iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VIII **Month of publication:** Aug 2023

DOI: https://doi.org/10.22214/ijraset.2023.55392

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VIII Aug 2023- Available at www.ijraset.com

Design and Verification of Serial Communication Interface for High Speed Data Transfer

Dr Sujatha Hirmath[1], Arjumanth Farraj[2]
Dept. of Electronics and Communication Engineering RV College of Engineering Bengaluru, India

Abstract: Inter-Integrated Controller is a crucial interface for device communication. I 2 C is utilized as an interface between EEPROMS since they need an interface for communication. The design's operation has been confirmed, and there has been no data transfer interruption. The proposed verification environment is designed using system Verilog which includes the constrained randomization for the stimulus generation. In the proposed system, a modified FSM is used in the data transfer process between the master and slave EEPROM. I2C typically operates at a 100MHz clock frequency in an FPGA with a reference clock used in the system creates a slower clock which is completely configurable in the TB for the bits to get transferred. The finite state machine also handles the acknowledge response where it stays in the same state until the acknowledge is received unlike going to the starting state. A memory controller was used to control the I2C transactions with the EEPROM slave and the data transfers were executed as expected. System Verilog is the HDL used for the design and creating the verification environment with Xilinx Vivado 2020.02 as the IDE used to run the simulations.

I. LITERATURE REVIEW

Serial communication buses are used in electronic systems to interconnect sensors and other devices using the Inter integrated circuit (I2C) which are vulnerable to bus wide failure. For the aerospace applications a new simple approach has been proposed to add a simple external circuit to a communication bus to prevent the bus from going to failure [1]. A flexible hardware architecture for slave device of I2C is implemented where a verification platform is built to verify the performance of the designated hardware architecture [2]. Energy consumption has become a bottleneck in modern integrated circuits (ICs). The energy demand of synchronous serial communication buses is effectively decreased by employing encoding techniques to reduce bit transactions of the transmitted bit streams with a 25% reduction in the energy consumption which is used in the serializer/deserializer devices [3]. The composition of three protocols i.e, SPI, I2C and UART comparison with work and the advantages are presented with UART being more stable and SPI has a variety of transmission modes to choose from. UART out performs SPI and I2C [4]. Paper 5 depicts about the functional verification of I2C controller through the I2C analyzer with UART controller is used to send data serially via COM ports [5]

II. INTRODUCTION

I2c is a 2wire interface where it uses 2 wires to communicate with the device. Verification becomes a bit tougher with the reduction of the available pin counts. The type of signals whether the control data, address or data has to be made known with only 2 lines along with the start and stop condition makes it complicated. Memory is used to verify the I2C protocols by writing and reading the data from the memory using two lines.

The protocol begins with a start condition, with I2C., many devices can be integrated. Certain bits can be used for read or write and the operation can be controlled. The data can be of three types 1. Control data (consists of specific operations that is needed to be performed), 2.) Address to recognize the devices connected to the bus, 3.) Data is the one which will be communicated with the device. The transactions will be stopped by using the stop condition. SDA and SCL are known for the 2-wire interface, these 2 pins are used for both controlling and data operation. Reference clock will be used where I2C usually works in a lower clock frequency with 100MHz of clock frequency is common in an FPGA. I2C works at 400Kbps where the bit by bit transaction takes place at this frequency. Reference clock is used to simplify a logic where in the first clock tick, SDA is high similarly SCL is also high. In the 2nd clock tick, SCL continues to stay high while SDA goes low and this is called as the start condition where the device is ready to take a write or read transaction. Address will be sent where the data will be stored in the memory with SDA is of size single bit and 7 bit address. 1 bit is used to read or write the data into the memory and at each edge of the reference clock the address will be sent. The address will be sent bit by bit and this will consume 8 clock cycles.

©IJRASET: All Rights are Reserved

Once the 8 bit address is sent where it includes the 7 bit address and 1 bit read or write, acknowledge will be received in the next clock tick. The acknowledgment wouldn't be missed if the intermediary write and read is happening.

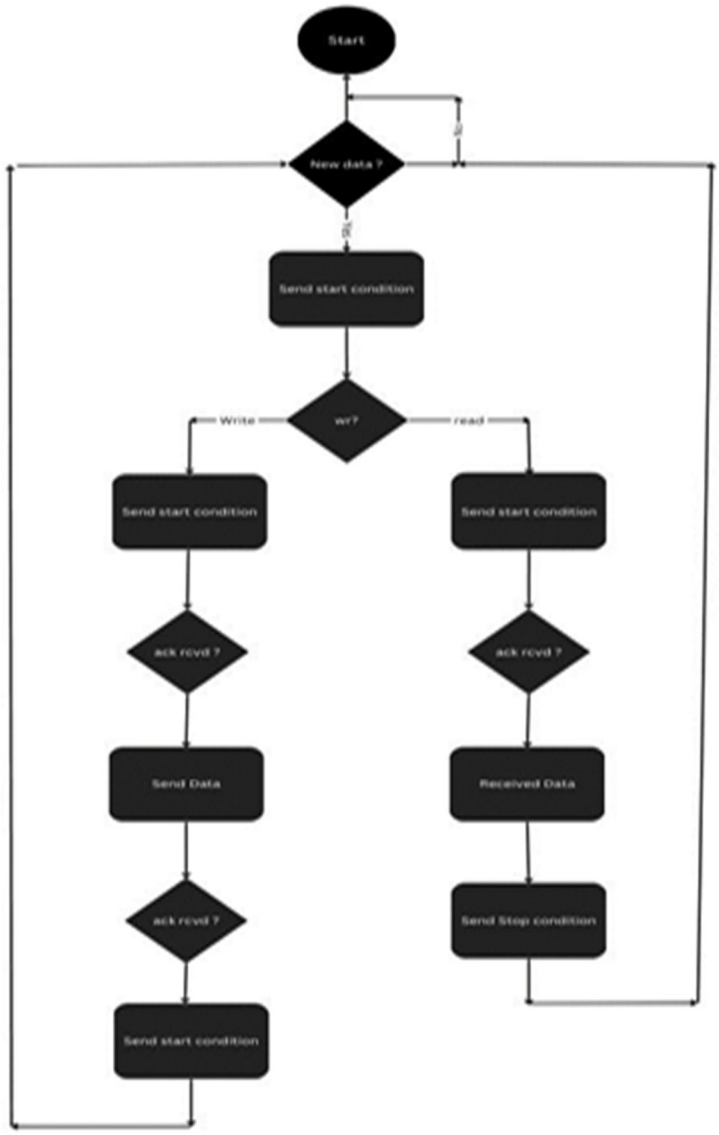


Figure 1: Methodology

FSM is designed in such a way that if the acknowledge is not received in the immediate next clock tick, we will be waiting in the same state until the acknowledge is given by the device. Once the 8 bit data is received the ack will be given by the slave by pulling the SDA line low. The stop condition is given by making the SDA line go high. For the start condition SCL should stay at 1 and for rest of the period SCL will follow the reference clock. If the LSB bit of address is 0 then write will happen else the read takes place. The SDA pin works both as input and output, during the output mode the data will be sent from the master to device and similarly during the input the data will be sent from device to master. During the write operation, the start condition will be initiated followed by sending the address. The ack will be sent by the memory then the data will be sent followed by the ack and finally the master sends the stop condition. Once the user requests for the new data, if the start condition is attained then read and write is checked using separate FSM's. During the write state the user wants to write the data into the memory where this begins by sending the address, and wait until the ack is received then the data is sent for the ack and the stop condition will be received. During the read operation the read address will be sent followed by the ack, the read data will be received followed by sending the stop condition.

A. Controller With Memory

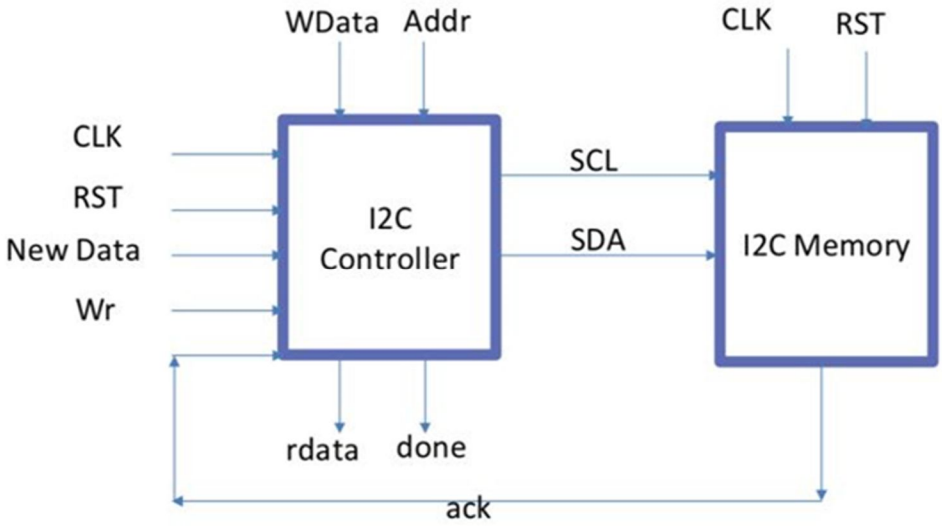


Figure 2: I2C Controller with Slave Memory

There are two behaviors for scl, sclt and sclk_ref, depending on which state the FSM is whether the wstart state, wstop state or rstop state. For sda, the behavior is dependent on sda_en signal which holds sdat value else holds high impedance when reading the data from the memory.

1) Slower Clock

I2C won't be working at a higher rate, the onboard clk frequency of an FPGA board is 50MHz to 100Mhz. It'll be down converted to 400KHz.

$$100MHz / 400K = N$$

N/2 is the logic to create a slower clock. The slower clock is adopted in the memory In the sensitivity list the sclk_ref and rst is added and during the reset condition, sclt, sdat and donet signals will be initialized to 0. In an ideal state both sda and scl will be 1 for newdata which is the first state after reset, for start condition scl will be 1 and sda will be 0 in the next clock tick. Addrt is a temporary variable which is of size 8 bit with 7 bits of MSB for addr and 1 bit of LSB for the read or write operation. The next state is the check write, as soon as the check write condition is reached the first bit of the address so that in the next clock tick the data can be sent. When write is high the LSB of the address is sent and jump to the next state which is wsend_addr where rest of the bit will be sent according to the N value with sdat getting the value from addrt in this case and in the next clock tick itself the slave should the acknowledge.

The behavior of the acknowledge has been modified in this condition where in the next clock tick if the acknowledge is not given then it'll jump to the initial state to sense the data and this is how the FSM's are designed in the current market. The next state is waddr_ack, if the acknowledge is 1, the next state will be jumped else it'll be in the same state until the acknowledge is given. Wdata represents the input bus, this has to be handled properly as wdata wont change till the transaction is completed, the wdata will be stored into a temp register. This register will be utilized in sending the data, if the data or addr changes in between before the transaction is completed so those changes won't be reflected. The new data will be sent to I2C, the new data that is getting changed before the transaction is completed won't get captured. The next state is wstop, where sda will be 1 and done signal will be to signify the stop condition.

2) I2C Memory

Series of states are present in the FSM, the start condition will be verified when scl =1 and sda =0. As soon as we have the start condition the address gets stored. The LSB of the address defines the type of the address and master will be waiting for the acknowledgement. The memory will be the slave and in the same clock tick when the acknowledge was captured, read and write operations takes place where the extra clock cycle is not used for this.

During the write operation the store data operation will be performed where master writes into the memory and after this an acknowledgement will be given and finally the memory will be updated. The stop condition will be given by the master. During the read operation, the data will be sent to the master which is named as the read mem operation. Memory is working in an I2C protocol with only 2 pins for clock and data. sda works in both ways where during the write operation, it'll be reading the data from the memory by sending the data to the master.

The four operations are:

a) Writing address during the write operation
b) Writing data during a write operation
c) Writing address during a read operation
d) Reading the data during a read operation

When reset is high, the elements of the memory will be 8'h91. During the start condition, when scl =1 and sda=0 where it jumps to the store address else it'll be waiting in the same state until a valid state is achieved. When sda_en =1, the memory address locations will be updated with the sda data. Once the 8 bit data is stored, an acknowledgement will be sent to the master. Acknowledgement will be high only for a single clock tick and the stop condition will be given by the master and jumps back to the start of the FSM.

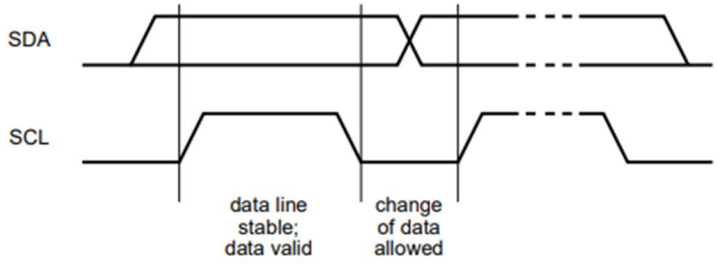


Figure 3 Behavior of data line

Throughout the high portion of the clock, the data on the SDA line must remain steady. Only when the clock signal on the SCL line is low can the data line's state be changed.

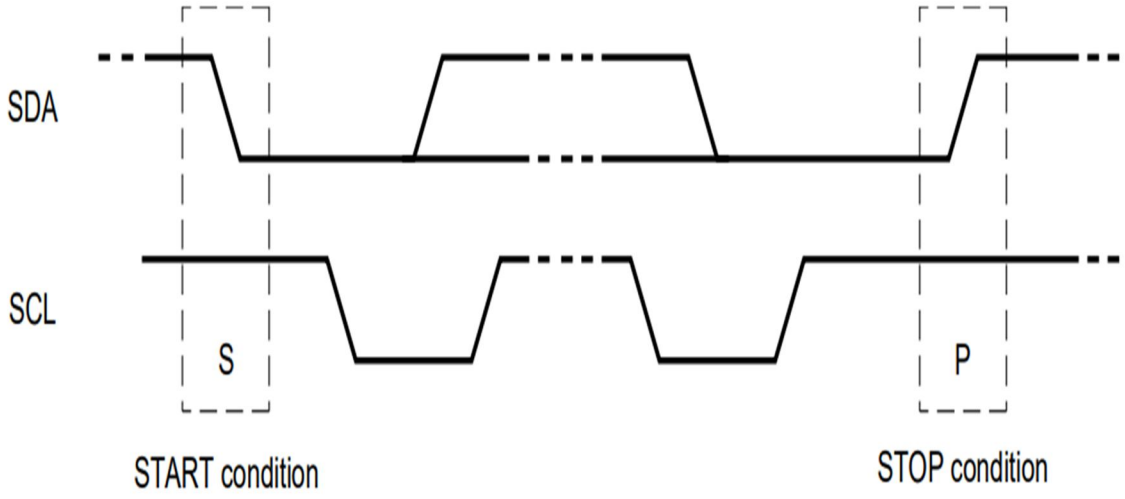


Figure 4 Start - Stop Condition

All transactions are started with a START and ended with a STOP.A START condition is established by a HIGH to LOW transition on the SDA line while the SCL is HIGH. A STOP condition is established when the SDA line goes from LOW to HIGH when the SCL is HIGH.

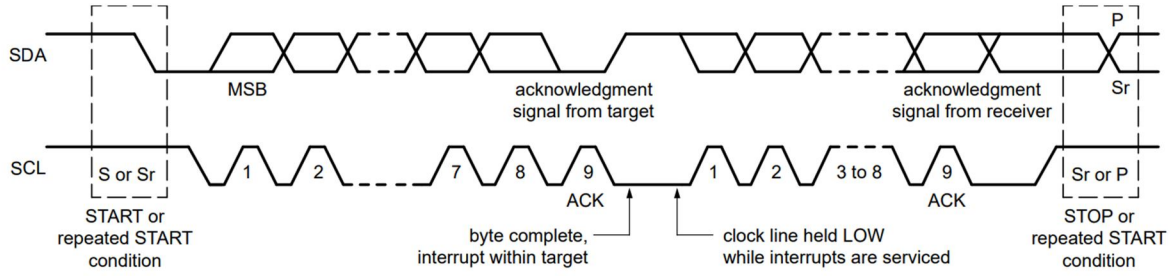


Figure 5 : Transaction Behavior

Eight bits must make up each byte that is sent across the SDA line. There are no limits on how many bytes can be transferred in a single transaction. An Acknowledge bit is required to come after every byte. The Most Significant Bit (MSB) of the data is transmitted first. Holding the clock line SCL low will make the controller enter a wait state if the target cannot receive or transmit another full byte of data until it has completed another task, such as handling an internal interrupt. When the target is ready for another byte of data and releases clock line SCL, data transfer resumes.

B. The verification Environment

The Interface has been designed in such a way that the I2C works in the sclk_ref which is the slower clock to give access to driver and monitor.

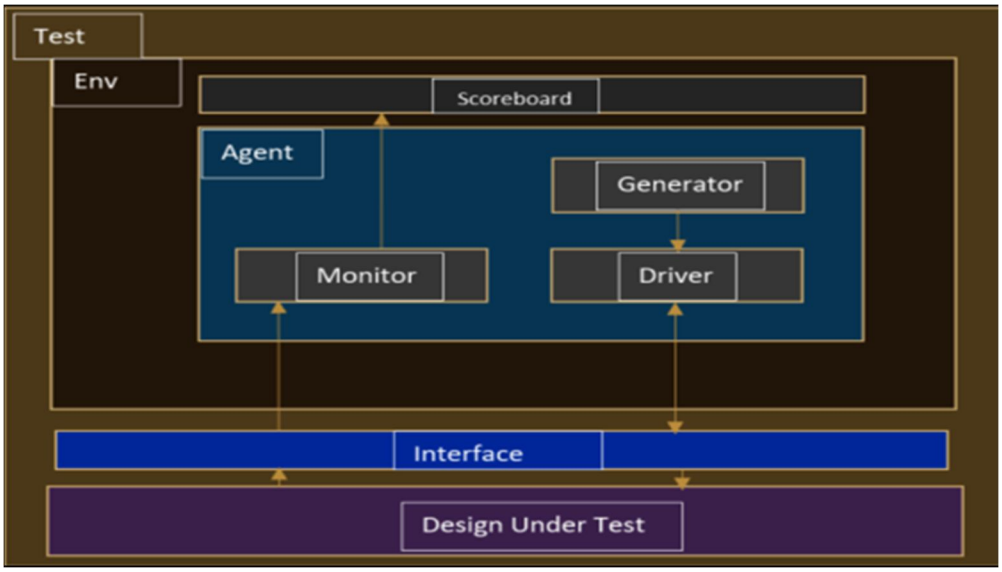


Figure 6: Verification Environment

1) Transaction

The verification of valid transactions will take place when the actual read or write . For write signal the pseudo random number generator will be used for both data and address with rand so that there will be some repetition which will be easier during the comparison phase. The display function will be with a valid tag so that the specific class prints a value. Deep copy is used to copy the data from the data members of the class.

2) Generator

The general working of a generator is to randomize the transactions and send it to the driver. Data container will be declared inside the generator and using the mailbox communication the data from the transaction data gets captured in the data container. Count variable will be declared so that the number of stimulus the user wishes can be sent through it. At the end when the required number of transactions are sent, a done event will be triggered which will be sensed in the testbench top or the env to notify that the simulation can be stopped.

3) Driver

A virtual interface is added in the driver class. Similarly using the mailbox communication, the data from the generator class comes to driver class where it gets stored in a data container. Driver basically consists of the reset and the main task of the system. Once the tasks are completed an event will be triggered which will give a notification that the driver has completed it task of reset and performing the main task.

4) Monitor

The main task of a monitor is to sample the data correctly. Whenever wr=0, we are reading the data. The data from the interface wdata, addr will be stored in the container. As soon as done=1, the sampling of data takes place with the data of the interface will be stored in the rdata of the data container. After correctly filtering the data and sending it through the mailbox, the put method will be called and a transaction will be in it.

5) Scoreboard

An event called sconext is declared inside scoreboard to tell that the scoreboard has completed its task and the generator can send the next transaction. A custom constructor is added with the argument as mailbox. The comparison algorithm is designed in such a way that the read data will be a temp variable with the data updated in the memory from the previous transaction and if in case the user is not updating the memory then a default value of decimal equivalent 91 will be stored.

6) Testbench Top

All the classes are added in the testbench top with 2 events that will be working with a generator and a driver, the other with generator and scoreboard. An interface will be declared here along with mailbox for communication from generator, driver and scoreboard. The connection of the interface with the dut takes place in the testbench top with a half clock period of 5 ns so that a 10ns clock will be generated. The event of generator and driver & scoreboard will be connected in the tb top. During the pretest phase, the reset of the DUT takes place. The run task will be triggered to perform the main operation of reading and writing from & to the memory.

III. RESULTS AND DISCUSSIONS

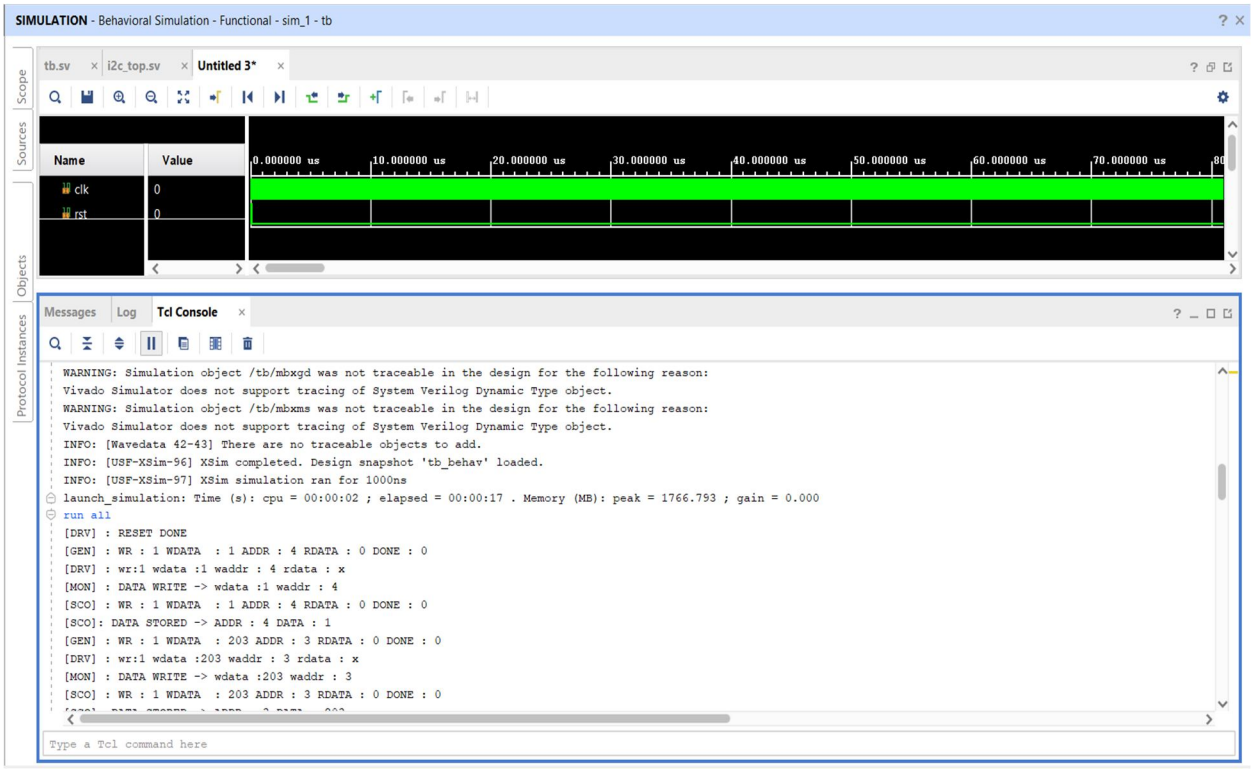


Figure 7: The reset phase

The above figure depicts about that after the reset phase the write is 1, address is 4 which is the randomized data sent through the generator. The read data and done will be 0 as there are no transactions done in the previous state.

Figure 8 TCL Console with data transactions

Figure 5 depicts the transactions taking place during the read and write. Read transactions takes place when write is 0, with driver the stimulus is sent to the DUT and scoreboard confirms the read and write transactions which took place.

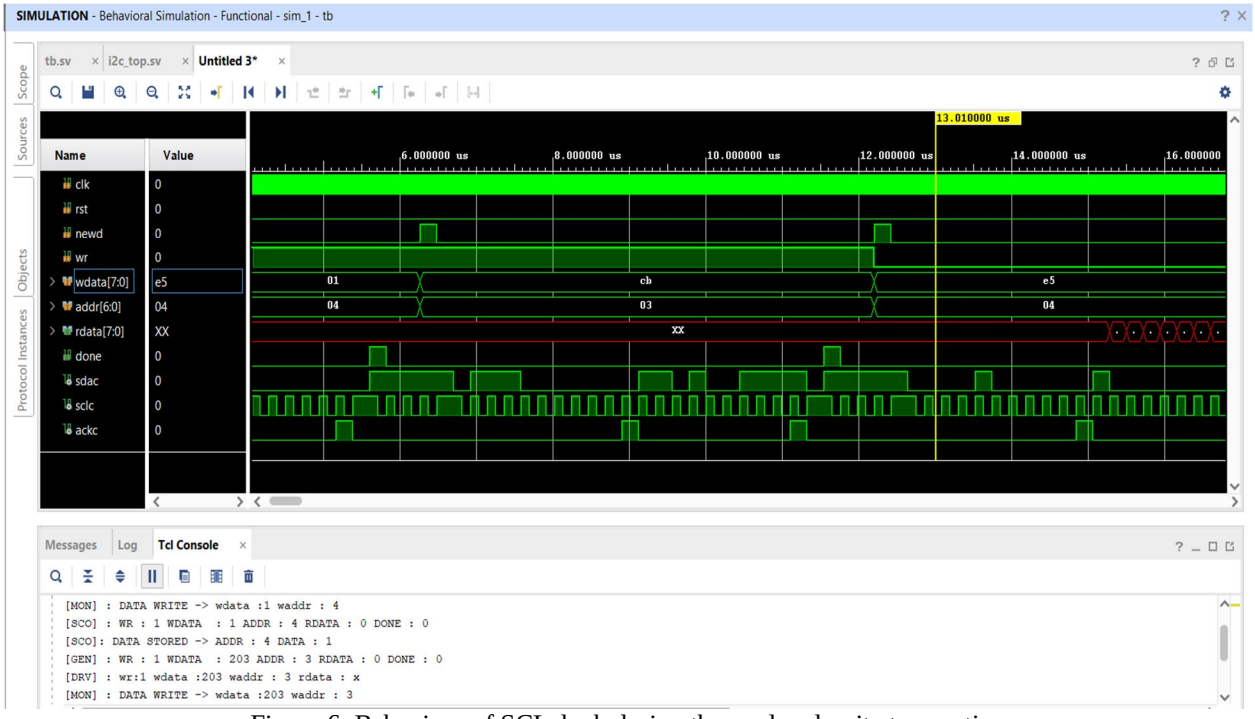


Figure 6: Behaviour of SCLclock during the read and write transaction

The transactions starts when sda and scl are 1, in the next clock tick sda is low and scl is high which is followed by the write transactions which begins by sending the address where an acknowledge will be given followed by writing the data to the memory. After writing the data to the memory the acknowledge will be given again and to end the transaction the sda will go low and scl as high.

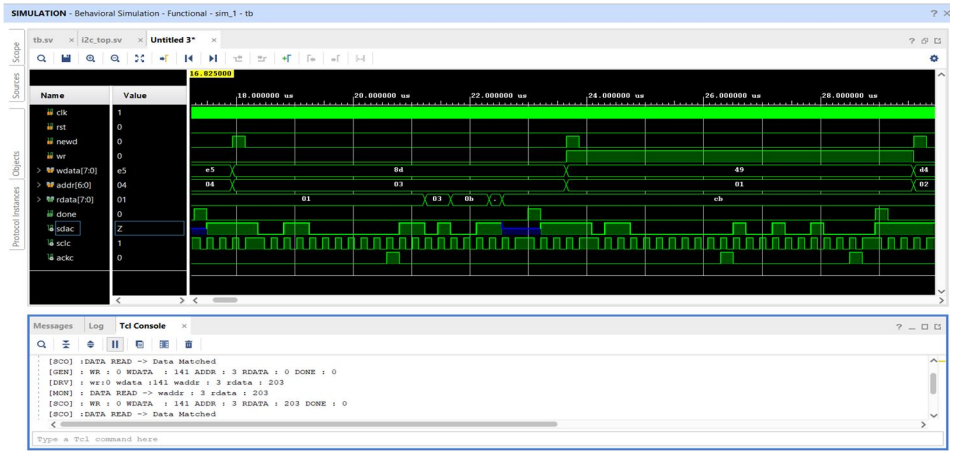


Figure 9: Behavior of SDA during the read condition

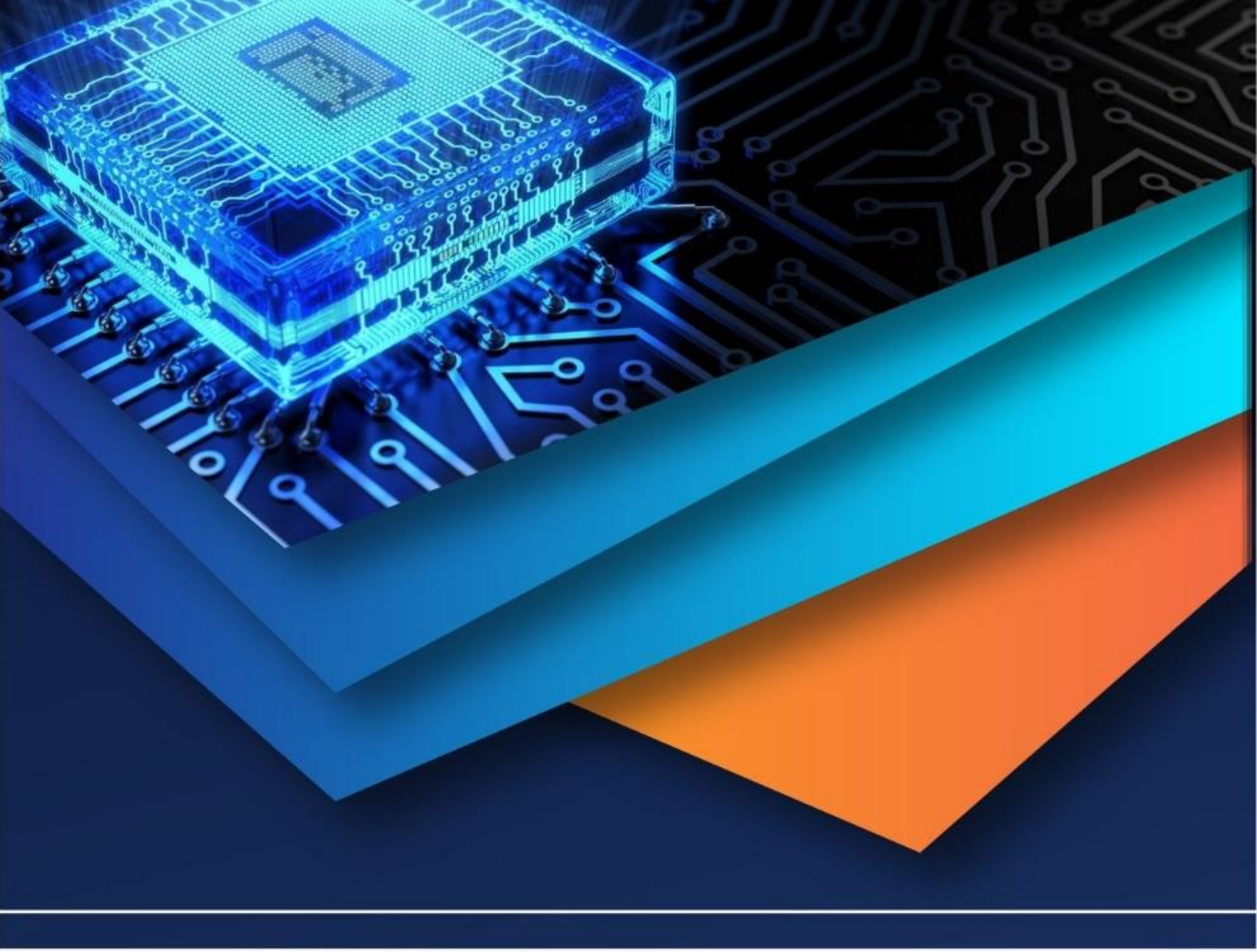
Figure 7 depicts about the behavior of SDA during the read condition, after the data has been written into the slave the read begins by pulling the sdac signal to high impedance during the read mode and low during the write mode which is evident in the waveform.

IV. CONCLUSION

In this paper, for the purpose of functionally verifying the I2C protocol, a system verilog testbench is proposed. Through the methodical execution of pertinent test cases, the testbench completely functionally verified I2C transactions using the memory as the slave and obtained 100% functional coverage. The simulation waveform shows how data can be successfully transferred between memory slave and I2C master controller using the SDA and SCL bus signals for various read and writes. Various scenarios were checked which includes writing the address and data during the write transaction, writing the address and reading the data during the read transaction at variable clock frequency which was proposed. The entire design was done using the system verilog and the testbench environment was done UVM using Xilinx vivado 2020.02 and Questasim 10.4 e for simulating the design.

REFERENCES

[1] M. Holliday, Z. Manchester and D. G. Senesky, "On-Orbit Implementation of Discrete Isolation Schemes for Improved Reliability of Serial Communication Buses," in IEEE Transactions on Aerospace and Electronic Systems, vol. 58, no. 4, pp. 2973-2982, Aug. 2022, doi: 10.1109/TAES.2022.3142713.
[2] C. Liu, Q. Meng, T. Liao, X. Bao and C. Xu, "A Flexible Hardware Architecture for Slave Device of I2C Bus," 2019 International Conference on Electronic Engineering and Informatics (EEI), Nanjing, China, 2019, pp. 309-313, doi: 10.1109/EEI48997.2019.00074.
[3] E. Maragkoudaki, W. Toms and V. F. Pavlidis, "Energy-Efficient Encoding for High-Speed Serial Interfaces," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 10, pp. 1484-1496, Oct. 2022, doi: 10.1109/TVLSI.2022.3194256.
[4] J. W. Bruce, M. A. Gray and R. F. Follett, "Personal digital assistant (PDA) based I2C bus analysis," in IEEE Transactions on Consumer Electronics, vol. 49, no. 4, pp. 1482-1487, Nov. 2003, doi: 10.1109/TCE.2003.1261257.
[5] V. Patel K.S and B. R, "Design and Verification of Wishbone I2C Master Device," 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS), Bangalore, India, 2018, pp. 1-5, doi: 10.1109/ICNEWS.2018.8904034.
[6] W. Conrads, "Integrated Feature TV Concept with Serial I2C-Bus-Control and Field Memory," in IEEE Transactions on Consumer Electronics, vol. CE-29, no. 4, pp. 469-474, Nov. 1983, doi: 10.1109/TCE.1983.356352.
[7] N. Bhuvaneswary, J. Deny and A. Lakshmi, "Exploration on Reusability of Universal Verification Methodology," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 1865-1868, doi: 10.1109/ICACITE53722.2022.9823570.
[8] M. Sukhanya and K. Gavaskar, "Functional verification environment for I2C master controller using system verilog," 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, India, 2017, pp. 1-6, doi: 10.1109/ICSCN.2017.8085732.
[9] P. Bagdalkar and L. Ali, "Interfacing of light sensor with FPGA using I2C bus," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 843-846, doi: 10.1109/ICACCS48705.2020.9074372.
[10] P. D. Mulani, "SoC Level Verification Using System Verilog," 2009 Second International Conference on Emerging Trends in Engineering & Technology, Nagpur, India, 2009, pp. 378-380, doi: 10.1109/ICETET.2009.205.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)