Peter the Great St. Petersburg Polytechnic University

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[Institute of Computer Science and Technology](http://english.spbstu.ru/structure/institut_computernikh_nauk_i_tekhnologiy/)

Department of Computer Systems & Software Engineering

Lecture

*SignalTap II Logic Analyzer: Introduction & Getting Started*

student:

Ivanov Il’ya

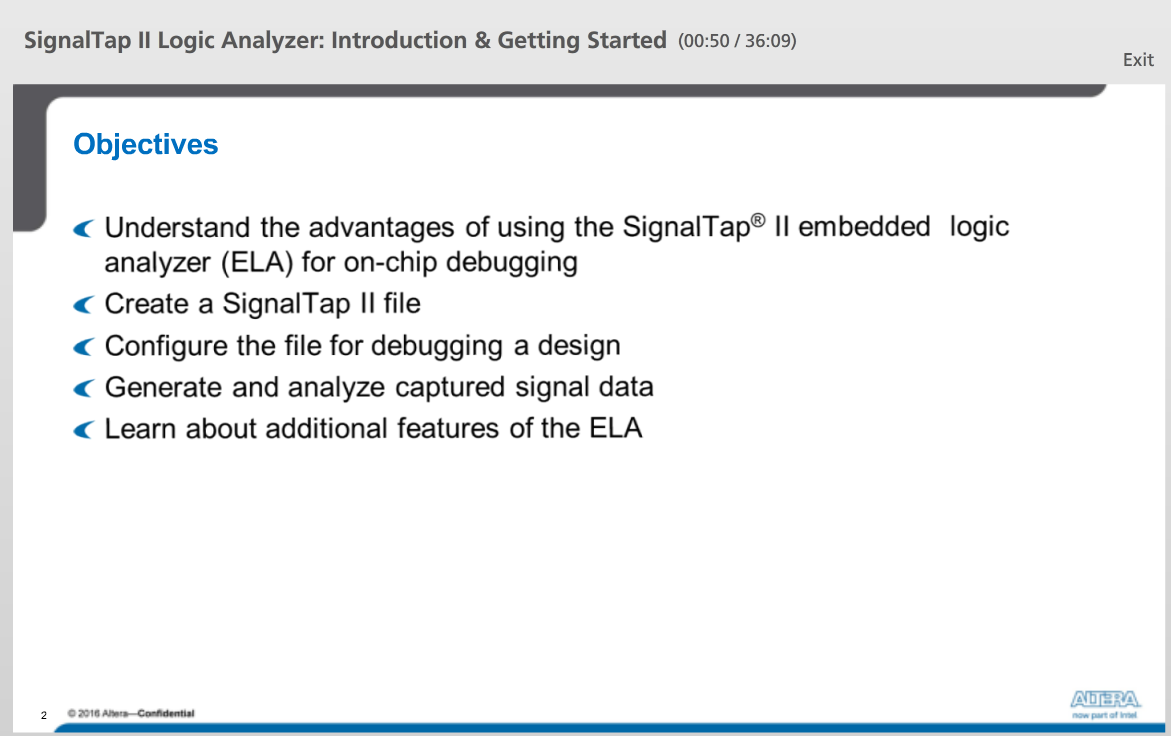
group: 3530901/70203

lecturer:

Antonov A.P.

Saint-Petersburg

2020



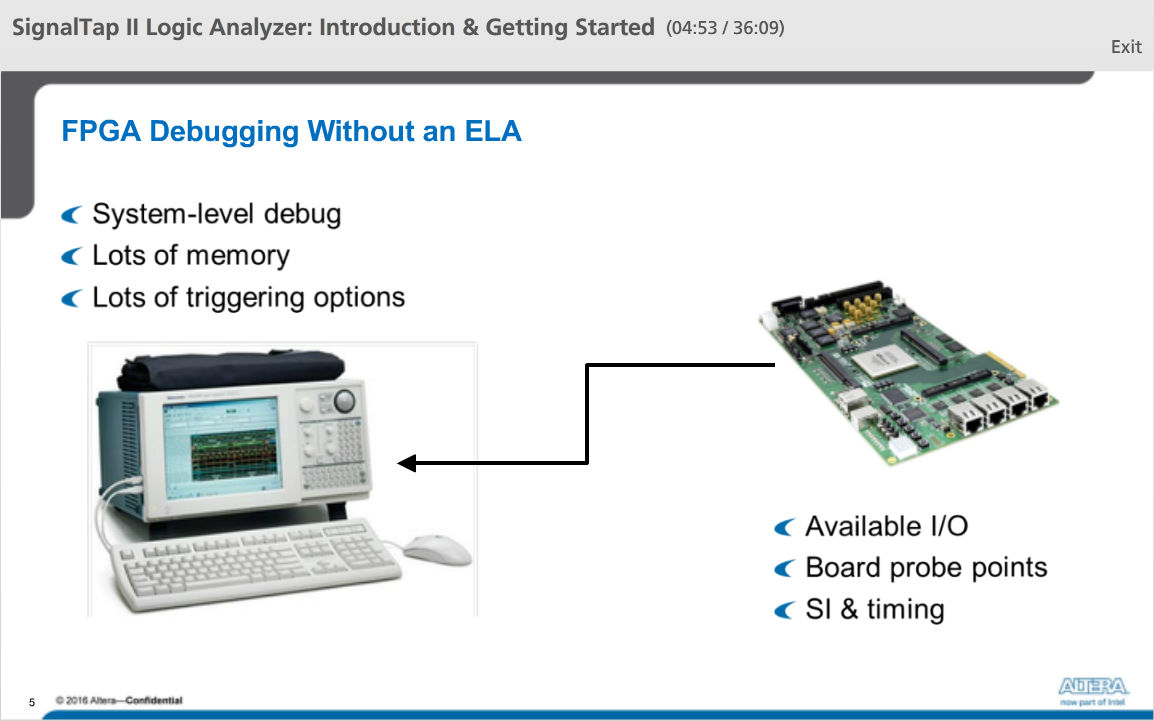
After this lecture, we will:

1) Understand the advantages of using SignalTap II for debugging FPGA design

2) See how easy it is to add the logic analyzer to our design and configure it

3) Be able to take the signal data captured by the logic analyzer and use it our purposes

4) Learn about some of the additional features of the ELA



How we can debug the functionality of an FPGA design in-system?

We can use the board that includes the FPGA device and some test equipment. To monitor internal signals, the FPGA must have available I/O pins and physical routing channels for routing out the signals we want to monitor. After programming the device, we can attach probes from the test equipment to probe points on the board.

Advantages:

1) Debugging at a system level

2) External test equipment typically has large amounts of buffer memory for storing large quantities of data samples

3) Test devices have many options for specifying what data to capture and when to capture it

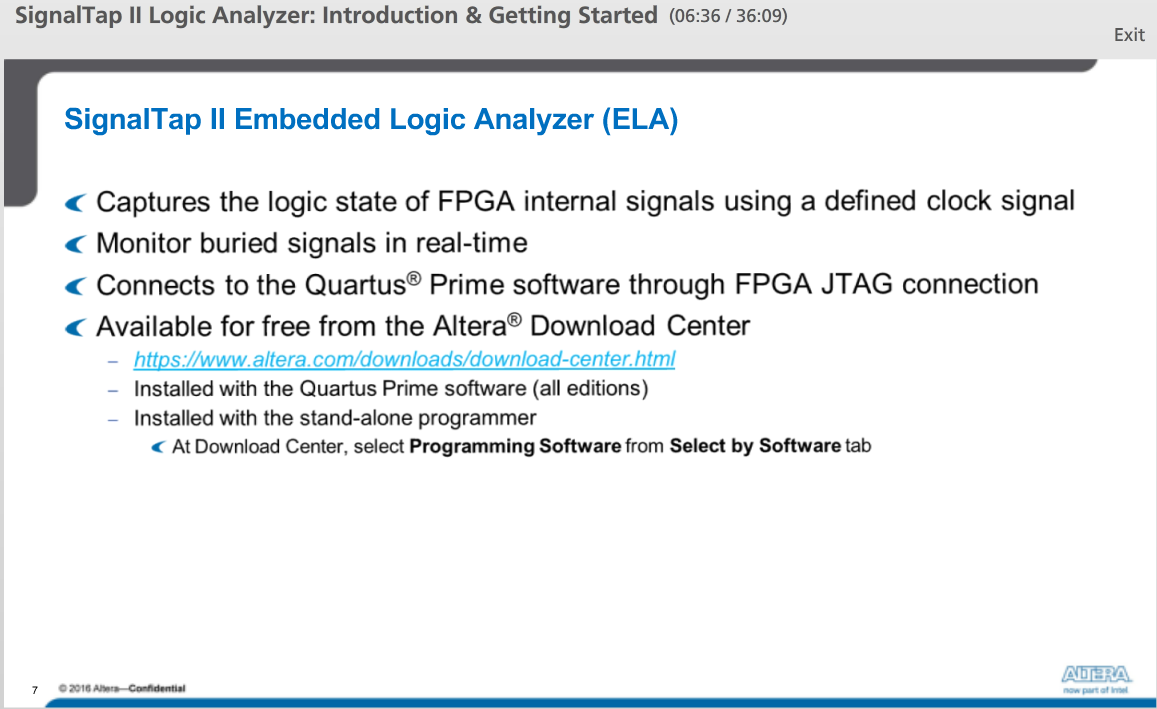
Disadvantages:

1) Any signals we want to monitor must be physically accessible on the board

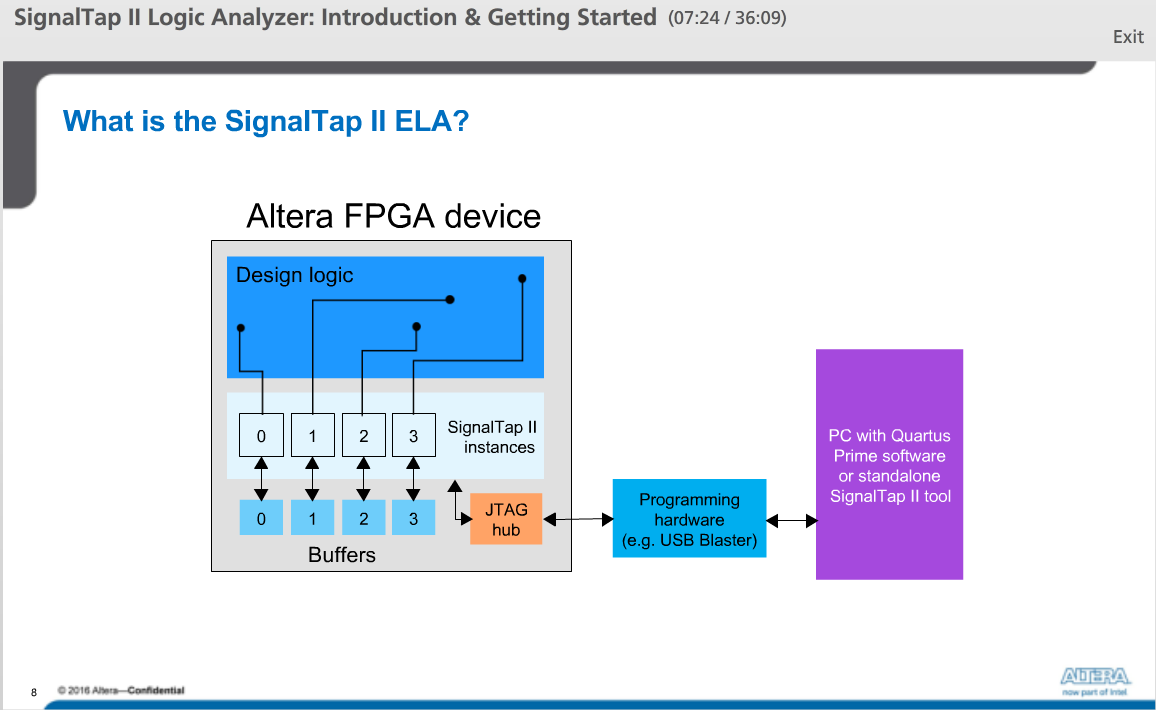
2) There is always the possibility that a device probe or other board component could affect signal integrity or timing on critical signals



The ELA takes the brain and features of a typical external logic analyzer and incorporates them into our FPGA design programmed into the target device on board. It is an embedded logic analyzer build using available FPGA resources, no I/O, board routing or probe points.

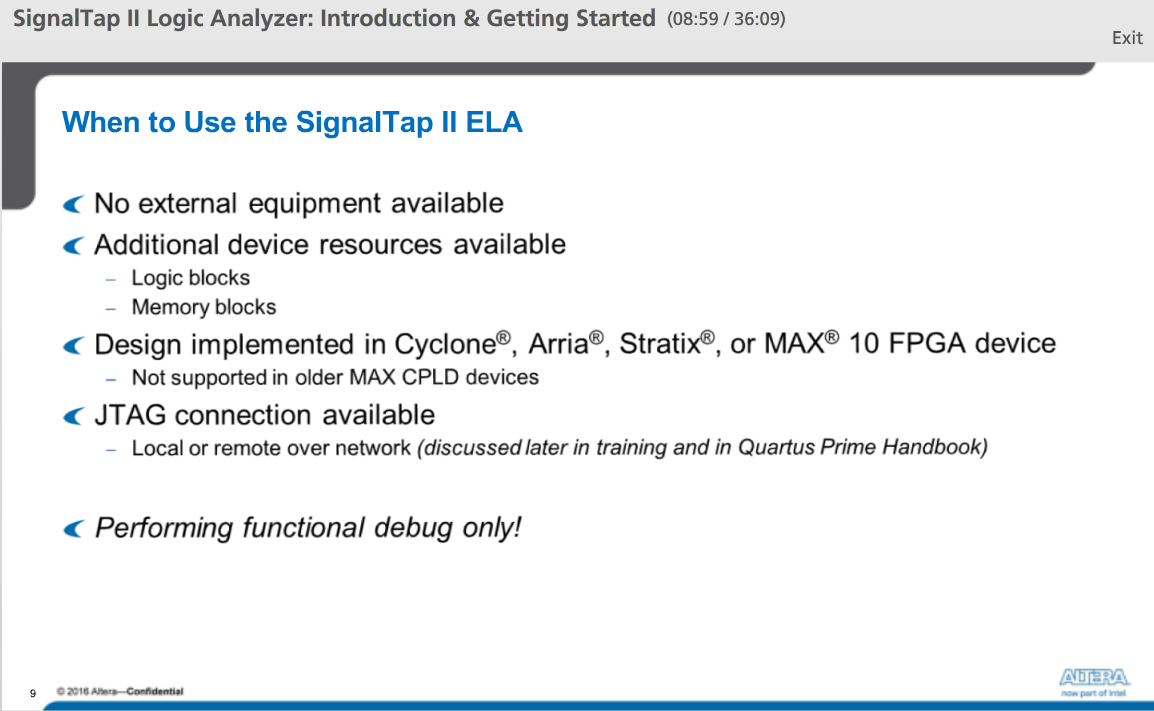


The SignalTap II logic analyzer emulates many of the features of an external logic analyzer. It samples, captures, and stores on-chip internal signal data on the rising edge of a specified sampling clock. We are able to control and view captured data on the fly through a standard JTAG connection. The ELA is available free.



The logic analyzer is created in the target device using unused logic elements and memory blocks.

The JTAG connection programs the device as well as controls the operation of the SignalTap II instances after device programming.



When to use the SignalTap II ELA:

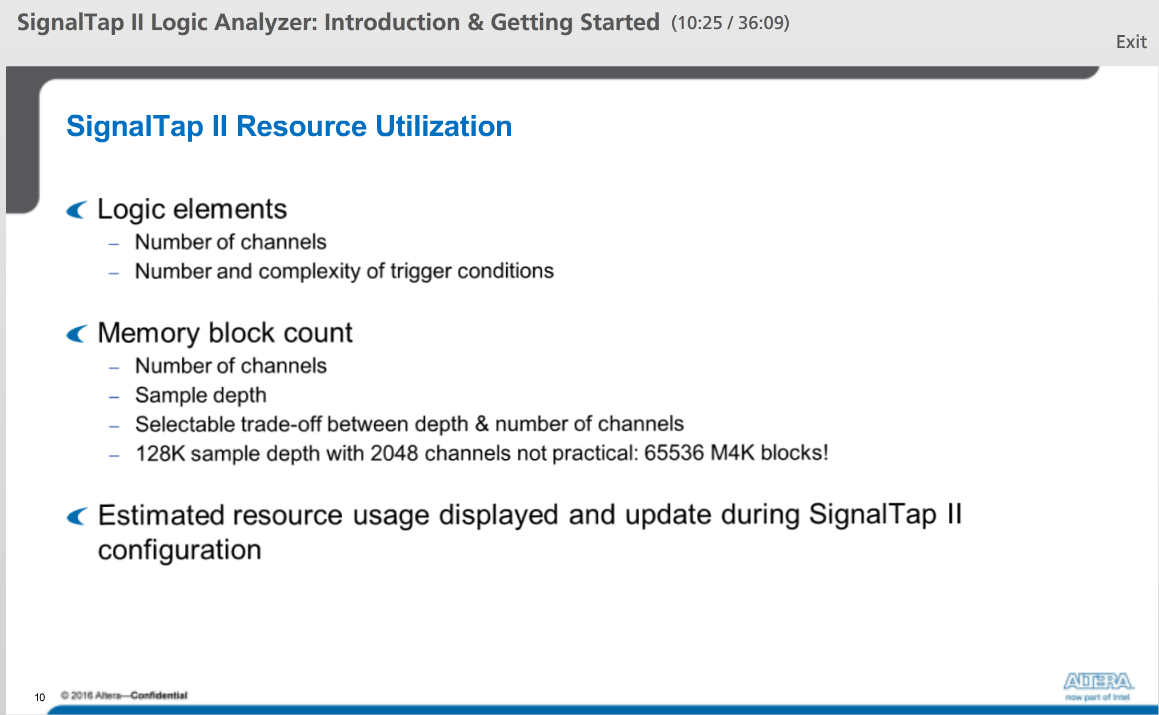
1) If we don’t have any external test equipment available

2) If we have additional logic and memory blocks available

3) ELA is only supported on Cyclone, Arria, Stratix and MAX 10 FPGA devices

4) We must have JTAG connection to the device under test

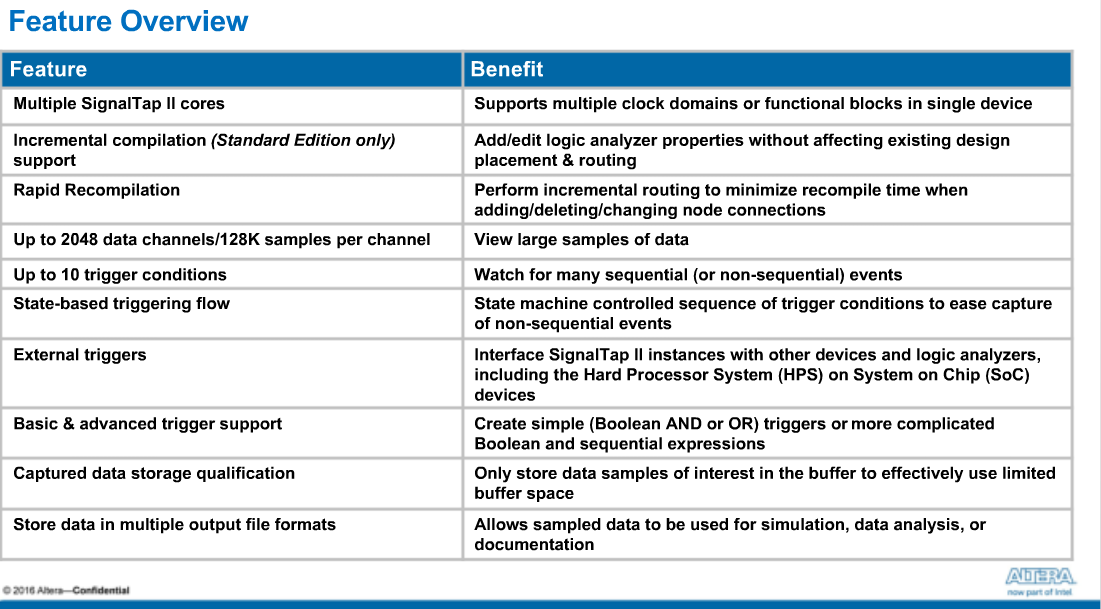
We can only perform a functional debug.



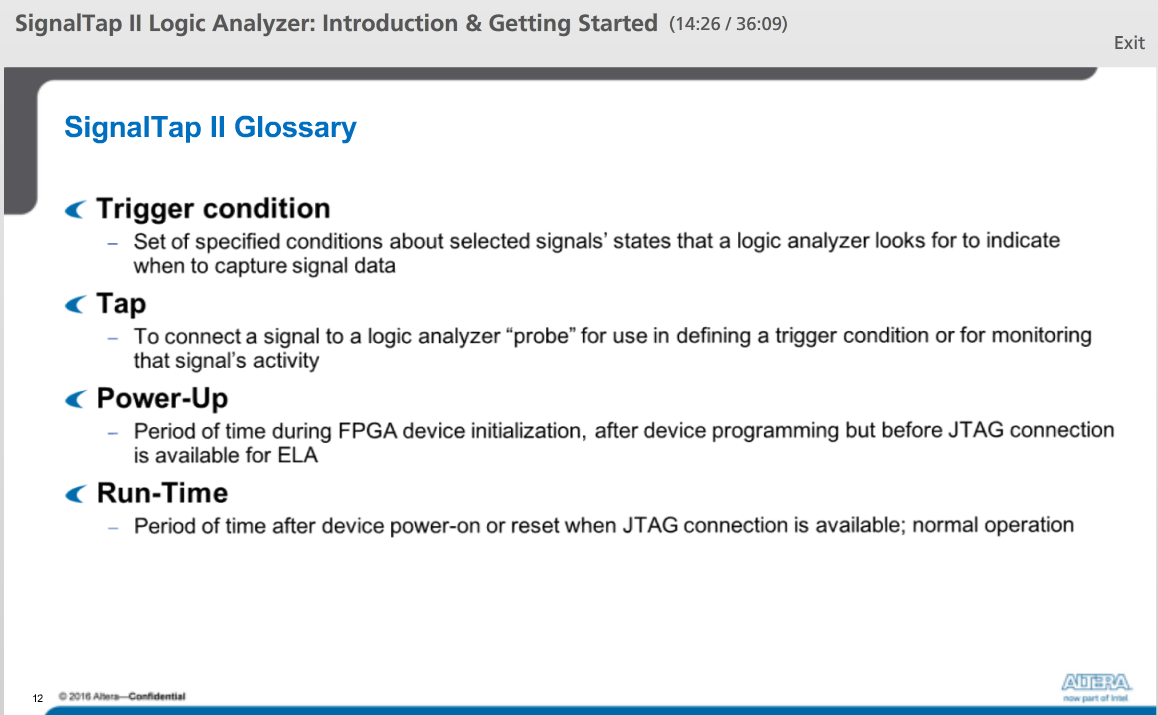
How the logic analyzer consumes device resources:

Logic and standard routing channels are used to implement the SignalTap II IP core, while memory blocks are used for sample storage.

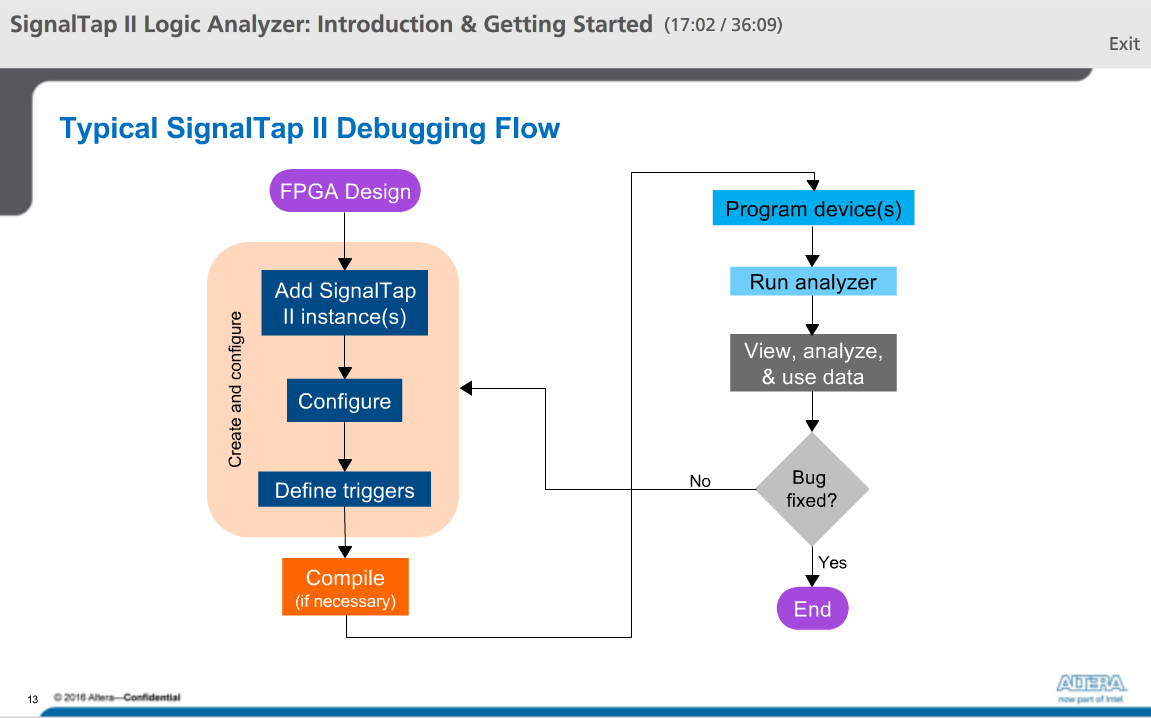
As we make changes to the logic analyzer configuration, a resource usage estimator constantly updates to keep track of the amount or resources required.



The SignalTap II logic analyzer provides many features and benefits to help debug a design.



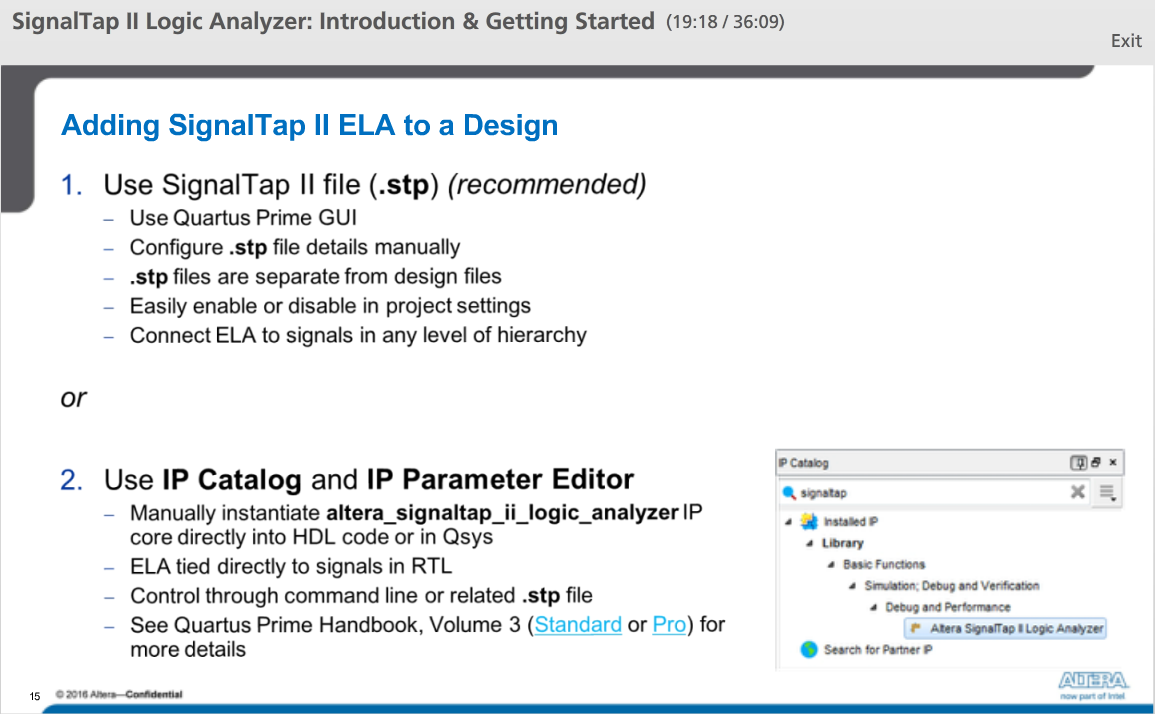
Some terminology that will be used throughout the training.



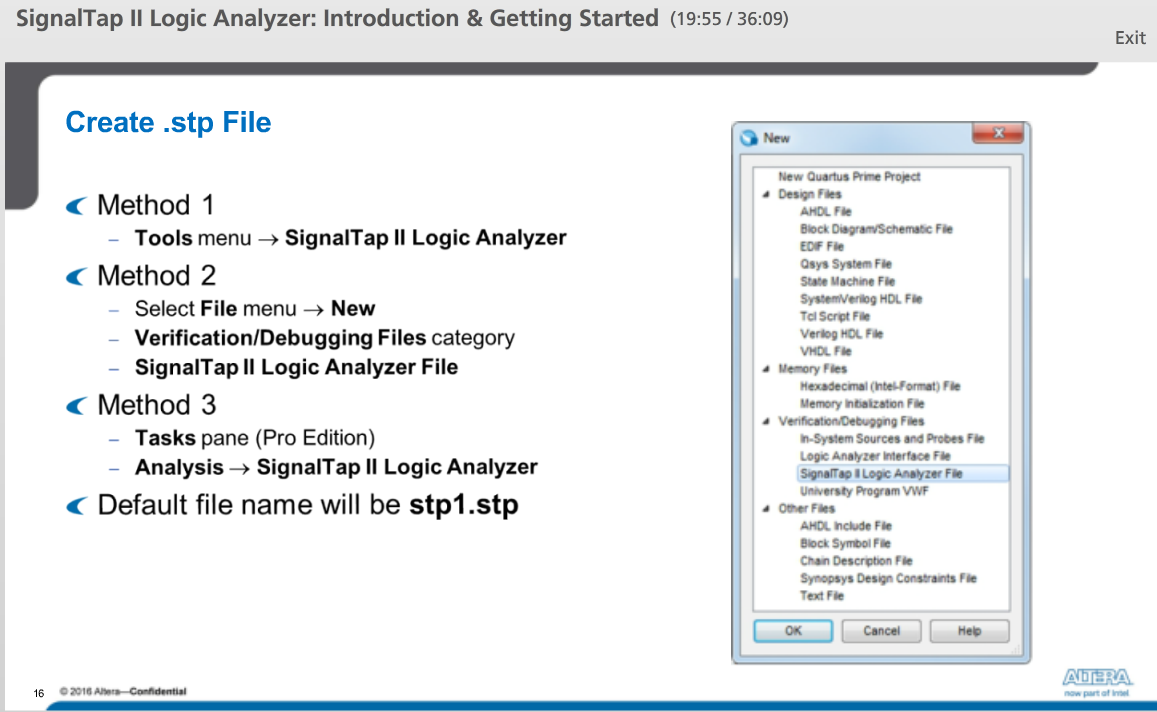
Typical debugging flow:

You start with a new or existing FPGA design. In the design, one or more instances of the logic analyzer are added. Each instance is then configured as desired for the target device and the debugging situation by selecting signals to monitor and specifying properties of the sampling buffer. Once an instance is configured, trigger conditions can be defined. These three main tasks can be repeated in different orders during the debugging process, so we'll group them together in the "create and configure" stage of the task flow.

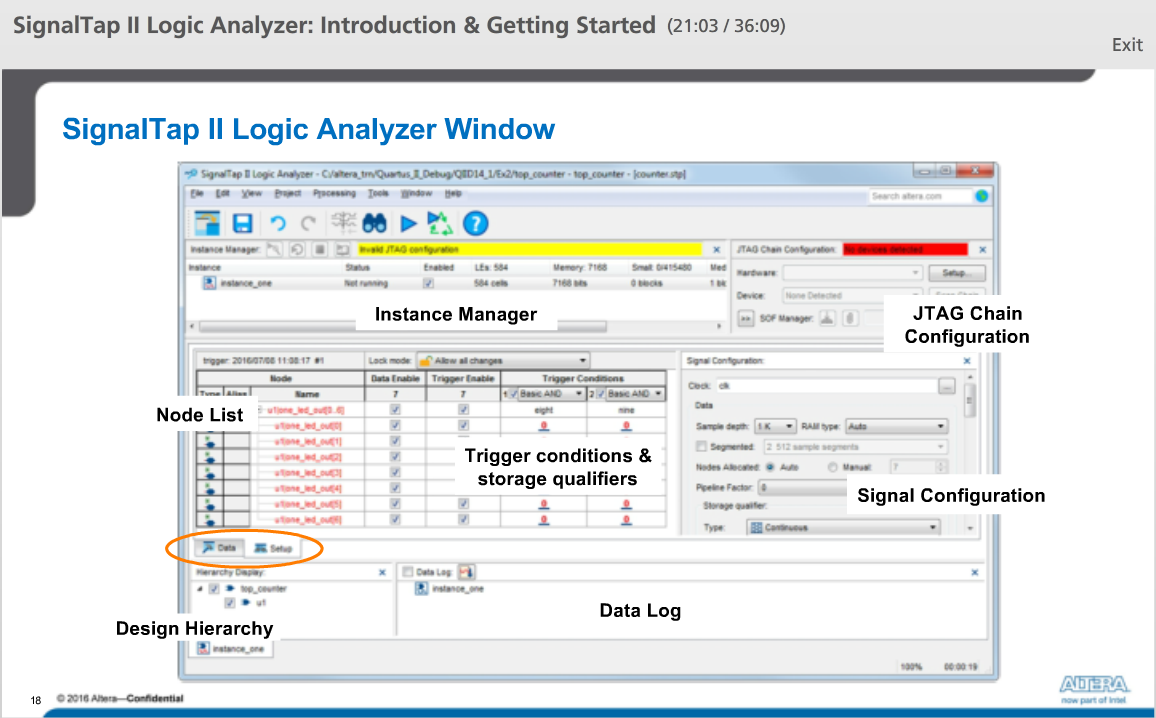
Once the instance or instances are properly configured, the project is compiled, if necessary. When compilation is complete, the target device or devices are programmed using the device's JTAG interface. Once the design that includes the logic analyzer is programmed into the device, it can be run and controlled through the JTAG connection. When the trigger occurs, the logic analyzer stops and the captured data is transferred to the SignalTap II file window where it can be viewed, analyzed, saved, or used to locate a bug in the design. If the bug is found and fixed, the debugging flow is complete. If not, the logic analyzer can be reconfigured or the trigger conditions adjusted to try looking for the problem again or to look for other bugs. All of the procedures mentioned in this training will fit within one of these main tasks.



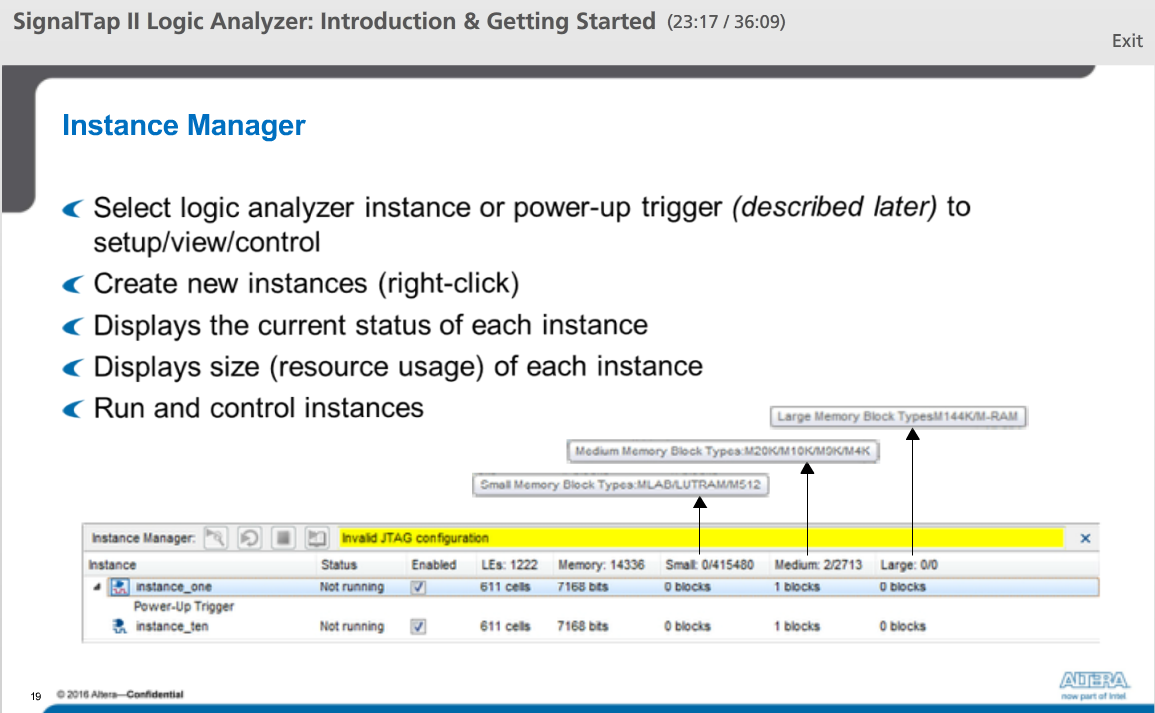
There are two ways to add the logic analyzer to a project.



There are several ways to create .stp file



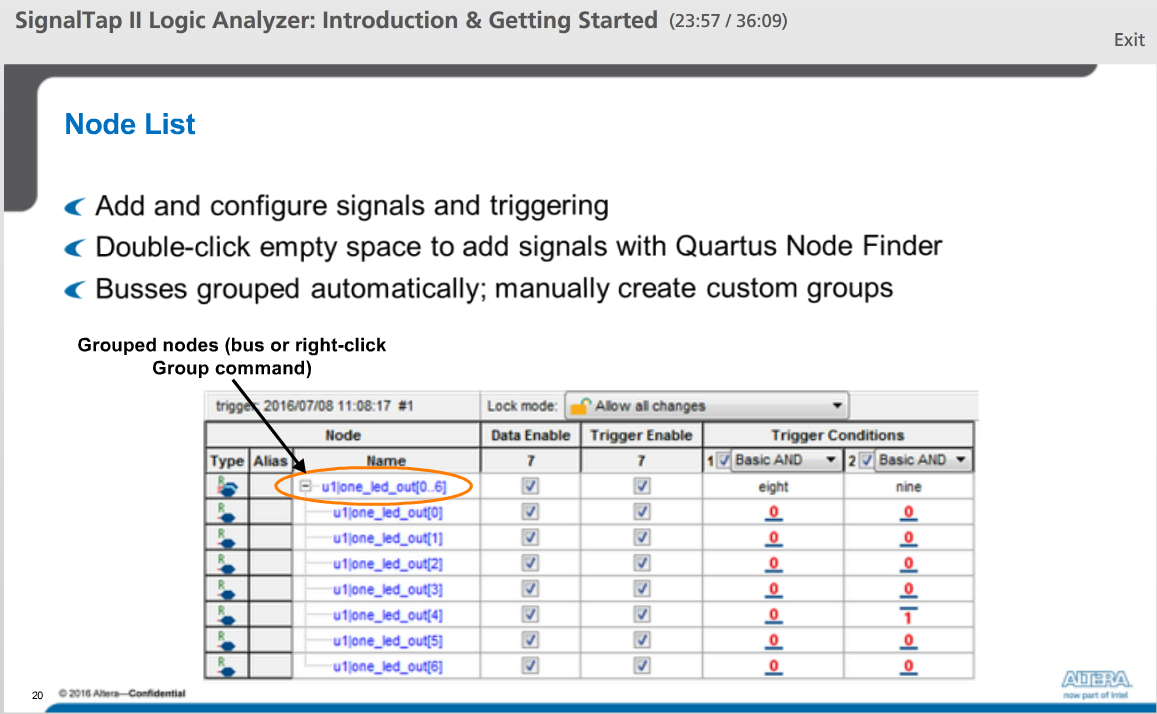
The main SignalTap II file window has a number of different panes, each with its own function. Each pane has a number of different settings and configuration options.



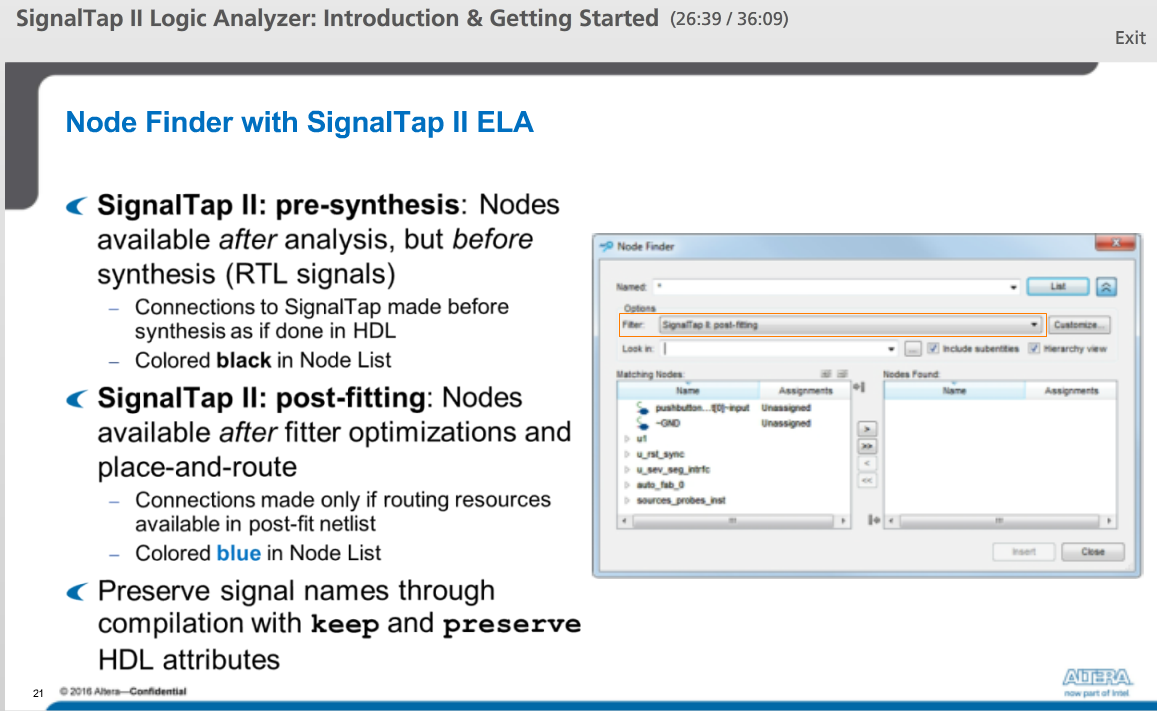
The Instance Manager includes a number of different controls and indicators to work with individual or multiple SignalTap II instances.

For each instance, the Instance Manager indicates the status of the instance as well as an estimate (or actual count) of the resources used by the instance.

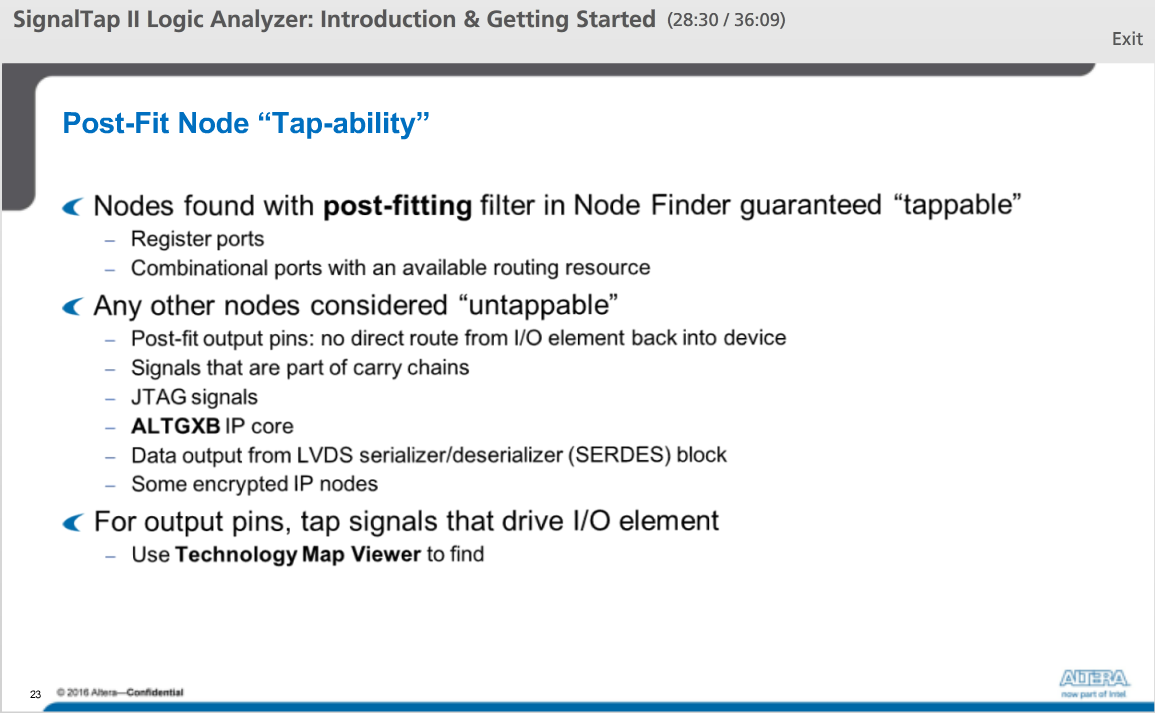
The health monitor at the top of the Instance Manager indicates the overall status of all SignalTap II instances and whether a recompilation is necessary to integrate updated SignalTap instances into the design before they can be run. Finally, the analysis control buttons next to the health monitor control how each instance is run to watch for the trigger conditions.



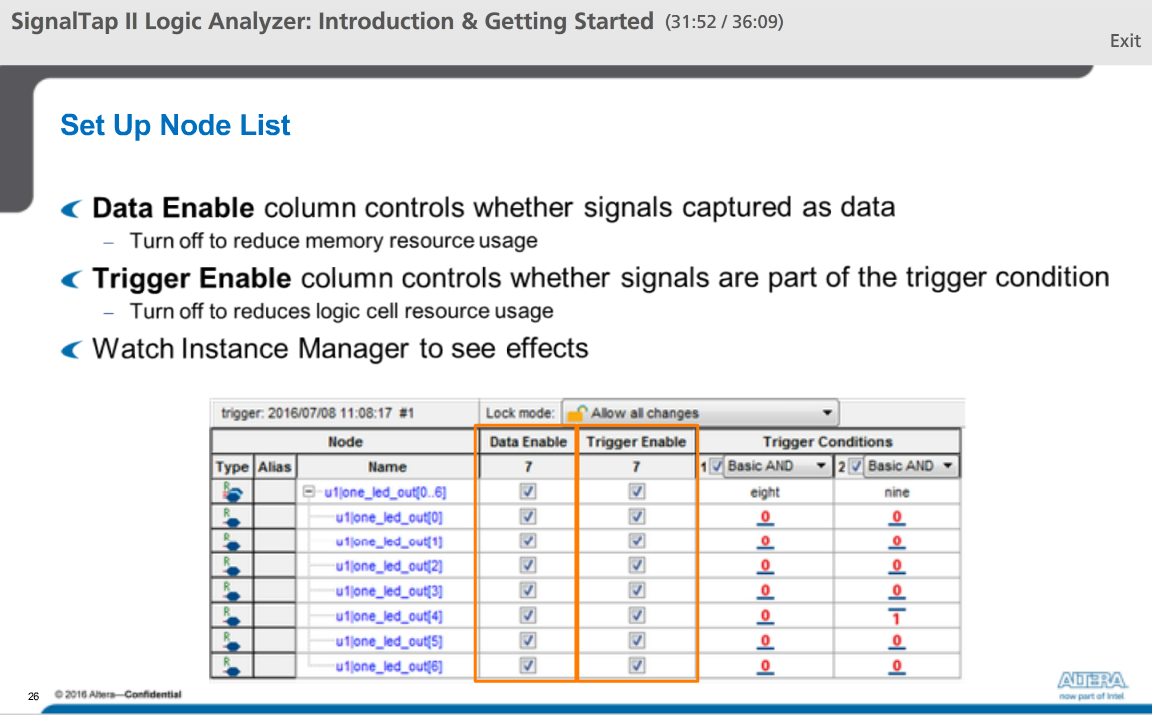
The Node List on the Setup tab in the SignalTap file window is used to select signals from your design for monitoring or triggering, as well as for constructing the actual trigger conditions.



The standard Node Finder found throughout the Quartus Prime software is used to select signals for use by the ELA. When accessed in the SignalTap II file window, two SignalTap-specific node filters, named SignalTap II pre-synthesis and SignalTap II post fitting, should always be used.



It is recommended to use the special SignalTap II filters in the Node Finder or cross-probe from the Technology Map Viewer. Using these methods guarantees that a selected node can be tapped, meaning that the node is “tap-able”.



After signals are added to the Node List, you can configure how the logic analyzer should use those signals. The Data Enable and Trigger Enable checkboxes on each row in the Node List are used to disable usage of the associated signal.