# ADM-XRC-KU1
# Standalone DDR4 Test
# FPGA Design
# Release: 1.1.0

**Document Revision: 1.3**
**29 Jun 2022**

| | Head Office | US Office |
|---|---|---|
| Address: | Suite L4A, 160 Dundee Street,<br>Edinburgh, EH11 1DQ, UK | 10822 West Toller Drive, Suite 250<br>Littleton, CO 80127 |
| Telephone: | +44 131 558 2600 | (303) 954 8768 |
| Fax: | +44 131 558 2700 | (866) 820 9956 - toll free |
| email: | sales@alpha-data.com | sales@alpha-data.com |
| website: | http://www.alpha-data.com | http://www.alpha-data.com |

# Table Of Contents



# List of Tables



# List of Figures

# 1 Introduction

**Supported Vivado versions**
This version of the **ADM-XRC-KU1 Standalone DDR4 Test FPGA Design** can be built with Vivado 2018.3 to 2022.1. Alpha Data cannot guarantee that this FPGA design will be fully compatible with releases of Vivado later than 2022.1.

This document describes the **ADM-XRC-KU1 Standalone DDR4 Test FPGA Design**. For a detailed discussion of using Xilinx DDR4 SDRAM (MIG) IP with the ADM-XRC-KU1 reconfigurable computing card, refer to the document Using Xilinx DDR4 SDRAM (MIG) IP with the ADM-XRC-KU1.

This FPGA design demonstrates the use of Xilinx DDR4 SDRAM (MIG) IP to interface to the DDR4 SDRAM fitted to an ADM-XRC-KU1 reconfigurable computing card.

It includes the following elements:

- Four DDR4 SDRAM controllers, one per bank, using **Xilinx DDR4 SDRAM (MIG)** IP.
- Four **memory exerciser** instances, one per DDR4 SDRAM controller.
- Four **Xilinx VIO cores**, one per **memory exerciser**, for controlling the memory test and issuing a reset to each bank.

The memory test can be initiated by manipulating the VIO cores using Vivado's Hardware Manager. Running the Standalone DDR4 Test in hardware is described in Section 4.

A Tcl script for creating a Vivado project is provided. Refer to Section 3 for a list of the available configurations and their respective Tcl scripts.

## 1.1 Structure of this package

The files and folders making up the Standalone DDR4 Test FPGA Design are organized as in Figure 1 below:

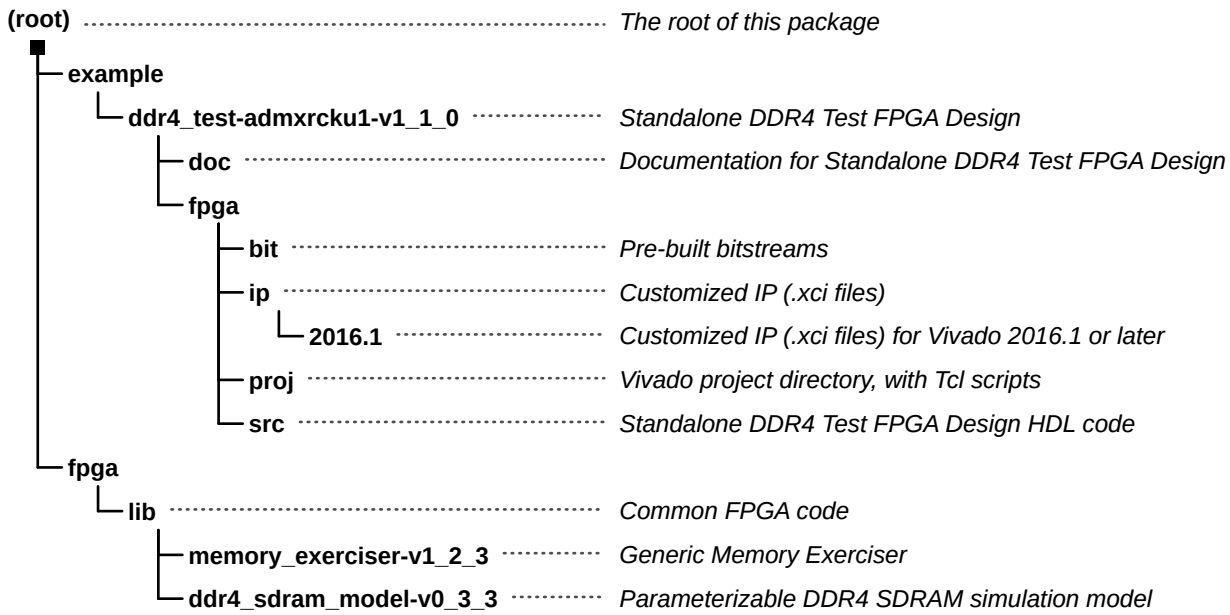


**Figure 1 : Structure of this package**

The root of this package, i.e. the directory which forms the root of the tree of directories and files making up this package, is referred to in the remainder of this document as **(root)**.

The base directory of the FPGA design, i.e. **(root)/example/ddr4_test-admxrcku1-v1_1_0** is referred to in the

remainder of this document as **(design)**.

# 2 Design description

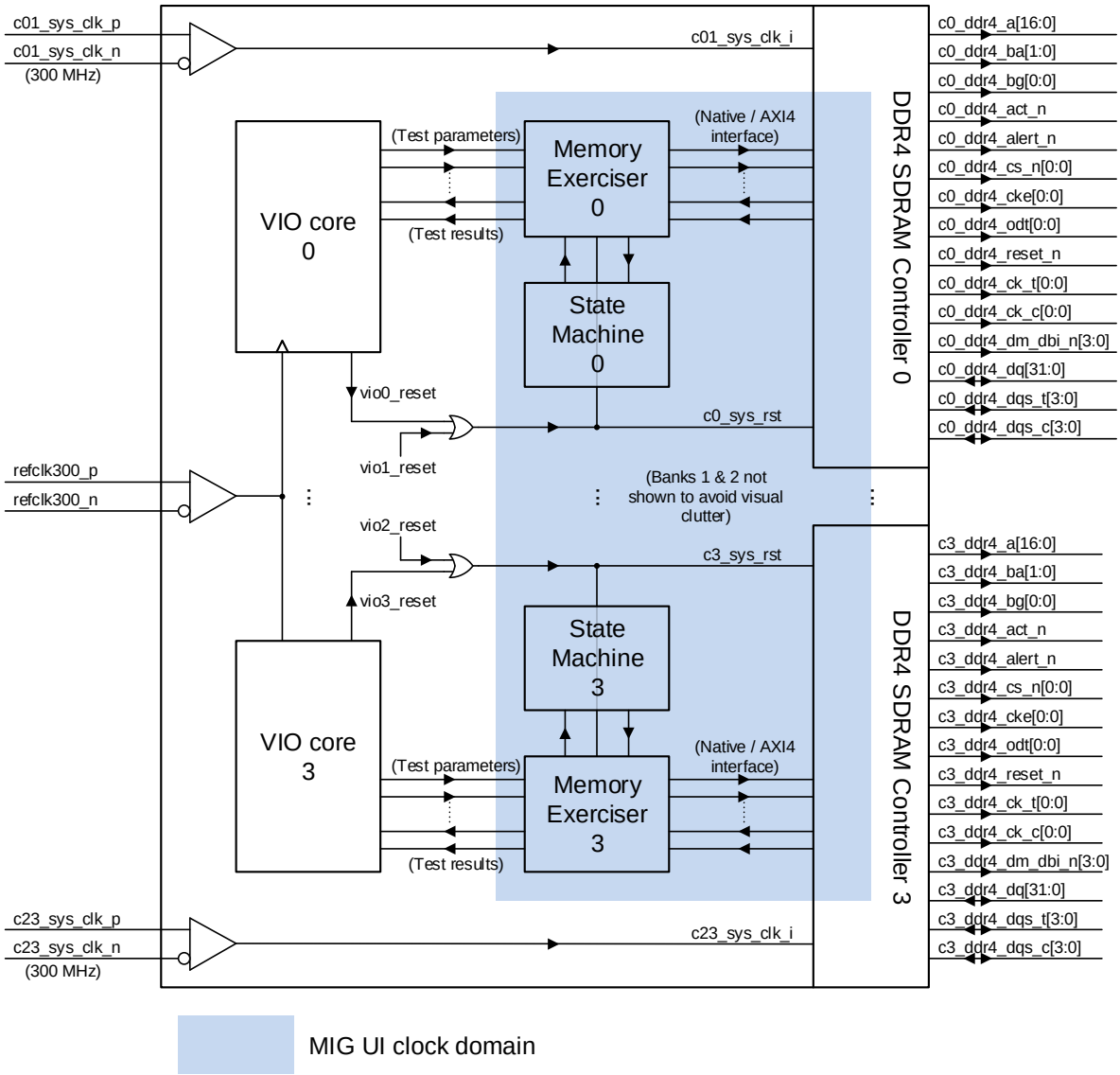


**Figure 2 : Block diagram of Standalone DDR4 Test FPGA Design**

The FPGA design depicted in Figure 2 has two main variants:

- Using the DDR4 SDRAM (MIG) IP with a 256-bit wide native interface. The top-level module is implemented by **ddr4_test_native.vhd**.
- Using the DDR4 SDRAM (MIG) IP with a 256-bit wide AXI4 interface. The top-level module is implemented by **ddr4_test_axi.vhd**.

In both variants, the structure of the FPGA design is the same; the difference is the protocol used by the DDR4 SDRAM controllers to communicate with the memory exercisers. In the native interface variant of the FPGA design, each memory exerciser drives the native interface of its associated DDR4 SDRAM controller. In the AXI4 interface variant of the FPGA design, the memory exerciser is an AXI4 master, driving the AXI4 slave interface of its associated DDR4 SDRAM controller.

The FPGA design consists of four similar sections, where each section contains a DDR4 SDRAM controller plus memory test logic and functions independently of the other sections.

Overall control of the system is accomplished via VIO cores 0 to 3. In order to perform a memory test, a VIO core pulses the **vioN_reset** signal, where N is 0 to 3. This asserts **cN_sys_rst**, which resets the corresponding

memory controller and memory test logic. Once **cN_sys_rst** is deassserted, state machine N begins its sequence.

As well as the the **vioN_reset** signal, each VIO core outputs parameters for the memory test: the starting logical address (**ctlN_offset**) number of logical words to test (**ctlN_length + 1**). A logical word is defined to be a word of data as transferred on the DDR4 SDRAM (MIG) UI, whether native or AXI4. Although these signals are brought from the VIO clock domain into the DDR4 SDRAM MIG UI clock domain without any synchronization logic, they are effectively static during the memory test as long as the user does not change them after pulsing **vioN_reset**. The memory test parameters can be different for each bank of DDR4 SDRAM.

When **cN_sys_rst** is asserted and deasserted, the corresponding section of the system executes the following sequence independently of the other sections:

1. State machine N waits until the associated DDR4 SDRAM controller asserts **cN_init_calib_complete** and deasserts **cN_ui_clk_sync_rst**.
2. State machine N then pulses **ctlN_go** for its associated memory exerciser, and transitions permanently to a "halt" state.
3. When memory exerciser N, which at this point is in its idle state, sees **ctlN_go** asserted, it begins to execute a memory test. This consists of a number of phases of reads and writes of the corresponding bank of DDR3 SDRAM. Additionally, the memory exerciser deasserts **ctlN_done** signal to indicate that a memory test is in progress. During the memory test, the memory exerciser checks data read back against expected data.
4. When memory exerciser N finishes the memory test, it asserts its **ctlN_done** signal to indicate that a memory test is not in progress. The **ctlN_done** signal, among others, can be monitored in Vivado Hardware Manager via VIO core N.

The constraints required by the FPGA design are located in **.xdc** files in the **src** folder. The constraints are split over several files according to function and are listed in Table 1:

| Constraint File in **(design)/fpga/src/** | Description |
|---|---|
| refclk300.xdc | 300 MHz reference clock constraints |
| ddr4sdram.xdc | DDR4 SDRAM common constraints |
| ddr4sdram_locs_b0_x32.xdc | DDR4 SDRAM bank 0 constraints |
| ddr4sdram_locs_b1_x32.xdc | DDR4 SDRAM bank 1 constraints |
| ddr4sdram_locs_b2_x32.xdc | DDR4 SDRAM bank 2 constraints |
| ddr4sdram_locs_b3_x32.xdc | DDR4 SDRAM bank 3 constraints |
| ddr4_test_native.xdc | Native design variant constraints |
| ddr4_test_axi.xdc | AXI4 design variant constraints |
| bitstream.xdc | Bitstream generation constraints |

**Table 1 : Design Constraints**

## 2.1 Testbench

The testbench for simulating the design is **(design)/fpga/src/tb_ddr4_test.vhd**. When a simulation is performed, the testbench instantiates (a) the unit under test (UUT) and (b) a pair of DDR4 SDRAM chip models per controller (8 chip models in total), as shown in Figure 3 below.

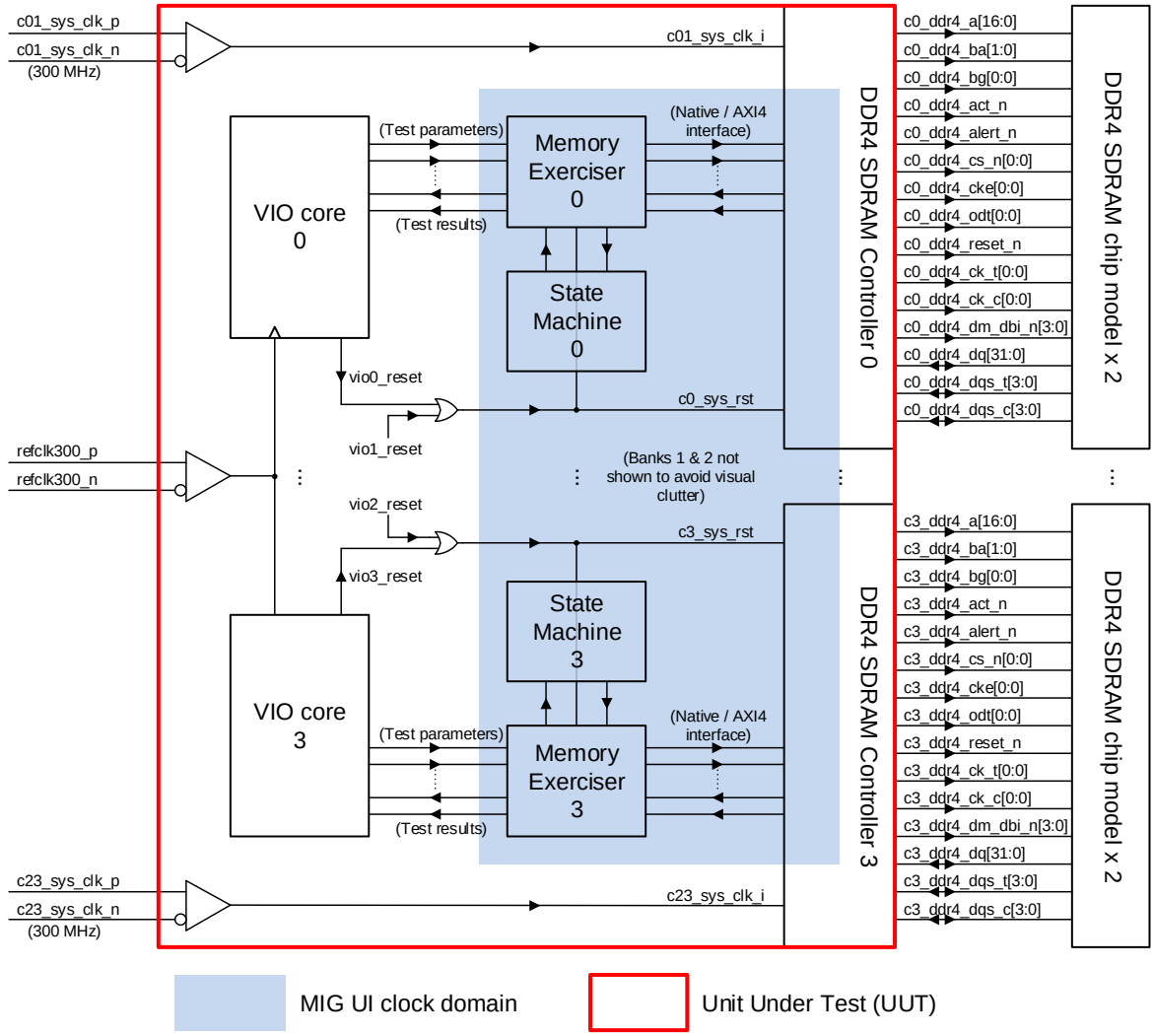


**Figure 3 : Block diagram of the testbench**

# 3 Building the Standalone DDR4 Test

Tcl scripts to create the Vivado projects for the various configurations of the FPGA design are found in the **(design)/fpga/proj/** directory. These can be **source**d within the Vivado GUI, or **source**d by Vivado in batch mode. The available Tcl scripts are listed in Table 2:

| SDRAM density & speed | MIG options | Project creation script in **(design)/fpga/proj/** |
|---|---|---|
| 8 GiB 2400 MT/s | Native interface | mkxpr-8g_2400_x32_native-ku060_2i.tcl |
| | | mkxpr-8g_2400_x32_native-ku115_2i.tcl |
| | AXI4 interface | mkxpr-8g_2400_x32_axi4-ku060_2i.tcl |
| | | mkxpr-8g_2400_x32_axi4-ku115_2i.tcl |

**Table 2 : Project creation scripts by configuration**

To generate a project, start a shell or command prompt, and issue a command of the following form:

```
cd /path/to/fpga/proj
vivado -mode batch -source <.tcl script>
```

(Windows users should use backslashes in the **cd** command, rather than forward slashes.)

For example, to generate the Vivado project for the configuration of DDR4 SDRAM (MIG) IP native interface and 8 GiB SDRAM @ 2400 MT/s (the 1st configuration in Table 2 above), invoke Vivado as follows:

```
cd /path/to/fpga/proj
vivado -mode batch -source mkxpr-8g_2400_x32_native-ku060_2i.tcl
```

(Windows users should use backslashes in the **cd** command, rather than forward slashes.)

After one or more projects have been generated using the scripts listed in Table 2, they can be opened in the Vivado GUI.

Scripts are also provided in the same directory to fully rebuild a Vivado project or all Vivado projects via the shell or command prompt. These are named similarly to the **mkxpr** scripts, except that the prefix is **rebuild**. For example, to rebuild the Vivado project for the configuration of DDR4 SDRAM (MIG) IP native interface and 8 GiB SDRAM @ 2400 MT/s, invoke Vivado as follows:

```
cd /path/to/fpga/proj
vivado -mode batch -source rebuild-8g_2400_x32_native-ku060_2i.tcl
```

(Windows users should use backslashes in the **cd** command, rather than forward slashes.)

**Note**
The pre-built bitstreams, under **(design)/fpga/bit/<configuration>**, are **not** overwritten when the FPGA design is built.

# 4 Using the Standalone DDR4 Test

To use the Standalone DDR4 Test, first determine the configuration of interest; for example, **8g_2400_x32_native-ku060_2i**. For convenience, **.bit** and **.ltx** files are provided pre-built for each configuration, under the **(design)/fpga/bit/<configuration>/** directory. The available configurations are as per Table 2.

Vivado Hardware Manager should then be used to configure the FPGA with the **.bit** file. The associated **.ltx** file tells Vivado Hardware Manager about debug probes for VIO cores etc., and so must not be omitted when configuring the FPGA.

For convenience, a debug Tcl script is provided with each pre-built **.bit** file (in the same directory), for automating the process of opening Vivado Hardware Manager, connecting to the FPGA and downloading the **.bit** file. For a particular configuration, the path of the debug Tcl script is:

**(design)/fpga/bit/<configuration>/debug-<configuration>.tcl**

Each debug Tcl script is intended to be **source**d within the Vivado GUI, and is designed to work correctly whether executed on the development machine with the corresponding Vivado project already open, or on a lab machine which does not have the corresponding Vivado project already open.

## 4.1 Monitoring & controlling the Standalone DDR4 Test using VIO cores

In Vivado Hardware Manager, the four VIO core instances are used to control and monitor the Standalone DDR4 Test.

To control the Standalone DDR4 Test, execute the debug Tcl script from the directory **(design)/fpga/bit/*<configuration>*/** for the configuration of interest. This will configure the FPGA with the appropriate **.bit** file and associate the ports of the VIO cores with recognizable signal names. At this point, Vivado Hardware Manager looks like this:

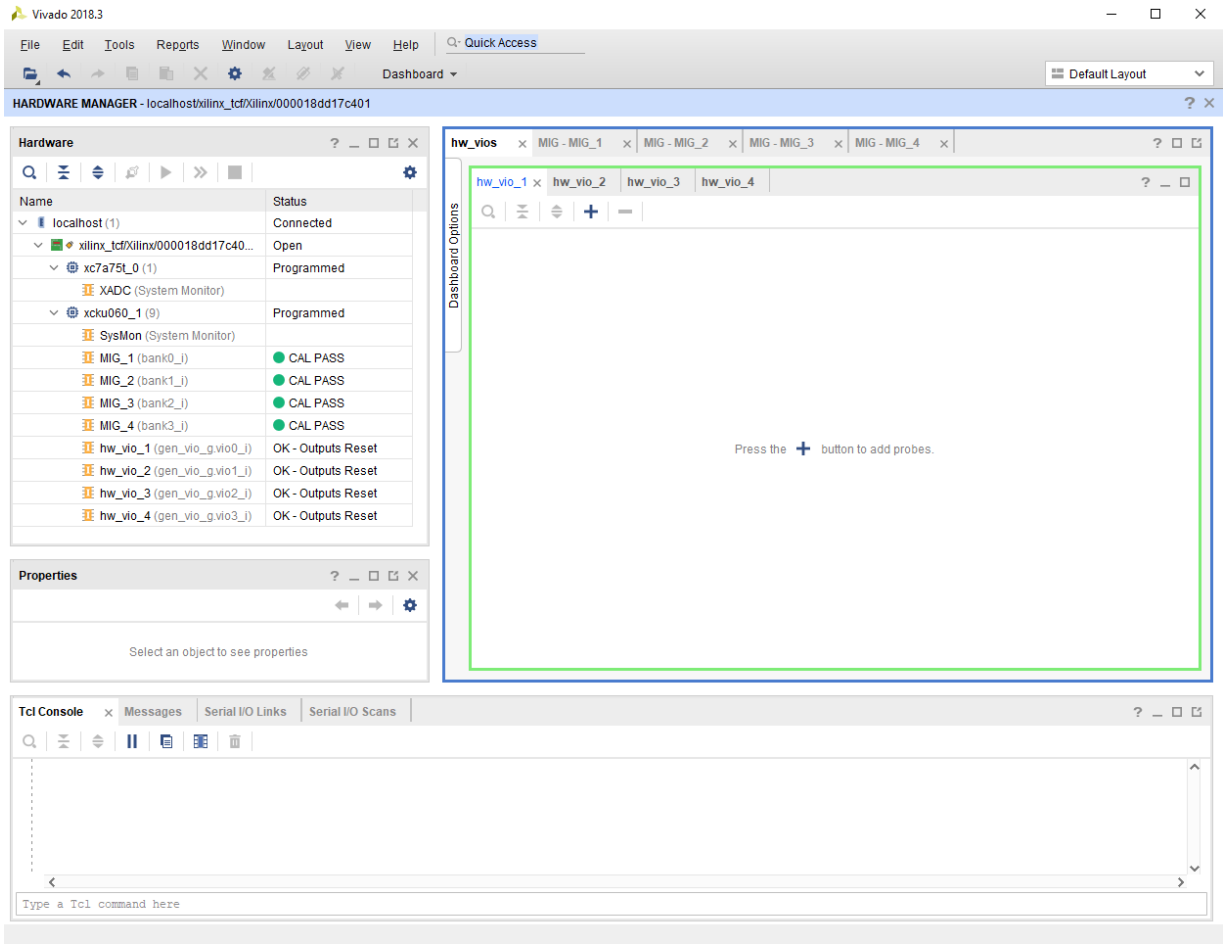


**Figure 4 : Vivado Hardware Manager after debug Tcl script**

Next, add the probes into the **hw_vios** tab. Note that the correspondence of **hw_vio_1**, **hw_vio_2**, **hw_vio_3** and **hw_vio_4** with the DDR4 SDRAMs may be reversed with respect to what might be expected. **hw_vio_1** may correspond to the fourth DDR3 SDRAM controller rather than the first, but it does not matter if this occurs.

Then, for each of **hw_vio_1**, **hw_vio_2**, **hw_vio_3** and **hw_vio_4** (where * represents 0, 1, 2, or 3):

- Configure the **vio*_reset** probe as **Active-High Button**. This can be clicked in order to reset the corresponding part of the FPGA design so that it performs calibration and memory test again.
- Configure the **c*_init_calib_complete** and **ctl*_done** probes as **LED** where **High Value Color** is **Green**.
- Configure the **ctl*_error** probe as **LED** where **High Value Color** is **Red**.

At this point, Vivado Hardware Manager looks like this:

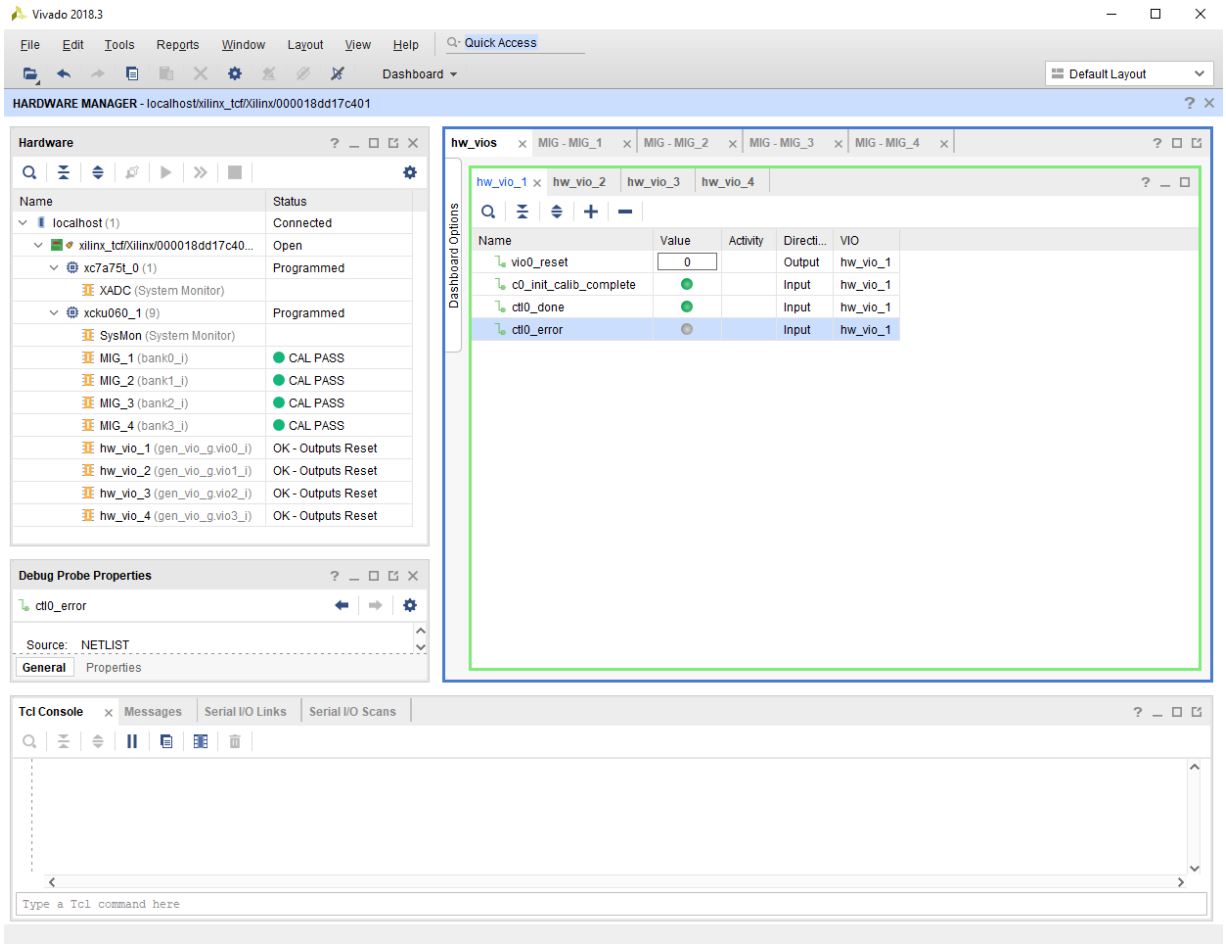


**Figure 5 : Vivado Hardware Manager after configuring debug probes**

It is likely that by the time the debug probes have been added to the **hw_vios** tab, the test that was initiated (automatically, when the FPGA was configured) has already completed. This situation is shown in Figure 5. To initiate another test, click the **vio0_reset**, **vio1_reset**, **vio2_reset** or **vio3_reset** virtual buttons.

**Note**
Alpha Data tests all ADM-XMC-KU1 reconfigurable computing cards before shipping to customers, so the **ctl_error** virtual LEDs should never be illuminated for any DDR4 SDRAM. The function of the debug probes and their usage in diagnosing a problem is currently outside the scope of this document, but may be added in a future version.

# 5 Simulating the Standalone DDR4 Test

Before simulating the Standalone DDR4 Test, the **Target simulator** and **Simulation set** must first be selected as desired, by selecting "Flow -> Simulation Settings" from the main menu:

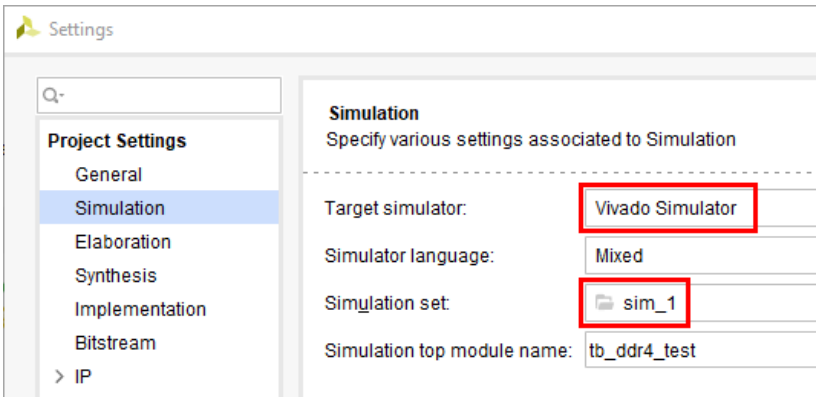


**Figure 6 : Simulation settings**

**Target simulator** may be any simulator supported by Vivado.

> **Note**
> If a simulation of the Standalone DDR4 Test has not been run before the first synthesis run, then subsequent simulations may not finish before the fixed timeout period. This issue and the workaround are discussed in Section 6.1.

To perform a simulation, use the **Run simulation** button in Vivado Flow Manager.

There are three possible ways for the simulation to terminate:

- If the memory test performed by the FPGA design finishes without data errors, the simulation outputs a message of the form **Simulation completed: PASSED** and terminates. This is the expected behaviour.
- If the memory test performed by the FPGA design finishes but detects data errors, the simulation outputs a message of the form **Simulation completed: FAILED** and terminates.
- If, for any reason, the memory test performed by the FPGA design does not finish after a fixed timeout period, the simulation outputs a message of the form **Simulation completed: TIMEOUT waiting for completion** and terminates.

# 6 Known issues

## 6.1 Workaround for incomplete DDR4 SDRAM IP simulation output products in Vivado

When simulating the design, you may encounter a **Simulation completed: TIMEOUT waiting for completion** with the preceding error **Error: Training not completed for bank n**. This is caused by an issue in the DDR4 SDRAM (MIG) IP in Vivado related to incomplete simulation output products.

This issue may be encountered if a simulation has not been run before the first synthesis run. To work around this issue, follow these steps:

(1) Open the project in the Vivado GUI.

(2) In the "Sources" window, select the "IP Sources" tab. Select the "ddr4sdram" ip and right-click to select "Generate Output Products...".

(3) A GUI box should appear which resembles the one shown in Figure 7. The list element "Behavioral Simulation" must be present.

(4) Select the "Generate" button to generate the IP output products.

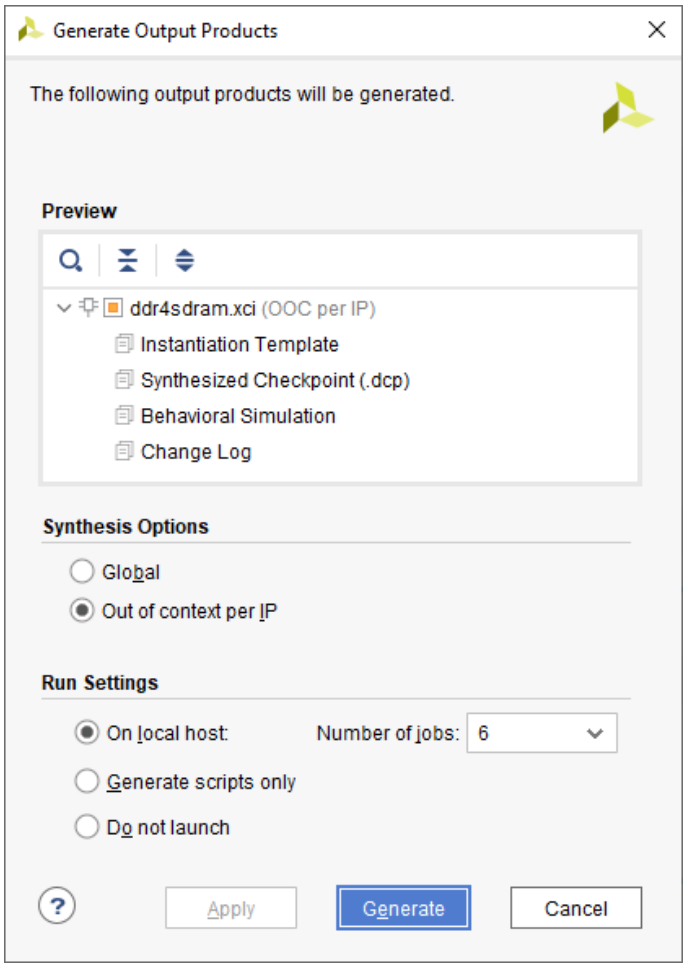


**Figure 7 : Generate Output Products GUI for ddr4sdram IP**

The Standalone DDR4 Test may now be simulated.

# Revision History

| Date | Revision | Nature of change |
|---|---|---|
| 22 June 2016 | 1.0 | Initial version. |
| 1 Aug 2016 | 1.1 | Updated for temperature grade change from E to I. |
| 17 Jan 2017 | 1.2 | Updated for Vivado 2016.4.<br>Updated to include DDR4 SDRAM model that works around XSIM VHDL access type issues. |
| 29 Jun 2022 | 1.3 | Updated for Vivado 2018.3 to 2022.1. |

Address: Suite L4A, 160 Dundee Street, Edinburgh, EH11 1DQ, UK
Telephone: +44 131 558 2600
Fax: +44 131 558 2700
email: sales@alpha-data.com
website: http://www.alpha-data.com

Address: 10822 West Toller Drive, Suite 250 Littleton, CO 80127
Telephone: (303) 954 8768
Fax: (866) 820 9956 - toll free
email: sales@alpha-data.com
website: http://www.alpha-data.com