

Alveo Data Center Accelerator Card Test

User Guide

UG1361 (v3.3.0) March 2, 2020



Revision History

The following table shows the revision history for this document.

Section	Revision Summary
03/02/2020 Version 3.3.0	
General Updates	<ul style="list-style-type: none"> • Add hardware watchdog support for all Compute Units • Software enhancement: json parsing, simplification of memory test configuration • Minor bug fixes
General Updates	Editorial/Style updates
11/06/2019 Version 3.2.1	
Power Test Case	Updates to Power Test Case
General Updates	Editorial/Style updates
10/25/2019 Version 3.2.1	
Platform Definition	Added Shell Definition json file concept and usage information.
Chapter 3: Test Case Description	Updated test case information for clarity.
10/18/2019 Version 3.2	
Initial Xilinx release.	N/A

Table of Contents

Revision History	2
Chapter 1: Overview	4
Summary.....	4
Alveo Data Center Accelerator Card Compatibility.....	5
Architecture Overview.....	5
Hardware Overview.....	6
Software Overview.....	7
Chapter 2: Xbtest Installation and Usage	9
Dependencies.....	9
Installation.....	9
Usage.....	11
Platform Definition.....	12
Chapter 3: Test Case Description	14
Supported Test Cases.....	15
Verify Test Case.....	16
DMA Test Case.....	18
Memory Test Case.....	19
Power Test Case.....	21
GT MAC Test Case.....	22
Chapter 4: Test JSON User Guide	27
Test JSON File Structure.....	27
Test JSON Members.....	28
Appendix A: Additional Resources and Legal Notices	54
Xilinx Resources.....	54
Documentation Navigator and Design Hubs.....	54
References.....	54
Please Read: Important Legal Notices.....	55

Overview

Summary

The Alveo™ Card Validation Test Solution application can be used to validate that the Alveo card hardware is operating correctly within the host server environment. The application monitors system health and validates the functionality of the essential hardware and software components of the platform.

The Alveo™ Card Validation Test Solution is also referred as "x`btest`".

You can configure `xbtest` to run the following tests:

- Validate that Compute Units (CUs) can be installed and run
- The host can communicate with target memory and CUs at the required rate
- Dissipate a programmable amount of power

The Alveo OEM Validation Test Solution is targeted at the following scenarios:

- Card installation
- After host system upgrades
- Validation prior to deploying a new application
- Validation after an outage or trouble condition



IMPORTANT! Prior to using the Card Validation Test Solution, hardware and software must be installed as described in *Getting Started with Alveo Data Center Accelerator Cards* ([UG1301](#)).

Alveo Data Center Accelerator Card Compatibility

The Alveo™ Card Test Validation Solution supports the U50, U200, U250, and U280 cards. Please contact [Xilinx support](#) for information about supported Target Platforms.

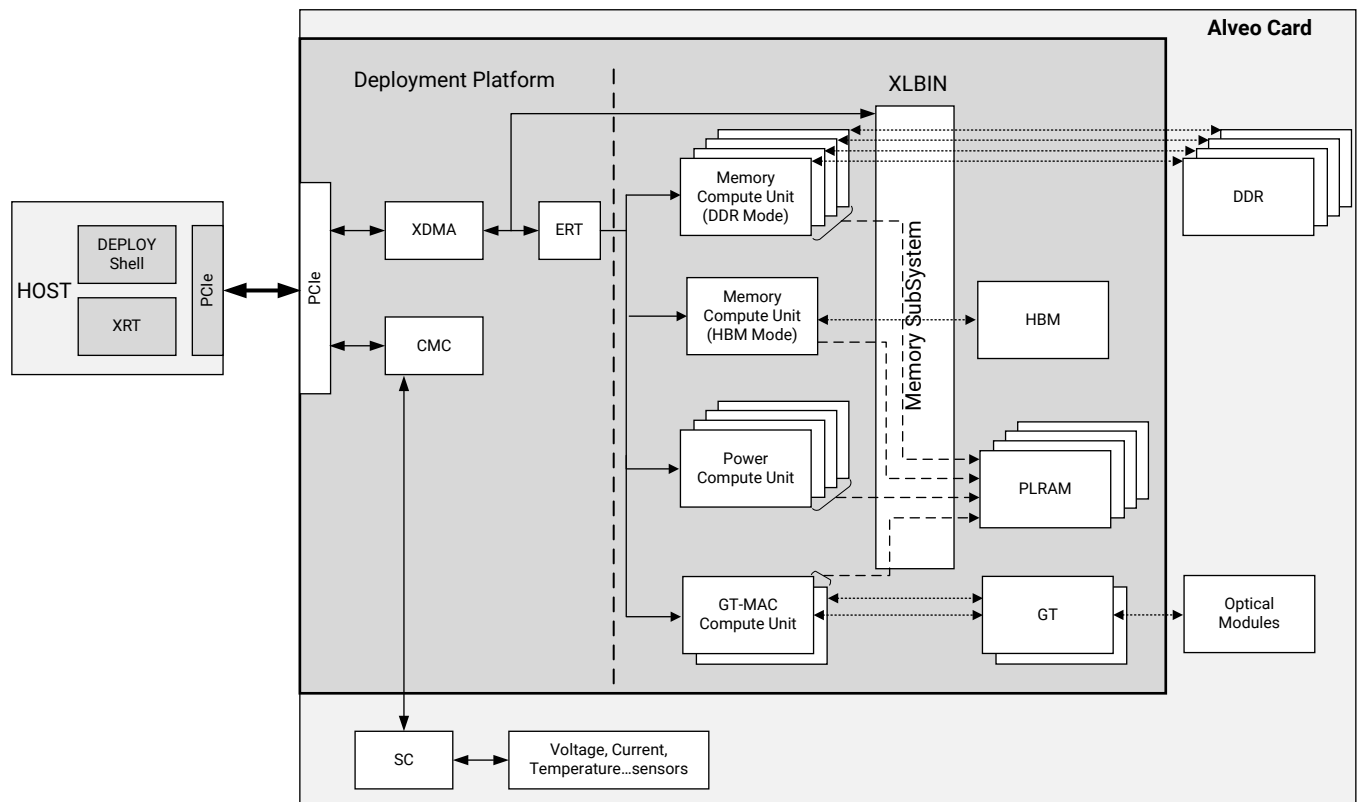
Architecture Overview

The solution comprises Application Software, which runs on the host and an `xclbin`, which is downloaded to the board. The `xclbin` may contain Power, Memory, and GT MAC Compute Units (CUs).

The following block diagram shows an Alveo™ card with an example of Deployment Platform and Compute Units (part of the Xclbin). Per Compute Unit type (Power, Memory, and GT MAC), the presence and the quantity of available Compute Units depends on the board and the platform capabilities (refer to the respective documentation).

Note: The content of the Deployment Platform depends on the platform used while the number of DDRs, HBMs, and GTs depends on the Alveo™ card used.

Figure 1: Alveo Card Block Diagram



X22886-022720

Hardware Overview

Application Software and xclbin are used in conjunction to consume power while testing/ checking memories (DDR and/or HBM) and GTs.

The basic operation of the solution consists of:

- Defining a toggle rate which will be used to consume power for a specified period of time. The host instructs the Power CU to throttle the clocks of the FF, DSP, BlockRAM, and UltraRAM available in the Power CU according to a user defined value.
- Checking the bandwidth and data integrity of all available memories (DDR and HBM).
- Checking the GT transceivers of the Alveo card at 10 Gigabit Ethernet (10GbE) and 25 Gigabit Ethernet (25GbE) line rates.

Note: There are several hardware safety mechanisms present in the various Compute Units:

- **Watchdog:** Stops the Compute Unit activity after 15 seconds in the case of the Application Software failing to perform the watchdog reset.
- **Status Register:** Detects and prevent multiple instances of Application Software trying to control/ access the same Alveo™ card.

Software Overview

The Application Software is common for all supported platforms and multiple Alveo™ cards can be tested simultaneously by running additional instances of the application. The Application Software automatically detects how many and which type of Compute Units (CUs) are present in the `xc1bin` (Power, Memory, or GT MAC).

The Application Software also provides a DMA test (Server <-> Alveo card memories).

Each test case (Power, Memory, and GT MAC) can run independently. The DMA test case runs prior to all test cases.

The Application Software uses two JSON configuration files:

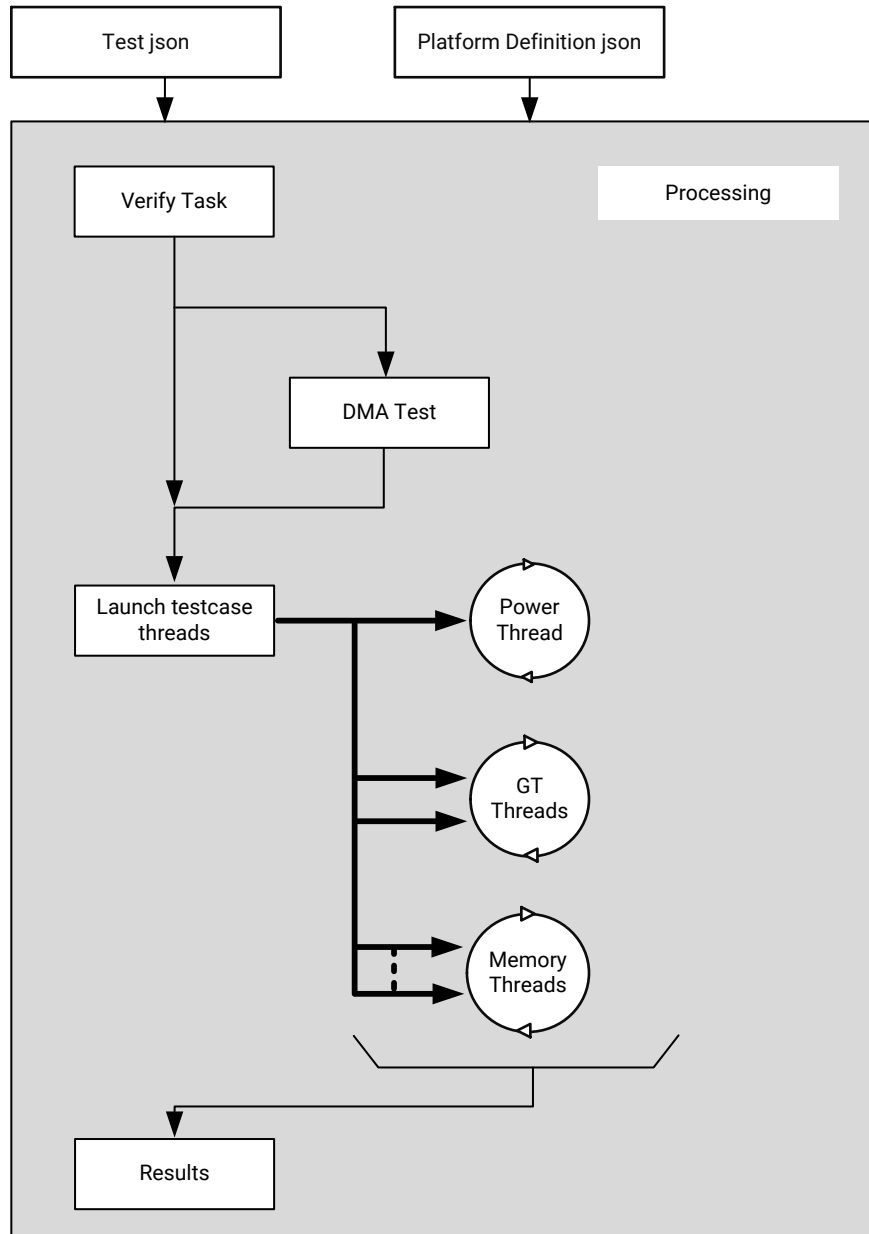
1. **Test JSON File:** this file contains the descriptions of the tests to perform.
2. **Platform Definition JSON File :** this file specifies the characteristics and the limits of the platform.



IMPORTANT! *The Platform Definition File should not be edited/modified.*

The Application Software also manages the watchdog present in the different Compute Units and checks that the Alveo™ card is not in use by another instance of the application.

Figure 2: Software Model



X23157-022120

Xbtest Installation and Usage

Dependencies



IMPORTANT! The Application Software depends on Xilinx® Runtime (XRT) (*xbutil* utility). See *Getting Started with Alveo Data Center Accelerator Cards (UG1301)* for how to install and validate Xilinx® Runtime (XRT) and the Deployment Platform.

The Application Software depends on the `json-glib-1.0` package. The following command is used to install this package:

- For CentOS and Red Hat:

```
$ sudo yum install json-glib
```

- For Ubuntu:

```
$ sudo apt install libjson-glib-1.0
```

Related Information

[Platform Definition](#)

Installation

This `xbtest` version is delivered as an archive file which can be [downloaded](#) from the Xilinx website. The archive can be extracted to a location of your choice as it does not require other installation steps. Check [Xilinx support](#) for server architecture and operating system compatibility. The archive contains the following files:

- `xbtest`: executable Application Software; compiled for a `x86_64` architecture.
- Platform specific files:
 - Two `xclbins`:
 - `xbtest.xclbin`: Contains various Compute Units, for example GT MAC, Memory (DDR and/or HBM)

- `xbtest_stress.xclbin`: Contains the same Compute Units as `xbtest.xclbin` with the addition of the Power Compute Unit
- o A set of pre-canned tests `json` files:
 - For example: `verify`, `dma`, `memory`, `gt_mac power`, `stress...`



IMPORTANT! *The toggle rate used by the pre-canned test is given as example. It may require manual adjustments as the actual board power consumption depends on the environmental test conditions.*

- o Platform Definition `json` file



CAUTION! *These files are specific to each Deployment Platform. Ensure the correct files are associated to your Deployment Platform.*

The following steps can be used to verify that `xbtest` is installed correctly:

1. Check that your card is installed and operating correctly: See *Getting Started with Alveo Data Center Accelerator Cards (UG1301)* for instructions on how to install Xilinx® Runtime (XRT) and the Deployment Platform.
2. Extract the `xbtest` archive in, for example, `<my_folder>`.
3. Move into `xbtest` folder.
 - `cd <my_folder>/xbtest_v3_3_0`
4. Launch `xbtest` with the following command:

```
./bin/xbtest -j lib/xilinx_u280_xdma_201920_3/test/verify.json
```



IMPORTANT! *Check Host Application Usage and Dependency for more information on how to run `xbtest`.*



TIP: *Optionally use "`-d`" and "`-l`" to target different cards. See Usage for command line options definition.*



TIP: *The pre-canned tests can be used as templates to create your own tests. In such case, do not forget to update all filepaths defined in your `test.json` to point to their `<my_folder>` location (see Test Environment Members).*

Related Information

[Test Environment Members Usage](#)

Usage

The Application Software is named `xbtest`. The following example command launches it in the current working directory:

```
./bin/xbtest -j my_test.json
```

The available command line options are shown in the following table:

Table 1: Application Software command line options

Command Line Option	Description
-j	Launch <code>xbtest</code> with user defined <code>json</code> file: <ul style="list-style-type: none"> <code>./xbtest -j PathTo/mytests.json</code>
-v	Display the version: <ul style="list-style-type: none"> <code>./xbtest -v</code>
-h	Display the help: <ul style="list-style-type: none"> <code>./xbtest -h</code>
-d	Overwrite the <code>device_idx</code> member in the <code>json</code> file. Allow the selection of the Alveo card within the user server. The integer value to provide with "-d" options is the index returned by the Xilinx® Runtime (XRT) command <code>xbutil scan</code> or <code>xbmgmt scan</code> : <ul style="list-style-type: none"> <code>xbtest -j my_tests.json -d 1</code>
-l	Define a logging directory in which all <code>output_file</code> defined in any test case will be stored. All messages displayed during the execution will be also stored into a <code>xbtest.log</code> file available in the provided directory. <ul style="list-style-type: none"> <code>xbtest -j my_tests.json -l PathTo/LogDir</code> <p>Note: When this option is used, all <code>output_file</code> must be a file name only (absolute or relative path must not be present in <code>output_file</code>).</p>
-x	Overwrite the <code>xclbin</code> member present in the <code>json</code> file: <ul style="list-style-type: none"> <code>xbtest -j my_tests.json -x PathTo/File.xclbin</code>
-e	Overwrite the <code>xbtest_pfm_def</code> member present in the <code>json</code> file: <ul style="list-style-type: none"> <code>xbtest -j my_tests.json -e PathTo/xbtest_pfm_def.json</code>

Notes:

- "-j", "-d", "-l", "-x", "-e" can be used simultaneously within the same command line.
- "-v" & "-h" will prevent any tests being launched.



TIP: Use `"-d"` and `"-l"` options together to test simultaneously multiple Alveo card with the same test json and store all various files into different directories:

- `xbttest -j test.json -d 0 -l log_0`
- `xbttest -j test.json -d 1 -l log_1`

There are currently four test cases.

Note: Power, Memory, and GT MAC test cases are run in parallel, while all tests within a test case are performed serially.

There is no communication between test cases; a failure/error in one test case does not affect the progress of the other test cases. Within a test case, the tests are performed serially according to the `test_sequence` provided and an error within a test sequence will not impact the next test case.

Note: Reaching any physical limit (temperature or power) of the Alveo card will halt all tests (see Platform Definition).

Related Information

[Platform Definition](#)

Platform Definition

`Xbttest` uses a Platform Definition `json` file which is provided with `xbttest`. The file specifies the characteristics and the limits of the target platform (for example, quantity and size of DDR/HBM) and is specific to each target platform. The content and structure of the Platform Definition `json` file is beyond the scope of this document.

These definitions drive the limits and the pass/fail criteria of `xbttest` for:

- Minimum and maximum DMA bandwidths.
- Minimum and maximum Memory bandwidths.



IMPORTANT! A path to the Platform Definition `json` file must be present in the `test.json` file.

Based on the physical characteristics of the Alveo card present in the Platform Definition `json` file, the test `json` defined by the user is verified. For example, no DDR test can be run if there is no valid DDR definition.



TIP: The Platform Definition parameters are displayed at the beginning of each test (as information with the tag `"XBT_PFM_DEF"`).

Related Information

[Memory Test Case Members](#)

[Test Environment Members](#)

Test Case Description

All test cases are configured via one common test `json` file. This `json` file contains various schema - one per test case. Schemas can be removed/added to disable/enable test cases.

```
{
  "device": "xilinx_u280_xdma_201920_3",
  "device_idx": 0,
  "xclbin": "../xbtest.xclbin",
  "xbtest_pfm_def": "../xbtest_pfm_def_u280.json",
  "testcases":
  [
    {
      "type": "dma",
      "parameters":
      {
        "test_sequence": [[10, "DDR", "all", 256], [10, "HBM", 2, 256]],
        "output_file": "dma_output_file"
      }
    },
    {
      "type": "power",
      "parameters":
      {
        "test_sequence": [[60, 15], [120, 50]],
        "test_sequence_mode": "config_duration_toggle",
        "output_file": "power_output_file"
      }
    },
    {
      "type": "memory_ddr",
      "parameters":
      {
        "test_sequence": [{"Alternate_Wr_Rd", 60}],
        "output_file": "memory_output_file"
      }
    },
    {
      "type": "memory_hbm",
      "parameters":
      {
        "test_sequence": [{"Alternate_Wr_Rd", 60}],
        "output_file": "memory_output_file"
      }
    },
    {
      "type": "gt_mac",
      "parameters":
      {
        "test_sequence": [[2, "conf"], [1, "clr_stat"], [10, "run"]],
        "line_rate": "10gbe",
        [1, "status"]],

```

```

        "output_file": "gt_mac_output_file"
    }
}
    ]
}

```

Note: Refer to the test `json` examples provided for accurate test configuration (some members may not be available depending on your Alveo card and platform).

Supported Test Cases

A test case (DMA, Power, Memory or GT MAC) is composed of:

- A series of tests (also called test sequence). The duration and configuration of each test can be specified individually.
- Common configurations for all tests within a test case (for example, results file).

Each type of test case has a different sets of pass/fail criteria Test case selection and configuration is performed via a test `json` file.

Verify

- Detects and report available CUs.
- Checks compatibility.
- Displays Platform Definition.
- Checks all CUs capabilities and limitations.
- Basic host-CUs communication test: read/write of a scratch register.

DMA

- Check data transfer between the host and the memories available (DDR and/or HBM).
- DDR and HBM are tested separately.
- Uses counter for data integrity check.
- Measure Read and Write Host-Memory bandwidths.

Memory

- Check data transfer between the Compute Unit and the memories available (DDR and/or HBM).

- DDR and HBM are tested separately.
 - The configuration settings to check the DDR are different to those of the HBM. HBM test configurations apply to *all* HBM while DDR test configurations apply to *all* DDR.
- Uses PRBS31 for data integrity check.
- Measure Read and Write bandwidths.

Power

- Set the power consumed by the board by controlling the toggle rate of the resource present in the Compute Unit..

Note: Power CU only exercises the Programmable Logic (PL) of the `xclbin`. The maximum and minimum power depends on:

- The Alveo™ card selected.
- The platform used (see Platform Definition).
- The configuration of other tests running simultaneously to the power test.

Related Information

[Platform Definition](#)

GT MAC

- Checks the GT transceiver of the Alveo card at 10 Gigabit Ethernet (10GbE) and 25 Gigabit Ethernet (25GbE) line rates.
- Uses the Xilinx XXV Ethernet IP core (see *10G/25G High Speed Ethernet Subsystem Product Guide (PG210)*).
- Includes a packet generator which allows an Alveo card with simple loopback cables to generate packets, and to verify that these packets are received error free.

Verify Test Case

This test case requires:

- A device name: The platform name (for example `xilinx_u280_xdma_201920_3`).
- A path to an `xclbin` compatible with the target platform specified above.
- (Optionally) a `device_idx`, in case the system contains multiple identical target platforms.
- A path to the Platform Definition `json` file of the platform (for example, `xbtest_pfm_def_u280.json`).

These parameters are defined in the `test json` file and constitute the minimum requirements to run `xbtest` Application Software.

The *Verify* test case is always executed at the start and is not configurable. If any of its checks described below fails, none of the test cases are executed.

Application Software automatically detects the Compute Units (CUs) present in the `xclbin`.

The *Verify* test case crosschecks the contents of `xclbin` with regard to:

- Its compatibility with the Application Software
- The content of the `test json` file

Xclbin vs. Application Software

Once uniquely identified, each CU (Power, Memory, and/or GT MAC) is checked for:

- Compatibility with:
 - Application Software version
 - CUs stat information
- Basic communication:
 - Multiple read/write to each CU scratch register are performed

Application Software and Platform Definition vs. test json

An initial sanity check of the `test json` file is performed when the Application Software starts. All `json` members are checked for validity and compatibility (value type and range) with the Application Software and the Platform Definition `json` file. This also includes a check of the `test_sequence` parameters for each test case.

Xclbin vs. test json file

If a test case is described in the `test json` file but the `xclbin` does not contain the associated CU, the “verify” test case fails. If a test case is not described in the `test json` file, the associated CU will stay in its idle state and this is *not* considered as a Verify test case failure.

DMA Test Case

The goal of this test case is to check communications between host and DDR and/or HBM memory through the PCIe DMA. Data integrity and write/read bandwidths are checked.

The DMA test case consists of writing and reading back a certain quantity of data (`total_size`, in MB) over and over during a certain period of time. A Write-Read -Check Cycle is never interrupted, meaning that:

- The `total_size` can be separately specified for DDR and for HBM
- The `total_size` of data is always fully sent, read back and checked for data integrity
- If required, a test duration may be extended to perform all Write-Read-Check Cycle operations.

The data is generated via 8-bit counter which is randomly initialized at the beginning of each Write-Read-Check Cycle.

The `total_size` is predefined for each type of memory (HBM or DDR) but can be changed (see DMA Test Case Members). Per memory type (HBM or DDR), the `total_size` default value is the size defined in the Platform Definition; capped at 4096MB.

The `total_size` of data is split into buffers (`buffer_size`) which are transferred (via OpenCL) to or from the Alveo™ card. The `buffer_size` can be changed (see DMA Test Case Members)

The bandwidths are computed after each `total_size` of data transfer and the values are averaged over the test duration.



IMPORTANT! *By default, the average read and write bandwidths are not checked against any pass/fail criteria but this can be overruled (see DMA Test Case Members).*

Related Information

[Test JSON User Guide](#)

[DMA Test Case Members](#)

Main Test Steps

A DMA test consists of the following steps:

1. Allocate N host buffers of `buffer_size` MB (aligned with memory page size), where N equals `total_size` divided by `buffer_size`.
2. Allocate and initialize the reference buffer used to check data integrity.

3. The following steps are repeated for the duration of the test:
 - a. Set host buffers with reference data (8-bit counter).
 - b. Write host buffers to the board memory, measure bandwidth.
 - c. Reset host buffers to 0.
 - d. Read from the board memory, measure bandwidth.
 - e. Check that the host buffers contain the same data as the reference buffers (data integrity).

Note: These steps constitute a Write-Read-Check Cycle which is always entirely executed.

4. Compute write and read minimum, maximum, and average bandwidths.
5. If enabled, compare the average read and write bandwidths against their thresholds.
6. Release all host buffers.

Test Parameters

Following are the mandatory test configuration parameters, more are available (see DMA Test Case Members):

- `duration`: The duration of the test in seconds.
- `mem_type`: Type of memory to access (DDR, or HBM).
- `mem_index`: Index of the memory to access.

Memory Test Case

The goal of this test case is to check communications between the Memory CUs and the memories available on the Alveo card (DDR) or on the FPGA (HBM).

Note: In the DMA test case, data is transferred between the host and memories. In the Memory test case, the Application Software commands the Memory CUs to transfer data between the CU and their associated memories.

The memory test case includes the following features:

- All memories are tested in parallel.
- Sequences are defined for memory types (DDR or HBM), and all memories of a particular type are tested with the same sequence.
- Data integrity is checked using PRBS31 generator/checker within the CU.
- Write and read bandwidths are measured.

- All transfers are performed using linear addressing.

Prior to the first test and if enabled (see Memory Test case member), a verification of the communication between the Memory CUs and the memories is performed. For each memory, PRBS data are written with a known quantity of errors, then read back and validated (see Preliminary Test).

Related Information

[Memory Test Case Members](#)

[Verbosity Level](#)

[Preliminary Test](#)

Main Test Steps

1. The Preliminary Test is run (if enabled).
2. For each test configuration the following steps are repeated:
 - a. The test is run for at least the defined duration. The entire range of the memory is always checked, meaning that, if needed, the test duration is extended.
 - b. Every second, the Application Software requests status and measurements from the CU:
 - The instantaneous read and write bandwidths.
 - The data integrity status; an error message is displayed in case of mistake.
 - c. After the test completes, the average read and write bandwidths are displayed. They are only checked against thresholds if the following condition is met:
 - Test duration is greater than 20s.

Related Information

[Memory Test Case Members](#)

Preliminary Test

If enabled, this test validates that the correct memory is under test. The following steps are performed:

1. PRBS31 data with a known quantity of errors is sent to the memory. The quantity of errors differs for each memory.
2. The memory is read back and verified against the expected quantity of errors.
3. Data integrity of the memory is re-established by writing valid PRBS31 data.

Test Parameters

The mandatory test configuration parameters are as follows (see Memory Test Case Members for more details):

- `mem_test_mode`: Describes data transfer mode: "Alternate_Wr_Rd", "Only_Wr", "Only_Rd".
- `duration`: The test duration in seconds.



IMPORTANT! To perform any "Only_Rd", the memory content must have filled with valid PRBS31 data. Within your `test_sequence`, define at least 1 Only_Wr test prior to Only_Rd test

Related Information

[Memory Test Case Members](#)

Power Test Case

Power consumption is controlled by adjusting the toggle rate of the clock driving all Flip-Flops, DSPs, block RAMs, and UltraRAMs present in the Power Compute Unit (CU). The toggle rate range is between 0% and 100%. The power is measured every second. It is computed based on the current and voltage measurements available via Xilinx® Runtime (XRT).

The following basic safety mechanisms are in place to limit potential damage to the Alveo card in cases of accidental misuse or demanding test environmental conditions:

- **Temperature Limit:** All tests are aborted if the temperature limit is reached.
- **Power Limit:** All tests are aborted if the power limit is reached.

Temperature and power limits are defined in Platform Definition File (see Platform Definition).



WARNING! The Power CU has been intentionally designed to exceed the power capacity of the card. You may damage your card and cause your server/workstation to reboot if you try to for example (but not limited to):

- Use a high toggle rate
 - Use a particularly demanding test sequence (for example, alternating between 0% and a high toggle rate for a short period of time).
-

Note: No checks are made on the power consumed, so the power test always passes. The user is responsible of monitoring the consumed power which is displayed by the Application Software.



TIP: You should calibrate your target platform and card against the toggle rate according to your test environment in order to determine:

- The maximum permissible toggle rate

- The relationship between the toggle rate and the power consumed
-

A simple calibration method could be as follows: Starting from 0%, increase the toggle rate by 5% and, for each toggle rate step, let the power and temperature stabilize for two minutes. This could help you establish the relationship between toggle rate and power.



IMPORTANT! *Ensure that the environmental conditions (for example temperature) used during calibration are similar to those used during the tests.*

Memory (DDR or HBM) and GT MAC tests have a significant impact on the power consumed:

- **DDR:** When running four DDRs simultaneously, the memory test consumes approximately 20W (write mode) or 15W (read mode).
- **HBM:** For example, when eight HBM ports are used, the memory test consumes between 7W and 8W.
- **GT MAC:** When two GT MAC CUs are present, the 25GbE mode uses $\pm 6W$ more than the 10GbE mode.
- **Logic:** Memory and GT MAC CU, and the memory subsystem also consume several watts.

Note: These values are indicative and may vary from card to card. Moreover they also depend on test environmental conditions such as cooling, etc.

Care must be taken when mixing test types or when changing the mode of other tests while the power test is running. For example, power will increase if the DDR memory test changes from `Only_Rd` to `Only_Wr` mode. Power will decrease when a Memory (DDR or HBM) or GT MAC test sequence ends.

For active cards, the fan power is part of the measured power. Temperature fluctuations will result in variations of the fan speed/power.

Note: The U50 card's total power budget is 75W but 10W are used by the HBM, leaving up to 65W controllable via toggle rate.

Related Information

[Power Test Case Members](#)

[Verbosity Level](#)

[Platform Definition](#)

GT MAC Test Case

The goal of this test case is to allow verification of GT transceivers on Alveo cards at 10GbE and 25GbE line rates.

This Compute Unit (CU) instantiates the 10G/25G High Speed Ethernet Subsystem IP core and allows the core to be configured from a test `json` file. (See *10G/25G High Speed Ethernet Subsystem Product Guide (PG210)*).

The CU also includes a packet generator, which allows a card with simple electrical or optical loopback cables to generate packets, and also verify that the generated packets have been received back, error free.

Note: Under some circumstances the packet generator may not be able to launch packets at the requested rate. This is most likely to occur when operating at 25 GbE, with small (for example 64 byte) packets. Reducing the number of active MACs and/or increasing the packet size should allow the maximum rate to be achieved (see GT MAC Test Case Members).

A test is generally composed of four steps and a definition of the hardware environment (see GT MAC Test Case Members).

Typical test steps are as follows:

1. Configuration
2. Clear Status
3. Run
4. Report/Check Status



WARNING!

- All GT MAC CUs present in the `xclbin` are tested with the same test sequence.
 - By default, after `xclbin` download, the GT MAC CU generates IDLE packets at 25GbE rate.
 - When the test sequence is over, the GT MAC CU continues to send traffic as per its last configuration.
 - It is not possible to configure the source nor the destination MAC address.
-

Related Information

[GT MAC Test Case Members](#)

MAC_STAT Status Description

Each time the `status` command is executed, the following registers are read from the MAC hardware for each active MAC lane.

All registers are stored in hardware using 48 bits, and extended to 64 bits when read by software.

Note: The 48 bits `RX_TOTAL_BYTES`, `RX_TOTAL_GOOD_BYTES` and `TX_TOTAL_BYTES` counters could saturate after approximately 25 hours of maximum rate operation at 25GbE. It is therefore recommended that the test duration does not exceed 24 hours between `status` commands.

In addition, if the RX_TOTAL_GOOD_PACKETS count is equal to 0, then a "No Good Packets received" message is reported, and the test will fail.

The optional parameter `match_tx_rx` causes the values of RX_TOTAL_GOOD_PACKETS and TX_TOTAL_PACKETS to be compared, and the values of RX_TOTAL_GOOD_BYTES and TX_TOTAL_BYTES to be compared. A mismatch will cause the messages "Tx vs Rx Packets mismatch." and/or "Tx vs Rx Bytes mismatch." to be reported, and the test will fail.

If the transmitter of one MAC instance is connected to the receiver of another MAC instance, then using the `tx_mapping_n` option allows the `match_tx_rx` to be used, with the comparison being made between the correct MAC Tx and Rx instances.

In the table following table, the "Register Type" represents how the register is verified:

- Check: the content of the register must be null. Any other value will generate an error.
- Info: the content of the register is displayed as information (no verification performed).

Register Name	Register Type	Description
<code>cycle_count</code>	Info	Number of transceiver clock domain cycles (approximately 1.5625e8 / sec at 10 GbE and 3.90612e8 / sec at 25 GbE). Note: This count may return inaccurate values.
<code>fec_inc_cant_correct_count</code>	Check	This count indicates how many uncorrected bit errors in the corresponding Clause 74 FEC Frame.
<code>fec_inc_correct_count</code>	Check	This count indicates how many corrected bit errors in the corresponding Clause 74 FEC Frame.
<code>rx_bad_code</code>	Check	This count indicates how many cycles the RX PCS receive state machine is in the RX_E state as defined by IEEE Std. 802.3.
<code>rx_bad_fcs</code>	Check	The value of this count indicates packets received with a bad FCS, but not a stomped FCS. A stomped FCS is defined as the bitwise inverse of the expected good FCS.
<code>rx_broadcast</code>	Info	Increment for good broadcast packets.
<code>rx_error</code>	Check	This count indicates a mismatch occurred for the test pattern in the RX core.
<code>rx_fragment</code>	Check	Increment for packets shorter than <code>stat_rx_min_packet_len</code> with bad FCS.
<code>rx_framing_err</code>	Check	This count is used to keep track of sync header errors. The <code>stat_rx_framing_err</code> output indicates how many sync header errors were received.
<code>rx_inrangeerr</code>	Check	Increment for packets with Length field error but with good FCS.
<code>rx_jabber</code>	Check	Increment for packets longer than <code>ctl_rx_max_packet_len</code> with bad FCS.
<code>rx_multicast</code>	Info	Increment for good multicast packets.
<code>rx_oversize</code>	Check	Increment for packets longer than <code>ctl_rx_max_packet_len</code> with good FCS.
<code>rx_packet_64_bytes</code>	Info	Increment for good and bad packets received that contain 64 bytes.

Register Name	Register Type	Description
rx_packet_65_127_bytes	Info	Increment for good and bad packets received that contain 65 to 127 bytes.
rx_packet_128_255_bytes	Info	Increment for good and bad packets received that contain 128 to 255 bytes.
rx_packet_256_511	Info	Increment for good and bad packets received that contain 256 to 511 bytes.
rx_packet_512_1023_bytes	Info	Increment for good and bad packets received that contain 512 to 1,023 bytes.
rx_packet_1024_1518_bytes	Info	Increment for good and bad packets received that contain 1,024 to 1,518 bytes.
rx_packet_1519_1522_bytes	Info	Increment for good and bad packets received that contain 1,519 to 1,522 bytes.
rx_packet_1523_1548_bytes	Info	Increment for good and bad packets received that contain 1,523 to 1,548 bytes.
rx_packet_1549_2047_bytes	Info	Increment for good and bad packets received that contain 1,549 to 2,047 bytes.
rx_packet_2048_4095_bytes	Info	Increment for good and bad packets received that contain 2,048 to 4,095 bytes.
rx_packet_4096_8191_bytes	Info	Increment for good and bad packets received that contain 4,096 to 8,191 bytes.
rx_packet_8192_9215_bytes	Info	Increment for good and bad packets received that contain 8,192 to 9,215 bytes.
rx_packet_bad_fcs	Check	Increment for packets between 64 and <code>ctl_rx_max_packet_lenbytes</code> that have FCS errors.
rx_packet_large	Info	Increment for all packets that are more than 9,215 bytes long.
rx_packet_small	Check	Increment for all packets that are less than 64 bytes long. Packets that are less than 4 bytes are dropped.
rx_pause	Info	Increment for 802.3x Ethernet MAC Pause packet with good FCS.
rx_rsfec_corrected_cw_inc	Check	This count will increment if the RS-FEC decoder detected and corrected a bit errors in the corresponding frame.
rx_rsfec_err_count0_inc	Check	Increment for RS-FEC detected errors.
rx_rsfec_uncorrected_cw_inc	Check	This count will increment if the RS-FEC decoder detected uncorrectable bit errors in the corresponding frame.
rx_stomped_fcs	Check	The value of this count indicates packets were received with a stomped FCS. A stomped FCS is defined as the bitwise inverse of the expected good FCS.
rx_test_pattern_mismatch	Check	This count indicates how many mismatches occurred for the test pattern in the RX core.
rx_toolong	Check	Increment for packets longer than <code>ctl_rx_max_packet_len</code> with good and bad FCS.
rx_total_bytes	Info	Increment for the total number of bytes received.
rx_total_good_bytes	Info	Increment for the total number of good bytes received. This value is only non-zero when a packet is received completely and contains no errors.
rx_total_good_packets	Info	Increment for the total number of good packets received. This value is only non-zero when a packet is received completely and contains no errors.
rx_total_packets	Info	Increment for the total number of packets received.

Register Name	Register Type	Description
rx_truncated	Check	This count indicates that the number of packets truncated due to their length exceeding <code>ctl_rx_max_packet_len[14:0]</code> .
rx_undersize	Check	Increment for packets shorter than <code>stat_rx_min_packet_len</code> with good FCS.
rx_unicast	Info	Increment for good unicast packets.
rx_user_pause	Info	Increment for priority-based pause packets with good FCS.
rx_vlan	Info	Increment for good 802.1Q tagged VLAN packets.
tx_total_bytes	Info	Increment for the total number of bytes transmitted by the packet generator.
tx_total_packets	Info	Increment for the total number of packets transmitted by the packet generator.

Test JSON User Guide

Test JSON File Structure

Following is a test `json` file example. Note that the various schemas present in test cases can be listed in any order. Test cases are executed in parallel with exception of:

- **Verify:** executed prior to any tests, as a preliminary check.
- **DMA:** executed prior to all other listed tests (Memory, Power, or GT MAC).

The `.json` file has the following properties:

- If the same Member is repeated throughout your file, only the last value is used.
- Comments are not supported.

```
{
  "device": "xilinx_u280_xdma_201920_3",
  "device_idx": 0,
  "xclbin": "../xbtest.xclbin",
  "xbtest_pfm_def": "../xbtest_pfm_def_u280.json"
  "testcases":
  [
    {
      "type": "dma",
      "parameters":
      {
        "test_sequence": [[10,"DDR","all",256],[10,"HBM",2,256]],
        "output_file": "dma_output_file"
      }
    },
    {
      "type": "power",
      "parameters":
      {
        "test_sequence_mode": "config_duration_toggle",
        "test_sequence": [[60,15],[120,50]],
        "output_file": "power_output_file"
      }
    },
    {
      "type": "memory_ddr",
      "parameters":
      {
        "test_sequence": [{"Alternate_Wr_Rd",60}],
        "output_file": "memory_output_file"
      }
    }
  ]
}
```

```

    }
  },
  {
    "type": "memory_hbm",
    "parameters":
    {
      "test_sequence": [ ["Alternate_Wr_Rd", 60] ],
      "output_file": "memory_output_file"
    }
  },
  {
    "type": "gt_mac",
    "parameters":
    {
      "test_sequence" : [ [2, "conf"], [1, "clr_stat"], [10, "run"],
      "line_rate": "10gbe",
      "output_file": "gt_mac_output_file"
    }
  }
]
}

```

Test JSON Members

Test Environment Members

Table 2: Test Environment Members

Member	Sub-Member	Mandatory/ Optional	Description
device	-	Mandatory	Target platform: For example: xilinx_u280_xdma_201920_3 Use the command <code>xbutil scan</code> to retrieve the target platforms available on your system.
device_idx	-	Optional	If your system contains multiple cards of the same device (see above), this field allows you to select a particular target board. The integer value to provide is the index returned by the Xilinx® Runtime (XRT) command <code>xbutil scan</code> . When not specified, the Application Software searches for all cards compatible with the specified device and selects the first one. This first one found may not be available (if, for example another acceleration program is currently running on it).
xclbin	-	Mandatory	Relative/Absolute path to your <code>xclbin</code> .
xbtest_pfm_def	-	Mandatory	Relative/Absolute path to your Platform Definition <code>json</code> file.
verbosity	-	Optional	Possible values: 0 to 6; default: 0: status, see the following table for descriptions of all levels. This controls the verbosity level of the entire test. This will be applied to any test case which does not have its own verbosity defined. If the verbosity is defined at a certain level, all messages with equal and greater levels will be displayed.

Table 2: Test Environment Members (cont'd)

Member	Sub-Member	Mandatory/ Optional	Description
testcase	-	Optional	List of the test cases to be executed. Note that these are performed in parallel. A failure in a particular test case does not stop other test cases. Define which test is performed. As test cases runs in parallel, you can define 0, 1 or more. When not defined, the verify test case is still performed.
-	type	Mandatory if testcase defined	Possible values: <ul style="list-style-type: none"> • "dma" • "memory_dds" • "memory_hbm" • "power" • "gt_mac"
-	parameters	Mandatory if testcase defined	Test case parameter definition depends on the test case type.

DMA Test Case Members

Following is an example of DMA test case for a platform containing DDR and HBM.

```
{
  "type": "dma",
  "parameters": {
    "test_sequence": [[10, "DDR", "all", 256], [10, "HBM", "all", 256]],
    "output_file": "dma_output_file"
  }
}
```

Where:

```
[10, "DDR", "all", 256]
```

- > buffer size
- > mem_index
- > mem_type
- > duration

The following table shows all members available for this test case. More details are provided for each member in the subsequent sections.

Table 3: DMA Test Case Members

Member	Sub-Member	Mandatory/ Optional	Description
type	-	mandatory	Test case type definition; must be "dma"
parameters	-	mandatory	Test case parameter definition.
-	test_source	optional	Select where the test_sequence originates.

Table 3: DMA Test Case Members (cont'd)

Member	Sub-Member	Mandatory/ Optional	Description
-	test_sequence	mandatory	Describes the sequence of tests to perform. A test is defined by the following values: <ul style="list-style-type: none"> Duration: Test duration in seconds mem_type: Targeted memory type "DDR" or "HBM" mem_index: Targeted memory index buffer_size: Size in MB of buffers to transfer
-	check_bw	optional	Enable bandwidth checking. Disabled by default.
-	output_file	optional	Relative path to the files used to store all DMA measurements.
-	verbosity	optional	Controls the verbosity level of the entire test case.
-	ddr_total_size	optional	Total amount of data (MB) per bandwidth measurement for DDR tests.
-	hbm_total_size	optional	Total amount of data (MB) per bandwidth measurement for HBM tests.
-	hi_thresh_wr_ddr hi_thresh_rd_ddr hi_thresh_wr_hbm hi_thresh_rd_hbm	optional	Overwrite high threshold of the DDR/HBM write/read bandwidth (MB/s).
-	lo_thresh_wr_ddr lo_thresh_rd_ddr lo_thresh_wr_hbm lo_thresh_rd_hbm	optional	Overwrite low threshold of the DDR/HBM write/read bandwidth (MB/s)

test_source

Optional. When not specified, it uses the default value.

- Possible Value: "json" (default), "file".

Select where the `test_sequence` originates:

- When "json" is used, the `test_sequence` (see below) present in the `.json` file is interpreted according the `test_sequence_mode` specified, with each element of the list representing one test.
- When "file" is used, the content of the files provided in the `test_sequence` will be used.

Note: Multiple files can be listed and will be read sequentially.

test_sequence

Mandatory. Describes the sequence of tests to perform. Tests are performed serially, and a failure in one test does not stop the sequence (the next test will be launched).

The following modes are supported:

`json` mode:

In this mode, the `json` file contains the test to perform. This field contains a list of grouped values. The grouped value is interpreted according to `test_source` (see above).

This is a list of values, each separated by a comma: [[], [], []]

The values are interpreted as: [`duration`, `mem_type`, `mem_index`, `buffer_size`].

- `duration`: The duration of the test in seconds: Range [1, 2³²
- `mem_type`: Type of memory to access: -1] "DDR", "HBM"
- `mem_index`: Index of the memory to access: integer or "all"
 - The index must be within the range specified in the platform definition file.
 - "all" is used when all the available memory for the specified types are tested consecutively.

The test fails when the memory to test is not connected in the `xclbin`.

Note: `mem_type` and `mem_index` information can be retrieved using the `xbutil query` and checking the tag associated with each memory.

- `buffer_size`: Size in MB of buffers to transfer.
 - The size must be within the range:
 - [1 MB, 1024 MB] when `mem_type` is DDR
 - -1[1 MB, 256 MB] when `mem_type` is HBM
 - The total size of data per bandwidth measurement is defined in the `json` file for DDR and HBM.
 - For example, if the `ddr_total_size` is 4096 MB and the buffer size is 256 MB, then 16 buffers of 256 MB will be transferred.
 - The total size of data (`ddr_total_size` or `hbm_total_size`) must be a multiple of buffer size.

Example:

- **Single test:** [[50, "DDR", 0, 1024]]
- **Multiple test:** [[50, "DDR", 0, 512], [1, "DDR", 1, 1024], [10, "DDR", 2, 256]]

Note: If, in an XCLBIN, only DDR[0] and DDR[1] are available, then [[5,"DDR",0,256] , [5,"DDR",1,256]] is equivalent to [[5,"DDR","all",256]]

There is no limit to the length of the `test_sequence`.

file mode:

In this mode there is no limit to the number of files you can use. All files must be listed as in `json` mode, as a list of a list.

- Example:
 - Single file: [["../test_cfg/single_file_example.txt"]]
 - Multiple test: [["../test_cfg/test_file_1.txt"] , ["../test_cfg/test_file_2.txt"]]

The content of each file must be *csv-like* with one test configuration per line. Similar to the `json` file, the number must be in decimal format and positive.

The fields are interpreted the same way as the value provided in the `test_sequence` in `json` test source mode.

The following is an example `../test_cfg/test_file_1.txt` syntax:

```
50, "DDR" , 0 , 512
1, "DDR" , 1 , 1024
10, "DDR" , 2 , 256
```

Note:

- No comments are allowed in the `json` file nor in the `csv` file.
- The value of `mem_type` is always quoted in both `json` and `file` test source.

Table 4: Valid/Invalid Syntax

	Invalid Syntax	Valid Syntax
json	[[1,DDR,0,256]]	[[1,"DDR",0,256]]
csv	1, DDR,0,256	1, "DDR",0,256

check_bw

By default, none of bandwidth values (read or write) are checked against any pass/fail criteria; they are just reported as information. By setting this member to `true`, all bandwidth results will be compared to thresholds specified inside the platform definition file. These thresholds can be overwritten (see the following section).

ddr_total_size

Optional: When not specified, the default value is used.

Possible values: Minimum: 1; Maximum: depends on the platform definition file. Default: 4096.

This is the total amount of data (MB) per transfer cycle for DDR tests. This must be a multiple of the `buffer_size` defined in the `test_sequence`.

hbm_total_size

Optional: When not specified, the default value is used.

Possible values: Minimum: 1; Maximum: depends on the platform definition file. Default: 256.

This is the total amount of data (MB) per transfer cycle for HBM tests. This must be a multiple of the `buffer_size` defined in the `test_sequence`.

hi_thresh_wr_ddr, hi_thresh_rd_ddr, hi_thresh_wr_hbm, hi_thresh_rd_hbm

Optional. Overwrite high threshold of the DDR/HBM write/read bandwidth (MB/s) specified in the Platform Definition File.

Range [1, $2^{32}-1$]

After all bandwidth measurements made during the test duration are complete, if the measured bandwidth is greater than this threshold, the test fails.

lo_thresh_wr_ddr, lo_thresh_rd_ddr, lo_thresh_wr_hbm, lo_thresh_rd_hbm

Optional. Overwrite low threshold of the DDR/HBM write/read bandwidth (MB/s) specified in the Platform Definition File.

Range [1, $2^{32}-1$]

After all bandwidth measurements made during the test duration are complete, if the measured bandwidth is lower than this threshold, the test fails.

verbosity

Optional. This controls the verbosity level of the entire test case. This overwrites the Test Environment Members verbosity.

- Possible Values: 0 to 6
- Default Value: verbosity defined in Test Environment Members. See Verbosity Level for descriptions of all possible values.

output_file

Optional. Absolute or Relative path to the files used to store all DMA measurements.



IMPORTANT! If the "-l" command line option is used while calling the Application Software, `output_file` must strictly be the name of a file.

By convention, the `.csv` file extension is appended by the software to the value given by the `output_file` member, and therefore is not specified in the `.json` file.

The values are stored in a `.csv` type format with one column per each information type.

Up to 3 `.csv` files are created:

- `output_file.csv`: this file is always created. It contains the bandwidth measurement done for each of the Write-Read Cycle (See DMA Test Case) for all memory types (DDR and HBM).

Table 5: output_file.csv for a platform containing 2 DDRs and 1 HBM of 32 Pseudo Channels

Tag	buffer size	iteration	Write BW	Read BW
DDR[0]	256	0		
DDR[0]	256	1		
...
DDR[1]	256	0		
DDR[1]	256	1		
...		
HBM[0]	256	0		
...		
HBM[1]	256	0		
...		
...				
HBM[31]	256	0		
HBM[31]	256	...		

- **Tag:** Tested memory name
- **Buffer size:** Size of buffers (MB) transferred during the test
- **Iteration:** Index of the Write-Read Cycle: the quantity of cycle depends on `duration`, `total_size` and `buffer_size`
- **Write BW:** DMA write BW measurement (MB/s)
- **Read BW:** DMA read BW measurement (MB/s)

- `output_file_ddr/hbm.csv`: if a memory type is present in the Platform, its equivalent `.csv` file will be created (and maybe not populated if the memory is not tested). For convenience (for example, to plot figures), the `output_file.csv` values are also sorted per memory tag and grouped into sets of columns. Each set is containing 4 columns:
 - buffer size
 - iteration
 - write bandwidth
 - read bandwidth

Table 6: `output_file_ddr/hbm.csv` for a platform containing 2 DDRs and 1 HBM of 32 Pseudo Channels

DDR[0]:set of columns	DDR[1]:set of columns	HBM[0]:set of columns	HBM[1]:set of columns	HBM[...]:set of columns	HBM[31]:set of columns
values					
	values				
		values			
			values		
				values	
					values

Related Information

[Test JSON File Structure](#)

[DMA Test Case](#)

Memory Test Case Members

Following is an example of DDR and HBM test case. Note that:

- The same output file name can be used for DDR and HBM test, a suffix will be automatically be inserted.
- The memory must be initialized, via `Only_Wr` test, prior to `Only_Rd` test.

```

{
  "type": "memory_ddr",
  "parameters": {
    "test_sequence": [ ["Alternate_Wr_Rd", 60] ],
    "output_file": "memory_output_file"
  }
},
{
  "type": "memory_hbm",

```

> duration
> mem_test_type

```

"parameters" :
{
  "test_sequence": [{"Only_Wr", 1}, {"Only_Rd", 60}],
  "output_file": "memory_output_file"
}
    
```

The following table shows all members available for this testcase. More details are provided for each member in the subsequent sections.

Some Alveo cards have multiple 16 GB of DDR and some variants have HBM. This information is defined in the Platform Definition file. The HBM stack can be split into Pseudo Channels (PCs) (see *AXI High Bandwidth Controller LogiCORE IP Product Guide (PG276)*).

Table 7: Memory Test Case Members

Member	Sub-Member	Mandatory/ Optional	Description
type	-	mandatory	Testcase type definition: Must be "memory_hbm" or "memory_ddr".
parameters	-	mandatory	Test case parameter definition.
-	test_source	optional	Select where the test_sequence originates.
-	test_sequence	mandatory	Describes the sequence of tests to perform.
-	check_bw	optional	Enable the check of the bandwidths. Enabled by default.
-	output_file	optional	Relative path to the files used to store all memory measurements.
-	verbosity	optional	controls the verbosity level of the entire test case.
-	hi_thresh_alt_wr hi_thresh_alt_rd hi_thresh_only_wr hi_thresh_only_rd	optional	Overwrite high threshold of write/read bandwidth (MB/s).
-	lo_thresh_alt_wr lo_thresh_alt_rd lo_thresh_only_wr lo_thresh_only_rd	optional	Overwrite low threshold of the write/read bandwidth (MB/s).
-	error_insertion	optional	Enable preliminary error insertion test. Disabled by default.

test_source

Optional. When not specified, it uses the default value.

- Possible Values: "json" (default), "file".

Select where the test_sequence originates:

- When "json" is used, the test_sequence (see below) present in the .json file is interpreted according the test_sequence_mode specified, with each element of the list representing one test.

- When "file" is used, the content of the files provided in the `test_sequence` will be used.

Note: Multiple files can be listed and will be read sequentially.

test_sequence

Mandatory. Describes the sequence of tests to perform. Tests are performed serially and a failure in one test does not stop the sequence (the next test will be launched).

json mode:

In this mode, the `.json` file contains the test to be performed. This field contains a list of grouped values. The grouped value is interpreted according to `test_sequence_mode` (see above).

This is a list of values, each separated by a comma: [[], [], []]

The values are interpreted as: [mem_test_type, duration]

During the Verify test case, the Application Software automatically detects the number of ports connected to the HBM by checking the `test_sequence` value based on the hardware (For all `xclbin` and the Platform Definition file).

Note: `mem_test_type`, the full range of the memory is written and/or read, meaning that, if required, a test duration may be extended.

- `mem_test_type`
 - "Alternate_Wr_Rd": The memory range is fully written then fully read. These operations are repeated until the duration is reached.
 - Only_Wr: The memory range is fully written. This cycle is repeated until the duration is reached.
 - Only_Rd: The memory range is fully read. This cycle is repeated until the duration is reached.
- `duration`: The test duration in seconds: Range [1, 2³²-1]
- Example:
 - Single test:
 - [["Alternate_Wr_Rd", 60]]
 - [["Only_Wr", 60]]
 - [["Only_Rd", 60]]
 - Multiple test:
 - [["Alternate_Wr_Rd", 60], ["Only_Wr", 60], ["Only_Rd", 60]]

- [{"Only_Wr",1}, {"Only_Rd",60}]
- [{"Alternate_Wr_Rd",1}, {"Only_Rd",60}]

There is no limit to the length of the `test_sequence`.

file mode:

In this mode, there is no limit to the number of files you can use. All files must be listed as in `json` mode, as a list of a list.

- Example:

- Single file: [["../test_cfg/single_file_example.txt"]]
- Multiple test: [["../test_cfg/test_file_1.txt"] , ["../test_cfg/test_file_2.txt"]]

The content of each file must be *csv-like*, with one test configuration per line. As with the `.json` file, the number must be in decimal format and positive.

Following is an example `../test_cfg/test_file_1.txt` syntax:

```
"Alternate_Wr_Rd" , 60
```

```
"Only_Wr" , 60
```

```
"Only_Rd" , 60
```

Note: No comments are allowed in the `json` file, nor in the `csv` file.

check_bw

By default, all bandwidth values (read or write) are checked against pass/failed criteria. By setting this member to *false*, not all bandwidth values will be compared.

Default bandwidth limit is defined by the Platform Definition File but can be overwritten by the user (see below). The default values are displayed at the beginning of each test.

hi_thresh_alt_wr, hi_thresh_alt_rd, hi_thresh_only_wr, hi_thresh_only_rd

Optional: Overwrite high threshold of the write/read bandwidth (MB/s) specified in the Platform Definition File.

Range: [1, 2³²-1]

`hi_thresh_alt_wr` is the write threshold when `mem_test_type` is "Alternate_Wr_Rd"

`hi_thresh_alt_rd` is the read threshold when `mem_test_type` is "Alternate_Wr_Rd"

`hi_thresh_only_wr` is the write threshold when `mem_test_type` is "Only_Wr"

`hi_thresh_only_rd` is the read threshold when `mem_test_type` is "Only_Rd"

After all bandwidth measurements made during the test duration are complete, if the maximum bandwidth is greater than this threshold, the test fails.

`lo_thresh_alt_wr`, `lo_thresh_alt_rd`, `lo_thresh_only_wr`, `lo_thresh_only_rd`

Optional: Overwrite low threshold of the write/read bandwidth (MB/s) specified in the Platform Definition File.

Range: [1, $2^{32}-1$]

`lo_thresh_alt_wr` is the write threshold when `mem_test_type` is "Alternate_Wr_Rd"

`lo_thresh_alt_rd` is the read threshold when `mem_test_type` is "Alternate_Wr_Rd"

`lo_thresh_only_wr` is the write threshold when `mem_test_type` is "Only_Wr"

`lo_thresh_only_rd` is the read threshold when `mem_test_type` is "Only_Rd"

After all bandwidth measurements made during the test duration are complete, if the maximum bandwidth is lower than this threshold, the test fails.

verbosity

Optional. This controls the verbosity level of the entire test case. This overwrites the Test Environment Members verbosity.

- Possible Values: 0 to 6
- Default Value: verbosity defined in Test Environment Members. See Verbosity Level for descriptions of all possible values.

This controls the verbosity level of the entire test case and overwrites the Test Environment Members verbosity.

output_file

Optional. Absolute or Relative path to the files used to store all memory measurements.



IMPORTANT! If the `-l` command line option is used while calling the Application Software, `output_file` must strictly be the name of a file.

By convention, the `.csv` file extension is appended by the software to the value given by the `output_file` member, and therefore is not specified in the `json` file.

The values are stored in `.csv` type format with one column for each information type.



IMPORTANT! Different CSV files are used to store all DDR and HBM memory test results. These are stored with different files extensions. For example, extension `_ddr_0` is used for DDR[0] and `_hbm_2_3` is used for HBM[2:3]. So the same `output_file` name can be used for HBM and DDR test case.

For each memory, two files are written with `_detail.csv` and `_result.csv` extensions.

- The `_details.csv` contains all intermediate bandwidths computed every second, based on information retrieved from the Memory Compute Unit. There is one line of result for every second of each test of the `test_sequence`. For readability, there are columns for live and average value for each `mem_test_type`.

Table 8: `_details.csv` for 3 different `mem_test_type` (2s duration for each test)

test	Alt Wr Bw	Alt Wr Bw live	Alt Rd Bw	Alt Rd Bw live	Only Wr Bw	Only Wr Bw live	Only Rd Bw	Only Rd Bw live
1	4477	4477	12253	12253				
1	8139	11378	8192	4599				
2					15524	15524		
2					15525	15526		
3							17219	17219
3							17220	17221

- **Test:** Test index within the `test_sequence`
- **Alt Wr Bw:** Current average Write BW (MB/s) for “Alternate_Wr_Rd” tests
- **Alt Wr Bw live:** Current live Write BW (MB/s) for “Alternate_Wr_Rd” tests
- **Alt Rd Bw:** Current average Read BW (MB/s) for “Alternate_Wr_Rd” tests
- **Alt Rd Bw live:** Current live Read BW (MB/s) for “Alternate_Wr_Rd” tests
- **Only Wr Bw:** Current average Write BW (MB/s) for “Only_Wr” tests
- **Only Wr Bw live:** Current live Write BW (MB/s) for “Only_Wr” tests
- **Only Rd Bw:** Current average Read BW (MB/s) for “Only_Rd” tests
- **Only Rd Bw live:** Current live Read BW (MB/s) for “Only_Rd” tests

Note: The “live” value are for internal use only of the Application Software.

- The `_results.csv` contains only the final BW computed for each test of the `test_sequence`.

Table 9: `_results.csv` for 3 different `mem_test_type` (2s duration for each test)

Test	Mode	Duration	wr_start_addr	wr_burst_size	wr_num_xfer	rd_start_addr	rd_burst_size	rd_num_xfer	Alt Wr Bw	Alt Rd Bw	Only Wr Bw	Only Rd Bw
1	Alternate_Wr_Rd	2	0	64	268435456	0	64	268435456	8139	8192		
2	Only_Wr	2	0	64	268435456	0	64	268435456			15525	
3	Only_Rd	2	0	64	268435456	0	64	268435456				17220

- **Test:** Test Index
- **Mode:** Test mode
- **Duration:** Duration of the test
- **wr_start_addr:** Starting address for the write transfer.
- **wr_burst_size:** Write burst size: AXI 4 Beat size
- **wr_num_xfer:** Write transfer size: Quantity of AXI transfers to write the entire memory; based on the AXI4 beat size and the memory definition (see Platform Definition)
- **rd_start_addr:** Starting address for the read transfer
- **rd_burst_size:** Read burst size: AXI 4 Beat size
- **rd_num_xfer:** Read transfer size: Quantity of AXI transfers to read the entire memory; based on the AXI4 beat size and the memory definition (see Platform Definition)
- **Alt Wr Bw:** Average Write BW (MB/s) for “Alternate_Wr_Rd” tests
- **Alt Rd Bw:** Average Read BW (MB/s) for “Alternate_Wr_Rd” tests
- **Only Wr Bw:** Average Write BW (MB/s) for “Only_Wr” tests
- **Only Rd Bw:** Average Read BW (MB/s) for “Only_Rd” tests

Related Information

[Memory Test Case](#)

Power Test Case Members

Here is an example of Power test case running for 60 seconds at a toggle rate of 15%

```

{
  "type": "power",
  "parameters":
  {
    "test_sequence_mode": "config_duration_toggle",
    "test_sequence": [[60,15]],
    "output_file": "power_output_file"
  }
}
    
```

> percent
> duration

The following table shows all members available for this test case. More details are provided for each member in the subsequent sections.

Table 10: Power Test Case Members

Member	Sub-Member	Mandatory/Optional	Description
type	-	mandatory	Test case type definition; must be "power".
parameters	-	mandatory	Test case parameter definition
-	test_source	optional	Select where the test_sequence originates.
-	test_sequence	mandatory	Describes the sequence of tests to perform. A test is defined by the following values: <ul style="list-style-type: none"> Duration: In seconds Target: Toggle rate
-	test_sequence_mode	mandatory	Define the Power CU mode. The only supported setting is config_duration_toggle.
-	output_file	optional	Relative path to the files used to store all DMA measurements.
-	verbosity	optional	controls the verbosity level of the entire test case.

test_source

Optional. When not specified, it uses the default value:

- Possible Values: "json" (default), "file"
- Select where the test_sequence originates:
 - When "json" is used, the test_sequence (see below) present in the .json file will be interpreted according to the test_sequence_mode specified with each element of the list is representing one test.

- When "file" is used, the content of the files provided in the `test_sequence` will be used.

Note: Multiple files can be listed and will be read one after the other.

test_sequence_mode

Mandatory. Must be specified.

Possible Values: "config_duration_toggle".

The following describes how to interpret the value provided in the `test_sequence`.

- `config_duration_toggle`: A pair of values respectively corresponding to:
 - Time in seconds: Range [1, $2^{32}-1$]
 - Toggle rate in %; range [0 - 100]

test_sequence

Mandatory. Describes the test to perform. Tests are performed serially and an error in one test does not stop the sequence (the next test will be launched). The test sequence interpretation depends on the `test_sequence` mode (json or file mode, set by `test_source`).

json mode:

In this mode, the Test `json` file contains the test to perform. This field contains a list of tests, each test being defined by a list of parameters : [[], [], []]

The values are interpreted as: [duration, target]

- `duration`: The duration of the test in seconds; range [1, $2^{32}-1$]
- `Target`: Toggle rate; range [0, 100]
- Example:
 - Single test: [[40, 75]]
 - Multiple test: [[40, 15], [240, 30], [120, 40], [20, 50]]

There is no limitation to the length of the `test_sequence`.

file mode:

In this mode, there is no limit to the number of files you can use. All files must be listed as in `json` mode, a list of a list.

- Example:
 - Single file: [["../test_cfg/single_file_example.txt"]]

- **Multiple test:** [["../test_cfg/test_file_1.txt"] , ["../test_cfg/test_file_2.txt"]]

The content of each file must be *csv-like* with one test configuration per line. As per the `.json` file mode, the number must be in decimal format and positive.

For example, `../test_cfg/test_file_1.txt` syntax:

```
40, 15
240, 30
120, 40
20, 50
```

Note: No comments are allowed.

output_file

Optional. Absolute or Relative path to a file used to store all power measurements.



IMPORTANT! If the `-l` command line option is used while calling the Application Software, `output_file` must strictly be the name of a file.

By convention, the `.csv` file extension is appended by the software to the value given by the `output_file` member, and therefore is not specified in the `.json` file.

The values are store in `.csv` type format with one column per *type* of information. All measurements from all `test_sequence` are combined into a single file.

At a minimum, the following values are recorded:

- FPGA Temperature
- Fan RPM: For passive cards, the column contains 0.
- Raw Power: Value measured.
- Toggle Rate.

The `.csv` file also contains

- filtered power, target power, error, error filtered. These columns are reserved for future used and must be ignored
- the detailed measurements for each power rail. The power rails (voltage and current) to monitor are defined in the Platform Definition. The Platform Definition file also contains their respective names which are similar to those found when using the XRT command `"xbutil query"` (see *Getting Started with Alveo Data Center Accelerator Cards (UG1301)*). If the names are not defined a generic naming convention is used: e.g. `voltage[0]`, `current[0]`, `power[0]`. As the Platform Definition is displayed at the beginning of the test (as information with the tag `"XBT_PFM_DEF"`), the link between the index `[0]` and the actual power rail can be done manually

verbosity

Optional. Controls the verbosity level of the entire test case. This overwrites the Test Environment Members verbosity.

- Possible Values: 0 to 6
- Default Value: verbosity defined in Test Environment Members. See Verbose Level for descriptions of all possible values.

Related Information

[Verbose Level](#)

[Platform Definition](#)

[Power Test Case](#)

GT MAC Test Case Members

Following are examples of GT MAC test cases:

GSFP: Electrical / Optical loopback

Note: The default Tx/Rx lane mapping is used with QSFP.

```
{
  "type": "gt_mac",
  "parameters": {
    "test_sequence" : [[1, "conf"], [1, "clr_stat"], [30, "run"],
    [1, "status"]],
    "line_rate": "25gbe",
    "match_tx_rx": true,
    "utilisation": 100,
    "output_file": "gt_mac_out"
  }
}
```

SFP-DD: Electric Cross-Connect with Molex 4 pair 30AWG, 1m, P/N 2049621101

Note: The Tx/Rx lane mapping must be defined.

```
{
  "type": "gt_mac",
  "parameters": {
    "test_sequence" : [[1, "conf"], [1, "clr_stat"], [30, "run"],
    [1, "status"]],
    "line_rate" : "25gbe",
    "match_tx_rx" : true,
    "tx_mapping_0": 2,
    "tx_mapping_1": 3,
  }
}
```

```

        "tx_mapping_2": 0,
        "tx_mapping_3": 1,
        "output_file": "gt_mac_out"
    }
}
    
```

SFP Electric/Optic loopback

Note: Two of the four lanes are disabled.

```

{
  "type": "gt_mac",
  "parameters": {
    "test_sequence" : [[1, "conf"], [1, "clr_stat"], [30, "run"]],
    [1, "status"]],
    "match_tx_rx": true,
    "line_rate"    : "10gbe",
    "active_mac_1": false,
    "active_mac_3": false,
    "output_file" : "gt_mac_out"
  }
}
    
```

The following table shows all members available for this test case. More details are provided for each member in the subsequent sections.

Table 11: GT MAC Test Case Members

Member	Sub-Member	Mandatory/ Optional	Description
type	-	mandatory	Test case type definition; must be "gt_mac"
parameters	-	mandatory	Test case parameter definition
-	test_sequence	mandatory	Describes the sequence of tests to perform. A test is defined by the following values: <ul style="list-style-type: none"> Duration in seconds Command
-	active_mac	optional	Enable/disable packet generator for all lines.
-	active_mac_n (n = 0..3)	optional	Selectively enable/disable packet generator for each line.
-	line_rate	optional	Set line rate: 10 or 25 GbE.
-	utilisation	optional	Transmit utilization (0-100%).
-	set_test_pat	optional	Enable test pattern IEEE 802.3 clause 45.
-	fec_mode	optional	Select the Line FEC mode to be used for both transmitter and receiver.
-	traffic_type	optional	Define the content of the payload area of the packets.
-	packet_cfg	optional	Define the packet length.
-	match_tx_rx	optional	Enable RX and TX packet count match when loopback is present.

Table 11: GT MAC Test Case Members (cont'd)

Member	Sub-Member	Mandatory/ Optional	Description
-	tx_mapping_n (n = 0..3)	optional	Specify the TX line number which will be checked against RX status.
-	gt_tx_diffctrl	optional	Configure the GTY Transceiver TXDIFFCTRL input to all transmitters. See <i>UltraScale Architecture GTY Transceivers User Guide (UG578)</i> for details.
-	gt_tx_pre_emph	optional	Configure the GTY Transceiver TXPRECURSOR input to all transmitters. See <i>UltraScale Architecture GTY Transceivers User Guide (UG578)</i> for details.
-	gt_tx_post_emph	optional	Configure the GTY Transceiver TXPOSTCURSOR input to all transmitters. See <i>UltraScale Architecture GTY Transceivers User Guide (UG578)</i> for details.
-	gt_rx_use_lpm	optional	Configure the GTY Transceiver RXLPMEN input to all transmitters. See <i>UltraScale Architecture GTY Transceivers User Guide (UG578)</i> for details.

test_sequence

Mandatory. Describes the sequence of tests to perform. Tests are performed serially, and a failure in one test does not stop the sequence (the next test will be launched).

Only `json` mode is supported (no external file support), the sequence must be entirely defined with the test `json`.

The `json` file contains the test to perform. This field contains a list of grouped values. The grouped value is interpreted according to `test_source` (see above).

This is a list of values, each separated by a comma: [[], [], []]

The values are interpreted as: [duration, command]

- `duration`: The duration of the test in seconds: range [1,2³²-1].
- `command` (see the following table).

Table 12: test_sequence Commands

Command	Description
<code>conf</code>	[optional] <code>conf</code> applies the settings specified in the configuration parameters to the mac hardware. Part of the configuration process is to issue a reset to the mac hardware, so the <code>conf</code> operation will always result in the Ethernet link dropping and restarting, even if the configuration is identical to the previous test.
<code>clr_stat</code>	[optional] <code>clr_stat</code> is provided to read and clear the mac status registers, but ignore the values returned in the counters. It is intended to be used after a <code>conf</code> operation to clear the status errors caused by the link dropping and restarting.

Table 12: test_sequence Commands (cont'd)

Command	Description
run	[optional] run enables the packet generator Any test sequence entry without a run will disable the packet generator. If the final test sequence entry contains a run then packet generation will continue after execution of xbttest has terminated
status	[optional] status reads the mac status registers, and for any mac instances that have active_mac = true it checks for any counter values that indicate an error has occurred, or the received number of packets indicates a fault on the link. If an error is detected the overall test will be flagged as a fail.

An example of a test_sequence is as follows:

```
"test_sequence" : [ [2, "conf" ] , [10, "clr_stat" ] , [60, "run" ], [1, "stat" ] ]
```

This will:

- Apply the configuration to the MACs, reset them, and wait for two seconds.
- Clear the status registers and wait for 10 seconds.
- Start the packet generators for any active macs, and wait for 60 seconds.
- Read the Status registers and check the results, then clear the status registers, stop the packet generator and wait for one second before exiting.

active_mac

Optional. Possible Values: true or false; default: true

This configuration is applied to all lines of the GT MAC Compute Unit (CU).

When true the packet generator of all mac instances will be enabled, and receiver statistics will be gathered and used to determine the overall pass/fail result of the test.

When false no packets are generated by any mac instance, and all receiver statistics are ignored.

active_mac_n

Optional. Possible Values: true or false; default: true

This configuration can be applied individually to one or more of the four lines connected to the individual transceivers.

When true the packet generator of the selected line will be enabled, and receiver statistics will be gathered and used to determine the overall pass/fail result of the test.

When `false` no packets are generated by the selected line, and receiver statistics from that instance are ignored.

line_rate

Optional. Possible values: `10gbe` or `25gbe`; default: `10gbe`.

This configuration is applied to all lines of the GT MAC CU.

The line rate for all mac cores can be selected as either `10GbE` (10.3125 Gbit/s) or `25GbE` (25.78125 Gbit/s).

utilization

Optional. Possible values: from 0 to 100; default: 50.

This configuration is applied to all lines of the GT MAC CU.

This sets the transmit utilization of the link in the range 0 to 100 (percent). The parameter is used to set the approximate link utilization for the packet generator, by adjusting the delay between packets.

Setting the utilization to 100 causes packets to be generated at the maximum achievable rate.

Setting the utilization to 0 disables the packet generator completely.

set_test_pat

Optional. Possible values: `true` or `false`; default: `false`

This configuration is applied to all lines of the GT MAC CU.

This causes the MAC to generate the test pattern specified in IEEE 802.3 Clause 45 MDIO bit 3.42.3.

Note: If a loopback cable is used to pass this test pattern to the receiver, it will not report any count values.

fec_mode

Optional. Possible values: `none`, `clause_74` or `rs_fec`; default: `none`.

This configuration is applied to all lines of the GT MAC CU.

This parameter selects the Line FEC mode to be used for both transmitter and receiver

- `none`: Disables both FEC options, and uses 66b words with two bit sync headers.
- `clause_74`: Enables the Forward Error Correction mode specified in IEEE 802.3 Clause 74. It may be used for both 10GbE and 25GbE line rates.

- `rs_fec`: Enables the Forward Error Correction mode specified in IEEE 802.3by Clause 91. It may only be used in 25GbE mode. If selected in 10 GbE mode, it will be ignored, and the link will operate in 66b mode (equivalent to `none`).

traffic_type

Optional. Possible values: `0x00`, `0xff`, `count` or `pattern`; default: `count`.

This configuration is applied to all lines of the GT MAC CU.

The test packets produced by the traffic generator consist of a statically configured Destination Address (48 bits), Source Address (48 bits) and Ethertype (16 bits) followed by a payload area and a CRC (32 bits).

The content of the payload area is controlled by this parameter.

If set to `0x00` the whole payload area will be filled with bytes of value `0x00`.

If set to `count` the payload area will be filled with a byte count sequence. The byte following Ethertype will be `0x00`, the next `0x01`, with each successive byte incrementing to `0xFF`, and rolling over to `0x00` and repeating to the end of the payload area.

If set to `pattern`, the payload area will be filled with the pattern (`0x00`, `0x55`, `0xAA`, `0xFF`) repeating for the number of bytes in the payload area.

If set to `0xFF` the whole payload area will be filled with bytes of value `0xFF`.

packet_cfg

Optional. Possible values: `sweep` or value from 64 to 1518 or 9500 to 10000; default: `sweep`.

This configuration is applied to all lines of the GT MAC CU.

This parameter sets the packet length to be generated, or specifies that all packets in the range 64 to 1518 bytes are to be generated in sequence.

If a single numeric value is used, and this is in the range 64 to 1518 or 9500 to 10000 is supplied, then all generated packets shall be this size.

If the value `sweep` is specified, then a sequence of 1455 packets will be generated continuously with sizes between 64 and 1518 bytes.

Note: Note that the receive MTU is adjusted to match the configured transmit packet size:

- If the transmit size is less than or equal to 1518, then the receive MTU is set to 1518.
- If the transmit size is greater than 1518 but less than or equal to 9600, then the receive MTU is set to 9600.
- If the transmit size is greater than 9600 then the receive MTU is set to 10000.

match_tx_rx

Optional. Possible values: `true` or `false`; default: `false`

Specifies if RX is checked against TX.

This configuration is applied to all lines of the GT MAC CU.

If the transmit and receive interfaces of each active mac instance are connected via a loopback cable, then the transmitted packet and byte count may be expected to exactly match the equivalent received good packet and byte counters. Setting this parameter to `true` enables this comparison to be made, and included in the overall pass/fail assessment of the link.

When set to `false` the comparison of Tx and Rx counters is not included in the overall pass/fail assessment of the link.

tx_mapping_n

Optional. Possible values: 0 to 3; default: `n`

Specifies the line number of the *n*th TX which will be checked against RX status.

gt_tx_diffctrl

Optional. Possible values: from 0 to 31; default: 11.

This configuration is applied to all lines of the GT MAC CU.

This parameter sets the TXDIFFCTRL input to all transmitters.

Further details can be found in Chapter 3 of *UltraScale Architecture GTY Transceivers User Guide (UG578)*.

gt_tx_pre_emph

Optional. Possible values: from 0 to 31; default: 0.

This configuration is applied to all lines of the GT MAC CU.

This parameter sets the TXPRECURSOR input to all transmitters.

Further details can be found in Chapter 3 of *UltraScale Architecture GTY Transceivers User Guide (UG578)*.

gt_tx_post_emph

Optional. Possible values: Possible values: from 0 to 31; default: 0

This configuration is applied to all lines of the GT MAC CU.

This parameter sets the TXPOSTCURSOR input to all transmitters.

Further details can be found in Chapter 3 of *UltraScale Architecture GTY Transceivers User Guide (UG578)*.

gt_rx_use_lpm

Optional. Possible values: `false` or `true`; default `false`.

This configuration is applied to all lines of the GT MAC CU.

When `FALSE` this parameter causes the GTY Transceiver to use the DFE Receive equalizer. When `TRUE` it causes the GTY transceiver to use the LPM Receive equalizer.

Further details can be found in Chapter 4 of *UltraScale Architecture GTY Transceivers User Guide (UG578)*.

output_file

Optional. Absolute or Relative path to a file used to store all GT measurements.



IMPORTANT! If the `-l` command line option is used while calling the Application Software, `output_file` must strictly be the name of a file.

By convention, the `.csv` file extension is appended by the software to the value given by the `output_file` member. It is therefore not specified in the `.json` file.

The values are store in `.csv` type format with one column per type of information. All measurements from all `test_sequence` values are combined into a single file.

For each GT MAC CU, one file is generated per line with the suffix/extension.

`"_gt<gt_index>_<line_index>.csv"` with:

- `gt_index` being the index of the GT MAC CU
- `line_index` being the index of the line

each line of the `.csv` file contains:

- Overall result: set to "FAIL" as soon as one test fails, otherwise set to "PASS."
- Overall result: set to "FAIL" if the current test fails, otherwise set to "PASS."
- Values of all status registers (see MAC_STAT Status).

verbosity

Possible Values: 0 to 6.

Default Value: verbosity defined in Test Environment Members. See [Verbosity Level](#) for descriptions of all possible values.

Related Information

[Verbosity Level](#)

[MAC_STAT Status Description](#)

Verbosity Level

If the verbosity is defined to a level, all messages with equal or higher level will be displayed.

Table 13: Verbosity Level

Level	Message Type	Description
0 (default)	STATUS	Message showing intermediate progress of a test; for example, measurements.
1	INFO	General information about the progress of a test; for example, the start of a particular step of the test sequence.
2	WARNING	Message that <i>does not</i> alter the progress of the test. User action may be taken or may be reserved.
3	CRITICAL WARNING	Message that <i>does</i> alter the progress or the results of the test. User action is strongly recommended.
4	PASS	Message returned in the case of successful check.
5	ERROR	Message returned in the case of failed check. This does not interrupt any test cases or tests.
6	FAILURE	The only message level which aborts <i>all</i> test cases/tests. For example, in cases of: <ul style="list-style-type: none"> SW/HW incompatibility json file typo or wrong settings/members Critical FPGA temperature User interruption: CTRL+C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Xilinx Support](#).

Documentation Navigator and Design Hubs

Xilinx[®] Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado[®] IDE, select **Help** → **Documentation and Tutorials**.
- On Windows, select **Start** → **All Programs** → **Xilinx Design Tools** → **DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the [Design Hubs](#) page.

Note: For more information on DocNav, see the [Documentation Navigator](#) page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *Getting Started with Alveo Data Center Accelerator Cards* ([UG1301](#))
2. *10G/25G High Speed Ethernet Subsystem Product Guide* ([PG210](#))
3. *UltraScale Architecture GTY Transceivers User Guide* ([UG578](#))

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright

© Copyright 2019-2020 Xilinx, Inc. Xilinx, the Xilinx logo, Alveo, Artix, Kintex, Spartan, Versal, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.