**UNIVERSITY OF ENGINEERING AND TECHNOLOGY  
 LAHORE, PAKISTAN**

****

**RTL Design of Quad Serial Peripheral Interface (Quad-SPI)**

May 25, 2022.

**Submitted By:**

Wajahat Riaz 2021-MS-EE-36

**Department of Electrical Engineering**

**Introduction:**

A microcontroller can send multiple bits to a peripheral device by using multiple wires or by sending multiple bits in series over a single wire. The former is called parallel I/O and the latter is called serial I/O. Serial I/O is popular, especially when pins are limited because it uses few wires and is fast enough for many applications. Indeed, it is so popular that many standards for serial I/O have been established and microcontrollers offer dedicated hardware to easily send data via these standards. Some of the commonly found standards include an inter-integrated circuit (I2C), universal serial bus (USB), etc. This report introduces Quad Serial Peripheral Interface and implementations in SystemVerilog.

**Serial Peripheral Interface (SPI):**

Serial peripheral interface (SPI) is one of the most widely used interfaces between the microcontroller and peripheral ICs such as sensors, ADCs, DACs, shift registers, SRAM, and others. SPI is a synchronous, full-duplex master-slave-based interface. The data from the master or the slave is synchronized on the rising or falling clock edge of the clock. Both master and slave can transmit data at the same time. The SPI has a 1-bit data bus as shown below:

1-bit   
Data Bus

Flash Memory/

SD Card

(Slave Module)

Quad SPI

(Master Module)

Slave

A typical SPI device has the following four signals:

1) Clock signal

2) Chip select

3) Master out slave in (MOSI)

4) Master in slave out (MISO)

The device that generates the clock signal is called the master. Data transmitted between the master and the slave is synchronized to the clock generated by the master. SPI devices support much higher clock frequencies. SPI interfaces can have only one master and can have one or multiple slaves. The chip select signal from the master is used to select the slave. This is normally an active low signal and is pulled high to disconnect the slave from the SPI bus. When multiple slaves are used, an individual chip select signal for each slave is required from the master. MOSI and MISO are the data lines. MOSI transmits data from the master to the slave and MISO transmits data from the slave to the master. A diagram illustrating the signals is shown below:

SCK

SDO

SDI

CS

SCK

SDO

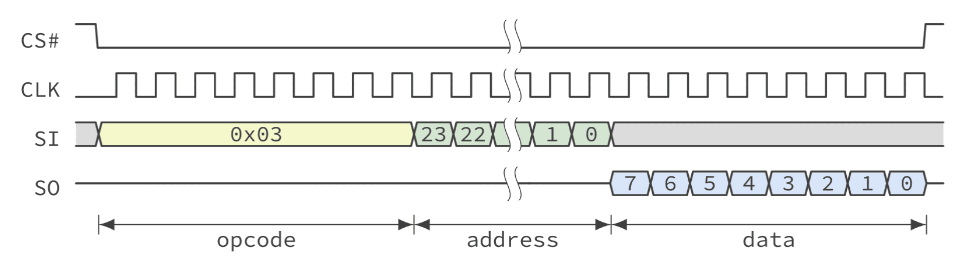
SDI

CS

Slave

Master

A waveform of SPI protocol is given below:



**Quad Serial Peripheral Interface (Quad-SPI):**

Quad-SPI, also known as QSPI is the extension of the traditional Serial Peripheral Interface (SPI) protocol. is a peripheral that can be found in most modern microcontrollers This serial interface allows the communication on four data lines between a master (for example AHB, AXI, Wishbone, etc) and an external flash memory that supports this interface. It is especially useful in applications that involve a lot of memory-intensive data like multimedia where on-chip memory is not enough. A block diagram along with its characteristic waveform is given below:

4-bit   
Data Bus

Flash Memory/

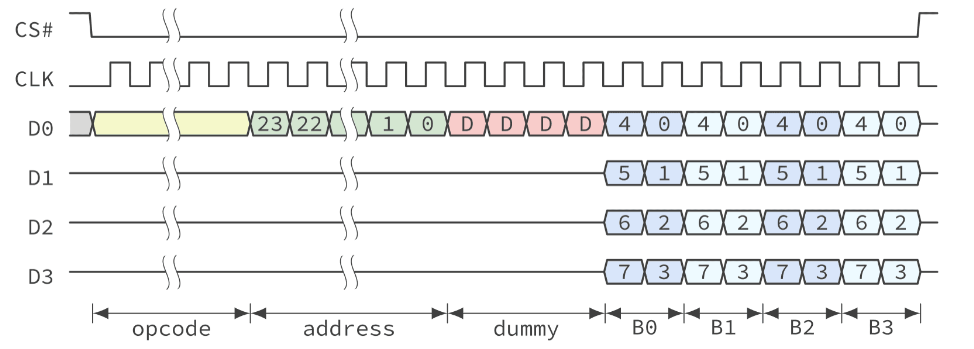
SD Card

(Slave Module)

Quad SPI

(Master Module)

Slave



The reason why we require a Quad-SPI protocol can be deduced from the following debate:

**Why did the engineers come up with yet another protocol? Wasn’t SPI enough for all applications?**

SPI was enough for most of the use cases like reading data from sensors and sending data to actuators or output devices and is still one of the most preferred protocols for on-chip serial communication but engineers struggled with one important problem, which is adding more memory to their embedded devices.

**What is the problem with flash memory, that makes it need a special interface?**

Flash memory is cheap and durable which makes it an attractive option for embedded applications. But it is slow by nature and this will create a bottleneck and impact the performance of embedded applications. Even though SPI is fast, the flash devices were not able to send out data at this rate through a single data line to make it work as fast as the on-chip memory.

**What solution did we use before quad SPI?**

Earlier before quad-SPI came, the solution was to use parallel memory where 8, 16 or 32 pins (depending on the address range) can be used to connect the external memory device with the microcontroller to achieve fast performance. But this approach had two major disadvantages:

1) It made the PCB design complicated.

2) It also meant that all these pins are fixed to one particular chip and cannot be used anymore for literally anything else.

To eliminate the problems discussed above, the Quad-SPI protocol was made.

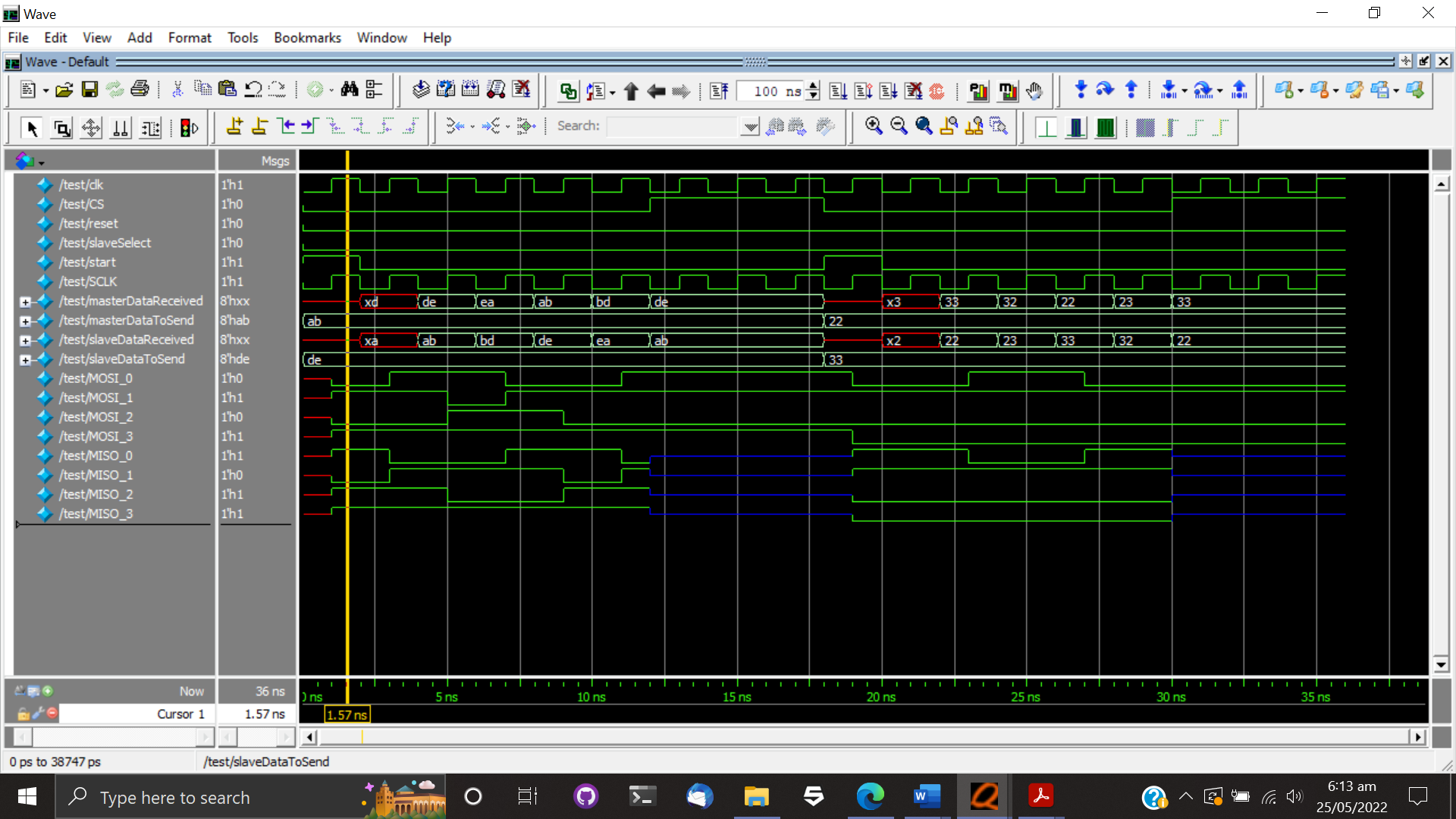
**Advantages of using Quad-SPI:**

Some of the key advantages of the protocol are listed in the table below:

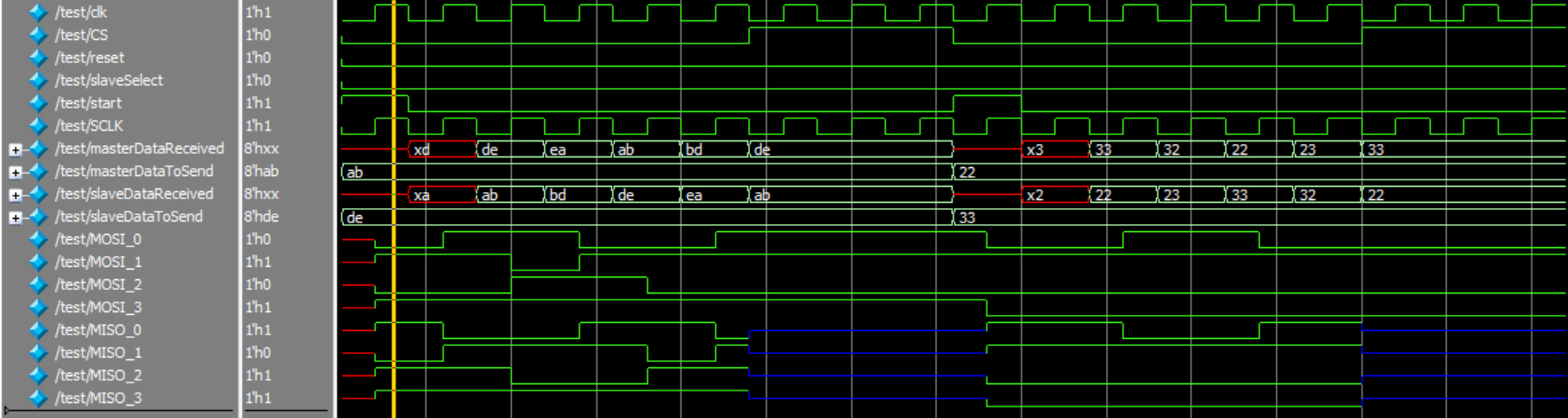
|  |  |
| --- | --- |
| **Benefits** | **Comments** |
| Low pin-count | Saves GPIOs to be used for other purposes. Can support single, dual, and quad memories. |
| Easier PCB design | Allows easier and faster PCB design thanks to a reduced pin count. |
| Save space for smaller size applications | Can be used in small-size applications due to its small footprint. |
| Cost-effective | Easier and faster design permits a lower development cost. |
| Executable | Extends limited on-chip flash memory allowing Quad-SPI memory to be seen as an internal memory |
| Extended size for data storage | Memory-mapped mode allows Quad-SPI memory to be accessed autonomously by any master e.g., AHB, AXI, Wishbone, etc. |
| High performances | Throughput is multiplied by four versus traditional SPI |

**RTL of Quad-SPI Protocol:**

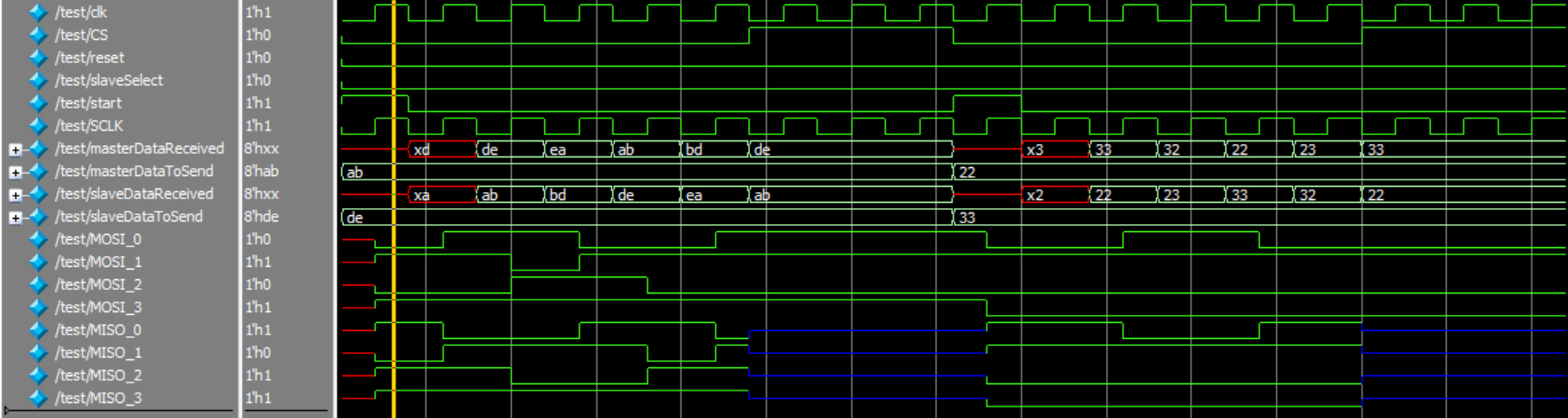
The protocol was implemented on QuestaSim using SystemVerilog. Two modules were made i.e., a master and a slave. A simplified testbench was implemented and the waveforms related to that are given below:



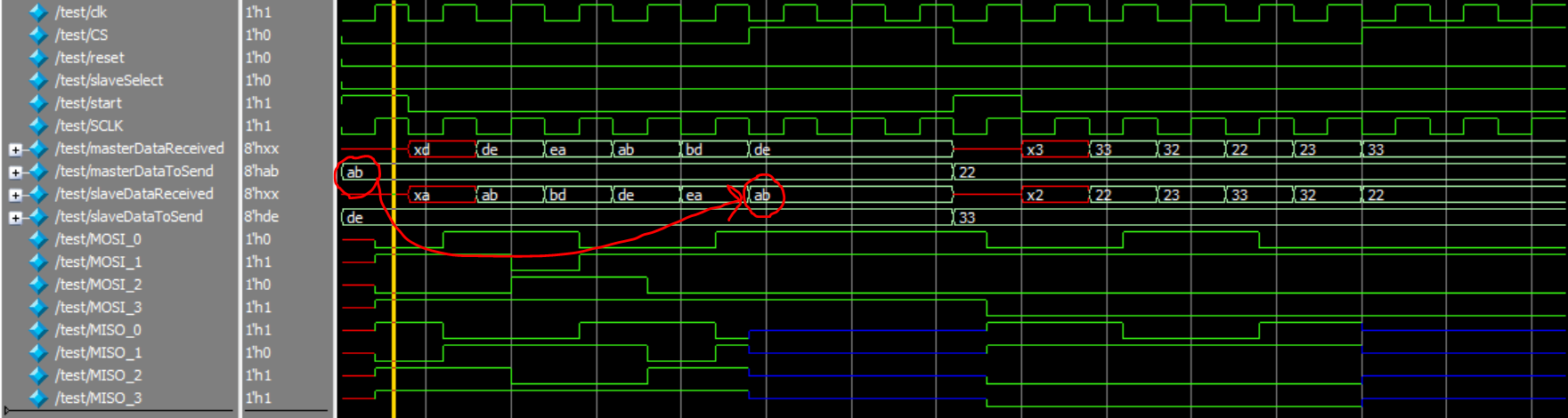
Initially, the data we want to send to the slave from the master is 8’hAB stored in **masterDataToSend**. Therefore, on the clock edge **SCLK** the most significant nibble 4’hA = 4’b1010 is being pushed onto the **MOSI\_x** lines. This is highlighted in the given figure below:

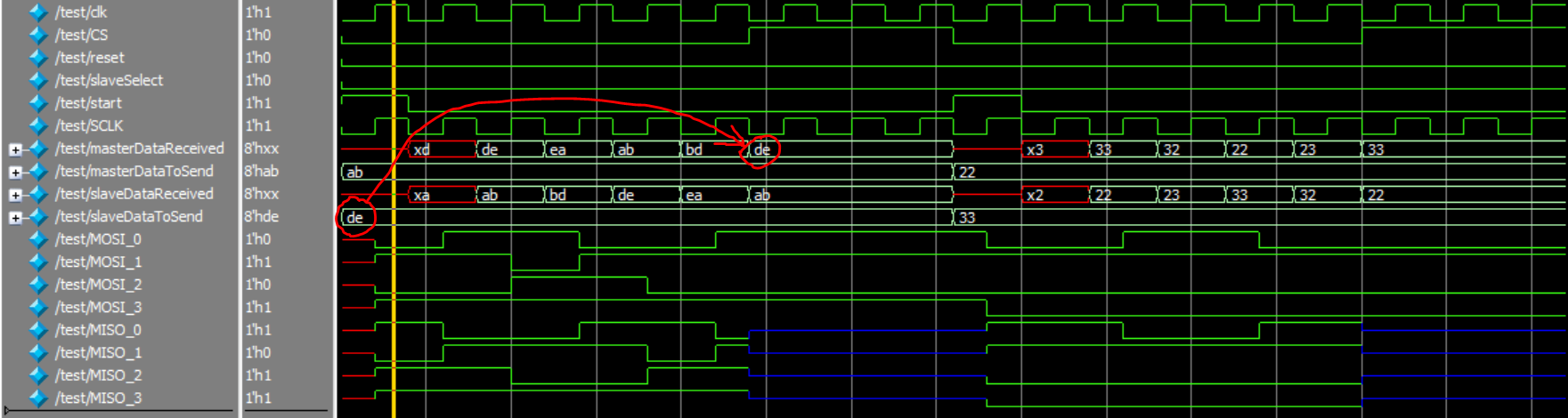


Similarly, the data we want to send to the master from the slave is 8’hDE stored in **slaveDataToSend**. On the clock edge **SCLK,** the most significant nibble 4’hD=4’b1101 is being pushed onto the **MISO\_x** lines. This is illustrated below:

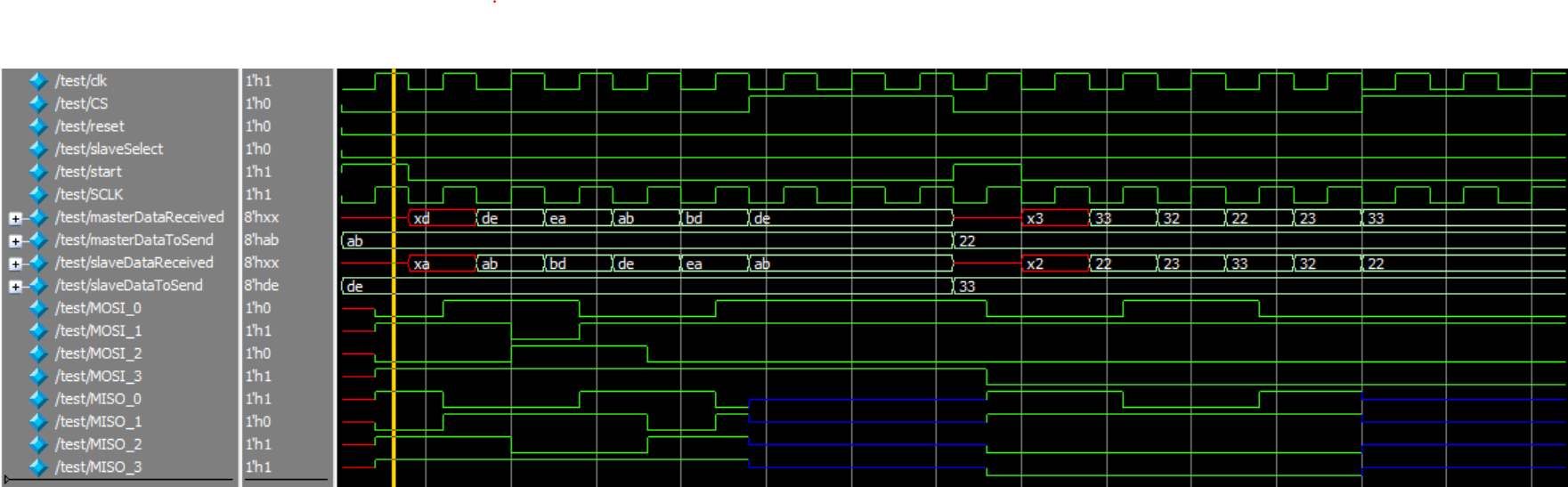


We know that initially, we want to send 8’hAB from master to slave, and similarly, we want to send 8’hDE from slave to the master. You can see in the figures below that the received data in both slave and master are an exact match of the data we sent.





Once the communication happens the chip select signal CS is set to high which disconnects the slave from the master and introduces dummy values. This phenomenon is encircled below:



Github link: <https://github.com/WajahatRiaz/QuadSPI>