Lab 04

Part B – Floating-Point Division

Description

This activity simulates an implementation of floating-point division.

Procedure

Background

- In this module, we will use the simplified floating-point number form introduced in the lecture.
  - Consider a number N, whose *fraction part* F and *exponent part* E are both **4-bit two's complement numbers**.
  - We will use two numbers of this form:

$$N_1 = F_1 \times 2^{E_1} \quad \text{and} \quad N_2 = F_2 \times 2^{E_2}$$

- Dividing two floating-point numbers is basically multiplying the fraction part of the first number, F_1, by the reciprocal of the fraction of the second number, ${F_2}'$, and subtracting the exponents.

$$N = \frac{F_1}{F_2} \times 2^{(E_1 - E_2)} = F_1 \times {F_2}' \times 2^{(E_1 - E_2 + a)}$$

  - where ${F_2}' = \frac{1}{F_2}$ and a is the *normalization factor* for the exponents.
- The normalization factor is used to keep all numbers within the same 4-bit two's complement format.
- The tables below on the next page show all possible positive and negative values, respectively.

F (decimal)	F (binary)	F_2' (decimal)	F_2' (binary)	a	$F_2' \times 2^a$ (decimal)
0.5	`0.100`	2	`0.100`	2	2
0.625	`0.101`	1.6	`0.111`	1	1.75
0.75	`0.110`	1.333	`0.110`	1	1.5
0.875	`0.111`	1.1429	`0.101`	1	1.25

F (decimal)	F (binary)	F_2' (decimal)	F_2' (binary)	a	$F_2' \times 2^a$ (decimal)
-1	`1.000`	-1	`1.000`	0	-1
-0.875	`1.001`	-1.1429	`1.011`	1	-1.25
-0.75	`1.010`	-1.333	`1.010`	1	-1.5
-0.625	`1.011`	-1.6	`1.001`	1	-1.75

Project Creation and Hardware Review

1. Download the VHDL file `lab04.vhd` from Canvas and place it in a new folder titled `Lab04.
2. Open Vivado and create a new project.
    - From the Quick Start menu, select `Create Project`.
    - Click `Next`.
    - On the second page, if desired, give the project an appropriate and change the location. Click `Next`.
    - Ensure `RTL Project` is selected. Click `Next`.
    - On the `Add Sources` page, click `Add Files` and add `lab03.vhd` from where you downloaded it.
    - Ensure `Copy sources into project` is checked. Click `Next`.
    - Click next – constraints are not needed for this lab.
    - Select the following:
        - Category: `General Purpose`
        - Family: `Artix-7`
        - Package: `cpg236`
        - Speed: `-1`
    - Finally, select the middle option: `xc7a35tcpg236-1`.
    - Click `Next`.
    - Click `Finish`.

3. When Vivado finishes creating the project, run synthesis.
    - From the left side pane, click `Run Synthesis`.
        - If a popup appears, click `OK`.
    - After synthesis completes, you may see a popup with the options `Run Implementation`, `Open Synthesized Design`, and `View Reports`.
        - Here you can either click `Cancel`, or select `View Reports` and click `OK`.
4. While we wait for synthesis to complete, open the file in the editor, and follow along with the instructor to review the hardware description.
    - To do this: inside the `Project Manager` pane, under the `Sources` subpane, and under the `Design Sources` folder, double-click the `lab03.vhd` file.

Hardware Simulation

5. Make a copy of the template datasheet for your lab group's simulation results: https://docs.google.com/spreadsheets/d/19YSJcf-SXu-2kO7T7ZpZDHC-RVsgEt5-2tghXgTB0-k/edit?usp=sharing
6. Setup the behavioral simulation.
    - In Vivado, from the left side pane, click `Run Simulation`, then `Run Behavioral Simulation`.
    - Minimize the two panes on the left with `Scope` and `Objects` tabs to make the wave view bigger.
    - Optionally, minimize the bottom pane (`Tcl Console/Messages/Log`).
    - Optionally, drag the handle next to `Value` to make the columns larger.
    - Set the radix for all of the signals that are by default included in the waveform to `Binary`.
        - To select: within the waveform graph, under the `Name` column, click first signal, then hold `Shift` and click last signal.
        - Right-click one input and click `Radix`.
        - Select `Binary`.

7. Based on the first row of your datasheet, enter the values F_1, E_1, F_2, and E_2 **in binary, as two's complement**. Run the simulation for 10 ns.
   - Right-click an input (ex. `f1`) and select `Force constant`.
   - Change the value radix to `Binary`.
   - Enter the binary value of the input (ex. $F_1 =$ 0111) as the `Force value`.
   - Click `OK`.
   - To run the simulation for a specified time, click the play button with `(T)` underneath, or press `Shift+F2`.
   - While holding `Ctrl`, you can use the mouse scroll wheel to zoom in and out in the waveform view.
   - While holding `Shift`, you can use the mouse scroll wheel to navigate horizontally along the waveform view.
   - If you accidentally close the waveform view, go to `Window` in the toolbar, and click `Waveform`.
8. In your datasheet, record the outputs $P = F_1 \times F_2'$ (8 bits) and $S = E_1 - E_2 + a$ (4 bits)
9. From P and S, find the final answers F and E.
   - F is a normalized fraction that can only be one of the eight values shown in the first column of the tables in the Background section.
   - E is a four bit number between -8 and 7.
     - Indicate if there is an exponent overflow.
10. Use F and E to calculate the decimal value of $N = F \times 2^E$.
11. Repeat steps 7 through 10 for the remaining rows in your datasheet.
12. Once you have completed simulation, calculate the decimal value of the actual N, that is:

$$N_{\text{actual}} = \frac{F_1 \times 2^{E_1}}{F_2 \times 2^{E_2}}$$

13. Calculate the percentage error between N and N_{actual} with:

$$\%\ \text{Error} = \frac{N_{\text{actual}} - N}{N_{\text{actual}}} \times 100$$

   - Note that within Excel or Google Sheets, formatting as percent will automatically multiply by 100.

Discussion

14. Is N close to N_{actual}?
15. What could we do to improve the accuracy?
16. How can we change the hardware description to support higher precision for the reciprocal?

Deliverables

- Include the following items in your **informal report** for Lab 04:
  - Your datasheet
  - A screenshot of your wavefrom from the simulation
  - Answers to the discussion questions (14, 15, and 16) from this activity

Outcomes

- Practice working with VHDL.
- Practice using Vivado for hardware simulation.
- Understand a simple floating-point representation.
- Understand how floating-point division is implemented in hardware.