Quartus Software Debug Tools

Class Agenda

SignalProbe Incremental Routing

In-System Memory Content Editor (ISMCE)

In-System Sources & Probes

SignalTap II embedded logic analyzer

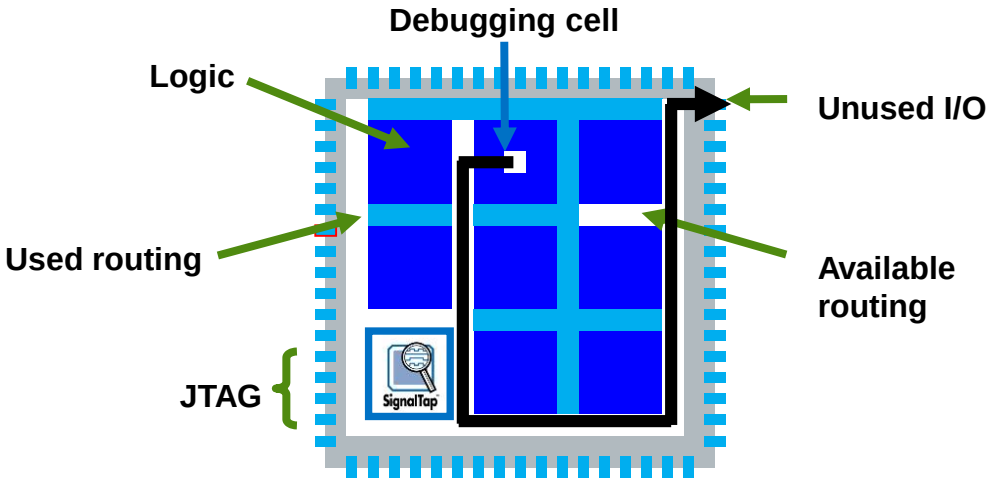
Logic Analyzer Interface (LAI)

SignalProbe Tool

SignalProbe Incremental Routing

Fast incremental routing of debugging signals to spare/reserved I/O pins

Uses any available routing without full recompilation



When to Use SignalProbe

- Additional I/O available to use as debug ports
- External equipment available
- Little or no additional internal device resources required
- Avoid possibly long compile times
- Quickly route internal signals to debug ports
- No JTAG connection required

SignalProbe Advantages

- Simple to use
  - Fitter handles signal routing using incremental routing
    - Fast compilation time
  - User only specifies source node & destination pin
- User can test output of any hard node
- Compiler reports delay times from node to pin
- Placement of compiled design remains unaffected
  - Timing of signal being debugged remain mostly unchanged

Steps to Use SignalProbe Routing

1. Reserve SignalProbe outputs in Pin Planner
2. Assign and configure SignalProbe source nodes
   - Requires prior full compilation (post-fit netlist)
3. Perform SignalProbe incremental compilation
   - Start Check & Save All Netlist Changes from SignalProbe dialog box

     or
   - Processing menu → Start → Start SignalProbe Compilation
4. Program Device

More Information

- Quartus Prime Standard/Pro Edition Handbook
  - “Quick Design Debugging Using SignalProbe” Chapter (Volume 3)

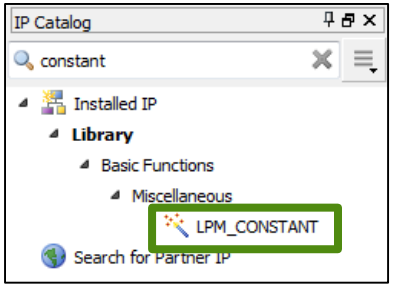
In-System Memory Content Editor

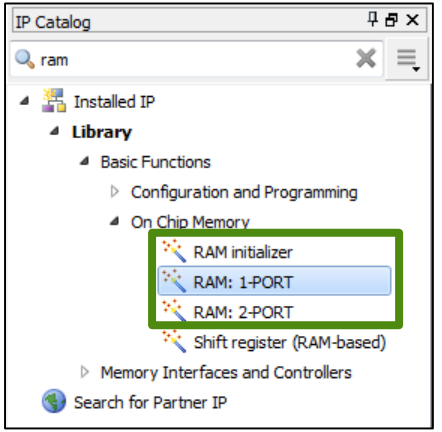
In-System Memory Content Editor

- Allows
  - Viewing contents of memories & constants via JTAG
  - Updating contents of memories & constants via JTAG
- Example uses
  - Correct stored image/video data
  - Try different filter coefficients to change filter characteristics
  - View/correct parity bits
  - Insert data errors to check reliability

When to Use ISMCE

- Embedded memory blocks storing data
  - Verify memory contents during device operation
  - Update memory contents during device operation
- JTAG connection required
- No external equipment available
- Supported IP
  - Constants (**LPM_CONSTANT**)
  - Single-port ROM (**ROM: 1-PORT**)
  - Single-port RAM (**RAM: 1-PORT**)





Steps To Using the ISMCE

1. Enable in-system content editing for each memory & constant
2. Perform a full compilation & program device
3. Launch In-System Memory Content Editor
4. Perform reads/writes from/to in-system memories & constants

In-System Editor Options

- Import memory data file to update memories or constant
  - **.hex** & **.mif**
- Export data displayed in hex editor
  - **.rif**, **.hex**, **.mif**, **.vcd**
- Open multiple editors to access multiple devices in JTAG chain
- Tcl scripting support

More Information

- Quartus Prime Standard/Pro Edition Handbook
  - “In-System Modification of Memories & Constants” chapter (Volume 3)

In-System Sources & Probes

In-System Sources & Probes (ISSP)

Features Overview

- Operates over JTAG
- Each ISSP instance can view (probe) up to 512 signals during run-time
- Each instance can drive and toggle (source) the values of up to 512 signals
- Create up to 128 instances of the ISSP IP using IP Catalog
- Graphical interface makes it easy to manage large numbers of sources and probes across multiple instances

When to Use Sources & Probes

- Simple functional debug needed
- Logic resources available for ISSP IP instance(s)
- JTAG connection required
- No external equipment available or needed

Example Uses

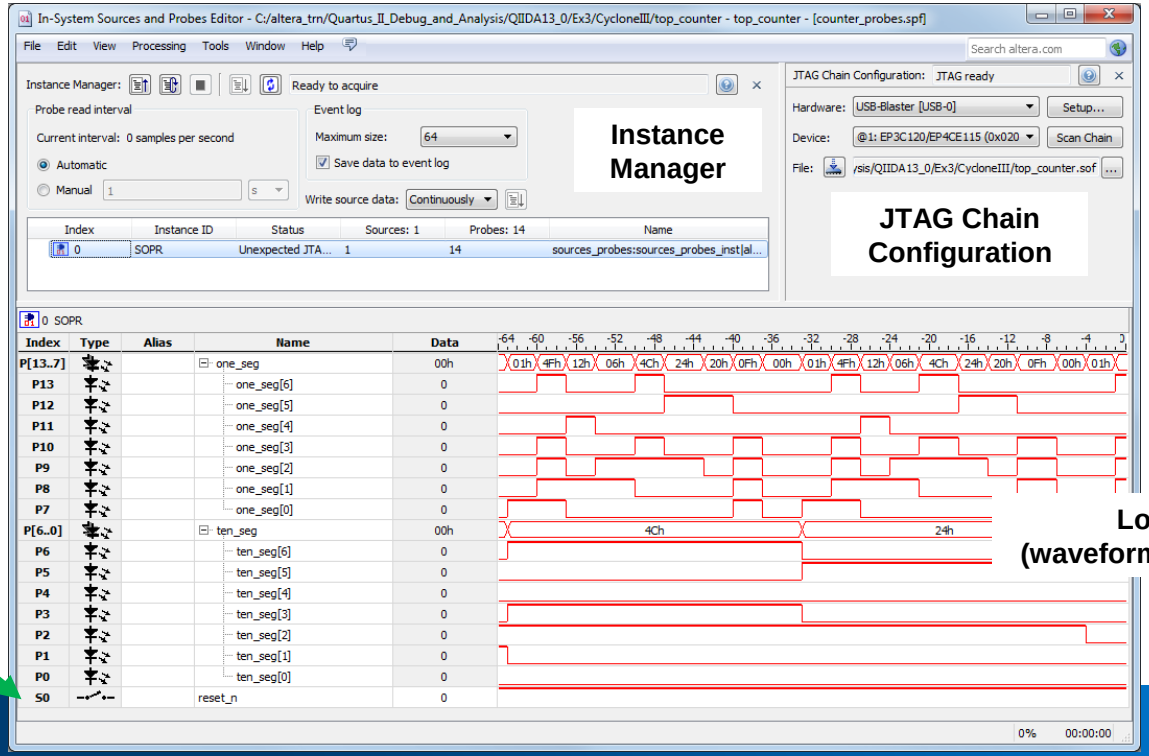
- Virtual push button control of signals in design
- Monitor results of changing design constants
- Extensive Tcl scripting support to create custom automated design control interfaces
- Force trigger conditions for debugging design with the SignalTap II Embedded Logic Analyzer *(described later)*

Using In-System Sources & Probes

1. Create In-System Sources & Probes IP instance(s) using the IP Parameter Editor or MegaWizard Plug-In Manager (device-dependent) (through the IP Catalog)
2. Instantiate in design and compile project
3. Program target device(s)
4. Create and use In-System Sources & Probes Editor (.spf file) to control sources and probes

In-System Sources & Probes Editor (.spf)

Quartus Tools menu → In-System Sources & Probes Editor



More Information

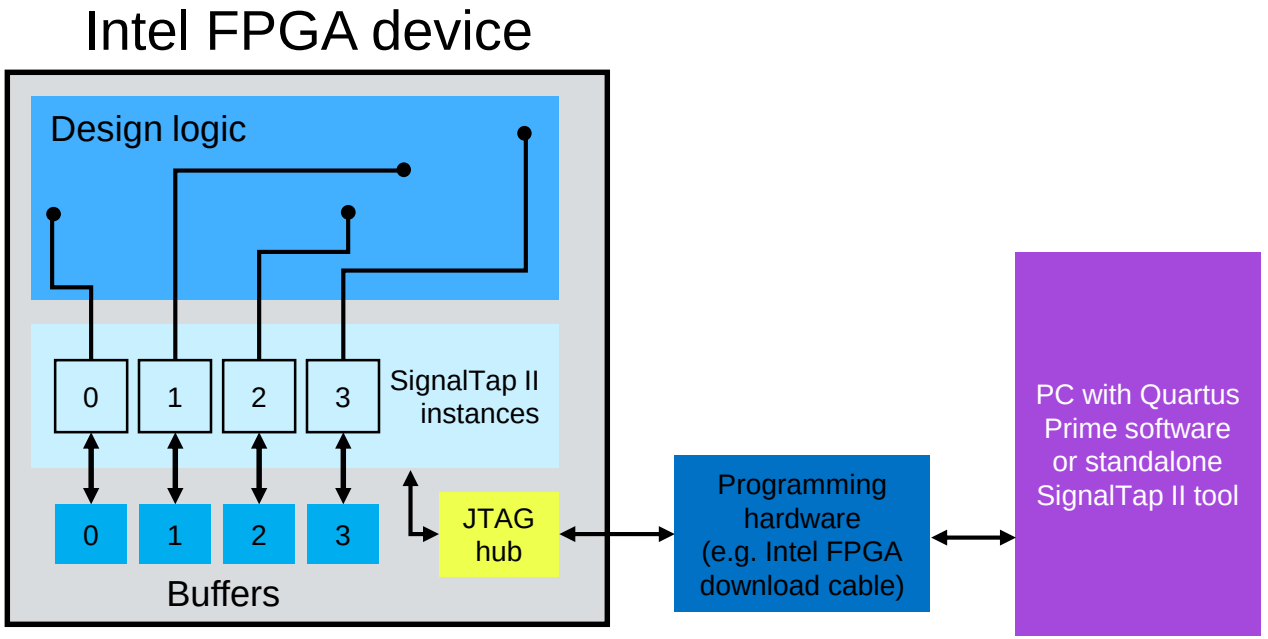
- Quartus Prime Standard/Pro Edition Handbook
  - “Design Debugging Using In-System Sources and Probes” Chapter (Volume 3)

SignalTap II Logic Analyzer

SignalTap II Embedded Logic Analyzer

- Captures the logic state of FPGA internal signals using a defined clock signal
- Monitor buried signals in real-time
- Connects to the Quartus® Prime software through FPGA JTAG connection
- Available for free from the Intel FPGA Download Center
  - *https://www.altera.com/downloads/download-center.html*
  - Installed with the Quartus Prime software (all editions)
  - Installed with the stand-alone programmer
    - At Download Center, select **Programming Software** from **Select by Software** tab

What is the SignalTap II ELA?



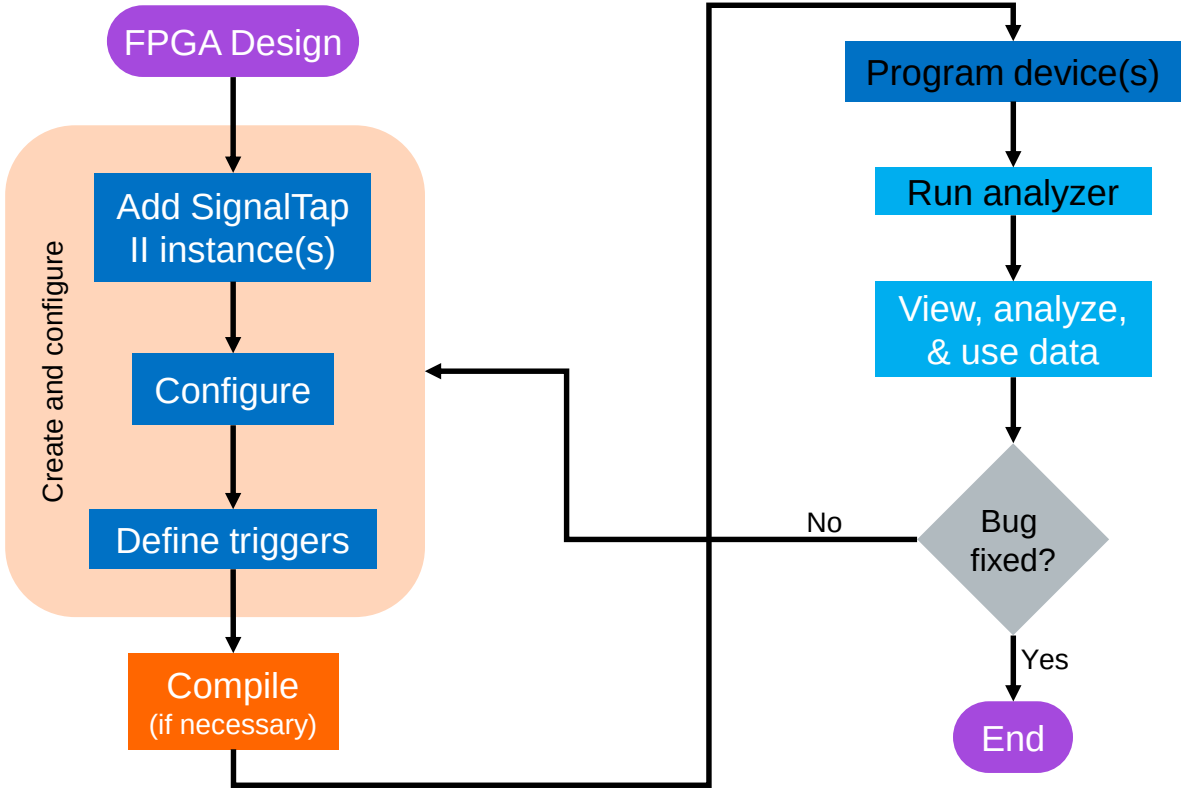
When to Use the SignalTap II ELA

- No external equipment available
- Additional device resources available
  - Logic blocks
  - Memory blocks
- JTAG connection available
  - Local or remote over network
- Only functional debug capabilities needed

SignalTap II Resource Utilization

- Logic elements
  - Number of channels
  - Number and complexity of trigger conditions
- Memory blocks
  - Number of channels
  - Sample depth
    - Selectable trade-off between depth & number of channels
  - 128K sample depth with 2048 channels not practical = 13107 M20K blocks
    - Largest Stratix® 10 device has ≈11000 M20K blocks
- Estimated resource usage displayed and update during SignalTap II configuration

Typical SignalTap II Debugging Flow



Adding SignalTap II ELA to a Design

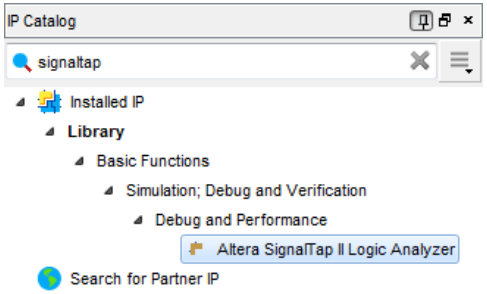
Use SignalTap II file (**.stp**) *(recommended)*

- Use Quartus Prime GUI
- Configure **.stp** file details manually
- Easily enable or disable in project settings

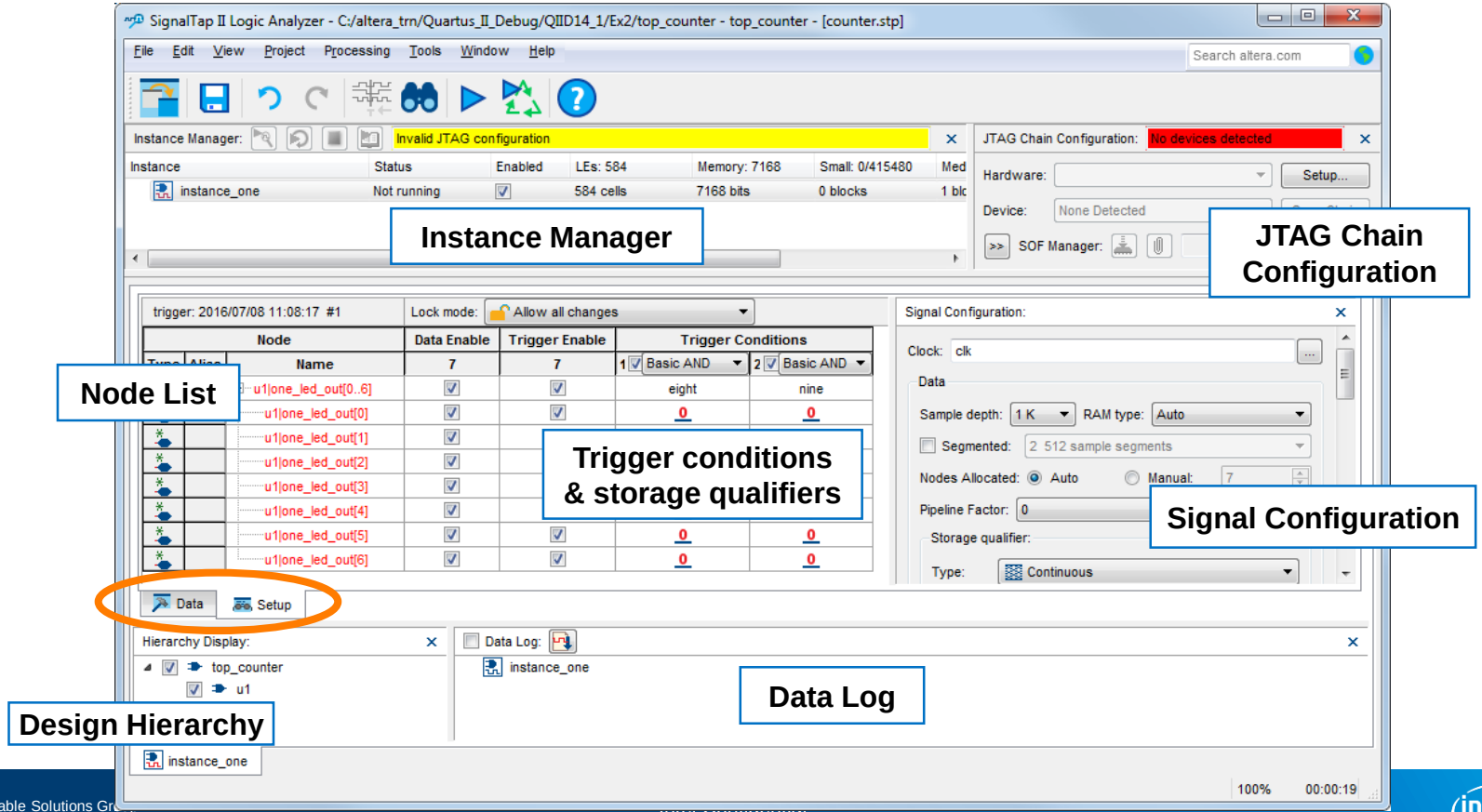
or

Use **IP Catalog** and **IP Parameter Editor**

- Manually instantiate **altera_signaltap_ii_logic_analyzer** IP core into HDL code or in Qsys
- ELA tied directly to signals in RTL
- Control through command line or related **.stp** file
- See Quartus Prime Handbook (Standard or Pro) for more details

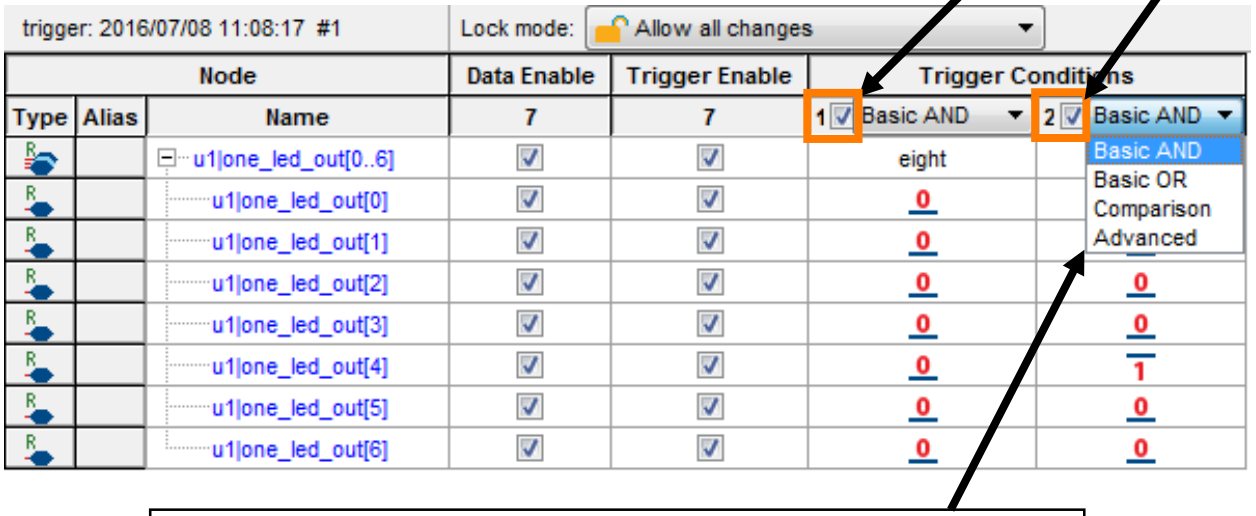


SignalTap II Logic Analyzer Window



Trigger Types and Options

Disable individual conditions to skip (sequential flow only; described later)



Basic AND: *all* signal levels/transitions must be true

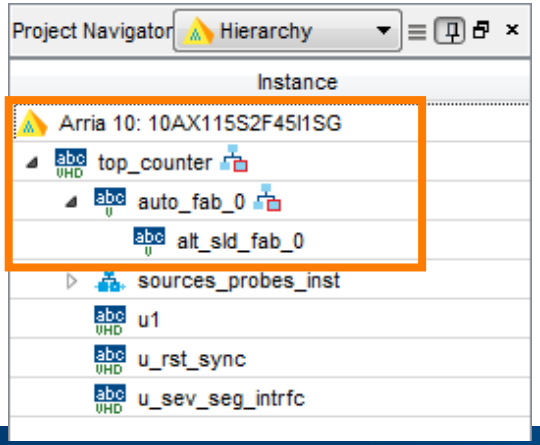
Basic OR: *any* levels/transitions true; bits in groups are OR reduced unless condition is a numerical value

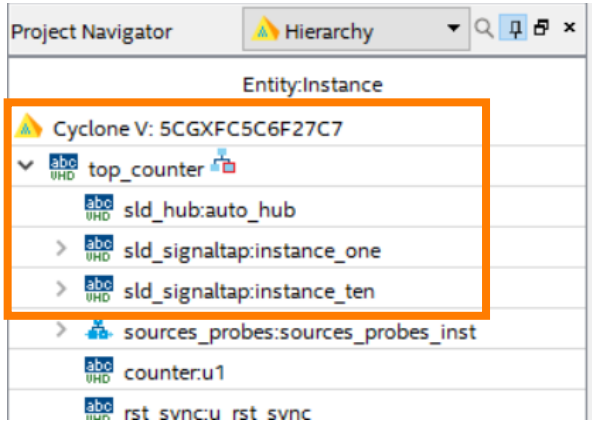
Comparison: extension of **Basic OR**; compare groups to expected integer value(s)

Advanced *(discussed later)*

Compile Project with SignalTap II ELA

- Full compilation (**Processing** menu → **Start Compilation**)
- Logic added to project to implement logic analyzer and JTAG hub connection
  - **auto_fab** with Arria 10 devices in Pro edition
  - **sld_signaltap** and **sld_hub** for other devices and Quartus versions





More Information

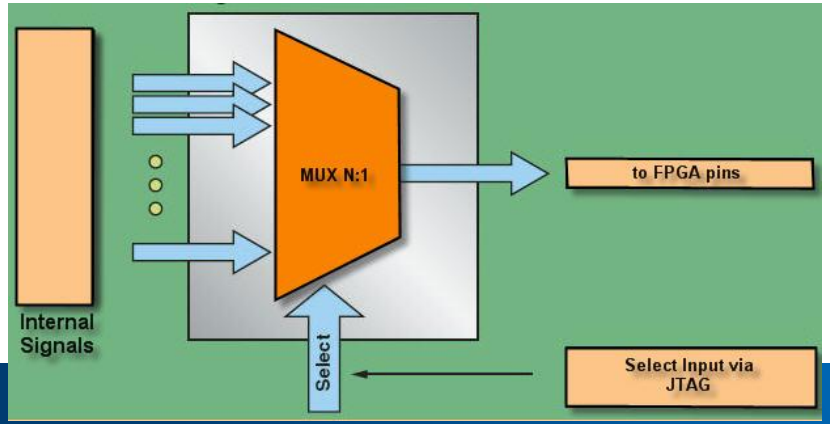
- Quartus Prime Standard/Pro Edition Handbook
  - “Design Debugging with the SIgnalTap II Logic Analyzer” Chapter (Volume 3)
- “SignalTap II Logic Analyzer” online trainings
  - *https://www.altera.com/support/training/course.html?courseCode=ODSW1164*

Logic Analyzer Interface

What Is the Logic Analyzer Interface?

Enables external visibility into internal FPGA signals

- Internal signals → I/O pins → external test equipment
  - Map up to 256 internal signals to each pin
  - Quickly switch internal probe points using Quartus Logic Analyzer Interface
    - Without recompilation or reconfiguration

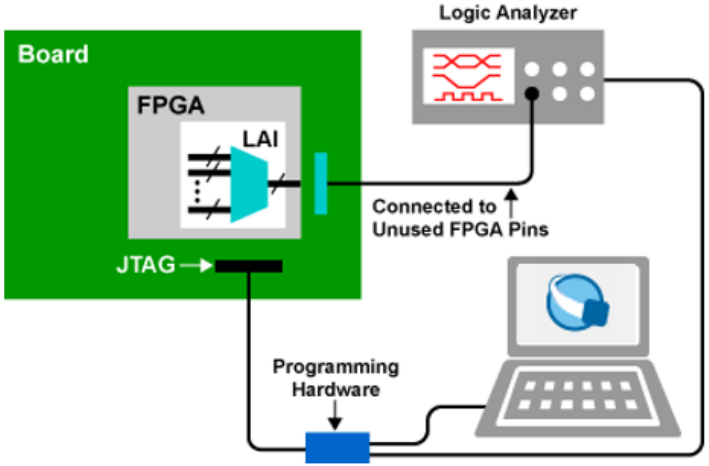


When to Use LAI

- External equipment available
- Large number of debug nodes but few remaining unused I/O
- Large sample depth required
- Limited device resources available
- JTAG connection available
- Performing functional or timing debug

Required Components

- Quartus software
- External logic analyzer
- Intel FPGA download cable
- Cable to connect FPGA to external logic analyzer (probes)

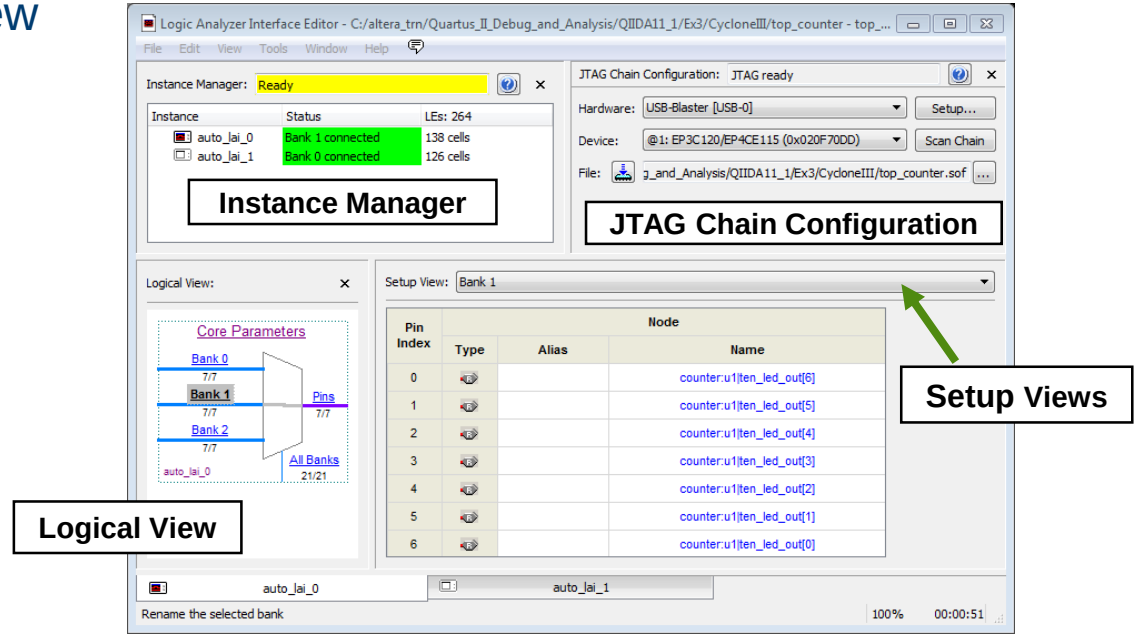


Using Logic Analyzer Interface File

1. Create Logic Analyzer Interface (.lai) file
   - Select number of banks
   - Assign signals to LAI file
   - Map LAI outputs to device I/O
2. Enable LAI file & compile
3. Program device
4. Select banks for debugging

Create LAI File

- Logic Analyzer Interface Editor (Tools menu)
- File menu → New



More Information

- Quartus Prime Standard/Pro Edition Handbook
  - “In-System Debugging Using External Logic Analyzers” Chapter (Volume 3)

Course Summary

Quartus software provides tools to aid and reduce the time spent on verification during the creation of your Intel FPGA design

- Debugging Tools
  - SignalProbe
  - In-System Memory Contents Editor
  - In-System Sources & Probes
  - SignalTap II Logic Analyzer
  - Logic Analyzer Interface


intel®
experience
what's inside™