

# Design Analysis Using Tcl Commands Demo Script

## Introduction

In this demonstration, you will perform design analysis by finding objects, object properties, and object connectivity by using Tcl commands.

### Preparation:

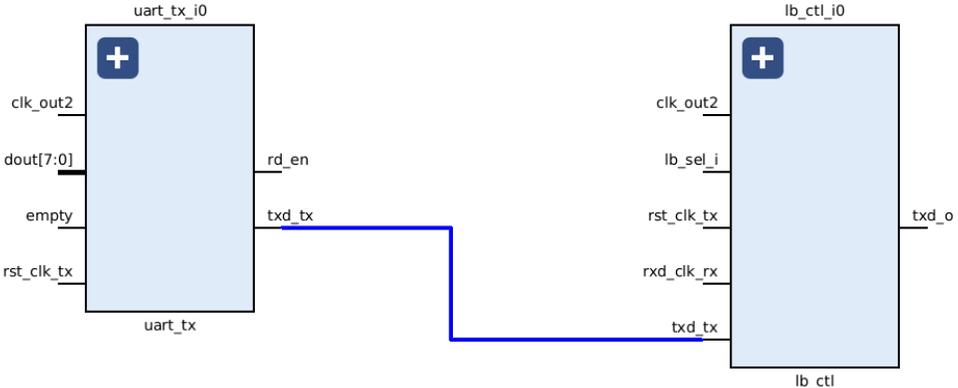
- Necessary project files are located in the following directory:  
`$TRAINING_PATH/Tcl_Dsgn_Analysis/demo/KCU105/verilog`
- Required hardware: None

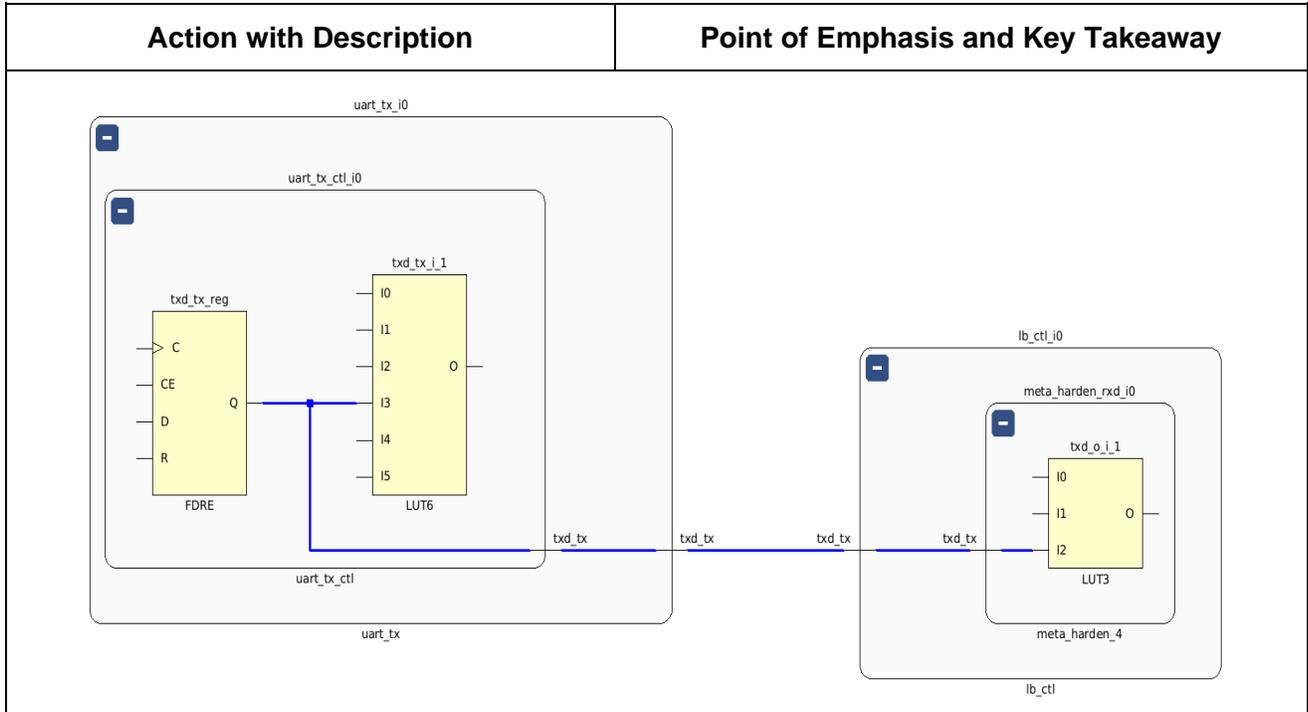
## Finding Objects

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>• Launch the Vivado™ Design Suite.</li> <li>• Unzip the project using the Tcl Console:  <pre>exec unzip \$:env(TRAINING_PATH)/Tcl_Dsgn_Analysis/demo/KCU105/verilog.zip -d \$:env(TRAINING_PATH)/Tcl_Dsgn_Analysis/demo/KCU105/verilog</pre> </li> <li>• Open the <b>wave_gen.xpr</b> project from the following directory:  <code>\$TRAINING_PATH/Tcl_Dsgn_Analysis/demo/KCU105/verilog</code></li> </ul>	<ul style="list-style-type: none"> <li>• You can easily open an existing Vivado IDE project via the Getting Started page.</li> </ul>
<ul style="list-style-type: none"> <li>• Open the implemented design.</li> </ul>	<ul style="list-style-type: none"> <li>• You can open the implemented design by using: <ul style="list-style-type: none"> <li>• The Flow Navigator</li> <li>• The Tcl Console</li> <li>• The horizontal toolbar</li> </ul> </li> <li>• The implemented design shows how the logic will be placed on resources. It allows you to generate various clock and timing report.</li> </ul>
<ul style="list-style-type: none"> <li>• Is the hierarchy maintained in the implemented netlist? <ul style="list-style-type: none"> <li>• Yes, hierarchy is maintained. You can view the netlist hierarchy in the Netlist window (<b>Window &gt; Netlist</b>). You can also review the hierarchy settings for the project in the synthesis settings.</li> </ul> </li> </ul>	

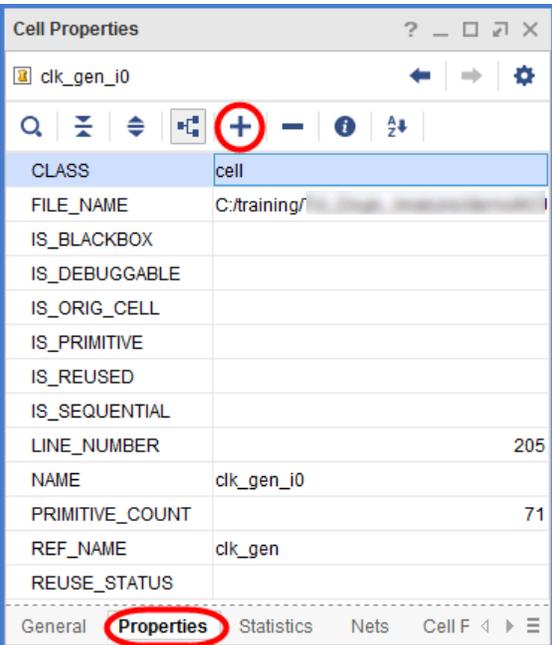
Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Enter <code>get_cells</code> in the Tcl Console to get all the cells in the design within the current level of the hierarchy.</li> </ul>	<ul style="list-style-type: none"> <li>The Tcl interface provides commands for getting objects. Some of them include: <ul style="list-style-type: none"> <li><code>get_cells</code>, <code>get_nets</code>, <code>get_ports</code>, <code>get_pins</code></li> </ul> </li> <li>With no options, these commands return a list of all objects at the current level of hierarchy. <ul style="list-style-type: none"> <li>The current level of hierarchy is the top level of the design unless changed by the user.</li> </ul> </li> <li><b>Note:</b> As you enter <code>get_</code> in the Tcl Console, the Vivado Design Suite lists all the commands supported by it.</li> </ul>
<ul style="list-style-type: none"> <li>What is the output of the <code>get_cells</code> command? Is this expected? How many cells are returned by the <code>get_cells</code> command? <ul style="list-style-type: none"> <li>The <code>get_cells</code> command returns the list of cells at the current level of hierarchy (in this case <code>top_level</code>). This list does not include any cells from the hierarchical cells (sub-blocks) under the top level.</li> <li>Enter the <code>llength [get_cells]</code> command to determine the number of cells returned by the <code>get_cells</code> command.</li> </ul> </li> </ul>	
<ul style="list-style-type: none"> <li>Enter <code>get_cells -hierarchical</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>This command lists all the cells in the design.</li> <li>The <code>-hierarchical</code> option can be used with cells, pins, or nets.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_cells uart_rx</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>If no object of that name given in the command exists, the command will return a null list with a warning message.</li> <li>The <code>uart_rx</code> cell is taken as an example here. You can use any other cell.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_cells uart_rx_i0</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>This returns the cell <code>uart_rx_i0</code> from the current level of the hierarchy.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_cells OBUF_dac_c*_n</code> in the Tcl Console to use a wildcard to match characters.</li> <li>Enter <code>get_cells OBUF_dac_c?_n</code>.</li> </ul>	<ul style="list-style-type: none"> <li>The names given in the command can have wildcards.</li> </ul>
<ul style="list-style-type: none"> <li>How did the wildcards <code>*</code> and <code>?</code> match the characters differently in each case? <ul style="list-style-type: none"> <li>The wildcard <code>*</code> matches any number of characters, so the command with <code>*</code> returns two cells <code>OBUF_dac_clr_n</code> and <code>OBUF_dac_cs_n</code>.</li> <li>The wildcard <code>?</code> matches a single character, so the command using <code>?</code> returns <code>OBUF_dac_cs_n</code>.</li> </ul> </li> </ul>	

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Enter <code>get_cells -regexp {[Rr]st.*}</code> in the Tcl console.</li> </ul>	<ul style="list-style-type: none"> <li>Using the <code>-regexp</code> option specifies that the search patterns are written as regular expressions.</li> <li>You can add <code>".*"</code> to the beginning or end of a search string to match any number of characters.</li> <li>This command searches for strings starting with <code>Rst</code> or <code>rst</code> and has any number of characters after that.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_cells -hierarchical uart_baud_gen_*</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>By default, the <code>get_*</code> commands only search for objects starting at the current level of hierarchy.</li> <li>The option <code>-hierarchical</code> will search at all levels of the hierarchy.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_pins samp_gen_i0/rst_clk_tx</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>Pin names are based on the instance they belong to.</li> <li>When searching for pins, you must use the hierarchy separator to separate the instance name and the pin name patterns.</li> <li>The command returns the <code>clk_tx</code> pin of the <code>samp_gen_i0</code> instance.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_pins -hierarchical */C</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>This command returns the C pins of any cell in the design as the wild card <code>*</code> is used to match pin names.</li> <li>Note that the pin names look hierarchical, but they are not.                             <ul style="list-style-type: none"> <li>For example, pin C of the instance <code>signal_meta_reg</code> is named as <code>signal_meta_reg/C</code>.</li> <li>This does not mean that the C pin is inside the hierarchy of the instance <code>signal_meta_reg</code>.</li> <li>The final <code>/</code> in a pin name is not a hierarchy separator, it is part of the pin name.</li> </ul> </li> </ul>

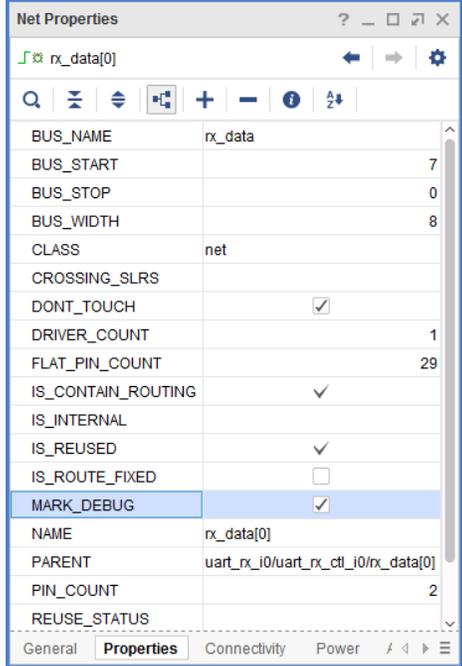
Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>• Enter <code>get_nets txd_tx</code> in the Tcl Console.</li> <li>• Enter <code>show_schematic [get_nets txd_tx]</code> in the Tcl Console.</li> <li>• Enter <code>select_objects [get_nets txd_tx]</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>• The <code>txd_tx</code> net is used as an example here, you can choose any other net as well.</li> <li>• The <code>get_nets txd_tx</code> command returns the net <code>txd_tx</code>, if present in the current level of hierarchy.</li> <li>• The <code>show_schematic</code> command creates a Schematic view containing the specified design objects.</li> <li>• The <code>select_object</code> command highlights the selected object in the GUI (including in the Schematic view).</li> </ul>
	
<ul style="list-style-type: none"> <li>• Enter <code>get_nets txd_tx -segments</code> to return all the segments of the net <code>rx_data[0]</code>.</li> <li>• Enter <code>show_schematic [get_nets txd_tx -segments]</code> in the Tcl Console.</li> <li>• Enter <code>select_objects [get_nets txd_tx -segments]</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>• A single logical net can have multiple segments. <ul style="list-style-type: none"> <li>• Within each hierarchical object, the net can have a different segment.</li> <li>• Each segment is a different net object.</li> </ul> </li> <li>• The <code>get_nets</code> command with <code>-segments</code> option can be used to find the segments of a net.</li> <li>• One of the segments will be considered the parent segment. All the other segments will have the <code>PARENT</code> property set to refer to this segment. Accessing the properties will be covered in the next part of the demo.</li> </ul>

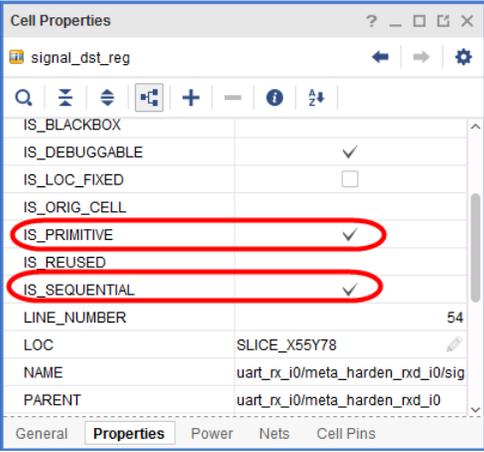


## Object Properties

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Select any cell from the netlist (clk_gen_i0, for example).</li> <li>View the class type and other properties from the Cell Properties window.</li> </ul> 	<ul style="list-style-type: none"> <li>Object properties can be seen by selecting an object. <ul style="list-style-type: none"> <li>The selection can be done in a schematic or any object list.</li> </ul> </li> <li>Only the most common properties and properties that are at their non-default value are shown in the Cell Properties window. Additional defined properties can be added to the view by right-clicking in the window and selecting <b>Add Properties</b> or by clicking the <b>+</b> sign. A popup window will show all possible properties that can be added to the view.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>report_property [get_cells clk_gen_i0]</code> in the Tcl Console.</li> <li>Enter <code>report_property -all [get_cells clk_gen_i0]</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>The <code>report_property</code> command returns the property name, property type, and property value for all of the properties on a specified object or class of objects.</li> <li>You can specify objects for <code>report_property</code> by using the <code>get_*</code> series of commands to get a specific object.</li> <li>The <code>-all</code> option returns all of the properties for an object even if the property value is not currently defined.</li> </ul>

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>• Enter <code>list_property -class net</code> in the Tcl Console to list all the properties of <code>net</code>.</li> </ul>	<ul style="list-style-type: none"> <li>• This command will list any property that can be had by any object of class-type <b>net</b>.</li> <li>• All objects of a given class have the same set of properties.</li> <li>• You can use <code>report_property</code> instead of <code>list_property</code> to get the description and type of the property.</li> </ul>
<ul style="list-style-type: none"> <li>• Enter the following commands in the Tcl Console and observe the output:             <ul style="list-style-type: none"> <li>• <code>get_property bus_width [get_nets rx_data[0]]</code> <ul style="list-style-type: none"> <li>▪ This command returns the value of the property <code>bus_width</code> of the object <code>net rx_data[0]</code>.</li> </ul> </li> <li>• <code>get_property slew [get_nets rx_data[0]]</code> <ul style="list-style-type: none"> <li>▪ This command returns nothing. Why?</li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• All property values can be obtained via the <code>get_property</code> Tcl command.</li> <li>• <code>get_property</code> gets the current value of the named property from the specified object or objects.</li> <li>• If the property is not currently assigned to the object, or is assigned without a value, then the <code>get_property</code> command returns nothing, or the null string.</li> </ul>

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Enter the following command in the Tcl Console:                             <ul style="list-style-type: none"> <li><code>set_property MARK_DEBUG 1 [get_nets {rx_data[0]}]</code> <ul style="list-style-type: none"> <li>This command sets the property <code>mark_debug</code> of the net <code>rx_data[0]</code>.</li> </ul> </li> </ul> </li> <li>How can you verify whether the property is set or not?                             <ul style="list-style-type: none"> <li>Use the <code>get_property</code> command:                                     <pre>get_property MARK_DEBUG [get_nets {rx_data[0]}]</pre> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Properties that are not read only can be set via the <code>set_property</code> Tcl command.</li> <li>The <code>set_property</code> command sets the property on object.</li> <li>Properties can also be modified directly in the Attributes pane of the Instance Properties window in the GUI by right-clicking the particular net and selecting Net Properties.</li> </ul>  <ul style="list-style-type: none"> <li>Note that the property you set through the Tcl command is reflected in the GUI. This shows the interoperability between the Tcl interface and the GUI.</li> </ul>

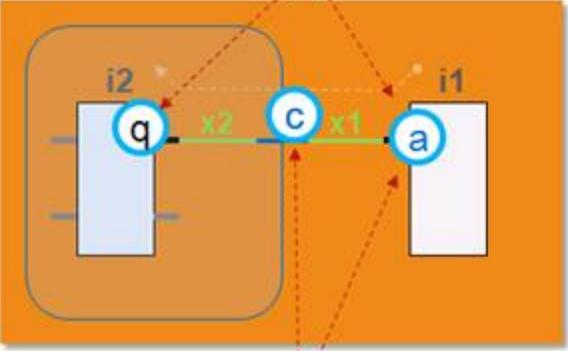
Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Select any primitive cell (uart_rx_i0/meta_harden_rxd_i0/Leaf Cells/signal_dst_reg, for example) from the Netlist window and view the properties from the Cell Properties window.</li> </ul>  	<ul style="list-style-type: none"> <li>All objects have a <b>CLASS</b> property, which determines which type of object it is.</li> <li><b>NAME</b> is the full hierarchical name of the object.</li> <li><b>IS_PRIMITIVE</b> indicates that it is a primitive cell (as opposed to a hierarchical cell).</li> <li><b>IS_SEQUENTIAL</b> indicates that it is a clocked element.</li> <li><b>REF_NAME</b> is the name of the library cell, module, or entity of which this object is an instance.</li> <li>The library cell attributes (i.e., MMCM, Block RAM, ...) are all properties.</li> <li>Similarly, you can select any hierarchical cell.</li> <li>A hierarchical cell is a module or block that contains one or more additional levels of logic and eventually concludes at primitive cells.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>report_property [get_ports led_pins[0]]</code> in the Tcl Console.</li> </ul> <p><b>Note:</b> You can open the elaborated design and select the port from the schematic and view the properties from the Property window</p>	<ul style="list-style-type: none"> <li><b>DIRECTION</b> is the direction of the port.</li> <li>I/O attributes are all properties of the port:             <ul style="list-style-type: none"> <li>IOSTANDARD, SLEW, DRIVE, PULLUP, PULLDOWN, KEEPER</li> </ul> </li> <li><b>PACKAGE_PIN</b> is the FPGA pin used for the port.</li> </ul>

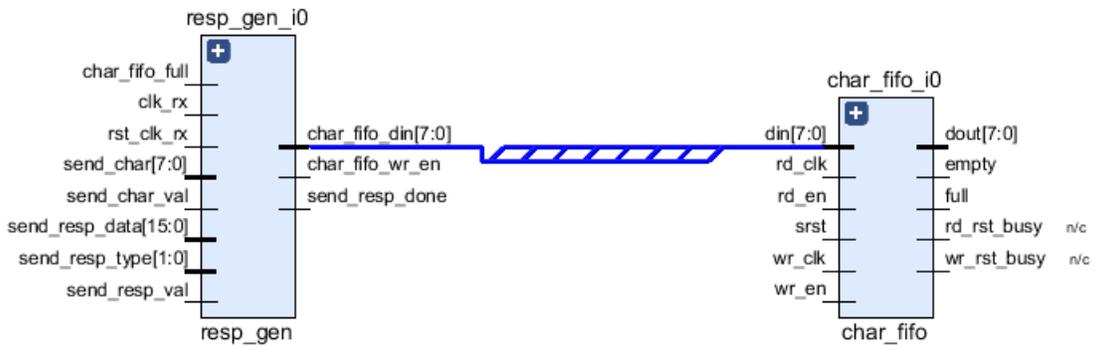
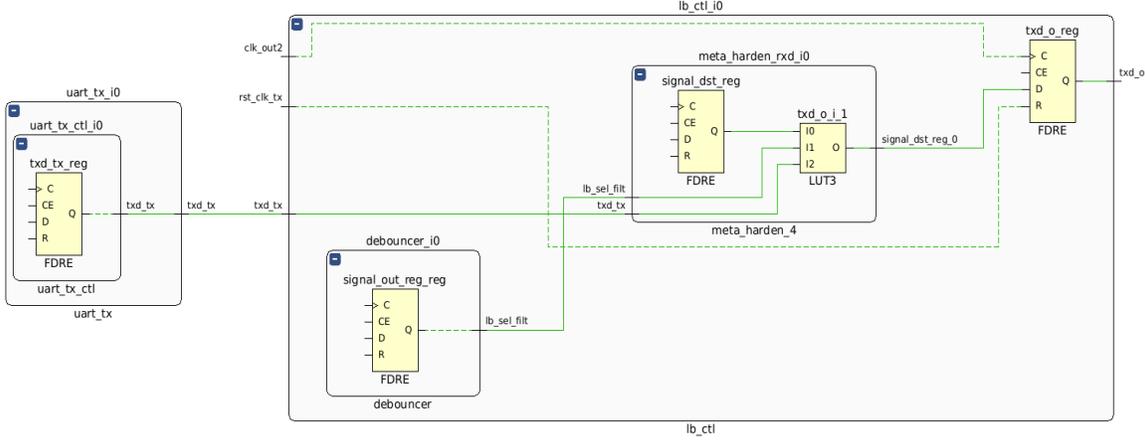
Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Select any pin from the schematic and view its properties or use the <code>report_property</code> and <code>get_pins</code> commands to view the properties of the pin. <ul style="list-style-type: none"> <li><code>report_property [get_pins rst_gen_i0/rst_clk_rx]</code></li> <li>This command returns the properties of the pin <code>rst_gen_i0/rst_clk_rx</code>.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li><b>DIRECTION</b> indicates if this pin is an IN, OUT, or INOUT.</li> <li><b>IS_LEAF</b> indicates if this is the pin of a leaf cell or a pin of a hierarchical cell.</li> <li><b>IS_CLOCK</b> indicates if this is a clock pin to a clocked object.</li> <li><b>IS_RESET, IS_PRESET</b> indicate if this pin is a reset or preset pin of a clocked object.</li> <li><b>IS_ENABLE</b> indicates if this pin is a clock enable pin of a clocked object.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>create_property -type bool MY_PROPERTY cell</code> in the Tcl Console.</li> <li>Creates the property <code>MY_PROPERTY</code> for all objects of the class <code>cell</code>, which can accept a Boolean value.</li> </ul>	<ul style="list-style-type: none"> <li>New properties can be added to a class of objects with the <code>create_property</code> Tcl command.</li> <li><code>create_property -type &lt;type&gt; &lt;name&gt; &lt;class&gt;</code> <ul style="list-style-type: none"> <li><code>&lt;type&gt;</code> can be <code>string</code>, <code>int</code>, <code>long</code>, <code>double</code>, or <code>bool</code>.</li> <li><code>&lt;class&gt;</code> can be <code>cell</code>, <code>pin</code>, <code>net</code>, or <code>port</code>.</li> <li><code>&lt;name&gt;</code> is the name of the new property to be created.</li> </ul> </li> <li>Once added to a class, the new property can be used in the same way as other properties.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>filter [get_ports] {DIRECTION == IN}</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>Properties can be used to filter lists of objects with the <code>filter</code> Tcl command.</li> <li><code>filter &lt;objects&gt; &lt;filter_expression&gt;</code> <ul style="list-style-type: none"> <li><code>&lt;objects&gt;</code> is a list of objects to filter.</li> <li><code>&lt;filter_expression&gt;</code> is the filter expression to use.</li> <li>Returns the list of objects from the <code>&lt;objects&gt;</code> list that match the <code>&lt;filter_expression&gt;</code>.</li> </ul> </li> </ul>

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>• Enter <code>get_cells -hierarchical -filter { PRIMITIVE_TYPE =~ *MMCME3_ADV* }</code> in the Tcl Console.</li> <li>• This command returns cells with the <code>primitive_type</code> property matching <code>*MMCME3_ADV*</code>.</li> </ul>	<ul style="list-style-type: none"> <li>• You can also filter a list of objects based on the property with the <code>-filter</code> option of the <code>get_*</code> command.</li> <li>• <code>get_cells -filter &lt;filter_expression&gt;</code> <ul style="list-style-type: none"> <li>• All options of the <code>get_cells</code> command can be used (<code>-hierarchy, ...</code>)</li> </ul> </li> <li>• Filter expressions are strings. They are usually enclosed in <code>{}</code>, but <code>"</code> can be used as well.</li> </ul>
<ul style="list-style-type: none"> <li>• Select <b>Edit &gt; Find</b>.</li> <li>• Select the cells in the Find section.</li> <li>• Select <b>PRIMITIVE_TYPE is CLOCK.PLL.MMCME3_ADV</b> in the Properties section.</li> <li>• Click <b>OK</b> to find the objects matching the search pattern.</li> </ul>	<ul style="list-style-type: none"> <li>• You can use the <b>Edit &gt; Find</b> command or the <code>&lt;Ctrl + F&gt;</code> keyboard shortcut to search for design objects.</li> <li>• The command bar in the Find window shows the Tcl command that is run to populate the Find Results window.</li> </ul>

## Object Connectivity

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Select <b>wave_gen &gt; Nets &gt; clk_pin_p</b> from the Netlist window.</li> <li>Press <b>&lt;F4&gt;</b> to show the schematic of the clk_pin_p net.</li> </ul>	<ul style="list-style-type: none"> <li>Note the pins and cells connected to the net.</li> <li>How can you get pins connected to a particular net?</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>get_pins -of_objects [get_nets clk_pin_p]</code> in the Tcl Console.           <ul style="list-style-type: none"> <li>This command returns the pins connected to the net clk_pin_p. Note that this net is connected to a port at one end and a pin at another end. Hence it results in only one pin name; i.e., clk_gen_i0/clk_pin_p.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Objects within the database are related to each other to form the connectivity of the netlist.</li> <li>This connectivity can be accessed by using the Tcl <code>get_*</code> commands.</li> <li>The <code>-of_objects</code> option returns the list of objects connected to the specified object. The <code>-of</code> option is the same as <code>-of_objects</code>.</li> <li>This enables you to find objects based on netlist connectivity.</li> </ul>

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Enter <code>get_pins -of_objects [get_nets clk_pin_p] -leaf</code> in the Tcl Console.</li> <li>How was the output different from the previous command?</li> </ul>	<ul style="list-style-type: none"> <li>When finding the pins connected to a net that crosses a hierarchical boundary, you can find either the pins of the hierarchical objects or the pins of the primitive objects. This is controlled by the use of the <code>-leaf</code> option.</li> <li>Without <code>-leaf</code>, the hierarchical objects pins will be returned.</li> <li>With <code>-leaf</code>, the pins of the primitive objects will be returned, regardless of where they are in the hierarchy.</li> </ul> <div style="text-align: center;"> <pre>get_pins -of [get_nets x1] -leaf</pre>  <pre>get_pins -of [get_nets x1]</pre> </div> <ul style="list-style-type: none"> <li>You can expand the cells from the schematic of <code>clk_pin_p</code> to understand the output.</li> </ul>
<ul style="list-style-type: none"> <li>Select <b>wave_gen &gt; Nets &gt; char_fifo_din</b> from the Netlist window and press <b>&lt;F4&gt;</b> to view the schematic.</li> </ul>	<ul style="list-style-type: none"> <li>What would be the Tcl command to get cell <code>resp_gen_i0</code> from the cell <code>char_fifo_i0</code> based on object connectivity?</li> </ul>

Action with Description	Point of Emphasis and Key Takeaway
	
<ul style="list-style-type: none"> <li>Multiple <code>get_*</code> commands can be nested with the <code>-of_objects</code> option to perform arbitrarily complex searches through the design database.</li> </ul>	<ul style="list-style-type: none"> <li>Apart from <code>resp_gen_i0</code>, what else did the command return? And why?                     <ul style="list-style-type: none"> <li>The <code>get_nets</code> in the command would return more than one object, and hence the resulting list of objects would include far more than just <code>resp_gen_i0</code>.</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>Select <b>wave_gen &gt; lb_ctl_i0 &gt; leaf_cells &gt; txd_o_reg</b> from the Netlist window and view the schematic.</li> </ul>	<ul style="list-style-type: none"> <li>Using the schematic, what are the fan_in objects of pin D of the <code>txd_o_reg</code> cell?                     <ul style="list-style-type: none"> <li>Double-click the cell pin D to trace the source.</li> <li>Double-click the three inputs of the LUT3.</li> </ul> </li> </ul>
	

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"> <li>Enter <code>all_fanin [get_pins lb_ctl_i0/txd_o_reg/D]</code> in the Tcl Console.</li> <li>If you want to visualize the output of the above command on multiple lines, you can use the built-in command <code>join</code> to insert a "newline" character in between each element of the list returned by the command                             <ul style="list-style-type: none"> <li><code>join [all_fanin [get_pins lb_ctl_i0/txd_o_reg/D]] \n</code></li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>The <code>all_fanin</code> command returns a list of pins that can combinatorially reach the specified pin, port, or net.</li> </ul>
<ul style="list-style-type: none"> <li>Use the <code>select_objects</code> command to select the output of the above command in the GUI:                             <ul style="list-style-type: none"> <li><code>select_objects [all_fanin [get_pins lb_ctl_i0/txd_o_reg/D]]</code></li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Did the command select all the pins you see in the schematic?                             <ul style="list-style-type: none"> <li>You can observe that the <code>uart_tx_i0/uart_tx_ctl_i0/txd_tx_reg/Q</code> pin is not highlighted as this pin is not in the same level of hierarchy as <code>lb_ctl_i0/txd_o_reg/D</code>.</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>select_objects [all_fanin [get_pins lb_ctl_i0/txd_o_reg/D] -flat -startpoints_only]</code> in the Tcl Console.</li> </ul>	<ul style="list-style-type: none"> <li>When the <code>-startpoints_only</code> option is used, none of the intermediate points in the fan-in network are returned. This option can be used to identify the primary driver(s) of the sinks.</li> <li>Similarly, the <code>-only_cells</code> option returns only the cell objects that are in the fan-in path of the specified sinks. This does not return pins or ports.</li> </ul>
<ul style="list-style-type: none"> <li>Enter <code>all_fanout [get_pins lb_ctl_i0/txd_o]</code> in the Tcl console.</li> </ul>	<ul style="list-style-type: none"> <li>The <code>all_fanout</code> command returns a list of all objects that can be combinatorially reached from a specified pin, port, or net.</li> <li>Has similar options as <code>all_fanin</code>.</li> <li>Use <code>-endpoints_only</code> to restrict it to sequential cells.</li> <li>Use different options as in <code>all_fanin</code> and observe the results.</li> </ul>
<ul style="list-style-type: none"> <li>Close the implemented design and exit the Vivado Design Suite.</li> </ul>	

## Summary

This demonstration illustrated how to find objects, and how to observe objects properties and object connectivity by using Tcl commands.

References:

- Supporting materials
  - *Vivado Design Suite Properties Reference Guide* (UG912)
  - *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
  - *Vivado Design Suite Tcl Command Reference Guide* (UG835)