. Introduction to I2C Communication

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave, packet-switched, single-ended, serial communication bus. It uses only two lines: SDA (Serial Data) for data transfer and SCL (Serial Clock) for synchronization. Each device on the bus is identified by a unique address. Communication is initiated by the master device, while the slave responds according to the received address.

## 2. Working of Address Translator

In cases where two devices share the same physical I2C address, an address translator is required. The translator sits between the master and the slaves, allowing the master to communicate with multiple slaves that would otherwise conflict. The master uses virtual addresses, which the translator converts to the actual physical address of the target slave.

For example, if two devices both use the physical address 0x48, the translator can assign virtual addresses (e.g., 0x21 and 0x22) to distinguish them. When the master sends a message to virtual address 0x21, the translator rewrites the address to 0x48 and forwards it to Slave 1. Similarly, for virtual address 0x22, the translator forwards the message to Slave 2.

## 3. FSM Flow of the Code

The provided Verilog code implements a Finite State Machine (FSM) to achieve address translation. The key states are:

• IDLE: Waits for a START condition (SDA goes low).

• ADDR\_CAP: Captures 7 bits of the incoming address.

• TRANSLATE: Checks if the captured address matches a virtual address, replaces it with the physical address, and sets the target channel.

• DATA\_PASS: Forwards the subsequent data to the appropriate slave device (SDA1 or SDA2).

• STOP: Resets outputs and returns to IDLE.

## 4. Flowchart of FSM Operation

The flowchart below summarizes the FSM operation of the address translator:

* Start → IDLE → ADDR\_CAP → TRANSLATE → DATA\_PASS → STOP → IDLE

## 5. Code Explanation

The code uses registers to store the captured address and translated address. The SDA input from the master is monitored and shifted into the address register. Once the address bits are captured, the module compares them with predefined virtual addresses (0x21 and 0x22). If a match is found, the FSM replaces the virtual address with the physical address (0x48) and forwards the data to either SDA1 or SDA2 depending on the target device. If no match is found, the FSM transitions to the STOP state.

## 6. Brief Documentation

### 6.1 Architecture Overview

The architecture consists of an I2C Master, an FPGA-based Address Translator, and two I2C Slave devices. The master communicates using virtual addresses. The translator intercepts the communication, maps the virtual addresses to the physical address (0x48), and directs the data to the correct slave output line (SDA1 or SDA2). This ensures that multiple slaves with identical physical addresses can coexist without conflict.

### 6.2 FSM/Logic Explanation

The Finite State Machine (FSM) governs the operation of the translator. It includes the following states:  
• IDLE – Waits for start condition.  
• ADDR\_CAP – Captures address bits from the SDA line.  
• TRANSLATE – Compares the captured address with predefined virtual addresses and replaces it with the physical address.  
• DATA\_PASS – Passes subsequent data bits to the appropriate slave channel.  
• STOP – Resets the system and returns to IDLE.  
This step-by-step FSM approach ensures proper synchronization with the I2C protocol.

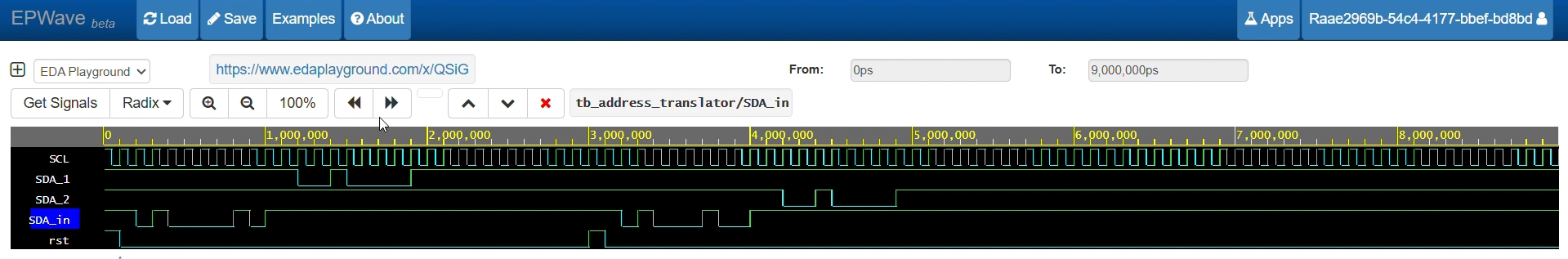
### 6.3 How Address Translation is Implemented

The translator maintains two predefined virtual addresses (0x21 and 0x22). When the incoming address matches one of these, it is replaced with the real physical address (0x48). The Read/Write (R/W) bit from the original transmission is preserved. The translated address and R/W bit are then forwarded to either SDA1 or SDA2 depending on which virtual address was detected.

### 6.4 Design Challenges Faced

Some of the challenges encountered in designing the address translator include:  
• Handling I2C timing: Ensuring that the translator introduces minimal latency while preserving synchronous operation with the SCL line.  
• Start/Stop detection: Correctly identifying start and stop conditions to synchronize the FSM.  
• Address bit shifting: Maintaining proper bit order while capturing and forwarding addresses.  
• Scalability: Extending support to more virtual addresses while keeping the FSM simple and efficient.

Output graph



Initially, when input is 0x21, the slave 1 data line is selected, and when it is 0x22, the slave 2 data line is selected, as the clock line is the same for both devices