Basic XDC Constraints

We know our FPGA has hundreds pins that connect to various electronic components, How do we access them? To access the pins we'll simply need to create a constraints file. A constraints file will let our Xilinx or Altera software tools know what pin we want to connect our ports to. To determine what pin connects where, you'll have to look at your development boards data sheet or master XDC file. Figure 1 provides an example of what an XDC constraint file would look like for the VHDL next to it. Please take a few minutes to memorize the syntax. To assign a pin we use keywords "set_property package_pin" followed by the pin location. We then specify what port we want our pin to connect to by using the "get_ports{port name}" function. Notice the name of the ports in the XDC file have the same names as the ports in our entity. We must make sure the names match, or our software tools won't know what we're tying to do. You'll also notice the ports in our entity are of type std_logic_vector. When we use vectors we must give every bit a pin location with the "[bit number]" syntax. Figure 1 highlights the pin location, the port name and the syntax for providing each bit of the port. If our ports were of type STD_LOGIC we wouldn't need to specify a particular bit, so we would simply use the port name without the "[pin number]". Once again, to find the pin location you must use the master XDC file provided by the vendor. Finally, you'll notice a line specifying the io standard.The io standard sets the voltage of the pins. Please check your data sheet to see what standards your pins can support. For discrete pins you'll typically use "LVCMOS33" for 3.3 volts and for differential you'll use "LVDS". There is also an "LVCOMS18" for 1.8 volts if needed.

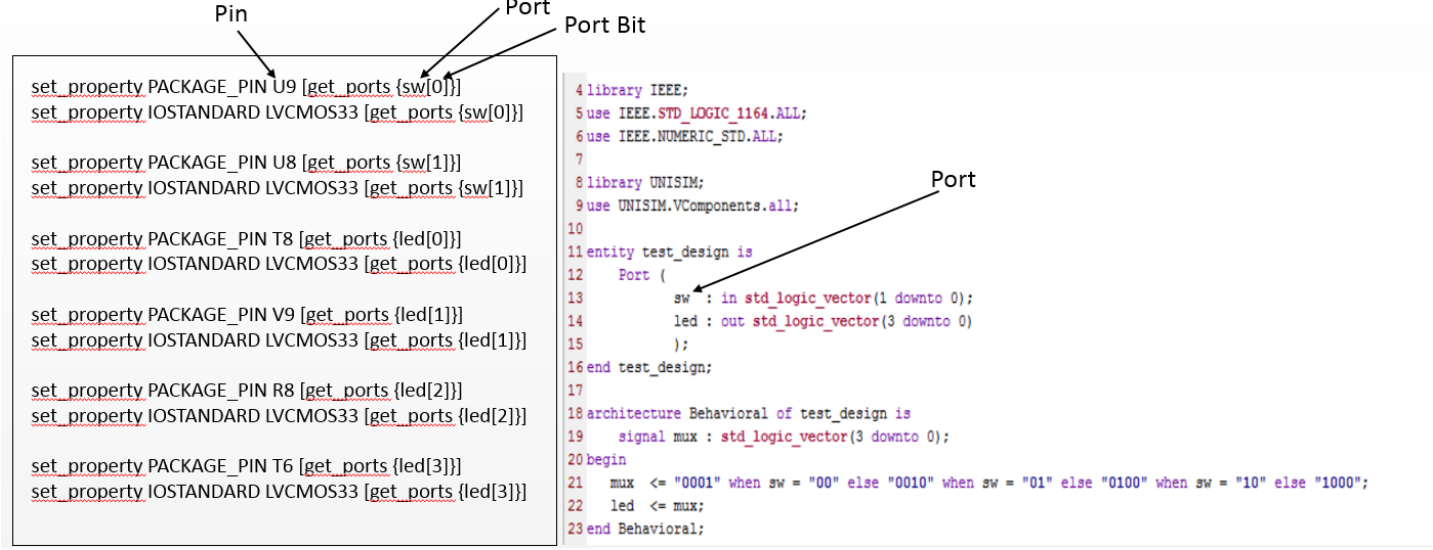

Figure 1: XDC Constraints

Another type of constraint you'll need is a clock constraint. Clock constraints allow you to constrain the frequency, phase and duty cycle. Vivado will use clock constraints to make sure your design meets timing. Meeting timing is probably the biggest hassle and you'll find out when you get out college and make into the industry. It's important to know you only need to constrain primary clocks. If you generate secondary clocks out of an MMCM, or with the clock wizard ip, vivado will derive the constraints for you behind the scenes. Figure 2 provides an example displaying how to constrain a clock. The period is in nanoseconds and the waveform parameters provide the absolute rise and fall times. By changing the rise time you can adjust the phase shift, or how many nano seconds before the clock starts. The fall time will allow you to adjust the duty cycle by specifying how many nano seconds before the clock falls. The period will of course allow you to adjust the clocks frequency. In this example the clock has a 5ns period, 0ns phase shift/rise time and a 5ns fall time. If the clock is differential and identifiable by an "_n" or "_p", then you should only have to use the create clock constraint for either the _p or _n, not both (clk_p,clk_n). However, you should provide a pin location and io standard constraint for both. You'll probably want to use LVDS as the standard if you're using a differential clock. On a final note, if you leave the -waveform{} out of the create clock constraint, vivado should default to a 0ns phase shift and 50 percent duty cycle. If you're new and unsure don't include the waveform parameters and let the tools default for you.

```
create_clock -add -name clock -period 5.00 -waveform {0 5} [get_ports clk]
set_property PACKAGE_PIN Y10 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
```

Figure 2: Clock Constraint