



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



BSEE



MSEE



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

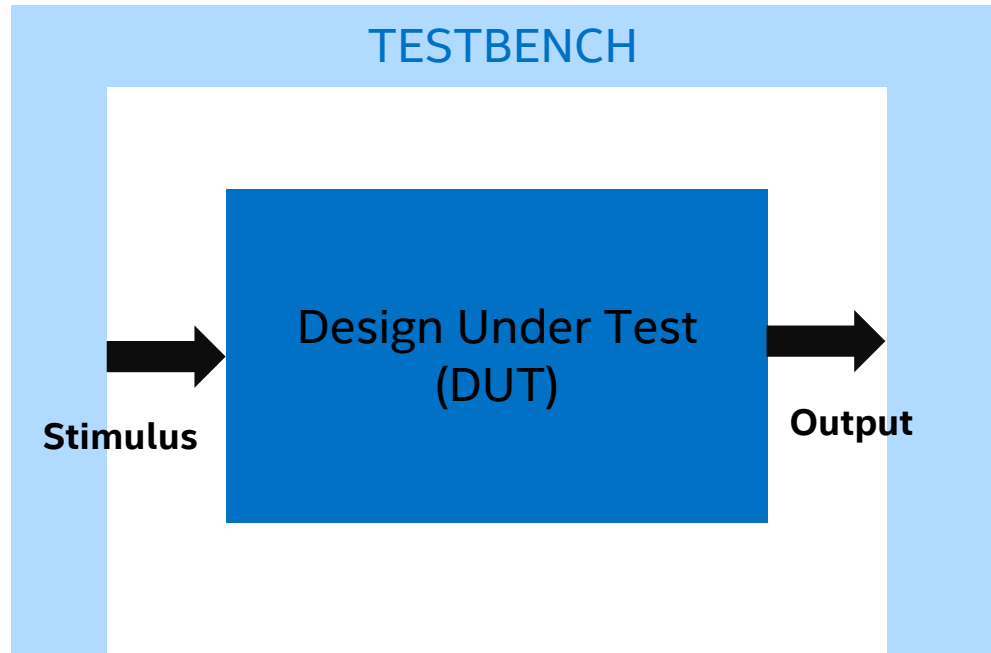
SIMULATION WITH MODELSIM

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
```

Timescale

1st number is units, second is timing resolution

```
module tb_counter()
```

```
reg clock, reset_pll, reset_count, counter_direction;
```

```
wire [3:0] count_up , count_down;
```

Inputs are reg, outputs are wires

Note: no module I/O

```
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;
```

Initial Block

Runs only once (vs always block)

```
#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;
```

Stimulus

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

```
#1000 $stop;
```

Use \$stop vs \$finish or simulator closes

```
end
```

```
always #10 clock = ~clock;
```

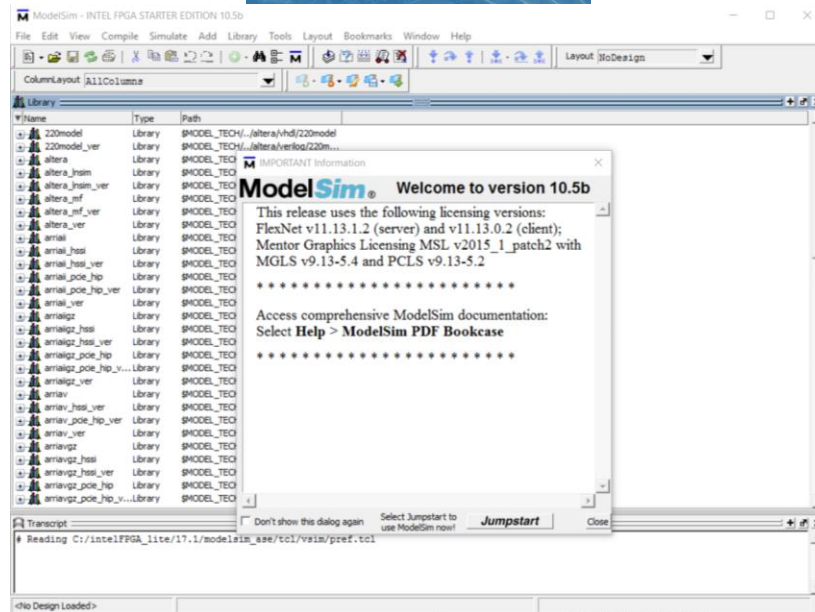
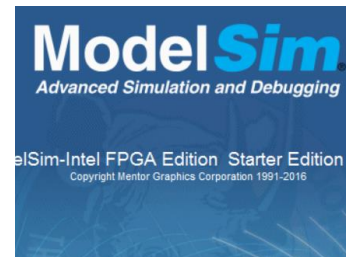
Clock

```
endmodule
```

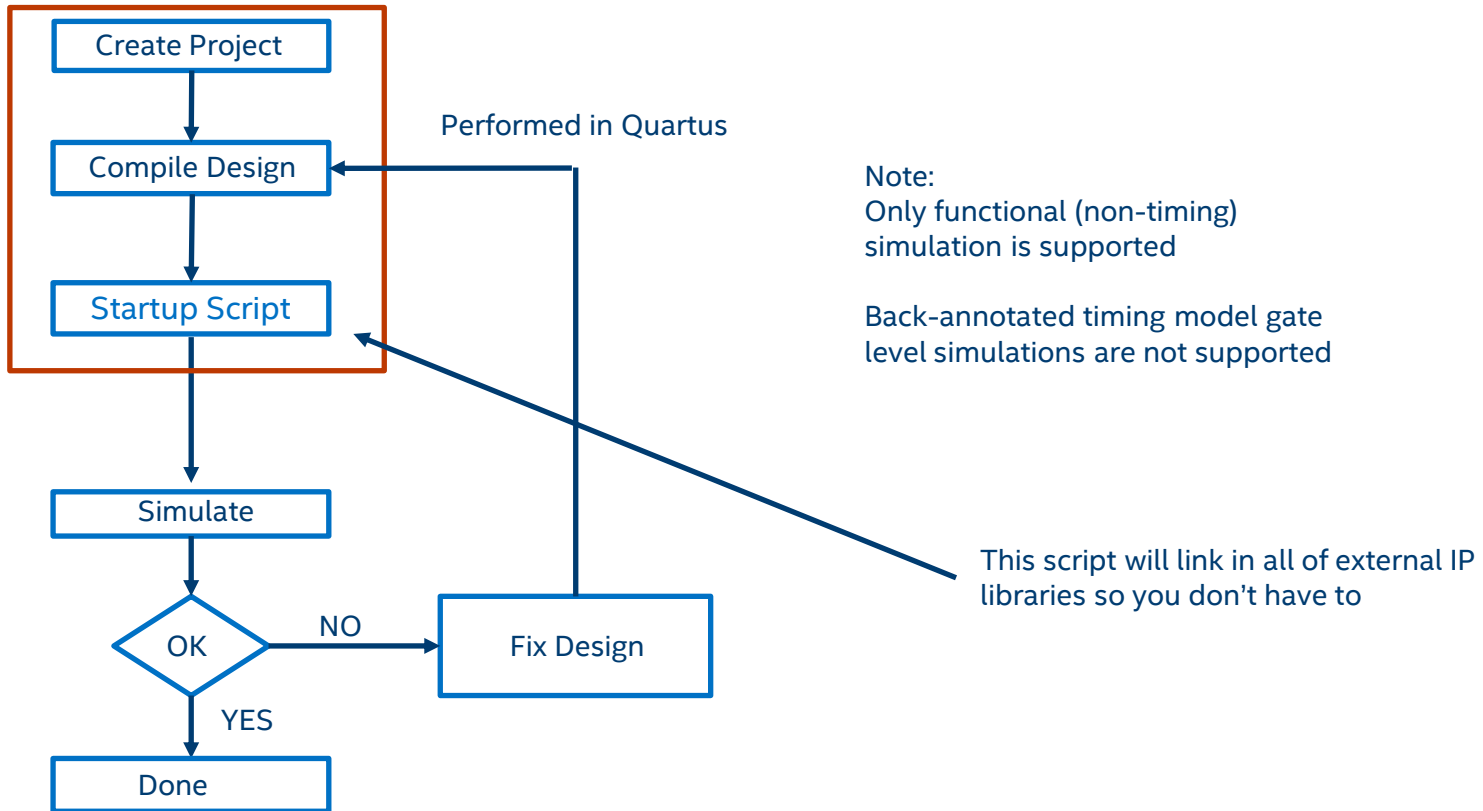
Mentor ModelSim Overview

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: $\leq 10K$ lines of code, runs slower



ModelSim

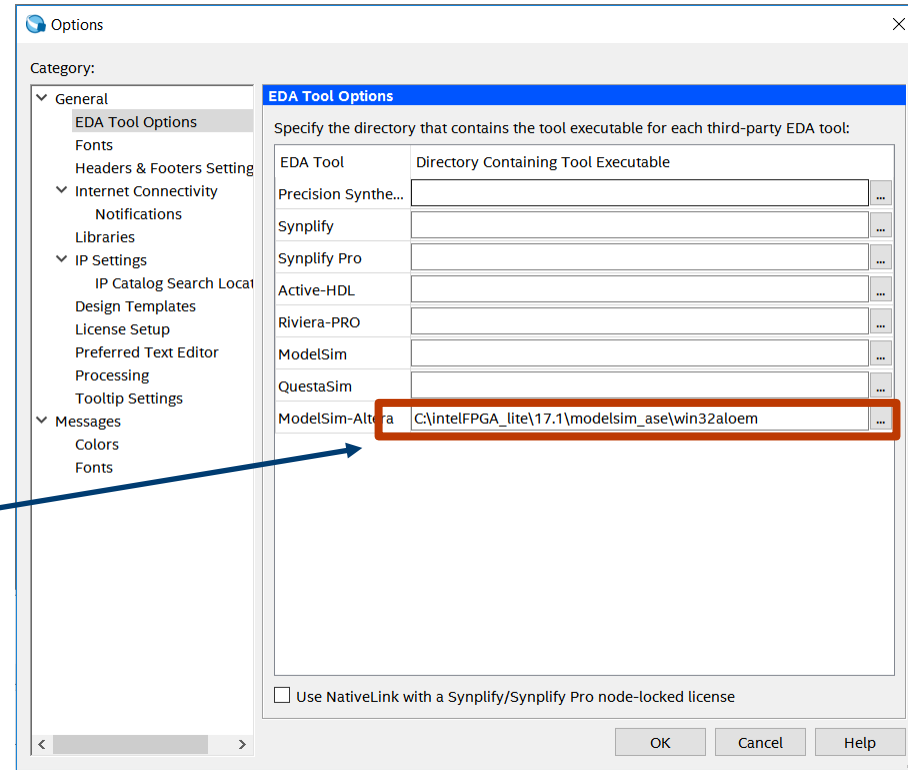


Setting up ModelSim from Intel Quartus

Specify EDA Tool Setting to generate simulation files.

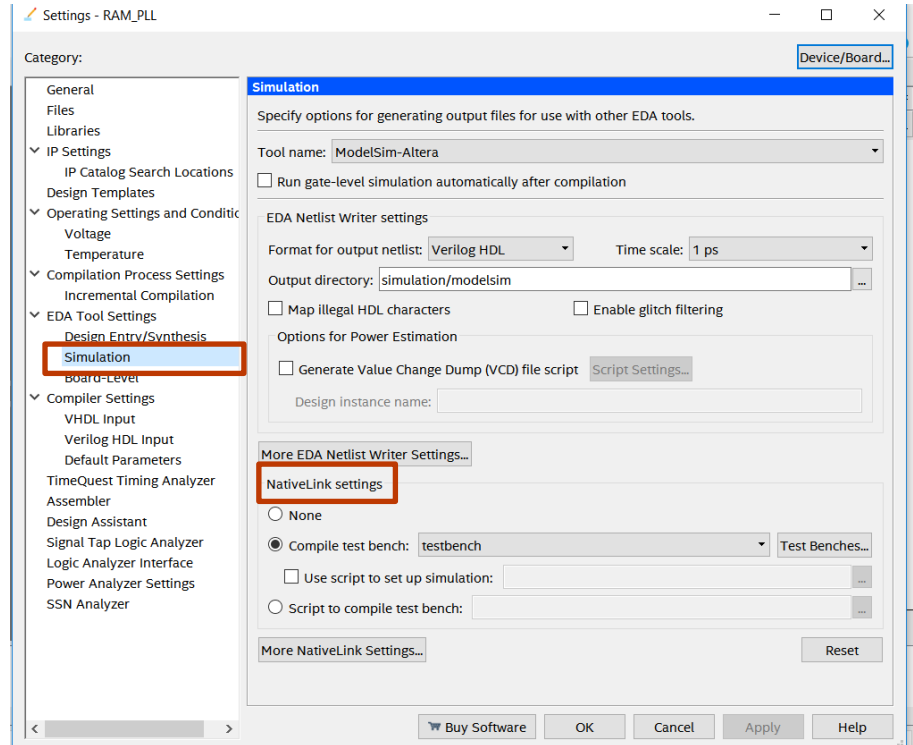
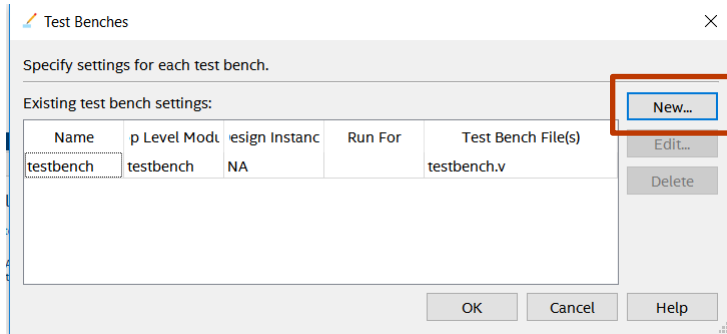
- Tools → Options → EDA Tool Options
In ModelSim-Altera, enter the executable path

This path might be different for your own installation!



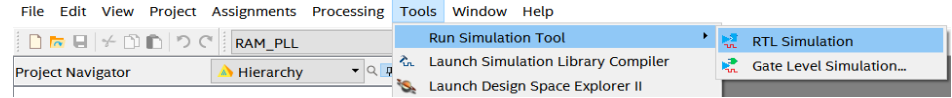
Setting up ModelSim from Intel Quartus

- 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

The screenshot shows the ModelSim - INTEL FPGA STARTER EDITION 10.5b interface. The 'Testbench File' window is open, showing the testbench code. The 'Command Transcript' window is also open, showing the simulation results. The 'Simulation Objects' window is also visible, showing the hierarchy of the design.

Testbench File

```
1 module testbench();
2   reg clock, reset, wren;
3   reg [7:0] offset;
4
5   RAM_PLL t1(.refclk(clock), .offset(offset), .wren(wren));
6
7   initial
8   begin
9     clock = 0;
10    reset = 0;
11    wren = 0;
12    offset = 0;
13
14    #20 wren = 1; offset = 8'b00000001;
15    #20 wren = 0; offset = 0;
16    #30 wren = 1; offset = 8'b00000001;
17    #10 wren = 0;
18    #10 wren = 1; offset = 8'b00000000;
19    #10 wren = 0;
20    #30 wren = 1; offset = 8'b00000010;
21    #20 wren = 0;
22  end
```

Command Transcript

```
Info: hierarchical_name = testbench.t1.P1.altera_pll_altera_pll_1_1098.new_model.gpll.no_need_to_gen
Info: reference_clock_frequency = 50.0 MHz
Info: output_clock_frequency = 75 MHz
Info: phase_shift = 0 ps
Info: duty_cycle = 50
Info: sram_additional_refclk_cycles_to_lock = 0
Info: output_clock_high_period = 6666.666667
Info: output_clock_low_period = 6666.666667
** Note: #finish : C:/intelFPGA_lite/17.1/RAM_PLL/testbench.v(25)
Time: 100150 ns Iteration: 0 Instance: /testbench
Break in Module testbench at C:/intelFPGA_lite/17.1/RAM_PLL/testbench.v line 25
```

LAB EXERCISE 1: MODELSIM

IN-SYSTEM SOURCES AND PROBES (ISSP)

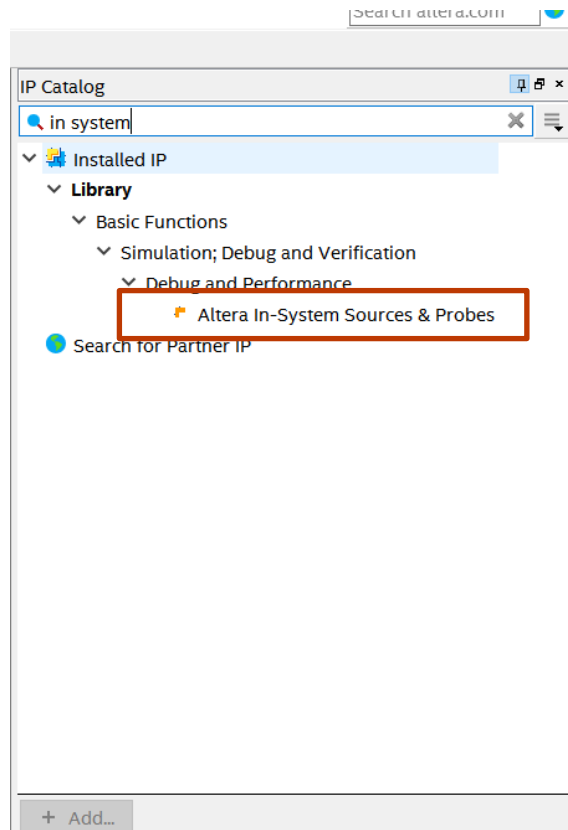
Why ISSP

- + 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

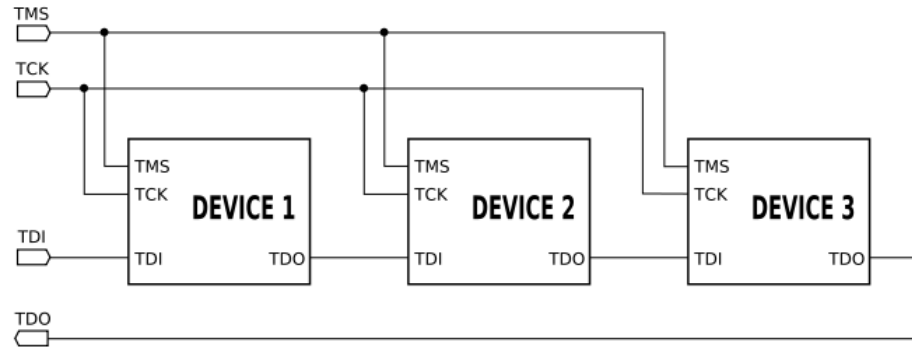


What is JTAG - Joint Test Action Group (IEEE 1149)

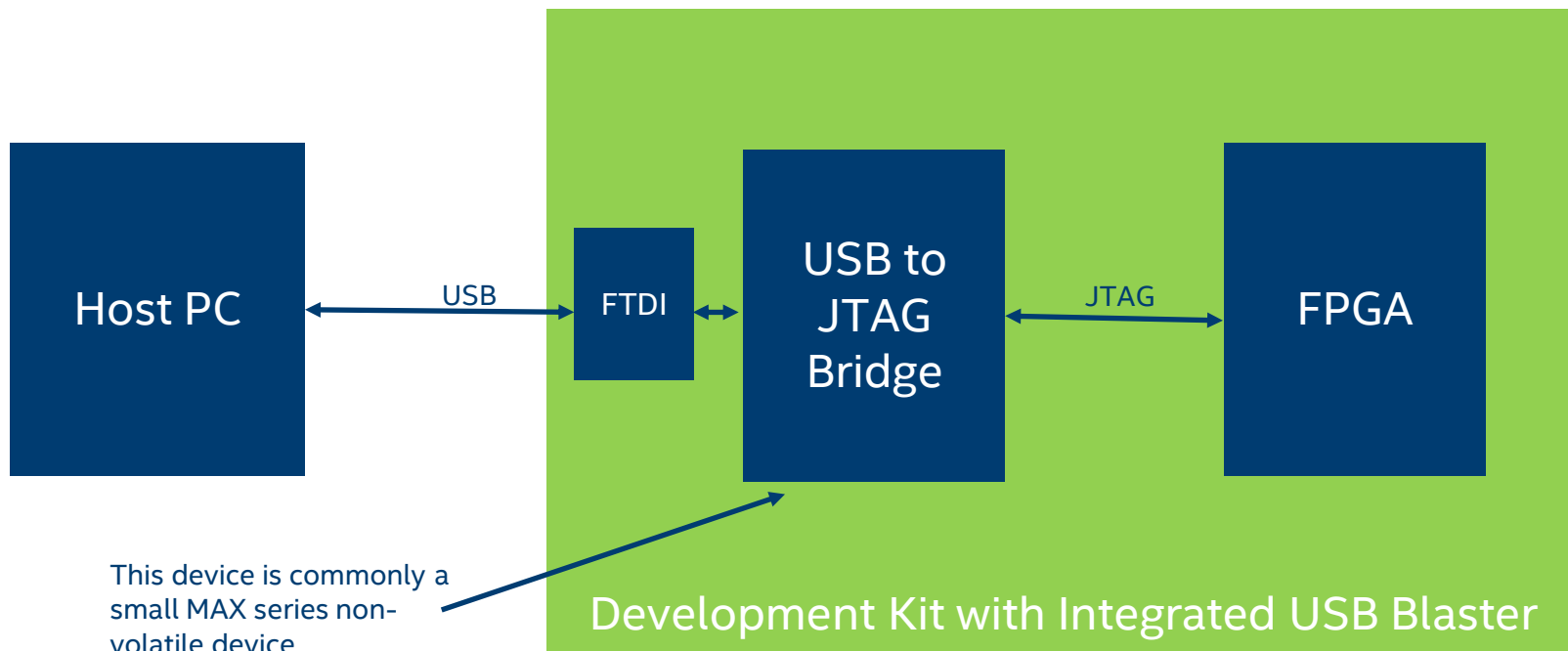
JTAG = JTAG is an industry standard for verifying designs and testing printed circuit boards after manufacture. JTAG implements standards for on-chip instrumentation in electronic design automation as a complementary tool to digital simulation.

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

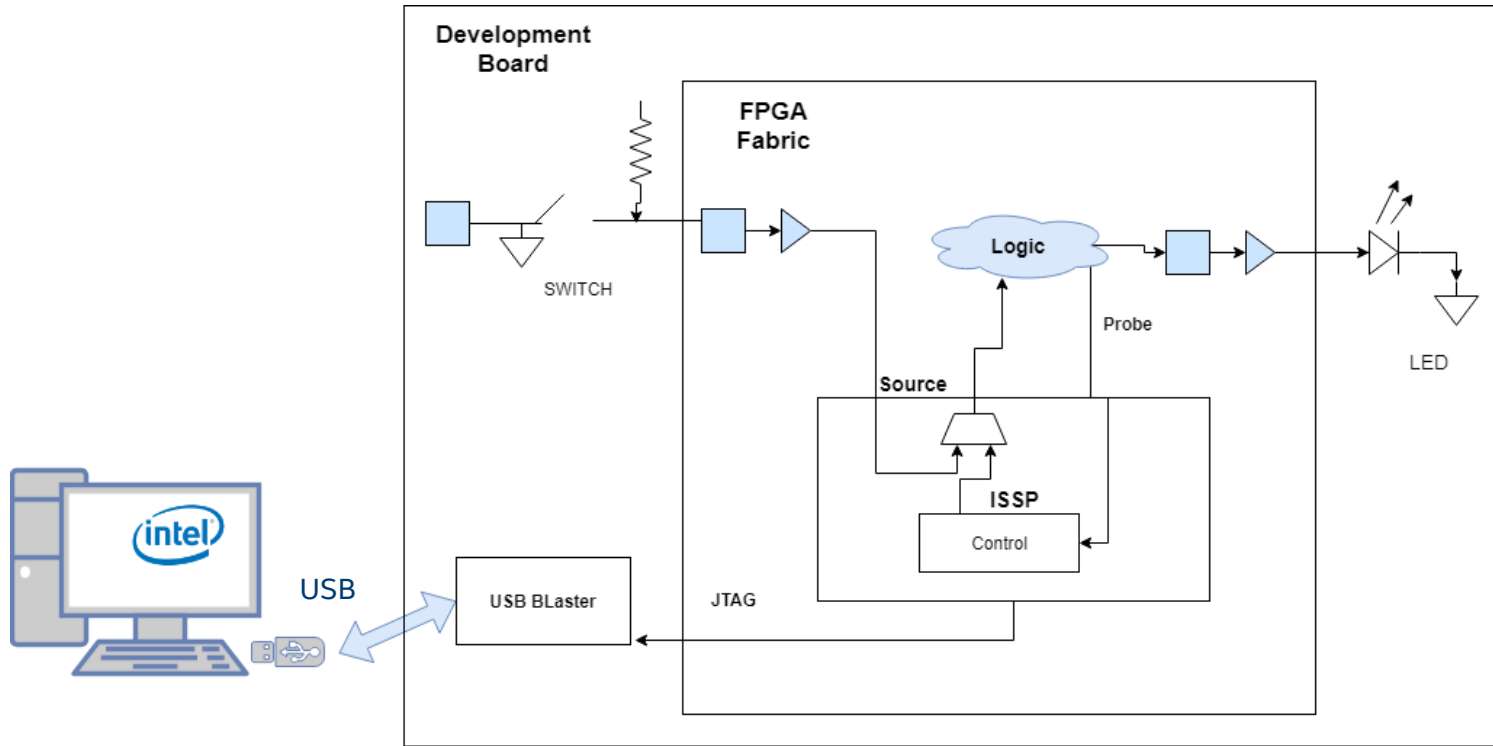
Standard 4 or 5 wire bus – used in many digital electronic devices for test and device specific configuration



USB Blaster Bridge Circuit



In-System Sources and Probes Block Diagram

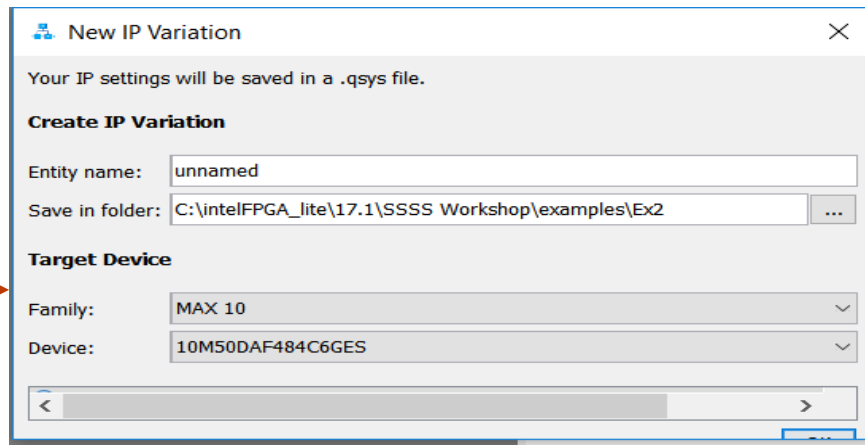
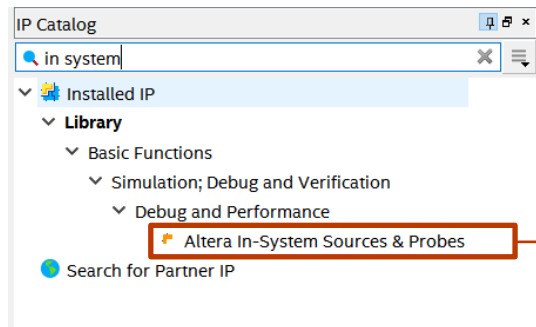


Example Uses

- 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

The screenshot displays the IP Parameter Editor for the Altera In-System Sources & Probes (ISSP) IP. The interface includes a menu bar (File, Edit, System, Generate, View, Tools, Help) and a toolbar. The main window is titled "Parameters" and shows the "System: unnamed" and "Path: in_system_sources_probes".

Annotations with arrows point to specific fields:

- Instance Index**: Points to the "Instance Index" field, which is set to 0.
- Instance ID**: Points to the "The 'Instance ID' of this instance (optional):" field, which is set to SOPR.
- # of Sources and Probes**: Points to the "Probe Port Width [0..512]:" field, which is set to 14.
- Synchronize writing of clks with sources**: Points to the "Use Source Clock" checkbox, which is unchecked.

The "Probe Parameters" section shows "Probe Port Width [0..512]:" set to 14. The "Source Parameters" section shows "Source Port Width [0..512]:" set to 1. The "Hexadecimal initial value for the Source Port:" field is set to 0.

The "Details" pane on the right shows a block diagram of the IP, labeled "in_system_sources_probes_0", with a "sources" block and a "probe[13..0]" block. The "Presets" pane at the bottom right shows "Presets for in_system_sources_probes_0" and a "Project" section with a "Library" section.

The "Messages" pane at the bottom left shows a table with columns "Type", "Path", and "Message". The status bar at the bottom indicates "0 Errors, 0 Warnings" and includes buttons for "Generate HDL..." and "Finish".

Using In-System Sources and Probes

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

In-System Sources and Probes Editor - C:/intelFPGA_lite/17.1/SSSS Workshop/examples/Ex2/top_counter - top_counter - [Spf2.spf]

File Edit View Processing Tools Window Help

Instance Manager: Ready to acquire

Probe read interval

Current interval: 9 samples per second

☒ Automatic

☐ Manual 1 s

Event log

Maximum size: 8

☒ Save data to event log

Write source data: Continuously

Instance Manager

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0] Setup...

Device: @1: 10M50DA[ES]/10M Scan Chain

File: kshop/examples/Ex2/top_counter.sof ...

JTAG Chain Configuration

Index	Instance ID	Status	Sources: 1	Probes: 14	Name
0	SOPR	Unloading ...	1	14	sources_probes:sources_probes_inst[altsource_probe_topin_system_sources_probes_0]altsource_probe:issp_impl

0 SOPR

Index	Type	Alias	Name	Data
P13			sources_probes:sources_pi	0
P12			sources_probes:sources_pi	0
P11			sources_probes:sources_pi	0
P10			sources_probes:sources_pi	0
P9			sources_probes:sources_pi	0
P8			sources_probes:sources_pi	1
P7			sources_probes:sources_pi	0
P6			sources_probes:sources_pi	0
P5			sources_probes:sources_pi	1
P4			sources_probes:sources_pi	0
P3			sources_probes:sources_pi	0
P2			sources_probes:sources_pi	1
P1			sources_probes:sources_pi	0
P0			sources_probes:sources_pi	0
S0			source[0]	1

Source

Log (Waveform Viewer)

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:

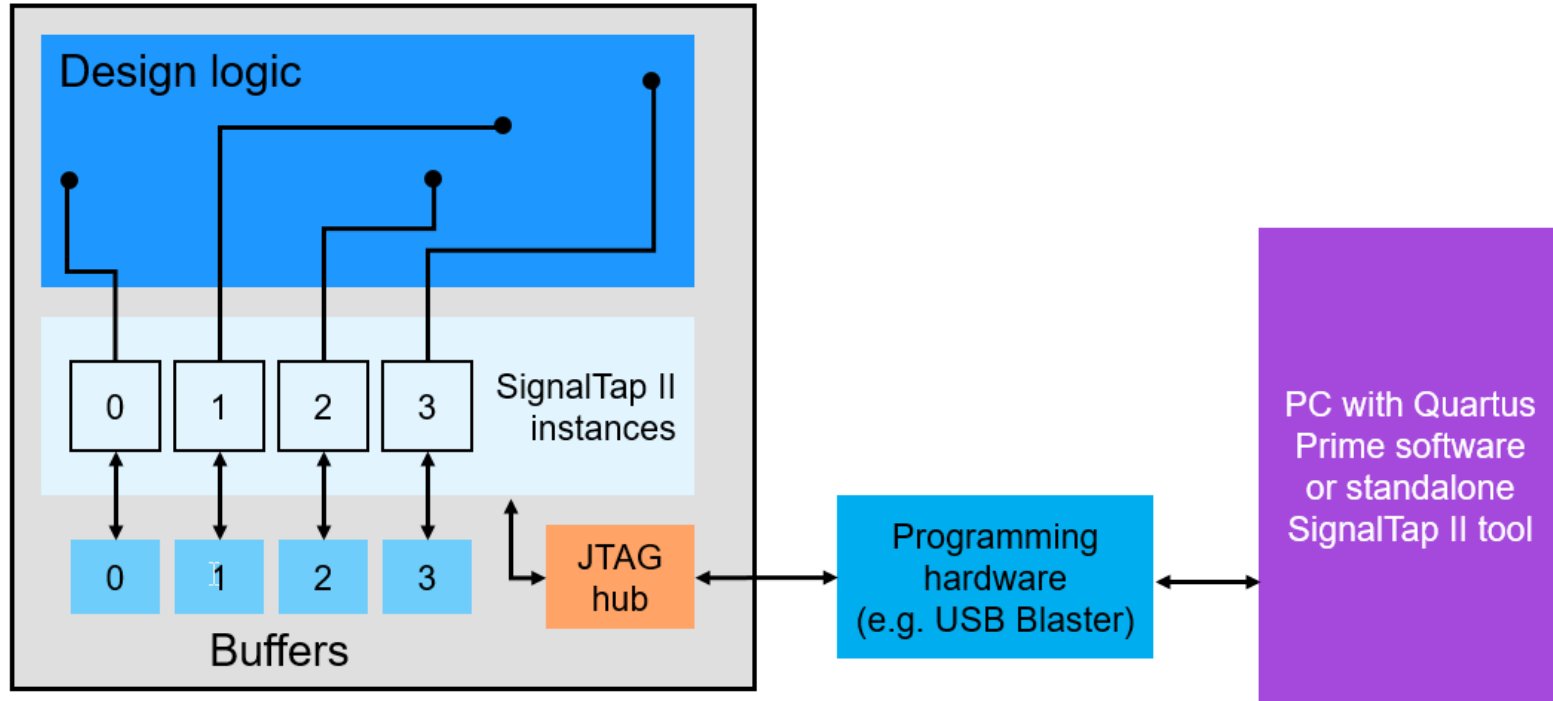
- Tap signals buried deep in the design
- No unassigned I/Os or routing needed
- Comes free with all versions of Quartus, no external test equipment required
- Tap new signals with the same board by reconfiguring, recompiling, and reprogramming (no re-spin!)

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 ?

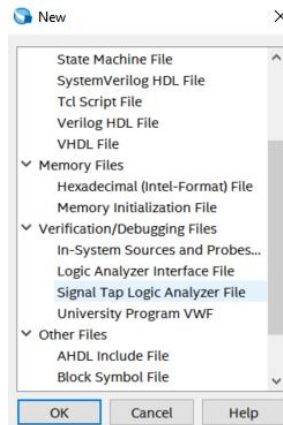
Altera FPGA device



Create a Signal Tap instance in two ways

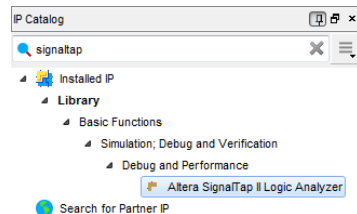
1. Use Signal Tap file (.stp) *(recommended)*

- 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)

The screenshot shows the SignalTap II Logic Analyzer interface. The Instance Manager window is highlighted with a red box, showing a table of instances. The Signal Configuration window is also visible, showing settings for the clock and data. The Hierarchy Display window shows the project structure.

Instance Manager: Invalid JTAG configuration

Instance	Status	Enabled	LEs: 584	Memory: 7168	Small: 0/415480	Med
instance_one	Not running	<input checked="" type="checkbox"/>	584 cells	7168 bits	0 blocks	1 bl

Signal Configuration:

Clock: clk

Data

Sample depth: 1 K RAM type: Auto

Segmented: 2 512 sample segments

Nodes Allocated: Auto Manual: 7

Pipeline Factor: 0

Storage qualifier:

Type: Continuous

Hierarchy Display:

- top_counter
 - u1

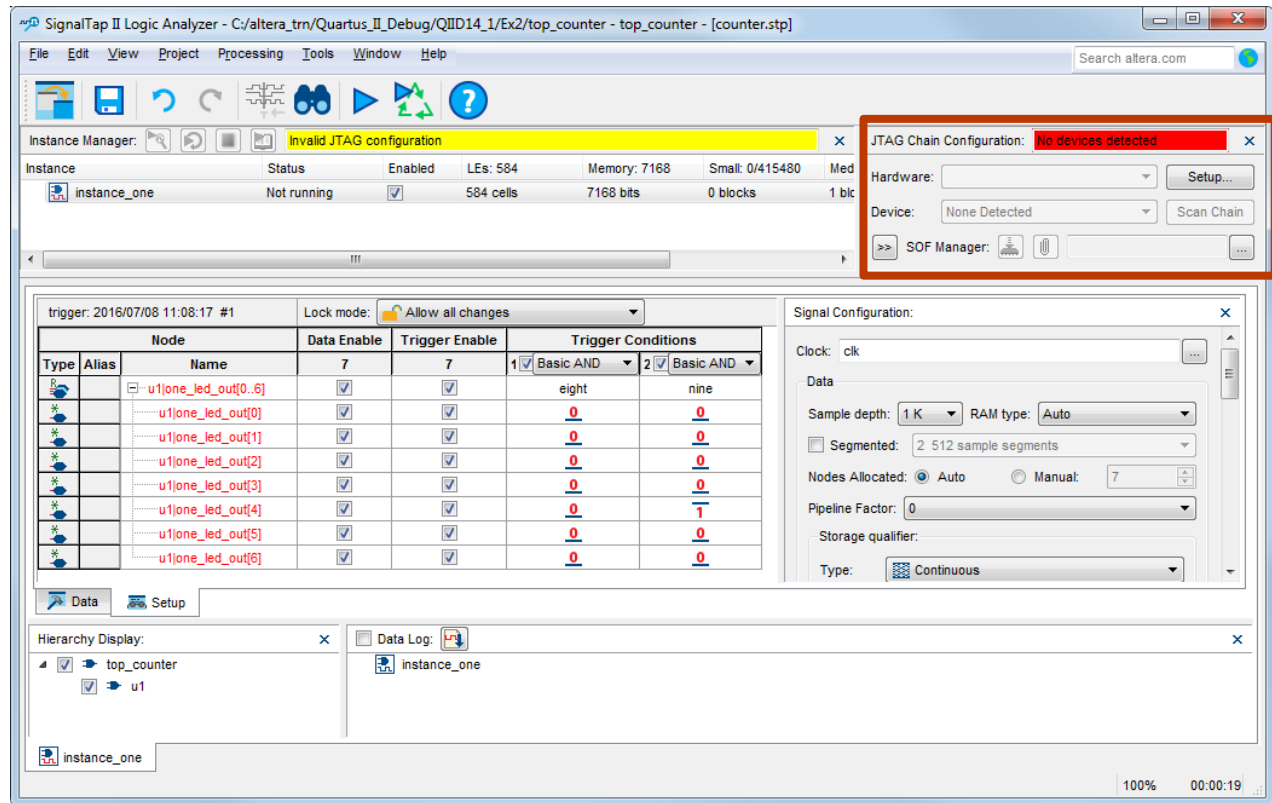
Data Log:

- instance_one

Signal Tap Logic Analyzer Window

JTAG Chain Configuration

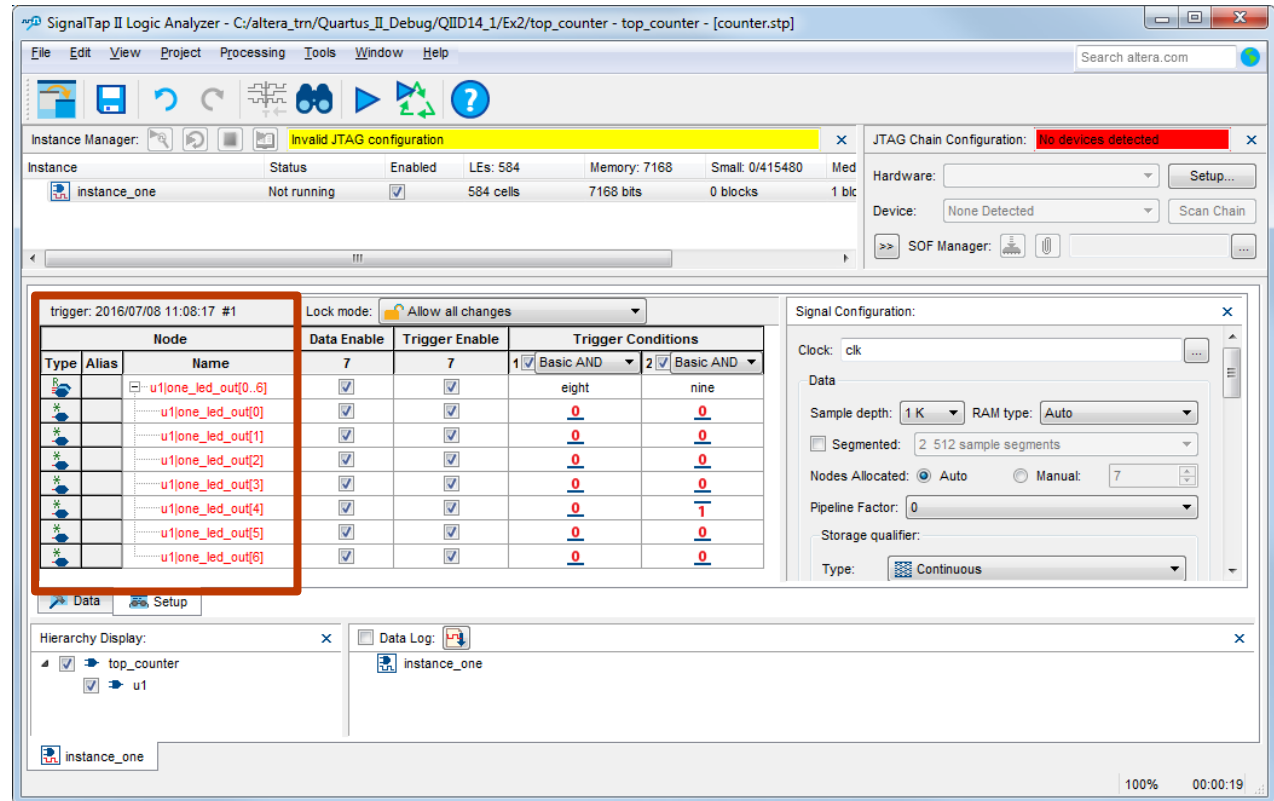
- Built in “Programmer”
- Scans the JTAG chain and identifies available devices



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)

The screenshot shows the SignalTap II Logic Analyzer interface. The main window displays the configuration for instance_one, which is currently not running. A yellow banner at the top indicates an "Invalid JTAG configuration". The "Instance Manager" table shows instance_one with 584 LEs and 7168 bits of memory. The "Signal Configuration" panel on the right shows the clock set to "clk" and the sample depth set to 1 K. The "Trigger Configuration" panel is highlighted with a red box, showing the trigger conditions for instance_one. The trigger is set to "Basic AND" and "Basic AND" with conditions "eight" and "nine". The "Data Enable" and "Trigger Enable" columns are both checked for all signals.

Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
		u1one_led_out[0..6]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	eight nine
		u1one_led_out[0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1one_led_out[1]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1one_led_out[2]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1one_led_out[3]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1one_led_out[4]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 1
		u1one_led_out[5]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1one_led_out[6]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0

Signal Tap Logic Analyzer Window

Trigger Conditions

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

The screenshot shows the SignalTap II Logic Analyzer interface. The main window displays a table of nodes and their trigger conditions. A red box highlights the 'Trigger Conditions' dropdown menu, which is open, showing options: Basic AND, Basic OR, Comparison, and Advanced. The table below shows the configuration for the 'u1|one_led_out[0..6]' node.

Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
		u1 one_led_out[0..6]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	1 <input checked="" type="checkbox"/> Basic AND 2 <input checked="" type="checkbox"/> Basic AND
		u1 one_led_out[0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	eight 0
		u1 one_led_out[1]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
		u1 one_led_out[2]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
		u1 one_led_out[3]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
		u1 one_led_out[4]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
		u1 one_led_out[5]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
		u1 one_led_out[6]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0

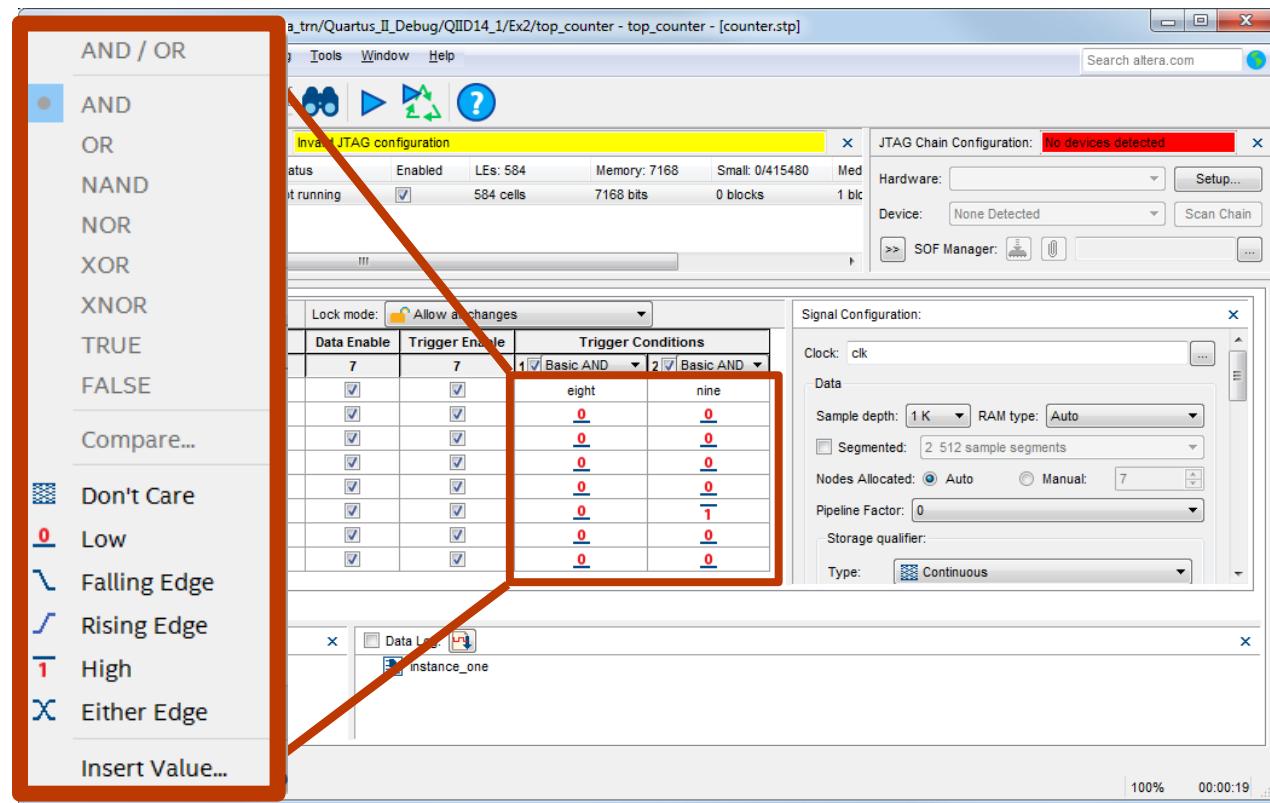
The 'Trigger Conditions' dropdown menu is open, showing the following options:

- Basic AND
- Basic OR
- Comparison
- Advanced

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

The screenshot shows the SignalTap II Logic Analyzer interface. The main window displays a table of signals and their configurations. A dialog box titled "Signal Configuration" is open, showing settings for the clock and data capture.

Signal Configuration Dialog:

- Clock: clk
- Sample depth: 1 K
- RAM type: Auto
- Segmented: 2 512 sample segments
- Nodes Allocated: Auto
- Pipeline Factor: 0
- Storage qualifier: Continuous

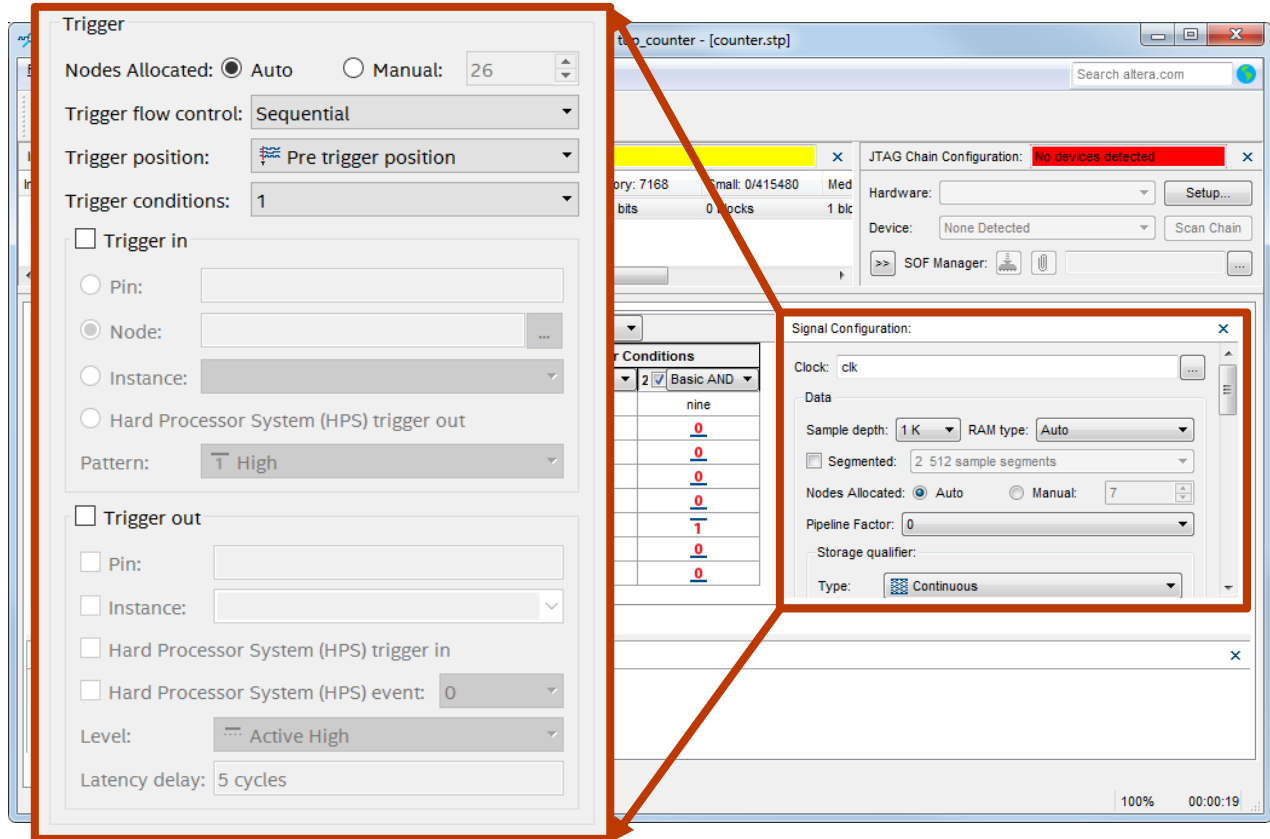
Main Window Table:

Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
		u1[one_led_out[0..6]	7	7	1 Basic AND 2 Basic AND
		u1[one_led_out[0]	7	7	0 0
		u1[one_led_out[1]	7	7	0 0
		u1[one_led_out[2]	7	7	0 0
		u1[one_led_out[3]	7	7	0 0
		u1[one_led_out[4]	7	7	0 1
		u1[one_led_out[5]	7	7	0 0
		u1[one_led_out[6]	7	7	0 0

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)

The screenshot shows the SignalTap II Logic Analyzer interface. The main window displays a table of nodes and their configuration. The 'Data' tab is selected, showing a list of nodes and their values. The 'Setup' tab is also visible, showing configuration options for the nodes and trigger conditions.

Instance Manager:

Instance	Status	Enabled	LEs: 584	Memory: 7168	Small: 0/415480	Med
instance_one	Not running	<input checked="" type="checkbox"/>	584 cells	7168 bits	0 blocks	1 bk

JTAG Chain Configuration: No devices detected

Hardware: Setup...

Device: None Detected Scan Chain

SOF Manager:

trigger: 2016/07/08 11:08:17 #1 Lock mode: Allow all changes

Type	Alias	Name	Data Enable	Trigger Enable	Trigger Conditions
		u1[one_led_out[0..6]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	eight nine
		u1[one_led_out[0]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1[one_led_out[1]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1[one_led_out[2]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1[one_led_out[3]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1[one_led_out[4]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 1
		u1[one_led_out[5]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0
		u1[one_led_out[6]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0 0

Signal Configuration:

Clock: clk

Data

Sample depth: 1 K RAM type: Auto

Segmented: 2 512 sample segments

Nodes Allocated: Auto Manual: 7

Pipeline Factor: 0

Storage qualifier:

Type: Continuous

Instance_one

LAB EXERCISE 2: IN-SYSTEM SOURCES AND PROBES

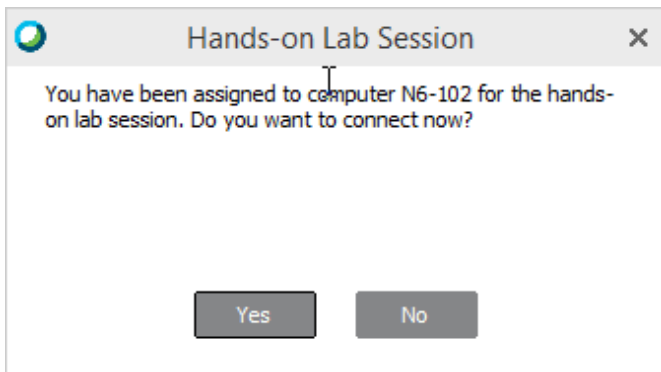
LAB EXERCISE 3: SIGNAL TAP

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**

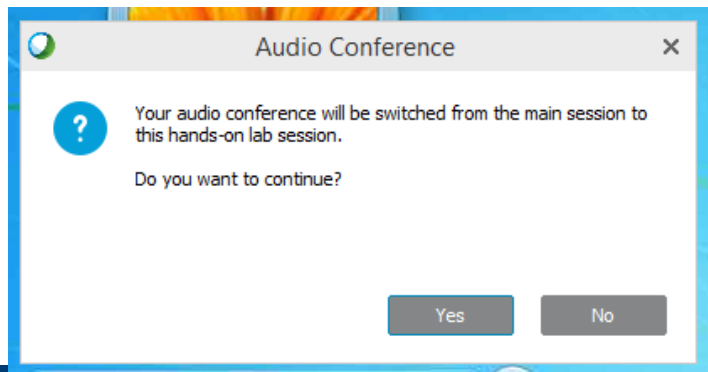


Lab Notes (2)

Instructor is able to view your screen if necessary

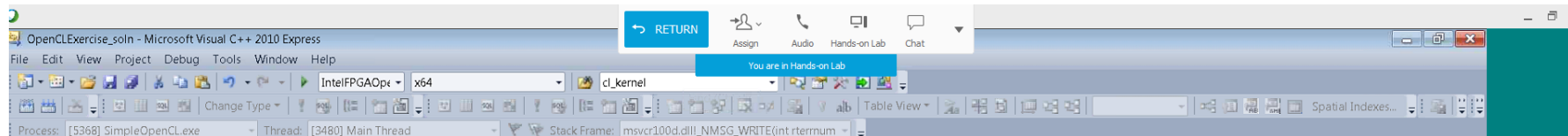
If you get stuck and need help, exit to the main conference room and type in the chat that you need help. If you type in chat window while connected to your PC, the instructor cannot see that you need assistance.

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



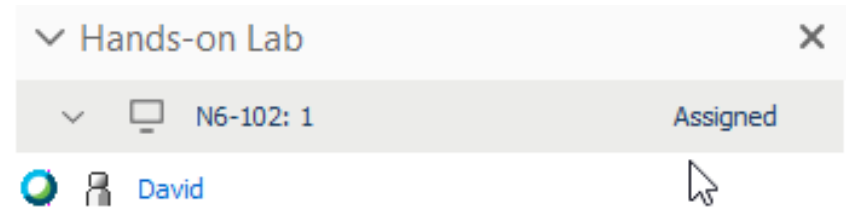
After clicking return, you will get a pop-up asking if you want to disconnect

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**



Disconnect

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

