# TESTBENCH BASICS

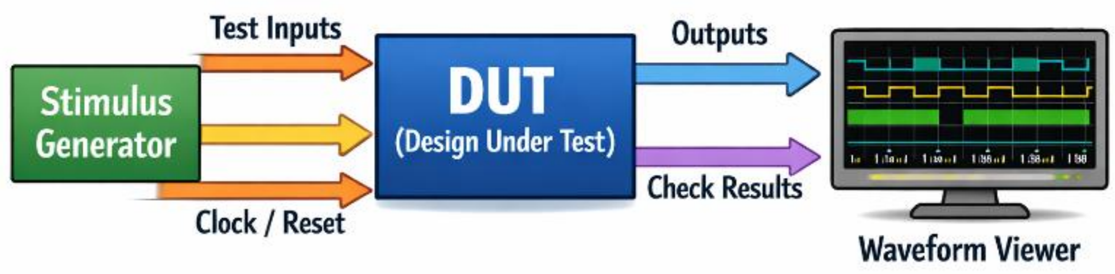
-LEEZA

# WHAT IS A TESTBENCH?

Test benches used for simulation to verify that design works correctly.

Not synthesizable hardware

- Purpose
  - Instantiates the design under test (DUT)
  - Drives inputs (stimulus)
  - Generate clock / reset
  - Observes outputs
  - Checks if behavior is expected against a golden model
  - Reports errors



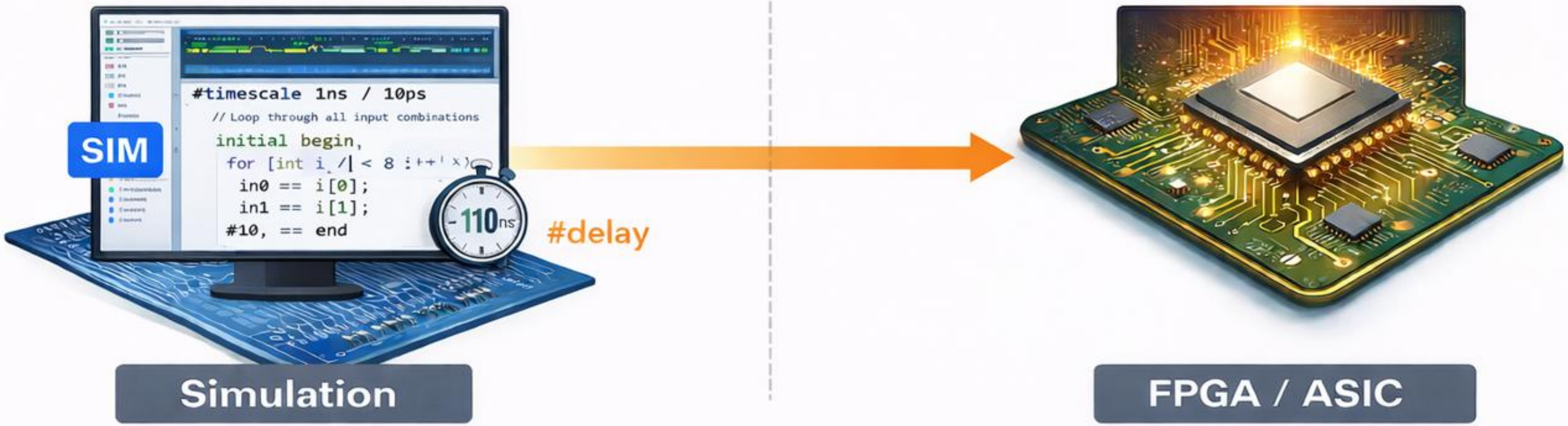
# SIMULATION VS SYNTHESIS

## Simulation

- Runs in a simulator
- Allows initial blocks, delays, and waits
- Functional behavior only
- Not timing or power accurate

## Synthesis

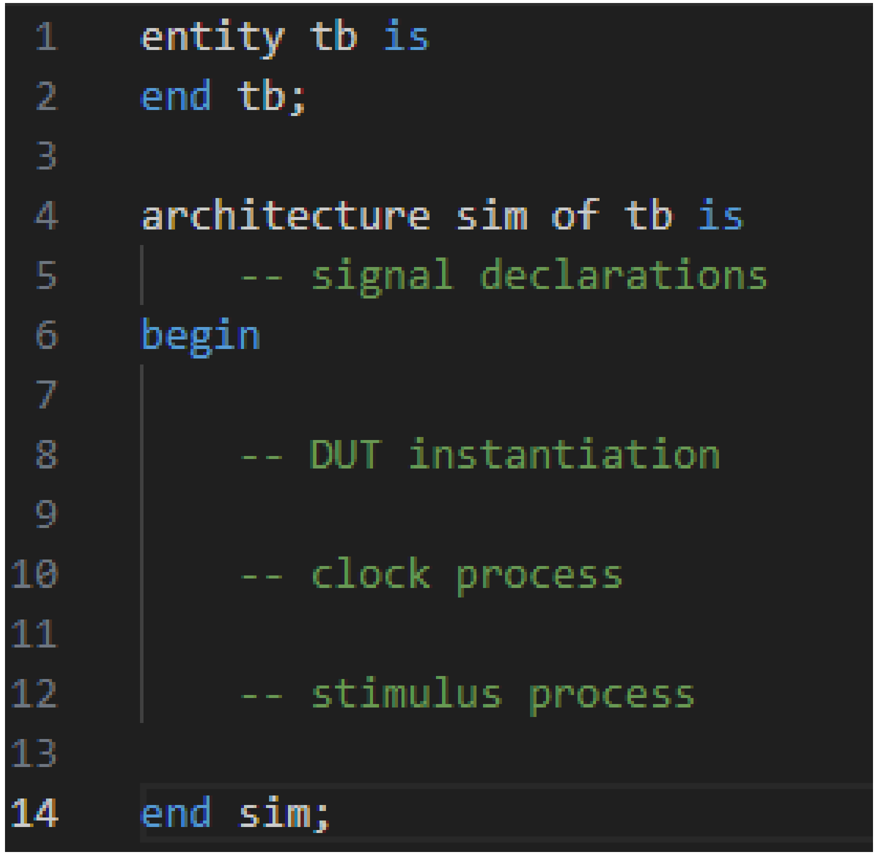
- Converts RTL to hardware
- Only synthesizable RTL allowed
- Runs on FPGA / ASIC



```
#timescale 1ns / 10ps
  // Loop through all input combinations
 initial begin,
  for [int i /| < 8 :++' x)
   in0 == i[0];
   in1 == i[1];
  #10, == end
```

# BASIC STRUCTURE OF A VHDL TB

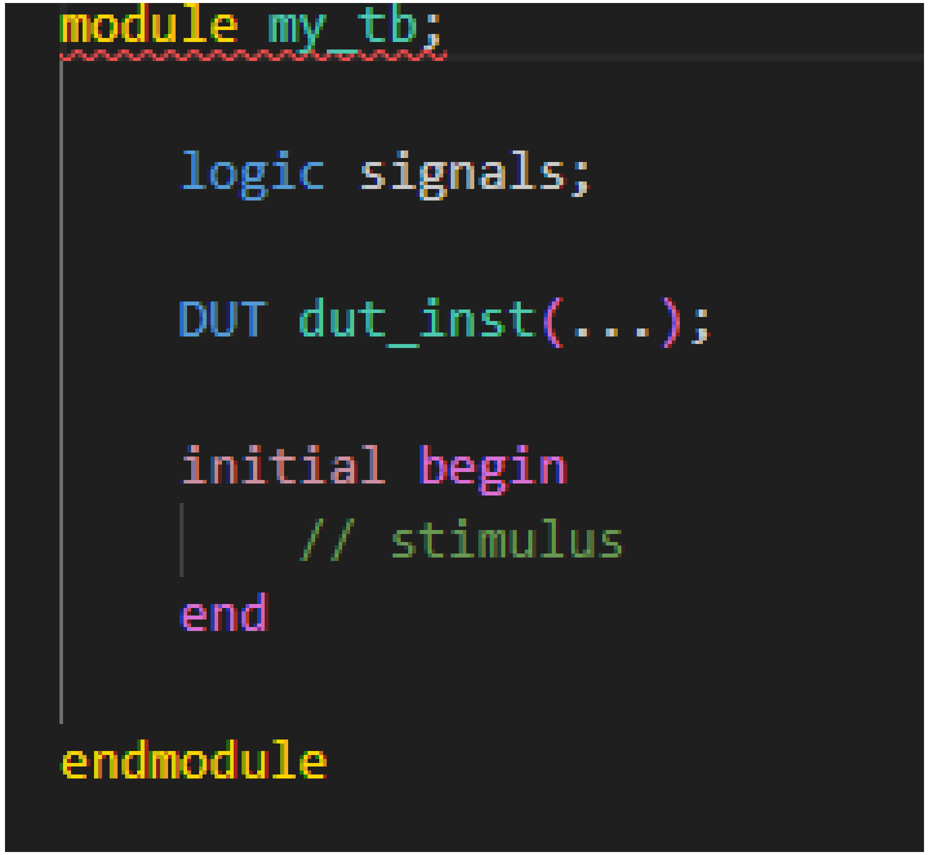
- No ports
- Process drives inputs
- Local signals connected to DUT



```vhdl
entity tb is
end tb;

architecture sim of tb is
    -- signal declarations
begin

    -- DUT instantiation

    -- clock process

    -- stimulus process

end sim;
```

# BASIC STRUCTURE OF A SV TB

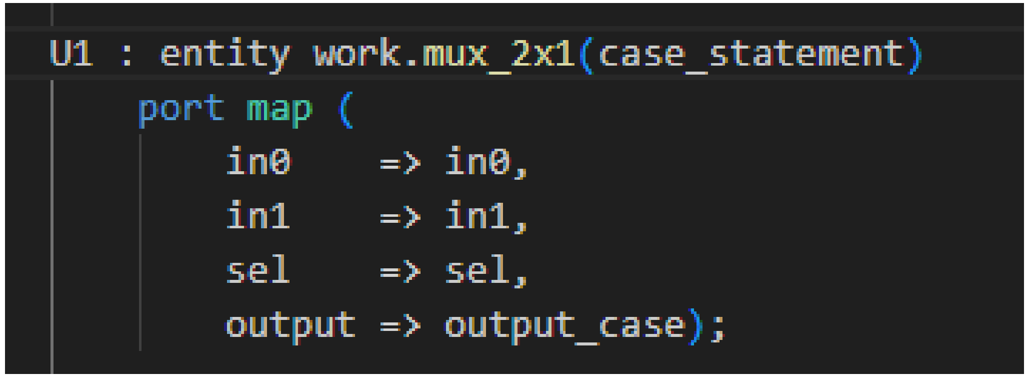
- Module with no ports
- Uses initial block
- Uses delays



```
module my_tb;

    logic signals;

    DUT dut_inst(...);

    initial begin
        // stimulus
    end

endmodule
```

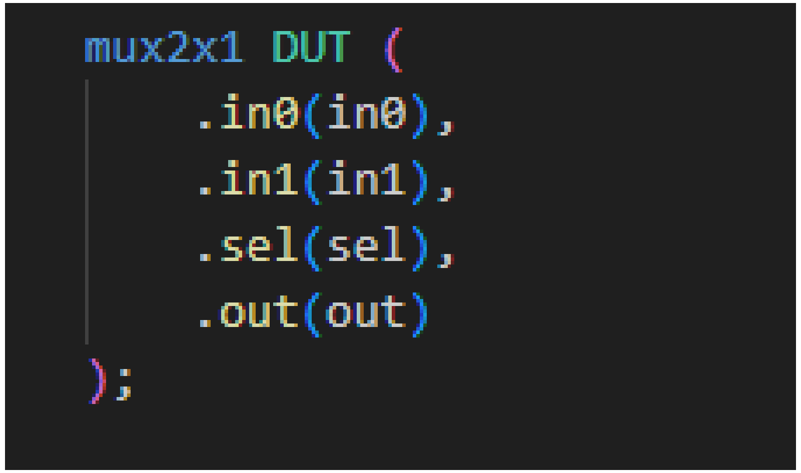
# INSTANTIATING THE DUT

VHDL:



```
U1 : entity work.mux_2x1(case_statement)
    port map (
        in0    => in0,
        in1    => in1,
        sel    => sel,
        output => output_case);
```

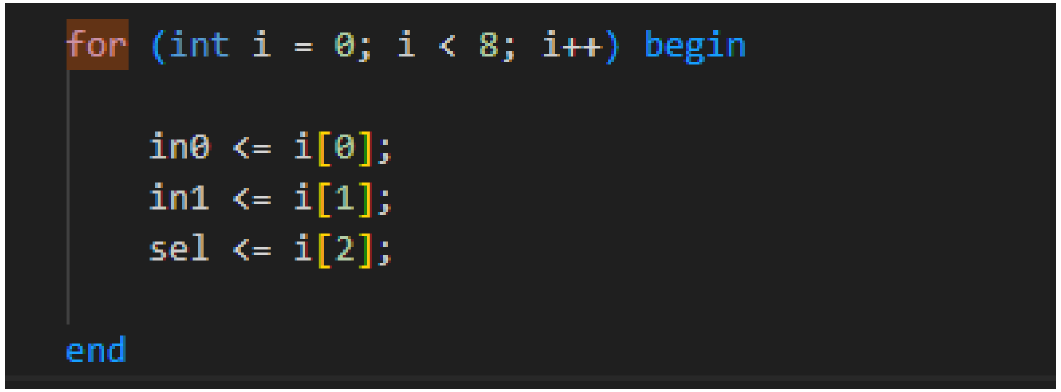
System Verilog:



```
mux2x1 DUT (
    .in0(in0),
    .in1(in1),
    .sel(sel),
    .out(out)
);
```

# DRIVING STIMULUS

- Can loop through combinations
- Apply values over time
- Use non-blocking assignments (<= to drive DUTs
  - Prevents race conditions

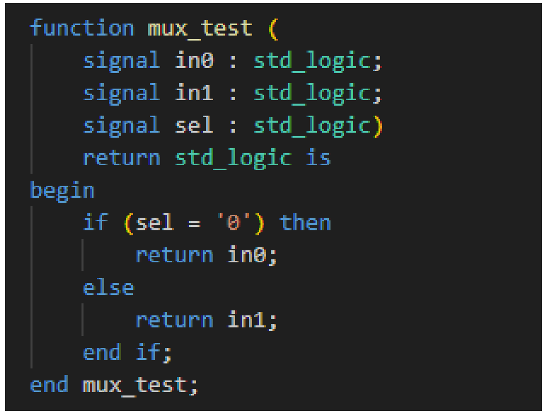


```
for (int i = 0; i < 8; i++) begin

    in0 <= i[0];
    in1 <= i[1];
    sel <= i[2];

end
```

# CHECKING OUTPUTS

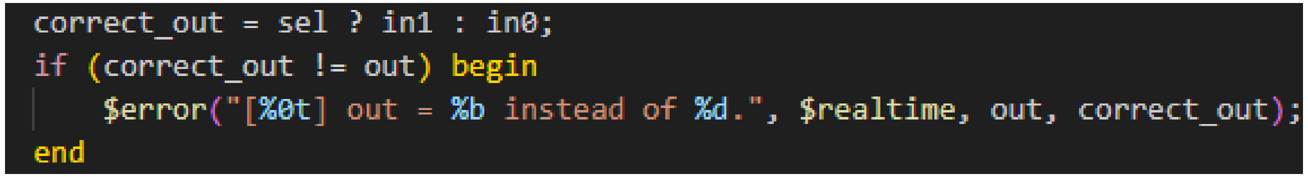
- Compute expected value (golden model)
- Compare
- Report mismatch

VHDL:

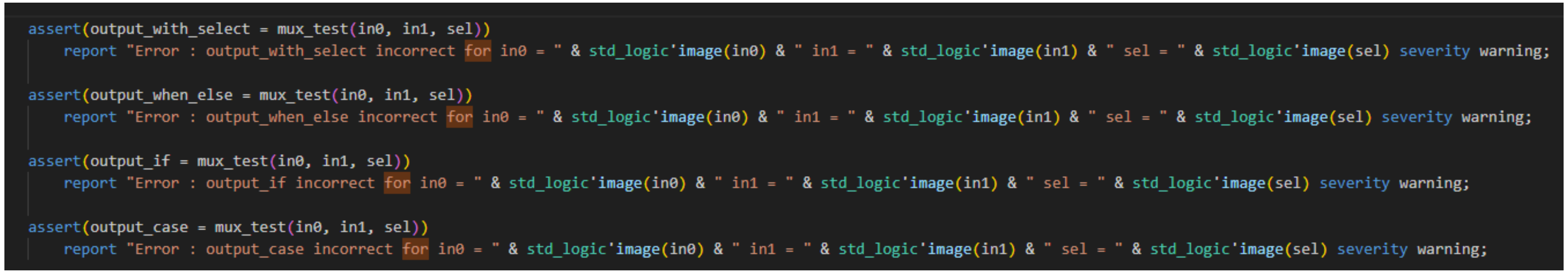


```
function mux_test (
    signal in0 : std_logic;
    signal in1 : std_logic;
    signal sel : std_logic)
    return std_logic is
begin
    if (sel = '0') then
        return in0;
    else
        return in1;
    end if;
end mux_test;
```

System Verilog:



```
correct_out = sel ? in1 : in0;
if (correct_out != out) begin
    $error("[%0t] out = %b instead of %d.", $realtime, out, correct_out);
end
```



```
assert(output_with_select = mux_test(in0, in1, sel))
    report "Error : output_with_select incorrect for in0 = " & std_logic'image(in0) & " in1 = " & std_logic'image(in1) & " sel = " & std_logic'image(sel) severity warning;

assert(output_when_else = mux_test(in0, in1, sel))
    report "Error : output_when_else incorrect for in0 = " & std_logic'image(in0) & " in1 = " & std_logic'image(in1) & " sel = " & std_logic'image(sel) severity warning;

assert(output_if = mux_test(in0, in1, sel))
    report "Error : output_if incorrect for in0 = " & std_logic'image(in0) & " in1 = " & std_logic'image(in1) & " sel = " & std_logic'image(sel) severity warning;

assert(output_case = mux_test(in0, in1, sel))
    report "Error : output_case incorrect for in0 = " & std_logic'image(in0) & " in1 = " & std_logic'image(in1) & " sel = " & std_logic'image(sel) severity warning;
```

# ENDING SIMULATION

System Verilog

- $finish
- $stop
- Disable block

VHDL

- Wait
- Simulation ends when no events remain