# Lab 09

## Part A – Adder/Subtractor Timing

### Description

In this lab module, you will modify the constraint file for the 8-bit adder/subtractor description from 08C to determine the fastest possible timing.

You will not need to use the FPGA development board for this lab module.

## Procedure

1. Download `full_adder.vhd`, `half_adder.vhd`, `lab09a.vhd`, and `lab09a.xdc` from Canvas and place them in a new folder titled `lab09a`.
2. Open Vivado and create a new project titled appropriately.
    - For your convenience, the board identifier is `xc7a35tcpg236-1`.
        - You can also find the board with these options: `General Purpose`, `Artix-7`, `cpg236`, `-1`.
3. Open `lab09a.vhd` in the editor, and follow along with the instructor to review the changes to this hardware description from 08C.
    - The most important change is the addition of registers before and after the hardware. This will allow timing analysis of the combinational logic between the two registers.
4. Open `lab09a.xdc` in the editor. There is only one constraint in this file: a timing constraint.
    - In this lab module, you will modify the timing constraint to maximize the clock speed.
    - Note: changing this constraint does not change the *actual* clock speed on the FPGA development board, which is generated by a 100 MHz crystal oscillator. This only changes the clock Vivado expects the FPGA to receive on that pin, and consequently Vivado's timing analysis, which is what we will investigate in this lab module.

5. Run synthesis and implementation with the constraint file as is, but do not generate the bitstream. Look at the `Design Runs` pane at the bottom of the screen, and pay attention to the `WNS` and `WHS` statistics.
   - `WNS` stands for *Worst Negative Slack*, and `WHS` stands for *Worst Hold Slack*.
   - These metrics compare the critical path to the timing constraint (clock period).
     - It follows that using a faster clock will typically decrease these values, but we want to avoid negative values (displayed in red) as this means the design has failed timing.
       - In our case, that means the timing constraint is too aggressive, so you would need to increase the clock period (slow it down).
6. As the constraint is now, at a period of `10.000` ns (100 MHz), the design will pass timing, but we can do better. Change the constraint file as follows: Change `-period 10.000` to `-period 5.000` and `-waveform {0.000 5.000}` to `-waveform {0.000 2.500}`.
   - The `-period` argument specifies the clock period in nanoseconds. We'll halve it and see how that affects the timing.
   - The `-waveform` argument contains the edges of the clock signal waveform. `{0.000 5.000}` means it has an edge at 0 ns and an edge at 5 ns, so it is active half of the time.
     - This is also called a *duty cycle* of 50% (in this case).
7. Repeat steps 5 and 6 with lower clock periods until the design fails to meet timing. Take a screenshot of the implementation results at this time.
8. Next, try slightly higher clock periods until the design meets the timing constraint. Take a screenshot at the lowest clock period that meets timing.
   - Hint: a binary search type approach may be the most effective – divide the clock period in half until you cannot anymore, then increase by half of the last increment, and repeat.

## Deliverables

- Include as part of your **informal report**:
  - A screenshot of the run results for the first failed timing run and the last successful timing run.

## Outcomes

- Practice working with VHDL.
- Practice using Vivado for hardware synthesis and implementation.
- Understand VHDL components, instantiation, and structural connections.
- Understand elements of timing analysis including clock constraints and critical paths.