An Implementation of I2C Slave Interface Using Verilog HDL

Table of Contents:

1. Introduction:

The I2C Bus is a two wire serial communication protocol. One is SDA(Serial data) and the other is SCL(Serial Clock). The two wires in the I2C bus carry a 7 bit address, R/W control bit, and a data bit. The data (SDA) wire carries the data, while the clock (SCL) wire synchronizes the sender and receiver during transfer. Each device has its own unique address, and can work either as a transmitter or as a receiver. I2C master is the device that initiates communication and generates a clock for the same. Any device addressed by the master is called a slave. The I2C bus has three modes of operation: standard mode (0 to 100kbps), fast mode (0 to 400kbps), and high-speed mode (0 to 3.4mbps).

2. I2C Protocol Working:

Master will issue an 'Attention' signal to all of the connected devices known as START. Then the Master sends the ADDRESS of the device it wants to communicate with and a Read or a Write bit. All devices compare it with their own address and the one whose address is matched will produce a response signal as an Acknowledgement. If it doesn't match, they simply wait until the bus is released. On receiving acknowledgement, the master can start transmitting or receiving DATA. The transferred data is 8 bits long followed by an acknowledgement bit which is repeated till the whole transmission takes place. When all is done, the Master will issue the STOP condition.

2.1 General Characteristics:

2.1.1 Write Operation: It is done when the R/W bit is 0.

2.1.2 Read Operation: It is done when the R/W bit is 1.

2.1.3 Start and Stop condition: Start bit is indicated by the high-to-low transition of SDA while the SCL keeps high. And the stop bit is indicated by a low-to-high transition on the SDA line keeping SCL high and rest of the transitions of SDA line take place with SCL low.

2.1.4 Data Validity: The data on the SDA line is valid for the high period of clock pulse.

2.1.5 Transmitting a byte to a slave device: After the start condition, a byte transmitted will identify the slave on the bus and will select the mode of operation. If the R/W bit in the address was set to 0, transmission of byte is done.

2.1.6 Receiving a byte from slave device: Once the slave has been addressed and the slave has acknowledged this, a byte can be received from the slave if the R/W bit in the address was set to 1.

2.2 Operation Algorithm:

1. Master generates a START signal and all the slave devices listen to the bus.

2. Master sends a slave device ADDRESS + R/W bit.

3. Slave sends an ACKNOWLEDGE bit to the master if the address is matched.

4. The R/W bit is checked and the direction of data transfer is decided.

5. Slave sends 8-bit DATA to the master when R/W bit is 1 and receives 8-bit DATA from the master when R/W bit is 0.

6. An ACKNOWLEDGE signal is sent after receiving each 8-bit DATA by the receiver unit.

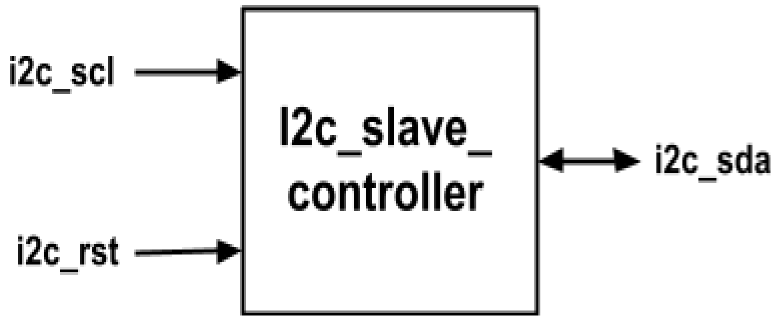
7. The DATA is repeatedly sent or received if there is no STOP signal.

8. The Master sends a STOP signal if data transmission is done.

3. I2C Slave Controller:

It is the top module which encapsulates a RAM and a FSM. Slave follows Master. The I2C module provides an interface to the I2C bus and the peripheral devices that are connected to the I2C bus.

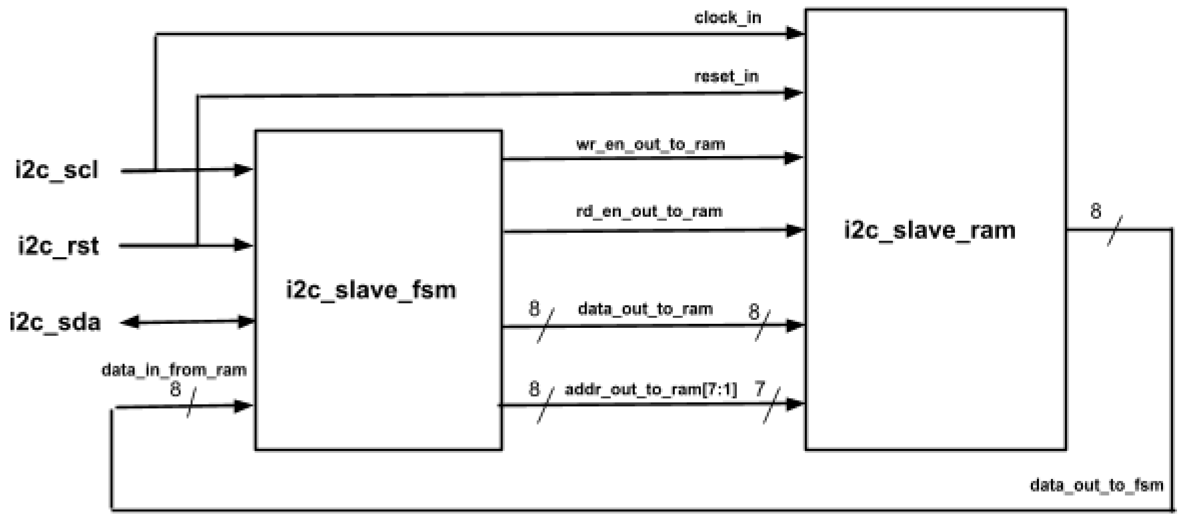
3.1 Block Diagram:



3.2 Port Declaration:

Signals	Declaration	Description
i2c_rst	input	Active high reset signal
i2c_scl	input	Serial clock line
i2c_sda	inout	Serial data line

3.3 Micro Architecture of I2C Slave Top:



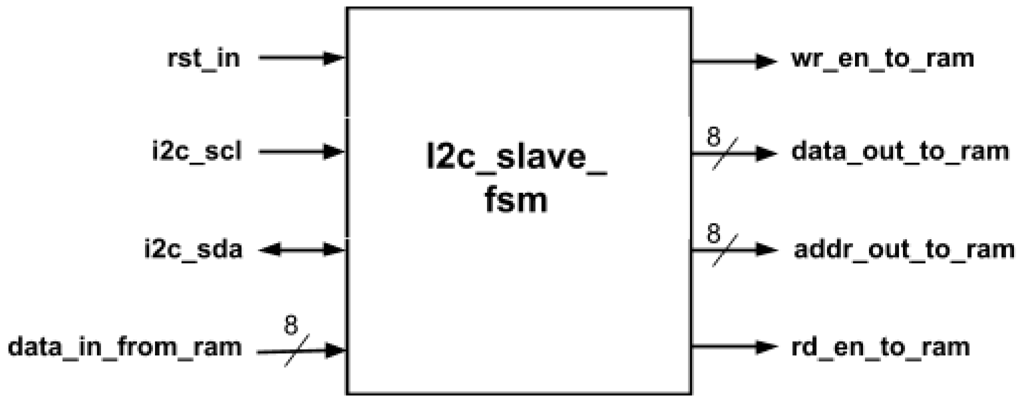
3.4 Functional Description:

The I2C slave module designed here supports the major features of I2C specification. It responds to the commands issued by an I2C master. The I2C slave acknowledges the successful receipt of address or data by pulling down the sda line during the ACK stage. Whenever write operation is occured Master writes 8 bit of data into Slave RAM memory and whenever read operation is occurred Master reads 8 bit of data from Slave RAM Memory.

4. I2C Slave FSM:

FSM is the main component of the I2C Slave Controller. The FSM is defined as a sequential circuit which uses the finite number of states to track the history of operations. Five states are present in the FSM to get the actual working of the I2C Slave Controller.

4.1 Block Diagram:



4.2 Port Description:

Signals	Declaration	Description
rst_in	input	Active high reset signal
i2c_scl	input	Serial clock line
i2c_sda	inout	Serial data line
data_in_from_ram	input	8 bit input data coming from data_out of RAM
wr_en_to_ram	output	After 8 clock pulses it goes to high and data_out_to_ram is written into RAM memory based on adrr value
rd_en_to_ram	output	If addr[0] value is high then this signal enables it to read data from RAM memory based on addr value.

data_out_to_ram	output	8 bit output data going to data_in of RAM.

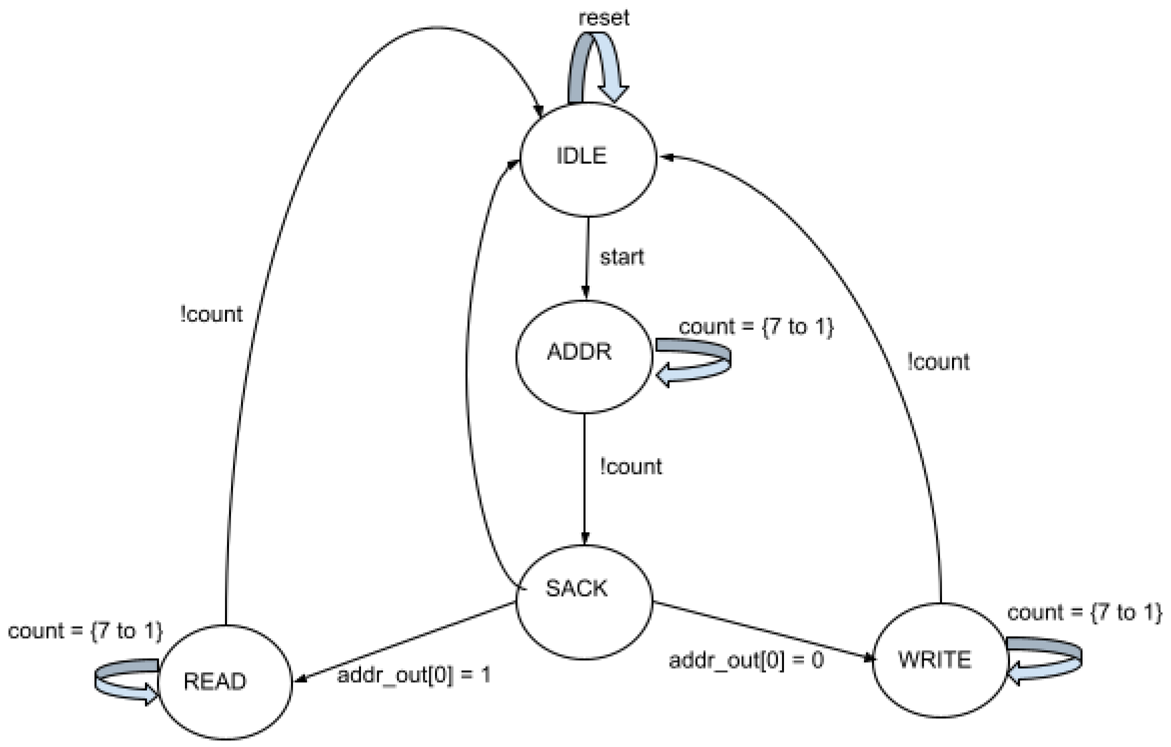
addr_out_to_ram	output	8 bit addr_out in which addr[7:1] going to the ram address and addr[0] is used for read and write operation.
		Addr[0] = 1, for read operation
		Addr[0] = 0, for write operation

4.3 Functional Description:

Algorithm:

Stage1: ***STATE_IDLE*** **:** I2C Bus does not perform any operation (i.e SCL and SDA remains high). If start(internal signal for start logic generation) is high it enters into the next state STATE_ADDR else it is in IDLE state..
Stage2: ***STATE_ADDR*** **:** In this ADDR state, Master sends the slave address serially to the slave through i2c_sda port. An internal counter is used to count no of bits transferred and as it becomes 0, it enters into the next state SEND_ACK and that 8 bits passed into adrr_out port.
Stage3: ***SEND_ACK*** **:** In this state Slave sends an acknowledgement bit to Master by pulling SDA line to low(here we have used Slave RAM so no need to check Slave_address).
If addr_out[0] is high then it enters into STATE_READ state and makes rd_en port high for one clock cycle.
Else if addr_out[0] is low then it enters into STATE_WRITE state.
Else state is in IDLE state.
Stage4: ***STATE_READ*** **:** In this state, 8 bit data is read from the Slave RAM by Master through i2c_sda line. After transmitting all the bits to master it enters into IDLE state.
Stage5: ***STATE_WRITE*** **:** In this state, 8 bit data is written into the Slave RAM by Master through i2c_sda line. Counter is used to count the no of bits, when it reaches 0, it enters into IDLE state and at that time wr_en port is high for one clock cycle. So the 1 Byte of data is written into RAM.

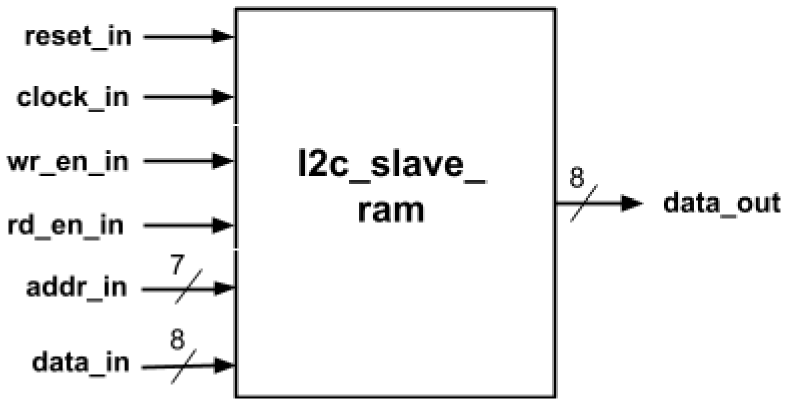
StateMachine:



5. I2C Slave RAM:

The width of the address is 7 bits and data is 8 bits for this design. Whenever wr_en is high it writes 8 bit data into RAM and whenever rd_en is high it reads 8 bit data from RAM.

5.1 Block Diagram:



5.2 Port Description:

Signals	**Declaration**	**Description**
reset_in	input	Active high reset signal
clock_in	input	Clock signal(serial clock signal i.e SCL)
wr_en_in	input	If high 8 bits of data is written into RAM
rd_en_in	input	If high 8 bits of data is read from RAM
addr_in	input	7 bits address signal, the location in which data is written into RAM or read from RAM
data_in	input	RAM input data width of 8 bits
data_out	output	RAM output data width of 8 bits

5.3 Functional Description:

The RAM designed in this project is dual port RAM. I have declared a reg_file of width 8 bits and depth 128.

Case1: Writing data into RAM
If wr_en_in is high then reg_file[addr_in] = data_in.
Case2: Reading data from RAM
If rd_en_in is high then data_out = reg_file[addr_in].