MaxelerOS
Installation and Administration Guide

Version 2014.2

Contents

1 Introduction

This document describes the installation and setup of a Maxeler System and the MaxelerOS. A Maxeler System consists of dataflow engines (DFEs) which are either on cards locally in the host node, or in an MPC-X system connected to the host. Typically a Maxeler System may contain:

- A workstation, MPC-C or MPC-N containing a number of DFEs.
- MPC-X systems of DFEs connected via an Infiniband network.
- A node with local DFEs can also be connected to MPC-X systems.

MaxelerOS is the software installation that is required to run an application on DFEs. A MaxelerOS installation contains four major software components:

Driver – provides low level access to local cards (via device files `/dev/maxeler0 - N-1`).

Daemon – runs in the background to configure and monitor the local cards. The MaxelerOS daemon must be running for Maxeler applications to access the cards.

Utilities – a set of command line utilities for managing local cards and MPC-X systems, including `maxtop` to report card and DFE status, `maxstatuscheck` to monitor system health and `maxdiag` to run hardware diagnostics.

`libmaxeleros` – shared library used by MaxCompiler SLiC interface to communicate with local cards or a remote MPC-X system. see the MaxCompiler Tutorial document for more information on SLiC.

A CPU node accessing remote DFEs in an MPC-X system, only require the utilities and `libmaxeleros`. The driver and daemon run only where there are local DFEs in the node.

2 MaxelerOS Software Installation

The MaxelerOS software should be installed on each compute host node containing Maxeler cards and/or requiring access to an MPC-X system. It is important to select the correct RPM to install. The basic RPM is distributed with support for local cards only, and as such has no dependency on Infiniband libraries. If the host node will be used with an MPC-X system then the MPC-X variant should be installed. The MPC-X variant includes support for local cards so only one RPM needs to be installed on each host.

2.1 Supported Operating Systems

MaxelerOS is provided as an RPM for 64-bit Red Hat Enterprise Linux (RHEL) or CentOS 6.x and 5.x.

There are separate RPMs for versions 6.x and 5.x of RHEL/CentOS. Please ensure you install the correct version for your installed operating system, either:

- The RPM for 6.x is `maxeleros-2014.2-1.el6.x86_64.rpm`.
- The RPM for 5.x is `maxeleros-2014.2-1.el5.x86_64.rpm`.

or with MPC-X support:

- The RPM for 6.x is `maxeleros-mpcx-2014.2-1.el6.x86_64.rpm`.
- The RPM for 5.x is `maxeleros-mpcx-2014.2-1.el5.x86_64.rpm`.

You can ascertain the version of your installed operating system by running the command:

```
cat /etc/redhat-release
```

2.2 Dependencies

MaxelerOS depends on the `dkms` package. The `dkms` package enables kernel device drivers to be automatically rebuilt when a new kernel is installed.

Maxeler can provide an RPM of `dkms` for convenience. The package only contains scripts, so is itself kernel independent. CentOS users can also obtain the `dkms` package freely by first retrieving the package from this URL:

```
http://packages.sw.be/rpmforge-release/rpmforge-release-0.5.2-2.el5.rf.x86_64.
rpm
```

and then running the following commands as root:

```
[root@machine ~]# rpm -i rpmforge-release-0.5.2-2.el5.rf.x86_64.rpm
[root@machine ~]# yum install dkms
```

2.2.1 MPC-X specific dependencies

The MaxelerOS rpm with MPC-X support has a further dependence on `libibverbs` (Infiniband library). It will be necessary to perform the following installation if not already installed.

```
[root@machine ~]# yum install libibverbs
```

This command accesses the rpmforge repository configured above.

2.3 Installation

MaxelerOS is installed using the `rpm` command:

```
[root@machine ~]# rpm --install maxeleros-2014.2-1.el6.x86_64.rpm
pre-install maxeleros-2014.2 1 /
post-install maxeleros-2014.2 1 /

Creating symlink /var/lib/dkms/maxeleros/2014.2/source ->
                 /usr/src/maxeleros-2014.2

DKMS: add Completed.

Kernel preparation unnecessary for this kernel.  Skipping...

Building module:
cleaning build area....
make KERNELRELEASE=2.6.18-164.el5 -C /lib/modules/2.6.18-164.el5/build M=/var/
lib/dkms/maxeleros/2014.2/build....
cleaning build area....

DKMS: build Completed.

maxeleros.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/2.6.18-164.el5/extra/
Adding any weak-modules

depmod....

DKMS: install Completed.
Starting MaxelerOS driver/daemon
Starting MaxelerOS driver:[  OK  ]
Starting MaxelerOS daemon:[  OK  ]
```

RPM installation will fail with a conflict if an earlier version of MaxelerOS is already installed: use `rpm --upgrade` instead.

2.4 Installation files

MaxelerOS will install the following files:

- `/opt/maxeler/maxeleros`
  - `lib/`
    - `libmaxeleros.so` – shared library for libmaxeleros (used by MaxCompiler SLiC interface for low-level card or DFE operations)
  - `utils/`
    - `maxtop` – lists cards and DFEs in system, helps debug hardware/MaxelerOS issues, shows what is running on the cards and for how long
    - `max2diag, max3diag, max4diag` - runs diagnostics on the installed cards
    - `ch2diag` – runs diagnostics on the installed CH2 network interface cards
    - `maxsyscheck` – checks the status of the installed MaxelerOS system
    - `maxstatuscheck` – queries the status of the daemon and installed cards
    - `maxidentify/maxsysmon` – deprecated utilities for monitoring MaxelerOS
    - `maxforceidle` – forces either a specific DFE or all DFEs into the idle state
    - `maxflash, max3flash` – updates card firmware (should not need to change from the factory firmware)
    - `maxmanifest` – archives system configuration to help remote debugging by Maxeler engineers (RPMs installed, flexlm licenses, machine configuration)
  - `daemon/`
    - `maxelerosd` – daemon binary (runs automatically on boot)
  - `doc/`
    - `MaxelerOSmanual.pdf` – this manual
- `/etc/init.d/maxeleros` - script to load driver and daemon on boot
- `/etc/logrotate.d/maxeleros` - settings to enable system rotation of log files


Warning: `maxflash` should not be used unless advised by a Maxeler engineer.

2.5 Upgrading

The MaxelerOS installation will automatically rebuild the driver when the kernel is updated or changed to ensure compatibility (using `dkms`).

To upgrade to a newer version of MaxelerOS, use the `rpm` command with the argument `-U` or `--upgrade` with the new RPM, which will uninstall the old package and replace it with the new one:

```
[root@machine ~]# rpm --upgrade maxeleros-2014.2-1.el6.x86_64.rpm
```

2.6 Uninstalling

To remove MaxelerOS, simply remove the MaxelerOS RPM:

```
[root@machine ~]# rpm --erase maxeleros-2014.2
pre-uninstall maxeleros-2014.2 0 /
Killing maxeleros driver/daemon.
Stopping MaxelerOS daemon:[  OK  ]
Stopping MaxelerOS driver:[  OK  ]

-------- Uninstall Beginning --------
Module:  maxeleros
Version: 2014.2
Kernel:  2.6.18-164.el5 (x86_64)
-------------------------------------

Status: Before uninstall, this module version was ACTIVE on this kernel.
Removing any linked weak-modules

maxeleros.ko:
 - Uninstallation
   - Deleting from: /lib/modules/2.6.18-164.el5/extra/
 - Original module
   - No original module was found for this module on this kernel.
   - Use the dkms install command to reinstall any previous module version.

depmod....

DKMS: uninstall Completed.

------------------------------
Deleting module version: 2014.2
completely from the DKMS tree.
------------------------------
Done.
post-uninstall maxeleros-2014.2 0 /
```

3 Administration

Sections 3.1, 3.2, and 3.3 are only relevant when utilizing local DFEs and are not required when using MPC-X systems.

3.1 Starting and Stopping the Driver and Daemon

A script `maxeleros` for starting and stopping the driver and daemon is installed in `/etc/init.d`. This automatically runs on boot by default.

This script can also be used to manually start and stop the driver and daemon.


Note: starting and stopping MaxelerOS needs to be done as the root user.

The usage is:

```
/etc/init.d/maxeleros [restart|stop|status|restart|condrestart]
```

To start the service:

```
[root@machine ~]# /etc/init.d/maxeleros start
Starting MaxelerOS Driver and Daemon (maxelerosd):         [  OK  ]
```

To stop the service:

```
[root@machine ~]# /etc/init.d/maxeleros stop
maxelerosd (pid 19811) is running...
Shutting down MaxelerOS Driver and Daemon (maxelerosd):    [  OK  ]
```

To restart the service:

```
[root@machine ~]# /etc/init.d/maxeleros restart
maxelerosd (pid 1289) is running...
maxelerosd (pid 1289) is running...
Shutting down MaxelerOS Driver and Daemon (maxelerosd):    [  OK  ]
Starting MaxelerOS Driver and Daemon (maxelerosd):         [  OK  ]
[root@machine ~]# /etc/init.d/maxeleros stop
maxelerosd (pid 19811) is running...
Shutting down MaxelerOS Driver and Daemon (maxelerosd):    [  OK  ]
[root@machine ~]# /etc/init.d/maxeleros restart
maxelerosd is stopped
Starting MaxelerOS Driver and Daemon (maxelerosd):         [  OK  ]
```

To restart the service *only if it is already running*:

```
[root@machine ~]# /etc/init.d/maxeleros condrestart
maxelerosd (pid 1289) is running...
maxelerosd (pid 1289) is running...
Shutting down MaxelerOS Driver and Daemon (maxelerosd):    [  OK  ]
Starting MaxelerOS Driver and Daemon (maxelerosd):         [  OK  ]
[root@machine ~]# /etc/init.d/maxeleros stop
maxelerosd (pid 19811) is running...
Shutting down MaxelerOS Driver and Daemon (maxelerosd):    [  OK  ]
```

```
[root@machine ~]# /etc/init.d/maxeleros condrestart
maxelerosd is stopped
```

To query the status of the service:

```
[root@machine ~]# /etc/init.d/maxeleros status
MaxelerOS driver loaded.
maxelerosd (pid 19914) is running...
MaxelerOS Daemon running.
```

3.2 Daemon Configuration

The MaxelerOS daemon can be configured using the file `/etc/maxeler/maxelerosd.conf`. The configuration settings are categorized for ease of reference in this section by system monitoring, card monitoring, idle configuration initialization, logging, and miscellaneous settings.

3.2.1 System Monitoring

These options pertain to monitoring the operation of main CPU board and host operating system.

EnableTempMonitor

Purpose: Enable the system temperature monitor.
Parameters: `[yes|no]`
Example: `EnableTempMonitor yes`
Default: `no`

IPMIDevice

Purpose: Set the IPMI device that the system temperature monitor uses.
Parameters: `<device name>`
Example: `IPMIDevice /dev/ipmi0`
Default: `/dev/ipmi0`

You can use the `ipmitool` command in the `freeipmi` package to retrieve a list of available temperature sensors in your system.

```
[root@machine ~]# ipmitool sdr type Temperature
BB Temp          | 30h | ok  |  7.1 | 31 degrees C
Front Panel Temp | 32h | ok  | 12.1 | 18 degrees C
MCH Therm Margin | 33h | ok  |  7.1 | -54 degrees C
Mem Therm Margin | 48h | ok  | 32.1 | -44 degrees C
P1 Therm Margin  | 99h | ok  |  3.1 | -50 degrees C
P1 Therm Ctrl %  | C0h | ok  |  3.1 | 0 unspecified
Proc 1 VRD Hot   | C8h | ok  |  3.1 |
```

The `freeimpi` package is installable by this command:

```
[root@machine ~]# yum install freeipmi
```

The `freeipmi` package is not supported by Maxeler and not necessarily compatible with all system configurations.

TempSensorId

Purpose: Set the IPMI temperature sensor that the system temperature monitor uses.
Parameters: `<temperature sensor id>`
Example: `TempSensorId Front Panel Temp`
Default: `BB Temp (base board temperature sensor)`

TempThreshold

Purpose: Set the threshold temperature for the system temperature monitor in *Celsius*.
Parameters: `<temperature>`
Example: `TempThreshold 70`
Default: `67`

EnableWatchdog

Purpose: Force the machine to reboot automatically if the MaxelerOS daemon becomes unresponsive for a specified period of time or terminates unexpectedly.
Parameters: [`yes|no`]
Example: `EnableWatchdog yes`
Default: `no`

WatchdogTimeout

Purpose: Set period of unresponsivness in *seconds* sufficient to warrant a reboot if `EnableWatchdog` is in effect.
Parameters: `<timeout>`
Example: `WatchdogTimeout 120`
Default: `600`

3.2.2 Card monitoring

These options pertain to monitoring the operation of the cards.

EnableBoardTempMonitor

Purpose: Enable monitoring the temperature of all cards in the system.
Parameters: [`yes|no`]
Example: `EnableBoardTempMonitor yes`
Default: `yes`

Every `BoardTempInterval` seconds, the daemon checks whether each card temperature is above the thresholds `BoardTempWarning` or `BoardTempIdle`.

If a card temperature is above `BoardTempWarning`, a warning is written to the daemon log.

If a card temperature is above `BoardTempIdle`, `EnableBoardTempSignal` is enabled and an application is using the card, the application is killed and a message is written to the daemon log. The DFE will be left configured with the application maxfile; if the DFE temperature does not drop below `BoardTempIdle` temperature within `BoardTempInterval` seconds, then the idle configuration will be loaded onto the card.

If a card temperature is above `BoardTempIdle` and no application is using the card, the idle configuration is loaded onto the card and a message is written to the daemon log.

EnableBoardTempSignal

Purpose: Enable killing of a process that is using the card if the card overheats (see `EnableBoardTempMonitor`).
Parameters: `[yes|no]`
Example: `EnableBoardTempSignal no`
Default: `yes`

BoardTempWarning

Purpose: Temperature (in *Celsius*) above which a warning is printed in the daemon log saying that a device is overheating (see `EnableBoardTempMonitor`).
Parameters: `<temperature>`
Example: `BoardTempWarning 80`
Default: `85`

BoardTempIdle

Purpose: Temperature (in *Celsius*) above which the idle configuration is loaded into the overheating card or an application using the card is killed (see `EnableBoardTempMonitor`).
Parameters: `<temperature>`
Example: `BoardTempIdle 90`
Default: `95`

BoardTempInterval

Purpose: Interval (in *seconds*) for polling the temperature of each card in the system (see `EnableBoardTempMonitor`).
Parameters: `<time interval>`
Example: `BoardTempIdle 60`
Default: `60`

EnableUsageMonitor

Purpose: Enable polling the DFE and system usage every second so that MaxTop can report the usage
Parameters: `[yes|no]`
Example: `EnableUsageMonitor yes`
Default: `yes`

3.2.3 Idle Configuration Initialization

The DFEs can be reinitialized to idle bit streams at various configurable times by options documented in this section.

LoadIdleOnStart

Purpose: Load the idle configuration onto the DFE at daemon startup.
Parameters: [yes|no]
Example: `LoadIdleOnStart yes`
Default: `yes`

LoadIdleOnShutdown

Purpose: Load the idle configuration onto the DFE at daemon shutdown.
Parameters: [yes|no]
Example: `LoadIdleOnShutdown yes`
Default: `yes`

LoadIdleOnTimeout

Purpose: Load the idle configuration onto the DFE if the DFE has not been used for longer than a timeout.
Parameters: [yes|no]
Example: `LoadIdleOnTimeout no`
Default: `yes`

IdleTimeout

Purpose: Set the timeout after which the idle configuration will be loaded onto the DFE in *seconds*.
Parameters: `<timeout>`
Example: `IdleTimeout 120`
Default: `600`

3.2.4 Logging

Information about events and conditions of the daemon useful for system administration is normally written to log files as noted in *subsection 3.3*. These configuration settings allow some further logging preferences to be selected.

LogLevel

Purpose: Control the verbosity of log files. Setting 0 is for errors only, 1 is normal, 2 is verbose, and 3 is more verbose. Settings 2 and 3 cause very large log files and should be used for debuggng only.
Parameters: [0|1|2|3]
Example: `LogLevel 3`
Default: `1`

EnableEventMonitor

Purpose:	(applies only to MAX3/Vectis) Enable monitoring of card events. These events are interrupts from the power supplies that warn of pending or actual failures. These events are written to the daemon log. A Maxeler engineer may require this log information to diagnose any power supply issues.
Parameters:	[yes|no]
Example:	`EnableEventMonitor no`
Default:	`yes`

EnableSyslog

Purpose:	Log all messages about daemon activity in the system log file. This option will cause messages that are normally written in `/var/log/maxelerosd.log` to be written additionally to `/var/log/syslog` or to `/var/log/messages`, depending on the setting in `/etc/syslog.conf`. See *subsection 3.3*.
Parameters:	[yes|no]
Example:	`EnableSyslog yes`
Default:	`no`

3.2.5 Allocation Governor Settings

MaxelerOS can manage groups of dataflow engines automatically to help applications use them interchangeably. In some circumstances, engines may be reallocated from underutilized groups to those in greater demand. The options documented in this section affect the way MaxelerOS performs these allocations. The default settings of these options are optimal for most environments. Varying them is inadvisable except in consultation with a Maxeler engineer.

These options are best understood in terms of an **allocation governor** working within MaxelerOS. The allocation governor alternately performs a sampling sequence and sleeps for a set interval. During each sampling sequence, the allocation governor computes two metrics based on a set number of samples taken at regular intervals. One metric describes the demand for each group. The other metric describes the unused capacity of each group.

- The demand is measured as the number of processes seen to be waiting to use a member of the group at the moment a sample is taken, summed over all samples taken during the sequence.
- The unused capacity is measured as the number of members of the group seen to be idle at the moment a sample is taken, summed over all samples taken during the sequence.

EnableAllocationGovernor

Purpose:	Allow the allocation governor to adjust group sizes or not.
Parameters:	[yes|no]
Example:	`EnableAllocationGovernor yes`
Default:	`yes`

AllocationGovernorInterval

Purpose: The time (in seconds) from the end of one sampling sequence to the beginning of the next
Parameters: `<floating point number of seconds>`
Example: `AllocationGovernorInterval 1`
Default: `1`

AllocationGovernorGrowthThreshold

Purpose: The level of demand sufficient to warrant increasing the size of a group
Parameters: `<integer demand metric>`
Example: `AllocationGovernorGrowthThreshold 5`
Default: `5`

AllocationGovernorShrinkThreshold

Purpose: The level of excess capacity sufficient to warrant decreasing the size of a group
Parameters: `<integer capacity metric>`
Example: `AllocationGovernorShrinkThreshold 5`
Default: `5`

AllocationGovernorSampleCount

Purpose: The number of samples taken during each sampling sequence
Parameters: `<number of samples>`
Example: `AllocationGovernorSampleCount 10`
Default: `10`

AllocationGovernorSampleInterval

Purpose: The length of time (in microseconds) between consecutive samples within a sampling sequence
Parameters: `<number of microseconds>`
Example: `AllocationGovernorSampleInterval 100`
Default: `100`

3.2.6 Miscellaneous Configuration Settings

Some further configuration settings not belonging to any of the previous classifications are noted here.

VerifyBitstream

Purpose: Enable reading back of the bitstream after writing to the card.
Parameters: `[yes|no]`
Example: `VerifyBitstream yes`
Default: `no`

Note: Bitstream verification will make configuration of the DFE much slower, so it is not recommended unless advised by a Maxeler engineer.

ClientConnections

Purpose: Set the maximum number of concurrent client connections that the daemon will accept. If host application processes attempt individually or collectively to exceed this number, the daemon will log a message of "Maximum connections exceeded" and deny access.
Parameters: `<number of connections>`
Example: `ClientConnections 120`
Default: `900`

In some cases, the daemon may impose a lower limit than the requested maximum and log a message to that effect. This action may be necessary because the daemon is constrained to a total of 1024 connections by most operating systems, and reserves six connections per device for internal use. If this number is insufficient, the system administrator may be able to resolve the issue by increasing the `OPEN_MAX` system configuration (`sysconf`) parameter, which the daemon interrogates to ascertain the limit. One way of doing so is as follows.

```
[root@machine ~]# getconf OPEN_MAX
1024
[root@machine ~]# ulimit -n 2048
[root@machine ~]# getconf OPEN_MAX
2048
```

The `ulimit` command affects only the shell in which it is invoked. To apply it to the daemon, the administrator may prefer to add it to the `/etc/init.d/maxeleros` script or that of its parent process.

MaxCoredumpSizeMB

Purpose: Enable core dump files of optionally limited sizes in the event of a MaxelerOS daemon crash. A limit of 0 disables core dumps, and -1 allows core dumps of unlimited size. A limit of 8192 is recommended.
Parameters: `<size in megabytes>`
Example: `MaxCoredumpSizeMB 8192`
Default: `0`

WorkingDirectory

Purpose: Nominate a directory to hold core dumps. To use this option, you should select a directory that already exists or that you create.
Parameters: `<absolute path>`
Example: `WorkingDirectory /var/maxcoredumps`
Default: `/`

3.2.7 Default Configuration File

The default configuration file contents are:

```
LoadIdleOnStart yes
LoadIdleOnTimeout yes
LoadIdleOnShutdown yes
IdleTimeout 600
EnableWatchdog no
WatchdogTimeout 600
EnableTempMonitor no
TempThreshold 67
IPMIDevice /dev/ipmi0
TempSensorId BB Temp
VerifyBitstream no
EnableBoardTempMonitor yes
BoardTempWarning 85
BoardTempIdle 95
BoardTempInterval 60
EnableEventMonitor yes
EnableBoardTempSignal yes
```

3.3 Log Files

MaxelerOS produces a log of daemon activity in `/var/log/maxelerosd.log`, and optionally in `/var/log/syslog`, and rotated log files `/var/log/maxelerosd.log.X`. Example log output:

```
[user@machine ~]$ cat /var/log/maxelerosd.log
Tue Oct  4 15:43:28 2011 --- Running with MaxelerOS 2014.2
Tue Oct  4 15:43:28 2011 --- MaxelerOS library version 2014.2
Tue Oct  4 15:43:28 2011 --- LoadIdleOnStart : yes
Tue Oct  4 15:43:28 2011 --- LoadIdleOnTimeout : yes
Tue Oct  4 15:43:28 2011 --- LoadIdleOnShutdown : yes
Tue Oct  4 15:43:28 2011 --- IdleTimeout : 600s
Tue Oct  4 15:43:28 2011 --- EnableWatchdog : no
Tue Oct  4 15:43:28 2011 --- WatchdogTimeout : 600s
Tue Oct  4 15:43:28 2011 --- EnableTempMonitor : no
Tue Oct  4 15:43:28 2011 --- TempThreshold : 67C
Tue Oct  4 15:43:28 2011 --- IPMIDevice : /dev/ipmi0
Tue Oct  4 15:43:28 2011 --- TempSensorId : BB Temp
Tue Oct  4 15:43:28 2011 --- VerifyBitstream : no
Tue Oct  4 15:43:28 2011 --- EnableBoardTempMonitor : yes
Tue Oct  4 15:43:28 2011 --- EnableBoardTempSignal : yes
Tue Oct  4 15:43:28 2011 --- BoardTempInterval : 60s
Tue Oct  4 15:43:28 2011 --- BoardTempWarning : 85C
Tue Oct  4 15:43:28 2011 --- BoardTempIdle : 95C
Tue Oct  4 15:43:28 2011 --- EnableEventMonitor : yes
Tue Oct  4 15:43:28 2011 --- EnableAllocationGovernor : yes
Tue Oct  4 15:43:28 2011 --- AllocationGovernorInterval : 1s
```

```
Tue Oct  4 15:43:28 2011 --- AllocationGovernorGrowthThreshold : 5
Tue Oct  4 15:43:28 2011 --- AllocationGovernorShrinkThreshold : 5
Tue Oct  4 15:43:28 2011 --- EnableMaxElogServer : no
Tue Oct  4 15:43:28 2011 --- MaxElogLogFile : (none)
Tue Oct  4 15:43:28 2011 --- EnableSyslog : no
Tue Oct  4 15:43:28 2011 --- 1 Maxeler devices found in system.
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0: Spurious CH2 detection. Ignoring.
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0: Device initialised board id =
13424, serial number = 00025560.
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0:         Interface FPGA - Creation
date: 20110418 Version: 1
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0:         Interface FPGA Checksum:
e6f8d8a85b9de003f95eadecaa903e5bd6aa17883e74f8b3f38801c985cfff56
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0:         Interface FPGA FLASH -
Creation date: 20110418 Version: 1
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0:         Interface FPGA FLASH
Checksum: e6f8d8a85b9de003f95eadecaa903e5bd6aa17883e74f8b3f38801c985cfff56
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0:         Interface FPGA SAFE  -
Creation date: 20110418 Version: 1
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0:         Interface FPGA SAFE
Checksum: e6f8d8a85b9de003f95eadecaa903e5bd6aa17883e74f8b3f38801c985cfff56
Tue Oct  4 15:43:30 2011 --- /dev/maxeler0: Loaded idle bitstream
```

3.3.1 Log File Configuration

The MaxelerOS RPM installs settings for rotating log files in `/etc/logrotate.d/maxeleros`. See the manual pages for `logrotate` for information on the format of this file.

3.4 Forcing DFEs to Idle

During the development process, it can be useful to force the DFE to load the idle configuration to ensure that the DFE will be reconfigured when running your application. This can be done using `maxforceidle`, which can either force *all* DFEs to load the idle configuration with the option `-a` or force a specific DFE to load the idle configuration with the option `-d` followed by the DFE number.

For example, to force all DFEs to idle:

```
[user@machine ~]$ maxforceidle -a
Force Idle Tool - MaxelerOS 2014.2
Loading idle configuration onto 2 devices...
        DFE 0 : OK - idle configuration loaded
        DFE 1 : OK - idle configuration loaded
```

To force only the device `0` to idle:

```
[user@machine ~]$ maxforceidle -d 0
Force Idle Tool - MaxelerOS 2014.2
Loading idle configuration onto 1 device...
        DFE 0 : OK - idle configuration loaded
```

The example below shows the output for an MPC-X series node on the IP address 10.101.101.2:

```
[user@machine ~]$maxforceidle -a -r 10.101.101.2
Force Idle Tool - MaxelerOS 2014.2
Loading idle configuration onto 8 devices...
        DFE 0 : OK - idle configuration loaded
        DFE 1 : OK - idle configuration loaded
        DFE 2 : OK - idle configuration loaded
        DFE 3 : OK - idle configuration loaded
        DFE 4 : OK - idle configuration loaded
        DFE 5 : OK - idle configuration loaded
        DFE 6 : OK - idle configuration loaded
        DFE 7 : OK - idle configuration loaded
```

3.5 MaxManifest

MaxelerOS includes the tool `maxmanifest`, which archives system configuration to help remote debugging by Maxeler engineers (e.g. RPMs installed, flexlm licenses, machine configuration). It is recommended to run this from a terminal using the `sudo -E` command to preserve the user's environment.

```
[user@machine ~]$ sudo -E maxmanifest
MaxManifest V2014.2

Generating manifest for machine.company.com ...

Dumping bios information
Obtaining ethernet MAC addresses
Obtaining drive information
Obtaining raid array information
Performing dmidecode dump
Performing lspci dump
Performing dmesg dump
Performing full smartctl
Performing hal dump
Performing ifconfig dumps
Performing route dumps
Performing set dump
Performing iptables dump
Checking redhat-release
Copying log files
Copying maxq logs
Checking kernel version
Creating BIOS dump
Dumping SEL
Dumping sensor status
Checking BMC information
Channel 2 is not a LAN channel
Channel 3 is not a LAN channel
Dumping list of installed packages
```

```
Dumping environment variables
Dumping CPU/RAM/diskspace info
Dumping running processes info
Dumping Maxeler card status

Creating archive...complete

Archive file is: MaxManifest-machine.company.com-20101216-165428.tar.bz2
```

4 MaxTop

MaxTop is a utility for reporting the status of Maxeler cards.

MaxTop is run from a terminal using the command `maxtop`. For example, on a host with multiple MAX3/Vectis cards:

```
[user@machine ~]$ maxtop
MaxTop Tool 2014.2
Found 4 Maxeler card(s) running MaxelerOS 2014.2
Card 0: MAX3A Vectis (P/N: 13424) S/N: 1801010021 Mem: 24GB
Card 1: MAX3A Vectis (P/N: 13424) S/N: 1801010019 Mem: 24GB
Card 2: MAX3A Vectis (P/N: 13424) S/N: 1801010010 Mem: 24GB
Card 3: MAX3A Vectis (P/N: 13424) S/N: 1801010017 Mem: 24GB

Load average: 0.16, 0.03, 0.01

DFE  %BUSY  TEMP   MAXFILE      PID    USER       TIME     COMMAND
 0   55.1%   54.6C  7f9585...   2618   user       00:00:05  acoustic_
 1   54.9%   50.7C  7f9585...   2635   user       00:00:05  acoustic_
 2   54.8%   58.6C  7f9585...   2651   user       00:00:05  acoustic_
 3   55.0%   55.1C  7f9585...   2662   user       00:00:05  acoustic_
```

And on a host with a single MAX2 card:

```
[user@machine ~]$ maxtop
MaxTop Tool 2014.2
Found 1 Maxeler card(s) running MaxelerOS 2014.2
Card 0: MAX2C (P/N: 24414) S/N: 4294967295 Mem: 24GB

Load average: 0.00, 0.00, 0.00

DFE  %BUSY  TEMP   MAXFILE      PID    USER       TIME     COMMAND
 0   0.0%   43.7C  93332fad...  -      -          -        -
 1   0.0%   38.0C  IDLE (r7)    -      -          -        -
```

An MPC-X Series node can be queried remotely using MaxTop with the `-r` option to specify an IP address.

The IP address given to MaxTop must be that of an InfiniBand port on the target MPC-X Series node.

The example below shows the output for an MPC-X Series node on the IP address `10.101.101.2`:

```
[user@machine ~]$ maxtop -r 10.101.101.2
MaxTop Tool 2014.2
Found 8 Maxeler card(s) running 2014.2
Card 0: MAX3A Vectis (P/N: 13424) S/N: 2362601010007 Mem: 24GB
Card 1: MAX3A Vectis (P/N: 13424) S/N: 2339801010012 Mem: 24GB
Card 2: MAX3A Vectis (P/N: 13424) S/N: 2362601010003 Mem: 24GB
Card 3: MAX3A Vectis (P/N: 13424) S/N: 2362601010006 Mem: 24GB
```

```
Card 4: MAX3A Vectis (P/N: 13424) S/N: 2339801010005 Mem: 24GB
Card 5: MAX3A Vectis (P/N: 13424) S/N: 2339801010007 Mem: 24GB
Card 6: MAX3A Vectis (P/N: 13424) S/N: 2362601010001 Mem: 24GB
Card 7: MAX3A Vectis (P/N: 13424) S/N: 2362601010005 Mem: 24GB

Load average: 0.00, 0.00, 0.00

DFE  %BUSY  MAXFILE      HOST         PID    USER       TIME      COMMAND
 0   0.0%   IDLE (r7)    -            -      -          -         -
 1   0.0%   IDLE (r7)    -            -      -          -         -
 2   0.0%   IDLE (r7)    -            -      -          -         -
 3   0.0%   IDLE (r7)    -            -      -          -         -
 4   0.0%   IDLE (r7)    -            -      -          -         -
 5   0.0%   IDLE (r7)    -            -      -          -         -
 6   0.0%   IDLE (r7)    -            -      -          -         -
 7   0.0%   IDLE (r7)    -            -      -          -         -
[user@machine ~]$
```

4.1 Card Information

MaxTop reports the number of cards installed and which version of MaxelerOS they are running. The installed cards are enumerated from 0. For each card, the serial number (`S/N`), part number (`P/N`), amount of memory installed (`Mem`) and the number and type of user DFEs installed are reported. MAX3/Vectis cards have only one user DFE, whereas MAX2 cards have two user DFEs.

4.2 Load Average

The load average section reports three exponential moving averages of this metric

$$C + \sum_{p \in P} R(p)$$

sampled once every second, where

$$\begin{aligned} C &= \text{the number of DFEs currently allocated} \\ P &= \text{the set of processes running on the system} \\ R(p) &= \text{the number of DFEs requested by a process } p \end{aligned}$$

The number of requested DFEs given by $R(p)$ includes cards currently allocated to process p and pending requests.

The three exponential moving averages have different decay parameters chosen to enable monitoring of activity over the past 1 minute, 5 minute, and 15 minute intervals. Load activity prior to these intervals does not contribute significantly to their respective averages.

A distinction exists between load averages and similar metrics noted in the next section. Load averages derive entirely from system-level process statistics. Busy percentages and DFE usage are a measure of the efficiency of hardware utilization resulting from application specific features of the computation.

4.3 DFE Status

The status of each DFE is reported in a table with the following headings:

- `DFE`: the DFE number in the system.
- `%BUSY`: the percentage of time that the DFE has been active during the most recent sampling period.
- `TEMP`: the temperature of the DFE measured in Celsius.
- `MAXFILE`: one of three things,
  - the first few characters of the checksum of the configuration currently loaded onto the DFE
  - the word `IDLE (r`*n*`)` with the revision number *n* if the DFE has an idle configuration currently loaded
  - the word `<DISABLED>` if the DFE is blacklisted due to overheating or any prior failed attempt at configuration
- `PID`: the process ID of a user application currently connected to the DFE.
- `USER`: the username of the user who launched the application currently connected to the DFE.
- `TIME`: the time for which the user process has been running.
- `COMMAND`: the command which launched the user process.

If the pass-through example from MaxCompiler (modified to run long enough to run MaxTop simultaneously) is running, then the following DFE status information is shown:

```
DFE  %BUSY   TEMP   MAXFILE        PID    USER       TIME      COMMAND
 0   0.0%    48.3C  5933d5...      18791  user       00:00:17  PassThrou
 1   0.0%    51.1C  IDLE (r4)      -      -          -         -
```

This shows that `user` launched the process ID `18791` from a command starting with the characters `PassThrou` and has been running for 17 seconds.

Once the application has closed its connection to the DFE, the process information no longer appears, but the DFE is still configured with the same maxfile until reconfigured:

```
DFE  %BUSY   TEMP   MAXFILE        PID    USER       TIME      COMMAND
 0   0.0%    47.6C  5933d5...      -      -          -         -
 1   0.0%    50.1C  IDLE (r4)      -      -          -         -
```

The checksum for a design is stored in the maxfile and can be retrieved using the `tail` command:

```
[user@machine ~]$ tail -n 2 PassThrough.max
CHECKSUM("5933d5f0bcf189550d0bc0cc3b5f45660b18bd2c058d98f6131e3f4423175bd0")
#endif
```

4.4 Verbose MaxTop Output

More information can be retrieved using the `-v` option on MaxTop. The output is slightly different between different DFEs.

4.4.1 MAX2 Verbose MaxTop Output

```
[user@machine ~]$ maxtop -v
MaxTop Tool 2014.2
Found 1 Maxeler card(s) running MaxelerOS 2014.2
Card 0: MAX2C (P/N: 24414) S/N: 4294967295 Mem: 24GB

DFE  %BUSY   TEMP   MAXFILE        PID    USER       TIME      COMMAND
 0   68.3%   55.8C  dbe06a...      23120  user       00:08:10  pFDmodel2
 1   67.9%   53.3C  dbe06a...      23120  user       00:08:10  pFDmodel2

/dev/maxeler0:
        MAX2C (P/N: 24414) S/N: 000905005
        Mem: DDR2-24GB
        Device capabilities: 'MAX2REVC,SXT240_2C,DDR2_24GB'
        FPGA: V5-SXT240
                Temperature: FPGA 55.8C
                Voltages: VccInt 0.964V  VccAux 2.505V
        Bitstream: app_id: -1 app_rev:-1 checksum:
dbe06a5770e1348173719bfa0919fec8eeed4ef5516b165438033eb51c9fc561
        PCIe: x8 gen1
        Device usage:
                Performance monitoring core version: 1
                FPGA usage: 68.3%
                DRAM: Parity disabled - ECC disabled
        Inter FPGA links: 1 connection
                Connection to device /dev/maxeler1 has capabilities '
FPGA_ON_LOCAL_CARD'
...
```

- `MAX2C (P/N: 24414) S/N`: the serial number of the device.
- `Mem`: the type and amount of DRAM on the card.
- `FPGA`: the part number of the FPGA on the card.
- `Bitstream`: details about the currently loaded bitstream.
  - `app_id`: ID of the application as set when the design was built.
  - `app_rev`: revision number of the application as set when the design was built.
  - `checksum`: the full checksum of the bitstream.
- `PCIe`: PCI-Express link information. In the example above, the card is running an 8-lane PCI-Express generation 1 link.
- `Device usage`: exponential moving average of Kernel activity sampled at 0.1 second intervals, and for applications built performance monitoring core version 2 or later, also 1 second intervals
- `Inter FPGA links`: MaxRing topology reporting (see *subsection 4.5*).
- `Temperature`: the current temperature of the FPGA.

- `Voltage`: current voltages on the card. A Maxeler engineer debugging potential hardware issues may require this voltage information.

4.4.2 MAX3/Vectis Verbose MaxTop Output

```
[user@machine ~]$ maxtop -v
MaxTop Tool  2014.2
Found 1 Maxeler card(s) running MaxelerOS 2014.2
Card 0: MAX3A Vectis (P/N: 13424) S/N: 901010007 Mem: 24GB FPGA(s): 1 V6-SXT475
        /dev/maxeler0

Load average: 0.00, 0.00, 0.00

ENGINE     %BUSY   TEMP   MAXFILE        PID    USER       TIME      COMMAND
maxeler0   0.0%    36.9C  bee533...      -      -          -         -

/dev/maxeler0:
        MAX3A Vectis (P/N: 13424) S/N: 901010007
        Mem: DDR3-24GB
        Device capabilities: 'MAX3REVA,SXT475_2ES,DDR3_24GB'
        Interface FPGA: V6-LXT75
                Temperature: FPGA 41.3C
                Voltages: VccInt 0.999V  VccAux 2.517V
                Voltages: Slot12V0 11.935V  Aux12V0 12.005V
                Voltages: Slot3V3 3.402V  Flash1V8 -
                Interface FPGA checksum:
f81384001c2b4b7804b6dc57d75d67eca84d76b7a03f9b2cd54c1b5293fa6d71
                Interface FPGA build date 20110708, rev2
                Interface FPGA FLASH checksum:
f81384001c2b4b7804b6dc57d75d67eca84d76b7a03f9b2cd54c1b5293fa6d71
                Interface FPGA FLASH build date 20110708, rev2
                Interface FPGA SAFE checksum:
f81384001c2b4b7804b6dc57d75d67eca84d76b7a03f9b2cd54c1b5293fa6d71
                Interface FPGA SAFE build date 20110708, rev1
        Compute FPGA: V6-SXT475
                Temperature: FPGA 36.9C
                Voltages: VccInt 0.999V  VccAux 2.520V
        Bitstream: app_id: 0 app_rev: 0 checksum:
bee533ad510d9792f35825f6ba960b8fe333384c52d70821057e812199a2a8b8
        PCIe: x8 gen2
        Device usage:
                Performance monitoring core version: 1
                FPGA usage: 0.0%
                DRAM: Parity disabled - ECC disabled
        Inter FPGA links: 0 connection
        CH2 network adapter (P/N: 311) S/N: 2164602010008
                SFP1 MAC B8:CD:A7:57:99:81
                SFP2 MAC B8:CD:A7:57:99:82
```

```
                CX4  MAC B8:CD:A7:57:99:84
```

- `MAX3A (P/N: 3324) S/N`: the serial number of the device.
- `Mem`: the type and amount of DRAM on the card.
- `Device capabilities`: a string that can be used by a design to check specific capabilities for a card.
- `Interface FPGA`: the part number of the PCI Express interface FPGA on the card.
  - `Temperature`: the current temperature of the interface FPGA.
  - `Voltages`: current voltages related to the interface FPGA.
- `Compute FPGA`: the part number of the PCI Express compute FPGA on the card.
  - `Temperature`: the current temperature of the compute FPGA.
  - `Voltages`: current voltages related to the compute FPGA.
- `Bitstream`: details about the currently loaded bitstream.
  - `app_id`: ID of the application as set when the design was built.
  - `app_rev`: revision number of the application as set when the design was built.
  - `checksum`: the full checksum of the bitstream.
- `PCIe`: PCI-Express link information. In the example above, the card is running an 8-lane PCI-Express generation 1 link.
- `Device usage:` exponential moving average of Kernel activity sampled at 0.1 second intervals, and for applications built performance monitoring core version 2 or later, and also 1 second intervals
- `Inter FPGA links`: MaxRing topology reporting (see *subsection 4.5*).
- `CH2 network adapter`: Serial number and recommend MAC addresses of CH2 network adapter cards (on MaxNode10G systems only)

4.4.3 Coria & Maia Verbose MaxTop Output

```
[user@machine ~]$ maxtop -v
MaxTop Tool 2014.2
Found 1 Maxeler card(s) running MaxelerOS 2014.2
Card 0: Maia (P/N: 4848) S/N: 2401901010003 Mem: 48GB

Load average: 0.33, 0.68, 0.61

DFE  %BUSY  TEMP   MAXFILE        PID    USER       TIME      COMMAND
 0   0.0%   40.4C  d994fa81c2...  -      -          -         -

/dev/maxeler0:
        Maia (P/N: 4848) S/N: 2401901010003
```

```
        Mem: DDR3-48GB
        Device capabilities: 'MAIAREVA,5SGSMD8N2F45C2,DDR3_48GB'
        FPGA: SV-D8
                Temperature: FPGA 40.4C
                Voltages: VccInt - VccAux -
        Bitstream: app_id: -1 app_rev:-1 checksum:
d994fa81c251207f1f067a59ba33c2efc60b95a5463d83e20f796d6ab084134f
        Estimated power usage: 19.56 W
        PCIe: x4 gen1
        Device usage:
                Performance monitoring core version: 2
                FPGA usage: 0.0%
                FPGA medium-term usage: 0.0%
                DRAM: Parity enabled - ECC enabled
                DRAM: Uncorrected errors: 0
                DRAM: Corrected errors: 0
        Inter FPGA links: 0 connections
```

4.5 MaxTop Topology Reporting

The verbose output of MaxTop reports the topology of the links between cards within a system.

4.5.1 MAX3/Vectis Topology Reporting

The example below shows abridged output for a system with 4 MAX3/Vectis cards:

```
[user@machine ~]$ maxtop -v
MaxTop Tool 2014.2
Found 4 Maxeler card(s) running MaxelerOS 2014.2
...
DFE  %BUSY   TEMP   MAXFILE        PID    USER       TIME      COMMAND
 0   0.0%    40.2C  IDLE (r5)      -      -          -         -
 1   0.0%    38.4C  IDLE (r5)      -      -          -         -
 2   0.0%    43.0C  IDLE (r5)      -      -          -         -
 3   0.0%    42.6C  IDLE (r5)      -      -          -         -
...
/dev/maxeler0:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler1 has capabilities 'MAXRING_A'
                Connection to device /dev/maxeler3 has capabilities 'MAXRING_B'
/dev/maxeler1:
...
        Inter FPGA links: 1 connection
                Connection to device /dev/maxeler0 has capabilities 'MAXRING_A'
/dev/maxeler2:
...
        Inter FPGA links: 1 connection
                Connection to device /dev/maxeler3 has capabilities 'MAXRING_A'
/dev/maxeler3:
```

```
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler0 has capabilities 'MAXRING_B'
                Connection to device /dev/maxeler2 has capabilities 'MAXRING_A'
```

The links between devices on a MAX3/Vectis card are all MaxRing links between devices on different cards. A MaxCompiler Manager design specifies whether a `MAXRING_A` or `MAXRING_B` link is required for each MaxRing connection.

The topology for the output shown again is shown in *Figure 1*.

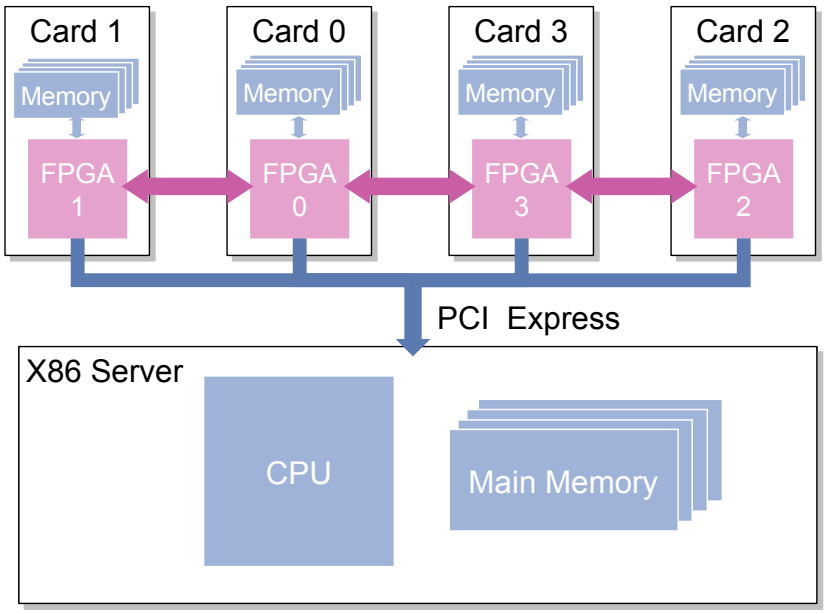

Figure 1: Topology of MAX3/Vectis cards matching MaxTop output in *subsubsection 4.5.1*.

4.5.2 MPC-X Topology Reporting

The example below shows abridged output for an MPC-X system with 8 MAIA cards:

```
[user@machine ~]$ maxtop -v -r 10.101.101.2
MaxTop Tool 2014.2
Found 8 Maxeler card(s) running MaxelerOS 2014.2
...
DFE  %BUSY  MAXFILE      HOST         PID    USER       TIME       COMMAND
 0   0.0%   IDLE (r7)    -            -      -          -          -
 1   0.0%   IDLE (r7)    -            -      -          -          -
 2   0.0%   IDLE (r7)    -            -      -          -          -
 3   0.0%   IDLE (r7)    -            -      -          -          -
 4   0.0%   IDLE (r7)    -            -      -          -          -
 5   0.0%   IDLE (r7)    -            -      -          -          -
 6   0.0%   IDLE (r7)    -            -      -          -          -
 7   0.0%   IDLE (r7)    -            -      -          -          -

/dev/maxeler0:
```

```
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler1 has capabilities 'MAXRING_A'
                Connection to device /dev/maxeler4 has capabilities 'MAXRING_B'
/dev/maxeler1:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler0 has capabilities 'MAXRING_A'
                Connection to device /dev/maxeler2 has capabilities 'MAXRING_B'
/dev/maxeler2:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler1 has capabilities 'MAXRING_B'
                Connection to device /dev/maxeler3 has capabilities 'MAXRING_A'
/dev/maxeler3:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler2 has capabilities 'MAXRING_A'
                Connection to device /dev/maxeler6 has capabilities 'MAXRING_B'
/dev/maxeler4:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler0 has capabilities 'MAXRING_B'
                Connection to device /dev/maxeler5 has capabilities 'MAXRING_A'
/dev/maxeler5:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler4 has capabilities 'MAXRING_A'
                Connection to device /dev/maxeler7 has capabilities 'MAXRING_B'
/dev/maxeler6:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler3 has capabilities 'MAXRING_B'
                Connection to device /dev/maxeler7 has capabilities 'MAXRING_A'
/dev/maxeler7:
...
        Inter FPGA links: 2 connections
                Connection to device /dev/maxeler5 has capabilities 'MAXRING_B'
                Connection to device /dev/maxeler6 has capabilities 'MAXRING_A'
```

5 Monitoring System Health

With local DFEs MaxelerOS supports “out of band” system health checking, for monitoring the health of running systems while applications are in use. A number of checks are supported:

- Checking that the available system hardware configuration matches the expected configuration.
- Checking for error events on cards, such as power supply or over-temperature errors.

Errors due to either kind of check can be accessed either through reading the MaxelerOS daemon log file, or by running `maxstatuscheck` to report if any errors are present (see *subsection 5.2*).

5.1 Checking system configuration

By default on system start-up MaxelerOS will enumerate the available cards in a system and make those available for use by applications. The available hardware can be displayed by running `maxtop`.

However, in a production setting it may be useful to inform MaxelerOS that a certain hardware configuration is expected, and that if the configuration differs for this to be reported as an error. For example, if a compute node contains 4 Vectis cards, but only 3 are available on boot, this would allow MaxelerOS to alert the system administrator that one card was not working correctly.

MaxelerOS allows this to be done by specifying the required configuration in the hardware topology file: `/etc/maxeler/maxelerosd.topology`

When it is loaded the MaxelerOS daemon will evaluate whether the available hardware matches the specification and log an error if there is a problem.

Hardware topology files are specific to particular hardware configurations and a file for your particular compute node configuration will be provided by Maxeler. An example topology file contents is shown below:

```
D:/dev/maxeler0:V6SXT475_2ES
D:/dev/maxeler1:V6SXT475_2ES
C:0:1:MAXRING_A
```

Entries beginning with ’D’ describe a required device, entries beginning with ’C’ describe a required connection between devices. In the example above, two cards are required, `maxeler0` and `maxeler1`, both of which are Vectis cards with V6-SXT475 FPGAs. A MaxRing connection is expected between the cards. The general format for a required device is: `D:<devicename>:<FPGAtype>`.

The general format for a required connection is: `C:<device1>:<device2>:<connectiontype>`.

If you change the configuration of your hardware you will need to modify the topology file in order to reflect the new configuration. To disable system configuration checking all together, simply delete or rename the topology file.

5.2 Checking the System Status

The MaxelerOS daemon logs system health information to its log file (`/var/log/maxelerosd.log`). The information can also be accessed in an easily machine readable format for integration into automatic monitoring systems by running `maxstatuscheck`.

```
[user@machine ~]$ maxstatuscheck
Daemon status check tool - MaxelerOS 2014.2
Devices match expected configuration: Yes
Power supply events: 0
```

```
FPGA temperature warnings: 0
FPGA temperature idle requests: 0
FPGA DRAM uncorrected errors: 0
FPGA DRAM corrected errors: 0
```

The exit status value can be used in an application or script to check the status programmatically, with the following usage (obtained by running `maxstatuscheck -h`):

```
usage: maxstatuscheck [-c] [-e] [-i] [-p] [-t] [-w]
   -c: ignore FPGA DRAM corrected errors
   -e: ignore FPGA DRAM uncorrected errors
   -i: ignore device configuration status
   -p: ignore power supply events
   -t: ignore FPGA temperature idle requests
   -w: ignore FPGA temperature warnings

Unless ignored, the bits in the exit status are:
   bit 1: Invocation or communication error
   bit 2: Device status mismatch
   bit 3: Non-zero power supply events
   bit 4: Non-zero FPGA temperature warnings
   bit 5: Non-zero FPGA temperature idle requests
   bit 6: Non-zero FPGA DRAM uncorrected errors
   bit 7: Non-zero FPGA DRAM corrected errors
```

`maxstatuscheck` is designed to be incorporated into test scripts with minimal effort. Particular classes of error can be masked out using either command line flags (`-c, -e, -i, -p, -t, -w`) or by masking the returned error code externally.

5.2.1 Power Supply Events

The figure reported for power supply events by `maxcheckstatus` refers to the total number of entries logged in `/var/log/maxelerosd.log` containing any of the following diagnostics:

```
MAX3_PMBUS_ALERT
MAX3_DDR_EVENT
MAX3_VREFCA_EVENT
MAX3_VREFDQ_RIGHT_EVENT
MAX3_VREFDQ_LEFT_EVENT
MAX3_CTL1_POWER_EVENT
MAX3_CTL2_POWER_EVENT
```

A moderate number of power supply events can be expected, but the log file should be checked if an usually large number is reported.

5.2.2 Temperature Idle Requests

This figure refers to the total number of messages in `/var/log/maxelerosd.log` of a card's temperature exceeding its designated safe level. On those occasions, an entry of this form is logged:

⟨*time stamp*⟩ ⟨*device*⟩ `FPGA temperature` ⟨*temp*⟩ `- too high`

The card is also automatically taken out of service by being loaded with an idle configuration.

Temperature idle requests are more extreme conditions than temperature warnings, which pertain to lower temperature levels that do not warrant taking a device out of service. The log file entries for temperature warnings take a similar form but omit the words "`too high`". See *subsubsection 3.2.2* for related information.

6 Card Diagnostics

MaxelerOS includes diagnostic utilities for performing hardware checks on installed Maxeler cards. MaxDiag checks the different subsystems on a card, including power supplies, PCI Express bus and on-card DRAM memory. For systems with CH2 network interface cards installed, `ch2diag` checks the connections between the FPGA card, the MaxRing, the PHY (physical layer interface) chip and the SFP modules.

The diagnostics operate by loading a special test configuration onto the DFE. You must therefore stop other applications that might be using the DFE before you can run a diagnostic. For a more general high-level system health check that can run while applications are using the cards, you can run `maxstatuscheck` (see *section 5*).

6.1 Running Diagnostics

The diagnostic capabilities of different cards vary, so you will need to run the appropriate diagnostic utility for the specific hardware you are using.

Card	MaxDiag
MAX2	`max2diag`
MAX3/Vectis	`max3diag`
Maia & Coria	`max4diag`
CH2	`ch2diag`

You can identify the cards present in your system by running `maxtop`. You can also verify the presence of a CH2 network interface card by running `maxtop -v`.

These utilities output detailed information to the console, the test output can be captured by redirecting the console output to a file. The utilities output PASSED/FAILED status, and will return a code of zero if successful, or non-zero if any test has failed. In the event of a failure, the test output can be useful to diagnose a particular fault with the card, memory modules or the host system.

MaxDiag Running `max3diag --help` (or equivalent for `max2diag` and `ch2diag`) will display a status message showing the available command line options. To select the card to test, specify the device with the option `-d` followed by the DFE number. For example, to test the second MAX3/Vectis device in a system:

```
[user@machine ~]$ max3diag -d 1
```


There is no default setting for the `-d` option.

The `max3diag` utility takes two additional options. `-q` selects a quick test mode, where only an abridged memory test will be performed. A full MAX3/Vectis diagnostic test will take several minutes, but the quick test can complete in a few seconds. `-l` enables detailed logging for the full memory test (assuming not in quick test mode).

ch2diag The `ch2diag` utility takes the additional options `-p` (or `--packets`) and `-l` (or `--loopback`).

- The `--packets` option followed by an unsigned integer specifies the number of packets to transfer during the test. The default is 1000000. A higher number requests a longer running test that is more likely to detect rarely manifested errors.
- If the `--loopback` option is omitted, a non-loopback mode test is indicated. The non-loopback test relies on the presence of a 10G optical cable to carry test data between the two SFP ports on a single CH2 card. This cable must be physically connected by the user or system administrator prior to the test, thereby taking the card off the local network for the duration.
- If the `--loopback` option is specified, a loopback test is indicated. A loopback test can be done without disconnecting the card from the network because it does not cause any data to be transmitted externally, but it requires the SFP modules to be installed (as shipped), even if the card is not connected to the network.
- The `--loopback` option can be followed on the command line by a mode parameter determining how thoroughly the hardware is to be tested. The mode can be one of `pma_system`, `pcs`, or `xgxs_system`. Setting this parameter is harmless but not useful except in consultation with a Maxeler engineer.

6.2 `max3diag` Output

`max3diag` prints a large quantity of diagnostic information to the console. A sample is shown below:

```
[user@machine ~]$ max3diag -d 0
------------------------------------------------------------
MAX3 Diagnostics
------------------------------------------------------------
Found part number: 3424, serial number:801010013 for /dev/maxeler0
Successfully opened /dev/maxeler0

------------------------------------------------------------
Test 1 - FPGA Temperatures and Voltages
------------------------------------------------------------
1.1 - Interface FPGA
    Temperature     41.342 [C]  OK
    VccInt           0.996 [V]  OK
    VccAux           2.508 [V]  OK
1.2 - Compute FPGA
    Temperature     51.678 [C]  OK
    VccInt           0.996 [V]  OK
    VccAux           2.511 [V]  OK
1.3 - Board Voltages:
    PCIe aux power (12V0)      0.095 [V]    OK
    PCIe slot power (12V0)    11.864 [V]    OK
    PCIe slot power (3V3)      3.331 [V]    OK
    Config FLASH power (1V8)   1.825 [V]    OK
```

6.2.1 Test 1 - FPGA Temperatures and Voltages

This test checks the core temperatures and critical power supplies for FPGAs on the MAX3/Vectis card. A failure on a temperature test may indicate a fan/airflow failure in the host system. A failure on any of the PCIe power supplies (PCIe aux or slot power) indicates that the host system is not supplying the correct power to the FPGA card, or that the on-card instrumentation has failed.

6.2.2 Test 2 - Power Controller Temperatures and Voltages

The MAX3/Vectis card incorporates a digitally controlled power supply that continually monitors all voltages, currents and temperatures of key power supply components. A failure in this section indicates that an on-card power supply has failed or that FPGAs or memory on the card are drawing a significantly higher or lower than expected current while running the test.

This test also dumps the contents of internal status registers and device information that may be useful to Maxeler in diagnosis of power supply faults in the field.

6.2.3 Test 3 - DIMM Information

This test dumps certain information from the memory SODIMM modules on the MAX3/Vectis card, including part and serial numbers for each module. A failure in this test indicates a memory module is missing or that the EEPROM on that module has failed.

6.2.4 Test 4 - Memory PHY Test

A failure in this test indicates a critical failure of the memory system on the card. DDR3 memory requires a read/write leveling process to complete before any data or test patterns can be read/written. This process requires a minimum level of functionality and may fail if, for example, a SODIMM is not seated correctly or a device has failed. If the training process fails, the diagnostic will dump the status of the training registers which may be used by Maxeler to narrow down a specific fault.

6.2.5 Test 5 - Bus Test

This test writes test patterns to a subset of the memory and identifies any faults in connectivity between the FPGAs and memory devices. In the event of a failure, the test outputs a summary of errors by SODIMM module (to narrow down to a specific module) and byte lane (to narrow down to a specific device on a module).

The example output below shows a failure of byte lane 4 on the SODIMM module in socket J3.

```
5.1 - Walking Ones Test                Failed to verify data looped back into
DIMMs.
NumErrs:  By0   By1   By2   By3   By4   By5   By6   By7
DIMM J1   0     0     0     0     0     0     0     0
DIMM J2   0     0     0     0     0     0     0     0
DIMM J3   0     0     0     0     8     0     0     0
DIMM J4   0     0     0     0     0     0     0     0
DIMM J5   0     0     0     0     0     0     0     0
DIMM J6   0     0     0     0     0     0     0     0
```

6.2.6 Test 6 - Memory Hammer Test

This test writes test patterns to the entirety of the memory and identifies hard faults (for example stuck at one, stuck at zero) in the memory. Writing and reading back the entire memory can take a few minutes, depending on host system speed, so this test is skipped if the -q option is supplied. In the event of a failure, the failed data patterns are output to the console together with an error summary in the same format as the Bus Test.

6.3 `max4diag` Output

`max4diag` prints a large quantity of diagnostic information to the console. A sample is shown below:

```
[user@machine ~]$ max4diag -d 0
------------------------------------------------------------
MAX4 Diagnostics
------------------------------------------------------------
Found part number: 4848 for /dev/maxeler0
Found serial number: 2401901010003 for /dev/maxeler0
Successfully opened /dev/maxeler0

------------------------------------------------------------
Test 1 - FPGA Temperatures and Voltages
------------------------------------------------------------
1.1 - FPGA
        Temperature             39.007 [C]      OK
        VccInt                   0.000 [V]      N/A
        VccAux                   0.000 [V]      N/A
```

6.3.1 Test 1 - Test 7

The various tests present information similar to that of `max3diag`.

6.4 `max2diag` Output

`max2diag` prints a large quantity of diagnostic information to the console. A sample is shown below:

```
[user@machine ~]$ max2diag -d /dev/maxeler0

Found part number: 24414, serial number 928003 for /dev/maxeler0
Testing memory DIMM size = 4GB

------------------------------------------------------------
Test 1, FPGA U28 :            System monitor
    1.1.1 :               Temperature OK    Temp=51.854

------------------------------------------------------------
Test 1, FPGA U14 :            System monitor
    1.2.1 :               Temperature OK    Temp=53.271

------------------------------------------------------------
```

6.4.1 Test 1 - System monitor

Each FPGA on the MAX2 includes a temperature monitor. A failure in this test indicates a fan/airflow failure in the host system or that the on-card monitoring infrastructure has failed.

6.4.2 Test 2 - Card monitor registers

This test checks the status of the Inter-FPGA MaxRing link between the two FPGAs and the DDR2 memory training on the MAX2 card. The DDR2 memory training fails if there is a critical memory error, for example if a SODIMM module is not seated correctly or if a device has failed. In the event of a DDR2 training failure, the test outputs the failed FPGA (FPGA 1 or 2) to the console.

6.4.3 Test 3 - Loopback

This test loops back test patterns over the PCI Express and both PCI Express and Inter-FPGA links. A failure here indicates that the link between the host and FPGA has failed.

6.4.4 Test 4 - Bus Test

This test writes test patterns to a subset of the memory and identifies any faults in connectivity between the FPGAs and memory devices. In the event of a failure, the test outputs the socket(s) containing failed SODIMM module(s).

The example output below shows a failure of the SODIMM module in socket J5.

```
------------------------------------------------------------
Test 4, FPGA U28 :                 Bus tests
        4.1.1 :              Walking ones
                SODIMM in J5 on FPGA U28 failed.
```

6.4.5 Test 5 - RAM tests

This test writes test patterns to the entirety of the memory and identifies hard faults (for example stuck at one, stuck at zero) in the memory. Writing and reading back the entire memory can take a few minutes, depending on host system speed, so this test is skipped if the -q option is supplied. In the event of a failure, the test outputs the failed SODIMM module(s) details in the same format as the Bus Test.

6.5 `ch2diag` Output

Here is an example of an abbreviated console session for a successful run of `ch2diag`.

```
[root@machine ~]# ch2diag -d /dev/maxeler0 -lpma_system
CH2 Diagnostics
===============

Found part number: 3224, serial number:7086 for /dev/maxeler0
Opening and configuring FPGA.
CH2 Device Found. serial number:2164602010005, part number:311

Status Regs:
    CH2_SFP1 - sync_status: 0xF
```

```
    CH2_SFP1 - align_status: 0x1

⋮

MAC configurations:
    CH2_SFP1_MAC.RX_CONFIG_0:           0x00000000
    CH2_SFP1_MAC.RX_CONFIG_1:           0x10000000

⋮

MDIO configurations:
    CH2_SFP1_MAC PMA/PMD Control 1:       0x0006
    CH2_SFP1_MAC PMA/PMD Status 1:        0x2006

⋮

Setting MAC addresses. SFP1: 0xB8CDA7579951, SFP2: 0xB8CDA7579952
Disabling pause
Enabled jumbo frames
Enabled PMA System Loopback
Setting Stream Clock frequency to 180 MHz

Running loopback for 1000000 packets...
Packets RX SFP1: 296426, SFP2: 296204, out of 1000000
Packets RX SFP1: 592586, SFP2: 592064, out of 1000000
Packets RX SFP1: 888196, SFP2: 887519, out of 1000000
Packets RX SFP1: 1000000, SFP2: 1000000, out of 1000000

End of test status (SFP1):
    packets: 1000000
    dest_mac: 0xB8CDA7579952
    source_mac: 0xB8CDA7579951
    ethertype: 0x0080
    lfsr64_init: 0x0123456789ABCDEF
    tx_busy: 0
    rx_busy: 0
    packets_sent: 1000000
    packets_received: 1000000
    packets_good: 1000000

⋮

    CH2_SFP1_MAC Frames TX: 1000000
    CH2_SFP1_MAC Frames RX: 1000000
    CH2_SFP2_MAC Frames TX: 1000000
    CH2_SFP2_MAC Frames RX: 1000000
TEST PASSED!
Disabled Internal Loopback
```

The main item of interest is the test result shown on the penultimate line. A test can pass as shown above, or fail due to disconnected cables or SFP modules among other reasons. In those cases, the output may include the lines

```
TEST FAILED!
Make sure the optical cable is connected properly.
```

for a non-loopback test, or

```
TEST FAILED!
Make sure the SFP modules are connected properly.
```

for a loopback test. If the reason for a failure is other than a simple connection problem, the remaining output is useful to a Maxeler engineer in diagnosing it.

7 Troubleshooting

This section describes how to detect and resolve some potential issues with MaxelerOS.

7.1 Failed to connect to local MaxelerOS daemon

If you get this error when running an application:

```
SLiC Error #101 @ topology_internal.c:1503 - Failed to connect:
        * If you are running a simulation, check that "use_simulation=<
simulator_name>" is set correctly in your SLiC configuration
          (via the SLIC_CONF environment setting, or in your \$SLIC_CONF_FILE
or ~/.MaxCompiler_slic_user.conf file.)
        * If you are running on hardware, you may need to restart the MaxelerOS
 daemon.
                (socket: (null), MaxelerOS daemon error: Socket connect/close
failure)
SLiC Error #506 @ group_standard_internal.c:402 - Topology request failed
SLiC Error #518 @ group.c:26 - Error reported from function "max_load_group".
```

This is because the MaxelerOS daemon is not running. This can be confirmed by running MaxTop:

```
[user@machine ~]$ maxtop
MaxTop Tool 2014.2
Maxeler Error #101 - Failed to connect to MaxelerOS daemon, restart MaxelerOS
daemon
Maxeler Error #101 - Failed to connect to MaxelerOS daemon, restart MaxelerOS
daemon
Maxeler Error #184 - Failed to connect to MaxelerOS driver, reload MaxelerOS
driver.
Found Maxeler card 1bbf:0004 at location 17:00
```

The driver and daemon can be restarted following the instructions in *section 3*: *Administration*.

7.2 Failed to connect to MPC-X MaxelerOS daemon

If you get this error when running an application for an MPC-X:

```
SLiC Error #101 @ topology_internal.c:1503 - Failed to connect:
        * If you are running a simulation, check that "use_simulation=<
simulator_name>" is set correctly in your SLiC configuration
          (via the SLIC_CONF environment setting, or in your \$SLIC_CONF_FILE
or ~/.MaxCompiler_slic_user.conf file.)
        * If you are running on hardware, you may need to restart the MaxelerOS
 daemon.
                (socket: pandora://44.44.44.44:18515, MaxelerOS daemon error:
Socket connect/close failure)
SLiC Error #506 @ group_standard_internal.c:402 - Topology request failed
SLiC Error #518 @ group.c:26 - Error reported from function "max_load_group".
```

This is because the MPC-X is not connected to the network or not ready. This can be confirmed by running MaxTop:

```
[user@machine ~]$ maxtop -r 44.44.44.44
MaxTop Tool 2014.2
Maxeler Error #101 - Failed to connect to MaxelerOS daemon, restart MaxelerOS
daemon
```

7.3 Cards not present or not detected

If no Maxeler cards are present in the system or they have not been detected by the operating system, then MaxTop will report this error:

```
[user@machine ~]$ maxtop
MaxTop Tool 2014.2
Maxeler Error #101 - Failed to connect to MaxelerOS daemon, restart MaxelerOS
daemon
Maxeler Error #101 - Failed to connect to MaxelerOS daemon, restart MaxelerOS
daemon
Maxeler Error #184 - Failed to connect to MaxelerOS driver, reload MaxelerOS
driver.
Maxeler Error #184 - No Maxeler cards found on PCI bus
Maxeler Error #184 - No Maxeler cards present on the system
```

If there are no cards in the node and this command was intended for an MPC-X system then supply the `-r` option with IP address.

If there is a card in the system, power-cycling the system will force the PCI-Express interface to reinitialize.

If you have access to the back of the machine, you can check the status LEDs on the card. On MAX2 cards, a lit green LED means that the card has started correctly and a red LED indicates an error. On a MAX3/Vectis card, illumination of three green LEDs means it has started correctly, and illumination of fewer than three indicates an error.

If the problem persists, contact Maxeler.