**I2C Slave to Read/write 8 bit of data data**

A project Report Based On

**SUBMITTED BY**

SHYAM SARVAIYA (ID NO. 19EL090)

**LAB INSTRUCTOR**

Mr. Chintan Patel



Department of EL,

BIRLA VISHWAKARMA VIDYALAYA,ANAND



Birla Vishvakarma

Mahavidyalaya

(AN AUTONOMOUS INSTITUTION) ELECTRONICS ENGINEERING ACADEMIC YEAR:2020-21)

SUBJECT: 3EL42 - DIGITAL SYSTEM DESIGN

Institute Vision:

“Produce globally employable innovative engineers with core values”

Institute Mission:

Re-engineering curricula to meet global employment requirements

Promote innovative Practices at all levels

Imbibe core values

Reform policies, systems and processes at all levels

Develop faculty and staff members to meet the challenges

Department: Electronics Engineering

Vision: “Produce globally employable, innovative Electronics engineers with core values”

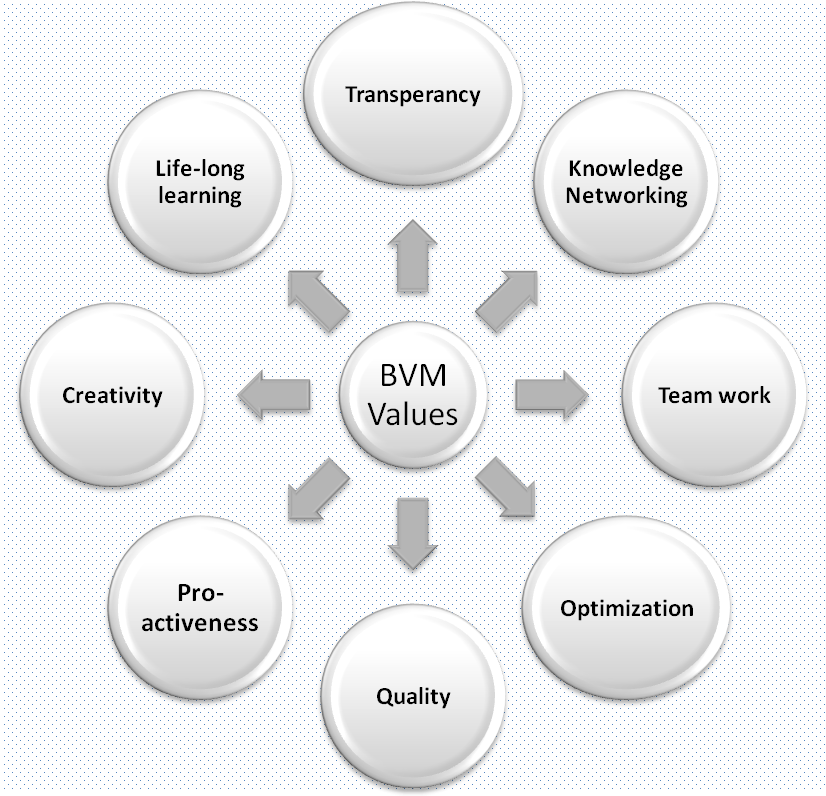
Mission:

* Promote Innovative Practices to strengthen teaching and learning process in electronics engineering
* Develop faculty and staff to meet challenges in Electronics engineering
* Adapt Engineering curricula to meet global requirements for Electronics engineering programme
* Reform policies, systems and processes at all levels
* Imbibe core Values.

**Program Educational Objectives (PEOs):**

* Study and analysis of Electronics engineering systems.
* Adapt state-of-art developments in Electronics engineering and eco-friendly technologies.
* Design and develop Electronic hardware and software-based applications.

**BVM CORE VALUES**



**ELECTRONICS ENGG DEPARTMENT**

BIRLA VISHWAKARMA MAHAVIDYALAYA ENGG COLLEGE V V NAGAR

**CERTIFICATE**

This is to certify that Mr. SHYAM SARVAIYA ID No.(19EL090) of B.Tech. (Electronics Engineering) SEM-V has satisfactorily completed the term work of the subject 3EL042 prescribed by BVM an Autonomous Institution during the Academic Year 2021-2022

Lab Teacher Course Coordinator & Guide Chintan Sir Chintan Sir

**Acknowledgment**

Firstly, we offer thanks to our guide and Course coordinator **Chintan sir** Electronics Department, Birla Vishvakarma Mahavidyalaya, our Lab teacher: **chintan sir** Lecturer, Birla Vishvakarma Mahavidyalaya, Electronics Department, for their invaluable support, guidance and advice given throughout this semester and for helping to establish our direction.

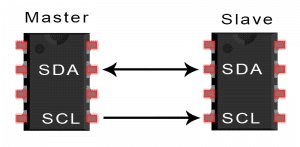
We express our deep and sincere sense of gratitude to specially,**chintan sir** Electronics Department, Birla Vishvakarma Mahavidyalaya, who has given us invaluable support and given opportunities to learn and develop researcher skills and has been unending source of inspiration to us. We also thankful to all our other faculty members for their consistence support and guidance.

Shyam Sarvaiya (19EL090)

**INTRODUCTION**

I2C combines the best features of SPI and UARTs. With I2C, you can connect multiple slaves to a single master (like SPI) and you can have multiple masters controlling single, or multiple slaves. This is really useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.

Like UART communication, I2C only uses two wires to transmit data between devices:

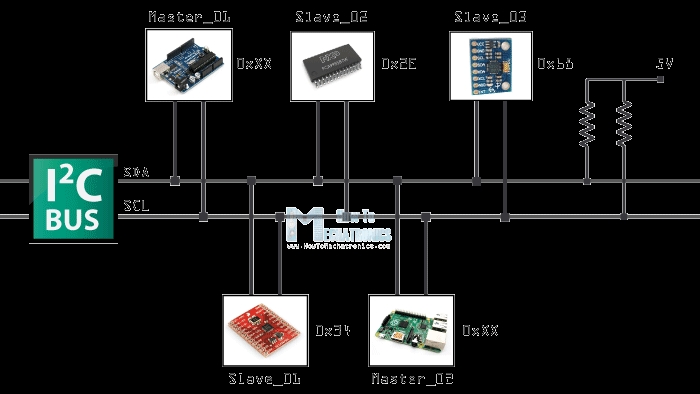
[](https://www.circuitbasics.com/wp-content/uploads/2016/01/Introduction-to-I2C-Single-Master-Single-Slave.png)

**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.

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).

**Diagram Of Project**

*~~~~*

**Verilog code**

1. /\*\*
2. I2C Slave to Read/Write 8 bits of data only
3. \*/
4. `timescale 1ns / 1ps
5. Module Slave(
6. Inout wire SDA,
7. Input wire SCL);
9. Reg [4:0] IDLE = 4’b0000;
10. Reg [4:0] START = 4’b0001;
11. Reg [4:0] READ\_ADDRESS = 4’b0010;
12. Reg [4:0] READ\_WRITE = 4’b0011;
13. Reg [4:0] DATA = 4’b0100;
14. Reg [4:0] DATA\_ACK = 4’b0101;
15. Reg [4:0] STOP = 4’b0110;
16. Reg [4:0] ADDRESS\_ACK = 4’b0111;
18. Reg [4:0] state = 4’b0010;
20. Reg [6:0] slaveAddress = 7’b000\_1000;
21. Reg [6:0] addr = 7’b000\_0000;
22. Reg [6:0] addressCounter = 7’b000\_0000;
24. Reg [7:0] data = 8’b0000\_0000;
25. Reg [6:0] dataCounter = 7’b000\_0000;
27. Reg readWrite = 1’b0;
28. Reg start = 0;
29. Reg write\_ack = 0;
31. Assign SDA = (write\_ack == 1) ? 0 : ‘b1z;
33. Always @(negedge SDA) begin
34. If ((start == 0) && (SCL == 1))
35. Begin
36. Start <= 1;
37. addressCounter <= 0;
38. dataCounter <= 0;
39. End
40. End
42. Always @(posedge SDA) begin
43. If (state == DATA\_ACK && SCL == 1)
44. Begin
45. Start <= 0;
46. State <= READ\_ADDRESS;
47. End
48. End
50. Always @(posedge SCL)
51. Begin
52. If (start == 1)
53. Begin
54. Case (state)
55. READ\_ADDRESS:
56. Begin
57. Addr[addressCounter] <= SDA;
58. addressCounter <= addressCounter + 1;
59. If (addressCounter == 6)
60. Begin
61. State <= READ\_WRITE;
62. End
63. End
64. READ\_WRITE:
65. Begin
66. readWrite <= SDA;
67. State <= ADDRESS\_ACK;
68. End
69. ADDRESS\_ACK:
70. Begin
71. Write\_ack <= 1;
72. State <= DATA;
73. End
74. DATA:
75. Begin
76. Write\_ack <= 0;
78. Data[dataCounter] <= SDA;
79. dataCounter <= dataCounter + 1;
80. If (dataCounter == 8)
81. Begin
82. State <= DATA\_ACK;
83. Write\_ack <= 1;
84. End
85. End
86. DATA\_ACK:
87. Begin
88. Write\_ack <= 0;
89. State <= STOP;
90. End
91. STOP:
92. Begin
93. Start <= 0;
94. State <= READ\_ADDRESS;
95. End
96. Endcase
97. End
98. End

101. Endmodule

**TESTBENCH**

**/\*\***

**Testing I2C Slace for reading/writing 8 bits of data only**

**\*/**

**`timescale 1ns / 1ps**

**module Slave\_TB ();**

**reg clk;**

**wire SDA;**

**wire SCL;**

**pullup(SDA);**

**pullup(SCL);**

**reg [6:0] addressToSend = 7'b000\_1000; //8**

**reg readWite = 1'b1; //write**

**reg [7:0] dataToSend = 8'b0110\_0111; //103 = 0x67**

**integer ii=0;**

**initial begin**

**clk = 0;**

**force SCL = clk;**

**forever begin**

**clk = #1 ~clk;**

**force SCL = clk;**

**end**

**end**

**Slave #() UUT**

**(.SDA(SDA),**

**.SCL(SCL));**

**initial**

**begin**

**$display("Starting Testbench...");**

**clk = 0;**

**force SCL = clk;**

**#11**

**// Set SDA Low to start**

**force SDA = 0;**

**// Write address**

**for(ii=0; ii<7; ii=ii+1)**

**begin**

**$display("Address SDA %h to %h", SDA, addressToSend[ii]);**

**#2 force SDA = addressToSend[ii];**

**end**

**// Are we wanting to read or write to/from the device?**

**$display("Read/Write %h SDA: %h", readWite, SDA);**

**#2 force SDA = readWite;**

**// Next SDA will be driven by slave, so release it**

**release SDA;**

**$display("SDA: %h", SDA);**

**#2; // Wait for ACK bit**

**for(ii=0; ii<8; ii=ii+1)**

**begin**

**$display("Data SDA %h to %h", SDA, dataToSend[ii]);**

**#2 force SDA = dataToSend[ii];**

**end**

**#2; // Wait for ACK bit**

**// Next SDA will be driven by slave, so release it**

**release SDA;**

**// Force SDA high again, we are done**

**#2 force SDA = 1;**

**#100;**

**$finish();**

**end**

**initial**

**begin**

**// Required to dump signals to EPWave**

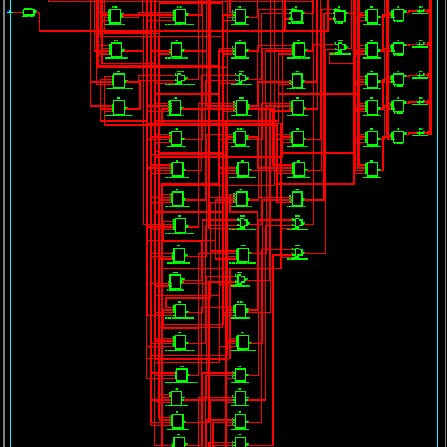
**$dumpfile("dump.vcd");**

**$dumpvars(0);**

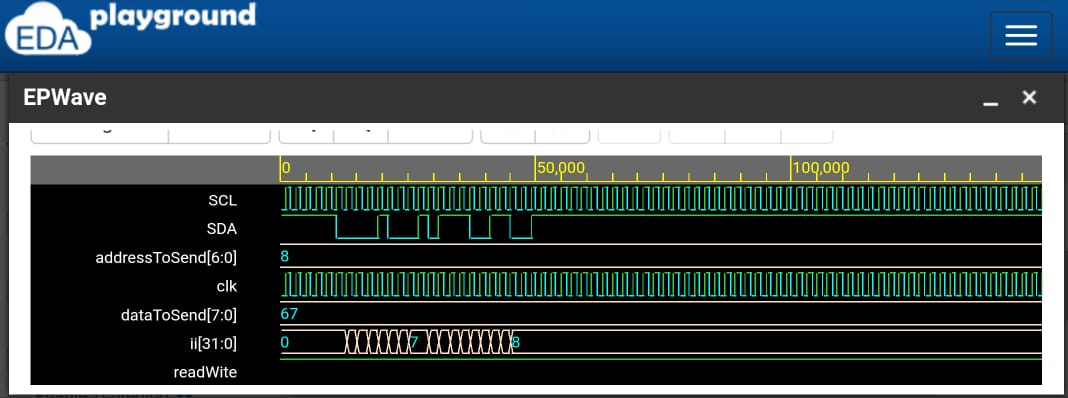
**end**

**endmodule**

**RTL Schematic**

****

**Simulation result:**

****