Ain Shams University
Faculty of Engineering
Computer and Systems Department

CO 2 Project

PCI Slave In Verilog

Team: 31

Team Members:-

1700106	**أحمد عاطف محمد سعيد محمد**
1700204	**احمد مصطفى نيازى عبدالقادر**
1700477	**خالد طارق عبدالفتاح ابراهيم**
1701427	**مصطفى سمير محمد موسى محمد**
1701444	**مصطفى محسن عبدالفتاح الديب**

Table of content:

1.0 BLOCK DIAGRAMS

1.1 Block Diagram

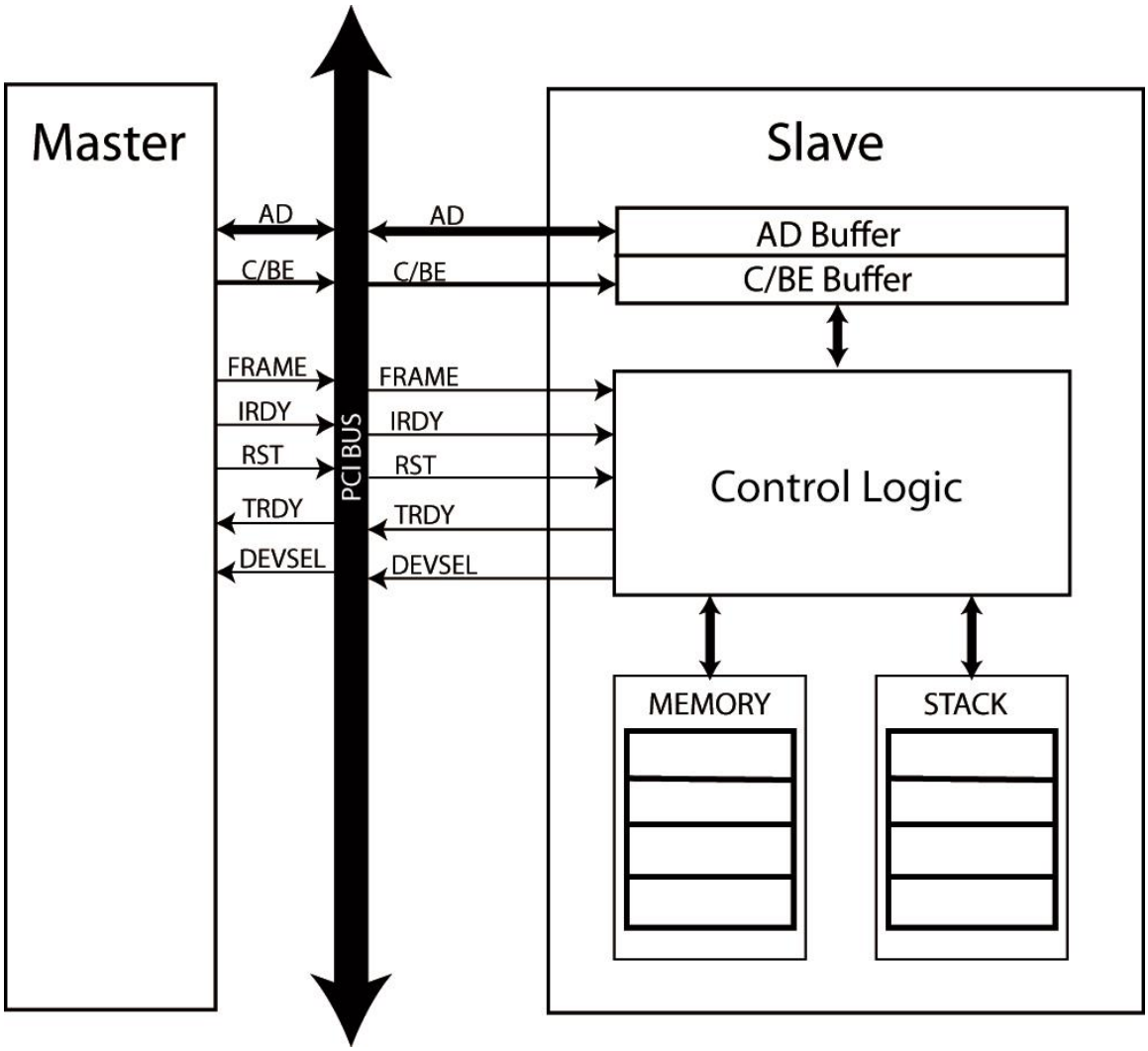

Figure 1-BLOCK DIAGRAM

1.2 Block Diagram In Details

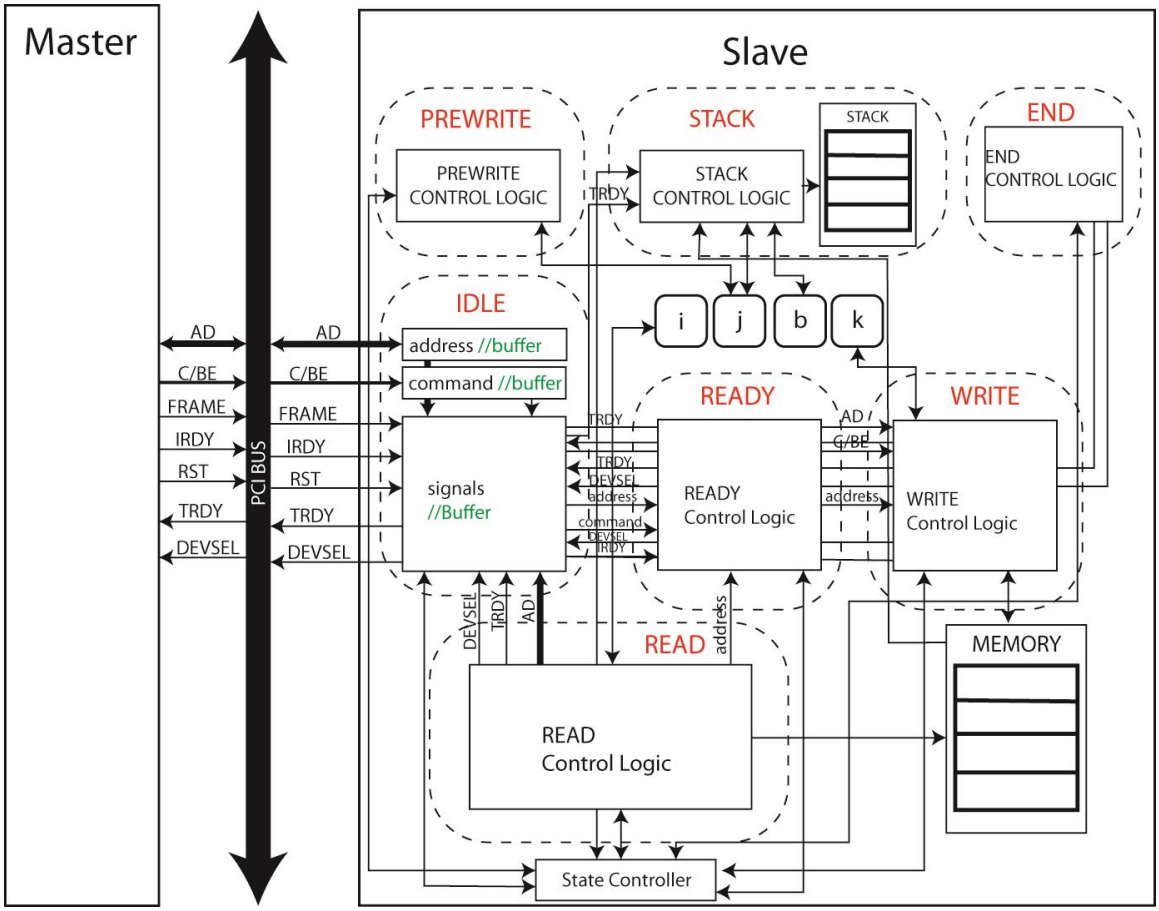

Figure 2-Block Diagram In Details

2.0 SIGNALS DESCRIBTION

Signal	Description
PCI BUS Signals	
AD[31:00]	Stands for Address & Data Inout port used to transfer the address from MASTER to SLAVE and transfer Data in both directions. When the **FRAME** is asserted the **AD** port acts as input to SLAVE to check the address of the device. When the **IRDY** & **TRDY** is asserted The **AD** port acts as output from SLAVE in *READ* operation and input in *WRITE* operation.
C/BE	Stands for Bus Command & Byte Enables. Input port used to tell SLAVE what's the command operation. And to tell the SLAVE what bytes It's allowed to take.
FRAME	Driven by MASTER to indicate the beginning the end of access. When **FRAME** is asserted the SLAVE take the address and the command and put them in buffers. Then the SLAVE goes to *READY* state waiting the MASTER to assert **IRDY**. When **FRAME** is 1 again SLAVE's state will be *IDLE*.
IRDY	Stands for Initiator Ready Indicate the MASTER finished sending the address and the command and ready to send or get Data.
TRDY	Stands for Target Ready Indicate the SLAVE ability to get or send data.
DEVSEL	Stands for DEVICE SELECT Indicate that the SLAVE has known that it's been selected.
CLK	Stands for Clock. The device responds to MASTER signals on CLK negative edge.
RST	Stands for Reset When asserted the SLAVE return to *IDLE* state
MASTER – Test Bench Signals	
din	Stands for Direct Input

	Used in test bench in assign the value of **AD** when the **AD** acts as input signal.
cbuffer	Stands for Command Buffer Used in test bench to save the operation command when **IRDY** is asserted. We do this we prevent the conflict between byte enable and command in **C/BE** signal in *WRITE* operation
mem	SLAVE'S memory consists of 4 registers.
stack	A memory used to save the data in SLAVE's memory before writing.
address	A buffer to save the address from **AD** in *READY* state.
command	A buffer to save the address from **C/BE** in *READY* state.
Parameters	
Command Parameters	
READ_CMD	Equal in binary 0010. Used to recognize the read command.
WRITE_CMD	Equal in binary 0011. Used to recognize the write command.
State Parameters	
IDLE	Equal in binary 000 The original state. The SLAVE waits the MASTER to assert **FRAME**.
READY	Equal in binary 111. The SLAVE waits the MASTER to assert **IRDY** to catch address and the command. And to indicate the next state.
READ	Equal in binary 001. MASTER reads SLAVE's memory.
PREWRITE	Equal in binary 101. SLAVE indicates whether it needs to transfer data to stack or not.
WRITE	Equal in binary 010. MASTER writes in SLAVE's memory in this state.
STACK	Equal in binary 011. In this state the SLAVE transfer the existing data in memory to stack to not lose them.
END	Equal in binary 100. The last state used after *READ* & *WRITE* operation the SLAVE return into *IDLE* state after it.

SLAVE Signals	
state	Used in SLAVE controller to save the current state.
i	Used in SLAVE controller to read the all data in Memory When MASTER *READ* every word , **i** increases 1.
j	Used in SLAVE controller to transfer the current data in memory to the stack. When word transfer from memory to stack , **j** increases 1.
k	Used in SLAVE controller to write the all data in Memory When MASTER *WRITE* every word , k increases 1.
b	Stands for Base Used to not write on the same word in stack again.

3.0 SCENARIOS

SLAVE memory Content

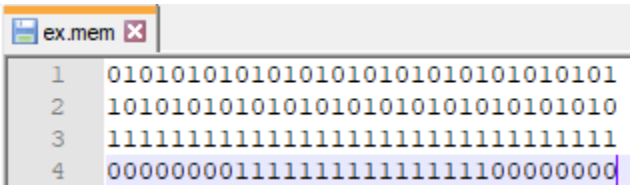

Figure 3-Memory content

ex.mem file is used to initiate the content of the memory.

Figure 3 shows the content in the memory.

3.1 READ Scenarios

3.1.1 Full Read Scenario

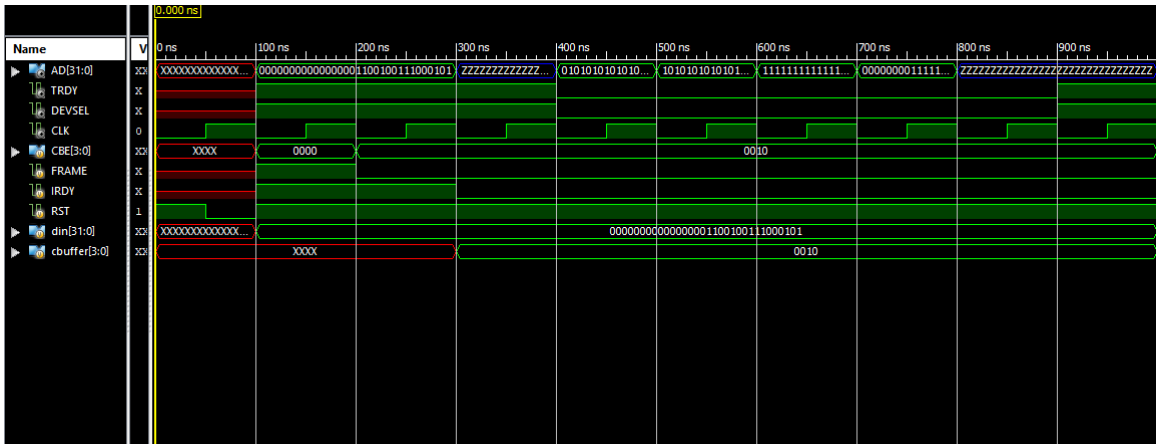

Figure 4- Full Read Scenario Timing Diagram

In this scenario the MASTER reads the 4 bits of the SLAVE's memory. First the MASTER sends the address and the command then the AD converted from input to output to show the content of data as shown in figure 4.

3.1.2 FRAME is up while Reading

Figure 5 FRAME is up while Reading Timing Diagram

In this scenario the MASTER reads the 2 bits of the SLAVE's memory. As the FRAME goes up in the clock no 6.

First the MASTER sends the address and the command then the AD converted from input to output to show the content of data as shown in figure 5. Then the SLAVE goes into IDLE state waiting the FRAME to be low again.

3.1.3 CBE is changed while Reading

Figure 6 CBE is changed while Reading Timing Diagram

In this scenario the MASTER reads the 4 bits of the SLAVE's memory. There's no effect from changing the C/BE while reading. SLAVE ignored it. First the MASTER sends the address and the command then the AD converted from input to output to show the content of data as shown in figure 6.

3.1.4 Read in between address

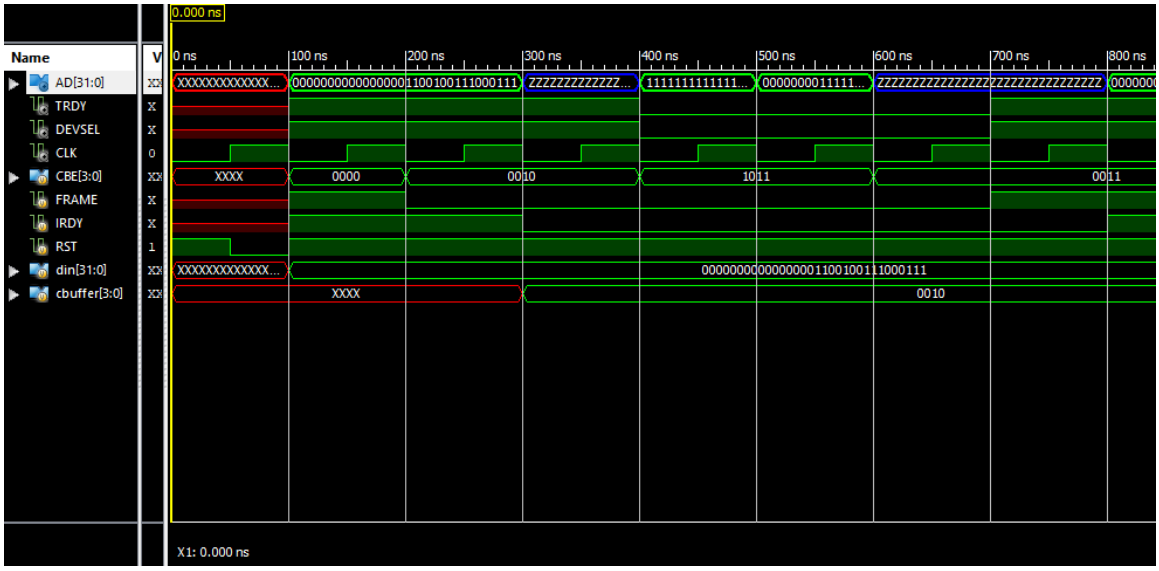

Figure 7 Read in between address Timing Diagram

In this scenario the MASTER reads the last 2 bits of the SLAVE's memory. First the MASTER sends the address of register #3 and the command then the AD converted from input to output to show the content of data as shown in figure 7.

3.2 WRITE Then READ Scenarios

3.2.1 Full Read Full Write Scenario

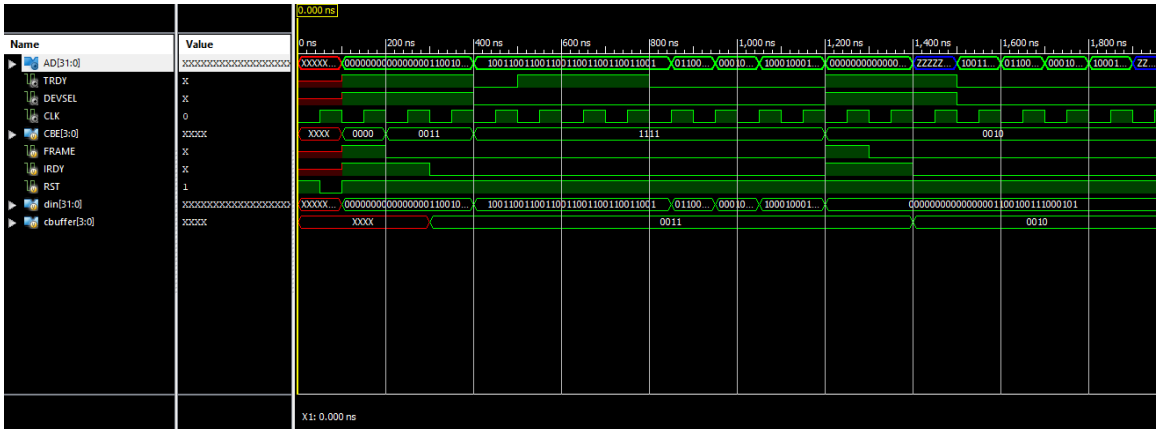

Figure 8 Full Read Full Write Scenario Timing Diagram

In this scenario the MASTER writes the 4 bits of the SLAVE's memory. Then Read the content of the memory.

figure 8 shows that the TRDY signal is Up for 3 clock cycles. Because SLAVE is transferring the existing data in memory into its stack. Then FRAME is up to announce the end of writing then is lowed again to begin reading the content of data

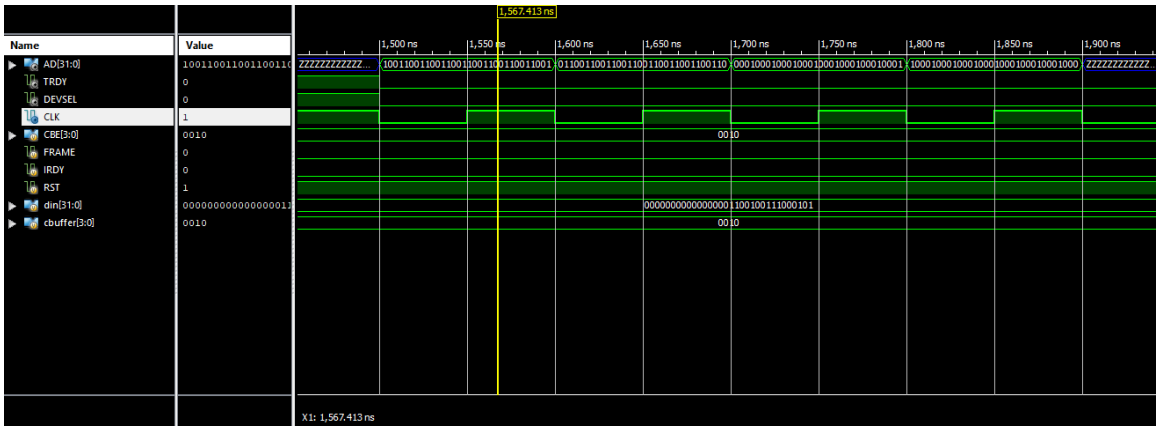

Figure 9 the content of memory after writing Timing Diagram

Figure 9 show the read operation of the memory after writing.

3.2.2 Full Read Full Write Scenario with different Byte Enable

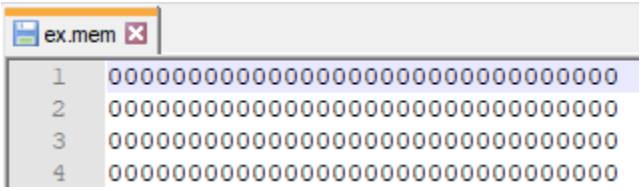

Figure 10 the new content of memory

In this scenario we need to make memory as shown in figure 10. To show the effect of byte enable.

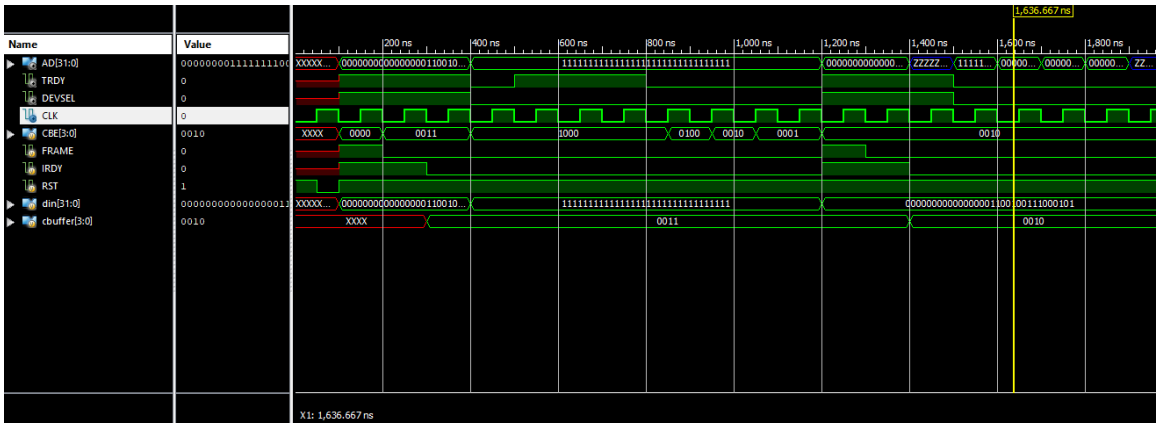

Figure 11 2 Full Read Full Write Scenario with different Byte Enable Timing Diagram

In this scenario the MASTER writes the 4 bits of the SLAVE's memory. Then Read the content of the memory.

Figure 11 shows that the Data in AD port is identical we just changed the byte enable signal in C/BE.

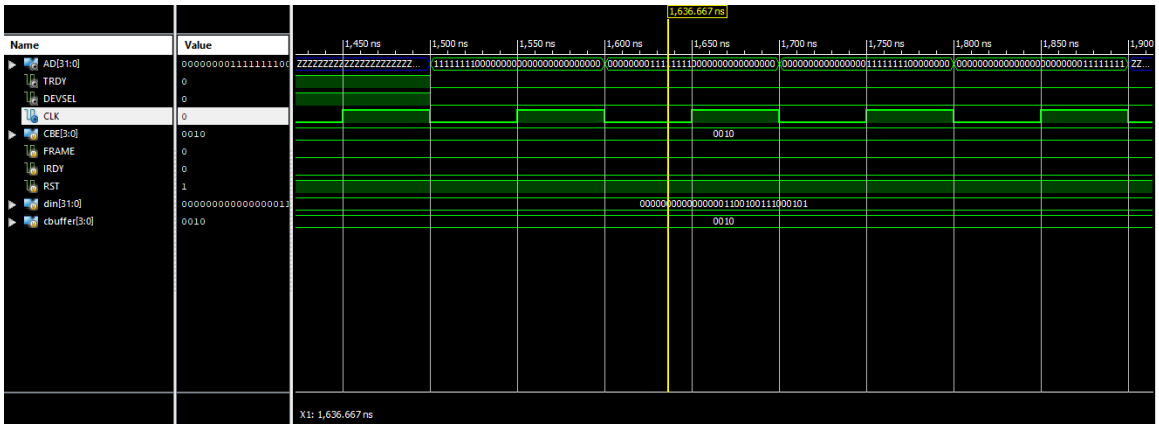

Figure 12 the content of memory after writing Timing Diagram

Figure 12 shows the read operation of the memory after writing.

We can see that the only written bits is the bits we enabled in write operation.

3.2.3 FRAME is up while WRITING

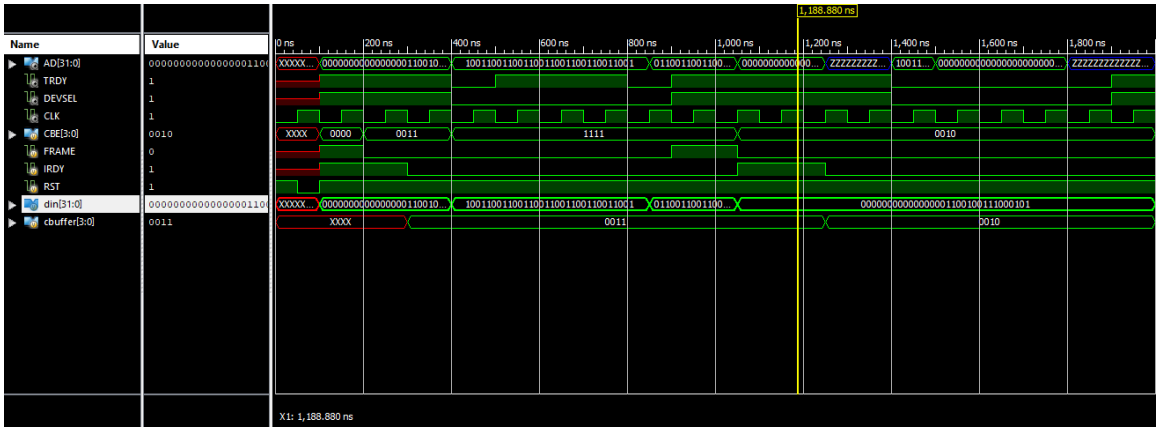

Figure 13 FRAME is up while WRITING Timing Diagram

Figure 13 shows that in this scenario the MASTER writes only 1 bits of the SLAVE's memory after that FRAME goes high so the write stops. Then MASTER Read the content of the memory.

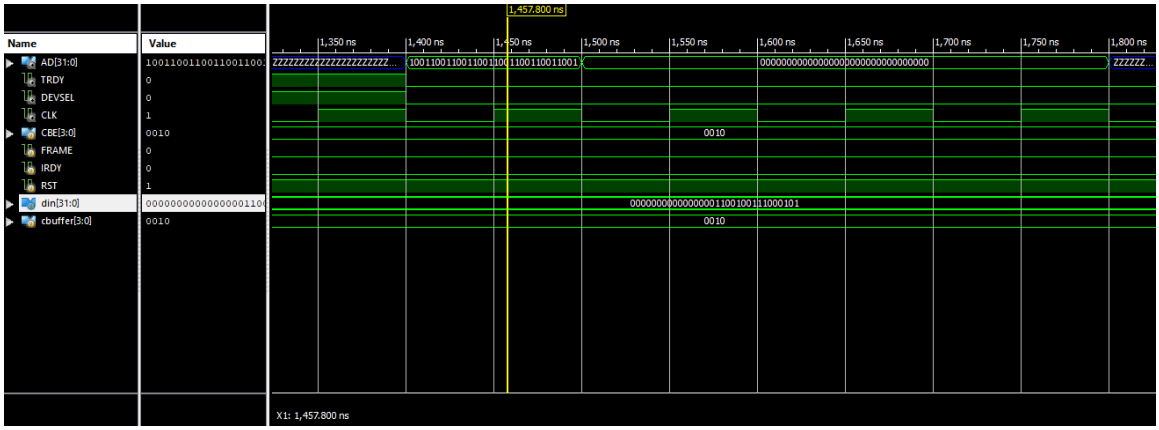

Figure 14 the content of memory after writing Timing Diagram

Figure 14 shows the content of the memory. Only the first register has been written.

3.2.4 Write in specific address

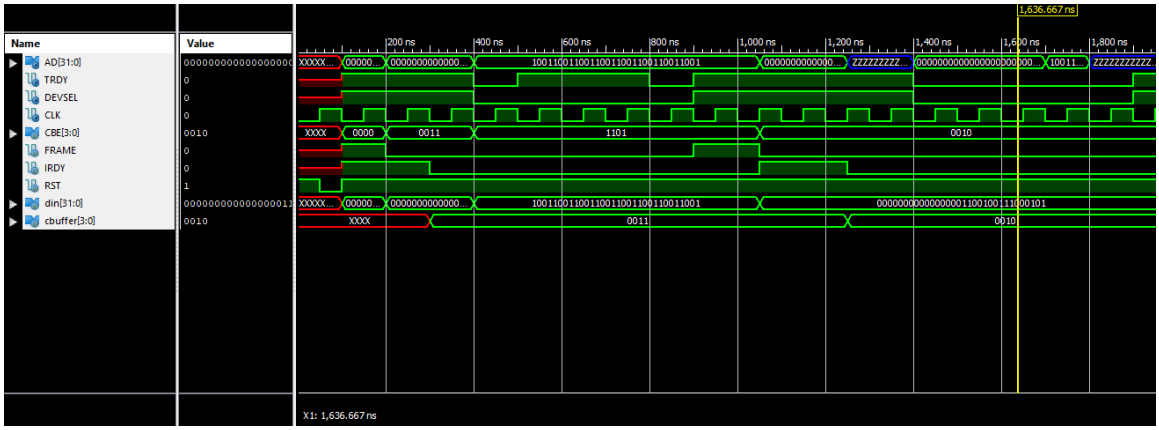

Figure 15 Write in specific address Timing Diagram

Figure 15 shows that in this scenario the MASTER writes only the last bit of the SLAVE's memory as the register called is the last one.

Then MASTER Read the content of the memory.

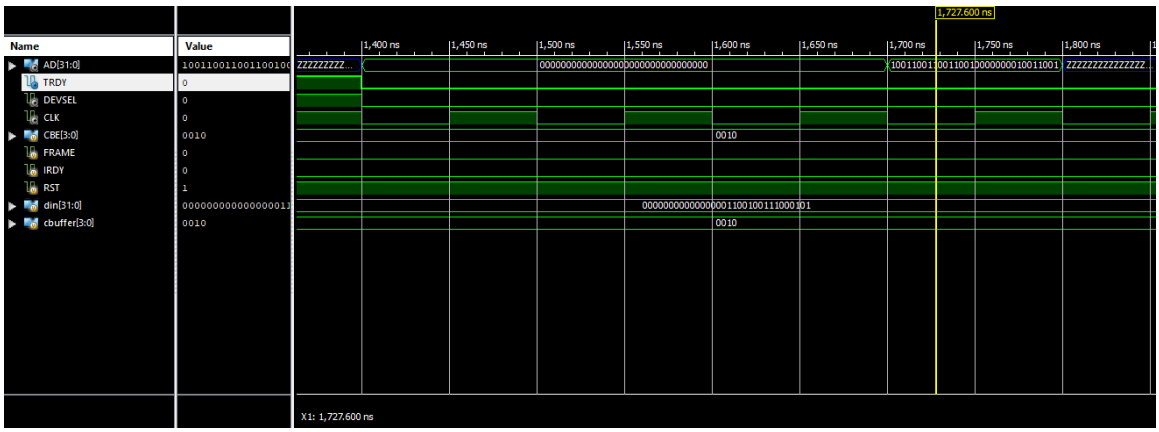

Figure 16 the content of memory after writing Timing Diagram

Figure 16 show the content of the memory.

Only last bit is written.

3.4 Wrong Signals Scenarios

3.4.1 Wrong address

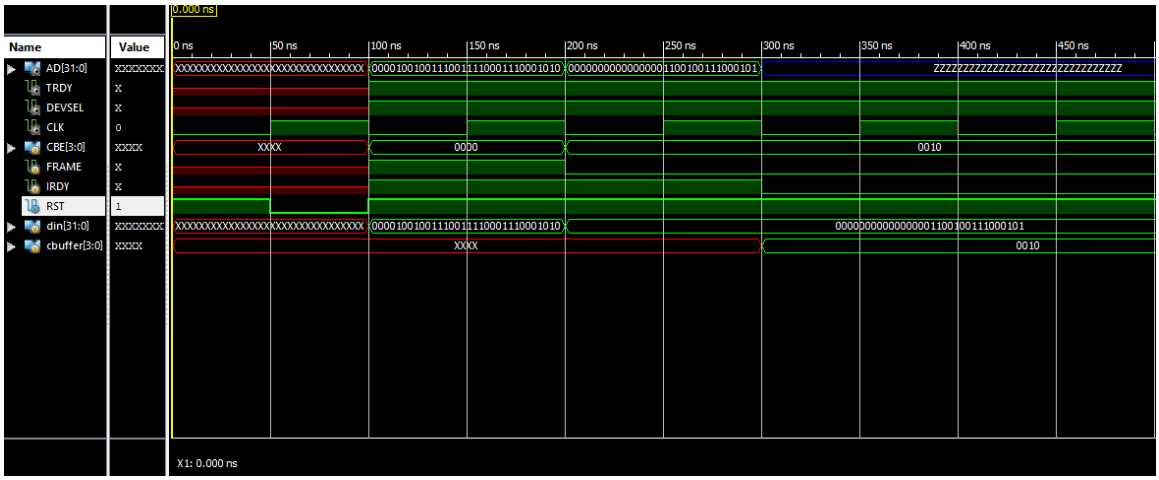

Figure 17 Wrong address timing diagram

As shown in Figure 17 the SLAVE didn't lowed TRDY & DEVSEL as the address is wrong so SLAVE ignored the FRAME signal.

3.4.2 FRAME wasn't asserted

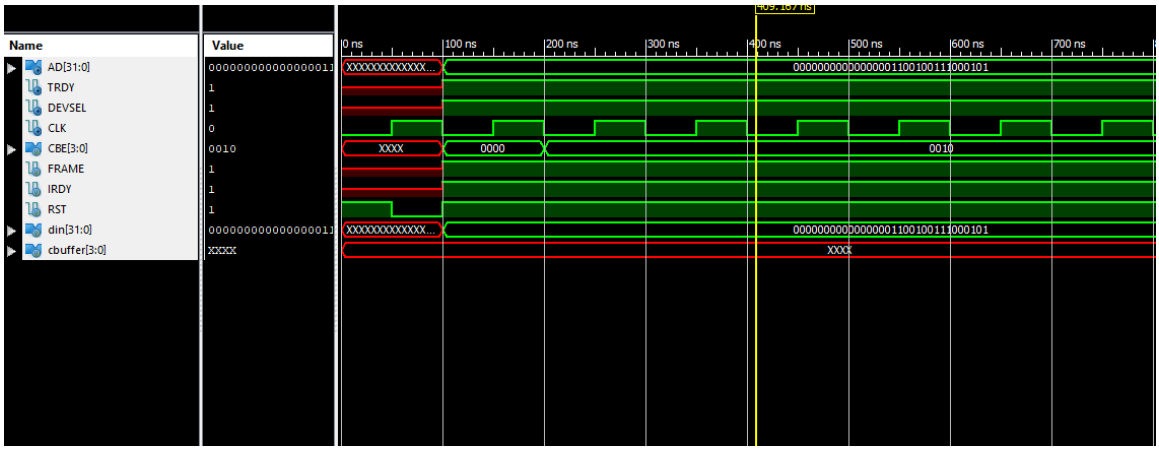

Figure 18 FRAME wasn't asserted timing diagram

In figure 18 Even the address and command are correct but the FRAME signal wasn't asserted. So SLAVE won't respond.

4.0 Enhancements Made

- Design the SLAVE as Finite State Machine.
- Transfer the data in memory into stack to not lose them.
- Call specific register address to read or write from it.

5.0 Contribution

Preparing Presentation **Debugging the Code**	أحمد عاطف محمد سعيد محمد
Preparing the Report **Writing Scenarios & Test Bench**	احمد مصطفى نيازى عبدالقادر
Preparing Presentation **Debugging the Code**	خالد طارق عبدالفتاح ابراهيم
Writing Source Code **Writing Scenarios & Test Bench**	مصطفى سمير محمد موسى محمد
Writing Source Code **Writing Scenarios & Test Bench**	مصطفى محسن عبدالفتاح الديب