Synthesis

Objectives

- **After completing this module, you will be able to:**
  - Elaborate a design and perform analysis
  - Make basic timing constraints with the Constraints viewer
  - Use the `check_timing` report to verify constraint coverage of your design
  - Build useful design reports that will help you avoid the most common design mistakes and assure design success
  - Use the `clock_interaction` report to verify constraint coverage on datapaths between clock domains

XILINX

Outline

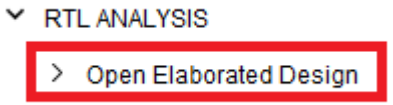
- *Elaboration*
- Synthesis
- Basic Timing Constraints
- Synthesis Reports
- Summary

Elaboration

- **Elaboration is the RTL optimization to an FPGA technology**
- **Vivado IDE allows designers to import and manage RTL sources**
  - Verilog, System Verilog, VHDL, NGC, or testbenches
- **Create and modify sources with the RTL Editor**
  - Cross-selection between all the views
- **Sources view**
  - Hierarchy view: Display the modules in the design by hierarchy
  - Libraries view: Display sources by category

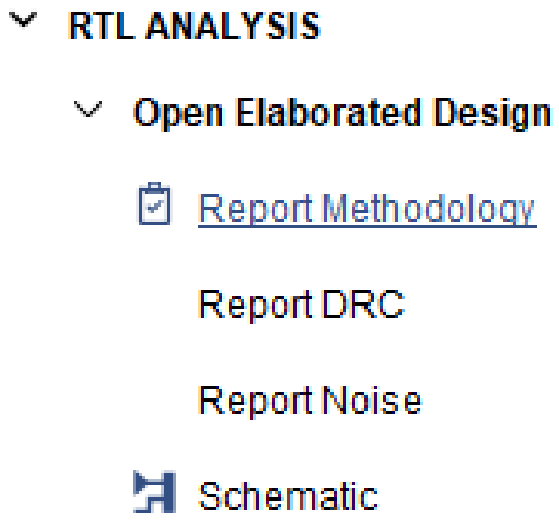
Elaboration and Analysis

- **In a RTL based design, elaboration is the first step**
- **Click on Open Elaborated Design under RTL Analysis to**
  - Compile the RTL source files
  - Load the RTL netlist for interactive analysis
- **You can check RTL structure, syntax, and logic definitions**
- **Analysis and reporting capabilities include:**
  - RTL compilation validation and syntax checking
  - Netlist and schematic exploration
  - Design rule checks
  - Early I/O pin planning using an RTL port list
  - Ability to select an object in one view and cross probe to the object in other views, including instantiations and logic definitions within the RTL source files



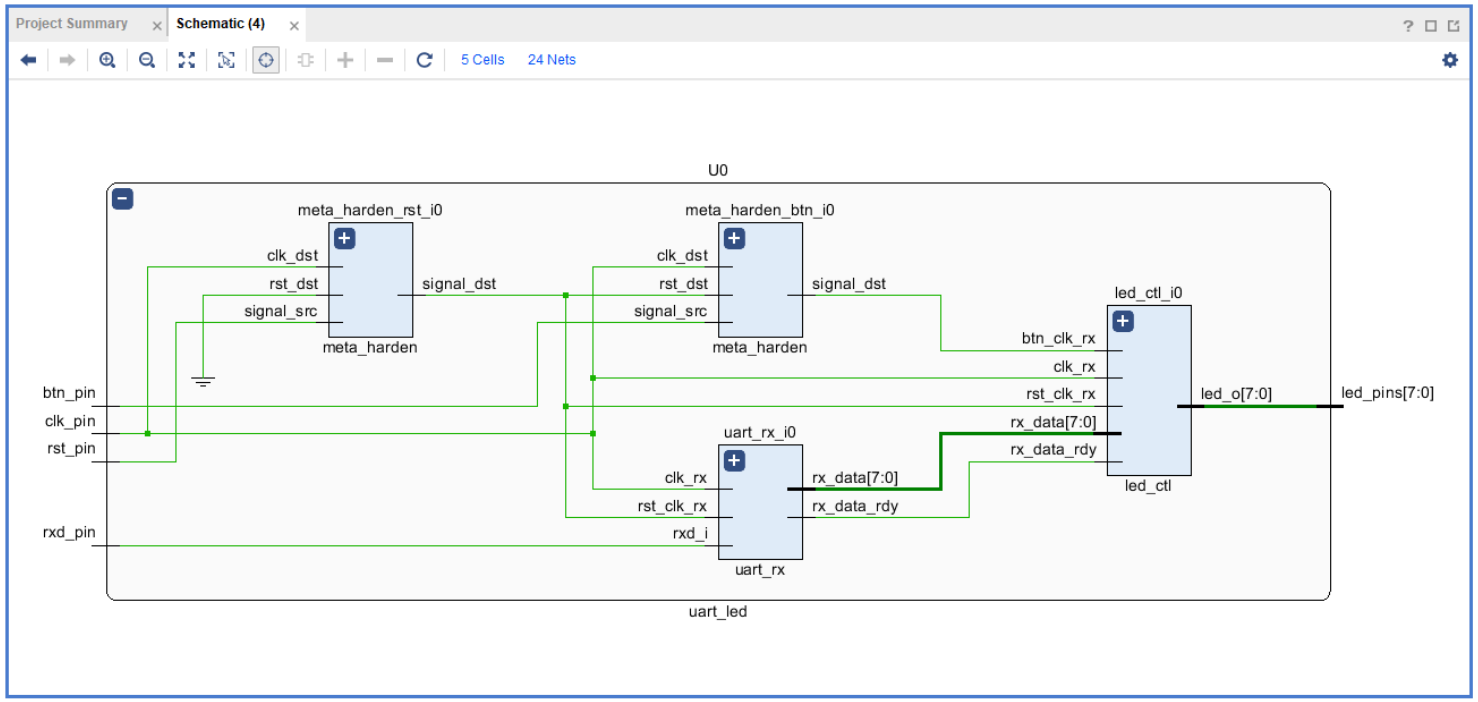
Analysis of an Elaborated Design

- **Three options available after opening an Elaborated Design**
  - Report DRC
    - Runs DRC on the design
  - Report Noise
    - Checks the SSO on the design based on a XDC file
  - Schematic
    - Opens the schematic



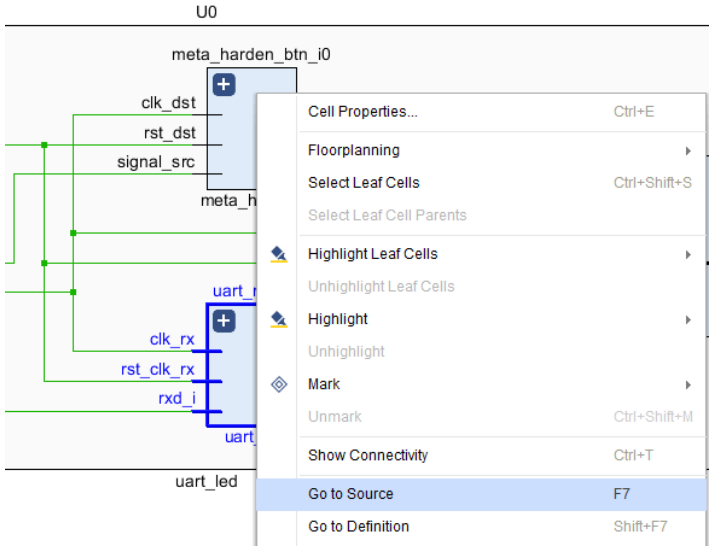
Schematic View of an Elaborated Design

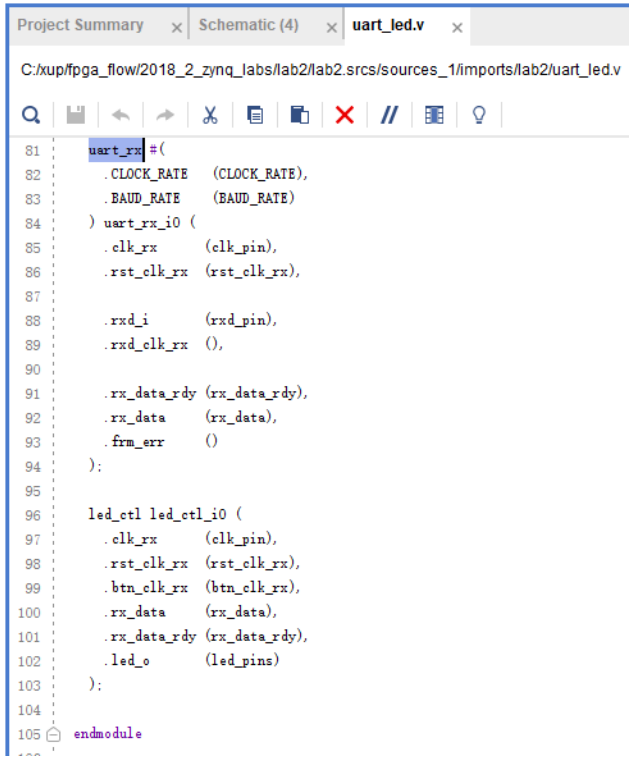
- **When Schematic is clicked under the Elaborated Design, the schematic is opened showing the hierarchical blocks**
  - Note that no I/O buffers are inferred at this stage
  - Each block opens up to reveal underlying logic and sub-modules in the hierarchy
  - Closest representation to the actual coded design



Cross Probing

> **Select an object in the schematic, right-click, and select Go To Source to view where the object is defined in the source file**

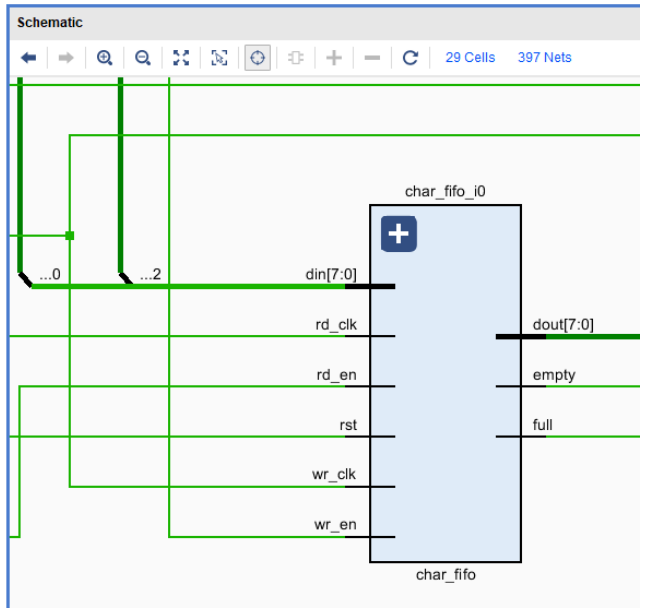




Synthesis

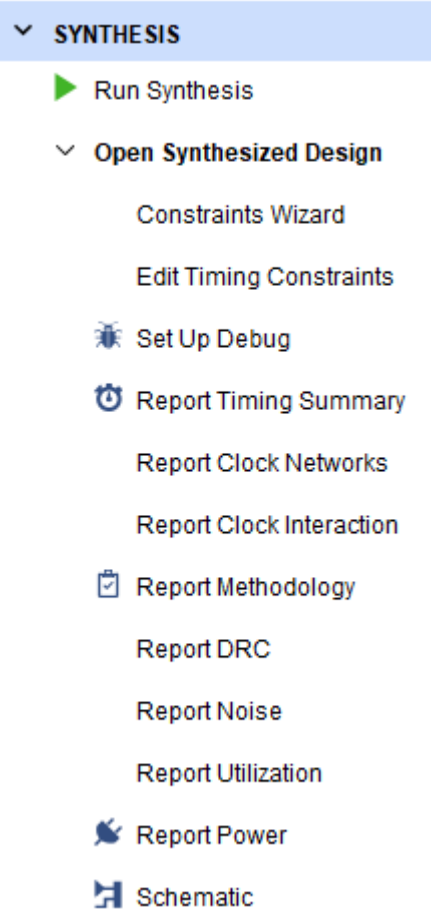
Synthesis: Logic Optimization and Mapping to Device Primitives

- **Synthesis of an RTL design**
  - Optimizes the gate-level design
  - Maps the netlist to Xilinx primitives
    - Sometimes called technology mapping
- **The figure shows a generic FIFO behavioral component mapped to the dedicated FIFO hardware in a 7-Series FPGA**



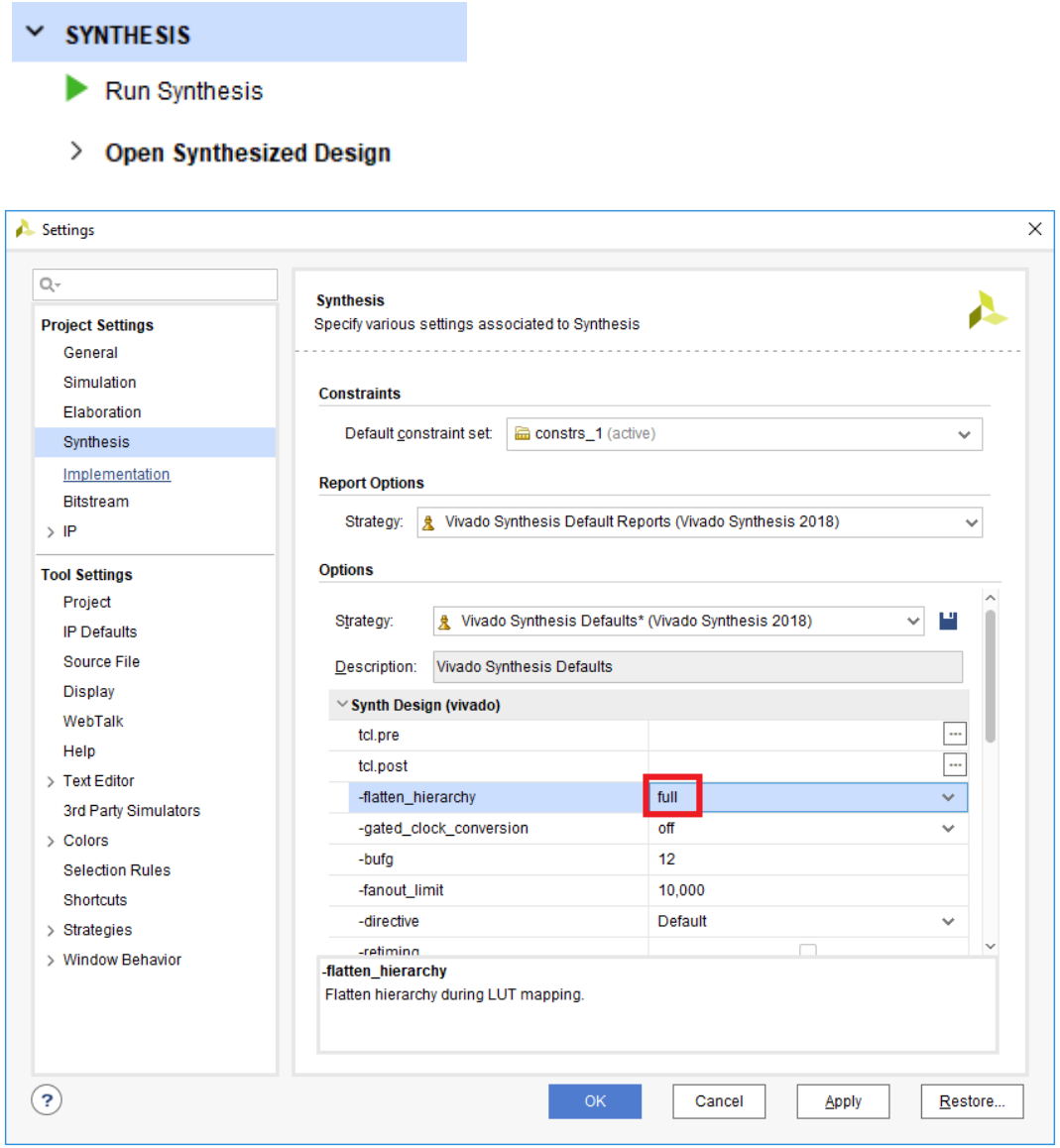
Vivado IDE Synthesis

- **Applicable only for RTL (HDL) design flows**
  - EDIF is black boxed and linked after synthesis
- **Synthesis tool uses XDC constraints to drive synthesis optimization**
  - XDC file must exist
- **Timing constraints considerations**
  - Design must first be synthesized without timing constraints for constraints editor usage
  - Constraints Wizard is available after synthesis to define rudimentary timing constraints
- **Synthesis settings provide access to additional options**
- **Note the changes in the Flow Navigator when the synthesized design has been opened**
  - Set Up Debug allows for integration of debug feature



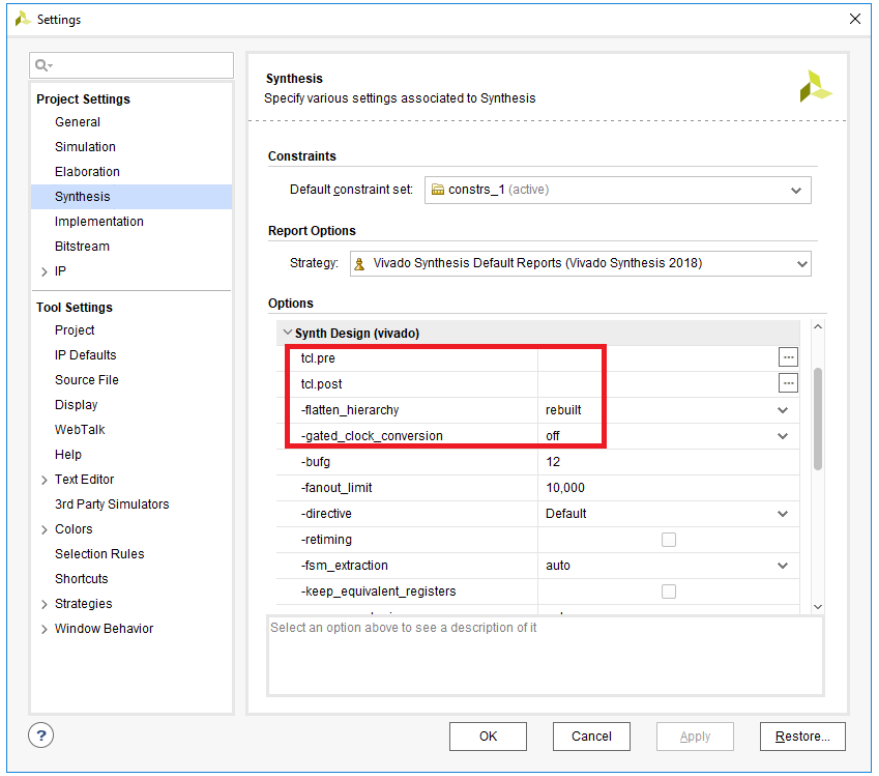
Vivado IDE Synthesis Settings

- **Synthesis Settings brings up Project Settings dialog box**
  - Central location for all project settings
- **Select the Default Constraint Set as the active *constraint set*. Two types of the design constraints**
  - **Physical constraints** define pin placement, and absolute, or relative, placement of cells such as block RAMs, LUTs, Flip-Flops, and device configuration settings.
  - **Timing constraints**, written in industry standard SDC, define the frequency requirements for the design. Without timing constraints, the Vivado Design Suite optimizes the design solely for wire length and placement congestion.



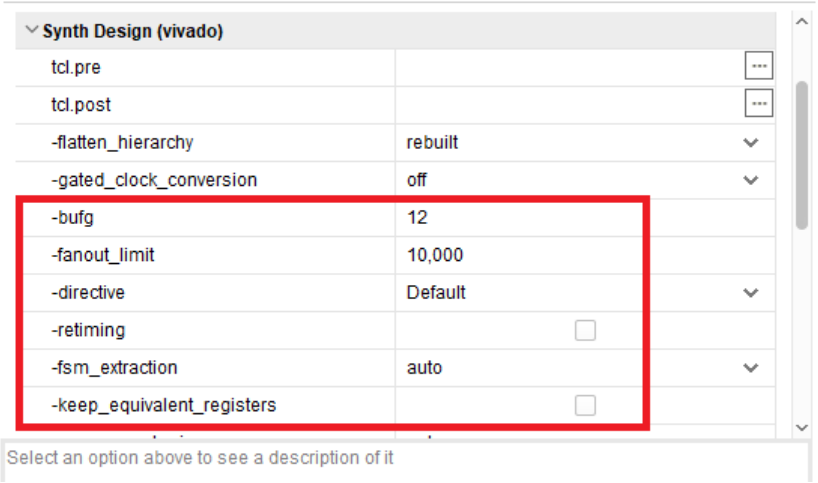
Vivado IDE Synthesis Settings

- The **tcl.pre** and **tcl.post** options are hooks for Tcl files that run immediately before and after synthesis
- **flatten_hierarchy**: Determines how synthesis controls hierarchy
  - none: Do not flatten the hierarchy.
    - Maintains the exact same hierarchy as the original RTL
  - full: Fully flatten the hierarchy leaving only the top level
  - rebuilt: Flatten the hierarchy, perform synthesis, and then rebuild the hierarchy based on the original RTL
    - Allows the QoR benefit of cross-boundary optimizations, with a final hierarchy that is similar to the RTL for ease of analysis
- **gated_clock_conversion:** Turns on and off the synthesis tools ability to convert clock_logic with enables



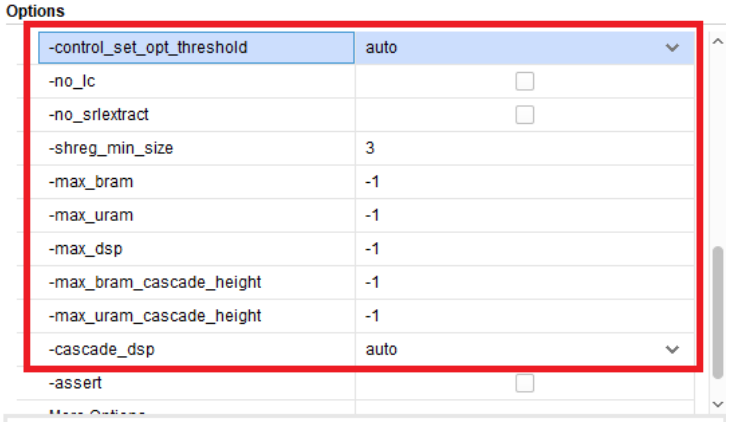
Vivado IDE Synthesis Settings

- **bufg**: Controls how many BUFGs the tool infers in the design
  - This option is used when other BUFGs in netlists are not visible to the synthesis process
- **fsm_extraction**: Encode the state machine in a specific encoding type: one_hot, sequential, johnson, gray, or auto
- **keep_equivalent_registers**: Prevents registers with the same input logic from being merged
- **resource_sharing**: Sets the sharing of arithmetic operators between different signals
  - The values are auto, on and off



Vivado IDE Synthesis Settings

- **control_set_opt_threshold: Sets the threshold for clock enable optimization to the lower number of control sets**
- **no_lc: When checked, this option turns off LUT combining**
- **shreg_min_size: Is the threshold for inference of SRLs**
  - This sets the number of sequential elements that would result in the inference of an SRL for fixed delay chains (static SRL)
- **max_bram: Limits the number of BRAM allowed in the design.**
  - -1 default lets the tool pick as many BRAM as it can, limited by number of BRAMs in device
- **max_dsp: Similar in concept to max_bram, except this applies to DSPs.**
- **To see explanation of each option, click on the name of each option to highlight it.**



Synthesis RTL Attributes Supported

Attribute	Description
translate_off/_on	Tells the tool to ignore blocks of code
full_case	Tells that all possible case values are specified
parallel_case	Case statement should be built as a parallel structure
keep	Tells tool to keep the signal the attribute is placed on
keep_hierarchy	Used to prevent optimizations along the hierarchy boundaries
buffer_type	Tells tool what buffer type to use on an input
max_fanout	Tells the tool the limits for fanout on registers and signals
ram_style	Tells the tool how to infer memory
rom_style	Tells the tool how to infer ROM memory
use_dsp48	Tells the tool how to deal synthesis arithmetic structures
black_box	Turns a whole level of hierarchy off and enables synthesis to create a black box for that module/entity
gated_clock	Allows the conversion of gated clocks; must be enabled
shreg_extract	Tells the tool on whether to infer structures
iob	Not a synthesis attribute but is passed to the implementation tool indicating if a register should be in IOB

Synthesis RTL Attributes Supported, cont'd…

Attribute	Description
async_reg	Tells the tool that a register can receive asynchronous data at D input
srl_style	Specifies how SRL is inferred in design
clock_buffer_type	Specifies a buffer other than the (default) BUFG for synthesis
dont_touch	Similar to KEEP attribute, use in place of KEEP. Attribute is forward-annotated to place & route
fsm_encoding	Specifies a specific FSM encoding scheme: one_hot, sequential, johnson, gray, auto (default), none
fsm_safe_state	Place on state machine state registers, used to define a safe state in the machine
IOB	Not a synthesis attribute, used by Vivado implementation. Specifies if a register is packed into IOB
io_buffer_type	Instructs the tool to not automatically infer I/O buffers for a specific top-level port.
MARK_DEBUG	Specifies that a net to be marked for debug.

Attributes Example in RTL

- **No support for timing constraints embedded in RTL**
- **Example of KEEP attribute**

VHDL

```
signal sig1 : std_logic;
attribute KEEP : string;
attribute KEEP of sig1 : signal is "true";
sig1 <= in1 and in2;
out1 <= sig1 and in3;
```

Verilog

```
(* KEEP = "true" *) wire sig1;
assign sig1 = in1 & in2;
assign out1 = sig1 & in3;   // without the attribute sig1 will be optimized away if it not
                            // being used anywhere else in the model
```


XILINX

Attributes Example in RTL

> **Attributes do not work with Verilog 2001 module syntax**

Does Not Work

```
module top ((* buffer_type = "none" *) input sys_clock,
                                        input sys_reset,
            (* buffer_type = "none" *) input serDataIn,
```

Works

```
module top (    sys_clock,
                sys_reset,
                serDataIn,
                …
            );
(* buffer_type = "none" *) input  sys_clock;
                           input  sys_reset;
(* buffer_type = "none" *) input  serDataIn;
```

Mark_Debug RTL Attribute

- **MARK_DEBUG support with Vivado synthesis**
  - Tag signals in HDL
  - Corresponding nets preserved in netlist (DONT_TOUCH behavior)
  - Tagged nets appear in the logic analyzer Unassigned nets view
  - XDC and elaborated design netlist support available

- **You can also mark nets after synthesis**

Apply attribute in HDL

C:/xup/fpga_flow/2018_2_zynq_labs/lab4/lab4.srcs/sources_1/imports/lab4/cmd_parse.v

```
129
130
131 //*****************************************************************
132 // Reg declarations
133 //*****************************************************************
134
135   reg [2:0]          state;     // State variable
136
137   reg                old_rx_data_rdy; // rx_data_rdy on previous clock
138
139   reg [6:0]          cur_cmd;   // Current cmd - least 7 significant bits of char
140   reg [4*MAX_ARG_CH-5:0]         arg_sav;   // All but last char of args
141   reg [clogb2(MAX_ARG_CH)-1:0]   arg_cnt;   // Count the #chars in an argument
142
143   reg [NSAMP_WID:0] nsamp;     // Number of samples to send
144                                // Width is one more to code 2*N naturally
145 (* mark_debug = "TRUE" *)    reg [PRE_WID-1:0] prescale; // Clock prescaler
146 (* mark_debug = "TRUE" *)    reg [SPD_WID-1:0] speed;    // Speed
147 (* mark_debug = "TRUE" *)    reg               nsamp_new;
148 (* mark_debug = "TRUE" *)    reg               prescale_new;
149 (* mark_debug = "TRUE" *)    reg               speed_new;
```

After Synthesis

Tcl Console | Messages | Log | Reports | Design Runs | Debug

Name	Driver Cell	Driver Pin	Probe Type
Unassigned Debug Nets (35)			
U0_wave_gen/cmd_parse_i0/prescale (16)	Multiple	Q	
U0_wave_gen/cmd_parse_i0/speed (16)	Multiple	Q	
U0_wave_gen/cmd_parse_i0/nsamp_new	FDRE	Q	
U0_wave_gen/cmd_parse_i0/prescale_new	FDRE	Q	
U0_wave_gen/cmd_parse_i0/speed_new	FDRE	Q	

Basic Timing Constraints

XILINX

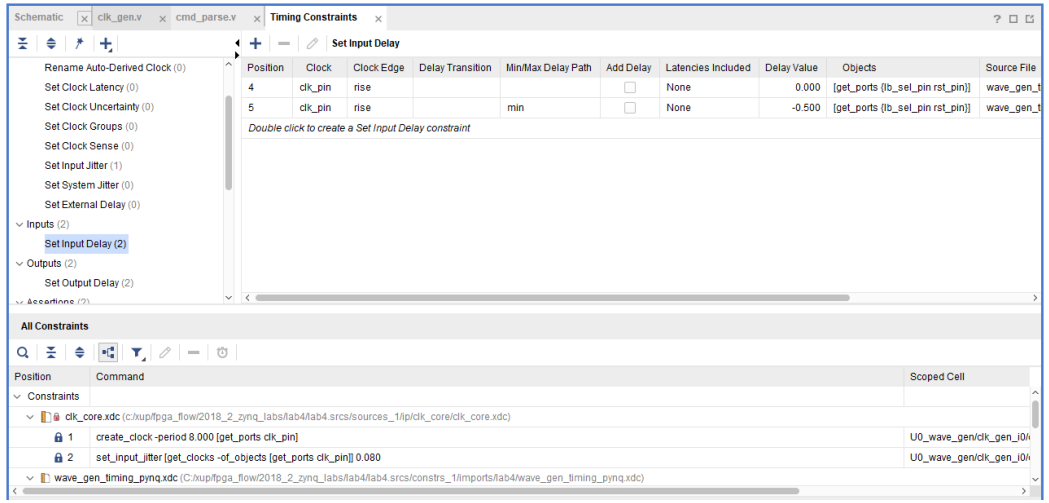
Basic Timing Constraints

- **There are three basic timing constraints applicable to a sequential machine**
  - Period
    - Paths between synchronous elements clocked by the reference clock net
      - Synchronous elements include flip-flops, latches, synchronous RAM, and DSP slices
    - Use `create_clock` to create the constraint
  - Input Delay
    - Paths between input pin and synchronous elements
    - Use `set_input_delay` to create the constraint
  - Output delay
    - Paths between synchronous elements and output pin
    - Use `set_output_delay` to create the constraint

- **More about this in the module on Xilinx Design Constraints**

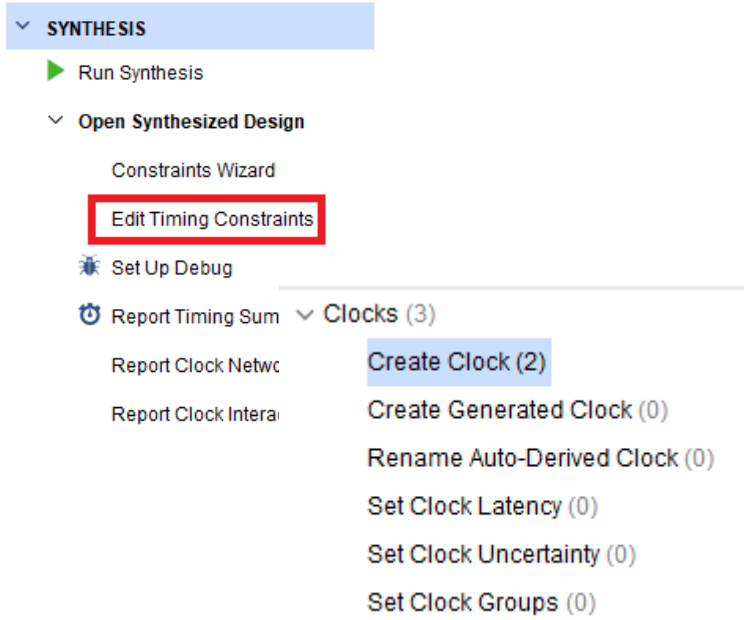
Constraints Viewer

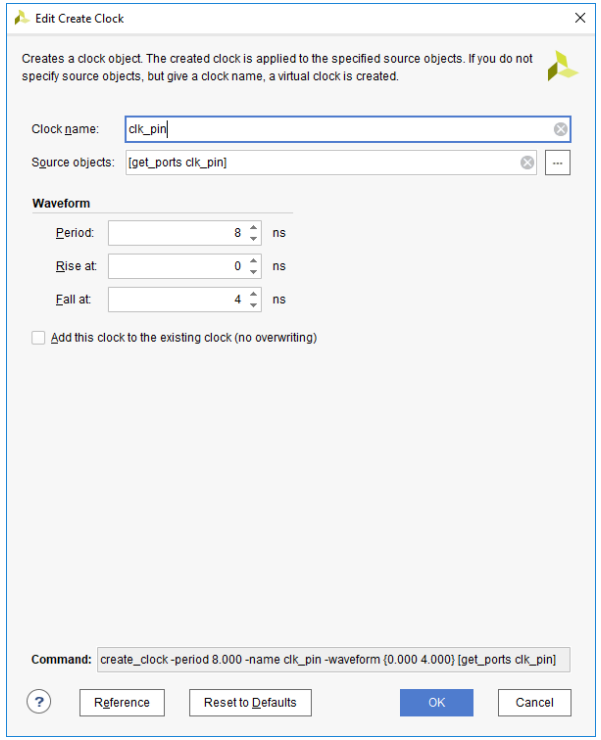
- **Constraints Viewer allows you to see/edit the timing constraints contents of the XDC file**
  - All timing constraints are supported with the GUI
  - Right-clicking the constraint in the Create window allows you to modify/delete an existing constraint
  - Select a different constraint by clicking it in the All Constraints window
  - To create a different constraint, select the type from the GUI
- **After constraint entry is completed, constraints must be "saved" to have them stored in the XDC**
  - Changes will be written into the specified Target XDC only

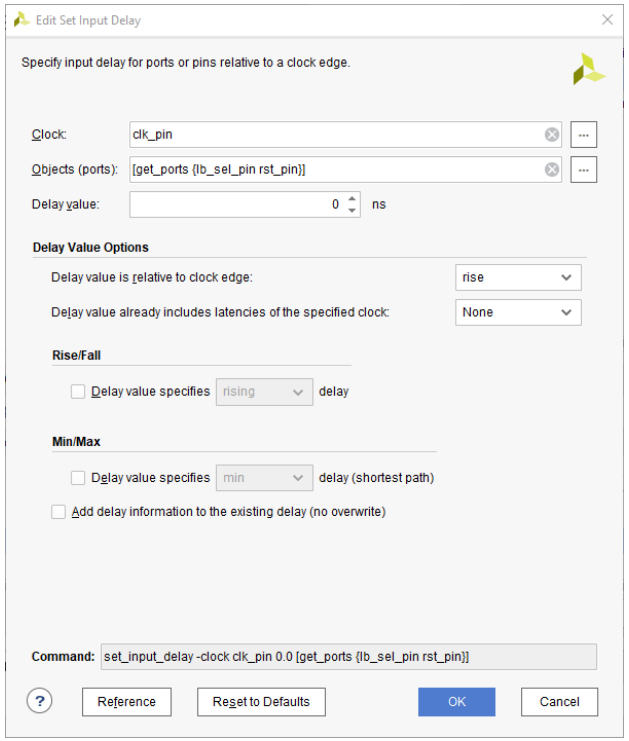


Creating Basic Timing Constraints in Vivado IDE

- Run Synthesis
- Open the synthesized design
- Invoke constraints editor





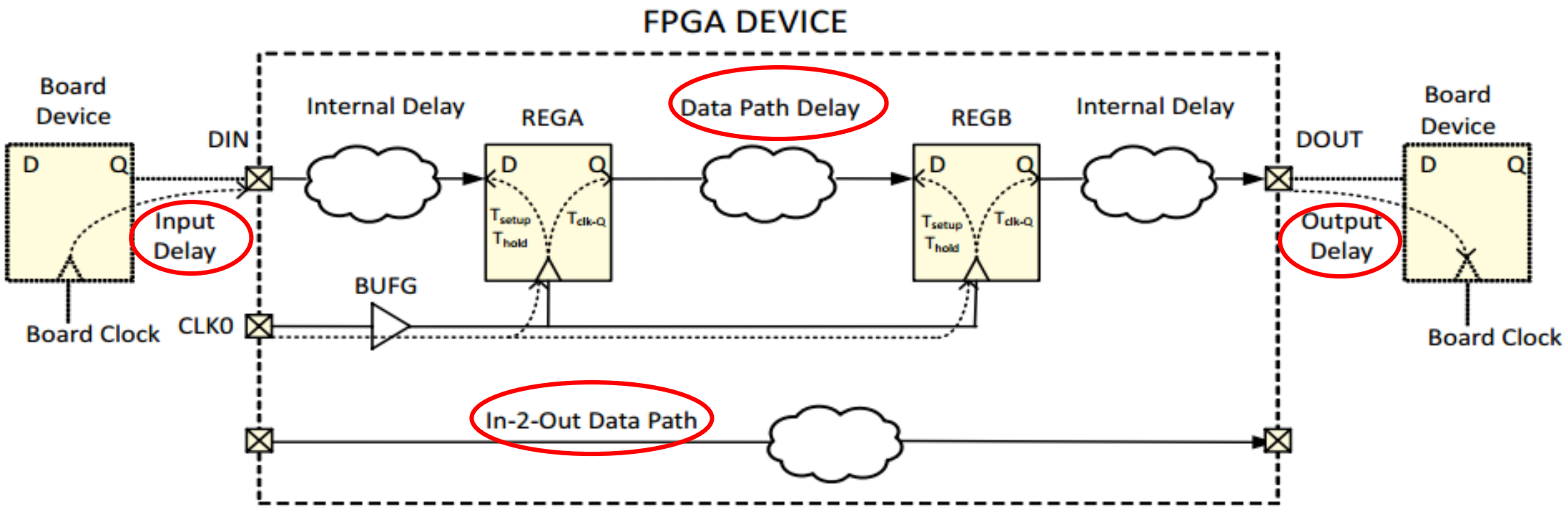


Creating Basic Timing Constraints using Tcl

- **Period constraint using create_clock**
  - create_clock –period 10 [get_ports clk_pin]
  - create_clock –period 10 –waveform {0.000 5.000} –name clk [get_ports clk_pin]
- **set_input_delay**
  - The input delay external to FPGA
  - set_input_delay –clock clk_pin 2.000 [all_inputs]
  - set_input_delay –clock clk_pin 3.000 [get_ports in1]
  - set_input_delay –clock clk_pin 2 [get_ports l_msn*]
- **set_output_delay**
  - The output delay external to FPGA
  - set_output_delay –clock clk_pin -2 [all_output]
  - set_output_delay –clock [get_clocks clk_pin] -2 [get_ports out1]

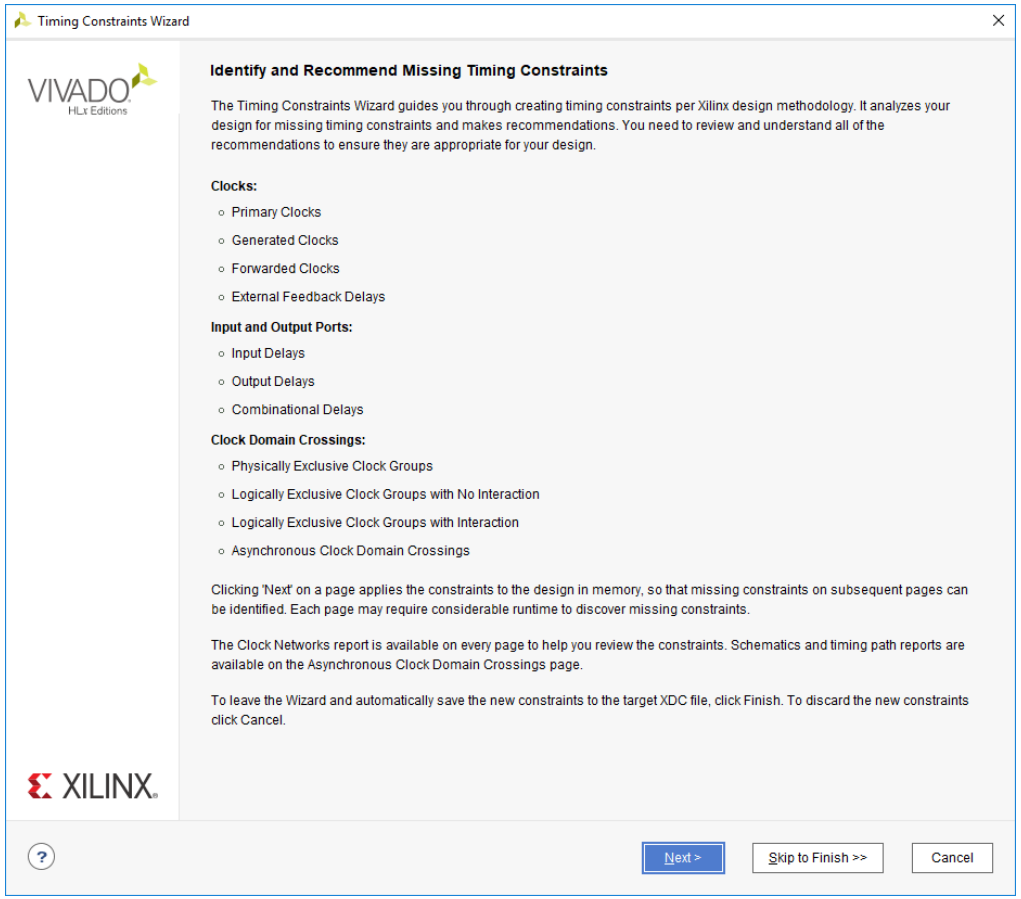
XILINX

Timing Paths Example



Constraints Wizard

- **Allows Vivado to *suggest* timing constraints missing from the design**
  - Recommended flow for an initially unconstrained design
  - User can ignore the recommended constraints
    - Option to deselect constraints at every stage of the wizard
  - Opens the Timing Constraints Editor with recommended constraints
    - User needs to consciously add the constraints to the target XDC file
  - A great alternative to using the Timing Constraints Editor in Vivado
    - User can still use the editor to add constraints



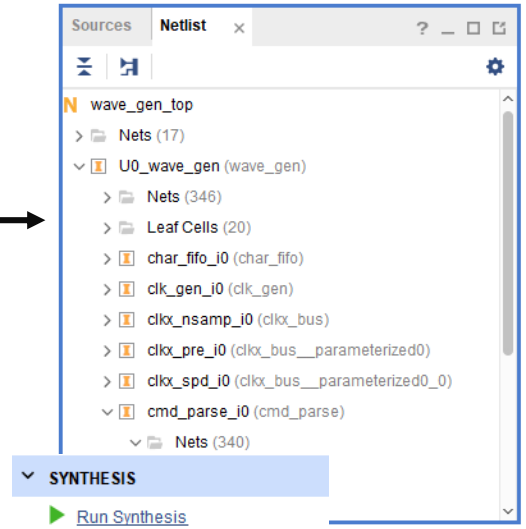
Synthesis Reports

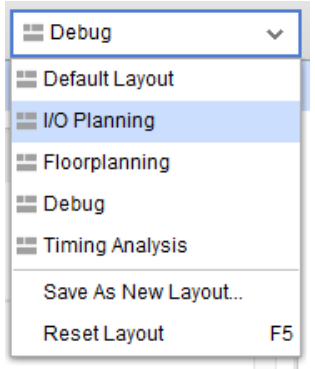
XILINX

After Synthesis

In the Synthesized Design view:

- **Sources tab does not change**
  - RTL Netlist tab (Elaborated Design) changes to Netlist (Synthesized Design)
- **However, the Flow Navigator now includes:**
  - Constraints Wizard, Edit Timing Constraints, Set Up Debug, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report DRC, Report Noise, Report Utilization, Report Power, Schematic
- **Views can selected by purpose**
  - All timing information is only an estimate (until implementation has completed)
  - Setup debug tool



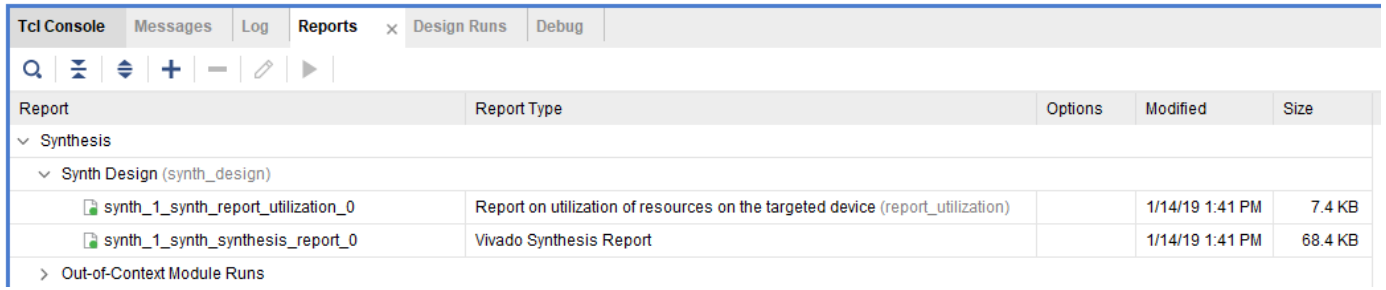


Synthesized Design

- **Accessed through the Flow Navigator by selecting Open Synthesized Design**
- **Representation of the design after synthesis**
  - Interconnected netlist of hierarchical and BELs
    - Instances of modules/entities
    - BELs
      - LUTs, flip-flops, carry chain elements, wide MUXes
      - Block RAMs, DSP cells
      - Clocking elements (BUFG, BUFR, MMCM, …)
      - I/O elements (IBUF, OBUF, I/O flip-flops)
- **Object names are the same as names in the elaborated netlist when possible**

Synthesis Reports

- **While the Flow Navigator points to the most important reports, the Reports tab contains several other useful reports**
  - Vivado Synthesis Report shows
    - HDL files synthesized, synthesis progress, timing constraints read, and RTL primitives from the RTL design
    - Timing optimization goals, technology mapping, removed pins/ports, and final cell usage (technology-mapped cell usage)
  - Utilization Report shows
    - Technology-mapped cell usage in an easy-to-read tabular format



Synthesis Utilization Report

> **Reports slice logic, memory, DSP slice, IO, clocking, and other resources used by the design**

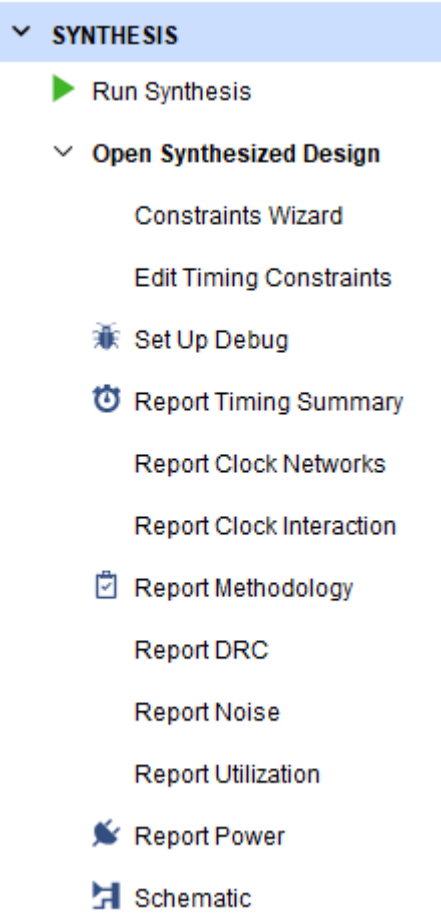
Schematic | clk_gen.v | cmd_parse.v | synth_1_synth_report_utilization_0 - synth_1

C:/xup/fpga_flow/2018_2_zynq_labs/lab4/lab4.runs/synth_1/wave_gen_top_utilization_synth.rpt

```
Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2018.2 (win64) Build 2258646 Thu Jun 14 20:03:12 MDT 2018
| Date         : Mon Jan 14 13:41:35 2019
| Host         : XSHWELI30 running 64-bit major release  (build 9200)
| Command      : report_utilization -file wave_gen_top_utilization_synth.rpt -pb wave_gen_top_utilization_synth.pb
| Design       : wave_gen_top
| Device       : 7z020clg400-1
| Design State : Synthesized
------------------------------------------------------------------------------------------


Utilization Design Information

Table of Contents
-----------------

1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------


+----------------------------+------+-------+-----------+-------+
|          Site Type         | Used | Fixed | Available | Util% |
+----------------------------+------+-------+-----------+-------+
```

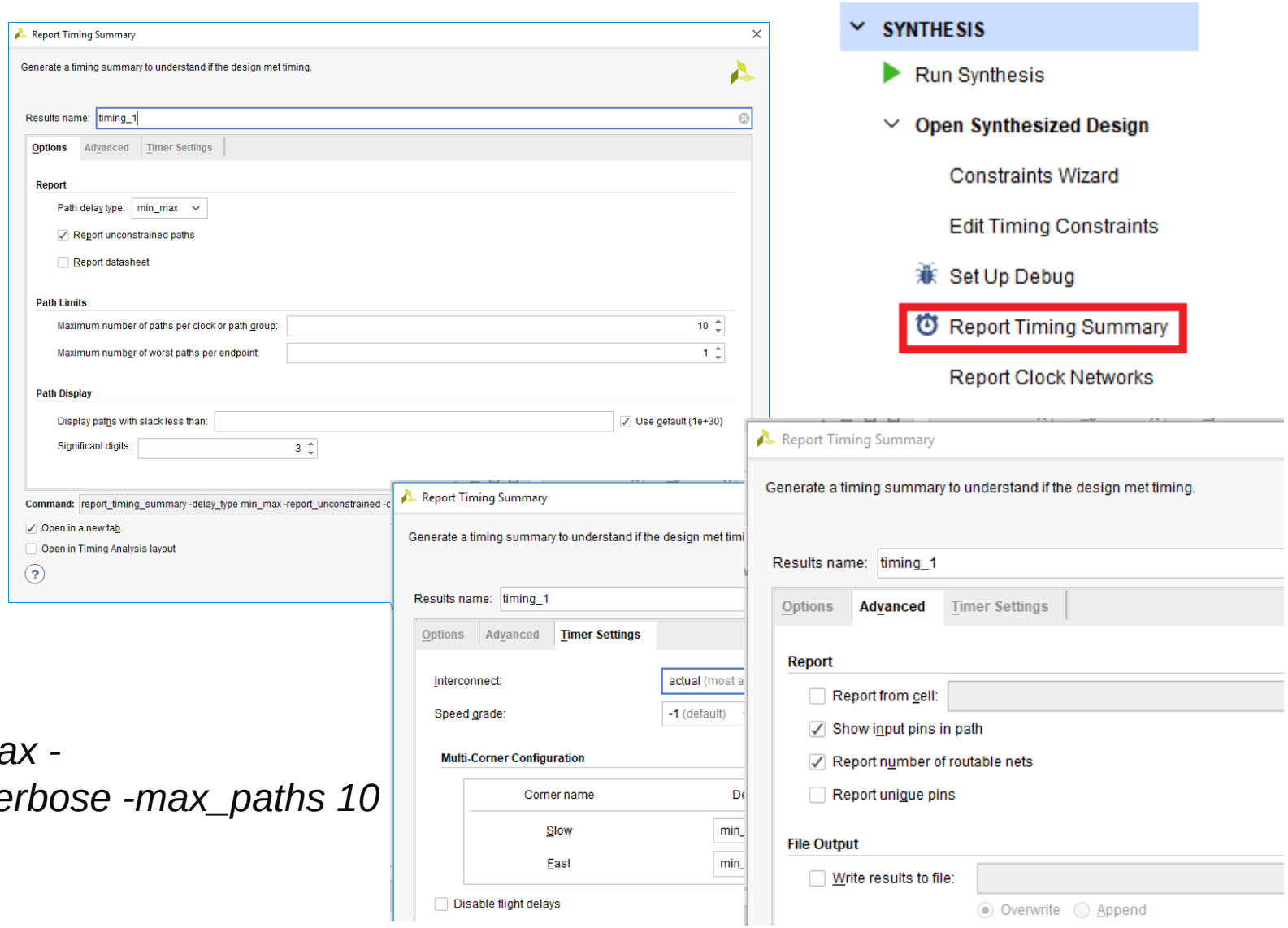
Commands Available After Synthesis

- **Flow Navigator is optimized to provide quick access to the options most frequently used after synthesis**
  - Constraints Wizard: Already mentioned
  - Edit Timing Constraints: Launch the timing constraints tab
  - Set Up Debug: Launch the view for marking nets for debug
  - Report Timing Summary: Generate a default timing report
  - Report Clock Networks: Generates a clock tree for the design
  - Report Clock Interaction: Verifies constraint coverage on paths between clock domains
  - Report DRC: Performs design rule check on the entire design
  - Report Noise: Performs an SSO analysis of output and bidirectional pins in the design
  - Report Utilization: Generates a graphical version of the Utilization Report
  - Report Power: Detailed power analysis reports
  - Schematic: Opens the Schematic viewer



Report Timing Summary

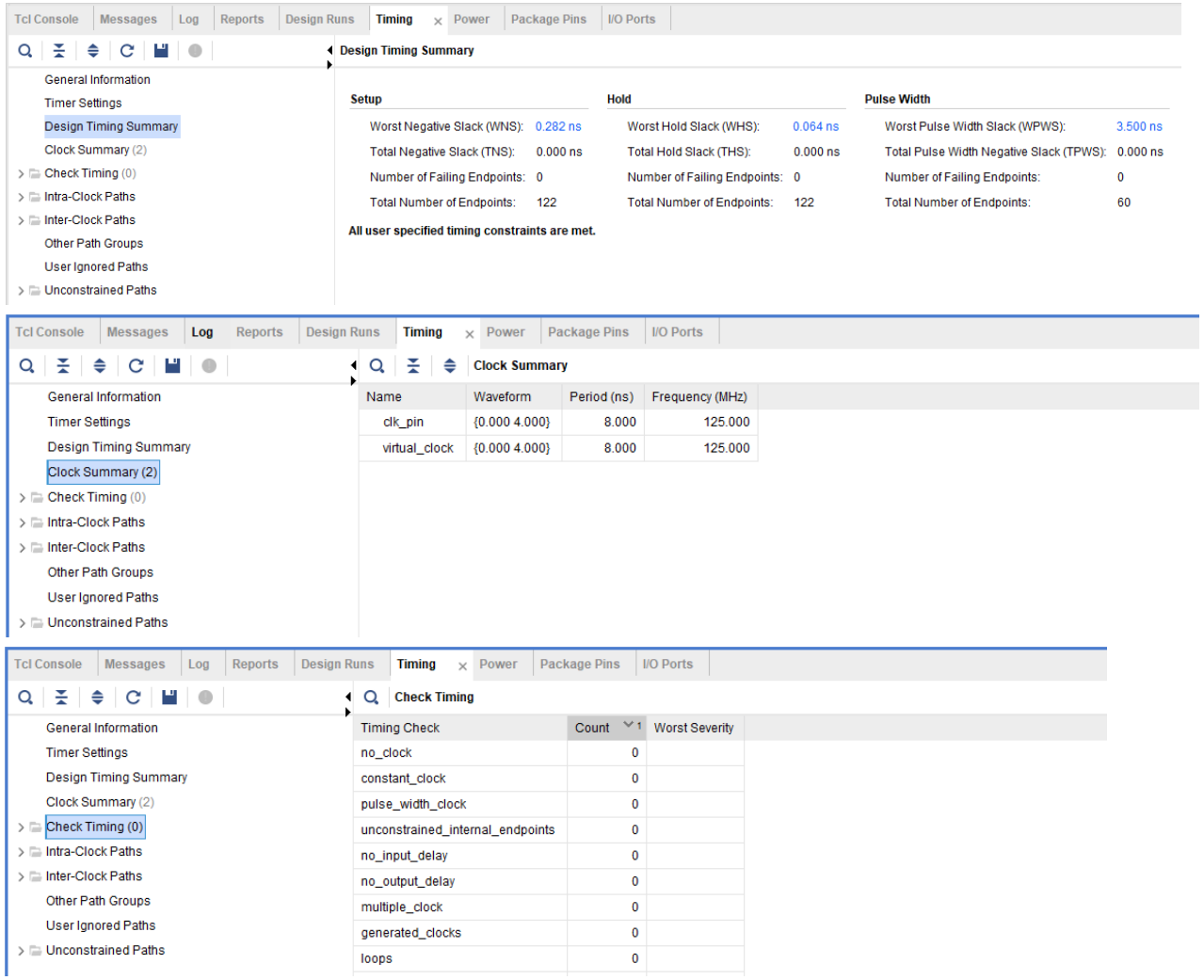
- **Vivado IDE**
- **Options tab**
  - Maximum number of paths
- **Advanced tab**
  - Write to a file
- **Timer Settings**
  - Interconnect delay can be ignored
  - Flight delays can be disabled



Tcl command: *report_timing_summary report_timing_summary -delay_type max -report_unconstrained -check_timing_verbose -max_paths 10 -input_pins -name timing_1*

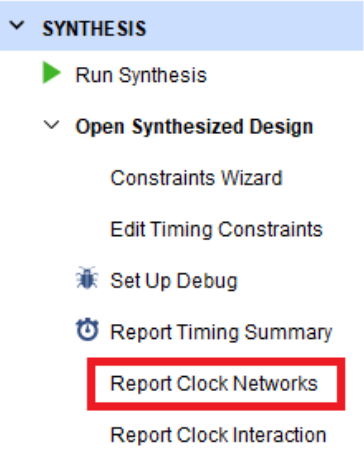
Report Timing Summary

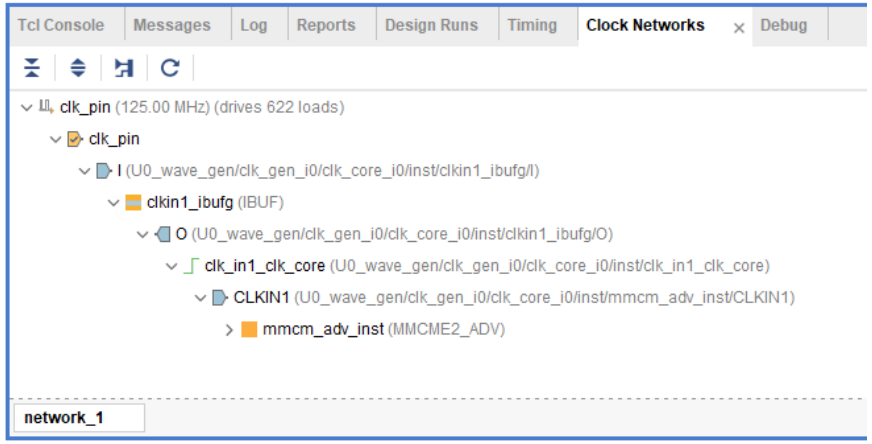
- **Design Timing Summary**
  - WNS, TNS, total number of endpoints are of interest
- **Clock Summary**
  - Primary and derived clocks
- **Check Timing**
  - Number of unconstrained internal endpoints



Report Clock Networks

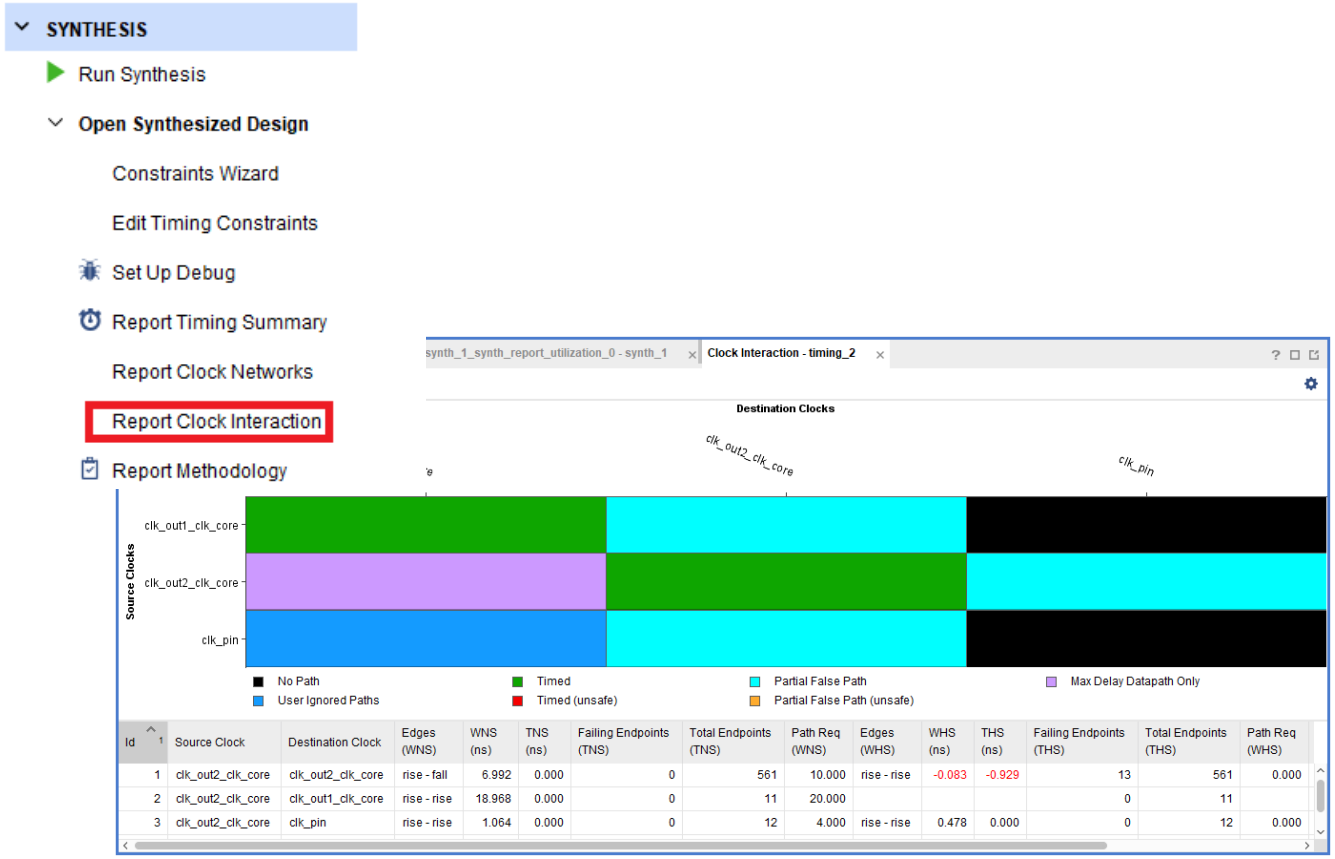
- **To generate a report on the clock networks in a design, use the tcl command**
  - `report_clock_networks`
  - Creates a tree view of all the logical clock trees found in the design, annotated with existing and missing clock definitions on the roots of these trees
- **Vivado IDE**





Report Clock Interaction

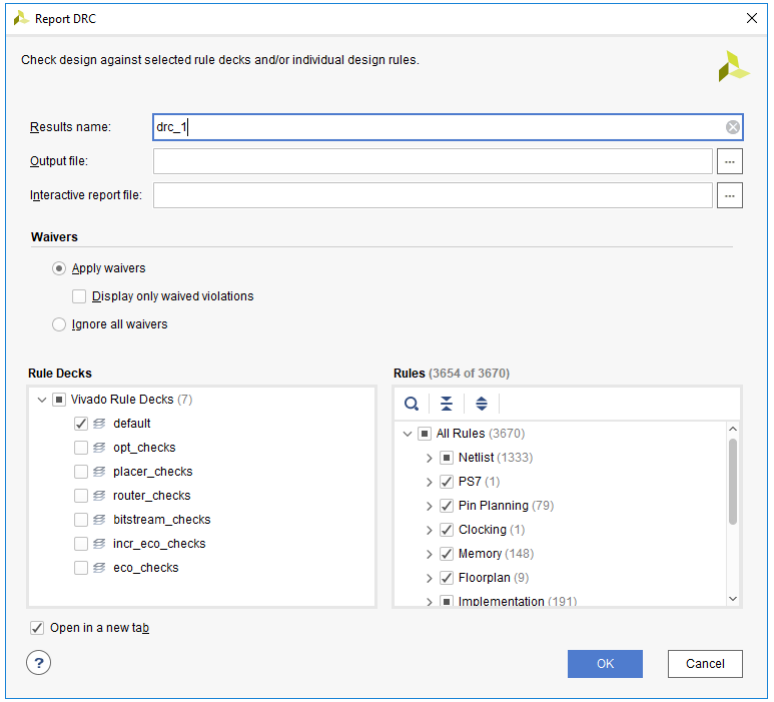
- **Use Tcl command**
  - `report_clock_interaction -significant_digits 3 -name timing_1`
- **Reports paths on inter-clock domain paths and unclocked registers**
  - Uses an inter-clock path matrix to show clock relationships and group paths
  - Tells if timing is asynchronous and if paths are constrained (green)
  - Uses the worst-slack value for each clock grouping

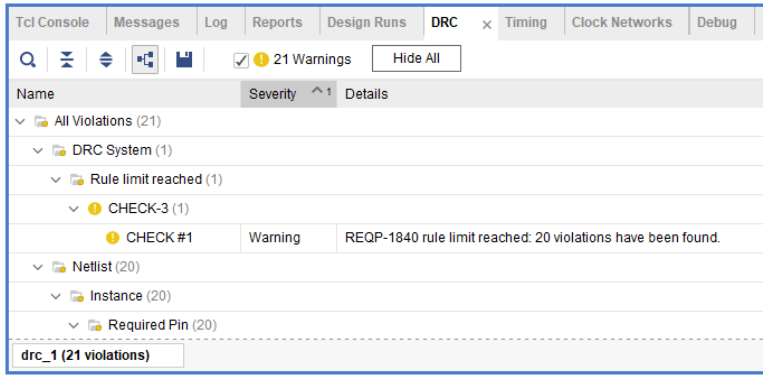


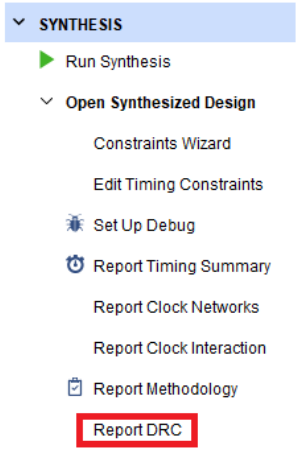
Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Path Req (WNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	Path Req (WHS)
1	clk_out2_clk_core	clk_out2_clk_core	rise - fall	6.992	0.000	0	561	10.000	rise - rise	-0.083	-0.929	13	561	0.000
2	clk_out2_clk_core	clk_out1_clk_core	rise - rise	18.968	0.000	0	11	20.000				0	11	
3	clk_out2_clk_core	clk_pin	rise - rise	1.064	0.000	0	12	4.000	rise - rise	0.478	0.000	0	12	0.000

Report DRC

- **Use Tcl command**
  - `report_drc -name drc_1`
- **DRCs performed early in the design flow allow for correction before a full implementation**
  - Objects listed in violations are cross-selectable with HDL sources
  - Can select which DRCs to run, or run all
    - This check is more thorough than the I/O DRC checks that occur during pin planning
  - Any problems will open a DRC window at the bottom of the GUI
  - Run implementation for the final sign-off DRC
- **Vivado IDE**

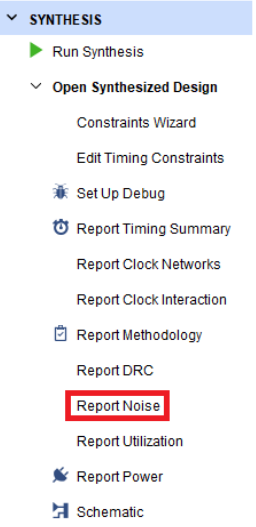


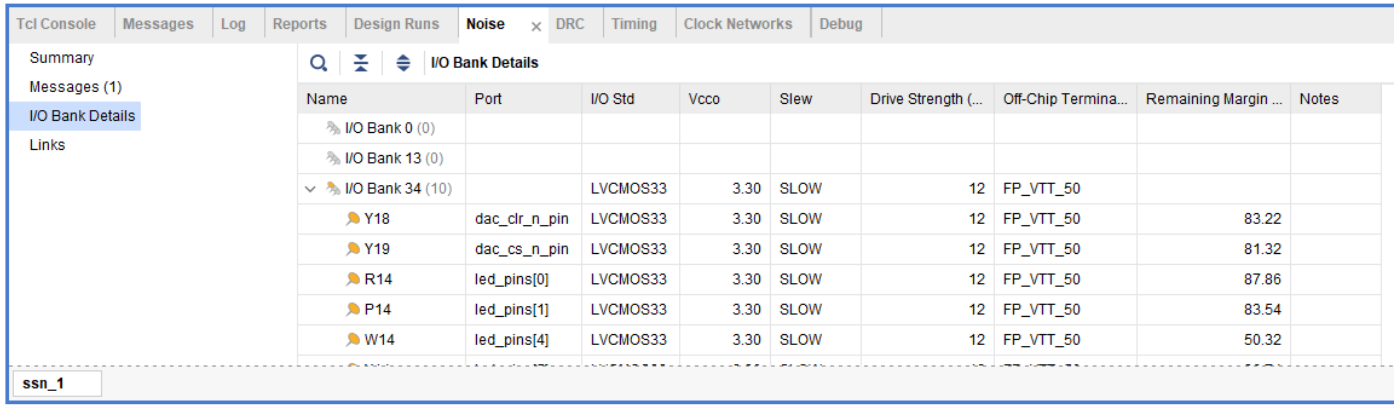




Report Noise

- **Use Tcl command**
  - `report_ssn -name ssn_1`
- **Performs a SSN analysis of the planned I/O layout**
  - This report is looking to gauge the number of pins, I/O standard, and drive strength on a bank-by-bank basis
    - Banks that exceed what is recommended will be flagged in the Summary tab
    - SSN analysis can only be done on output and bidirectional ports
- **Vivado IDE**

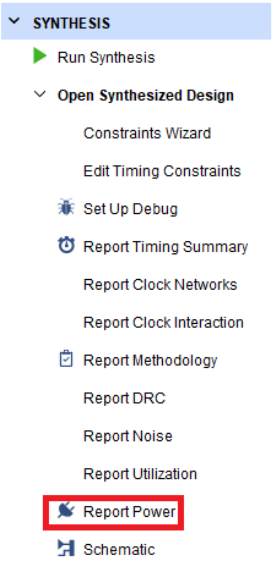


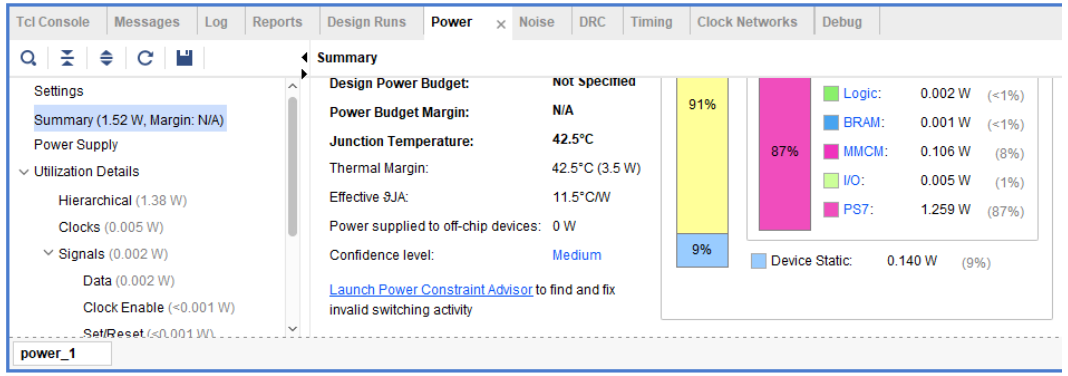


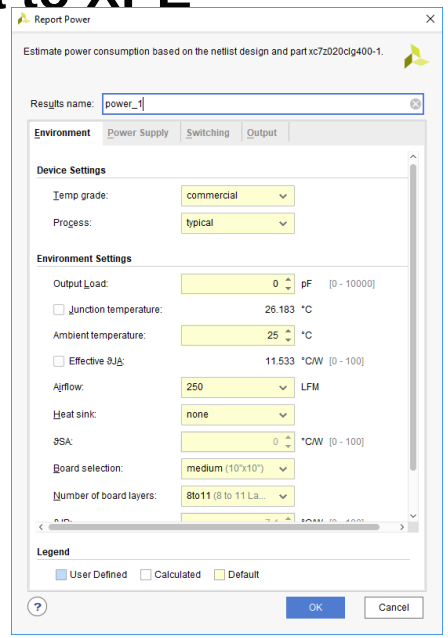
Name	Port	I/O Std	Vcco	Slew	Drive Strength (...	Off-Chip Termina...	Remaining Margin ...	Notes
I/O Bank 0 (0)								
I/O Bank 13 (0)								
I/O Bank 34 (10)		LVCMOS33	3.30	SLOW	12	FP_VTT_50		
Y18	dac_clr_n_pin	LVCMOS33	3.30	SLOW	12	FP_VTT_50	83.22	
Y19	dac_cs_n_pin	LVCMOS33	3.30	SLOW	12	FP_VTT_50	81.32	
R14	led_pins[0]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	87.86	
P14	led_pins[1]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	83.54	
W14	led_pins[4]	LVCMOS33	3.30	SLOW	12	FP_VTT_50	50.32	

Report Power

- Use Tcl command
  - `report_power -results {power_1}`
- Accurate power and thermal analysis
- Power estimates at every stage after synthesis
- Perform what-if analysis by varying switching activity
- Extensive debug capabilities through cross-probing
- Exports data to XPE







Power Utilization Details

> **Select different resource types to see more details**

Tcl Console | Messages | Log | Reports | Design Runs | Power | Noise | DRC | Timing | Clock Networks | Debug

Hierarchical

- Settings
- Summary (1.52 W, Margin: N/A)
- Power Supply
- Utilization Details
  - Hierarchical (1.38 W)
  - Clocks (0.005 W)
  - Signals (0.002 W)
    - Data (0.002 W)
    - Clock Enable (<0.001 W)
    - Set/Reset (<0.001 W)

Utilization	Name	Clocks (W)	Signals (W)	Data (W)	Clock Enable (W)	Set/Reset (W)	Logic (W)	BRAM (W)	Clock Manager (W)	MMCM (W)	I/O (W)
1.38 W (91% of total)	wave_gen_top										
1.259 W (83% of total)	U1_system_wrapper (system_wrapper)	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
0.12 W (8% of total)	U0_wave_gen (wave_gen)	0.005	0.002	0.002	<0.001	<0.001	0.002	0.001	0.106	0.106	0.005

power_1

Tcl Console | Messages | Log | Reports | Design Runs | Power | Noise | DRC | Timing | Clock Networks | Debug

Logic

Engine | Constraint | Estimated | Default | Calculated

- Clocks (0.005 W)
- Signals (0.002 W)
  - Data (0.002 W)
  - Clock Enable (<0.001 W)
  - Set/Reset (<0.001 W)
  - Logic (0.002 W)
  - BRAM (0.001 W)
  - Clock Manager (0.106 W)
  - I/O (0.005 W)
  - PS7 (1.259 W)

Utilization	Name	Type	Clock (MHz)	Clock Name	Signal Rate (Mtr/s)	% High
0.002 W (<1% of total)	wave_gen_top					
<0.001 W (<1% of total)	bcd_out[3]_i_1 (LUT5)	LUT	N/A	Async	49.980	49.699
<0.001 W (<1% of total)	bcd_out[3]_i_3 (LUT6)	LUT	N/A	Async	60.049	20.991
<0.001 W (<1% of total)	bcd_out[2]_i_1 (LUT6)	LUT	N/A	Async	46.555	40.787
<0.001 W (<1% of total)	bcd_out[1]_i_1 (LUT5)	LUT	N/A	Async	40.199	52.264
<0.001 W (<1% of total)	bcd_out[4]_i_3 (LUT6)	LUT	N/A	Async	60.000	59.987
<0.001 W (<1% of total)	bcd_out[2]_i_3 (LUT6)	LUT	N/A	Async	71.514	44.677
<0.001 W (<1% of total)	bcd_out[4]_i_1 (LUT6)	LUT	N/A	Async	37.974	63.179

power_1

XILINX

Summary

XILINX

Summary

- Elaborated design allows early drc checks, noise analysis, and cross-probing between the source file and the generated hierarchical design
- Synthesis performs logic optimization and technology mapping to Xilinx primitives
- Vivado IDE supports VHDL, Verilog, and SystemVerilog
- Vivado IDE supports the use of Xilinx Design Constraint (XDC) syntax, which is used to drive synthesis design optimization
- check_timing report is used to verify that a design has been sufficiently constrained by the designer
- Vivado IDE includes numerous reporting mechanisms for the designer to improve performance, avoid common design mistakes, and get the most out of the FPGA