ECE 526

Final project: I2C protocol

Jagrat Rao

5/15/21

Statement

For this project, my goal was to implement a I2C master module. The I2C is a two-wire serial bus protocol (like UART or SPI) with addressing capability. The two wires in interest are SDA (Serial Data); The line for the master and slave to send and receive data. SCL (Serial Clock); The line that carries the clock signal (as seen in figure 1). With I2C, data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame.

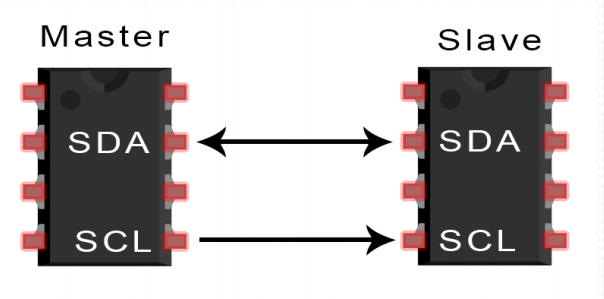


Figure 1 Block diagram for i2c protocol

Diagram, table

Description automatically generated

Figure 2 SDA line arrangement

Methodology

The bits were meant to have the following characteristics; Start Condition- The SDA line switches from a high to low before the SCL line switches from high to low. Read/Write Bit- A single bit specifying whether the master is writing data to the slave (active low) or reading data from it (active high). Address Frame- A 7- or 10-bit sequence unique to each slave that identifies the slave when the master wants to talk to it. ACK/NACK Bit- Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device (active high). Stop Condition - The SDA line switches from a low to high after the SCL line switches from low to high.

For simplification purposes only one data frame(as seen in figure 2) was used and SCL=system clk. Also, additions inputs were added to model slave module behaviours on the SDA line (i.e. start, stop, wr…)

Results

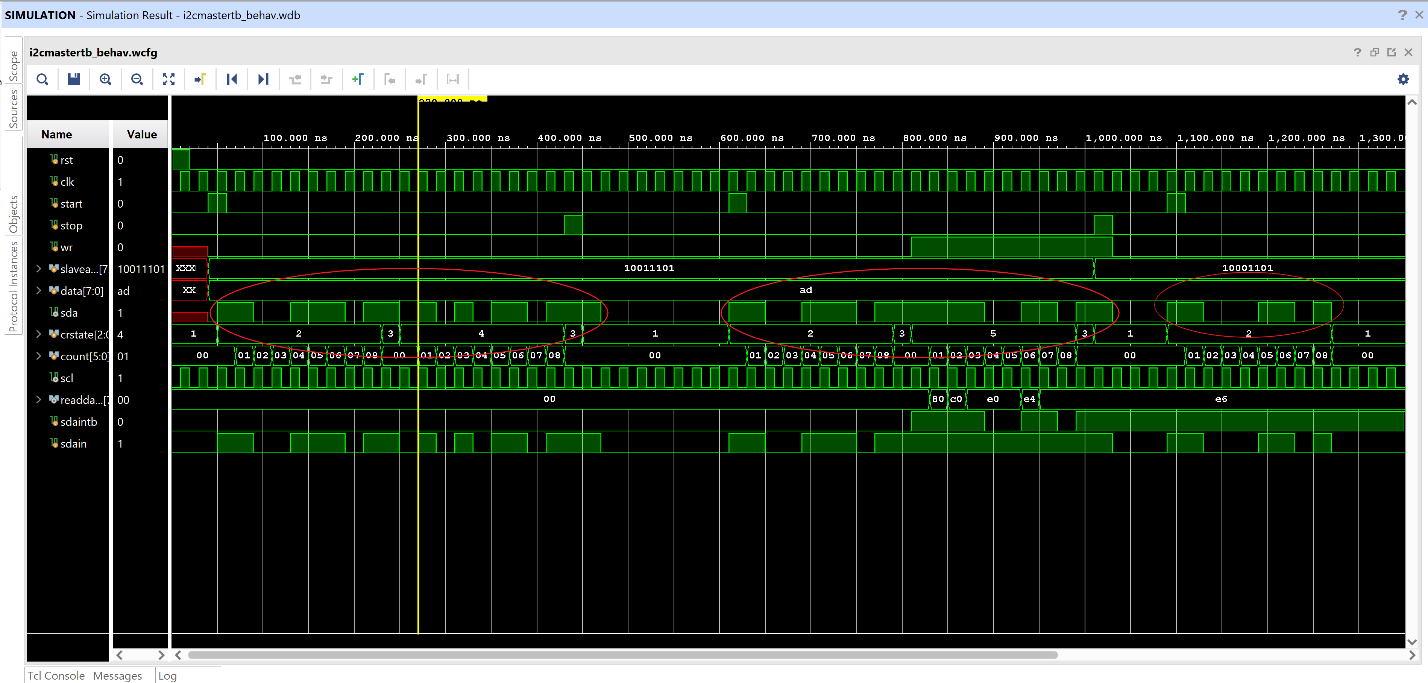


Figure 3- Simulation waveform for 3 messages.

As seen in figure above the first message (highlighted by the red circle) is a write operation. To describe the message we can see the following; the start bit is set to high, the address within the master (the salve address it is looking for) is put on the SDA line, we get the ACK bit to go high saying the addresses match, read/write is 0 indicating that it’s a write operation, the data is written to the SDA line and we get another ACK bit saying data has been written, lastly, we see the stop bit set to high indicating the message is over.

The same can be seen for message 2, but this time it’s a read operation. We see that the read/write bit is 0 indicating it’s a red operation and see the readdata line slowly get populated with the values read in. Finally, for message 3 we see that the slave address has changed and so after the address is put on the SDA, we get no ACK bit set since the addresses don’t match and the I2C module goes back to state one(idle).

Conclusion

In conclusion I was able to implement a basic version I2C master using states and an inout SDA port. A lot of debugging was required to gain the correct output.

y attest that this lab report is entirely my own work. I have not copied either code or text from anyone, nor have I allowed or will I allow anyone to copy my work. Name (printed) \_\_\_\_\_\_\_\_\_\_\_\_Jagrat rao\_\_\_\_\_\_\_\_\_\_\_\_ Name (signed) \_\_\_\_\_jagrat rao\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date \_\_2-25-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_