Ain Shams University
Faculty of Engineering
3rd.Electrical Department
Computer System Engineering

PCI PROJECT

By:

GROUP (1)

Submitted To:

Computer Organization T.S

Cairo 2018

TABLE OF CONTENTS

Team Members

NAME	SECTION	ID
Mohammed Hesham Hassan	3	33883
Omar Ahmad Mahmoud	2	33830
Omar Ibrahim Mohamed	2	33829
Ammar Yasser Baraka	2	33828
Lamees Emad	2	33850
Matthew Emile	2	33852

Design Choices

Module Name	Device
Outputs	• REQ: 1 bit signal going to the arbiter to request the bus
Inputs	• GNT: 1 bit signal coming from the arbiter to grant this device the bus • CLK: The clock syncing the whole system • RESET: 1 bit signal to reset the device • forceReq: 1 bit signal for testing purposes to trigger the request signal

	• phaseWire: 2 bits input for testing purposes to set the number of data phases in each transaction • address: 32 bits input for testing purposes to set the address of the device • addressToContact: 32 bit input for testing purpose to set the address of the target device
InOuts	• **AD:** 32 multiplexed bits used to transmit address of the target to be and the data itself • **C/BE:** 4 control bits sent from the initiator to the target to set the type of the transaction (read, write, …) in the address phase, then used to send the byte enable in the data phase to decide which bytes used in read/write • **Frame:** 1 bit signal shared by all devices and arbiter, high means the bus is idle (no initiator), while low means there is a n ongoing transaction • **IReady:** 1 bit signal set by the initiator when it is ready to send/receive data • **TReady:** 1 bit signal set by the target when it is ready to send/receive data • **DevSel:** 1 bit signal set by the target when the address on the AD in the address phase matches the target's address
Internal registers/integers	• **mem:** The internal memory of the device, consists of 10 words • **data:** A small memory containing three words (AAAAAAAA, BBBBBBBB, CCCCCCCC) to decide which word should be the output depending on the address of the device • **i:** A counter for the number of transactions depending on how many cycles the forceReq signal was asserted • **pointer:** Points to the last empty place in the memory (like the stack pointer) • **numberOfTransactions:** Saves the number of transactions counted by the i

	• **posEdge:** Determines which edge of the clock is happening • **isData:** Determines the current phase (address or data) • **isMaster:** Determines of the device is a master (initiator) • **isSlave:** Determines if the device is a slave (target)

Module Name	Arbiter
Outputs	• GNT: 3 bits signal going to each device granting them the bus one at a time
Inputs	• REQ: 3 bits, one coming from each device used to request the bus from the arbiter • CLK: The clock syncing the whole system • RESET: 1 bit signal resets the arbiter by removing all grant signals (setting them to 1)
InOuts	___
Internal registers/integers	___

Test Cases

Wave Diagram Legend

Scenario “1”

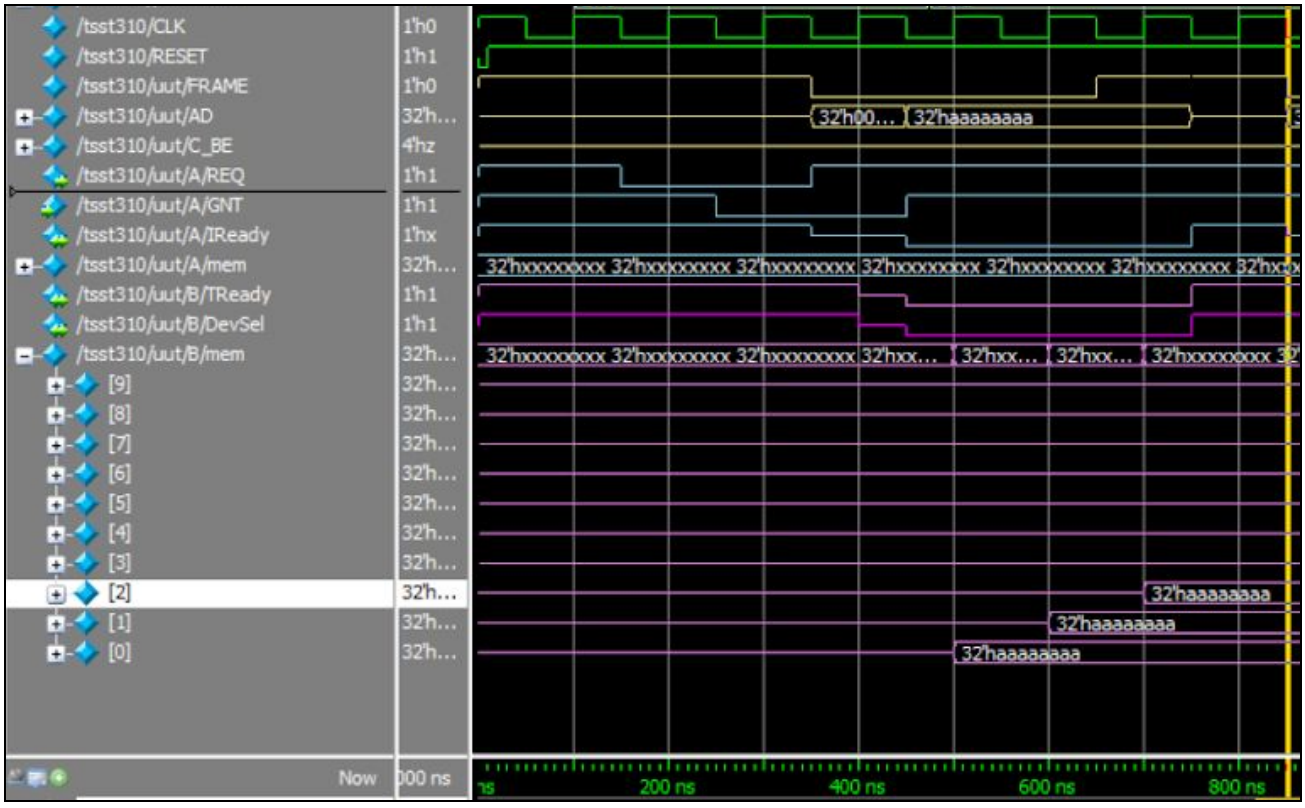
Device A requests the bus in order to communicate with device B and sends three data words in the transaction.

➔ **Sequence of expected events:**

1. Device A asserts its Request signal.
2. Device A receives a Grant signal from the arbiter at the next negative clock edge.
3. Since A is granted the bus and the bus is IDLE at the next clock cycle, device A asserts the global Frame signal and is said to have the bus. At the same clock edge, device A deasserts its Request signal since it does not require more than one transaction. The device also puts the address of the target it wants to write to on the global A/D multiplexed lines.
4. At the next clock cycle, the arbiter reads that the bus is now in use and thus deasserts the Grant signal of device A and grants the bus to the next requesting device. At the same clock cycle, device B, the target, recognizes its address on the A/D lines and therefore asserts its Devsel and TReady signals. Device A also puts the data it wants to write to the target on the A/D lines.
5. All data transfer conditions are met in the next cycle i.e. the Frame signal, IReady and TReady are all asserted. The data word ‘AAAA_AAAA’ is therefore written in the memory of device B over this cycle and the next two.
6. Device A deasserts the Frame signal at the negative edge of the cycle before the positive edge at which the last word of data is transferred.

7. After the transfer of the last word of data, the device A deasserts the IReady signal and removes data from the A/D lines. At the same clock cycle, device B deasserts the TReady and Devsel signals since the data transfer is over.

➔ **Wave Diagram:**



Scenario "2"

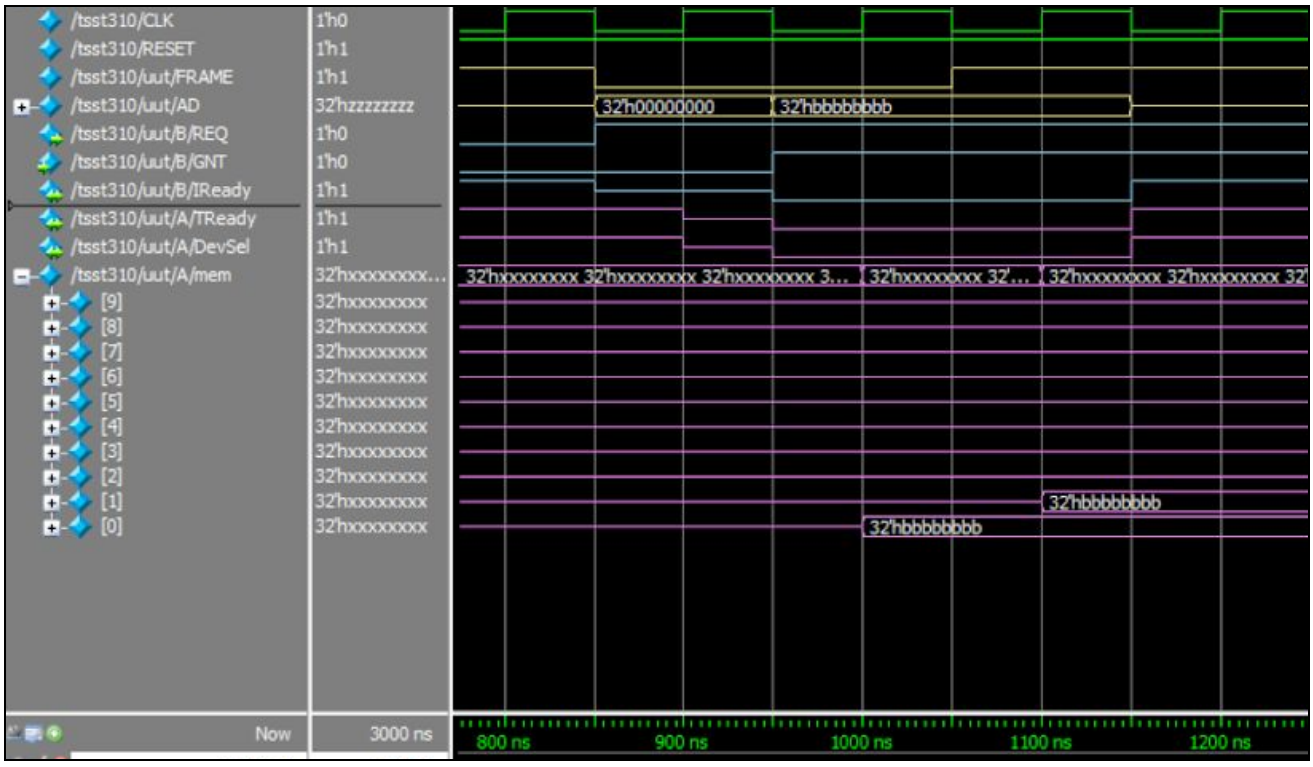
Device B requests the bus in order to communicate with device A and send two data words.

➔ **Sequence of expected events:**

1. Device B asserts its request signal.
2. Device B receives a Grant signal from the arbiter at the next negative clock edge.
3. Since B is granted the bus and the bus is IDLE at the next clock cycle, device B asserts the global Frame signal and is said to have the bus. At the same clock edge, device B deasserts its Request signal since it does not require more than one transaction. The device also puts the address of the target it wants to write to on the global A/D multiplexed lines.

4. At the next clock cycle, the arbiter reads that the bus is now in use and thus deasserts the Grant signal of device Band grants the bus to the next requesting device. At the same clock cycle, device A, the target, recognizes its address on the A/D lines and therefore asserts its Devsel and TReady signals. Device B also puts the data it wants to write to the target on the A/D lines.
5. All data transfer conditions are met in the next cycle i.e. the Frame signal, IReady and TReady are all asserted. The data word 'BBBB_BBBB' is therefore written in the memory of device A over this cycle and the next one.
6. Device B deasserts the Frame signal at the negative edge of the cycle before the positive edge at which the last word of data is transferred.
7. After the transfer of the last word of data, the device B deasserts the IReady signal and removes data from the A/D lines. At the same clock cycle, device A deasserts the TReady and Devsel signals since the data transfer is over.

➔ **Wave Diagram:**

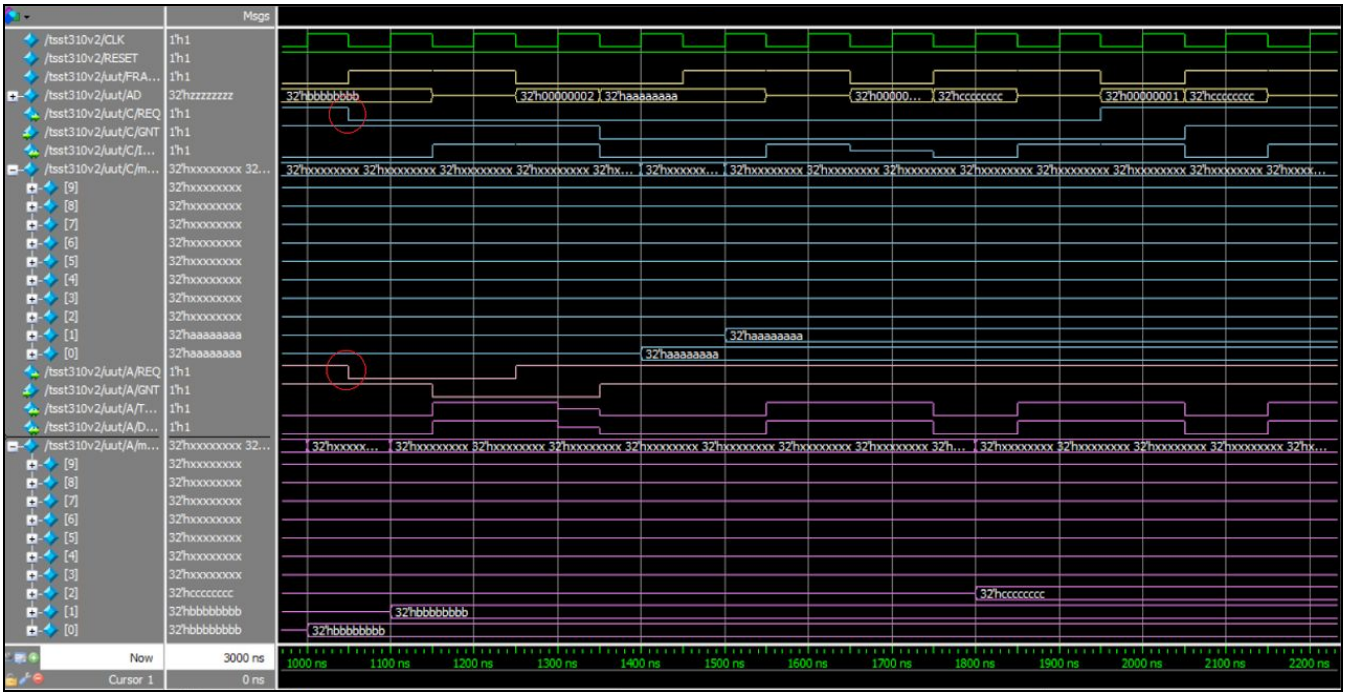


Scenario “3”

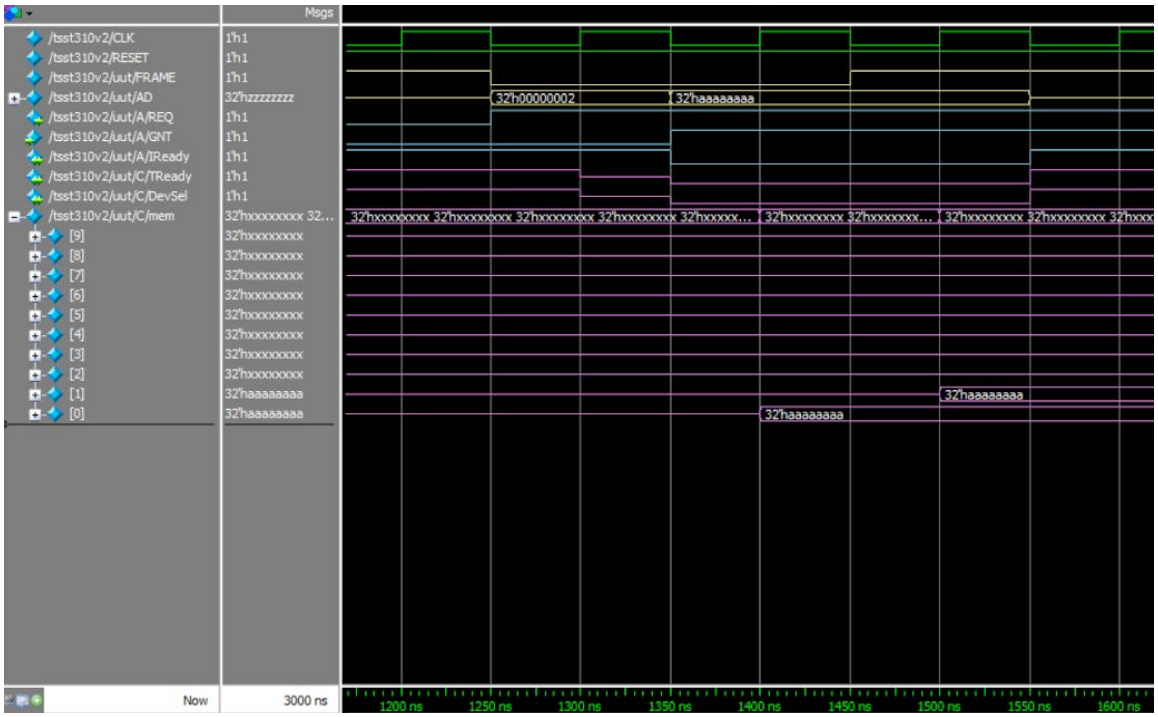
Device C requests the bus for two transactions; One sends two words ‘CCCC_CCCC’ to device A and the other sends one word ‘CCCC_CCCC’ to device B. At the same time device A requests the bus again to communicate with Device C and send two words ‘AAAA_AAAA’. What should happen here is as follows :

- The arbiter will grant device A since it has a higher priority, then device A will send two data words to device C.
- The arbiter must then grant the bus to device C, C now wants to send data to device A, send one word.
- Device C still wants the bus to communicate with device B, and send another data word to B. So it will have the bus for a second transaction since both A & B don’t request the bus any more

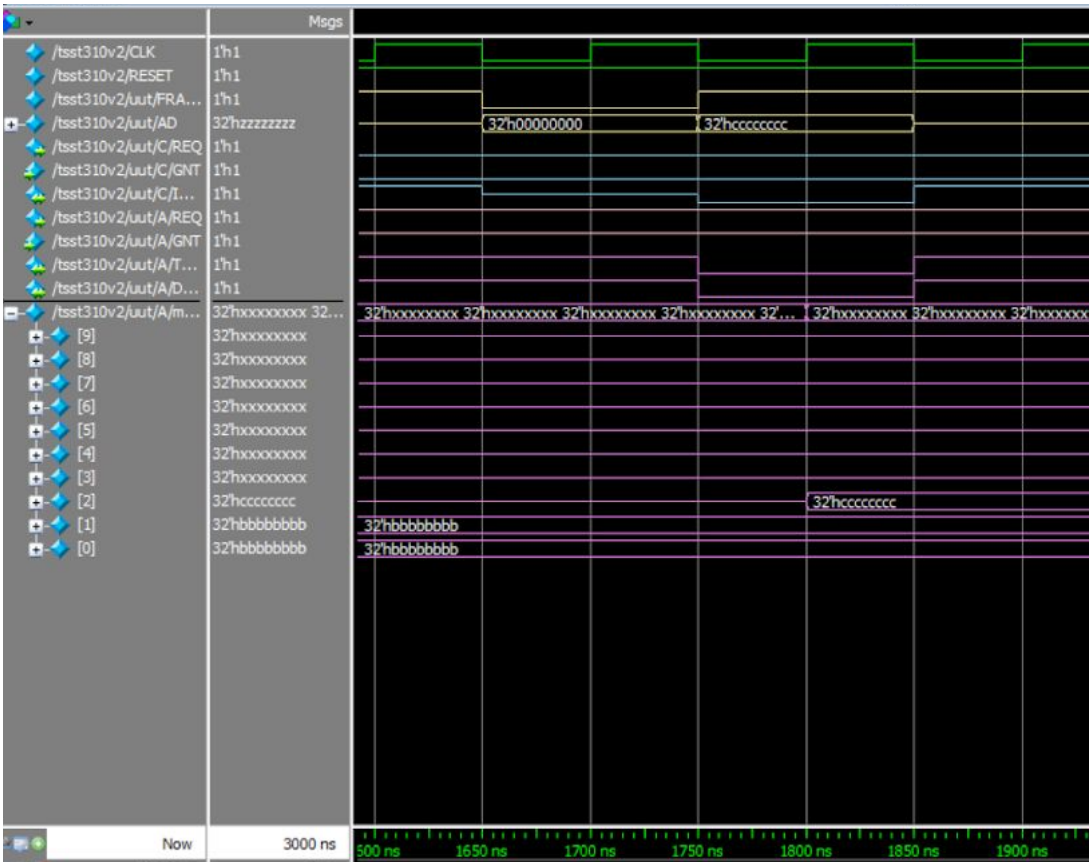
➔ **Wave Diagram “1”: Overview of the Whole Scenario**



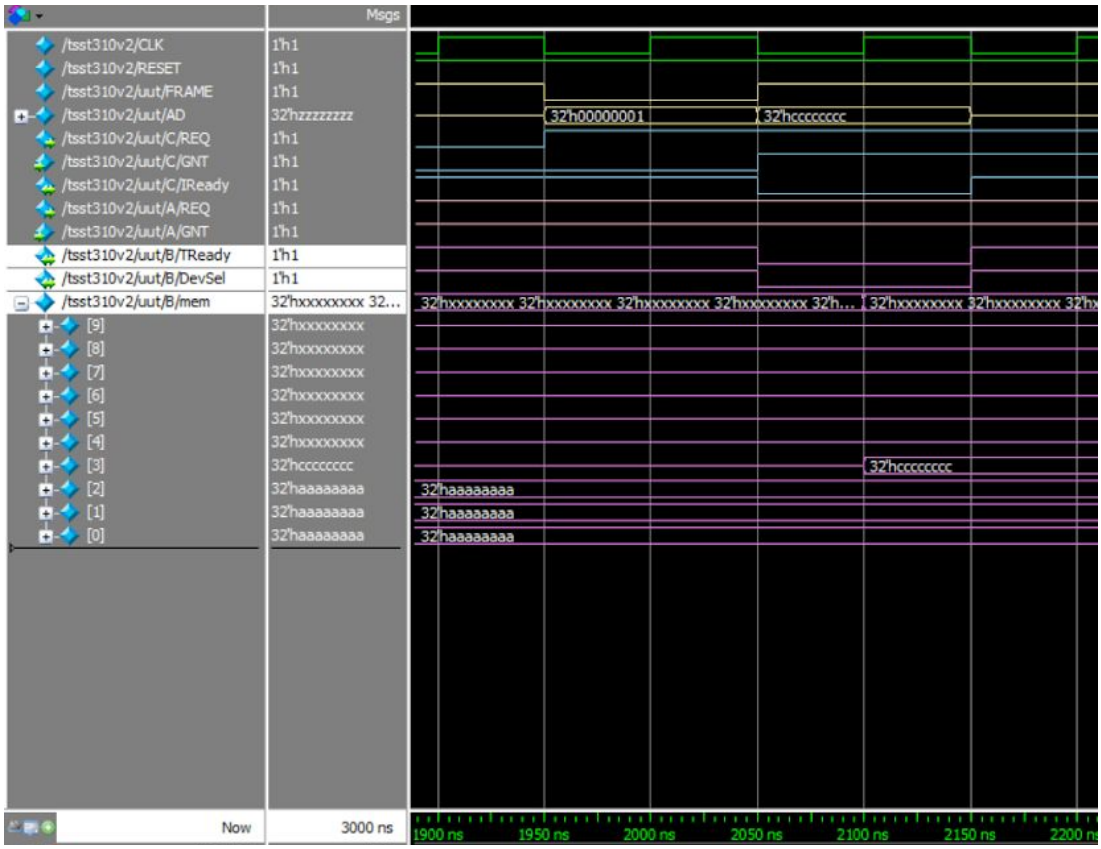
➔ **Wave Diagram “2”: First Transaction of the Scenario (A to C)**



➔ **Wave Diagram "3": Second Transaction of the Scenario (C to A)**



➔ **Wave Diagram “4”: Third Transaction of the Scenario (C to B)**



Extra Scenario “1”

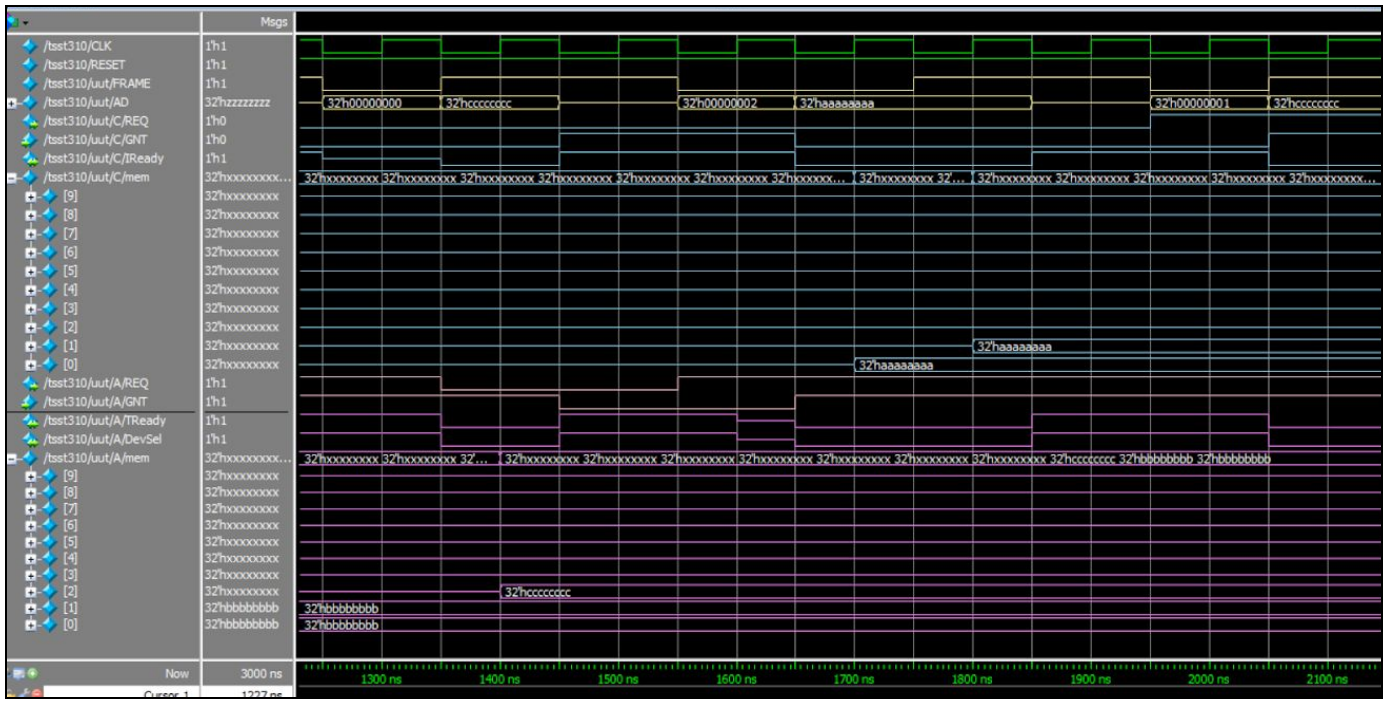
Device C requests the bus for two transactions; The first transaction is sending two words ‘CCCC_CCCC’ to device A, and the second one is sending one word ‘CCCC_CCCC’ to device B. During device C’s first transaction, device A requests the bus in order to send two words ‘AAAA_AAAA’ to device C.

What should happen here is as follows :

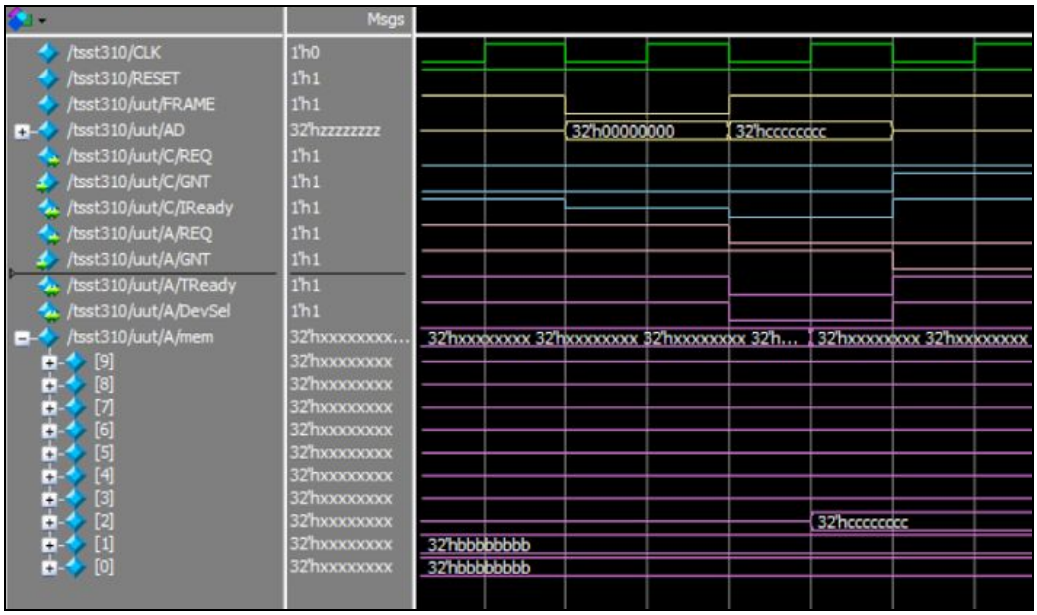
- The arbiter will grant device C the bus for its first transaction. Device C will thus send two data words to device A.
- When the first transaction is over, the arbiter will remove the grant from device C since device A, a device of higher priority, is requesting the bus. The arbiter will grant device A the bus. Device A thus sends two data words to device C.

- Device C still wants the bus to communicate with device B, therefore the arbiter will will give the grant back to device C once device A's transaction is over. Device C sends two data words to device B.

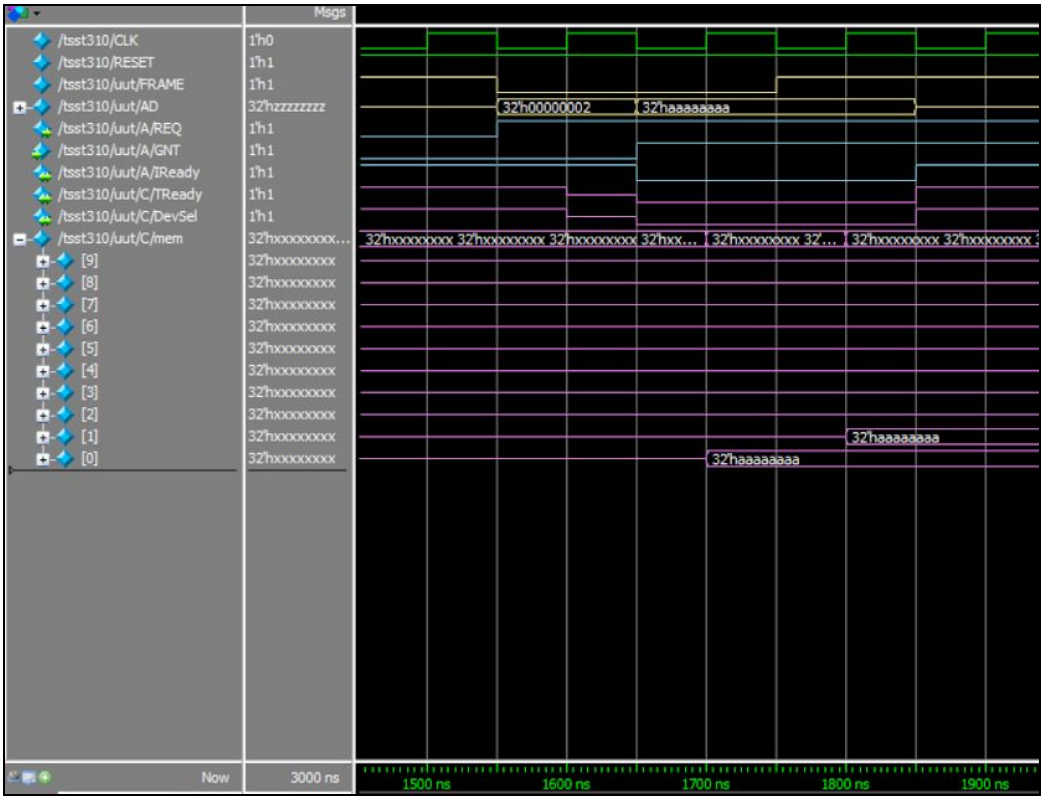
➔ **Wave Diagram "1": Overview of the Whole Scenario**



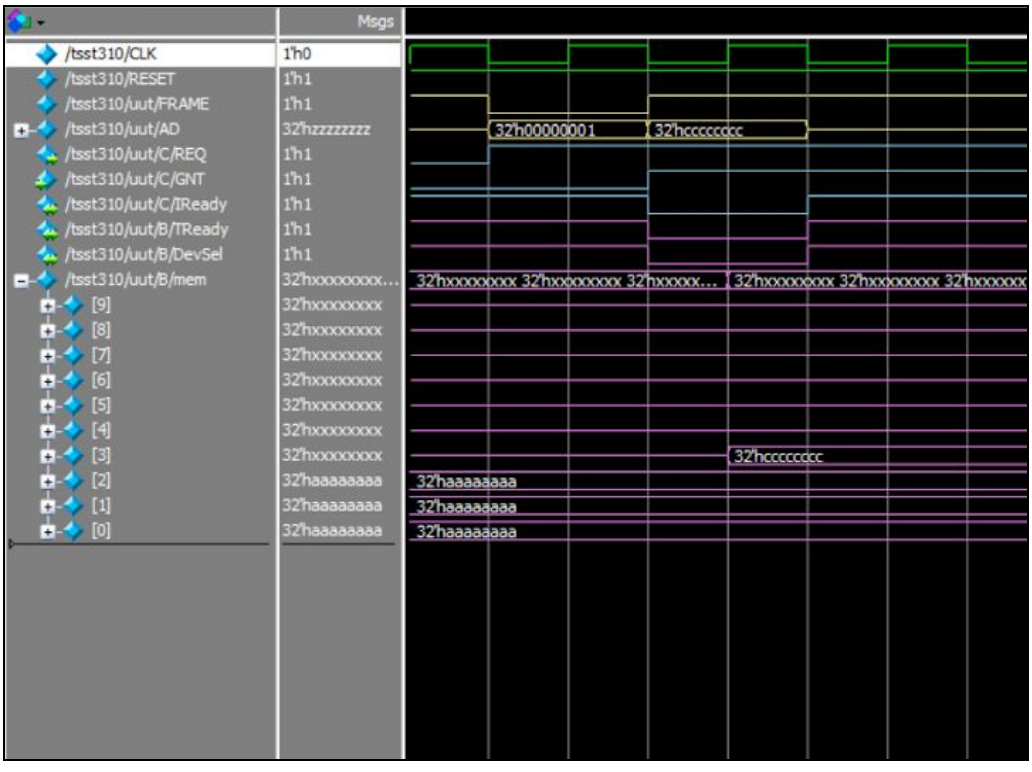
➔ **Wave Diagram "2": First Transaction of the Scenario (C to A)**



➔ **Wave Diagram “3”: Second Transaction of the Scenario (A to C)**



➔ **Wave Diagram “4”: Third Transaction of the Scenario (C to B)**



Other Extra Scenarios

Extra scenarios can be implemented using the GUI.

EasyVerilog IDE

Description

A software version of the protocol is implemented using C# and .NET Technologies, “easyVerilog” is an IDE that attempts to make verilog programmers’ life easier. It features loading, editing, saving of easyverilog and verilog files. It also includes a GUI that dynamically generates a test bench that can be compiled, simulated, and analyzed as a wave diagram.

Functions

The IDE can be divided into a set of functions each has its own inputs and outputs as follow.

Editor

The editor is used to load, edit, and save files. It features a night-mode coloring theme.
It allows multi-opening of files, and quick save.

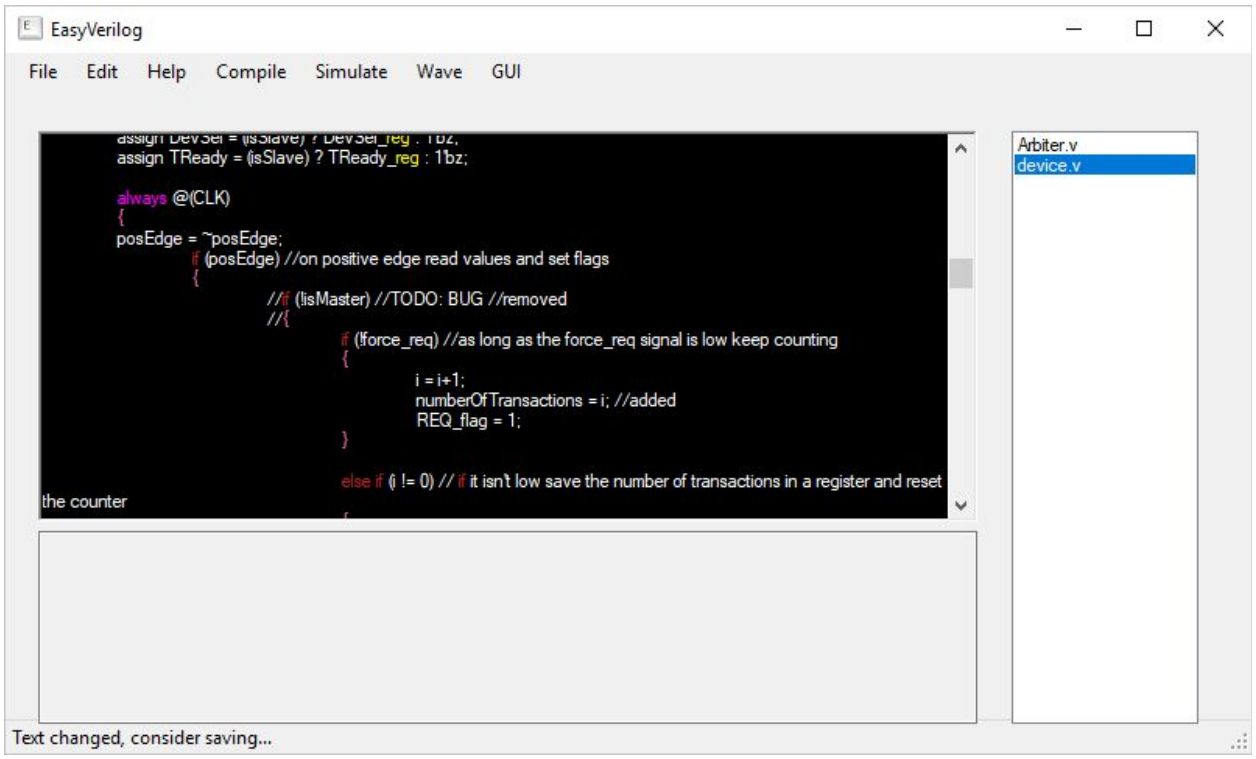
Execution

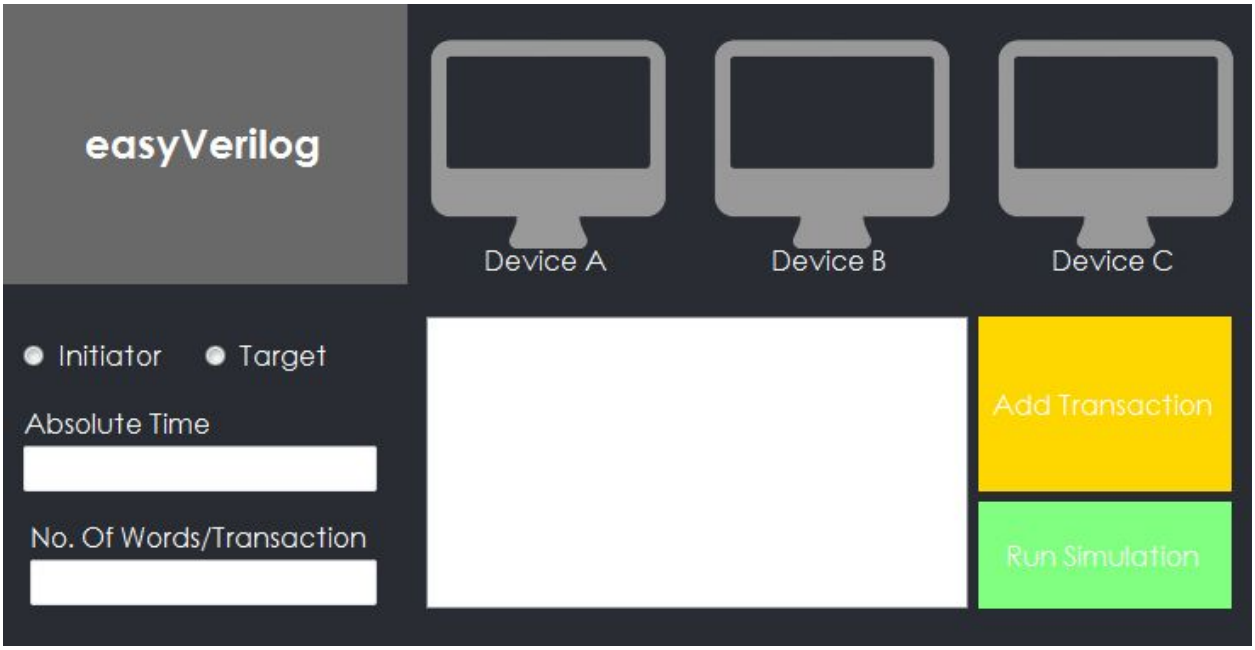
The IDE allows compiling, simulation, and wave analyzing. The whole process is quick and you can choose which actions to take and which actions to do later. At any stage you can see whether its successful and or not and what output was generated.

GUI

The user interface is designed to allow the user to select the Initiator Device, the Target Device, the number of words to be transferred, and the time of transaction through a set of textboxes and buttons.
Then a set of C# functions are executed to generate the testbench verilog code, then the code is passed to the next stage where Icarus verilog compiler is called.

Interfaces


EasyVerilog
File
Edit
Help
Compile
Simulate
Wave
GUI
assign TReady = (isSlave) ? TReady_reg : 1'bz;
always @(CLK)
posEdge = ~posEdge;
if (posEdge) //on positive edge read values and set flags
//if (!isMaster) //TODO: BUG //removed
if (!force_req) //as long as the force_req signal is low keep counting
i = i+1;
numberOfTransactions = i; //added
REQ_flag = 1;
else if (i != 0) // if it isn't low save the number of transactions in a register and reset
the counter
Arbiter.v
device.v
Text changed, consider saving...


easyVerilog
Device A
Device B
Device C
Initiator
Target
Absolute Time
No. Of Words/Transaction
Add Transaction
Run Simulation

Icarus

Icarus Verilog is a free compiler implementation for the IEEE-1364 Verilog hardware description language. Icarus is maintained by Stephen Williams and it is released under the GNU GPL license, we are using icarus as our verilog compiler as it is open source,can be accessed through the terminal and it can be implemented easily through our windows application.

GTKWave

GTKWave is a fully featured GTK+ based wave viewer for Unix, Win32, and Mac OSX which reads LXT, LXT2, VZT, FST, and GHW files as well as standard Verilog VCD/EVCD files and allows their viewing.

Output

Where you specify a sort of scenario to test and the program will draw the timing diagram cycle by cycle, showing (by colors) which objects are currently communicating with each other, all the signal values and so on.