# LAB04
# AXI Traffic Generation (ATG)



POWER & ENERGY CONVERSION LABORATORY

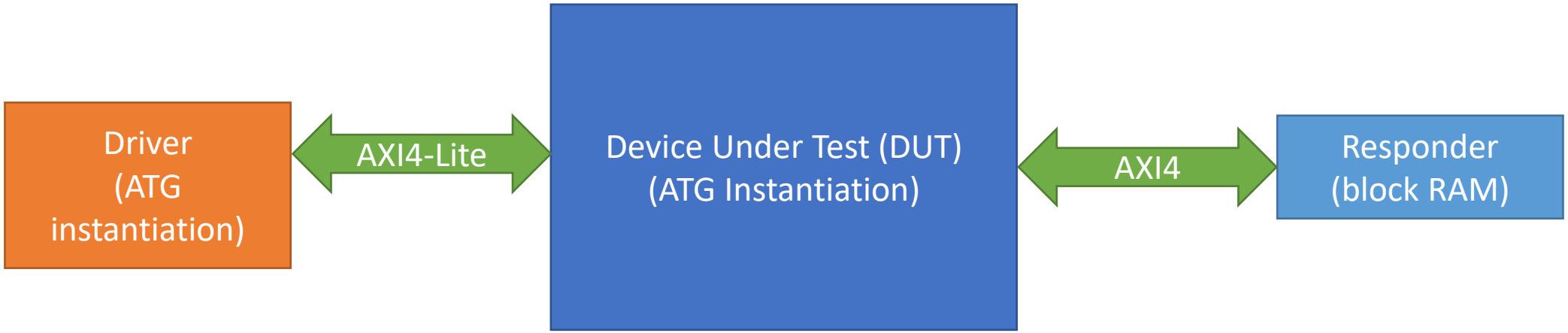
Electrical Engineering Department
Pontificia Universidad Católica de Chile
peclab.ing.uc.cl

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

# AXI Traffic Generator

## What is a ATG?

- In this laboratory we will use two IP-cores ATG.
- First ATG is named driver and it is a custom profile Test Mode ATG.
- Second ATG is named DUT and it is a custom profile Advanced Mode ATG




**Driver**: An instantiation of the ATG in AXI4-Lite mode. This module is used to generate AXI4 Lite transactions to program the DUT module.

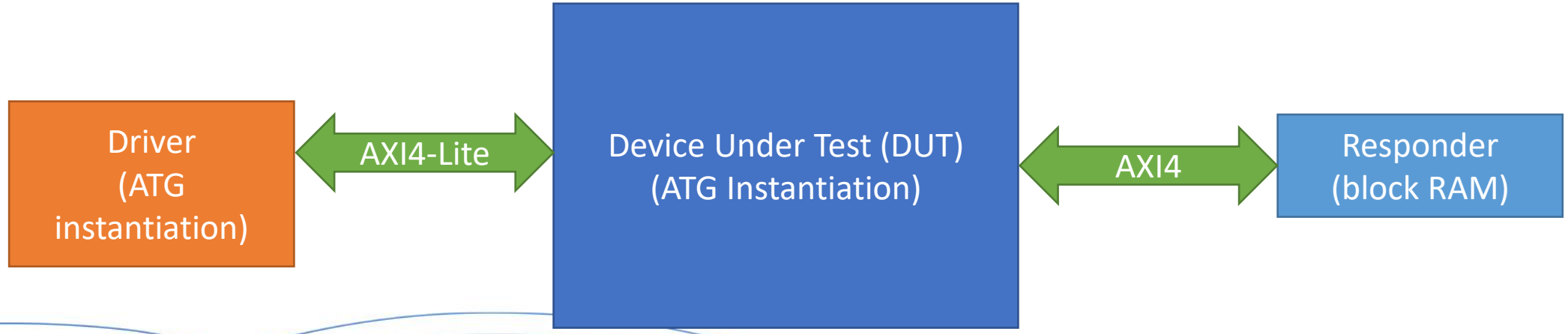
**DUT**: Device under test. A second instantiation of an ATG, this time in AXI4 mode. This module is used to generate AXI4 transaction to the Responder.

**Responder**: An instantiation of a block RAM controller that will accept the generated traffic from the ATG. This differs from the block RAM transaction buffers that are internal to the two ATGs.

# AXI Traffic Generator

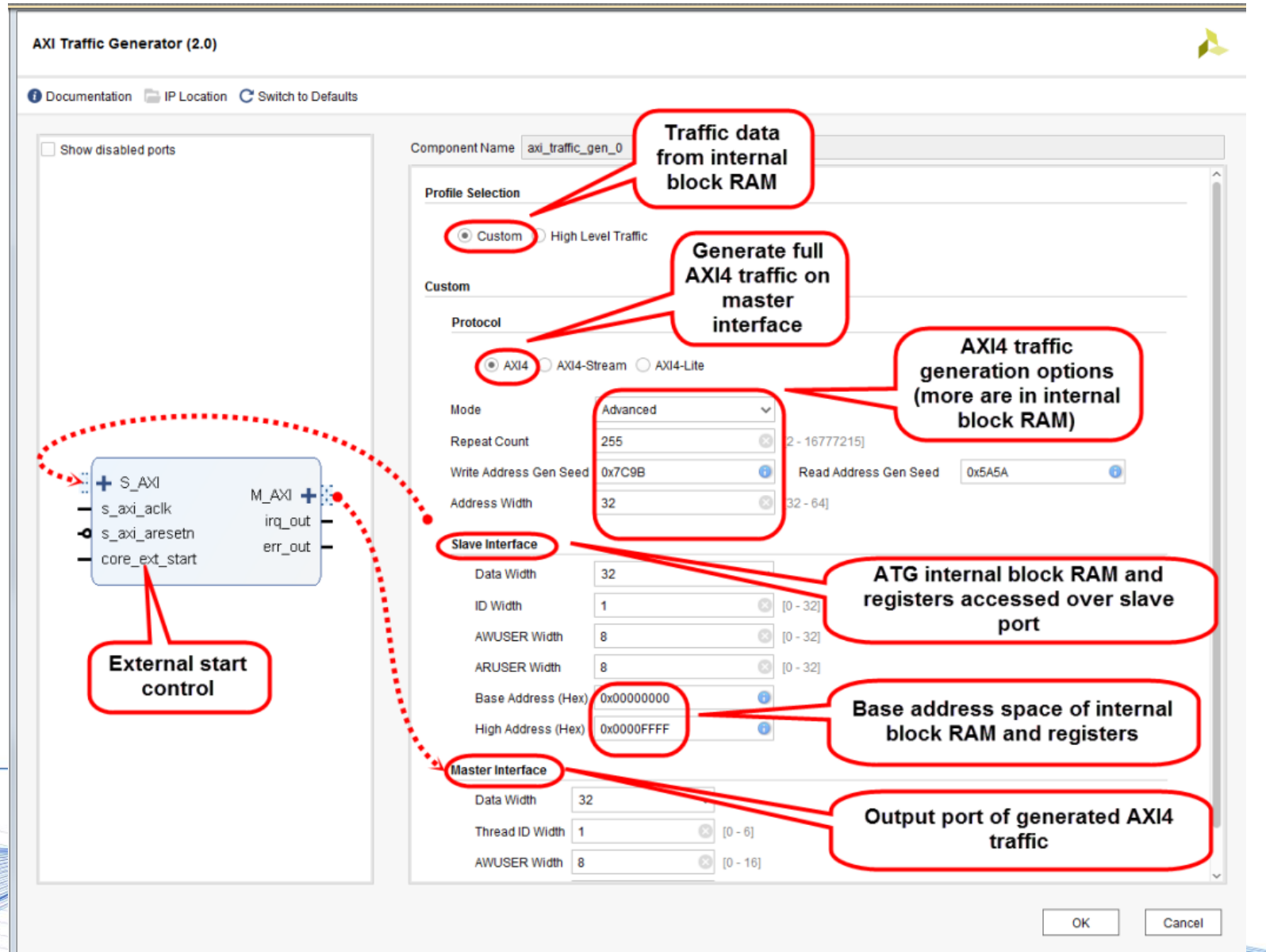
## What is a ATG?

1. The driver will read the Master Register from DUT. (just for fun!)
2. The driver will fill the MSTRAM with data. (from COE FILES). This data Will be later used by the DUT.
3. The driver will fill the CMDRAM to set read and write the transactions that DUT shall implement with the block RAM. (from COE FILES)
4. The driver set the Master Register of DUT to start DUT operation.
5. The DUT perform the transactions sabes into its CMDRAM.
6. Operation is finished.



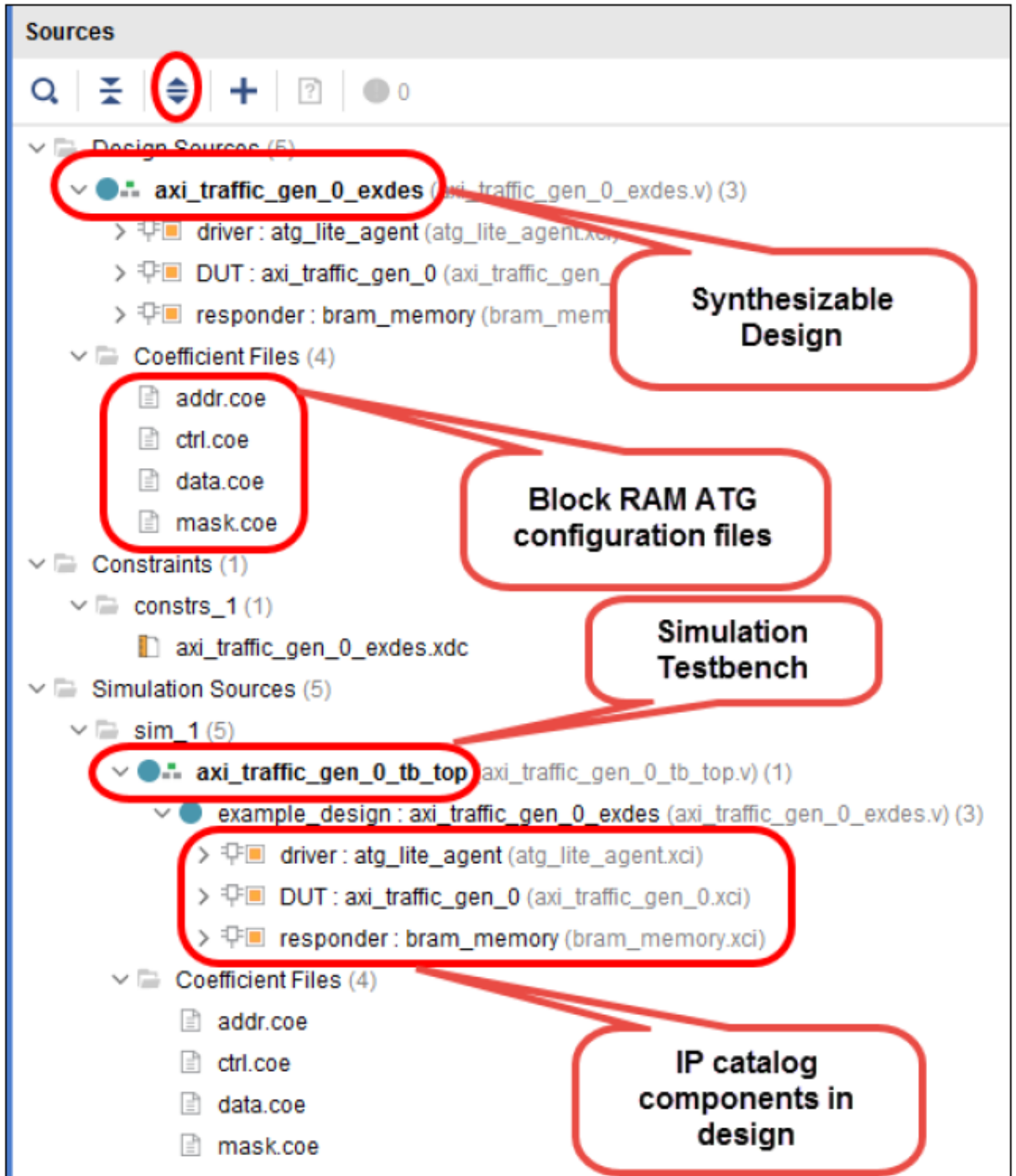
# AXI Traffic Generator

## ATG Customization



# AXI Traffic Generator

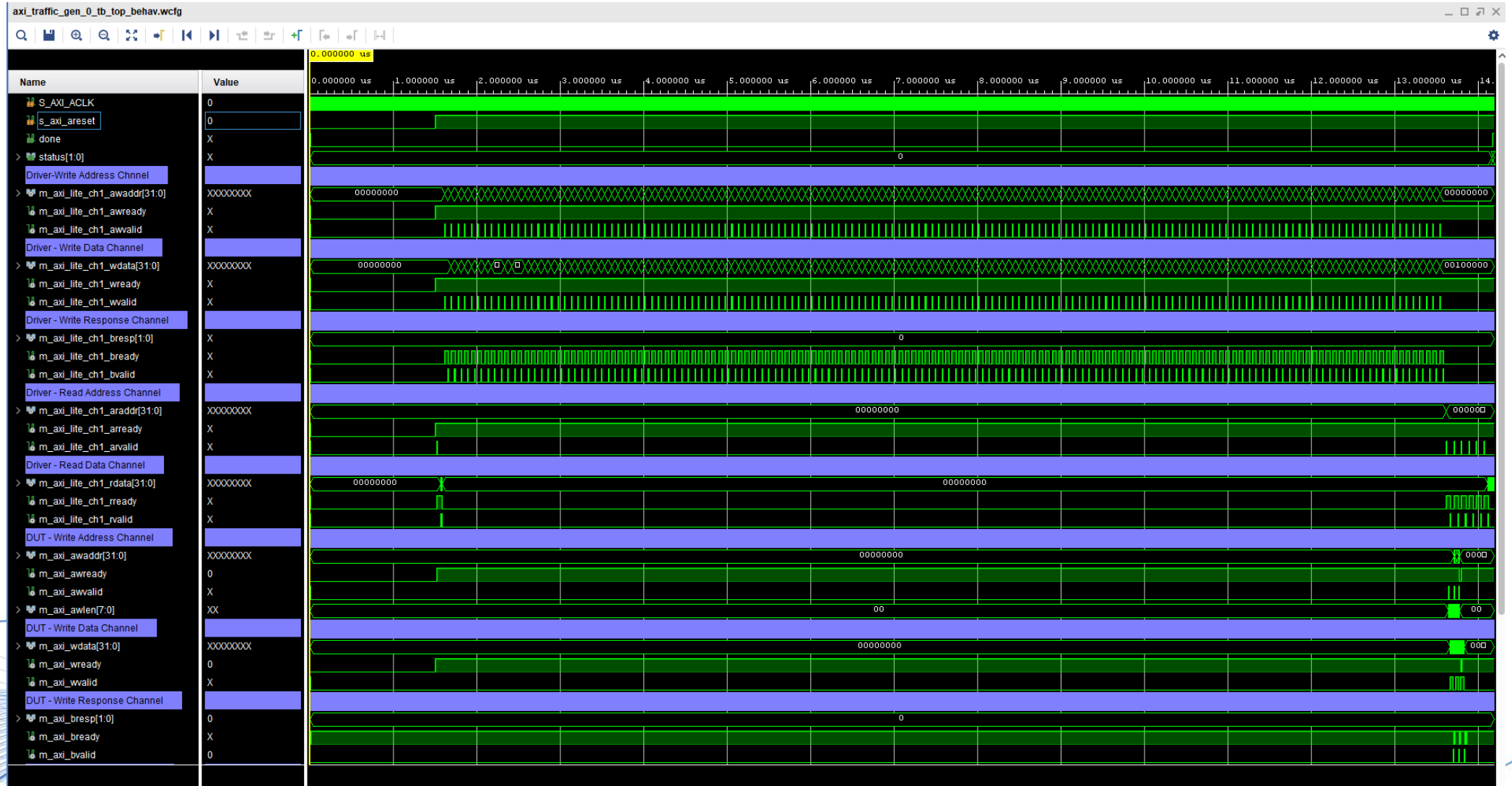
## Example: Driver-DUT-RAM



# AXI Traffic Generator

## Simulation

After Simulating for 15us you should see something like this:

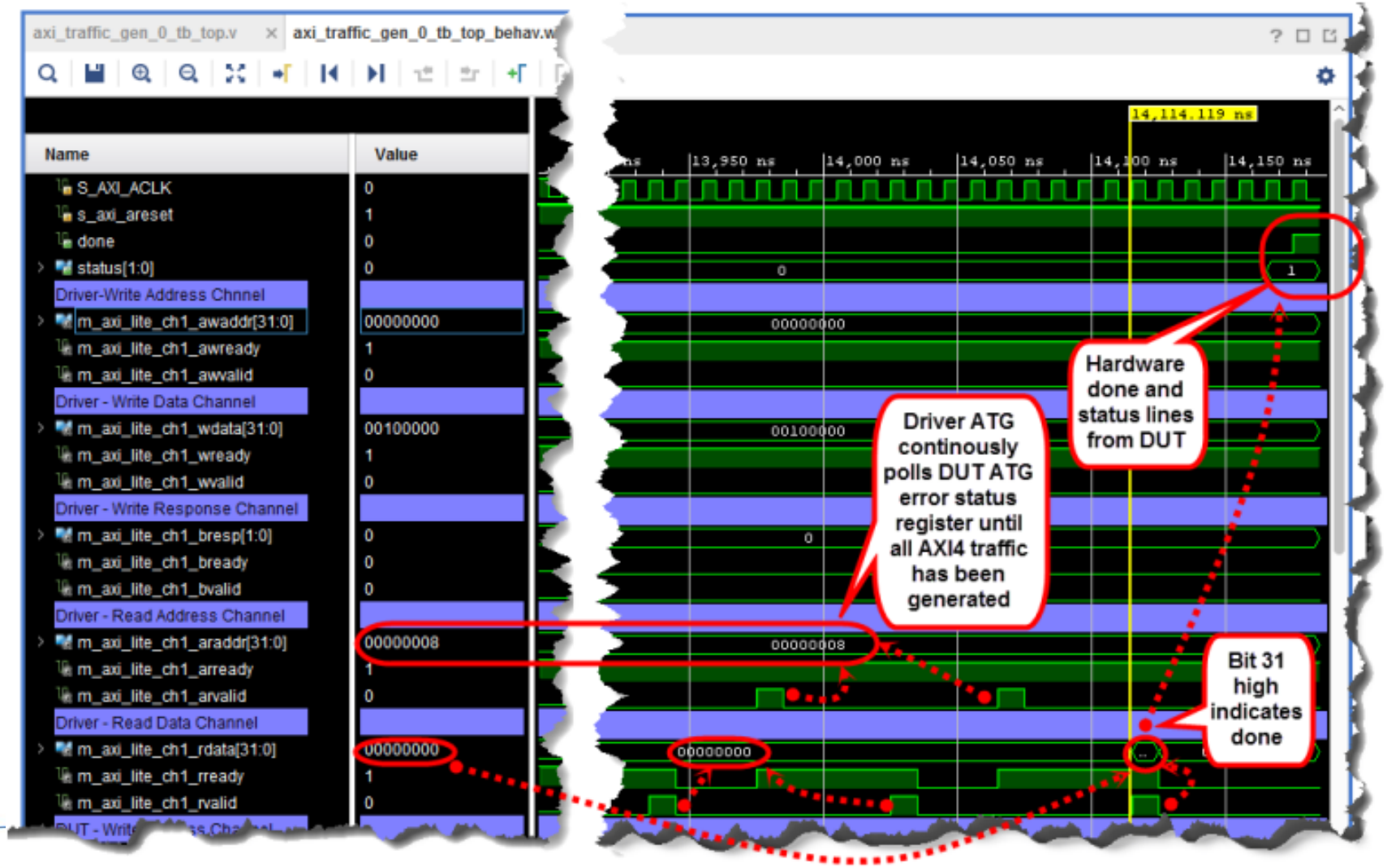


# AXI Traffic Generator

## Do the following (Driver):

1. How many AXI ports can you identify? Which are their channels?
2. Check all names of the signals and match them with theory. Which two signals are present in each channel? (handshaking signals)
3. Zoom in around time 1.5us to 1.7us:
   1. Which is the first transaction of the driver? How do you identify it?
   2. Which is the second transaction of the driver? How do you identify it?

   Note: To start DUT transaction the bit 20 of the Master Control Register (0x00000000) must be set to 1. (ATG guideline). Find where this happends.
4. Go to time 13us to 14us and identify the end of writting transaction frm driver to LUT.


axi_traffic_gen_0_tb_top.v
axi_traffic_gen_0_tb_top_behav.w
Name
Value
S_AXI_ACLK
s_axi_areset
done
status[1:0]
Driver-Write Address Chnnel
m_axi_lite_ch1_awaddr[31:0]
m_axi_lite_ch1_awready
m_axi_lite_ch1_awvalid
Driver - Write Data Channel
m_axi_lite_ch1_wdata[31:0]
m_axi_lite_ch1_wready
m_axi_lite_ch1_wvalid
Driver - Write Response Channel
m_axi_lite_ch1_bresp[1:0]
m_axi_lite_ch1_bready
m_axi_lite_ch1_bvalid
Driver - Read Address Channel
m_axi_lite_ch1_araddr[31:0]
m_axi_lite_ch1_arready
m_axi_lite_ch1_arvalid
Driver - Read Data Channel
m_axi_lite_ch1_rdata[31:0]
m_axi_lite_ch1_rready
m_axi_lite_ch1_rvalid
00000000
00100000
00000008
14,114.119 ns
13,950 ns
14,000 ns
14,050 ns
14,150 ns
Hardware done and status lines from DUT
Driver ATG continously polls DUT ATG error status register until all AXI4 traffic has been generated
Bit 31 high indicates done

# AXI Traffic Generator

## Do the following (DUT):

1. Go to time 13.65us identify how many transactions make the LUT.
2. Which difference can you identify compared to AXI Lite transactions.

Note: Awlen[7:0]: defines the number of bursts

Using awreadt, wvalid, rready and rvalid determine the number of beats per transaction.

3. Complete the following table for all DUT transactions

| Transaction | Type | Address | Number Data Beats | Burst Length awlen or arlen |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |

# AXI Traffic Generator

## Do the following (DUT):

1. Go to time 13.65us identify how many transactions make the LUT.
2. Which difference can you identify compared to AXI Lite transactions.

Note: Awlen[7:0]: defines the number of bursts

Using awreadt, wvalid, rready and rvalid determine the number of beats per transaction.

3. Complete the following table for all DUT transactions

| Transaction | Type | Address | Number Data Beats | Burst Length awlen or arlen |
|---|---|---|---|---|
| 1 | write | 0x00000000 | 3 | 2 |
| 2 | write | 0x00000040 | 4 | 3 |
| 3 | write | 0x00000080 | 4 | 3 |
| 4 | read | 0x00000000 | 3 | 2 |
| 5 | read | 0x00000040 | 4 | 3 |

# AXI Traffic Generator

## Address COE FILE:

addr.coe × axi_traffic_gen_0_tb_top_behav.wcfg ×

c:/training/AXItransactions/support/addr.coe

```
1  memory_initialization_radix = 16;
2  memory_initialization_vector =
3  00000000
4  00008000
5  00008004
6  00008008
7  0000800C
8  00008010
9  00008014
10 00008018
11 0000801C
12 00009000
13 00009004
14 00009008
15 0000900C
16 00009010
17 00009014
18 00009018
19 0000901C
20 00009020
21 00009024
22 00009028
23 0000902C
24 00009000
```

First two blocks at 0x00008000 represent two reads

Each block of four addresses represents a DUT 128-bit (four Dwords) configuration to initiate an AXI4 transaction

Address of static AXI values transaction ordering & burst length static AXI values

Four Dwords(128 bits) represent AXI4 traffic entry

Three blocks beginning at 0x00009000

1. The address COE file contains a list of addresses that the Driver reads and writes.
2. Read and write operations are coded into crt.coe
3. The data written or read (predicted) is contained in data.coe
4. The DUT is configured to make 5 transactions.
5. Each DUT transaction require 128bits total (all AXI data/config). It is divided into 4 data of 32bits, name dword.
6. Within DUT ATG there are two identical structures, one for reading (starting at 0x00008000). And for writing, beginning at address 0x00009000.

# AXI Traffic Generator

## Data COE File:

axi_traffic_gen_0_tb_top_behav.wcfg | data.coe

c:/training/AXItransactions/support/data.coe

```
memory_initialization_radix=16;
memory_initialization_vector=
20000000
00000000
80002402
00006400
00000000
00000040
80002403
00000010
00000000
00000000
80002402
00000000
00000000
00000040
80002403
00000010
00000000
00000080
80002403
00000020
00000000
00000111
```

Configuration write data for two read

Address static AXI values transaction ordering & burst length static AXI values

Burst m_axi_awlen[7:0] is low order 8-bits of third configuration

Write data for three writes

1. Each entry corresponds to the data associated to the same line on the address coe file.
2. When we write to adress 0x8000 and 0x9000 we must write 128bit data. That is why we habe blocks of four data (12bits each).
3. First two blocks is the data that configure the two reading transactions.
4. Blocks 3,4,and 5 configure three reading transactions.

# Homework

1. Modify the COE files to change the address read and written by the DUT.

2. Modify the COE files to change the lenght of the read and written burst.

3. Modify the COE files to change the data written to the RAM by the DUT.

Remember that changing the COE files you only change the data that the driver is sending to the DUT. However, this data is strategically sent to configure the transactions of the DUT.

Now you can add AXI drivers (with predefined data) to connect them with your own blocks that posses AXI-lite communication protocol.

You can also configure an AXI4-full interface to connect i to any of your own blocks designed as AXI4-full slaves.

# AXI Traffic Generator
## Overview


POWER & ENERGY CONVERSION LABORATORY

Electrical Engineering Department
Pontificia Universidad Católica de Chile
peclab.ing.uc.cl