How to create a Quartus project, generate a bitstream, and run SignalTap to test NPU on an FPGA

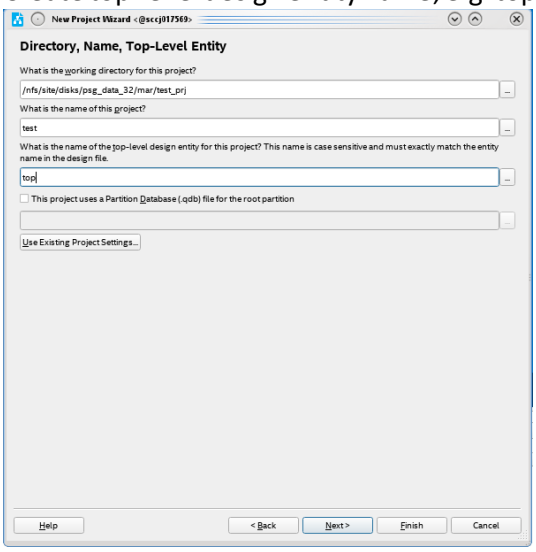
1. Create a source RTL, e.g. top.v

```
`timescale 1ps/1ps
module top (
    input  ref_clk100MHz,
    output [3:0] usr_led);
endmodule
```

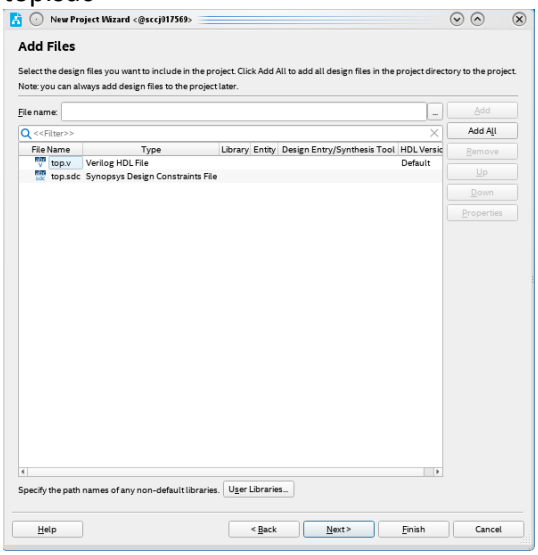
2. Create timing constraint file, e.g. top.sdc

```
create_clock -name {reconfig_clk} -period 10.000 -waveform { 0.000 5.000 }
[get_ports {ref_clk100MHz}]
set_clock_groups -exclusive -group {reconfig_clk}
```

3. Open Quartus -> File -> New Project Wizard
   Create project name, e.g. test.
   Create top-level design entity name, e.g. top.



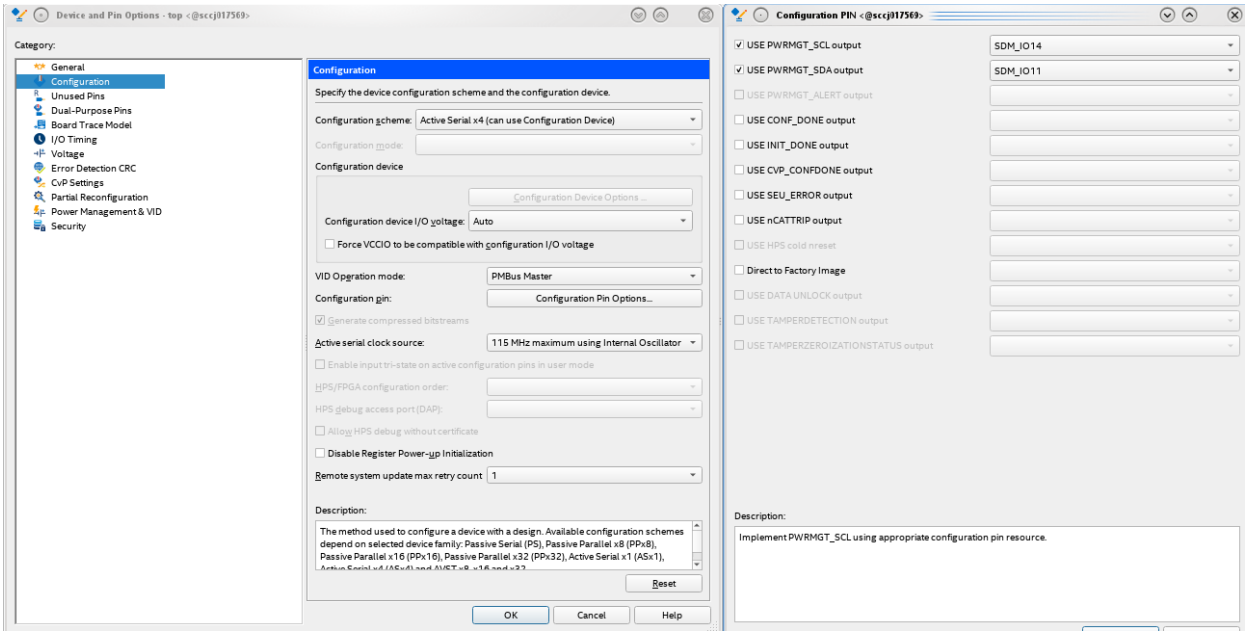
4. Add RTL source -> Finish
   top.v
   top.sdc



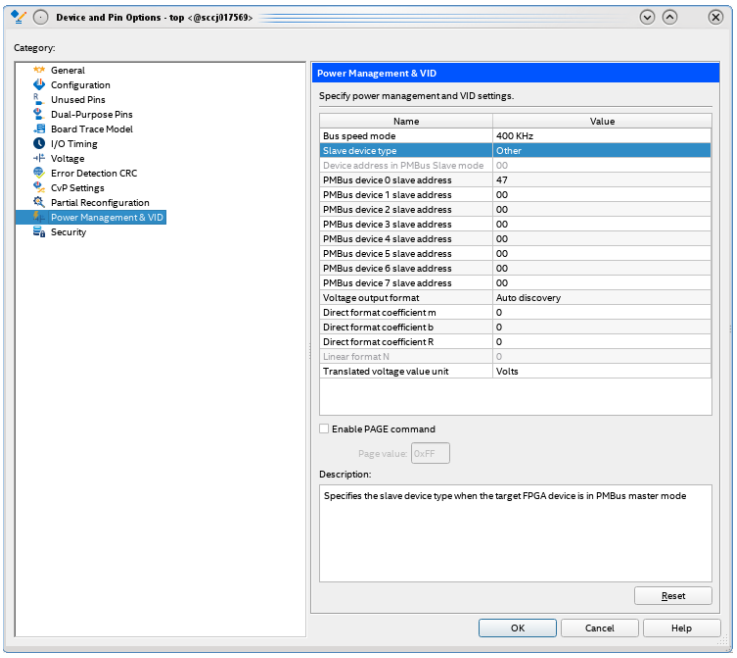
5. Close Quartus, at project directory, open <project name>.qsf file with text editor.

Find two lines setting “device” and “family”. Change them to:
set_global_assignment -name DEVICE 1SN21BHU2F53E2VGI2
set_global_assignment -name FAMILY "Stratix 10"

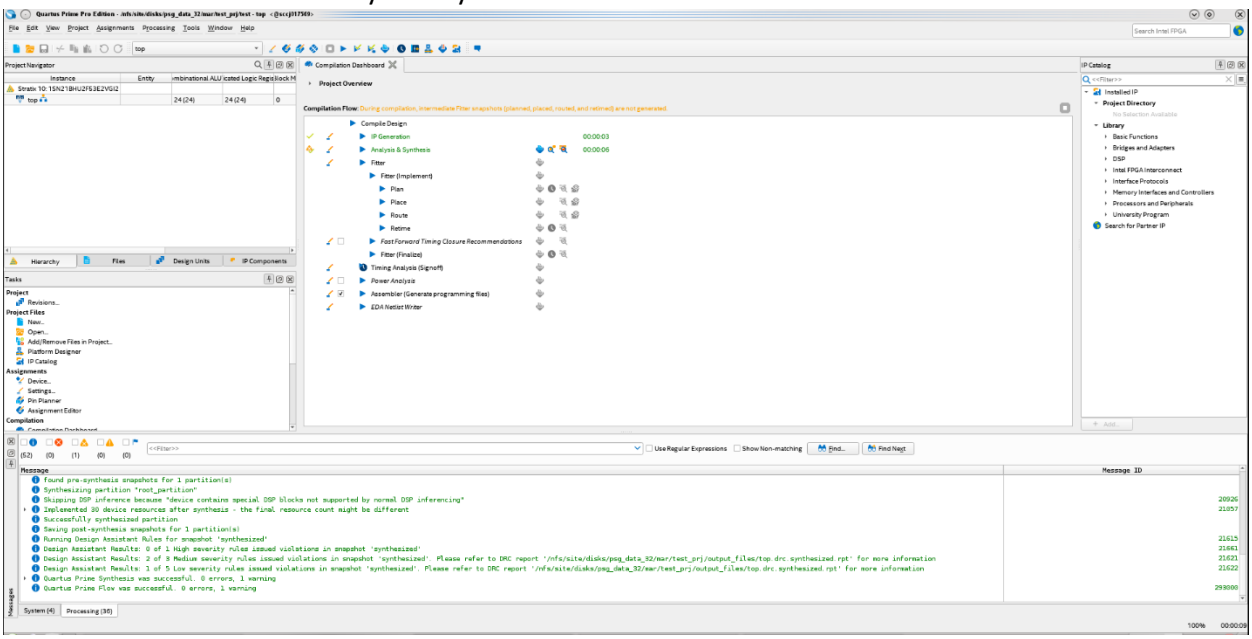
6. Reopen Quartus -> File -> Open Project -> <project name>.qpf
7. Assignments -> Device -> Device & Pin Options -> Configuration -> Configuration Pin Options
   Enable “USE PWRMGT_SCL output” and set value to SDM_IO14
   Enable “USE PWRMGT_SDA output” and set value to SDM_IO11



8. Assignments -> Device -> Device & Pin Options -> Power Management & VID
   BuS speed mode: 400 KHz
   Slave device type: Other
   PMBus device 0 slave address: 47

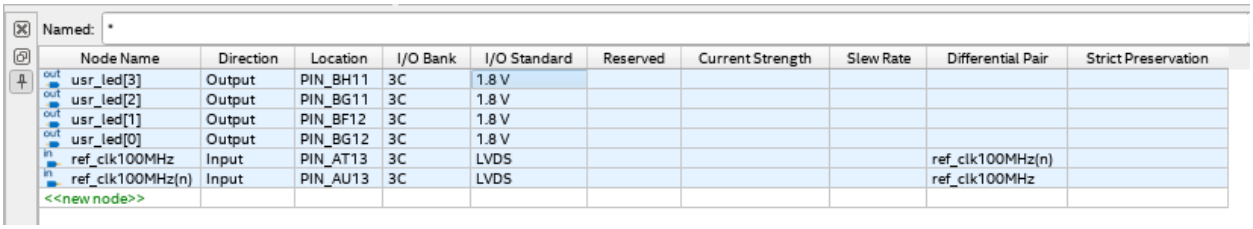


9. Click the start button on “Analysis & Synthesis”



10. Assignment -> Pin Planner -> Set the assignments as illustrated in the table and then the final assignment should look like the picture

	Location	I/O Standard
usr_led[3]	PIN_BH11	1.8V
usr_led[2]	PIN_BG11	1.8V
usr_led[1]	PIN_BF12	1.8V
usr_led[0]	PIN_BG12	1.8V
ref_clk100MHz	PIN_AT13	LVDS



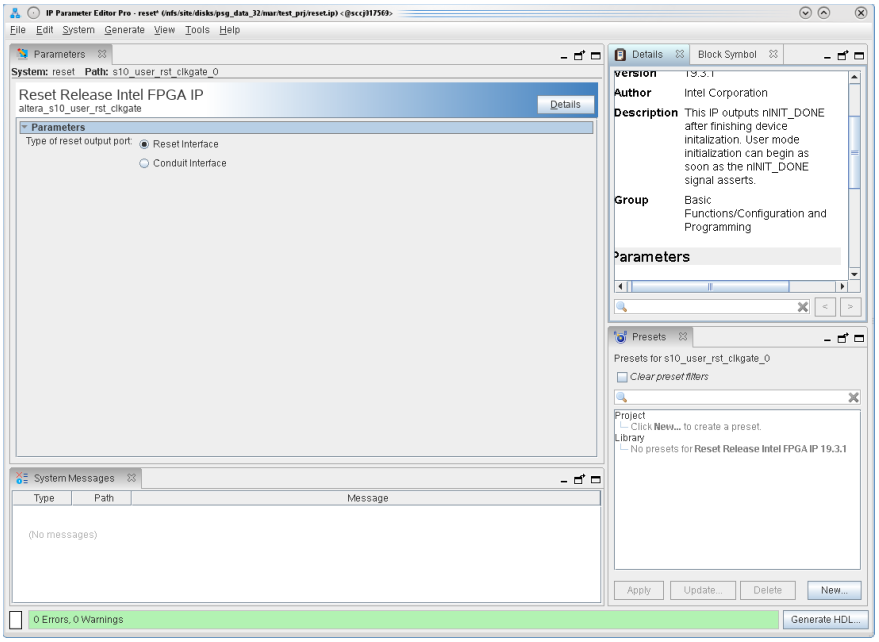
Named: *

Node Name	Direction	Location	I/O Bank	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
usr_led[3]	Output	PIN_BH11	3C	1.8 V					
usr_led[2]	Output	PIN_BG11	3C	1.8 V					
usr_led[1]	Output	PIN_BF12	3C	1.8 V					
usr_led[0]	Output	PIN_BG12	3C	1.8 V					
ref_clk100MHz	Input	PIN_AT13	3C	LVDS				ref_clk100MHz(n)	
ref_clk100MHz(n)	Input	PIN_AU13	3C	LVDS				ref_clk100MHz	
<<new node>>									

11. Tools -> IP Catalog -> Installed IP -> Library -> Basic Functions -> Configuration and Programming -> Reset Release Intel FPGA IP
    a. Save the ip file, e.g. “reset.ip”
    b. Choose “Reset Interface”
    c. Generate HDL, leave all options as is for the rest of the generation
    d. Add the following code into **top.v**

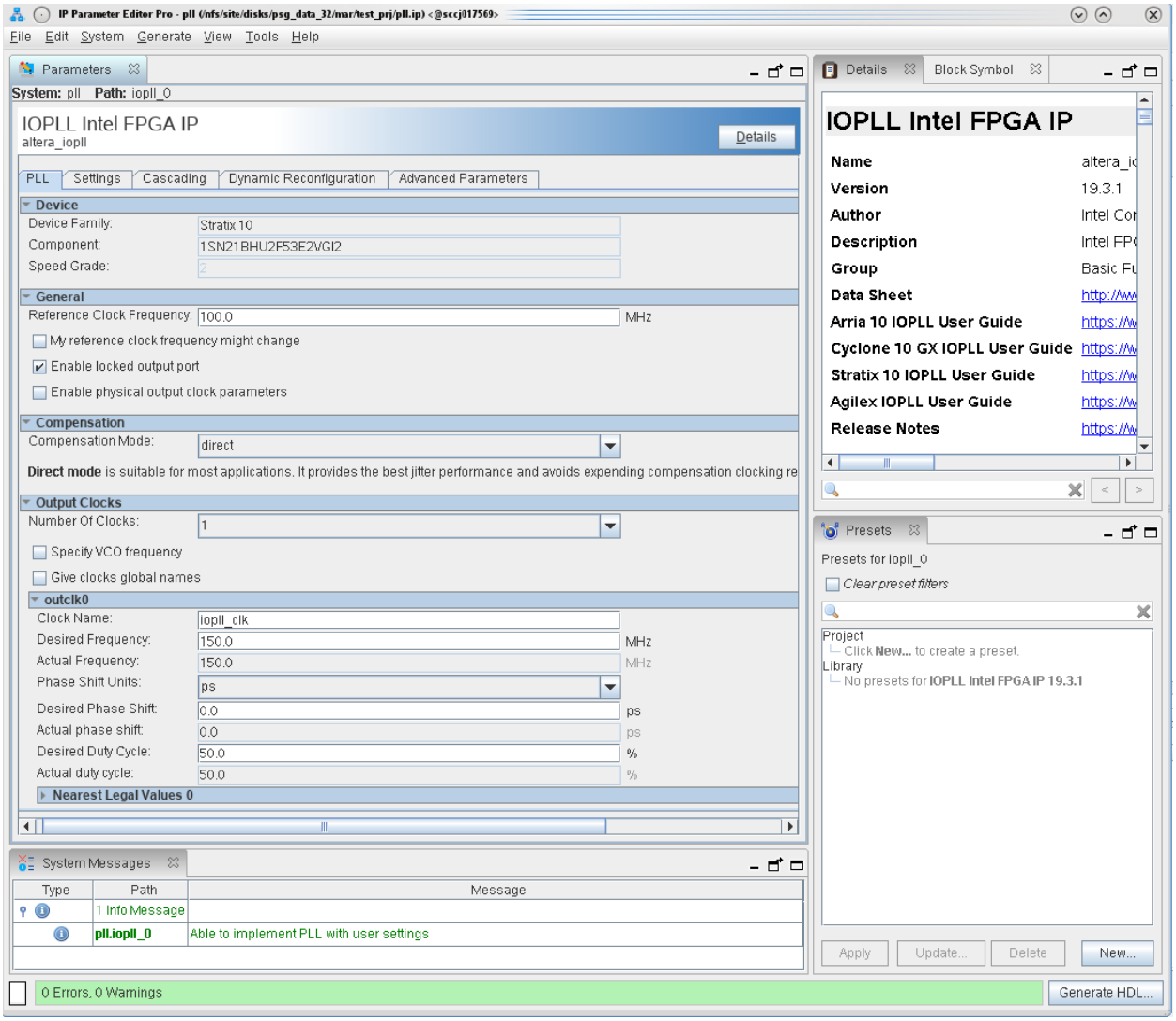
```
wire init_done;
wire ninit_done;
reset s10_reset (
```

```
    .ninit_done (ninit_done)
);
assign init_done = ~ninit_done;
```



12. Tools -> IP Catalog -> Installed IP -> Library -> Basic Functions -> Clocks; PLLs and Resets -> PLL -> IOPLL Intel FPGA IP
    a. Save the ip file, e.g. pll.ip
    b. Set clock name: Output Clocks -> outclk0 -> Clock Name, e.g. clk200
    c. Set output frequency: Output Clocks -> outclk0 -> Desired Frequency, e.g. 200
    d. Keep everything else as default, Generate HDL
    e. Add the following code into **top.v**

```
wire iopll_clk;
wire iopll_locked;
wire usr_rst;
pll pll (
    .rst      (ninit_done),
    .refclk   (ref_clk100MHz),
    .locked   (iopll_locked),
    .outclk_0 (iopll_clk)
);
assign usr_rst = ninit_done | (~iopll_locked);
```



13. Drop in NPU

    Project -> Add/Remove Files in Project -> Add the following NPU files

```
top_sched.sv
tester_rom.sv
tanh.sv
star_interconnect.sv
sigmoid.sv
self_tester_shim.sv
ram.sv
prime_dsp_tensor_int8.sv
pipeline_interconnect.sv
nx_dot_product_int8.sv
nx_dot6_int8.sv
nx_axbs_slice.sv
nx_axbs_core.sv
nx_axbs.sv
npu.sv
mvu_vrf.sv
mvu_tile.sv
mvu_sched.sv
```

```
mvu.sv
mrf_ram.sv
mfu_sched.sv
mfu.sv
ld_sched.sv
ld.sv
inst_ram.sv
inst_fifo.sv
fifo.sv
evrf_sched.sv
evrf.sv
dpe_mrf.sv
dpe.sv
daisy_chain_interconnect.sv
bram_accum.sv
axbs.sv
asynmmetric_fifo.sv
```

14. Open “npu.vh” with any editor -> Make sure that the define statement on line 21, RTL_DIR, points to the right directory for RTL source code, e.g.

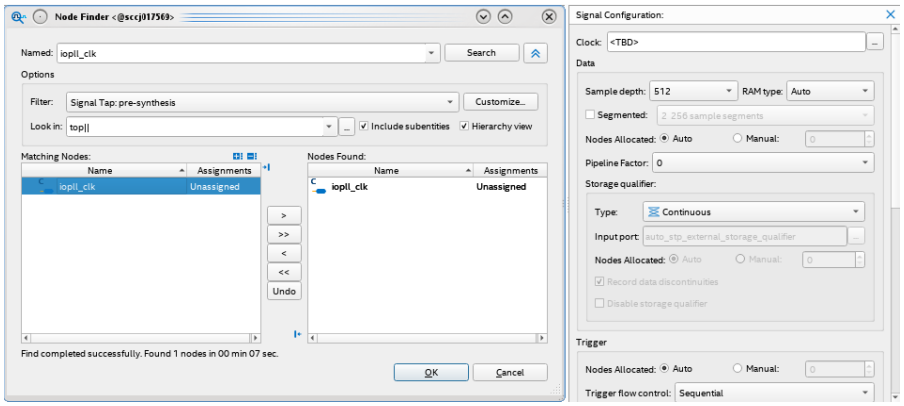
```
`define RTL_DIR                          "<my_directory>"
```

15. Hook NPU, add the following logic to **top.v**

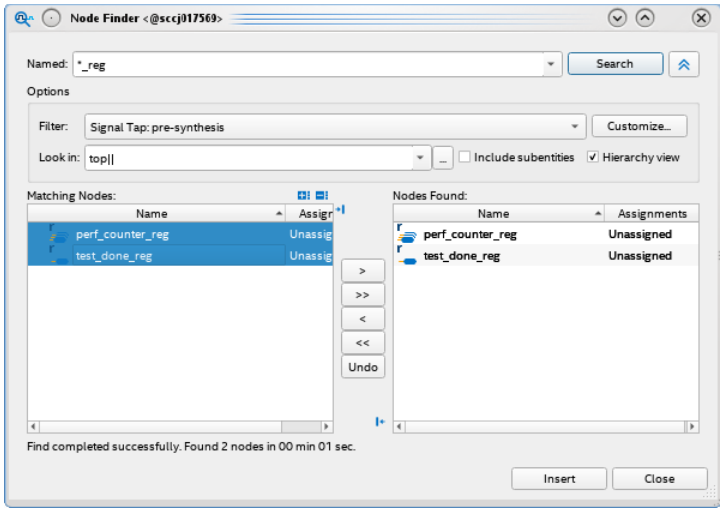
```
reg [30:0] count = 'b0;
wire [2:0] test_status;
wire [31:0] result_count;
wire [31:0] perf_counter;
wire test_done;
reg [31:0] perf_counter_reg /* synthesis preserve */;
reg test_done_reg /* synthesis preserve */;
always @(posedge iopll_clk) begin
    if (usr_rst) begin
        count <= 'd0;
    end else if (!count[30]) begin
        count <= count + 31'd1;
    end
    if (!count[30]) test_done_reg <= 'b0;
    else test_done_reg <= test_done;
    perf_counter_reg <= perf_counter;
end

self_tester_shim npu_self_tester(
    .clk(iopll_clk),
    .reset(~count[30]),
    .o_test_status(test_status),
    .o_result_count(result_count),
    .o_perf_counter(perf_counter),
    .o_test_done(test_done)
);
assign usr_led = {1'b0, test_status};
```

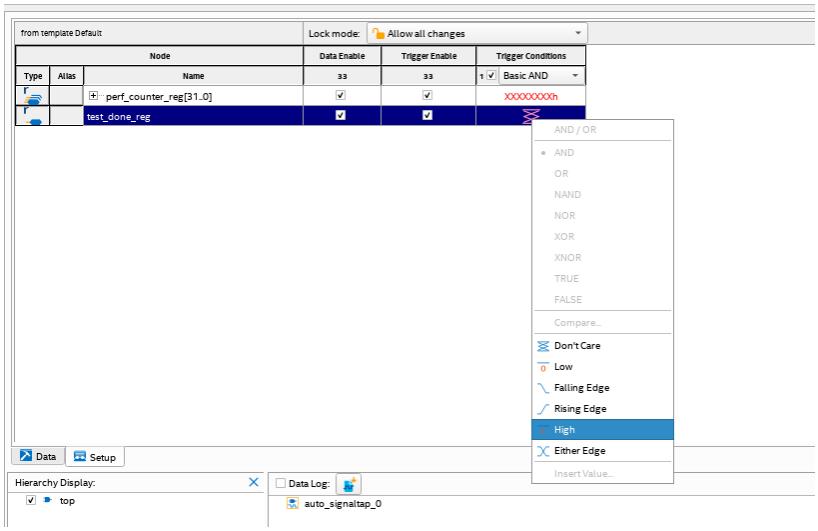
16. Click the start button on “Analysis & Synthesis”
17. Dump out performance counter in SignalTap
    a. Tools -> Signal Tap Logic Analyzer
    b. Quick Start Group -> Default(default selection) -> Create
    c. Signal Configuration -> Clock -> Search “iopll_clk” in “top” module -> add the signal from “Matching Nodes” window to “Nodes Found” window -> OK



d. Double click the blank area in the “setup” window -> Search for “perf_counter_reg” and “test_done_reg” -> add the signals from the “Matching Nodes” window to “Nodes Found” window -> Insert



e. In the “setup” window, right click the “test_done_reg” signal in the “Trigger conditions” column -> click “High”



f. File -> Save -> Click Yes in the popped up window asking if to enable the stp file for the current project -> Exit SignalTap

18. Processing -> Start Compilation. Bitstream will be generated.
19. Use Programmer to program the FPGA using the bitstream. the LEDs on the board will show the test status (i.e., 000: idle, 001: Running, 010: Success, 100: Fail). Furthermore, SignalTap can be used to see the performance counter value of the running design.