Assignment Homework 2

Due date: Check Canvas

Student Version

Please team up with a fellow student and start working on the following tasks (100 pts total). In addition, you *must* consider the following:

- Vivado creates a default template including a comment block with:
  - "Company" – insert team name/number
  - "Engineer" – put team member names here
  - "Design Name" – based on task description (*do not leave empty*)
  - "Project Name" – Assignment 2
  - "Description" – based on task description (*do not leave empty*)
- To prepare your assignment for submission through Canvas, you must use:
  `File > Project > Archive`
- You work as a team. Please alternate between one student creating the testbench, the other student creating the implementation.

We will start this homework assignment together through Zoom as a lab. Please try to follow along and ask questions if you get stuck somewhere. Your homework assignment includes all tasks mentioned here (even those, that you manage to complete during our joint lab time).

1 Task

Please implement the following (in Verilog):

1. A basic edge detector (`detect_edge`)
2. A n x m bit shift register (`nxmBitShiftReg`)

Including test benches to confirm that your design is working.

`detect_edge` After learning about `posedge` you might be inclined to use this macro on regular signals. However, common FPGA/HDL wisdom is *not* to do this (as in: never) and instead, create an edge detection module for rising and falling edges. After brainstorming with a fellow student, you come up with the module description in Figure 1 and the preliminary waveform shown in Figure 2. As a hardware engineer, you will think about the basic gates needed (FFs? Boolean gates?) and design a gate-level diagram prior to writing the HDL code. How would your design change if adding a third output that creates a pulse on any edge? Please make sure to follow the given timing diagram accurately, i.e., the outputs must *not* be delayed by a clock cycle.

`nxmBitShiftReg` Shift registers are a hugely popular design element due to requiring few resources only while ensuring good timing behavior (=high clock frequency). However, as you are about to graduate from Oregon State University, you do not just want to implement *any* shift register but the best possible and most versatile one. This results in the following description:

- The register shall support a generic number of words, called `n`, with a bit width of `m` bits per word (this is beneficial, for not only rotating binary values, but to support a shift register of a different radix than binary, e.g., when using a shift register holding hexadecimal values).
- `OPMODE` defines the mode of operation and can be adjusted during run-time.
- `SR` initializes the complete register with `SRINIT`.
- `CE` (de)activates the normal working mode but does not override the reset mechanism
- `DOUT` has the size of one word and contains the word *shifted out* of the register in shift mod resp. *the last word rotated* in rotation mode.
- `DOUT_F` always provides the full register state.

Table 1: Modes of operation for the `n x m` bit shiftregister

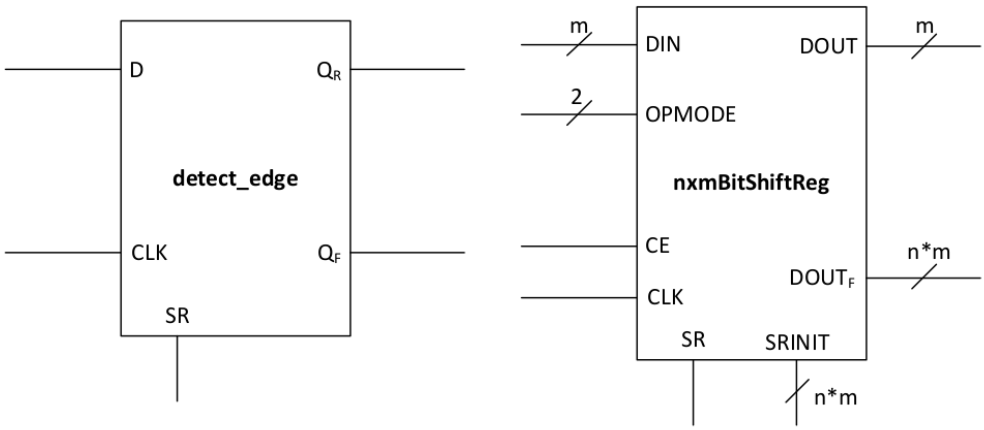
`OPMODE`	MODE	Description
00	SHR	Shift right one word
01	SHL	Shift left one word
10	ROR	Rotate right one word
11	ROL	Rotate left one word

2 General Design

For now, we continue to mostly ignore the recommended Verilog coding style to keep things as simple as possible. Please keep the naming of the signals consistent to the provided figures. Additional hints:

- Your can reuse your modules from the previous assignment but this is not strictly required.

- There are different ways to build the shift register. Generate statements and parameters are likely to be helpful.
- If you look at the two bit input `OPMODE`, you notice that each bit has its own function (rotate vs. shift, left vs. right). You can use an internal signal, assign the input directly to this internal signal and declare an alias for each bit of the internal signal. Afterwards, you should use the alias to make sure your code can be understood more easily (your fellow student will appreciate this when reading the code).
- To implement rotation or shifting, look at the concatenation operator. Indexing and concatenation are convenient ways to implement left/right rotation.



(a) D-FlipFlop **(b)** Latch

Figure 1: Module/interface description of assignment 2.

3 Output

3.1 Submission

Submit your report/program code on Canvas (all files compressed in one `.zip` in addition to the archived project). The archive you submit must contain all .v files. Please make sure that your code passes the syntax check and comment anything which is not obvious (use the Verilog comments // comment).

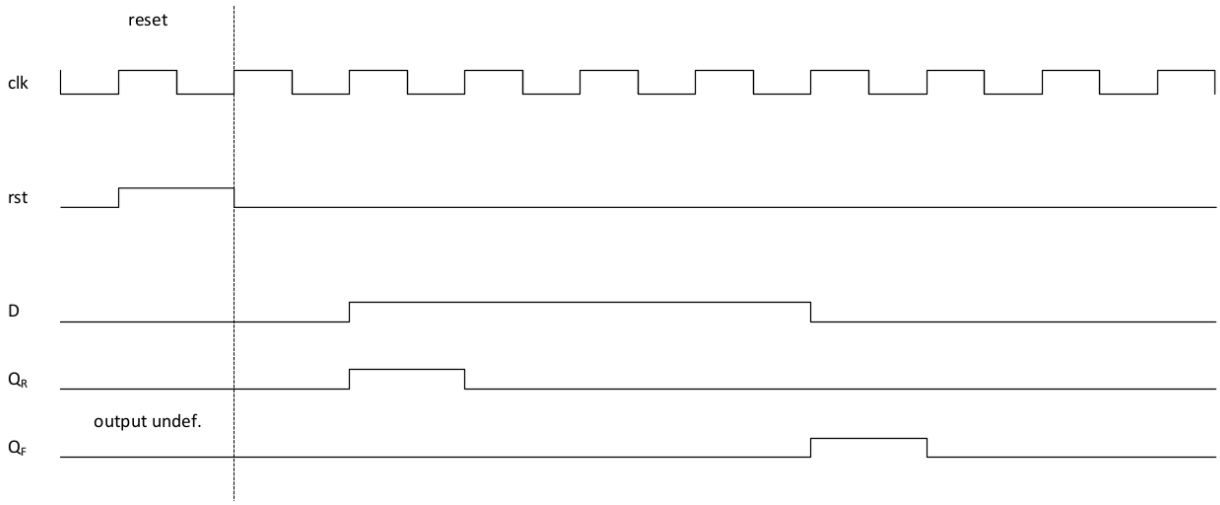


Figure 2