Digital Logic
Design with FPGA

Design Flow with Quartus

Simulation & HW Debug

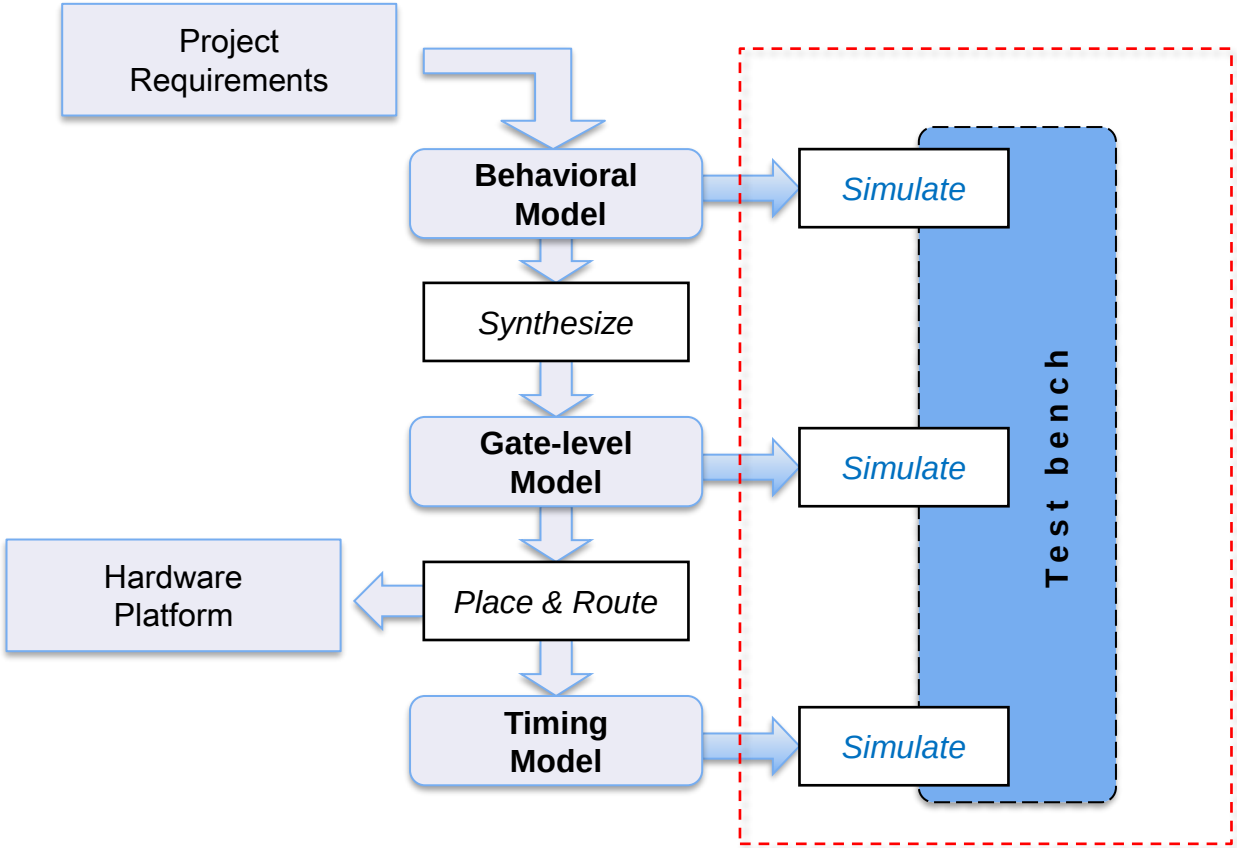
Outline

- Modeling digital systems
- Simulation in typical design flow
  - Simulation tools
  - Timing simulation
- Hardware debug
  - In-System Sources and Probes (ISSP)
  - Signal Tap Internal Analyzer

Modeling Digital Systems

- Simulation and modeling at the system level
  - performance analysis
- Behavior specification at the algorithmic level
  - preliminary functional verification of algorithms
  - division into hardware and software
  - high-level synthesis
- Simulation behavioral models of standard elements
- Functional simulations at the system/package level
  - full
  - bus
- Synthesizable models at the RTL (register transfer level)
  - full functional specification of the project
- Model of the system environment (testbench)
- Simulation models of library cells from integrated circuit manufacturers (VITAL standard) –
  - time verification of ASIC/FPGA systems

Simulation in typical design flow

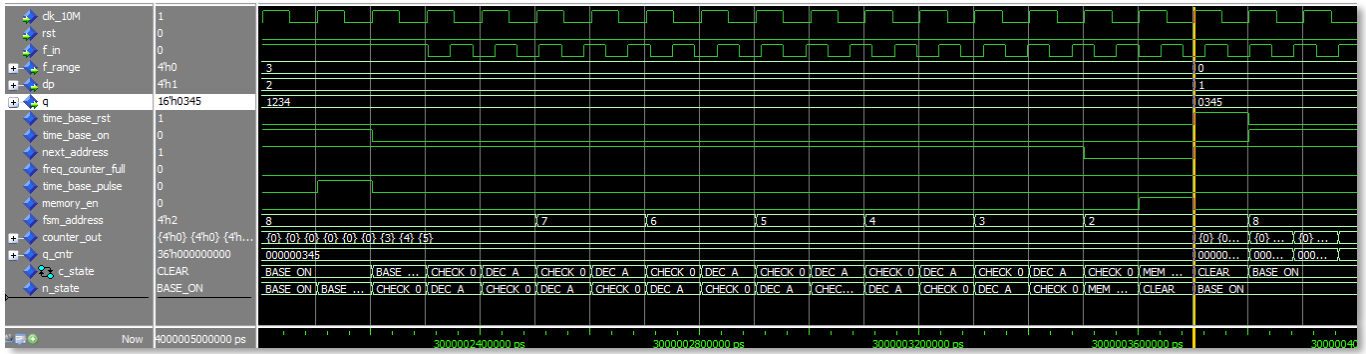


Simulation

- Provide results that are impossible to measure in HW prototype
- Include wide range of analyses
- Reduce development costs
- Minimize time-to-market
- Delivering visibility of all signals in design

…

- Designer has to create stimulus that matches device behavior
- Can take very long time to run for large designs

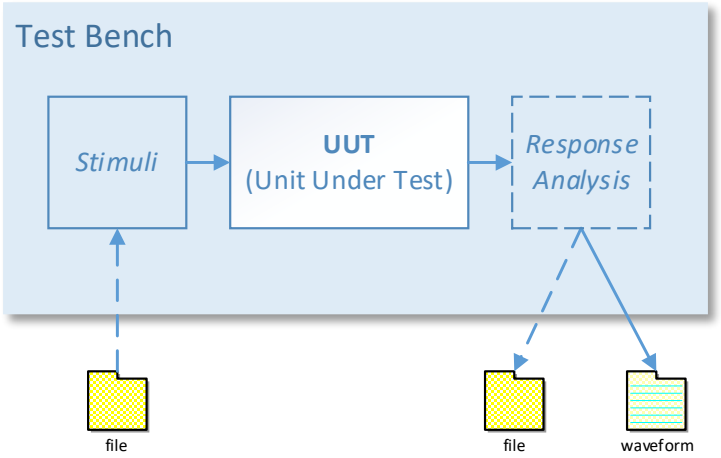


source: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-software/fpga-development-tools-support.html

HDL Testbench

- Testbench is a specific design unit described in HDL
- It has no ports and is intended to simulate the designed device (UUT)
- Defines forces (stimuli) and how to interpret simulation results

- Creating a testbench:
  - written "by hand" by the designer
  - generated automatically using graphic editors based on the description of UUT ports



source: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-software/fpga-development-tools-support.html

HDL Testbench

```
6   library IEEE;
7   use IEEE.STD_LOGIC_1164.ALL;
8   use std.env.all;
9
10  entity d_cntr4ceo_tb is
11      generic(T: time:= 1 ns);
12  end entity d_cntr4ceo_tb;
13
14  architecture behav of d_cntr4ceo_tb is
15      signal clk,rst: std_logic := '0';
16      signal ce,ceo,tc : std_logic;
17      signal q: std_logic_vector(3 downto 0);
18
19      constant PERIOD : delay_length := 20*T;
20
21      procedure clk_gen(signal s: out std_logic; period: delay_length := 10 ns) is
22       begin
26
27      procedure pulse (signal s: out std_logic; Hpulse,Lspace: delay_length) is
28       begin
32
33      procedure stop_after_falling_edge(signal trig: in std_logic; delay: delay_length := 0 ns) is
34       begin
41
42      procedure stop_after_Nfalling_edge(signal trig: in std_logic; delay: delay_length := 0 ns; N: natural:=2) is
44       begin
57
58  begin
59
60      UUT: entity work.d_cntr4ceo
61      port map(clk,rst,ce,tc,ceo,q);
62
63      pulse(ce,105*PERIOD,2*PERIOD);
64      pulse(rst,2*PERIOD,12*PERIOD);
65      clk_gen(clk,PERIOD);
66      stop_after_Nfalling_edge(ceo,PERIOD,5);
67
68  end architecture behav;
```

Entity without I/O

Stimulus definition

UUT instantiation

Stimulus run

Questa Overview

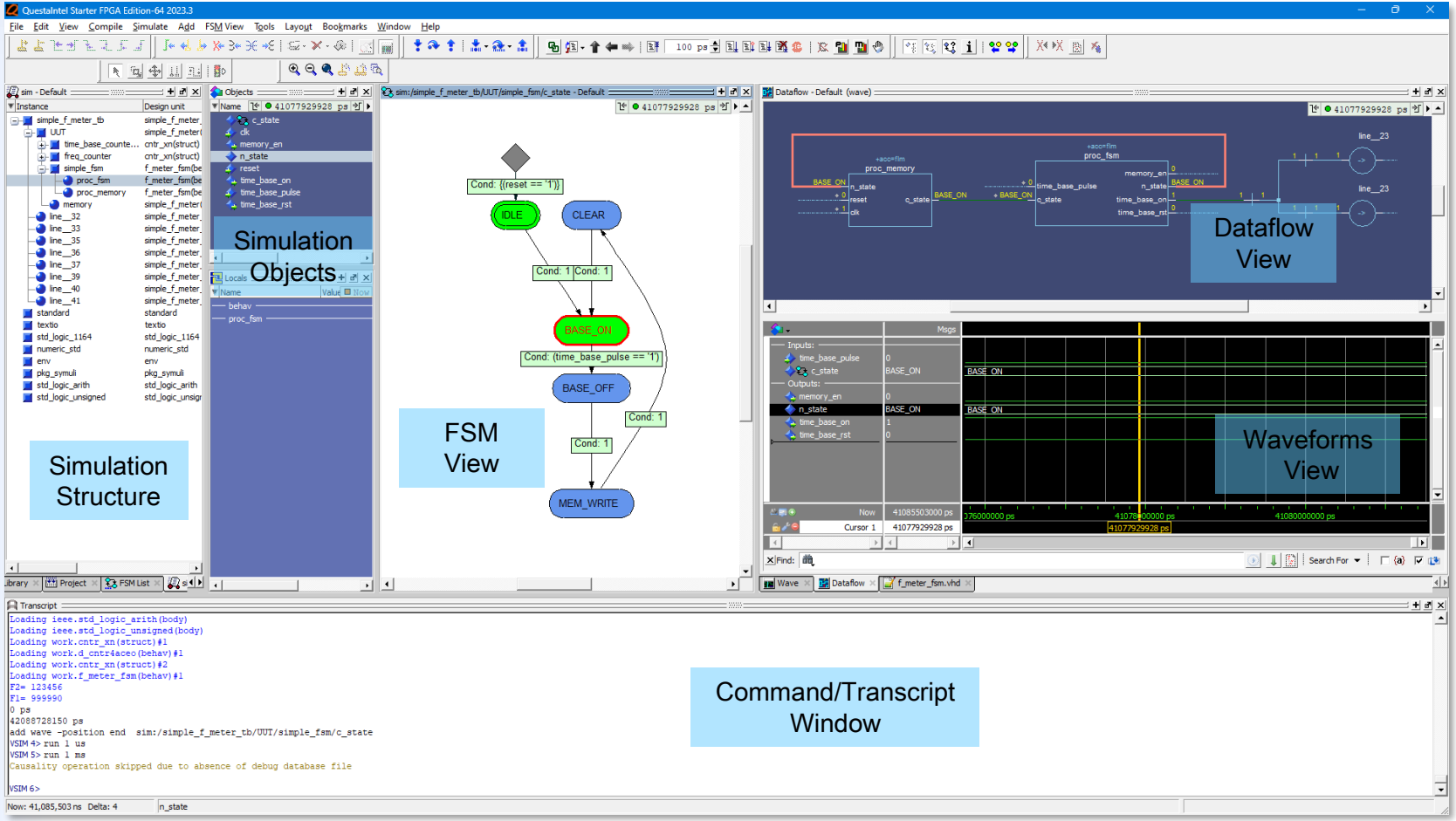
- Multi-language HDL simulation environment
- It can be used independently or Quartus can create startup scripts

- Two versions of the Questa* Intel FPGA simulator available:
  - Questa* Intel® FPGA Edition
    - Licenses are required and must be purchased
    - 2-6X slower than Questa Core
  - Questa* Intel® FPGA Starter Edition
    - Licenses are required but are free
    - 40% slower than paid edition

- Both the free and paid editions require licenses for performing elaboration and simulation, but not for compilation

source: https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-software/fpga-development-tools-support.html

Questa Overview



Digital Logic Design with FPGA

Questa Simulation Types

Simulation Type	Description	Occurs
RTL	Simulation of an RTL design consisting of one or more RTL files. The RTL files can instantiate low level blocks, such as primitives, basic IP functions, and ATOMs.	Can perform before synthesis
Post-Synthesis (Gate-Level)	The Quartus EDA Netlist Writer tool generates the post-synthesis netlist. The post-synthesis netlist is a netlist of low level blocks called ATOMs. The post-synthesis netlist is a purely functional netlist.	Must perform after synthesis
Post-Fit (Gate-Level)	The Quartus EDA Netlist Writer can generate a Verilog HDL or VHDL gate-level netlist after the Fitter stage completes (post-fit netlist). The post-fit netlist is a netlist of ATOMs that the Fitter placed and routed on the FPGA device. The post-fit netlist is a purely functional netlist. **Note:** The post-fit netlist includes chip locations of ATOM instances in commented lines. The post-synthesis netlist does not include this data.	Must perform after fitting

Note: The Quartus Prime software supports post-fit functional simulation, but does not support post-fit timing simulation.

source: https://www.intel.com/content/www/us/en/docs/programmable/730191/24-1/supported-simulation-types.html

In-System Sources and Probes (ISSP)

- Hardware and Software Requirements
  - Intel Quartus Prime Lite Edition
  - Download Cable (USB-Blaster download cable or ByteBlaster cable)
  - Intel FPGA development kit or user design board with a JTAG connection to device under test

- The In-System Sources and Probes Editor supports the following device families:
  - Arria series
  - Stratix series
  - Cyclone series
  - MAX series

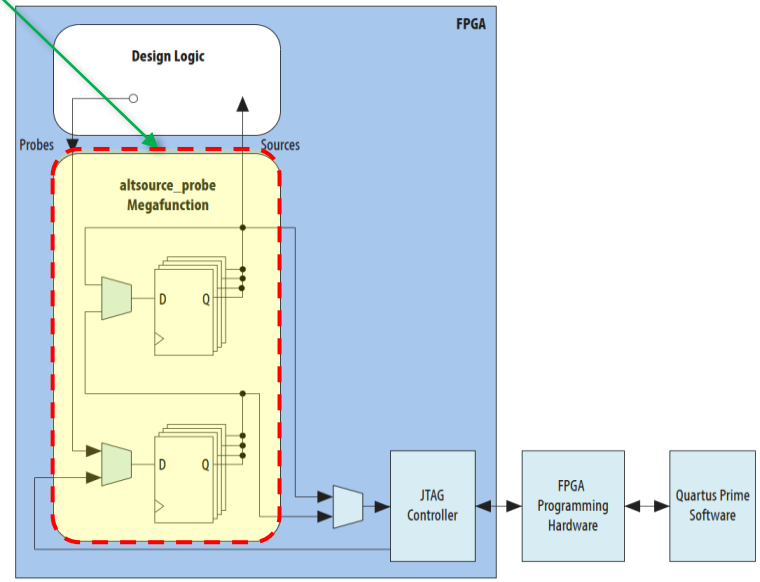
- ✓ Quickly set signal to constants: pins or internal nodes
- ✓ Easily monitor signals (no-triggered continuous display)
- ✓ Works on actual hardware
- × Not-triggered – might miss activity!

source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/hardware-and-software-requirements-46738.html

ISSP

The ISSP system consists of the ALTSOURCE_PROBE IP core and an interface to control the ALTSOURCE_PROBE IP core instances during run time

- ISSP Editor consists of a probe function and interface to control the instances during run-time
- Allows an easy way to drive and sample signals in hardware
- Operates over JTAG
- Each ISSP instance can view/probe up to 512 signals
- Each ISSP instance can drive/source up to 512 signals



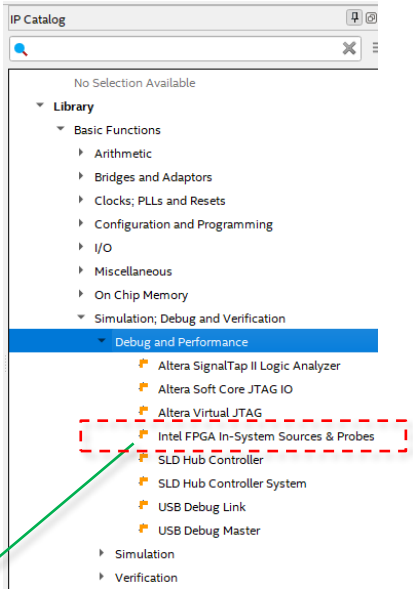
ISSP block diagram

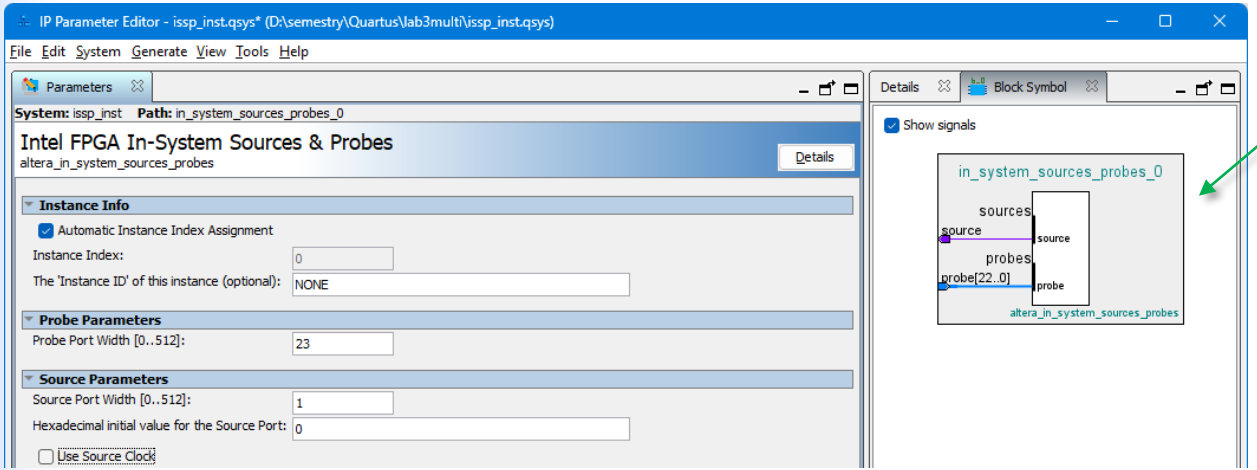
source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-using-in-system-sources-45607.html

ISSP

Using In-System Sources and Probes

- Create ISSP IP instance using the IP Parameter Editor
- Instantiate in design and compile project
- Program target device
- Create and use ISSP Editor (.spf file) to control sources and probes





ISSP IP core setup

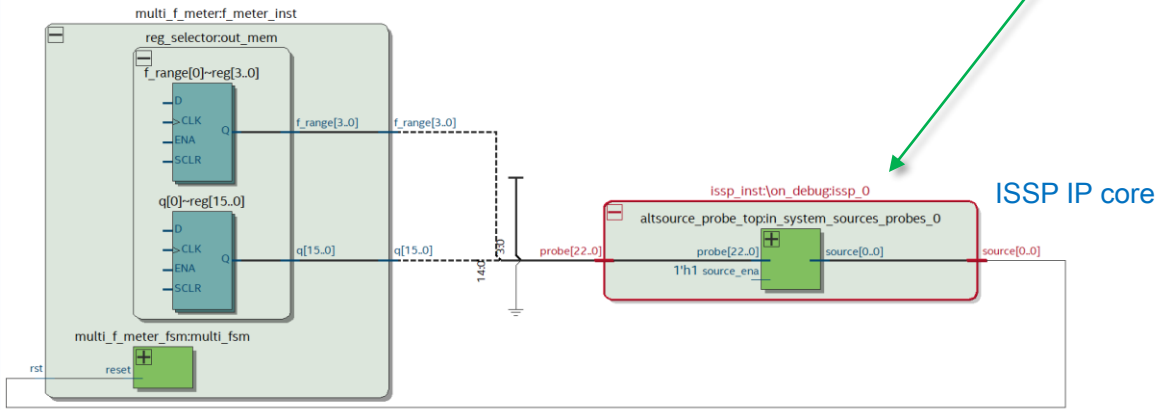
source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-using-in-system-sources-45607.html

ISSP

Using In-System Sources and Probes

- Create ISSP IP instance using the IP Parameter Editor
- Instantiate in design and compile project
- Program target device
- Create and use ISSP Editor (.spf file) to control sources and probes

```
57      on_debug:
58        if DEBUG generate
59            issp_0: issp_inst port map(source => source, probe => q(22 downto 0));
60            rst <= source(0);
61        end generate;
62      no_debug:
63        if not DEBUG generate
64            rst <= not key(0);
65        end generate;
```

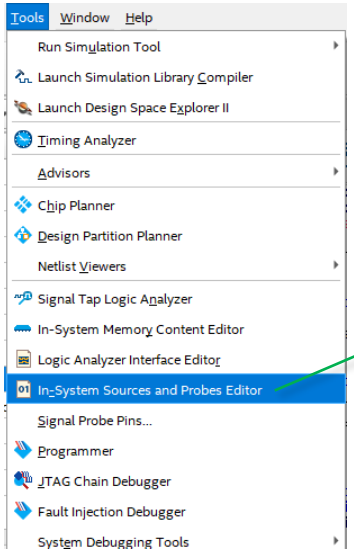


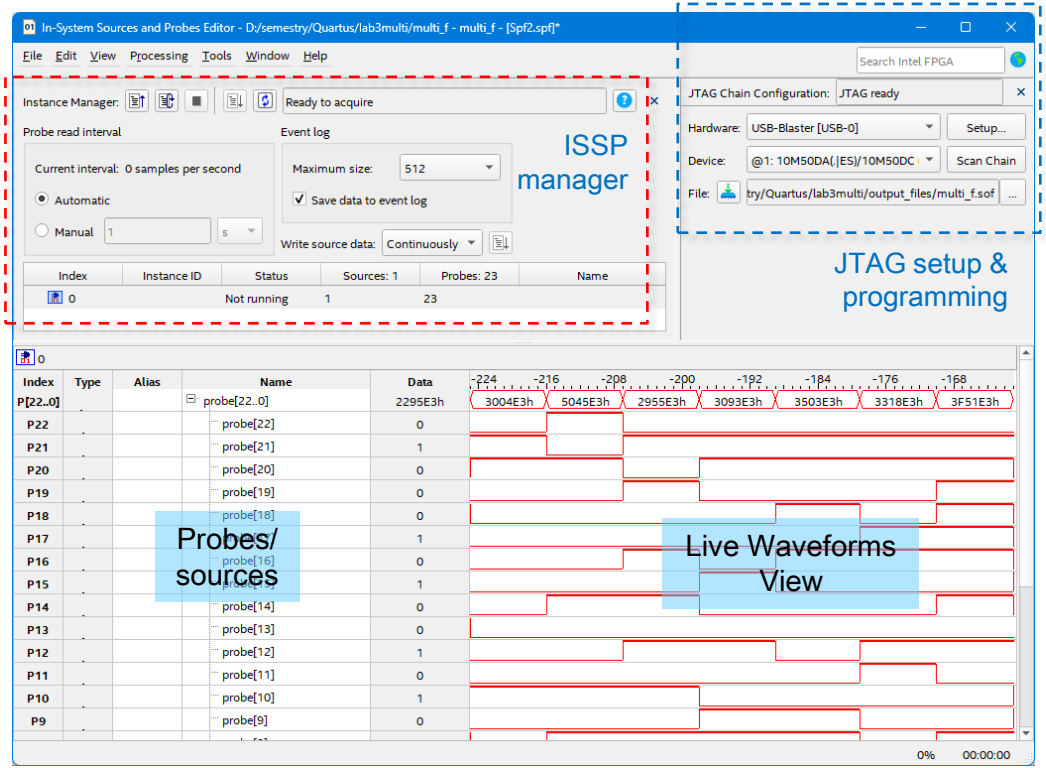
source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-using-in-system-sources-45607.html

ISSP

Using In-System Sources and Probes

- Create ISSP IP instance using the IP Parameter Editor
- Instantiate in design and compile project
- Program target device
- Create and use ISSP Editor (.spf file) to control sources and probes



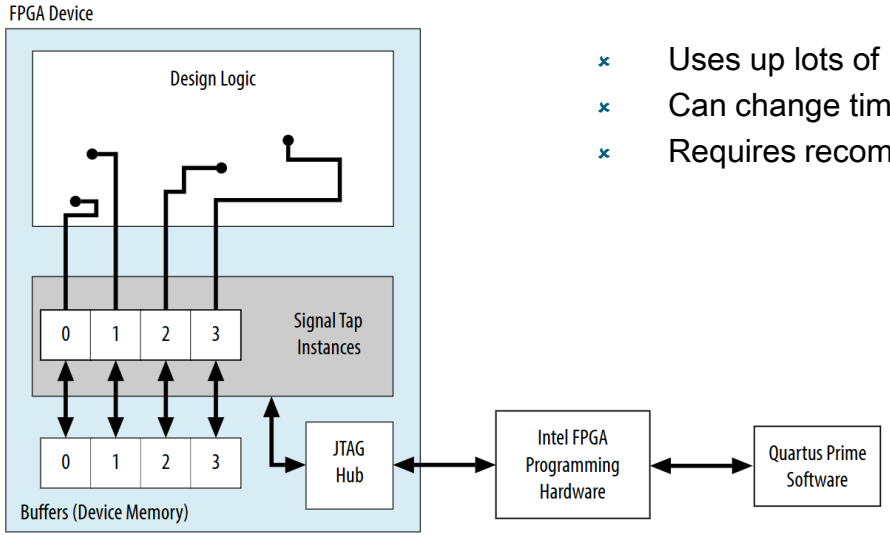


source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-using-in-system-sources-45607.html

Signal Tap

The Signal Tap Logic Analyzer captures and displays real-time signal behavior in an FPGA design using a defined clock signal

- ✓ 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 FPGA
- × Can change timing of design
- × Requires recompile witch takes time

source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

- Hardware and Software Requirements
  - Signal Tap Logic Analyzer (following software includes the Signal Tap):
    - Intel Quartus Prime Design Software
    - Intel Quartus Prime Lite Edition
    - Alternatively, use the Signal Tap standalone software and standalone Programmer software
  - Download Cable (USB-Blaster download cable or ByteBlaster cable)
  - Intel FPGA development kit or user design board with a JTAG connection to device under test

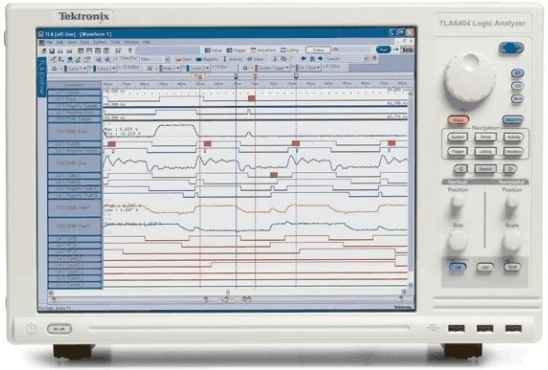
Note: During data acquisition, the memory blocks in the device store the captured data, and then transfer the data to the logic analyzer over a JTAG communication cable.

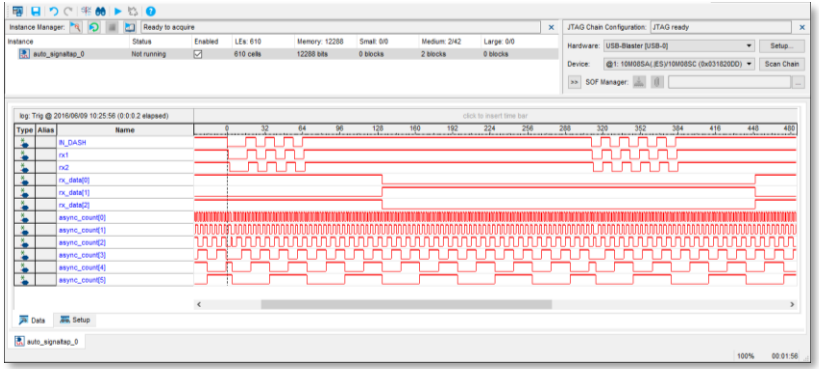
Opening the Standalone Signal Tap Logic Analyzer GUI

```
quartus_stpw <stp_file.stp>
```


source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap LA vs. External LA





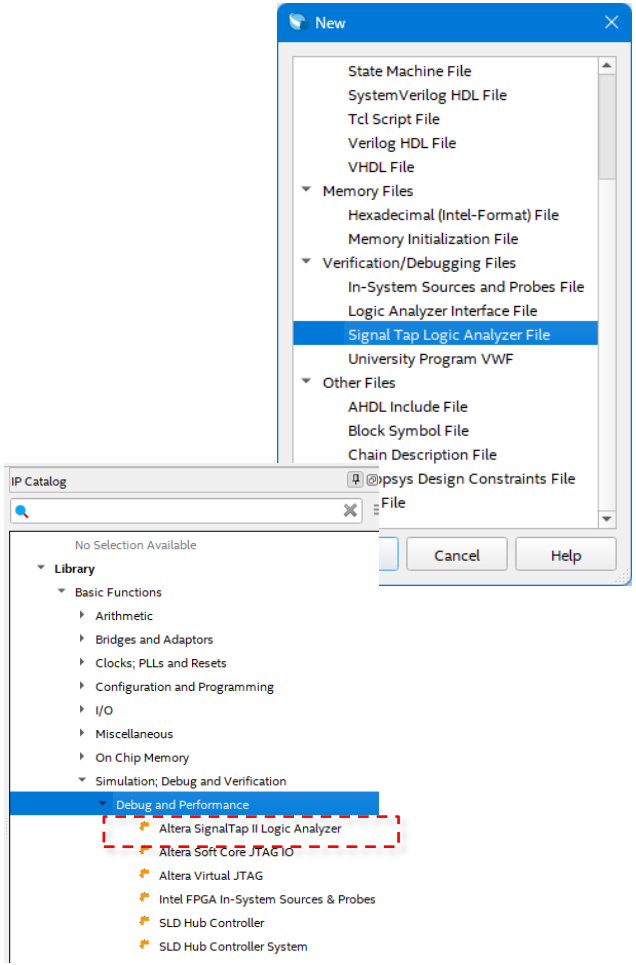
- System-level debug
- Can store large quantities of data
- Flexible trigger condition

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

source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

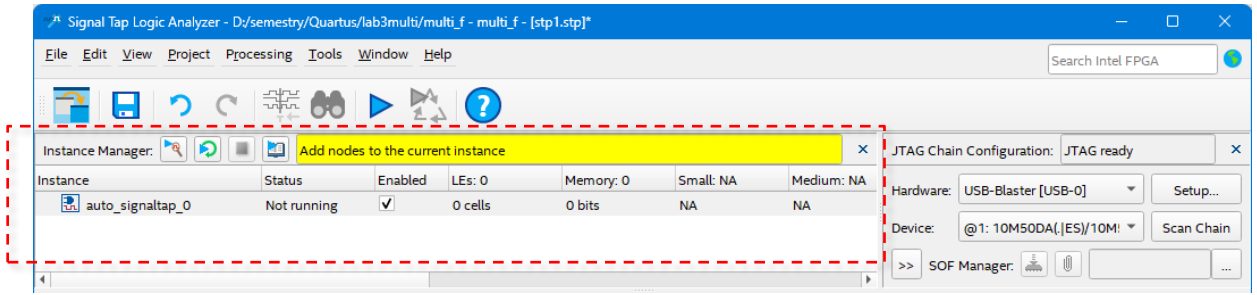
- Create Signal Tap instance
  - Use Signal Tap file (.stp)
    - Creates a file separate from design files
    - Convenient features and GUI
  - Use IP Catalog
    - Manually instantiate IP core directly into HDL code
    - Ties the ELA to the signals directly in RTL



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

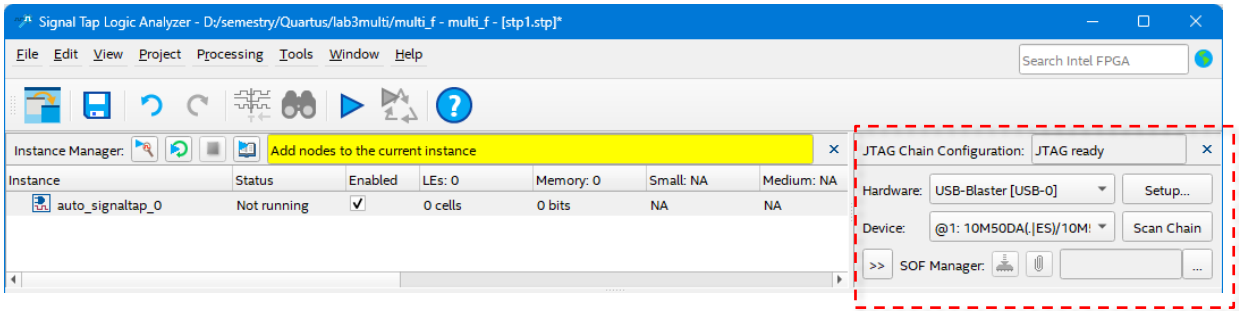
- Instance Manager
  - Identifies which instance is being edited in the GUI
  - Enable/disable instances quickly
  - Gives status and resource utilization



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

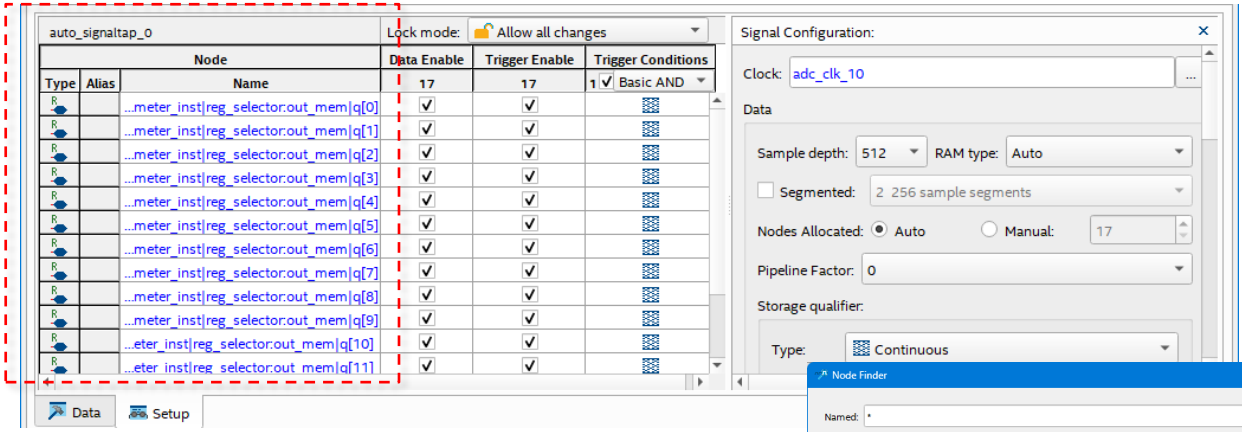
- JTAG Chain Configuration
  - Graphical setup for JTAG
  - Buit-in Programmer
  - Scans the JTAG chain and identifies available devices



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

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

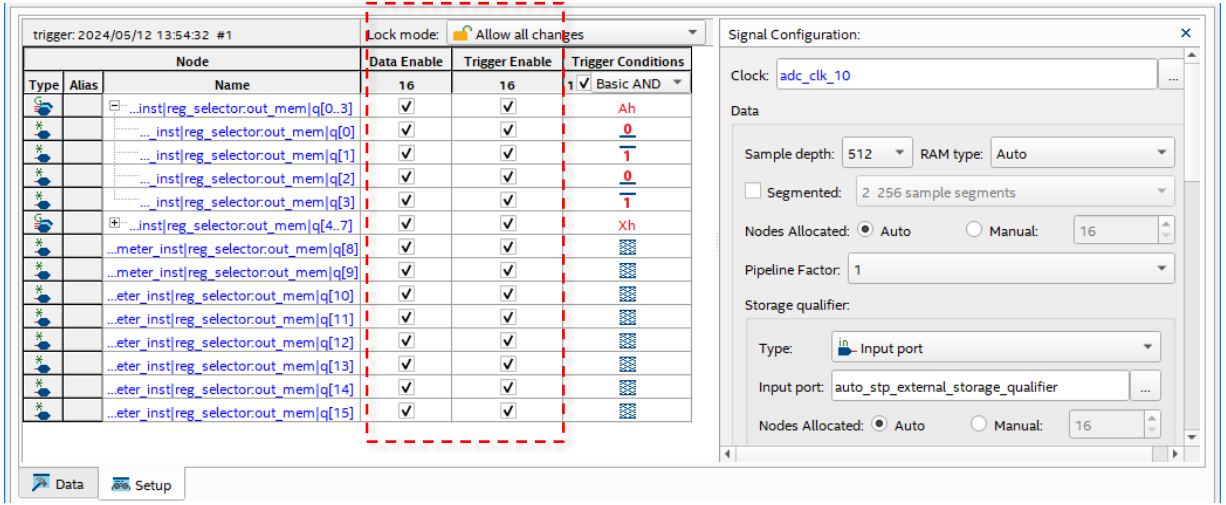


Node Finder

source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

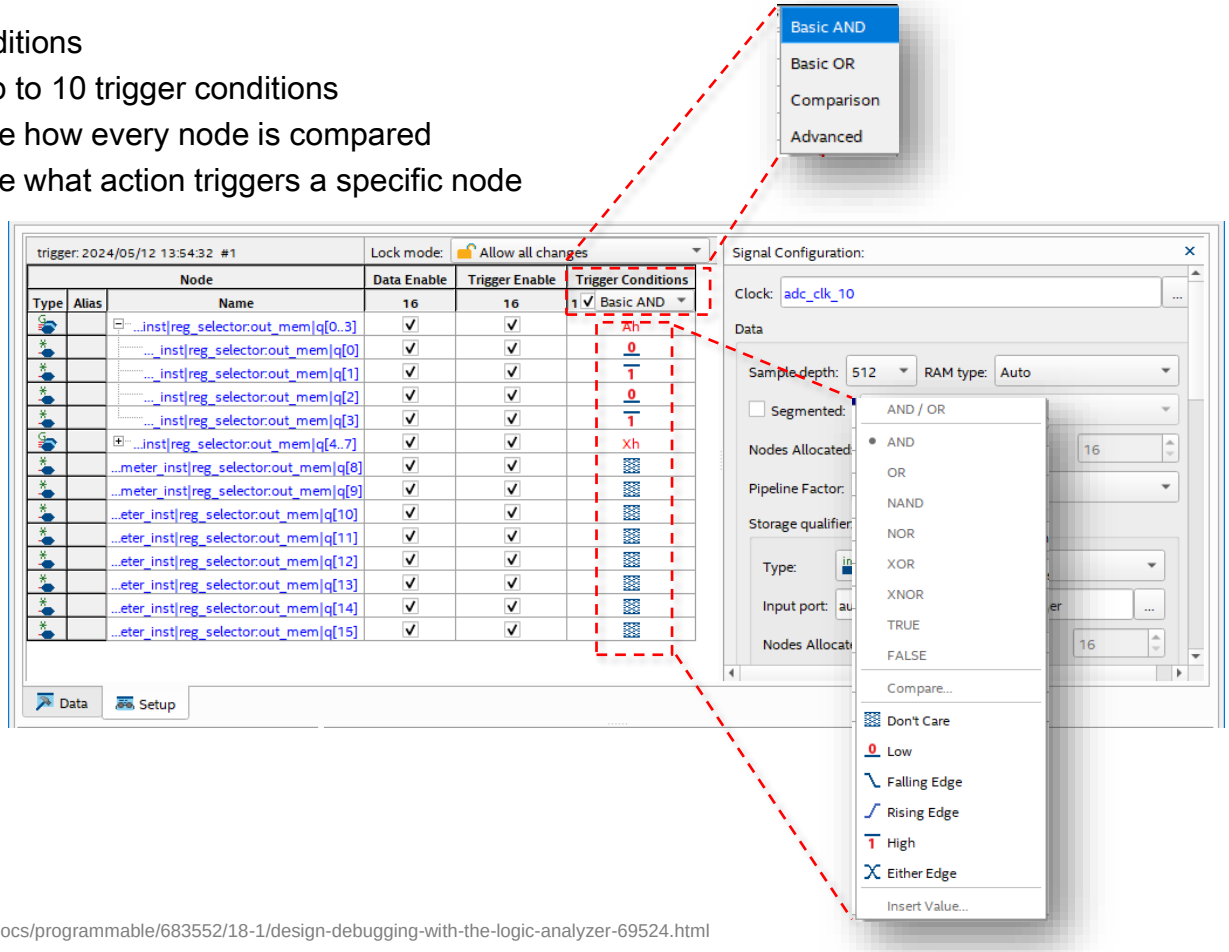
- Trigger Conditions and Qualifiers
  - Data Enable: saves signal data (disable to save memory)
  - Trigger Enable: signal is part of the trigger condition (disable to save LEs)



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

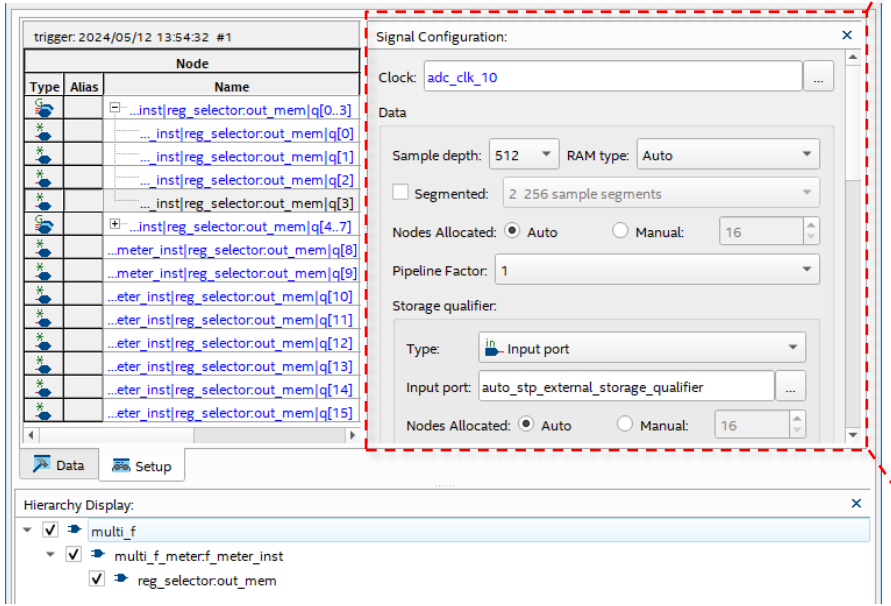
- Trigger Conditions
  - Add up to 10 trigger conditions
  - Choose how every node is compared
  - Choose what action triggers a specific node



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

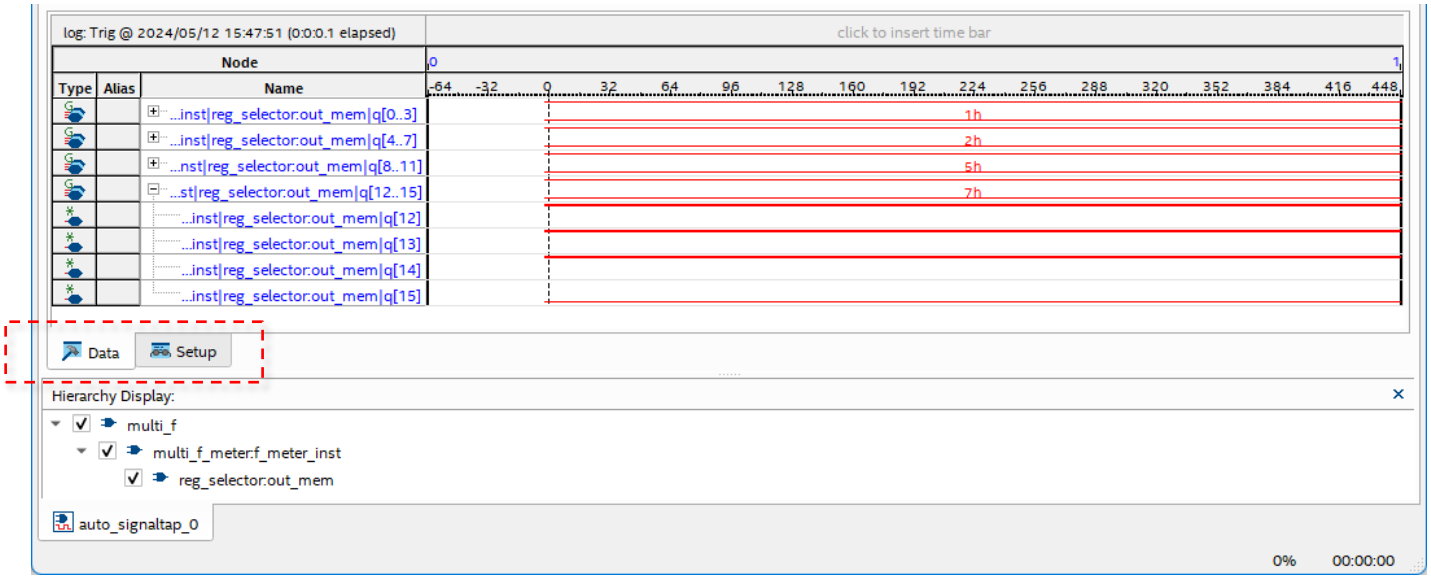
- Signal Configuration
  - Advanced trigger control
  - Select the number of trigger conditions
  - Trigger In/Out options



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html

Signal Tap

- Data/Setup Window
  - Setup allows configuration of nodes and trigger condition
  - Data shows the acquired signal information



source: https://www.intel.com/content/www/us/en/docs/programmable/683552/18-1/design-debugging-with-the-logic-analyzer-69524.html