

VectorCAST RSP Xilinx Zync-7000 QEMU

Version 1.0

2023-03-17

Application Note AN-ACT-1-108

Author	Tim Schneider
Restrictions	Public Document
Abstract	Information on the VectorCAST RSP Xilinx Zync-7000 QEMU

Table of Contents

1	Overview	2
2	Setup	3
2.1	Use Case: Selecting the Proper Configuration.....	3
2.2	Use Case: Configuring a Xilinx Project in Vitis	3
2.3	Use Case: Environment Setup	4
2.4	Use Case: Building Environments.....	5
2.5	Use Case: Testcase Execution	5
2.6	Use Case: Debugging	5
3	Additional Resources	9
4	Trademarks	9
5	Contacts.....	9

1 Overview

VectorCAST Runtime Support Package (RSP) is an extension of the VectorCAST product. It provides an interface layer that allows the user to use the VectorCAST testing techniques and methods on an embedded target processor. The VectorCAST RSP will always run on the host platform (the same platform as the compiler). Only the VectorCAST generated test harness will be downloaded to, and executed on, the embedded target.

A VectorCAST RSP is always customized to a particular Target CPU, Cross Compiler, and Run-Time environment (or kernel). As such, the information presented here contains sections for this specific RSP.

This application note provides details on configuring VectorCAST to run tests on QEMU's Xilinx Zync-7000 system emulation model. VectorCAST uses the Xilinx Vitis compiler and linker to build the VectorCAST test harness into an ARM ELF application, which is loaded into the QEMU emulation model.

VectorCAST provides an off the shelf RSP for the Xilinx Zync-700 QEMU. If the user needs support for a different architecture, need help configuring VectorCAST for the processor or debugger, or run into any issues configuring VectorCAST, please contact support@vector.com.

2 Setup

2.1 Use Case: Selecting the Proper Configuration

When configuring the Xilinx Zync-7000 QEMU RSP, select the correct processor as the target. The processor on the Zync-7000 is an ARM Cortex-A9 (Figure 1)

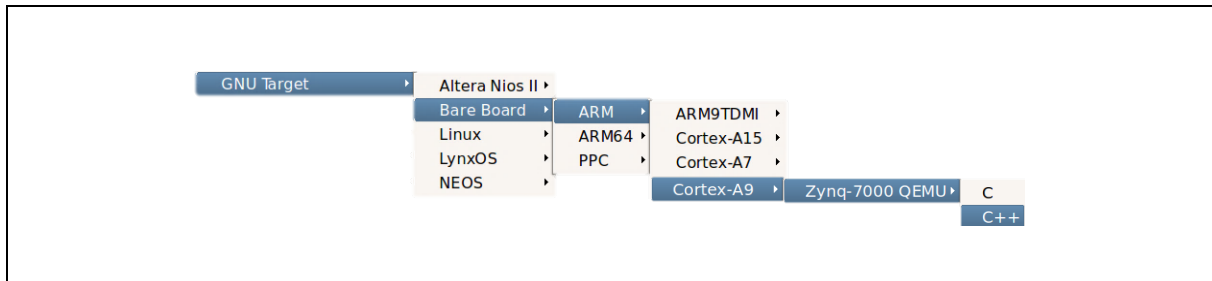


Figure 1 – Selecting the Zynq-7000 Configuration

After that, select the type of code to test. Choose C for only testing C code, choose C++ for either C++ or a combination of C and C++.

Once the type of code is selected, the configuration will show up in the project and the compiler node will be populated with the proper data, seen in Figure 2.

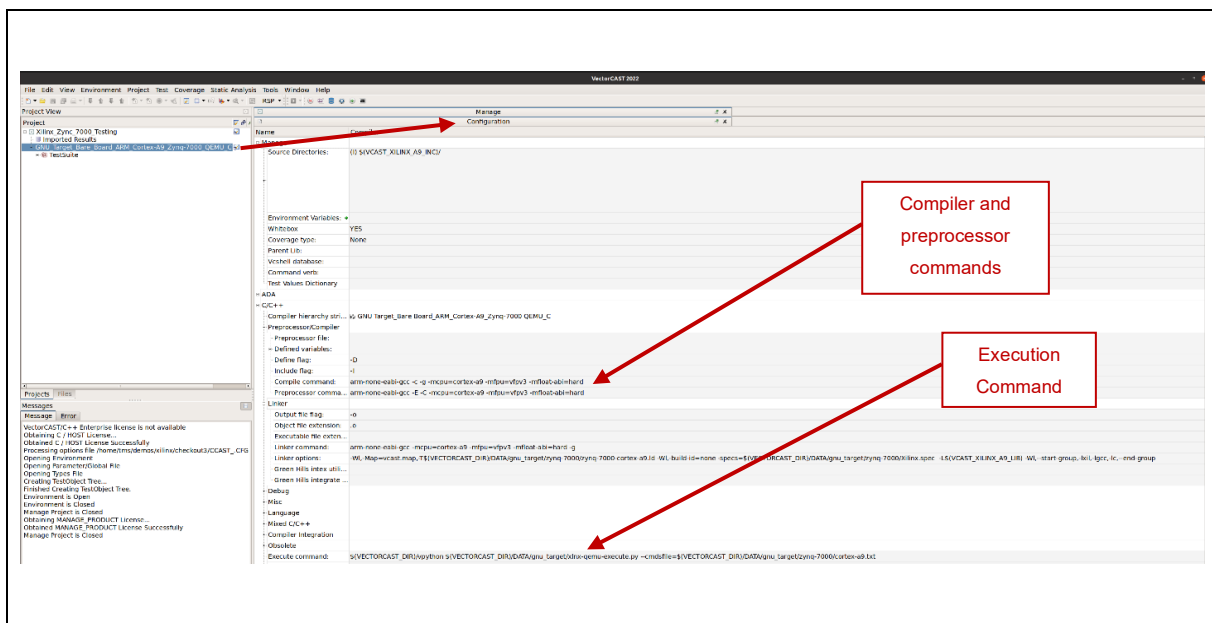


Figure 2 – Project Configuration

2.2 Use Case: Configuring a Xilinx Project in Vitis

The user must create or use a platform project from Xilinx Vitis to produce the proper libraries needed by VectorCAST to execute on the Xilinx Zync-7000 QEMU target. If the program already has an existing platform project, then use that one. Otherwise, use the Xilinx Vitis tools to create the platform.

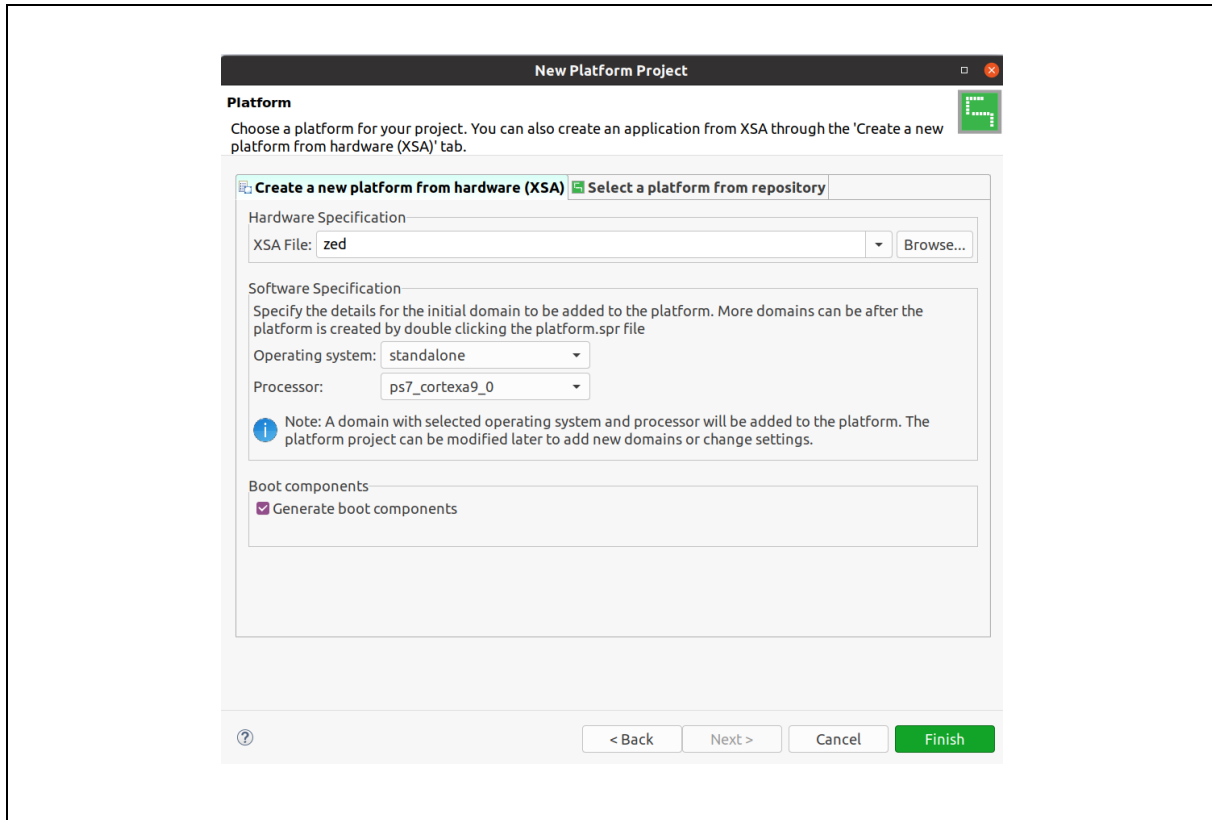


Figure 3 – Creating a Platform Project

2.3 Use Case: Environment Setup

The Xilinx Zync-7000 RSP uses environment variables to determine where the Xilinx tools have been installed and where the platform project libraries are located. Additionally, the path to the Xilinx compiler, IDE and QEMU binary directories need to be on the path environment variable. The following environment variables are used in this RSP:

- VCAST_XILINX_INSTALL_DIR – Set to the base directory of the Vitis tools installation

```
VCAST_XILINX_INSTALL_DIR=/tools/Xilinx/Vitis/2020.2
```

- VCAST_XILINX_A9_LIB_DIR – Set to the appropriate Xilinx Platform project BSP library export directory.

```
VCAST_XILINX_A9_LIB_DIR=/home/user/workspace/demo/platform_A53/export/platform_A9/s  
w/platform_A53/standalone_domain/bsplib/lib
```

- VCAST_XILINX_DTB_FILE – Set to path for device tree blob

```
VCAST_XILINX_DTB_FILE=$VCAST_XILINX_INSTALL_DIR/data/emulation/dtbs/zynq/zynq-  
arm.dtb
```

- PATH – The path will need to be updated to include the compiler, IDE, and QEMU binary directories

```
o $VCAST_XILINX_INSTALL_DIR/data/emulation/qemu/unified_qemu_v5_0/sysroots/x86_64  
-petalinux_linux/usr/bin/  
o $VCAST_XILINX_INSTALL_DIR/bin
```

```
o $VCAST_XILINX_INSTALL_DIR/gnu/aarch32/lin/gcc-arm-none-eabi/bin
```

VectorCAST recommends putting these environment variables in a shell script that will also launch VectorCAST. The **start_vectorcast.sh** may end up looking something like this:

```
export VCAST_XILINX_INSTALL_DIR=/tools/Xilinx/Vitis/2020.2
export
VCAST_XILINX_A53_LIB_DIR=/home/user/workspace/demo/platform_A53/export/platform_A53/sw/platform_A53/standalone_domain/bsplib/lib
export VCAST_XILINX_DTB_FILE=$VCAST_XILINX_INSTALL_DIR/data/emulation/dtbs/zynq/zynq-arm.dtb
export
PATH=$VCAST_XILINX_INSTALL_DIR/data/emulation/qemu/unified_qemu_v5_0/sysroots/x86_64-petalinux-linux/usr/bin:$VCAST_XILINX_INSTALL_DIR/bin:
$VCAST_XILINX_INSTALL_DIR/gnu/aarch32/lin/gcc-arm-none-eabi/bin:$PATH
$VECTORCAST_DIR/vcastqt -lc &
```

Use **start_vectorcast.sh** to launch VectorCAST to build and execute unit test environments. If the user needs to run VectorCAST from the command line, then create a similar shell script called **setup_vectorcast.sh** and remove the last line that launches the VectorCAST GUI. The **setup_vectorcast.sh** script must be run in order to execute commands from the command line.

2.4 Use Case: Building Environments

If the user followed the setup instructions in this App Note, then they should be able to build VectorCAST environments for the Zync-7000. If the user is having trouble building environments, contact support@vector.com.

2.5 Use Case: Testcase Execution

If the user followed the setup instructions in this App Note, then they should be able to execute testcases on Zync-7000 QEMU model without any further modifications. If they are having trouble executing tests, contact support@vector.com.

VectorCAST uses the Execute command in the VectorCAST options to execute tests. For the Zync-7000, VectorCAST will call QEMU directly and use QEMU stdout to collect the test results (Figure 4).

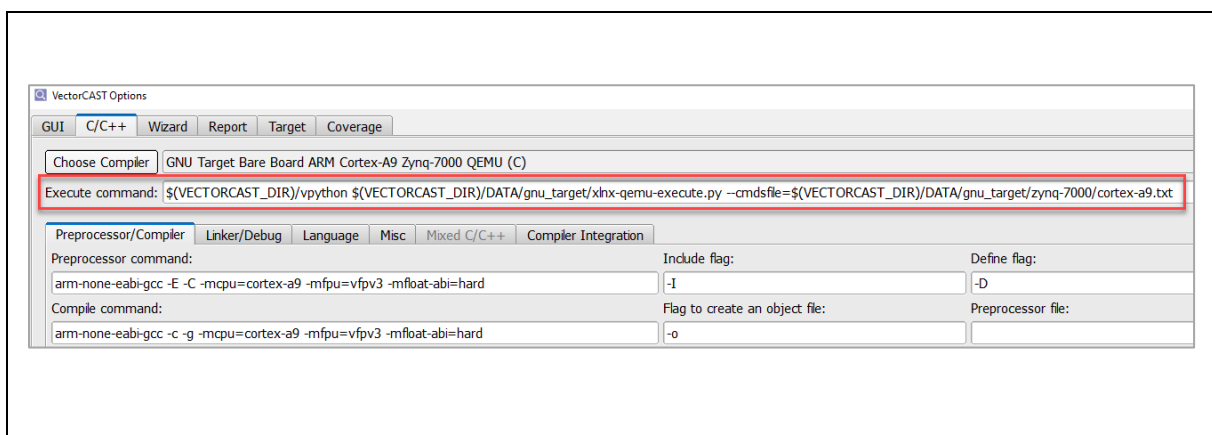


Figure 4 – Testcase Execution

2.6 Use Case: Debugging

When right clicking on a testcase and selecting '**Execute with Debug**' VectorCAST will launch the following information panel. This information panel instructs the user on how to use Vitis to debug a testcase (Figure 5).

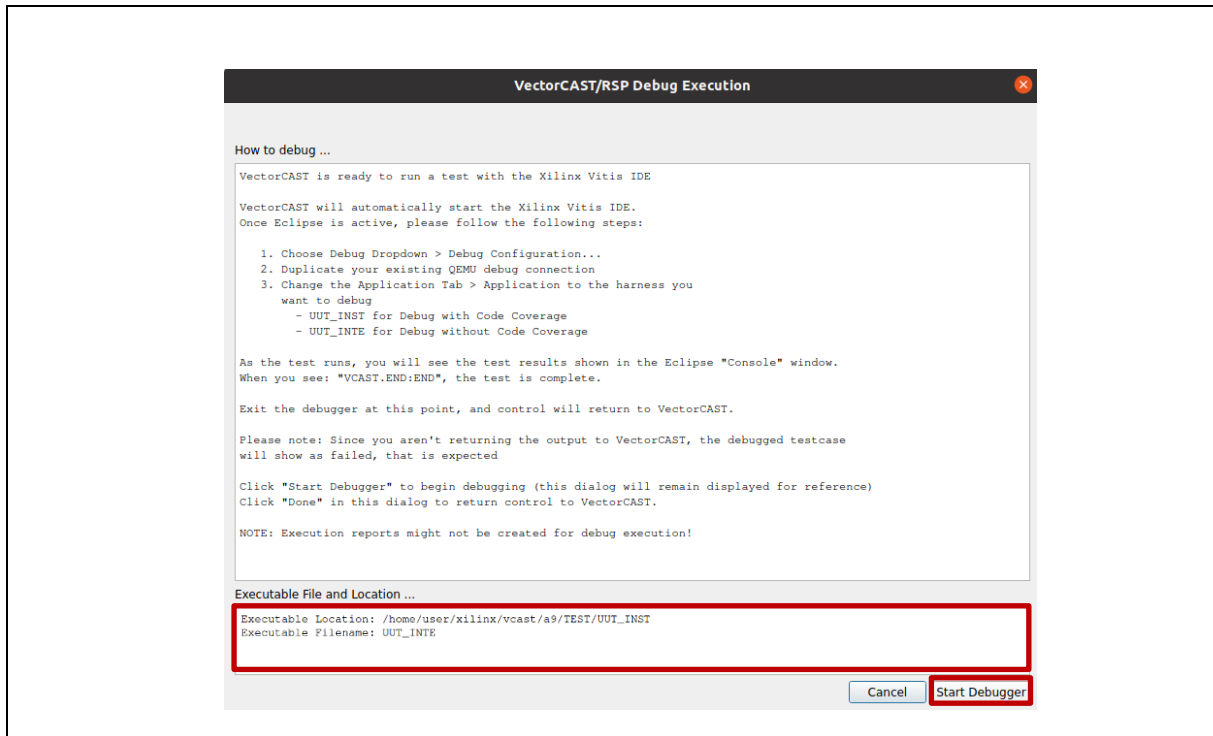


Figure 5 – Debug Information Panel

Note the Executable Location and Executable Filename. The executable location will show the user where the source code is located for the test harness. The Execute Filename would give an indication if the user selected Execute with Coverage or Execute without Coverage.

If the Execute Filename is UUT_INST then Execute with Coverage has been selected and users should use <filename>_inst.c to use for debugging. If the Execute Filename is UUT_INTE then Execute without Coverage has been selected and users should use <filename>_vcast.c to use for debugging. This difference will be important when opening files and setting breakpoints. Click on the <Start Debugger> button to open Vitis and use the instructions above to setup a debug session.

Figure 6 is an example of the modifications needed to update a debug configuration to launch the VectorCAST harness as an external application.

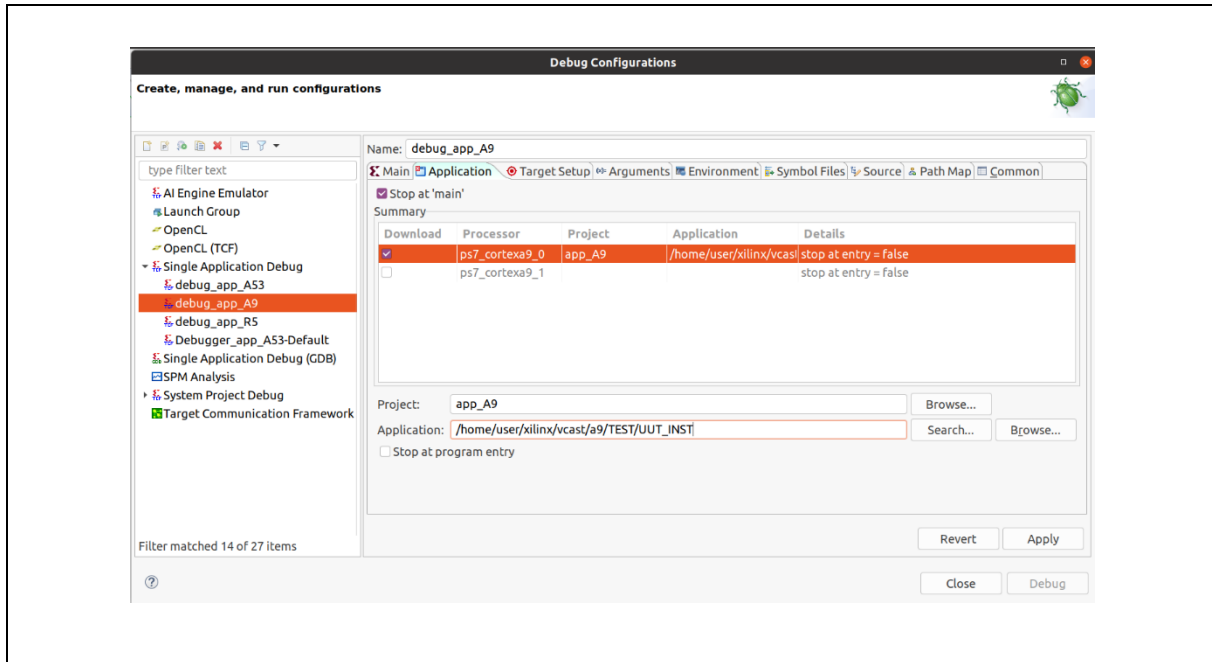


Figure 6 – Vitis Debug Configuration

Once the Debugger starts up, the user can open the file under test using the filename modifications above (with coverage or without coverage) and set a breakpoint on the function under test. From there, the user can debug as normal with Xilinx Vitis (Figure 8).

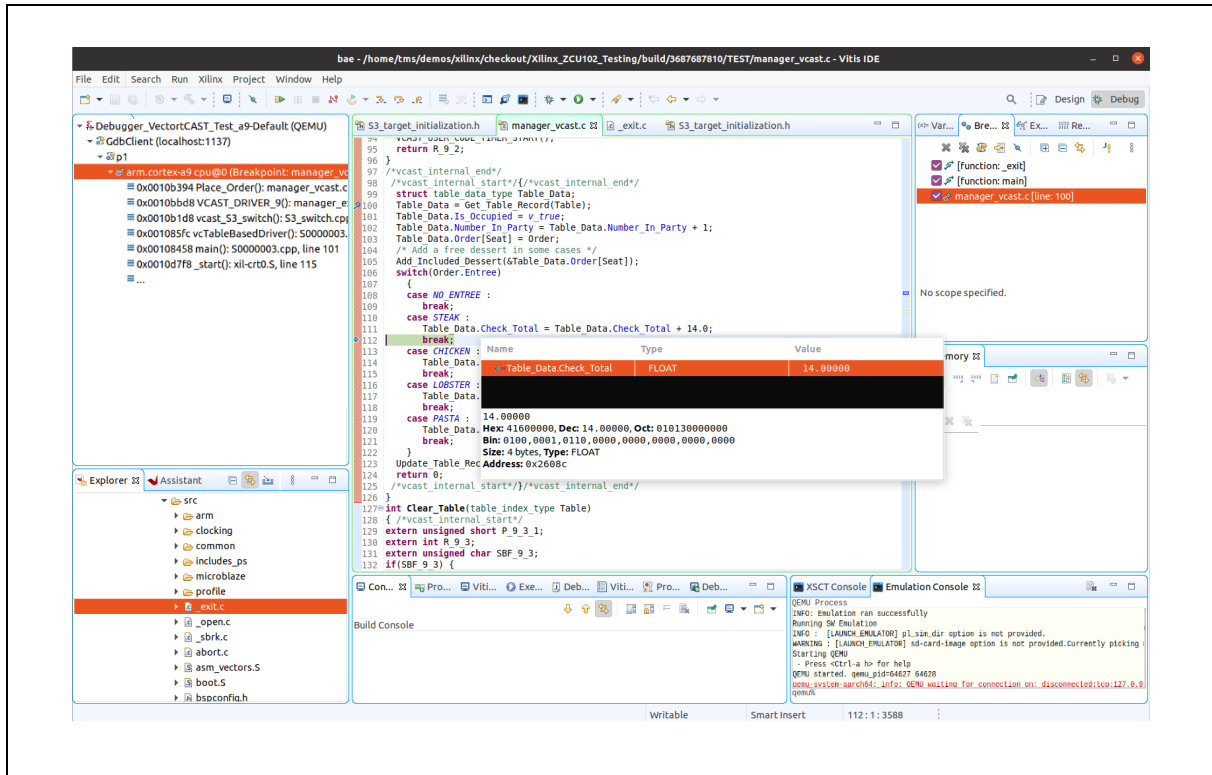


Figure 7 – Debug Window

After the user has completed debugging, close Xilinx Vitis. The user will need to indicate to VectorCAST that the debugging session has completed by clicking on the <Done> button.

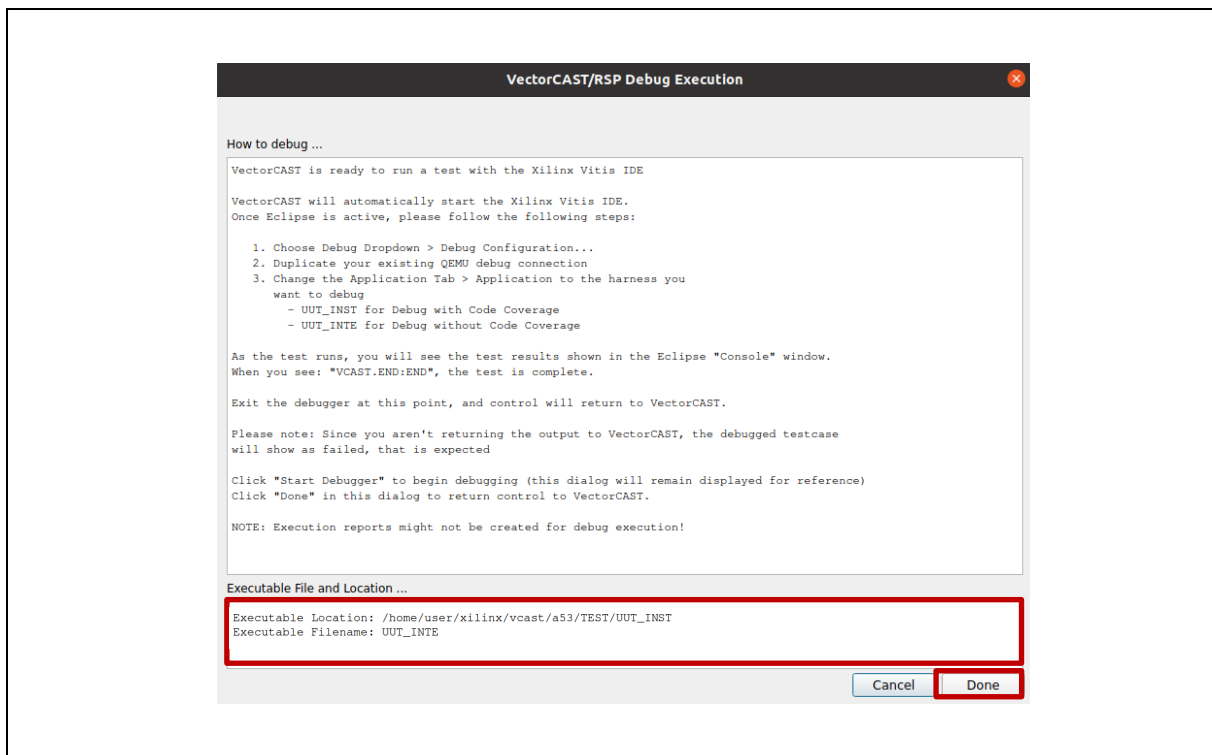


Figure 8 – Debug Information Window

3 Additional Resources

<https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html>

4 Trademarks

All mentioned names are either registered or unregistered trademarks of their respective owners.

5 Contacts

For a full list with all Vector locations and addresses worldwide, please visit <http://vector.com/contact/>.