San Jose State University
Department of Electrical Engineering
EE178, Fall 2018, Crabill/Nguyen

Laboratory Assignment #3

Objectives

This lab is an introduction to the creation of basic constraints files with Xilinx Vivado 2016.2. Basic constraints include specification of pin placements and signaling standards, as well as timing specifications appropriate for designs fully synchronous to a single clock.

No new logic design concepts are presented in this lab. The goal of this lab is for you to become more familiar with the tools. Please read carefully, pay attention, and take your time. This lab is not a race to see who gets done first.

In order to receive credit for this lab, you must complete the tasks and submit your project to the instructor.

Bibliography

This lab draws heavily from documents on the Xilinx website http://www.xilinx.com. I would like to thank Xilinx for making this material available. This lab is effectively a customized introduction to Xilinx Vivado using Verilog-HDL and the Digilent Basys3.

Xilinx Design Constraints

Previously, you have been served basic implementation constraints to copy and paste into a newly created Xilinx Design Constraints file (hereafter referred to as an XDC file) added to your project. It is a common practice to re-use constraints in this manner, particularly for a known hardware platform like the Digilent Basys3.

As you gain confidence in specifying constraints, you may elect to leverage constraints you have used previously, even if they are not an exact match for your needs, and manually edit them in a text editor. However, Vivado provides a powerful capability to interactively create constraints and save them to an XDC file, a feature exceptionally useful to everyone. As a novice, perhaps you know nothing of the available constraints and their syntax – Vivado will guide you. As an expert, perhaps you need advanced capability to query elements in your design to which you will apply complex constraints – Vivado will assist you.

Referring to the previous assignments if necessary, open Vivado and create a new project. After you have created a new project, create a new design source named fsm.v and replace the automatically generated contents of the file with the following:

```
// File: fsm.v
// This is the top level design for EE178 Lab #3.

// The `timescale directive specifies what the
// simulation time units are (1 ns here) and what
// the simulator time step should be (1 ps here).

`timescale 1 ns / 1 ps

// Declare the module and its ports. This is
// using Verilog-2001 syntax.
```

```
module fsm (
  input wire pause,
  input wire restart,
  input wire clk,
  input wire rst,
  output reg [1:0] state,
  output reg odd,
  output reg even,
  output reg terminal
  );

  parameter [1:0] FIRST  = 2'b11;
  parameter [1:0] SECOND = 2'b01;
  parameter [1:0] THIRD  = 2'b10;

  always @(posedge clk or posedge rst)
  begin
    if (rst) state <= FIRST;
    else
    begin
      case(state)
        FIRST:   if (restart || pause) state <= FIRST;
                 else state <= SECOND;
        SECOND:  if (restart) state <= FIRST;
                 else if (pause) state <= SECOND;
                 else state <= THIRD;
        THIRD:   if (!restart && pause) state <= THIRD;
                 else state <= FIRST;
        default: state <= FIRST;
      endcase
    end
  end

  always @*
  begin
    odd = (state == FIRST) || (state == THIRD);
    even = (state == SECOND);
    terminal = (state == THIRD) && (restart || !pause);
  end

endmodule
```

You will notice this is a version of the FSM example discussed in class. Hopefully you understand its behavior, but its value in this assignment is the relative simplicity it provides. We will not be simulating the design, nor attempting to exercise it in hardware – but we will be applying constraints to it.

After you have saved and closed the new design source, create a new constraint source named fsm.xdc. Verify the new constraint file is empty, save it, and close it. You are now ready to work on constraints.

Pin Placement and Signaling Standard Constraints

Most designs have top-level ports that map to device pins. Prior to synthesis, enough information can be extracted from the elaborated design sources to facilitate assignment of top-level ports to FPGA device pins. The FPGA device pins are highly (but not infinitely) flexible, and as a result it is necessary to select analog characteristics of the FPGA device pins (e.g. signaling voltage, output drive strength, etc…)

concurrently with pin placement to ensure the desired combination can be correctly implemented. Under RTL Analysis in the Flow Navigator, click Open Elaborated Design. You may receive a notice that the design elaboration settings allow pin placement and constraint-related work, but can result in longer elaboration times. If you receive this notice, accept it to proceed with elaboration.

The elaborated design opens to an RTL Schematic as shown in Figure 1. RTL stands for register transfer level, a level of abstraction typically higher than the gate level output of a synthesis tool. Take a moment to explore the RTL Schematic and familiarize yourself with the buttons to scroll and zoom.

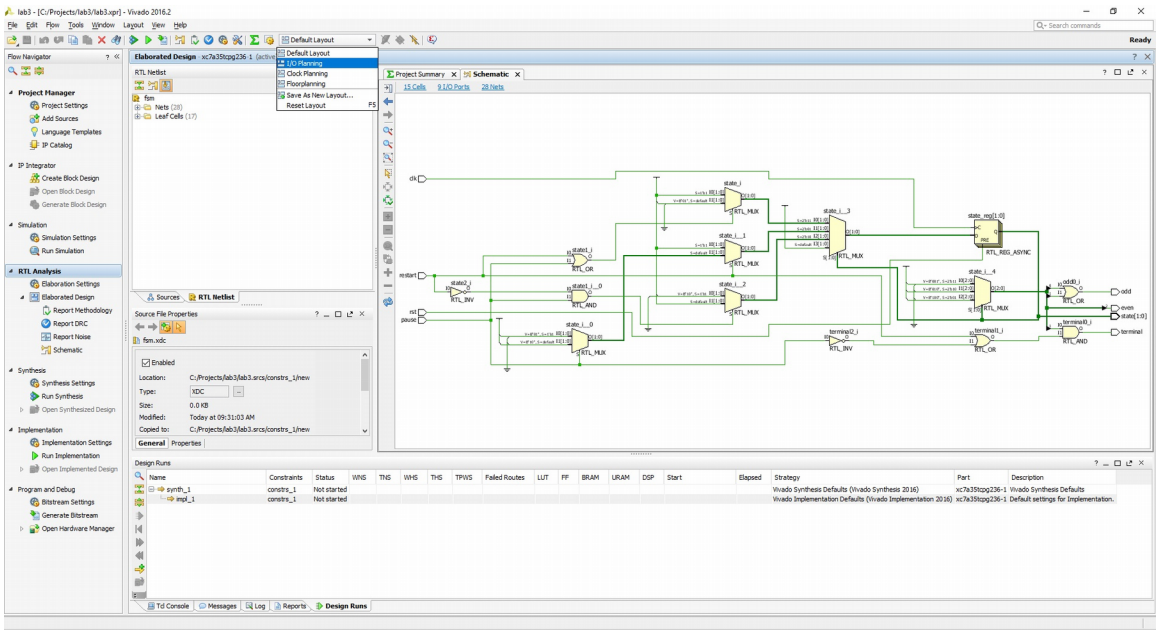

Figure 1: Elaborated Design Schematic

Now, switch from the Default layout to the I/O Planning layout, using the selection box that is also shown in Figure 1 at the top of the window.

The Package View of the device should appear as shown in Figure 2. This is a physical representation of the packaged device you selected during creation of the project. This is a ball grid array package, so it technically has balls, not pins, and they are identified by a row position (a letter, down the left edge) and a column position (a number, across the top edge). If you hover over an individual ball with the pointer, an informational note will appear to provide additional detail.

As you might expect, a number of pins are dedicated to power and ground. Other dedicated pins include those for special analog functions and exchanging programming information and status with the device. As far as programmable pins suitable for mapping to top-level design ports, these are shown as solid gray circles and solid cyan hexagons. Those shown as solid cyan hexagons are “clock capable” and have preferred internal connections to clocking resources.

Also notice the grouping of programmable pins suggested by the coloration of the heavy lines drawn around each group. These groups are called I/O banks, and the programmable pins in a given I/O bank must share certain characteristics such as the I/O power supply voltage.

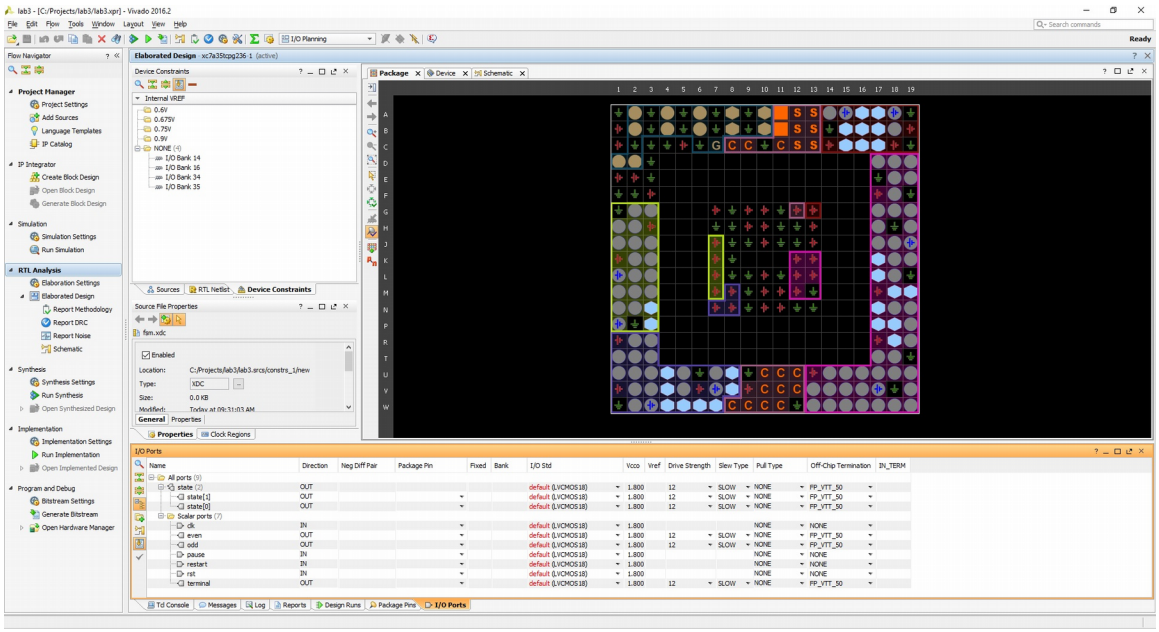

Figure 2: Package Diagram for I/O Planning

If the list of I/O Ports at the bottom of the window is not already expanded as shown in Figure 2, expand it, and start assigning the top-level ports to pins. You can do this by dragging and dropping. Simply select the port symbol next to the I/O Port name, and drag it onto a ball in the Package View. Begin by assigning the clk port to package pin W5. This particular selection is dictated to you to ensure the clk port is assigned to a "clock capable" location, although any other "clock capable" location would be appropriate. In the I/O Ports list, update the other columns for the clk port to match what is shown in Figure 3, which shows the desired end result.

We will use LVCMOS33, a 3.3v LVCMOS standard. In general, we can put an LVCMOS33 pin anywhere we want, but at the point we have made I/O placements, the I/O banks in which the placements have been made will require specific I/O power supply voltages. There are rules that specify what I/O standards can co-exist in a bank, given a specific I/O power supply voltage. For example, you cannot put LVCMOS18, a 1.8v LVCMOS standard, into a bank with LVCMOS33, a 3.3v LVCMOS standard.

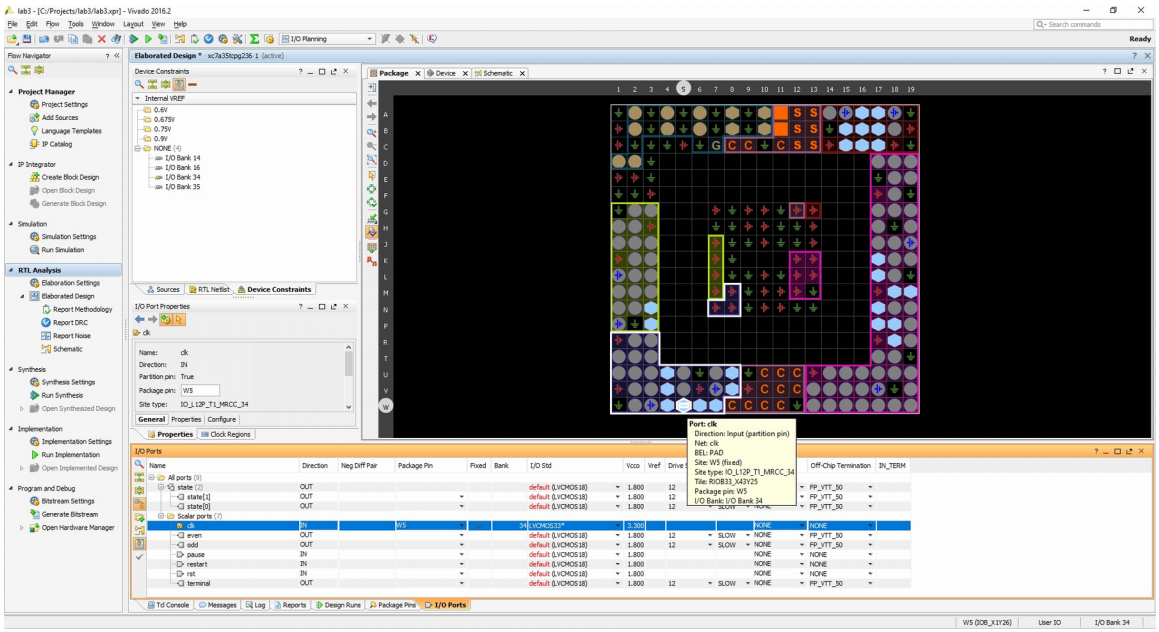

Figure 3: Design Clock Port Assigned to a Clock Capable Pin

Use the drag and drop feature to assign all the remaining I/O Ports to pins. In practice, you might have no freedom in your assignments if the board design for this device has already been created. On the other hand, you may find yourself providing pin assignments to a board designer, or possibly engaged in an interactive pin-planning discussion to yield an optimal result.

For this assignment, you may select pin locations randomly, or strive for a theme such as “close together” or “spread out” (this is a tutorial, we are experimenting). You will notice that Vivado won’t let you drag and drop when the result would cause an I/O power supply voltage conflict. In this case, you may need to change the pin I/O standard before you drag and drop. Figure 4 shows one possible final result – yours may differ.

A random or thoughtless assignment is not a good idea for a real project, because the performance of a design is affected by its pinout – and once you commit to a board layout based on a pinout, it costs time and money to correct bad choices in the pinout.

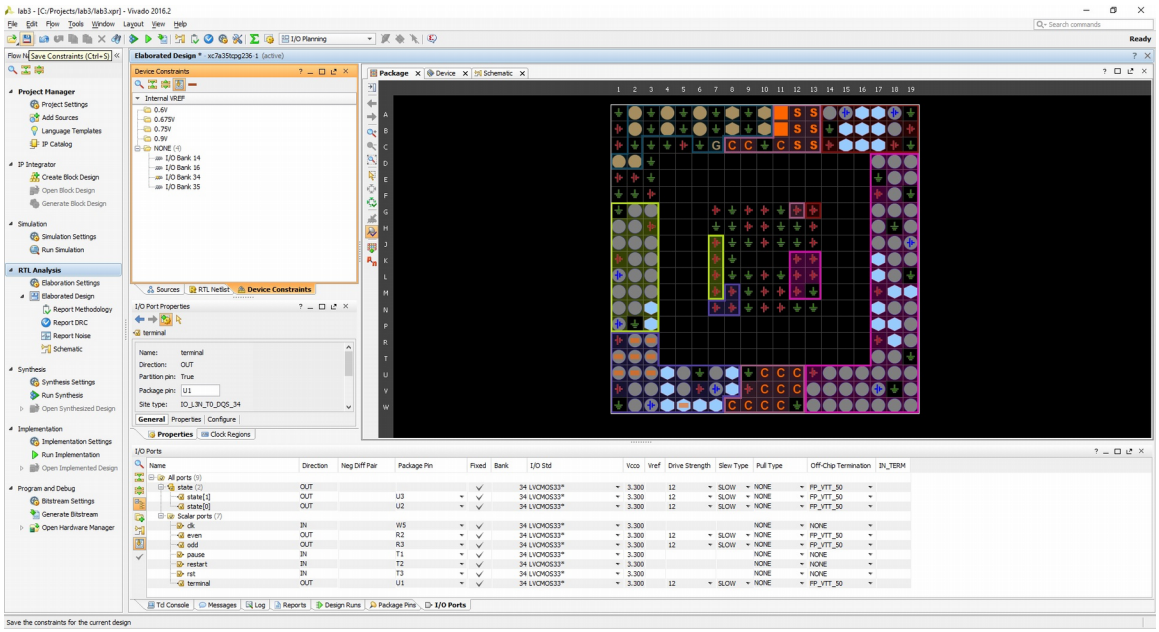

Figure 4: Remaining Design Ports Assigned to Other Pins

When you are done, click on the Save Constraints button in the toolbar, also shown in Figure 4. A dialog box will appear, asking if you want to save the constraints to a new XDC file, or to an existing XDC file in the project. Save the constraints to the fsm.xdc file that's already in the project.

At this time, close the elaborated design by clicking on the "X" in the light blue Elaborated Design title bar.

Fully Synchronous Single Clock Timing Constraints

Timing constraints involve the specification of delays between objects in the design. It is only after the completion of synthesis that all objects in the design are known. Under Synthesis in the Flow Navigator, click Run Synthesis, and when it completes, Open Synthesized Design. You can review the reports later, if you wish.

There are a variety of constraints you can interactively apply to the synthesized design. For interactive timing constraint entry, click on Edit Timing Constraints in the Flow Navigator. This is listed under Synthesis in the Flow Navigator, but you may have to expand the Synthesized Design options to see it.

For synchronous design, timing analysis is fundamentally asking one question: For each signal path that exists from a flip-flop output, possibly through combinational logic, to a flip-flop input, can the successful transfer of information be guaranteed? This type of timing analysis is called static timing analysis because it involves evaluation of signal path delays without regard to circuit activity.

If you have a complex design with lots of paths, this question may need to be posed millions of times to yield a complete result. Nobody has time (or wants) to do this task by hand, which is why static timing analysis tools exist.

Review the FSM structure shown in Figure 5. As mentioned, our interest is in the analysis of paths from flip-flops to flip-flops. Two assumptions are made:

- Inputs to the circuit originate from synchronous flip-flops, although we can't "see" them
- Outputs from the circuit terminate at synchronous flip-flops, although we can't "see" them

With these assumptions, it is possible to analyze all the paths that exist. However, for paths involving inputs and outputs, the fact that we can't "see" everything complicates analysis because it may be necessary to account for additional delays.

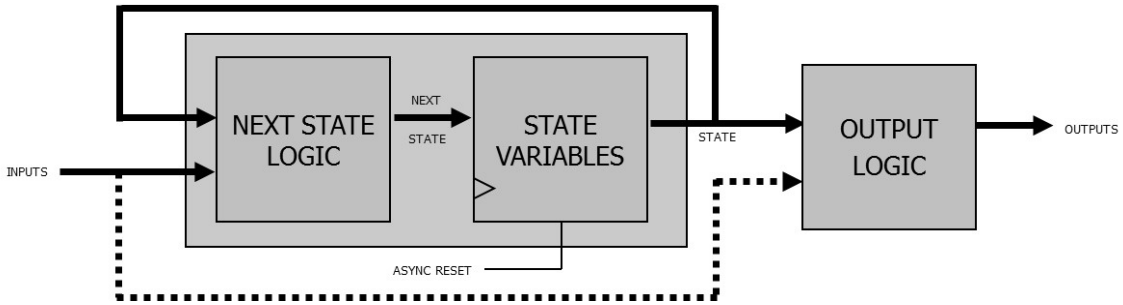

Figure 5: Common FSM Structure

Although we assume the inputs to the circuit are originating at flip-flops, there may be combinational logic delay and wire delay on the paths. Similarly, although we assume the outputs from the circuit are terminating at flip-flops, these paths may also have additional delay. To account for these possibilities, four different path types exist:

1. Clock (period) constraints, covering paths from flip-flops in the design to flip-flops in the design.
2. Input delay constraints, covering paths from flip-flops outside the design which we cannot "see", through inputs, to flip-flops in the design.
3. Output delay constraints, covering paths from flip-flops in the design, through outputs, to flip-flops outside the design which we cannot "see".
4. Path delay constraints, covering paths from flip-flops outside the design which we cannot "see", through inputs to outputs, to flip-flops outside the design which we cannot "see".

As shown in Figure 6, expand the available constraint types in the Timing Constraints window and double click on Create Clock. A series of dialog boxes begin to guide you through the creation of the clock constraint.

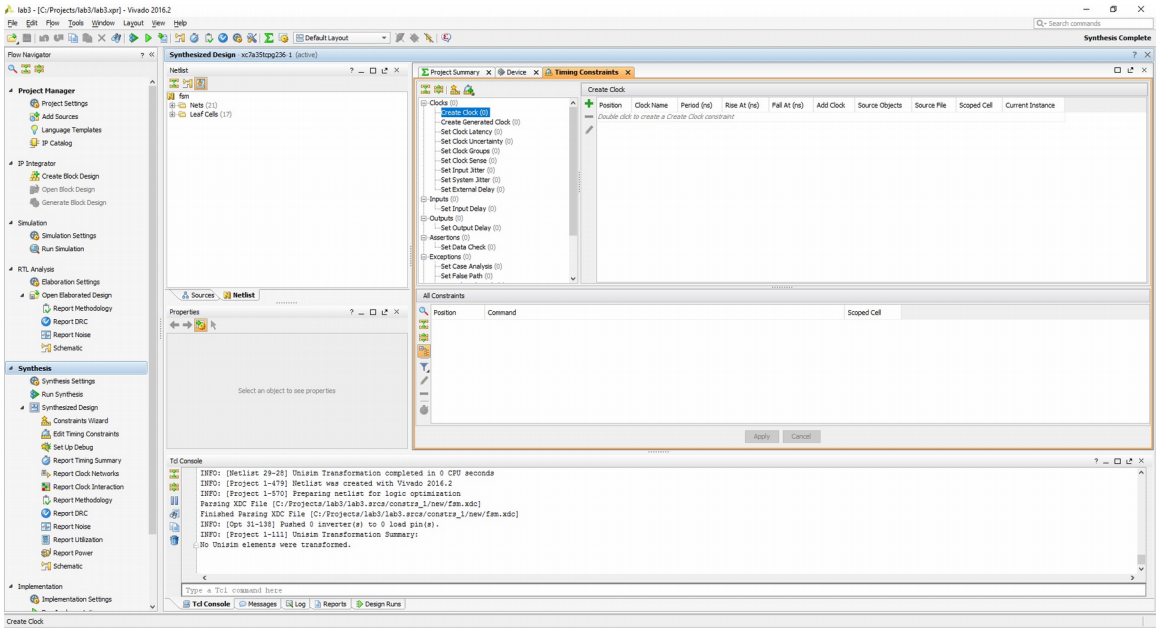

Figure 6: Create Clock – Constraint Creation

The first dialog asks you to provide a friendly nickname for your clock. You will later need to identify where the clock is coming from and sometimes that name be unfriendly or unwieldy, so the opportunity to create a nickname is useful.

In addition to the nickname, the clock waveform parameters need to be specified. A 100 MHz clock with a 50% duty cycle rises at 0 ns, falls at 5 ns, and has a total period of 10 ns. The choice of clock waveform in this exercise is arbitrary, but in a real design it would be based on the actual clock needed for the implementation. Copy the settings shown in Figure 7, and then click on the "…" to specify the source objects.

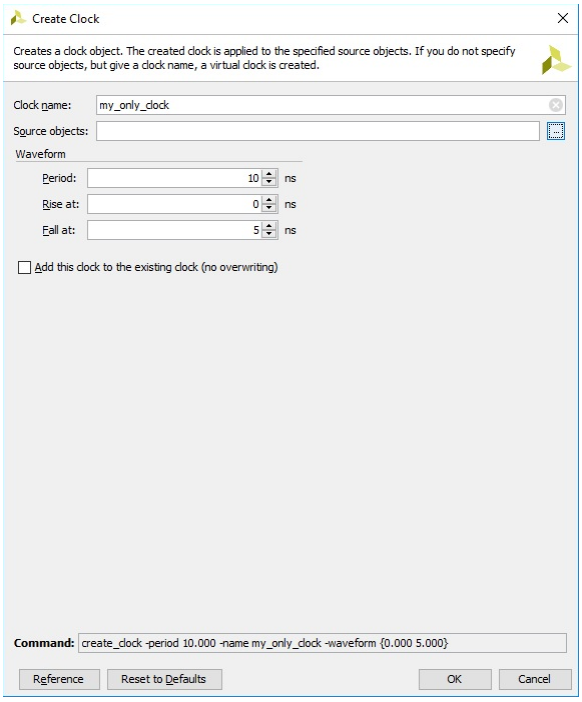

Figure 7: Create Clock – Provide Name and Waveform, then Identify Source

In this design, the clock signal is coming from a top level port, so it happens to have a friendly name and is easy to locate. However, this might not always be the case. For this design, though, use the settings shown in Figure 8 to find design ports that are inputs.

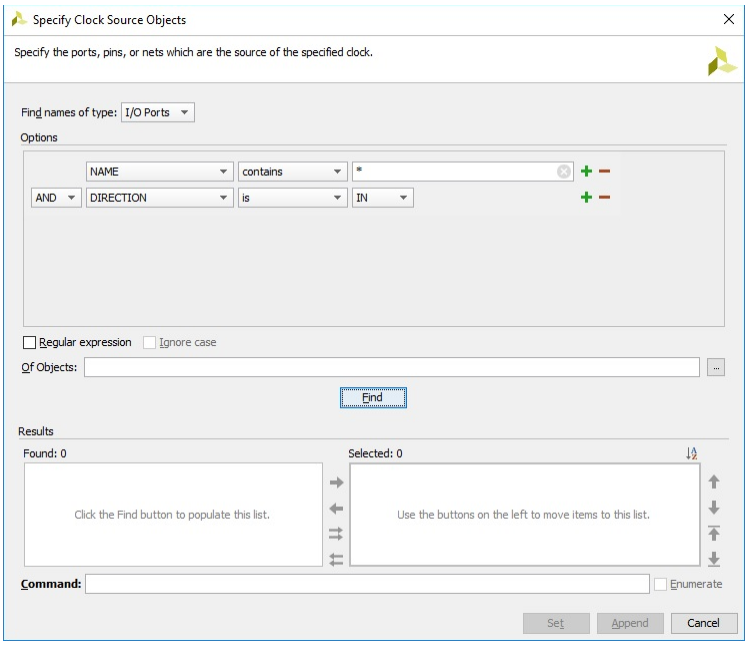

Figure 8: Create Clock – Find Ports

After you click Find, the results should list all the input ports to the design as shown in Figure 9. You need to select the design clock, clk, from the list of ports.

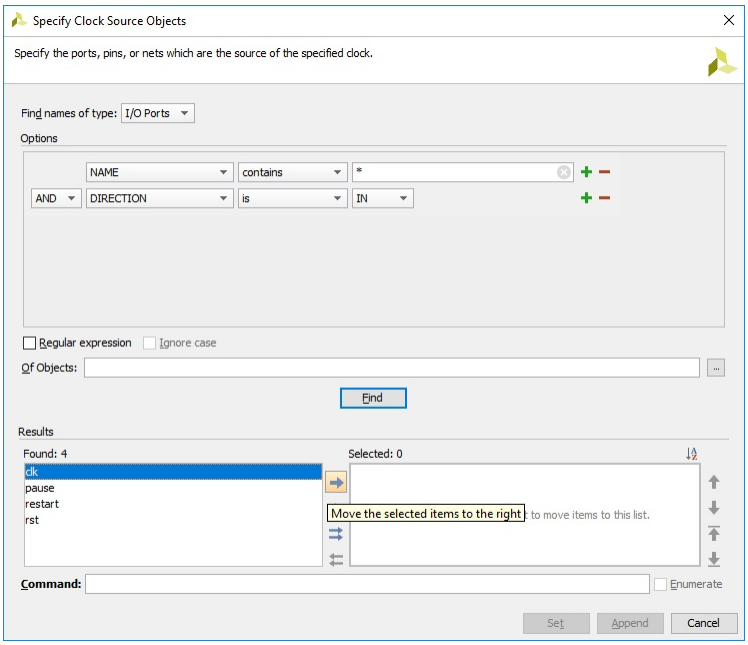

Figure 9: Create Clock – Select Port Sourcing Clock

Promote the clock port to the list of selected objects by clicking on the button with the arrow pointing to the right. The result should look like Figure 10. Then, set the selection to return to the previous dialog.

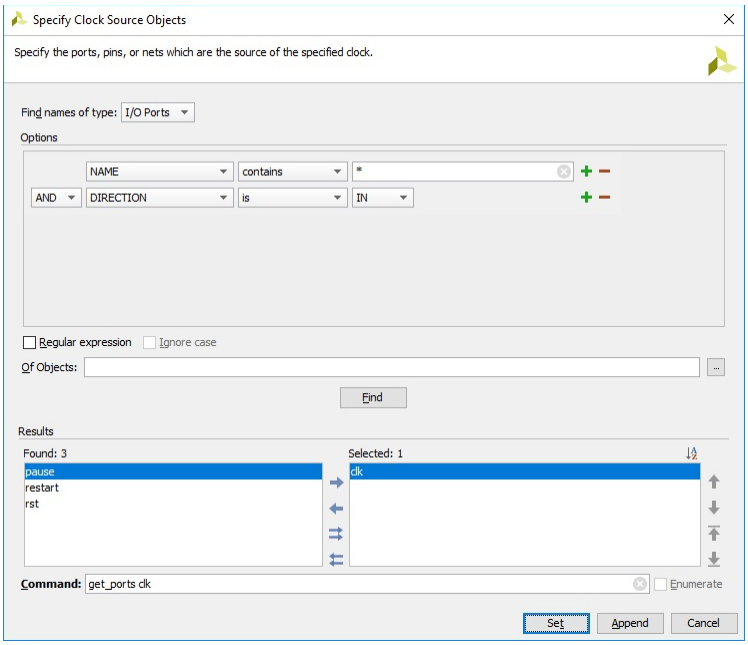

Figure 10: Create Clock – Add Port to Selection List and Accept

When you return to the dialog shown in Figure 11, no further editing is necessary. Review the settings for consistency, and notice the "command" shown at the bottom of the dialog. This is the actual constraint that is generated from the settings. There is no need to copy and paste it; Vivado will add all your new constraints into the XDC file when we have completed constraint entry. Accept the constraint, which will return you to the window shown previously in Figure 6.

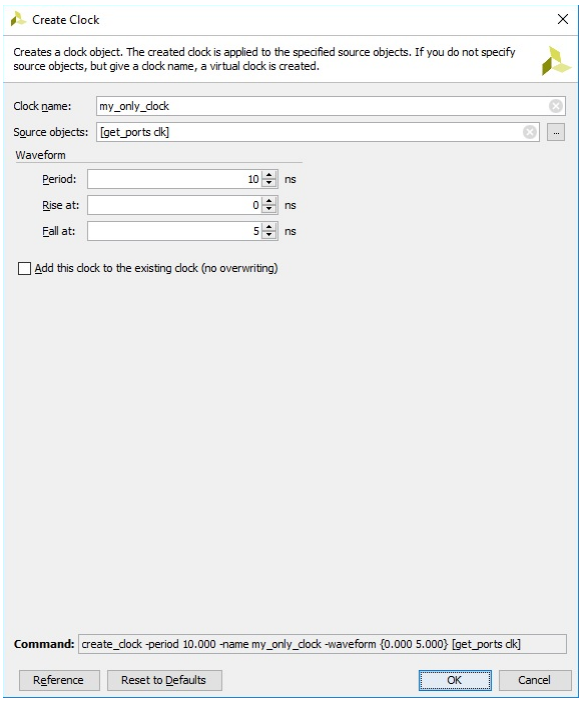

Figure 11: Create Clock – Accept Constraint

Next, create an input delay constraint. We are going to create a single constraint for all inputs, although it is possible to create different constraints for each input. The input delay constraint dialog is shown in Figure 12.

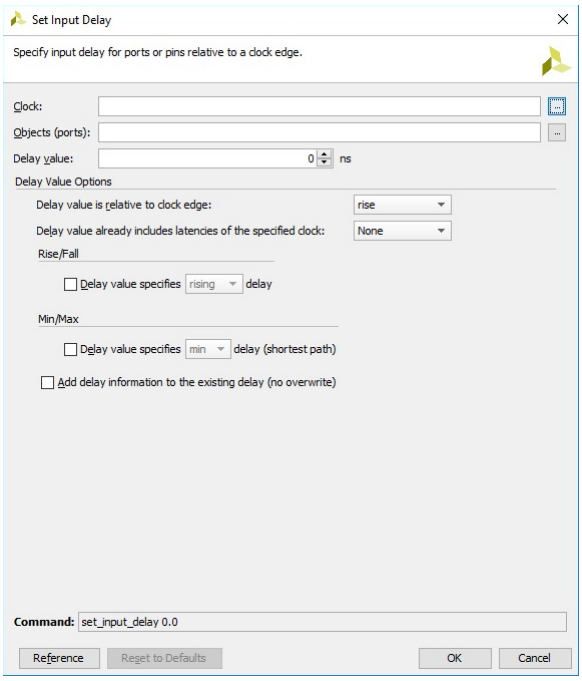

Figure 12: Input Delay – Identify Clock, Ports, and Specify Delay

First, click on the "…" to the right of the clock name textbox. This allows you to search for the design clock related to the input delay constraint. Figure 13 shows how to find clocks in your design, you should only find one, it will be the clock you identified while creating the clock constraint, and it will be listed by its nickname. Promote the clock to the selection list and accept the selection. You will return to the main input delay dialog.

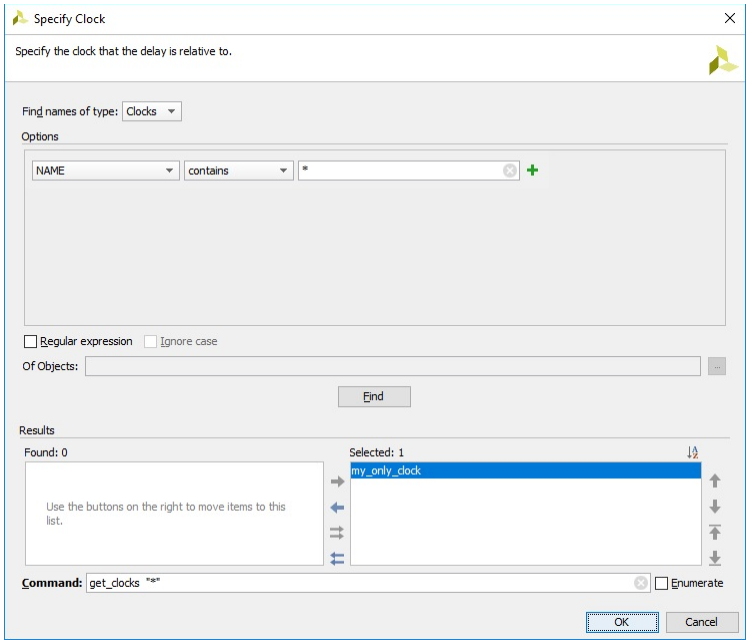

Figure 13: Input Delay – Identify Clock and Accept

Next, click on the "…" to the right of the objects name textbox. This allows you to search for the design ports related to the input delay constraint. Figure 14 shows how to find ports in your design, you should select pause and restart ports only, and promote them to the selection list. Accept the list of selected ports.

Although the clk port is listed in the port search results, it is not (usually) a data signal so you would not want it in the list of selected ports.

The rst port is also listed in the port search results, it is an asynchronous reset. If we assume the source of it is truly asynchronous, we can ignore the timing on this signal because we have no idea when it will assert or for how long it will assert, and it won't be possible to guarantee any particular behavior through a timing constraint. Note, however, that sometimes designers drive asynchronous control signals from synchronous sources such as flip-flops, and may wish to analyze the timing by adding a constraint.

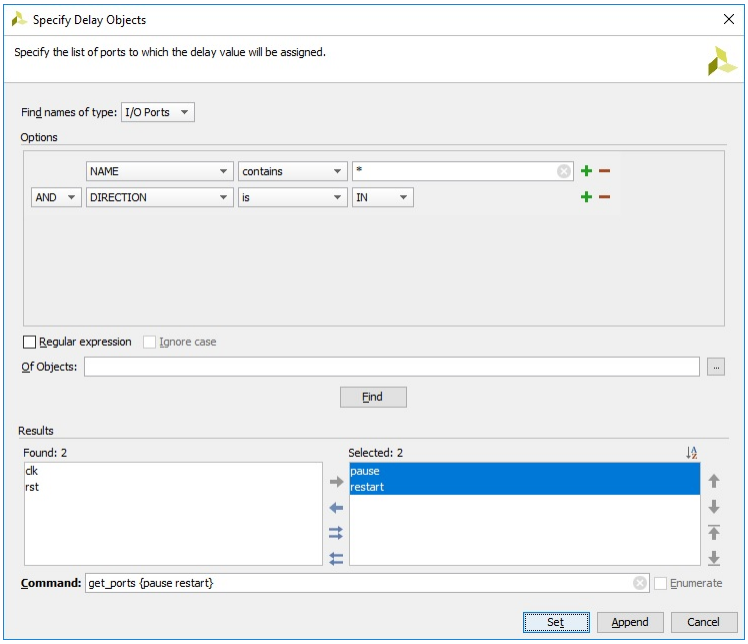

Figure 14: Input Delay – Identify Ports and Accept

When you return to the dialog shown in Figure 15, specify the delay value as 5 ns, as shown. This delay value is how you express the delay that cannot be "seen" from the source flip-flops outside the design up to the input ports of the design. The remainders of the paths from the input ports to flip-flops inside the design are visible to the timing analysis tool. The 5 ns is arbitrary, but a fair sharing of the 10 ns clock period between our design and some other invisible design that cannot be "seen".

Review the settings for consistency, and notice the "command" shown at the bottom of the dialog. This is the actual constraint that is generated from the settings. There is no need to copy and paste it; Vivado will add all your new constraints into the XDC file when we have completed constraint entry. Accept the constraint, which will return you to the window shown previously in Figure 6.

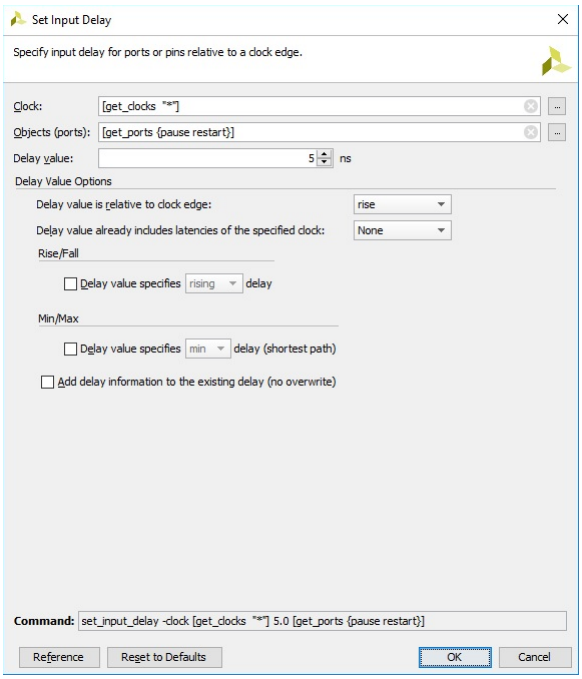

Figure 15: Input Delay – Specify Delay and Accept

Next, create an output delay constraint. We are going to create a single constraint for all outputs, although it is possible to create different constraints for each output. The output delay constraint dialog is shown in Figure 16.

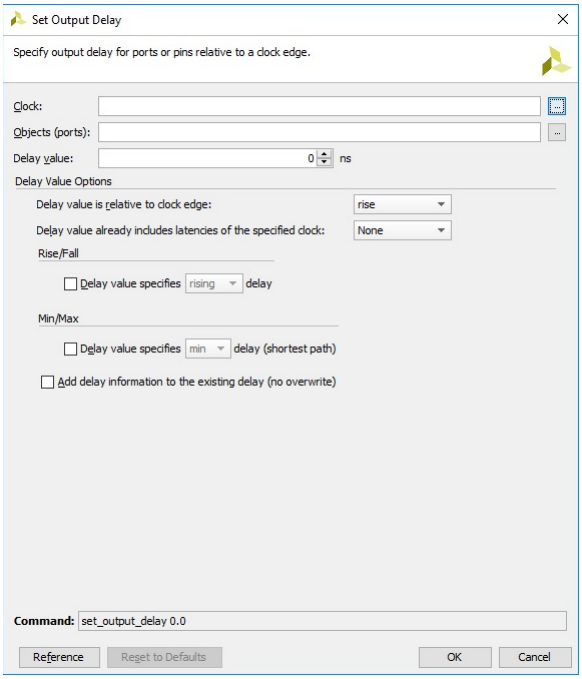

Figure 16: Output Delay – Identify Clock, Ports, and Specify Delay

First, click on the "…" to the right of the clock name textbox. This allows you to search for the design clock related to the output delay constraint. Figure 17 shows how to find clocks in your design, you should only find one, it will be the clock you identified while creating the clock constraint, and it will be listed by

its nickname. Promote the clock to the selection list and accept the selection. You will return to the main output delay dialog.

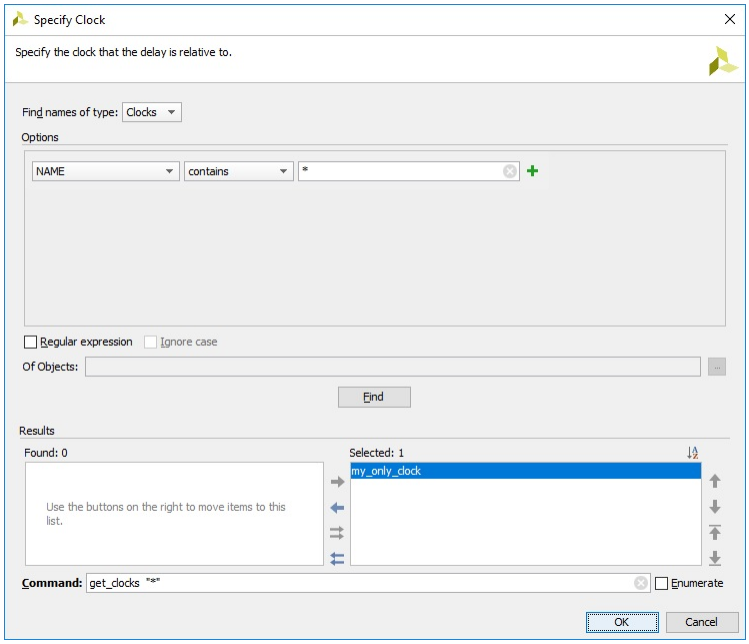

Figure 17: Output Delay – Identify Clock and Accept

Next, click on the "…" to the right of the objects name textbox. This allows you to search for the design ports related to the output delay constraint. Figure 18 shows how to find ports in your design, you should select all the output ports, and promote them to the selection list. Accept the list of selected ports.

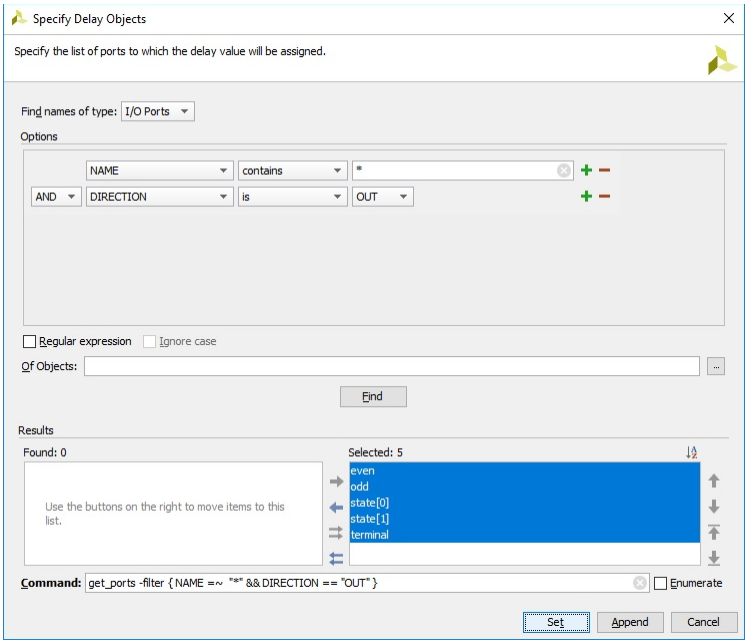

Figure 18: Output Delay – Identify Ports and Accept

When you return to the dialog shown in Figure 19, specify the delay value as 5 ns, as shown. This delay value is how you express the delay that cannot be "seen" from the output ports of the design up to the flip-flops outside the design. The remainders of the paths from the flip-flops inside the design to the output

ports are visible to the timing analysis tool. The 5 ns is arbitrary, but a fair sharing of the 10 ns clock period between our design and some other invisible design that cannot be “seen”.

Review the settings for consistency, and notice the “command” shown at the bottom of the dialog. This is the actual constraint that is generated from the settings. There is no need to copy and paste it; Vivado will add all your new constraints into the XDC file when we have completed constraint entry. Accept the constraint, which will return you to the window shown previously in Figure 6.

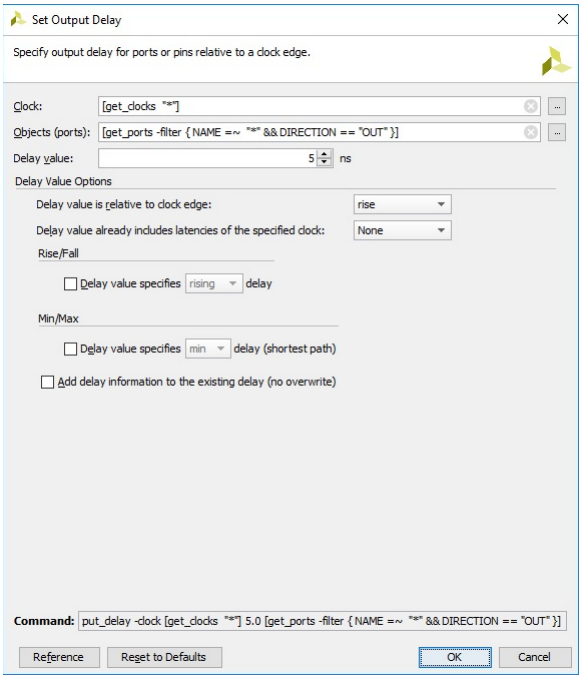

Figure 19: Output Delay – Specify Delay and Accept

At this point, you should see the three new constraints you have entered, listed under All Constraints in the Timing Constraints window, as shown in Figure 20. Below the main menu bar, you will find a small disk icon to save your constraints; click on it.

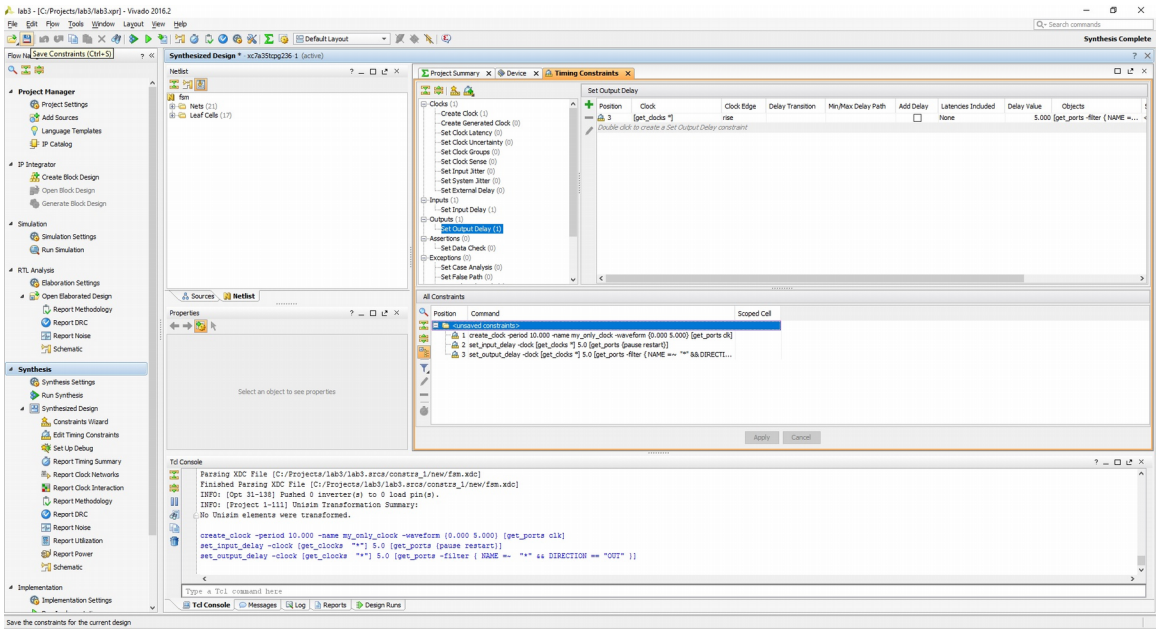

Figure 20: Save Timing Constraints

As you have created constraints, the synthesis results may now be out-of-date, and Vivado will display an Out of Date Design warning. This is because some constraints have an effect on the synthesis results. For this exercise, we will completely re-run synthesis to ensure all constraints have been applied during synthesis. Dismiss this message, but accept any subsequent message to confirm saving of the new constraints.

At this time, close the synthesized design by clicking on the "X" in the light blue Synthesized Design title bar.

Reset Design Runs, Re-Run with Constraints

Open the fsm.xdc constraint file in the text editor. You originally added a blank file, now it is populated with the constraints you applied. Add the following additional constraints manually and then save the file:

```
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
```

The completed constraints should look similar to Figure 21. In the bottom window, select the Design Runs tab; right click on the synthesis run. The context menu provides an option to Reset Runs. Reset the synthesis run. Resetting the synthesis run allows you to re-run it from scratch.

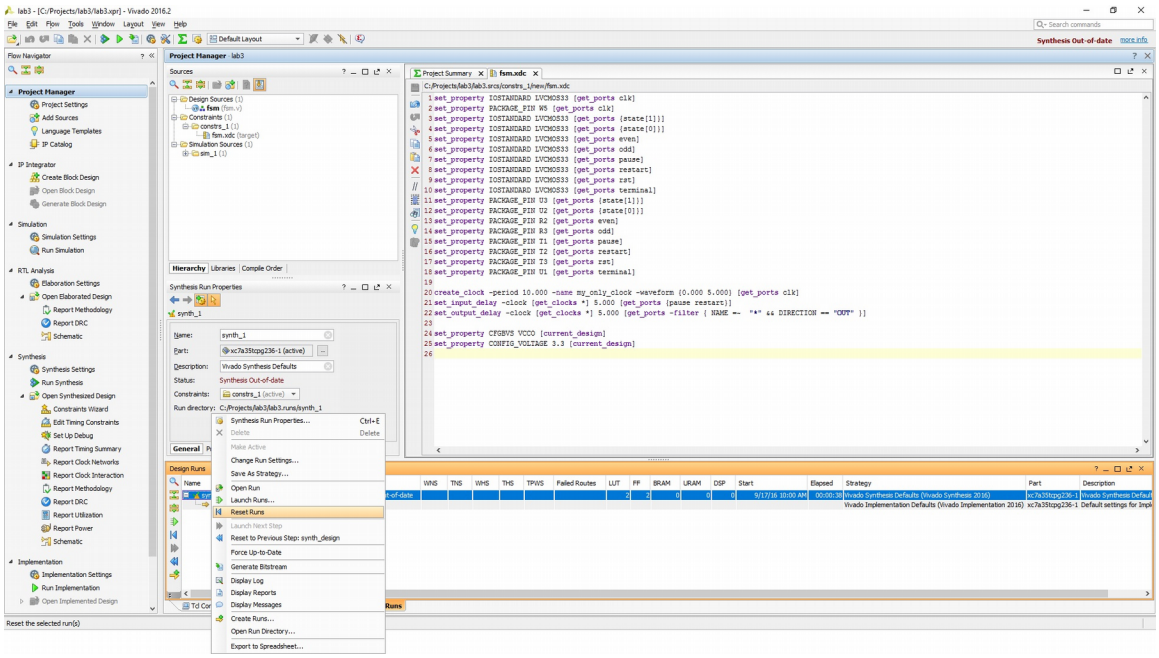

Figure 21: Inspect XDC File, Reset Design Runs

When you select to reset a design run, you are prompted for confirmation. Accept the confirmation request and be sure to delete the generated files in the working directory.

In the Flow Navigator, click on Run Implementation. Since you have reset the design run, this will re-run all of the design flow steps up through implementation. When the implementation run has completed, open the implemented design. You can review the reports later, if you wish.

Reviewing Implementation Results

The initial presentation of the timing results for the implemented design, shown in Figure 22, has a navigation tree on the left and a high level summary on the right. Expand the navigation tree, keeping the Design Timing Summary selected. Failures are indicated by the use of red text. The navigation tree indicates that failures exist on intra-clock paths of the declared clock signal my_only_clock, and those failures are setup failures.

Positive slack on a timing constraint is the portion of the constraint that has no “strain” on it, indicating the actual timing of the path being constrained exceeds the requirement by some amount. This is generally what you would consider a satisfied or passing constraint. If you achieve positive slack on all constraints, it is called timing closure.

Negative slack, on the other hand, is generally undesired and indicates the path being constrained is not meeting the requirement by some amount. If you have more than one path under analysis, the term worst negative slack (sometimes referred to as WNS) is the negative slack of the greatest magnitude and may be a useful guide to where the design needs work to achieve timing closure. The term total negative slack (sometimes referred to as TNS) is the sum of all negative slacks in the design, and may be a useful guide to how much the design needs work to achieve timing closure.

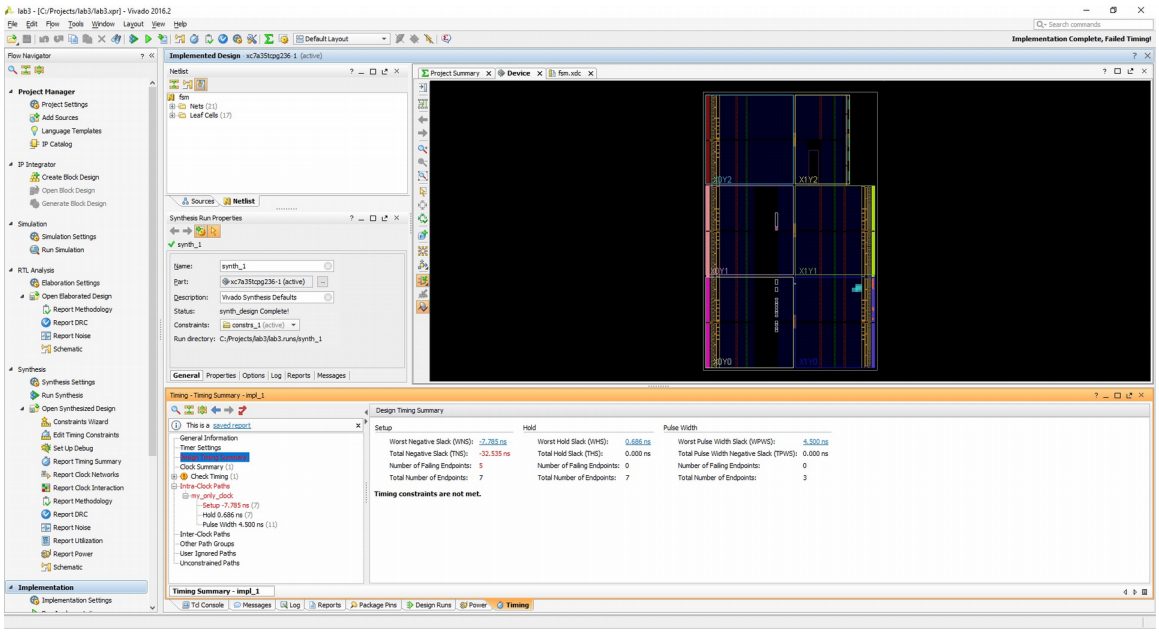

Figure 22: Timing Summary, Constraints Failing

As shown in Figure 23, in the navigation tree under Intra-Clock Paths for my_only_clock, select the failing Setup category. This will display a list of paths in this category. The list of paths may be sorted by several characteristics. By default they are sorted in ascending order of slack (meaning, large negative slacks at the top of the list, because these show where the design needs improvement).

Also notice, in Figure 23, that selection of a path in the list generates arrows on the device view to provide insight into the physical path. You can zoom and scroll to look at these arrow diagrams more closely. When you are evaluating why a path is failing its constraint, understanding its physical path may provide insight into the failure – or it may not. Please note your path may not match what is shown in the figure.

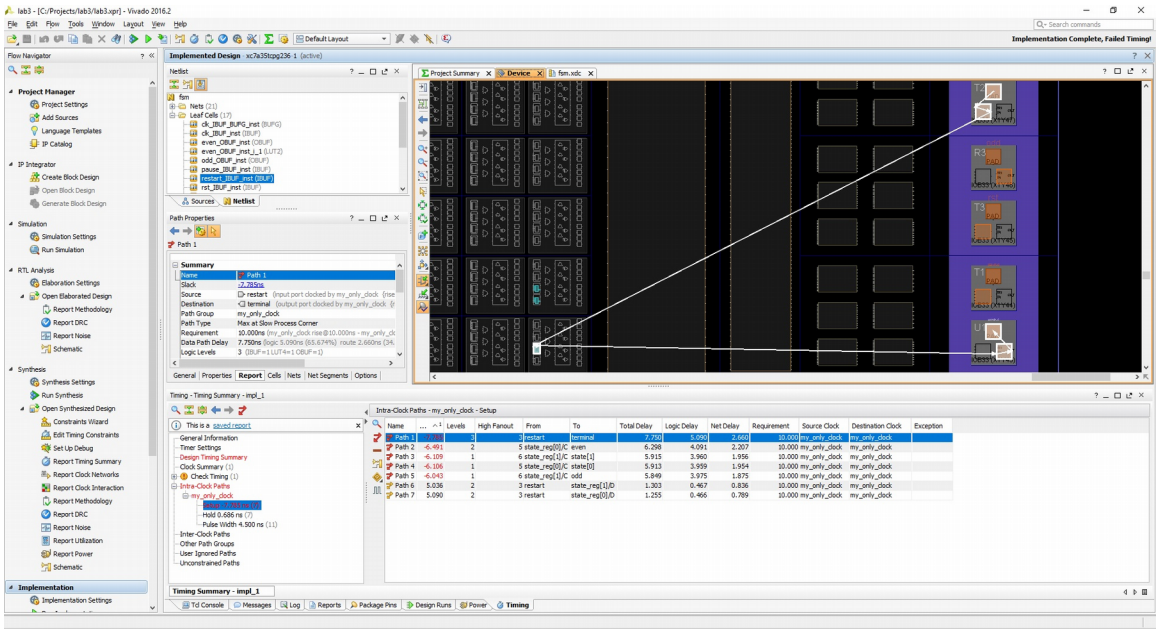

Figure 23: Setup Paths, Worst Offender, Graphical Illustration

Double click on a path in the timing summary list to open a detailed analysis of that path. Figure 24 provides an example of a detailed analysis. We will discuss these reports in class.

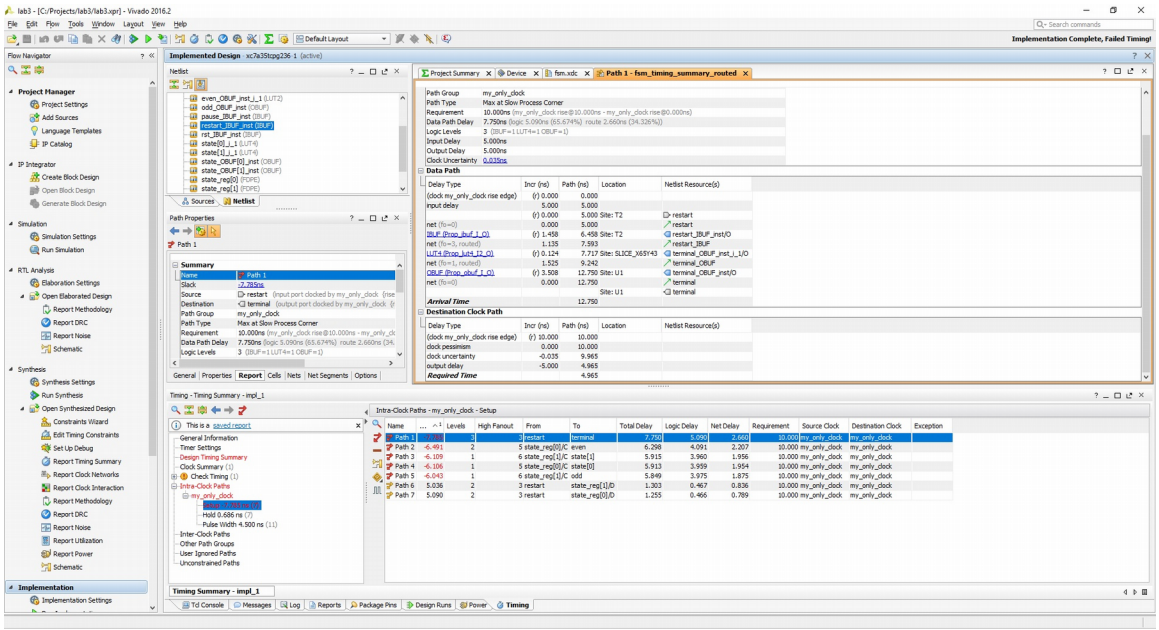

Figure 24: Worst Offender, Timing Detail

Laboratory Hand-In Requirements

Once you have completed this exercise, prepare for the submission process. Within four hours of the scheduled class time on the due date, you are required to submit your entire project directory in the form of a compressed ZIP archive. Use WinZIP to archive the entire project directory, and name the archive lab3_nguyen_tan.zip. For example, if I were to make a submission, it would be lab3_crabill_eric.zip. Then email the archive to the instructor. Only WinZIP archives will be accepted.