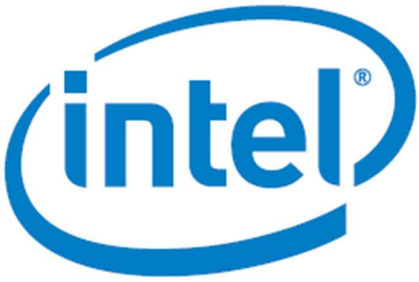


H-Tile On-Die Instrumentation through Avalon Memory Mapped Access

Date:September 24, 2019

Revision: 1

Contents

1.0 Introduction

This application note describes how to use the Stratix 10 L/H-Tile transceiver's On-die Instrumentation (ODI) feature through a microcontroller or processor. The ODI allows users to capture a 2-D eye diagram of the received signal after the RX CTLE and DFE to analyze the eye width and height. There is hard logic to sweep through horizontal and vertical points in the eye and determine whether the data at the point matches the recovered data. This allows users to determine the eye opening. This helps users at the system bringup stage so they can evaluate the optimal TX and RX settings.

Users can access the hard logic through the transceiver's Avalon memory mapped (AVMM) registers. There are three methods to access the registers:

- Transceiver toolkit through Quartus requires users to have enabled the ADME
    - Advantage: built-in functions allow users to sweep settings and capture the eye for each setting
    - Disadvantage: speed limited by USB blaster. Requires Quartus, so not field deployable
- RTL state machine using steps in KDB https://www.intel.com/content/altera-www/global/en_us/index/support/support-resources/knowledge-base/hsio/2018/have-there-been-any-updates-to-the-steps-required-to-enable-and-.html
    - Advantage: fast and users can customize the eye sweeping
    - Disadvantage: debugging complex and compilation takes time. Need to meet timing
- Processor (either on the FPGA or external)
    - Advantage: Flexible and scalable.
    - Disadvantage: requires memory to store the function

This app note will focus on the last method. An example design containing an embedded NIOS processor and 64 GXT transceiver channels is included. The NIOS contains a command line interface that allows users to capture the eye diagram for any channel in the design. The C function to access the AVMM registers, capture the eye, and display the eye diagram is included and users can integrate it into their design and software architecture.

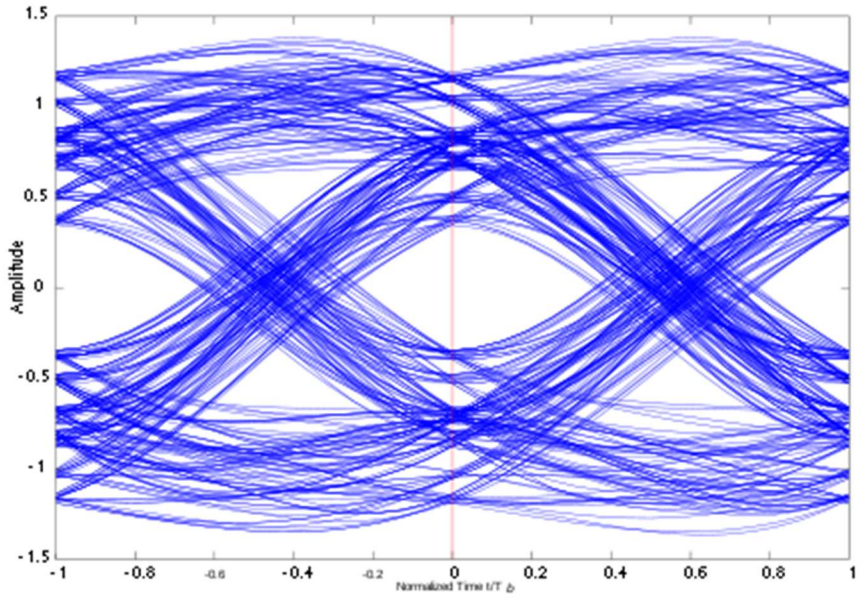
Native PHY user guide:
https://www.intel.com/content/www/us/en/programmable/documentation/wry1479165198810.html

TTK: https://www.intel.com/content/www/us/en/programmable/products/design-software/fpga-design/quartus-prime/features/swf-transceiver-toolkit.html

2.0 Concept

A serial data can become degraded across a link and the signal can become closed. The receiver's CTLE and DFE can help open the signal. The ODI block allows users to see how much the eye has been opened. If the recovered data (the serial data is sampled at the center of the eye) is error free, then sweeping the horizontal phase and vertical voltage and XORing the data with the error free recovered data will give the eye-opening width and height.

The ODI block works on any data pattern (PRBS or user) as long as there is sufficient data transitions (0->1, 1->0, 0->0, 1->1). The ODI block is non-destructive and users do not need to stop their user logic when running ODI.



If users have a microcontroller or microprocessor (either located on the Stratix 10 FPGA or external to the device), they can call a C function that will interface with the transceiver's AVMM bus to collect the eye information and display it.

3.0 Implementation details

If the user has implemented the design so the processor has access to N Native PHY instances and each instance has M channels with shared reconfiguration interfaces, users can use the C function below to capture the eye on any channel in any Native PHY instance.

The C function has the following definition:

void do_eye_measurement_top (int phy, int SelectedChannel, int Tile, int Runtime, int Bandwidth, int verbose, int background_cal);

Parameter	Range	Description
Phy	N-1 to 0	The number corresponds to the Native PHY instance
SelectedChannel	M-1 to 0	The number corresponds to a channel in the selected instance
Tile	0 to 3	L Tile and H Tile have different PMA analog features that affect the ODI result.

		0: for H-Tile production device 1: for H-Tile ES2 device 2: for L-Tile device 3: for H-Tile ES device
Runtime	0 to 6	Users can select the ODI dwell time. 0: 2^16 bits 1: 1,000,000 bits 2: 10,000,000 bits 3: 100,000,000 bits 4: 300,000,000 bits 5: 1,000,000,000 bits 6: 4,000,000,000 bits The total number of bits tested at each point is approximately equal to 4 * the runtime value when the DFE is not enabled and 8 * the runtime value when the DFE is enabled.
Bandwidth	0 to 3	In order for the ODI to generate accurate clock phases, users must provide the datarate If the device is H-Tile production: 0: > 25Gbps 1: 16Gbps to 25Gbps 2: 10Gbps to 16Gbps 3: < 10Gbps 4: not supported Otherwise: 0: > 20Gbps 1: 12.5Gbps to 20Gbps 2: 6.5Gbps to 12.5Gbps 3: < 6.5Gbps
background_cal		On H-Tile production devices, background calibration is recommended for datarates > 17.4Gbps in order to improve transceiver performance. 0: background calibration is not used 1: background calibration is used. The function will disable background calibration before running ODI and will enable background calibration afterwards Background calibration is supported in Quartus 18.1

Note: The do_eye_measurements_top function will call other PMA functions and also functions to access the AVMM.

Users need to specify how the PHYs map to the processor's memory space.

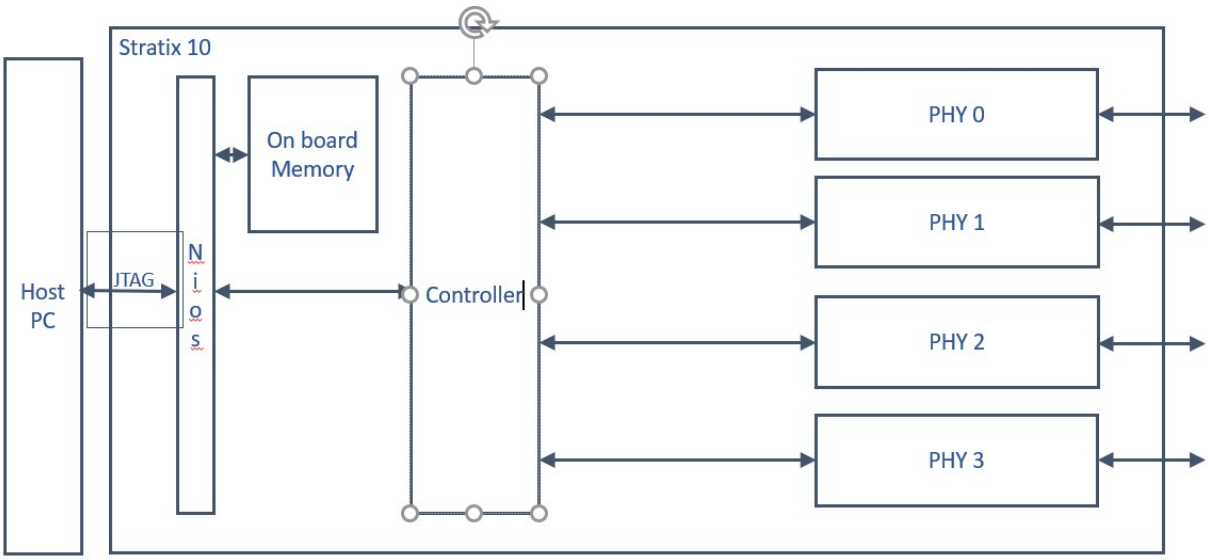
Users need to create bridge logic so the processor can access the AVMM registers as though it's a memory location.

The do_eye_measurement_top function defines two 130x130 2-dimensional floating point arrays to store the number of error bits and the number of bits tested.

4.0 Example design

An example design has been provided with this app note. It contains 64 transceiver channels running at 25.78Gbps and targets the Stratix 10 SX dev kit. The target device is 1SX280HU2F50E2VG.

The design contains the following blocks:



NIOS: embedded soft processor in the FPGA core

Onboard memory: 512K memory in the FPGA core to hold the NIOS's program code and data

PHY0 to 3: Native PHY instances, each containing 16 GXT transceiver channels

Controller: platform designer logic to allow NIOS to access the PHYs' AVMM register space and the onboard memory. Users specify the address space here.

Users interact with the Nios using a host PC through the Jtag connection.

Not shown: ATXPLLs acting as the TX PLLs for the transceivers.

Users can create their own memory mapping logic through knowledge of the AVMM interface and their microprocessor instead of using platform design to create the glue logic.

More information about the NIOS soft processor and how to use platform designer can be found at:

NIOS: https://www.intel.com/content/www/us/en/products/programmable/processor/nios-ii.html

Platform designer: https://www.intel.com/content/www/us/en/programmable/products/design-software/fpga-design/quartus-prime/features/qts-platform-designer.html

AVMM:
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

Test results

A FMC loopback card has been attached to the SX dev kit.

The following test has been performed:

1) Open the design using Quartus 18.1.1
2) Compile the design
3) Download the design using Quartus 18.1.1 Programmer
4) Open the NIOS command shell
5) cd to the directory devkit_demo_18_1_1_261_restored/software/app/
6) Type “jtagconfig” in NIOS command shell. The dev kit board name will be listed.
    a. “Stratix 10L SoC Dev Kit [USB-1]” in this case
7) Open run1.bsh and change the board name to the one in the previous step
8) cd devkit_demo_18_1_1_261_restored/software/app/Stratix10GX_x_Ch. If the C code has not been compiled before, type “./create-this-app” in NIOS command shell
    a. If you make changes to the C code, type “./make” in NIOS command shell

9) Type "./run1.bsh" in the NIOS command shell to download the NIOS firmware
   a. The firmware will initialize the Native PHY instances, start the PRBS generator and checker and will show the main options window as well as the current transceiver status

```
PHY  0  Located in Tile Top Right (TR) (Tile6)
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
Channel           :| 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|15|
                   |--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
PrbsPattern       :|31|31|31|31|31|31|31|31|31|31|31|31|31|31|31|31|
LockToRef         :| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
Serial Loop       :| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
Rev. Serial Loop  :|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  P=POSTCDR,R=PRECDR
CTLE,VGA Mode     :| A| A| A| A| A| A| A| A| A| A| A| A| A| A| A| A|
DFE Adaptation Mode:| A| A| A| A| A| A| A| A| A| A| A| A| A| A| A| A|  -=OFF,1=1TAP,A=ALL TAPS Adaptive
Channel Type      :| C| C| C| C| C| C| C| C| C| C| C| C| C| C| C| C|  C=CHIP2CHIP,B=BACKPLANE,S=SHORT,D=DAC,O=OPTICAL
Connection Type   :| F| F| F| F| F| F| F| F| F| F| F| F| F| F| F| F|  F=FMC,Q=QSFP28,B=BKPLANE,S=SMA,L=LPBK,M=MXP
Rx Polarity Invert :|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
Tx Polarity Invert :|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
PLLLocked         :| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1| 1|
FreqLocked        :| 1| 1| 1| 0| 1| 1| 0| 1| 1| 1| 1| 1| 1| 0| 1| 1|
PrbsLocked        :| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0|
Errorcount        :|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Test Time PHY 0          :  0h  3m 21s
Tx Clkout Frequency      :  201414 kHz
Recovered Clock Frequency :  139011 kHz
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------+
```

```
Select Action :
===============
1. Reset all transceiver Phys and restart PRBS generation/checking
2. Reset all transceivers in Selected Phy
3. Select Channel To Control in Selected Phy
4. Reset PrbsChecker on all Channels in the Selected Phy
5. Select Phy to control
6. Reset ErrorCounter on Selected Channel in Selected Phy
7. Show Status
8. Enable/Disable Serial loopback in Selected Phy
9. Enable/Disable Reverse Serial Loopback (post-CDR) in Selected Phy
0. Enable/Disable Reverse Serial Loopback (pre-CDR) in Selected Phy
A. Select Prbs Pattern on Selected Channel
B. Show detailed Errorcount
C. Change adaptation mode on the Selected Phy
E. Show Transceiver PMA Settings on all channels
G. Input new ODI Runtime
P. Recalibrate ATX PLL in Selected Phy
R. Recalibrate Selected channel in Selected Phy
S. Store all channel information in memory of Selected Channel
T. Store and compare with values stored in process S
Z. Dump channel content of selected channel
O. Perform 2D+1D Eye Measurement on selected channel
U. Sweep PMA settings and program best setting on selected channel
Y. Measure and dump 2D eye on all channels
To stop the test press CTRL-C (this will automatically create output.log file

  Enter Choice :5
```

   b. Enter "5" to change the PHY to control. The PHY associated with the FMC loopback card is #2

```
5. Select Selected Phy
Current Phy : 0
New Phy (0- 3)         : 2
 New phy 2 :
```

c. The firmware default channel is 0, which is connected to the FMC loopback card. Users can enter “3” to select a different channel

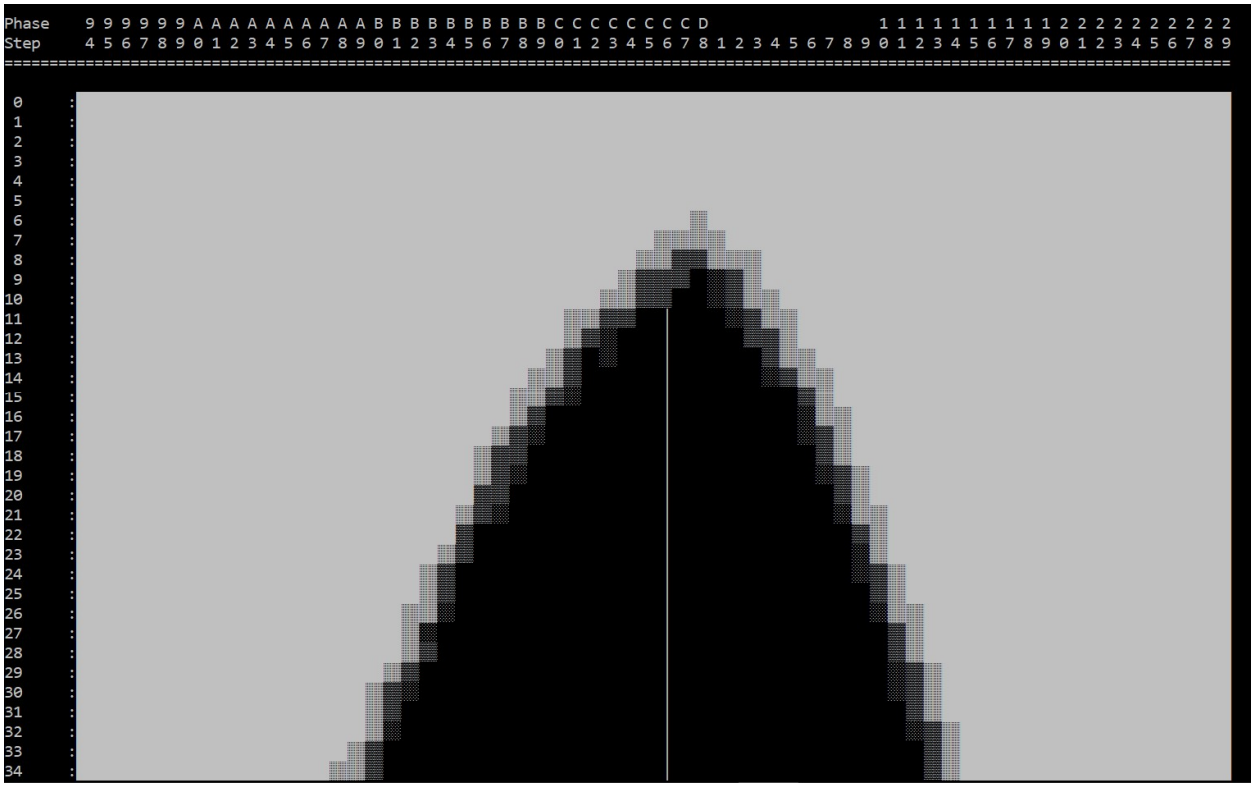
d. The firmware default runtime is for 2^16 bits. Users can enter “G” to change the runtime.

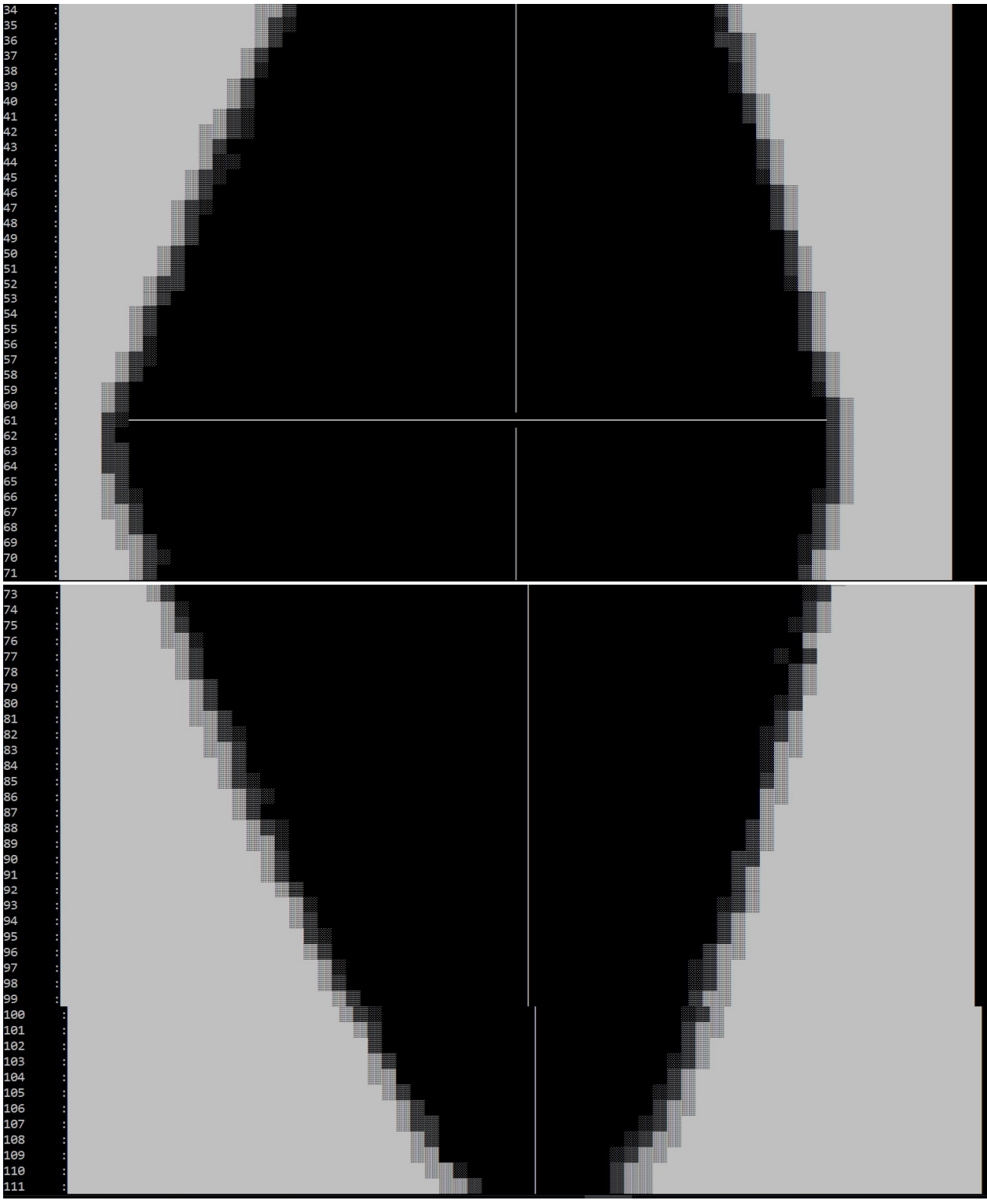
e. The firmware default RX adaptation mode is adaptive with all taps DFE. Users can enter “C” to change the adaptation mode
Users can enter “O” to capture the eye. Here is the eye with on channel 0 of phy 2 with 2^16 bits and adaptive with all taps DFE:
The routine will scan the eye and first report eye statistics:

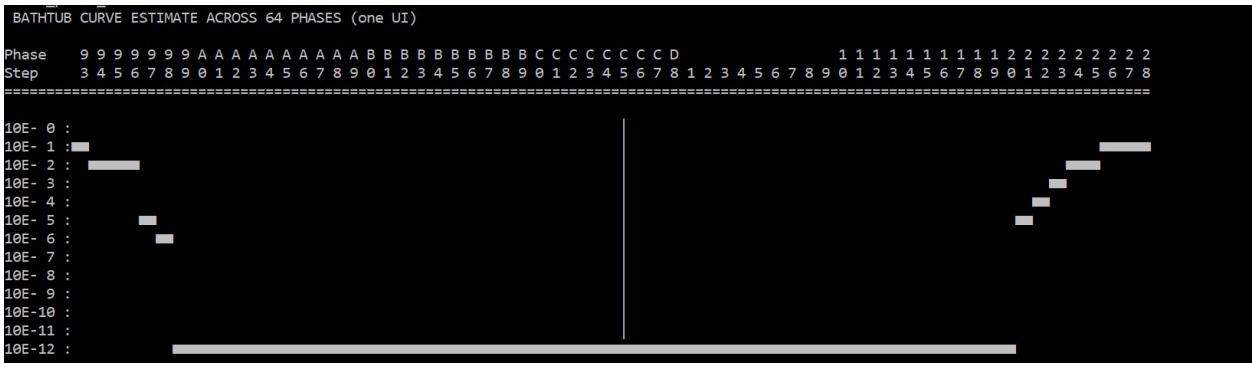
```
  Enter Choice :o

Measuring 2D and 1D Eye  ....

After scanning horizontal openining
left_phase = 99
right_phase = 23
eye_top =  9
eye_bot = 114
area = 3216
```

There are 64 phase steps in 1 UI.
Next, it will report the 2D eye

Followed by the bathtub curve at 0 vertical phase:







(These are the major sections to include)

- Table of Contents
- Theory of Operation
- How to download the completed design to development kit and what behavior to expect: lights, display, text on Eclipse console, etc. Reference which Quartus version the design example is constructed with.
- How to reconstruct the design (eg open Qsys, generate HDL files, compile in Quartus, build SW in Eclipse, etc)

(save your .doc Word document to .pdf and post on the alterawiki. Create a link from your Design Store design example posting to the wiki document).

Author	Comments	Revision Date
Leon Zheng	•	9/30/2019