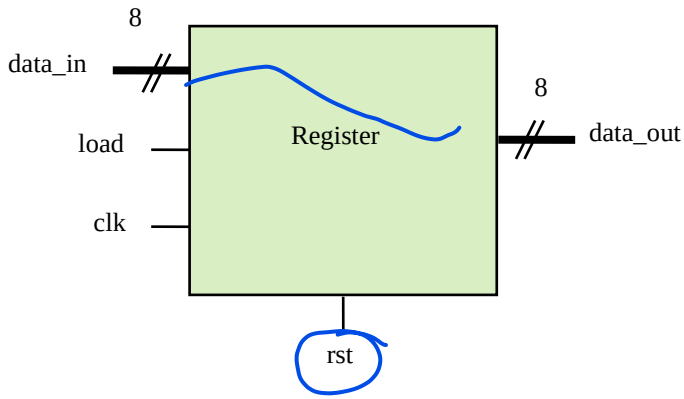
Module 7 Register

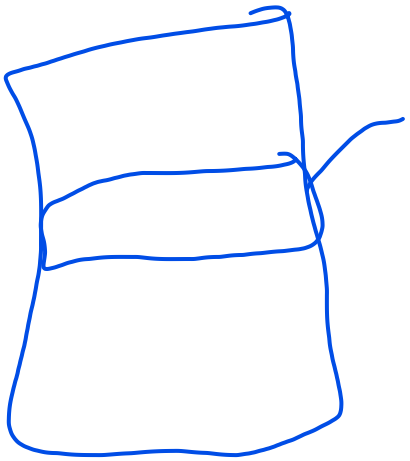
Module 7 Modeling a Generic Register

Objective: To use nonblocking assignments to describe a register.

The RISC CPU contains an accumulator register and an instruction register. One generic register definition can serve both purposes.

In this lab, you create a simple register design as per the specification and verify it using the provided testbench. Please make sure to use the same variable name as the ones shown in block diagrams.





Specifications

- `data_in` and `data_out` are both 8-bit signals.
- `rst` is synchronous and active high.
- The register is clocked on the rising edge of `clk`.
- If `load` is high, the input data is passed to the output `data_out`.
- Otherwise, the current value of `data_out` is retained in the register.

Designing a Register

1. Change to the *lab-2* directory and examine the following files.

register_test.v	Register test
filelist.txt	File listing all modules to simulate

2. Use your favorite editor to create the *register.v* file and to describe the register module named "*register* ".

   a. Parameterize the register data input and output width so that the instantiating module can specify the width of each instance.

   b. Assign a default value to the parameter as 8.

   c. Write the register model using the Verilog `always` procedural block.

Verifying the Register Design

1. Using the provided test module, check your register design using the following command with VCS

   ```
   vcs register.v register_test.v (Batch Mode)
   ./simv
   ```

   or

   ```
   vcs register.v register_test.v -R -gui ( GUI Mode)
   ```

2. You might find it easier to list all the files and simulation options in a text file and pass the file into the simulator using the `-f vcs` option.

   ```
   vcs-f filelist.txt -access rwc
   ```

   You should see the following results.

   ```
   At time 20 rst=0 load=1 data_in=01010101 data_out=01010101
   At time 30 rst=0 load=1 data_in=10101010 data_out=10101010
   At time 40 rst=0 load=1 data_in=11111111 data_out=11111111
   At time 50 rst=1 load=1 data_in=11111111 data_out=00000000
   TEST PASSED
   ```

3. Correct your register design as needed.