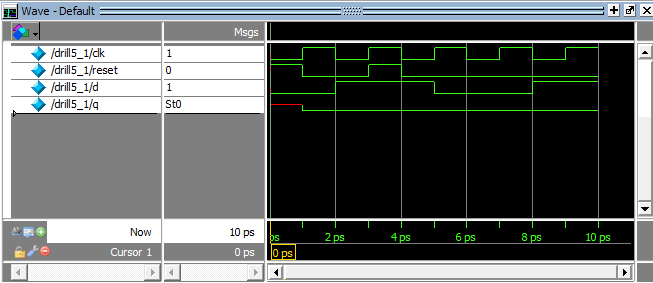
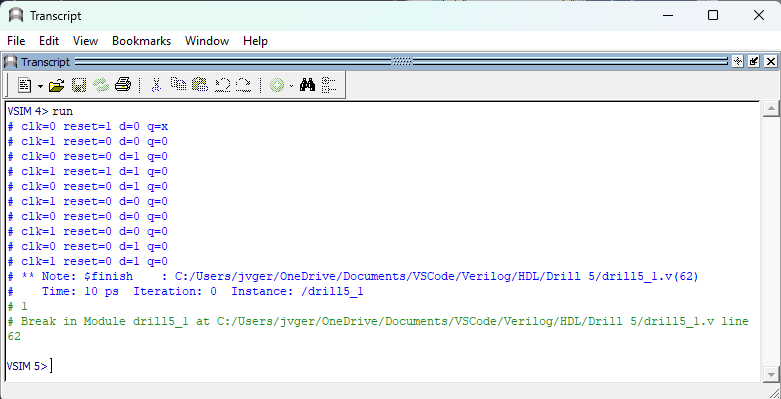
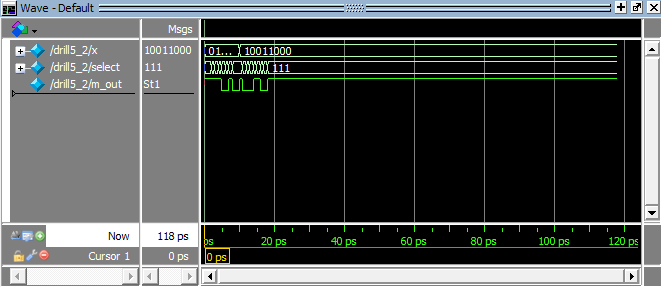
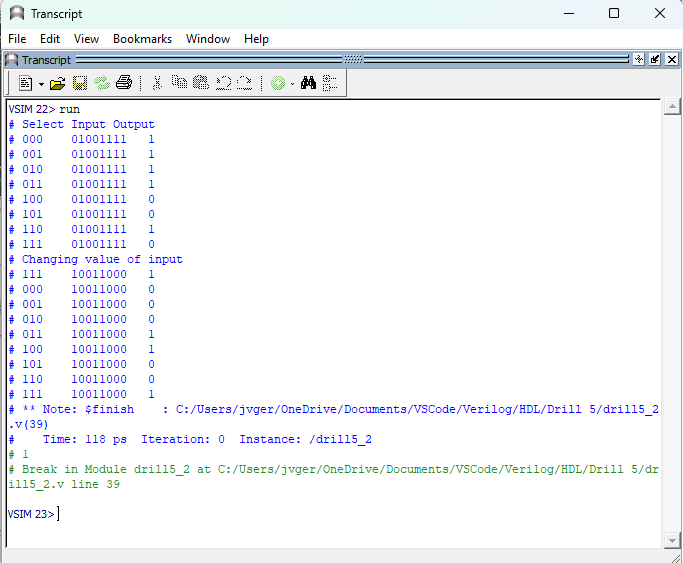
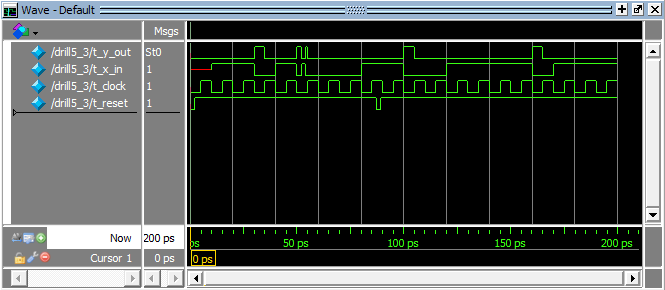
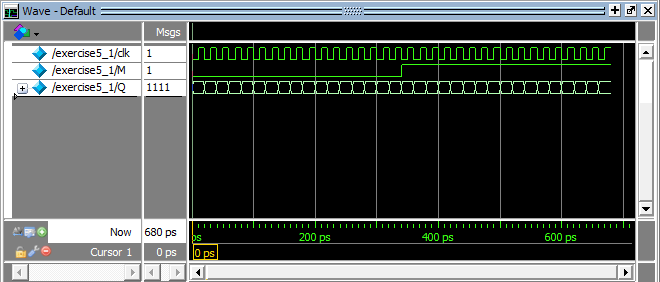
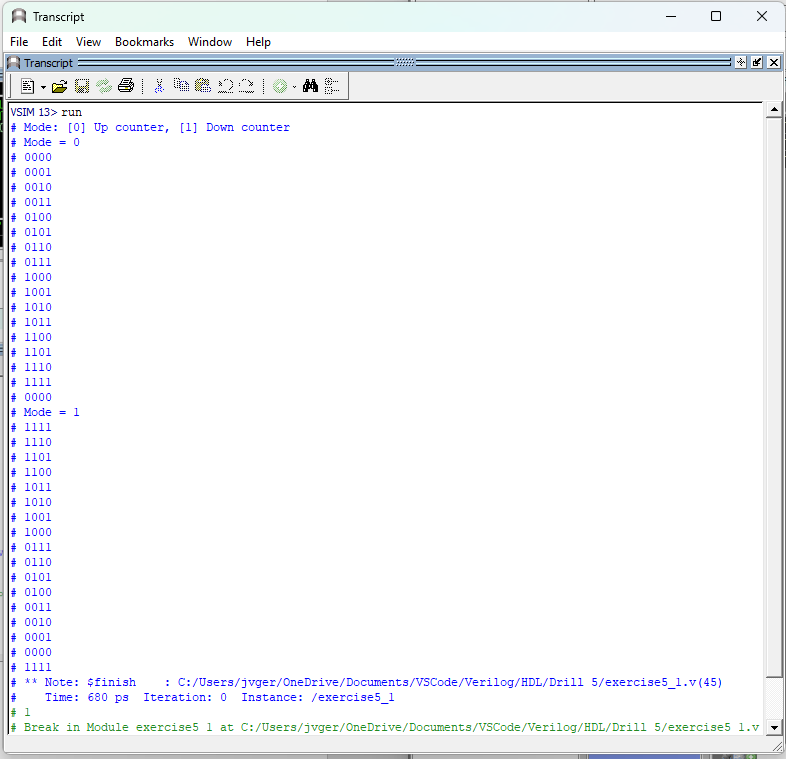
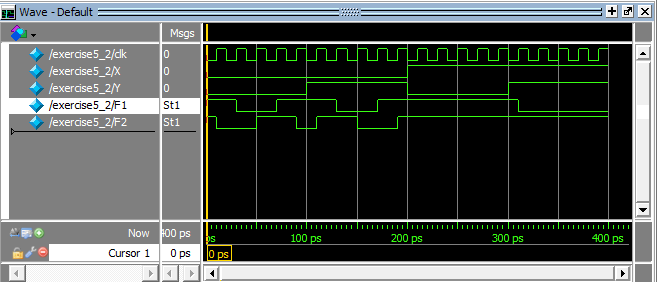
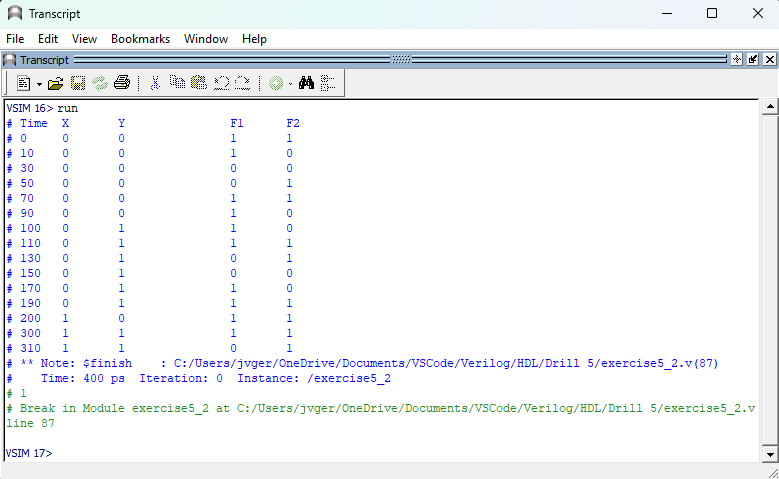
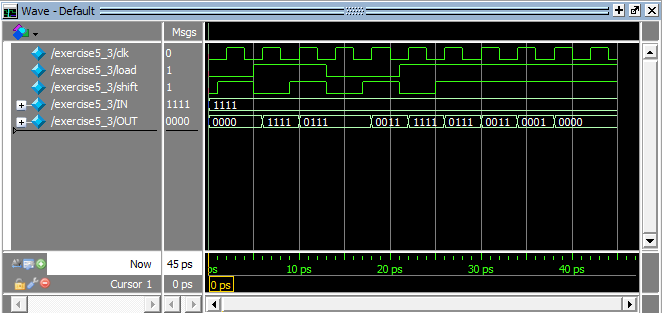
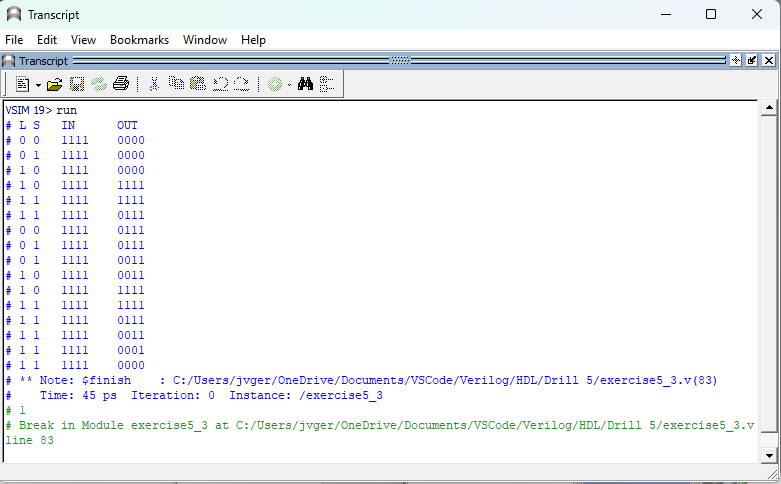
DRILL 5

* **Drill5\_1.v**
  + Testbench
  + Transcript
* **Drill5\_2.v**
  + Testbench
  + Transcript
* **Drill5\_3.v**
  + Testbench
  + Transcript



* **Exercise5\_1.v**
  + Testbench
  + Transcript
* **Exercise5\_2.v**
  + Testbench
  + Transcript

**Exercise5\_3.v**

* + Testbench
  + Transcript

**Review Questions**

1. **How is race condition experienced in Verilog HDL programming?**

*In Verilog, a race condition occurs when multiple statements are set to execute during the same simulation time-step, and changing the order in which they execute leads to different results. Essentially, this occurs when two or more threads are sharing the same resources.*

1. **Is there a difference between the instructions Always #1 a =! a; and forever #1 a = !a; ? If yes, what is/are their difference/s?**

*Yes, the statement "always #1 a = !a" executes the instruction every 1 unit of time and can be used in module, while "forever #1 a =!a" is typically used in test benches and also executes the same instruction at a 1 unit time interval but can’t be used in module.*

1. **Differentiate the selection constructs used in Verilog with those used in high-level languages (C++, C#, etc).**

*Verilog is a Hardware Description Language that includes the concept of time, which is crucial for simulating logical circuits. On the other hand, high-level programming languages like C++ and C# do not have a built-in concept of time, as they are primarily used for computer software programming.*

1. ***How are multiple always@ blocks executed within a given program?***

*In Verilog, multiple always@ blocks are executed within a program and are enclosed in parentheses with a begin and end statement.*

1. ***What are sensitivity lists? How do they affect the entire behavioral model description of a program?***

*The sensitivity list is a concise method of specifying the set of signals or events that can trigger a process to resume execution. It is defined after the "process" keyword and is used to list all the signals that will cause the code within the process to be evaluated whenever they change state.*

1. ***Why does the register operation of state transitions use a non-blocking operator (<=)?***

*In the state transition of register operations, a non-blocking operator is used to allow for a one-scheduled assignment without blocking the program's processing. Non-blocking assignments are executed sequentially as well.*