Introduction to AXI – AXI4-Lite Transactions Demo Script

Introduction

This demonstration will introduce you to AXI4-Lite transactions through the familiarization and analysis of simulated waveform output. This demonstration highlights the use of the AXI Traffic Generation (ATG) IP core accessing an AXI block RAM controller IP core.

Preparation:

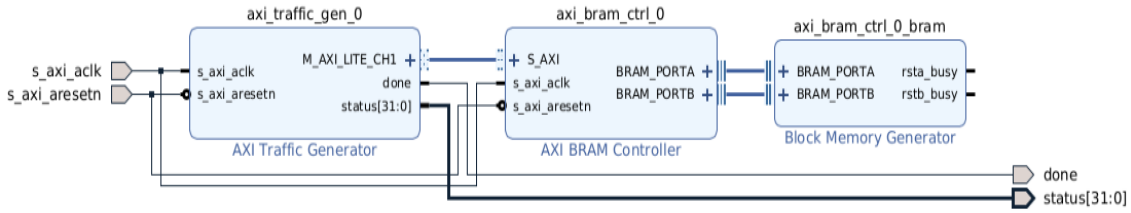
- Required files (all found in the `support` directory for this topic cluster):
  - `AXI_transactions_demo_completer.tcl`
  - Four coefficient files for controlling the message generation
  - `axi_traffic_gen_0_tb_top_behav.wcfg` waveform file
  - `axi_traffic_gen_0_tb_top.v`
- Required software: Vivado™ Design Suite (any edition)
- Required hardware: none
- Basic background of the ATG core (not required for demo):
  - *AXI Traffic Generator LogiCORE IP Product Guide* (PG125) - available through DocNav

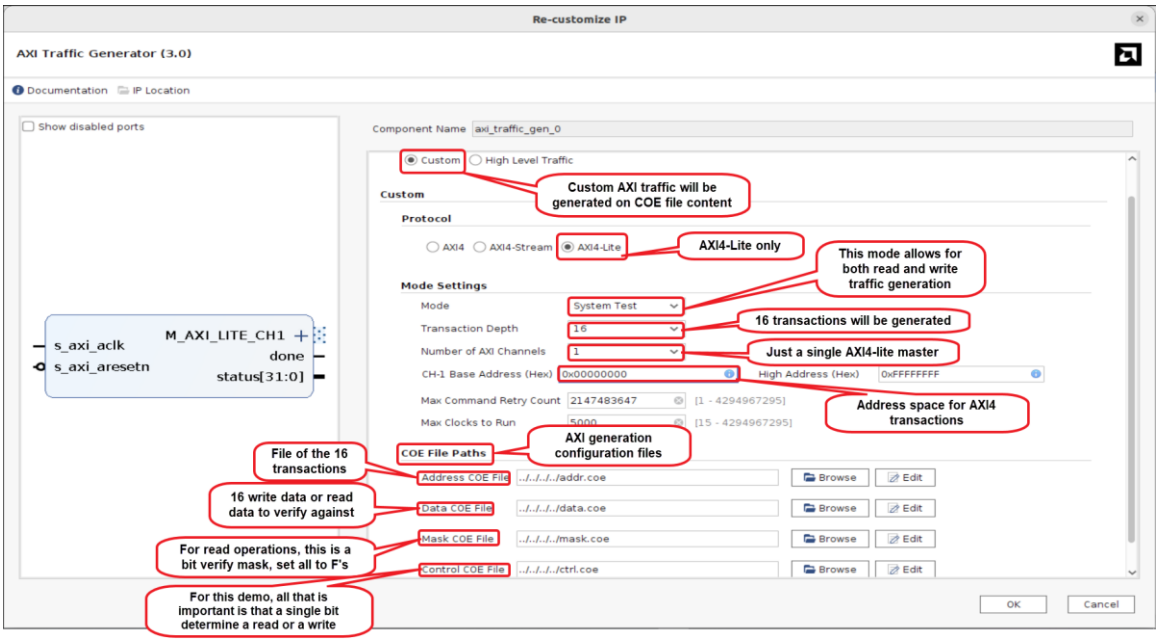
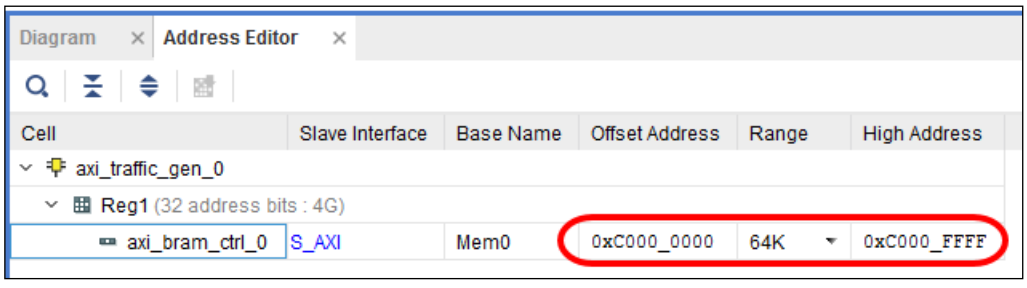
AXI4-Lite Transactions

Action with Description	Point of Emphasis and Key Takeaway
• Launch the Vivado Design Suite. • Open the `AXItransactions.xpr` project from the following directory: `$TRAINING_PATH/AXItransactions/demo` • The project will build and leave the IPI block design open.	Use the provided Tcl script to quickly build the project and include all of the files. You can easily open an existing Vivado IDE project via the Getting Started page.

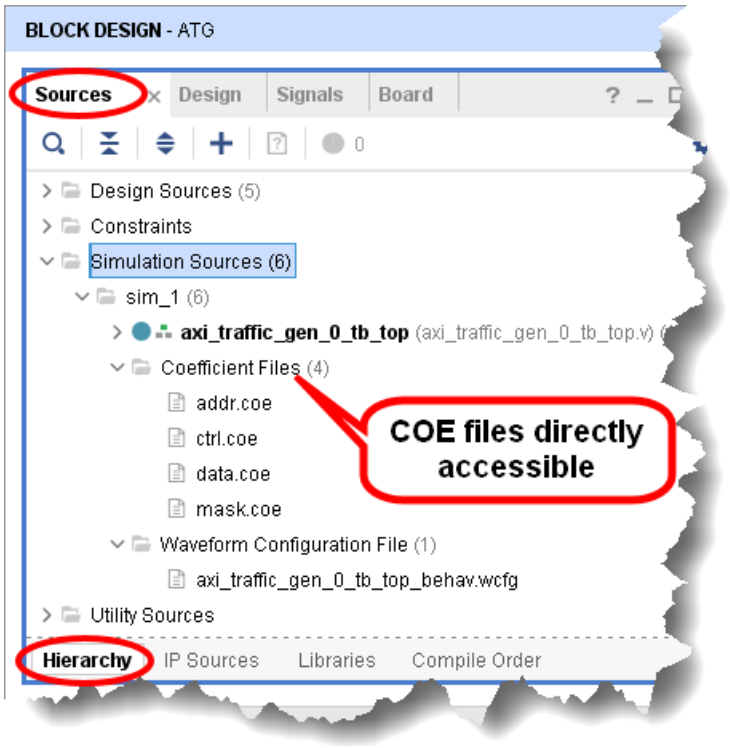
Action with Description	Point of Emphasis and Key Takeaway

• Maximize the IPI block design view, and quickly summarize the project.
• The design consists of three components:
  • AXI Traffic Generator (ATG) IP core
  • AXI BRAM Controller IP core
  • Block Memory Generator
• The purpose of this demo will be to examine and study the AXI transactions generated by the ATG.
	The Vivado IP integrator is used to create subsystems, including embedded subsystems. These IPI subsystems are referred to as block designs. The components of the design are constructed with IP cores from the IP catalog.

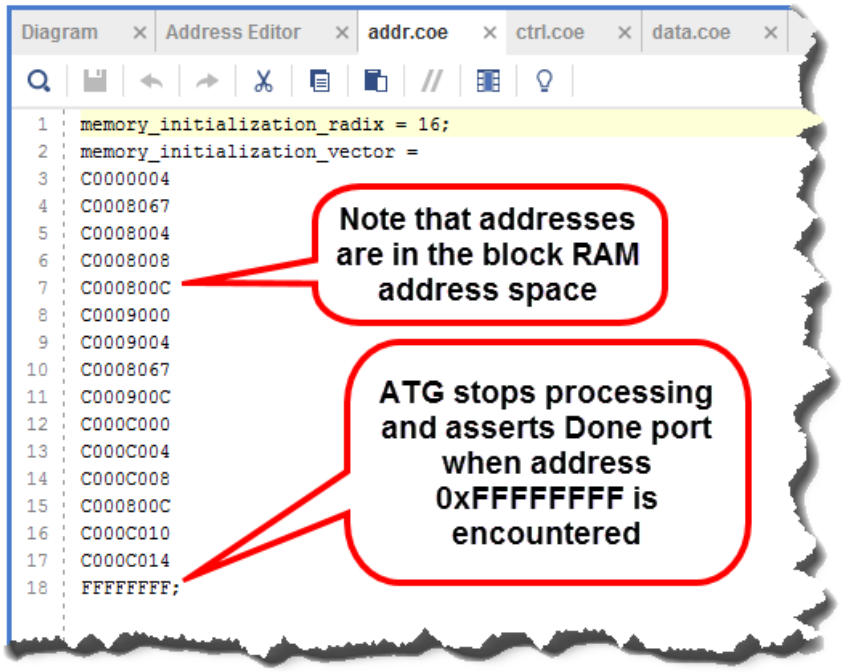
Action with Description	Point of Emphasis and Key Takeaway
• Double-click the **ATG** core to open its properties. • Review the properties as shown below. • If you feel familiar with the core, feel free to explore with the class, but do not change anything. • Cancel out of the properties to avoid any inadvertent changes.	Introduce the student to the ATG core. The ATG in AXI4-Lite mode generates AXI4-Lite transactions based on the contents of four block RAMs that are internal to the ATG. These block RAMs are initialized via COE files that are used in simulation and also injected into the bitstream during implementation. The contents of these four COE files govern the operation of the ATG. Refer to the *AXI Traffic Generator LogiCORE IP Product Guide* (PG125) for further information.
	
• Select the **Address Editor** tab. Note the auto-generated address of 0xC0000000 – 0xC000FFFF of the block RAM.	This is important to know so that the AXI transactions can be addressed in this range.
	

Action with Description	Point of Emphasis and Key Takeaway
• Note that the entire design is contained in the block diagram. • The ATG configuration COE files are available as sources, so they could be modified without opening the ATG IP parameters in the block diagram. • These COE files are loaded into the block RAM simulation model when the simulation tool is launched. • During implementation, the COE files are injected into the bitstream so that they initialize the block RAM contents upon device power up. • Select the **Sources** tab on top of the pane and the **Hierarchy** tab on the bottom of the pane. • Expand **Simulation Sources** to gain access to the testbench and COE files.	Become familiar with the design sources.

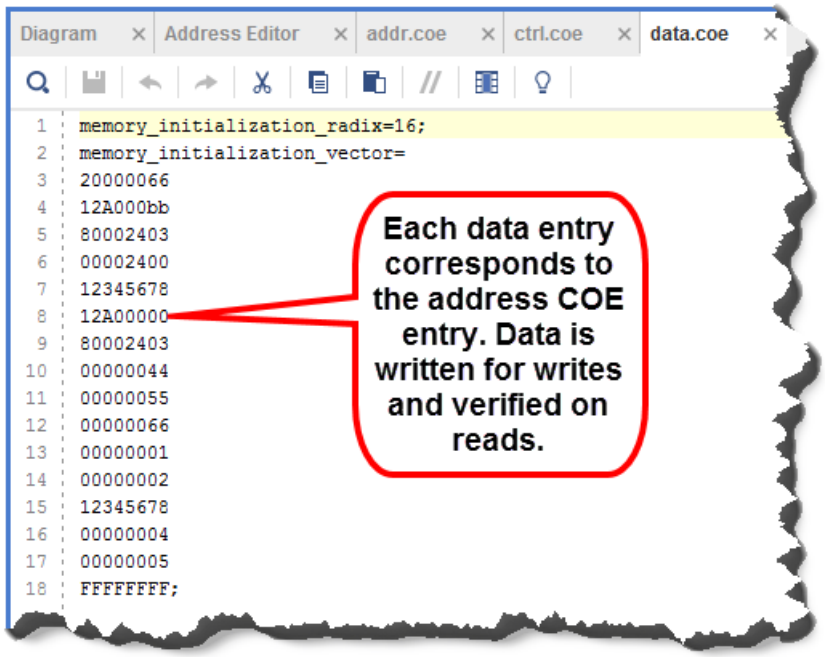


Action with Description	Point of Emphasis and Key Takeaway

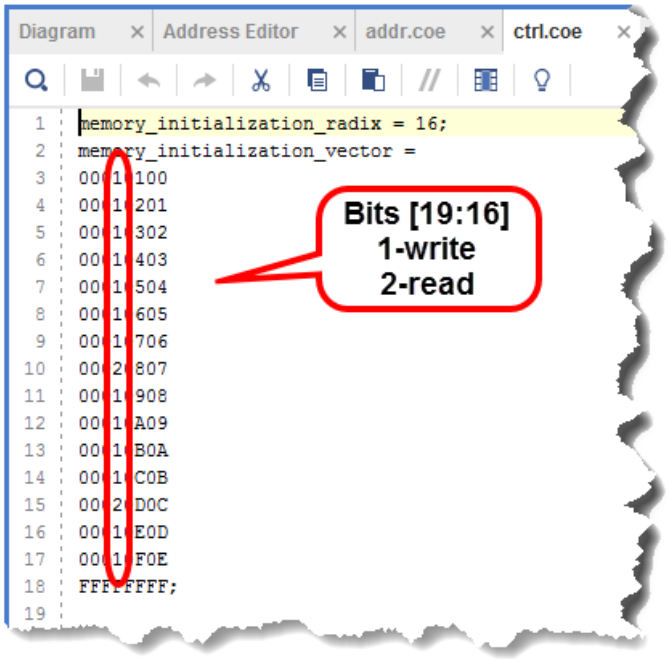
• Open the testbench file.
• Quickly show that the entire testbench consists of:
  • Clock stimulus
  • Reset stimulus
  • Monitoring of ATG done and status ports to end the simulation and report via the console
• Open the **addr.coe** file and explain that this contains a list of sixteen 32-bit addresses of the AXI4-Lite traffic that will be generated.
	When this ATG component is used in this mode, the testbench would probably be used "as is", i.e. no changes. The COE configuration files, however, can be modified to customize the generated AXI transactions.
	Note that one AXI transaction is generated per line in a COE file.

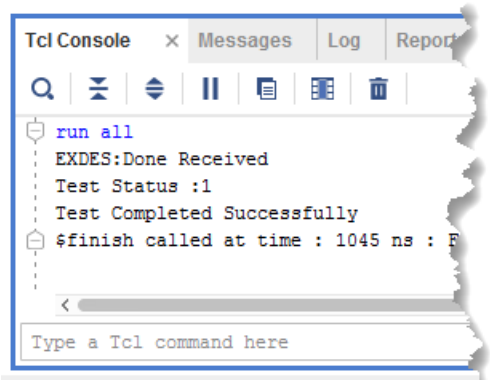


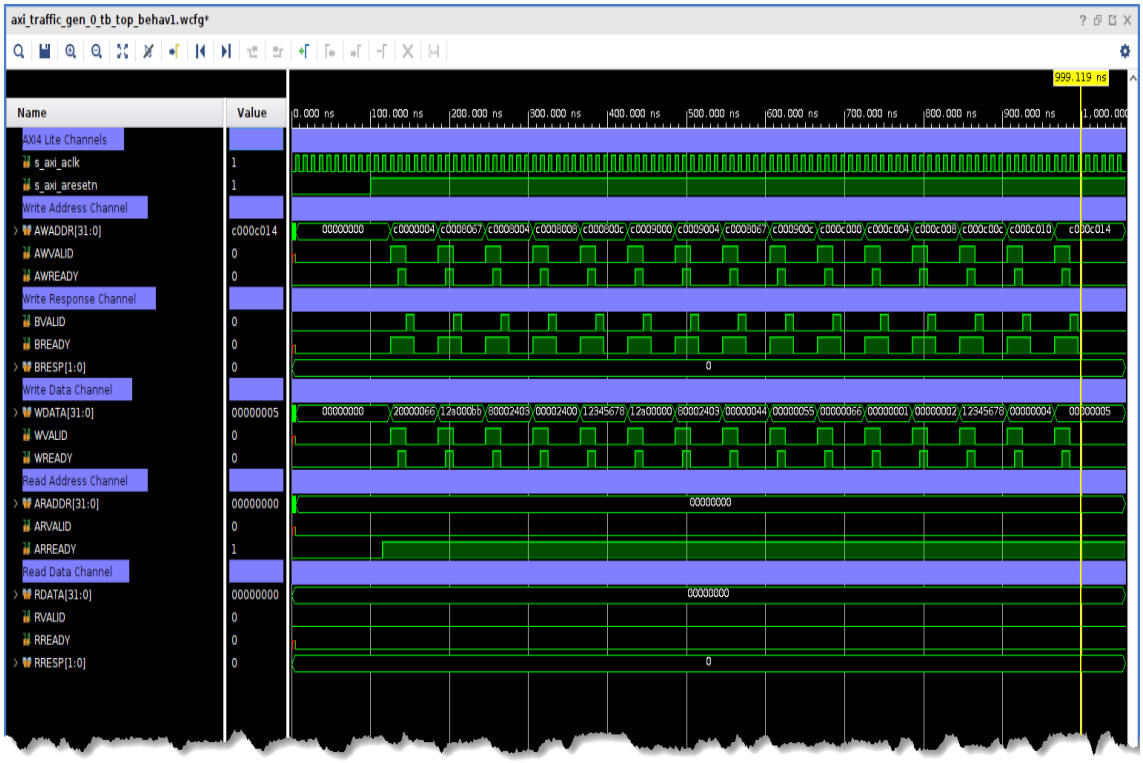
Action with Description	Point of Emphasis and Key Takeaway
• Open the **data.coe** file and explain that this contains a list of sixteen 32-bit data words corresponding to the address entries of the *addr.coe* file.	Data can be modified in this file to change write values and read verify values. If a read data does not correspond to the entry in the *data.coe* file, an internal ATG error counter is incremented.



Action with Description	Point of Emphasis and Key Takeaway
• Open the **ctrl.coe** file and explain that this contains a list of sixteen 32-bit control bits corresponding to the address entries of the *addr.coe* file. Note that it is beyond the scope of this demo to review exactly how the control word works. The important concept to understand is that a control entry determines if an AXI4-Lite transaction will be a write or read.	A full specification of how the control word is specified in PG125. For now, it is just used to determine if the transaction will be a read or write.



Action with Description	Point of Emphasis and Key Takeaway
• Click **Run Simulation** under Flow Navigator > Simulation > Run Behavioral Simulation to launch the behavioral simulation.  • After the simulation compiles and the waveform window opens (about 1 - 2 minutes), click the **Run All** icon ( ) to execute the simulation to completion. • Focus on the Tcl Console and point out the testbench-generated message of a successful completion.	The testbench monitors the ATG core status and Done output ports and report via the console traffic generation status.
 	

Action with Description	Point of Emphasis and Key Takeaway
• Focus on the waveform tab. • Click the **Zoom Fit** icon ( ). • Zoom into some of the reads and writes and explain how the valid/ready handshake mechanism works. • Verify that a read transaction is returning the same data written to the corresponding address. There is a write to 0xC000_8067 and 0xC0000_800C, which corresponds to the addresses read. The write and read operations are just random activity to the block RAM.	The AXI4-Lite traffic from the ATG to the block RAM controller can now be examined. Note that the waveform is divided into the individual AXI channels. Also note that there are a lot of AXI4-Lite write operations in the beginning that are initializing the contents of the block RAM.
	
• Select **File** > **Exit** to close the project.	

Summary

This demonstration illustrated how to generate AXI4-Lite transactions using the ATG IP core via simulation in the Vivado simulator. The resultant waveform provides insight into the various AXI channels, handshaking, and some of the critical control signals.