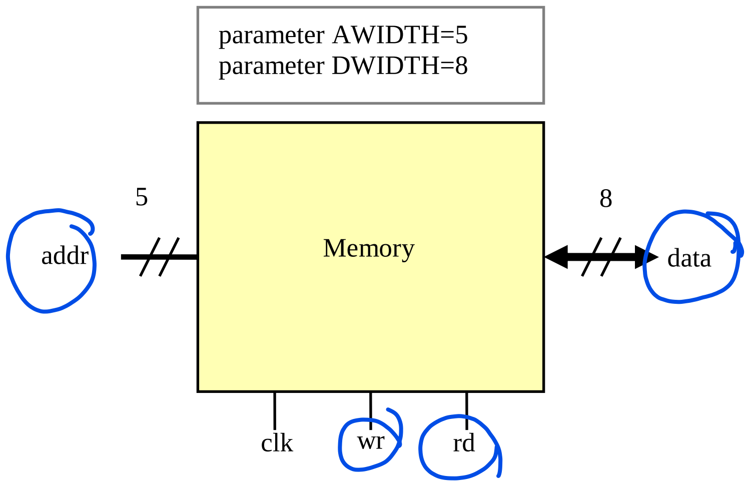
Lab 8-1 Modeling a Single-Bidirectional-Port Memory

Objective: To use continuous and procedural assignments to describe a memory.

The VeriRISC CPU uses the same memory for instructions and data. The memory has a single bidirectional data port and separate write and read control inputs. It cannot perform simultaneous write and read operations.

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.

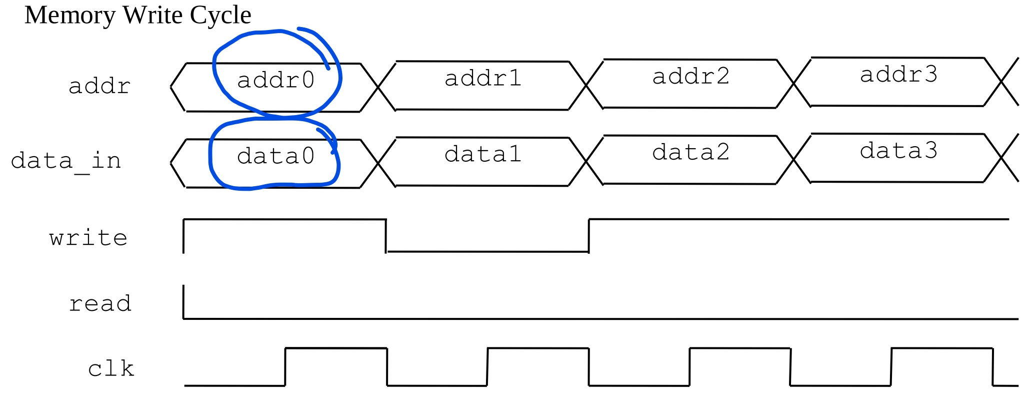


Specifications

- `addr` is parameterized to 5 and `data` is parameterized to 8.
- `wr` (write) and `rd` (read) are single-bit input signals.
- The memory is clocked on the rising edge of `clk`.
- Analyze the memory read and write operation timing diagram, as shown in the following graphic.

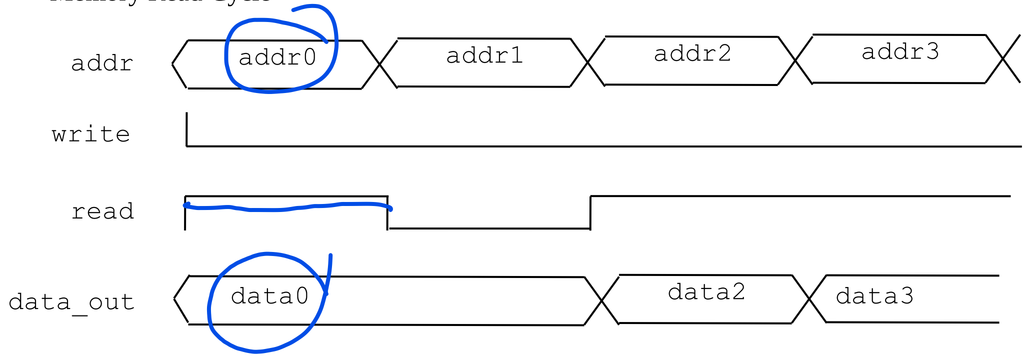
- **Memory write:** `data_in` is written to memory[addr] on the positive edge of `clk` when `wr` (write) =1.

Memory Write Cycle



- **Memory read**: `data_out` is assigned from memory[addr] when `rd` (read) =1.

Memory Read Cycle



Designing a Memory

1. Change to the *lab9-mem* directory and examine the following files.

memory_test.v	Memory test

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

3. Parameterize the address and data widths so that the instantiating module can specify the width and depth of each instance.

4. Assign default values to the parameters.

5. Write data on the active clock edge when the *wr* input is true and drive data when the *rd* input is true.

6. Perform the write operation in a procedural block and perform the read operation as a continuous assignment.

Verifying the Memory Design

1. Using the provided test module, check your memory design using the following command with Xcelium™.

   ```
   xrun memory.v memory_test.v (Batch Mode)
   ```

   or

   ```
   xrun memory.v memory_test.v -gui -access +rwc ( 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 xrun` option.

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

   You should see the following results.

   ```
   At time 370 addr=11111 data=00000000
   At time 380 addr=11110 data=00000001
   At time 390 addr=11101 data=00000010
   At time 400 addr=11100 data=00000011
   At time 410 addr=11011 data=00000100
   At time 420 addr=11010 data=00000101
   At time 430 addr=11001 data=00000110
   At time 440 addr=11000 data=00000111
   At time 450 addr=10111 data=00001000
   At time 460 addr=10110 data=00001001
   At time 470 addr=10101 data=00001010
   At time 480 addr=10100 data=00001011
   At time 490 addr=10011 data=00001100
   At time 500 addr=10010 data=00001101
   At time 510 addr=10001 data=00001110
   At time 520 addr=10000 data=00001111
   At time 530 addr=01111 data=00010000
   At time 540 addr=01110 data=00010001
   At time 550 addr=01101 data=00010010
   At time 560 addr=01100 data=00010011
   At time 570 addr=01011 data=00010100
   ```

```
At time 580 addr=01010 data=00010101
At time 590 addr=01001 data=00010110
At time 600 addr=01000 data=00010111
At time 610 addr=00111 data=00011000
At time 620 addr=00110 data=00011001
At time 630 addr=00101 data=00011010
At time 640 addr=00100 data=00011011
At time 650 addr=00011 data=00011100
At time 660 addr=00010 data=00011101
At time 670 addr=00001 data=00011110
TEST PASSED
```

3. Correct your memory design as needed.

End of Lab