INTRODUCTION TO FPGA SIMULATION AND DEBUG

- Larry Landis – University Outreach Senior Manager, Intel Programmable Solutions Group
- 36 years in electronics industry – design, sales, marketing, training (Altera 2004)
- Adjunct Lecturer for Digital Electronics Course – Santa Clara University

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE

BSEE

LMU | LA
Loyola Marymount University

MSEE

SANTA CLARA BRONCOS

Adjunct Lecturer

Intel® FPGA Academic Ecosystem

- Train the next generation of FPGA Designers
- Increase Intel® FPGA presence in academia

- Academic access to the latest generation of Intel FPGAs

- Nurture the talent pipeline for Intel and our customers
- Engage research on Intel FPGAs

INTEL® FPGA EE COURSEWORK OFFERINGS

- **Undergraduate**
  - **Digital Logic**
  - **Digital Systems**
  - **Computer Organization**
  - **Embedded Systems**

Objective

- Understand and select appropriate debugging tools for FPGA designs.
- Hands on use of four different FPGA debug tools
  - Simulation
    - Modelsim
  - Actual Hardware
    - In-System Sources and Probes
    - Signal Tap Logic Analyzer
    - System Console Instrumentation

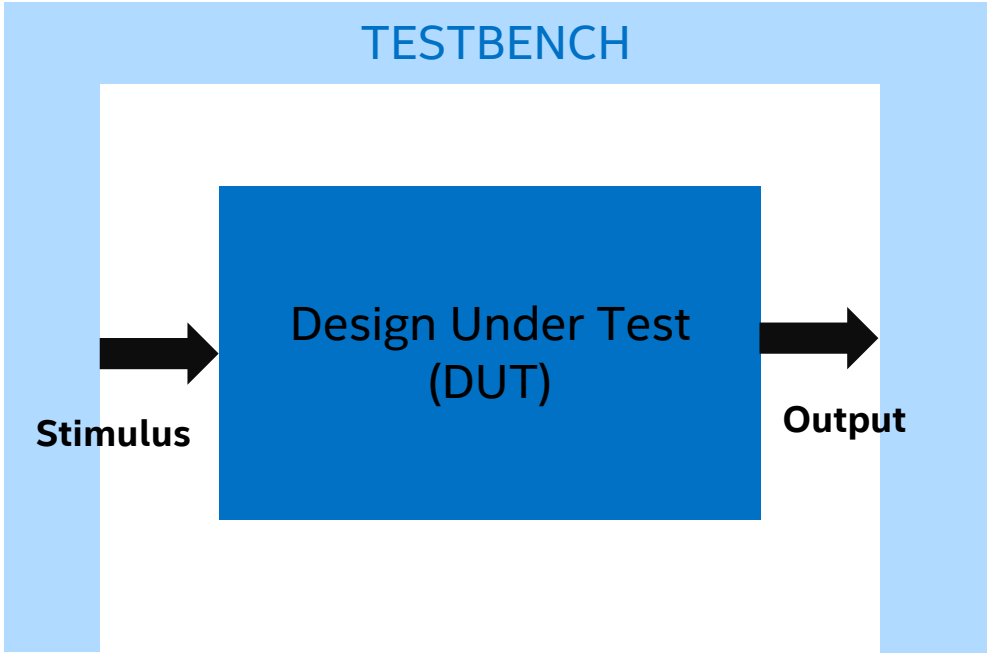
SIMULATION WITH MODELSIM

Why Simulation?

- + Include wide range of analyses
- + Reduce development costs
- + Brings innovative products faster to market
- + Provide results that are impossible to measure on physical prototype.
- + High visibility of all signals in design
- - Can take a very long time to run for large designs or excessive stimulus
- - Designer has to predict and create stimulus that matches actual behavior

Testbenches

A **test bench** or **testing workbench** is an environment used to verify the correctness or soundness of a design or model.



Verilog Testbench Constructs

```
`timescale 1ns/10ps

module tb_counter();

reg clock,reset_pll,reset_count,counter_direction;
wire [3:0] count_up , count_down;

top_counter DUT ( .refclk(clock) ,
                  .reset_pll(reset_pll) ,
                  .reset_count(reset_count),
                  .counter_direction(counter_direction),
                  .count_up(count_up),
                  .count_down(count_down)  );

    initial
    begin
        clock = 0;
        reset_pll =1;
        reset_count=1;
        counter_direction =0;

        #10 reset_pll=0;
        #30 reset_count=0;
        #10 counter_direction=1;
        #30 counter_direction=0;
        #10 counter_direction=1;
        #30 counter_direction=0;
        #10 counter_direction=1;
        #30 counter_direction=0;

        #1000 $stop;

    end

    always #10 clock = ~clock;

endmodule
```

Timescale

1st number is units, second is timing resolution

Inputs are reg, outputs are wires

Note: no module I/O

Initial Block

Runs only once (vs always block)

Stimulus

Best to change stimulus on the inactive edge of the clock – easier to read waveforms

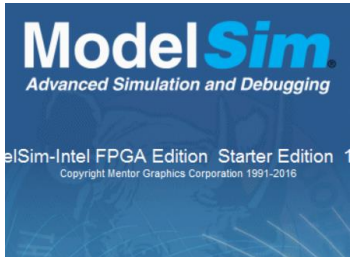
Use $stop vs $finish or simulator closes

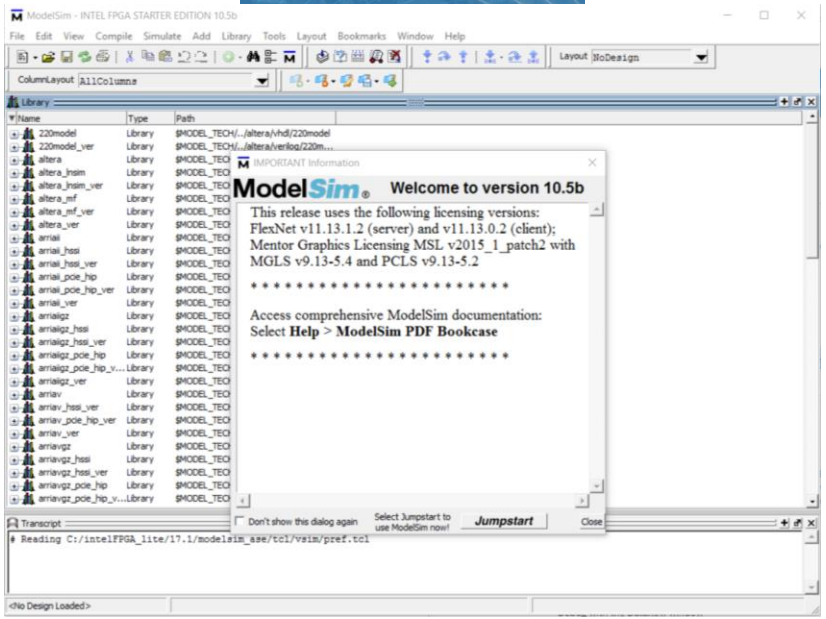
Clock

Mentor ModelSim Overview

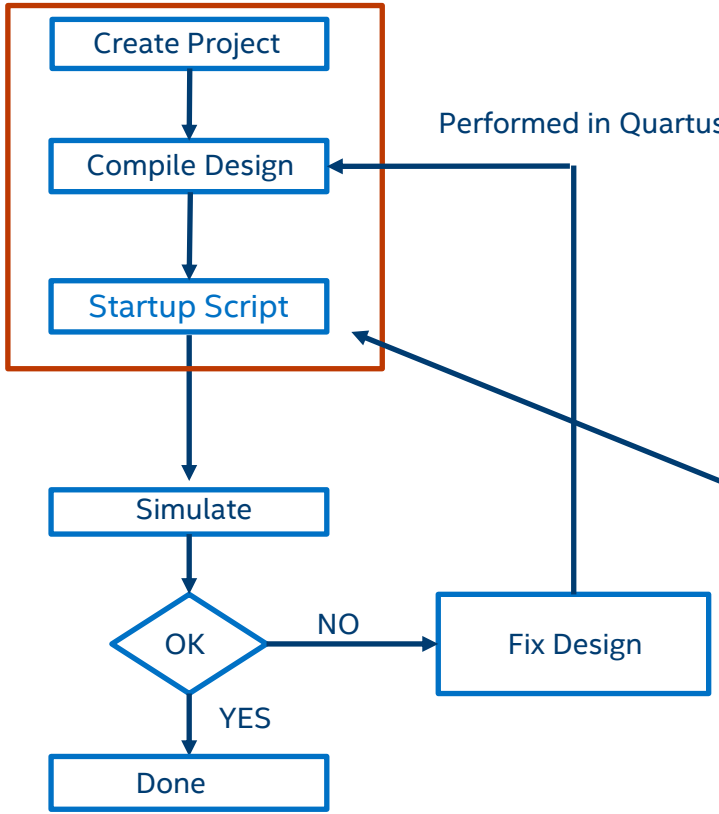
ModelSim is a multi-language HDL (Verilog/VHDL) simulation environment. It can be used independently or Intel Quartus can create startup scripts and link designs to ModelSim.

- Intel Quartus has a license to distribute Modelsim-Altera with Quartus .
- Free Starter Edition: <=10K lines of code, runs slower





ModelSim



Note:
Only functional (non-timing) simulation is supported

Back-annotated timing model gate level simulations are not supported

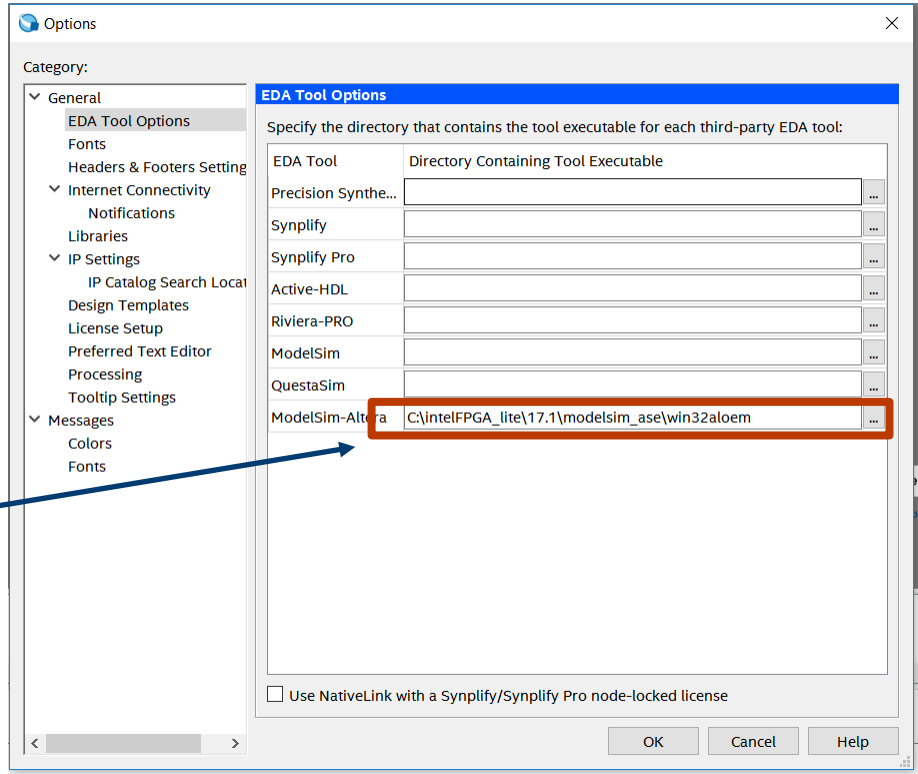
This script will link in all of external IP libraries so you don't have to

Setting up ModelSim from Intel Quartus

Specify EDA Tool Setting to generate simulation files.

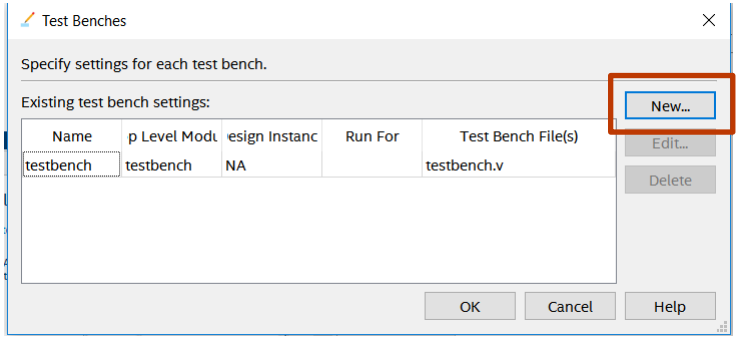
- Tools → Options → EDA Tool Options
  In ModelSim-Altera, enter the executable path

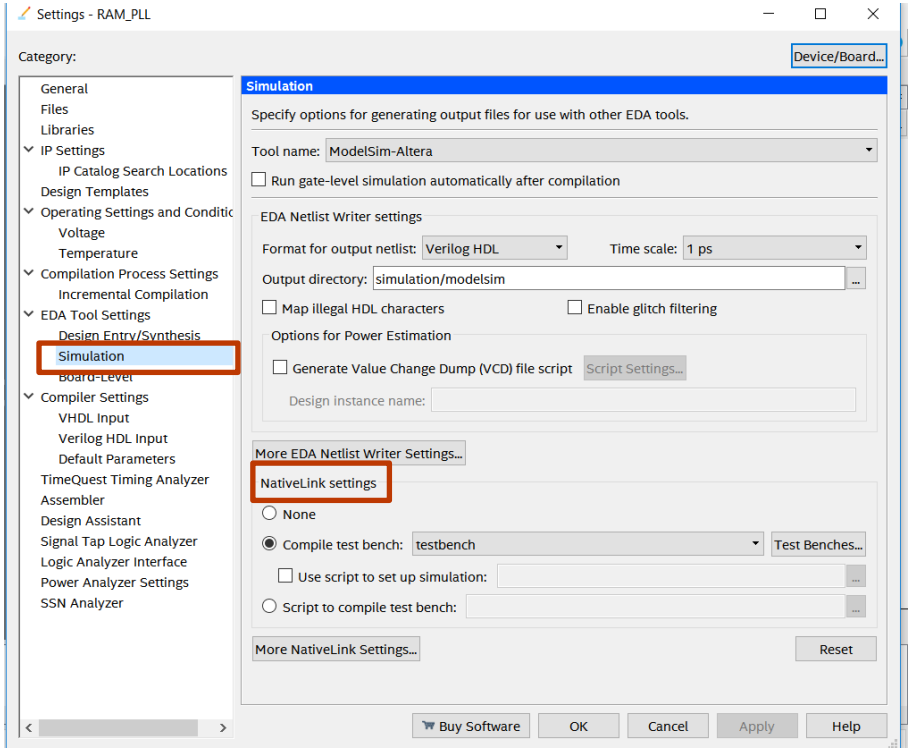
This path might be different for your own installation!



Setting up ModelSim from Intel Quartus

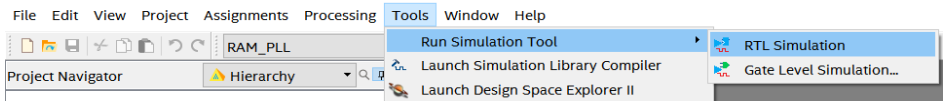
- Assignments → Settings→ EDA Tools Settings → Simulation
- In NativeLink Settings → Test Benches → NEW
- Add New testbench → OK



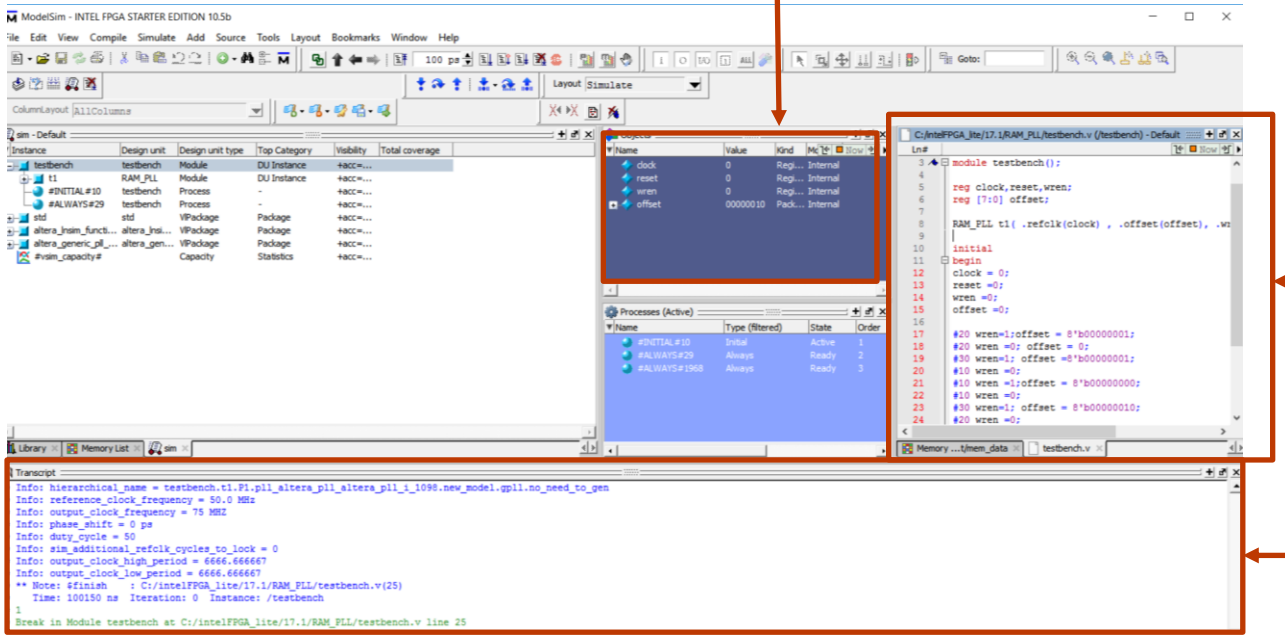


ModelSim GUI

Launching ModelSim from Quartus Tools →Run Simulation Tool → RTL Simulation



Simulation Objects



Testbench File

Command Transcript Window

LAB EXERCISE 1: MODELSIM

IN-SYSTEM SOURCES AND PROBES (ISSP)

Why ISSP

+ Quickly set signals to constants: pins or internal nodes

+ Easily monitor signals – non-triggered continuous display

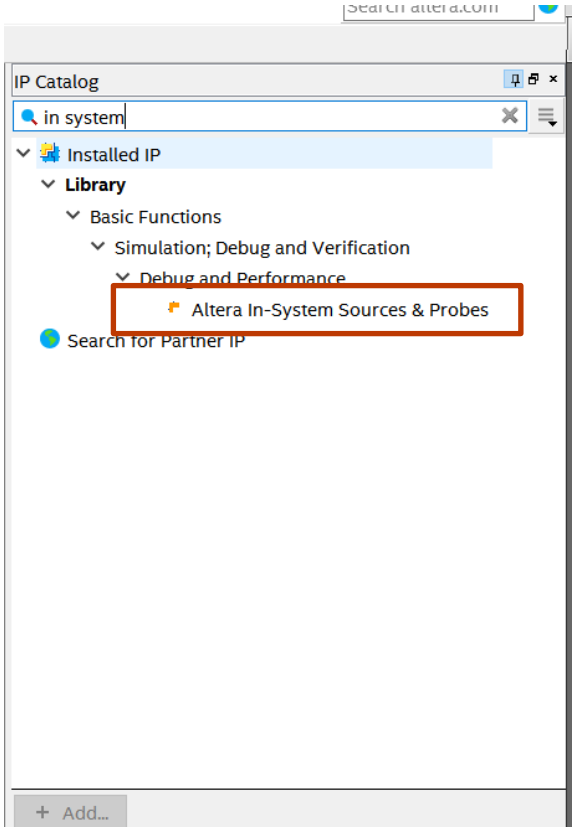
+ Works on actual hardware

- Not triggered – might miss activity

In-System Sources and Probes

ISSP allows an easy way to drive and sample signals in hardware and provides a dynamic debugging environment.

- ISSP Editor consists of a probe function and interface to control the instances during run time.
- It is operated over JTAG.
- Each instance can drive and toggle values up to 512 signals.
- Can create up to 128 instances of ISSP using IP Catalog

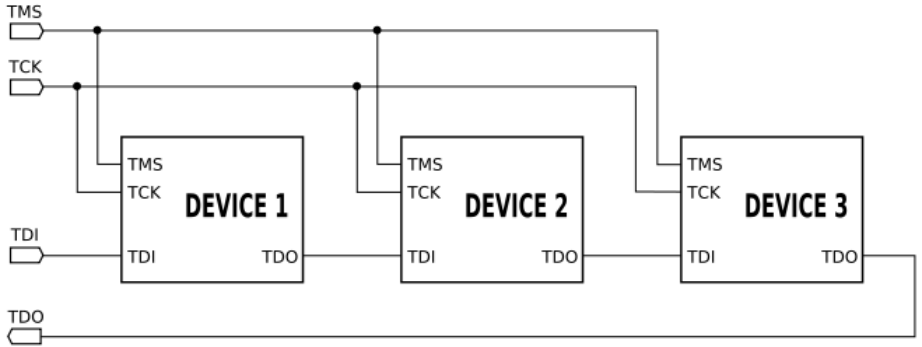


What is JTAG - Joint Test Action Group (IEEE 1149)

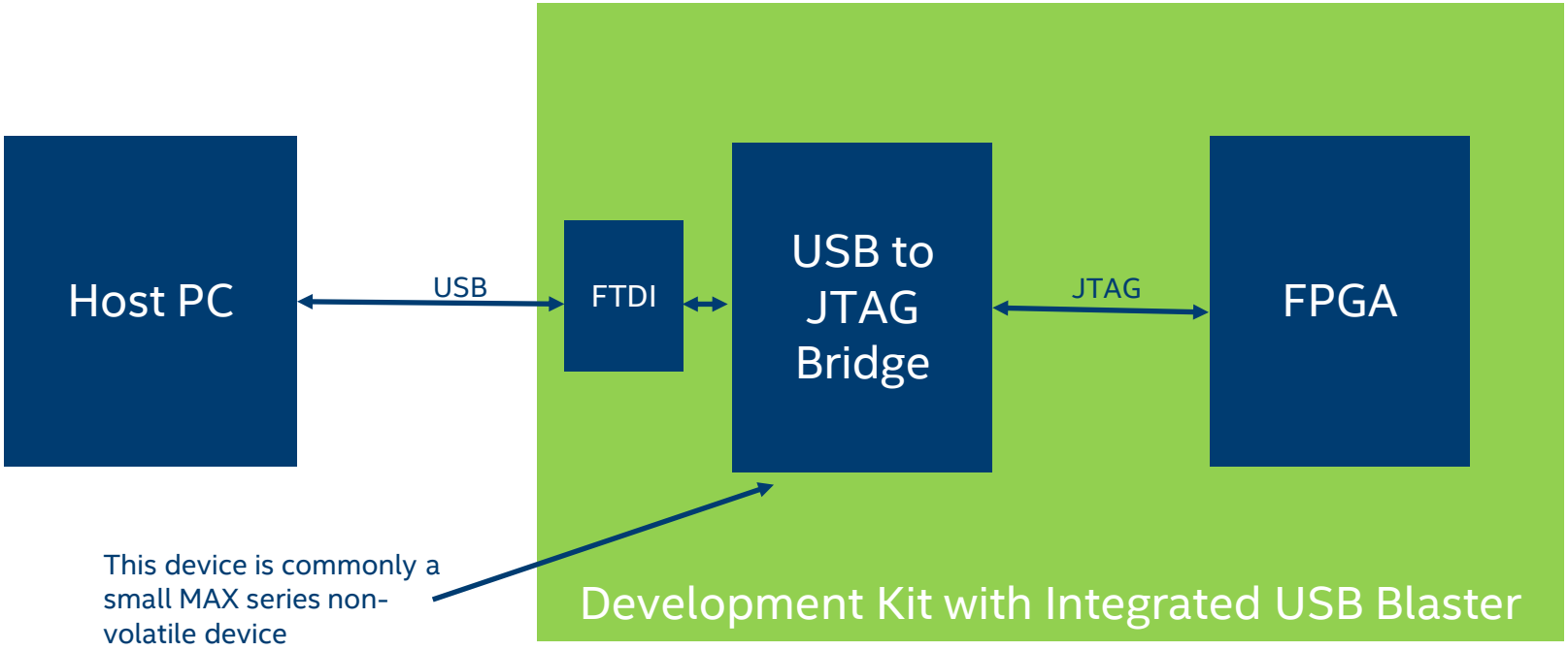
JTAG = JTAG is an industry standard for verifying designs and testing printed circuit boards after manufacture. JTAG implements standards for on-chip instrumentation in electronic design automation as a complementary tool to digital simulation.

- FPGAs use this bus as one of the means to configure the device and interface with internal structures in the device

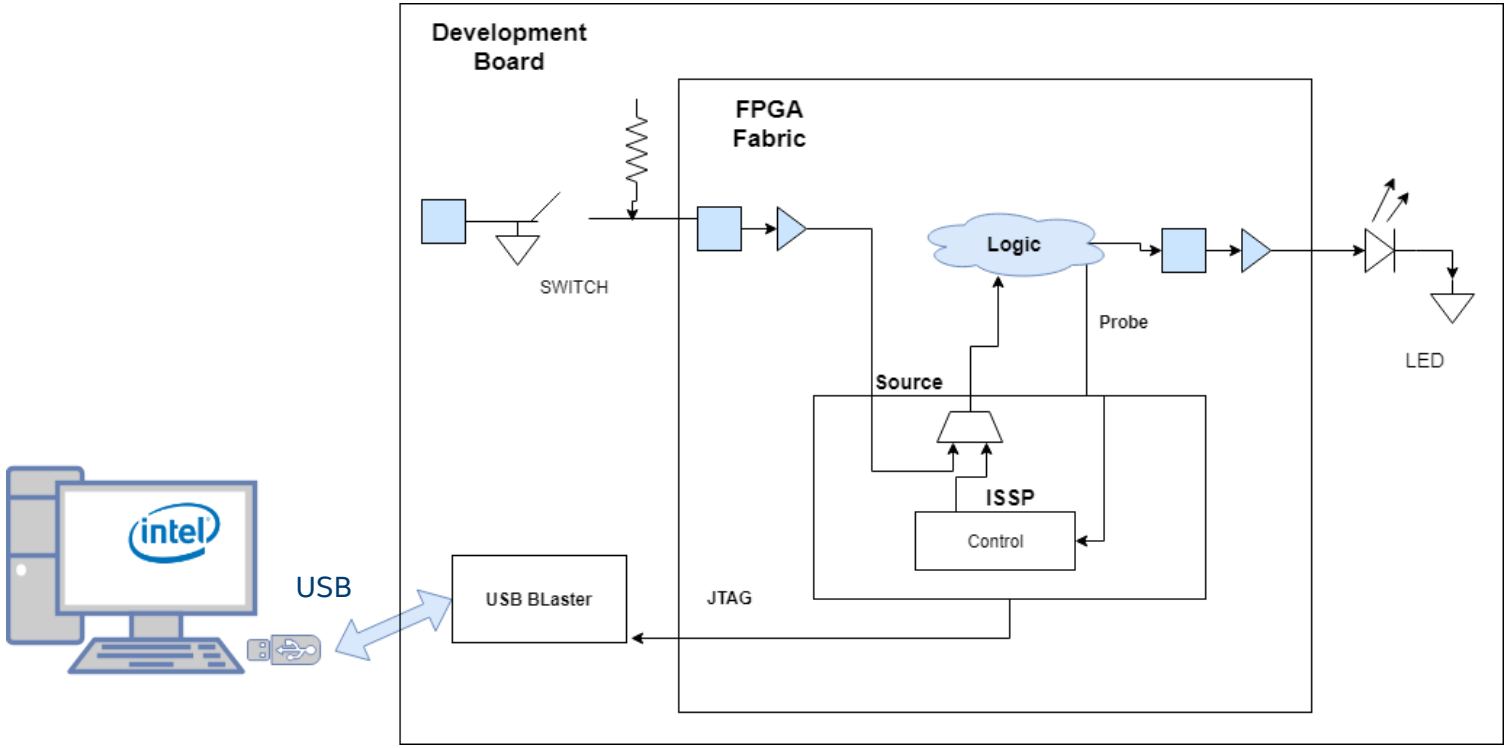
Standard 4 or 5 wire bus – used in many digital electronic devices for test and device specific configuration



USB Blaster Bridge Circuit



In-System Sources and Probes Block Diagram

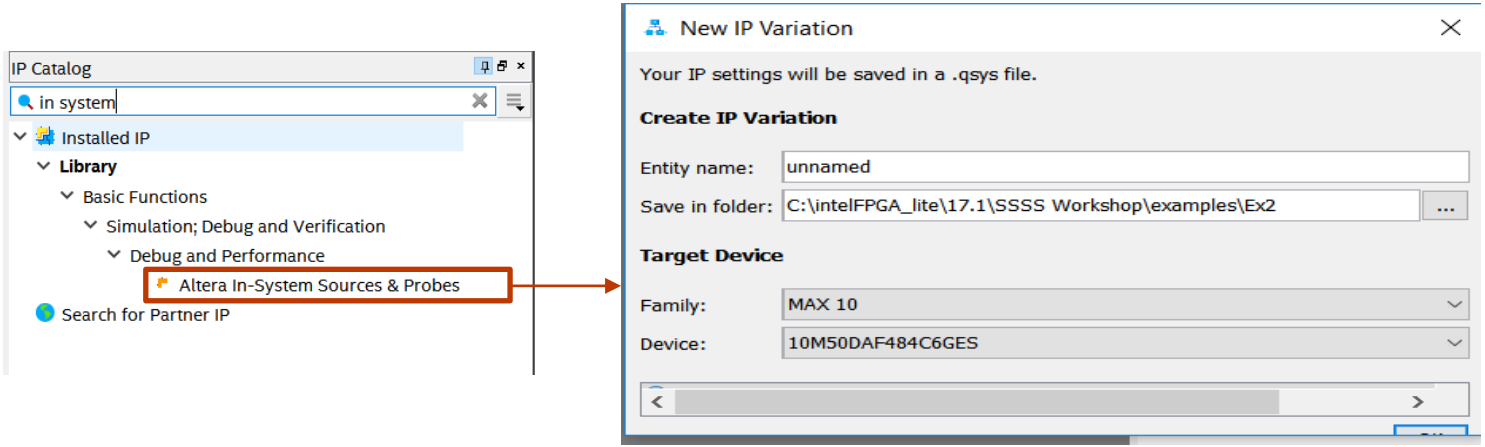


Example Uses

- Prototype a front panel with virtual buttons for a FPGA Design
- Monitor results of changing design constants
- Extensive TCL scripting support to create custom automated design control interfaces

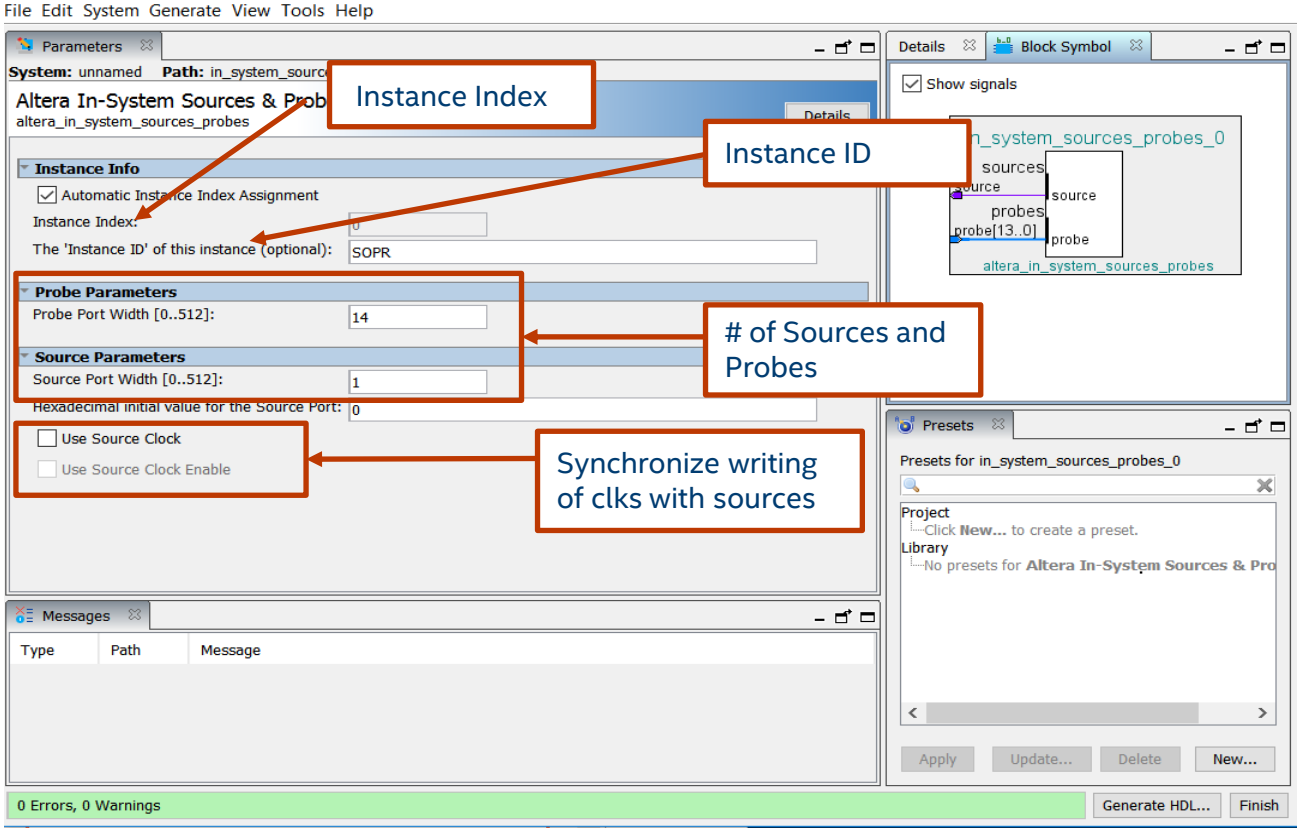
Using In-System Sources and Probes

- Create In-system Sources and Probes IP instances using IP parameter Editor /MegaWizard Plug-in Manager (through IP catalog)



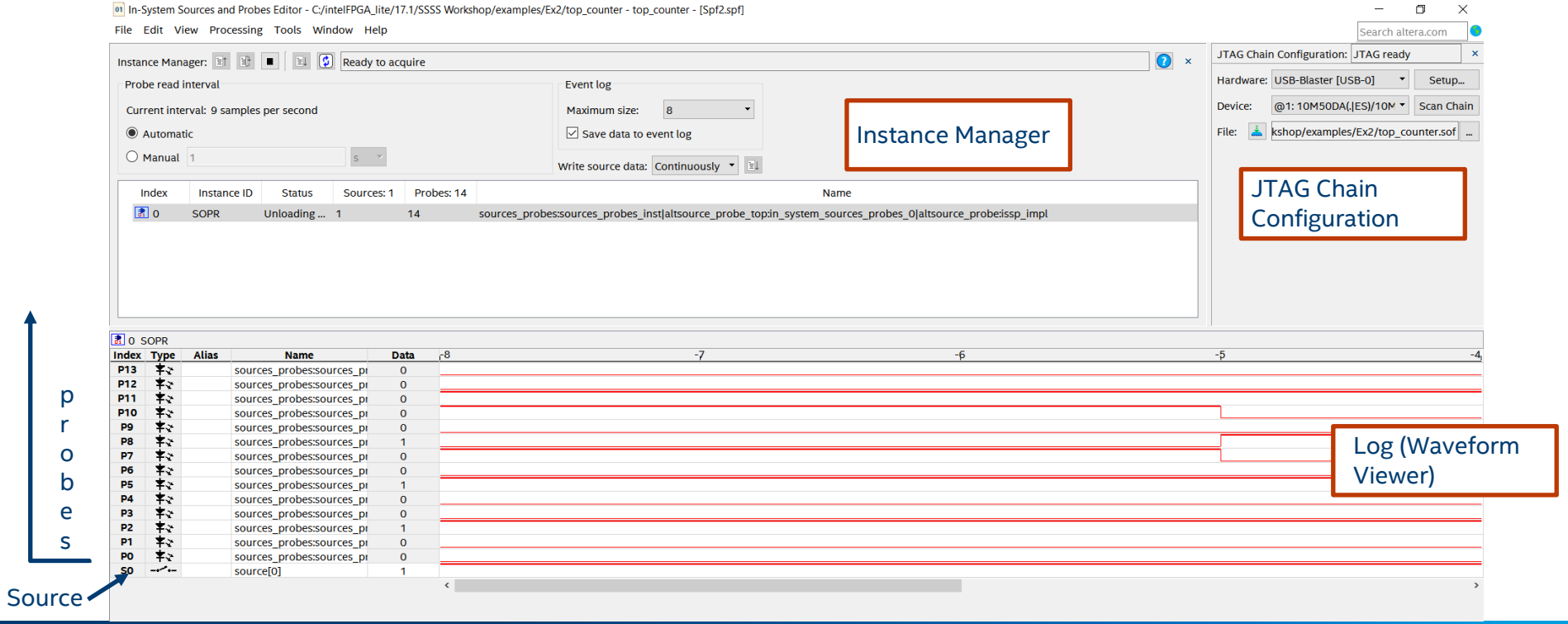
- Instantiate in design & compile
- Program target device

IP Parameter Editor for ISSP



Using In-System Sources and Probes

Create and use ISSP Editor (.spf file) to control sources and probes



SIGNALTAP

Embedded Logic Analyzer

Why Signal Tap

+ Easily monitor signals – using simple to elaborate triggering schemes
+ No external equipment required
+ Don’t need to figure out stimulus since its based on actual hardware
- Uses up lots of memory resources inside the chip
- Can change timing of design
- Requires recompile which takes time

Debug of a Design with an External Logic Analyzer

Pros:

- System-level debug
- Can store large quantities of data
- Flexible trigger condition

Cons

- Signals must be physically accessible on the board by a probe
- FPGA must have available I/O
- If you need a new signal that isn't accessible, you must make a new board
- Probe equipment can potentially effect signal integrity
  - High quality probes prevents this, but tend to be expensive
- Equipment expensive for hobbyist

Signal Tap Logic Analyzer

SignalTap is a logic analyzer made of available resources *inside* the FPGA

- Uses available logic elements to implement the Logic Analyzer
- Samples on-chip signals on the rising edge of a specified clock signal
- View captured data through the standard JTAG connection typically used for programming the device

Debug of a Design with Signal Tap

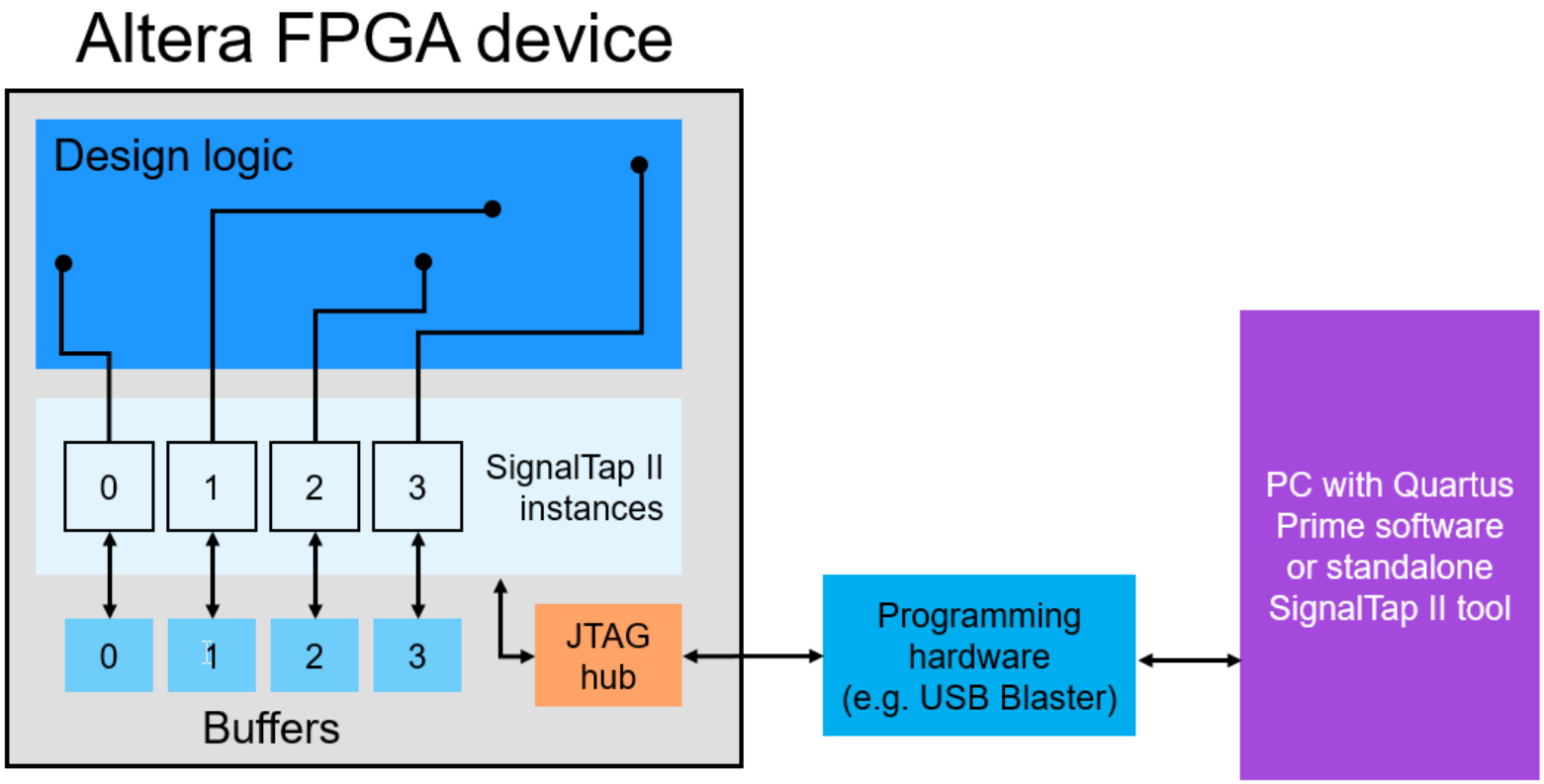
Pros:

- Tap signals buried deep in the design
- No unassigned I/Os or routing needed
- Comes free with all versions of Quartus, no external test equipment required
- Tap new signals with the same board by reconfiguring, recompiling, and reprogramming (no re-spin!)

Cons

- Requires additional device resources (memory and logic elements) – doesn't change base function, but changes timing
- Must have an active JTAG connection

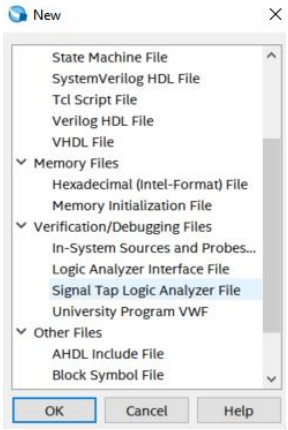
What is Signal Tap ?



Create a Signal Tap instance in two ways

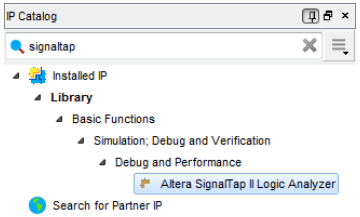
1. Use Signal Tap file (.stp) (*recommended)*

- Creates a file (.stp) separate from design files
- Convenient features and GUI



2. Use IP Catalog and IP Parameter Editor

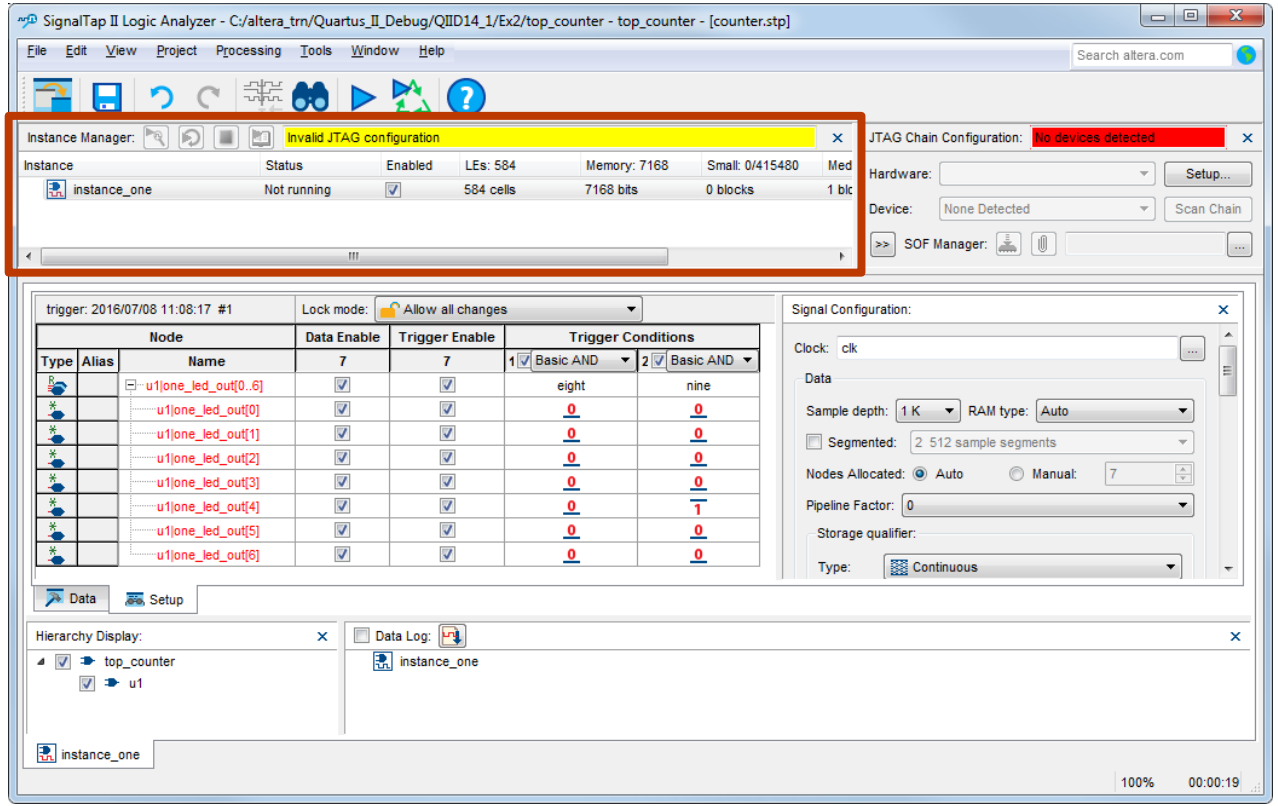
- Manually instantiate **altera_signaltap_ii_logic_analyzer** IP core directly into HDL code or Qsys (Platform Designer)
- Ties the ELA to the signals directly in RTL



Signal Tap Logic Analyzer Window

Instance Manager

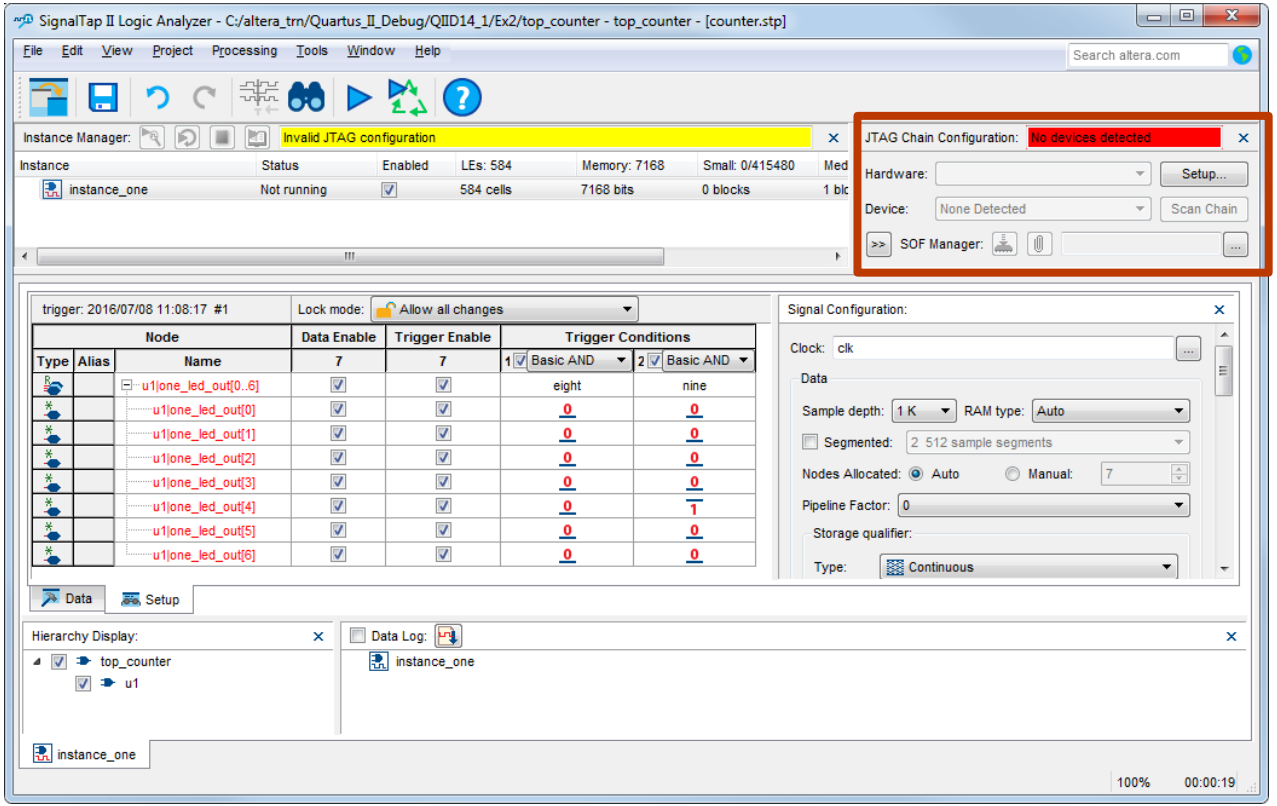
- Identifies which instance is being edited in the GUI
- Enable/Disable instances quickly
- Gives status and resource utilization (LEs and memory)



Signal Tap Logic Analyzer Window

JTAG Chain Configuration

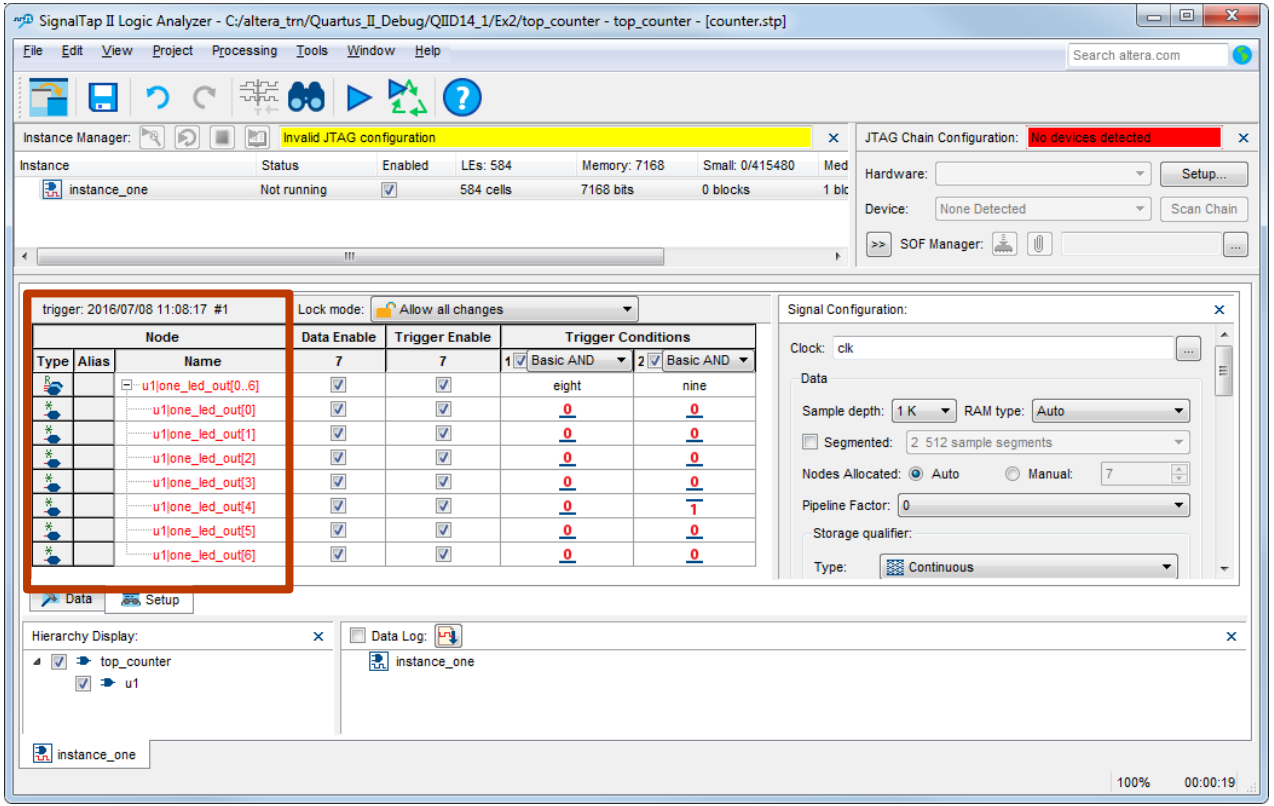
- Built in “Programmer”
- Scans the JTAG chain and identifies available devices



Signal Tap Logic Analyzer Window

Nodes List

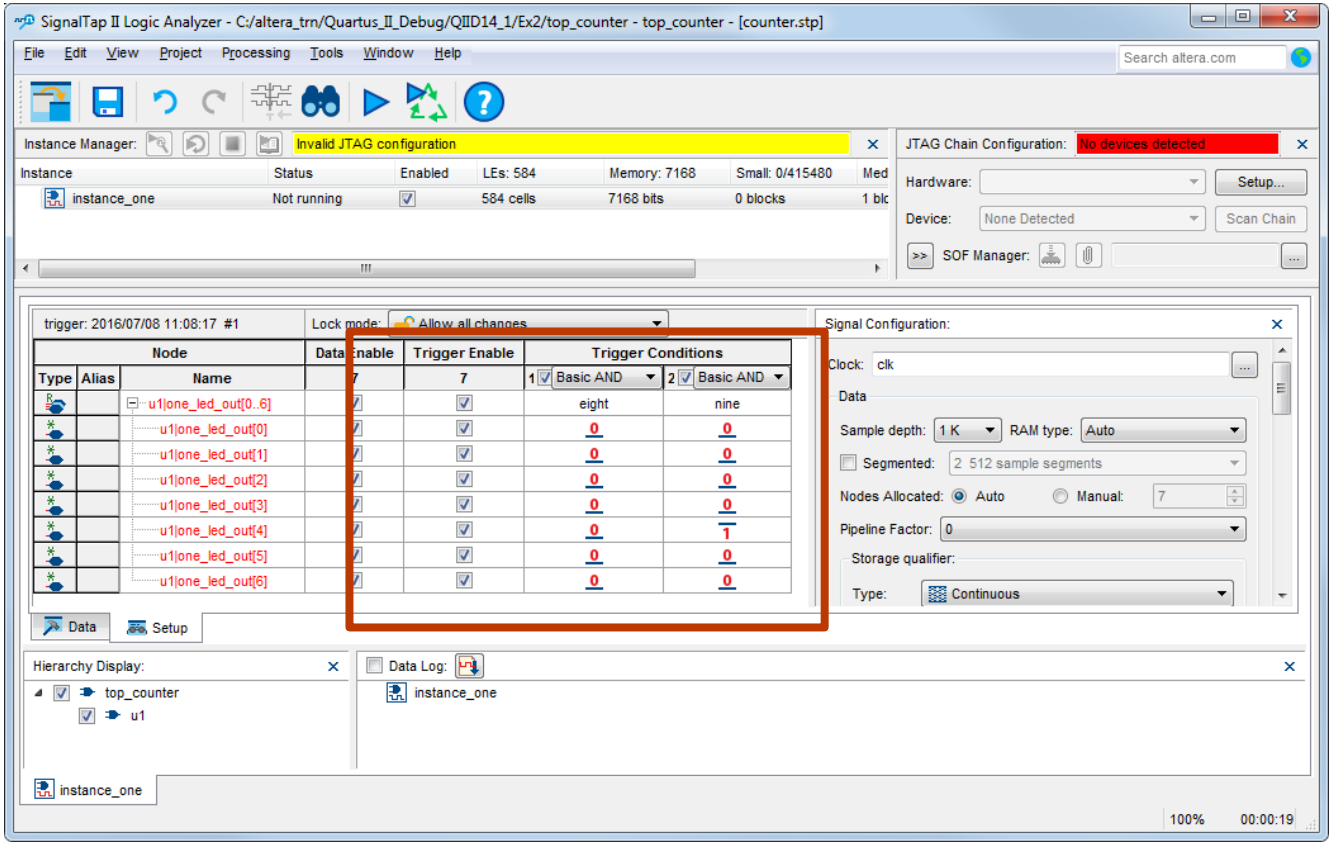
- Use the Node Finder to add signals to be tapped
- Automatically groups busses together and create custom groups



Signal Tap Logic Analyzer Window

Trigger Conditions and Qualifiers

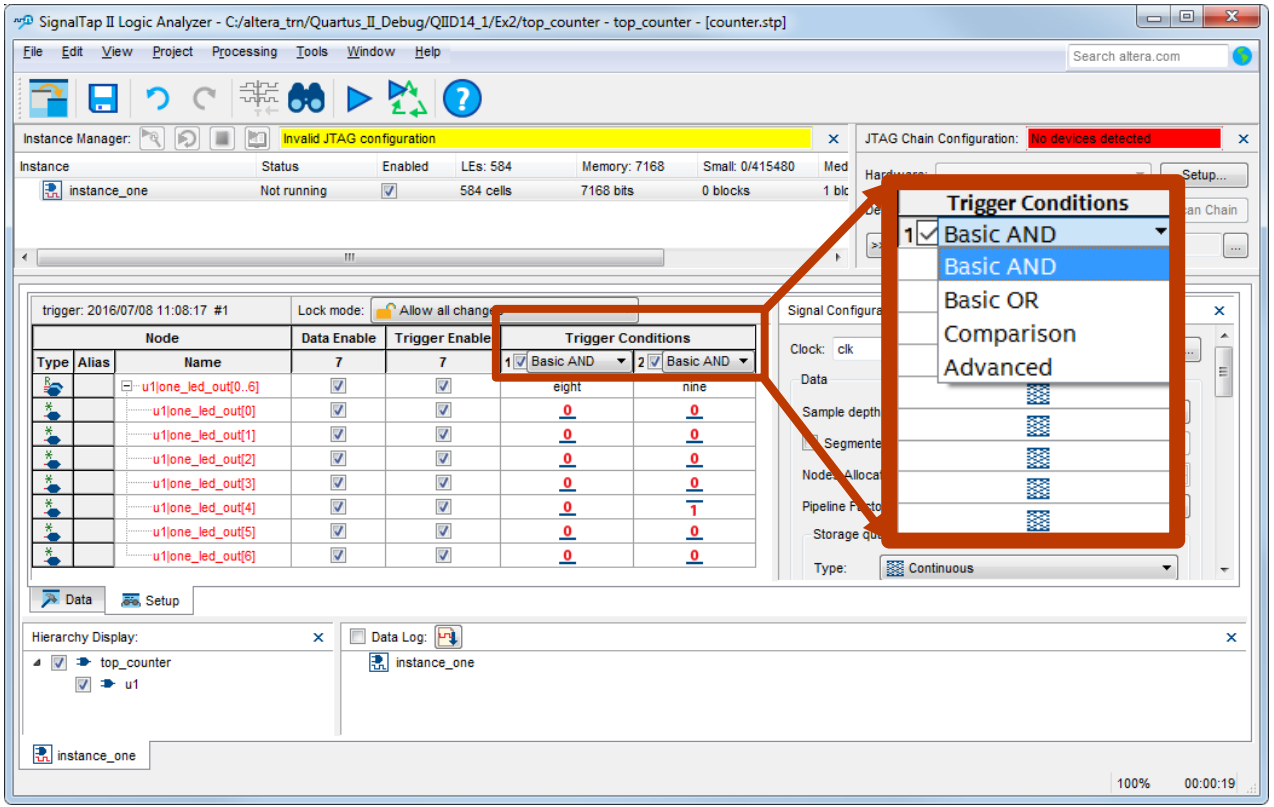
- Data Enable: Saves signal data (disable to save memory)
- Trigger Enable: Signal is part of the trigger condition (disable to save LEs)



Signal Tap Logic Analyzer Window

Trigger Conditions

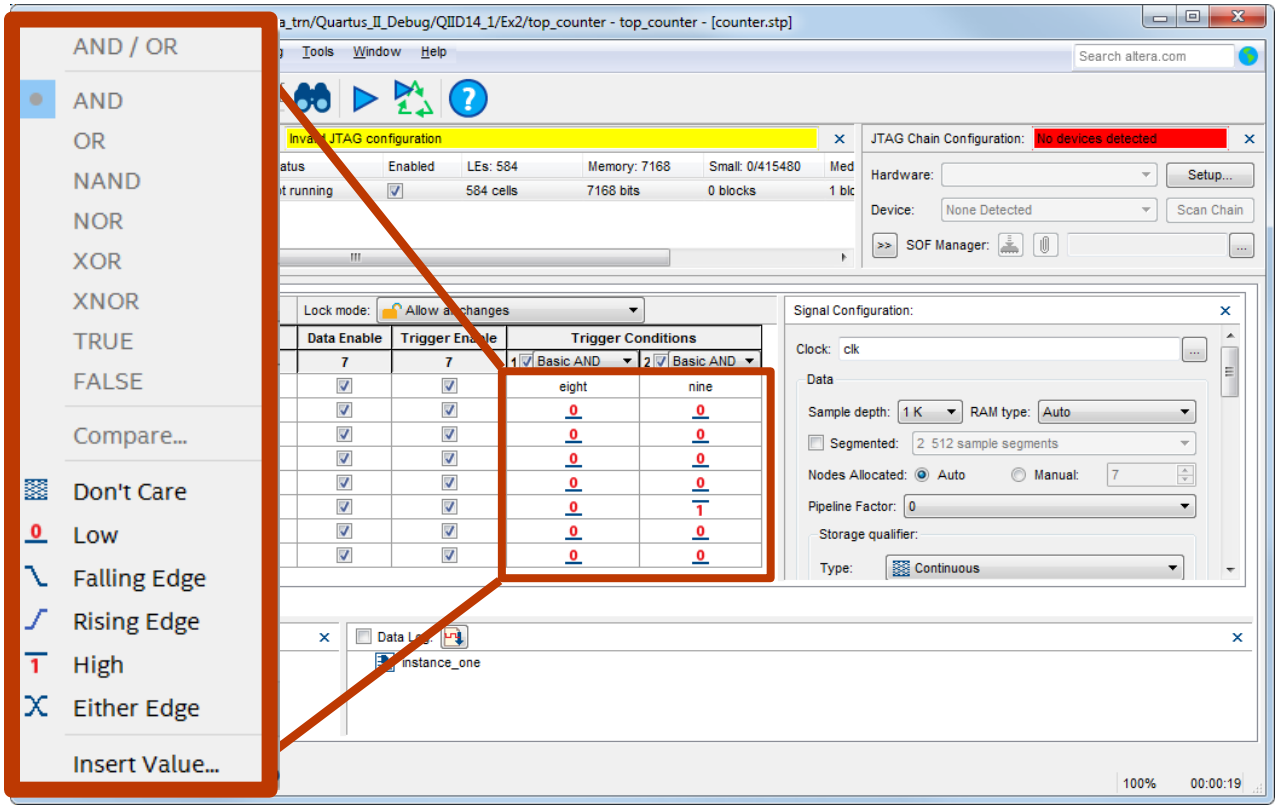
- Add up to 10 trigger conditions
- Choose how every node is compared



Signal Tap Logic Analyzer Window

Trigger Conditions

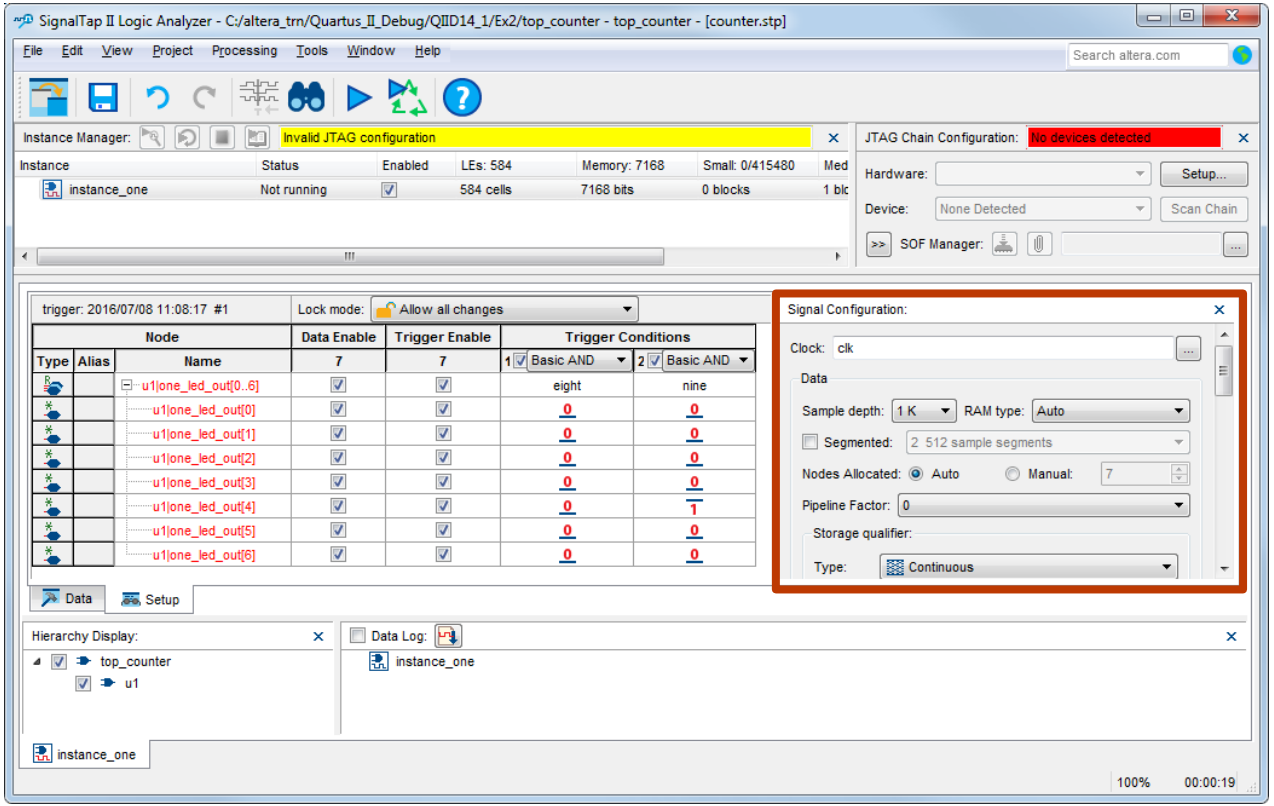
- Add up to 10 trigger conditions
- Choose how every node is compared
- Choose what action triggers a specific node



Signal Tap Logic Analyzer Window

Signal Configuration

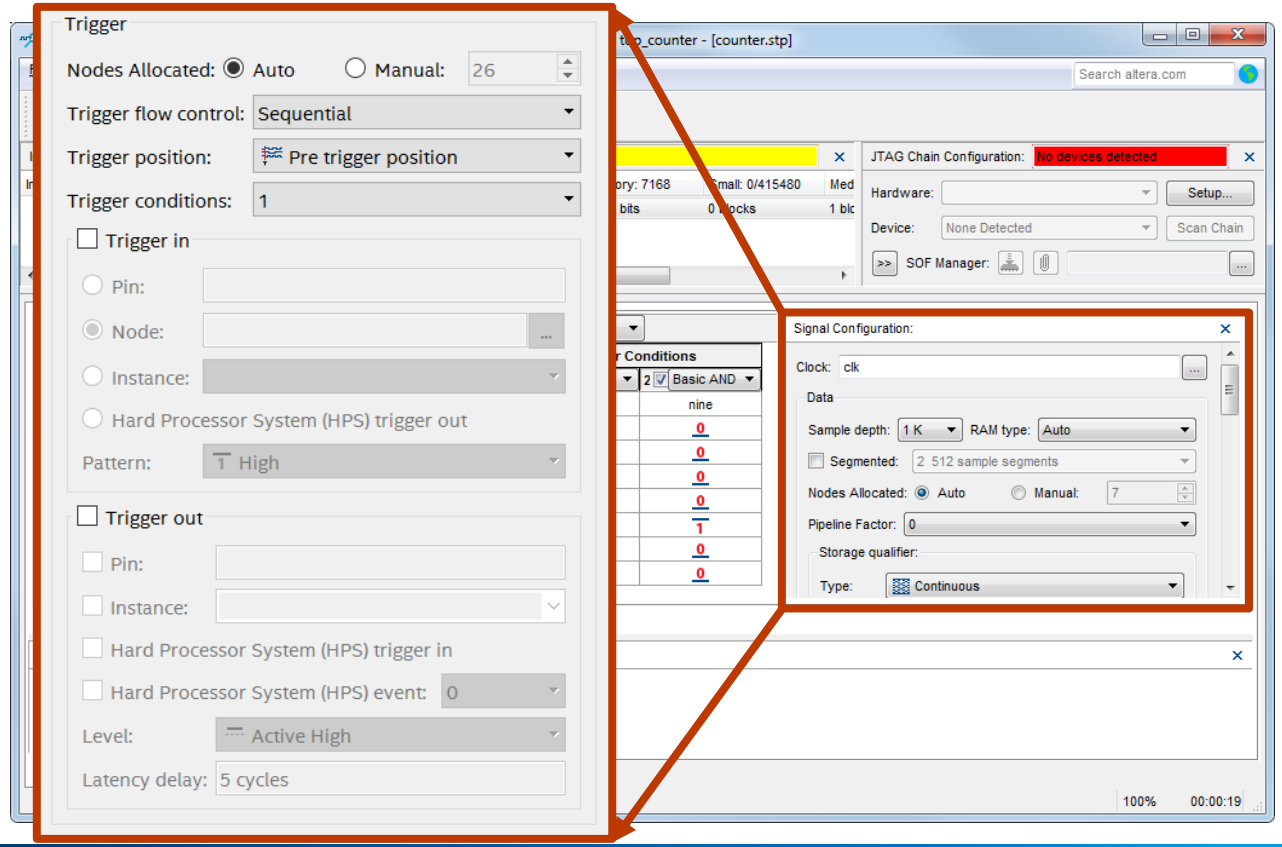
- Select which clock runs the instance
- **Sample Depth:** how much data from each signal is stored



Signal Tap Logic Analyzer Window

Signal Configuration

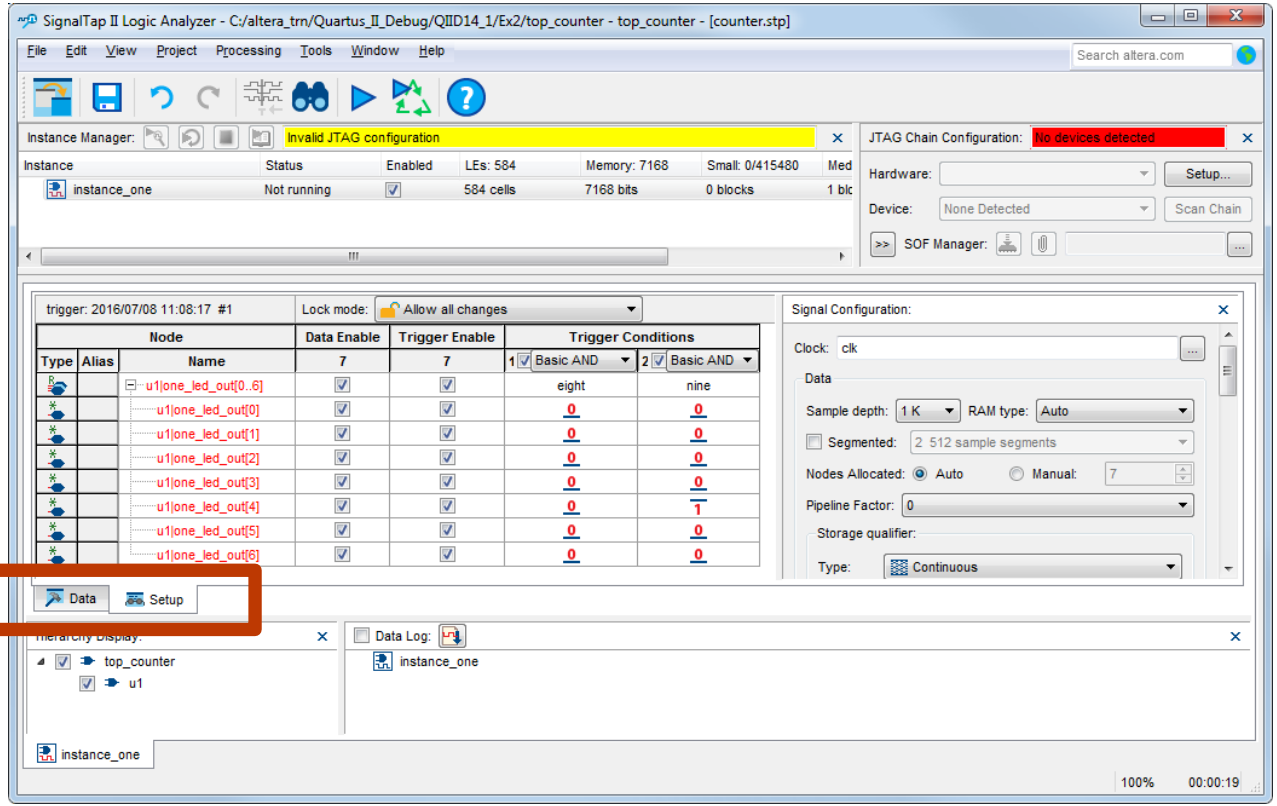
- Advanced trigger control
- Select the number of trigger conditions
- Trigger In/Out options



Signal Tap Logic Analyzer Window

Data/Setup Window

- **Setup** allows configuration of nodes and trigger conditions (for making edits)
- **Data** shows the acquired signal information (for viewing results)



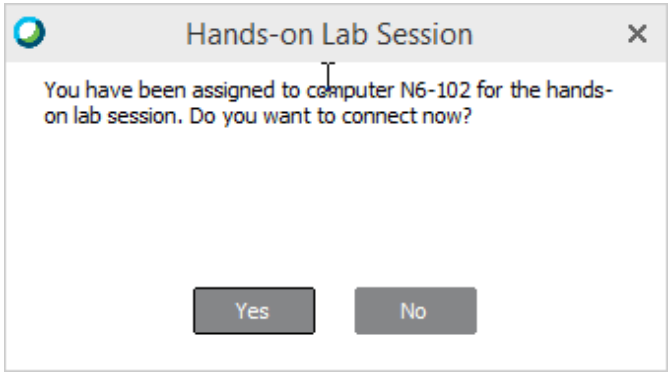
LAB EXERCISE 2: IN-SYSTEM SOURCES AND PROBES
LAB EXERCISE 3: SIGNAL TAP

Lab Notes (1)

Once the lab has started and you have been assigned to a lab computer, the pop-up below will appear.

Click Yes to connect to the lab computer.

Login: **Student**
Password: **QPrime.1**

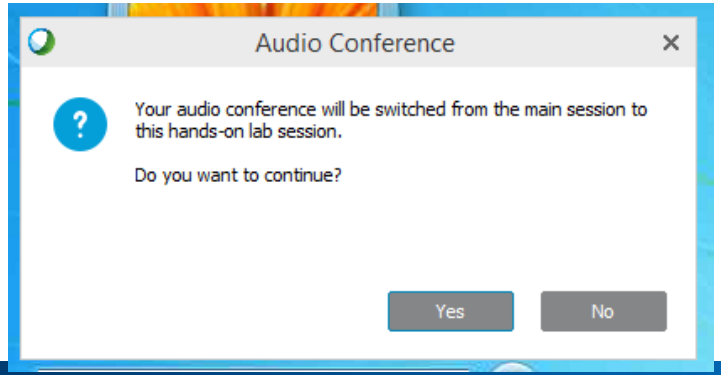


Lab Notes (2)

Instructor is able to view your screen if necessary

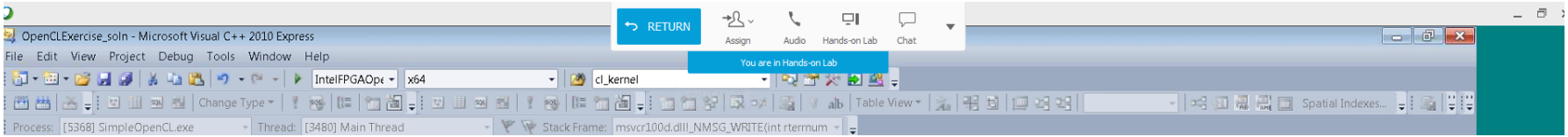
If you get stuck and need help, exit to the main conference room and type in the chat that you need help. If you type in chat window while connected to your PC, the instructor cannot see that you need assistance.

When he/she connects, click *YES* if the following pop-up appears; this is important. This will move your audio to a chat room connected to your machine



Lab Notes (3)

To return to the main presentation WebEx window, move the mouse pointer to the top of the window; click RETURN in the drop-down



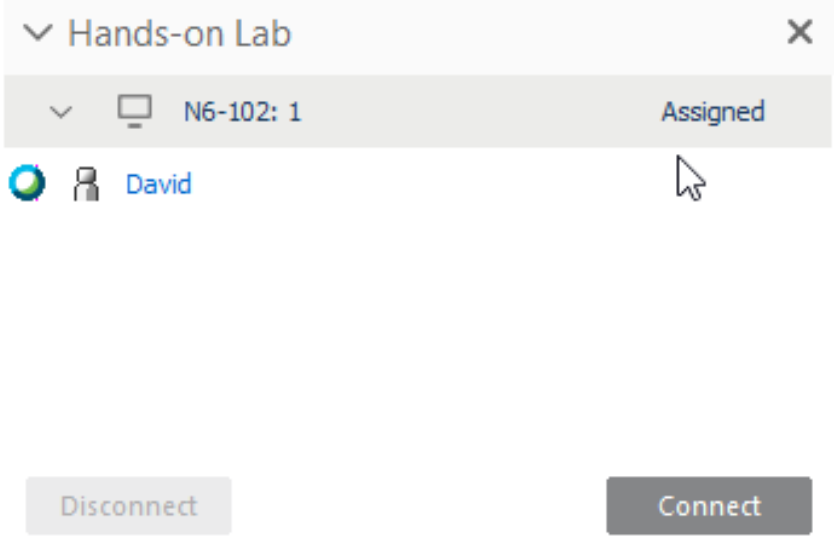
After clicking return, you will get a pop-up asking if you want to disconnect

It is ok to disconnect; you can reconnect later

NOTE: * keep track of which PC you are connected to

Lab Notes (4)

To reconnect to the lab computer just highlight the computer number and click **Connect**

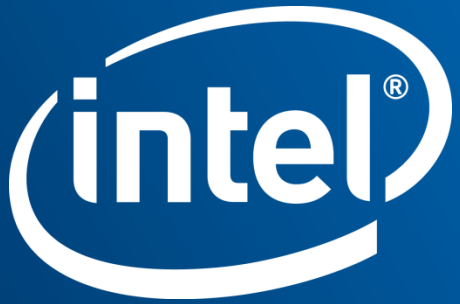


Final Lab Notes

Make note of your computer number; you will need to know which one to reconnect to

It is possible to cut and paste text into the virtual computer

- Notepad, Notepad++, and gvim all work well


intel®