ISE Simulation Overview

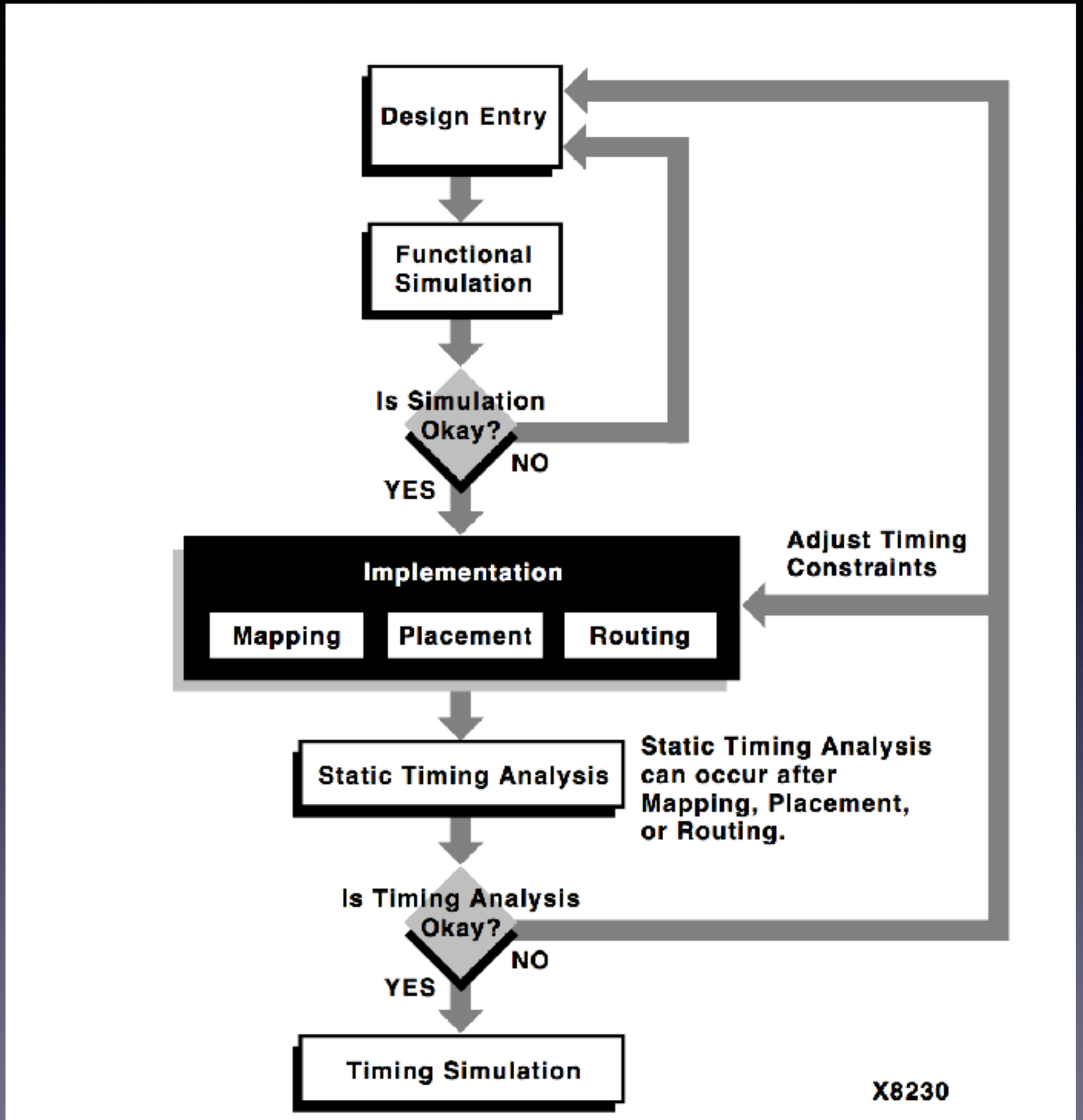
CSULB
CECS 360

https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ise_c_timing_analysis_overview.htm

Introduction

- The Xilinx ISE provides the means to verify several representations of your design
- Verification involves both the functional design (does it do what it is supposed to?) and the timing design (will it meet timing once programmed on the FPGA?)
- This presentation walks through the mechanics of the ISE tool. The next presentation will detail the timing analysis of the design

Design Flow


Design Entry
Functional Simulation
Is Simulation Okay?
NO
YES
Implementation
Mapping
Placement
Routing
Adjust Timing Constraints
Static Timing Analysis
Static Timing Analysis can occur after Mapping, Placement, or Routing.
Is Timing Analysis Okay?
NO
YES
Timing Simulation
X8230

Methodology

- To efficiently analyze timing, a top-down method is recommended, which begins with an inspection of the overall performance of the design, followed by an inspection of the different categories of constraints, a single constraint, and finally a specific path in the design

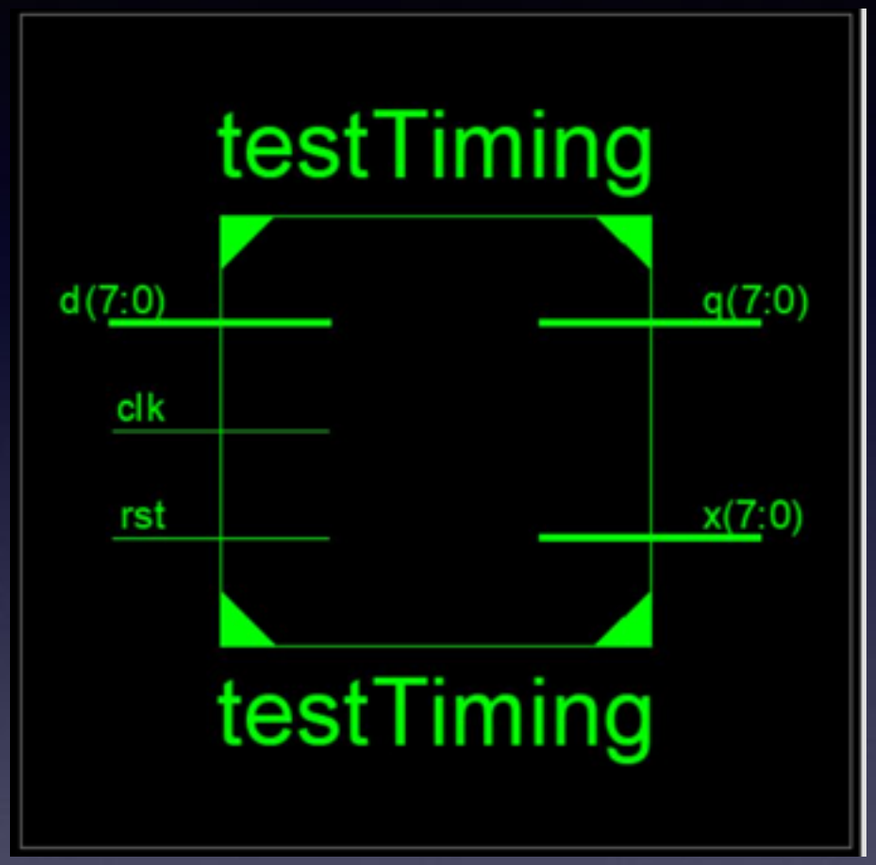
- After analysis is complete, a detailed report of that analysis is created, which can be customized to include only the information you need.

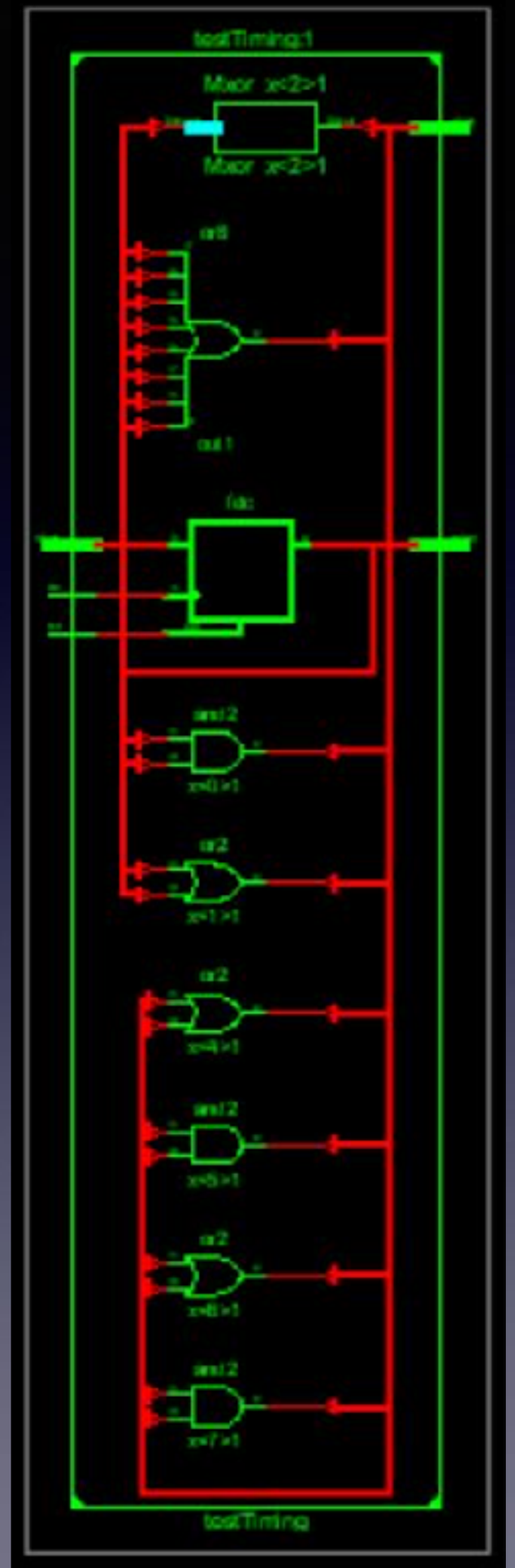
Defining the Clock

- The ucf file is used to specify the association of the design's top level signals with pins on the device
- The ucf file is also used to provide information regarding the design to the tools
- An example is the definition of the clock

```
## Clock signal
NET "clk"    LOC = "E3"   | IOSTANDARD = "LVCMOS33";
NET "clk" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50% INPUT_JITTER 50 ps PRIORITY 1;
```

Our Design

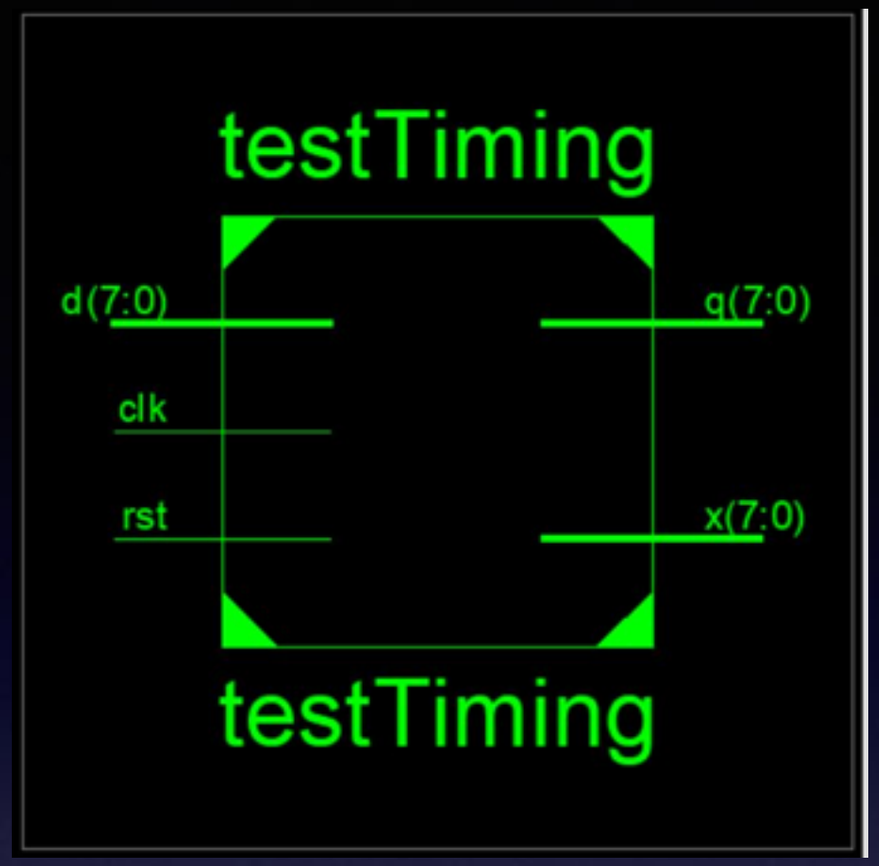




- d[7:0] captured in q[7:0]
- q[7:0] combined with gates to produce x[7:0]

Design Levels

- Each x is produced by different combinational constructs
- The result is that the signals pass through a different number of gates (levels)



Signal	# Levels
x[0] = q[1] & q[0]	1
x[1] = q[3] | q[2]	1
x[2] = q[5] ^ q[4]	1
x[3] = |q	1
x[4] = x[1] | x[0]	2
x[5] = x[3] & x[2]	2
x[6] = x[5] | x[4]	3
x[7] = x[6] & x[5]	4

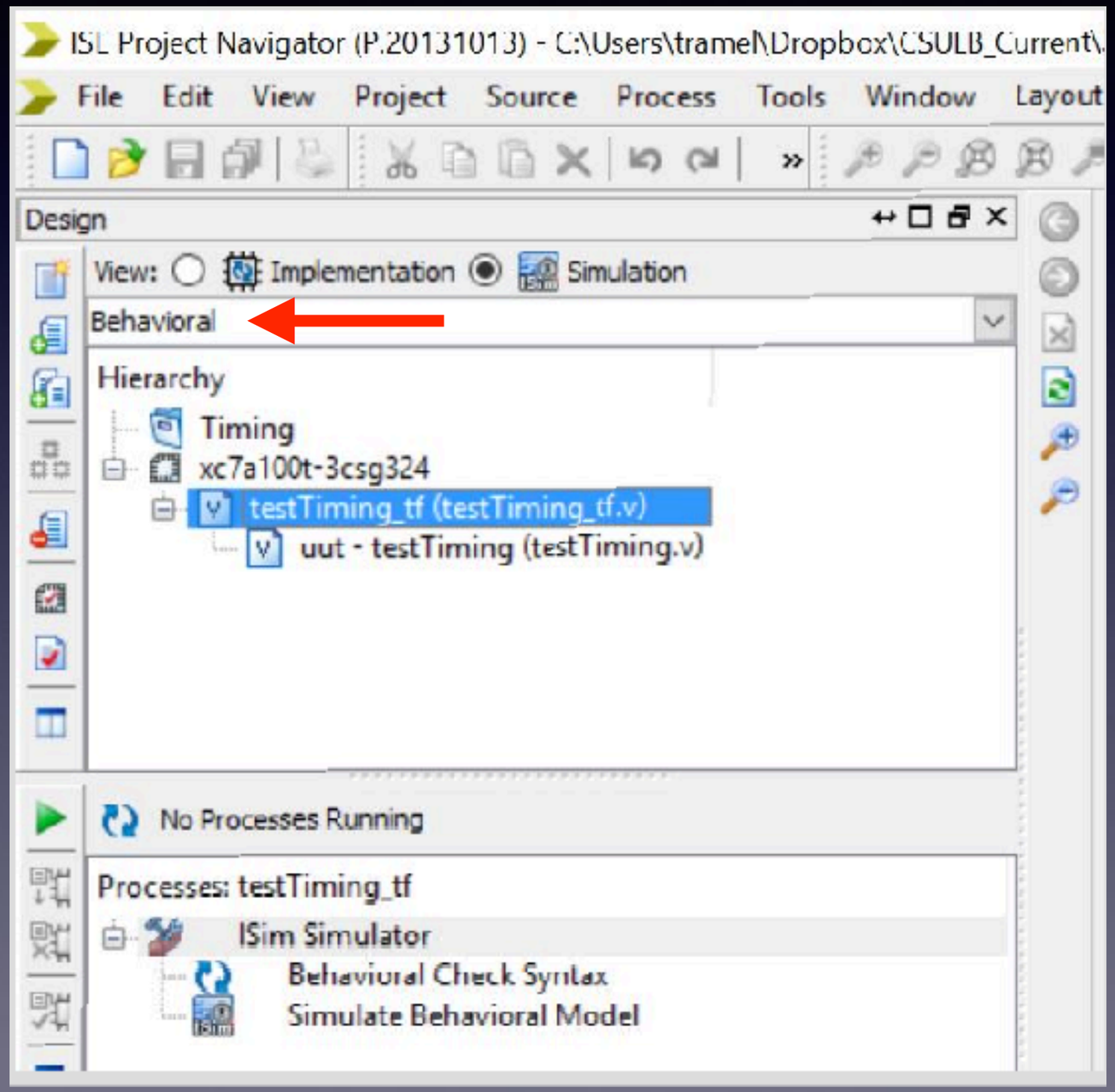
Functional Simulation Early in the Design Flow

- When creating large hierarchical HDL designs it is recommended to perform separate simulations on each module before testing your entire design
- This approach implies that each module has a sufficient amount of logic to make the simulation worth the effort - in other words - this approach assumes that simple circuit functions (flops, muxes, etc) are included in the body of the module and are not separate modules

Test Fixture

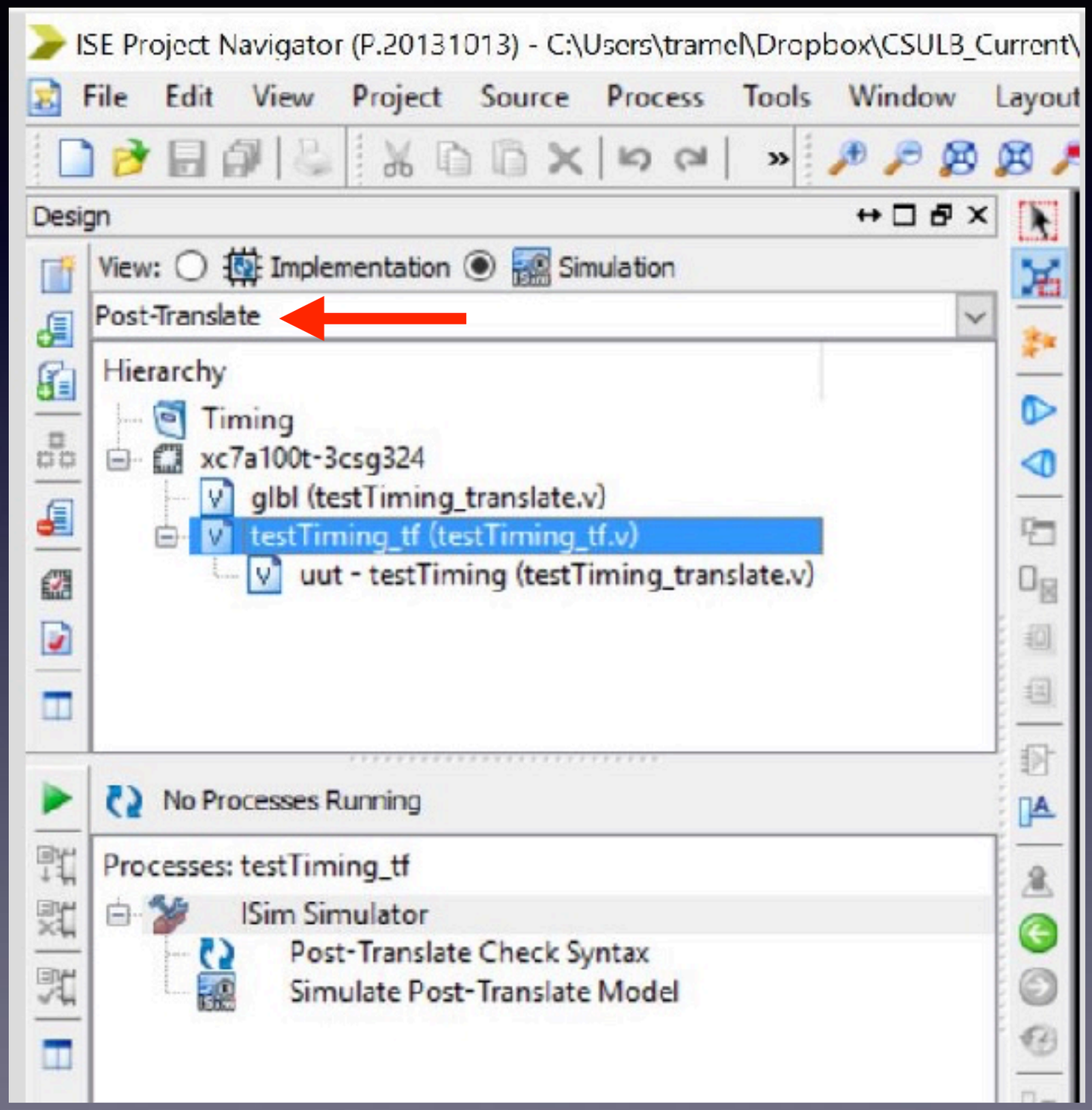
- A test fixture should be created for each module to be verified
- Care should be taken when creating the top-level test fixture
  - Responsible to verify complete functionality
  - Will be reused as the design progresses from RTL through synthesis, translation and mapping

Behavioral Simulation



- All simulation to this point has utilized the default simulation mode “Behavioral”
- The only consideration of time here has been the active edge of the clock
- All switching is assumed to be ideal (i.e. $\Delta t = 0$)

Post-Synthesis Simulation
-or- Post-Translate Simulation



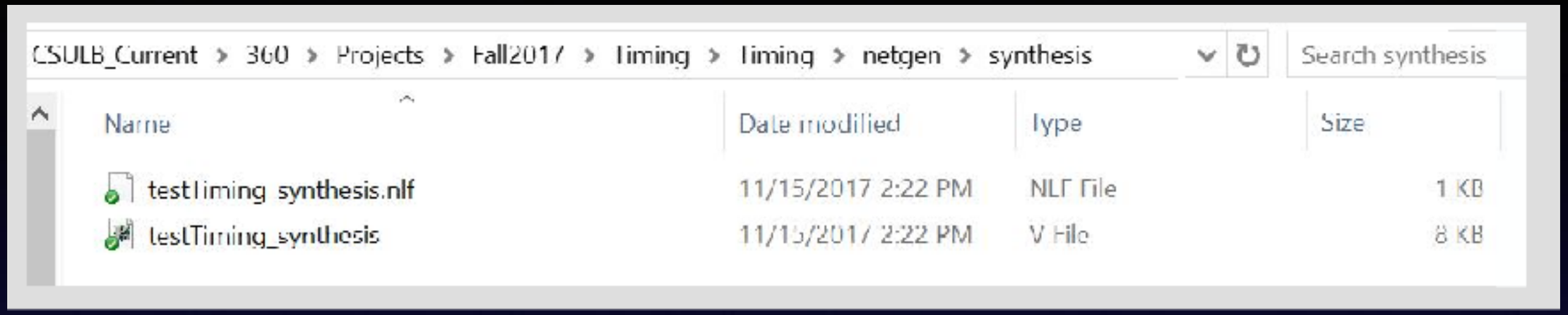
- Now simulating the initial translation of your design
- The switching of flops are assigned fixed (i.e. Clk-Q Δt = 100ps)
- All switching of gates are assumed to be ideal (i.e. $\Delta t = 0$)

Post-Synthesis Simulation
-or- Post-Translate Simulation

- This can be useful when there is suspected problems with the initial translation of your HDL

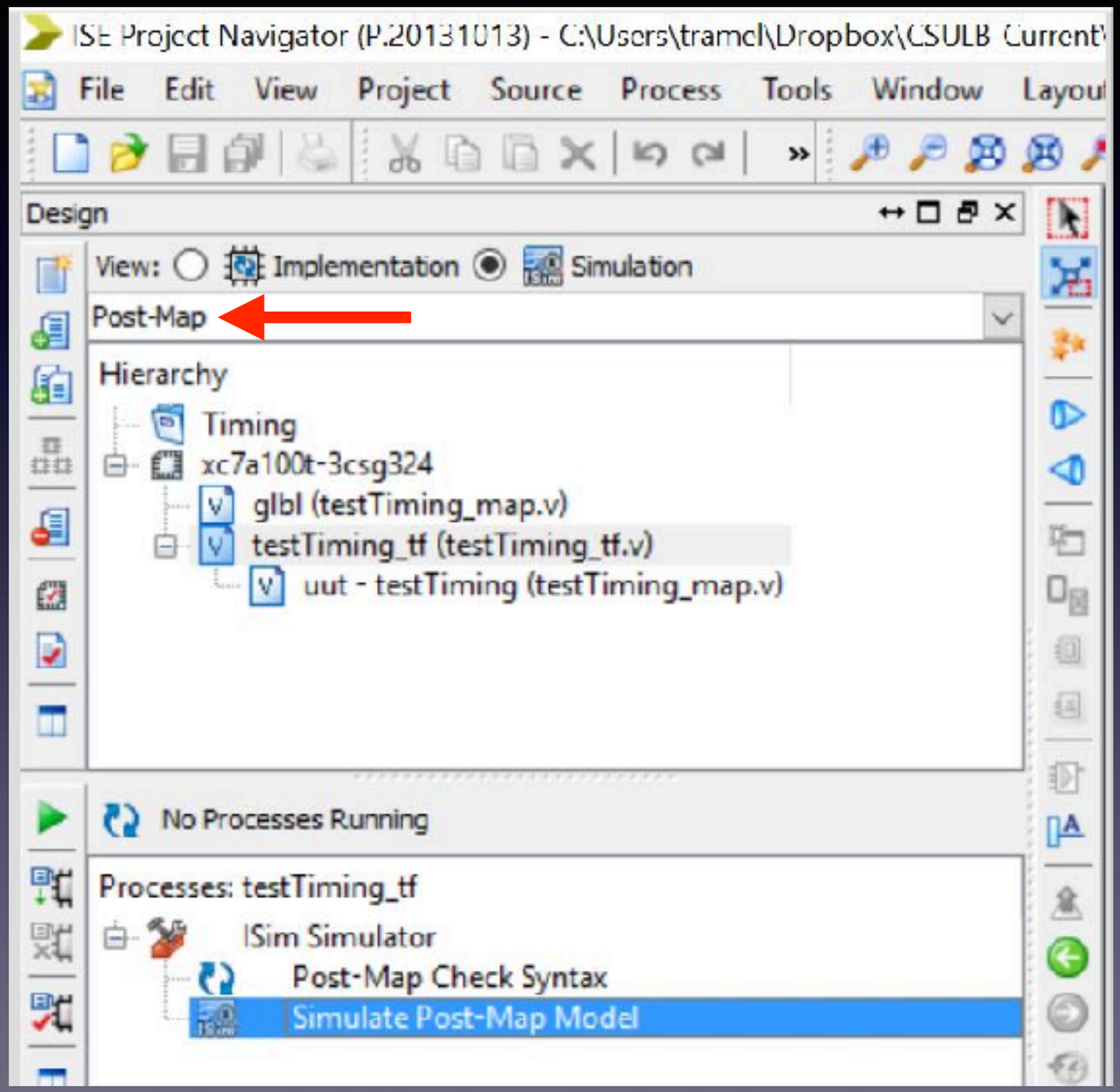
- One example is that synthesis is comprised of two steps: translate & optimize. There are times when portions of the design disappear. This could be helpful there

Post-Synthesis File Generation



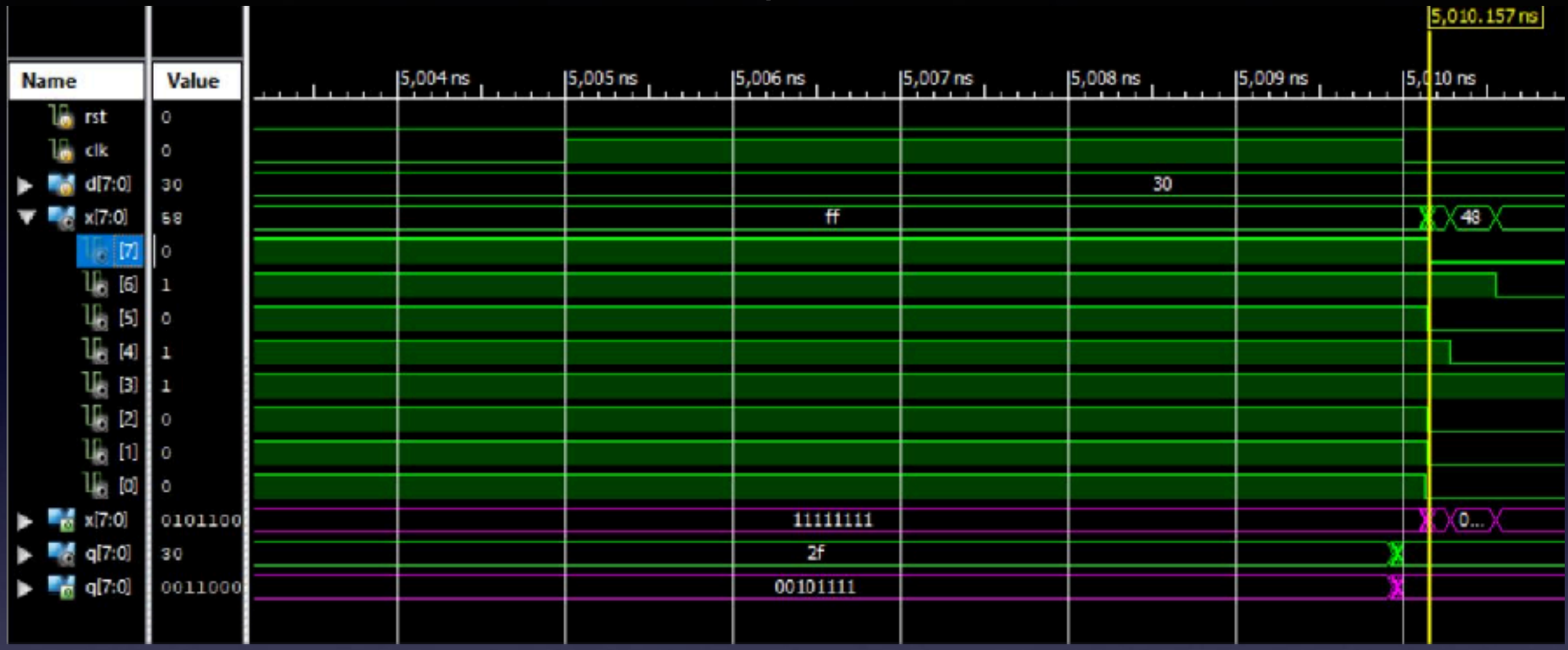
- The netgen directory is the repository for the files that will be generated
- For synthesis there is the netlist (.v) file utilized in the simulation
- The .nlf file is a type of log file recording what has been done

Post-Map Simulation



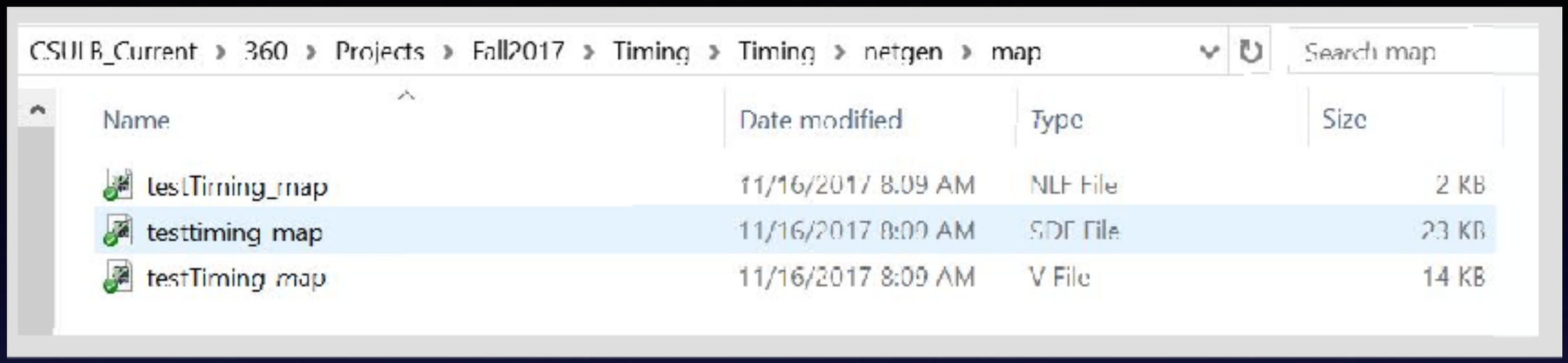
- Now simulating the initial translation of your design
- The switching of flops are assigned delays from the library via the sdf file
- The switching of gates are assigned delays from the library via the sdf file

Post-Map Simulation



- This simulates the final netlist that has been build for implementing the FPGA
- The switching characteristics of the components are loaded into the models using an “sdf” file and represent the library estimates

Post-Map File Generation



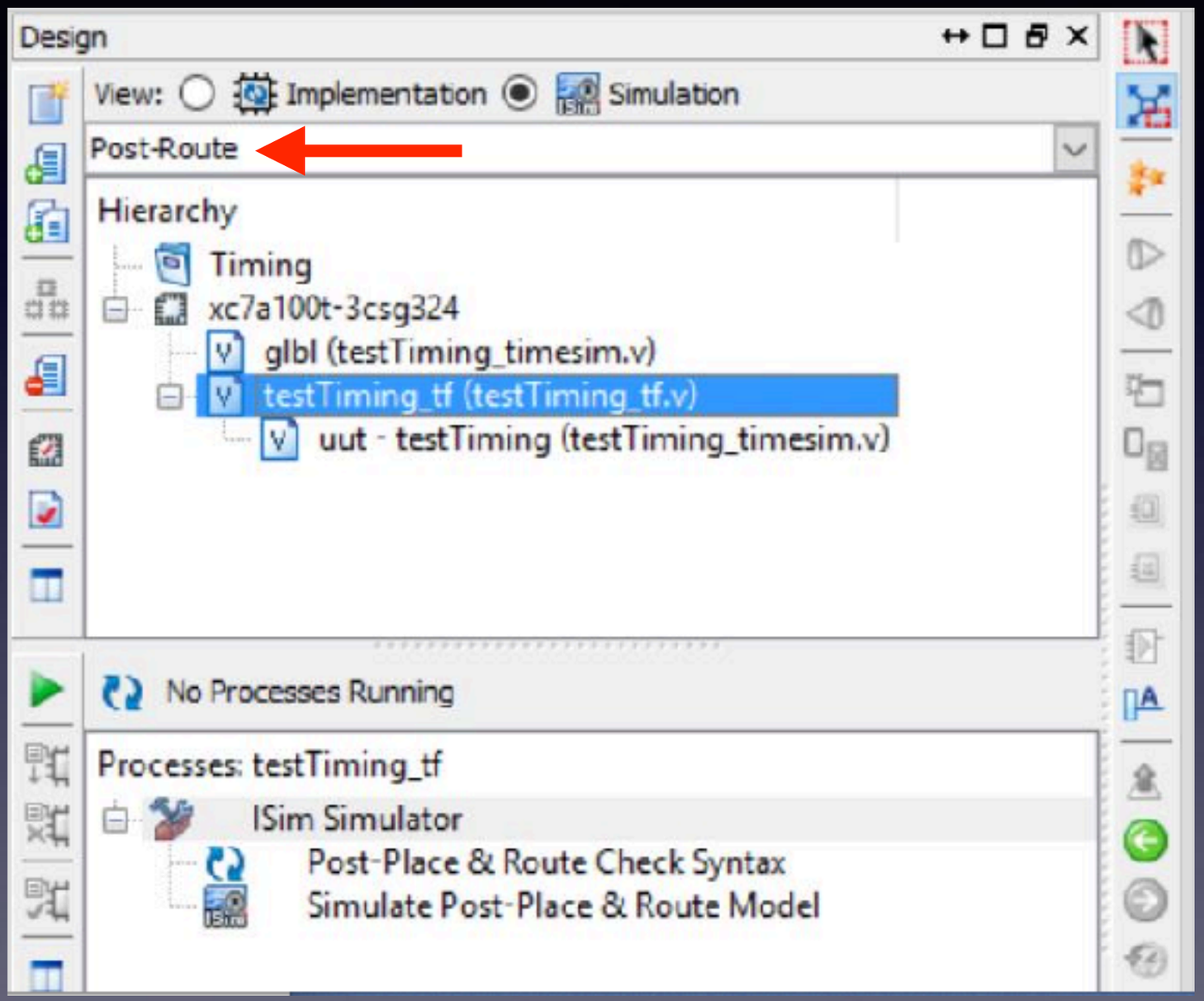
- The netgen directory is the repository for the files that will be generated
- For mapping there is the netlist (.v) file utilized in the simulation
- The .sdf file (Standard Delay Format) contains the switching characteristics of all of the components in the netlist

Post-Map sdf File

```
(DELAYFILE
  (SDFVERSION "3.0")
  (DESIGN "testTiming")
  (DATE "Thu Nov 16 08:09:06 2017")
  (VENDOR "Xilinx")
  (PROGRAM "Xilinx SDF Writer")
  (VERSION "P.20131013")
  (DIVIDER /)
  (VOLTAGE 0.95)
  (TEMPERATURE 85)
  (TIMESCALE 1 ps)
  (CELL (CELLTYPE "X_BUF")
    (INSTANCE d_7_IBUF)
      (DELAY
        (PATHPULSE (50))
        (ABSOLUTE
          (IOPATH I O (275:1356:1356)(275:1356:1356))
        )
      )
  )
  (CELL (CELLTYPE "X_OBUF")
    (INSTANCE q_6_OBUF)
      (DELAY
        (ABSOLUTE
          (PORT I ( 0 )( 0 ))
          (IOPATH I O (1255:3176:3176)(1255:3176:3176))
        )
      )
  )
  (CELL (CELLTYPE "X_OBUF")
    (INSTANCE q_7_OBUF)
      (DELAY
```

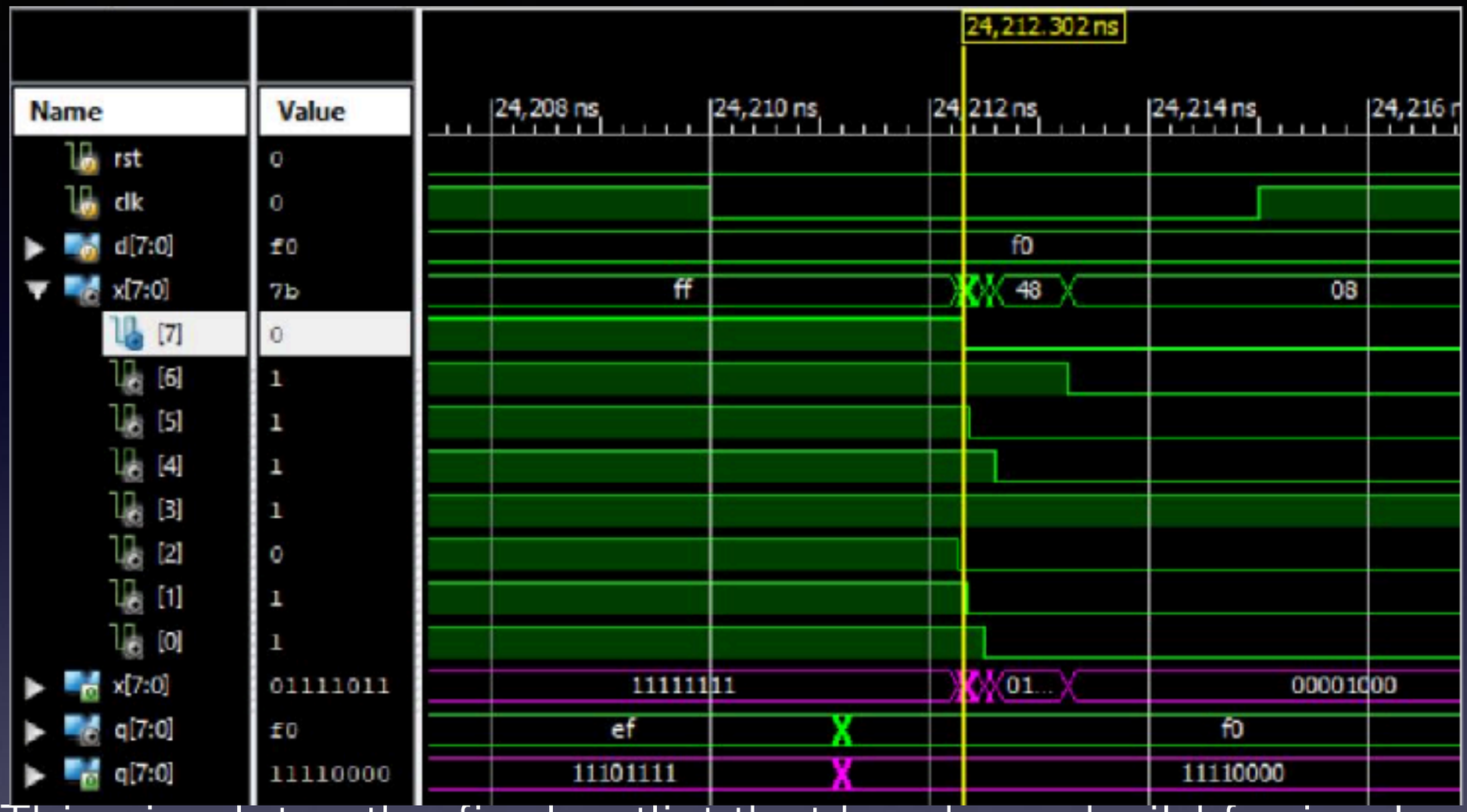
- The sdf file documents the assumptions that were made during the creation of the file
- Each cell has its switching characteristics defined in terms of (rise)(fall) with each define in terms of (min:typ:max)
- Simulation may choose which delay mode (m:t:mx) to use

Post-Route Simulation



- Now simulating the initial translation of your design
- The switching of flops are assigned delays from the library via the sdf file
- The switching of gates are assigned delays from the library via the sdf file

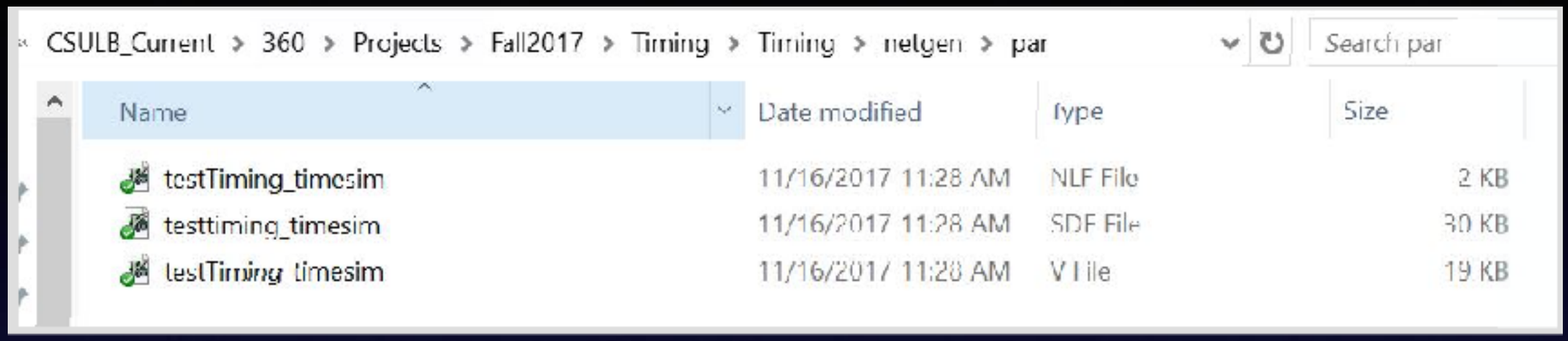
Post-Route Simulation



- This simulates the final netlist that has been build for implementing the FPGA

- The switching characteristics of the components are loaded into the models using an “sdf” file and represent the estimated values after the design has been “placed” and “routed”

Post-Route File Generation



- The netgen directory is the repository for the files that will be generated
- For PAR there is the netlist (.v) file utilized in the simulation
- The .sdf file (Standard Delay Format) contains the switching characteristics of all of the components in the netlist

Post-Route sdf File

```
(CELL (CELLTYPE "% FF")
  (INSTANCE q_2_1)
    (DELAY
      (ABSOLUTE
        (PORT CLK ( 0 )( 0 ))
        (PORT I ( 0 )( 0 ))
        (PORT RST (1054:1054:1979)(1054:1054:1979))
        (IOPATH CLK O (177:395:395)(177:395:395))
        (IOPATH RST O (301:718:718)(301:718:718))
      )
    )
    (TIMINGCHECK
      (PERIOD (posedge CLK) (1263))
      (SETUPHOLD(posedge I) (posedge CLK) (404:667:667)(-93:-117:-117))
      (SETUPHOLD(negedge I) (posedge CLK) (404:667:667)(-93:-117:-117))
      (SETUPHOLD(posedge CE) (posedge CLK) (59:315:315)(572))
      (SETUPHOLD(negedge CE) (posedge CLK) (59:315:315)(572))
      (RECREM(negedge RST) (posedge CLK) (104:219:219)(0))
      (RECREM(negedge SET) (posedge CLK) (104:219:219)(0))
    )
)
```

- The sdf not only defines the switching characteristics of the library cell but also defines the timing constraints that will be enforced when simulating the post-route netlist
- If the constraints are violated the user is informed through the log file
- This is referred to as “dynamic” timing analysis