PCI-Compliant Target Device

{Implementation Using Verilog}

Report on CSE311 project

G5

Presented by:	IDs
Ahmed Essam El-Din Ahmed El-Mogy	1300151
Islam Mohammed Abdel-Aziz Abdel-Aal	1700253
Islam Hisham Mohammed Abul-Fadl Awad	1700258
Ahmed Sayed Aabed Ahmed	1700078

Contents

1. Introduction

Hardware description languages (HDLs) like Verilog and VHDL are of extreme importance to the industry of hardware design when it comes to verifying the functionality of the system before the implementation phase. This project focuses on using Verilog HDL to design a PCI-bus-compliant-target (slave) I/O device.

1.1 Overview

The project has in total 5 Verilog code files including: 1 source code file named "Target_IO.v" and 4 testbench files divided based on the functionality and test scenarios implemented on differently configured instances.

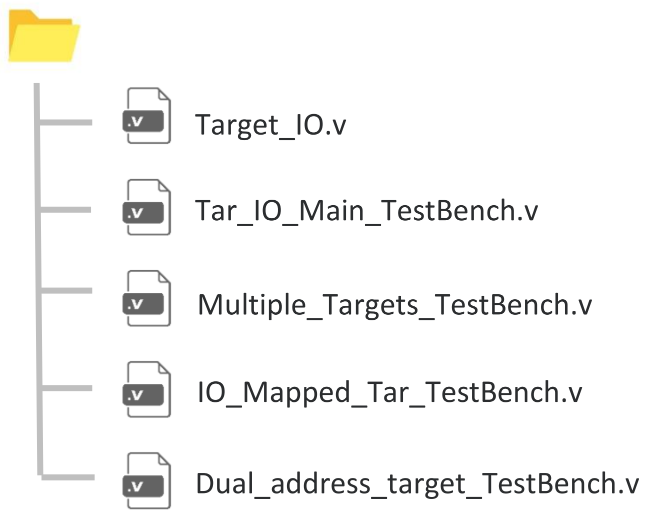

Target_IO.v
Tar_IO_Main_TestBench.v
Multiple_Targets_TestBench.v
IO_Mapped_Tar_TestBench.v
Dual_address_target_TestBench.v

Figure 1 Verilog files in the project

1.2 Block Diagram

This Target module internal blocks are designed to be implemented as shown:

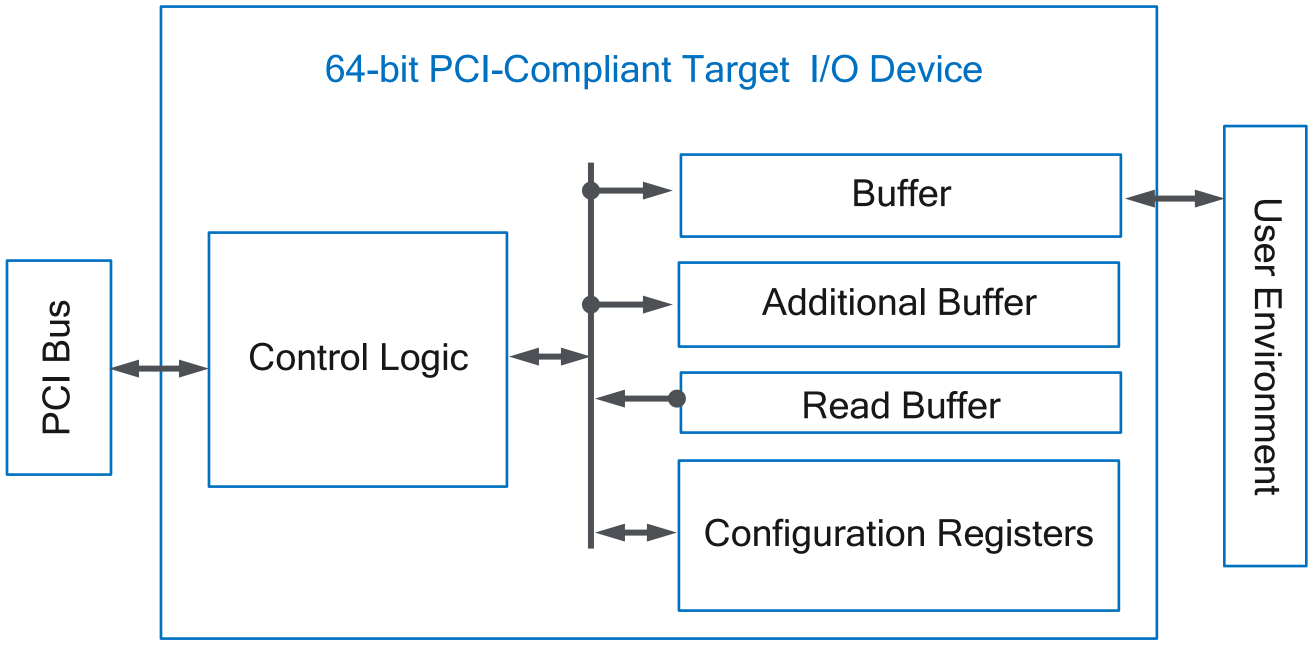


Figure 2 Block diagram of the Target Module

1.3 Addresses

The target device in this project is designed to have 4 DWORDs for the internal buffer and similar for the additional buffer.

All instances of the target module are configured so that for each device: the first address least significant 2 bits are 00, so we can take the lower 2 bits as a pointer to certain register in the buffer.

Based on that, the addresses of the 4 registers share the same highest 30 bits, so the target only need to match this shared 30 bits with the coming address.

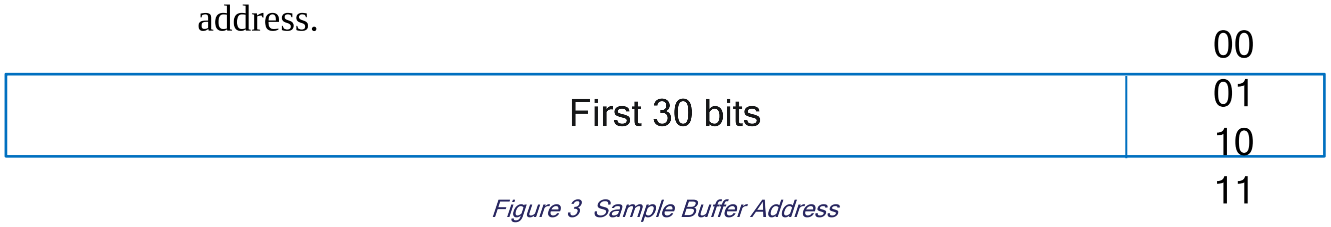


Figure 3 Sample Buffer Address

2. Multiple Instances

A scenario where 3 instances are created, configured, and done on them Memory write and read operations.

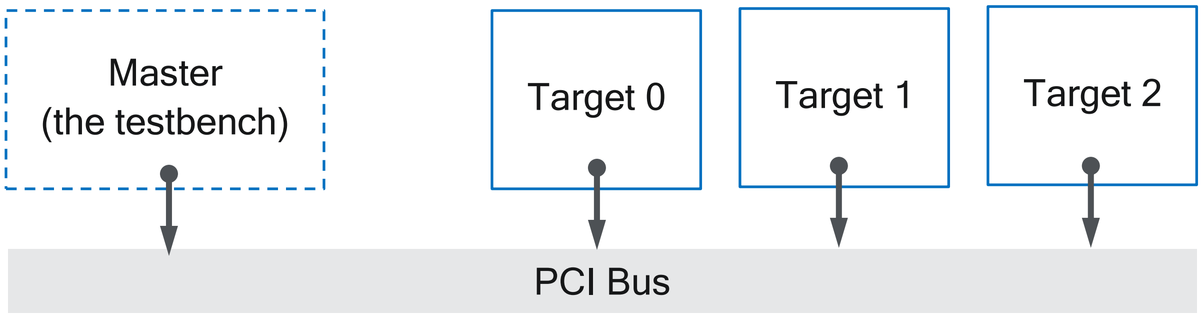


Figure 4 Three Target devices diagram

2.1 Configuration of the 3 instances

First, we send a **Configuration_Read** command to read the cash line sizes for the 3 devices, to know how much space the device needs to store its buffer addresses. And we communicate with each of them through asserting its slot's IDSEL port.

Normally, the configuration space is 256 bytes, we did a simplified version of it with 6 registers divided as shown:

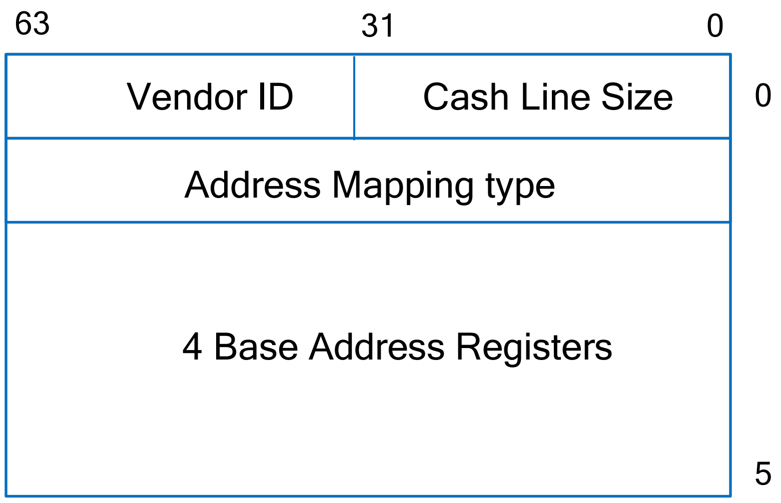


Figure 5 Configuration memory

During the transaction address phase we send the register address in the configuration space with byte-enabling the needed half (in this case second half of the first register). The data to be sent in the address phase on AD port is as shown:

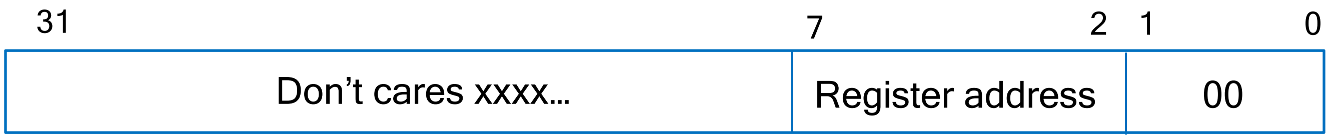


Figure 6 AD address data during Configuration Read

The wave form of the configuration read transaction for the 3 instances is shown below:

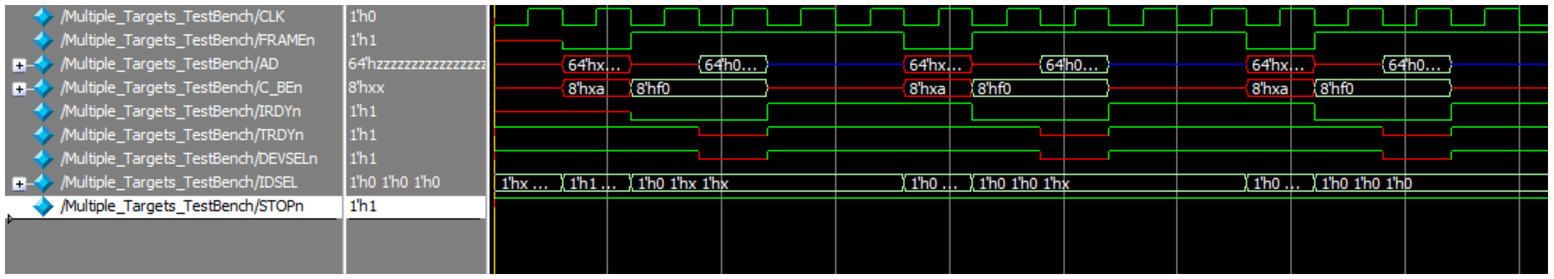

/Multiple_Targets_TestBench/CLK
/Multiple_Targets_TestBench/FRAMEn
/Multiple_Targets_TestBench/AD
/Multiple_Targets_TestBench/C_BEn
/Multiple_Targets_TestBench/IRDYn
/Multiple_Targets_TestBench/TRDYn
/Multiple_Targets_TestBench/DEVSELn
/Multiple_Targets_TestBench/IDSEL
/Multiple_Targets_TestBench/STOPn
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
1'h0 1'h0 1'h0
1'h1
64'hx...
64'h0...
8'hxa
8'hf0
1'hx ...
1'h1 ...
1'h0 1'hx 1'hx
1'h0 ...
1'h0 1'h0 1'hx
1'h0 1'h0 1'h0

Figure 7 Configuration Read

We then allocate addresses on the Memory to the buffers of each instances.

Then, we send **Configuration_Write** command to put these addresses on the Base Address Registers on the Configuration spaces of each instance.

We also send the type of the mapping of addresses whether it’s memory mapped or IO mapped to the address mapping type on the configuration space.

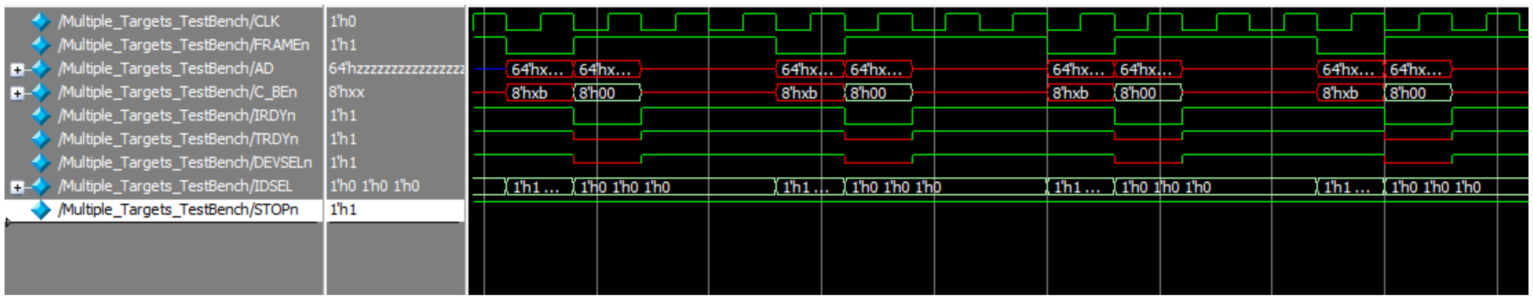

/Multiple_Targets_TestBench/CLK
/Multiple_Targets_TestBench/FRAMEn
/Multiple_Targets_TestBench/AD
/Multiple_Targets_TestBench/C_BEn
/Multiple_Targets_TestBench/IRDYn
/Multiple_Targets_TestBench/TRDYn
/Multiple_Targets_TestBench/DEVSELn
/Multiple_Targets_TestBench/IDSEL
/Multiple_Targets_TestBench/STOPn
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
1'h0 1'h0 1'h0
1'h1
64'hx...
8'hxb
8'h00
1'h1 ...
1'h0 1'h0 1'h0

Figure 8 Configuration Write

2.2 Memory write / read on the 3 devices

After we've done with configuration, we did simple Write then Read operations to test what we've done and the address calling…

Writing 4 DWORDs on the 3 instances:

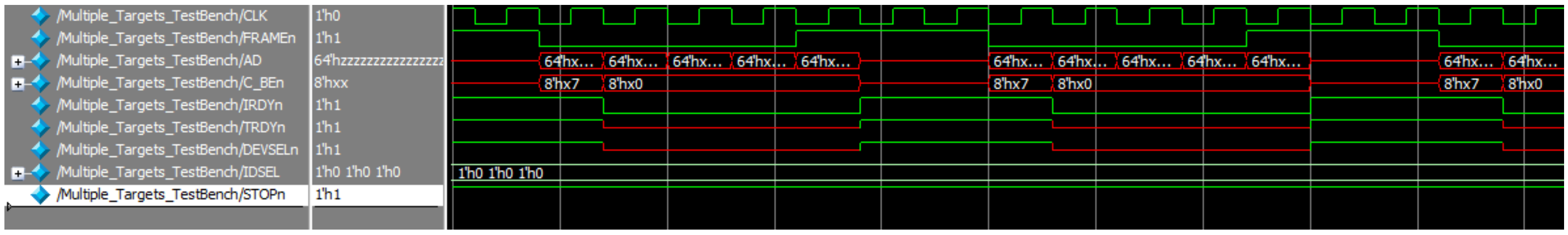

/Multiple_Targets_TestBench/CLK
/Multiple_Targets_TestBench/FRAMEn
/Multiple_Targets_TestBench/AD
/Multiple_Targets_TestBench/C_BEn
/Multiple_Targets_TestBench/IRDYn
/Multiple_Targets_TestBench/TRDYn
/Multiple_Targets_TestBench/DEVSELn
/Multiple_Targets_TestBench/IDSEL
/Multiple_Targets_TestBench/STOPn
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
1'h0 1'h0 1'h0
1'h1
64'hx...
8'hx7
8'hx0
1'h0 1'h0 1'h0

Figure 9 Memory Write

And then reading them:

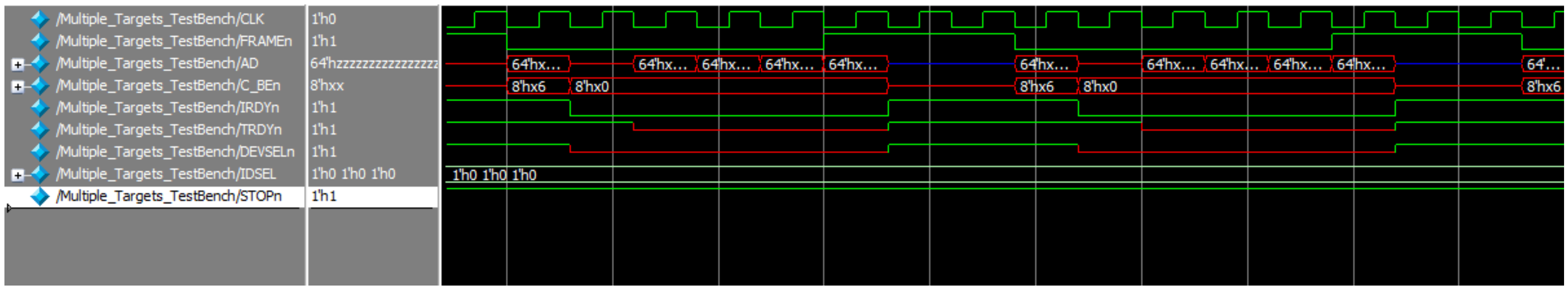

/Multiple_Targets_TestBench/CLK
/Multiple_Targets_TestBench/FRAMEn
/Multiple_Targets_TestBench/AD
/Multiple_Targets_TestBench/C_BEn
/Multiple_Targets_TestBench/IRDYn
/Multiple_Targets_TestBench/TRDYn
/Multiple_Targets_TestBench/DEVSELn
/Multiple_Targets_TestBench/IDSEL
/Multiple_Targets_TestBench/STOPn
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
1'h0 1'h0 1'h0
1'h1
64'hx...
8'hx6
8'hx0
1'h0 1'h0 1'h0

Figure 10 Memory Read

3. Simple Byte Enable Test

A scenario where we write four words on the target and then read the second two and with the Byte Enable referring to the lower 16 bits:

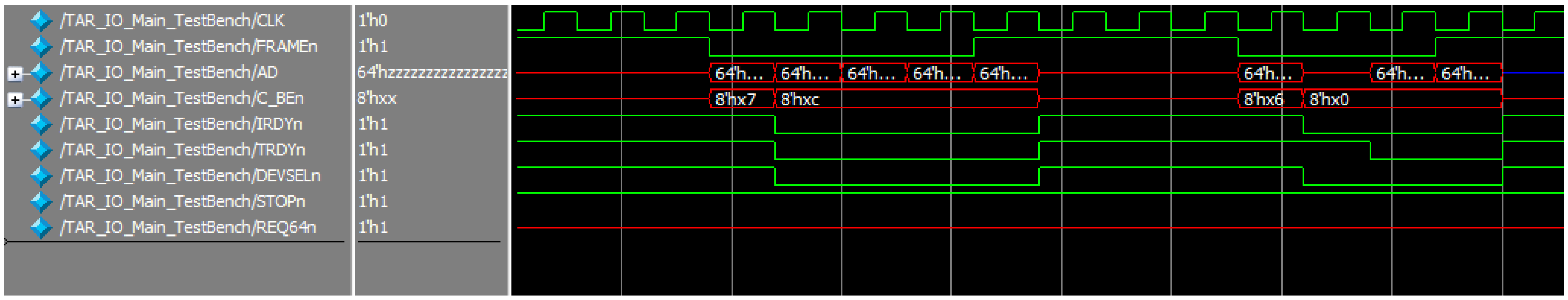

/TAR_IO_Main_TestBench/CLK
/TAR_IO_Main_TestBench/FRAMEn
/TAR_IO_Main_TestBench/AD
/TAR_IO_Main_TestBench/C_BEn
/TAR_IO_Main_TestBench/IRDYn
/TAR_IO_Main_TestBench/TRDYn
/TAR_IO_Main_TestBench/DEVSELn
/TAR_IO_Main_TestBench/STOPn
/TAR_IO_Main_TestBench/REQ64n
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
1'h1
1'h1
64'h...
8'hx7
8'hxc
8'hx6
8'hx0

Figure 11 Memory Write then Read transaction with Byte Enabling

4. Enabling 64-bit Transactions

A scenario where we write four 64-bits words on a target and then read them,: testing the functionality of enabling 64-bit transactions through asserting REQ64#.

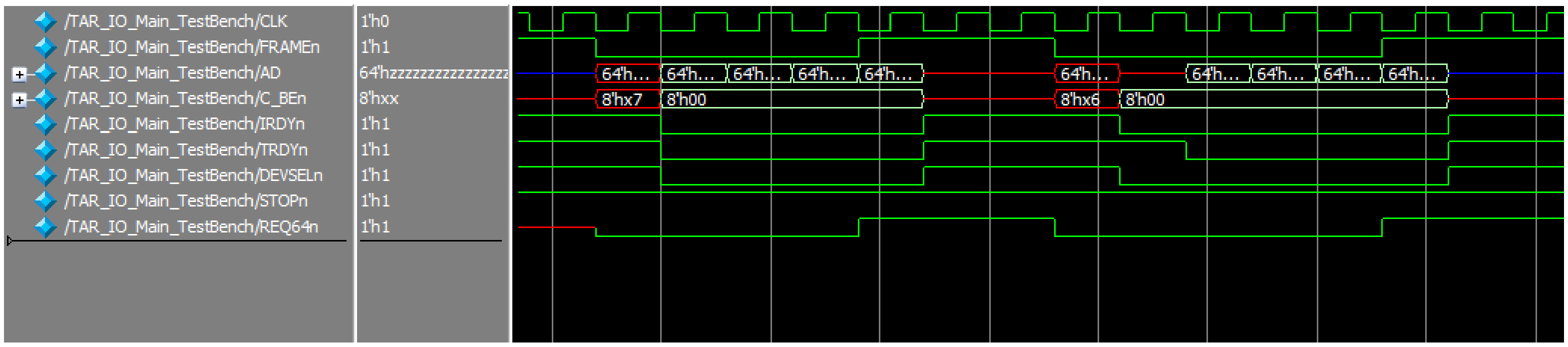
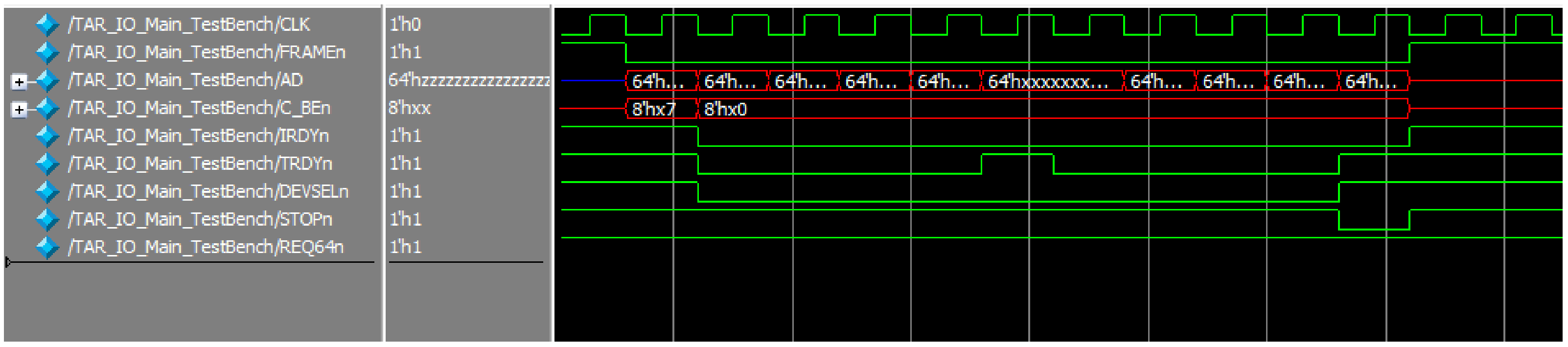

/TAR_IO_Main_TestBench/CLK
/TAR_IO_Main_TestBench/FRAMEn
/TAR_IO_Main_TestBench/AD
/TAR_IO_Main_TestBench/C_BEn
/TAR_IO_Main_TestBench/IRDYn
/TAR_IO_Main_TestBench/TRDYn
/TAR_IO_Main_TestBench/DEVSELn
/TAR_IO_Main_TestBench/STOPn
/TAR_IO_Main_TestBench/REQ64n
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
1'h1
1'h1
64'h...
8'hx7
8'h00
8'hx6
8'h00

Figure 12 64-bit Memory Write and Read

5. Write and read data bigger than cash line size

A scenario where we attempt to write 12 words on a target and then read what it's supposed to have written which is 8 (4 on the Buffer, and 4 on the Additional Buffer),: testing the functionality of the response of asserting TRDY# and STOP# ports.

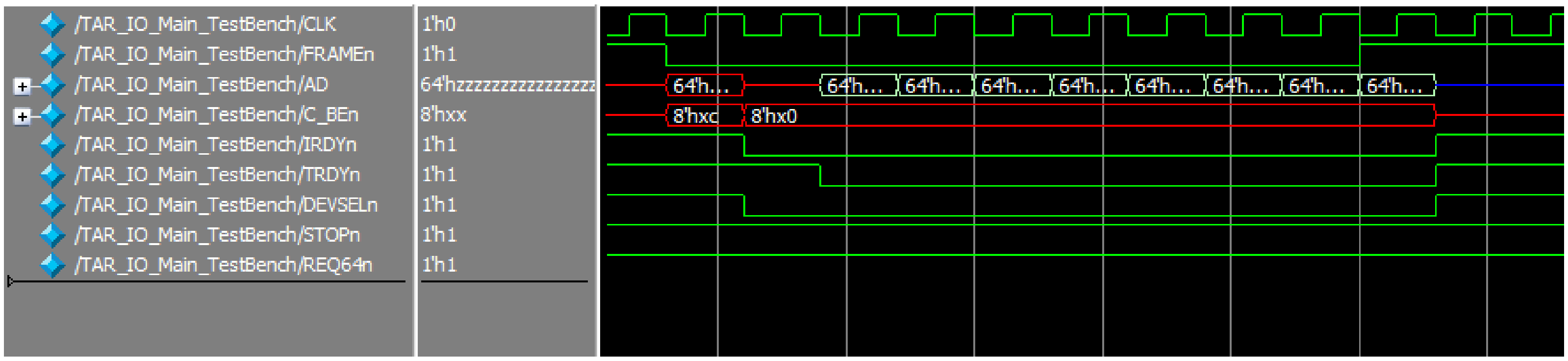
First: Attempting to write 12 words: TRDY# is asserted high upon receiving the fourth word to fetch data on the additional buffer, and then after words STOP# is asserted low to force the master to stop the transaction which it does after one cycle.



/TAR_IO_Main_TestBench/CLK 1'h0
/TAR_IO_Main_TestBench/FRAMEn 1'h1
/TAR_IO_Main_TestBench/AD 64'hzzzzzzzzzzzzzzzz
/TAR_IO_Main_TestBench/C_BEn 8'hxx
/TAR_IO_Main_TestBench/IRDYn 1'h1
/TAR_IO_Main_TestBench/TRDYn 1'h1
/TAR_IO_Main_TestBench/DEVSELn 1'h1
/TAR_IO_Main_TestBench/STOPn 1'h1
/TAR_IO_Main_TestBench/REQ64n 1'h1
64'h... 64'h... 64'h... 64'h... 64'h... 64'hxxxxxxxx... 64'h... 64'h... 64'h... 64'h...
8'hx7 8'hx0

Figure 13 Memory Writing 8 words

Then, Reading 8 words to receive the written data:



/TAR_IO_Main_TestBench/CLK 1'h0
/TAR_IO_Main_TestBench/FRAMEn 1'h1
/TAR_IO_Main_TestBench/AD 64'hzzzzzzzzzzzzzzzz
/TAR_IO_Main_TestBench/C_BEn 8'hxx
/TAR_IO_Main_TestBench/IRDYn 1'h1
/TAR_IO_Main_TestBench/TRDYn 1'h1
/TAR_IO_Main_TestBench/DEVSELn 1'h1
/TAR_IO_Main_TestBench/STOPn 1'h1
/TAR_IO_Main_TestBench/REQ64n 1'h1
64'h... 64'h... 64'h... 64'h... 64'h... 64'h... 64'h... 64'h... 64'h...
8'hxc 8'hx0

Figure 14 Reading 8 words

6. I/O Read and I/O Write

A scenario where we make 2 instances of the target, one we tell it is memory mapped through configuration, with its addresses been put on a memory called "Memory Address Space" , and the other instance we tell it is I/O mapped through configuration with its addresses been put on a memory called "IO Address Space". But the addresses are the same for two of them.

Then we do Memory Write and Memory Read on the first one. And then do IO Write and IO Read on the second one through the same address: Testing the ability of the device to start implementing only commands that target it based on its address space type.

First, Memory Writing and Reading on the Memory mapped same address target:

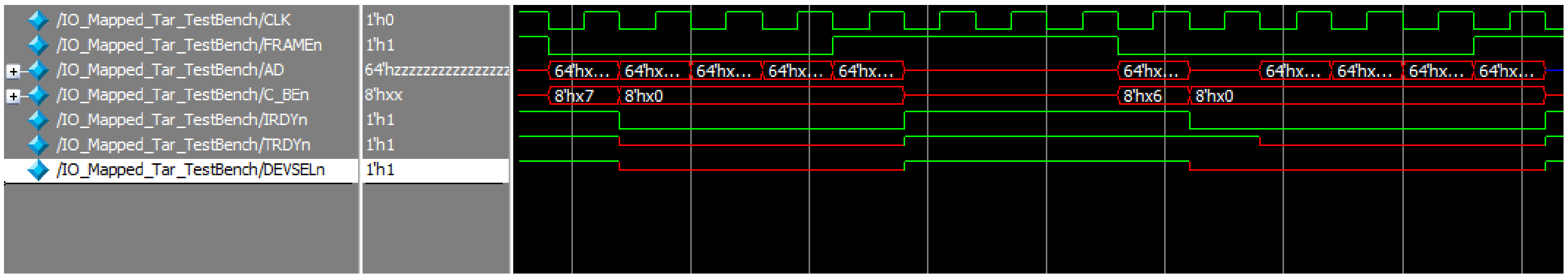

/IO_Mapped_Tar_TestBench/CLK
/IO_Mapped_Tar_TestBench/FRAMEn
/IO_Mapped_Tar_TestBench/AD
/IO_Mapped_Tar_TestBench/C_BEn
/IO_Mapped_Tar_TestBench/IRDYn
/IO_Mapped_Tar_TestBench/TRDYn
/IO_Mapped_Tar_TestBench/DEVSELn
1'h0
1'h1
64'hzzzzzzzzzzzzzzzz
8'hxx
1'h1
1'h1
1'h1
64'hx...
8'hx7
8'hx0
8'hx6
8'hx0

Figure 15 Memory Writing and Reading

Second, I/O Writing and Reading on the I/O mapped same address target:

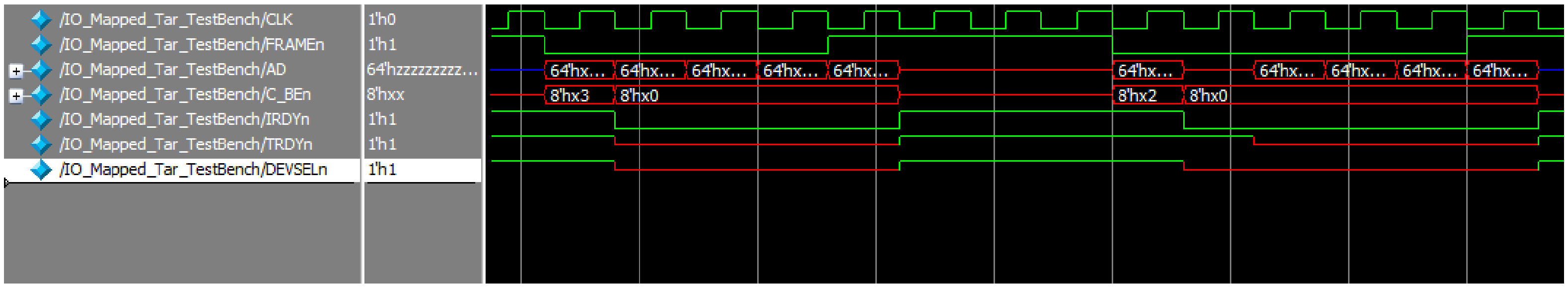

/IO_Mapped_Tar_TestBench/CLK
/IO_Mapped_Tar_TestBench/FRAMEn
/IO_Mapped_Tar_TestBench/AD
/IO_Mapped_Tar_TestBench/C_BEn
/IO_Mapped_Tar_TestBench/IRDYn
/IO_Mapped_Tar_TestBench/TRDYn
/IO_Mapped_Tar_TestBench/DEVSELn
1'h0
1'h1
64'hzzzzzzzzz...
8'hxx
1'h1
1'h1
1'h1
64'hx...
8'hx3
8'hx0
8'hx2
8'hx0

Figure 16 IO Read and Write

7. Dual Address Target

A scenario where we make an instance and configure it’s addresses to be stored on 64-bits. And then Writing 4 words on it: Testing the command Dual Address which signals to the target to hold to receive second half then receive the command.

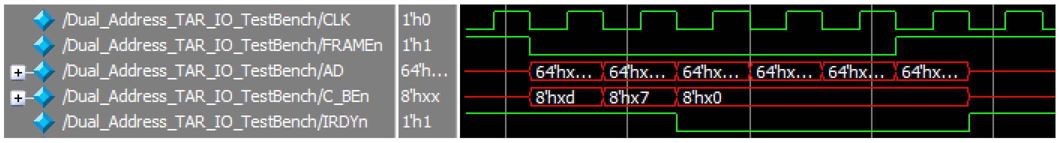

/Dual_Address_TAR_IO_TestBench/CLK
1'h0
/Dual_Address_TAR_IO_TestBench/FRAMEn
1'h1
/Dual_Address_TAR_IO_TestBench/AD
64'h...
/Dual_Address_TAR_IO_TestBench/C_BEn
8'hxx
/Dual_Address_TAR_IO_TestBench/IRDYn
1'h1
64'hx...
64'hx...
64'hx...
64'hx...
64'hx...
64'hx...
8'hxd
8'hx7
8'hx0

Figure 17 Dual Address Write

8. Signal Description

Bar Type	Memory mapped (0) or IO mapped (1)
Targeted	Address is matched
Transaction	Frame is low
Transaction End	Last word to be written
Dual Cycle Flag	Hold for the other half of 64-bit addresses
Enable 64	Enabling 64-bit Transactions
PCI slot select	Targeting the device while configuration
Config End	Last word in configuration to be written

9. Appendix: Who did what

Ahmed Essam El-Din Ahmed El-Mogy

- Memory Write, IO write, IO Read, Config. Read/Write, Mem Read Multiple.
- Test benches for the mentioned commands.

Islam Mohammed Abdel-Aziz Abdel-Aal

- Dual Address, 64-bit enabling.
- Test benches for the mentioned command and 64-bit enabling.

Ahmed Sayed Aabed Ahmed

- Memory Read.
- Write Multiple (TRDY# and STOP# timing).

Islam Hisham Mohammed Abul-Fadl Awad

- Testbench Memory Read.
- Testbenches for asserting TRDY in different commands.

10. Additional Information

- We designed the target module as a FSM (Finite State Machine) with type Meely as its output depends on the state and the incoming signals.

- All the transactions made in this report are made on 32 bits ignoring the high 32 bits except for the case with the 64-bit enabling.