Digital Design

Verification

Fenerbahce University

Content

- Verification
  - Verilog Based Testbench
  - ISIM Simulation Tool

Verification Approaches

- Implemented RTL design with Verilog /VHDL.
- No errors in syntax, bitstream generated, but...
- What needs to be done to make sure the design works correctly?
  - Configure and try method.
    - Lots of trials
    - Observing problems may not be easy.
    - It is very costly in terms of time.

Verification Approaches

- In modern digital system verification approaches, functional verification is performed.

- The sum of the effort spent on completing a design
  - 30% by design
  - 70% verification processes

Verification Approaches

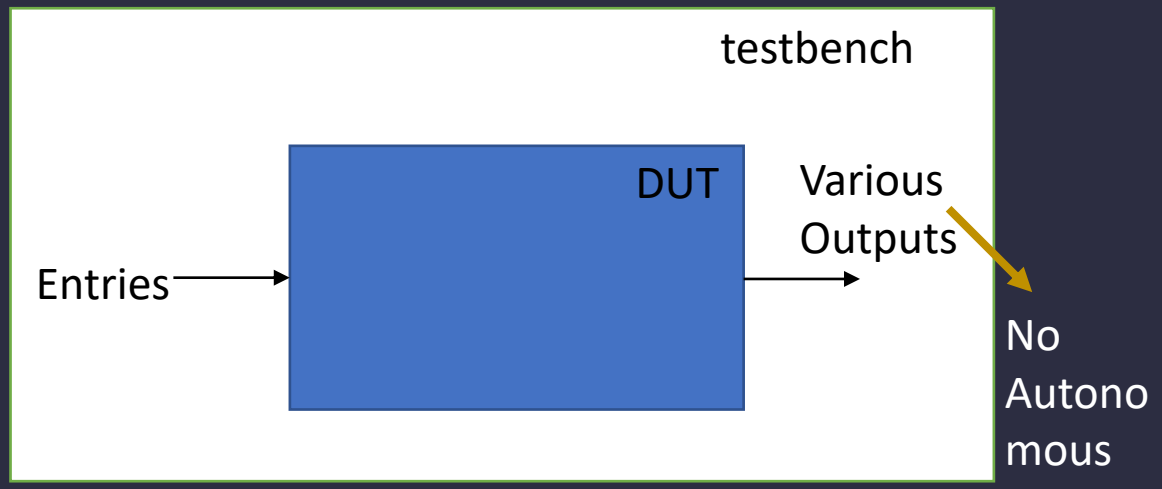
- Design to FPGA Before configuring, we verify it with simulation tools on the computer.
- Simulation to FPGA Advantages over configure and try
  - Very fast retry possibility when changes are made to the design
  - all signals in the design with cycle sensitivity

Verification Approaches

- Simulate the module after design complated.
- By coding Verilog / VHDL, a module is written that produces inputs and controls the outputs for the module to be tested.
- However, test modules do not have to be synthesizable. In other words , some structures that cannot be synthesized in verilog such as for , while .. can be used in testbench codes.

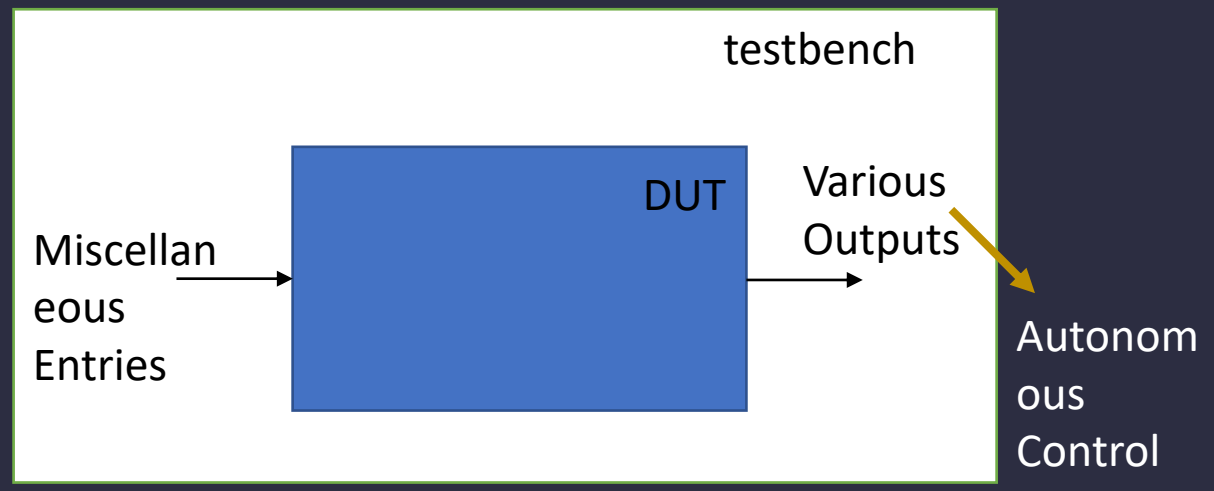
Verification Approaches

- There are 3 different type testbench
  - Simple Testbench : Inputs are fed to the input of the module, the outputs produced by the module are examined by the designer and it is decided whether it works correctly or not.
  - Instantiate name of tested modules usually DUT (Design Under Test) is chosen



Verification Approaches

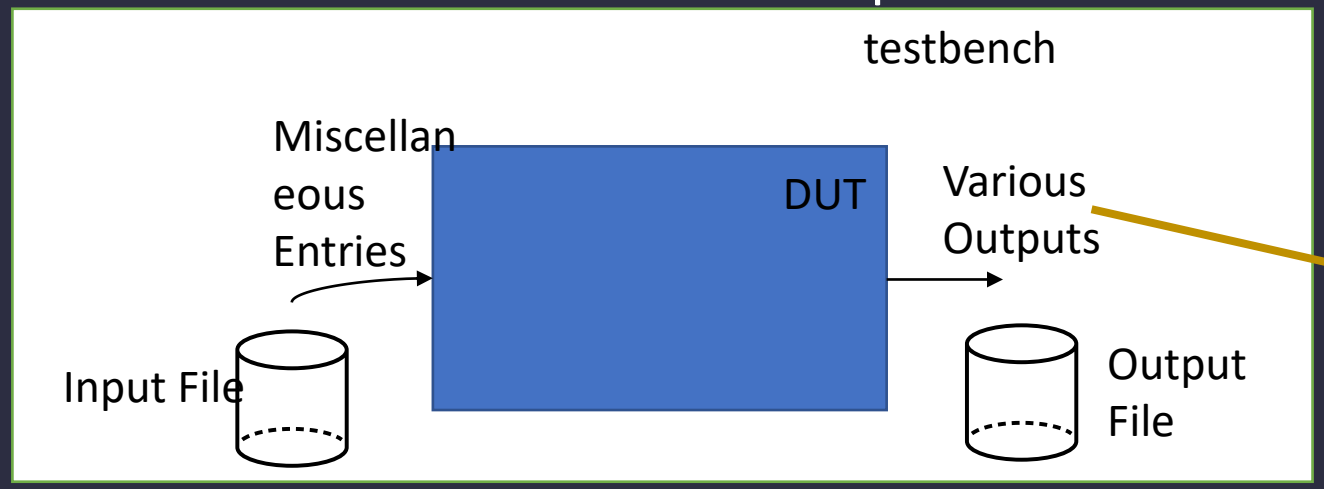
- There are 3 different type testbench
  - Self checking : According to the entries given in these testbenches , the signals produced by the tested module are not observed manually by the designer. Testbench is designed to automatically check whether the signals output by the tested module are true or false.



Verification Approaches

- There are 3 different type testbench

  - Self checking with test vectors with vectors: In this testbench , the inputs to be given from the module to be tested and the outputs expected to be produced by the module are prepared in a file beforehand. Testbench reads from this file at the desired times, feeds input to the module and checks whether the result produced by the module is the same as the expected result.



Autono
mous
Control
Yes

Verification Approaches

Sample:

- Design $y = (b \cdot c) + (a \cdot b)$ circuit and verify it in the testbench environment.

Verification Approaches

- $y = (b \cdot c) + (a \cdot b)$

```
module exampleRTL ( input a, b, c, output reg y);
 always @(*)
y = ~b & ~c | a & ~b;
endmodule
```

Verification Approaches

exampleRTL module.

```
` timescale 1ns / 1ps

module testbench ();
   reg a, b, c;
   wire y;

   exampleRTL DUT (.a(a), .b(b), .c(c), .y(y) );

   initial begin
a = 0;
b = 0;
c = 0;
#10;
c = 1;
#10;
b = 1;
c = 0;
#10;
c = 1;
#10;
   end

endmodule
```

Verification Approaches

The initial block will only be executed once when the simulation starts.

Simple testbench approach. The inputs are given automatically, but the correct or incorrect outputs are not automatically checked.

```
` timescale 1ns / 1ps

module testbench ();
   reg a, b, c;
   wire y;

   exampleRTL DUT (.a(a), .b(b), .c(c), .y(y) );

   initial begin
a = 0;
b = 0;
c = 0;
#10;
c = 1;
#10;
b = 1;
c = 0;
#10;
c = 1;
#10;
   end

endmodule
```

Verification Approaches

self-checking testbench

- It contains control mechanisms, if there is an error, it can print an error message to warn the user.
- $ display command is used to print an information message to the screen in the simulation tool.

```
`  `timescale 1ns / 1ps

   module testbench2();
      reg a, b, c;
      wire y;

      exampleRTL berry(.a(a), .b(b), .c(c), .y(y));

      initial begin
   a = 0;
   b = 0;
   c = 0;
   #10;
         if (y !== 1)
   $ display ("1st exit incorrect .");
   c = 1;
   #10;
         if (y !== 0)
   $ display ("2nd exit incorrect .");
   b = 1;
   c = 0;
   #10;
         if (y !== 0)
   $ display ("3rd exit incorrect .");
      end
   endmodule
```

Verification Approaches

self-checking testbench

- With this approach, if there are a lot of inputs that need to be tested, it can be very difficult to manually export them.
- testbench can be prepared by reading from a file and giving it as input .

```
`  `timescale 1ns / 1ps

module testbench2();
   reg a, b, c;
   wire y;

   exampleRTL berry(.a(a), .b(b), .c(c), .y(y));

   initial begin
a = 0;
b = 0;
c = 0;
#10;
      if (y !== 1)
$ display ("1st exit incorrect .");
c = 1;
#10;
      if (y !== 0)
$ display ("2nd exit incorrect .");
b = 1;
c = 0;
#10;
      if (y !== 0)
$ display ("3rd exit incorrect .");
   end
endmodule
```

Verification Approaches

An example of an RTL with a Clock

```
`timescale 1ns / 1ps

module counter ( clk , reset, enable, count);
input clk , reset, enable;

output reg [3:0] count = 0;
reg [3:0] countNext = 0;

always @ ( posedge clk ) begin
count <= #1 countNext ;
end

always@(*) begin
      countNext = count;
if (reset == 1'b1) begin
        countNext = 0;
end else if ( enable == 1'b1) begin
        countNext = count + 1;
end
end

endmodule
```

Verification Approaches

An example of an RTL testbench with Clock

```
module counter_tb ;
reg clk , reset, enable;
wire [3:0] count;

counter U0 (
. clk ( clk ),
.reset (reset),
.enable (enable),
.count (count)
);

initial
begin
   clk = 0;
reset = 0;
enable = 0;
#10;
enable = 1;
end

always #5 clk = ! clk ;

endmodule
```

Verification Approaches

Simulation tools frequently used in industry

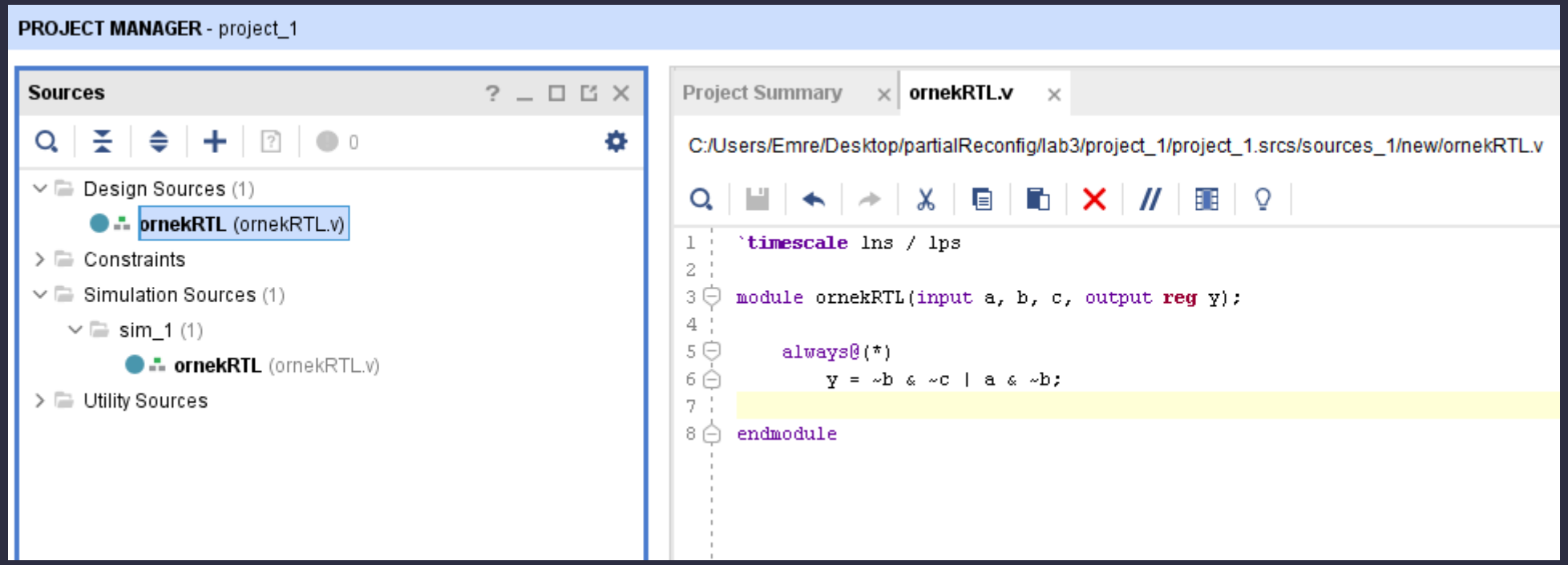
- Vivado NAME
- Modelsim / Questa (Mentor)
- VCS (Synopsys)
- Icarus Verilog (Open source)
- Verilator (Open source)

Verification Approaches

Simulation

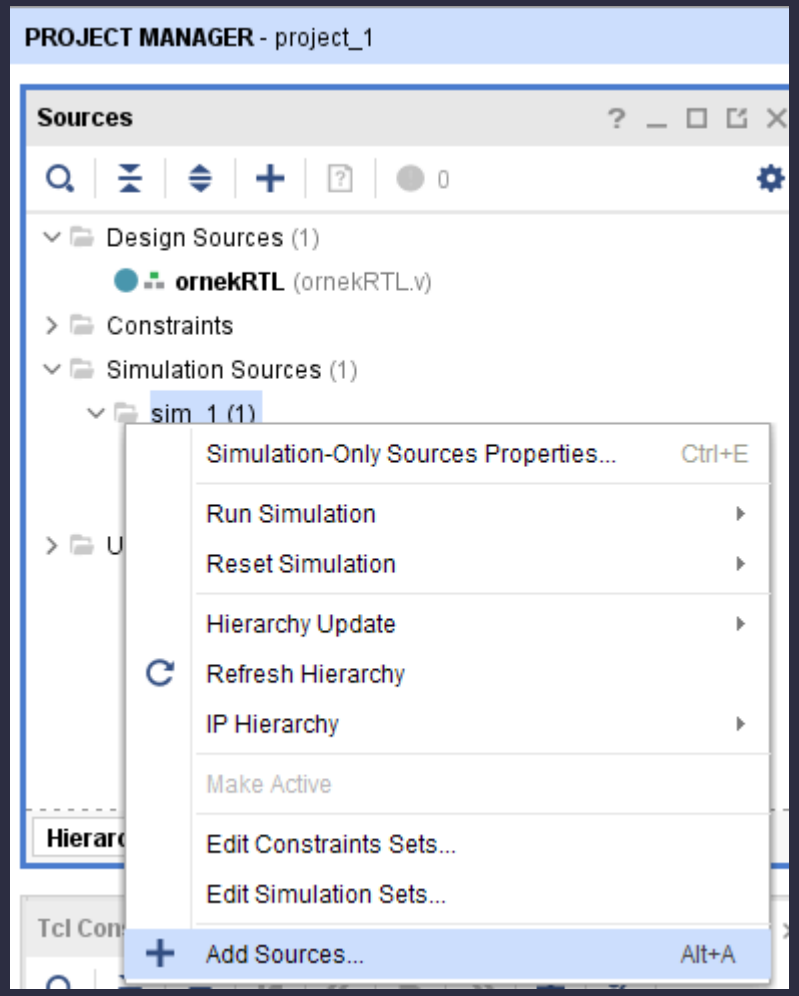
Verification Approaches

- Simulation



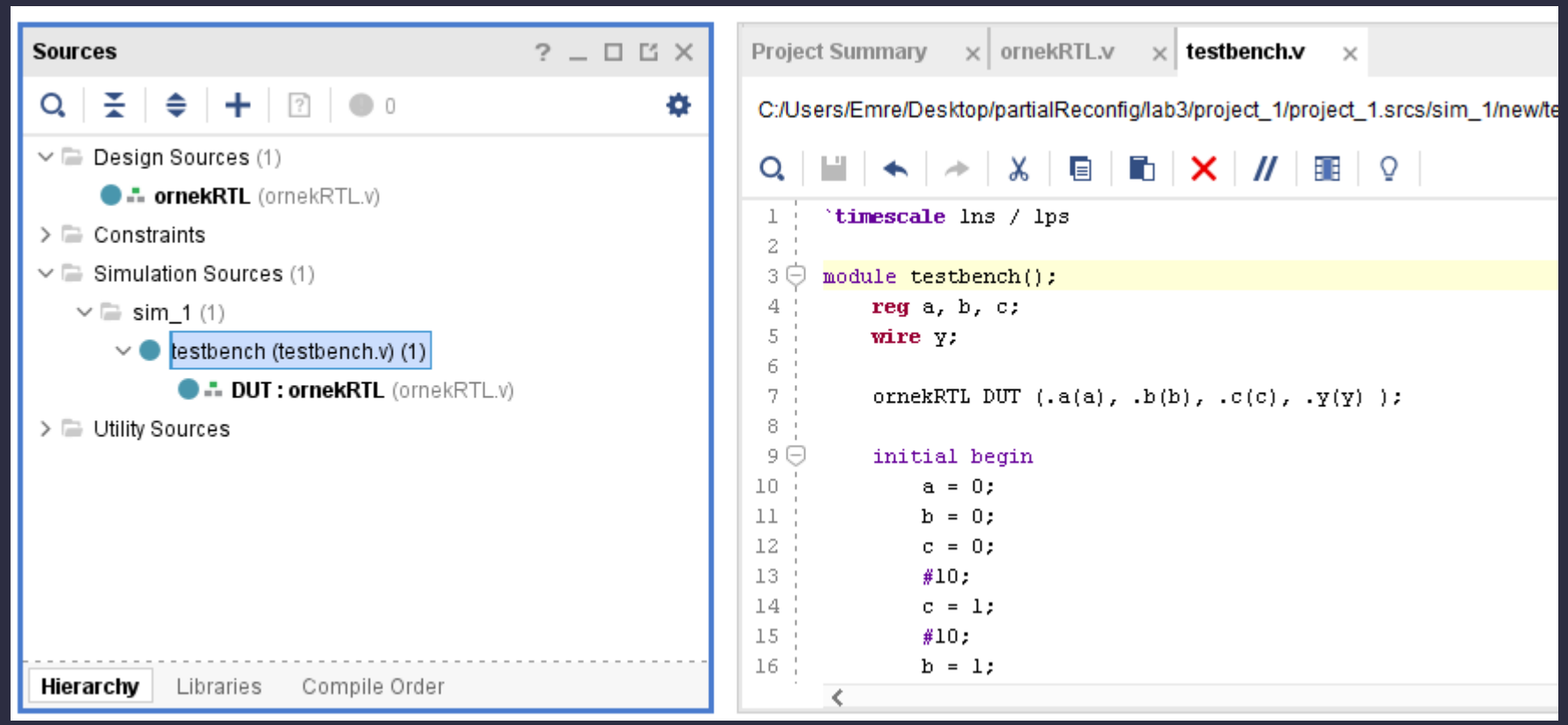
Verification Approaches

- Add a testbench file to the project in Vivado. In the Sources section, right click to simulation source. Click to Add Sources … tab. Proceed just like adding a design file and an empty testbench file is obtained. This file will created inside sim_1.



Verification Approaches

- Testbench codes are written into the new file added .



Verification Approaches

- At this stage, there is a point to be noted. Just like in design files, there is the concept of top module in simulation files. The top module of the simulation codes to be used to simulate a design must be specified in vivado .

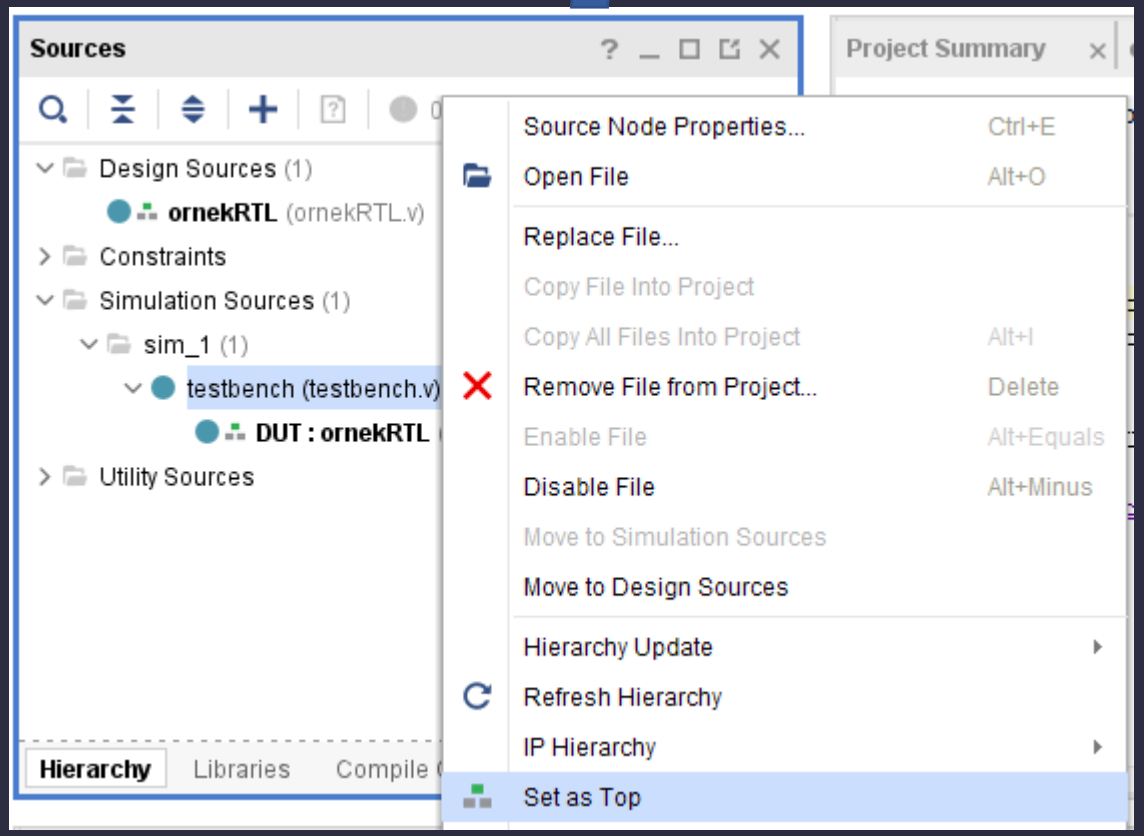
Verification Approaches

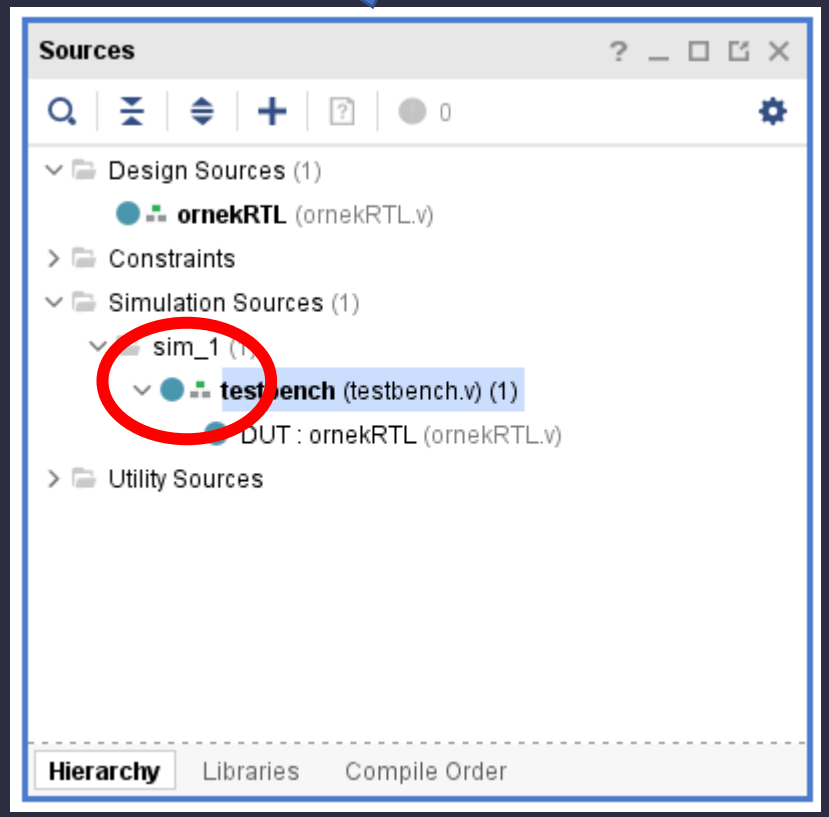
- The design file is added in Vivado, this file is added to both the design and simulation folders, and the top module is automatically determined for both cases.

- So in this case , the newly designed test module should be selected as the simulation top module.

Verification Approaches

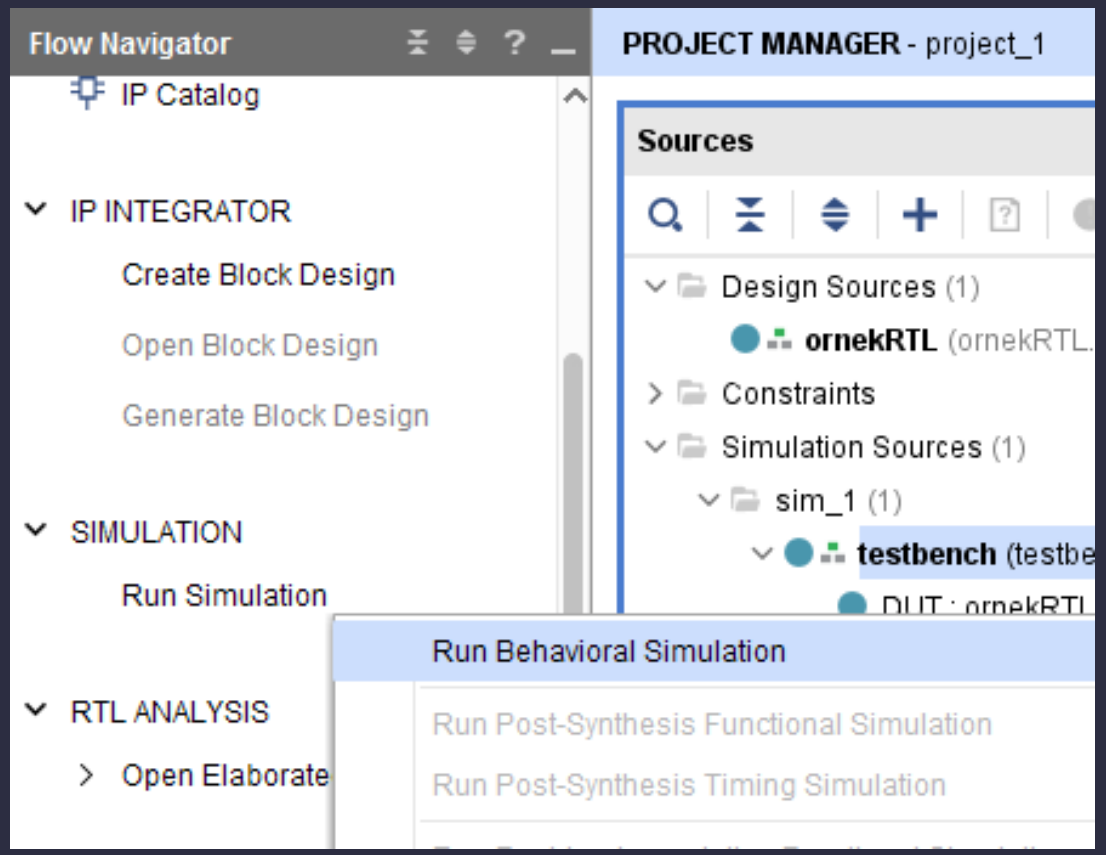
- simulation top module





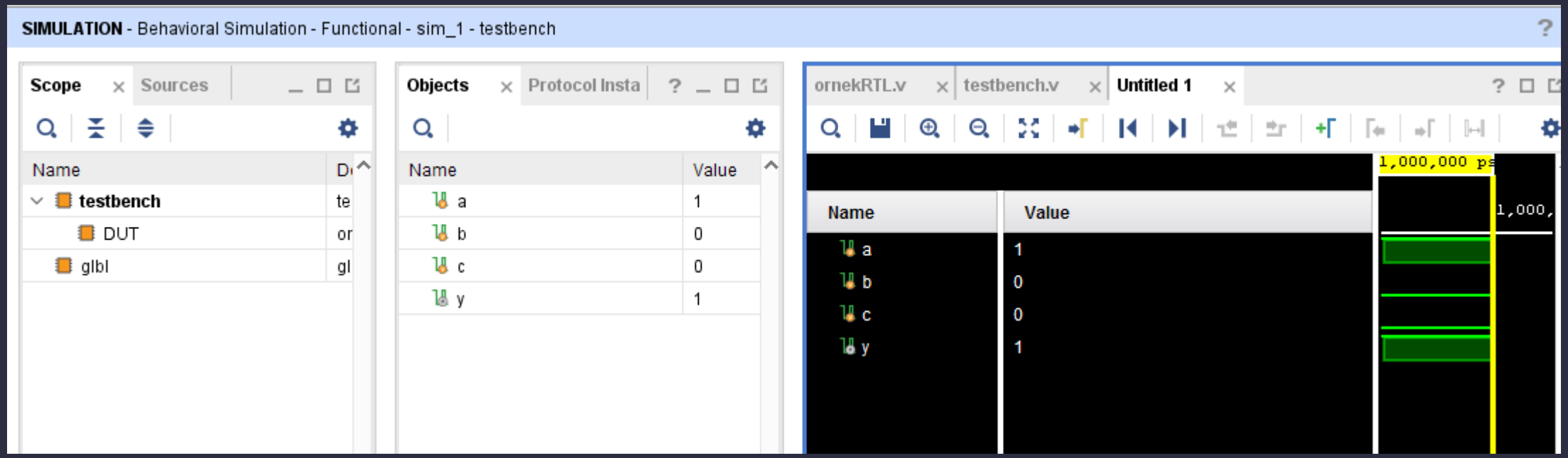
Verification Approaches

- After this step, the simulation can be started. For this, click the "Run Behavioral Simulation "

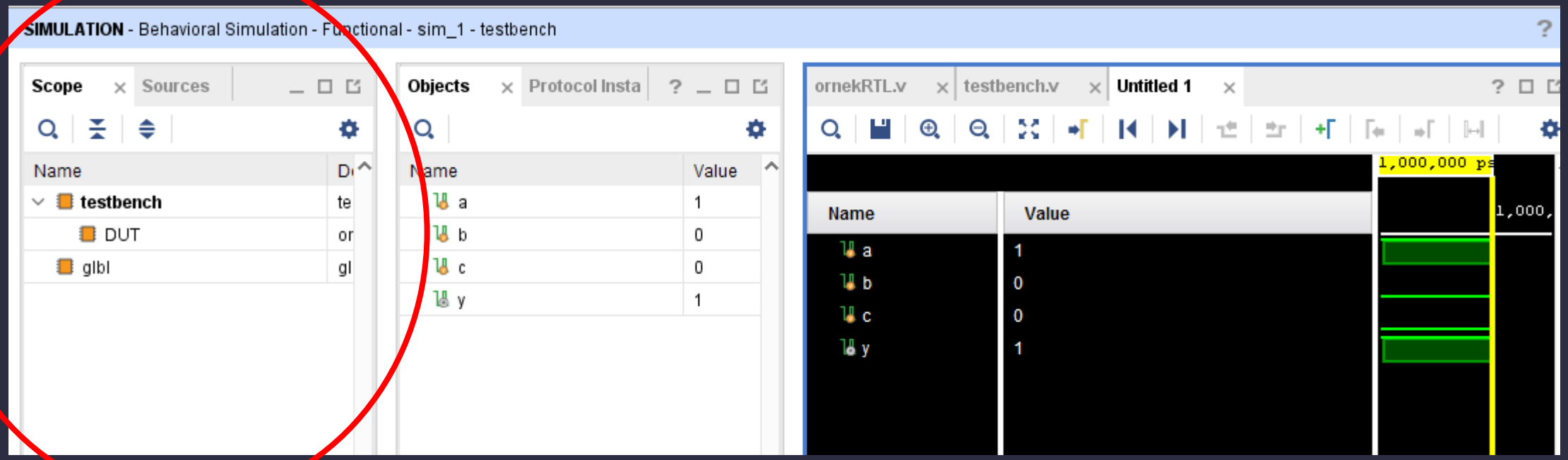


Verification Approaches

- Simulation window is given below

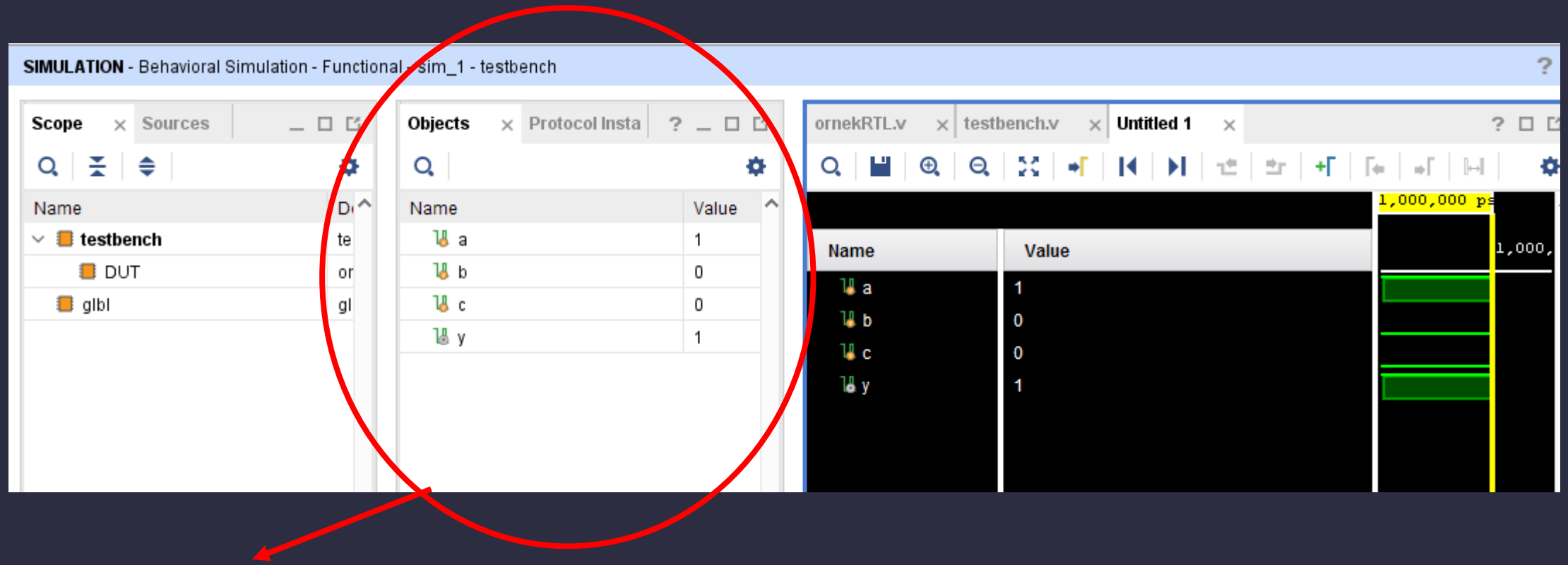


Verification Approaches



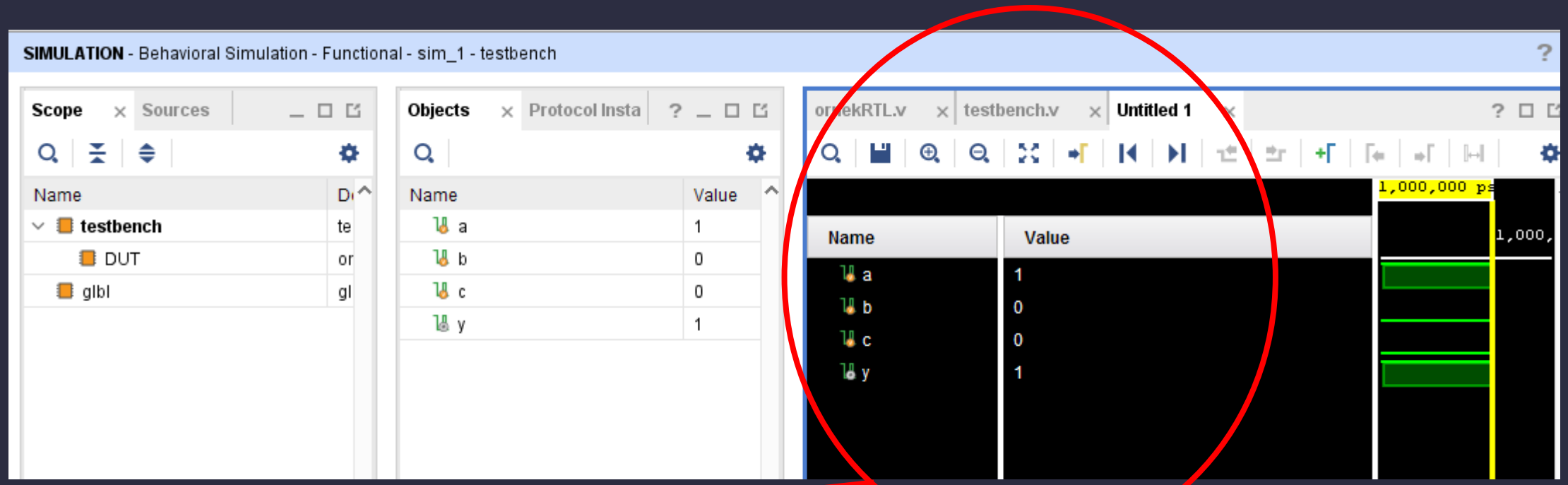
Scope window, lists modules

Verification Approaches



Object window, lists input/output and signals inside of selected module from Scope window

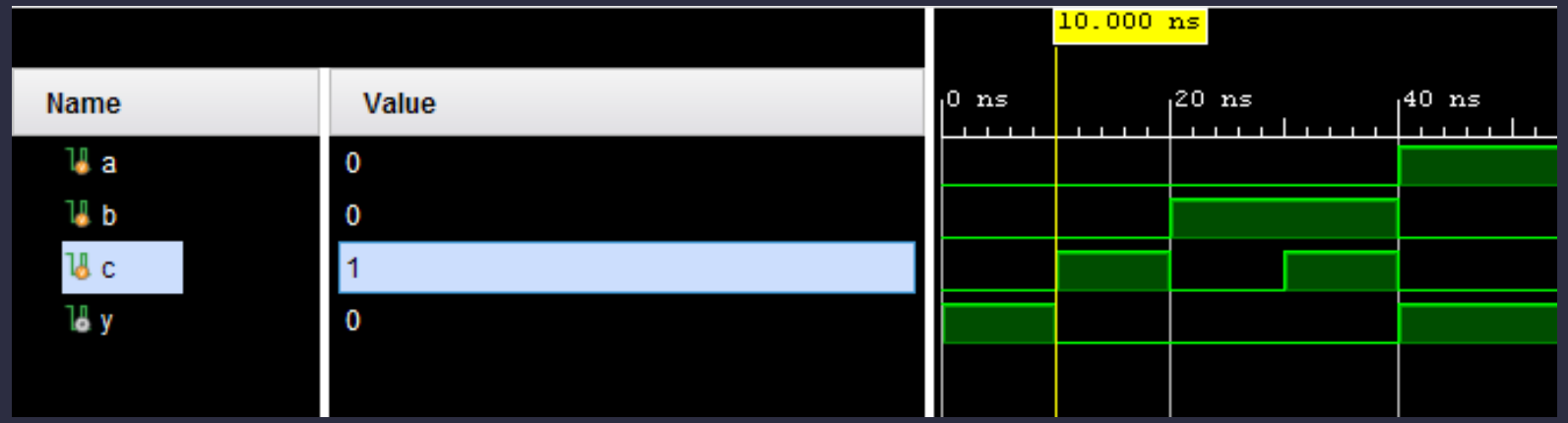
Verification Approaches



Added signals in Waveform (ISIM tool adds signals from testbench top module by default) when simulation is opened .

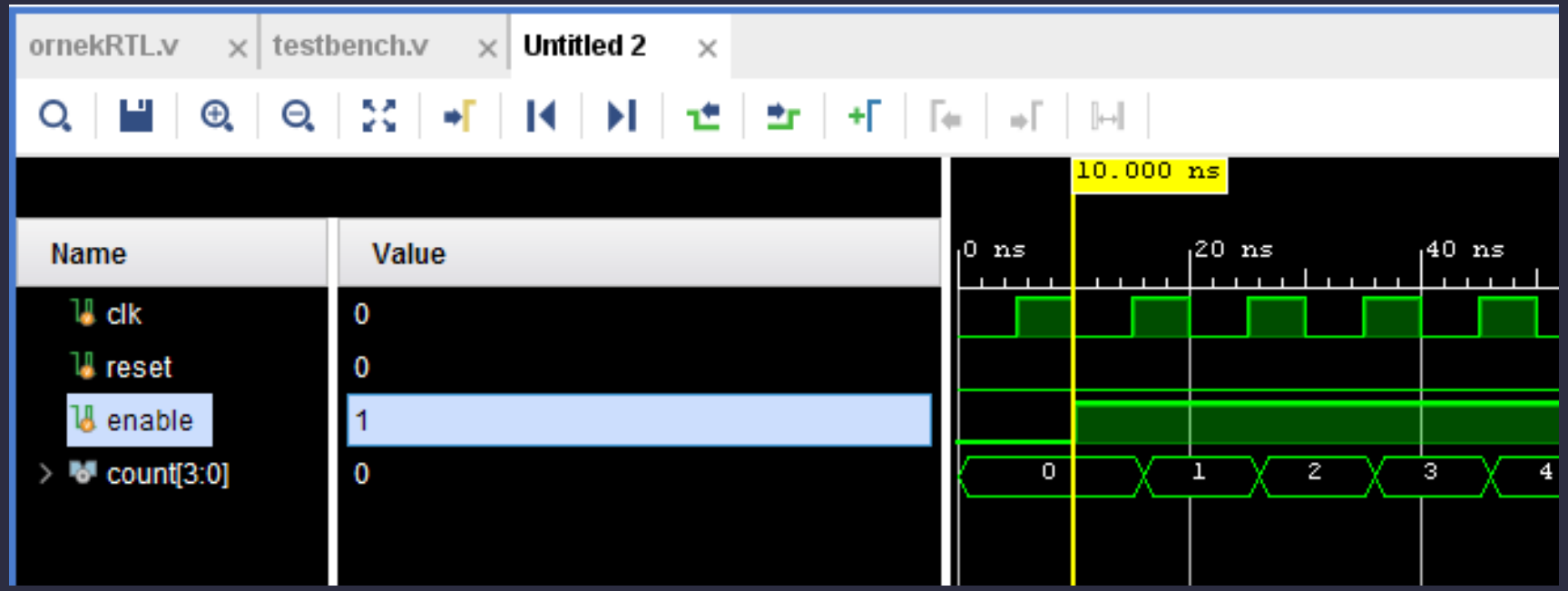
Verification Approaches

- Simulation starts, the waveform is started with too much zoom -in. Zoom -out can be achieved by right-clicking on the waveform or by turning ctrl + mouse middle wheel back .
- Zoom-out is done and when you go to the beginning of the simulation , the image in the figure below can be seen.



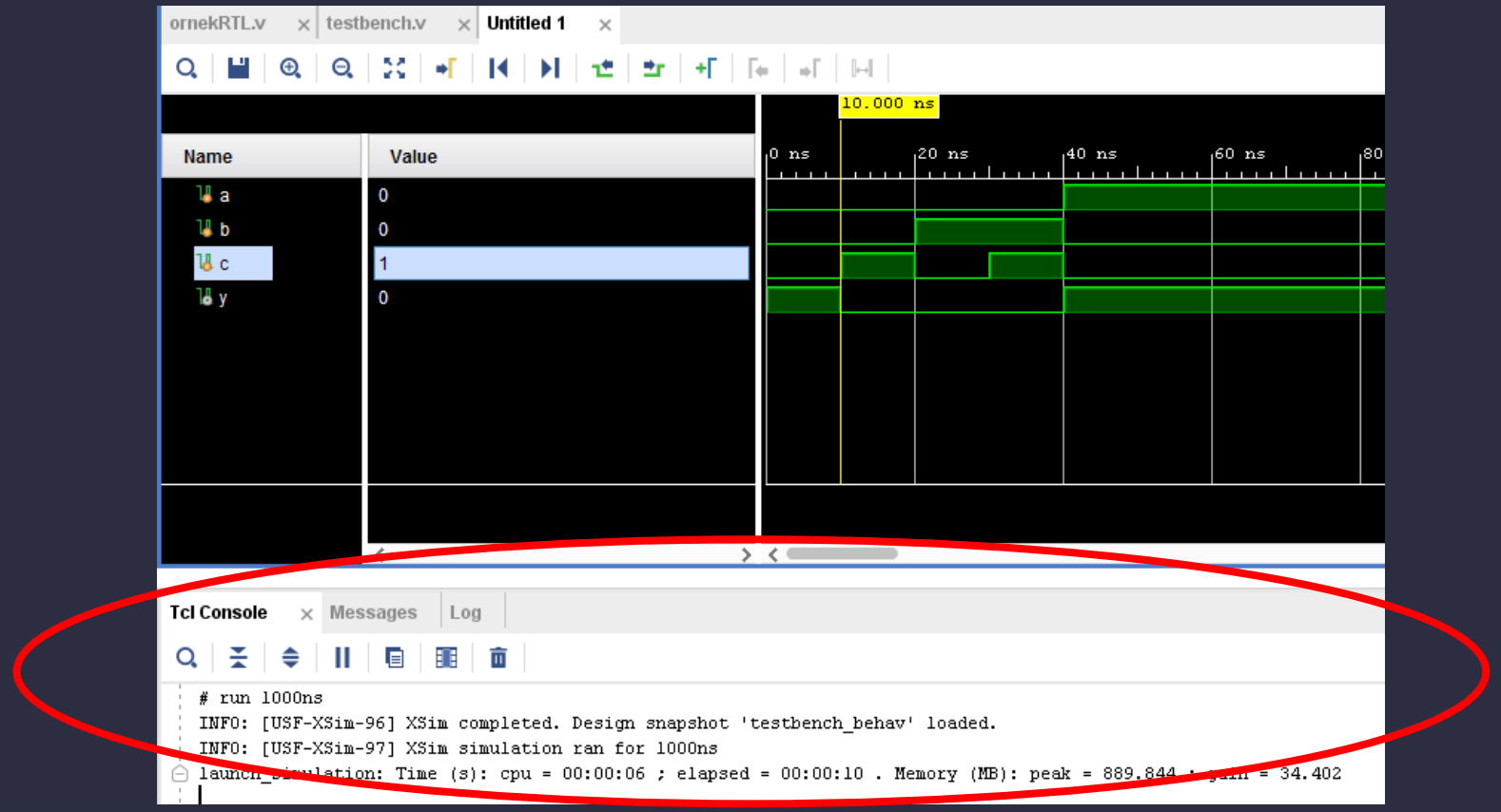
Verification Approaches

- Counter design simulation output



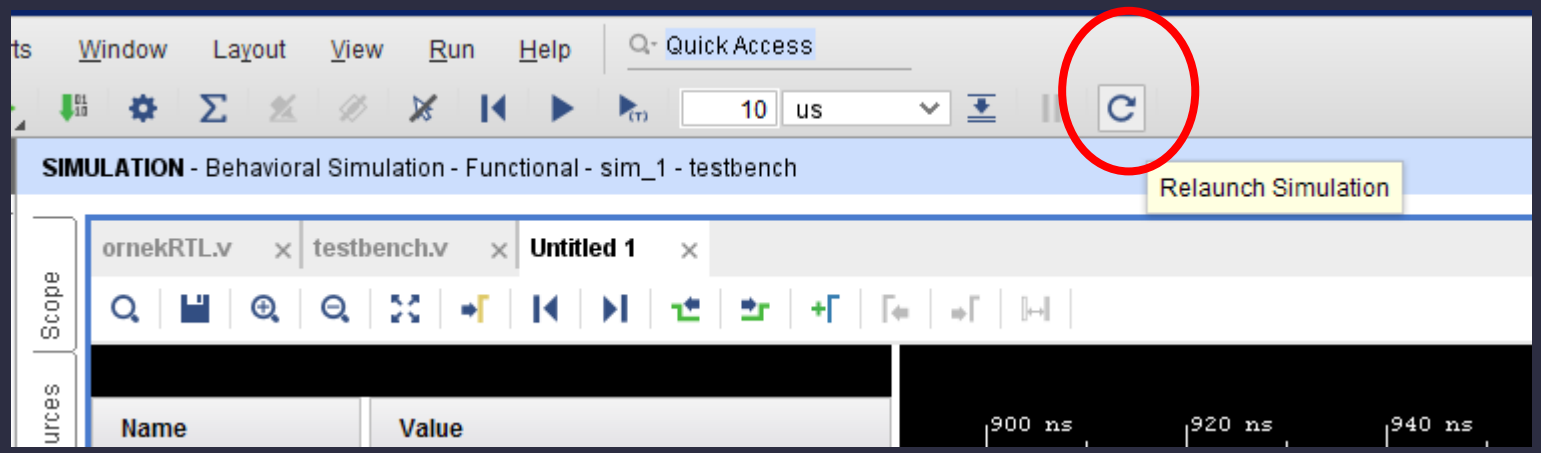
Verification Approaches

- In the TCL Console, outputs can be observed



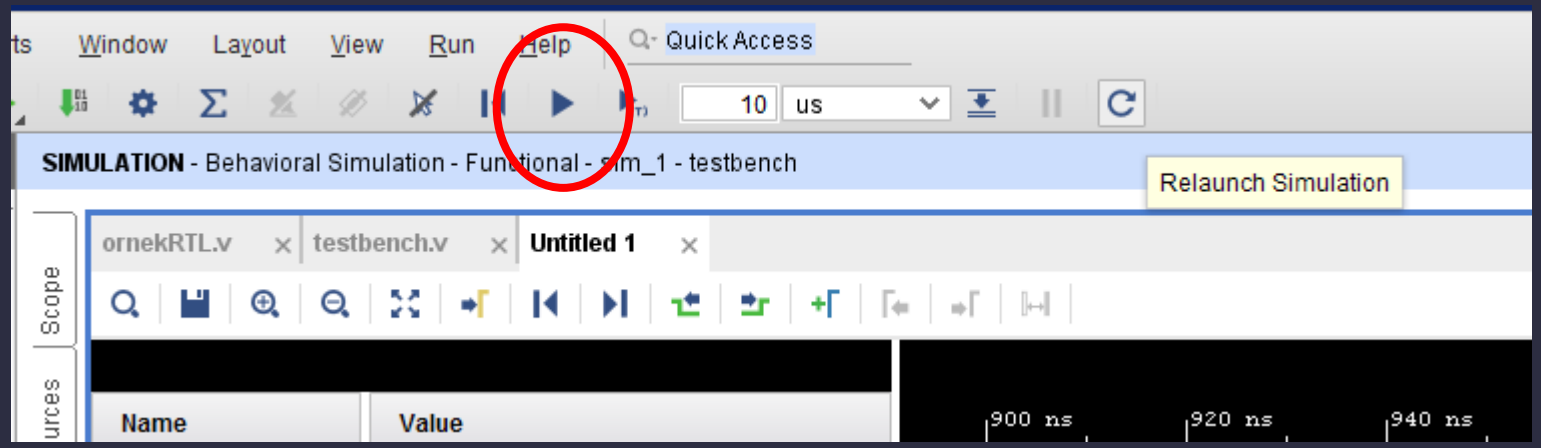
Verification Approaches

- When a change is made in the design, the button shown below can be used to simulate again.



Verification Approaches

- Simulation starts for 10 microseconds by default . If you want to continue the simulation, the play button shown below can be pressed.



Verification Approaches

- If the value of the signals appears as X in the simulation tool, it means that the initial assignment of that signal has not been assigned. It started from an unknown situation.

Verification Approaches

- Difficulty with verification on simulation is there can be lots of input combinations
- For example, for the circuit that calculates the sum of two 32-bit numbers, 2^64 different inputs can be fed. It may take years to try all possible entries.
- So instead of trying every combination, it should be tested by feeding critical inputs.

Verification Approaches

- Generally, when an algorithm is requested to implement its chip, these algorithms are first coded in languages such as C, C++, Matlab.
- For verification these C, C++, Matlab coded algorithms inputs and outputs are written a file.
- These files are used to feeding inputs in to RTL with Testbench and control the outputs produced with reference file.

Verification Approaches

- A working design in Testbench does not mean will work on configured real environment FPGA.
- Generally , there are problems encountered on the development chip due to reasons such as latch or frequency errors that cannot be reached.