GE2/1 Optohybrid Board Version 2

Testing Manual

Rice University

13 July 2020

Introduction

The GE2/1 Optohybrid Version 2 (OH2) board provides readout and trigger interfaces for 12 VFAT3 ASICs residing on the GEB board. A block diagram of the OH board and top and bottom views of the board are shown on Fig.1-3 respectively. Board specification, schematics, various configuration files, presentations and other documentation related to VFAT3, GEB, custom electronics parts can be found at

https://twiki.cern.ch/twiki/bin/view/CMS/GE21OHBoardTwiki

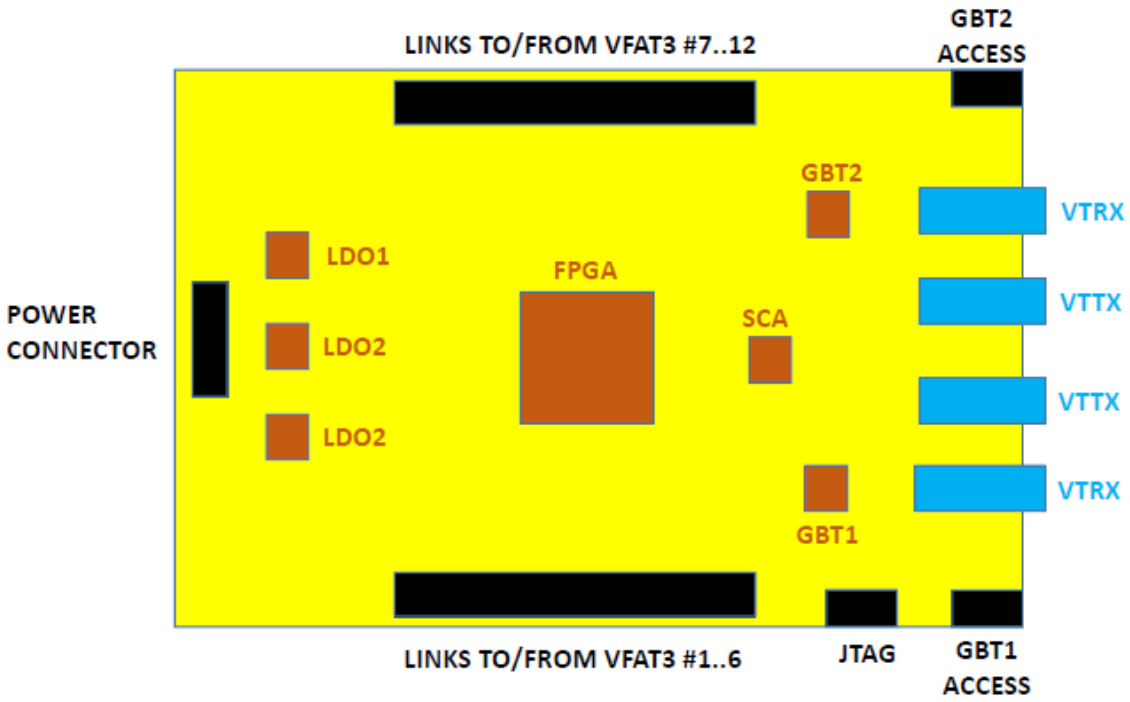


Figure 1: Block diagram of the OH board

Note that in these manual and on the printed circuit board the GBT ASICs are numbered as GBT1 (U2) and GBT2 (U3) while in software they are numbered as GBT0 and GBT1 respectively.

Specific important acceptance criteria are marked green in this manual and must be met.

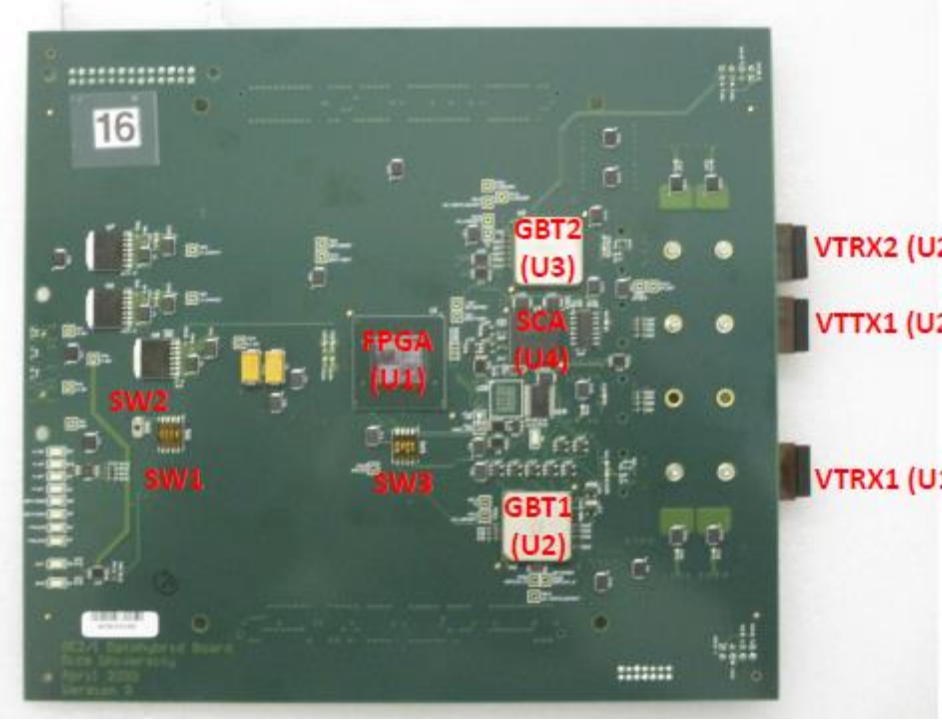


Figure 2: Top side of the OH board

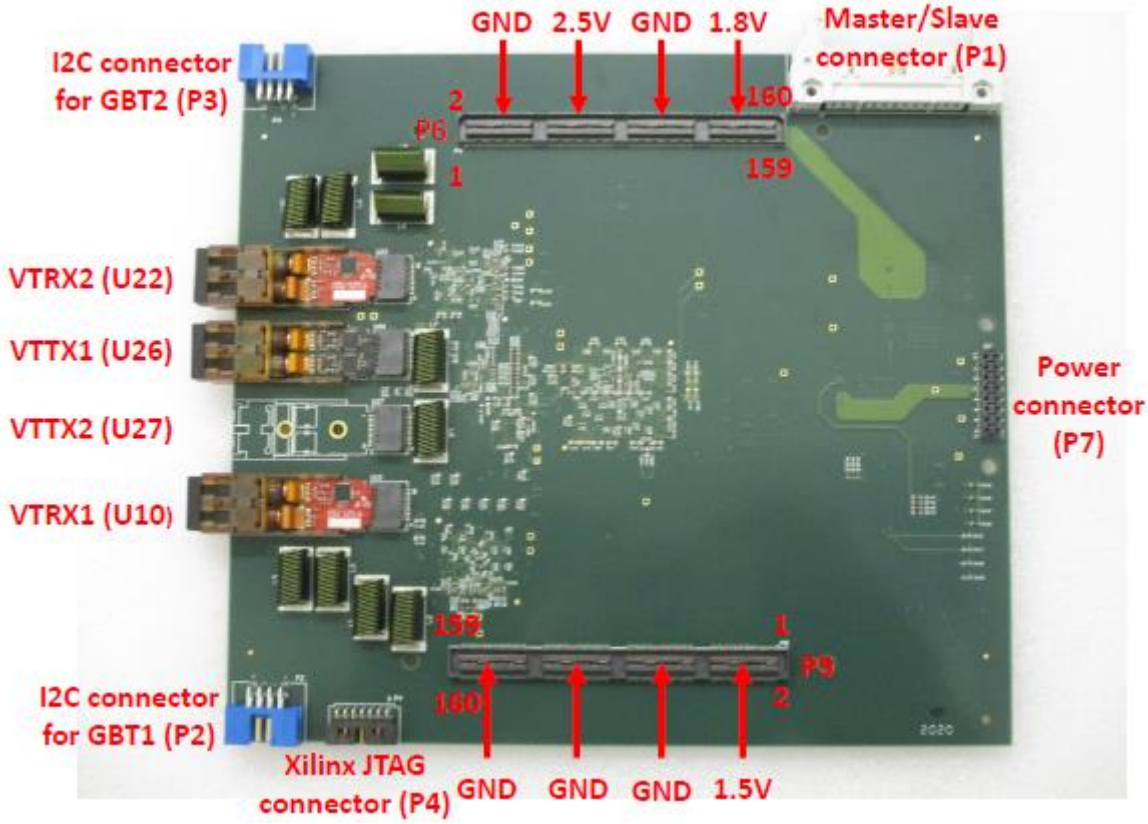


Figure 3: Bottom side of the OH board

The following documentation should be studied in detail before running tests:

Main source: https://twiki.cern.ch/twiki/bin/viewauth/CMS/GE21OHBoardTwiki

[1] OHv2 specification
https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/OH2_spec_030420.pdf

[2] OHv2 schematics https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/OHv3_042020_sch.PDF

[3] OH-GEB interface specification
https://twiki.cern.ch/twiki/pub/CMS/GE21GEBBoardTwiki/GE21_GEB_Interfaces_Specification_v2.pdf

Specific components

[4] GBTx ASIC Manual https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/gbtxManual.pdf

[5] SCA ASIC Manual https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/GBT-SCA-UserManual.pdf

[6] GBT Programming tools https://espace.cern.ch/GBT-Project/VLDB/Control/Forms/AllItems.aspx

[7] USB-I2C Dongle Manual https://espace.cern.ch/GBT-Project/VLDB/Manuals/Dongle%20Hardware%20manual.pdf

[8] VFAT3 presentation
https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/VFAT3_presentation.pdf

[9] VFAT3 Manual
https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/VFAT3_User_Manual_v2.3.pdf

[10] VFAT3 communication port
https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/VFAT3_comm_port.pdf

[11] Xilinx Artix-7 programming guide
https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

[12] Calorimeter Trigger Processor CTP7 https://pages.hep.wisc.edu/~pamc/pspdf/Klabbers_Layer1.pdf

[13] FlexPCB and PlugIn card https://twiki.cern.ch/twiki/bin/view/CMS/GE21FlexPCBPlugInCardsTwiki

[14] VTTX and VTRX optical parts https://cms-docdb.cern.ch/cgi-bin/PublicDocDB/ShowDocument?docid=4802

1. Visual Inspection

1.1. Make sure all parts are installed on both sides of the PCB with the exception of: U25, R67, R89, R50.
1.2. Make sure the active parts are correctly oriented (check for pin 1)
1.3. Make sure polar capacitors are installed properly
1.4. Make sure temperature sensors R151-R154 are mounted face up

2. Check OH board for shorts and GEB powers

Measure the resistance between GND point and six power points (2.5V, 1.8V, 1.5V, 1.0V_AVCC, 1.2V, 1.0V_VCCINT). Typically, it varies from 60 Ohm to 150 Ohms.

Make sure the GEB provides 2.5V, 1.8V and 1.5V powers to P7 connector (Fig.4).

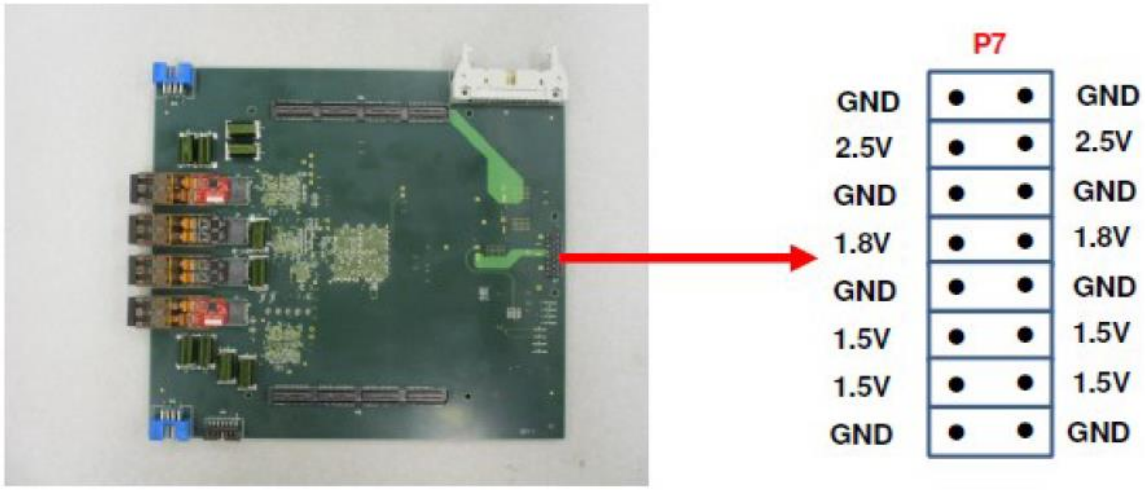


Figure 4: Pin assignment of the P7 power connector

3. Set jumpers for initial debugging

3.1 The EPROM (U25) is not needed, so make sure the jumper between TP18 and TP21 is installed (to close the JTAG loop)

3.2. Select SW1-1/2/3=”110” for Slave SelectMAP configuration mode (FPGA is loaded from GBT1) and SW1-4=”0” (Master/Slave=0) (Fig.5).

3.3. Set SW3-1/2/3/4 (Fig.5) to:
Select Enable GBT =1 (SW3-1)
Select TESTA= 1 (Enable JTAG access to FPGA from SCA ASIC, SW3-2)
Select CCLK to the FPGA from GBT path (SW3-3)
Select CONFIGSEL=1 to allow fusing of both GBT ASICs from the dongle (SW3-4).

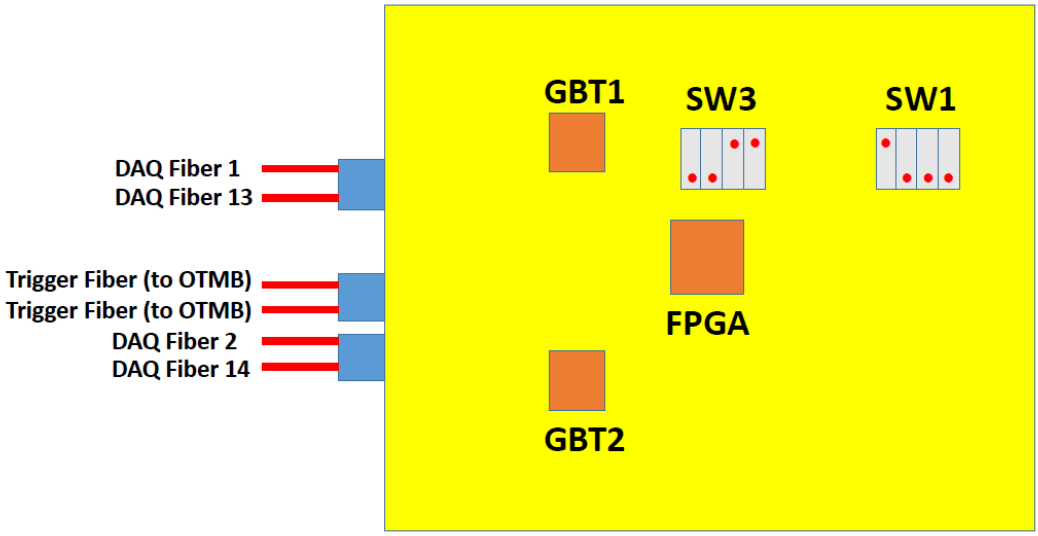


Figure 5: Switches on OH0 board for debugging, top view

4. Install VTTX and VTRX optical parts

4.1. Install two VTRX optical transceivers U10 and U22 (red boards)
4.2. Install one VTTX1 optical transmitter U26 (blue board)
4.3. Attach all three optical parts with screws (6 total)

5. Install OH board under test on the GEB board and attach it with 4 screws

Carefully install OH board on the GEB board; make sure it's fully plugged into 3 connectors. Power up the GEB board. **Measure 6 powers on the OH board with a multimeter in 6 points (2.5V, 1.8V, 1.5V, 1.0V_AVCC, 1.2V, 1.0V_VCCINT) and make sure the voltage tolerance is within 5% of the nominal values**. Make sure corresponding green LEDs D3, D4, D10, D11 are "on".

6. Attach optical fibers

Attach DAQ optical fibers 1 and 13 to VTRX1 U10 (GBT1) and fibers 2 and 14 to VTRX2 U22 (GBT2), see Fig.1. These connections are for OH0. For OH1 (if connected), use fibers 2 and 15 for U10 and fibers 4 and 16 for U22.

7. Load GBT configuration files into both GBT ASICs via I2C dongle

7.1. Connect dongle cable to PC to GBT1 (blue connector P2)
7.2. Start configuration software
7.3. Import configuration file (version 01/17/2020 for GBT1)
https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/GBTX_GE21_OHv2_GBT_0_minimal_2020-01-17.txt

7.4. Program chosen file (push “write” and “read” in the programmer GUI), make sure the link status changes to “idle”; close programming window.
7.5. Disconnect dongle from GBT1 and connect to GBT2 (blue connector P3)
7.6. Start configuration GUI
7.7. Import configuration file (version 01/31/2020 for GBT2) https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/GBTX_GE21_OHv2_GBT_1_minimal_2020-01-31.txt
7.8. Program chosen file (push “write” and “read” in the programmer GUI), make sure the link status changes to “idle”; note this is a “wide-bus” mode; close programming window.

8. Check communication with the CTP7

CTP7 has a command line utility that lets you read and write registers, here's how to start it:

1) connect to the CTP7:

```
ssh texas@eagle42
```

2) Run this command to open a register access interface

```
reg
```

When you have the register interface running, you can type help to get the list of supported commands, or help to get more information about a particular command. The only commands that you'll need for now are: * `read` : this reads the value of the given register * `readKW` : this reads all registers that match the pattern (no wildcards, just give a substring like OH_LINKS, and it will read all registers that have that substring in their name) * `write` : this writes a given value to the given register.

Note when the communication with the CTP7 is established, both GBT1TXRDY (D8) and GBT2TXRDY (D9) green LEDs are permanently on.

Resetting the AMC13

Sometimes the AMC13 needs to be restarted and the clock needs to be reset.

A script in the gem2 riceuser's bin directory can be used for this. Connect to the AMC13

```
cd /bin/

amc13
```

Enables the second slot for clock and TTC distribution and enable the AMC13 TTC loopback mode.

```
en 2 t
```

After the clock is stable, the CTP7 firmware should be reloaded (see below).

Login to the CTP7

```
ssh texas@eagle42
```

Load the CTP7 firmware

```
cd ~/tamu && cold_boot_invert_rx.sh
```

Load the OH firmware to the CTP7 RAM

```
cd ~/tamu

gemloader_configure.sh #OHv1

gemloader_configure_v2.sh #OHv2 with PRBS test support

gemloader_configure_v2_full.sh #OHv2 sBit test support
```

Configure the GBTs

The first argument is the OH number, and the second argument is the GBT number within the OH

```
$OH=1{2}

python gbt_ge21_map.py $OH 0 config
~/gbt_config/GBTX_GE21_OHv2_GBT_0_minimal_2020-01-17.txt

python gbt_ge21_map.py $OH 1 config
~/gbt_config/GBTX_GE21_OHv2_GBT_1_minimal_2020-01-31.txt
```

Check the GBTx transmission to CTP7 (VTRX)

Run this command

```
readKW OH_LINKS.OH0.GBT
```

This will read GBT status of the first OH0. You will get an output like this (if both GBT links on OH0 only are “ready”):

```
0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT0_READY 0x00000001

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT1_READY 0x00000001

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT2_READY 0x00000000

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT0_WAS_NOT_READY 0x00000000

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT1_WAS_NOT_READY 0x00000000
```

```
0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT2_WAS_NOT_READY 0x00000001

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT0_RX_HAD_OVERFLOW 0x00000000

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT1_RX_HAD_OVERFLOW 0x00000000

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT2_RX_HAD_OVERFLOW 0x00000000

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT0_RX_HAD_UNDERFLOW 0x00000001

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT1_RX_HAD_UNDERFLOW 0x00000001

0x65800400 r     GEM_AMC.OH_LINKS.OH0.GBT2_RX_HAD_UNDERFLOW 0x00000001
```

You are only interested in GBT0 and GBT1. If the GBT#_READY is 1 it means that currently CTP7 is receiving valid GBT data on that link. If the GBT#_WAS_NOT_READY is 0 it means that this link never saw invalid GBT frames on that link since last reset. Similarly you want to see GBT#_RX_HAD_OVERFLOW and GBT#_RX_HAD_UNDERFLOW to be 0, that indicates that recovered clock from the GBT links on CTP7 side matches the TTC clock. After you connect the OH you will want to clear the `WAS_NOT_READY, HAD_OVERFLOW, HAD_UNDERFLOW` flags, and you can do that by issuing a so-called "link reset" by running this command:

```
write GEM_AMC.GEM_SYSTEM.CTRL.LINK_RESET 1
```

Check the SCA ASIC

For the SCA you can do the following things: 1) check that SCA is receiving and sending valid data on the elinks:

Clear the error counters and reset the SCA

```
write GEM_AMC.SLOW_CONTROL.SCA.CTRL.MODULE_RESET 1
```

Check the following registers

This should be 1

```
read GEM_AMC.SLOW_CONTROL.SCA.STATUS.READY
```

This should be 0

```
read GEM_AMC.SLOW_CONTROL.SCA.STATUS.CRITICAL_ERROR
```

This should be 0 or a small number. We normally see 2 errors after reset here, but the important thing is that this is not incrementing.

```
read GEM_AMC.SLOW_CONTROL.SCA.STATUS.NOT_READY_CNT_OH0
```

2) Because hard reset signal is mapped to GPIO 31, which is the same as on GE1/1, you can check this signal by issuing a hard reset to the OH, and checking if you see a pulse on U34-4. The pulse is quite long because these GPIOs are quite slow, so you should expect to see a pulse several microseconds long. To issue the hard-reset do this:

```
write GEM_AMC.SLOW_CONTROL.SCA.CTRL.OH_FPGA_HARD_RESET 1
```

9. Fuse initial configuration into both GBT ASICs

If both links (step 8) are well established and stable, a permanent configuration should be programmed (fused) into GBT ASICs. **Note this is one time programming and it must be done with great care since the configurations for GBT1 and GBT2 are different. If programmed in a wrong order, the board may be not usable at all**. Follow the following steps:

9.1. Disconnect the dongle from the OH or from the computer (if connected)

9.2. Power-cycle the OH (this will make sure that the GBTX is reset; note that if you don't disconnect the dongle, the GBTX remembers the configuration, so it seems that it stays somewhat powered through the dongle)

9.3. Reconnect the dongle to the GBT1 (connector P2)

9.4. Restart the programmer software

9.5. Import the configuration file https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/GBTX_GE21_OHv2_GBT_0_minimal_2020-01-17.txt

9.6. Click “write GBTX”

9.7. Also “click read GBTX”

9.8. Check that GBT1 is ready on the CTP7

9.9. In the programmer software go to the tab labeled "Fuse my GBTX!"

9.10. Click "update view"

9.11. Scroll through the table on the left to make sure that all rows are green -- that means that the readback values match the configuration file values (a few rows at the very bottom can be red, it's fine because those few registers are read-only and expected to not match, but all others should be green)

9.12. Click "select not zero values" -- this will select all registers with non-zero values to be fused

9.13. Check both checkboxes in the "Fuse GBTX" box

9.14. Click the FUSE button (only takes a second or so to finish).

9.15. That's it, the chip is fused, now we need to check if everything went fine, that is that the chip remembers the configuration after a power cycle, and also that all bits match the configuration file. To do that, you need to do steps 1 to 11 again, but skip step 6 (do not write GBTX) and if you see that the table is all green in step 9, then everything is fine (remember that it's very important to disconnect the dongle and power-cycle the OH at the beginning, otherwise the GBTX won't reset its configuration).

9.16. Connect dongle to GBT2 (connector P3) and repeat steps 9.1-9.15. Use configuration file https://twiki.cern.ch/twiki/pub/CMS/GE21OHBoardTwiki/GBTX_GE21_OHv2_GBT_1_minimal_2020-01-31.txt

9.17. If everything goes well, you can disconnect the dongle, and if you power-cycle the OH, the CTP7 should see that both GBT devices are ready right after you power them

up. If that's the case, then you can disable the I2C interface (that will enable configuration over the fiber) with SW3-4 (set CONFIGSEL=0).

10. Test of the FPGA access via JTAG cable and Master/Slave Interface

10.1. Install SW3 switches per Fig.6 to be able to access the FPGA via JTAG cable

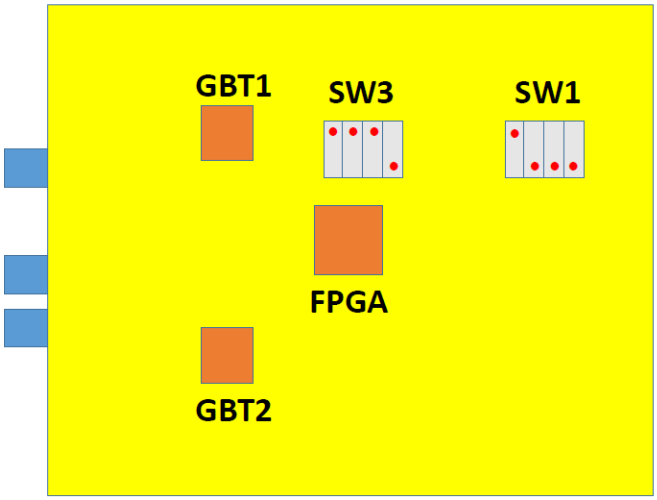


Figure 6: Access to the FPGA via JTAG cable

10.2. Connect Xilinx USB cable to connector P4

10.3. Run IMPACT software from Xilinx ISE 14.7. Check “initialize JTAG chain” and make sure there is one device (XC7A200T).

10.4. Select http://padley.rice.edu/cms/OHv2_PRBS7_060520.bit

file and load it to XC7A200T. Make sure both INIT (D2) and DONE (D1) LEDs are on and two FPGASEL1 (D6) and FPGASEL2 (D7) are blinking at a frequency of ~4Hz (that means the clock is provided from GBT1 and the FPGA is running the firmware loaded). This firmware configures all 4 MGT serializers of the FPGA into Pseudo Random Binary Sequence (PRBS7) data transmission mode at 3.2Gbps rate using a 160Mhz reference clock provided by the GBT1.

10.5. This version of firmware also translated 6 outputs of the continuously running 40MHz counter to 6 differential LVDS outputs of the Master/Slave connector P1. They can be checked with an oscilloscope (pins 1&2: 20MHz, 3&4: 10 MHz, 5&6: 5MHz, 7&8: 2.5MHz, 9&10: 1.25MHz, 11&12: 625kHz).

11. Test of VTTX Optical Links

There are three alternative methods to test the integrity of the VTTX optical links, all based on PRBS7 data patters generated from the FPGA. In one case (11.1.) the receiver is the Optical Trigger Motherboard (OTMB2019). In the second case (11.2) the receiver is the Optical Data Acquisition Motherboard (ODMB). The third method (11.3) uses dedicated CTP7 receivers.

11.1. Test of the VTTX optical links with the OTMB2019

The file per 10.4 should be loaded to the OH FPGA. The OTMB2019 needs the following firmware for the Virtex-6 FPGA:

http://padley.rice.edu/cms/OTMB2019_PRBS7_060720.bit

This firmware allows to check both the 12-channel Firefly receiver and the 4-channel Firefly transceiver residing on the OTMB2019 mezzanine card. 10 test points on the bottom of the OTMB2019 mezzanine are used to monitor PRBSERROUT outputs of the MGT receivers. Use a standard 12LC-to-MTP optical adapter for this test. The mapping of optical channels is the following:

TP10: receiver from LC fiber 1 (12-channel Firefly receiver)

TP9: receiver from LC fiber 2 (12-channel Firefly receiver)

TP8: receiver from LC fiber 3 (12-channel Firefly receiver)

TP7: receiver from LC fiber 4 (12-channel Firefly receiver)

TP6: receiver from LC fiber 1 (4-channel Firefly transceiver)

TP5: receiver from LC fiber 3 (4-channel Firefly transceiver)

TP4: receiver from LC fiber 4 (4-channel Firefly transceiver)

TP3: receiver from LC fiber 2 (4-channel Firefly transceiver)

TP2: OR of PRBSERROUT outputs of fibers 3 and 4

TP1: OR of PRBSERROUT outputs of fibers 1 and 2.

If only one VTTX is installed, it is recommended to connect fibers 1 and 2. MTP connector can be attached to either 12-channel Firefly receiver or 4-channel Firefly transceiver. In case of 12-channel receiver TP10 and TP9 (or their OR in TP1) should be monitored and must be "0". Any "1" pulses on these outputs indicate transmission errors. In case of 4-channel transceiver TP3 and TP6 should be monitored and must be "0". Run the test for at least 5 minutes; this is equivalent to ~10^{12} bit transfers. If both VTTX are under tests, use fibers 1..4 and monitor more test points (above).

11.2. Test of the VTTX optical links with the ODMB

The file per 10.4 should be loaded to the OH FPGA. The ODMB FPGA needs the following file:

http://padley.rice.edu/cms/ODMB_PRBS7_OH_060520.bit

(or **ODMB_PRBS7_OH_060520.mcs** for XCF128X PROM). Connect both channels of VTTX (using fibers 1 and 2 of the 12LC-to-MTP optical adapter) to ODMB MTP input. The ODMB firmware configures 12 GTX receivers into continuous PRBS7 receiving mode at 3.2Gbps. Each receiver has its own PRBSERROUT output which is routed to TP14 and TP15 test points (with

"0" levels if there are no errors). Errors, if any, for channels 1 and 2 are also counted by an internal counters. Five lower bits of these counters are provided to LEDs on the front panel of the ODMB (shown in Fig.7). Push button PB1 is needed to reload the FPGA from its XCF128X PROM. The PB2 is needed to reset internal error counters. The lower PB3 is not used.

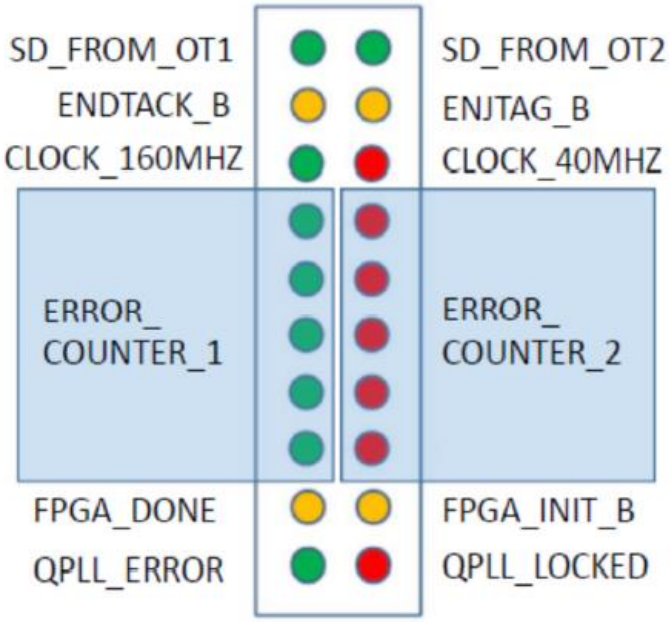


Figure 7: LEDs on the front panel of ODMB, as used in the test firmware

When the firmware is loaded to both OH and ODMB boards, the test starts automatically. Reset the internal error counters on the ODMB and continue monitoring LEDs on the front panel for ~5 minutes. That's enough to transmit 10^{12} bits. Both counters should be "0". Test points TP14 and TP15 can be monitored as well. **The bit error rate (BER) of below 10^{-12} is an acceptance criteria for trigger links.**

11.3. Test of the VTTX optical links with the CTP7

For this you first need to connect the VTTXs to trigger inputs on the CTP7. The trigger inputs are currently on CXP3 (the bottom transceiver that is sticking out on the CTP7 board, there's 3 of them in total, you need the bottom one). Just connect both fibers of the one of the VTTXs to fibers 1 & 2 of that transceiver (that will be mapped to "OH0"), and the other VTTX should connect to fibers 3 & 4 (that will be mapped to "OH1").

Also you need to load the FPGA with the trigger firmware. Then you can check the trigger link health by doing the following.

Reset the trigger module:

```
write GEM_AMC.TRIGGER.CTRL.MODULE_RESET 1
```

Open the register interface

```
reg
```

Read the first VTTX link

```
readKW GEM_AMC.TRIGGER.OH0
```

You will get an output like this:

```
0x66000400 None     GEM_AMC.TRIGGER.OH0

0x66000400 r     GEM_AMC.TRIGGER.OH0.TRIGGER_RATE 0x02638e98

0x66000404 r     GEM_AMC.TRIGGER.OH0.TRIGGER_CNT 0xffffffff

0x66000440 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_0_RATE 0x00000000

0x66000444 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_1_RATE 0x0000056b

0x66000448 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_2_RATE 0x0000aab8

0x6600044c r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_3_RATE 0x000be2b7

0x66000450 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_4_RATE 0x003109bc

0x66000454 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_5_RATE 0x008c675e

0x66000458 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_6_RATE 0x00d2d3ed

0x6600045c r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_7_RATE 0x009d6828

0x66000460 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_8_RATE 0x00294e8f

0x66000480 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_0_CNT 0x00a9d2bf

0x66000484 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_1_CNT 0x1ca7cd25

0x66000488 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_2_CNT 0xffffffff

0x6600048c r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_3_CNT 0xffffffff

0x66000490 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_4_CNT 0xffffffff

0x66000494 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_5_CNT 0xffffffff

0x66000498 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_6_CNT 0xffffffff

0x6600049c r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_7_CNT 0xffffffff

0x660004a0 r     GEM_AMC.TRIGGER.OH0.CLUSTER_SIZE_8_CNT 0xffffffff

0x66000680 r     GEM_AMC.TRIGGER.OH0.LINK0_SBIT_OVERFLOW_CNT 0x0000ffff

0x66000680 r     GEM_AMC.TRIGGER.OH0.LINK1_SBIT_OVERFLOW_CNT 0x0000ffff

0x66000684 r     GEM_AMC.TRIGGER.OH0.LINK0_MISSED_COMMA_CNT 0x0000ffff

0x66000684 r     GEM_AMC.TRIGGER.OH0.LINK1_MISSED_COMMA_CNT 0x0000ffff
```

```
0x6600068c r     GEM_AMC.TRIGGER.OH0.LINK0_OVERFLOW_CNT 0x00000000

0x6600068c r     GEM_AMC.TRIGGER.OH0.LINK1_OVERFLOW_CNT 0x00000000

0x66000690 r     GEM_AMC.TRIGGER.OH0.LINK0_UNDERFLOW_CNT 0x0000ffff

0x66000690 r     GEM_AMC.TRIGGER.OH0.LINK1_UNDERFLOW_CNT 0x0000ffff

0x66000694 r     GEM_AMC.TRIGGER.OH0.LINK0_SYNC_WORD_CNT 0x00000000

0x66000694 r     GEM_AMC.TRIGGER.OH0.LINK1_SYNC_WORD_CNT 0x00000000
```

Here the registers at the top you don't really care about, but the ones you do care about are:

```
GEM_AMC.TRIGGER.OH0.LINK0_MISSED_COMMA_CNT

GEM_AMC.TRIGGER.OH0.LINK1_MISSED_COMMA_CNT

GEM_AMC.TRIGGER.OH0.LINK0_OVERFLOW_CNT

GEM_AMC.TRIGGER.OH0.LINK1_OVERFLOW_CNT

GEM_AMC.TRIGGER.OH0.LINK0_UNDERFLOW_CNT

GEM_AMC.TRIGGER.OH0.LINK1_UNDERFLOW_CNT
```

All of these should be 0 or a small value like 1 or 2, and not incrementing between multiple reads. If that's the case, it means the CTP7 is receiving valid trigger data from the OH.

You can also read the second VTTX link

```
readKW GEM_AMC.TRIGGER.OH1
```

12. Load FPGA from GBT1

When both GBT ASICs are fused in with the copper cable, CONFIGSEL switch (SW3-4) should be connected to “0” to allow configuration changes from the optical link (fig.8).

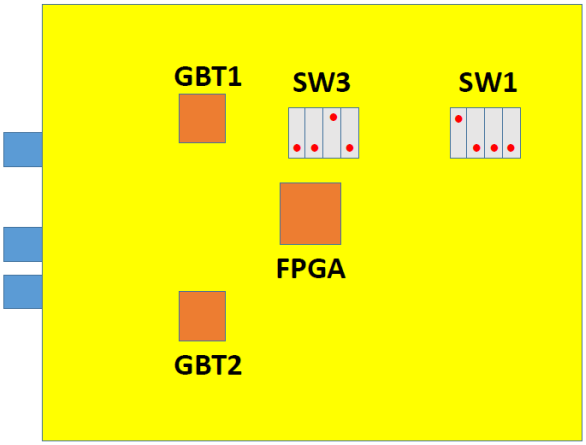


Figure 8: Switches on the OH board for regular operation, top view

This test allows to check the GBT1-to-FPGA 8-bit downloading path.78 Mbit files is loaded at 80Mhz data rate resulting in 120 ms latency for Xilinx XC7A200T-2FBG484C FPGA. The process involves:

- Send Hard_Reset pulse from SCA ASIC to the FPGA
- Confirm DONE=0 from the FPGA (not loaded)
- Program from GBT1
- Confirm DONE=1 (loaded successfully)

This process may be repeated N times. Make sure the SCA is ready (READY = 1)

```
read GEM_AMC.SLOW_CONTROL.SCA.STATUS.READY
```

Run the following python script:

```
python ge21_promless_test.py 1 1000
```

The first argument is the OH "bitmask" e.g. 1 will do it on OH0 only, 2 will do it on OH1 only, 3 will do it on both OH0 and OH1, etc. It is recommended to only do it over one OH at a time. The second argument is the number of iterations to do. 1000 for full test. 10 or 100 for quick test. Both DONE (D1) and INIT (D2) green LEDs should be on. **Based on tests performed on GE1/1 electronics we specify that 1000 reloading cycles without errors is an acceptance criteria for this path. This takes ~10 minutes to perform this test.**

13. Read ADCs from SCA ASIC

This test allows to check 32-channel ADC from the SCA ASIC and measure voltages and currents on the OH and GEB boards as well as temperatures in 4 points on the OH board. Run the following script:

```
python sca.py 1 adc-read
```

The first argument here is again the OH "bitmask" (so 1 for OH0, 2 for OH1).
The 32 ADC channels are assigned as follows:

- IN0: +1.0 FPGA core voltage
- IN1: +1.0AVCC voltage (MGTAVCC)
- IN2: +1.2AVTT voltage (MGTAVTT)
- IN3: +1.8V
- IN4: +1.5V
- IN5: +2.5V
- IN6: VTRX1_RSSI
- IN7: VTRX2_RSSI
- IN8: MONITOR1 current monitor from FEAST1 (1.8V for the OH board)
- IN9: MONITOR2 current monitor from FEAST2 (1.5V for the OH board)
- IN10: MONITOR3 current monitor from FEAST3 (2.5V for the OH board)
- IN11: MONITOR4 current monitor from FEAST4 (analog 1.2V for all VFAT slots)
- IN12: MONITOR5 current monitor from FEAST5 (digital 1.2V for all VFAF slots)

- IN13: PT1000 Temperature sensor near U2 (GBT1), see Table 1
- IN14: PT1000 Temperature sensor near U3 (GBT2), see Table 1
- IN15: PT1000 Temperature sensor near U22 (VTRX), see Table 1
- IN16: PT1000 Temperature sensor near U8 (1.5V voltage regulator), see Table 1
- IN17: High precision 1kOhm resistor connected to GND

Table 1: PT1000 Temperature vs Resistance Chart

°C	°F	Ω	°C	°F	Ω
-10	14	961	55	131	1213
-5	23	980	60	140	1232
0	32	1000	65	149	1252
5	41	1019	70	158	1271
10	50	1039	75	167	1290
15	59	1058	80	176	1309
20	68	1078	85	185	1328
25	77	1097	90	194	1347
30	86	1117	95	203	1366
35	95	1136	100	212	1385
40	104	1155	105	221	1404
45	113	1175	110	230	1423
50	122	1194	115	239	1442

Resistance values of the Pt1000-sensors

Below is a typical reading from ADCs:

```
Channel  0  OH  0:  1019  counts  (0x3fb)  =  248.840049mV
Channel  1  OH  0:  1026  counts  (0x402)  =  250.549451mV
Channel  2  OH  0:  1240  counts  (0x4d8)  =  302.808303mV
Channel  3  OH  0:  1876  counts  (0x754)  =  458.119658mV
Channel  4  OH  0:  1515  counts  (0x5eb)  =  369.963370mV
Channel  5  OH  0:  2562  counts  (0xa02)  =  625.641026mV
Channel  6  OH  0:  1987  counts  (0x7c3)  =  485.225885mV
Channel  7  OH  0:  1954  counts  (0x7a2)  =  477.167277mV
Channel  8  OH  0:  38  counts  (0x26)  =  9.279609mV
Channel  9  OH  0:  320  counts  (0x140)  =  78.144078mV
Channel  10  OH  0:  128  counts  (0x80)  =  31.257631mV
Channel  11  OH  0:  109  counts  (0x6d)  =  26.617827mV
Channel  12  OH  0:  140  counts  (0x8c)  =  34.188034mV
Channel  13  OH  0:  461  counts  (0x1cd)  =  112.576313mV
Channel  14  OH  0:  461  counts  (0x1cd)  =  112.576313mV
Channel  15  OH  0:  462  counts  (0x1ce)  =  112.820513mV
Channel  16  OH  0:  455  counts  (0x1c7)  =  111.111111mV
Channel  17  OH  0:  414  counts  (0x19e)  =  101.098901mV
Channel  18  OH  0:  770  counts  (0x302)  =  188.034188mV
Channel  19  OH  0:  754  counts  (0x2f2)  =  184.126984mV
Channel  20  OH  0:  807  counts  (0x327)  =  197.069597mV
Channel  21  OH  0:  669  counts  (0x29d)  =  163.369963mV
```

Channel	22	OH	0:	743	counts	(0x2e7)	=	181.440781mV
Channel	23	OH	0:	683	counts	(0x2ab)	=	166.788767mV
Channel	24	OH	0:	970	counts	(0x3ca)	=	236.874237mV
Channel	25	OH	0:	744	counts	(0x2e8)	=	181.684982mV
Channel	26	OH	0:	832	counts	(0x340)	=	203.174603mV
Channel	27	OH	0:	1341	counts	(0x53d)	=	327.472527mV
Channel	28	OH	0:	840	counts	(0x348)	=	205.128205mV
Channel	29	OH	0:	841	counts	(0x349)	=	205.372405mV
Channel	30	OH	0:	854	counts	(0x356)	=	208.547009mV
Channel	31	OH	0:	2703	counts	(0xa8f)	=	660.073260mV

If, for example, channel 0 measurement is 249mV, the input FPGA core voltage is (taking into account an input resistor divider) 249mV x 4 = 996 mV, or well within the 5% tolerance. Expected values for channels 0..5 are 250mV, 250mV, 300mV, 450mV, 375mV, 625mV respectively.

Channels 6 and 7 are the Receiver Strength Signal Indicator, RSSI, for both VTRX transceivers. Expected measurements are 450..470mV with the active optical receiver.

MONITOR[5:1] are the outputs of five MAX4372T current sense amplifiers with voltage output. They allow to measure the currents of five FEAST DC-DC converters located on the GEB boards. MAX4372T device provides 20V/V gain and with a current sensing resistor of 10mOhm on the GEB board the MONITOR[5:1] outputs are limited to 2V. These signals are provided to the SCA with a resistor divider 3kOhm/1kOhm since the SCA analog input range is between 0 and 1.0V. If the ADC output is, for example, 70mV, then the MONITOR is 70mV x 4 = 280mV (due to a resistor divider) and the current would be 280mV/10mOhm/20 = 1.4A.

For temperature measurements (channels 13-16) the 100uA current source should be enabled for each of these channels in the SCA ASIC. Then, if the output measurement is, for example, 110mV, the resistance would be R=V/I=110mV/100uA=1100 Ohm which corresponds to ~29C (Table 1).

Channel 17 is connected to a precise 1kOhm resistor. Channels 18-30 are not used. Channel 31 of the SCA ADC is connected to the internal temperature sensor. The conversion graph (Fig.9) is taken from the SCA Manual.

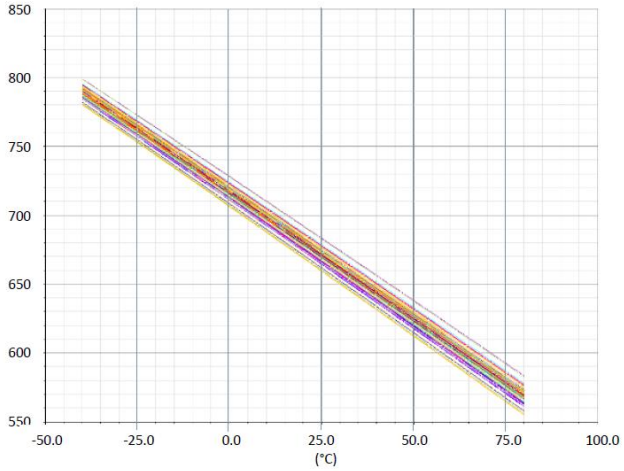


Figure 9: Conversion graph for the SCA internal temperature sensor

14. Integrated Tests of Internal Links

Continuously running Pseudo Random Binary Sequence (PRBS) of predefined patterns allows to check both GBT links as well as an embedded trigger link.

Loopback testing sends a signal from the CTP7 to the GBTx to the FPGA and then back to the CTP7. This checks both that the elinks between the GBTx -> FPGA and the path to the CTP7 are good.

For PRBS tests, by default the gemloader_configure_v2.sh loads the OHv2 loopback firmware, and when you run the ge21_promless_test.py it will get loaded to the OH FPGA. At that point in order to start the loopback test you have to open the reg_interface by just running "reg"

```
reg
```

Select the OH to be used. It is 0 if you just use the first OH, 1 for the second OH

```
 write GEM_AMC.GEM_TESTS.OH_LOOPBACK.CTRL.OH_SELECT 0
```

Put the CTP7 into a loopback testing mode. Note that VFAT or OH communication won't work while the CTP7 is in this mode, so e.g. phase scans and OH programming are not possible, so always make sure this mode is off when doing the other tests and only put this on for PRBS test)

```
write GEM_AMC.GEM_SYSTEM.TESTS.GBT_LOOPBACK_EN 1
```

Read the loopback registers

```
readKW GEM_AMC.GEM_TESTS.OH_LOOPBACK
```

This will print lots of stuff which includes 3 registers for each elink: 1) PRBS_LOCKED -- if this is 1 that means the CTP7 is receiving back a valid PRBS7 stream from this elink 2) MEGA_WORD_CNT -- this counts the total number of words checked by the PRBS7 checker after the PRBS_LOCKED was asserted (this counts in units of one million bytes) 3) ERROR_CNT -- number of PRBS errors found

The output will look like this:

```
0x66804040 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_0.PRBS_LOCKED
0x00000000

0x66804044 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_0.MEGA_WORD_CNT
0x00000000

0x66804040 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_0.ERROR_CNT
0x00000000

0x66804048 None     GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_1

0x66804048 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_1.PRBS_LOCKED
0x00000000

0x6680404c r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_1.MEGA_WORD_CNT
0x00000000

0x66804048 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_1.ERROR_CNT
0x00000000
```

```
0x66804050 None     GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_2

0x66804050 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_2.PRBS_LOCKED
0x00000000

0x66804054 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_2.MEGA_WORD_CNT
0x00000000

0x66804050 r GEM_AMC.GEM_TESTS.OH_LOOPBACK.GBT_0.ELINK_2.ERROR_CNT
0x00000000
```

..... the elinks are numbered in natural order in terms of GBTX: ELINK_0 is DIN0, ELINK_1 is DIN4, ELINK_2 is DIN8, ELINK_3 is DIN12, ... ELINK_10 is DIO1, ELINK_11 is DIO5, ELINK_12 is DIO9, ELINK_13 is DIO13. Not all of these elinks are connected to the OH FPGA, so do not expect all of them to lock, you should only see these elinks lock:

```
GBT_0.ELINK_6

GBT_0.ELINK_7

GBT_0.ELINK_8

GBT_0.ELINK_9

GBT_1.ELINK_6

GBT_1.ELINK_7

GBT_1.ELINK_8

GBT_1.ELINK_9

GBT_1.ELINK_10

GBT_1.ELINK_11

GBT_1.ELINK_12

GBT_1.ELINK_13
```

Don't forget to switch the CTP7 back to normal mode: write

```
GEM_AMC.GEM_SYSTEM.TESTS.GBT_LOOPBACK_EN 0
```

You can also try to inject a PRBS error on the CTP7 TX, and see if it comes back through the RX and is detected by the CTP7: write

```
GEM_AMC.GEM_TESTS.OH_LOOPBACK.CTRL.INJECT_ERR 1
```

To reset the loopback counters use this reset

```
write GEM_AMC.GEM_TESTS.OH_LOOPBACK.CTRL.RESET 1
```

10^{12} bits to be transmitted without errors per each elink is an acceptance criteria. This takes ~50 minutes.

15. Integrated Test of VFAT Interface

VFAT3 is an ASIC with 128 identical channels of a preamplifier, shaper and comparator. Following the comparator are digital circuits used for generating the “FAST-OR” outputs, data storage and data packet construction. One very useful feature is an internal programmable pulse generator which can deliver a charge pulse to a given channel for test and calibration purposes. Variables include channel selection, the polarity and amount of charge and the phase of the charge delivery within one clock period. The amount of charge is controlled by an internal DAC.

The pulse scan allows to characterize such parameters as noise and threshold. A channel is selected and an input charge swept through a fixed threshold providing data for S-curve analysis. An S-curve (having the “S” shape) is a histogram of hits on a selected channel as a function of the input pulse amplitude.

15.1. Interaction with VFAT3 ASICs

1) after power-up or VFAT reset, the VFAT is not doing anything, except waiting for a SYNC command to determine which 320MHz rising edge corresponds to the 40MHz rising edge, so that it can align it's 8bit deserializer and also determine the phase of the 40MHz analog sampling clock.
2) when you write to GEM_AMC.GEM_SYSTEM.CTRL.LINK_RESET, the CTP7 will send out the special SYNC command
3) every orbit after a BC0 the CTP7 is sending out a SYNC_VERIFY command, and if the VFAT receives it correctly, it will respond with SYNC_VERIFY_ACK
4) After sending the SYNC_VERIFY command, the CTP7 starts a countdown of 1500 BX, and if it does not receive a SYNC_VERIFY_ACK from that VFAT in that time, it counts it as a communication error
5) if there are no communication errors for 10 orbits in a row (meaning that 10 consecutive SYNC_VERIFY and SYNC_VERIFY_ACK cycles have completed successfully), then CTP7 sets the "LINK_GOOD" to 1 for that VFAT.
6) if the link is already declared good, and CTP7 detects 100 communication errors in a row for that VFATs, it will reset the LINK_GOOD flag to 0.

So in summary the CTP7 is exchanging messages in both directions with each VFAT once per orbit, and counts any errors, and if the link is stable for long enough, it will say LINK_GOOD = 1 for that VFAT.

Here is an example: write 1 to GEM_AMC.GEM_SYSTEM.CTRL.LINK_RESET to synchronize all VFATs, and then read GEM_AMC.OH_LINKS.OH0.VFAT15 (the GBT0 elink 0 is mapped to VFAT15 in firmware), which shows this:

```
0x658004b8 r    GEM_AMC.OH_LINKS.OH0.VFAT15.LINK_GOOD 0x00000001
0x658004b8 r    GEM_AMC.OH_LINKS.OH0.VFAT15.SYNC_ERR_CNT 0x00000000
0x658004b8 r    GEM_AMC.OH_LINKS.OH0.VFAT15.DAQ_EVENT_CNT 0x00000000
0x658004b8 r    GEM_AMC.OH_LINKS.OH0.VFAT15.DAQ_CRC_ERROR_CNT     0x00000000
```

As you can see the link is clean without any single error (you can wait several minutes to allow it to accumulate a large number of SYNC_VERIFY messages, but see no errors). All other VFATs report LINK_GOOD = 0.

The first test of would be to read chipID's from all attached VFAT3 cards (use numbers VFAT0…VFAT11); the example below is for VFAT0 accessible by OH0:

CTP7 > read GEM_AMC.OH.OH0.GEB.VFAT0.HW_CHIP_ID

The returned value looks like this: 0x00000a08

15.2. Phase Scan

Perform a phase scan over the GBTs

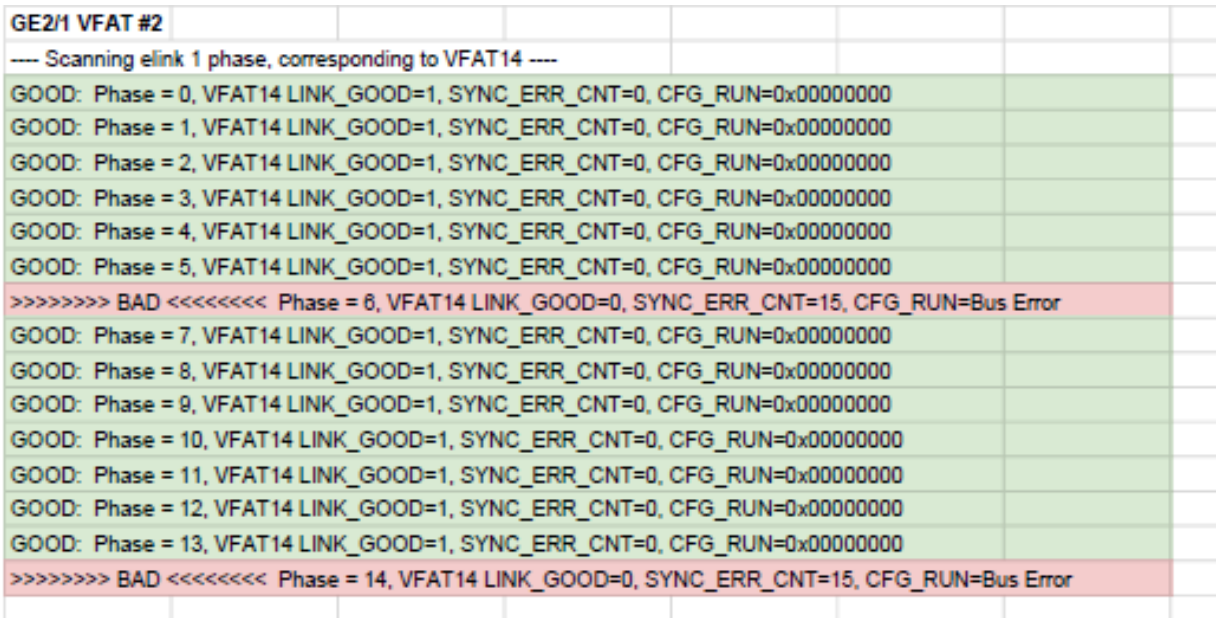
Used to perform a phase scan; checks the connection between the OH and 12 VFAT3 ASICs via dedicated links. Outputs a list of good/bad phases for each of the 12 VFATs. Not all phases need to be good, but there should be a 'safe window' of operation for each VFAT. A phase scan is done with 10K repeated register read operations of three VFAT3 registers after each phase has been set.

Notice the configuration file is different when you do it on GTB1 (GBT mode) and GBT2

```
$OH=1{2}

python gbt_ge21_map.py $OH 0 ge21-phase-scan
~/gbt_config/GBTX_GE21_OHv2_GBT_0_minimal_2020-01-17.txt

python gbt_ge21_map.py $OH 1 ge21-phase-scan
~/gbt_config/GBTX_GE21_OHv2_GBT_1_minimal_2020-01-31.txt
```



GE2/1 VFAT #2
---- Scanning elink 1 phase, corresponding to VFAT14 ----
GOOD: Phase = 0, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 1, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 2, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 3, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 4, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 5, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
>>>>>>>> BAD <<<<<<<< Phase = 6, VFAT14 LINK_GOOD=0, SYNC_ERR_CNT=15, CFG_RUN=Bus Error
GOOD: Phase = 7, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 8, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 9, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 10, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 11, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 12, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
GOOD: Phase = 13, VFAT14 LINK_GOOD=1, SYNC_ERR_CNT=0, CFG_RUN=0x00000000
>>>>>>>> BAD <<<<<<<< Phase = 14, VFAT14 LINK_GOOD=0, SYNC_ERR_CNT=15, CFG_RUN=Bus Error

Figure 9: An example of the “safe window”

15.2. Program “best” phases

Note that you have to make sure that communication to all VFATs is good before you start the s-bit test. Start with the phase scan for both GBTs, and note the “best” phases (for each slot pick a number in the middle between the two “bad” phases printed in red. The “good” window is normally 7-step wide, so the good phase is usually 3 phases away from any “bad” spot). After any power-cycle or a phase scan run, you have to program the desired phases to both GBT (the six numbers at the end have to be replaced by your picked best phases for each slot):

For GBT1:

python gbt_ge21_map.py 0 0 ge21-program-phases ~/gbt_config/GBTX_GE21_OHv2_GBT_0_minimal_2020-01-17.txt 8 8 6 8 9 9

For GBT2:

python gbt_ge21_map.py 0 1 ge21-program-phases ~/gbt_config/GBTX_GE21_OHv2_GBT_1_minimal_2020-01-31.txt 8 8 6 8 9 9

This will also print the communication status of each slot with the provided phases (make sure they all are green).

15.3. S-bit Test

There are 64 S-bit channels per VFAT, representing a binary value of a hit information from two strips. These trigger bits are multiplexed in time (8 different channels from each VFAT to the OH

FPGA). S-bits are scanned for all VFATs over the DAC comparator threshold (from 0 to 255 units). If an S-bit is seen at a particular threshold, it is recorded as a hit.

The script checks each sbit differential pair connectivity and signal quality by going through each sbit differential pair input, and doing a timing scan while pulsing the corresponding VFAT channel, and extracting a "good window" position and size. This test allows to verify 12 x 9 = 108 differential trigger lines from 12 VFAT3 ASICs to the OH FPGA, and the functionality of the FPGA. Execution will take ~15 minutes for OH0. **The acceptance criteria is that all channels must be labeled as good by the script.** To perform the s-bit test run the following script:

S-bit testing

```
$OH_NUMBER=0{1}

$VFAT_MIN=0

$VFAT_MAX=11

cd ~/apps/reg_interface

python sbit_timing_scan_oh_sbit_hitmap.py $OH_NUMBER $VFAT_MIN $VFAT_MAX
```

The results for 12 ASICs will look like this:

```
-----------------------------------------------------------------------------------------------------------------------
eagle42:~/apps/reg_interface$ python sbit_timing_scan_oh_sbit_hitmap.py 1 0 11

Loading shared library: /mnt/persistent/texas/shared_libs/librwreg.so

Parsing /mnt/persistent/texas/gem_amc_top.xml ...


#########################################################################################################

Scanning VFAT 0

#########################################################################################################


   2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: x - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit3: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit4: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit5: x x x - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - -

Sbit6: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit7: x x - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x

min center = 18
```

sot : best_dly= 18

bit0: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit1: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit2: center= 0 best_dly= 0 width= 36 (2.808000 ns)

bit3: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit4: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit5: center= 1 best_dly= 1 width= 34 (2.652000 ns)

bit6: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit7: center= 1 best_dly= 1 width= 34 (2.652000 ns)

##

Scanning VFAT 1

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x x x

Sbit1: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x x

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x x

Sbit4: - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit5: - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x x x

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x x x

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 34 (2.652000 ns)

bit1: center= 0 best_dly= 1 width= 35 (2.730000 ns)

bit2: center=-1 best_dly= 0 width= 35 (2.730000 ns)

bit3: center=-1 best_dly= 0 width= 35 (2.730000 ns)

bit4: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit5: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit6: center=-1 best_dly= 0 width= 34 (2.652000 ns)

bit7: center=-1 best_dly= 0 width= 34 (2.652000 ns)

##

Scanning VFAT 2

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit1: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit2: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit3: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit4: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit5: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit6: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit7: center=-1 best_dly= 0 width= 37 (2.886000 ns)

##

Scanning VFAT 3

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x x

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit7: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

min center = 16

sot : best_dly= 2

bit0: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit1: center=-1 best_dly= 1 width= 34 (2.652000 ns)

bit2: center=-1 best_dly= 1 width= 34 (2.652000 ns)

bit3: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit4: center=-2 best_dly= 0 width= 32 (2.496000 ns)

bit5: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit6: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit7: center=-2 best_dly= 0 width= 33 (2.574000 ns)

##

Scanning VFAT 4

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit1: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit2: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit3: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit4: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit5: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

Sbit7: - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - -

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit1: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit2: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit3: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit4: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit5: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit6: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit7: center=-1 best_dly= 0 width= 37 (2.886000 ns)

```
##################################################################################################

Scanning VFAT 5

##################################################################################################


    2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit5: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

min center = 17

sot :         best_dly= 1

bit0: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit1: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit2: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit3: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit4: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit5: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit6: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit7: center= 0 best_dly= 1 width= 36 (2.808000 ns)


##################################################################################################

Scanning VFAT 6

##################################################################################################


    2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -
```

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit1: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit2: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit3: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit4: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit5: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit6: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit7: center=-1 best_dly= 0 width= 37 (2.886000 ns)

##

Scanning VFAT 7

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x x

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit3: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 35 (2.730000 ns)

bit1: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit2: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit3: center= 0 best_dly= 1 width= 35 (2.730000 ns)

bit4: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit5: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit6: center= 0 best_dly= 1 width= 36 (2.808000 ns)

bit7: center=-1 best_dly= 0 width= 37 (2.886000 ns)

##

Scanning VFAT 8

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit1: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit2: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit3: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit4: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit5: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit6: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit7: center=-1 best_dly= 0 width= 37 (2.886000 ns)

##

Scanning VFAT 9

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: x x - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit3: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit4: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit5: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - x

Sbit7: x - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

min center = 18

sot : best_dly= 18

bit0: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit1: center= 0 best_dly= 0 width= 35 (2.730000 ns)

bit2: center= 0 best_dly= 0 width= 36 (2.808000 ns)

bit3: center= 0 best_dly= 0 width= 36 (2.808000 ns)

bit4: center= 0 best_dly= 0 width= 36 (2.808000 ns)

bit5: center= 0 best_dly= 0 width= 36 (2.808000 ns)

bit6: center= 0 best_dly= 0 width= 36 (2.808000 ns)

bit7: center= 0 best_dly= 0 width= 36 (2.808000 ns)

##

Scanning VFAT 10

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - x x x x x

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - x x x x x

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - x x x x x

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - x x x x x

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - x x x x

min center = 16

sot : best_dly= 2

bit0: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit1: center=-2 best_dly= 0 width= 32 (2.496000 ns)

bit2: center=-2 best_dly= 0 width= 32 (2.496000 ns)

bit3: center=-2 best_dly= 0 width= 32 (2.496000 ns)

bit4: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit5: center=-2 best_dly= 0 width= 33 (2.574000 ns)

bit6: center=-2 best_dly= 0 width= 32 (2.496000 ns)

bit7: center=-2 best_dly= 0 width= 33 (2.574000 ns)

##

Scanning VFAT 11

##

2 1 0 F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2

Sbit0: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit1: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit2: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit3: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit4: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit5: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit6: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

Sbit7: - - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - - -

min center = 17

sot : best_dly= 1

bit0: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit1: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit2: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit3: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit4: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit5: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit6: center=-1 best_dly= 0 width= 37 (2.886000 ns)

bit7: center=-1 best_dly= 0 width= 37 (2.886000 ns)
