ELEC3342: Digital System Design
First Semester, 2022–23

Lab 4

`Objective: To experiment with sequential circuits.`

1 Getting the files

Download the files for this lab from

`http://www.eee.hku.hk/~elec3342/fa22/handout/elec3342lab4.tar.gz`

2 Inferring a Flip-flop

In this part of the lab, you will experiment with various ways that you can infer flip-flops and registers using VHDL. These techniques will lay the foundation on which you can build larger sequential circuits and systems.

Start with inferring a simple D-flip-flop (DFF) using VHDL. The fundamental idea is to use a *synchronous process* in VHDL to infer any sequential circuit.

- In Vivado, start a new project named `dff1`.
- Add the file `rtl/dff1.vhd` into the project as *design source*.
- Open the file `dff1.vhd` and complete the missing lines to implement DFF *with* enable signal.

  Hint: You will need an `if-then` statement to implement enable.
- Open the elaborated design.

2.1 Checkoff 1

Based on the above elaborated design, answer the following:
- How many flip flops do you see in the inferred design? What type of FF is that?
- If you change the polarity of the enable signal (disabling the FF when enabled), how does it change the elaborated design?

Submit your (i) answer above, (ii) modified `dff1.vhd` and (iii) a screenshot showing the elaborated design.

3 Inferring Multiple FFs

Each process is a concurrent statement in VHDL. When you have multiple processes in a VHDL, they run concurrently. Now start by making a circuit with 2 DFFs using 2 VHDL processes.

- In Vivado, start a new project named `dff2`.

- Add the file `rtl/dff2.vhd` into the project as *design source*.
- In file `dff2.vhd`, you will see we already implement a DFF with enable. Now implement another DFF in the file by creating a *separate process* that is similar to the one already included. Your DFF should be a simple DFF *without* enabled signal. Use signal `b` as input, and signal `n` as output.

3.1 Checkoff 2

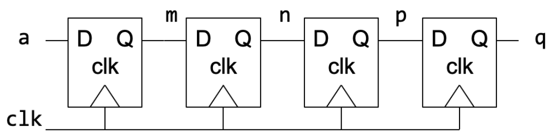
Based on the above elaborated design, answer the following:

- How many flip flops do you see in the inferred design? What types of FF are they?
- Is there any difference between between the two inferred flip flops?

Submit (i) your answer above, (ii) your modified file `dff2.vhd`, (iii) a screenshot showing the elaborated design.

4 Delay Line

A delay line with n DFFs can delay the input signal by n clock cycles. We show a 2 DFFs delay line in lecture. Here, you will implement a 4 DFFs delay line in this section. The 4 FFs should form a chain like this:



- In Vivado, start a new project named `delay_line`.
- Add the file `rtl/delay_line.vhd` into the project as *design source*.
- Add the file `tb/delay_line_tb.vhd` into the project as *simulation source*.
- In the file `delay_line.vhd`, you will see we have already implemented two DFFs. Now implement the other two DFFs in another two processes.

4.1 Check Yourself

Based on the above elaborated design:

- How many flip flops do you see in the inferred design? What type of FF is that?
- Do they look like a delay line?

Based on the simulation waveform:

- Check the output signal, has the input signal been delayed properly?

4.2 Checkoff 3

Submit (i) your modified file `delay_line.vhd`, (ii) a screenshot showing the elaborated design, (iii) a screenshot of your simulation output.

5 Inferring delay line in one process

When desingin larger circuits, often time you will need to infer multiple registers/FFs within one process. In this part of the lab, you will practice implementing the same delay line above in a single process.

To do that, remember statements within a process are evaluated sequentially. Also, an **assignment statement** (`<=`) is a sequential statement within the context of a process. Despite the fact that an assignment statement is evaluated sequentially within a process, its function remains the same: that it *assigns* a **signal** value to another **signal**.

For a delay line, you can implemen you should note that each an every intermediate signals between the FFs is an individual **signal**. You will notice we have already defined the signals and named them `m`, `n`, `p` for you.

- In Vivado, start a new project named `delay_line_1proc`.
- Add the file `rtl/delay_line_1proc.vhd` into the project as *design source*.
- Add the file `tb/delay_line_tb.vhd` into the project as *simulation source*.
- In file `delay_line_1proc.vhd`, implement the delay line in one process. **HINT:** your code should consist of just multiple assignment statements. Think about what values the signals in the circuit (`m`, `n`, `q`) should be assigned to.

5.1 Check Yourself

Based on the above elaborated design:
- Compare the delay line created using a single process and the one created with 4 separate processes, are they the same? If not, what are the differences?
- Simulate the deign. Is your delay line with a single process behaving the same as the one with 4 processes?

5.2 Checkoff 4

Submit the file `delay_line_1proc.vhd` ready and take a screenshot showing the elaborated design.

5.3 A BUGGY Design Now create a similar project named `delay_line_wrong`. We have prepared a similar delay line design with a single process, but it is NOT functioning correctly.

- Add `rtl/delay_line_wrong.vhd` as a design source.

5.4 Checkoff 5

Examine the content of the file `delay_line_wrong.vhd` and compare to the file `delay_line_1proc.vhd`, what are the main differences?
Now open the **elaborated** design. Is the elaborated circuit schematic the same as the delay line created in `delay_line_1proc.vhd`. Explain the differences.
Submit your answers above for this checkoff.

6 Submission

Compress your files from previous checkoffs to a zip file and upload it to Moodle:

```
<your-zip-file>
├─ dff1.vhd
├─ dff1.jpg
├─ dff2.vhd
├─ dff2.jpg
├─ delay_line.vhd
├─ delay_line.jpg
├─ delay_line_1proc.vhd
├─ delay_line_1proc.jpg
└─ answers to the checkoff questions
```

Screenshots can be `.jpg` or any other image format.