Exercise 02: Intro CocoTB Exercises

These exercises will introduce you to:

- The (Make) flow for running CocoTB simulations
- CocoTB constructs

0.0.1 Exercise A: Simple *cocotb* Test for Adder Design

Objective: *Introduction to the cocotb tests.*

Task: *Run cocotb tests for simple RTL and view waveforms.*

Look at the test example for the adder in

`<ROOT>/exercises/E02_intro_cocotb_exercises/A_example_adder`

The example can be run by going to the `test`-folder, make sure that the virutal enviroment is activated. Run the tests using `make` and the flag `WAVES=1` to generate a waveform-file. The waveforms can be seen by opening the file in e.g. gtkwave

```
# Run the tests
(.venv) [<username>@<servername> test]$ make WAVES=1

# visualize the waveforms
(.venv) [<username>@<servername> test]$ gtkwave sim\_build/adder.fst
```

0.0.2 Exercise B: Simple *cocotb* Test for Multiplexer Design

Objective: *Introduction to the cocotb tests*

Task: *Development of a cocotb test for simple RTL.*

Create two *cocotb* tests similar to the adder example for the multiplexer design (basic and random tests), using the test-setup found `<ROOT>/exercises/E02_intro_cocotb_exercises/B_mux`.

NOTE: Create a python module called test_mux.py in

`<ROOT>/exercises/E02_intro_cocotb_exercises/B_mux/test`

HINT: Look at the RTL for MUX. It has different ports than the adder!

Create the two *cocotb* tests in the test_mux.py file:

- `async def mux_basic_test(dut)`: Drive a single transaction through the DUT
- `async def mux_randomized_test(dut)`: Drive 10 random transactions thorugh the DUT

Afterwards, modify the Makefile (if needed) and run the tests.

Cocotb Triggers Try using the different methods for increasing the time in simulation that can be imported from `cocotb.triggers`.

Example:

```
# generating clock signal, driving the clk of the DUT
cocotb.start_soon(Clock(signal=dut.clk,period=4,units='ns').start())

# allowing time to pass
await Timer(2, 'ns')
await ClockCycles(dut.clk, 2)
await RisingEdge(dut.clk)
```

Add the differnt triggers to the end of the mux_basic_test.

0.0.3 Exercise C: Parallel *cocotb* Test

Objective: *Introduction to the cocotb tests.*

Task: *Development of a cocotb test for simple RTL.*

This exercise will introduce usage of `Combine` and `First`. Do the following exercise using the files in

`<ROOT>/exercises/E02_intro_cocotb_exercises/C_parallel`

Create cocotb test for parallel design

Create a coroutine for each signal (A, B, C) that drives them at different intervals.

See example below.

```
await RisingEdge(dut.clk)

for _ in range(20):
    A = random.randint(0, 7)
    dut.A.value = A
    await ClockCycles(dut.clk, 3)

dut.A.value = LogicArray('x'*4)
```

Create a test that starts all the coroutines using `cocotb.start_soon()`.

Use `Combine` to await all coroutines.

Similarly, create a test that starts all the coroutines and uses `First`.

- What are the differences between the two triggers?
- In what use cases could they be useful?