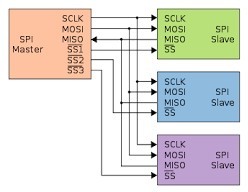


# Serial Peripheral Interface

**Advanced Logic Design**



## Team Members:

1. **Ahmed Alaa El-Sayed Arabi Zidan**
2. **Ahmed Ata**
3. **Menatalh Hossamalden**
4. **Sarah El-Zayat**

**Supervised by:**

* **Dr. Ihab Talkhan**
* **Eng. Mustafa Mahmoud**
* **Eng. Yahya Zakaria**

### Part I: Master Module

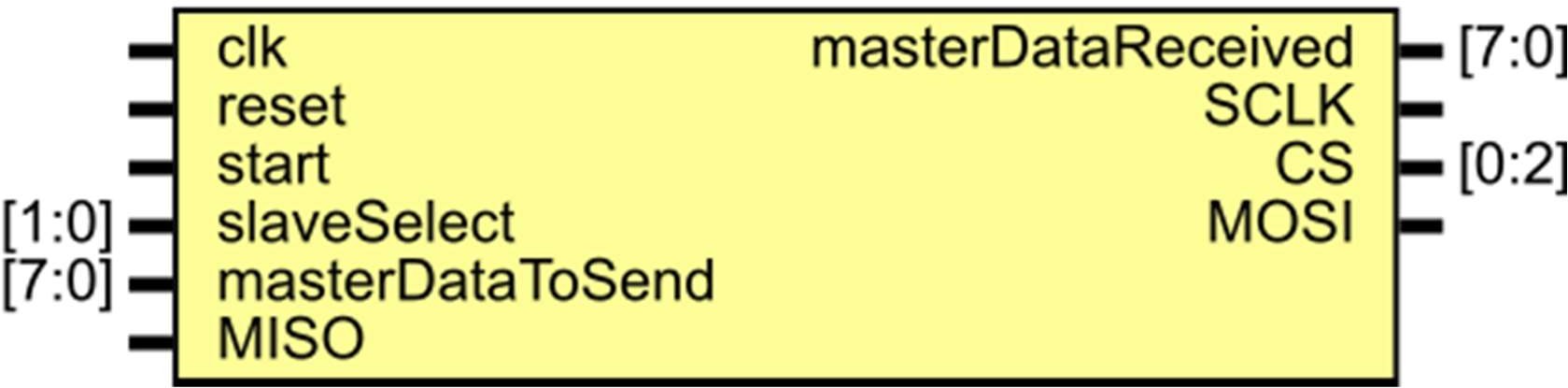


Figure 1: Master Prototype

#### Ports:

|  |  |  |
| --- | --- | --- |
| **Port Name** | **Type / Size** | **Description** |
| clk | Input | The synchronous clock generated from the Microcontroller feeding the master of SPI. |
| reset | Input | Resets the Register’s Data to 0 (Asynchronous reset). |
| start | Input | The bit which trigger the transmission and receiving process. |
| slaveSelect | Input [1:0] | The number of the slave to be connected to. |
| masterDataToSend | Input [7:0] | The 8-bits data we put in the Register to be sent in the transmission process. |
| MISO | Input | The bit received by the master from slave at each clock pulse. |
| masterDataRecieved | Output reg [7:0] | The 8-bits data which The Slave Receives at the end of receiving process. |
| SCLK | output | The Synchronous clock of The Master. |
| CS | Output [0:2] | Enable bit of The Transmission (Active-Low). |

|  |  |  |
| --- | --- | --- |
| MOSI | Output reg | The bit transmitted to the slave from the master at each clock pulse. |

1. **Local and Unported Signals:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type / Size** | **Description** |
| counter | integer | A counter that counts the number of bets |
| flag | reg | A bit raised to high when the  transmission and receiving process are terminated. |
| buffer | Reg [7:0] | * The Register inside the master * A normal shift Register with MISO as its IL and MOSI as its   IR. |

1. **Processes and The Data Flow:**
   1. At the raising edge of start:
      * The master read the slaveSelect.
      * Sent the CS bit for each slave (in the same time)
      * Initialize the masterDataRecieved by unknowns.
      * Initialize the buffer by masterDataToSend.
      * Initialize the counter by 1
      * Initialize the flag by zero
   2. At the rising edge of SCLK (shifting process):
      * Shifting the LSB in the buffer into the MISO.
   3. At the falling edge of SCLK (Sampling process):
      * Shifting the MOSI bit into the Register from the left.
      * Updating the SlaveDataReceived to contain the same bits as the Register.
   4. At the raising edge of flag:
      * Both transmission and receiving process are terminated.

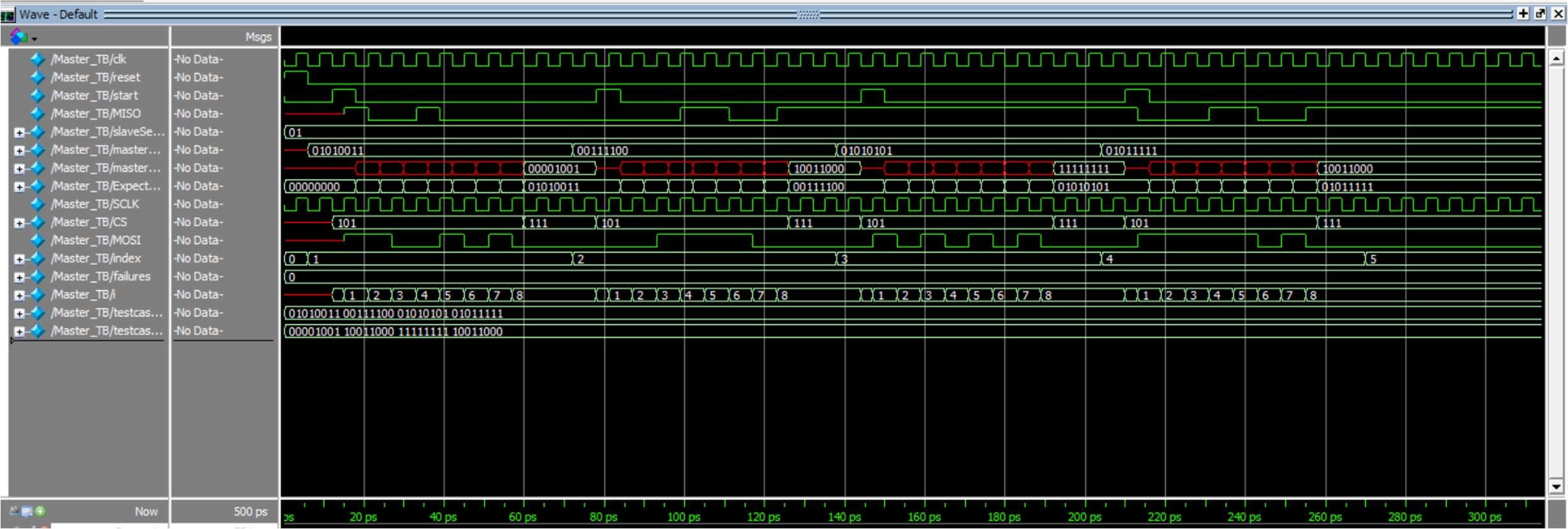
### PART II: Self-Checking Testbench for the Master Module

#### Local and Unported Signals:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **CLK** | **reg** | **Input to Master** |
| **Start** | **reg** | **Input to Master** |
| **Slave Select** | **reg [1:0]** | **Input to Master** |
| **reset** | **reg** | **Input to Master** |
| **MISO** | **reg** | **Input to Master** |
| **SCLK** | **wire** | **Output from Master** |
| **CS** | **wire [0:2]** | **Output from Master** |
| **MOSI** | **wire** | **Output from Master** |
| **masterDataToSend** | **reg [7:0]** | **Input to Master** |
| **masterDataReceived** | **wire [7:0]** | **output from master module** |
| **ExpectedSlaveDataToReceive** | **reg [7:0]** | **The 8-bits data which the Slave is expected to receive at the end of transmission** |
| **testcase\_SlaveDataToSend** | **wire [7:0] vector**  **[1:4]** | **The 4-element vector of 8-bit data which is expected to be**  **sent by Master in transmission** |
| **testcase\_MasterDataToSend** | **wire [7:0] vector**  **[1:4]** | **The 4-element vector of 8-bit data which is an input to Master in transmission (masterDataToSend)** |

1. **Processes and The Data Flow:**
   1. Initializing Local and unported signals:
      * Initialize start with 0 then 1 (triggering the start signal to start transmission).
      * Initialize input data loaded to Master and expected data to receive.
      * Initialize slave select.
   2. At the rising edge of CLK (mimicking shifting process of Slave):
      * Setting MISO with 𝑖th bit of testcase\_SlaveDataToSend.
      * Increasing i by 1.
   3. At the falling edge of CLK (sampling process):
      * Updating the ExpectedSlaveDataToReceive to contain all the bits shifted by Master and sampled by Slave.

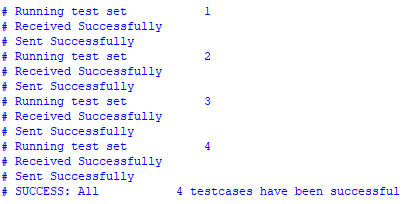
#### Wave of Master simulation



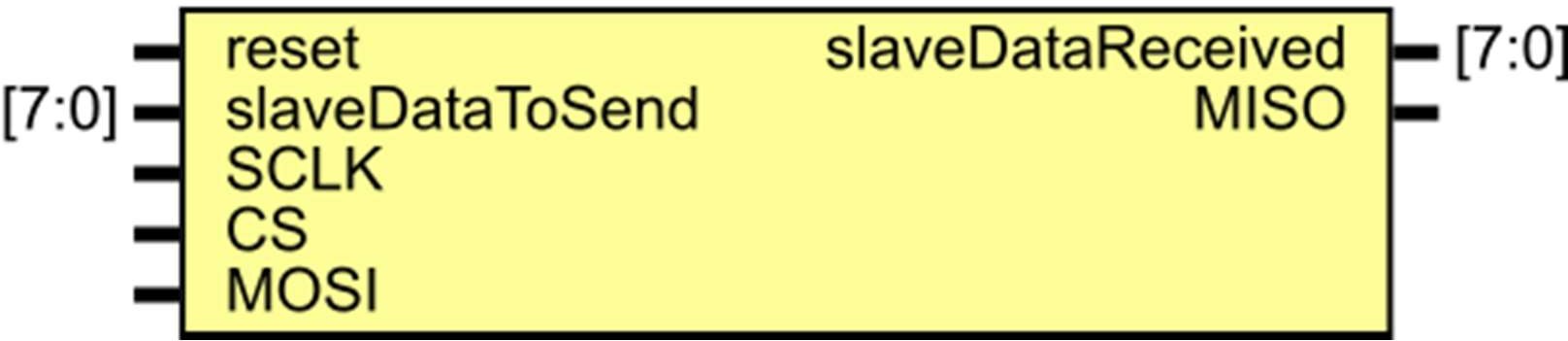
*Figure 2 : Wave form*

**D- Transcript**

*Figure 3 : Transcript*



### Part III: Slave Module



*Figure 4 Slave*

#### Ports:

|  |  |  |
| --- | --- | --- |
| **Port Name** | **Type / Size** | **Description** |
| reset | Input | Resets the Register’s Data to 0 (Asynchronous reset). |
| slaveDataToSend | Input [7:0] | The 8-bits data we put in the Register to send in the transmission process. |
| slaveDataReceived | output reg [7:0] | The 8-bits data which The Slave  Receives at the end of transmission. |
| SCLK | Input | The Synchronous clock of The Master. |
| CS | Input | Enable bit of The Transmission (Active-Low). |
| MOSI | Input | The bit received by slave from the master at each clock pulse. |
| MISO | Output reg | The bit transmitted to the master from slave at each clock pulse. |

1. **Local and Unported Signals:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type / Size** | **Description** |
| Reg Data | reg [7:0] | * The Register inside The Slave * A normal shift Register with MOSI as its IL and MISO as its IR. |

1. **Processes and The Data Flow:**
   1. At the falling edge of CS:
      * Indicates the start of the transmission.
      * The Slave loads the data that will be transmitted into the register.
   2. At the rising edge of CS:
      * Indicates the end of the transmission
      * Setting MISO to high-z.
   3. At the rising edge of SCLK (shifting process):
      * Shifting the LSB in the register into the MISO.
   4. At the falling edge of SCLK (Sampling process):
      * Shifting the MOSI bit into the Register from the left.
      * Updating the SlaveDataReceived to contain the same bits as the Register.

### Part IV: Self-Checking Testbench for the Slave Module

#### Local and Unported Signals:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| reset | reg | Input to Slave |
| SCLK | reg | Input to Slave |
| CS | reg | Input to Slave |
| MOSI | reg | Input to Slave |
| MISO | wire | output from Slave |
| slaveDataToSend | reg [7:0] | Input to Slave |
| slaveSataReceived | wire [7:0] | output from slave module |
| ExpectedMasterDataToReceive | reg [7:0] | The 8-bits data which the Master is expected to receive at the end of transmission |
| testcase\_MasterDataToSend | wire [7:0] vector [1:4] | The 4-element vector of 8-bit data which is expected to be sent by  Master in transmission |
| testcase\_SlaveDataToSend | wire [7:0] vector [1:4] | The 4-element vector of 8-bit data which is an input to Slave in transmission (slaveDataToSend) |

1. **Processes and The Data Flow:**
   1. Initializing Local and unported signals:
      * Initialize CS with 1 then 0 (triggering falling edge of CS).
      * Initialize input data loaded to Slave.
   2. Set CS =1:
      * Triggering rising edge of CS.
      * Checking if data received and sent by Slave is as Expected and display an indicative message.
   3. At the rising edge of SCLK (mimicking the shifting process of Master):
      * Setting MOSI with 𝑖th bit of testcase\_MasterDataToSend.
      * Increase i by 1.
   4. At the falling edge of SCLK (sampling process):

- Updating the ExpectedMasterDataToReceive to contain all the bits shifted by Slave and sampled by Master.

#### Simulation of Slave:

*Figure 5: Wave form*

#### 

Figure 6: Slave Displayed Messages

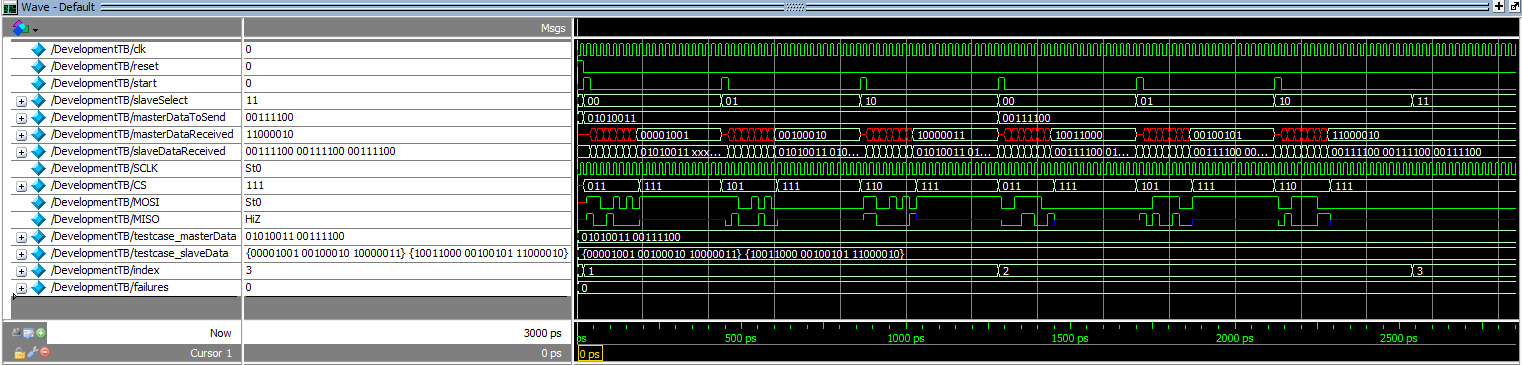
**D.Simulation of Development TestBench:**

Figure 7 : Development Wave Form

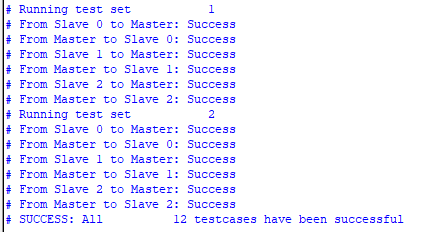


Figure 8 : Development Displayed Message