

**OPAE tools guide for Vista Creek**

**Version 1.7**

©2018 Intel Corporation. All rights reserved. INTEL, ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Intel Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Intel warrants performance of its semiconductor products to current specifications in accordance with Intel’s standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

**Revision history**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Changes** | **Reason** |
| 1.0 | Huang Wei You You | 02/20/2019 | First release |  |
| 1.1 | Huang Wei | 02/28/2019 | Update fpgaflash tool description |  |
| 1.2 | Tim Whisonant Huang Wei | 03/13/2019 | Add fpgad sections Update fpgaflash tool description |  |
| 1.3 | Huang Wei | 03/26/2019 | Add bmc factory image rsu Add parkvale eeprom rsu Update fpgaflash tool output information |  |
| 1.4 | Huang Wei | 04/15/2019 | Integrate 8x10G and 2x1x25G tools description |  |
| 1.5 | Huang Wei | 06/20/2019 | Add fvl bypass mode test tool description |  |
| 1.6 | Tim Whisonant | 07/10/2019 | Update fpgad section to describe sensor monitoring |  |
| 1.7 | Huang Wei | 07/30/2019 | Add fec mode configuration tool description |  |

Table of Contents

[1 Overview 5](#_Toc15397163)

[1.1 Hardware Overview 5](#_Toc15397164)

[1.2 Software Overview 6](#_Toc15397165)

[ **fpgainfo** 7](#_Toc15397166)

[ **fpgabist** 7](#_Toc15397167)

[ **fpgadiag** 7](#_Toc15397168)

[ **fpgaflash** 7](#_Toc15397169)

[ **fpgaconf** 7](#_Toc15397170)

[ **fpgaport** 7](#_Toc15397171)

[ **mmlink** 7](#_Toc15397172)

[ **fpgad** 7](#_Toc15397173)

[2 Functional Description 8](#_Toc15397174)

[2.1 Information Display 8](#_Toc15397175)

[2.2 PCIe Loopback Test 8](#_Toc15397176)

[2.3 Local Memory Test 8](#_Toc15397177)

[2.4 Network Loopback Test 8](#_Toc15397178)

[2.5 MAC ROM Test 8](#_Toc15397179)

[2.6 Mailbox Test 9](#_Toc15397180)

[2.7 FPGA Image and NIOS Firmware Update 9](#_Toc15397181)

[2.8 FVL Bypass Mode Test 9](#_Toc15397182)

[2.9 Graceful Shutdown 9](#_Toc15397183)

[2.10 FEC mode configuration 10](#_Toc15397184)

[3 Guide 11](#_Toc15397185)

[3.1 fpgainfo 11](#_Toc15397186)

[3.1.1 synopsis 11](#_Toc15397187)

[3.1.2 description 11](#_Toc15397188)

[3.1.3 example 11](#_Toc15397189)

[3.2 fpgadiag 12](#_Toc15397190)

[3.2.1 synopsis 12](#_Toc15397191)

[3.2.2 description 12](#_Toc15397192)

[3.2.3 example 14](#_Toc15397193)

[3.2.4 network loopback guide 14](#_Toc15397194)

[3.3 fpgabist 17](#_Toc15397195)

[3.3.1 synopsis 17](#_Toc15397196)

[3.3.2 description 17](#_Toc15397197)

[3.3.3 example 17](#_Toc15397198)

[3.4 fpgaflash 18](#_Toc15397199)

[3.4.1 synopsis 18](#_Toc15397200)

[3.4.2 description 18](#_Toc15397201)

[3.4.3 example 19](#_Toc15397202)

[3.4.4 procedure 20](#_Toc15397203)

[3.5 fpgaconf 22](#_Toc15397204)

[3.5.1 synopsis 22](#_Toc15397205)

[3.5.2 description 22](#_Toc15397206)

[3.5.3 example 23](#_Toc15397207)

[3.6 fpgaport 23](#_Toc15397208)

[3.6.1 synopsis 23](#_Toc15397209)

[3.6.2 description 23](#_Toc15397210)

[3.6.3 example 23](#_Toc15397211)

[3.7 mmlink 23](#_Toc15397212)

[3.7.1 synopsis 23](#_Toc15397213)

[3.7.2 description 23](#_Toc15397214)

[3.7.3 example 24](#_Toc15397215)

[3.8 fpgad 24](#_Toc15397216)

[3.8.1 synopsis 24](#_Toc15397217)

[3.8.2 description 24](#_Toc15397218)

[3.8.3 example 25](#_Toc15397219)

[3.8.4 defining sensor overrides in fpgad.cfg 25](#_Toc15397220)

[3.9 fecmode 25](#_Toc15397221)

[3.9.1 synopsis 25](#_Toc15397222)

[3.9.2 description 26](#_Toc15397223)

[3.9.3 example 26](#_Toc15397224)

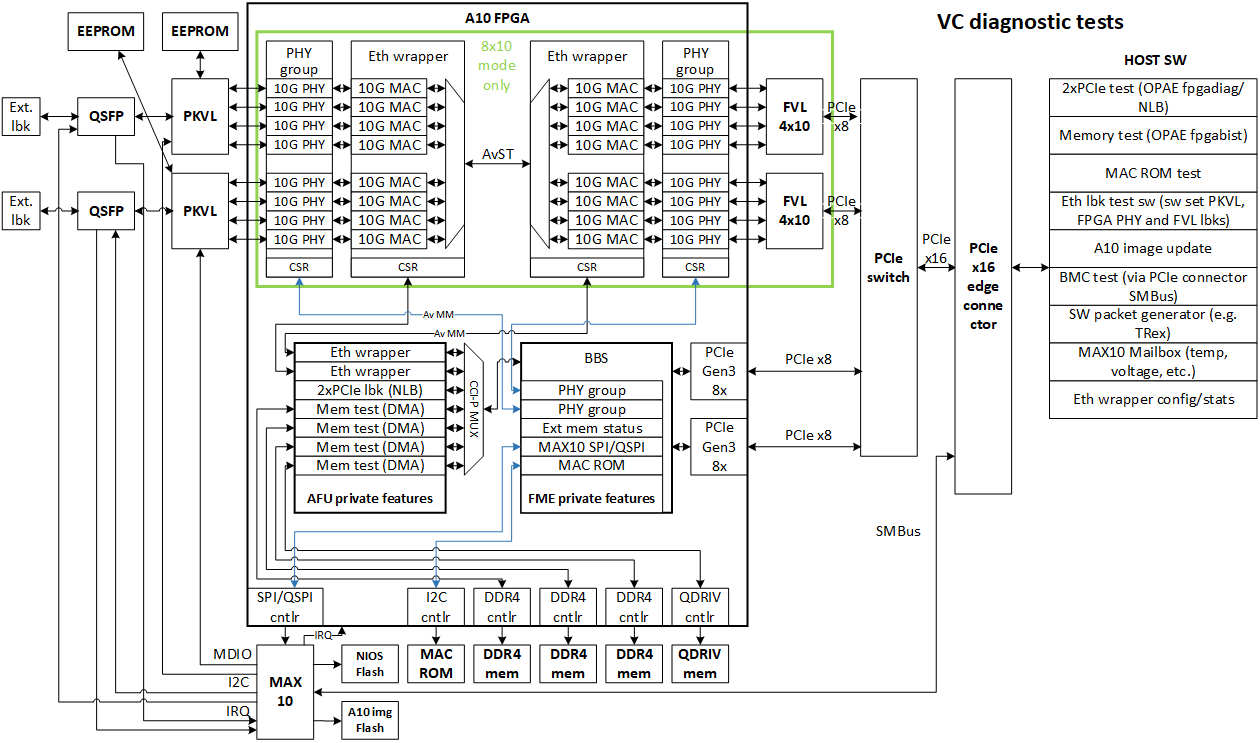
# Overview

## Hardware Overview

Intel® Vista Creek (VC) Programmable Acceleration Card (PAC) is a combination of the powerful ASIC NIC and FPGA technology connected as a “bump-in-the wire”. Below is a simple diagram for the HW components on VC PAC. Most of VC diagnostic tests shown on the left side will be performed through OPAE tools.

Currently there are two types of VC card, one is 8x10G, the other is 2x1x25G.

* 8x10G block diagram of VC



* 2x1x25G block diagram of VC



\*AFU - Accelerator Function Unit

## Software Overview

OPAE tools are user space application based on Linux kernel driver.

OPAE tools

OPAE kernel driver

sysfs

NVMEM dev

MTD dev

OPAE Lib

Vista Creek FPGA PCIe device

User Space

Kernel Space

hardware

char dev

VC applicable tools are list below. Although these tools are already existing, some of them need modification to meet VC board’s requirement.

### **fpgainfo**

It depends on

1) *sysfs* to read FPGA and MAX10 information

2) *char dev* to get PHY group information

3) *NVMEM dev* to read MAC address from ROM

### **fpgabist**

It depends on

1) *OPAE Lib* to find AFU, access MMIO registers, allocate DMA buffer

### **fpgadiag**

It depends on

1) *OPAE Lib* to find AFU, access MMIO registers, allocate DMA buffer

2) *NVMEM dev* to read MAC address from ROM

3) OPAE Python bindings(opae.fpga library) to get fpga mac statistics.

### **fpgaflash**

It depends on

1) *MTD dev* to write image or firmware to Flash

### **fpgaconf**

It depends on

1) *OPAE Lib* to find FIM, do PR

### **fpgaport**

It depends on

1) *char dev* to configure port virtualization

### **mmlink**

It depends on

1) *OPAE Lib* to find STP, access MMIO registers

### **fpgad**

It depends on

1) *OPAE Lib* to find AFU, access MMIO registers, access device sensors

# Functional Description

OPAE tools for Vista Creek are mainly used for below testing purpose.

## Information Display

Show information about FME, Port, errors, power, temperature, MAC and PHY.

This function is performed by **fpgainfo** tool.

## PCIe Loopback Test

NLB AFU read data from source buffer in host memory, then write data back to destination buffer in host memory, software compare the data between source and destination buffer.

This function can be performed by **fpgadiag** or **fpgabist** tool.

**Note**, fpgadiag tests specified PCIe interface configured by argument, fpgabist will automatically test two PCIe interfaces one by one without configuration.

## Local Memory Test

Step1. DMA AFU copy data from buffer in host memory to buffer1 in local memory, software clear buffer in host memory, then DMA AFU copy data from buffer1 in local memory to buffer in host memory, software verify the buffer data in host memory.

Step2. DMA AFU copy data from buffer1 in local memory to buffer2 in local memory, software clear buffer in host memory, then DMA AFU copy data from buffer2 in local memory to buffer in host memory, software verify the buffer data in host memory.

This function is performed by **fpgabist** tool.

## Network Loopback Test

Enable the loopback in FPGA PHY, use network test equipment or packet generator tool to send packets to specific port and check receive count manually.

This function is performed by **fpagdiag** tool.

**Note**, FVL loopback is implemented in i40e driver, it’s not in OPAE scope. The test data is generated by network test equipment or packet generator tool.

## MAC ROM Test

Read MAC address stored in ROM and compare them to MAC addresses stored in FVL flash.

This function is performed by **fpagdiag** tool.

## Mailbox Test

Read information from MAX10 by mailbox.

This function is performed by **fpagdiag** tool.

## FPGA Image and NIOS Firmware Update

Write image/firmware to corresponding Flash.

This function is performed by **fpgaflash** tool.

## FVL Bypass Mode Test

The FVL bypass image in 8x10G Ethernet configuration is depicted in the figure below.

A PRBS traffic generator/checker is integrated for test purpose. Tool write its registers to make it send ethernet packets via specified MAC and read its registers to check if the number of successful received packets is equal to the number of transmitted packets.

**Note**, to make the test work, ethernet interfaces must be loopbacked through external loopback table or internal loopback configuration.

This function is performed by **fpgadiag** tool.



## Graceful Shutdown

Detect and respond to power/temperature anomalies.

This function is performed by **fpgad** in conjunction with the OPAE kernel driver.

## FEC mode configuration

Configure FEC mode of externl ethernet PHY. There are three type of FEC mode can be configured:

“no” – No FEC

“kr” – BaseR FEC (Fire-Code) correction – 4 orders

“rs” – Reed-Solomon FEC correction – 7 orders

User can pass FEC mode configuration via intel-fpga-fme module parameter fec\_mode, e.g. modprobe intel-fpga-fme fec\_mode="no" .

As driver modules will be loaded automatically during system boot, it can't pass module parameters manually in that case, so user has to add below line to file /etc/modprobe.d/intel-fpga-fme.conf (suppose fec mode is “kr”),

options intel-fpga-fme fec\_mode="kr"

then modprobe will use above specified module parameters automatically when loading the module during system boot.

**Note**, even user can pass fec\_mode module parameter to intel-fpga-fme driver module, but the real mode in hardware could be different. The major reason is FPGA can only do one time configuration, any value passed to FPGA later is not accepted, in order to re-trigger FPGA initialization flow, user must power cycle the board or reboot the card.

Tool will modify intel-fpga-fme.conf, then reload intel-fpga-fme module with new parameter, at last reboot the card to make FEC mode effective.

This function is performed by **fecmode** tool in conjunction with the OPAE kernel driver.

# Guide

## fpgainfo

fpgainfo displays FPGA information derived from sysfs files. The command argument is one of the following: errors, power, temp, mac, phy, port or fme. Some commands may also have other arguments or options that control their behavior.

For systems with multiple FPGA devices, you can specify the BDF to limit the output to the FPGA resource with the corresponding PCIe configuration. If not specified, information displays for all resources for the given command.

### synopsis

**fpgainfo** **<***command***>** [**<***args***>**]

### description

|  |  |  |
| --- | --- | --- |
| command | args | description |
|  | --help, -h | prints help information and exit |
|  | --bus, -B | PCIe bus number of resource |
|  | --device, -D | PCIe device number of resource |
|  | --function, -F | PCIe function number of resource |
| errors fme | --clear, -c | show/clear errors of FME |
| errors port | --clear, -c | show/clear errors of port |
| errors all | --clear, -c | show/clear errors of both FME and port |
| power |  | show total power in watts that the FPGA hardware consumes |
| temp |  | show FPGA temperature values in degrees Celsius |
| port |  | show information about the port |
| fme |  | show information about the FME |
| bmc |  | show BMC sensors information |
| mac |  | show information about MAC ROM connected to FPGA |
| phy | --group, -G | show information about ethernet PHYs in FPGA |

### example

This command shows the FME information of FPGA on bus 0x28:

**fpgainfo** --bus 0x28 fme

This command shows the port information of FPGA on bus 0x85:

**fpgainfo** -B 0x85 port

This command shows the error information of FME and port of all FPGAs on PCIe bus:

**fpgainfo** errors all

This command shows temperature information of FPGA on bus 0x85:

**fpgainfo** --bus 0x85 temp

This command shows PHY group information of FPGA on bus 0xbe:

**fpgainfo** -B 0xbe phy

## fpgadiag

Perform several tests to diagnose, test, and report on the FPGA hardware.

lpbk1, read, write, trput, loopback and mactest can be set in mode argument to choose different test to run.

If there are multiple devices, use -B, -D, -F to specify the BDF for the specific device.

Note: fpgastats requires opae.fpga library, you can install it by *pip install opae.fpga*

### synopsis

**fpgadiag** [-m | --mode <*mode*>] [<*args*>]

### description

|  |  |  |
| --- | --- | --- |
| mode | args | description |
|  | --help, -h | prints help information and exit |
|  | --target, -t | specifies fpga (hardware) or ase (simulation) |
|  | --socket, -S | socket ID encoded in FPGA Interface Manager (FIM) |
|  | --bus, -B | bus number of the PCIe device |
|  | --device, -D | device number of the PCIe device |
|  | --function, -F | function number of the PCIe device |
|  | --dsm-timeout-usec | timeout in microseconds for test completion. The test fails if not completed by specified timeout  default=1000000 |
|  | --freq, -F | clock frequency (in Hz) used for bandwidth calculation  default=400000000 Hz (400 MHz) |
|  | --csv, -V | comma separated value format for text output |
|  | --suppress-hdr | suppress column headers for text output |
|  | --suppress-stats | suppress statistics output at the end of test |
|  | --guid, -G | AFU ID to enumerate |
|  | --begin, -b | index of first cache lines (1 ~ 65535)  default=1 |
|  | --end, -e | index of last cache lines (1 ~ 65535)  default=1 |
|  | --multi-cl, -u | number of cache line per read/write  default=1 |
|  | --timeout-usec  --timeout-msec  --timeout-sec  --timeout-min  --timeout-hour | timeout for --cont option  default=0 |
|  | --cont, -L | continuous test until time out |
| lpbk1 | --cache-policy=, -p | can be wrline-I, wrline-M, or wrpush-I  default=wrline-M |
| --cache-hint, -i | can be rdline-I or rdline-S  default=rdline-I |
| --read\_vc, -r | can be auto, vl0, vh0, vh1, random  default=auto |
| --write\_vc, -w | can be auto, vl0, vh0, vh1, random  default=auto |
| --wrfence-vc=, -f | can be auto, vl0, vh0, vh1  default=auto |
| --id, -I | NLB0 ID to enumerate, default is D8424DC4-A4A3-C413-F89E-433683F9040B |
| fpgalpbk | --side | can be line, host |
| --port | 0 ~ 7, all  default=all |
| --direction | can be remote, local |
| --type | Only support serial mode |
| --enable | enable loopback |
| --disable | disable loopback |
| fpgastats |  | print fpga mac statistics |
| mactest | --offset | read mac address from an offset at nvmem  default=0 |
| fvlbypass | --number, -n | number of the test packets to send per MAC  default=00000 |
| --length, -s | length of each test packet  default=128 |
| --loopback, -l | configure internal loopback automatically  Loopback is not configured by default |
| --clear, -c | clear statistics automatically  statistics are not cleared by default |
| --port, -p | select test port, format: port number or range seperated by space. port range is specified by two port numbers with dash between them. default=all |
| --debug, -d | display debug information during test |

### example

This command starts a lpbk1 test for the FPGA on bus 0x5e. The test copies from 8th to 15th cache lines, one line at a time:

**fpgadiag**  --bus 0x5e --mode lpbk1 --begin 8 --end 15

This command will do mac compare test, script will find Fortville interfaces automatically, and read MACs from nvmem at offset 0, and compare MACs between nvmem and Fortville interfaces:

**fpgadiag** -B 0x28 -m mactest

This command will do mac compare test, script will find Fortville interfaces automatically, and read MACs at offset 80, and compare MACs between nvmem and Fortville interfaces:

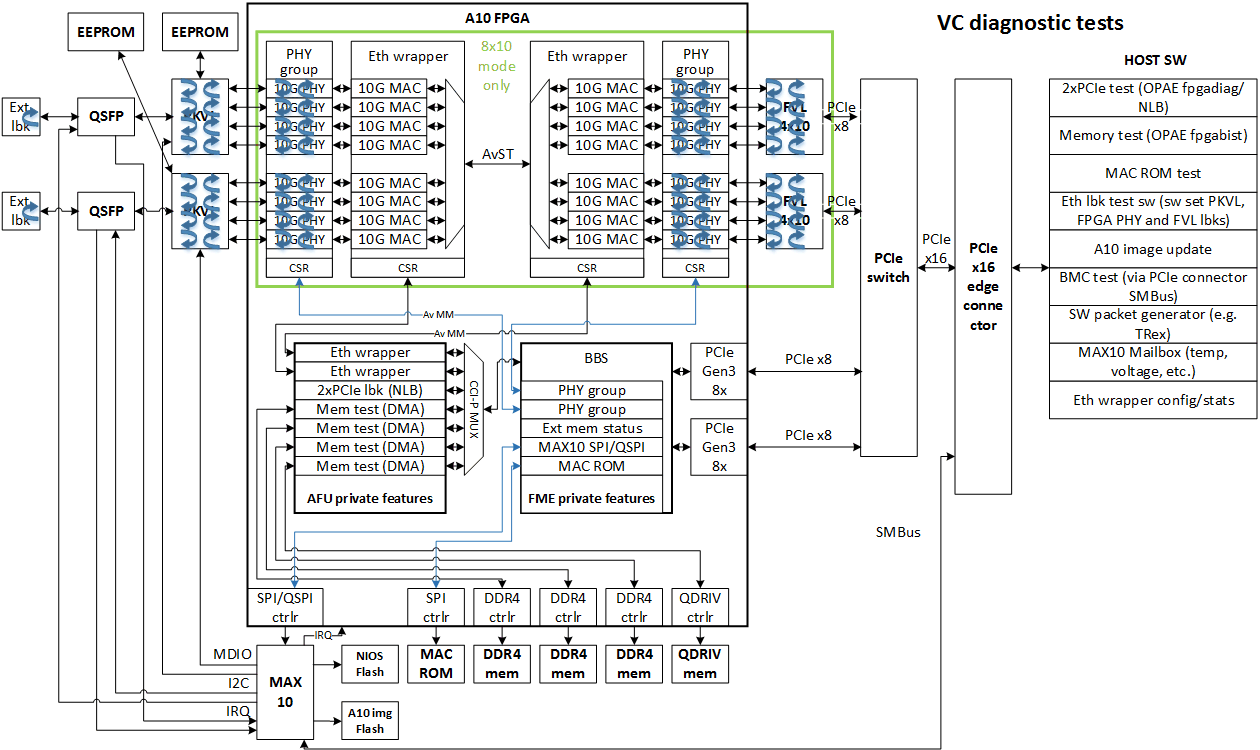
**fpgadiag** --bus 0x28 -mode mactest --offset 80

This command will do fvl bypass mode test, script will use packet generator in the FPGA to send test packet, the ports are loopbacked by software:

**fpgadiag** --bus 0x28 -mode fvlbypass --loopback

### network loopback guide

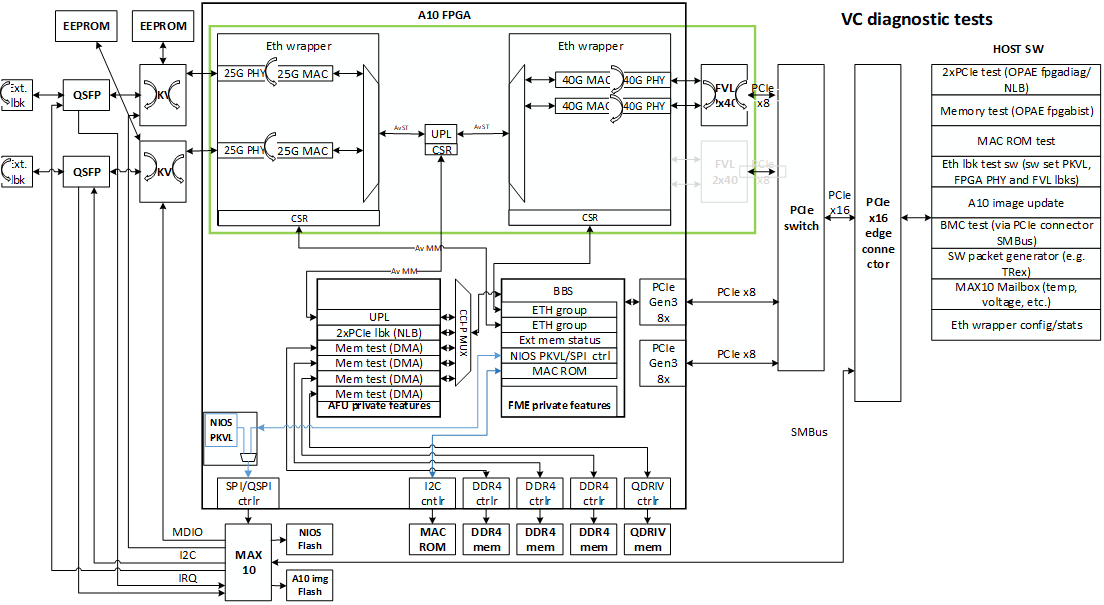
* 8x10G loopback modes of VC



In A10 FPGA, there are 4 loopback modes can be setup by fpgadiag tool in 8x10G VC card, they are called as below (from line side to host side):

1. FPGA line side remote loopback
2. FPGA line side local loopback
3. FPGA host side remote loopback
4. FPGA host side local loopback

* 2x1x25G loopback mode of VC



In A10 FPGA, there are 2 loopback modes can be setup by fpgadiag tool in 2x1x25G VC card, they are called as below (from line side to host side):

1. FPGA line side local loopback
2. FPGA host side remote loopback

**Note**, other loopbacks out of A10 FPGA in VC card can be setup by fiber, lanconf or other tools.

#### **Loopback support table by fpgadiag tool**

|  |  |  |
| --- | --- | --- |
|  | 8x10G | 2x1x25G |
| FPGA line side remote loopback | √ | N/A |
| FPGA line side local loopback | √ | √ |
| FPGA host side remote loopback | √ | √ |
| FPGA host side local loopback | √ | N/A |

#### **FPGA line side remote loopback example**

This command will enable remote loopback in all line side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side line --direction remote --type postcdr --enable

This command will disable remote loopback in all line side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side line --direction remote --type postcdr --disable

#### **FPGA line side local loopback example**

This command will enable local loopback in all line side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side line --direction local --enable

This command will disable local loopback in all line side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side line --direction local --disable

#### **FPGA host side remote loopback example**

This command will enable remote loopback in all host side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side host --direction remote --enable

This command will disable remote loopback in all host side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side host --direction remote –disable

#### **FPGA host side local loopback example**

This command will enable local loopback in all host side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side host --direction local --type precdr --enable

This command will disable local loopback in all host side ports of FPGA on bus 0x85:

**fpgadiag** -B 0x85 -m fpgalpbk --side host --direction local --type precdr --disable

## fpgabist

The fpgabist tool performs self-diagnostic tests on FPGA platforms.

The tool depends on the available AFUs integrated in FPGA shell BBS and runs appropriate tests and reports hardware issues.

fpgabist always uses fpgainfo to report system information before running any hardware tests.

If there are multiple devices, use -B, -D, -F to specify the BDF for the specific device.

### synopsis

**fpgabist** <*args*>

### description

|  |  |
| --- | --- |
| args | description |
| --help, -h | prints usage information |
| --device-id, -i | device ID for Intel FPGA |
| --bus, -B | bus number for specific FPGA |
| --device, -D | device number for specific FPGA |
| --function, -F | function number for specific FPGA |
| path\_to\_gbs ... | paths to GBS files of AFUs being used  not used in Vista Creek |

### example

This command runs fpgabist for FPGA on bus 0x85 with device ID 0x0b30:

**fpgabist** -B 0x85 -i=0x0b30

## fpgaflash

fpgaflash updates the static FIM image loaded from flash at power-on, and can also update the MAX10, NIOS firmware and EEPROM. Note, you need not power cycle the machine for loading A10 or MAX10 from user image with rsu enabled.

Once the data verification fails after programming A10 or MAX10, tool will retry the programming progress. If verification still fails, tool will erase the programming area to make it possible to load from factory image after power cycle.

If there are multiple devices in the system, fpgaflash must specify a BDF to select the correct device. If no BDF is specified, fpgaflash prints out the BDFs of any compatible devices.

### synopsis

**fpgaflash** {factory, factory\_only, user, rsu, bmc\_fw, bmc\_img, bmc\_factory, eeprom, dtb, phy\_eeprom} <file> [<*args*>]

### description

{factory, factory\_only, user, rsu, bmc\_fw, bmc\_img, bmc\_factory, dtb, eeprom, phy\_eeprom} - selects which type of image to program.

‘factory’ means programming both A10 factory and user image with option bits and dtb. Note, this operation will program both factory and user flash, it is risky, you have to recover the card through JTAG if programming failed. So, it should only be used by internal user. Moreover, please use the latest vc\_image\_convert tool to generate binary image for factory programming.

‘factory\_only’ means programming factory A10 image.

‘user’ means programming customer A10 image.

‘rsu’ reboot the board without programming flash. Note, although you still need to specify a file with it, but the file is not used

‘bmc\_fw’ means programming NIOS firmware image.

‘bmc\_img’ means programming customer MAX10 image.

‘bmc\_factory’ means programming factory MAX10 image.

‘phy\_eeprom’ means programming Parkvale EEPROM image.

‘dtb’ means update device tree.

‘eeprom’ means programming eeprom image.

<file> - specifies the Binary File (**bin**) to program into FPGA flash, dtb and eeprom.

specifies Intel HEX File (**ihex**) to program into NIOS firmware FLASH and pkvl.

specifies Raw Programming Data File (**rpd**) to program into MAX10 internal FLASH.

|  |  |
| --- | --- |
| args |  |
| --help, -h | print usage information |
| --rsu, -r | remote system upgrade feature, reload A10 or MAX10 without power cycle  Note, the PCIe bus number of FPGA may change after RSU, please check it immediately |
| --no-verify, -n | skip data verification after progamming flash |
| bdf | specifies the bus, device and function (BDF) of device to program such as 09:00.0 or 0000:25:00.0.  optional when there is a single device in the system |

### example

This command programs customer image included in a10\_flash.bin to user bank of flash on BDF 0000:09:00.0:

**fpgaflash** user a10\_flash.bin 0000:09:00.0 --rsu

This command programs whole a10\_flash.bin to flash on BDF 0000:09:00.0:

**fpgaflash** factory a10\_flash.bin 0000:09:00.0 --rsu

This command programs nios\_fw.ihex to NIOS flash on BDF 25:00.0:

**fpgaflash** bmc\_fw nios\_fw.ihex 25:00.0

This command update Parkvale EEPROM on BDF 25:00.0:

**fpgaflash** phy\_eeprom nios\_fw\_with\_pkvl\_eeprom.ihex 25:00.0

This command programs customer image max10\_system\_dual\_v1.0.6\_cfm0\_auto.rpd to MAX10 internal flash on BDF 25:00.0:

**fpgaflash** bmc\_img max10\_system\_dual\_v1.0.6\_cfm0\_auto.rpd 25:00.0 -r

This command programs factory image max10\_system\_dual\_v1.0.6\_cfm1\_auto.rpd to MAX10 internal flash on BDF 25:00.0:

**fpgaflash** bmc\_factory max10\_system\_dual\_v1.0.6\_cfm1\_auto.rpd 25:00.0 -r

This command programs device tree blob included in a10\_flash.bin to DTB bank of flash on BDF 85:00.0:

**fpgaflash** dtb a10\_flash.bin 85:00.0

This command programs mac\_rom.bin to eeprom connected to FPGA on BDF 25:00.0:

**fpgaflash** eeprom mac\_rom.bin 25:00.0

This command reboot the card which includes FPGA on BDF 25:00.0:

**fpgaflash** rsu /dev/null 25:00.0

### procedure

#### **Flash A10 image**

The flash has two partitions. One partition stores the factory or golden image. The other partition stores the user image. If the user image is corrupt, the Intel® Arria® 10 FPGA automatically uses the factory image.

0x0000000

golden image

user image

0x0020000

0x4000000

Above is the flash layout. You can flash factory or user image partition by fpgaflash tool.

After executing ‘fpgaflash user *xxx.bin* --rsu’ command you should see below output information:

/dev/mtd0

2019-03-26 10:44:29.912422 reversing bits

2019-03-26 10:44:41.425831 erasing 0x03800000 bytes starting at 0x04000000

2019-03-26 10:48:01.853762 writing 0x03800000 bytes to 0x04000000

2019-03-26 11:02:53.040423 actual bytes written 0x6a66000 - 0x4000000 = 0x2a66000

2019-03-26 11:02:53.042020 reading 0x02a66000 bytes from 0x04000000

2019-03-26 11:05:39.890107 verifying flash

2019-03-26 11:05:39.917383 flash successfully verified

2019-03-26 11:05:39.917974 performing remote system update

2019-03-26 11:05:40.686927 waiting for FPGA reconfiguration

2019-03-26 11:05:53.930483 pci bus rescanned

Note, last three lines are only output when --rsu option is used

After executing ‘fpgaflash factory *xxx.bin* --rsu’ command you should see below output information:

/dev/mtd0

2019-03-26 11:10:35.068505 reversing bits

2019-03-26 11:10:59.501084 erasing 0x07ff0000 bytes starting at 0x00010000

2019-03-26 11:17:41.779800 writing 0x077f0000 bytes to 0x00010000

2019-03-26 11:42:10.964524 actual bytes written 0x6a66000 - 0x10000 = 0x6a56000

2019-03-26 11:42:10.966157 reading 0x06a56000 bytes from 0x00010000

2019-03-26 11:47:49.431856 verifying flash

2019-03-26 11:47:49.499118 flash successfully verified

2019-03-26 11:47:49.499668 performing remote system update

2019-03-26 11:47:50.268013 waiting for FPGA reconfiguration

2019-03-26 11:48:03.576362 pci bus rescanned

Note, last three lines are only output when --rsu option is used

The flash erase, write, and verify process takes about 20 minutes for user image and about 40 minutes for factory image update to complete. If you have multiple Intel® PAC cards installed, you can specify the bus, device, and function (BDF) for the card to update using the following command. To find the BDF for your card, type the following command

lspci | grep 0b30

sample output:

09:00.0 Processing accelerators: Intel Corporation Device 0b30

So the BDF = 09:00.0, you can input the command like ‘fpgaflash user *xxx.bin* 09:00.0’.

You can check the fpga is loaded from factory or user image by below command

fpgainfo fme

‘Boot Page’ tells fpga is loaded from which image.

#### Flash NIOS firmware

The file used for fpgaflash is converted from orignal binary firmware file, please use below command to convert it.

objcopy --input-target=binary --output-target=ihex vista\_creek\_qspi\_xip\_v1.0.1.bin vista\_creek\_qspi\_xip\_v1.0.1.ihex

After executing ‘fpgaflash bmc\_fw *xxx.ihex*’ command you should see below output information:

2019-03-26 13:04:02.395204 erasing 0x00010000 bytes starting at 0x00000000

2019-03-26 13:04:03.471073 erasing 0x00010000 bytes starting at 0x00010000

2019-03-26 13:04:04.535096 writing 0x000160b0 bytes to 0x00000100

2019-03-26 13:04:05.947005 writing 0x00000100 bytes to 0x00000000

2019-03-26 13:04:06.257299 flash successfully verified

The flash erase, write, and verify process takes several seconds to complete. If you have multiple Intel® PAC cards installed, you can specify the bus, device, and function (BDF) for the card to update using the following command. To find the BDF for your card, type the following command

lspci | grep 0b30

sample output:

25:00.0 Processing accelerators: Intel Corporation Device 0b30

So the BDF = 25:00.0, you can input the command like ‘fpgaflash bmc\_fw *xxx.ihex* 25:00.0’.

#### **Flash MAX10 image**

The MAX10 internal flash also has two partitions. One partition stores the factory image. The other partition stores the user image. You can flash factory or user image partition by fpgaflash tool.

After executing ‘fpgaflash bmc\_img *xxx.rpd* --rsu’ command you should see below output information:

/dev/mtd0

2019-03-26 13:06:25.593829 reversing bits

2019-03-26 13:06:25.744230 erasing 0x000a8000 bytes starting at 0x000b8000

2019-03-26 13:06:25.750200 writing 0x000a8000 bytes to 0x000b8000

2019-03-26 13:07:22.528543 actual bytes written 0x125000 - 0xb8000 = 0x6d000

2019-03-26 13:07:22.528734 reading 0x0006d000 bytes from 0x000b8000

2019-03-26 13:07:23.918440 verifying flash

2019-03-26 13:07:23.918862 flash successfully verified

2019-03-26 13:07:23.919204 performing remote system update

2019-03-26 13:07:24.671911 waiting for FPGA reconfiguration

2019-03-26 13:07:38.052987 pci bus rescanned

Note, last three lines are only output when --rsu option is used

After executing ‘fpgaflash bmc\_factory *xxx.rpd* --rsu’ command you should see below output information:

/dev/mtd0

2019-03-26 13:10:29.898551 erasing 0x00048000 bytes starting at 0x00070000

2019-03-26 13:10:29.903046 reversing bits

2019-03-26 13:10:30.048621 erasing 0x00060000 bytes starting at 0x00010000

2019-03-26 13:10:30.053164 writing 0x000a8000 bytes to 0x00010000

2019-03-26 13:11:27.113585 actual bytes written 0x7d000 - 0x10000 = 0x6d000

2019-03-26 13:11:27.113806 reading 0x0006d000 bytes from 0x00010000

2019-03-26 13:11:28.502216 verifying flash

2019-03-26 13:11:28.502659 flash successfully verified

2019-03-26 13:11:28.503046 performing remote system update

2019-03-26 13:11:29.253953 waiting for FPGA reconfiguration

2019-03-26 13:11:42.541209 pci bus rescanned

Note, last three lines are only output when --rsu option is used

The flash erase, write, and verify process takes about 1 minutes to complete. If you have multiple Intel® PAC cards installed, you can specify the bus, device, and function (BDF) for the card to update using the following command. To find the BDF for your card, type the following command

lspci | grep 0b30

sample output:

24:00.0 Processing accelerators: Intel Corporation Device 0b30

So the BDF = 24:00.0, you can input the command like ‘fpgaflash bmc\_img *xxx.rpd* 24:00.0’.

You can check whether the fpga is loaded from user image by below command

fpgainfo fme

If the numerical value of ‘MAX10 Build version’ is 1.0.x, it indicates that MAX10 is loaded from user image.

If the numerical value of ‘MAX10 Build version’ is 255.255.x, it indicates that MAX10 is loaded from factory image.

## fpgaconf

fpgaconf configures the FPGA with the accelerator function unit (AFU). It also checks the AFU for compatibility with the targeted FPGA and the FPGA Interface Manager (FIM).

fpgaconf enumerates available FPGA devices in the system and selects compatible FPGAs for configuration. If more than one FPGA is compatible with the AFU, fpgaconf exits and asks you to be more specific in selecting the target FPGAs by specifying a socket number or a PCIe BDF.

### synopsis

**fpgaconf** <gbs> [<args>]

### description

<gbs> - specifies the green bit-stream file to program into flash

|  |  |
| --- | --- |
| args | description |
| --help, -h | prints usage information |
| --verbose, -v | prints more verbose messages while enumerating and configuring |
| --dry-run, -n | performs enumeration, skips any operations with side-effects such as the actual AFU configuration |
| --bus, -B | PCIe bus number of the target FPGA |
| --device, -D | PCIe device number of the target FPGA |
| --function, -F | PCIe function number of the target FPGA |
| --socket, -S | Socket number of the target FPGA |
| --force | don't try to open accelerator resource |

### example

This command programs "my\_afu.gbs" to a compatible FPGA:

**fpgaconf** my\_afu.gbs

## fpgaport

fpgaport enables and disables virtualization. It assigns and releases control of the port to the virtual function (VF). By default, the driver assigns the port to the physical function (PF) in the non-virtualization use case.

### synopsis

**fpgaport** {assign,release} <device> <port> [{--help,-h}]

### description

{assign,release} - assigns or releases control of the port to the physical function

<device> - FPGA device being targeted, such as ‘/dev/intel-fpga-fme.1’

<port> - specifies the number of the port

{--help,-h} - shows help message and exit

### example

This command assigns port 0 to physical function control

**fpgaport** assign /dev/intel-fpga-fme.0 0

This command release port 1 from physical function control

**fpgaport** release /dev/intel-fpga-fme.0 1

## mmlink

The Remote Signal Tap logic analyzer provides real-time hardware debugging for the Accelerator Function Unit (AFU). It provides a signal trace capability that the Quartus Prime software adds to the AFU. The Remote Signal Tap logic analyzer provides access to the Remote Signal Tap part of the Port MMIO space and then runs the remote protocol.

### synopsis

**mmlink** [<*args*>]

### description

|  |  |
| --- | --- |
| args | description |
| --bus, -B | PCIe bus number of the target FPGA |
| --device, -D | PCIe device number of the target FPGA |
| --function, -F | PCIe function number of the target FPGA |
| --socket, -S | Socket number of the target FPGA |
| --port, -P | TCP port number |
| --ip, -I | IP address of FPGA system |

### example

This command starts and listens for remote connection on signal tap MMIO space.

**mmlink** -B 0x5e -P 3333

## fpgad

fpgad monitors the device sensors, checking for sensor values that are out of the prescribed range. When any of the sensors is detected to be out of bounds, fpgad responds by removing the OPAE kernel driver for a configurable cool-down period.

When it is being removed, the OPAE kernel driver’s shutdown path will signal any application processes that are actively using an AFU to exit, and it will reset the hardware devices, allowing them to reach a quiescent state.

Once the cool-down period has expired, fpgad will force a PCIe bus rescan, reloading the OPAE kernel driver to resume normal operation. Any application processes that were forced to exit by the OPAE kernel driver will need to be manually restarted.

For systems with multiple FPGA devices, fpgad will monitor the sensors of each device.

**Note:** fpgad must be running (as root) and actively monitoring devices when a sensor anomaly occurs in order to initiate Graceful Shutdown. If fpgad is not loaded during such a sensor anomaly, the out-of-bounds scenario will not be detected, and the resulting effect on the hardware is undefined.

### synopsis

**fpgad** [**<***args***>**]

### description

|  |  |
| --- | --- |
| args | description |
| --help, -h | prints help information and exits |
| --daemon, -d | fpgad runs as a system daemon process |
| --logfile, -l | specify the log file name [default is fpgad.log] |
| --pidfile, -p | specify the pid file name [default is fpgad.pid] |
| --socket, -s | specify the name of the OPAE Events API socket [default is /tmp/fpga\_event\_socket] |
| --null-bitstream, -n | specify the path to the NULL GBS to be loaded on power overage.  not used in Vista Creek |
| --config, -c | specify location of fpgad configuration file.  [default is /var/lib/opae/fpgad.cfg] |

### example

This command starts fpgad as a system daemon process:

**sudo systemctl start fpgad**

### defining sensor overrides in fpgad.cfg

fpgad periodically monitors each of the Board Management Controller's on-board sensors. If a sensor supports a high-warn or low-warn threshold and that threshold is met, fpgad responds by disabling AER for the PAC. AER is disabled in order to avoid a system reset in the case that a sensor values reaches the high-fatal or low-fatal threshold. When high-fatal or low-fatal is reached, the Board Management Controller removes power from the PAC in order to avoid damage. When power is removed from the PAC, the kernel experiences a surprise device removal and responds by resetting the system. Disabling AER prior to the Board Management Controller's powering down the PAC avoids the surprise device removal and subsequent system reset.

fpgad's configuration file provides a mechanism for the user to specify additional sensor monitoring in the case that the Board Management Controller does not provide high-warn or low-warn thresholds for the sensor. To enable this feature, set the configuration file's "config-sensors-enabled" key to true, and specify the desired sensor and thresholds.

The following JSON syntax creates a sensor override for sensor25:

"config-sensors-enabled": true,

"sensors": [

{

"id": 25,

"low-warn": 11.40,

"low-fatal": 10.56

}

]

## fecmode

fecmode changes FEC mode of external ethernet PHY.

### synopsis

**fecmode** [*<mode>*][**<***args***>**]

### description

|  |  |  |
| --- | --- | --- |
| mode | args | description |
|  | --help, -h | prints help information and exit |
|  | --segment, -S | segment number of the PCIe devie |
|  | --bus, -B | bus number of the PCIe device |
|  | --device, -D | device number of the PCIe device |
|  | --function, -F | function number of the PCIe device |
|  | --rsu, -r | reboot card only if mode is not configured |
|  | --debug, -d | output debug information |
| no |  | no FEC |
| kr |  | BaseR FEC (Fire-Code) correction – 4 orders |
| rs |  | Reed-Solomon FEC correction – 7 orders |

### example

This command change FEC mode to “kr”:

**fecmode -B 0x25 kr**

This command reboot card (no need to specify bus number if there is only one card):

**fecmode -r**

This command display the current FEC mode:

**fecmode**