Constructive Computer Architecture

Tutorial 3

Debugging BSV

Andy Wright
6.175 TA

Review

- Last Tutorial:
  - Scheduling
  - Typeclasses
- Recent Lectures:
  - SMIPS ISA

Software Debugging
Print Statements

- See a bug, not sure what causes it
- Add print statements
- Recompile
- Run
- Still see bug, but you have narrowed it down to a smaller portion of code
- Repeat with more print statements…
- Find bug, fix bug, and remove print statements

BSV Debugging
Display Statements

- See a bug, not sure what causes it
- Add display statements
- Recompile
- Run
- Still see bug, but you have narrowed it down to a smaller portion of code
- Repeat with more display statements…
- Find bug, fix bug, and remove display statements

BSV Display Statements

- The $display() command is an action that prints statements to the simulation console
- Examples:
  - `$display(“Hello World!”);`
  - `$display(“The value of x is %d”, x);`
  - `$display(“The value of y is “, fshow(y));`

Ways to Display Values

Format Specifiers

- %d – decimal
- %b – binary
- %o – octal
- %h – hexadecimal
- %0d, %0b, %0o, %0h
  - Show value without extra whitespace padding

Ways to Display Values
fshow

- fshow is a function in the FShow typeclass
- It can be derived for enumerations and structures
- Example:

```
typedef emun {Red, Blue} Colors deriving(FShow);
Color c = Red;
$display("c is ", fshow(c));
```

Prints "c is Red"

BSV Debugging
Waveform Viewer

- Simulation executables can dump VCD waveforms
  - `./simMyTest -V test.vcd`
- Produces test.vcd containing the values of all the signals used in the simulator
  - Not the same as normal BSV signals
- VCD files can be viewed by a waveform viewer
  - Such as gtkwave
- The signal names and values in test.vcd can be hard to understand
  - Especially for structures and enumerations

BSV Debugging Example

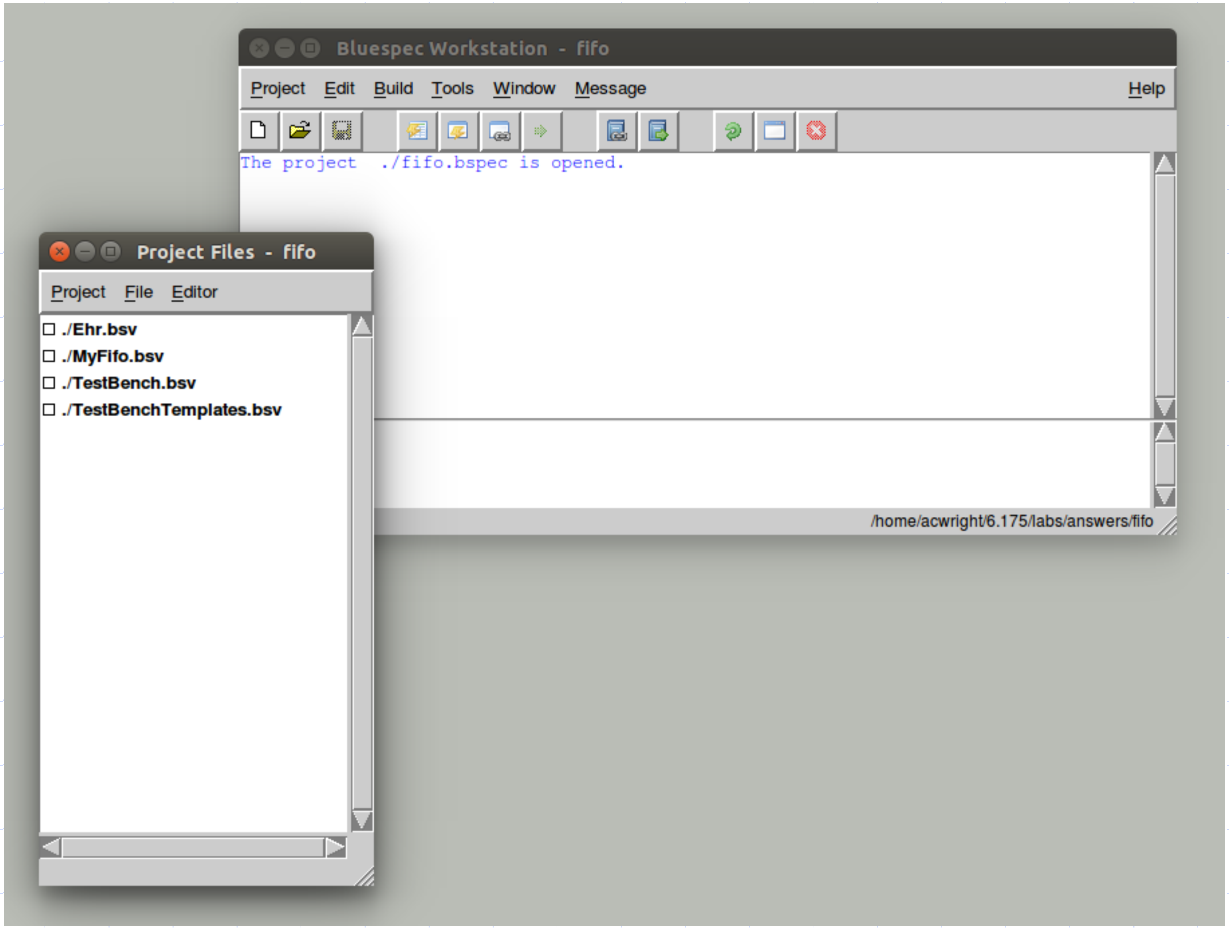
Using the Bluespec GUI and the GTKWave waveform viewer

Step 1
Generate VCD File

- Run ./sim*TestName* -V test.vcd

Step 2
Open Bluespec GUI

- Run “bluespec fifo.bspec”


Bluespec Workstation - fifo
Project Edit Build Tools Window Message Help
Project Files - fifo
Project File Editor

```
The project ./fifo.bspec is opened.
```

□ ./Ehr.bsv
□ ./MyFifo.bsv
□ ./TestBench.bsv
□ ./TestBenchTemplates.bsv

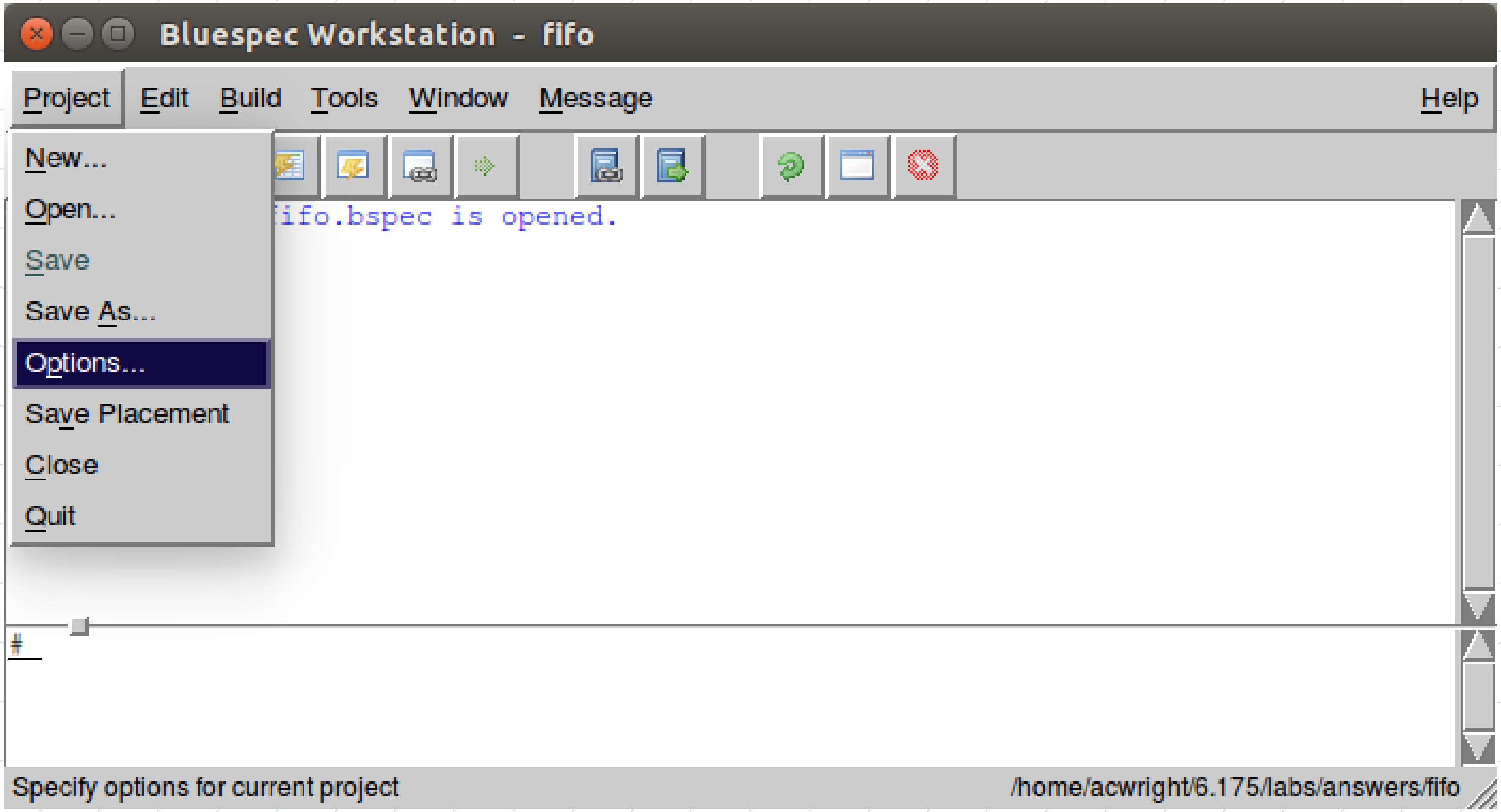
/home/acwright/6.175/labs/answers/fifo

Note, to run the GUI remotely, you need to SSH into the servers with the “ssh –X” command

For the fifo lab, fifo.bspec can be found in

Step 3
Set top module name

- Open project options



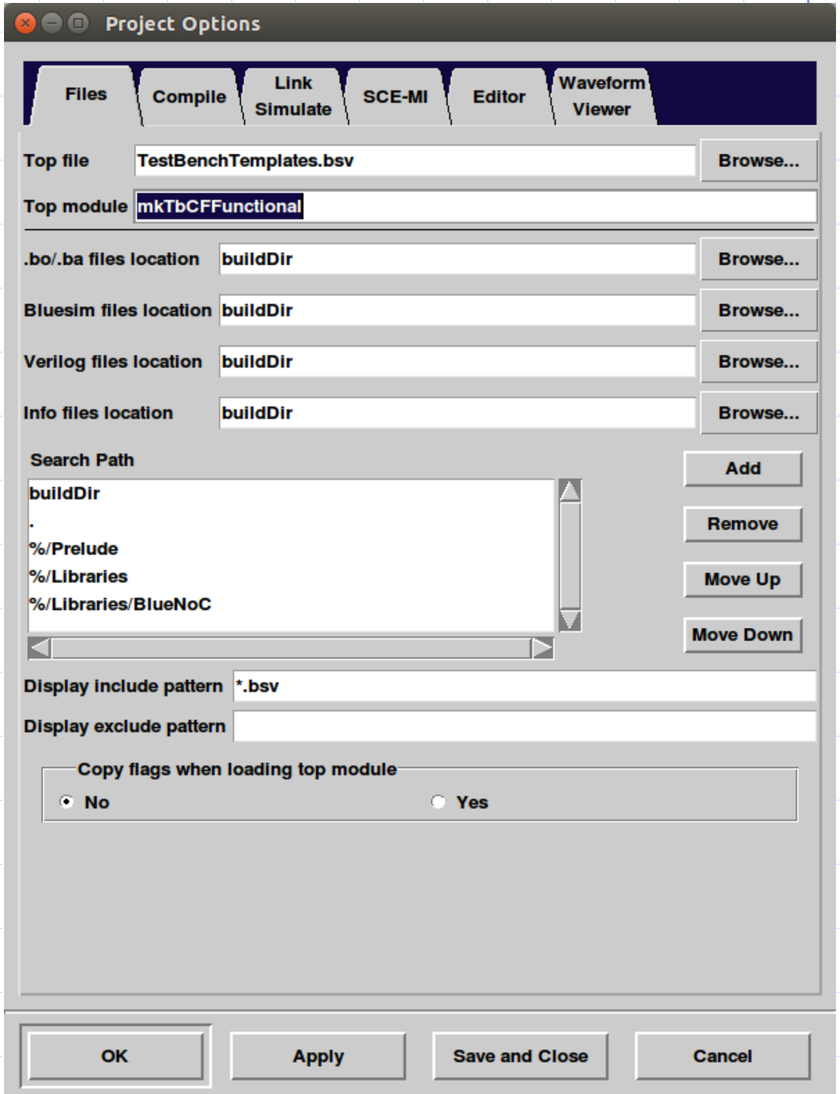
fifo.bspec is opened.

Specify options for current project

/home/acwright/6.175/labs/answers/fifo

Step 3
Set top module name

- Set the top module name to match the compiled module in TestBench.bsv



Project Options

Top file: TestBenchTemplates.bsv

Top module: mkTbCFFunctional

.bo/.ba files location: buildDir

Bluesim files location: buildDir

Verilog files location: buildDir

Info files location: buildDir

Search Path:

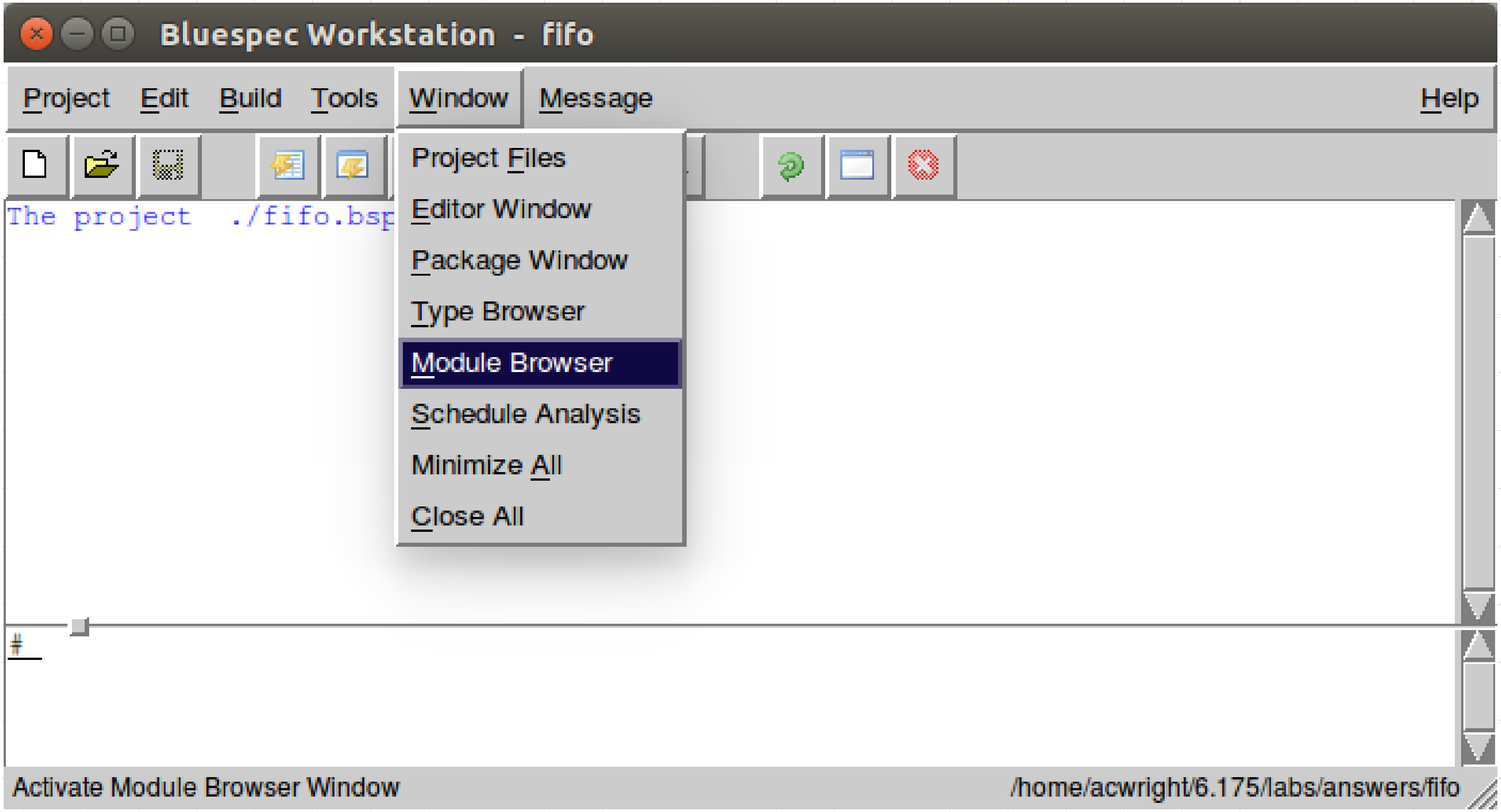
```
buildDir
.
%/Prelude
%/Libraries
%/Libraries/BlueNoC
```

Display include pattern: *.bsv

Display exclude pattern:

Copy flags when loading top module: ● No ○ Yes

Step 4
Open Module Viewer



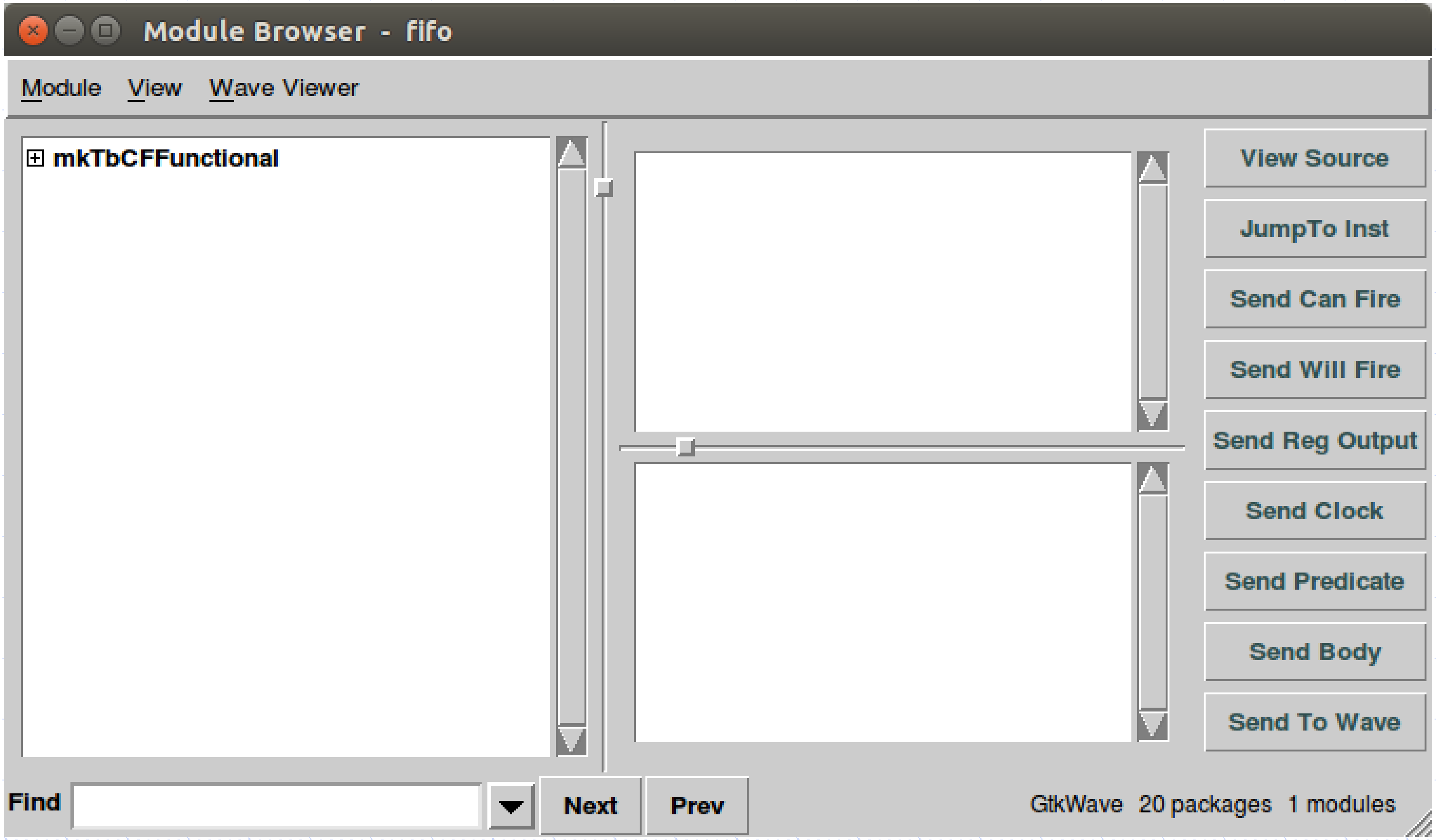
```
The project  ./fifo.bsp
```

```
#
```

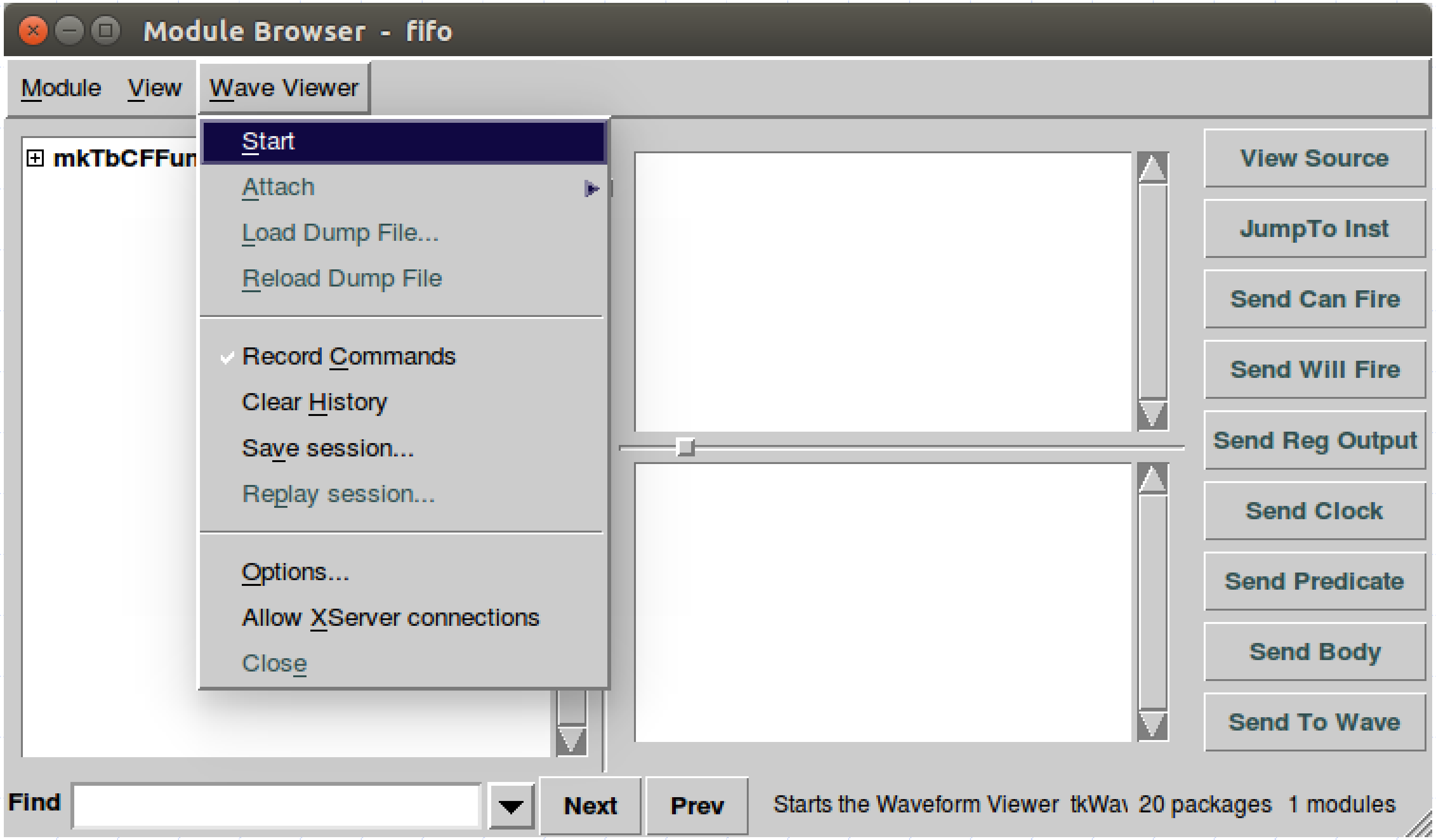
Activate Module Browser Window

/home/acwright/6.175/labs/answers/fifo

Step 4
Open Module Viewer

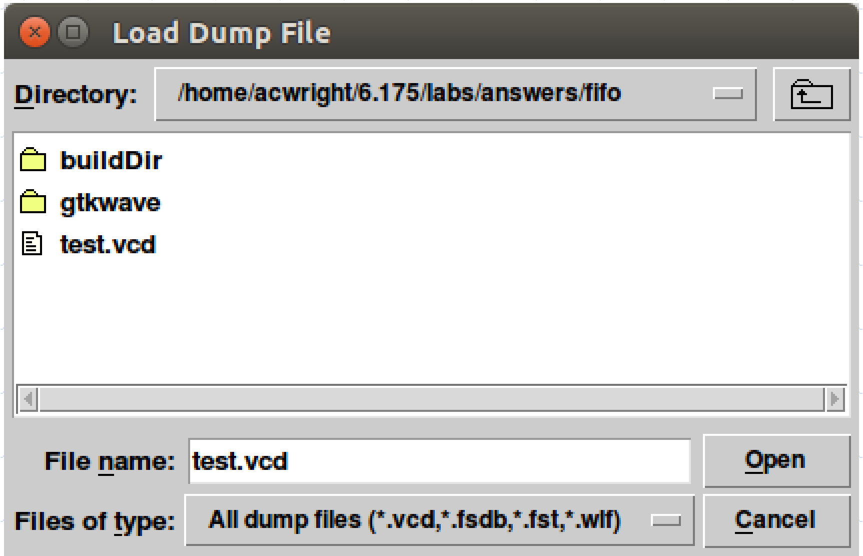


Step 5
Open Wave Viewer



Starts the Waveform Viewer tkWa\ 20 packages 1 modules

Step 5
Open Wave Viewer



Load Dump File

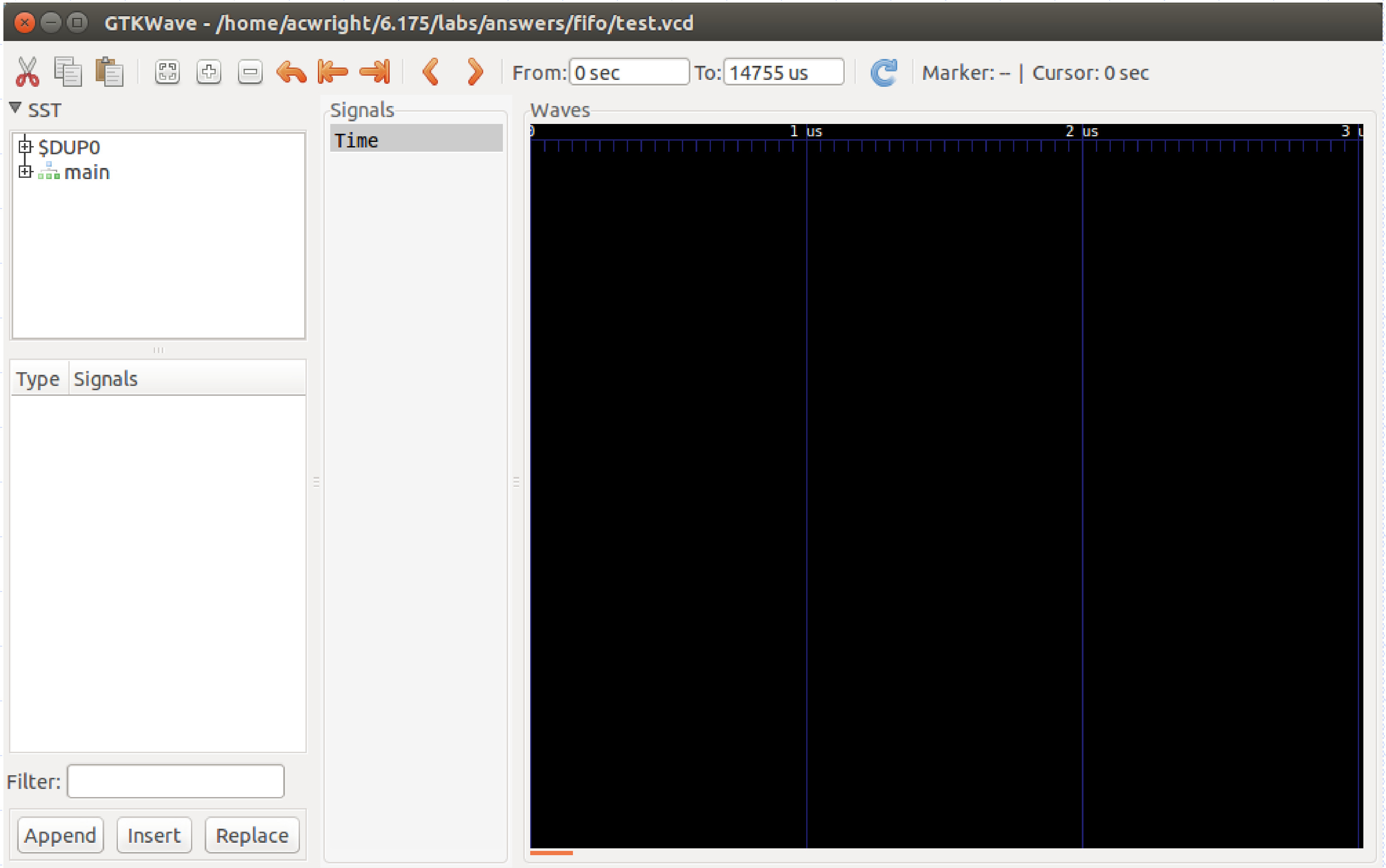
Directory: /home/acwright/6.175/labs/answers/fifo

buildDir
gtkwave
test.vcd

File name: test.vcd

Files of type: All dump files (*.vcd,*.fsdb,*.fst,*.wlf)

Step 6
Open Wave Viewer

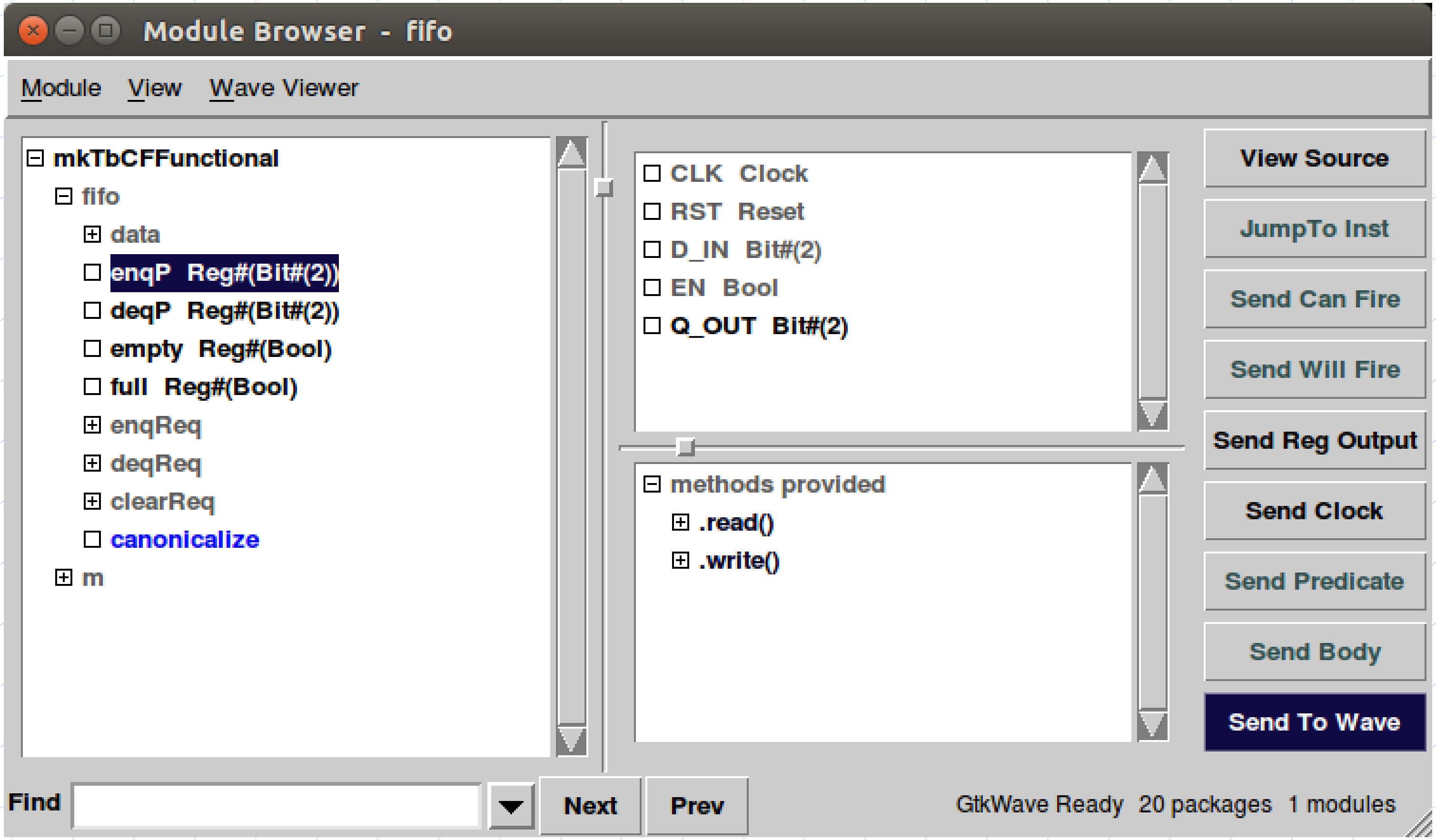


GTKWave - /home/acwright/6.175/labs/answers/fifo/test.vcd

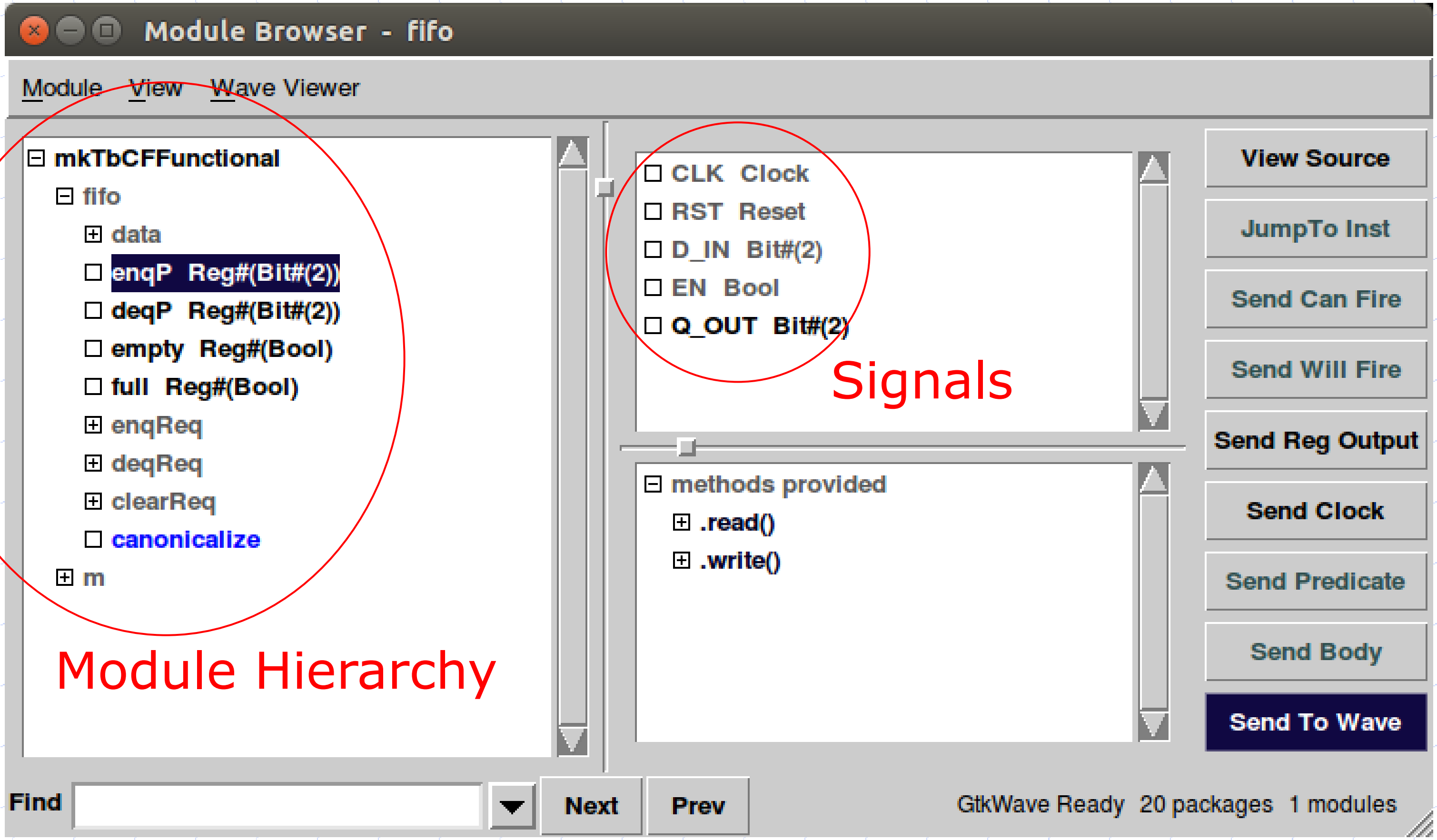
From: 0 sec To: 14755 us

Marker: -- | Cursor: 0 sec

Step 6
Add Some Signals



Step 6
Add Some Signals



⊟ mkTbCFFunctional
⊟ fifo
⊞ data
□ enqP Reg#(Bit#(2))
□ deqP Reg#(Bit#(2))
□ empty Reg#(Bool)
□ full Reg#(Bool)
⊞ enqReq
⊞ deqReq
⊞ clearReq
□ canonicalize
⊞ m

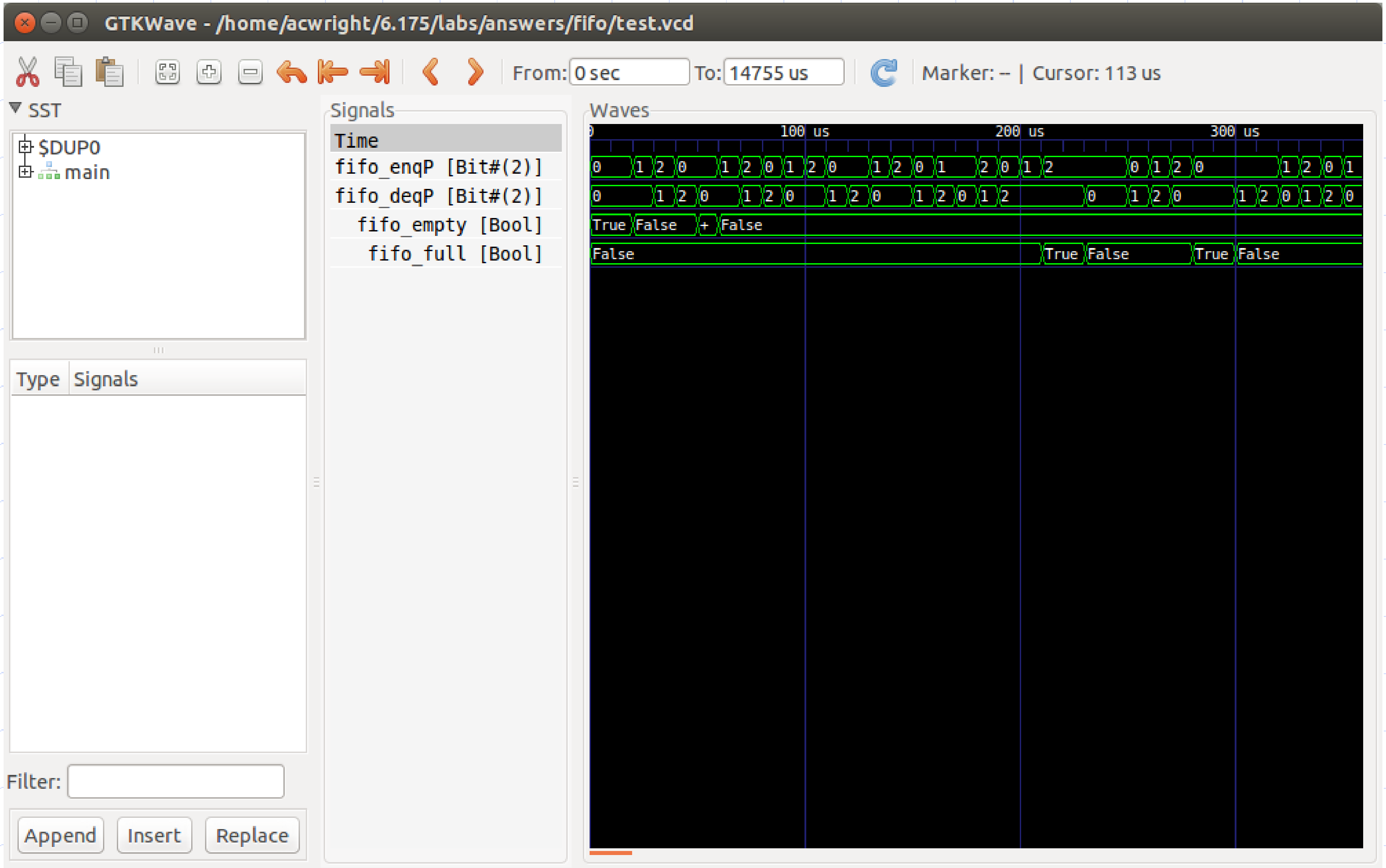
Module Hierarchy

□ CLK Clock
□ RST Reset
□ D_IN Bit#(2)
□ EN Bool
□ Q_OUT Bit#(2)

Signals

⊟ methods provided
⊞ .read()
⊞ .write()

Step 7
Look at the Waveforms



GTKWave - /home/acwright/6.175/labs/answers/fifo/test.vcd

From: 0 sec To: 14755 us

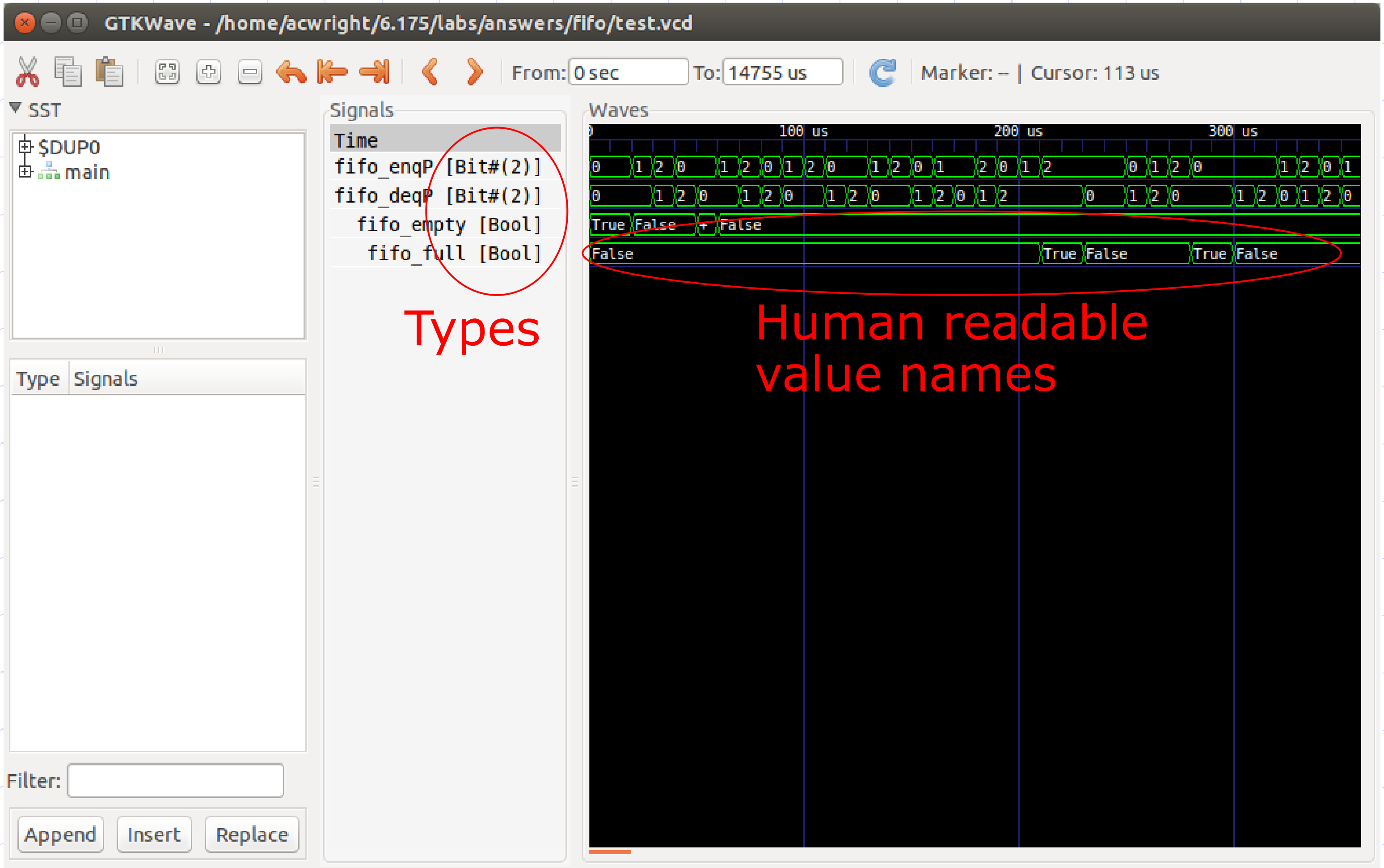
Marker: -- | Cursor: 113 us

Signals

Time
fifo_enqP [Bit#(2)]
fifo_deqP [Bit#(2)]
fifo_empty [Bool]
fifo_full [Bool]

Filter:

Step 7
Look at the Waveforms



GTKWave - /home/acwright/6.175/labs/answers/fifo/test.vcd

From: 0 sec To: 14755 us

Marker: -- | Cursor: 113 us

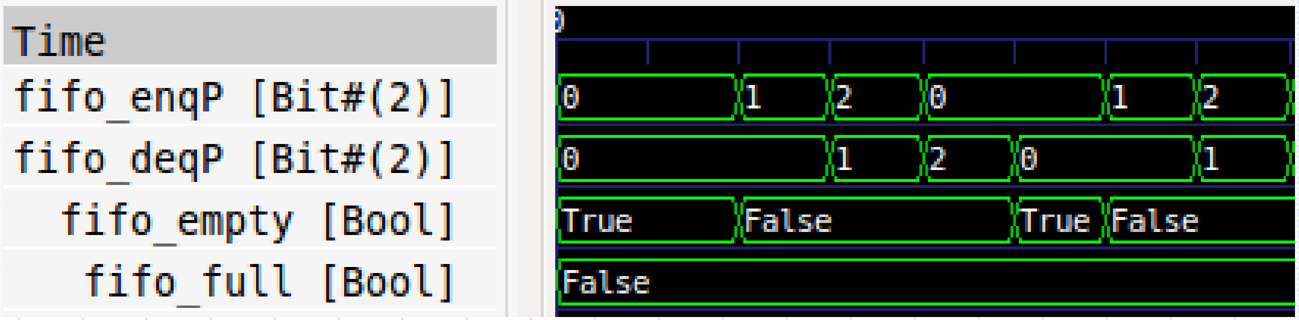
```
Time
fifo_enqP [Bit#(2)]
fifo_deqP [Bit#(2)]
  fifo_empty [Bool]
   fifo_full [Bool]
```

```
0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1
0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0
True False + False
False True False True False
```

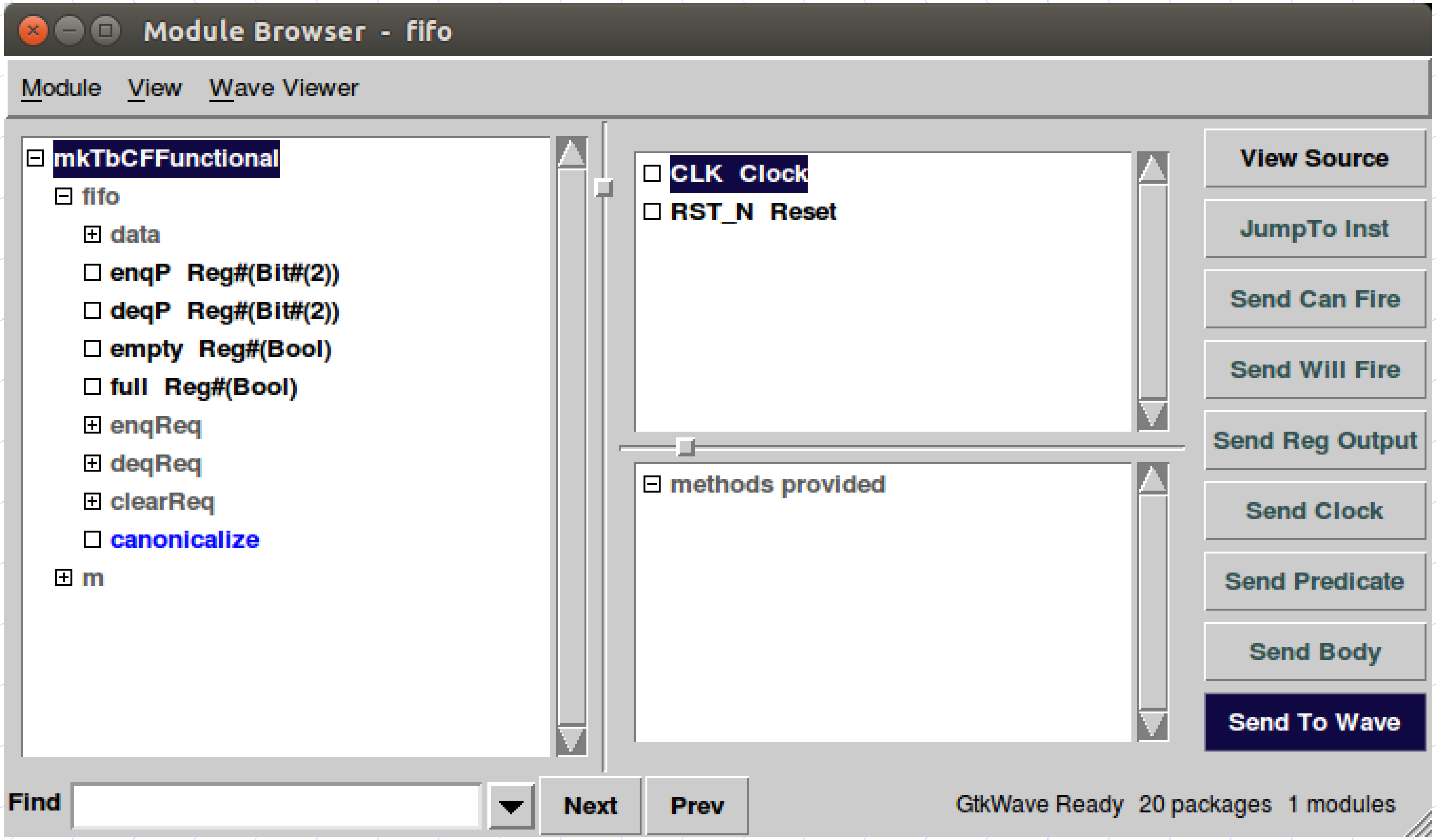
Types

Human readable value names

Step 7
Look at the Waveforms



Step 8
Add Some More Signals



Module Browser - fifo

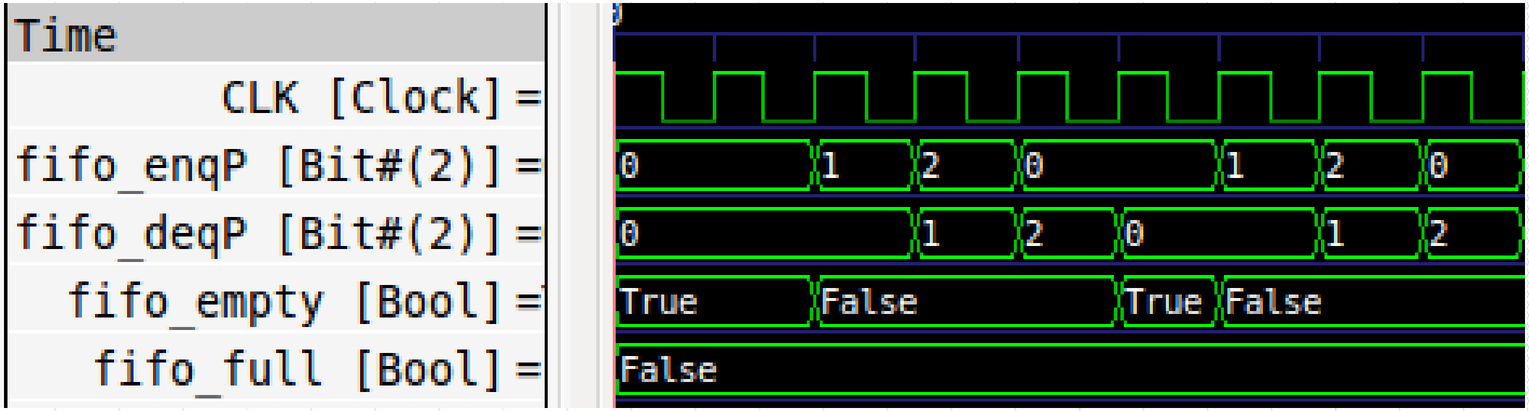
- ⊟ mkTbCFFunctional
  - ⊟ fifo
    - ⊞ data
    - □ enqP Reg#(Bit#(2))
    - □ deqP Reg#(Bit#(2))
    - □ empty Reg#(Bool)
    - □ full Reg#(Bool)
    - ⊞ enqReq
    - ⊞ deqReq
    - ⊞ clearReq
    - □ canonicalize
  - ⊞ m

□ CLK Clock
□ RST_N Reset

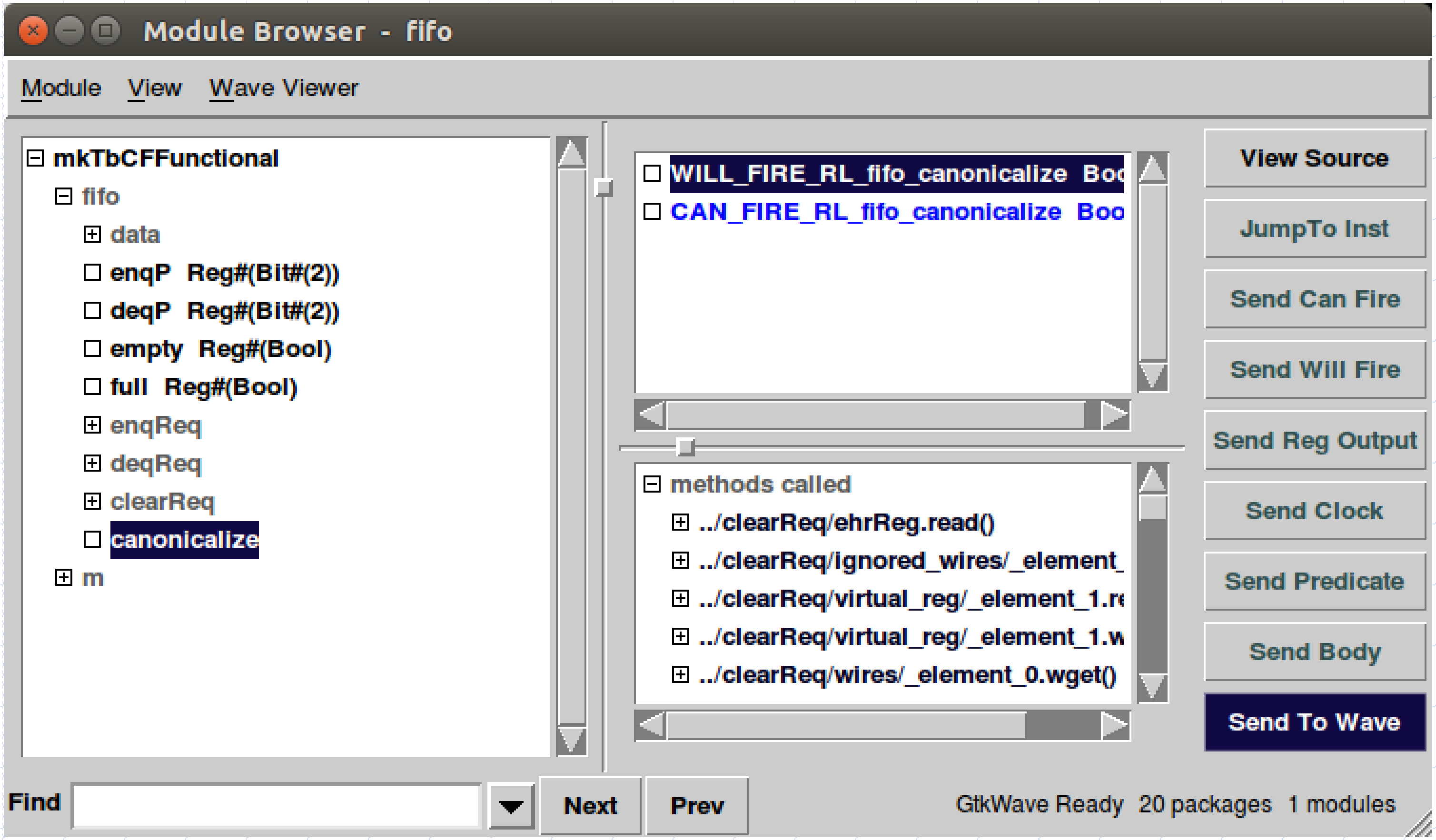
⊟ methods provided

GtkWave Ready 20 packages 1 modules

Step 8
Add Some More Signals



Step 9
Add Rules Too



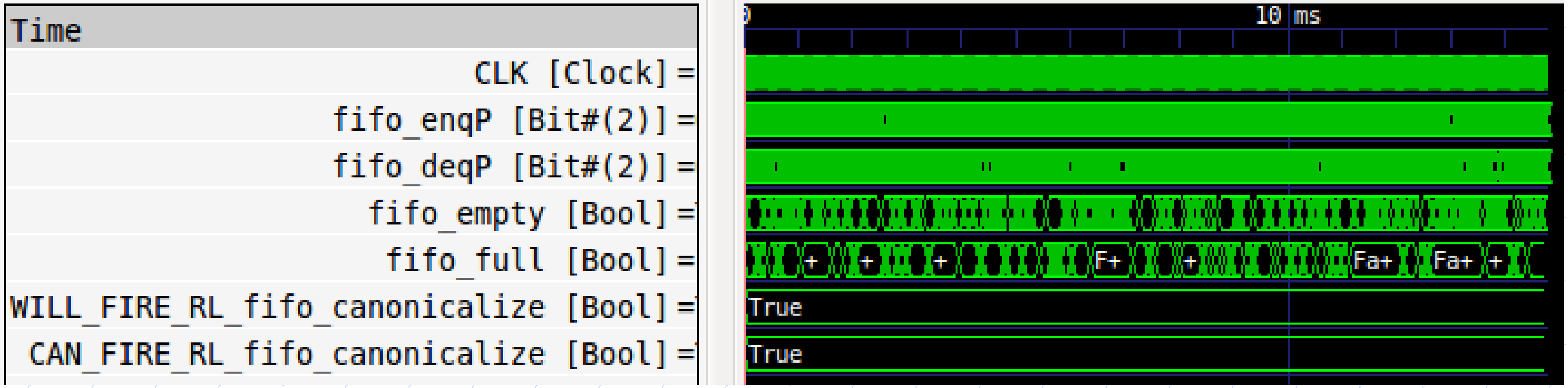
⊟ mkTbCFFunctional
⊟ fifo
⊞ data
□ enqP Reg#(Bit#(2))
□ deqP Reg#(Bit#(2))
□ empty Reg#(Bool)
□ full Reg#(Bool)
⊞ enqReq
⊞ deqReq
⊞ clearReq
□ canonicalize
⊞ m

□ WILL_FIRE_RL_fifo_canonicalize Boo
□ CAN_FIRE_RL_fifo_canonicalize Boo

⊟ methods called
⊞ ../clearReq/ehrReg.read()
⊞ ../clearReq/ignored_wires/_element_
⊞ ../clearReq/virtual_reg/_element_1.re
⊞ ../clearReq/virtual_reg/_element_1.w
⊞ ../clearReq/wires/_element_0.wget()

GtkWave Ready 20 packages 1 modules

Step 9
Add Rules Too



```
Time
CLK [Clock] =
fifo_enqP [Bit#(2)] =
fifo_deqP [Bit#(2)] =
fifo_empty [Bool] =
fifo_full [Bool] =
WILL_FIRE_RL_fifo_canonicalize [Bool] =
CAN_FIRE_RL_fifo_canonicalize [Bool] =
```