Remote Hardware Debugging over TCP/IP for Altera SoC

2015-05-11

AN693 Subscribe Send Feedback

You can perform remote debugging of your system with System Console. Debug equipment deployed in the field through an existing TCP/IP connection. Run a network stack on either a Nios II processor or a hard processor system (HPS) and piggyback on an existing remote administration setup. This application note focuses on the case of using an Altera SoC.

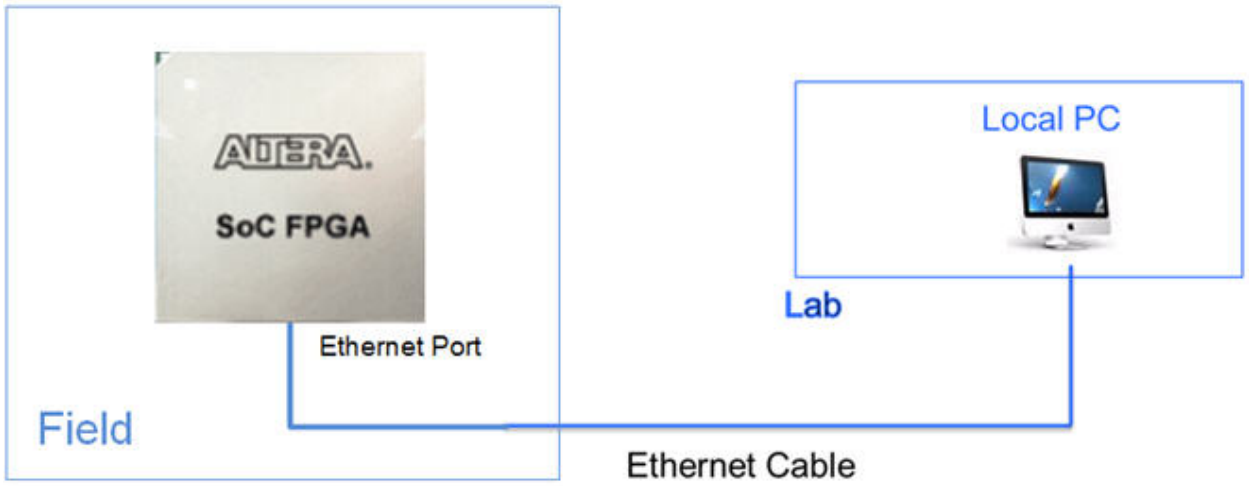
Related Information

Analyzing and Debugging Designs with System Console documentation

TCP/IP Communication Channel

Historically, the Altera System-Level Debugging (SLD) communication solution was based on the Altera JTAG Interface (AJI) which interfaced with the JTAG TAP controller (hard atom in Altera devices which implements the JTAG protocol). The SLD tools (SignalTap II Logic Analyzer, In-System Sources and Probes (ISSP), In-System Memory Content Editor) and the Nios II on-chip instrumentation (OCI) use the JTAG channel for communication between software and hardware. To support communication via TCP/IP, the SLD Hub Controller replaces the JTAG TAP controller.

Figure 1: TCP/IP Communication Channel Block Diagram



Use-Case Scenarios

Why would you use this feature?

- To debug a device when you cannot access the JTAG connections (due to mechanical restrictions or regulatory restrictions).
- To debug a device remotely, such as at a customer site.
- In a team-based situation with several people working on a single device.

Software Requirements

- Quartus II software version 13.0 or later installed on your local PC
- Familiarity with networking setup requirements
- SLD Hub Controller Linux driver (for SoC device)
- Linux running on the HPS

Hardware Requirements

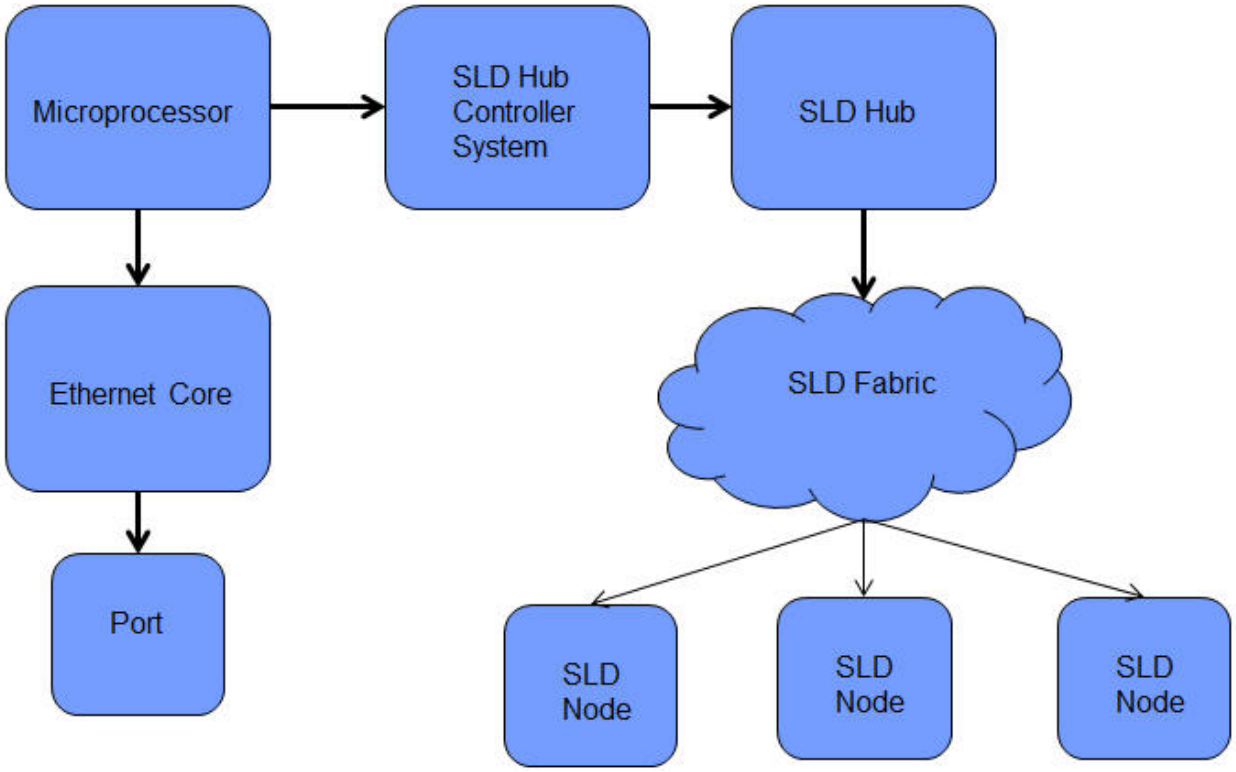
- SLD Hub Controller
- Microprocessor connected to the Altera FPGA (either internally or externally)
- Altera SoC with the HPS connected to an Ethernet port
- TCP/IP stack
- Programming device connected to an Ethernet cable at the remote location

Related Information

- **Ethernet and the NicheStack TCP/IP Stack - Nios II Edition documentation**
- **Using the NicheStack TCP/IP Stack - Nios II Edition Tutorial**

System Components

Figure 2: System Components Block Diagram



The Processor

Altera SoCs integrate an ARM-based hard processor system consisting of a processor, peripherals, and memory interfaces with the FPGA fabric using a high-bandwidth interconnect backbone. This application note assumes the HPS in the FPGA is running the Linux kernel. This simplifies the remote debugging feature. The processor monitors a TCP/IP socket for incoming transactions. The data bytes in these transactions are extracted and written directly to the SLD Hub Controller system hardware without modification. For outgoing data, the SLD Hub Controller system produces data that the processor packages into TCP/IP packets and transmits over the socket without modification.

SLD Hub Controller

The SLD Hub Controller component converts Avalon-ST packets to JTAG operations. This component connects most Altera SLD applications to hardware through a non-JTAG channel when used together with an appropriate driver in System Console.

Note: You cannot remote debug hardware using a JTAG connection through the JTAG TAP controller after instantiating the SLD Hub Controller component.

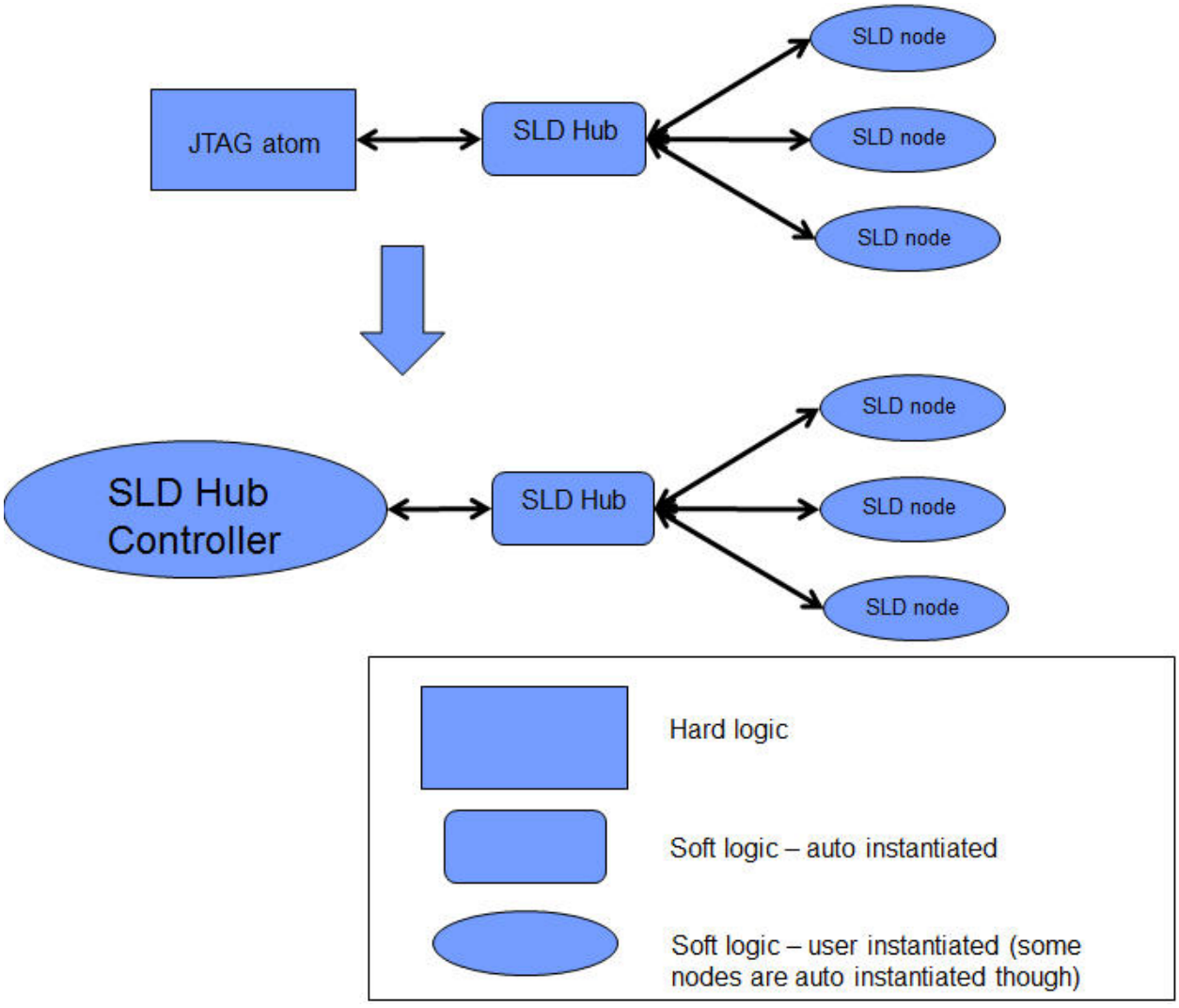
SLD Hub

The SLD Hub enables multiple nodes to share access to the user debug interface. The remote debugging feature supports the following system debugging tools over TCP/IP.

Supported System Debugging Tools

- SignalTap II Logic Analyzer
- In-System Sources and Probes
- In-System Memory Content Editor
- Logic Analyzer Interface
- SignalProbe
- System Console

Figure 3: Connecting SLD Hub to Adapter



How to Remote Debug with the SignalTap II Logic Analyzer

To implement remote debugging for the SignalTap II Logic Analyzer over TCP/IP, follow these steps:

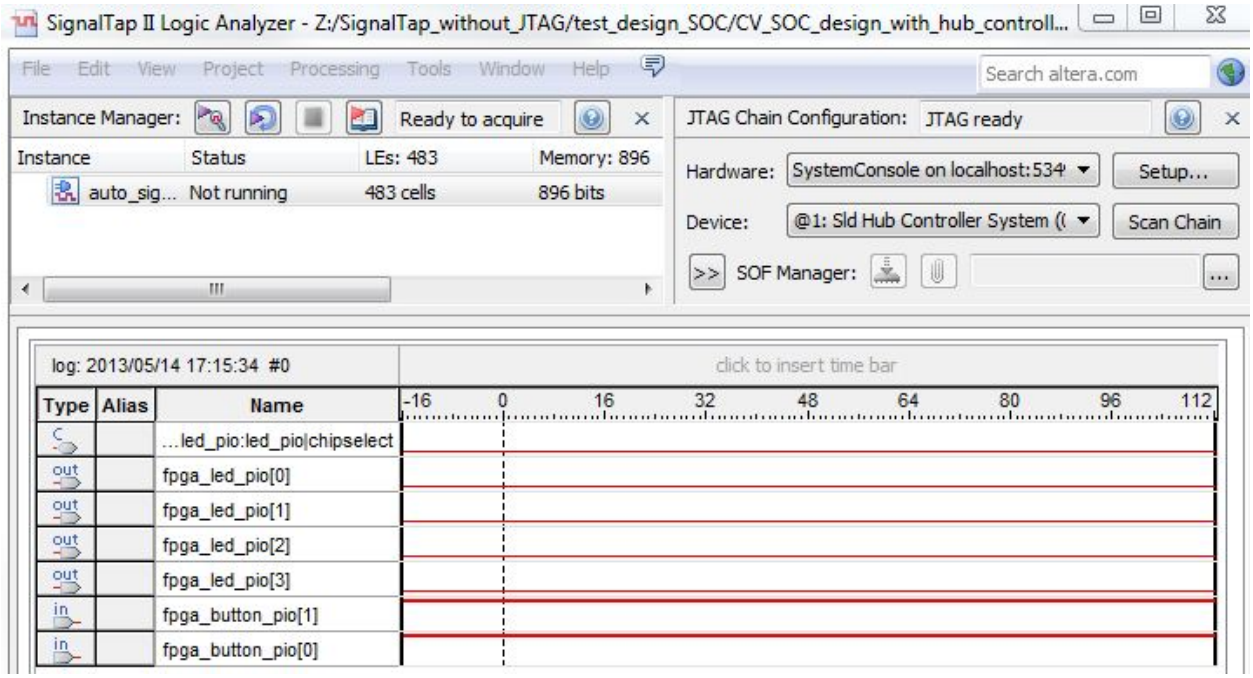
1. Verify your design can run a TCP server with memory-mapped access to the device.
   For Altera SoC, the Linux kernel handles the TCP/IP connection.
2. Instantiate the **SLD Hub Controller System** component from the IP Catalog.

Figure 4: SLD Hub Controller System

3. Generate the Qsys system.
4. Tap some nodes using SignalTap II Logic Analyzer.
5. Compile the design.
6. Using the existing remote configuration setup, update the remote board with firmware which contains the SLD Hub Controller System instantiated in the FPGA over the TCP server (enabled to listen for incoming debug connections).
7. Start System Console in JTAG server mode using the Tcl script in the Reference Design or using a custom script.

   `System-console-jtag_server-rc_script=mmlink_setup.tcl<path to .sof><IP><port number>`
8. After connecting, start the SignalTap II Logic Analyzer. You should see a System Console cable as an option under **Hardware**.

Figure 5: Successfully Connected in SignalTap II Logic Analyzer



Related Information

- **Design Debugging Using the SignalTap II Logic Analyzer documentation**
- **Creating a System with Qsys documentation**

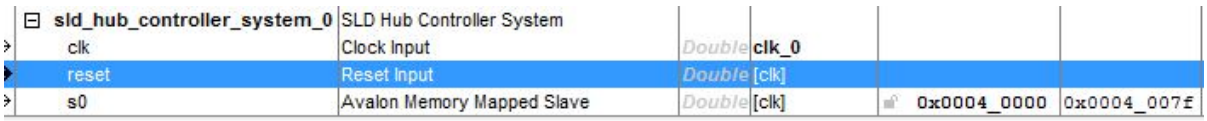
Example Implementation of Remote Debugging on an Altera SoC (5CSXSoC)

Requirements:

- microSD card for the Linux image
- USB cables
- Cyclone V SoC golden hardware reference design (GHRD)
- Linux drivers
- Setup Tcl script

1. Start Linux on the Altera SoC.
2. Get the USB serial UART cable and driver.
   a. Connect by serial to the board using PuTTY and selecting `serial` and the correct COM number with speed 57600.
   b. Press Enter and login as root.
   If the above fails, double-check that the serial driver is in the **Device Manager**.
   If you are sure the failure is not due to a configuration issue, then the Linux kernel might not be on the SD card.
3. After you get a connection via serial, get the board's IP with `>>ifconfig`
4. Add the SLD Hub Controller from the GHRD.(1)

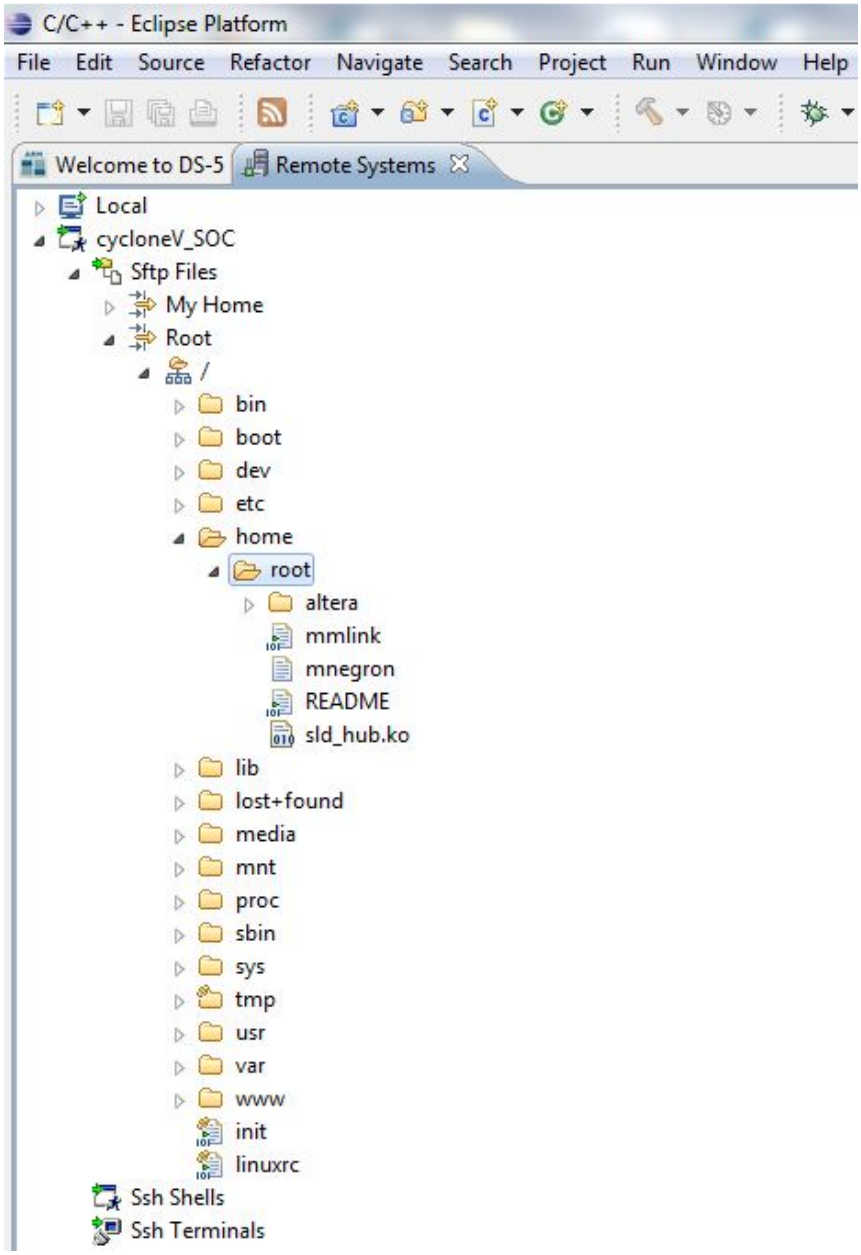
Figure 6: SLD Hub Controller in GHRD



5. Generate in Qsys.
6. Compile the design.
7. Program the board with the GHRD **.sof**.
8. Transfer the **sld_hub.ko** driver to the board via SFTP.
9. Transfer the mmlink user application to the board via SFTP.

(1) The SLD Hub Controller *Avalon Memory Mapped Slave* connects to h2f AXI Master (HPS).

Figure 7: mmlink Application



10. Run the SLD Hub Controller Linux driver and mmlink application.

```
>>insmod sld_hub.ko
>>./mmlink
```

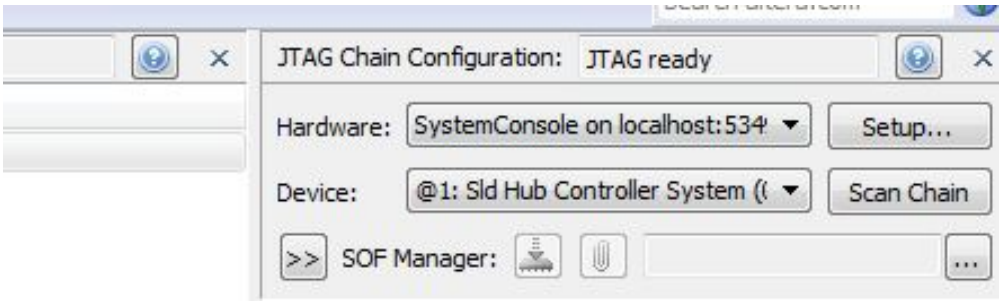
11. Use System Console to connect to the board.

```
>>system-console-jtag_server--rc_script=mmlink_setup.tcl<design>.sof<ip>3333
```

Figure 8: Connect with System Console

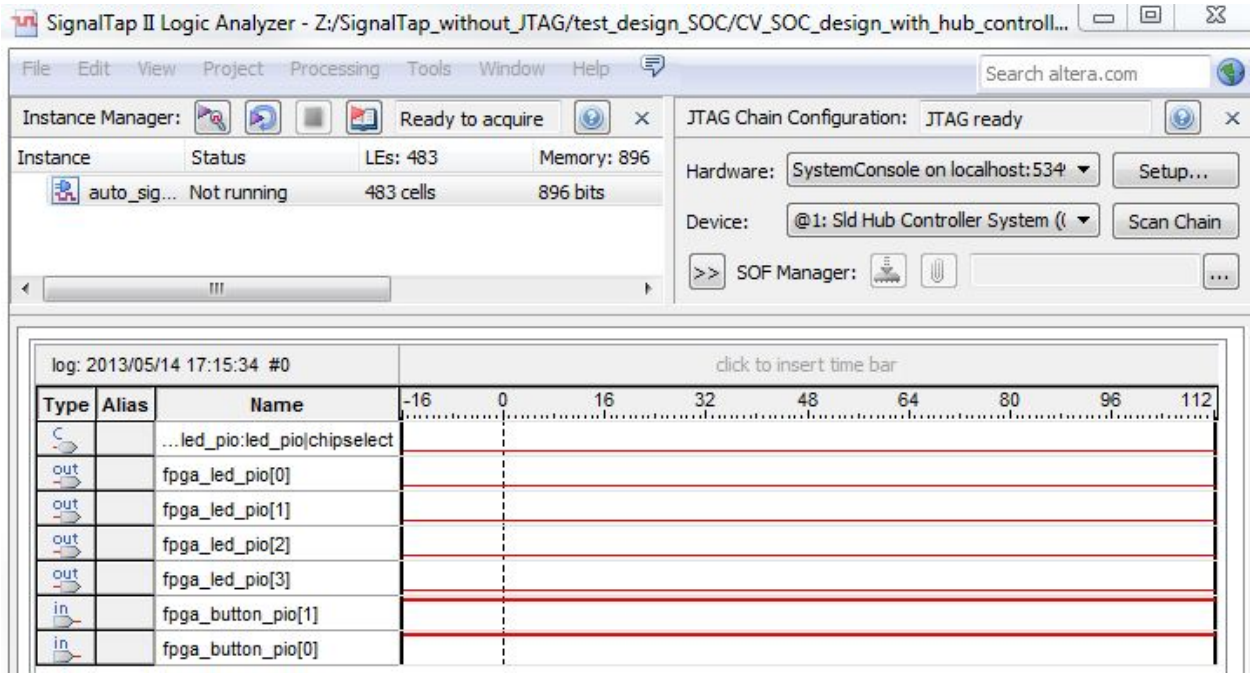
12. Open the SignalTap II Logic Analyzer and select SystemConsole on localhost:xxxx under **Hardware** and Sld Hub Controller System under **Device**.

Figure 9: Selecting Hardware and Device



13. Run analysis and view the waveforms.

Figure 10: Altera SoC Successfully Connected



Note: The SLD Hub Controller driver (**sld_hub.ko**) is compatible with Linux kernel 3.8.0-00069-g54902df-dirty.

Related Information

- **Rocketboards website**
  Altera SoC Golden System Reference Design information
- **Design Example Files**
- **Linux Drivers and Setup Tcl Script**
- **PuTTY website**
- **Cyclone V SoC Development Kit and SoC Embedded Design Suite website**
- **Design Debugging Using the SignalTap II Logic Analyzer documentation**

Appendix

Getting the Source Code for the SLD Hub Controller Driver

The SLD Hub Controller driver provided with Application Note 693 is only compatible with Linux kernel 3.8.0-00069-g54902df-dirty. You can get the source code for the **sld_hub.ko** driver from the RocketBoards.org website. RocketBoards is an open source community that provides resources for embedded solutions that allow you to explore and prototype applications for Altera SoC.

You can find detailed instructions to use the git repository on the RocketBoards website. The following is a high-level flow for using the git repository:

1. Clone the Linux kernel git tree.
2. Checkout the appropriate branch.
3. Configure the kernel to enable the driver. Use the following commands.

```
make menuconfig
```

   Note: Navigate to the **Device Drivers/Character Devices/Altera MM Debug Link Driver** and click **M** to compile the driver as a loadable module.
4. Compile the Linux kernel.

Driver Permissions

The user-mode app mmlink bridges between TCP/IP and the mm-debug-link device driver. The default device driver has permissions 0600, which can prevent you from opening the driver. If you encounter an error when you run the mmlink with these permissions in the driver, you may see an error such as below:

```
socfpga_cyclone5:~$ mmlink
failed to init driver: 13 (Permission denied)
```

The permissions of the driver can be modified, and the error avoided, by adding a udev rules file as follows:

```
root@testsocfpga# cat/etc/udev/rules.d/mm_debug_link.rules
KERNEL=="mm_debug_link", MODE="0666"
```

With this rules file in place, the next time the mm-debug-link driver is loaded, it will have permissions 0666, and any user can open it for read/write.

Alternatively, Altera has released an revised version of the device driver on RocketBoards.com that has the modified driver permissions already in place.

Related Information

- **Git website**
  Information about installing and using git
- **Using RocketBoards Git Trees**
  Information about using RocketBoards Git repository

Document Revision History

Table 1: Document Revision History

Date	Version	Changes
May 2015	2015.05.11	Added appendix. Added information about modifying driver permissions.
June 2014	2014.06.18	Added instructions to get source code for SLD Hub Controller driver.
December 2013	2013.12.05	Changed document title. Added setup Tcl script.
September 2013	2013.09.18	Initial release.