QEMU Emulation & Debug in SiFive Freedom Studio
Version 1.0

Copyright © 2022 by SiFive, Inc. All rights reserved.
QEMU Emulation & Debug in SiFive Freedom Studio

Proprietary Notice

Copyright © 2022 by SiFive, Inc. All rights reserved.

Information in this document is provided “as is,” with all faults.

SiFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose and non-infringement.

SiFive does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

SiFive reserves the right to make changes without further notice to any products herein.

Release Information

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>February 9, 2022</td>
<td>• Initial release</td>
</tr>
</tbody>
</table>
Contents

1 What Is QEMU? ................................................................. 2
2 Why Choose QEMU .......................................................... 3
3 How To Download SiFive Software Tools and Packages .......... 4
   3.1 How To Determine QEMU Version Installed in Freedom Studio ............................................... 5
4 Installing Freedom Studio Software ........................................ 6
5 How to Create and Build a QEMU Software Project in Freedom Studio ................................................................. 7
6 How to Debug Using QEMU .................................................. 12
7 Limitations Of Using QEMU In Freedom Studio ...................... 18
What Is QEMU?

QEMU stands for Quick Emulator and it is a generic and open source tool for running software emulation on supported underlying architectures, such as x86. QEMU supports many ISAs, including the RISC-V ISA.

SiFive provides pre-built QEMU binaries for several SiFive RISC-V cores and ISAs. For more details on QEMU please refer to https://www.qemu.org/.
Why Choose QEMU

QEMU is an architecturally accurate emulation tool that allows software teams to develop, port, and debug code in an emulation environment that targets a fixed configuration of SiFive’s RISC-V standard cores, and then run the same executables on RTL simulation or FPGA.

QEMU translates RISC-V instructions to the host CPU instruction set on the fly. It works as a behavioral model and uses the fast x86 host machines with large memory instead of an embedded core and is therefore much faster than an interpreted simulator.

QEMU can be used to execute and debug bare metal or Linux applications without any need for FPGA or hardware. This enables large software teams to develop code faster and more collaboratively, without requiring each developer to have target hardware platforms physically on their desks or IT teams to manage large, often complex, hardware farms, ultimately increasing developer productivity when developing for SiFive RISC-V cores.
How To Download SiFive Software Tools and Packages

QEMU is integrated and ships with the Freedom Studio bundle with fixed memory map BSP options for a fixed configuration of SiFive's RISC-V standard cores. In order to run and debug any available software examples through Freedom Studio, users need to download the latest version of Freedom Studio that is available through SCS (https://scs.sifive.com/) via download URLs.

In order to download the tool users need to log in to their SCS account, navigate to their Workspace to access the software deliverables list under the Customer Core Designs. Finally, choose and download the latest Freedom Studio suitable for your platform.

Additional information on how to access SiFive's software and tools is here: https://sifive.atlassian.net/servicedesk/customer/portal/47/article/2192998455?src=570211818

Users can download software development kits and all toolchains and utilities for RISC-V devices through https://www.sifive.com/software.

Please contact SiFive FAE or Support team for any assistance needed to download the software.
3.1 How To Determine QEMU Version Installed in Freedom Studio

1. Click on FreedomStudio and select About FreedomStudio
   a. FreedomStudio → About FreedomStudio
2. Then click on the Installation Details as shown in Figure 1.
3. Search for qemu in the search bar
4. This will point you to the QEMU Version installed
Installing Freedom Studio Software

Download the Freedom Studio tools package as shown in Chapter 3.

For Windows, unzip the Freedom Studio package and locate the "FreedomStudio" executable in the installation directory. Double-click it to launch Freedom Studio. For MacOS untar the Freedom Studio package and double click the FreedomStudio App to launch the application.
How to Create and Build a QEMU Software Project in Freedom Studio

Once Freedom Studio is set up on your system, you can select **Create a New Freedom E SDK Software Project** from the menu to create a new project as shown in Figure 1.

![Create a New Freedom E SDK Software Project](image)

**Figure 2:** Freedom Studio Create new Project
This will take you to the new project wizard where you can select the following options shown in Figure 3.

1. Select Freedom E SDK to use:

   Select the [bundled] repo which contains SiFive Board BSP (Board Support Package) to build the project. The built in QEMU targets are packaged into the [bundled] version.

   There are two bundled versions available as shown in Figure 4 and Figure 5. Select the standard freedom-e-sdk-v21.11.00.00 for the SiFive Essential Standard E,S and U targets and select the freedom-e-sdk-xtra-v21.11.01.00 for the X280 targets.
Figure 4: Select bundled Freedom E SDK For SiFive Essential Cores

Figure 5: Select bundled Freedom E SDK For SiFive Essential Cores

Figure 6: Select bundled Freedom E SDK for X280 Core
2. Select target:

When you first open the new project window, the target selection box might be empty. You need to select a qemu target from the options in the drop-down. You must select the target that most closely matches your core and ISA.

The Select Target dropdown option consists of the QEMU prebuilt binary as shown in Figure 7 below. It consists of a fixed number of ISA configurations for emulation. For example, from the figure below, users may select qemu-sifive-e31 for debugging, a 32-bit embedded core.

![Select QEMU Target](image)

**Figure 7:** Select QEMU Target

The following table describes the list of qemu targets and supported ISA options. Users can also customize the QEMU model for a desired ISA configuration through the command line. For more information please SiFive FAE or Support team.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qemu-sifive-e31</td>
<td>Supports RV32IMAC targets</td>
</tr>
<tr>
<td>qemu-sifive-s51</td>
<td>Supports RV64IMAC targets</td>
</tr>
<tr>
<td>qemu-sifive-u54</td>
<td>Supports RV64IMAFDC targets</td>
</tr>
<tr>
<td>qemu-sifive-u54mc</td>
<td>Supports RV64IMAFDC multi core targets</td>
</tr>
</tbody>
</table>

SiFive is continually improving our QEMU offering and works to add more targets and ISA options. Future releases of FreedomStudio/Qemu may include new targets and RISCV ISA support.

3. Select example program:

Select an example program. Several examples are provided and each one demonstrates different features sets of the core. Here, we are selecting the “hello” program from the available software examples. This demonstrates a printf and prints "Hello World!" to the console.
4. Select toolchain:

Select the compiler you want to use for this project. Choosing “auto-select” will automatically choose the newest compiler from all compilers in the install tree or use the dropdown selector to explicitly select the bundled version as shown in Figure 3.

5. Select project name:

The project name is automatically generated based on your target and example selections. If you prefer an alternative name you can change it.

6. QEMU as debug launch:

Finally, you can choose to automatically build and create a debug launch configuration for your new project. Choose “QEMU”, since you are using the QEMU target.

When you click the Finish button as shown in Figure 3, Freedom Studio will create your new project and build it, which can take a few minutes. After compiling and linking the software example, the Debug Launch Configuration dialog will automatically open shown in Figure 6.
How to Debug Using QEMU

Use the **Debug Launch Configuration** dialog to configure and launch the debug configuration as shown in Figure 8.

![Debug Configuration Window](image)

*Figure 8: Debug Configuration Window*

The design.dts (dts - Device Tree String) located in the **Target DTS** tab contains the design target information such as **Startup CPU Frequency**, **Device Memory Map**, etc as shown in Figure 9. The design.dts file is located in the `/bsp/` directory of the workspace.
Select the QEMU tab to view or change the debug configuration. The QEMU setup and configuration options are shown in Figure 10.
1. Start QEMU locally
   Automatically start/stop QEMU as a GDB server locally for each debug session.

2. More debug options:

   Users can provide more configuration options for debug, for example -d trace, trace_mr, unimp, guest_errors to get lots of tracing messages, or -monitor stdio to activate the monitor.

3. Debug console:

   This option allocates an Eclipse console where the output generated by QEMU will be displayed.

   When you click the Debug button, Freedom Studio will build the project and start the debugging process. The debug control stops at breakpoint which is usually set at main() function as shown in Figure 11.
Step through the code using the debug control options shown in the Run Toolbar in Figure 10. Without having to connect to any FPGA or RTL, users of QEMU can quickly build and debug any software examples and debug software. The disassembly window displays the RISC-V assembly code execution and the address location and you can view the registers being updated in the Registers section. In addition users can also probe memories, query CSRs, set breakpoints, step through code, add watchpoints and more.

Figure 11: Freedom Studio IDE QEMU Debug

Figure 12: Run Toolbar Options
The **State Browser** located in the bottom right corner below the disassembly window (shown below in Figure 13) provides a view for browsing the registers fields of all the peripherals in the system under debug and for reading and writing their state.

![State Browser](image)

**Figure 13: State Browser**

Users can explicitly launch the debug window and change the configuration as shown in Figure 14. Since this is a qemu debug, select the **SiFive QEMU launch** for debug or select the **Debug Configuration** to customize your debug configuration options.
Figure 14: SiFive QEMU Launch
Limitations Of Using QEMU In Freedom Studio

- QEMU is an architectural simulation tool and not a cycle accurate simulation tool.
- It doesn't simulate any latencies associated with memories as emulation is only functional, so there is no timing accurate model of the pipeline or memory subsystem.
- Currently available Freedom-E-SDK BSP's only support fixed number of ISA configurations
- It only emulates a few SiFive peripherals (CLINT, PLIC, UART, GPIO) in fixed configuration