

***Quick Start Guide for
Compiling and Debugging on the
Altera Cyclone V SoC Kit***

Chris Esser
Staff Product FAE
6/21/2013

1. Introduction

So, you just got that shiny new SoC dev kit and don't know where to start? This guide will provide you with a step-by-step introduction on how to build your own FPGA design, compile, run, and debug the SW contained in the ARM core of the FPGA. It utilizes the bare metal HW Libs example design as a case for demonstrating single-step, cross-triggering, and trace. Although this example utilizes bare metal, it can quite easily be extended to any OS as well, such as Yocto Linux.

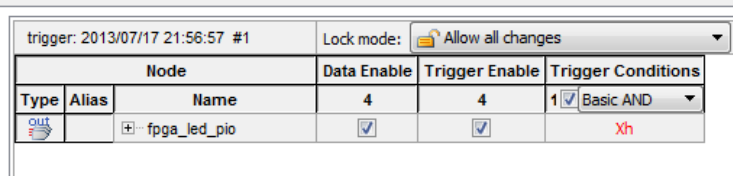
2. Design Example

The following example will utilize the Golden Hardware Reference Design as a means to demonstrate the debug capability of DS-5 and cross-triggering with SignalTap. Our first step will be to copy the hardware design over to a working directory, so we can compile and monitor various internal signals within SignalTap.

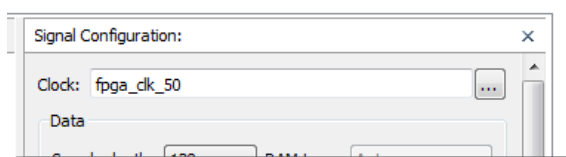
2.1. Compiling the FPGA image

The following steps should be used to create the configuration image (including SignalTap) for the FPGA:

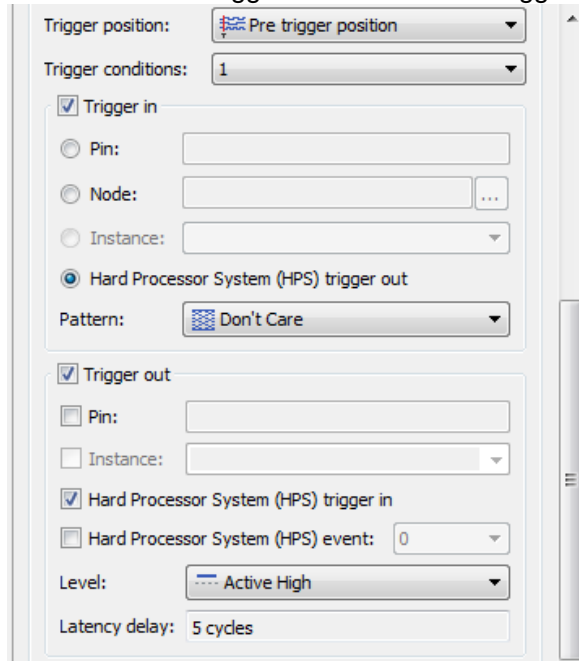
- 1) Create a directory (for instance, *SoC_Example*) that you can use for this demonstration
- 2) Copy the *cv_soc_devkit_ghrd* directory from your Quartus installation (for example, C:\altera\13.0sp1\embedded\examples\hardware\cv_soc_devkit_ghrd) to your *SoC_Example* working directory.
- 3) Open *soc_system.qpf* from the *cv_soc_devkit_ghrd* in Quartus
- 4) Open *soc_system.qsys* in Qsys and generate the system
- 5) Start **Analysis & Synthesis** on the design
- 6) After Analysis & Synthesis has completed, open SignalTap, add the “fpga_led_pio” signals for capture, and use the “fpga_clk_50” signal as the sample clock.



Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
OUT		fpga_led_pio	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1 Basic AND



- 7) Enable the HPS trigger out and the HPS trigger in

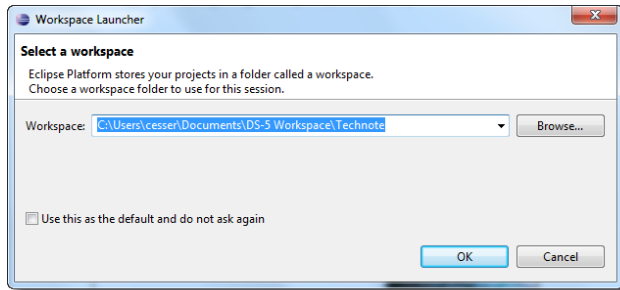


- 8) Save this SignalTap configuration, enable it for your project, and compile.

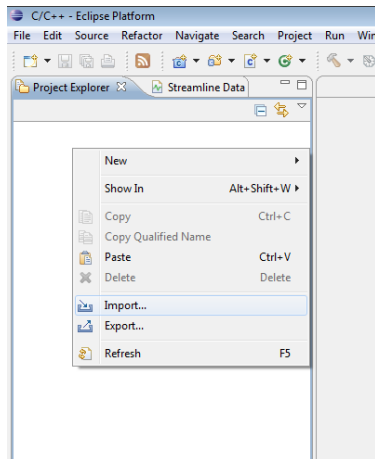
2.2. *Compiling the Software Code*

Here are the steps to build and execute the HWLIB example program:

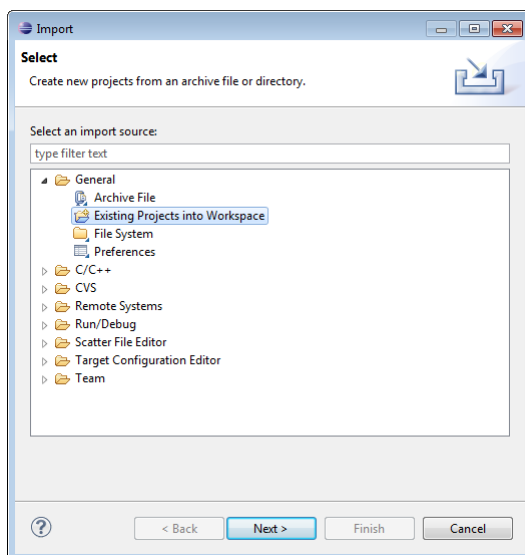
- 1) Start eclipse. Select an existing workspace or create a new workspace.



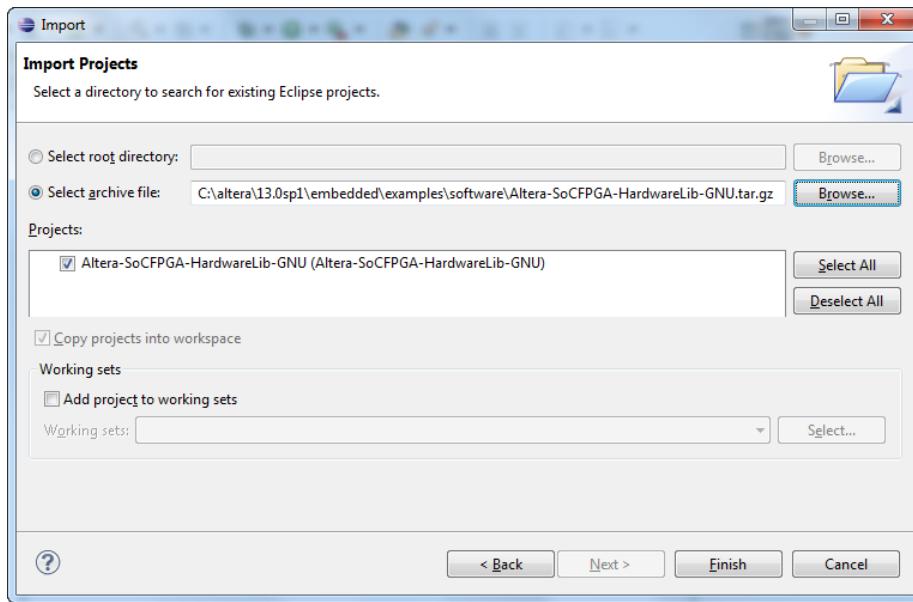
- 2) From the Program Explorer tab of Eclipse, use a right mouse button click to select **Import...**



- 3) From the Import dialog box select **General, Existing Projects into Workspace** and click the **Next** button.

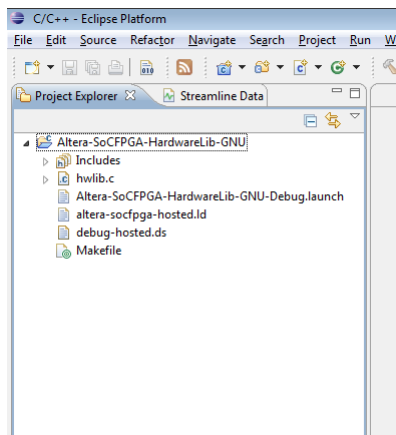


- 4) Select the **Select archive file:** radio button and use the **Browse...** button to navigate and select the **Altera-SoCFPGA-HardwareLib-GNU.tar.gz** file (For example, on my machine, it is located in C:\altera\13.0sp1\embedded\examples\software).



Click the Finish button.

- 5) The **Altera-SoCFPGA-HardwareLib-GNU** project should appear in the Eclipse Project Explorer frame.



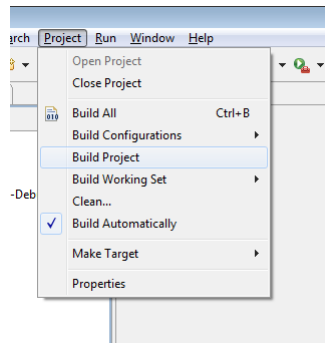
- 6) Double-click the “Makefile” in the Project Explorer to edit the file. Change line 39 from:

```
soc_system.sof:  
$(SOCEDS_ROOT)/examples/hardware/cv_soc_devkit_ghrd/output_files/soc_system.sof
```

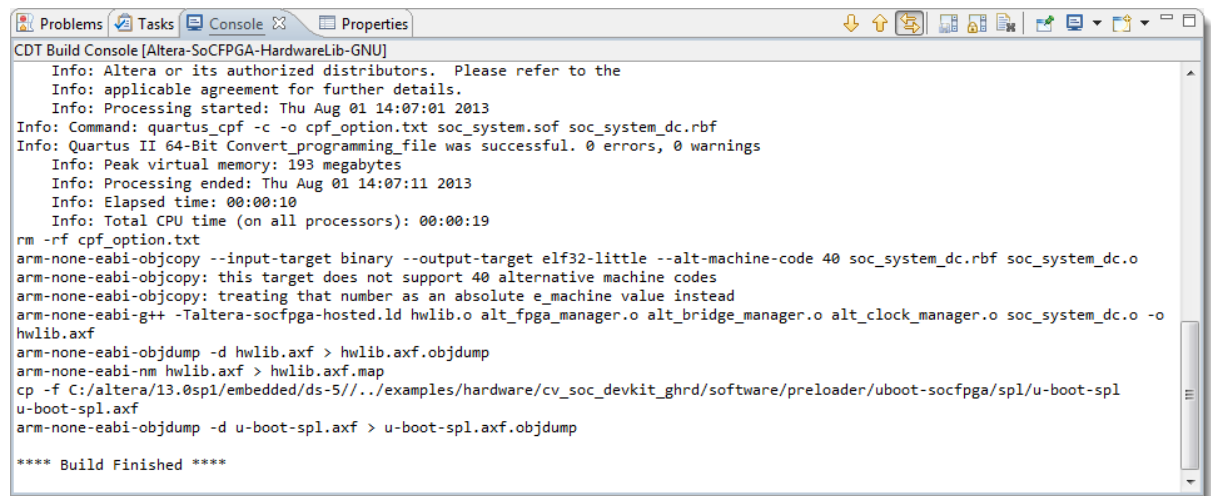
to point to the location of the soc_system.sof file that you had created in the section above (2.1, Compiling the FPGA image). For example, I changed mine to:

```
soc_system.sof:  
C:/work/Altera/TechNotes/SoC_QuickStart/ExampleDesign/cv_soc_devkit_ghrd/output_files/soc_system.sof
```

7) Select **Project**→**Build Project** from the Eclipse menu.



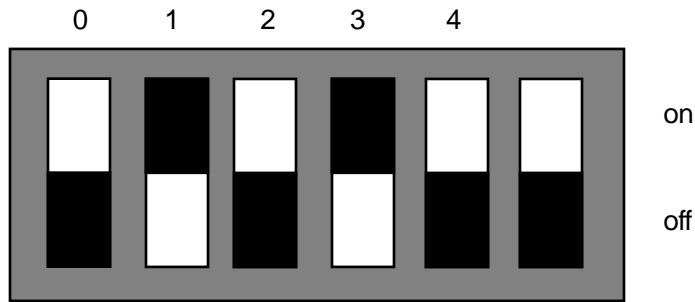
8) The Eclipse console should reflect a successful project build.



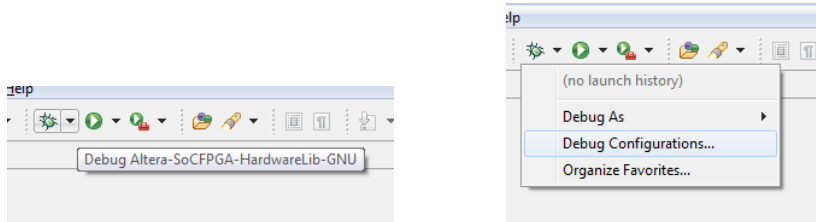
2.3. *Setting up the DS-5 Debug Configuration*

Here are the steps to execute the HW Lib example program under the control of the debugger.

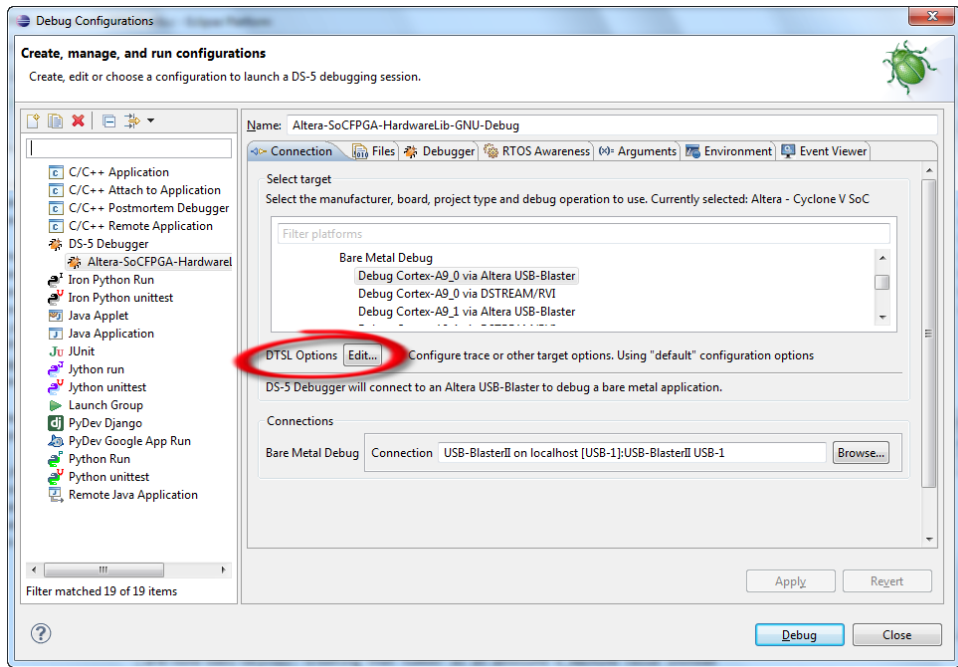
- 1) Verify that the Development Board is configured properly to execute the HW Lib example program. The MSEL DIP switches on Development Board should be configured as follows:



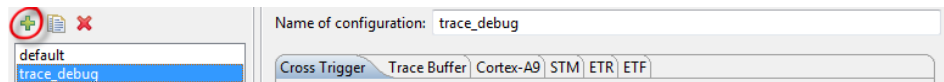
- 2) Connect the Development Board to the host computer using the USB Blaster II cable.
- 3) Power on the Development Board.
- 4) Select **Debug Configurations...** from the Debug button selection on the Eclipse tool bar.



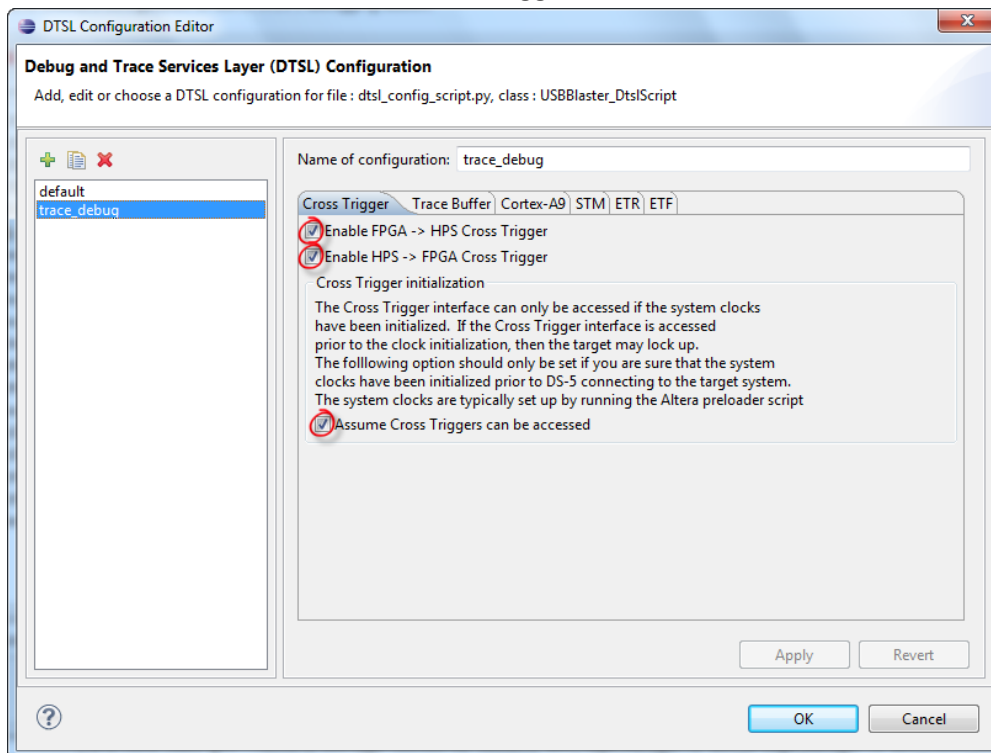
- 5) From the Debug Configurations dialog box select the **DS-5 Debugger, Altera SoC FPGA-HardwareLib-GNU-Debug** configuration. Click the DTSL Options “Edit” button.



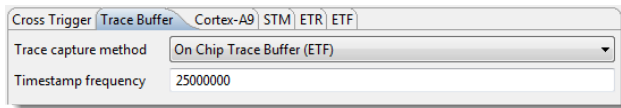
- 6) Click the green “+” icon to add a new configuration. Call it “trace_debug”.



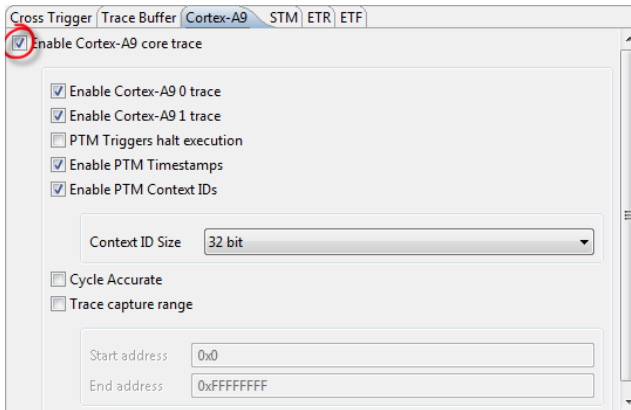
- 7) Select all of the checkboxes in the Cross Trigger tab:



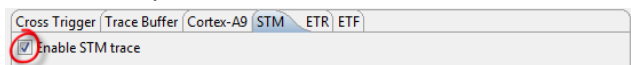
- 8) Select the On Chip Trace Buffer (ETF) in the Trace Buffer tab:



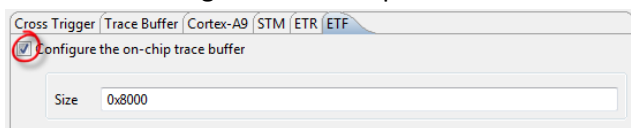
- 9) Enable the Cortex-A9 core trace



- 10) Enable the system trace module



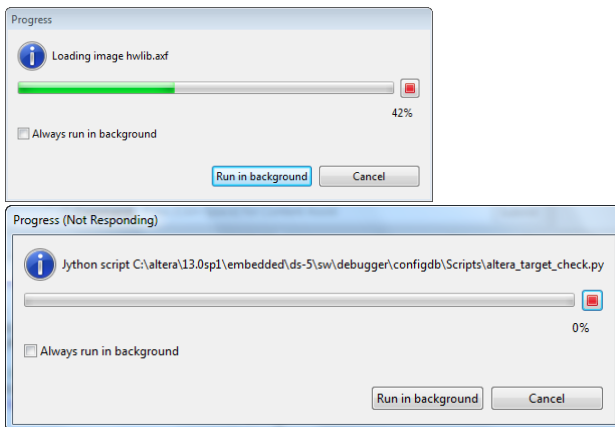
- 11) Enable and configure the on-chip trace buffer for its maximum size of 0x8000.



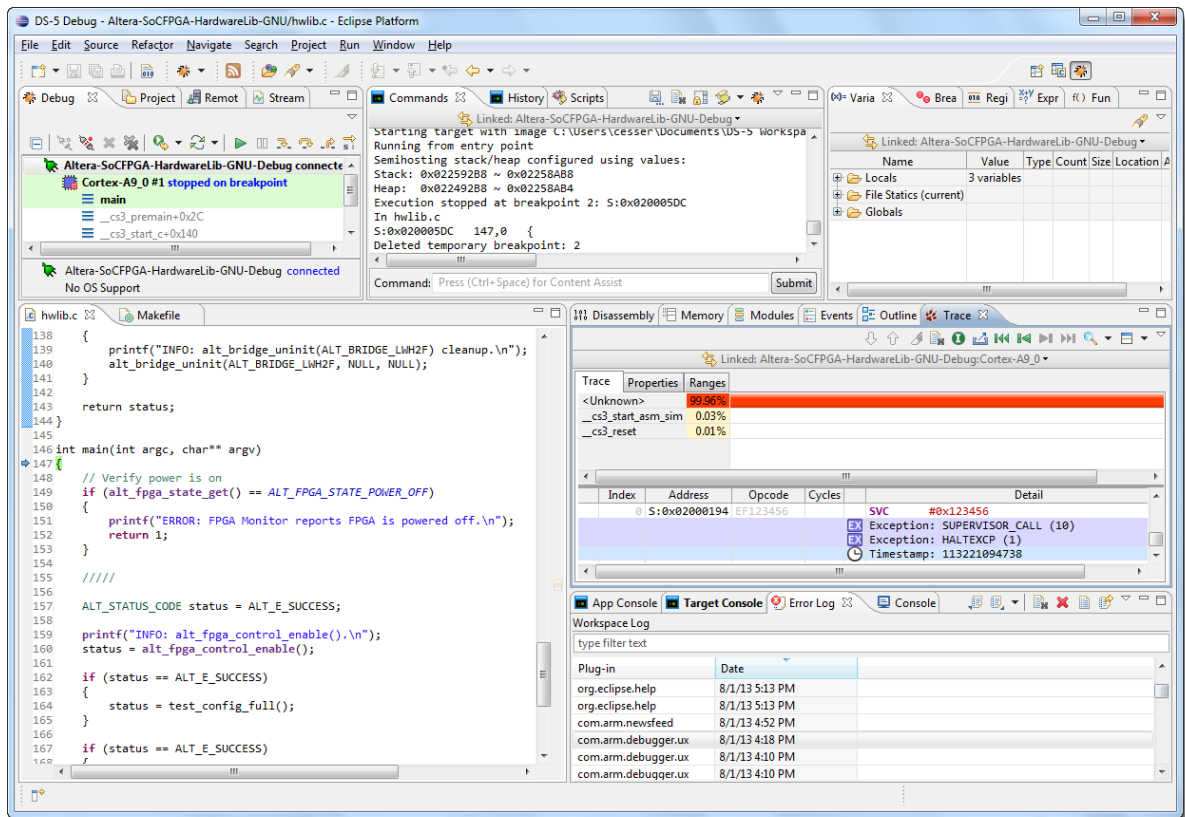
- 12) Select "OK" to close out the DTSL configuration.

- 13) Select "Debug" to apply these configuration options, and switch to the debug perspective.

- 14) The following dialog boxes should appear indicating that the HWLIB example program is loading onto the Development Board. This process may take several minutes.




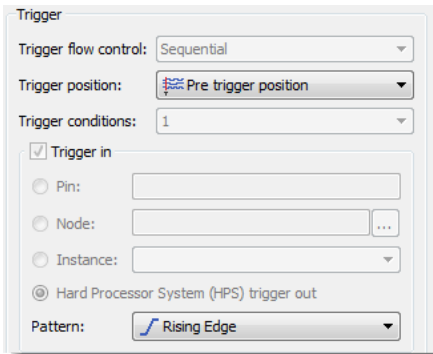
- 15) After the example program is loaded, the debugger should be stopped at main(). You may step through the example program and selectively examine the execution state using the DS-5 debug commands.





2.4. Using Breakpoints to Trigger SignalTap

Since the FPGA configuration is loaded by the application code, we need to set a breakpoint AFTER it has been configured, before we can enable the trigger in SignalTap. Since the FPGA is configured in the `test_config_full()` function, we can place a breakpoint anywhere in the `test_bridge()` function (which is the function that is executed immediately afterwards), enable SignalTap, and proceed. Therefore,


- 1) Place a breakpoint anywhere within `test_bridge()` and click the “Continue”  icon. The code should execute to that line, and it will be highlighted in green. You should also see the trace window fill up with the previously executed instructions.
- 2) Start up SignalTap. Set the **Trigger In** pattern to “Rising Edge”

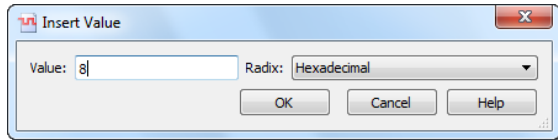




- 3) Click “Autorun Analysis”  to enable a continuous trigger.
- 4) Place a trigger on one of the statements inside the for loop that toggles the LEDs in the `hwlib.c` file.
- 5) Click the “Continue”  icon, and observe SignalTap triggering each time as you click. Also observe the trace buffer as it fills.

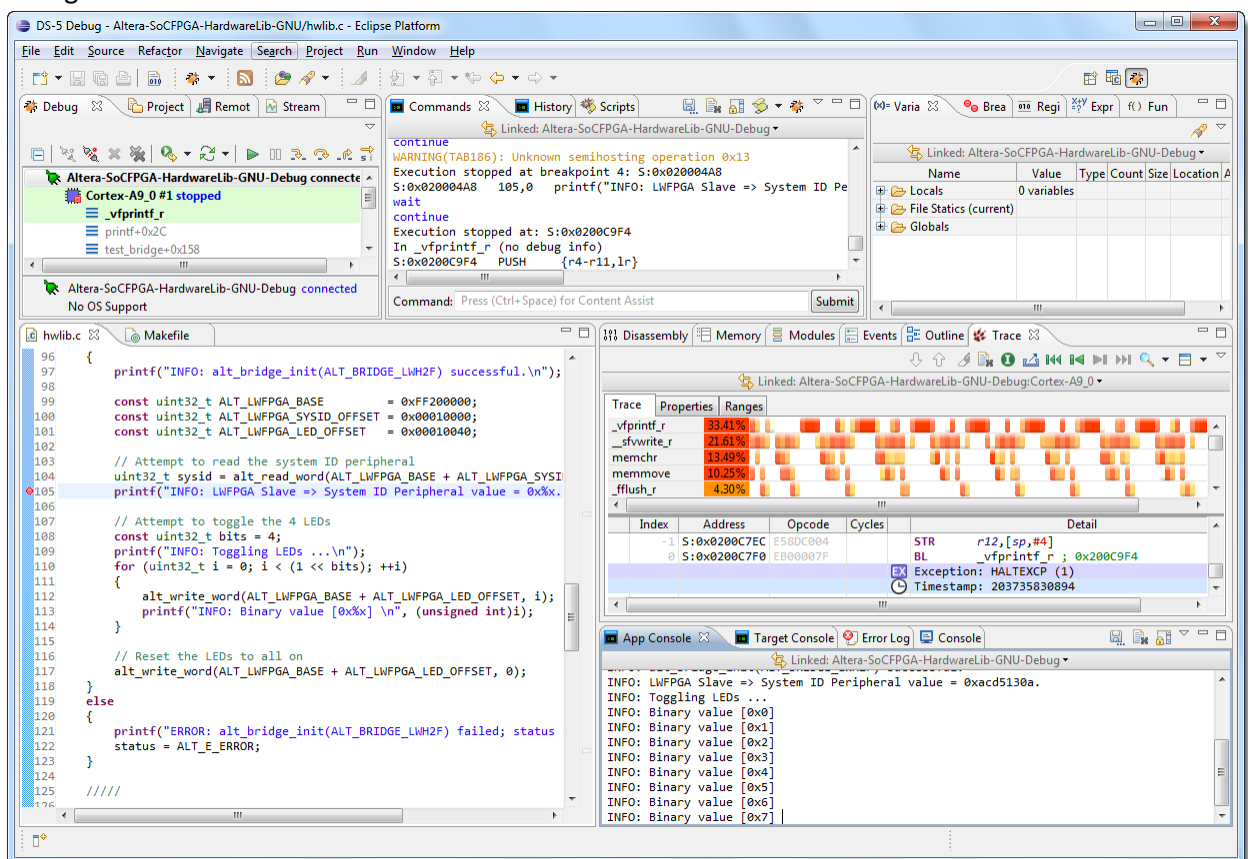
2.5. Using SignalTap to Stop Execution in DS-5

As we did before, we need to set a breakpoint AFTER the FPGA has been configured, before we can enable the trigger in SignalTap. Therefore,

- 1) Place a breakpoint anywhere within test_bridge() and click the “Continue”  icon. The code should execute to that line, and it will be highlighted in green. You should also see the trace window fill up with the previously executed instructions.
- 2) Start up SignalTap. Set the **Trigger In** pattern back to “Don’t Care”
- 3) Right click on the fpga_led_pio signal, and insert a trigger value between 0x1 and 0xF (I chose 8):



- 4) Click “Run Analysis”  to enable the trigger.
- 5) Place a trigger on one of the statements inside the for loop that toggles the LEDs in the hwlib.c file.
- 6) Click the “Continue”  icon, and observe that DS-5 stopped execution right after the LED was changed to 0x08.



3. Reference Material

The following references are immensely helpful in coming up to speed with the SoC development kit and DS-5:

<i>Cyclone V SoC Development Kit User Guide (PDF)</i>	Information about setting up the Cyclone V SoC Development Board and using the included software
<i>Cyclone V SoC Development Board Reference Manual (PDF)</i>	Detailed information about board components and interfaces
<i>Altera SoC Embedded Design Suite User Guide</i>	User's Guide for the Embedded Design Suite
<i>HPS Register Map</i>	Register map for all registers and peripherals within the HPS
<i>RocketBoards.org</i>	Linux community portal