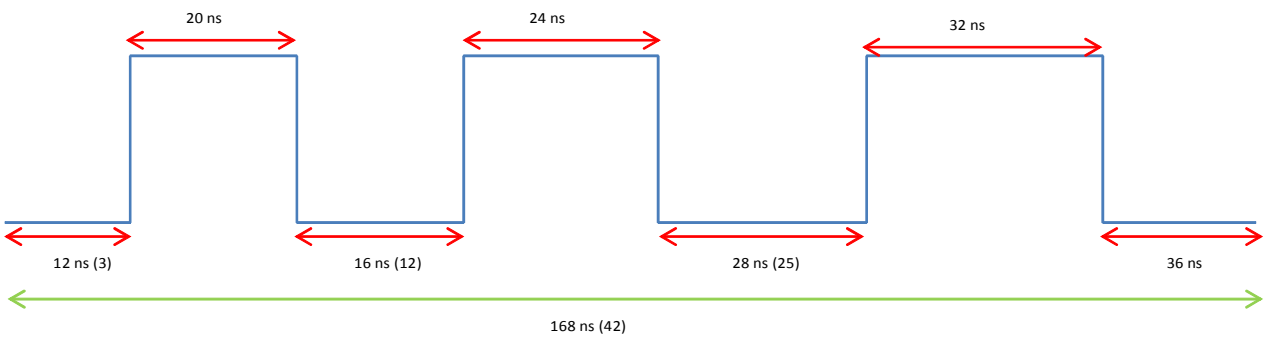
Custom PRI Testbench

This is a write up describing the stimulus in the testbench for generating custom pri trigger. Line 1000 in ***ii_x6_1000m_tb.vhd*** is where the stimulus area begins. The testbench provides a way to simulate wishbone reads and writes with app_read and app_write procedures respectively. The register values; for the reads and writes; are replaced by constant names to improve readability. The constant declarations can be found in ***x6_1000m_pkg.vhd***. Generics defined in the entity declaration allow to select which data flow direction gets simulated and its trigger mode.

The comments should describe what is happening in the section below it. For example, in the regiser sanity test section, a register is written to, and the value read back and compared to what was written to make sure they match.

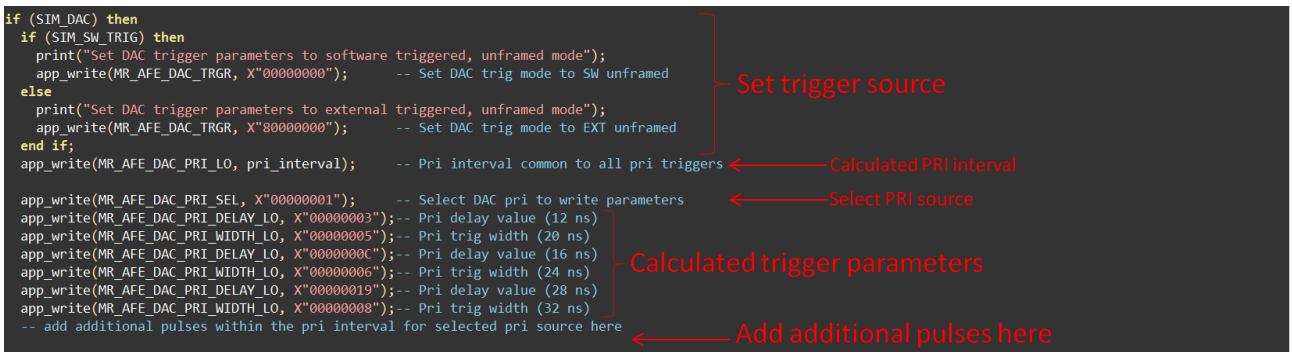
After the Initial device setup, the PLL spi interface is exercised and after configuring it, we wait for the model to indicate PLL lock. We then setup the ADC and DAC interfaces if their respective generics are set to true. In the ADC setup, they are powered up and we wait for their SPI interface to become ready. The channels are then enabled and the enable calibration bit is pulsed. The calibration process causes the adc model to output the forwarded ADC clock. When the calibration is done, we pulse app_rst to reset logic running on this forwarded clock and run calibration again to finish setting up the ADC interface.

The DAC setup is simpler. The DAC reset is toggled and the DAC SPI interface is exercised. We then enable the DAC channel and set it in the desired test mode. We then set up the ADC and DAC PRI parameters and assert the desired trigger. Calculating the values that need to be written for delay and width is as follows.



From the figure above we can calculate the PRI interval as 168 ns which is 42 counts and we write 41 to the interval register. The pulse width values are the desired width divided by 4 ns. The delay values are agrregate delays from the start of pri interval. In the above figure the delays are

1. 12 / 4 = 3
2. (12 + 20 + 16) / 4 = 12
3. (12 + 20 + 16 + 24 + 28) / 4 = 25
4. The last delay is factored in the pri interval.



The screenshot above shows the sequence in which the trigger parameters are written. We set the trigger source based on the value of the generic. The pri interval is same for all the pri sources so it is written next. We then select which pri source the parameters are written to. Bit '0' selects the trigger going to the interface while bits 1, 2, 3 and 4 select each of the trigger dios. We then proceed to write the delay and width values for all the pulses we need for the selected source. NOTE: THE DELAY VALUE NEEDS TO BE WRITTEN BEFORE THE WIDTH VALUE.

If you need to add pulses, you can do so under the comment. Keep in mind, you will have to widen the PRI interval appropriately if you choose to do so. After the parameters for all the sources are stored, we enable the PRI.

The if-else tree starting in line 1241 asserts the selected trigger and then we deassert it after 100 ns. We can do this because the PRI engine stops only when it is disabled, which is done after 5 us and the simulation is halted 1 us later.