Laboratory 1: Introduction to VerilogHDL and Digital Simulation

Google Doc Link:
https://docs.google.com/document/d/1RwdhU_V4fgYIIHKyREarglx_fv__xPal/edit?usp=sharing&ouid=110924416719706935733 &rtpof=true&sd=true

Objectives

1. Understand Verilog HDL.
2. Understand the combinational logic and sequential logic on Verilog.
3. Understand how digital simulation and testing is important in hardware designing.

Part 1: Revisiting Digital Logic

Please answer the questions in MyCourseVille in the assignment section.

Part 2: Introduction to Verilog

Verilog is a hardware description language (HDL) used to model electronic systems. In this part, you will learn how to design the circuits or modules using Verilog.

Setup

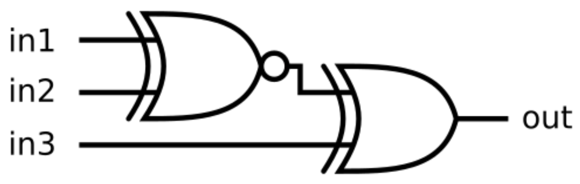
Clone the assignment from this repository:

https://github.com/2110363-HW-SYN-LAB/lab1

a. Your task is to edit Verilog files (.v files) in each problem inside "part2" directory. **You may specify *wire* or *reg* to inputs and outputs as you want.**
b. After editing codes, verify your codes by running testbench inside the "tests" directory of each problem. You can run by activating an environment that installs Cocotb (in case you use venv or Conda) and run the python file (`python <python_file>`).

Problem 1

Implement the module *xnor_xor*, which represents the following circuit.



Inputs

- *in1, in2, in3*: The inputs of the combinatorial circuit

Outputs

- *out*: The output of this combinatorial circuit

Problem 2

Implement the *full_adder_4_bit* module, which adds 2 4-bit inputs and carry-in and produces a sum and carry-out.

Inputs

- *a*: First 4-bit number to add
- *b*: Second 4-bit number to add

- *cin*: Carry-in

Outputs

- *sum*: 4-bit sum of 3 inputs
- *cout*: Carry-out

Problem 3

Implement the D flip-flop with synchronous reset, which reset when the clock signal is at the rising edges. Name the module as *dff_sync_reset*.

Inputs

- clk: Clock signal that controls when the flip-flop samples the input data
- rst: Synchronous reset signal (active high) used to reset the stored value to 0 - D: 1-bit data input to be stored in the flip-flop

Outputs

- Q: 1-bit output representing the stored value of the flip-flop

Problem 4

A JK flip-flop is a sequential logic circuit that stores a single bit of data, acting as a fundamental building block in digital systems, capable of Set, Reset, and Toggle operations, uniquely overcoming the invalid state of older SR flip-flops by changing (toggling) its output when both inputs (J & K) are high. The state of JK flip-flop is summarized in the table below.

J	**K**	**Current State (Q)**	**Action**
0	0	Qprev	Hold
0	1	0	Reset
1	0	1	Set
1	1	~Qprev	Toggle

Implement the *jkff* module, which represents the JK flip-flop as mentioned above.

Inputs

- clk: Clock signal that controls the flip-flop
- J: Set/toggle control input
- K: Reset/toggle control input

Outputs

- Q: 1-bit output representing the stored value of the flip-flop

**** Checkpoint: Call TA to check your work ****

Part 3: Introduction to Digital Simulation

Verifying and testing are critical in Verilog (and hardware design in general) because hardware designs, once implemented in silicon, cannot be easily modified. In this part, you will learn how to simulate and perform testing on your designs.

Setup

In the same repository as part 2, your task is to edit testbenches in the "tests" subdirectory in each problem in the "part3" directory.

Problem 1: Simulation on Combinational Logic Circuit

You are given a 2-bits full adder module, which adds two 2-bit inputs and carry-in. The module will give outputs as sum and carry-out.

Inputs

- A: 2-bit input of the first number.
- B: 2-bit input of the second number.
- Cin: 1-bit input of the carry-in.

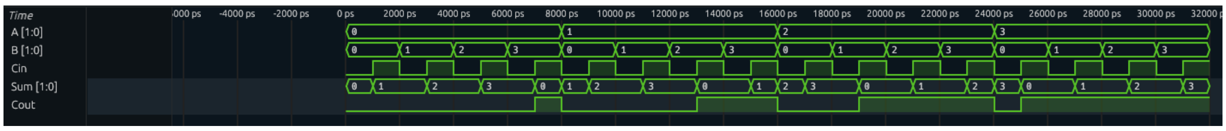
Outputs

- Sum: 2-bit output of the result of addition.
- Cout: 1-bit output of the carry-out of addition..

a. Inside *part3/part3-1/tests/fulladder_2bit_test.py*, edit the file by creating a testbench for the 2-bit full adder. You must iterate all of the possible combinations of inputs.
b. Run the testbench.

c. Open the result waveform.

Paste your result waveform picture here



Problem 2: Simulation on Sequential Logic circuit.

You are given a 2-bit counter module, which adds the counter by 1 every rising edge of the clock if the enable signal is high, and can be reset to 0 synchronously.

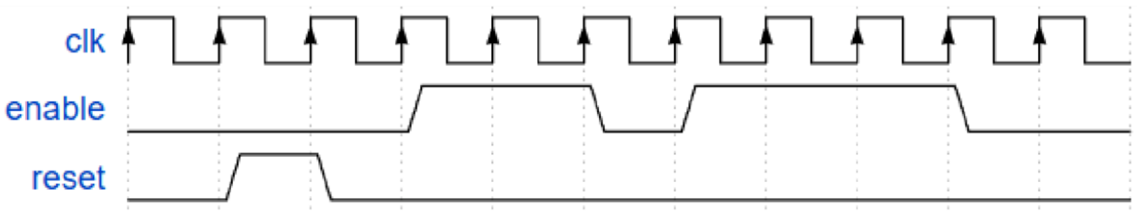
Inputs

- enable: 1 bit input that enables the system by adding the counter by 1 periodically.
- clk: 1 bit input of the clock of this system.
- reset: 1 bit synchronous reset that reset the counter value to 0 (active high).

Outputs

- counter_value: 2-bit output of the counter value.

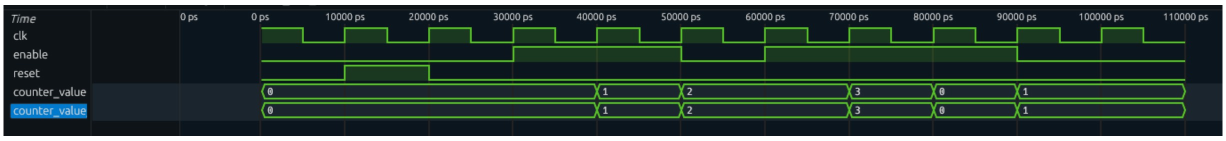
a. Inside *part3/part3-2/tests/counter_test.py*, edit the file by creating a testbench for the 2-bit counter module. Your test must follow the waveform below.



b. Run the testbench.

c. Open the result waveform.

Paste your result waveform picture here



**** Checkpoint: Call TA to check your work, export this document as PDF, zip all of your codes, and submit in MyCourseVille ****