Hardware Debugging

Objectives

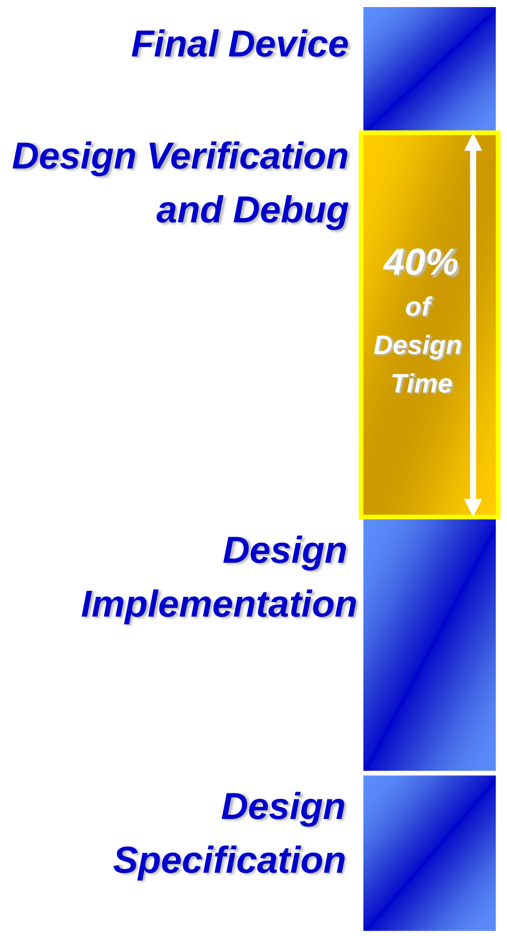
- **After completing this module, you will be able to:**
  - Understand the key benefits provided by the Vivado logic debug feature
  - List various debugging cores and their functionality
  - Describe the process of including debug tool sampling cores in the Vivado Design Suite

Outline

- *Introduction*
- Logic Debug Cores
- Logic Debug Probing Flows
- Summary

Debug and Verification is Critical

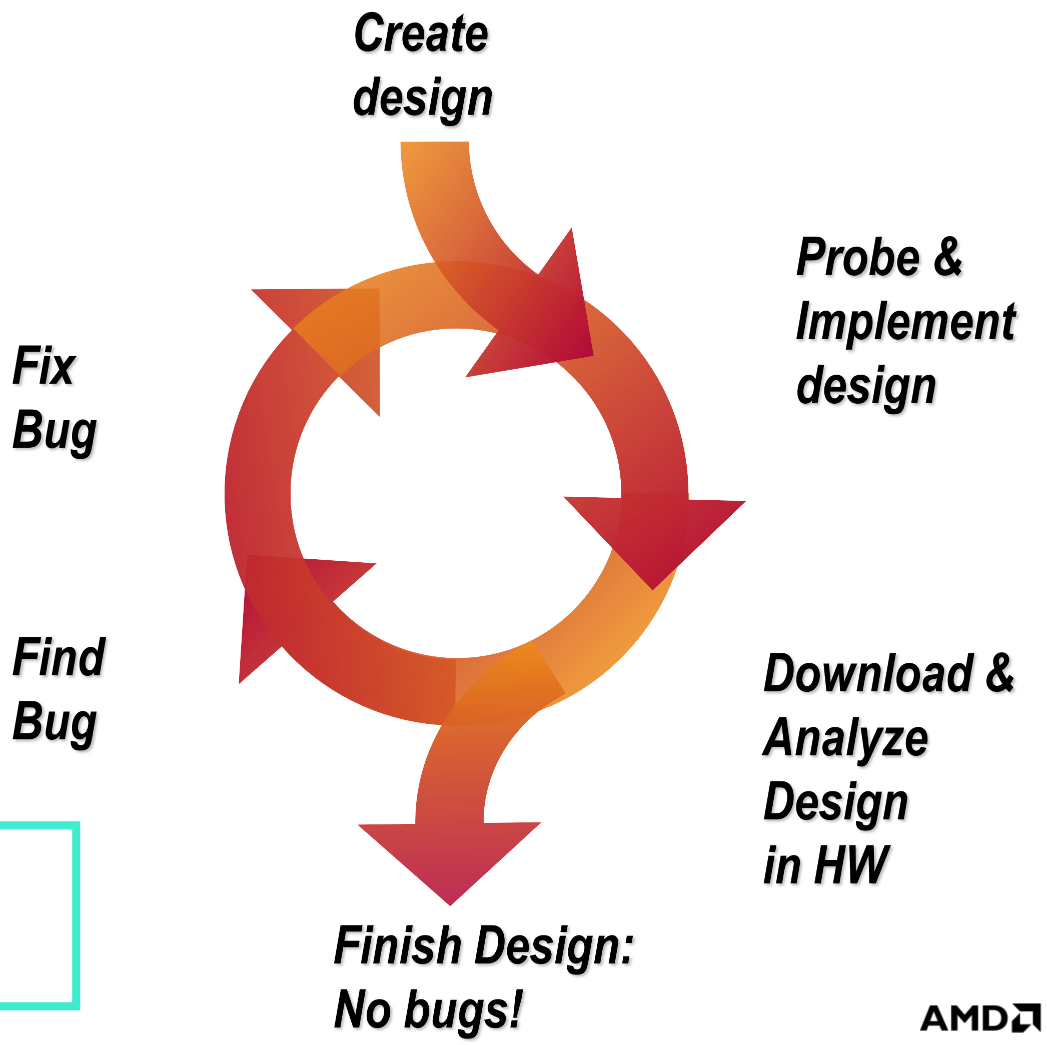
- Debug and verification can account for over 40% of creating a FPGA design
- Serial nature of debug and verification can make it difficult to optimize
- Inefficient strategy may result in product launch delay



Recommended Debug Methodology

- Engineers are trained to solve problems logically
  - *Break a problem into smaller parts*
  - *Simplify by reducing variables & variation*
  - *Make a prediction, verify the results*
  - *Plan how and where to debug early in the design cycles*
- FPGA design is an iterative process
- Debugging a FPGA design is an iterative process
  - ***1) Probe****: Adding/modify debug probes*
  - ***2) Implement****: Compile design w/probes*
  - ***3) Analyze****: Look for bugs using probes*
  - ***4) Fix****: Fix any bugs, repeat as necessary*

The reconfigurable nature of FPGAs facilitates the iterative debug process

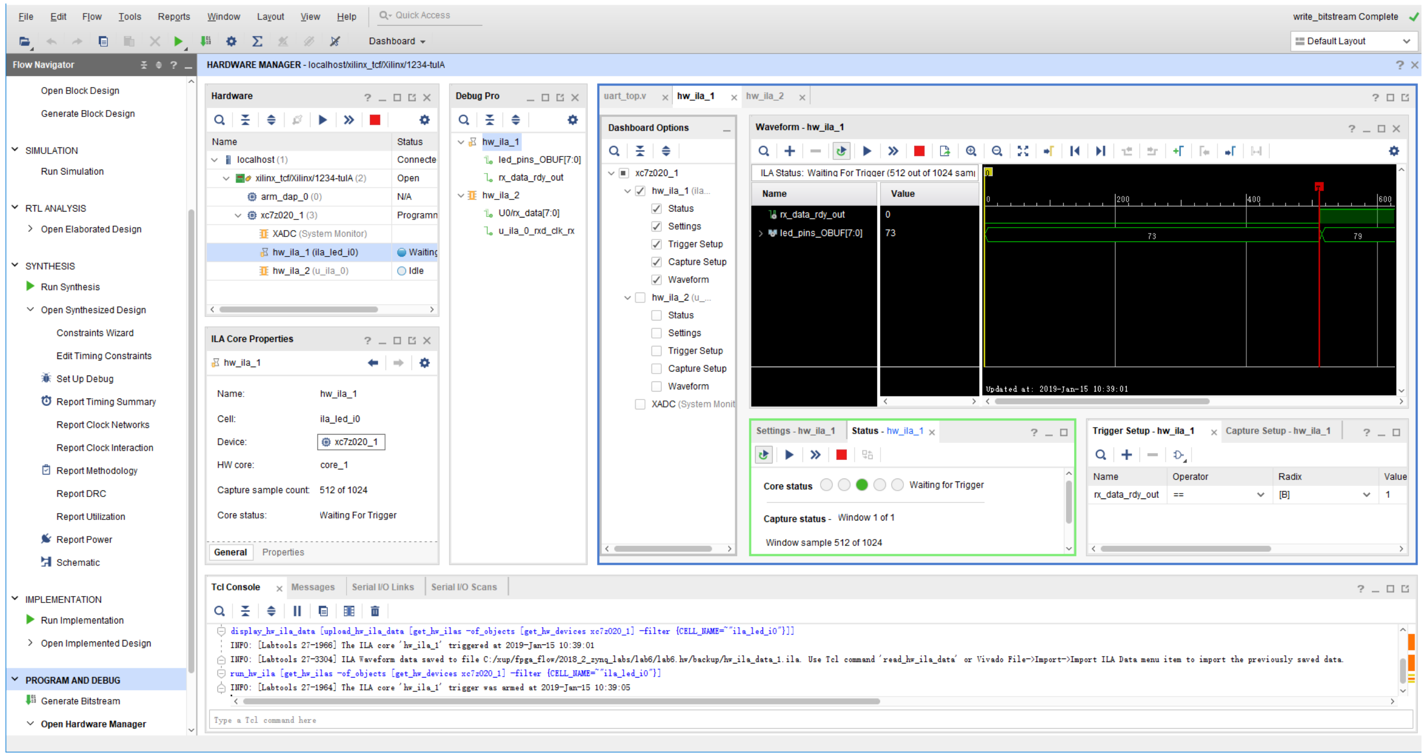


Xilinx Hardware Solutions for Debugging Designs

- Hardware debugging tool
  - Vivado logic analyzer for hardware
  - Functionally replaces need for external logic analyzer
- Now included in the Xilinx tool suite
- A single JTAG connection to the PC can be used for
  - Programming the programmable logic
  - Hardware debugging

Vivado Logic Analyzer Feature Overview

- Run-time software interface for interacting with ILA and VIO cores
- Supports simultaneous ILA waveform viewers
- Waveform formats can be saved and applied later for quick configurations



ILA Core Properties

Name:	hw_ila_1
Cell:	ila_led_i0
Device:	xc7z020_1
HW core:	core_1
Capture sample count:	512 of 1024
Core status:	Waiting For Trigger

ILA Status: Waiting For Trigger (512 out of 1024 samples)

Name	Value
rx_data_rdy_out	0
led_pins_OBUF[7:0]	73

Updated at: 2019-Jan-15 10:39:01

Core status: Waiting for Trigger
Capture status - Window 1 of 1
Window sample 512 of 1024

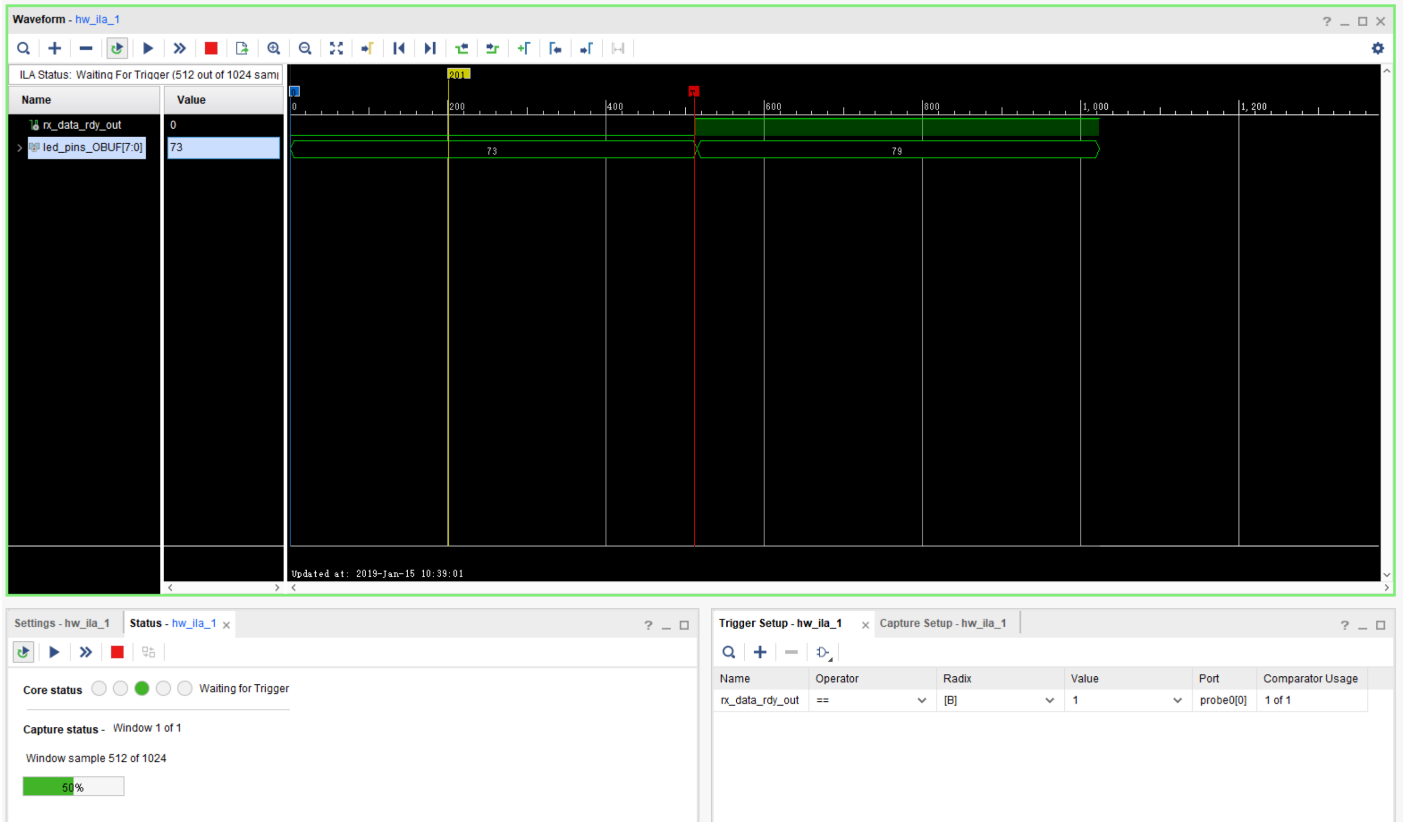
Name	Operator	Radix	Value
rx_data_rdy_out	==	[B]	1

```
display_hw_ila_data [upload_hw_ila_data [get_hw_ilas -of_objects [get_hw_devices xc7z020_1] -filter {CELL_NAME=~"ila_led_i0"}]]
INFO: [Labtools 27-1966] The ILA core 'hw_ila_1' triggered at 2019-Jan-15 10:39:01
INFO: [Labtools 27-3304] ILA Waveform data saved to file C:/xup/fpga_flow/2018_2_zynq_labs/lab6/lab6.hw/backup/hw_ila_data_1.ila. Use Tcl command 'read_hw_ila_data' or Vivado File->Import->Import ILA Data menu item to import the previously saved data.
run_hw_ila [get_hw_ilas -of_objects [get_hw_devices xc7z020_1] -filter {CELL_NAME=~"ila_led_i0"}]
INFO: [Labtools 27-1964] The ILA core 'hw_ila_1' trigger was armed at 2019-Jan-15 10:39:05
```

Vivado Logic Analyzer Feature

Common Waveform Viewer

- Shared across Vivado features
  - Simulator, Logic Analyzer, etc.
  - Reduces learning curve
  - Eases transition between features
- New functionality
  - Cursors and markers w/measure
  - Zoom mouse gestures
  - Custom colors
  - Find next/previous transition
  - Find signal name
  - Multiple radix selections
  - Integrated analog plot w/row resize



ILA Status: Waiting For Trigger (512 out of 1024 sam

Name	Value
rx_data_rdy_out	0
led_pins_OBUF[7:0]	73

Updated at: 2019-Jan-15 10:39:01

Core status Waiting for Trigger

Capture status - Window 1 of 1

Window sample 512 of 1024

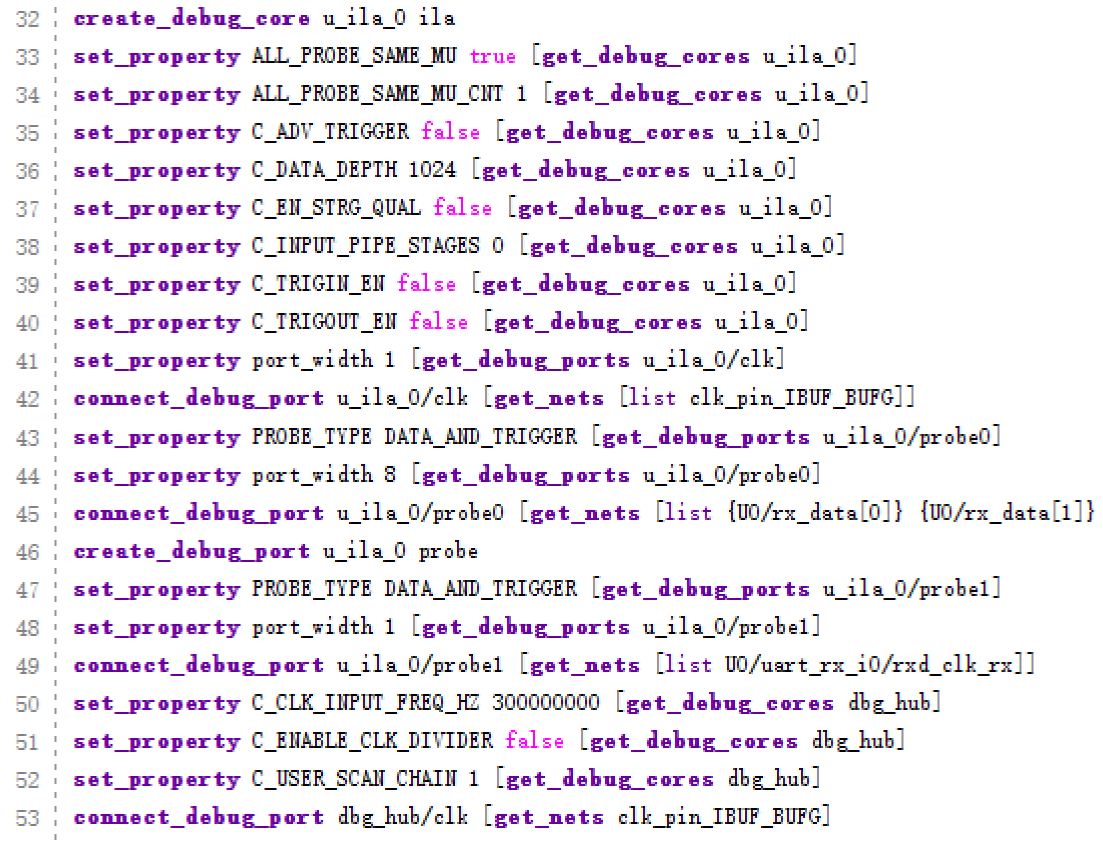
50%

Name	Operator	Radix	Value	Port	Comparator Usage
rx_data_rdy_out	==	[B]	1	probe0[0]	1 of 1

Vivado Logic Analyzer Feature

Tcl Scripting

- Enables automation of logic debug
  - Uses common Vivado Tcl engine and concepts
  - Run tests in interactive or batch mode
  - Save results for future viewing
- Allows you to create custom functions and tests
  - Create repeatable tests
  - Link custom Tcl to toolbar buttons in Vivado IDE
  - More easily integrate into custom test environments

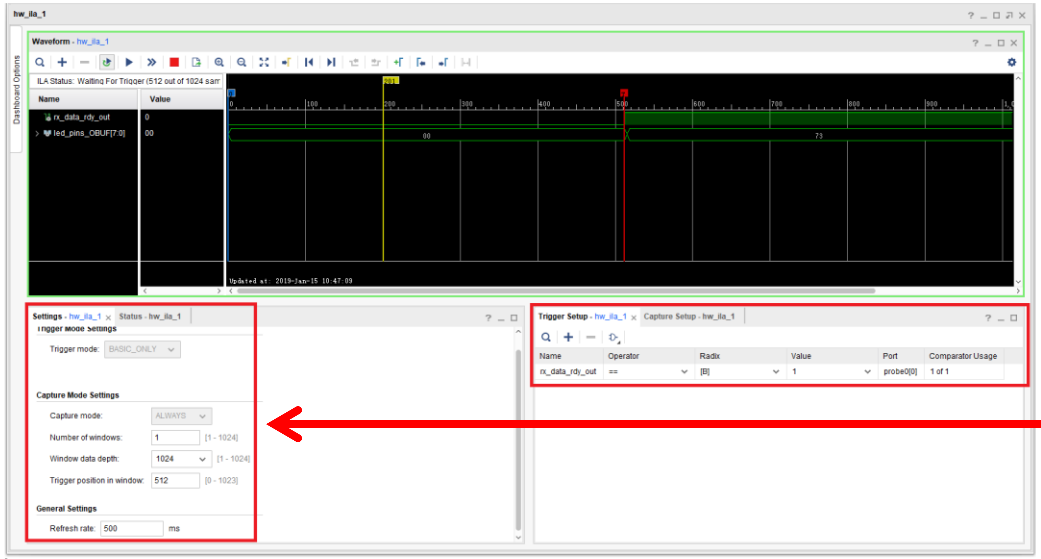


```
create_debug_core u_ila_0 ila
set_property ALL_PROBE_SAME_MU true [get_debug_cores u_ila_0]
set_property ALL_PROBE_SAME_MU_CNT 1 [get_debug_cores u_ila_0]
set_property C_ADV_TRIGGER false [get_debug_cores u_ila_0]
set_property C_DATA_DEPTH 1024 [get_debug_cores u_ila_0]
set_property C_EN_STRG_QUAL false [get_debug_cores u_ila_0]
set_property C_INPUT_PIPE_STAGES 0 [get_debug_cores u_ila_0]
set_property C_TRIGIN_EN false [get_debug_cores u_ila_0]
set_property C_TRIGOUT_EN false [get_debug_cores u_ila_0]
set_property port_width 1 [get_debug_ports u_ila_0/clk]
connect_debug_port u_ila_0/clk [get_nets [list clk_pin_IBUF_BUFG]]
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe0]
set_property port_width 8 [get_debug_ports u_ila_0/probe0]
connect_debug_port u_ila_0/probe0 [get_nets [list {U0/rx_data[0]} {U0/rx_data[1]}
create_debug_port u_ila_0 probe
set_property PROBE_TYPE DATA_AND_TRIGGER [get_debug_ports u_ila_0/probe1]
set_property port_width 1 [get_debug_ports u_ila_0/probe1]
connect_debug_port u_ila_0/probe1 [get_nets [list U0/uart_rx_i0/rxd_clk_rx]]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_pin_IBUF_BUFG]
```

Benefits of Vivado Logic Debug

Simplified Debugging

- Single trigger comparator type (a.k.a. “match unit”) per PROBE
  - All comparison types, bit values
- No ICON core instantiation required
  - Handled by auto detection and connections during implementation
- Most debug parameters are set during runtime
  - Minimize unnecessary re-implementation



ILA Status: Waiting For Trigger (512 out of 1024 sam

Name	Value
rx_data_rdy_out	0
led_pins_OBUF[7:0]	00

```
Updated at: 2019-Jan-15 10:47:09
```

Settings - hw_ila_1 | Status - hw_ila_1

Trigger Mode Settings

Trigger mode: BASIC_ONLY

Capture Mode Settings

Capture mode: ALWAYS

Number of windows: 1 [1 - 1024]

Window data depth: 1024 [1 - 1024]

Trigger position in window: 512 [0 - 1023]

General Settings

Refresh rate: 500 ms

Trigger Setup - hw_ila_1 | Capture Setup - hw_ila_1

Name	Operator	Radix	Value	Port	Comparator Usage
rx_data_rdy_out	==	[B]	1	probe0[0]	1 of 1

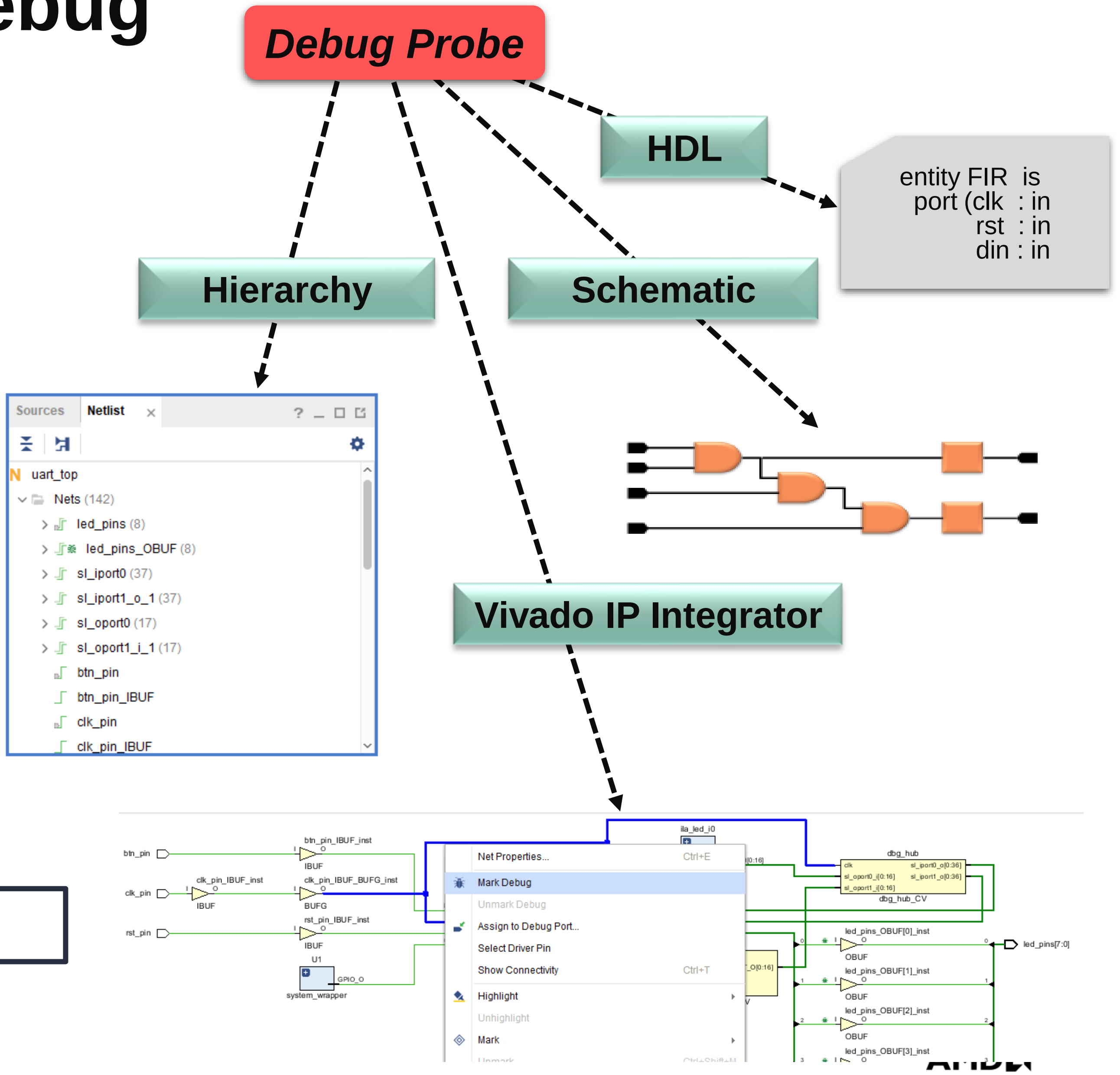
Focus on debugging the design, not on the debug core!

Benefits of Vivado Logic Debug

High-level Debugging

- Flexible, targeted probing of HDL design using MARK_DEBUG property
- Synthesized design probing in multiple views
- System-level probing inside of IP integrator view

Debug the design at the appropriate level



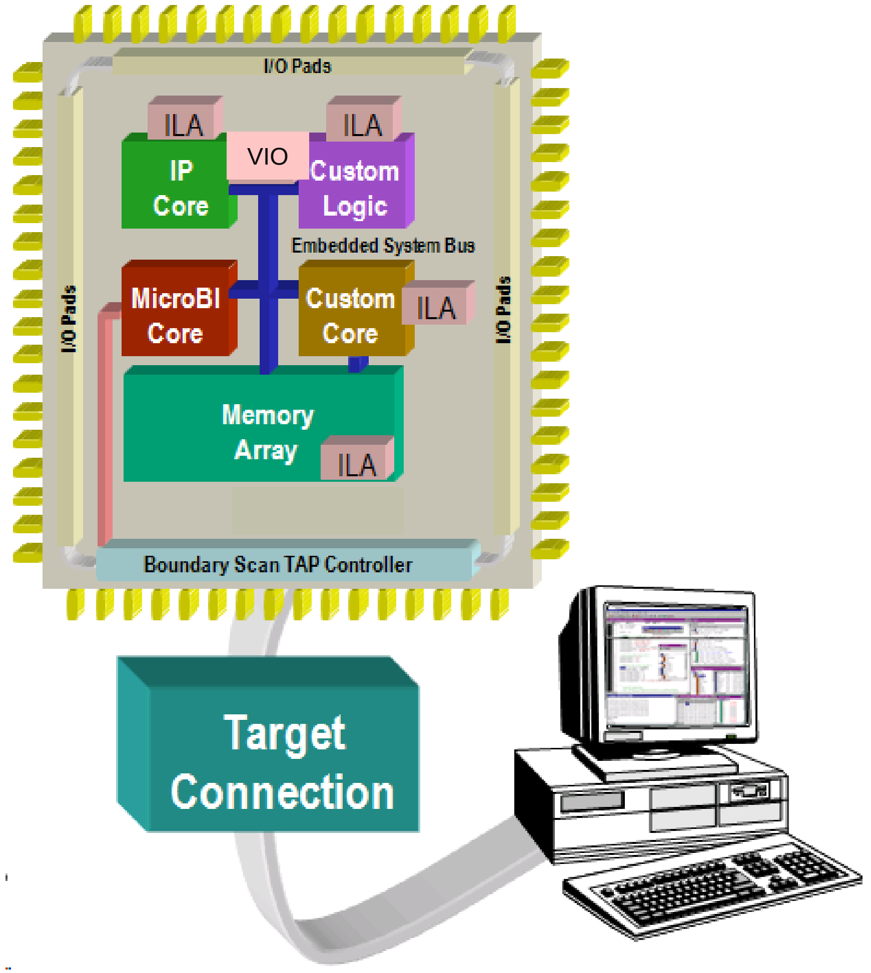
Sources | Netlist

- uart_top
  - Nets (142)
    - led_pins (8)
    - led_pins_OBUF (8)
    - sl_iport0 (37)
    - sl_iport1_o_1 (37)
    - sl_oport0 (17)
    - sl_oport1_i_1 (17)
    - btn_pin
    - btn_pin_IBUF
    - clk_pin
    - clk_pin_IBUF

Logic Debug Cores

Vivado Integrated Logic Analyzer System

- Vivado Design Suite debug cores provide internal visibility to all soft IP
  - Access to hard IP ports
  - Accesses all the internal signals, interfaces ports and nodes within the programmable logic (ILA v5.x)
  - Stimulus can be applied using the Virtual I/O core (VIO v3.x)
- Debugging occurs at, or near, system speeds
  - Debug on-chip using the system clock
- Minimize pins needed for debugging
  - Access via the JTAG interface (debug_core_hub)

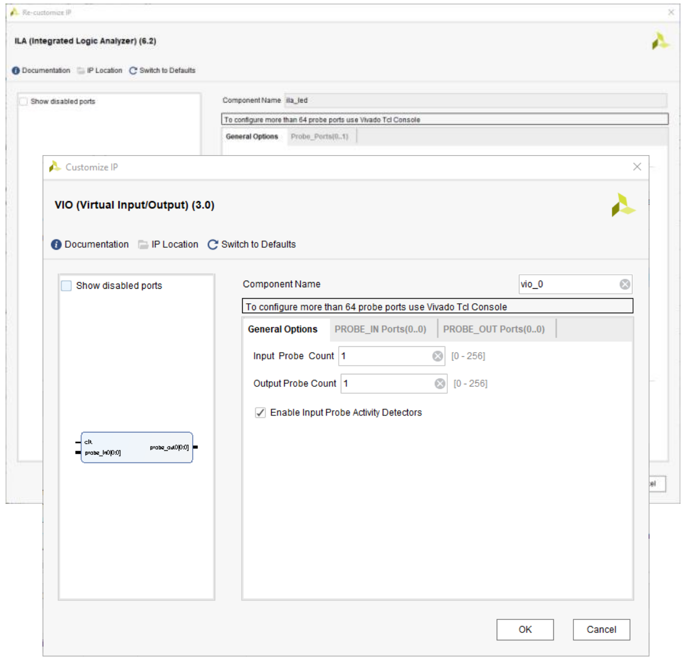


Vivado Logic Debug IP

- ILA 6.2
  - Vivado native Integrated Logic Analyzer debug IP core
  - Netlist insertion support
  - HDL instantiation support
- VIO 3.0
  - Vivado native Virtual Input / Output debug core
  - HDL instantiation

ILA 6.2 and VIO 3.0
HDL Instantiation

ILA 6.2
Netlist Insertion



Re-customize IP

ILA (Integrated Logic Analyzer) (6.2)

Show disabled ports

Component Name: ila_led

To configure more than 64 probe ports use Vivado Tcl Console

Customize IP

VIO (Virtual Input/Output) (3.0)

Show disabled ports

Component Name: vio_0

To configure more than 64 probe ports use Vivado Tcl Console

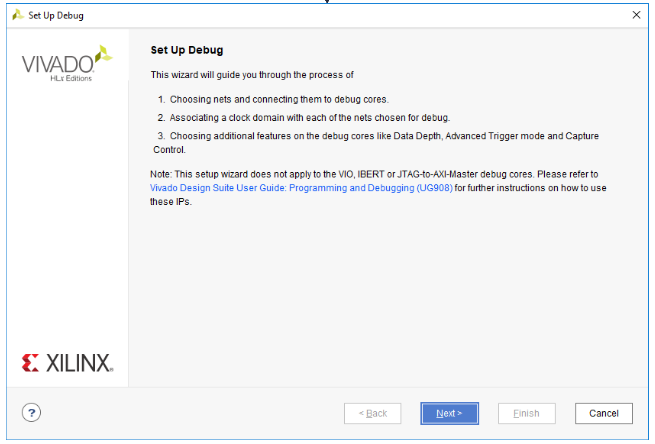
Input Probe Count: 1 [0 - 256]

Output Probe Count: 1 [0 - 256]

✓ Enable Input Probe Activity Detectors

```
ila_led ila_led_i0 (
    .clk(clk_pin), // input wire clk
    .probe0(rx_data_rdy_out), // input wire [0:0]  probe0
    .probe1(led_pins) // input wire [7:0]  probe1
);
```

Mark Debug Nets



Set Up Debug

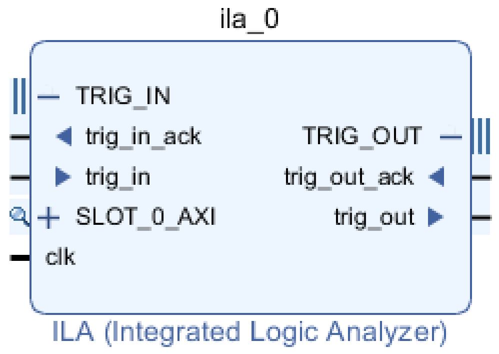
This wizard will guide you through the process of

1. Choosing nets and connecting them to debug cores.
2. Associating a clock domain with each of the nets chosen for debug.
3. Choosing additional features on the debug cores like Data Depth, Advanced Trigger mode and Capture Control.

Note: This setup wizard does not apply to the VIO, IBERT or JTAG-to-AXI-Master debug cores. Please refer to Vivado Design Suite User Guide: Programming and Debugging (UG908) for further instructions on how to use these IPs.

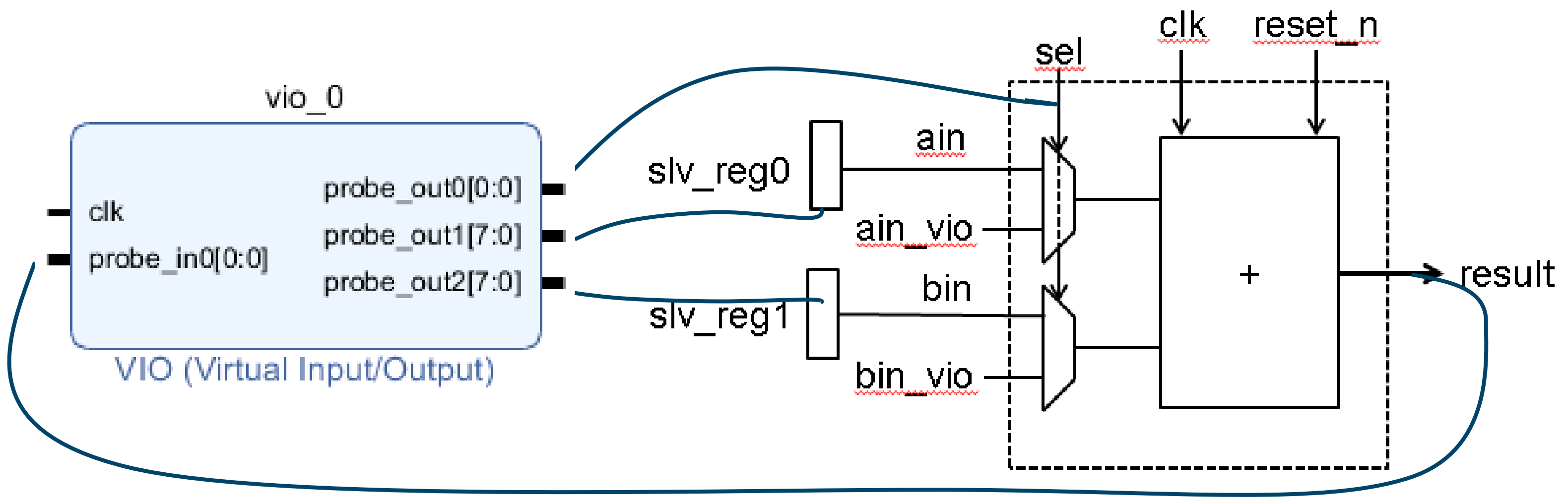
ILA Core

- Used for monitoring internal programmable logic signals and ports for post-analysis
- Multiple configurable ILA trigger units
  - Configurable trigger input widths and match types for use with different input signals types
- Separate data and trigger inputs
- Sequential triggering
- Storage qualification
- Trigger out signal for cross-probing
- Pre- and post-trigger buffering (capture data before, during, and after trigger condition is met)



VIO Core

- Support for monitoring and driving internal programmable logic signals in "real time"
- Probe input unit
- Probe output unit



Mark Debug

- Besides the IP cores insertion, Vivado logic debugging can be done by making nets in the HDL code using Mark Debug property
- VHDL syntax example

```
attribute mark_debug : string;

attribute mark_debug of char_fifo_dout: signal is "true";
```

- Verilog Syntax example

```
(* mark_debug = "true" *) wire [7:0] char_fifo_dout;
```

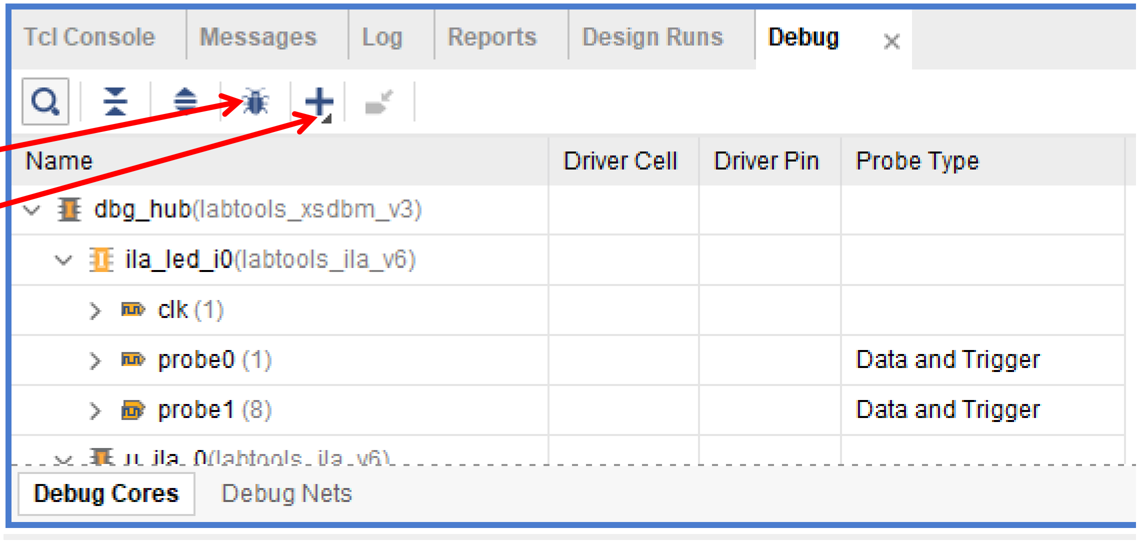
Logic Debug Probing Flows

Vivado Debug Probing Flows

- Netlist insertion flow (Highly recommended)
  - Most flexible with high predictability
  - Probing at different design levels (HDL, synthesized design, system design)
  - Compatible with various tool modes (Project, Non-project)
- HDL instantiation flow
  - Traditional flow for highest predictability, moderate flexibility
    - No longer requires ICON core instance
  - Probing at HDL design level only
  - Compatible with various tool modes (Project, Non-project)
- Netlist insertion and HDL instantiation flows can be mixed

Vivado Debug Tool Access Points

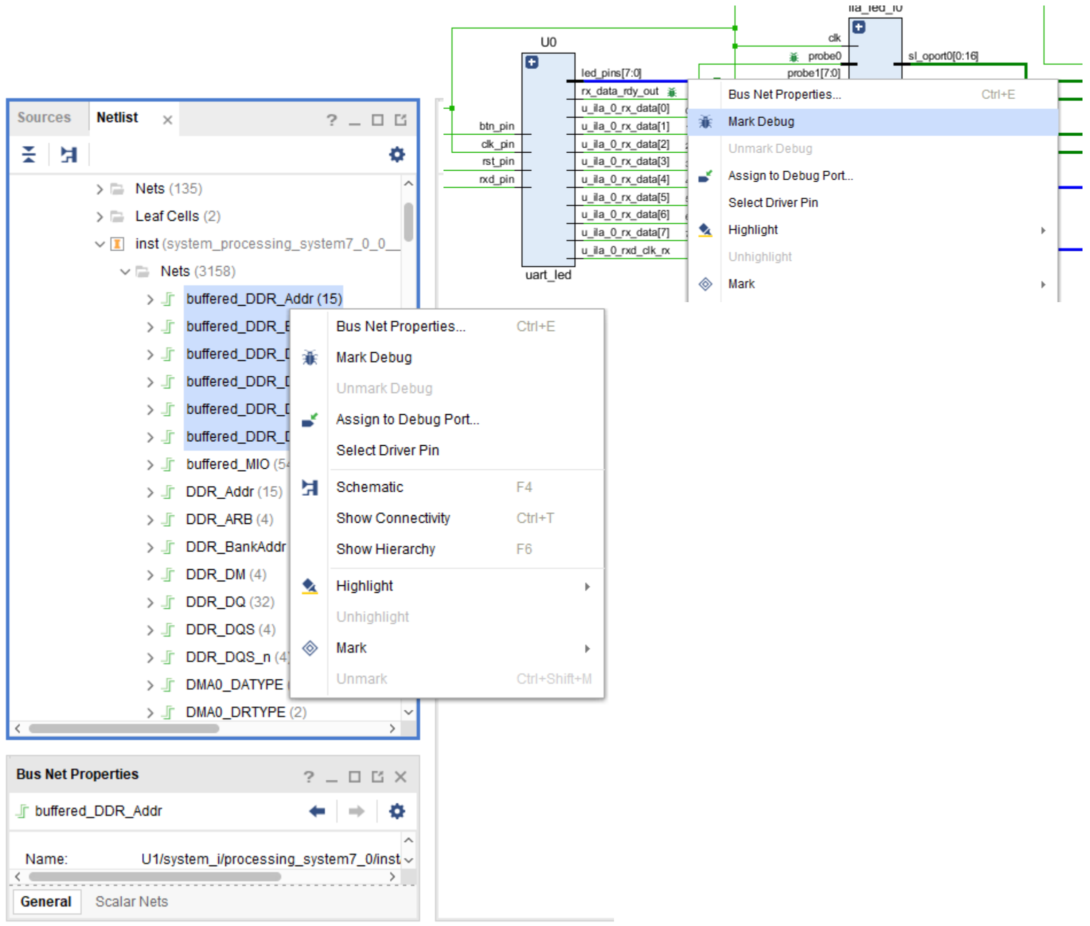
- Select Tools > Set up Debug to launch the Vivado Debug Wizard
- Debug tab appears in the Synthesized Design view
  - Click on the Set Up Debug icon to re-launch the wizard
    - Set up Debug
    - Create debug core or port



Name	Driver Cell	Driver Pin	Probe Type
dbg_hub(labtools_xsdbm_v3)			
ila_led_i0(labtools_ila_v6)			
clk (1)			
probe0 (1)			Data and Trigger
probe1 (8)			Data and Trigger
u_ila_0(labtools_ila_v6)			

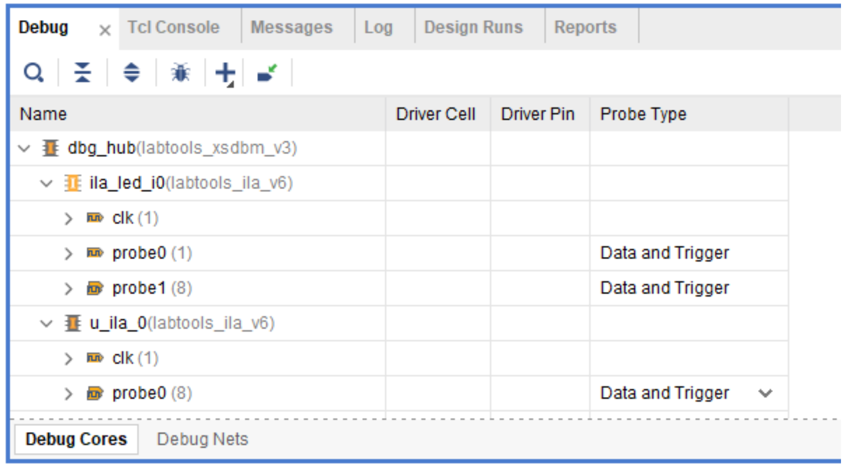
Selecting Signals to Debug

- Multiple ways to select nets in Vivado
  - Netlist view (nets folders)
    - Each level of logic hierarchy
  - Schematic
  - Find results
- Right-click the net and select Mark Debug
- Nets added to Unassigned Nets folder in the Debug tab view
  - Placeholder for probable nets prior to configuring cores
- Net name search also in the Set Up Debug Wizard

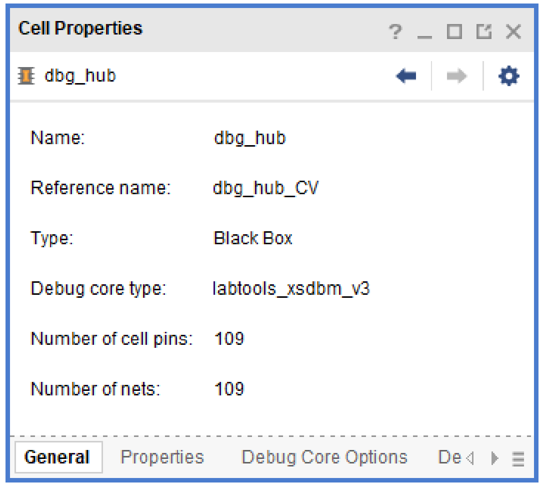


Debug Tool Configuration

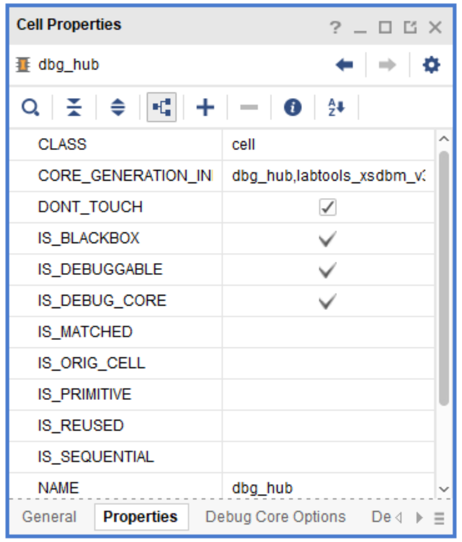
- The Vivado tool view displays core content and configuration
  - CLK, PROBE
  - Signal count
- Set options for cores and signals in the Properties view



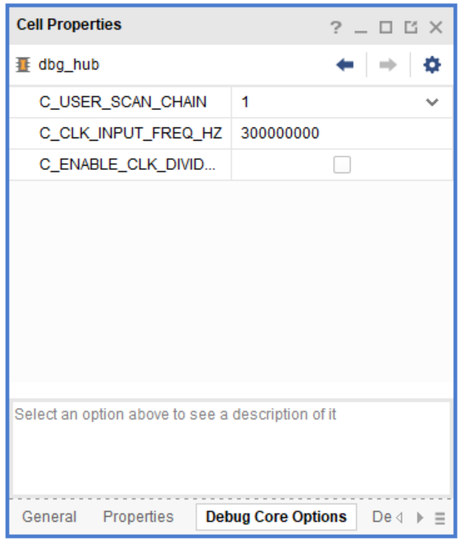
Name	Driver Cell	Driver Pin	Probe Type
dbg_hub(labtools_xsdbm_v3)			
ila_led_i0(labtools_ila_v6)			
clk (1)			
probe0 (1)			Data and Trigger
probe1 (8)			Data and Trigger
u_ila_0(labtools_ila_v6)			
clk (1)			
probe0 (8)			Data and Trigger



Name:	dbg_hub
Reference name:	dbg_hub_CV
Type:	Black Box
Debug core type:	labtools_xsdbm_v3
Number of cell pins:	109
Number of nets:	109



CLASS	cell
CORE_GENERATION_IN	dbg_hub,labtools_xsdbm_v:
DONT_TOUCH	✓
IS_BLACKBOX	✓
IS_DEBUGGABLE	✓
IS_DEBUG_CORE	✓
IS_MATCHED	
IS_ORIG_CELL	
IS_PRIMITIVE	
IS_REUSED	
IS_SEQUENTIAL	
NAME	dbg_hub



C_USER_SCAN_CHAIN	1
C_CLK_INPUT_FREQ_HZ	300000000
C_ENABLE_CLK_DIVID...	☐

Select an option above to see a description of it

Summary

Summary

- Vivado Logic debug features integration, common wave viewer, and Tcl command
- Debugging is an integral and important component of embedded system development
- The Vivado logic analyzer provides various cores to view the inside of a design—from individual signals to bus-level activities
  - ILA core
  - VIO core
- The Debug Configuration Wizard simplifies hardware connections and logic analyzer peripheral inclusion
- Three Vivado logic debug flows
  - HDL Instantiation
  - Netlist insertion
  - Mark Debug in netlist, schematic, and HDL

AMD
XILINX

Thank You

Disclaimer and Attribution