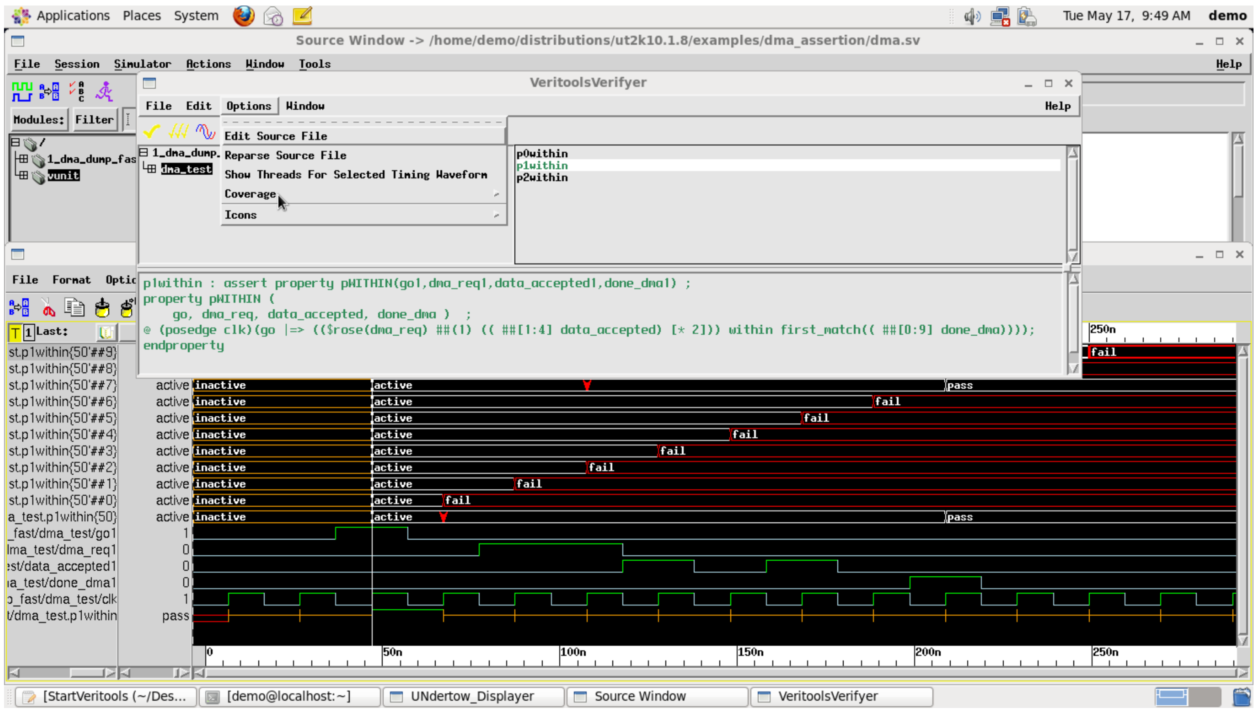
Tutorial for VeritoolsVerifyer, SystemVerilog Assertions

VeritoolsVerifyer



```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

Tutorial for SystemVerilog Assertions

To run the VeritoolsVerifyer, do the following steps:

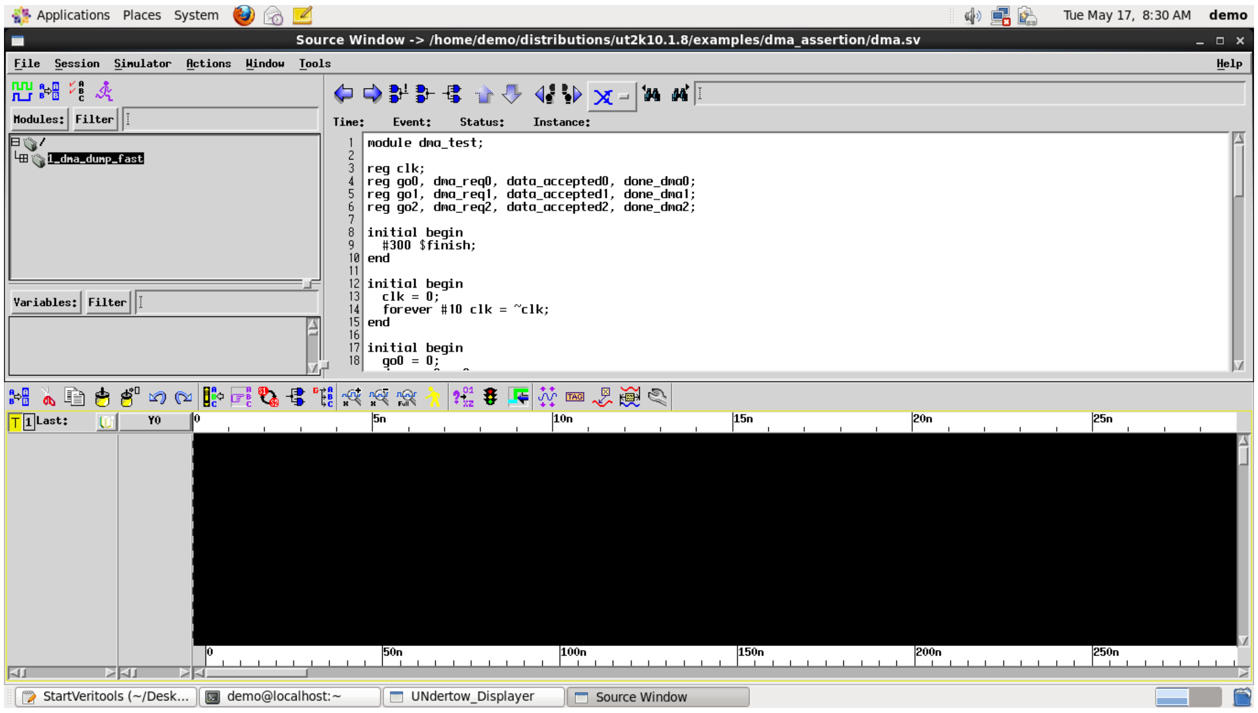
```
% cd /flexlm and launch License Manager

% Run lmgrd for VeritoolsVerifyer license

% cd /(distribution directory)/examples/dma_assertions

% run_ut_batch_new

(Script is: ut –iv –sysver –source –sigfile vt.dump)
```


Applications Places System
Tue May 17, 8:30 AM demo
Source Window -> /home/demo/distributions/ut2k10.1.8/examples/dma_assertion/dma.sv
File Session Simulator Actions Window Tools Help
Modules: Filter
1_dma_dump_fast
Variables: Filter
Time: Event: Status: Instance:
Last: Y0
StartVeritools (~/Desk...
demo@localhost:~
UNdertow_Displayer
Source Window

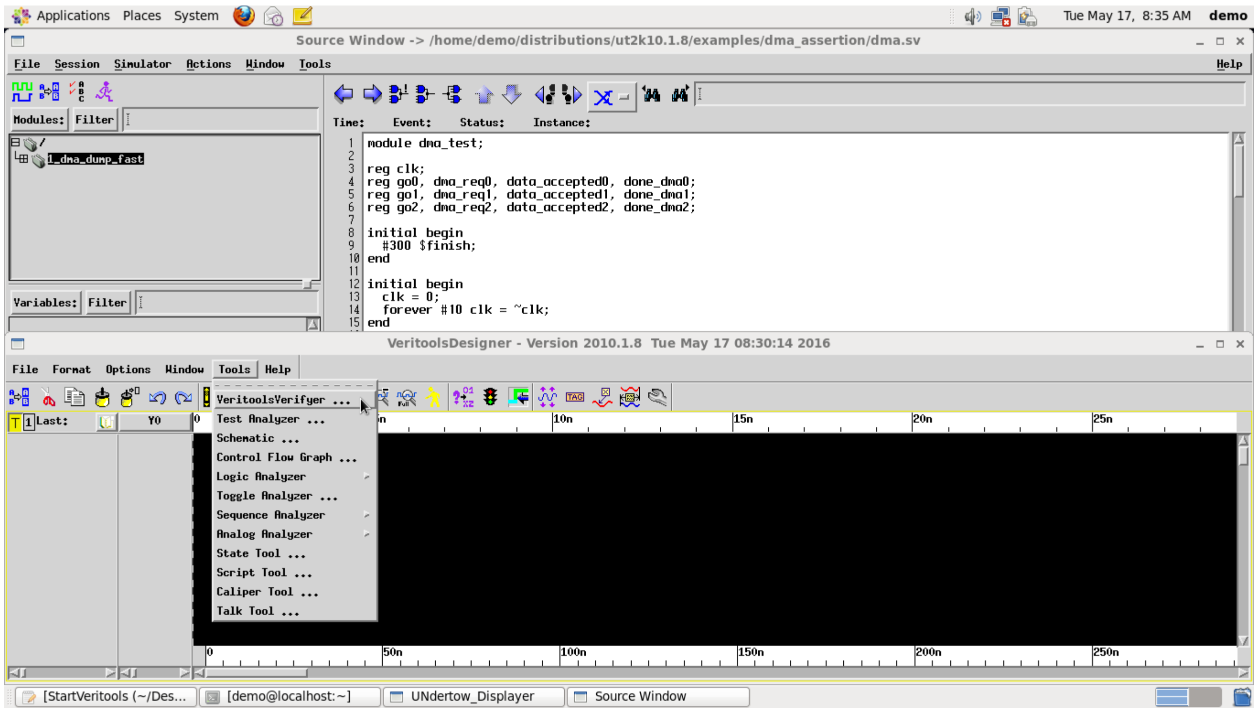
```
module dma_test;

reg clk;
reg go0, dma_req0, data_accepted0, done_dma0;
reg go1, dma_req1, data_accepted1, done_dma1;
reg go2, dma_req2, data_accepted2, done_dma2;

initial begin
  #300 $finish;
end

initial begin
  clk = 0;
  forever #10 clk = ~clk;
end

initial begin
  go0 = 0;
```



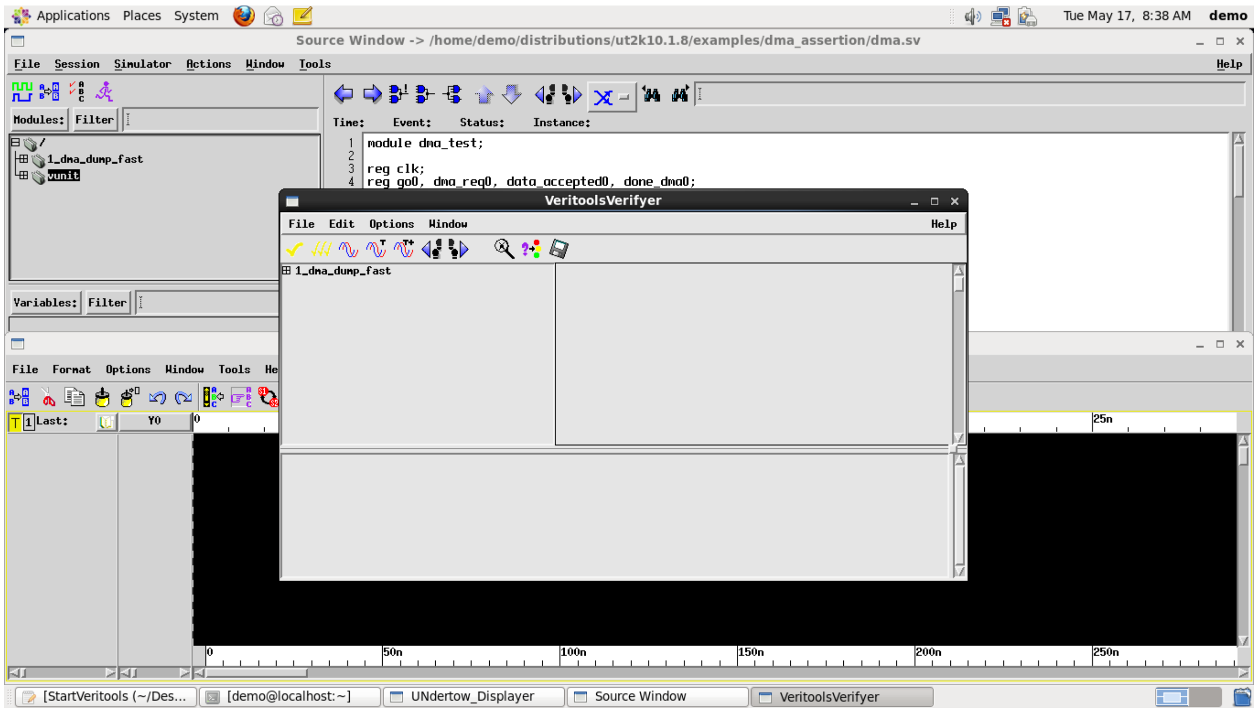
```
module dma_test;

reg clk;
reg go0, dma_req0, data_accepted0, done_dma0;
reg go1, dma_req1, data_accepted1, done_dma1;
reg go2, dma_req2, data_accepted2, done_dma2;

initial begin
  #300 $finish;
end

initial begin
  clk = 0;
  forever #10 clk = ~clk;
end
```

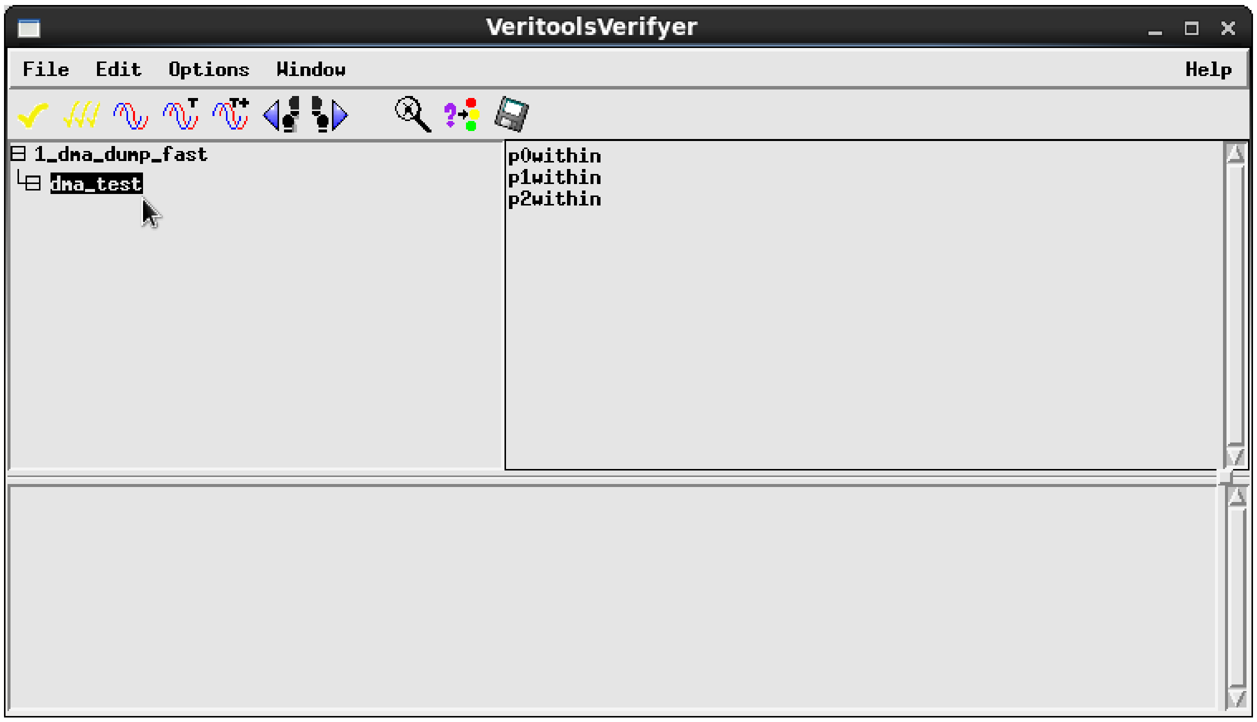
Start the VeritoolsVerifyer, by selecting the **Tools** menu item as shown above.


Applications Places System
Tue May 17, 8:38 AM demo
Source Window -> /home/demo/distributions/ut2k10.1.8/examples/dma_assertion/dma.sv
File Session Simulator Actions Window Tools Help
Modules: Filter
Time: Event: Status: Instance:
1_dma_dump_fast
vunit
Variables: Filter
VeritoolsVerifyer
File Edit Options Window Help
1_dma_dump_fast
File Format Options Window Tools He
Last: Y0
0 25n 50n 100n 150n 200n 250n
[StartVeritools (~/Des...
[demo@localhost:~]
UNdertow_Displayer
Source Window
VeritoolsVerifyer

```
module dma_test;

reg clk;
reg go0, dma_req0, data_accepted0, done_dma0;
```

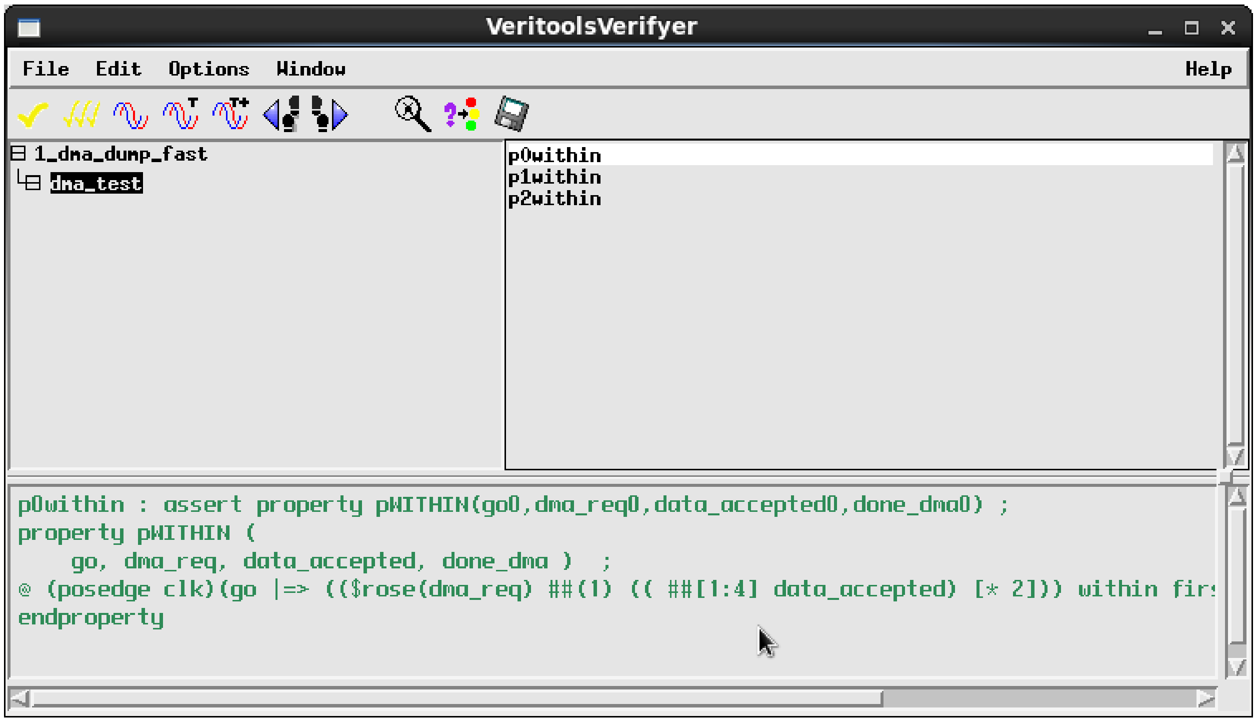
The VeritoolsVerifyer window contains an Icon Bar near the top under the Menu Items, a Heirarchy area in the Upper Right, the window holding the assertions for the selected module, and the Bottom Window displays the text for the selected assertion. Selecting the + in the Module Window opens the modules below this level (see next screen.)



```
1_dma_dump_fast
dma_test
```

```
p0within
p1within
p2within
```

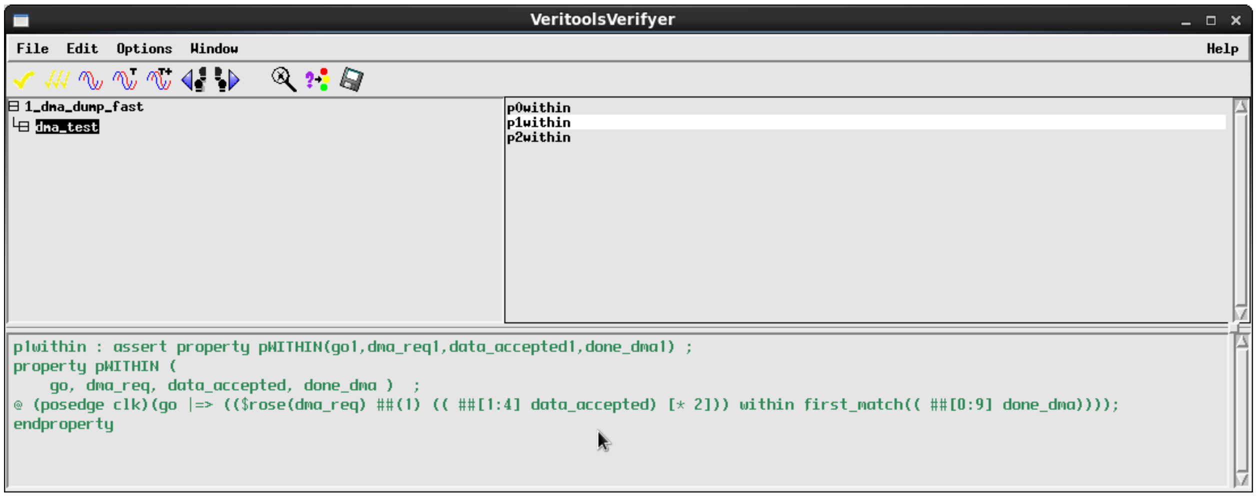
By selecting the Name of the Module, the assertions in that module are displayed in the Assertion List Window.



```
p0within
p1within
p2within
```

```
p0within : assert property pWITHIN(go0,dma_req0,data_accepted0,done_dma0) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within firs
endproperty
```

Selecting an Assertion in the Assertion List Window, displays the Assertion Text in the bottom window. In the screen below, the assertion **<u>p1within</u>** (background is highlighted) is selected:



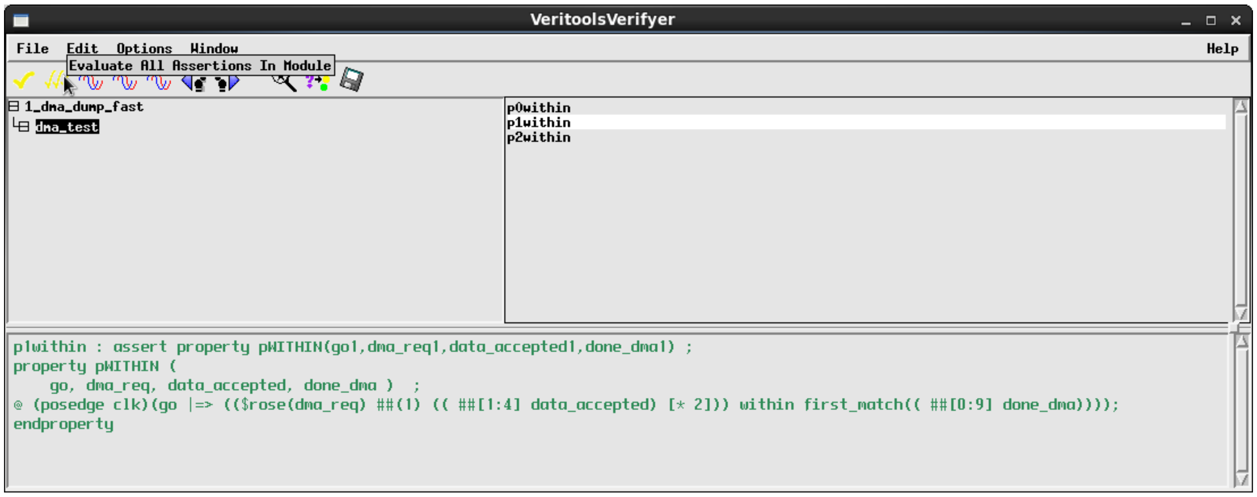
```
p0within
p1within
p2within
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

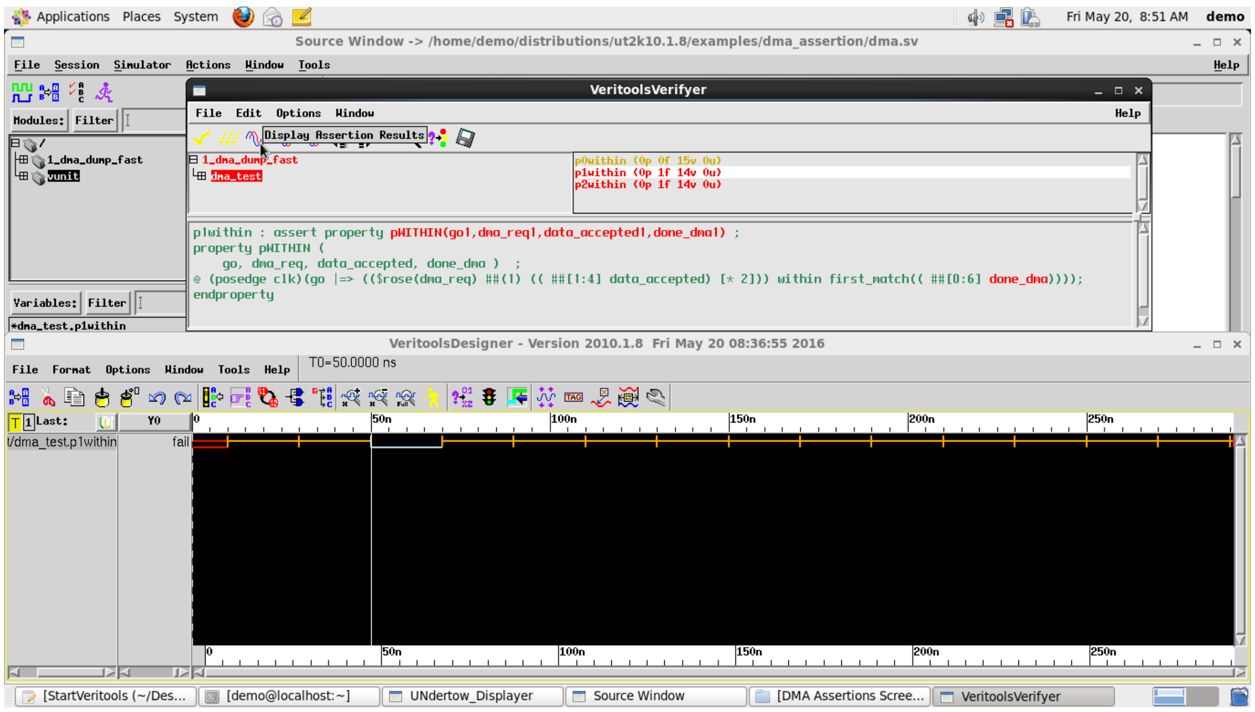
Next, use the **Evaluate All Assertions In Module** icon.

Note: All assertions at this level and all modules below this level will be evaluated.

Also, this will color code the hierarchy, and if even one assertion fails at this level or below this level, the entire hierarchy will be color coded **Red**.



```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma)));
endproperty
```


Source Window -> /home/demo/distributions/ut2k10.1.8/examples/dma_assertion/dma.sv
VeritoolsVerifyer
Display Assertion Results
VeritoolsDesigner - Version 2010.1.8 Fri May 20 08:36:55 2016
T0=50.0000 ns
/dma_test.p1within
fail

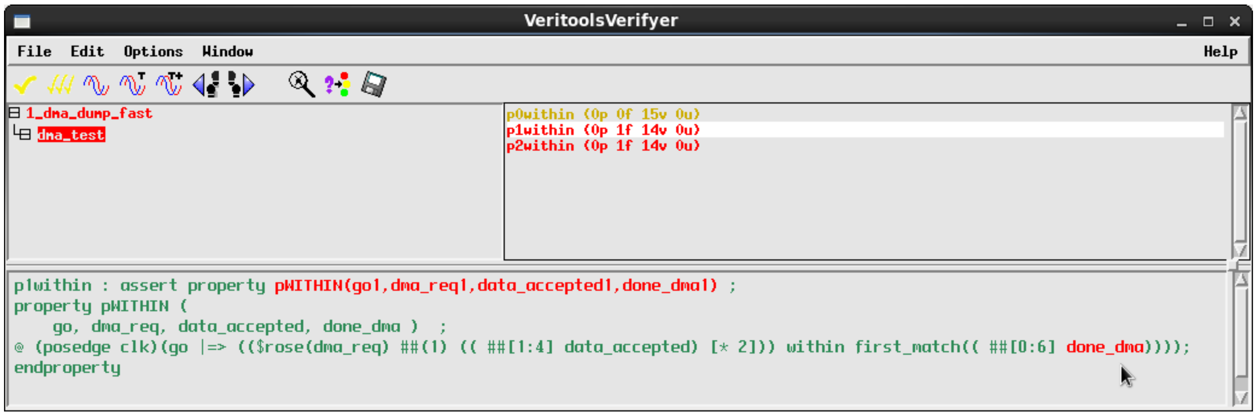
```
1_dma_dump_fast
  dma_test
```

```
p0within (0p 0f 15v 0u)
p1within (0p 1f 14v 0u)
p2within (0p 1f 14v 0u)
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:6] done_dma))));
endproperty
```

After assertions are evaluated, the assertions will turn Yellow if Vacuously True, Green if the assertion Passes and Red if it Fails or if it is still pending past the end of simulation. The next step is to press the **Display Assertion Results** icon. This displays the assertion result waveform in the waveform window, as shown above. A low going pulse indicates the assertion failed, a high going pulse indicates the assertion passed. The leading or falling edge of this pulse indicates where the assertion started.

Note: The cursor is automatically placed on the leading edge of the Assertion Result Pulse, which is a low going pulse if the assertion failed, and a high going pulse if the assertion passed.

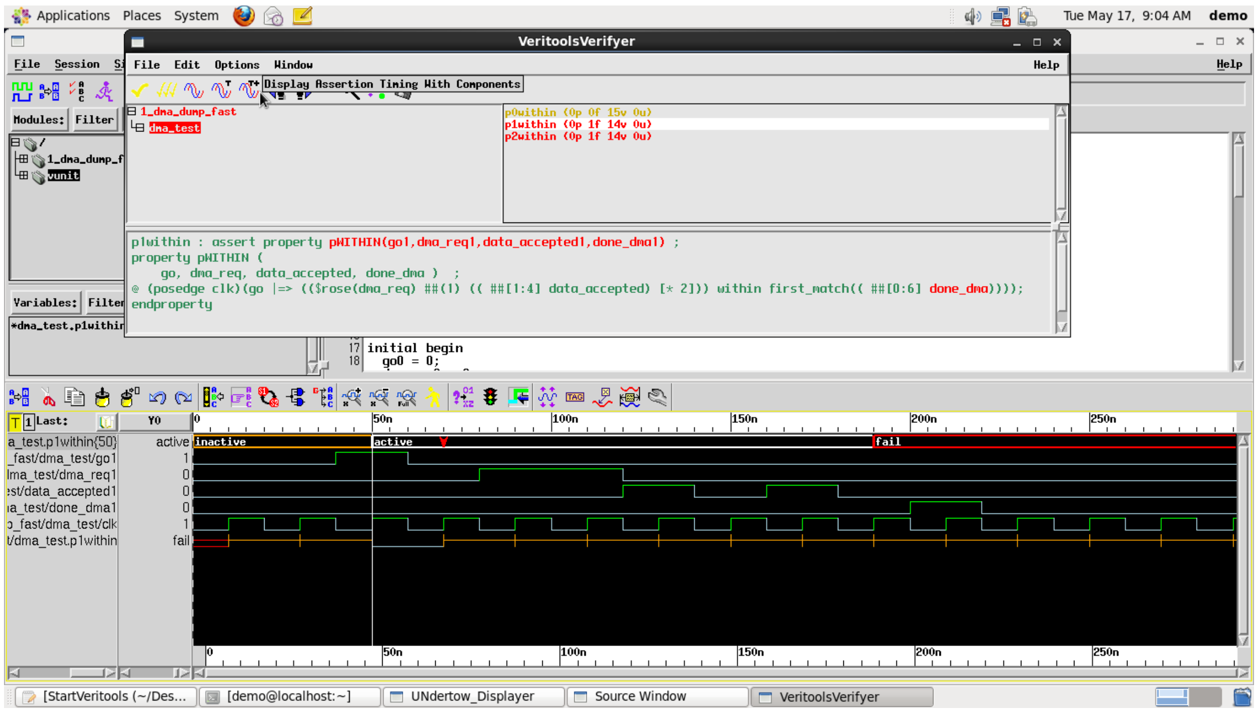


```
p0within (0p 0f 15v 0u)
p1within (0p 1f 14v 0u)
p2within (0p 1f 14v 0u)
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:6] done_dma))));
endproperty
```

After evaluating the assertion and displaying the result, the assertion text is automatically color coded, **Green** for the part of the assertion that Passes evaluation, **Red** for the part that does not pass evaluation.

Note: the assertion is evaluated only when the T0 cursor is on the leading edge of the assertion result pulse, which is automatically done when you press the **Display Assertion Result** icon.

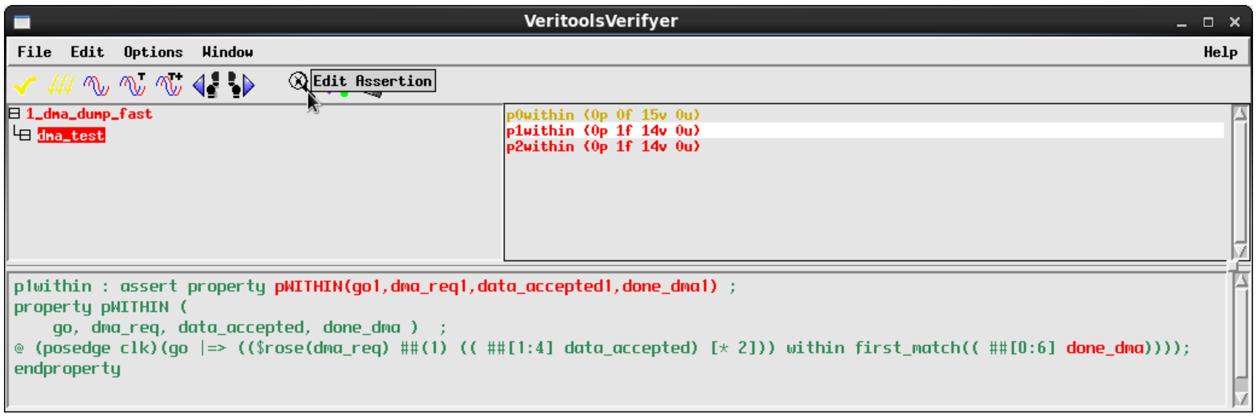


```
p0within (0p 0f 15v 0u)
p1within (0p 1f 14v 0u)
p2within (0p 1f 14v 0u)
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:6] done_dma))));
endproperty
```

Next, click the **Display Assertion Timing with Components** icon. See above.

Note: the assertion timing signal indicates when the assertion is inactive, when it goes active and the latest possible time when it goes to a True or Fail condition.

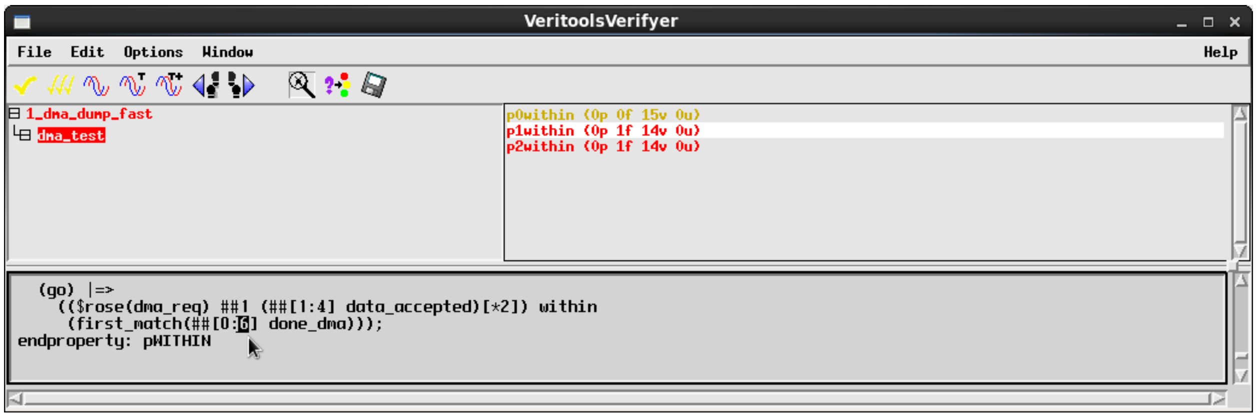


```
p0within (0p 0f 15v 0u)
p1within (0p 1f 14v 0u)
p2within (0p 1f 14v 0u)
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:6] done_dma))));
endproperty
```

To fix the error condition causing the assertion not to pass, look at the **Red** portion of the assertion. The condition causing the assertion to fail is the parameter defining the requirements for this signal. In this case, **dma_done** is required to come true within clocks 0 through 6. But as seen in the waveform viewer, it comes true on clock 7.

To fix this assertion, press the **Edit Assertion** icon, which turns the assertion text window into an “edit/what if” window. After pressing the **Edit Assertion** button, the text in the “edit/what if” window will no longer be color coded as shown in the next screen shot.

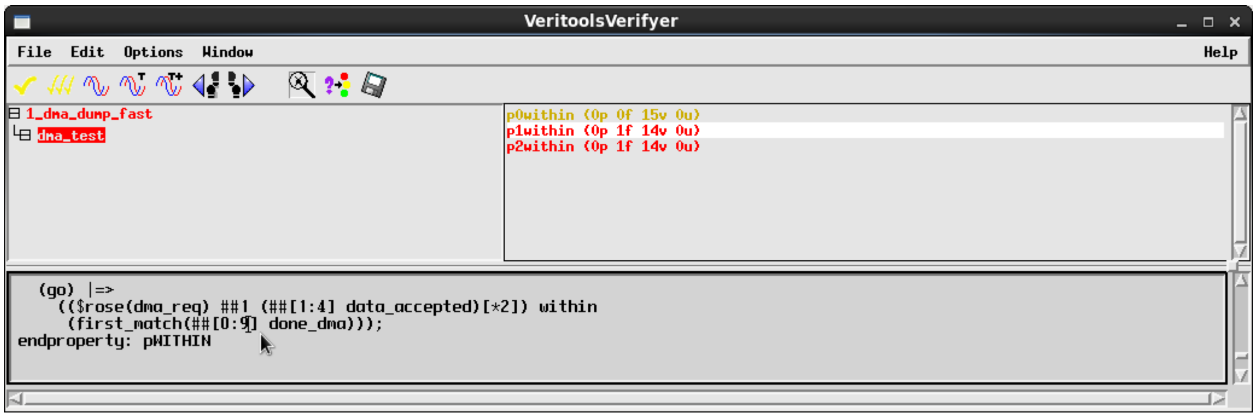


```
1_dma_dump_fast
  dma_test
```

```
p0within (0p 0f 15v 0u)
p1within (0p 1f 14v 0u)
p2within (0p 1f 14v 0u)
```

```
   (go) |=>
     (($rose(dma_req) ##1 (##[1:4] data_accepted)[*2]) within
      (first_match(##[0:6] done_dma)));
endproperty: pWITHIN
```

Replacing the number 6 with a number greater than 6 will cause this assertion to **Pass**.



```
1_dma_dump_fast
  dma_test
```

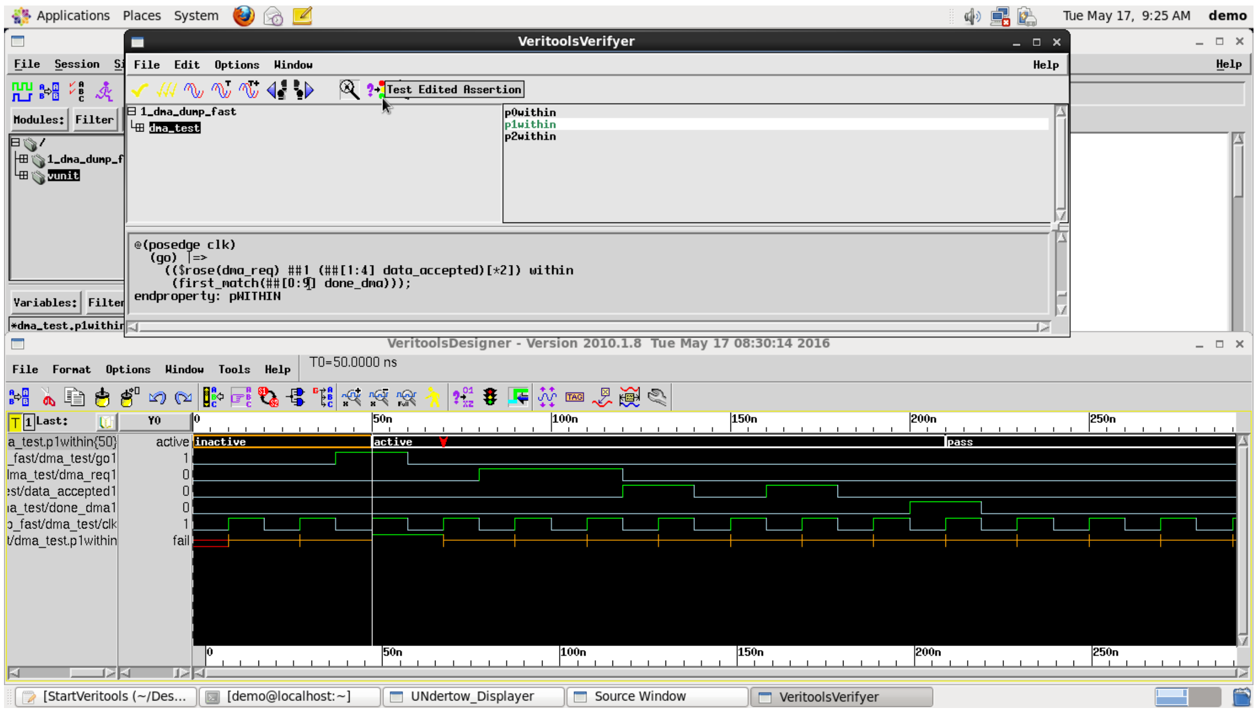
```
p0within (0p 0f 15v 0u)
p1within (0p 1f 14v 0u)
p2within (0p 1f 14v 0u)
```

```
   (go) |=>
     (($rose(dma_req) ##1 (##[1:4] data_accepted)[*2]) within
      (first_match(##[0:9] done_dma)));
endproperty: pWITHIN
```

After editing this assertion, run **Retest**, the icon to the immediate right of the **Edit Assertion** icon:

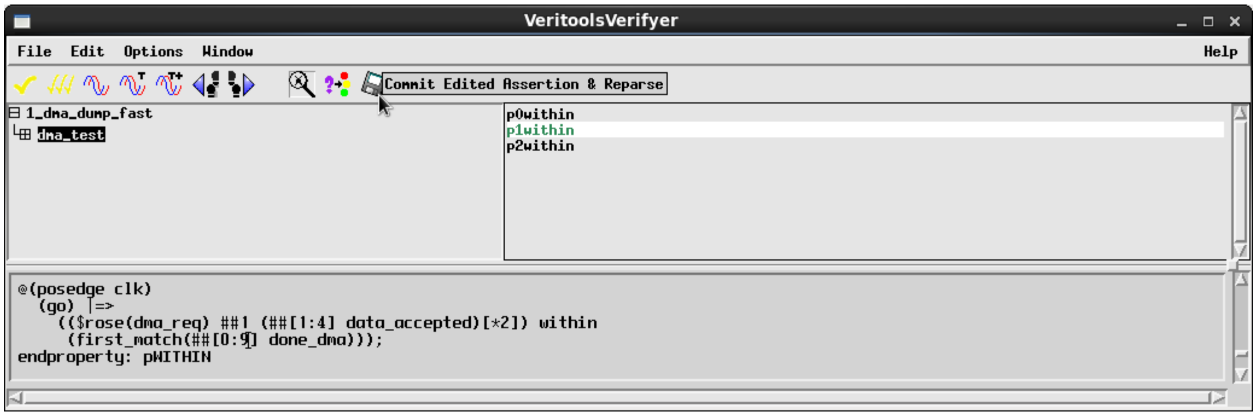
This action will retest the modified assertion code.

The Retest icon is only active when in Edit mode (when the **Edit Assertion** icon is depressed. This allows users to modify and retest this assertion as many times as necessary.



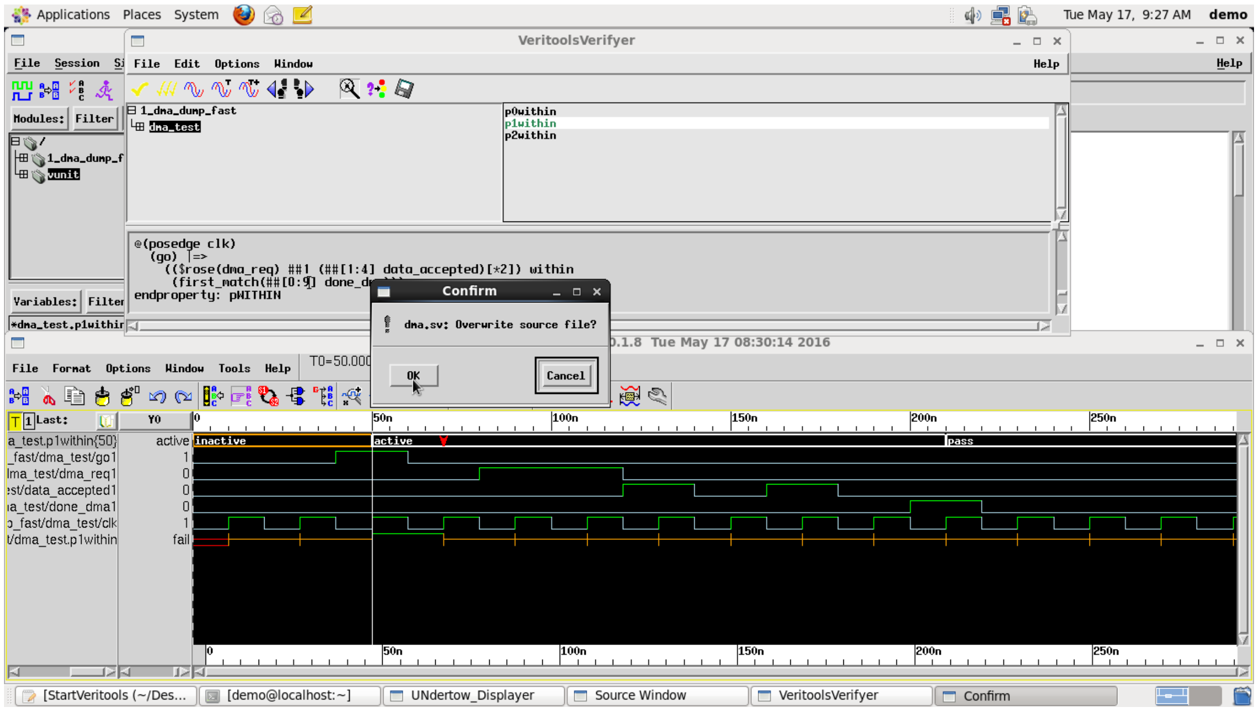
```
@(posedge clk)
  (go) |=>
    (($rose(dma_req) ##1 (##[1:4] data_accepted)[*2]) within
     (first_match(##[0:9] done_dma)));
endproperty: pWITHIN
```

Notice, after clicking **<u>Retest</u>**, if the assertion passes, the assertion timing signal will go inactive, active, pass, the assertion result will now show a high going pulse, and the assertion in the assertion list window will turn **green**. After the assertion passes, the user can then commit the modified code.



```
@(posedge clk)
  (go) |=>
    (($rose(dma_req) ##1 (##[1:4] data_accepted)[*2]) within
     (first_match(##[0:9] done_dma)));
endproperty: pWITHIN
```

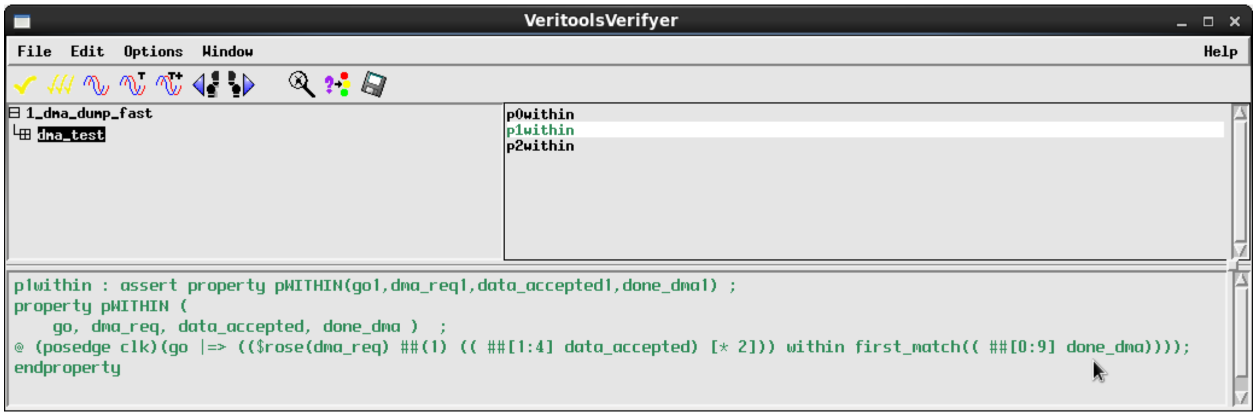
To commit, click the **Commit Edited Assertion & Reparse** icon as above:



```
@(posedge clk)
  (go) |=>
    (($rose(dma_req) ##1 (##[1:4] data_accepted)[*2]) within
     (first_match(##[0:9] done_d
endproperty: pWITHIN
```

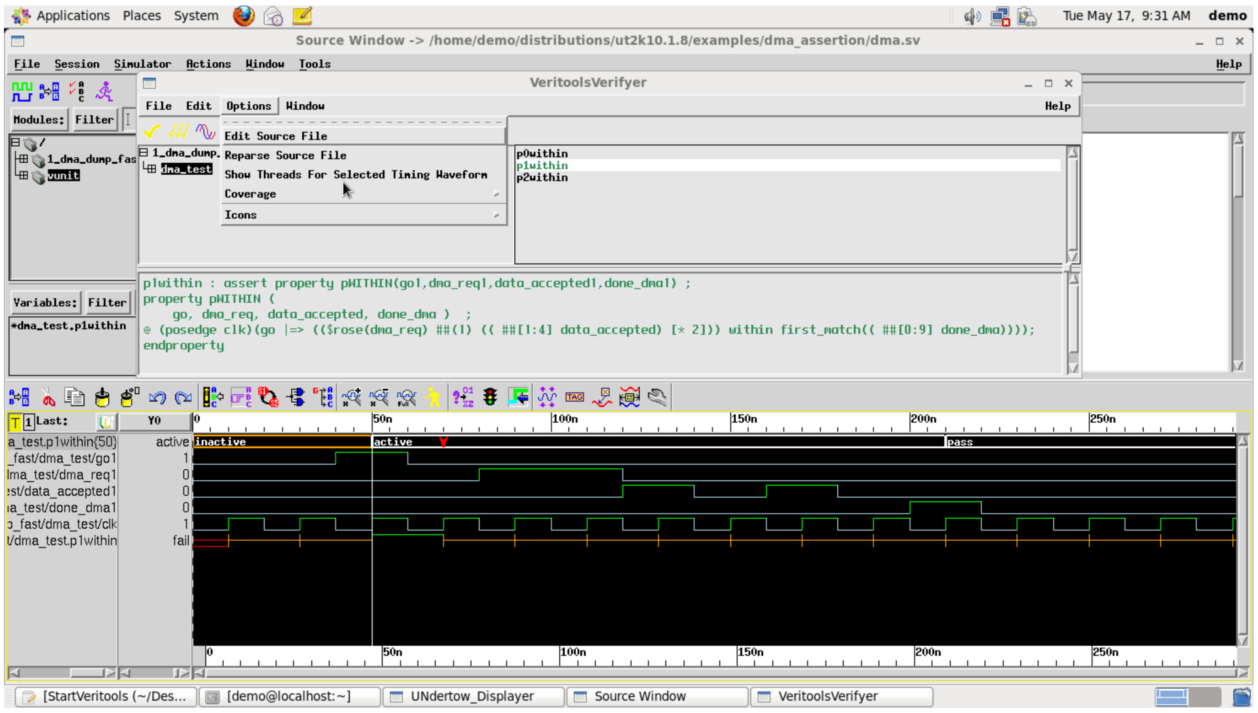
```
dma.sv: Overwrite source file?
```

Note: after pressing **Commit Edited Assertion & Reparse** a dialog box will ask if you wish to overwrite your source file for the assertion code.



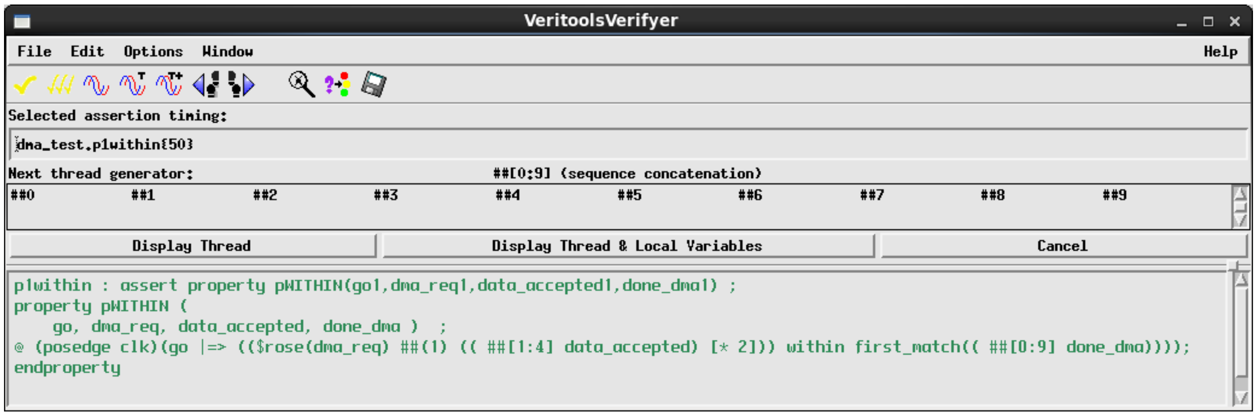
```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

Notice once the code is committed and reparsed, dma_done has changed from **Red** to **Green**, indicating the assertion now passes.


Source Window -> /home/demo/distributions/ut2k10.1.8/examples/dma_assertion/dma.sv
VeritoolsVerifyer
File Edit Options Window Help
Edit Source File
Reparse Source File
Show Threads For Selected Timing Waveform
Coverage
Icons
p0within
p1within
p2within
*dma_test.p1within
inactive
active
pass

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

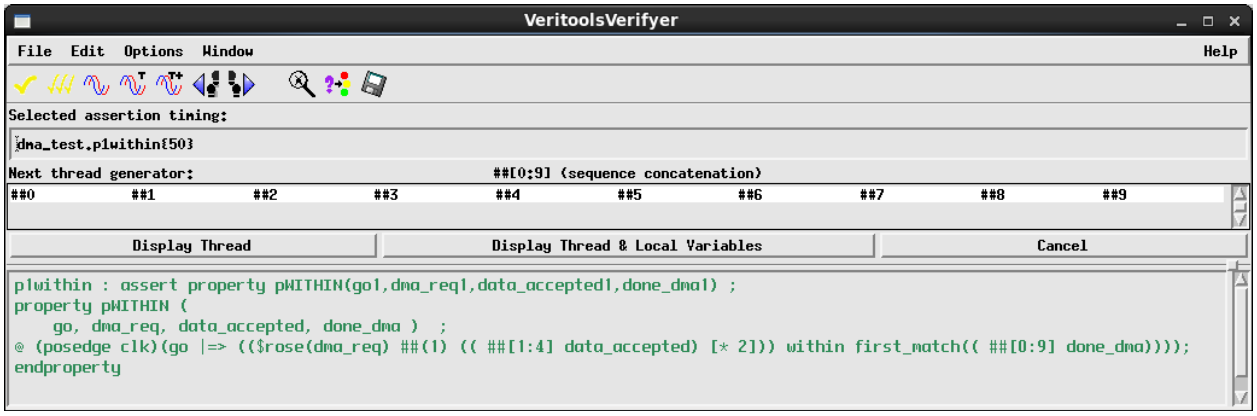
If there is a Red Downward Arrow in the assertion timing waveform (as shown above) it indicates this assertion has unique, independent execution threads. To display the unique, independent execution threads, select (click using left mouse button) the timing waveform (gray highlight, as shown above). Then press Options (pull down menu) -> and click **<u>Show Threads for Selected Timing Waveform</u>**.


VeritoolsVerifyer
File Edit Options Window Help
Display Thread
Display Thread & Local Variables
Cancel

```
Selected assertion timing:
dma_test.p1within{50}
Next thread generator:                    ##[0:9] (sequence concatenation)
##0     ##1     ##2     ##3     ##4     ##5     ##6     ##7     ##8     ##9
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

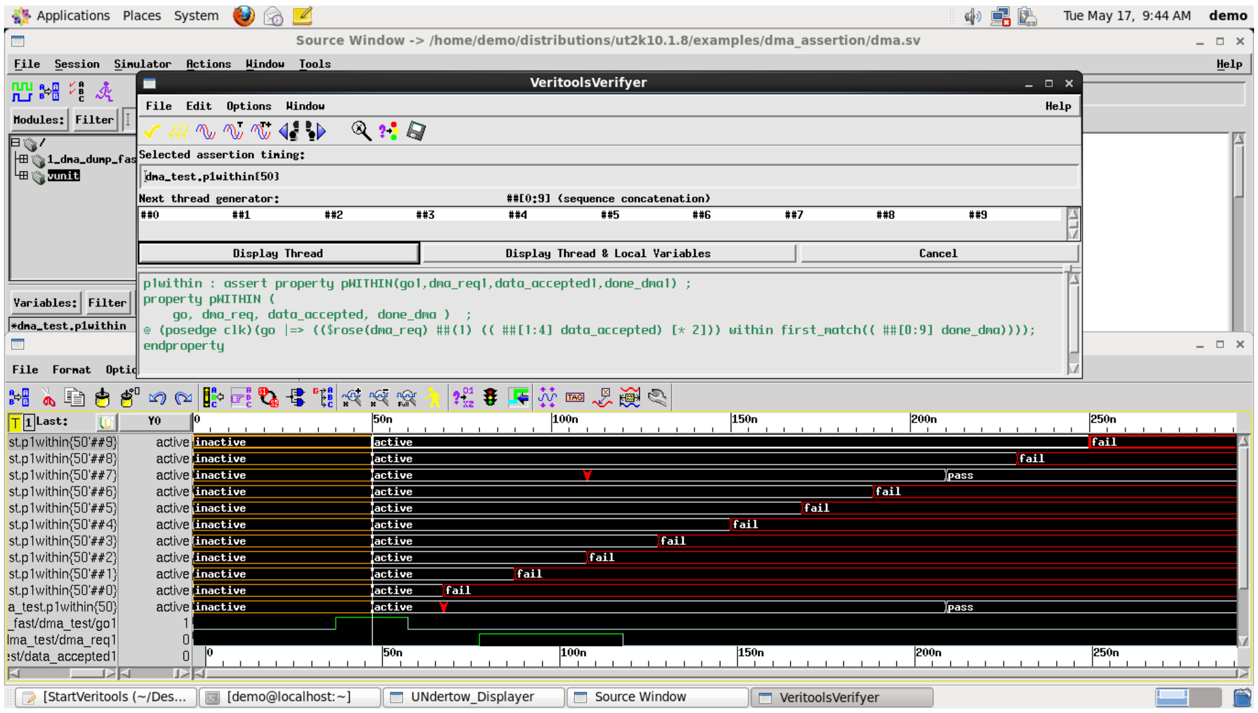
The heirarchy and assertion windows are replaced by the above windows: the **Selected assertion timing** window, and the **Next thread generator** window.


VeritoolsVerifyer
File Edit Options Window Help
Display Thread
Display Thread & Local Variables
Cancel

```
Selected assertion timing:
dma_test.p1within{50}
Next thread generator:                    ##[0:9] (sequence concatenation)
##0     ##1     ##2     ##3     ##4     ##5     ##6     ##7     ##8     ##9
```

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

Next, select the unique independent execution threads that you wish to display in the waveform display window. Click-and-hold with the left mouse button, and slide over the threads you wish to display. If you wish to select individual threads, select (left click) individually.



Selected assertion timing:

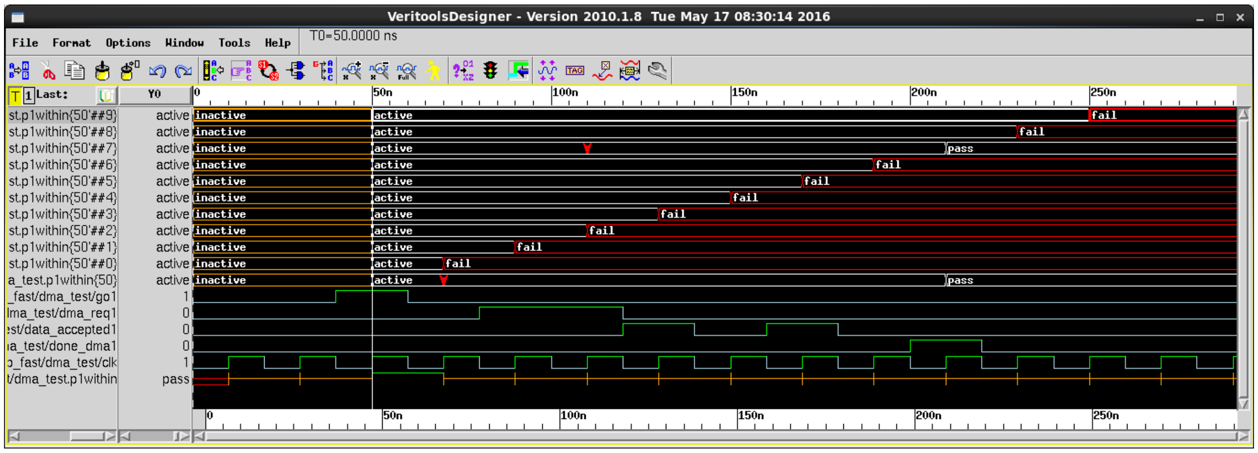
dma_test.p1within{50}

Next thread generator: ##[0:9] (sequence concatenation)

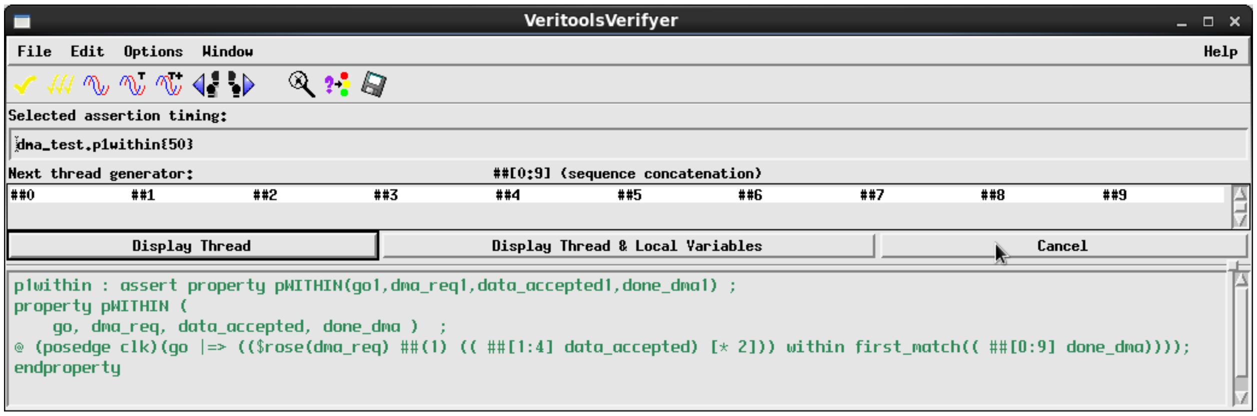
##0 ##1 ##2 ##3 ##4 ##5 ##6 ##7 ##8 ##9

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

Pressing the **<u>Display Thread</u>** button will display the selected threads in the waveform window. Notice within the threads shown, are additional unique independent execution threads, as shown by the downward pointing **Red** arrow, in the new assertion timing signals. This allows you to denest all of the unique independent execution threads in any given assertion.


VeritoolsDesigner - Version 2010.1.8 Tue May 17 08:30:14 2016
File Format Options Window Tools Help
T0=50.0000 ns
Last:
Y0
0
50n
100n
150n
200n
250n
st.p1within{50'##9} active
st.p1within{50'##8} active
st.p1within{50'##7} active
st.p1within{50'##6} active
st.p1within{50'##5} active
st.p1within{50'##4} active
st.p1within{50'##3} active
st.p1within{50'##2} active
st.p1within{50'##1} active
st.p1within{50'##0} active
a_test.p1within{50} active
_fast/dma_test/go1 1
lma_test/dma_req1 0
est/data_accepted1 0
ia_test/done_dma1 0
o_fast/dma_test/clk 1
t/dma_test.p1within pass
inactive
active
fail
pass

The above waveform window displays all the unique independent execution threads along with the timing signal for this assertion, the result signal for this assertion, and the signal components for this assertion.


VeritoolsVerifyer
File Edit Options Window Help
Display Thread
Display Thread & Local Variables
Cancel

Selected assertion timing:

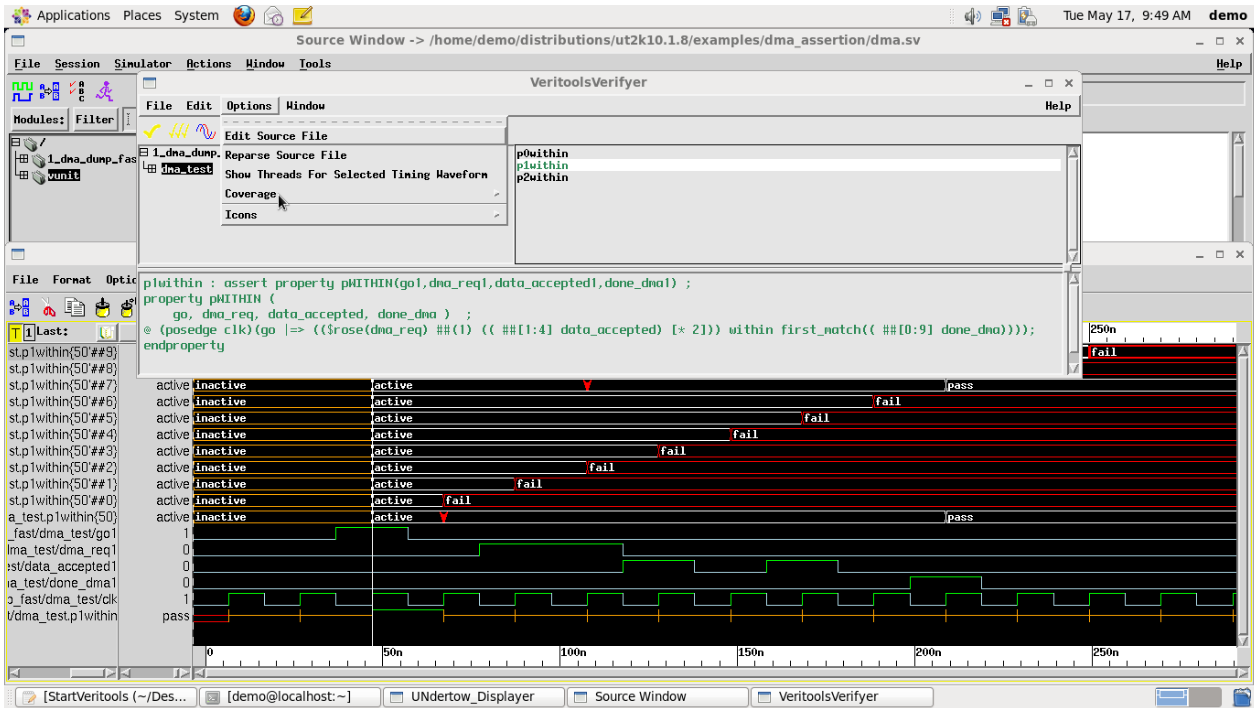
dma_test.p1within{50}

Next thread generator: ##[0:9] (sequence concatenation)

```
##0   ##1   ##2   ##3   ##4   ##5   ##6   ##7   ##8   ##9
```

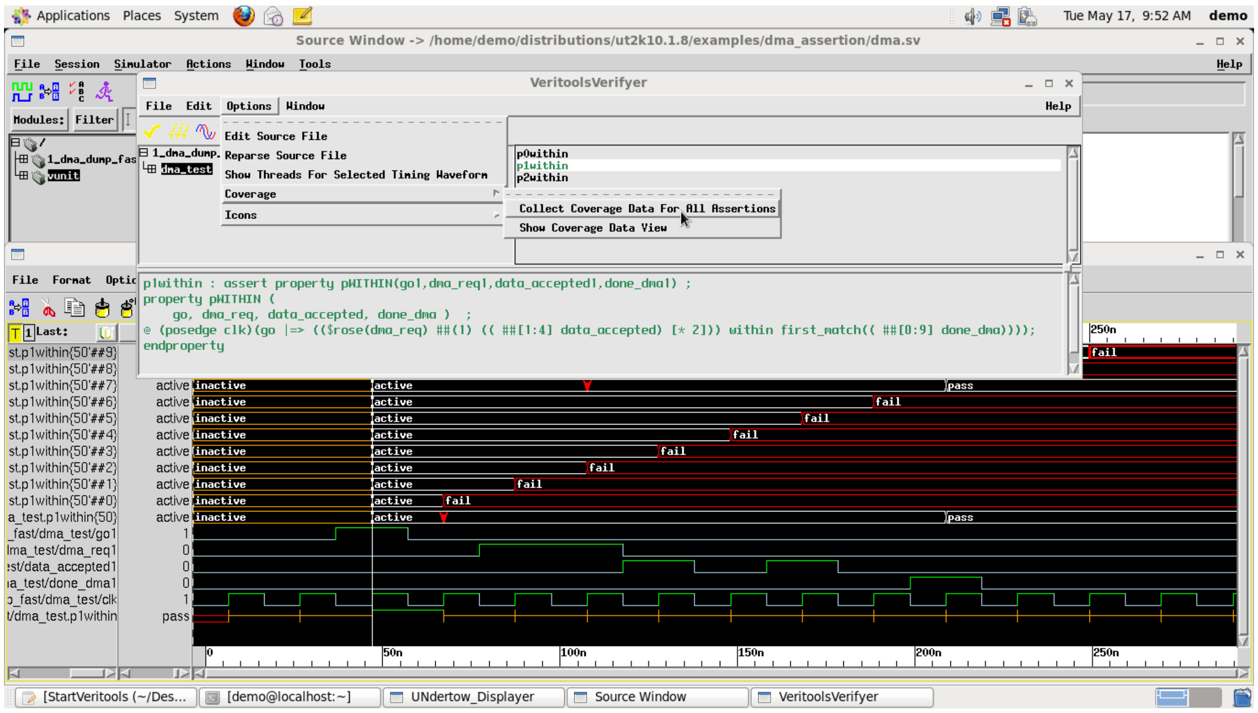
```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

To return to the assertion heirarchy window, click the **Cancel** button.


Source Window -> /home/demo/distributions/ut2k10.1.8/examples/dma_assertion/dma.sv
VeritoolsVerifyer
File Edit Options Window Help
Edit Source File
Reparse Source File
Show Threads For Selected Timing Waveform
Coverage
Icons
p0within
p1within
p2within

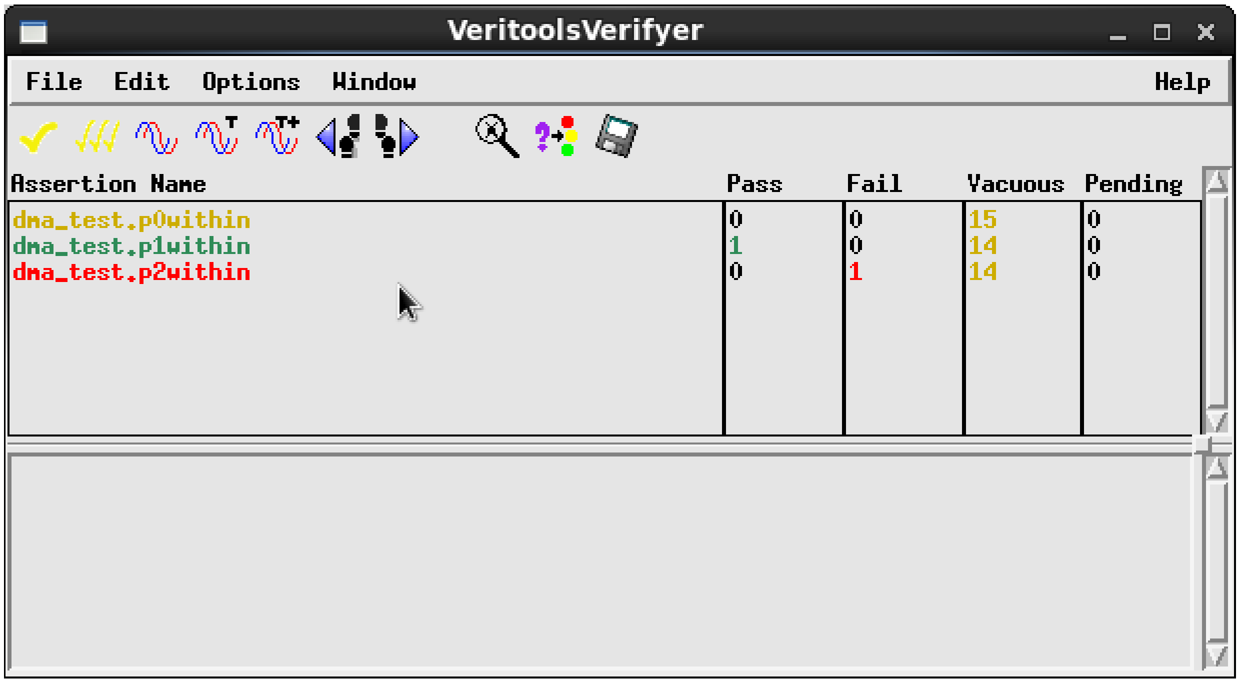
```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

To see Coverage information, select the **Options** pulldown menu, then press **Coverage** (as shown above) to get the sub menu (next screen).


Source Window -> /home/demo/distributions/ut2k10.1.8/examples/dma_assertion/dma.sv
VeritoolsVerifyer
Edit Source File
Reparse Source File
Show Threads For Selected Timing Waveform
Coverage
Icons
Collect Coverage Data For All Assertions
Show Coverage Data View
p0within
p1within
p2within

```
p1within : assert property pWITHIN(go1,dma_req1,data_accepted1,done_dma1) ;
property pWITHIN (
    go, dma_req, data_accepted, done_dma )  ;
@ (posedge clk)(go |=> (($rose(dma_req) ##(1) (( ##[1:4] data_accepted) [* 2])) within first_match(( ##[0:9] done_dma))));
endproperty
```

Click **<u>Collect Coverage Data For All Assertions</u>** as shown above.



Assertion Name	Pass	Fail	Vacuous	Pending
dma_test.p0within	0	0	15	0
dma_test.p1within	1	0	14	0
dma_test.p2within	0	1	14	0

This opens the coverage window, which shows the counts of each assertion: **Passes**, **Fails**, **Vacuously True** or **Pending** at the end of execution.