

QEMU: Introduction Demo Script

Introduction

This demonstration provides high-level instructions on how to run a bare-metal application on an Arm® Cortex®-R5 processor using QEMU command line options.

Preparation:

- Required files: `hello_world.elf`, `QEMU_demo_R5`, and `zynqmp-qemu-arm.dtb`

The above files can be found in the `support` directory at the following location for this topic cluster: `$TRAINING_PATH/QEMU_Intro/support`

- Required hardware: None
- Required software: QEMU

When run under Windows, VirtualBox is required to run Linux. AMD provides an appropriate Ubuntu Linux image that includes QEMU and the AMD tool set

QEMU: Introduction

Action with Description	Point of Emphasis and Key Takeaway
<p>If not already done, use the export command to set the <code>VITIS_PATH</code>. This path should include the version subdirectory.</p> <ul style="list-style-type: none">• Press <Ctrl + Alt + T> to open a new terminal. <pre>[host] \$ export VITIS_PATH=/opt/amd/Vitis/2024.2</pre> <ul style="list-style-type: none">• Set up the Vitis environment: <pre>[host] \$ source \$VITIS_PATH/ settings64.sh</pre>	<ul style="list-style-type: none">• In order to use the QEMU toolchain and launch the Vitis environment in Linux, you need to source <code>settings64.sh</code>.
<ul style="list-style-type: none">• Change the path to the demo directory: <pre>[host] \$ cd \$TRAINING_PATH/ QEMU_Intro/support</pre>	<ul style="list-style-type: none">• The <code>demo</code> directory contains all the files required for this demo.
<ul style="list-style-type: none">• Open and review the <code>QEMU_demo_R5.sh</code> file.	<ul style="list-style-type: none">• To save time and reduce errors, the QEMU command that needs to be run is given to you in the <code>QEMU_demo_R5.sh</code> file along with the necessary arguments.

Action with Description	Point of Emphasis and Key Takeaway														
<ul style="list-style-type: none"> The QEMU command along with the arguments from the <code>QEMU_demo_R5.sh</code> file looks like: <pre>\$VITIS_PATH/data/emulation/qemu/comp/qemu/sysroots/x86_64-petalinux- linux/usr/bin/qemu-system-aarch64 -nographic -M arm-generic-fdt -dtb \$TRAINING_PATH/tools/zynqmp-qemu-arm.dtb -device loader,file=\$TRAINING_PATH/QEMU_Intro/support/hello_world.elf,cpu- num=4 -device loader,addr=0xff5e023c,data=0x80008fde,data-len=4 -device loader,addr=0xff9a0000,data=0x80000218,data-len=4</pre>															
<p>Description of the command:</p> <p><code>-device loader,file=./hello_world.elf,cpu-num=<value>:</code></p> <p>Load an ELF onto Cortex-R5-0: The target CPU is indicated by a number from the table below.</p> <table> <tr> <th>Value</th><th>CPU</th></tr> <tr> <td>0</td><td>A53-0</td></tr> <tr> <td>1</td><td>A53-1</td></tr> <tr> <td>2</td><td>A53-2</td></tr> <tr> <td>3</td><td>A53-3</td></tr> <tr> <td>4</td><td>R5-0</td></tr> <tr> <td>5</td><td>R5-1</td></tr> </table> <p>The Cortex-R5 processors can be run in split mode or lock mode. Lock mode means that both processors will run the same code. Since you only need a single processor for this demo, you will run this in split mode. Two registers need to be configured to release the CPU reset and allow R5-0 to run in split mode.</p> <pre>-device loader,addr=0xff5e023c,data=0x80008fde,data-len=4 -device loader,addr=0xff9a0000,data=0x80000218,data-len=4</pre>		Value	CPU	0	A53-0	1	A53-1	2	A53-2	3	A53-3	4	R5-0	5	R5-1
Value	CPU														
0	A53-0														
1	A53-1														
2	A53-2														
3	A53-3														
4	R5-0														
5	R5-1														
<ul style="list-style-type: none"> To run the <code>hello_world</code> application on one of the Arm Cortex-R5 processors, enter the following command: <pre>[host] \$./QEMU_demo_R5.sh</pre>															
<ul style="list-style-type: none"> The output of running the script will be similar to below: <pre>Hello World</pre>															
<ul style="list-style-type: none"> Press <Ctrl + A> followed by X to terminate the QEMU session. 															

Summary

The QEMU program can be run with several arguments. These arguments can be used to select the target processor and configure the QEMU system and the binary that is to be executed. Here we showed how to target the R5_0 processing unit and run a simple Hello World program.