Bit Error Rate Software Specification



**Notice of Disclaimer**

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Ingenion products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Ingenion 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) Ingenion 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 Ingenion had been advised of the possibility of the same. Ingenion 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 the Limited Warranties; IP cores may be subject to warranty and support terms contained in a license issued to you by Ingenion. Unless otherwise stated, Ingenion 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 Ingenion products in Critical Applications.

© Copyright 2013–2023 Ingenion, Inc. Ingenion, the Ingenion logo, and other designated brands included herein are trademarks of Ingenion in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
| 4/25/19 | 1.1 | Initial draft |
| 12/5/2023 | 1.2 | BER TVS rework |
|  |  |  |

Contents

[Introduction 4](#_Toc519753871)

[Overview 4](#_Toc519753872)

[Applicable Documents 4](#_Toc519753873)

[Class Diagram 5](#_Toc519753874)

[Software Description 5](#_Toc519753875)

[API Control 5](#_Toc519753876)

[GUI 5](#_Toc519753877)

[Log File 5](#_Toc519753878)

[Operation 6](#_Toc519753879)

[Registers 6](#_Toc519753880)

# Introduction

## Overview

This software design is used during testing of TVS functionality. This software communicates with the FPGA design loaded into the TVS daughter cards at time of testing. The software will read testing via a register file to keep track of successful loopback connectivity for each corresponding input and output. All this information will be displayed to the user with a graphical interface, and it will be presented to the user to control TVS testing. After testing has completed, a log file will be filled with status on each input and output tested as well as device information. The software will be able to communicate with any FPGA using Opal Kelly’s FrontPanel API.

## Applicable Documents

The following documents contain background material that may be referenced in this document.

|  |  |  |
| --- | --- | --- |
| Document Number | Date | Title |
| ING-TVS-SPEC-001 | - | TVS User’s Manual |
| BER-TVS-FPGA-SPEC | - | Bit Error Hardware Description |
|  |  |  |

# Class Diagram

A screenshot of a computer

Description automatically generated

Figure 1 Software Block Diagram

# Software Description

## bertvs

This block communicates with the Opal Kelly FPGA. It initializes the FPGA to be used by the program and loads the bertvs bit file allowing access to all input and output registers used in the Opal Kelly Host Interface.

The methods within the bertvs class are as follows:

|  |  |  |
| --- | --- | --- |
| Class Methods | Input | Return |
| connect\_TVS | # of attempts to connect | - |
| disconnect\_TVS | - | - |
| get\_version | - | Version of bertvs bit file |
| get\_device\_info | - | Status code from Front Panel |
| reset\_ber\_test | - | - |
| device\_reset | - | - |
| init\_test\_registers | - | - |
| failsafe\_status | - | List of signals with invalid failsafe |
| read\_test\_registers | - | - |
| print\_offsets | - | - |
| print\_test\_registers | - | - |

**connect\_TVS()**

Connect TVS is used to establish the communication between the bertvs module and the Opal Kelly Host Interface. To connect to the TVS this function attempts to open the device via the USB C port on the XEM board and upload the BER TVS bit file.

**disconnect\_TVS()**

Disconnect TVS will close the bertvs module’s hold on the Opal Kelly Host Interface allowing another program to hold the TVS.

**get\_version()**

Get version reads the version register on the TVS. This register is only instantiated after the bertvs bit file is loaded.

**get\_device\_info()**

Get device info reads the information about the XEM board present within the TVS (product name, device version, serial number, device ID). This information is stored within bertvs instance variables of the same name.

**reset\_ber\_test()**

Reset BER test simply clears the test registers on the TVS as if no loopbacks have been connected.

**device\_reset()**

Device reset issues a reset flag that resets all hardware modules within the TVS.

**init\_test\_registers()**

Init test registers reads and parses the offset register on the TVS and sets up a NumPy array for as many signals are available.

**failsafe\_status()**

Failsafe status reads the failsafe status registers on the TVS and returns a list of all signals that have invalid failsafe statuses on startup.

**read\_test\_registers()**

Read test registers reads all the registers associated with signals on the TVS. This information contains the delay, BER, and bit lock status which is parsed and written into the test register status array generated by init\_test\_registers().

**print\_offsets()**

Print offsets will read, parse, and display the offset register on the TVS. Used for debugging.

**print\_test\_registers()**

Print test registers will read, parse, and display the test status registers while also formatting the values for human readability. Used for debugging. BER is set to a value between 1 and 0. Delay is multiplied by the clock period measured in nanoseconds (ns).

## Bertvs GUI

This block is the graphical interface which the user will interact with both during testing and for test initiation. The GUI will provide a way to start and stop testing, as well as show the status of each loopback test.

The bertvs\_gui file contains a class titled “GUI” in which all functions required for startup are called.

|  |  |  |
| --- | --- | --- |
| Class Methods | Input | Return |
| test\_window | - | The test display window |
| show\_device\_info | - | - |
| show\_fsPopup | - | A popup window object |
| clicked\_start | - | - |
| clicked\_stop | - | - |
| clicked\_connect | - | - |
| clicked\_disconnect | - | - |
| clicked\_file | - | - |
| clicked\_reset | - | - |
| save\_log | - | - |
| open\_info | - | - |
| failsafe\_check | - | - |
| read\_progress | - | - |

**test\_window()**

Test window constructs the window used for displaying test progress.

**show\_device\_info()**

Show device info updates the information shown in the bottom right corner of the control window.

**show\_fsPopup()**

Show failsafe popup constructs the window used to alert the user that one or more signal failsafe is invalid.

**clicked\_start()**

Clicked start is the callback function for the on click event used to start the test. This function will instantiate a new thread that handles running the BER test.

**clicked\_stop()**

Clicked stop is the callback function for the on click event used to stop the test. This function also enters the log information to a temporary file.

**clicked\_connect()**

Clicked connect is the callback function for the on click event used to connect the GUI to the Opal Kelly Host Interface. This will take control over the TVS, and no other connections can be made at the same time.

**clicked\_disconnect()**

Clicked disconnect is the callback function for the on click event used to disconnect the GUI from Opal Kelly’s Host Interface. After this function is called the TVS can be used by other programs.

**clicked\_file()**

Clicked file is the callback function for the on click event used to open the most recent log file.

**clicked\_reset()**

Clicked file is the callback function for the on click event used to reset the TVS test results and the log file.

**save\_log()**

Save log permanently saves the file “TVS\_Current\_Log.txt” if it is present. The name of that file will be in this format: “TVS\_xxxx\_mmddyy\_hhmm\_pass.txt”.

**open\_info()**

Open info creates a window displaying basic help and the credits for this program.

**failsafe\_check()**

Failsafe check reads the failsafe status registers and determines if any failsafe is invalid. If a failsafe is invalid show\_fsPopup() is executed to alert the user.

**read\_progress()**

Read progress is the function that reads all test status registers. When a test is complete it will update the test window’s progress for each respective connector.

## Log File

This block will store all test data from each input and output signal pair. Each loopback connection will be stored and logged with its BER (bit error rate), . Test information like TVS serial number, hardware version and FPGA information will be logged as well.

The Log file has these functions:

|  |  |  |
| --- | --- | --- |
| Class Methods | Input | Return |
| log\_info() | deviceInfo, entryU, entryS | - |
| log\_note() | tvsNotes | - |
| log\_test() | status, test\_state, pass\_fail | - |
| log\_end() | time | - |
| log\_failsafe\_violations() | List of low failsafes | - |
| log\_reset() | - | - |
| enter\_log() | Serial number and time | - |
| delay\_to\_type() | Delay in clock cycles | I/O standard detected |
| get\_log\_directory() | - | Path to log directory |

**log\_info()**

Log info generates the header for each BER test. This includes the test conductor’s name, the TVS serial number, and XEM boar information.

**log\_note()**

Log note adds any additional notes in the notes section of the log file.

**log\_test()**

Log test records the test status for all signals with test registers. This information includes the name of the signal pair, the pass/fail state, and the detected I/O standard. Additionally, if the test fails, log test will also add the debug string to the right of each fails signal pair.

**log\_end()**

Log end concludes the log file with a times stamp and break line.

**log\_failsafe\_violations()**

Log failsafe violations add the invalid failsafe and debug information to the beginning of the log file.

**log\_reset()**

Log reset overwrites the log file.

**enter\_log()**

Enter log permanently saves the log file in this format: “TVS\_xxxx\_mmddyy\_hhmm\_pass.txt”.

**delay\_to\_type()**

Delay to type interprets the TVS delay value defined in nanoseconds as the likely supported I/O standard.

**get\_log\_directory()**

Get log directory simply gets the directory where all the log files will be placed.

## Signal Map

The signal map module is used to read the signal map file into a Pandas data frame. This data frame object is then used to sort and list the connectors and signals the TVS has.

|  |  |  |
| --- | --- | --- |
| Class Methods | Input | Return |
| load\_signal\_map() | A file | The signals map dataframe |
| signals\_per\_connector() | A pin map | Number of signals |
| get\_signals\_per\_connector() | - | Number of signals |
| get\_connector\_list() | - | List of connectors |
| get\_debug\_info() | A signal | Debug info string |

**load\_signal\_map()**

Load signal map loads the file containing all information about the TVS signals pairs into a Pandas Dataframe.

**signals\_per\_connector()**

Signals per connector is a function that reads a Pandas Dataframe containing signal pair information.

**get\_signals\_per\_connector()**

Get signals per connector retrieves the signal\_map’s Dataframe attribute and reads all signals into a list.

**get\_connector\_list()**

Get connector list gets all the unique connector names present in the signal map and returns a list of them in order.

**get\_debug\_info()**

Get debug info retrieves the debug string from the signal map.

# Operation

The user will open or navigate to the folder containing the ber-tvs project. This folder should be the first opened after downloading and extracting the most recent ber-tvs release on git. Once the user has made their way to the proper directory, they should only need to run “python /py/bertvs\_gui.py”. Alternatively, the folder can be opened in VS Code and run there.

Once run, the user will be prompted via an on-screen graphical interface. As shown below, this GUI has several input fields and buttons used to control the Opal Kelly Host Interface.

A screenshot of a test results

Description automatically generated

Initially, when the TVS is connected and powered on the window will populate with the information retrieved from the device in the bottom left corner. If the TVS is not connected, the user will be prompted to check the power and connection to the TVS.

Additionally, the test window should appear alongside the control window. The test window will be in an idle state until a BER test is initiated.

A close-up of a number

Description automatically generated

Before starting the BER test, the user should enter both their name and the device serial number into their respective boxes. Note that these entries are not required, but they are used in the log file and will be instrumental for keeping track of test results.

To start the BER test, simply press the start button.

Once the test has started the button should remain in a held state until pressed again, stopping the BER test. The test window should also become active, reading 0% and filling lights red for all connectors on the test window.

Begin testing by plugging in loopbacks until progress is read back on the test window.

If any tests do not state completion on the test window, despite loopbacks being plugged in this may indicate that a loopback error has occurred.

Once loopback testing is complete simply press Stop Test.

The control window will immediately tell the user if the test was successfully completed or if it failed. It will also save the results to a file called “TVS\_Current\_Log.txt”. To view the file the user can either press Open File or navigate to the directory “/py/TVS\_Logs/ “ where all the log files are stored. All log files that are permanently saved will contain the serial number, pass/fail, date, and time “xxxx\_mmddyy\_hhmm\_pass”.

The Save Log button can be pressed to save the log file permanently.

If an issue was detected during testing or a test was never performed on a connector, the log file will state the pass/fail and what I/O standard was detected. If the test says fail or no test a debug string will be displayed to the right of that signal detailing the path and all the points most susceptible failure.

To perform a second test with a clean start, press the Reset button.

# Registers

See BER-TVS-FPGA-SPEC register section for more detail.

|  |  |  |  |
| --- | --- | --- | --- |
| Address | R/W | Mnemonic | Description |
| 0x000 | RO | Version | Version of FPGA design |
| 0x001 | RO | Offset | Contains BER TVS address offsets |
| 0x002 | W | Command 1 | Used to control the BER test externally if requested |
| 0x003 | W | Command 2 | ^^^ |
| 0x004 – 0x007 | RO | Failsafe Status | Contains the failsafe statuses for signals 31:0 |
| 0x008 – 0x029 | RO | RS422In[0:39] | RS422 BER |
| 0x030 – 0x033 | RO | UartIn[0:3] | Uart BER |
| 0x034 – 0x03F | RO | LVDSIn[0:11] | LVDS BER |
| 0x040 – 0x053 | RO | XTTLOut[0:19] | XTTL BER |