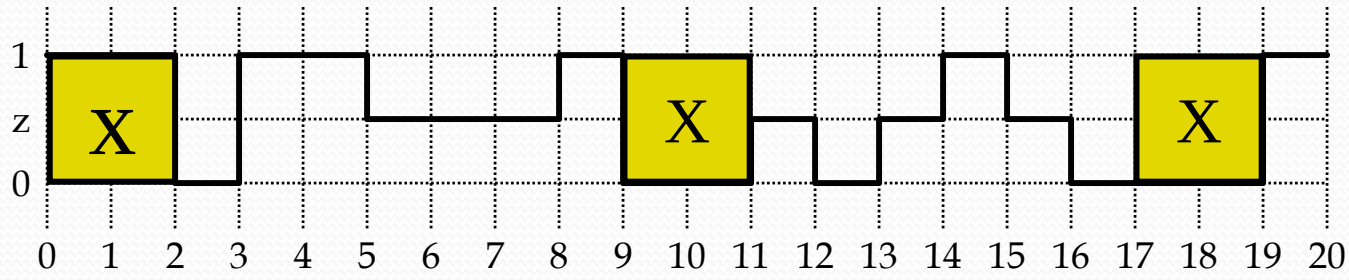
Lab 3.1

Lab 3.1

- In this lab, you will:
  - Learn how to use a waveform viewer
  - Learn to use the # statement
  - Learn how to use the @ statement
  - Learn the difference between various sampling techniques
  - Identify problems caused by event ordering and simulation startup

Lab 3.1

- In directory ~/lab3.1, you will find a file named "wave.v". It contains a declaration of a reg named "R" and an *initial* block with a few statements.
- Add the necessary statements to the *initial* block to complete a model generating the following waveform:



Lab 3.1

- If you run the simulation, nothing very interesting happens because this model does not produce any output.
- To verify that your waveform generator works as expected, add an *always* block that monitors any changes on the R register then displays the new value:

```
always @ (R)
begin
   $write("%0d: R = %b\n", $time, R);
end
```

- Simulate your model to verify that the correct waveform is generated.

Lab 3.1

- Waveforms are usually best displayed using a waveform viewer rather than using text statements.
- Waveform viewers are not part of the Verilog language, and as such are different for every tool you use.
- For the next lab step, please follow the instructions only for the waveform viewing tool you will be using . These instructions are presented on the following slides. Do not try all waveform viewers.
  - The remaining instructions for this lab are presented immediately after the instructions for the waveform viewing tools.

Lab 3.1

- ***IMPORTANT***: Most waveform viewers require that specific system tasks be compiled with the Verilog simulator. All tools may not be available in all simulators (error reported as "undefined task").
- The following instructions are by no means complete or thorough. Detailed tool training should be obtained through your vendor. Exploration of the waveform viewer features is strongly encouraged.

Lab 3.1: Icarus

- Compile your wave.v or whatever you called it as follows. In Windows you will need to open a Command Prompt window and go to the Icarus folder.

  iverilog -o wave wave.v
- Run it now since Icarus is a compiled verilog simulator.
  vvp wave
- Notice that it prints out nothing at all. Now add the following code inside your wave.v module
  initial $dumpvars(0,MODEL);
  Note: MODEL is the name of the module here
- Now compile and run again. Notice anything? A dump.vcd was produced in the folder containing waveform data.
- If running Windows Double click on the vcd file and GTKwave should open it. Now highlight MODEL and a list of signals should appear in the Signals window. INSERT each one and you should now see them. Highlighting one signal then using CTR-A will highlight them all, making it faster to INSERT the signals

Lab 3.1: ModelSim

- To use the ModelSim waveform viewer, you don't need to edit the the Verilog source file.
- During simulation you must use the graphical user interface.
- Create a new project and include your wave.v file
- Right click on the file and select Compile -> Compile Selected
- In the Transcript window type the following

  vsim work.MODEL
  You can also click on Simulate and then select work.MODEL from the work library
- Notice the Object window. Add each signal from it to the waveform window by right clicking on the signal then Add to Wave -> Selected Signal or by highlighting and dragging the signals into the waveform window

Lab 3.1: ModelSim

- Now all the signals should be in the Waveform window but still no signals displayed.
- Now save the Wavform information by selecting the Wave window (click on the top bar of that window)
- Now select File -> Save and a window should ask you to save the default name of wave.do. Just use that name for now. Now you will have this info for future simulations of this module if needed.
- In the Transcript Window type

  do wave.do

  run –all

  Answer NO to the popup window and it will then run, display the signals and bring up the .v file and show you where it stopped. You can close that window or just click on the waveform window tab to bring it to the front

Lab 3.1: ModelSim

- Now select the Zoom Full selection tool. It looks like a black magnifying glass
- You should now be able to see the signals

- Read the Tutorial which you can find in ModelSim by selecting Help-> PDF Documentation -> Tutorial

Lab 3.1

- Modify the monitor *always* block as follows, then simulate again. Why is the message sequence different?

```
always @ (posedge R) ...
```

- Repeat with:

```
always @ (negedge R) ...
```

- Repeat with:

```
always @ (posedge R or negedge R) ...
```

Lab 3.1

- In the same directory, you will find a file named "sample.v". It contains a waveform generator on a register named "R" and five *always* blocks that sample register "R" in various, similar-looking fashions.
- Using a waveform viewer, look at the waveforms of the sampler registers "S1", "S2", "S3", "S4", and "S5".
- Why are they different? Explain the differences.

Lab 3.1: Optional

- In the same directory, you will find a file named "puzzle.v".
- Run the simulation using your simulator of choice, then use another simulator (if available).
- Headers aside, are the simulation results different? Why?

Lab 3.2

Lab 3.2

- In this lab, you will:
  - Learn how to single-step through a model execution
  - Witness the sequential execution of parallel blocks
  - Examine the sequence of event-driven simulation

Lab 3.2

- In directory ~/lab3.2, you will find a file named "model.v". It contains the source for the example we have just walked through.
- Source-level debuggers are not part of the Verilog language, and as such are different for every tool you use.
- For the next lab step, you will need to discover how your particular simulator can do this.

Lab 3.2: Verilog XL

- Instruct the simulation to drop into the interactive command interpreter by using the '-s' option:

```
% verilog -s model.v
```

- Step through the code by using the ',' (comma) command at the "C1>" prompt.
- To exit, press Control-D or type "$finish;".

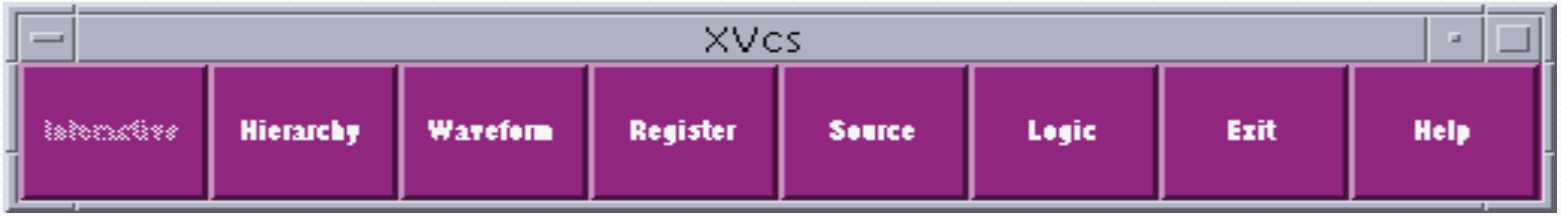
Lab 3.2: VCS

- To single step through code in VCS, invoke the VCS GUI environment. The following command line simulates and opens the VCS GUI debugging environment:

```
% vcs -line +cli+3 -RI model.v
```

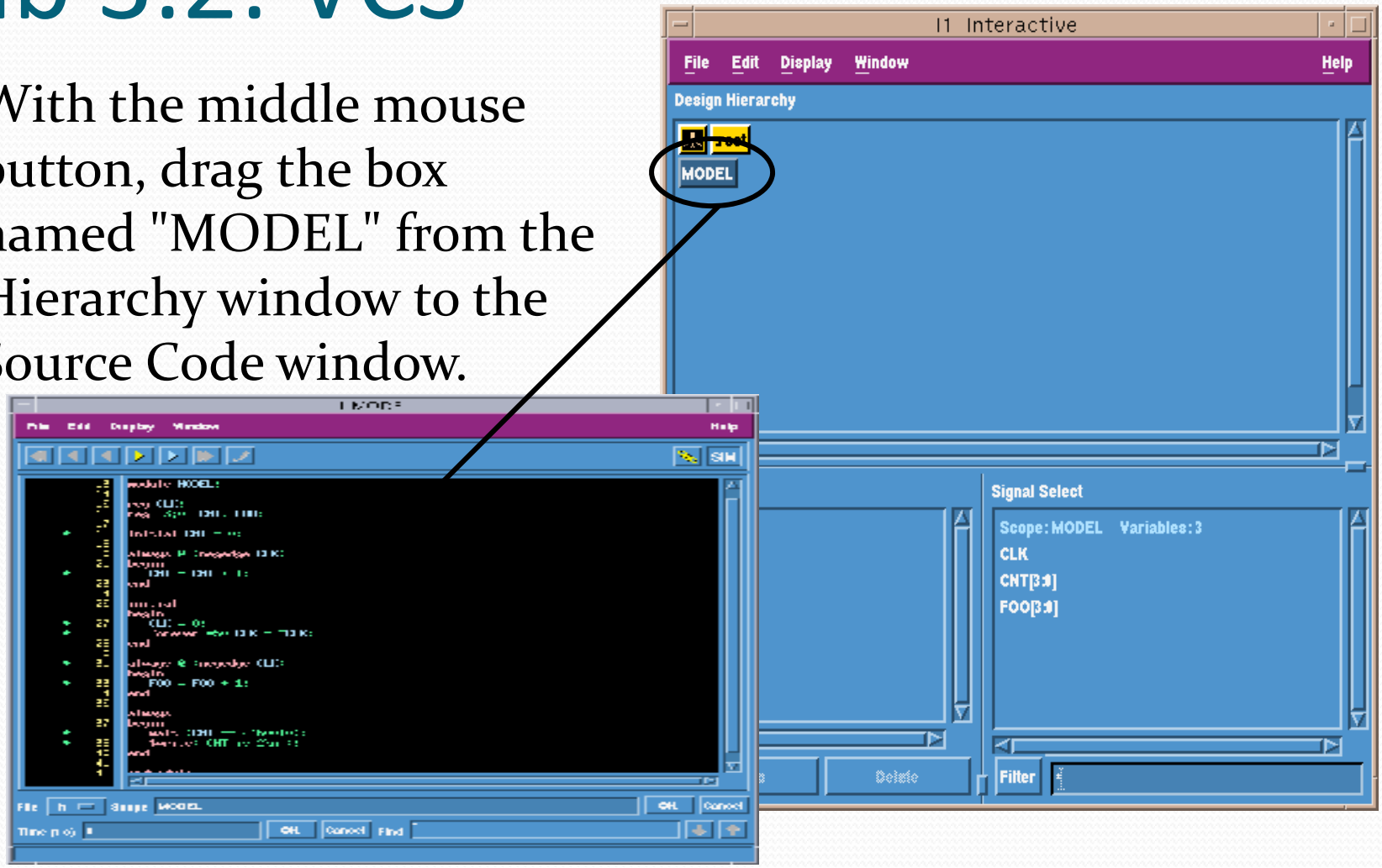
  - Note: If you see a linkage error, remove the +cli+3 option. Your version of VCS may not support this option.

- To open the Hierarchy browser and the Source Code viewer, use the button bar and click on the "Hierarchy" and "Source" buttons, respectively.



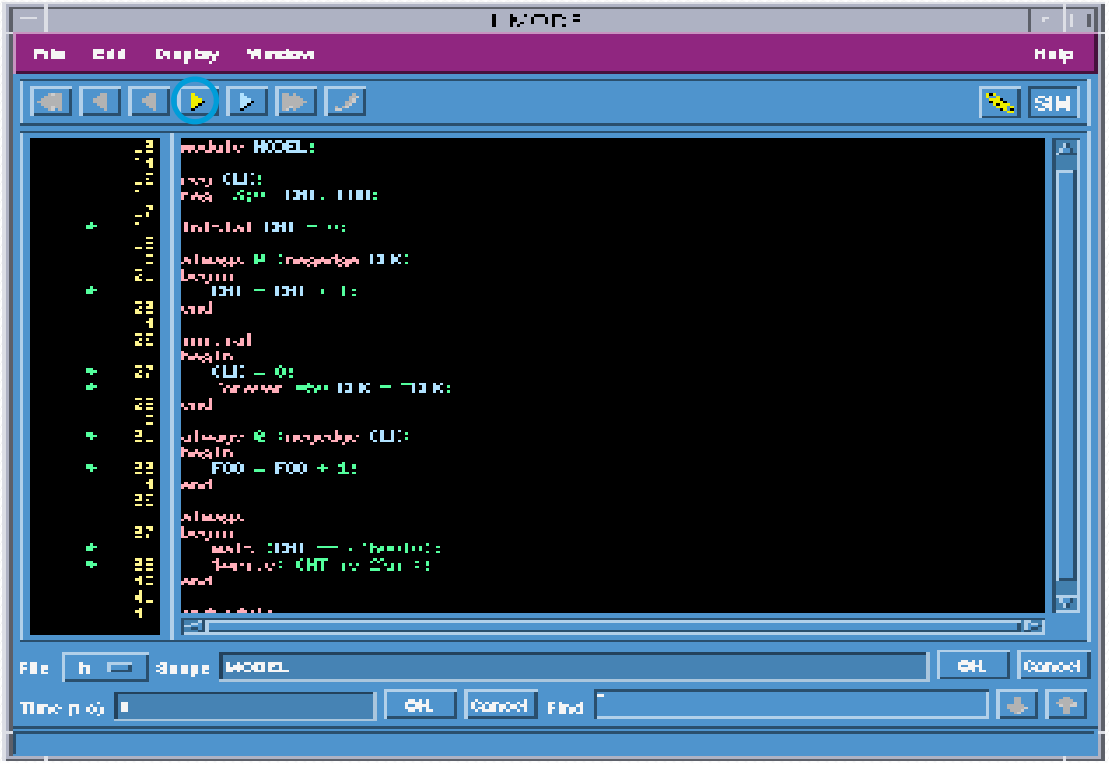
Lab 3.2: VCS

- With the middle mouse button, drag the box named "MODEL" from the Hierarchy window to the Source Code window.



Lab 3.2: VCS

- To single step through the code in the Source Code window, use the yellow arrow button. When you are done, click the Exit button in the main menu.



Lab 3.2: ModelSim

- To single step through the code in the Source window, use the "Step" button.