**CODE DESCRIPTION**

**Functions used:**

* **void i2c\_parse(ifstream & ,ofstream &) :**

ifstream and ofstream parameters are passed as reference to i2c\_parse function. This function includes verification of the file, reading from file and writing to the created file. Also, state machine function is called while reading each line of the file.

* **void addr\_transac\_details(int A\_D\_array[], string W\_R\_stat) :**

this function converts the address to its equivalent decimal value and sets the data operation bit to 0 if WRITE and 1 if READ. Moreover, print\_Addr() function is called.

* **void data\_transac\_details(int A\_D\_array[], string W\_R\_stat) :**

This function converts the data to its equivalent decimal value. Moreover, print\_Data() function is called.

* **void print\_Addr() :**

This function writes address’ equivalent hexadecimal value to the respective output file. Prints operation status i.e. MWR or MR to output file.

* **void print\_Data() :**

It prints hexadecimal value of received data of corresponding address.

* **void state\_mc(int line\_no,int sclk,int data\_in) :**

Following is the functioning of the state machine.

**STATE MACHINE FUNCTIONING**

**State 0:**

The initial state is the idle state where scl = 1 and sda = 1. As soon as scl = 1 and sda = 0, the code moves to the next state i.e. state 1, else, it stays idle.

**State 1:**

If scl = 1, the next state remains the same i.e. state 1, otherwise state 2 becomes the next state. When scl = 0, this is called as the start state.

**State 2:**

Machine remains in state 2 until scl = 0. The moment scl becomes 1, the state jumps from 2 to state 3.

**State 3:**

Now, scl = 1 i.e. the machine is in state 3. First 8 incoming bits are stored in an 8-bit temporary array.

**State 4:**

Here, the address is printed and after that the last bit of the array is used to analyze if it is a write or a read operation. If address counter (a\_cnt) is not equal to 0, state 2 will be the next state, else, it will jump to state 5.

**State 5:**

State 5 remains as present state until scl does not change to 1. If scl = 1, state 6 becomes the present state.

**State 6:**

After address bits are received, one verification bit is sent. This can be an ACK or a NACK. If NACK, transaction is complete, the next state is state 0, and NACK & transaction counters are incremented. If it is ACK, the next state is state 7 and ACK counter is incremented.

**State 7:**

Until scl = 0, it remains in state 7, else, it jumps to state 8.

**State 8:**

In state 8, incoming next 8 bits i.e. data bits, are stored in data array. Also, the data counter which was preinitialized is decremented by 1.

**State 9:**

While 8 bits are stored in data array in state 8, state 9 checks for data counter. If d\_cnt is not equal to 0, next state is state 7, else, next state is state 10 and the data is printed.

**State 10:**

If the received bit is ACK on the positive scl, then read/write counter is incremented according to the respective operation and next state is state 11. If same address receives multiple data, then print\_Addr() is called. Else, addr\_transac\_details(temp,op) and data\_transac\_details(Data,op) functions are called which convert address and data bits into their equivalent decimal value and then to their equivalet hexadecimal value.

If the received bit is NACK, next state is state 0. In addition to that NACK & transaction counters are incremented, and address & data counters are reset.

**State 11:**

If scl = 0, state remains the same i.e. state 11, else, it moves to state 12.

**State 12:**

Here it checks for two different conditions to be true at the same time. If scl = 1 and sda = 1, then the transaction is complete and it jumps to state 13, and transaction counter is incremented.

If scl = 0 and sda = 0, then next state is state 8 and d\_cnt is set to 7.

**State 13:**

If scl = 1 and sda = 1, both address and data count are reinitialized. Next state is state 0.