Test Benches

Last updated 1/19/21

Test Benches

These slides describe how to create testbench code

Upon completion: You should be able to write testbench code that allows visual verification of circuit operation via waveforms

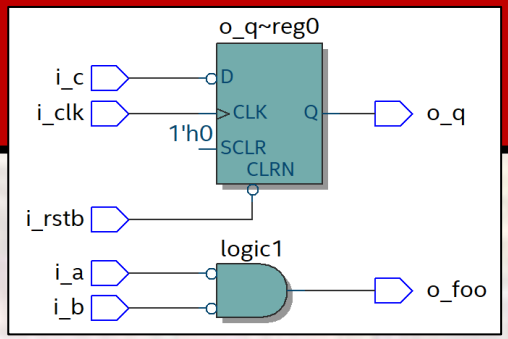
Test Benches

- Testbench
  - VHDL file used to run simulations on a block(s)
  - Testbenches have:
    - One or more blocks instantiated in it
    - Signals to drive the block's inputs
    - Signals to sense the block's outputs
    - Additional simulation related code
      - Generate input signals (with timing) to drive the block
      - Note – this makes the testbench un-synthesizable

Test Benches

- Testbench process
  - Create simulatable VHDL files (testbench) to drive the inputs of our design
    - Note: these are not synthesizable
  - Instantiate our design in the testbench file
  - Simulate the design
    - ModelSim is our simulation tool
  - View the resulting output waveforms and check for correctness

Test Benches



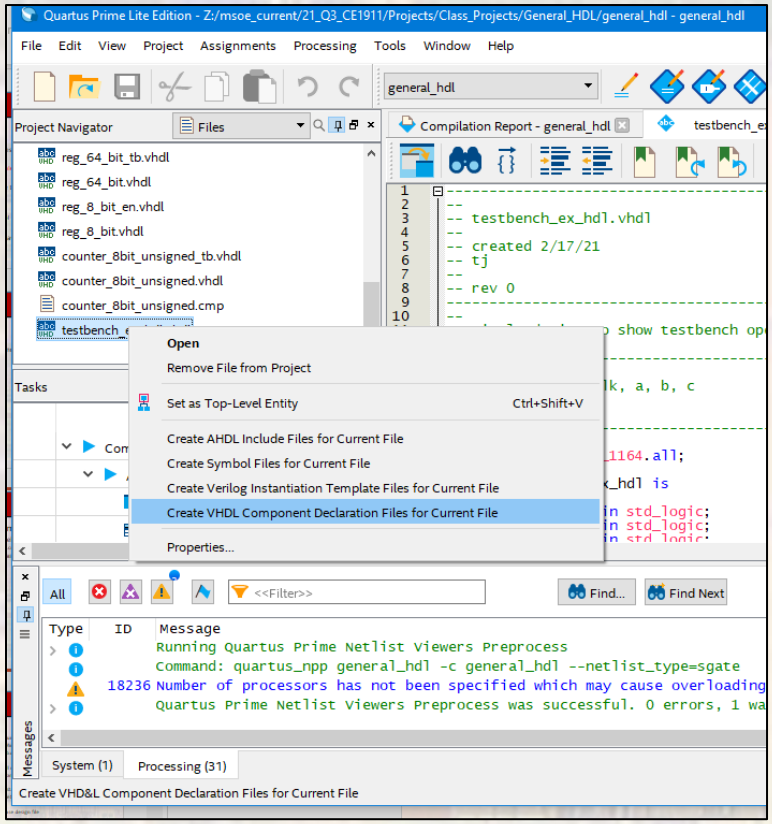
- Create Design
  - Arbitrary logic example
  - Set to top level, Elaborate and check for errors

```
----------------------------------------
--
-- testbench_ex_hdl.vhdl
--
-- created 2/17/21
-- tj
--
-- rev 0
----------------------------------------
--
-- Simple design to show testbench operation
--
----------------------------------------
--
-- Inputs: rstb, clk, a, b, c
-- Outputs: q, foo
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity testbench_ex_hdl is
   port(
         i_clk:   in std_logic;
         i_rstb:  in std_logic;
         i_a:     in std_logic;
         i_b:     in std_logic;
         i_c:     in std_logic;

         o_q:     out std_logic;
         o_foo:   out std_logic
   );
end entity;
```

```
architecture behavioral of testbench_ex_hdl is
   -- create a couple of itermediate signals
   signal one: std_logic;
   signal two: std_logic;

begin
   -- input logic processes
   logic1: process(i_a, i_b)
   begin
      if(i_a = '0' and i_b = '0') then
         one <= '1';
      else
         one <= '0';
      end if;
   end process;

   logic2: process(i_c)
   begin
      case i_c is
         when '0' => two <= '1';
         when '1' => two <= '0';
         when others => two <= '0';
      end case;
   end process;

   -- flipflop process
   process(i_clk, i_rstb)
   begin
      if(i_rstb = '0') then
         o_q <= '0';
      elsif(rising_edge(i_clk)) then
         o_q <= two;
      end if;
   end process;

   -- simple output mapping
   o_foo <= one;

end architecture;
```

Test Benches

- Create component prototype
  - Rt-click the design file and select Create VHDL Component…



.cmp file created in the project directory

Block_name.cmp → component prototype

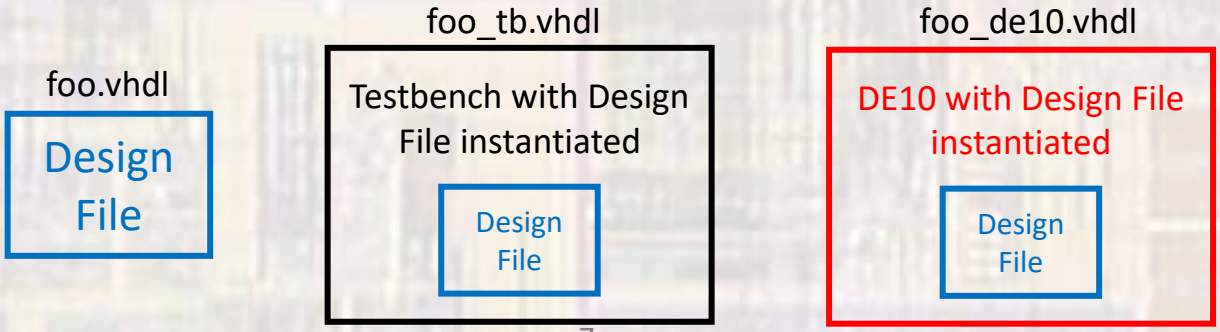
testbench_ex_hdl.cmp - Notepad

File Edit Format View Help

```
-- Copyright (C) 2018  Intel Corporation. All rights reserved.
-- Your use of Intel Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Intel Program License
-- Subscription Agreement, the Intel Quartus Prime License Agreement,
-- the Intel FPGA IP License Agreement, or other applicable license
-- agreement, including, without limitation, that your use is for
-- the sole purpose of programming logic devices manufactured by
-- Intel and sold by Intel or its authorized distributors.  Please
-- refer to the applicable agreement for further details.


-- Generated by Quartus Prime Version 18.1 (Build Build 625 09/12/2018)
-- Created on Wed Feb 17 09:45:47 2021

COMPONENT testbench_ex_hdl
        PORT
        (
                i_clk           :       IN STD_LOGIC;
                i_rstb          :       IN STD_LOGIC;
                i_a             :       IN STD_LOGIC;
                i_b             :       IN STD_LOGIC;
                i_c             :       IN STD_LOGIC;
                o_q             :       OUT STD_LOGIC;
                o_foo           :       OUT STD_LOGIC
        );
END COMPONENT;
```

Test Benches

- Create Testbench Code
  - Name your testbench the same as your design file but add _tb to the end
    - counter_8bit.vhdl → counter_8bit_tb.vhdl
    - shift_reg_nbit.vhdl → shift_reg_nbit_tb.vhdl
  - When we start using the DE10 we will do the same thing
    - Name your DE10 implementation the same as your design file but add _de10 to the end
    - counter_8bit.vhdl → counter_8bit_de10.vhdl
    - shift_reg_nbit.vhdl → shift_reg_nbit_de10.vhdl
  - In both cases we never change the base design file
    - Ensures what we designed is what we simulate, and what we simulated is what we build



Test Benches

- Create Testbench Code
  - Description and entity

```
--------------------------------
--
-- testbench_ex_hdl_tb.vhdl
--
-- created: 2/17/21
-- by: johnsontimoj
-- rev: 0
--
-- testbench for testbench setup example
-- of testbench_ex_hdl.vhdl
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity testbench_ex_hdl_tb is
   -- no entry - testbench
end entity;
```

Description

Note – entity statement required
nothing in it

testbench name

Test Benches

- Create Testbench Code
  - Testbench signals
  - Device to test

```
architecture testbench of testbench_ex_hdl_tb is
   --------------------------
   -- testbench signals to drive and sense simulation
   --------------------------
   signal CLK:     std_logic;
   signal RSTB:    std_logic;
   signal A:       std_logic;
   signal B:       std_logic;
   signal C:       std_logic;
   signal Q:       std_logic;
   signal FOO:     std_logic;

   constant PER:  time  := 20 ns;

   --------------------------
   -- Component prototype(s)
   --------------------------
   COMPONENT testbench_ex_hdl
   PORT
   (
      i_clk     :   IN STD_LOGIC;
      i_rstb    :   IN STD_LOGIC;
      i_a       :   IN STD_LOGIC;
      i_b       :   IN STD_LOGIC;
      i_c       :   IN STD_LOGIC;
      o_q       :   OUT STD_LOGIC;
      o_foo     :   OUT STD_LOGIC
   );
   END COMPONENT;

   begin

   --------------------------
   -- device under test
   --------------------------
   dut: testbench_ex_hdl
      port map(
               i_clk     => CLK,
               i_rstb    => RSTB,
               i_a       => A,
               i_b       => B,
               i_c       => C,
               o_q       => Q,
               o_foo     => FOO
      );
```

testbench is the type (can call it anything you want)

testbench name

Names used in the testbench for all inputs and outputs to your design – I use all caps and remove the o_ and i_

The period of 1 clock cycle on the DE10 (used later to make things easier)

top level name

Prototype for the component – type it in or let the tool generate it for you

Instantiation of the component – type it in (connect DUT wires to testbench wires)

DUT stands for device under test – you can label it anything you want

component signals

testbench signals

Test Benches

- Create Testbench Code
  - Test processes

```
   -- clock process
   clock: process          -- note: no sens list
   begin
      CLK <= '0';
      wait for PER/2;
      inf: loop
         CLK <= not CLK;
         wait for PER/2;
      end loop;
   end process;

   -- reset process
   reset: process          -- note: no sens list
   begin
      RSTB <= '0';
      wait for PER*2;
      RSTB <= '1';
      wait;
   end process;

   ----------------------------------------
   -- run process

   -- cycle a and c through 4 states and stop
   ac: process             -- note: no sens list
   begin
      -- initialize values
      A <= '0';
      C <= '0';
      -- wait for reset + a little bit
      wait for PER*4;
      -- cycle through values
      A <= '1';
      wait for PER*2;
      A <= '0';
      C <= '1';
      wait for PER*2;
      A <= '1';
      C <= '1';
      wait;
   end process;

   -- cycle b forever
   bonly: process          -- note: no sens list
   begin
      B <= '0';
      wait for PER*3;
      B <= '1';
      wait for PER*3;
   end process;

   ------------------------------------------------

end architecture;
```

Process for the clock signal
Runs continuously

Process for the reset_bar signal
Runs once then waits forever

Run process – cycles through a and c
then waits forever

Run process – cycles b and runs forever

Test Benches

- Create Testbench Code
  - The wait command

```
-- Clock process
clock: process          -- note: no sens list
begin
   CLK <= '0';
   wait for PER/2;
   inf: loop
      CLK <= not CLK;
      wait for PER/2;
   end loop;
end process;

-- reset process
reset: process          -- note: no sens list
begin
   RSTB <= '0';
   wait for PER*2;
   RSTB <= '1';
   wait;
end process;

-----------------------------------------
```

wait for xx causes the process block or construct to continue running (looping)

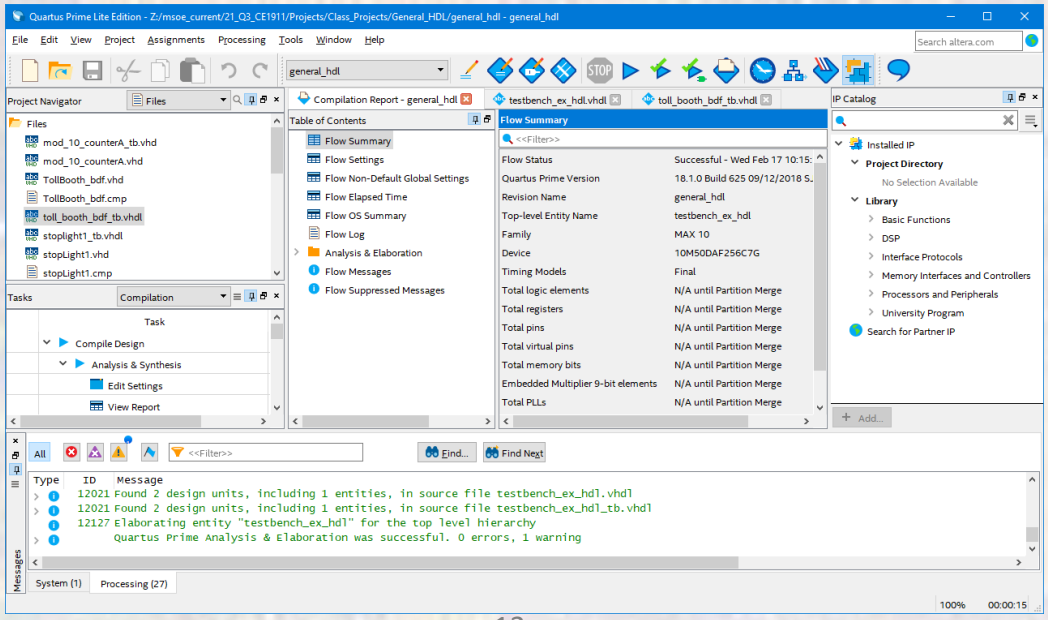
wait with no time causes the process block or construct to remain in its current state - signals no longer change

Test Benches

- Quartus Interaction and ModelSim
  - Test benches can contain un-synthesizable code
  - Quartus will generate errors on compilation if you set the testbench to the top level entity
    - Use the block under test as the top level entity

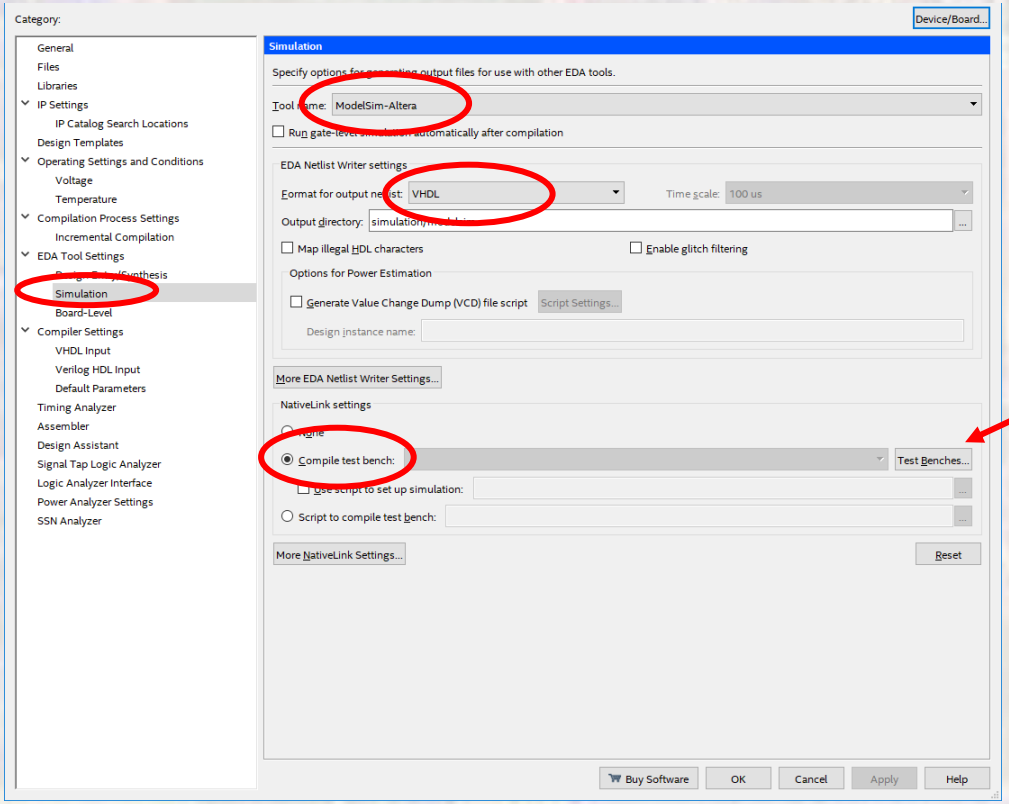
Test Benches

- Elaborate your design
  - Run Analysis and Elaboration
    - Do not do a full compile, it is a waste of time
    - Make sure your design (NOT the testbench) is selected as the top level block)
    - Checks the design and testbench for errors (checks everything in the project)
    - Creates the mathematical models of your design used in simulation



Test Benches

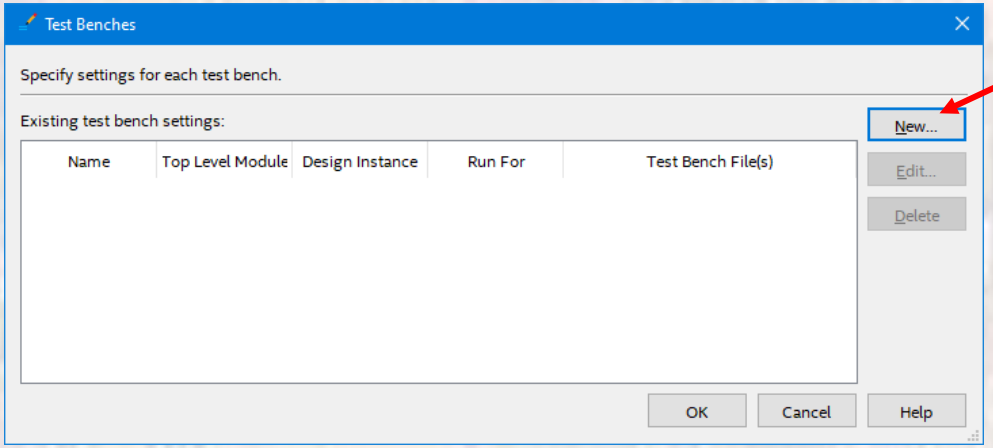
- Prepare for simulation
  - Provides ModelSim with the required information to run your testbench
  - Assignments → Settings → EDA Tool Settings → Simulation



Select Test Benches

Test Benches

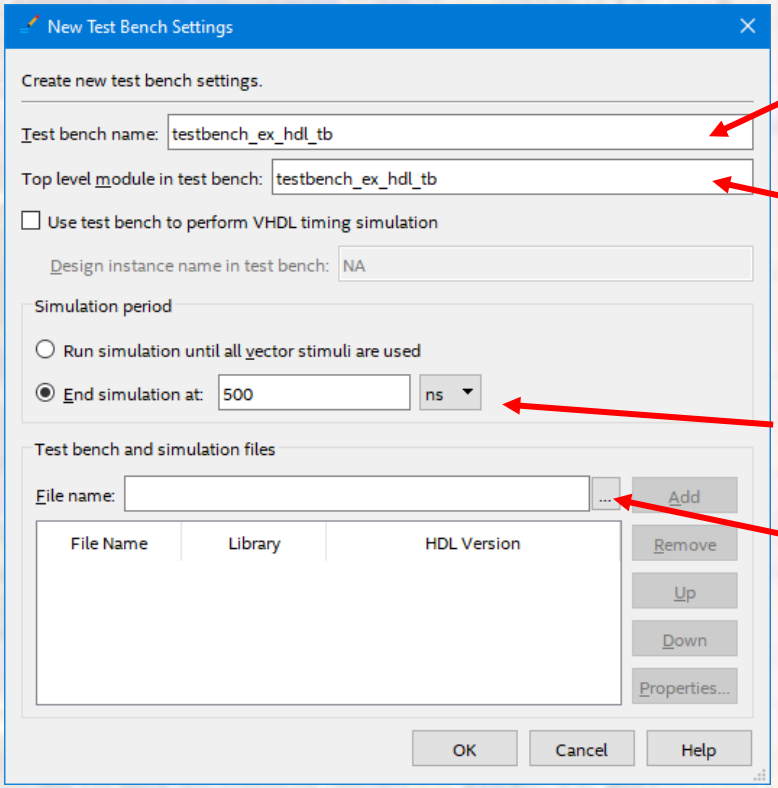
- Prepare for simulation
  - Provides ModelSim with the required information to run your testbench



Select New

Test Benches

- Prepare for simulation
  - Provides ModelSim with the required information to run your testbench



Type in name of your testbench file
no .vhdl extension

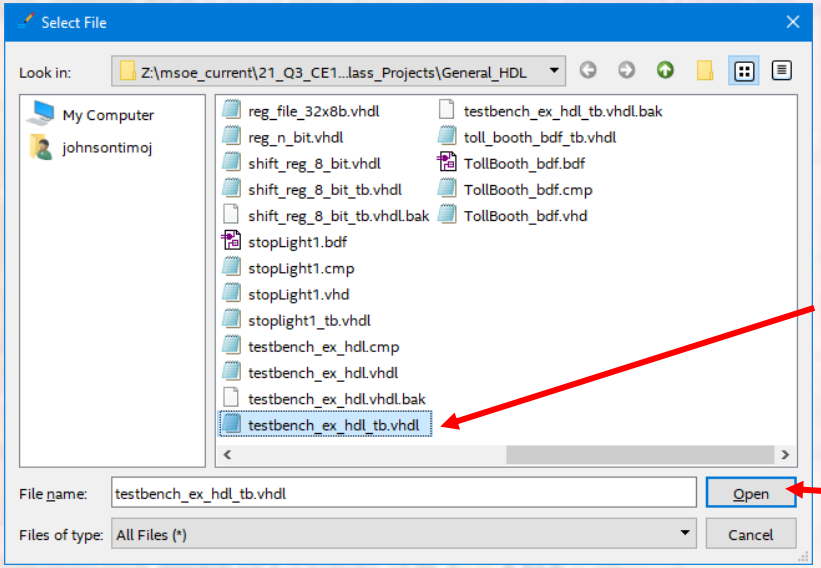
Automatically filled in

Select End simulation at button
Enter how long you want the simulation to run

Select ...

Test Benches

- Prepare for simulation
  - Provides ModelSim with the required information to run your testbench



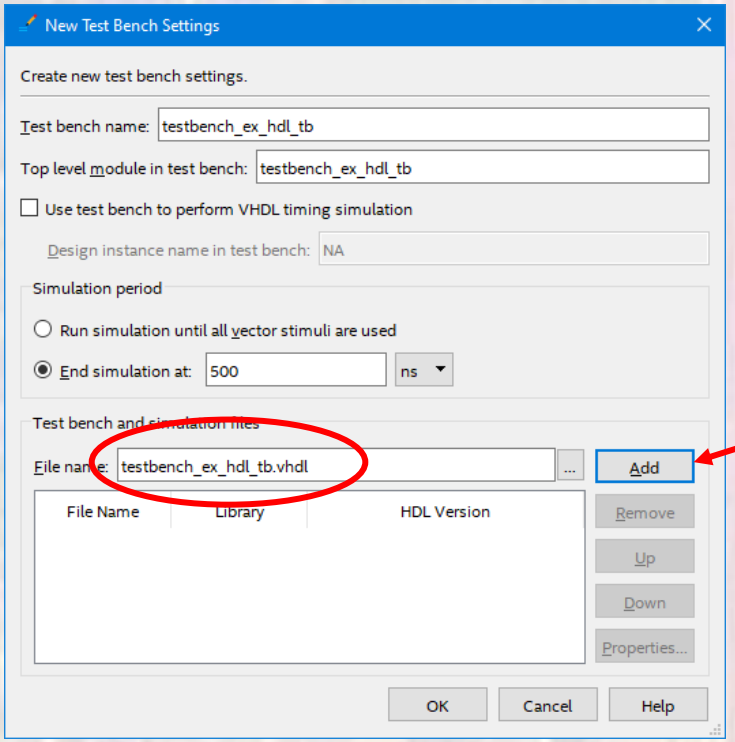
Select your testbench file

Be careful not to get the .bak version

Select Open

Test Benches

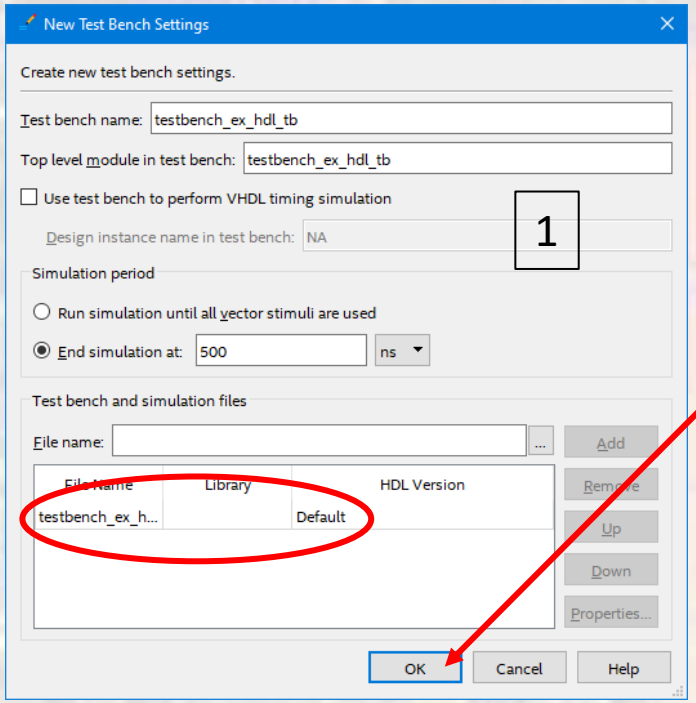
- Prepare for simulation
  - Provides ModelSim with the required information to run your testbench

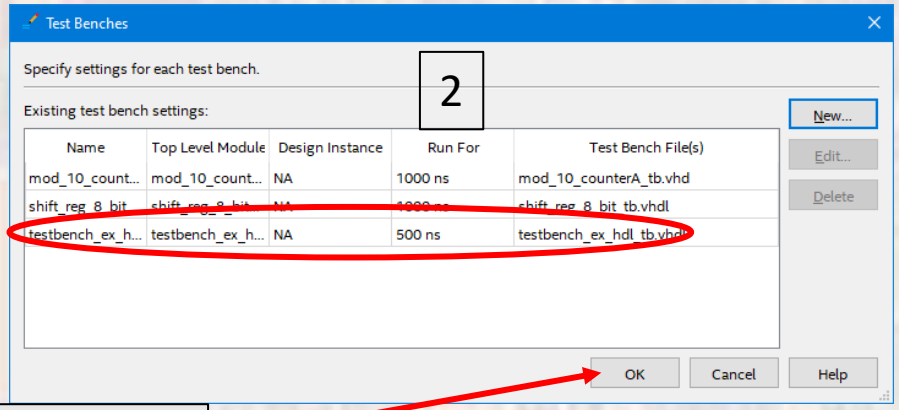


Select Add

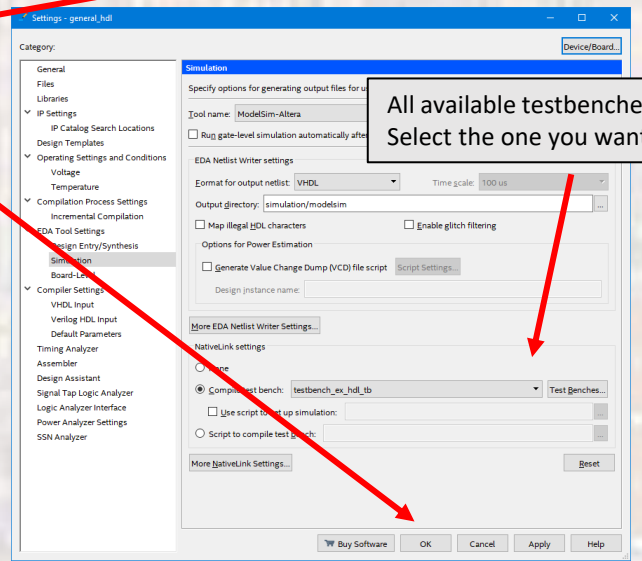
Test Benches

- Prepare for simulation
  - Provides ModelSim with the required information to run your testbench





Select OK

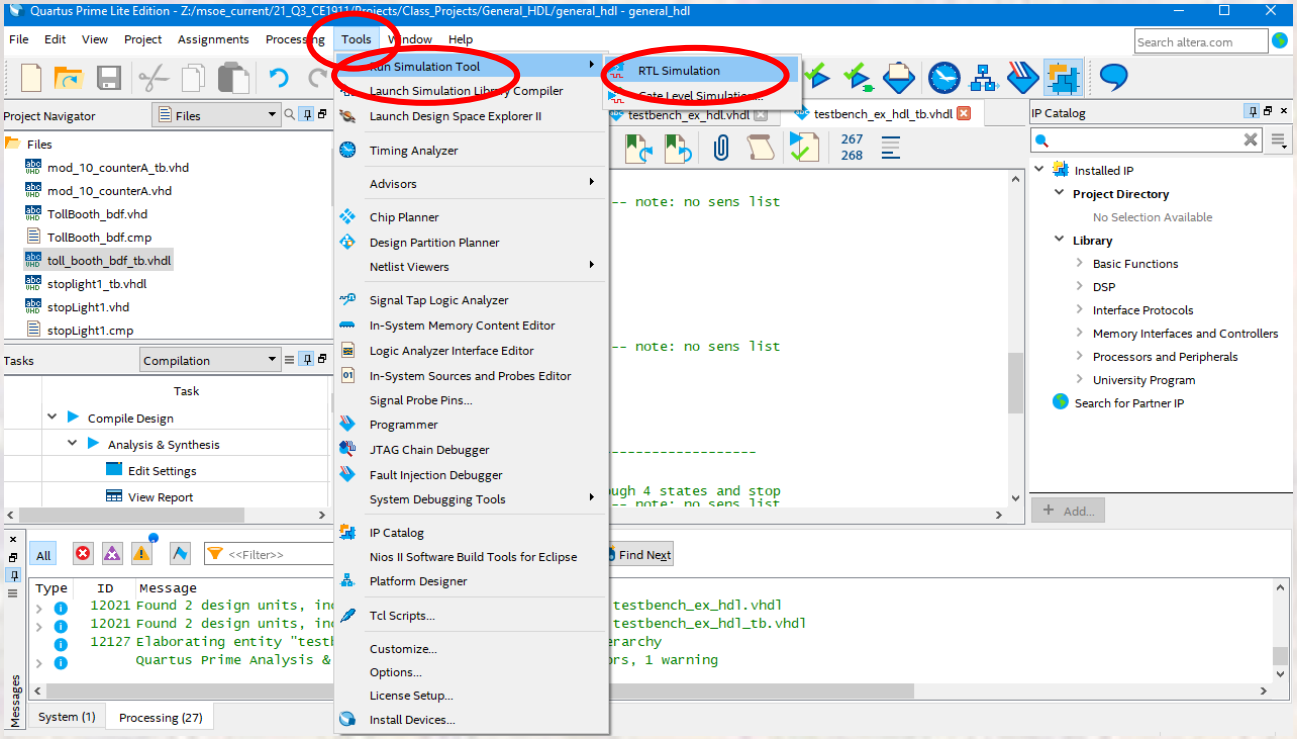


All available testbenches are here
Select the one you want to run

3

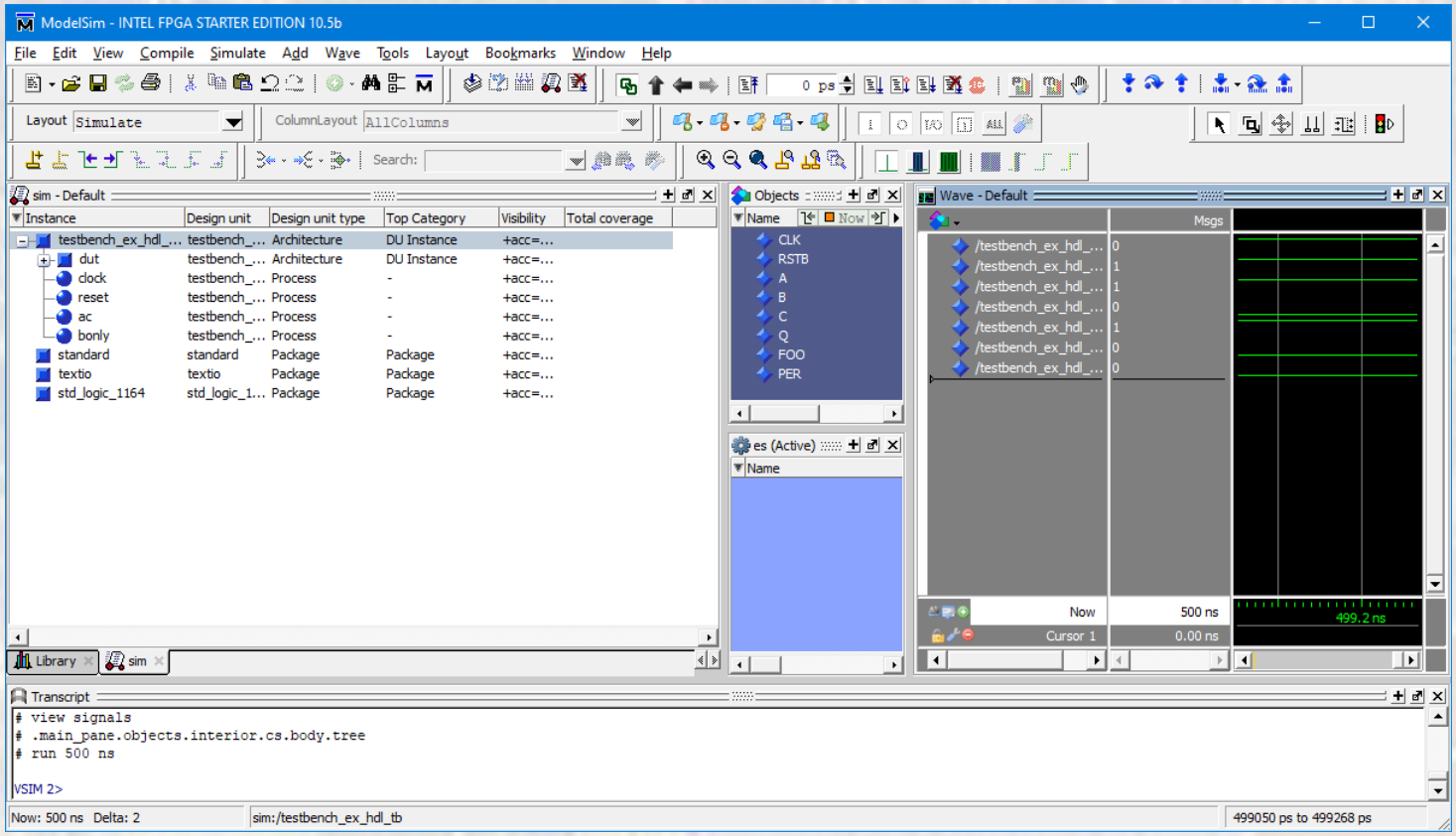
Test Benches

- Start ModelSim
  - Tools → Run Simulation Tool → RTL Simulation



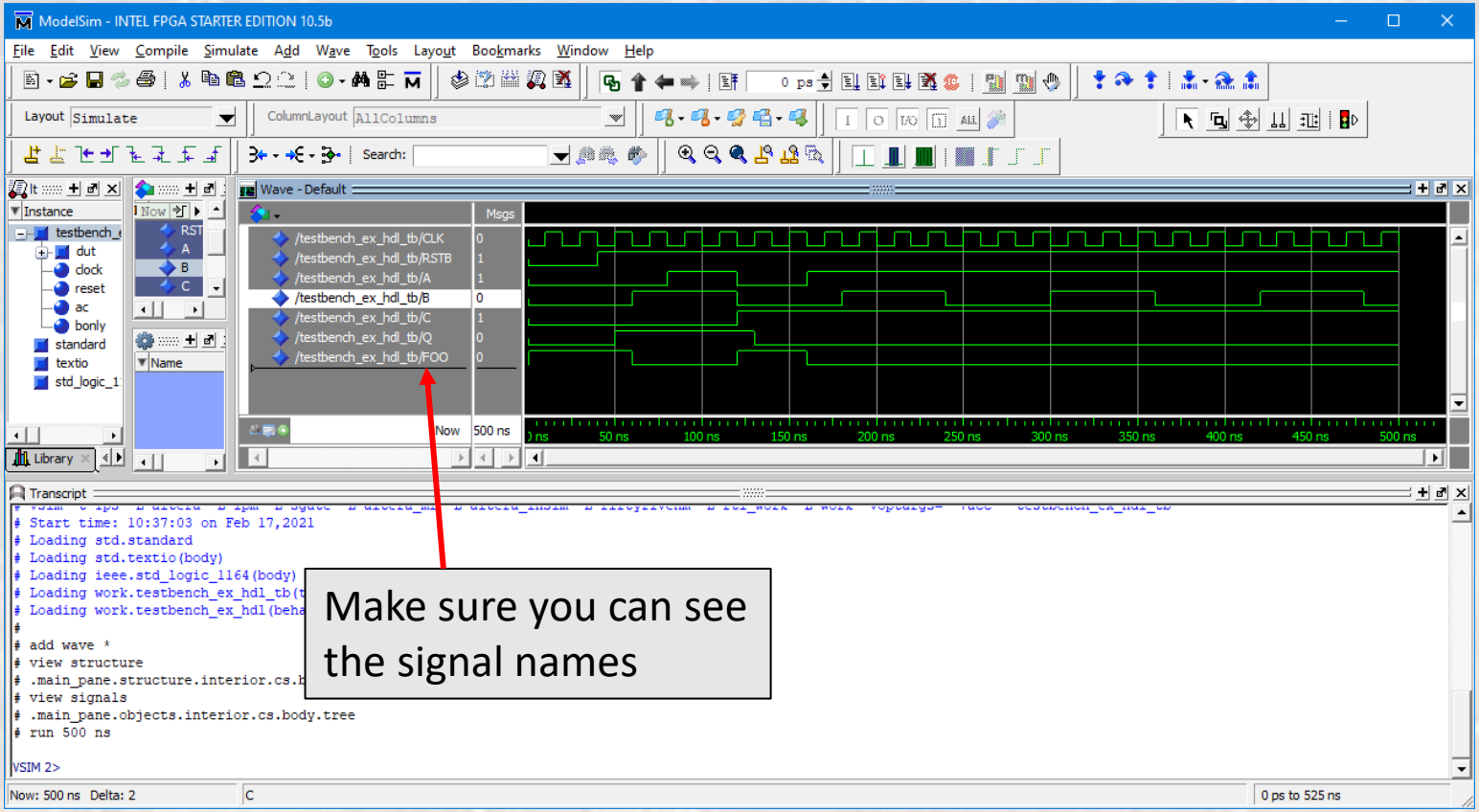
Test Benches

- Start ModelSim
  - ModelSim will start in a new window
    - Often is starts minimized – so you need to select it from the toolbar



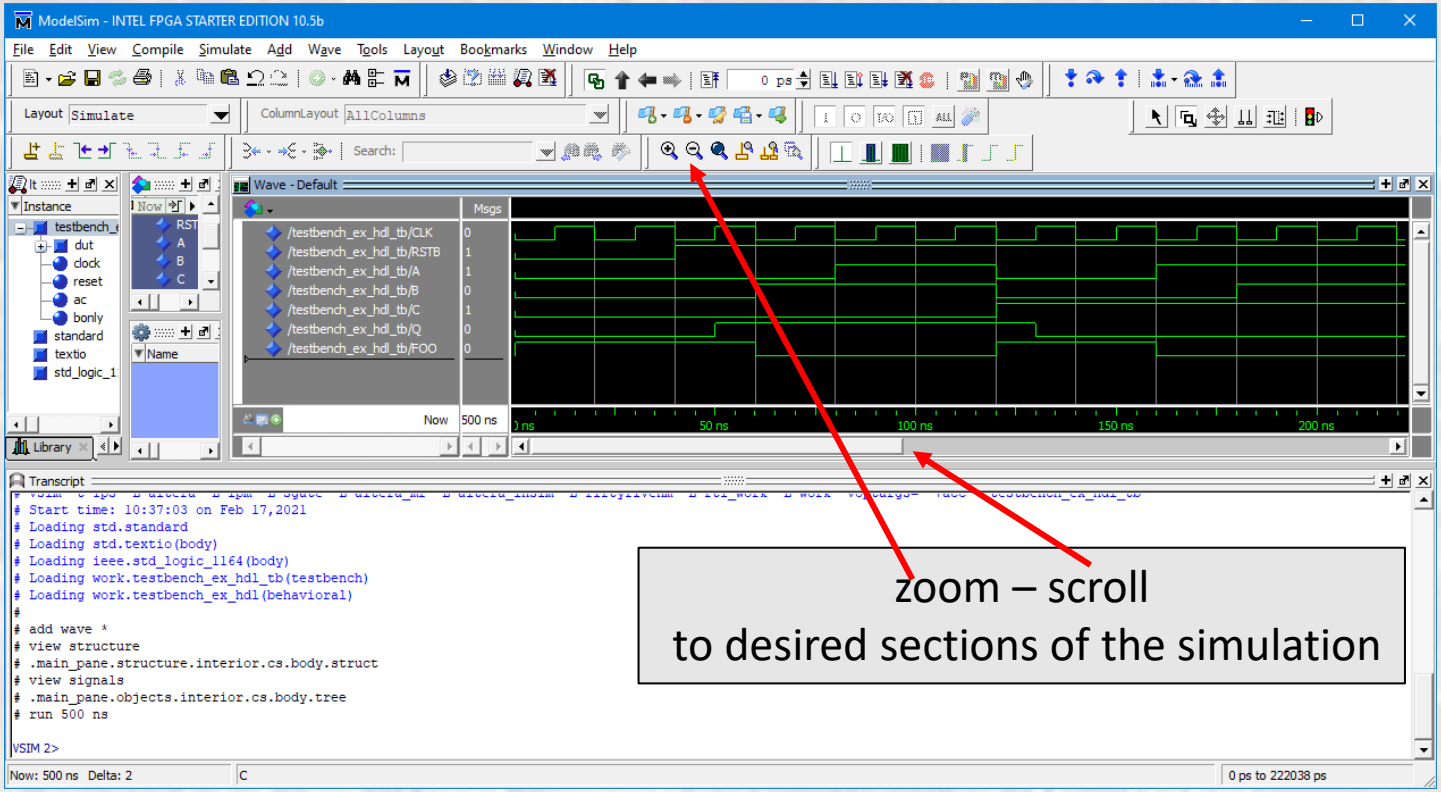
Test Benches

- Simulation Results
  - Resize windows as desired
  - Rt-click and select Zoom Full on the waveform window



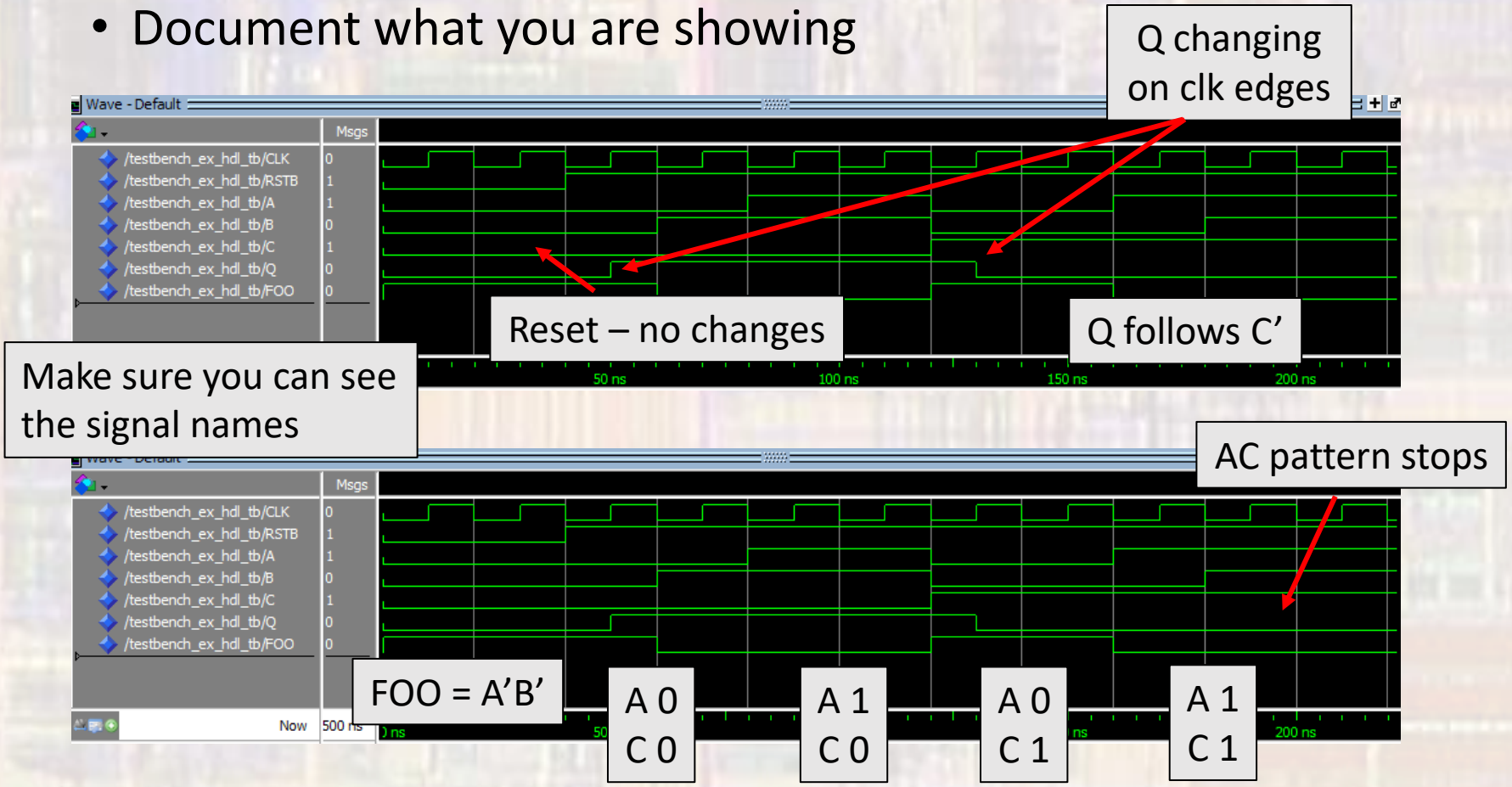
Test Benches

- Simulation Results
  - Move around in the simulation window



Test Benches

- Verify Design
  - Check critical sections to prove design works
  - Document what you are showing



Test Benches

- Run Process Examples

Up/dn counter – DIR changes the direction

```
-- Run Process
run: process       -- no sensitivity list
begin
   -- initialize inputs
   DIR <= '0';
   -- run code
   wait for 300*PER;
   DIR <= '1';
   wait;
end process run;
```

Enough clock cycles to make sure it wraps then - changes direction

Note: it then counts down forever

Test Benches

- Run Process Examples

8 bit shift register (L/R based on DIR) (shift/no-shift based on SHIFT)

```
run: process      -- no sensitivity list
   begin

      -- Initialize  inputs
      D <= '0';
      SHIFT <= '0';
      DIR <= '0';

      wait for 2*PER;    -- wait for reset

      -- verify no shift
      D <= '1'; wait for PER;
      D <= '0'; wait for PER;

      -- verify shift lt
      SHIFT <= '1';
      D <= '1'; wait for 8*PER;
      D <= '0'; wait for 8*PER;

      -- verify shift rt
      DIR <= '1';
      D <= '1'; wait for 8*PER;
      D <= '0'; wait for 8*PER;
end process run;

-------------------------------
-- End test processes
```

Initialize the input signals

Wait for reset to complete

No-shift

Shift 1 way – enough times to test all bits

Shift the other way – enough times to test all bits

No ‘wait’ at the end – repeats the whole process

Test Benches

- Run Process Examples

Multiple RUN process – used for independent signals

```
-- Run Processes
n_s: process
begin
    TNS <= '0';
    wait for 10*PER;
    TNS <= '1';
    wait for 15*PER;
    TNS <= '0';
    wait for 15*PER;
end process n_s;

ew: process
begin
    TEW <= '0';
    wait for 13*PER;
    TEW <= '1';
    wait for 7*PER;
end process ew;

-----------------------------------
-- End test processes
-----------------------------------
```

Independent N/S signal

Independent E/W signal