

Tutorial: Sonata's VHDL simulator

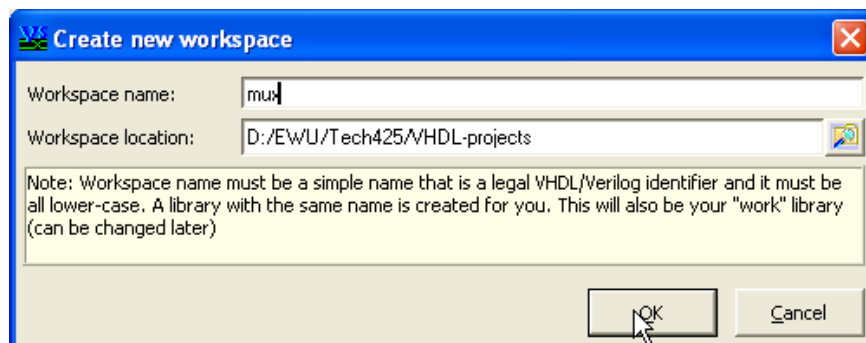
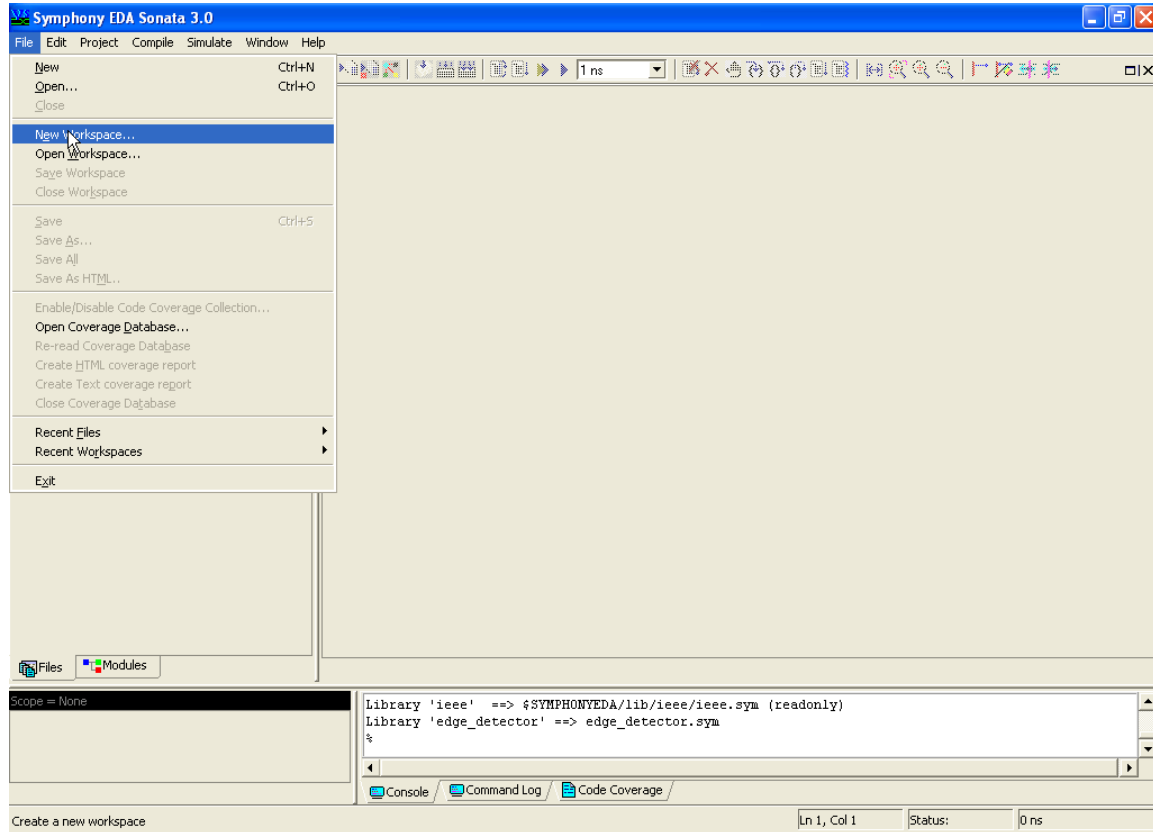
Step 1.

Invoke Sonata VHDL simulator

Start → All Programs → Symphony EDA → VHDL Simili 3 → Sonata

Step 2.

Create a new work space

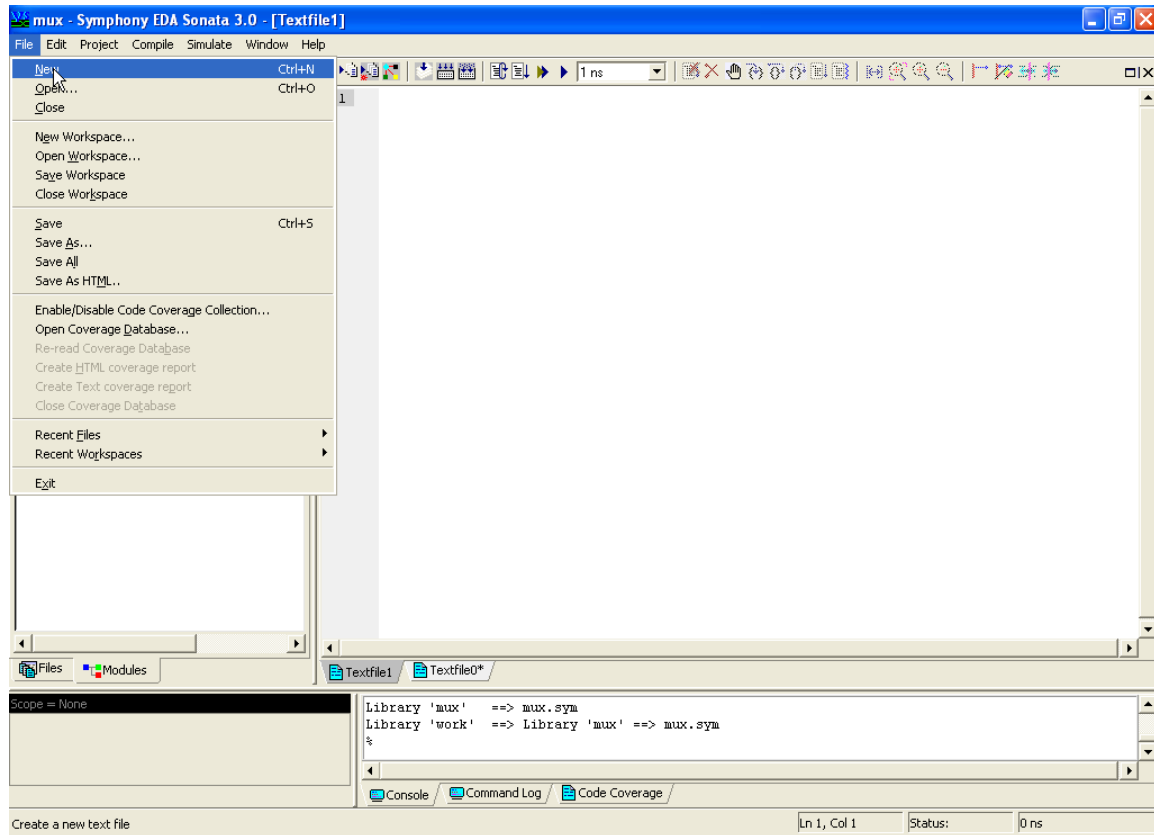


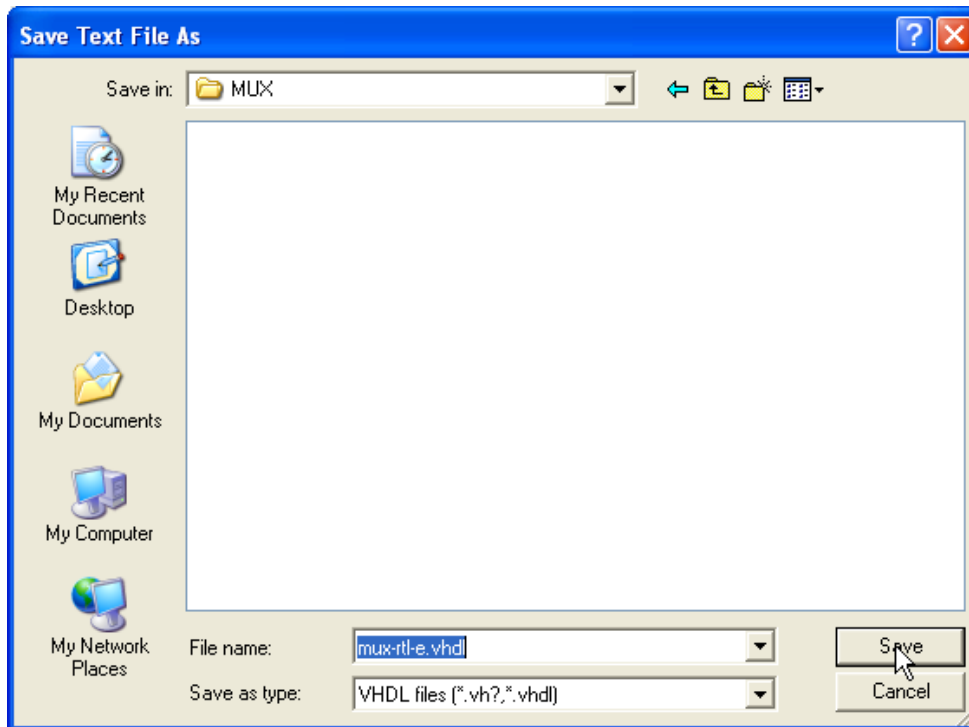
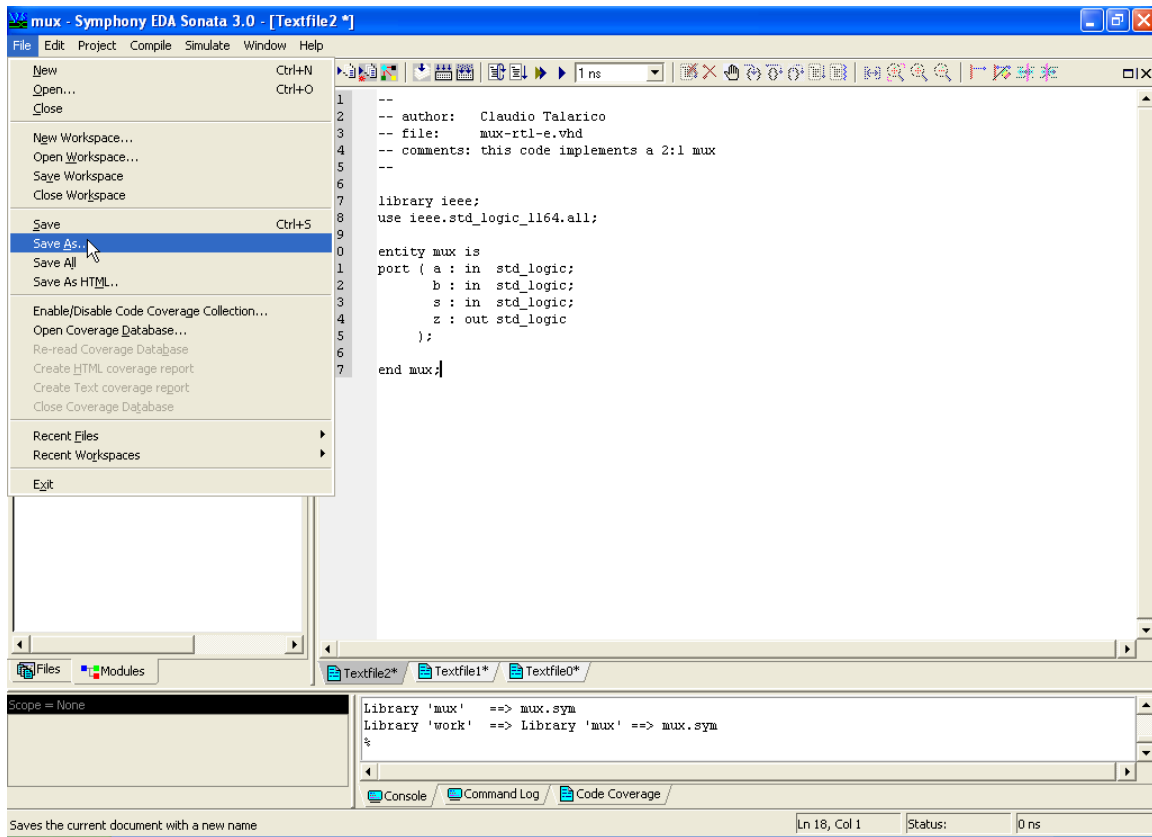
Step 3.

Create a directory (folder) to store your VHDL source files:
(D:/EWU/Tech425/VHDL-projects/MUX)

Step 4.

create the vhdl source files and save them
(mux-rtl-e.vhdl, mux-rtl-a.vhdl, mux-rtl-c.vhd, mux-tb-e.vhdl, mux-tb-a.vhdl, mux-tb-c.vhdl)





Do the same for the 5 remaining source files:

```
--  
-- author:   Claudio Talarico  
-- file:    mux-rtl-a.vhdl  
-- comments: this code implements a 2:1 mux  
--
```

```
architecture rtl of mux is  
begin
```

```
    mux_p: process(a,b,s)  
    begin  
        if(s='0') then  
            z <= a;  
        else  
            z <= b;  
        end if;  
    end process mux_p;
```

```
end rtl;
```

```
--  
-- author:   Claudio Talarico  
-- file:    mux-rtl-c.vhdl  
-- comments: configuration for 2:1 mux  
--
```

```
configuration mux_rtl_c of mux is  
for rtl  
end for;  
end configuration mux_c;
```

```
--  
-- author:   Claudio Talarico  
-- file:    mux-tb-e.vhdl  
-- comments: test-bench for 2:1 mux  
--
```

```
library ieee;  
use ieee.std_logic_1164.all;  
use std.textio.all;
```

```
entity mux_tb is  
end mux_tb;
```

```
--
-- author:   Claudio Talarico
-- file:    mux-tb-a.vhdl
-- comments: poor example of tb for 2:1 mux
--

architecture beh of mux_tb is
  -- signal and variable declaration
  signal a : std_logic;
  signal b : std_logic;
  signal s : std_logic;
  signal z : std_logic;

  component mux
    port ( a : in  std_logic;
          b : in  std_logic;
          s : in  std_logic;
          z : out std_logic);
  end component mux;

begin
  mux_instance: mux
  port map (
    a => a,
    b => b,
    s => s,
    z => z );

  tb: process
  begin

    a <= '0'; b <= '0'; s <= '0';
    wait for 10 ns;

    a <= '0'; b <= '0'; s <= '1';
    wait for 10 ns;

    a <= '0'; b <= '1'; s <= '0';
    wait for 10 ns;

    a <= '0'; b <= '1'; s <= '1';
    wait for 10 ns;

    -- Necessary for the simulator to advance
    wait for 10 ns;

    assert false
    report "End of TestBench"
    severity error;

  end process tb;

end architecture beh;
```

```

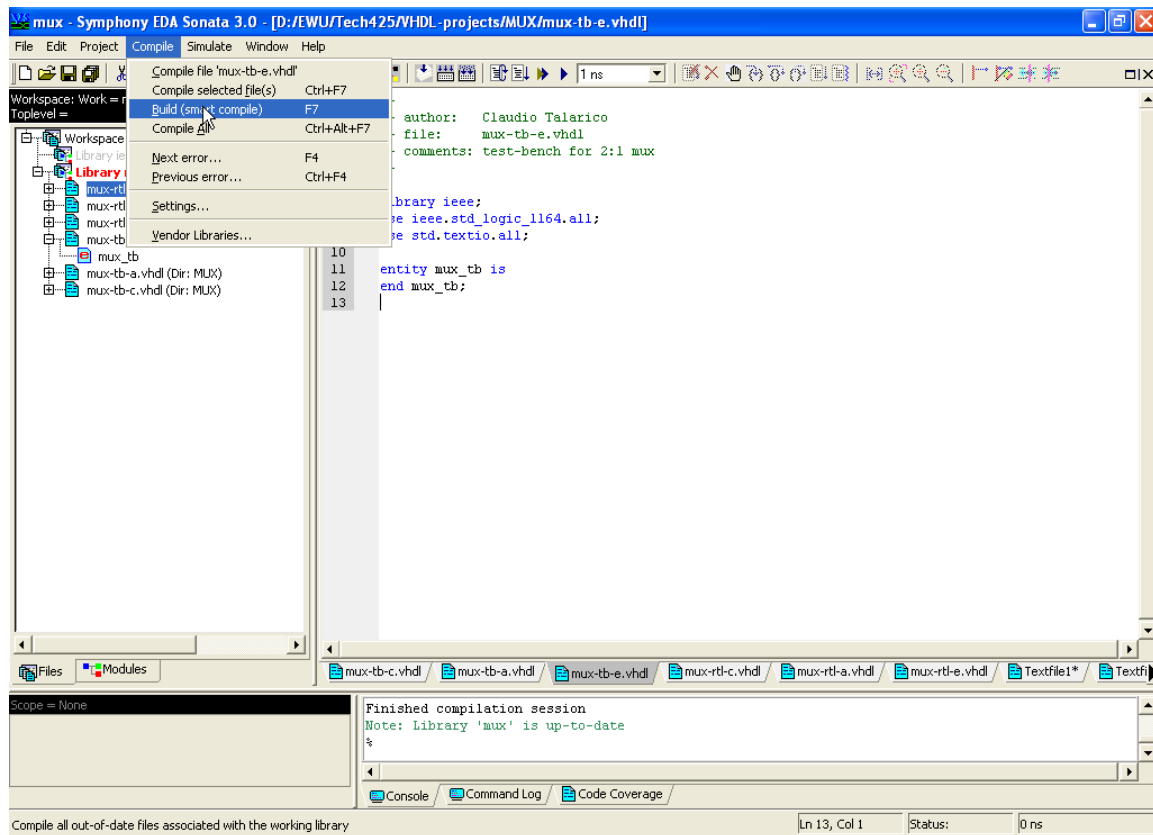
--
-- author:   Claudio Talarico
-- file:    mux-tb-c.vhdl
-- comments: test-bench's configuration for 2:1 mux
--
configuration mux_tb_c of mux_tb is
for beh
end for;
end configuration mux_tb_c;

```

Step 5.

Compile all source files. The compilation order is: mux-rtl-e.vhdl, mux-rtl-a.vhdl, mux-rtl-c.vhd, mux-tb-e.vhdl, mux-tb-a.vhdl, mux-tb-c.vhdl.

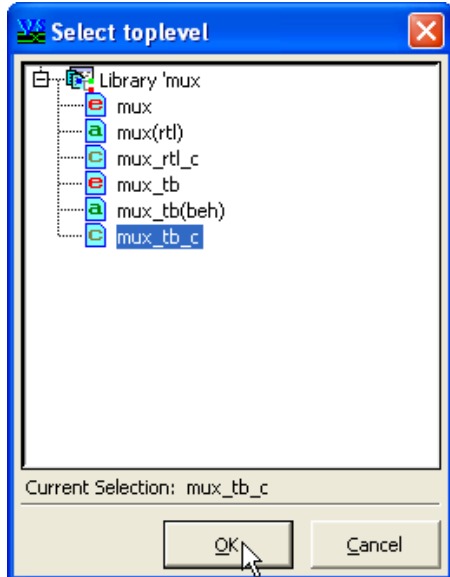
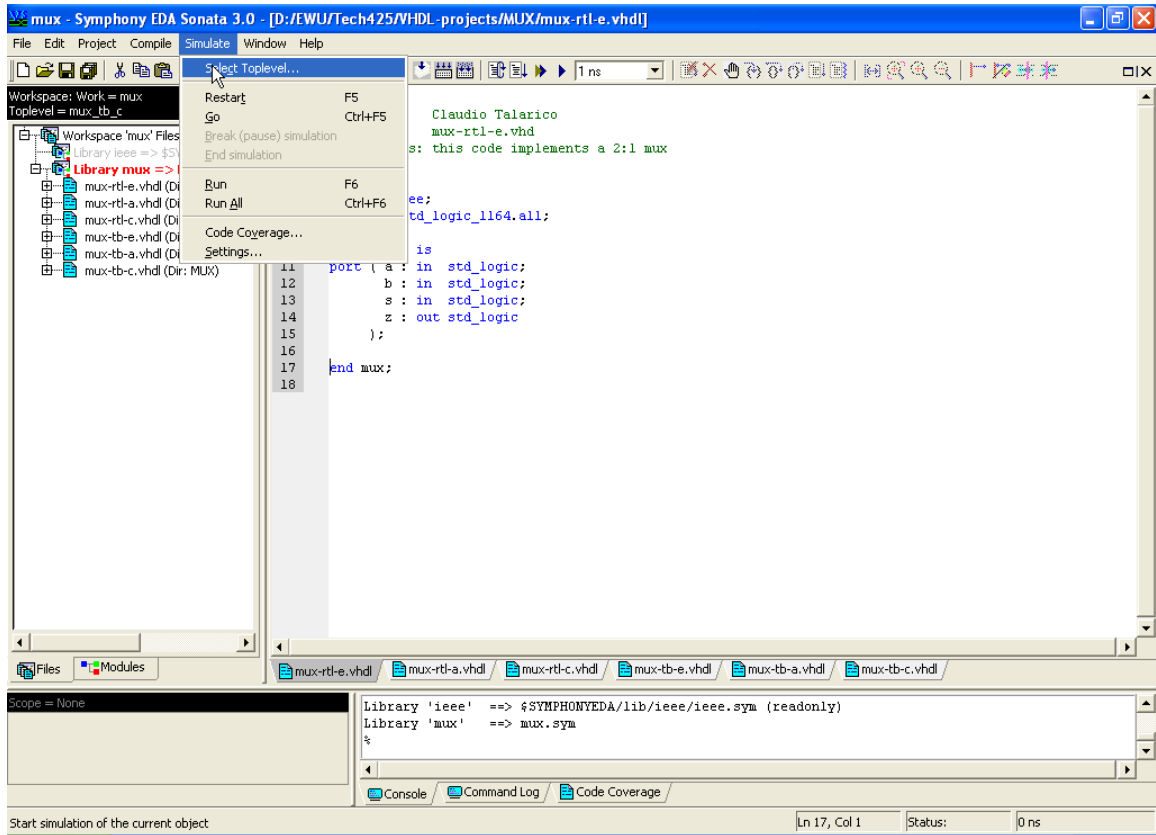
You can let the tool figuring out the compilation order as follow:



Step 6.

Simulate the mux to see if its behavior is correct.

1. You need to specify the top level of the "project". Select the testbench's configuration.



2. Start the simulation

The screenshot shows the Symphony EDA Sonata 3.0 simulation environment. The 'Simulate' menu is open, displaying the following options:

- Select Toplevel...
- Restart (F5)
- Go (Ctrl+F5)
- Break (pause) simulation
- End simulation
- Run (F6)
- Run All (Ctrl+F6)
- Code Coverage...
- Settings...

The 'Values' window is currently empty. The console window displays the following output:

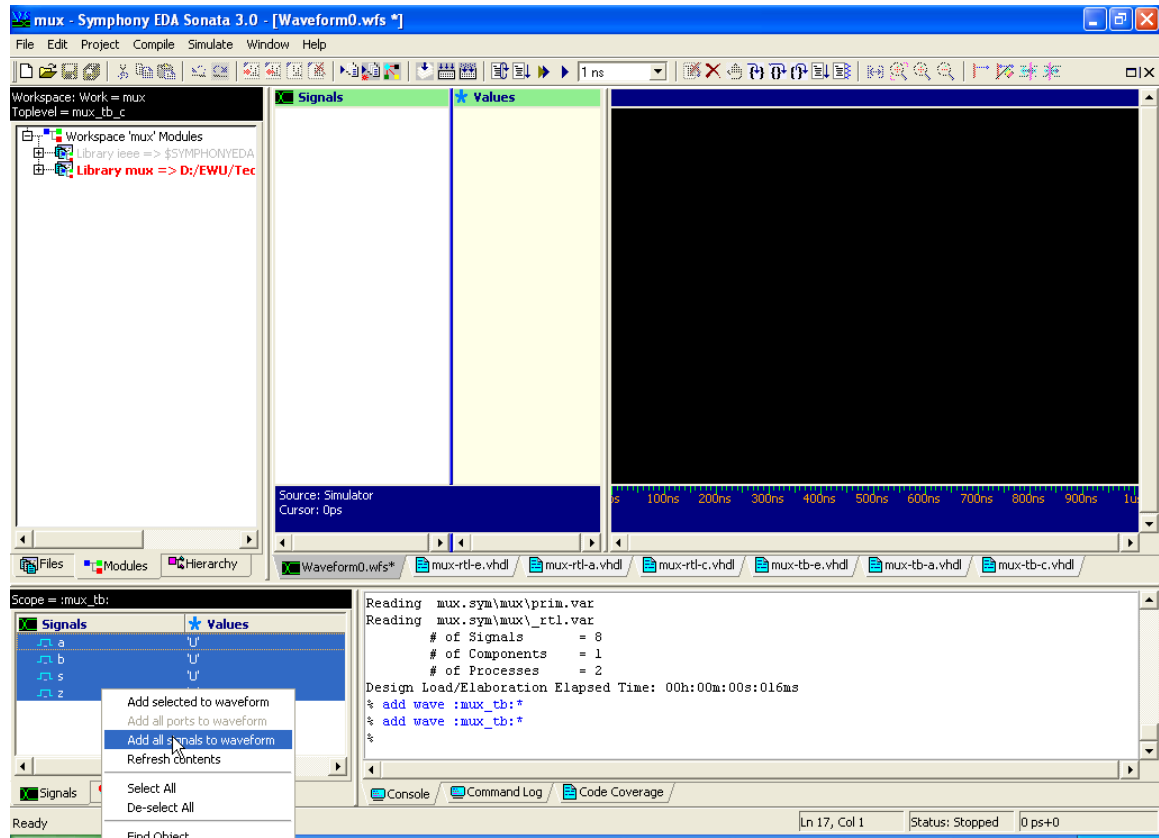
```

Reading mux.sym\mux_tb_beh.var
Reading $SYMPHONYEDA\lib\ieee\ieee.sym\std_logic_1164_body.var
Reading mux.sym\mux_prim.var
Reading mux.sym\mux_rtl.var
# of Signals      = 8
# of Components   = 1
# of Processes    = 2
Design Load/Elaboration Elapsed Time: 00h:00m:00s:016ms
%

```

The status bar at the bottom indicates: Build current lib and restart simulation | Ln 17, Col 1 | Status: Stopped | 0 ps+0

3. Add all signals/variables you'd like to observe to the waveform viewer



4. Go to the simulation settings:
 Simulate → Settings → Simulator
 and **apply** the following preferences:
 simulation stops on: ERROR
 resolution: 100 ps

5. Run the simulation and enjoy it ☺

