EECS 151/251A Homework 2

Due Monday, Feb 6th, 2023

For this HW Assignment

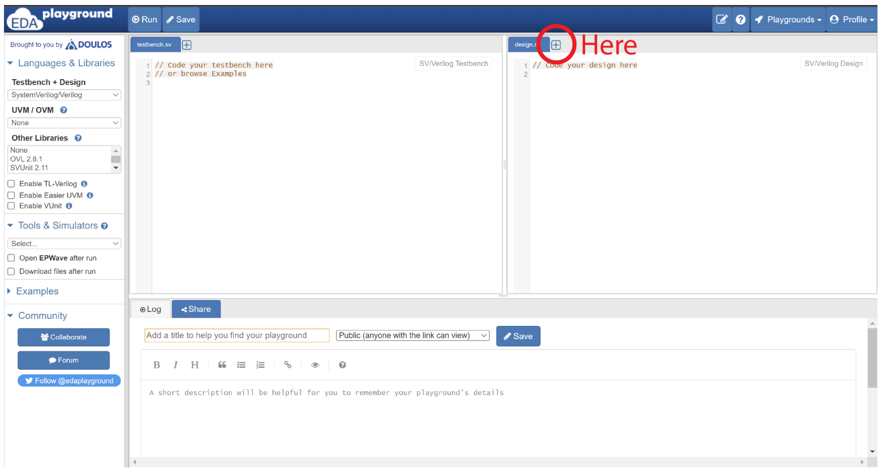
You will be asked to write several Verilog modules as part of this HW assignment. You will need to test your modules by running them through a simulator. As shown in discussion 2, a highly suggested simulator is `https://www.edaplayground.com` which is a free, online, Verilog simulator.

For all problems except 2(b), turn in:

1. Circuit diagram (neatly drawn by hand or with a tool). When drawing there are some rules we'd like you to follow:
    - Draw solder dots at wire junctions
    - Label bus widths for multi-bit wires
2. Verilog code
3. Testbench
4. Test result

***Warning*: We enforce no register inference policy in this class. You must use the register ibrary in `EECS151.v` whenever using registers in your Verilog. `EECS151.v` is located at `https://inst.eecs.berkeley.edu/~eecs151/sp23/files/lib/EECS151.v`. We will only accept solutions using this library!**

To import the library on *EDA playground*, click the plus button shown below and upload the file. Then, add `` `include "EECS151.v" `` to your Verilog code (`design.sv`). You can import other modules (such as modules you designed before) in the same way.



```
// Code your testbench here
// or browse Examples
```

```
// Code your design here
```

Version: 1 - 2023-02-01 04:58:09-08:00

Problem 1: Binary Decoder and Encoder

In this problem except (a)3, you are not allowed to use any of the following operators: {`!`, `~`, `&`, `|`, `~&`, `~|`, `*`, `&&`, `||`}.

(a) Write five different Verilog implementations of a 2-bit binary decoder, based on the formats listed below. For the first one (structural Verilog), also draw a gate-level circuit diagram. Write a single testbench that exhaustively tests all of your implementations (put all of them in `design.sv` on *EDA playground*).

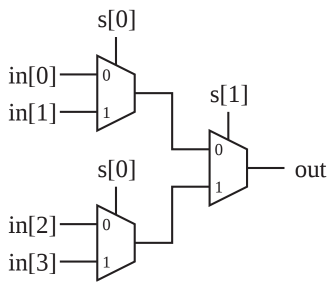
1. Structural Verilog
2. Behavioral Verilog using continuous assignment (i.e. no `always` blocks)
3. Behavioral Verilog using continuous assignment and bit-wise operators
4. Behavioral Verilog using `if`
5. Behavioral Verilog using `case`

(b) Write the behavioral Verilog using continuous assignment and the ternary operator (?:), for a 2-bit binary *encoder*. (A encoder has the inverse function of a decoder.) The input is guaranteed to be one-hot. Write a testbench using one of the decoders you designed (import the decoder in the testbench).

Problem 2: Decoder-Based Multiplexer

(a) Design a 4-to-1 multiplexer using one of the decoders you designed above. The select signals must be input to the decoder and must not be used anywhere else. Provide an exhaustive test.

(b) What could be a potential benefit of using this decoder-based multiplexer against the following design:



Problem 3: Equality Comparator Generator

An equality comparator is a combinational logic circuit that takes as input two N-bit signals and outputs 1 iff the two signals match in every bit position. Design a generator for equality comparators of size N in the structural Verilog. Write a testbench that tests $N = 1$ and $N = 4$ exhaustively.

Problem 4: Counter Generator

Write a Verilog implementation for an N-bit counter generator for counters with the following specification. Write a testbench for $N = 4$.

Specification:

- Counters have as input `clk` (the clock signal), `rst` (reset), and `en` (enable).
- Counters have an N-bit output named `cnt`.
- Counters hold the value of `cnt` constant when both `rst` and `en` are 0.
- Counters set `cnt` to 0 on a positive edge of `clk` if `rst` is 1.
- Counters increment `cnt` by 1 at a positive edge of `clk` if `rst` is 0 and `en` is 1.
- When `cnt` is the maximum possible value ($2^N - 1$ in N-bit counters), `cnt` will become 0 next time it is incremented.

Warning: Use the register library in `EECS151.v`.

Problem 5: Serial To Parallel Converter

Imagine a situation that you need to design an interface between two circuits A and B. A generates one-bit data at a time, while B receives 10-bit data in parallel. These circuits also partially implement the hand-shaking protocol. A raises `SerRdy` when the data starts coming out. On the other hand, B reads the data when `ParRdy` is on. Assume all circuits (including the one you are going to design) share the clock signal `clk`.

(a) Using the equality comparator generator and the counter generator you designed, design a saturating counter that saturates at 9 (it works the same as the original counter up to 9, but then it will get stuck at 9 until the reset signal is activated). Equip it with an extra output port that tells whether it is saturated (1) or not (0).

(b) Design a 10-bit shift-register with enable (it works as a shift-register if enable is 1, otherwise it just holds the current value).

(c) Design the interface circuit using the saturating counter and the shift-register. Its detailed specification is shown below. You may assume that `SerRdy` is 1 at the first positive edge.

Specification:

- Receives three binary inputs: `SerDat`, `SerRdy`, and `clk`.
- Generates two outputs: a 10-bit signal `ParDat` and a binary signal `ParRdy`.
- Stores a 10-bit sequence that `SerDat` takes at each positive edge of `clk` after `SerRdy` becomes 1.
- It is guaranteed that once `SerRdy` becomes 1, it will remain 1 until next positive edge of `clk`. Just after that, it will drop to 0 and remain 0 for at least 10 cycles.

- Turns on `ParRdy` when finished storing the 10-bit sequence, which is output as `ParDat`.
- Turns off `ParRdy` when next sequence comes in.