**Electrical and Computer Engineering Department**

**Portland State University**

**Spring - 2023**

**ECE 571 – Introduction to System Verilog**

**“Design and Verification of an I2C based memory subsystem ”**



**Batch 1:**

Divyasri Mandla

Goutham Kumar Reddy Palem

Naveen Babu Vanamala

Sai Sukitha Puli

Venkata Mohan Krishna Challa

Veera Venkata Sai Prasanna Korlapati

**Table of contents: -**

* 1.Objective.
* 2. Introduction.

1. I2C Protocol
2. How it work’s?

* 3. Design Implementation

1. Functional Unit.
2. I2C Interface.
3. Memory Controller and Memory.

* 4. Verification.
* 5. Code.
* 6. Results.

1. **Objective: -** The main objective is to design a reliable, effective memory system to verify an I2C (Inter-Integrated Circuit) communication protocol.
2. **Introduction: -**
3. **I2C Protocol:-**

I2C stands for **Inter-Integrated Circuit.**It is a bus interface connection protocol incorporated into devices for serial communication. In I2C, one device serves as the master and begins communication, while the other devices serve as slaves and carry out the master's orders. Multiple number of masters and slaves can coexist on the same bus because of the protocol's usage of a synchronous, multi-master, and multi-slave communication method.

1. **How I2C works:-**

I2C protocol uses only 2 bi-directional lines for data communication called Serial Data Line (SDA) and Serial Clock Line (SCL). While SCL provides the clock signal that synchronizes the data transmission, SDA is bidirectional and conveys the actual data that is being communicated between devices. Here, the data is transmitted in the form of packets and these packets are 7 or 9 bits long. For example, if we consider 9-bit long packets, out of which the first 8 bits are put in SDA line and the 9th bit is reserved for ACK/NACK i.e. Acknowledge or Not Acknowledge by the receiver.

The sequence of these bits are as follows–

The target device's 7or 9-bit slave address is sent by the master device after delivering a start condition to commence communication. The master can start reading or writing data to or from the slave device when the slave device accepts the address. Reading or writing a single byte or a stream of bytes is required for communication.

* Start Condition: The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.
* Stop Condition: The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to 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.
* Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
* 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.

In our situation, the functional unit and I2C interface function as a master and slave, respectively, while the memory controller serves as the slave with a single slave and master that can carry out fundamental functions.

1. **Design Implementation: -**

High-level representation of the circuit: -

A picture containing diagram, text, plan, technical drawing

Description automatically generated

The above block diagram represents the flow of the design and is as follows:

* The inputs are provided to the Functional unit which is a Shift Register, and it will generate the outputs.
* The I2C interface will collect the outputs generated by the Functional Unit, it transfers the address, wr\_rdn\_en and the collected outputs to the Memory Controller bit by bit through the SDA line.
* Based on the address, wr\_rdn\_en, data the Memory controller will either write the data into the Memory or read from it.

The following representation gives the detailed view of how the I2C and Memory controller have been implemented:

**Functional Unit FSM:**

A picture containing text, circle, screenshot, font

Description automatically generated

**I2C Interface FSM:**

A black background with white circles

Description automatically generated with low confidence

**Memory Controller FSM:**

A group of white circles with black text

Description automatically generated with low confidence

**Verification: -**

**Functional Unit:**

Used assertions to check for error when the outputs of functional unit are not becoming zero after reset and also, when the input is unknown.

Verified the functionality using golden model.

Used system Verilog constructs like Interfaces using modports, enum, always\_ff, always\_comb, tasks, assertions.

Used system Verilog constructs like tasks for displaying the inputs and outputs and debugging, emum,always\_ff,always\_comb.

Results after executing the file

|  |
| --- |
| --- All testcases are passed for ShiftRegister |

**I2C Interface:**

Used assertions to check for error when the START, STOP, RESET, M\_EN of I2C Interface when the start sequence, stop sequence, reset and master enable conditions are working according to the designed FSM for the Interface respectively.

Used tasks in testbench to test thoroughly each fsm state in the design and for displaying the inputs and outputs, and debugging.

Used system Verilog constructs like Interfaces using modports, enum, always\_ff, always\_comb, tasks, assertions.

Stimulus provided while execuiting:

// WRITE TESTING

ApplyWriteStimulus(1,2,3,1);

//read

ApplyReadStimulus(1,2,3,0);

Results after executing the I2C file:

|  |
| --- |
| # Collecting Address  # 65 bus.SDA\_OUT VALUE is 1 and temp\_addr[wait\_counter] = temp\_addr[0] = 1  # 75 bus.SDA\_OUT VALUE is 1 and temp\_addr[wait\_counter] = temp\_addr[1] = 1  # 85 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[2] = 0  # 95 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[3] = 0  # 105 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[4] = 0  # 115 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[5] = 0  # 125 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[6] = 0  #  # Collecting Data  # 155 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[0] = 0  # 165 bus.SDA\_OUT VALUE is 1 and temp\_data[wait\_counter] = temp\_data[1] = 1  # 175 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[2] = 0  # 185 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[3] = 0  # 195 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[4] = 0  # 205 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[5] = 0  # 215 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[6] = 0  # 225 bus.SDA\_OUT VALUE is 0 and temp\_data[wait\_counter] = temp\_data[7] = 0  #  # Collecting Address  # 325 bus.SDA\_OUT VALUE is 1 and temp\_addr[wait\_counter] = temp\_addr[0] = 1  # 335 bus.SDA\_OUT VALUE is 1 and temp\_addr[wait\_counter] = temp\_addr[1] = 1  # 345 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[2] = 0  # 355 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[3] = 0  # 365 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[4] = 0  # 375 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[5] = 0  # 385 bus.SDA\_OUT VALUE is 0 and temp\_addr[wait\_counter] = temp\_addr[6] = 0 |

**Memory Controller:**

Used assertions to check for error when the RESET, CLK has unknown values.

Used tasks in testbench to test thoroughly each fsm state in the design and for displaying the inputs and outputs, and debugging.

Used system Verilog constructs like Interfaces using modports, enum, always\_ff, always\_comb, tasks, assertions.

Results after executing the memory file by filling the memory with values from 0 to 127 and also with random data in the second case:

|  |
| --- |
| Filling with random Data  # bus.data\_out = 0  # bus.data\_out = 1  # bus.data\_out = 2  # bus.data\_out = 3  # bus.data\_out = 4  # bus.data\_out = 5  # bus.data\_out = 6  # bus.data\_out = 7  # bus.data\_out = 8  # bus.data\_out = 9  # bus.data\_out = 10  # bus.data\_out = 11  # bus.data\_out = 12  # bus.data\_out = 13  # bus.data\_out = 14  # bus.data\_out = 15  # bus.data\_out = 16  # bus.data\_out = 17  # bus.data\_out = 18  # bus.data\_out = 19  # bus.data\_out = 20  # bus.data\_out = 21  # bus.data\_out = 22  # bus.data\_out = 23  # bus.data\_out = 24  # bus.data\_out = 25  # bus.data\_out = 26  # bus.data\_out = 27  # bus.data\_out = 28  # bus.data\_out = 29  # bus.data\_out = 30  # bus.data\_out = 31  # bus.data\_out = 32  # bus.data\_out = 33  # bus.data\_out = 34  # bus.data\_out = 35  # bus.data\_out = 36  # bus.data\_out = 37  # bus.data\_out = 38  # bus.data\_out = 39  # bus.data\_out = 40  # bus.data\_out = 41  # bus.data\_out = 42  # bus.data\_out = 43  # bus.data\_out = 44  # bus.data\_out = 45  # bus.data\_out = 46  # bus.data\_out = 47  # bus.data\_out = 48  # bus.data\_out = 49  # bus.data\_out = 50  # bus.data\_out = 51  # bus.data\_out = 52  # bus.data\_out = 53  # bus.data\_out = 54  # bus.data\_out = 55  # bus.data\_out = 56  # bus.data\_out = 57  # bus.data\_out = 58  # bus.data\_out = 59  # bus.data\_out = 60  # bus.data\_out = 61  # bus.data\_out = 62  # bus.data\_out = 63  # bus.data\_out = 64  # bus.data\_out = 65  # bus.data\_out = 66  # bus.data\_out = 67  # bus.data\_out = 68  # bus.data\_out = 69  # bus.data\_out = 70  # bus.data\_out = 71  # bus.data\_out = 72  # bus.data\_out = 73  # bus.data\_out = 74  # bus.data\_out = 75  # bus.data\_out = 76  # bus.data\_out = 77  # bus.data\_out = 78  # bus.data\_out = 79  # bus.data\_out = 80  # bus.data\_out = 81  # bus.data\_out = 82  # bus.data\_out = 83  # bus.data\_out = 84  # bus.data\_out = 85  # bus.data\_out = 86  # bus.data\_out = 87  # bus.data\_out = 88  # bus.data\_out = 89  # bus.data\_out = 90  # bus.data\_out = 91  # bus.data\_out = 92  # bus.data\_out = 93  # bus.data\_out = 94  # bus.data\_out = 95  # bus.data\_out = 96  # bus.data\_out = 97  # bus.data\_out = 98  # bus.data\_out = 99  # bus.data\_out = 200  # bus.data\_out = 101  # bus.data\_out = 102  # bus.data\_out = 103  # bus.data\_out = 104  # bus.data\_out = 105  # bus.data\_out = 106  # bus.data\_out = 107  # bus.data\_out = 108  # bus.data\_out = 109  # bus.data\_out = 110  # bus.data\_out = 111  # bus.data\_out = 112  # bus.data\_out = 113  # bus.data\_out = 114  # bus.data\_out = 115  # bus.data\_out = 116  # bus.data\_out = 117  # bus.data\_out = 118  # bus.data\_out = 119  # bus.data\_out = 120  # bus.data\_out = 121  # bus.data\_out = 122  # bus.data\_out = 123  # bus.data\_out = 124  # bus.data\_out = 125  # bus.data\_out = 126  # bus.data\_out = 127  # memory[0] = 0  # memory[1] = 1  # memory[2] = 2  # memory[3] = 3  # memory[4] = 4  # memory[5] = 5  # memory[6] = 6  # memory[7] = 7  # memory[8] = 8  # memory[9] = 9  # memory[10] = 10  # memory[11] = 11  # memory[12] = 12  # memory[13] = 13  # memory[14] = 14  # memory[15] = 15  # memory[16] = 16  # memory[17] = 17  # memory[18] = 18  # memory[19] = 19  # memory[20] = 20  # memory[21] = 21  # memory[22] = 22  # memory[23] = 23  # memory[24] = 24  # memory[25] = 25  # memory[26] = 26  # memory[27] = 27  # memory[28] = 28  # memory[29] = 29  # memory[30] = 30  # memory[31] = 31  # memory[32] = 32  # memory[33] = 33  # memory[34] = 34  # memory[35] = 35  # memory[36] = 36  # memory[37] = 37  # memory[38] = 38  # memory[39] = 39  # memory[40] = 40  # memory[41] = 41  # memory[42] = 42  # memory[43] = 43  # memory[44] = 44  # memory[45] = 45  # memory[46] = 46  # memory[47] = 47  # memory[48] = 48  # memory[49] = 49  # memory[50] = 50  # memory[51] = 51  # memory[52] = 52  # memory[53] = 53  # memory[54] = 54  # memory[55] = 55  # memory[56] = 56  # memory[57] = 57  # memory[58] = 58  # memory[59] = 59  # memory[60] = 60  # memory[61] = 61  # memory[62] = 62  # memory[63] = 63  # memory[64] = 64  # memory[65] = 65  # memory[66] = 66  # memory[67] = 67  # memory[68] = 68  # memory[69] = 69  # memory[70] = 70  # memory[71] = 71  # memory[72] = 72  # memory[73] = 73  # memory[74] = 74  # memory[75] = 75  # memory[76] = 76  # memory[77] = 77  # memory[78] = 78  # memory[79] = 79  # memory[80] = 80  # memory[81] = 81  # memory[82] = 82  # memory[83] = 83  # memory[84] = 84  # memory[85] = 85  # memory[86] = 86  # memory[87] = 87  # memory[88] = 88  # memory[89] = 89  # memory[90] = 90  # memory[91] = 91  # memory[92] = 92  # memory[93] = 93  # memory[94] = 94  # memory[95] = 95  # memory[96] = 96  # memory[97] = 97  # memory[98] = 98  # memory[99] = 99  # memory[100] = 200  # memory[101] = 101  # memory[102] = 102  # memory[103] = 103  # memory[104] = 104  # memory[105] = 105  # memory[106] = 106  # memory[107] = 107  # memory[108] = 108  # memory[109] = 109  # memory[110] = 110  # memory[111] = 111  # memory[112] = 112  # memory[113] = 113  # memory[114] = 114  # memory[115] = 115  # memory[116] = 116  # memory[117] = 117  # memory[118] = 118  # memory[119] = 119  # memory[120] = 120  # memory[121] = 121  # memory[122] = 122  # memory[123] = 123  # memory[124] = 124  # memory[125] = 125  # memory[126] = 126  # memory[127] = 127  #  # Filling with random Data  # bus.data\_out = 36  # bus.data\_out = 129  # bus.data\_out = 9  # bus.data\_out = 99  # bus.data\_out = 13  # bus.data\_out = 141  # bus.data\_out = 101  # bus.data\_out = 18  # bus.data\_out = 1  # bus.data\_out = 13  # bus.data\_out = 118  # bus.data\_out = 61  # bus.data\_out = 237  # bus.data\_out = 140  # bus.data\_out = 249  # bus.data\_out = 198  # bus.data\_out = 197  # bus.data\_out = 170  # bus.data\_out = 229  # bus.data\_out = 119  # bus.data\_out = 18  # bus.data\_out = 143  # bus.data\_out = 242  # bus.data\_out = 206  # bus.data\_out = 232  # bus.data\_out = 197  # bus.data\_out = 92  # bus.data\_out = 189  # bus.data\_out = 45  # bus.data\_out = 101  # bus.data\_out = 99  # bus.data\_out = 10  # bus.data\_out = 128  # bus.data\_out = 32  # bus.data\_out = 170  # bus.data\_out = 157  # bus.data\_out = 150  # bus.data\_out = 19  # bus.data\_out = 13  # bus.data\_out = 83  # bus.data\_out = 107  # bus.data\_out = 213  # bus.data\_out = 2  # bus.data\_out = 174  # bus.data\_out = 29  # bus.data\_out = 207  # bus.data\_out = 35  # bus.data\_out = 10  # bus.data\_out = 202  # bus.data\_out = 60  # bus.data\_out = 242  # bus.data\_out = 138  # bus.data\_out = 65  # bus.data\_out = 216  # bus.data\_out = 120  # bus.data\_out = 137  # bus.data\_out = 235  # bus.data\_out = 182  # bus.data\_out = 198  # bus.data\_out = 174  # bus.data\_out = 188  # bus.data\_out = 42  # bus.data\_out = 11  # bus.data\_out = 113  # bus.data\_out = 133  # bus.data\_out = 79  # bus.data\_out = 59  # bus.data\_out = 58  # bus.data\_out = 126  # bus.data\_out = 21  # bus.data\_out = 241  # bus.data\_out = 217  # bus.data\_out = 98  # bus.data\_out = 76  # bus.data\_out = 159  # bus.data\_out = 143  # bus.data\_out = 248  # bus.data\_out = 183  # bus.data\_out = 159  # bus.data\_out = 92  # bus.data\_out = 91  # bus.data\_out = 137  # bus.data\_out = 73  # bus.data\_out = 208  # bus.data\_out = 215  # bus.data\_out = 81  # bus.data\_out = 150  # bus.data\_out = 12  # bus.data\_out = 194  # bus.data\_out = 200  # bus.data\_out = 119  # bus.data\_out = 61  # bus.data\_out = 18  # bus.data\_out = 126  # bus.data\_out = 109  # bus.data\_out = 57  # bus.data\_out = 31  # bus.data\_out = 211  # bus.data\_out = 133  # bus.data\_out = 120  # bus.data\_out = 200  # bus.data\_out = 73  # bus.data\_out = 63  # bus.data\_out = 42  # bus.data\_out = 88  # bus.data\_out = 134  # bus.data\_out = 142  # bus.data\_out = 156  # bus.data\_out = 250  # bus.data\_out = 38  # bus.data\_out = 115  # bus.data\_out = 163  # bus.data\_out = 47  # bus.data\_out = 179  # bus.data\_out = 95  # bus.data\_out = 68  # bus.data\_out = 247  # bus.data\_out = 203  # bus.data\_out = 230  # bus.data\_out = 90  # bus.data\_out = 41  # bus.data\_out = 237  # bus.data\_out = 218  # bus.data\_out = 101  # bus.data\_out = 181  # bus.data\_out = 223  # bus.data\_out = 121  # bus.data\_out = 68  # memory[0] = 36  # memory[1] = 129  # memory[2] = 9  # memory[3] = 99  # memory[4] = 13  # memory[5] = 141  # memory[6] = 101  # memory[7] = 18  # memory[8] = 1  # memory[9] = 13  # memory[10] = 118  # memory[11] = 61  # memory[12] = 237  # memory[13] = 140  # memory[14] = 249  # memory[15] = 198  # memory[16] = 197  # memory[17] = 170  # memory[18] = 229  # memory[19] = 119  # memory[20] = 18  # memory[21] = 143  # memory[22] = 242  # memory[23] = 206  # memory[24] = 232  # memory[25] = 197  # memory[26] = 92  # memory[27] = 189  # memory[28] = 45  # memory[29] = 101  # memory[30] = 99  # memory[31] = 10  # memory[32] = 128  # memory[33] = 32  # memory[34] = 170  # memory[35] = 157  # memory[36] = 150  # memory[37] = 19  # memory[38] = 13  # memory[39] = 83  # memory[40] = 107  # memory[41] = 213  # memory[42] = 2  # memory[43] = 174  # memory[44] = 29  # memory[45] = 207  # memory[46] = 35  # memory[47] = 10  # memory[48] = 202  # memory[49] = 60  # memory[50] = 242  # memory[51] = 138  # memory[52] = 65  # memory[53] = 216  # memory[54] = 120  # memory[55] = 137  # memory[56] = 235  # memory[57] = 182  # memory[58] = 198  # memory[59] = 174  # memory[60] = 188  # memory[61] = 42  # memory[62] = 11  # memory[63] = 113  # memory[64] = 133  # memory[65] = 79  # memory[66] = 59  # memory[67] = 58  # memory[68] = 126  # memory[69] = 21  # memory[70] = 241  # memory[71] = 217  # memory[72] = 98  # memory[73] = 76  # memory[74] = 159  # memory[75] = 143  # memory[76] = 248  # memory[77] = 183  # memory[78] = 159  # memory[79] = 92  # memory[80] = 91  # memory[81] = 137  # memory[82] = 73  # memory[83] = 208  # memory[84] = 215  # memory[85] = 81  # memory[86] = 150  # memory[87] = 12  # memory[88] = 194  # memory[89] = 200  # memory[90] = 119  # memory[91] = 61  # memory[92] = 18  # memory[93] = 126  # memory[94] = 109  # memory[95] = 57  # memory[96] = 31  # memory[97] = 211  # memory[98] = 133  # memory[99] = 120  # memory[100] = 200  # memory[101] = 73  # memory[102] = 63  # memory[103] = 42  # memory[104] = 88  # memory[105] = 134  # memory[106] = 142  # memory[107] = 156  # memory[108] = 250  # memory[109] = 38  # memory[110] = 115  # memory[111] = 163  # memory[112] = 47  # memory[113] = 179  # memory[114] = 95  # memory[115] = 68  # memory[116] = 247  # memory[117] = 203  # memory[118] = 230  # memory[119] = 90  # memory[120] = 41  # memory[121] = 237  # memory[122] = 218  # memory[123] = 101  # memory[124] = 181  # memory[125] = 223  # memory[126] = 121  # memory[127] = 68 |