Qualcomm Technologies, Inc.

Qualcomm® Hexagon™ QEMU

User Guide

80-N2040-52 Rev. AC

October 7, 2022

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Contents

1 Introduction

This document describes the Qualcomm® Hexagon™ QEMU-VP utility, which is an emulator that decodes and translates the instructions of a DSP program into instructions for a host architecture on a virtual platform.

1.1 Conventions

Computer text, code names, and code samples appear in a different font, for example, `printf("Hello world\n")`.

The following notation is used to define command syntax:

- Square brackets enclose optional items, for example, `[label]`.
- **Bold** indicates literal symbols, for example, `[`**`comment`**`]`.
- The vertical bar character, `|`, indicates a choice of items.
- Parentheses enclose a choice of items, for example, `(`**`add`**`|`**`del`**`)`.
- An ellipsis, `. . .` , follows items that can appear more than once.

1.2 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to CreatePoint, register for access or send email to qualcomm.support@qti.qualcomm.com.

2 Overview

QEMU is designed differently from the Hexagon simulator. Instead of simulating processor cycles, the instructions of a DSP program are decoded and dynamically translated into instructions for the host architecture. The execution will accurately emulate the architectural behavior of the Hexagon DSP. Instructions are translated just-in-time—as they are encountered during execution. This translation cost is only paid once; executing the same code does not require additional translation. Thus, the translation gives QEMU a performance advantage over the simulator.

With QEMU, the default relationship between guest instruction execution and guest system clocks is different from the simulator or the target. Some DSP instructions require more host instructions than others, which can cause some differences in how those clocks appear to elapse during guest code emulation.

The software on the emulated DSP (the guest) can access the host system via a set of semi-hosted *angel calls*, which are system calls that the emulator handles to access the file system or console. The QEMU interface matches the Hexagon simulator interface.

2.1 Host system requirements

Releases of QEMU are provided for Linux. The Linux binaries will work on Ubuntu 18.

2.2 Required platform libraries

QEMU binaries distributed with the Hexagon Tools require the following system libraries to be installed:

- libasound
- libc
- libepoxy
- libgcc_s
- libgio
- libglib
- libgmodule
- libgobject
- libm
- libpixman
- libpthread
- libpulse
- libutil
- libz

3 Use QEMU

QEMU provides many command line arguments to customize how the emulator is to run. Several important options for the Hexagon processor are described in this document. These options allow you to customize how the emulator is to run.

3.1 Command

The QEMU binary is called `qemu-system-hexagon`. To start the emulator from the command line:

`qemu-system-hexagon` [*option...*]

Where *option* is one or more of the following arguments for the Hexagon processor.

3.2 Options

Get help (Section 3.2.1)

```
--help
```

System configuration (Section 3.2.2)

```
-cpu any
-cpu any,count-gcycle-xt=on,usefs=<dir>
-m 6G
```

Timing (Section 3.2.3)

```
-icount [shift=N|auto]
```

Miscellaneous (Section 3.2.4)

```
-append 'x y z 1 2 3'
-gdb tcp::1234        |  -s
-kernel file
--machine V68N_1024   | -M V68N_1024
-monitor [none|stdio]
-S
```

3.2.1 Get help

-help

Display command usage information and then exit. For example, to show help on how to specify the system memory:

```
qemu-system-hexagon -m help
```

3.2.2 System configuration

-cpu any,[*option,option,...***]**

System configuration option that launches a system using a generic Hexagon CPU.

QEMU does not have the capability to select different architectures. The most recent DSP architecture is enabled, and it will support instructions and coprocessors introduced in older architectures.

Additional options are comma-separated and include the following:

count-gcycle-xt=on

Enables emulation of the `gcycle_nT` registers. Emulating this feature is expensive enough that it is not enabled by default.

For more information, see Section 3.4.1.

usefs=*<dir>*

If the emulated Hexagon program attempts to open a `.so` file and it cannot be found, prepends the path specified by *<dir>* to the `.so` filename and then tries to open the file again.

-m 6G

System configuration option that launches a guest Hexagon system with 6 gigabytes of RAM.

3.2.3 Timing

Because QEMU is designed differently from a cycle-accurate simulator, it can result in less predictable execution. The semantics of the architecture are always preserved, but you might experience results that do not exactly match target or simulator behavior, especially when interacting with clock devices. QEMU supports a special mode that might show more consistency.

-icount [shift=N|auto]

Dispatch instructions on a specified frequency that corresponds to the virtual clock. Using `-icount` mode will make results more deterministic.

shift=N

Virtual CPU will execute one instruction every 2N ns of virtual time.

auto

Virtual CPU speed is automatically adjusted to keep virtual time within a few seconds of real time.

NOTE: The `-icount` mode typically makes execution significantly slower, so use it with caution.

3.2.4 Miscellaneous

-append 'x y z 1 2 3'

Specify the space-delimited command-line arguments for the Hexagon program that is being executed in System mode.

The Hexagon guest program can access these arguments in the same way the simulator accesses them: a `SYS_GET_CMDLINE` angel call provides the access.

-gdb tcp::1234
-s

Launch QEMU with a debug server that is listening to local TCP port 1234.

-kernel *file*

Run a specified Hexagon program (for more information, see Section 3.3.1).

--machine V68N_1024
-M V68N_1024

Launch QEMU with a configuration table and memory map that match the V68N_1024 core.

Additional configurations:

Machine	CPUs
V66G_1024	4
V66_Linux	4
V68N_1024	6
V68_H2	4
V69NA_1024	6
V73NA_1024	6
V73_Linux	6
V75NA_1024	6
V75_Linux	6

NOTE: Values specified for this argument do not add or remove architectural or coprocessor features that are available when executing instructions.

-monitor [none|stdio]

Specify how to set QEMU's interactive monitor.

none

Disable the interactive monitor.

When the monitor is enabled, it consumes `stdin`, which will conflict with your guest program's use of `stdin`.

stdio

Connect the monitor to the console. Use the interactive module to investigate and trace the guest program execution.

Several features of this monitor are also available in `hexagon-lldb`.

-S

Halt execution at the startup point. This option is useful when using a debugger.

3.3 Usage

3.3.1 Run programs

The simplest way to run a Hexagon program is to enter the following command:

```
qemu-system-hexagon -kernel ./test_prog
```

QEMU will initialize the Hexagon system, load `test_prog` into memory, and start executing code at the entry point specified by `test_prog`.

3.3.2 Device emulation

The QTimer and L2VIC devices are provided as a part of the overall Hexagon system emulation. These devices are enabled by default and can be accessed via memory-mapped I/O using the addresses provided in the system configuration table.

The generic QEMU system provides the capability for using disk images as emulated storage devices. Hexagon QEMU does not support these devices. Instead, use the `-kernel` option to launch a program.

3.3.3 Debugging

To debug a simulation, you can use `hexagon-lldb` to either attach to an existing QEMU session or launch a QEMU session. Using `hexagon-lldb`, you can set watchpoints, breakpoints, step through code, and read and write registers the same way that you do with the simulator or the target.

For example, after starting QEMU with the `-s -S` options, you can connect to the session as follows:

```
$ hexagon-lldb
(lldb) file /path/to/executable.elf
Current executable set to '/path/to/executable.elf' (hexagon).
(lldb) gdb-remote 1234
Process 1 stopped
* thread #1, stop reason = signal SIGTRAP
    frame #0: 0x00000000
->  0x0:      { jump 0x44 }
```

3.3.4 Plug-ins

QEMU plug-ins allow you to add new functionality to QEMU. You can write a plug-in library that can inspect and alter the system state during translation and translation-block-execution.

The specifics of the interface are described at https://qemu.readthedocs.io/en/latest/devel/tcg-plugins.html and in the QEMU source in `./include/qemu/qemu-plugin.h` and `./docs/devel/tcg-plugins.rst`. Example plug-ins are included in the source distribution of QEMU in the `./contrib/plugins/` directory.

The source distribution for `qemu-system-hexagon` that comes with the Hexagon Tools is in `Tools/src/qemu_src.tgz`.

You can invoke QEMU and load one or more plug-in libraries as follows:

```
qemu-system-hexagon -kernel ./some-prog \
    -plugin libmy_trace.so,arg1=on,arg2=12 \
    -plugin libmy_debug.so
```

3.4 Architecture caveats

Some architecture features cannot be replicated with QEMU. You can expect to see the following deviations from the architecture.

3.4.1 Cycle counts

The system architecture defines the following registers that are intended to count cycles that have elapsed:

- `pcycleNt`
- `gpcycleNt` – Are expensive to model, so they are off by default. They can be enabled using a CPU property.
- `gpcyclelo/gpcyclehi` – Mirror `PCYCLEHI/PCYCLELO` registers in Guest mode. The values are only valid when the CE bit in SSR is set.
- `pcyclelo/pcyclehi` – Are incremented each time a packet starts. If the packet is replayed, `PCYCLEHI/PCYCLELO` is incremented each time.
- `upcyclelo/upcyclehi` – Mirror `PCYCLEHI/PCYCLELO` registers in User mode. The values are only valid when the CE bit in SSR is set.
- And so on

These registers are modeled in QEMU as counting single packets, regardless of the instructions in the packet. Thus, the cycle counts will not exactly match the counts reported by the simulator, but they should be able to coarsely measure the overall system utilization.

QEMU does not model stalls, so it cannot give an accurate cycle count. However, you can use the number of packets as a proxy.

3.4.2 HVX 64-bit mode

The system architecture defines HVX coprocessor support to have different precisions controlled by the V2X bit in SYSCFG. QEMU does not support the 64-bit mode (value 0 in the V2X bit).

3.4.3 Interrupts

The system architecture for interrupt handling changed between Hexagon V62 and V65.

- V62 and earlier versions have a different set of registers for accessing the `ipend` and `iad` fields. QEMU does not support these registers.
- QEMU supports only the interrupt registers in V65 and later versions.