GRAPHCORE

Graphcore Command Line Tools

Version latest

Graphcore Ltd

Nov 25, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

The Poplar SDK includes a number of software tools that provide information on the current status of the connected hardware.

These tools are in the `bin` directory of the Poplar SDK installation, typically `poplar-os_name-version/bin`. To set up your environment to use the tools, you will need to source the `enable.sh` script provided in that directory:

```
$ source enable.sh
```

Important: If the `http_proxy` and/or `https_proxy` environment variables have been set then you need to add the IP addresses for the IPU-M2000s to the `no_proxy` environment variable in order to allow the tools to access them.

For example, if the subnet address of an IPU-POD is **x.y.z**, you could set the `no_proxy` environment variable using the following command:

```
$ export no_proxy="$(for k in {1..8}; do echo x.y.z.$k ; done | tr '\n' ',')"
```

The commands available are listed below.

gc-docker Allows you to use PCIe-based IPU devices in Docker containers.

gc-info Determines what IPU cards are present in the system.

gc-inventory Lists device IDs, physical parameters and firmware version numbers.

gc-links Displays the status and connectivity of each of the IPU-Links that connect IPUs. See also *IPU-Link channel mapping* for connectivity in an IPU Server containing C2 cards.

gc-monitor Monitors IPU activity on shared systems.

gc-reset Resets IPU devices.

gc-exchangetest Allows you to test the internal exchange fabric in an IPU.

gc-hostsynclatencytest Reports the latency of transfers between the host machine and the IPUs (in both directions).

gc-hosttraffictest Allows you to test the data transfer between the host machine and the IPUs (in both directions).

gc-iputraffictest Allows you to test the data transfer between IPUS.

gc-memorytest Tests all the memory in an IPU, reporting any tiles that fail.

gc-powertest Tests power consumption and temperature of the IPU processors.

gc-exchangewritetest Tests direct writes to the IPU's tile memory via the host.

gc-gwlinkstraffictest Tests GW-Links on multi-rack IPU-POD systems.

gc-flops Allows you to benchmark the number of floating point operations per second on one or more IPU processors.

Some of these commands reference IPU devices by their ID. The ID mapping is shown in *C2 PCIe Device IDs and channel map*.

The use and output of each of these tools is described in the following sections.

CHAPTER

TWO

GC-DOCKER

This tool generates IPU device definitions for Docker, so the IPU devices can be used inside a container. It can be used either to start a container with `docker run` or to display the Docker command that would be executed.

2.1 Start a container with IPU devices

The default action is to start a Docker container. All options and arguments specified after `--` are passed directly to `docker run`.

Here are some examples where `{docker_opts}` are options passed to `docker run`:

```
$ gc-docker -- {docker_opts} # start a container with all IPU devices
$ gc-docker --device-id {id} -- {docker_opts} # start a container with specified device
$ gc-docker --binary {docker_bin} -- {docker_opts} # specify path to the docker binary
```

You can specify multiple `--device-id` options. You can find out the available device IDs with `gc-info` command. If the device ID refers to multiple IPU devices, all the IPUs that are part of that ID will be available in the container.

When you select a subset of device IDs, please note that inside the container the device ID sequence always starts from 0.

2.2 Show Docker command

The `--echo` option displays the Docker command with IPU device definitions.

An example:

```
$ gc-docker --echo -- {docker_opts}  # display docker command and don't start container
```

2.3 Usage

2.3.1 Commands

`-e, --echo`	Display docker command and don't start a container
`-h, --help`	Produce help message
`--version`	Version number

2.3.2 Command options

`-d {arg}, --device-id {arg}`	Device id
`-b {arg}, --binary {arg}`	Docker binary (default: /usr/bin/docker)
`--command`	arg

2.3.3 Examples

```
gc-docker -- {dockers_opts}                    # start a container with all available IPU devices
gc-docker --device-id {id} -- {docker_opts}    # start a container with specified IPU device(s)
gc-docker --binary {docker_bin} -- {docker_opts} # specify path to the docker binary
gc-docker --echo -- {docker_opts}              # display docker command and don't start container
```

2.3.4 Notes

- This command generates device definitions for docker run command.
- By default a container is started with 'docker run {docker_opts}'.
- If –echo option is defined, the docker command is displayed.
- You can set the path to docker with –binary option.
- By default '–ipc=host' will be set.
- Everything after '–' is passed directly to docker run.

CHAPTER

THREE

GC-EXCHANGETEST

This tool tests the IPU-Exchange fabric inside an IPU, ensuring that data can move properly between all the tile processors and memory locations.

It can be run as follows:

```
gc-exchangetest -d {device_id}
```

where `{device_id}` is the id number returned by the *gc-inventory* tool.

If the test finds no issues with the IPU-Exchange, then the output looks like this:

```
{
    "result": "pass"
}
```

If the test finds any issues with the IPU-Exchange, the output shows which tiles have failed. For example:

```
{
    "tile_results": [
        {
            "12": "fail"
        },
        {
            "1000": "fail"
        },
        {
            "0": "fail"
        }
    ],
    "result": "fail"
}
```

3.1 Usage

3.1.1 Allowed options

`-d {id}, --device-id {id}`	Device id
`-h, --help`	Produce help message
`-v, --verbose`	Verbose output
`--version`	Version number

CHAPTER

FOUR

GC-EXCHANGEWRITETEST

This tool tests direct writes to the IPU's tile memory via the host.

It can be run as follows:

```
gc-exchangewritetest -d {device_id}
```

where `{device_id}` is the id number returned by the *gc-inventory* tool, including MultiIPU devices.

If the test finds no issues, then the output looks like this:

```
{
    "result": "pass"
}
```

If the test finds any issues with the IPU tile memory writes, the output shows which tiles have failed. For example:

```
{
    "tile_results": [
        {
            "12": "fail"
        },
        {
            "1000": "fail"
        },
        {
            "0": "fail"
        }
    ],
    "affected_ipus": [
      {
          "serial": "0134.0004.918521"
      }
  ],

    "result": "fail"
}
```

4.1 Usage

4.1.1 Allowed options

`-d {id}, --device-id {id}`	Device id
`-h, --help`	Produce help message
`-v, --verbose`	Verbose output
`--version`	Version number

CHAPTER

FIVE

GC-GWLINKSTRAFFICTEST

This tool tests data transfer between IPU-Machines over GW-Links. It sends data from IPUs on one IPU-Machine across the GW-Links to an IPU-Machine on the other side of the link. Receiving IPUs verify the data is correct.

This program is only applicable to IPU-POD systems where IPU-POD racks have been connected by GW-Links.

This program has two completely separate modes of use:

- Standard mode, which tests entire IPU Fabric multi-ILD (IPU Link Domain) partitions.
- Single IPU mode, which provides a low-level mechanism to configure individual IPUs to test specific GW-Link configurations. **This mode is not meant for general use**. It is primarily intended to be used by the V-IPU diagnostic tools.

5.1 Standard mode

To use GW-Links, an IPU-POD system must have been configured with a cluster and non-reconfigurable partition containing at least two IPU Link Domains (ILDs) (Note that GW-Links cannot be used with reconfigurable partitions).

For example, the simplest such configuration would be two IPU-Machines, sitting side by side in racks connected only by GW Links. You could make a cluster and partition containing two ILDs, where each ILD contains one GCD. The V-IPU commands to do this from agents `ag1` and `ag2` would look like:

```
vipu-admin create cluster cl1x2 --agents ag1,ag2 --num-ipulinkdomains 2 --cluster-topology looped
vipu-admin create partition pa1x2 --size 8 --num-gcds 2 --gw-routing ring
```

This will create two IPUoF configuration files for the `pa1x2` partition, one per ILD/GCD. In this example they would be named `pa1x2_gcd0.conf` and `pa1x2_gcd1.conf`.

You can now use `gc-gwlinkstraffictest` to check this configuration is working and measure its performance.

For each ILD/GCD, you must run one instance of `gc-gwlinkstraffictest`. Each instance needs to be configured to attach to the correct partition and ILD. This can be done by setting the `IPUOF_CONFIG_PATH` environment variable to specify the relevant configuration file.

If your system uses a separate headnode per ILD then you would run one instance on the first headnode and the other on the second. Each instance will attach to the multi-IPU device containing all the IPUs in that ILD, and configure those devices to exchange data with devices in the other ILD. You should start both instances at roughly the same time.

```
IPUOF_CONFIG_PATH=/home/ipuuser/.ipuof.conf.d/pa1x2_gcd0.conf gc-gwlinkstraffictest

IPUOF_CONFIG_PATH=/home/ipuuser/.ipuof.conf.d/pa1x2_gcd1.conf gc-gwlinkstraffictest
```

Each instance will finish and report results separately. If the test was successful the results will look something like:

```
$ IPUOF_CONFIG_PATH=/home/justina/.ipuof.conf.d/pa1x2_gcd0.conf gc-gwlinkstraffictest
gwlinks: 0.7188 GiB in 0.08 seconds, 9.54 GiB/s, 81.93 Gbps

$ IPUOF_CONFIG_PATH=/home/justina/.ipuof.conf.d/pa1x2_gcd1.conf gc-gwlinkstraffictest
gwlinks: 0.7188 GiB in 0.08 seconds, 9.53 GiB/s, 81.83 Gbps
```

To get more detailed information use the verbose option, `-v`. Detailed JSON formatted results are available by specifying the `-j` option.

On each IPU-Machine in the partition, data is sent from IPUs #0 and #1 from the first ILD to IPUs #2 and #3 in the corresponding IPU-Machine in the second ILD, and vice versa. By default, IPU #0 will transmit through GW-Link port 1 and IPU #2 will use port 2. This behaviour can be changed with the `-p` option.

To run multiple iterations of the test, specify the `-i` option.

To test that the transfer rate does not fall below a specific threshold, use the `--min-gwlinks-bandwidth` option.

Note, if the message `Warning: SPCR.GSBGS2DIR is not set to SREQ_OUTPUT` is displayed, it means that the sync configuration of the partition is incorrect and the test will most likely fail. This can occur if another application changes the sync configuration from its original state. The `vipu-admin reset partition` command will restore the sync configuration to the correct state.

If an error occurs, the tool will display an error message and set a non-zero exit code.

5.2 Single IPU mode

This mode is *not intended for general use*, as such it will only be described briefly. Single IPU mode is only available on native IPU-Machine hardware and cannot be used over Fabric.

This mode is selected with the `--single-ipu-mode` option. Each IPU in an IPU-Machine must be configured individually with one of the following commands:

- `tx`: Configure the IPU to transmit to a device on the other side of the GW-Link connection. Requires a `--rx-device-id` parameter to specify the target IPU.
- `rx`: Configure the IPU to expect to receive data. One of these devices should be configured as the sync master, by specifying `--master-rx`.
- `dummy`: Take part in the sync mechanism but do not transfer any data.
- `run`: Use the sync mechanism to drive execution of the transmitter and receiver programs. Must be ran on the IPU that is the sync master. All 4 devices must have already been configured in either tx, rx or dummy mode for run to execute correctly.
- `check`: Check the received data is as expected. Must be ran on an IPU that was previous configured as a receiver.

5.3 Usage

CHAPTER

SIX

GC-HOSTSYNCLATENCYTEST

This tool tests the sync latency between host and IPU in microseconds.

To use it, run:

```
gc-hostsynclatencytest -d {device_id} -j
```

where `{device_id}` is the id number returned by the *gc-inventory* tool.

If JSON output is selected, the output will look something like this:

```
{
    "ipu_id": "0",
    "sync_mode": "Polling",
    "hsp": "hsp1",
    "delay_cycles": "5",
    "num_delays": "32",
    "iterations": "1",
    "mark_count": "4095",
    "split_increments": "false",
    "time_to_sync_us": "1485",
    "result": "pass"
}
```

The output will be plain text if the `-j` option is not specified. For example:

```
o Using device Id - 0
o Using sync mode: Polling, HSP: hsp1
  - Issuing 4095 syncs, each followed by 32 delay instructions of 5 cycles
o Loading binary on all tiles... done
o Syncs took on average 1485 us
o Result - Pass
```

If an error occurs during the test, then `gc-hostsynclatencytest` will return a non-zero exit code, and output an error message to the stdout.

More iterations can be added using the `-i` option to average the results to reduce the impact of system process scheduling.

When the host syncs with the IPUs it can set a "mark count". This is decremented every time the IPUs sync. There is a host function that will wait until the mark count hits zero. `gc-hostsynclatencytest` can set the initial value of the mark count and request the IPUs to issue a series of syncs. It then measures the time it takes for the mark count to reach zero.

A larger number of delay cycles, number of delays or value of mark count will result in longer load time onto the tiles of the IPU.

6.1 Usage

6.1.1 Allowed options

`-d {arg}, --device-id {arg}`	IPU device Id to be used
`--which-hsp {arg}`	hsp1|hsp2|both (default: hsp1)
`-i {arg}, --iterations {arg}`	Iterations (default: 1)
`--split-increment`	Split mark count increment into two calls
`-m {arg}, --sync-mode {arg}`	posted|polling|hybrid (default: polling)
`--delay-val {arg}`	Number of cycles for delay instruction. Max is 20 (default: 5)
`--delay-num {arg}`	Number of delay instructions (default: 32)
`-c {arg}, --mark-count {arg}`	Initial value of mark-count (default: 4095)
`--multiple-sync-groups`	split MultiIPU device into separate sync groups
`-v, --verbose`	Verbose output
`-j, --json-output`	Emit JSON output
`-h, --help`	Produce help message
`--version`	Version number

CHAPTER

SEVEN

GC-HOSTTRAFFICTEST

This tool tests the data transfer between the host machine and the IPUs (in both directions).

To use it, run:

```
gc-hosttraffictest -d {device_id} -j
```

where `{device_id}` is the id number returned by the *gc-inventory* tool.

If JSON output is selected, the output will look something like this:


```
{
  "repeat_0": {
      "tile_to_host": {
          "duration_sec": "2.8448334399999999",
          "kbytes_transferred": "12800000",
          "gbytes_transferred": "12.20703125",
          "gbps": "36.858959306946282"
      },
      "host_to_tile": {
          "duration_sec": "2.0582515429999999",
          "kbytes_transferred": "12800000",
          "gbytes_transferred": "12.20703125",
          "gbps": "50.944987922693372"
      }
  }
}
```


The output will be plain text if the `-j` option is not specified. For example:

```
$ gc-hosttraffictest --device 0
Running test 1/1
tile -> host: 12.2070 GiB in 2.84 seconds, 4.29 GiB/s, 36.87 Gbps
host -> tile: 12.2070 GiB in 2.07 seconds, 5.91 GiB/s, 50.72 Gbps
```

If an error occurs during the test, then `gc-hosttraffictest` will return a non-zero exit code, and output an error message to the terminal.

7.1 Usage

7.1.1 Allowed options

`-j, --json-output`	Emit JSON output
`-d {id}, --device-id {id}`	Device id
`-n {arg}, --num-tiles {arg}`	Number of tiles: 1 to 32 (default: 32)
`-m {arg}, --mode {arg}`	r|w|rw|cc (default: rw)
`-p {arg}, --payload-blocks {arg}`	Number of 64 byte blocks per transfer: 2|4 (default: 4)
`-i {arg}, --iterations {arg}`	Number of 4KB transfers per tile (default: 100000)
`--min-ipu-host-bandwidth {arg}`	Minimum bandwidth (Gbps) IPU to host expected - fails if not reached
`--min-host-ipu-bandwidth {arg}`	Minimum bandwidth (Gbps) host to IPU expected - fails if not reached
`-v, --verbose`	Verbose output
`-h, --help`	Produce help message
`--version`	Version number
`-r, --remote-buffers`	Use remote buffers
`-R {arg}, --repeat {arg}`	Number of times to repeat test (default: 1)

CHAPTER

EIGHT

GC-INFO

This tool lists detailed information about the IPUs present in the hardware platform. To extract some of the information, `gc-info` will need to lock access to IPUs. Therefore, most options (except `--list-devices`) cannot be used for IPUs that are already in use.

8.1 Sub-commands

A number of sub-commands are available as command line options to `gc-info`. The most useful are listed below.

Note: Several of the options to `gc-info` are intended primarily for use by Graphcore engineering and support staff. The *glossary* provides a brief explanation of some of the terminology, in case it is useful.

8.1.1 List devices

The `--list-devices` and `--list-all-devices` command options will list the IPUs in the system. The `--list-devices` option will list only IPUs directly connected to the server.

8.1.2 Tile overview

The `--tile-overview` command option will show a representation of the sync state of all the tiles of the IPU or IPUs this device is connected to.

```
$ gc-info -d 0 --tile-overview
     0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
  0 xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx:: xx::
  1 :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: ::::
  2 :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: ::::
  3 :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: :::: ::::
```

The representation is in terms of supertile (group of 4 tiles), arranged by tile column and distance from the exchange. Columns 0-7 being north tile columns (moving eastwards), 8-15 being south tile columns (moving back westwards). The rows are the supertile index as you move away from the exchange.

The groupings of 4 show the state of each tile within a supertile, the first pair showing tiles 0 and 1 in supertile, and the second pair tiles 2 and 3.

Specifically for row 0, the first pair for each supertile (marked `xx` above) are the first tiles connected to the exchange in that column.

The symbol position is related to the physical tile number by:

physical tile number = 64 x *row* + 4 x *column* + *tile*

Where:

- *row* is the supertile row index
- *column* is the column index
- *tile* is 0..3 (tile index within the supertile)

Note: There are two numbering schemes used to index tiles. A *physical tile number* is commonly used by low level tools such as `gc-info`. Internally Poplar uses a *virtual tile number*, but will report both.

Whilst the layout takes a little interpretation, the representation is compact and the most useful feature is to identify discrepancies between sync types. Identifying one tile that is not of a particular sync type for an IPU or group of IPUs can help you target debug.

Table 8.1: Key for tile-overview

X	(non debug) exception
x	debug exception (trap)
r	waiting on sync (receiving data)
:	waiting on multiple syncs (sans instruction)
'	waiting on local sync (workers in tile)
.	waiting on internal sync (other tiles within IPU)
AB	waiting on external sync including host (A=GS1, B=GS2)
abcd	waiting on external sync excluding host (a=GS1, b=GS2, c=GS3, d=GS4)
*	executing (some other instruction)
–	tile unloaded
?	inaccessible / unknown

8.1.3 Register dump

The `--tile-status` command dumps the register state from the individual tiles in the IPU. There are lots of options to control what it dumps. This is useful for low-level debugging of an application or IPU fault.

Some examples:

```
$ gc-info --device-id 0 --tile-status 0  # dumps out all tile registers on device 0
$ gc-info --device-id 0 --tile-status 0 --context SU # dumps out all tile registers on device 0 for the supervisor context
$ gc-info --device-id 0 --tile-status 0 --context SU --register PC # dumps out the PC for the supervisor context on tile 0
$ gc-info --device-id 0 --tile-status - # dumps out the tile status for all tiles.
$ gc-info --device-id 0 --tile-status - -c SU -r PC # Dumps out the PC for the supervisor context from every tile
```

The SU context displays information from the supervisor execution context used for managing worker threads and exchanges.

The W*n* (or just *n*) context displays information from worker thread *n*.

There are also commands to display various SoC registers for low-level debugging; for example, `--xb-status`, `--gsp-status`, `--pci-status`.

8.1.4 Dump tile memory

The `--dump-mem` command displays the contents of memory on the specified tile. It takes three arguments: the tile number, the start address and the number of bytes to display.

For example, the following command dumps 16 bytes of memory from address 0x4c000 on tile 0:

```
$ gc-info --device-id 0 --dump-mem 0 0x4c000 16
```

8.2 Usage

8.2.1 Commands

`-l, --list-devices`	List devices
`-a, --list-all-devices`	List all devices
`--chip-id`	Show IPU chip ID
`-t {tile_id}, --tile-status {tile_id}`	Tile register dump
`-k, --tile-clock-speed`	Tile clock speed
`-i, --device-info`	Device info
`-m {arg}, --dump-mem {arg}`	{tile_num} {start_address} {size_in_bytes}
`--tr-status`	Trunk Router status
`-x, --xb-status`	XB status
`--gsp-status`	GSPs status
`--nlc-status`	NLCs status
`--pci-status`	SoC PCI status registers
`--ss-status`	System services registers
`--sxp-status`	Secure exchange pipe registers
`--ipu-arch`	Display IPU arch name
`--ipu-count`	Display the number of IPUs installed
`-r {arg}, --register {arg}`	Select register to print from tiles (‘-‘ is all registers) (default: -)
`-c {arg}, --context {arg}`	Select register context (‘-‘ is supervisor and TDI) (default: -)
`--group-output`	Set to group tile status output by value. Only valid with –register
`--phy-summary`	PCI PHY summary
`--phy-dump`	PCI PHY dump
`--show-insn`	List the instruction at the current supervisor’s $PC, for all tiles.
`--tile-overview`	Show an overview of the state of all tiles.
`--fw-pub-keys`	Display part of firmware public keys
`--remote-id {arg}`	Remote device id in format HOSTNAME:DEVICE_ID
`-j, --json-output`	Emit JSON output
`-h, --help`	Produce help message
`--version`	Version number

8.2.2 Command options

`-s, --disassemble`	Disassemble memory dump
`-d {id}, --device-id {id}`	Device id

8.2.3 Examples

```
gc-info --list-devices
gc-info --device-id {id} --tile-status {tile_num}
gc-info --device-id {id} --tile-status {tile_num} --register {reg}
gc-info --device-id {id} --tile-status {tile_num} --context - --register {reg}
gc-info --device-id {id} --tile-status {tile_num} --context SU --register {reg}
gc-info --device-id {id} --tile-status {tile_num} --context W0 --register {reg}
gc-info --device-id {id} --tile-status {tile_num} --context TDI --register {reg}
gc-info --device-id {id} --tile-status {tile_num} --context 0 --register {reg}
gc-info --device-id {id} --device-info
gc-info --device-id {id} --dump-mem {tile_num} {start_address} {size_in_bytes} [--disassemble]
gc-info --device-id {id} --xb-status
gc-info --device-id {id} --gsp-status
gc-info --device-id {id} --show-insn
gc-info --device-id {id} --fw-pub-keys
```

8.3 Glossary

External exchange Data communication between IPUs or between an IPU and the host.

GSP Global Sync Peripheral. An IPU subsystem that manages the synchronization of IPUs over multiple PCI cards.

NLC Network Link Controller. The interface between a *PCI* controller used for external exchange and a *trunk router*.

PCI Peripheral Component Interconnect. The bus standard used for connecting IPUs to the host, and for IPU-Links between IPUs.

SoC System on Chip. A device, such as the IPU, that contains processor, memory, external interfaces and on-chip peripherals,

SS System services registers.

Supertile A group of four colocated tiles.

TDI Tile debug interface.

TR Trunk Router. Connects exchange traffic between the *exchange block* and the *PCI* controller.

XB Exchange block. A subsystem that manages external exchange on behalf of the tiles in the IPU.

CHAPTER

NINE

GC-INVENTORY

This tool lists device IDs, physical parameters and firmware version numbers of the IPUs present in the system.

Unlike *gc-info*, this command will not attempt to lock access to any IPUs. This means that it can be used to gather information about a system even if some of the IPUs are in use. However, this also means that it is not able to provide as much information as *gc-info*.

The output will be similar to this:

```
 {
 "devices": [
     {
         "id": "0",
         "target": "Fabric",
         "Attached": "0",
         "Config Domain": "94891151096224",
         "Driver version": "1.0.50",
         "Fabric Reconfigurable": "true",
         "Firmware Major Version": "2",
         "Firmware Minor Version": "1",
         "Firmware Patch Version": "3",
         "IPU": "1",
         "IPU sync utilisation": "0.00%",
         "IPU version": "ipu2",
         "In use": "0",
         "Location": "10.1.5.12",
         "Num Replicas": "1",
         "PCI Id": "3",
         "PCI slot": "3",
         "Partition ID": "pa12_ipuof",
         "Routing Id": "0",
         "Routing Type": "DNC",
         "Server version": "1.5.4",
         "Sync Type": "c2-compatible",
         "clock": "1330MHz",
         "gateway software version": "2.0.10",
         "graph streaming": "true",
         "hexoatt total size": "16911433728",
         "hexoatt used size": "0",
         "hexopt total size": "268435456",
         "hexopt used size": "0",
         "link correctable error count": "0",
         "link speed": "16 GT\/s",
         "link width": "8",
         "remote buffers": "1",
         "serial number": "0024.0002.8203321",
         "type": "M2000"
     },
...
]
}
```

If the IPU is in use, the output will contain some additional fields relating to the attached process:

```
{
"devices": [
```

(continues on next page)

(continued from previous page)


```
    {
        "id": "0",
        "target": "Fabric",
        "Attached": "0",
        "Config Domain": "94615721195936",
        "Driver version": "1.0.50",
        "Fabric Reconfigurable": "true",
        "Firmware Major Version": "2",
        "Firmware Minor Version": "1",
        "Firmware Patch Version": "3",
        "IPU": "1",
        "IPU sync utilisation": "0.00%",
        "IPU version": "ipu2",
        "In use": "0",
        "Location": "10.1.5.12",
        "Num Replicas": "1",
        "PCI Id": "3",
        "PCI slot": "3",
        "Partition ID": "pa12_ipuof",
        "Routing Id": "0",
        "Routing Type": "DNC",
        "Server version": "1.5.4",
        "Sync Type": "c2-compatible",
        "average board temp": "N\/A",
        "average die temp": "N\/A",
        "clock": "1330MHz",
        "gateway software version": "2.0.10",
        "graph streaming": "true",
        "hexoatt total size": "16911433728",
        "hexoatt used size": "0",
        "hexopt total size": "268435456",
        "hexopt used size": "0",
        "link correctable error count": "462807",
        "link speed": "16 GT\/s",
        "link width": "8",
        "process start time": "20210415T171938Z",
        "remote buffers": "1",
        "serial number": "0024.0002.8203321",
        "total board power": "N\/A",
        "type": "M2000",
        "user executable": "gc-powertest",
        "user name": "emmaf",
        "user process id": "17870"
    },
...
]
}
```


9.1 Usage

9.1.1 Allowed options

`-2, --json-v2`	Emit JSON output only using version 2 attribute names
`-j, --json-output`	Emit JSON output with v2 and legacy attribute names (deprecated)
`-m, --multi-devices`	Include MultiIPU devices in output
`-h, --help`	Produce help message
`--version`	Version number

CHAPTER

TEN

GC-IPUTRAFFICTEST

This tool tests the data transfer between IPUs.

To use it to test data transfer between device 2 and device 3, for example, run:

```
gc-iputraffictest --device0 2 --device1 3 -j
```

The device numbers used are those returned by the *gc-info* tool.

The output will be plain text if the `-j` option is not specified. The JSON output will look something like this:


```
{
  "tile_to_tile": {
      "boards": {
          "0024.xxxx.8203321": {
              "power_sensors": {
                  "XDPE132G5C:0": "25.5",
                  "XDPE132G5C:1": "21"
              },
              "device_ids": [
                  "3",
                  "2"
              ],
              "type": "M2000",
              "board_power": "46.5"
          }
      },
      "device0": "2",
      "device1": "3",
      "duration_sec": "0.079266418000000005",
      "kbytes_transferred": "4096000",
      "gbytes_transferred": "3.90625",
      "gbps": "423.31207649625344",
      "errors": "0",
      "power": "46.5"
  }
}
```


The reported bandwidth is the simultaneous bi-directional bandwidth: the sum of the bandwidth from `device0` to `device1` and the bandwidth from `device1` to `device0`.

If an error occurs during the test, then `gc-iputraffictest` will return a non-zero exit code, and output an error message.

You can use this tool to run a "soak test" of all the IPU-to-IPU links by running:

```
$ gc-iputraffictest --all-links
```

There is also a `--forever` option that will run either a point-to-point or all-link soak forever (until interrupted by `Ctrl-C`). The `-j` option is not available when `--forever` is used.

Note: The `--device0` and `--device1` arguments must be IDs for single IPUs, not groups of IPUs. (The test will connect to the smallest group containing both the sender and receiver.)

10.1 Usage

10.1.1 Allowed options

`-j, --json-output`	Emit JSON output
`--device0 {arg}`	First IPU of exchange
`--device1 {arg}`	Second IPU of exchange
`-i {arg}, --iterations {arg}`	Number of (64)KB transfers per tile (default: 1000)
`-v, --verbose`	Verbose output
`--superverbose`	Even more verbose output
`--amp`	Run high power consumption AMP loop on idle tiles
`--sync-before-amp`	AMP loop tiles sync with host before starting
`--pulseamp`	[=arg(=300)] '–amp' option but with a delay between iterations.Interval specified in units of 1000 cycles
`--ipum-loopback`	Test using IPU-M loopback config
`--all-links`	Exercise every link at once
`--min-bandwidth {arg}`	Minimum bandwidth (Gbps) expected - fails if not reached
`--max-errcount {arg}`	Maximum error count expected - fails if exceeded
`--forever`	Transmit data forever
`-h, --help`	Produce help message
`--version`	Version number

CHAPTER

ELEVEN

GC-LINKS

This tool displays the status and connectivity of each of the IPU-Links that connect the IPUs. It does this by "training" the links with some data, and then checking that the data can be retrieved across all the links.

To use it, run:

```
gc-links -j
```

Note: On IPU-POD systems, the IPU-Links are trained when the partition is created. When running `gc-links`, the output will look something like this:

On a system where link training is available, the output will look similar to the following example:


```
{
  "ipu to ipu": {
      "from pcie id": "5",
      "to pcie id": "7",
      "channel": {
          "from": "NLC_E_2A",
          "to": "NLC_E_3A",
          "status": "passed",
          "gen": "4",
          "lanes": "8"
      },
      "channel": {
          "from": "NLC_E_2B",
          "to": "NLC_E_3B",
          "status": "passed",
          "gen": "4",
          "lanes": "8"
      }
  },
  "num ipus": "16",
  "overall result": "passed",
  "training fails": "0"
}
```


The "status" field shows the training status for each link. The "lanes" field shows the number of lanes being trained, and the "gen" field shows what generation of link is tested.

When `gc-links` finds that a link fails to train, the output looks like this:


```
{
  "ipu to ipu": {
    "from pcie id": "7",
    "to pcie id": "6",
    "channel": {
        "from": "NLC_W_1B",
        "to": "NLC_W_1B",
        "status": "passed",
        "gen": "4",
        "lanes": "8"
    },
    "channel": {
        "from": "NLC_W_1C",
```


(continues on next page)

(continued from previous page)


```
        "to": "NLC_W_1C",
        "status": "failed",
        "gen": "0",
        "lanes": "0"
    }
  }
}
```


This output shows that it failed to train the link from device 7 to device 6, using link NLC_W_1C.

You can run *gc-inventory* to show more information on devices 6 and 7:

```
Device:
  id: 6
  type: C2
  Firmware Major Version: 1
  Firmware Minor Version: 0
  Firmware Patch Version: 4
  IPU: 1
  IPU version: ipu0
  PCI Id: 0000:43:00.0
  link speed: 8 GT/s
  link width: 8
  physical slot: PCIe Slot 12
  serial number: 0020.0004.018471
Device:
  id: 7
  type: C2
  Firmware Major Version: 1
  Firmware Minor Version: 0
  Firmware Patch Version: 4
  IPU: 1
  IPU version: ipu0
  PCI Id: 0000:44:00.0
  link speed: 8 GT/s
  link width: 8
  physical slot: PCIe Slot 13
  serial number: 0027.0004.018481
```

You can see, in this example, that there's an issue with the link between the card in slot 12 and 13.

When training a chassis full of IPUs, the tool outputs additional information on IPU-Link failures, for example:


```
{
  "failures": [
    {
        "cable_id": "IPUL-00"
    },
    {
        "cable_id": "IPUL-24"
    },
    {
        "cable_id": "IPUL-25"
    }
  ]
}
```


The physical location of these failing cables is shown in *IPU-Link channel mapping*.

11.1 Usage

11.1.1 Allowed options

`-j, --json-output`	Emit JSON output
`-n {arg}, --num-retries {arg}`	Number of link training retries (default: 3)
`-d {id}, --device-id {id}`	Device id (default is largest group)
`-i {arg}, --num-iterations {arg}`	Number of times to train each link (default: 1)
`-v, --verbose`	Verbose output
`-p, --phy-summary`	Print PHY summary after all training runs
`-h, --help`	Produce help message
`--until-trained`	Run the training until it succeeds
`-f, --phy-summary-failure`	Print PHY summary after training failures
`-l {path}, --load-phy-firmware {path}`	Load PHY firmware from specified path
`--allow-all-presets`	Allow all NLC TX presets when training, default is preset 4 only
`--allow-full-swing`	Allow full swing on the transmitter, default is quarter swing, does nothing if allow-all-presets is set
`--check-fom`	Check the FOM, declare training fail if it is too low
`--version`	Version number

CHAPTER

TWELVE

GC-MEMORYTEST

This tool tests all the tile memory in an IPU, reporting any tiles that fail.

To use it, run:

```
gc-memorytest -d {device_id}
```

where `{device_id}` is the id number returned by the *gc-inventory* tool. If the test reveals no issues with the device memory, the output will look like this:

```
{
  "result": "pass"
}
```

If the test reveals any issues with the device memory, the output shows which tiles have failed, and looks like this:

```
{
  "tile_results": [
    {
        "12": "fail"
    },
    {
        "1000": "fail"
    },
    {
        "0": "fail"
    }
  ],
  "result": "fail"
}
```

12.1 Usage

12.1.1 Allowed options

`-d {id}, --device-id {id}`	Device id
`-v, --verbose`	Verbose output
`--vddcheck`	Check that the IPU works at different VDD levels
`-h, --help`	Produce help message
`--version`	Version number

CHAPTER

THIRTEEN

GC-MONITOR

You can use this command to monitor IPU activity without affecting users of the IPUs. This can be used to:

- Check and monitor what's currently running on which IPU in shared systems.
- Make sure code is correctly running on an IPU.
- Monitor performance: the power and temp will increase, and the clock rate will drop when an IPU is heavily loaded.

To get a continuously updated display, use it with the `watch` command:

```
$ watch gc-monitor
```

13.1 Output

The output shows information about the IPUs in the system and information on the processes running on the machine. The fields available vary slightly depending on the system.

The card information table for a 4-IPU partition shows:

1. Information about the partition.
2. IPU-Machine address.
3. Serial number of the board.
4. ICU firmware revision.
5. IPU card type.
6. Fabric server version.
7. ID of the IPU as used by other tools to address the IPUs.
8. PCIe ID.
9. Routing type.
10. Gateway software version.

Typical output is shown below.

```
+--------------+-------------------------------------------------------------------------------+
|  gc-monitor  |                 Partition: 'pa12' has 4 reconfigurable IPUs                   |
+-------------+-------------------+----------+------+--------------+----+---------+-------+----------+
|    IPU-M    |       Serial      |  ICU FW  | Type |Server version| ID | PCIe ID |Routing|   GWSW   |
+-------------+-------------------+----------+------+--------------+----+---------+-------+----------+
|  10.1.5.12  | 0024.0002.8203321 |  2.1.3   |M2000 |    1.5.4     | 0  |    3    |  DNC  |  2.0.10  |
|  10.1.5.12  | 0024.0002.8203321 |  2.1.3   |M2000 |    1.5.4     | 1  |    2    |  DNC  |  2.0.10  |
+-------------+-------------------+----------+------+--------------+----+---------+-------+----------+
|  10.1.5.12  | 0024.0001.8203321 |  2.1.3   |M2000 |    1.5.4     | 2  |    1    |  DNC  |  2.0.10  |
|  10.1.5.12  | 0024.0001.8203321 |  2.1.3   |M2000 |    1.5.4     | 3  |    0    |  DNC  |  2.0.10  |
+-------------+-------------------+----------+------+--------------+----+---------+-------+----------+
```

(continues on next page)

(continued from previous page)

```
+-----------------------------------------------------------+-----------------------+-----------------+
|                    Attached processes                     |          IPU          |      Board      |
+--------+---------------------------+--------+-------------+----+----------+--------+--------+--------+
|  PID   |          Command          |  Time  |    User     | ID |  Clock   |  Temp  |  Temp  | Power  |
+--------+---------------------------+--------+-------------+----+----------+--------+--------+--------+
| 10629  |       gc-powertest        |   5s   |   emmaf     | 0  | 1330MHz  |  N/A   |  N/A   |  N/A   |
+--------+---------------------------+--------+-------------+----+----------+--------+--------+--------+
```

The card information for a PCIe card shows:

1. Physical PCIe slot location, if available.
2. Serial number of the board.
3. ICU firmware revision.
4. Installed kernel module driver version number.
5. IPU card type.
6. PCIe width and speed.
7. ID of the IPU as used by other tools to address the IPUs.
8. Which IPUs are on the same card.
9. PCIe ID.
10. IPU number (which IPU on a card it is).

The process information displayed includes:

1. The process ID (PID) using the IPU.
2. The process name using the IPU.
3. The time it has been running.
4. The username of the user using the IPU.
5. The ID of the IPU in use.
6. The current measured IPU clock rate.
7. The current IPU temperature.
8. The current board temperature.
9. The current board power consumption.

By default, the temperature and power data are not available. To enable these, the process running on the IPU must be launched with the `GCDA_MONITOR` environment variable set. For example, to add the temperature and power data to the output when monitoring a program called `test`, the following commands could be used:

```
$ GCDA_MONITOR=1 test
$ watch gc-monitor
```

13.1.1 Usage

13.2 Allowed options

`--no-card-info`	Don't display card information
`-j, --json-output`	Emit JSON output
`-x, --xml-output`	Emit XML output
`-c, --csv-output`	Emit CSV output
`-f {arg}, --fields {arg}`	Comma separated field names to be displayed in attached process info. Use -f list to see the choices. If empty all supported fields are displayed.
`--no-headers`	Don't display headers (csv only)
`--version`	Version number
`-h, --help`	Produce help message

13.3 Notes

- The number of link errors (LnErr) is printed using scientific notation.
- By default, gc-monitor shows the processes running on attached IPUs, the users running them, the processes' PIDs and the IPUs' IDs & clock speeds.
- Also by default, the temperature and power data is not available. To enable it for a specific process running on the IPU, that process must be launched with the GCDA_MONITOR environment variable set to 1. For example, to add the temperature and power data to the output when monitoring a gc-powertest process, the following commands could be used:

  $ GCDA_MONITOR=1 gc-powertest -d 0 $ watch gc-monitor

13.4 Examples

```
$ GCDA_MONITOR=1 python main.py # Run main.py, enabling gc-monitor power
and temp output for the IPUs it uses
$ watch -n1 gc-monitor          # Continually monitor IPUs every
second, visually
$ gc-monitor -j >> data.json    # Append one JSON gc-monitor reading
to a data file
```

CHAPTER

FOURTEEN

GC-POWERTEST

This tool is used to test power consumption and temperature of the IPU processor.

Note that:

- The program runs indefinitely until killed (with `Ctrl-C`).
- It can be run multiple times to start code on other IPUs.
- Temperatures are reported in degrees Celsius.
- After running `gc-powertest` it's a good idea to reset the IPUs (using the *gc-reset* tool) otherwise they'll be left running at high power.

To use this tool, run:

```
gc-powertest -d {device_id}
```

where `{device_id}` is the id number returned by the *gc-inventory* tool.

Some typical output is shown below (split into three separate columns here). This example is for a single IPU in an IPU-M2000. Similar output will be generated for other systems.

The output is constantly updated until the process is killed. The first column shows power readings taken from the board:

```
|                    CARD LEVEL
-------------------------------------
|            XDPE132G5C             |
-------------------------------------
| 0001: 024.50W  021.50W  T046.00W  |
| 0002: 024.50W  022.00W  T046.50W  |
| 0003: 024.50W  021.00W  T045.50W  |
| 0004: 024.50W  021.50W  T046.00W  |
```

The second column shows temperature readings

```
     CARD LEVEL                                |
-----------------------------------------------
| B_COL0:0 B_COL1:0 B_IN:0   B_MID:0  B_OUT:0  |
-----------------------------------------------
| 041C     039C     038C     034C     039C     |
| 041C     039C     038C     034C     039C     |
| 041C     039C     038C     034C     039C     |
| 041C     039C     038C     034C     039C     |
```

The third column shows temperature and clock readings measured on the IPU:

```
|                      IPU#1                     |
-------------------------------------------------
| 0:PVT0    0:PVT1    0:PVT2    0:PVTE    CLOCK    |
-------------------------------------------------
| 039.4C    037.2C    038.3C    033.3C    1330MHz  |
| 039.1C    037.2C    038.3C    033.3C    1330MHz  |
| 038.8C    037.2C    038.5C    033.6C    1330MHz  |
| 039.1C    036.9C    038.5C    033.3C    1330MHz  |
```

By default, the power test will run with 0% IPU processing load. The load can be set with the `-p` option. This must be in multiples of 10%. For systems older than IPU-M2000 only, 20%, 50% or 100% loads can be requested.

For example:

```
gc-powertest -d {device_id} -p 50
```

14.1 Usage

14.1.1 Allowed options

`-p {arg}, --percent-load {arg}`	IPU load percentage (default: 0)
`-d {id}, --device-id {id}`	Device id
`-b {binary}, --binary {binary}`	Use custom IPU binary file
`-v, --verbose`	Verbose mode
`-j, --json-output`	Emit JSON output
`-t {time}, --time {time}`	Time in seconds of the measurements (0 is infinite) (default: 0)
`-h, --help`	Produce help message
`--version`	Version number

CHAPTER

FIFTEEN

GC-RESET

This tool resets the IPU devices. For example:

```
gc-reset -d {device_id}
```

Where `{device_id}` is the id number returned by the *gc-info* tool. This can refer to one IPU or a group of IPUs.

15.1 Usage

15.1.1 Allowed options

`-t, --teardown-links`	Teardown IPU links
`-m, --reset-memory`	Reset tile memory and registers (for debugging)
`-d {id}, --device-id {id}`	Device id
`-h, --help`	Produce help message
`--version`	Version number

15.1.2 Examples

```
gc-reset           # reset all IPUs
gc-reset -d 0      # reset a single IPU
gc-reset -d 30 -t  # tear down a link
gc-reset -d 0 -m   # reinitialize memory and registers
```

15.1.3 Notes

- When tearing down links, device ID is the MultiIPU group you want to untrain the links for. Typically you would want to use the biggest MultiIPU group.
- Links cannot be torn down on IPUoF partitions; using the -t option on these platforms will have no effect.

CHAPTER

SIXTEEN

GC-FLOPS

This tool allows you to benchmark the number of floating point operations per second on one or more IPU processors. The tool supports Mk2 architectures only.

16.1 Precision

The `--fp` and `-p` command options select the floating point precision. You can choose between `FP16` (the default), which is 16 bit floating point also known as half-precision floating point, or `FP32` for single precision floating point.

16.2 Device

The `--device-id` and `-d` command options will let you specify the IPU device to benchmark.

Note: The tool reports benchmark results in gigaFLOPS. The clock speed of the IPU affects these results. The tool measures and reports the clock speed to the user. For Multi IPU device, the tool reports the lowest value from all IPUs.

CHAPTER

SEVENTEEN

C2 PCIE DEVICE IDS AND CHANNEL MAP

The following information applies to C2 cards in a PCIe server. For IPU-Machine and IPU-POD systems, see the appropriate product documentation.

17.1 Device IDs

The diagram below shows how single IPUs and then, hierarchically, groups of IPUs are numbered in a 16-IPU system. For example, device ID 25 corresponds to the four IPUs numbered 0 to 3. These numbers are the IDs used by tools such as *gc-info*.

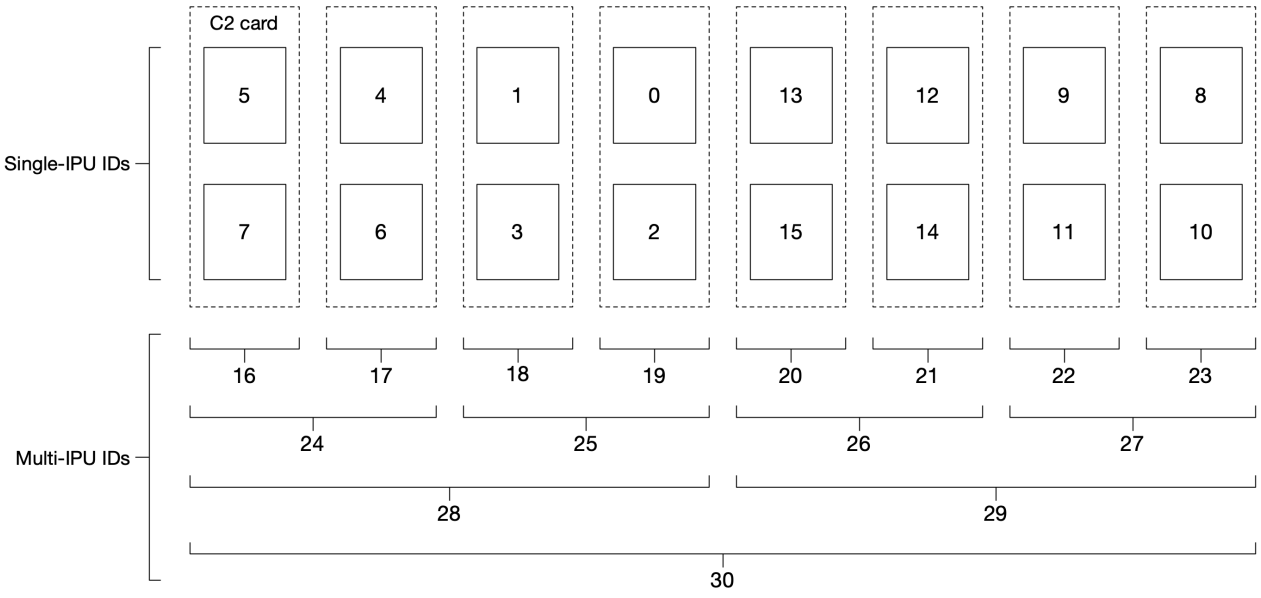

Fig. 17.1: IPU device IDs

17.2 IPU-Link channel mapping

The diagram below shows the how channels connect between the cards in a system.

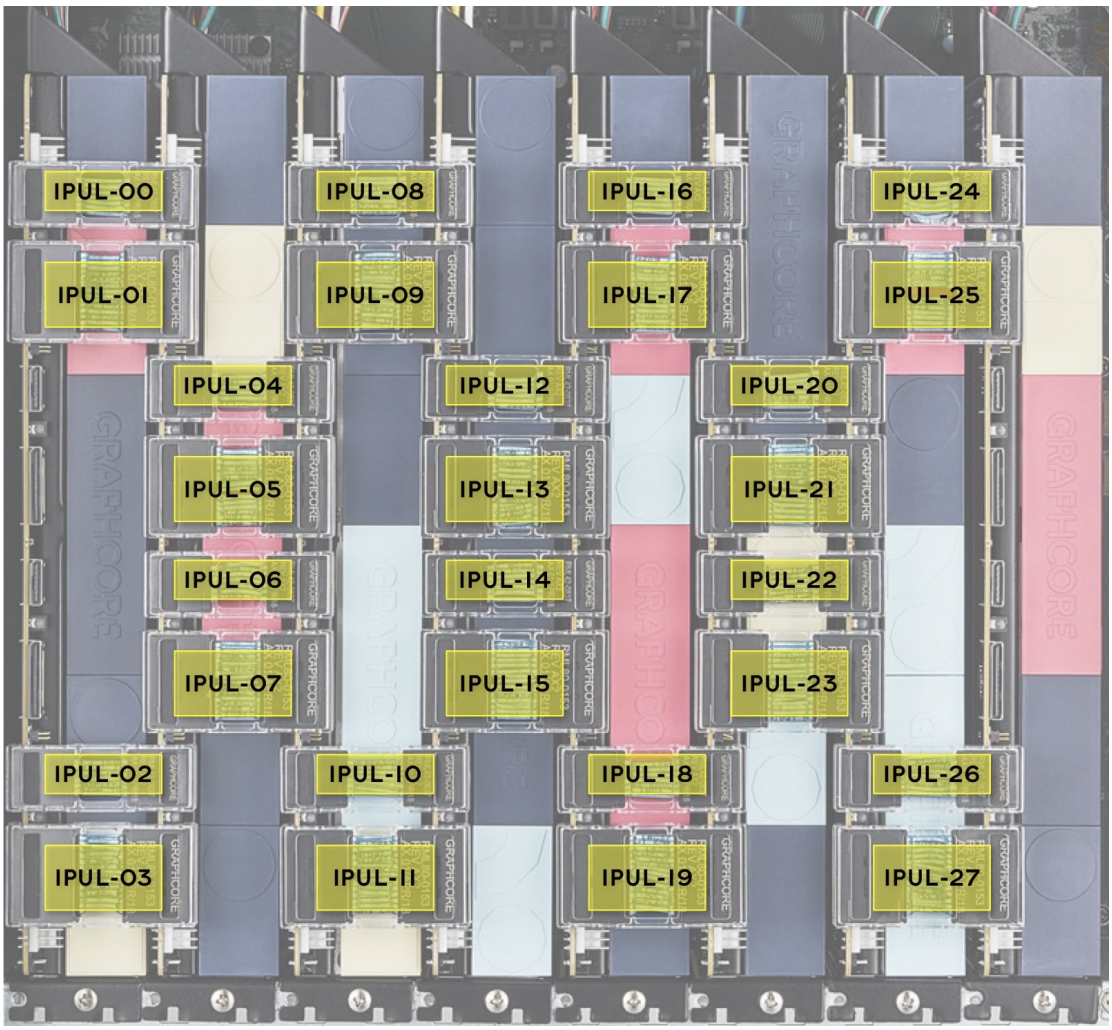

Fig. 17.2: IPU-Link card-to-card channel map

17.3 PCIe ID to slot mapping

The Graphcore tools typically deduce which IPU cards are in which PCIe slot via a table in the SMBIOS. Alternatively, a JSON configuration file may be provided by setting the environment variable `GCDA_CONFIG_FILE`. The file must have the following format:

```
{
  "ipu_card_mapping": [
    "0000:0f:00.0",
    "0000:0e:00.0",
    "0000:0d:00.0",
    "0000:0c:00.0",
    "0000:0b:00.0",
    "0000:0a:00.0",
    "0000:09:00.0",
    "0000:08:00.0",
    "0000:07:00.0",
    "0000:06:00.0",
```

(continues on next page)

(continued from previous page)

```
    "0000:05:00.0",
    "0000:04:00.0",
    "0000:03:00.0",
    "0000:02:00.0",
    "0000:01:00.0",
    "0000:00:00.0"
  ]
}
```

Where “ipu_card_mapping” is an ordered list of the PCIe IDs for the Graphcore cards, from one side of the chassis to the next. Each pair of IDs should match a single card, and the next pair should be in the next physical slot.

CHAPTER

EIGHTEEN

TRADEMARKS & COPYRIGHT

INDEX