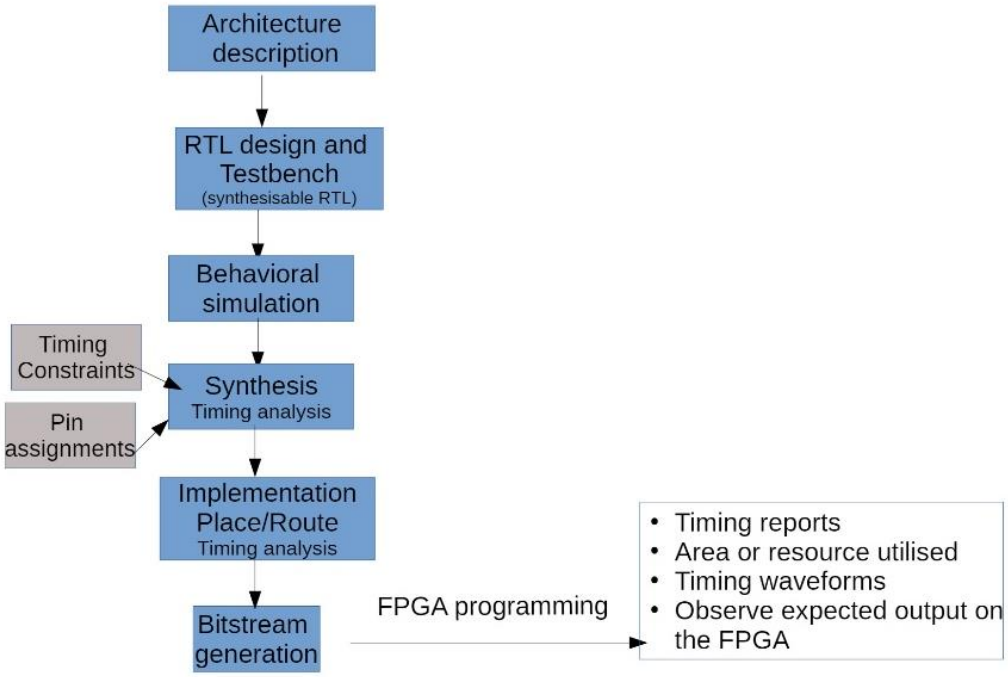
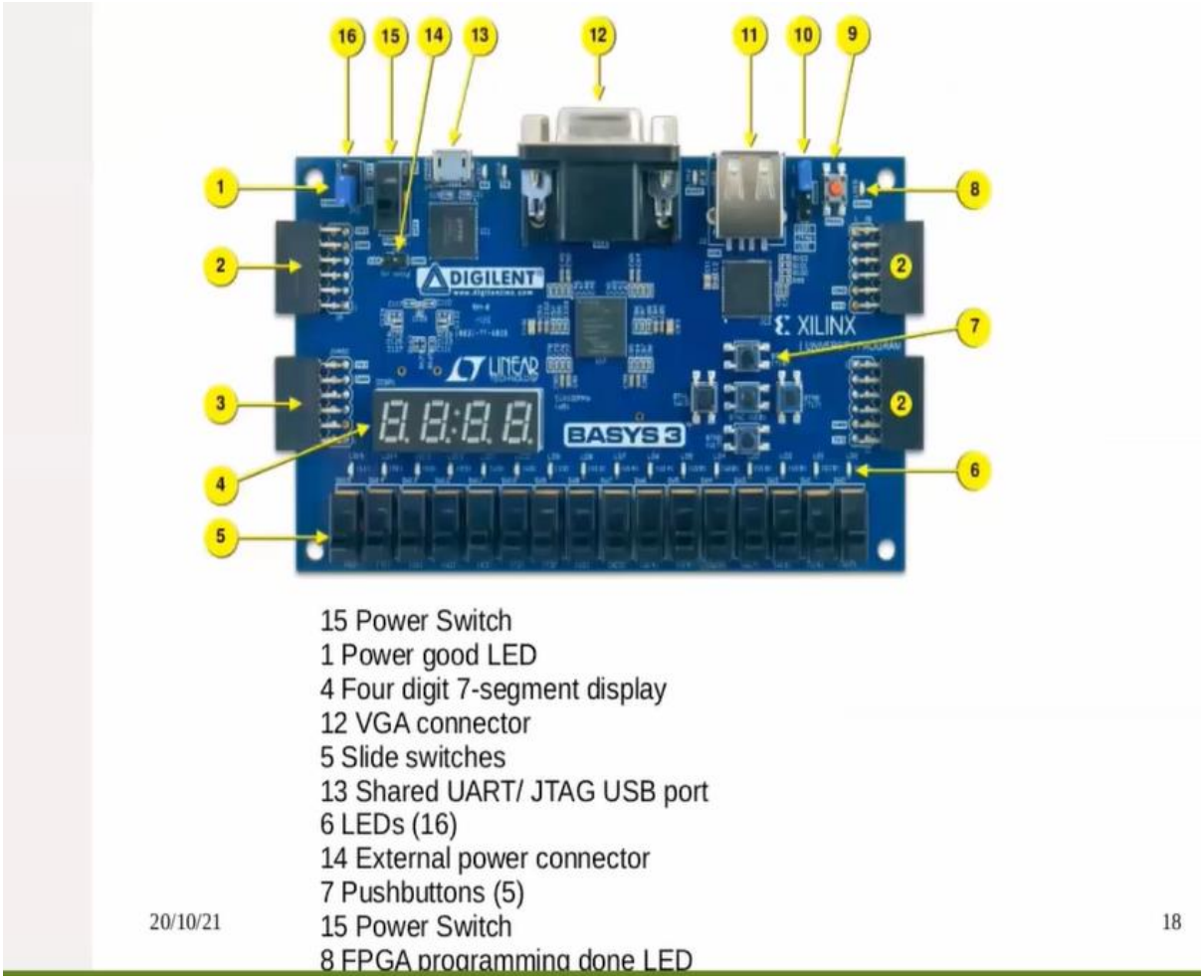
FPGA - Fabric, Design and Architecture

This repository contains all the information studied and created during the FPGA - Fabric, Design and Architecture workshop.

Vivado / Open FPGA
Day 1
FPGA Introduction
FPGA Design Flow



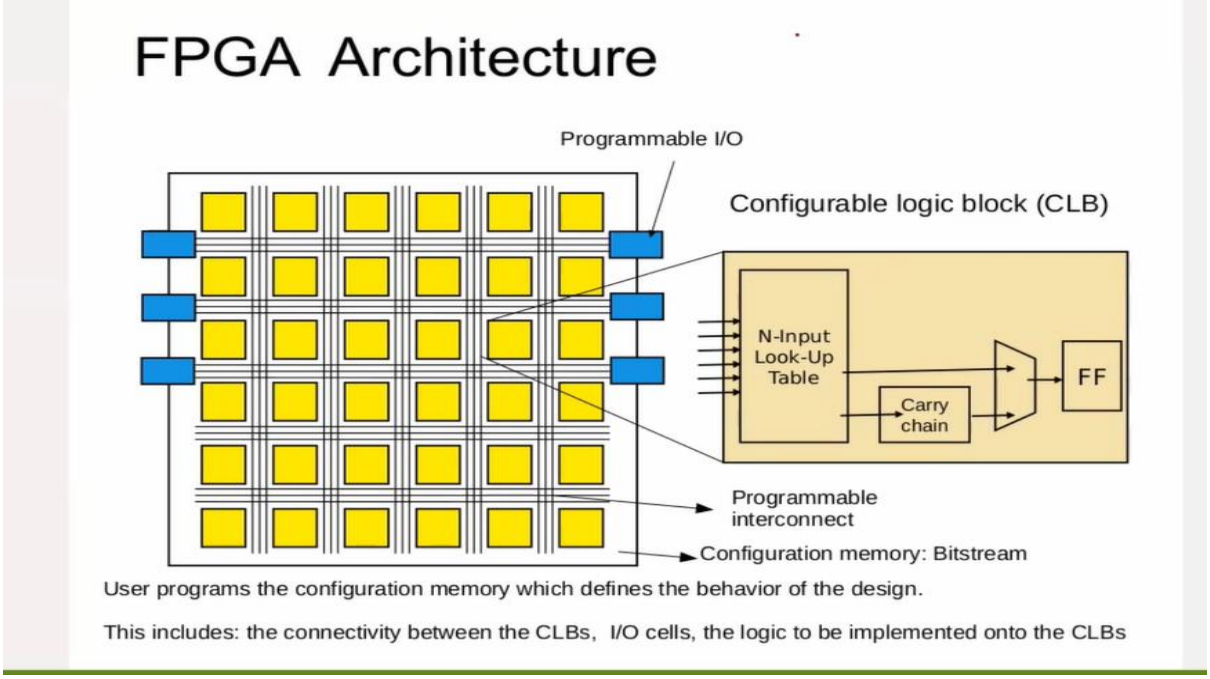
Design of a 4-bit counter through Vivado on Basys3 FPGA



Basys 3 Board for Implementation of Design
Use of Virtual Input/Output (VIO)
Flow of VIO

Day 1 - Exploring FPGA Basics and Vivado

FPGA Architecture



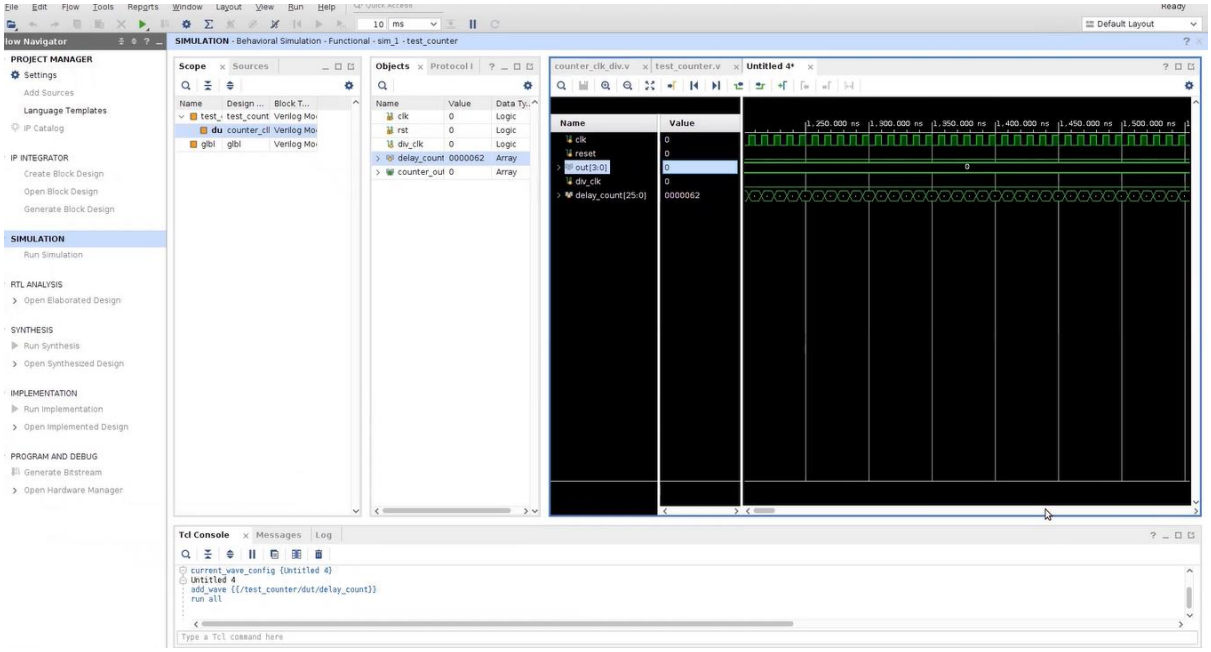
Counter Example in Vivado

A 4-bit up counter is being used for exploring the Vivado tool and OpenFPGA. Below mentioned the RTL for the counter modules that is being used

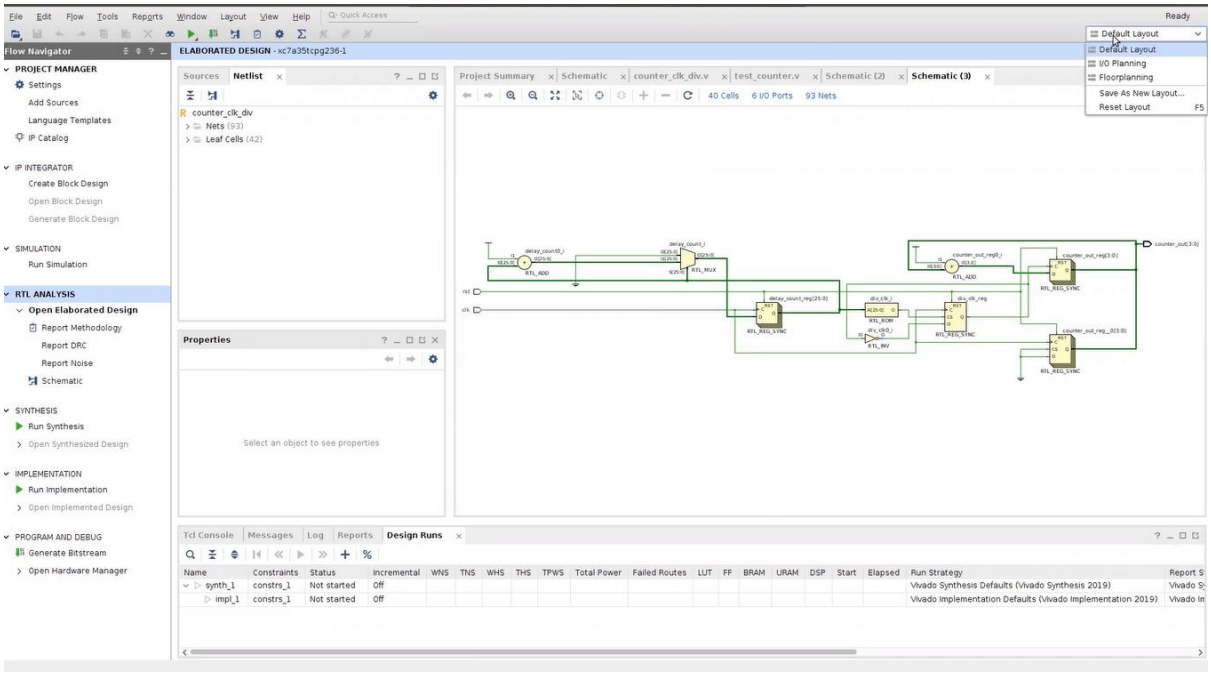
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Description: 4 bit counter with source clock (100MHz) division.

module counter_clk_div(clk,rst,counter_out);

    input clk,rst;
    reg div_clk;
    reg [25:0] delay_count;
    output reg [3:0] counter_out;

    //////////clock division block////////////////////
```

```
    always @(posedge clk) begin

        if(rst) begin
            delay_count<=26'd0;
            div_clk <= 1'b0; //initialise div_clk
        end
        else
            if(delay_count==26'd212) begin
                delay_count<=26'd0; //reset upon reaching the max value
                div_clk <= ~div_clk;  //generating a slow clock
            end
            else begin
                delay_count<=delay_count+1;
            end
        end
    end

    //////////////4 bit counter block//////////////////
    always @(posedge div_clk) begin

        if(rst) begin
            counter_out<=4'b0000;
        end
        else begin
            counter_out<= counter_out+1;
        end
    end

endmodule
```

Counter Simulation and Elaboration



The snippet below is the schematic of the counter design after elaboration.



Constraints

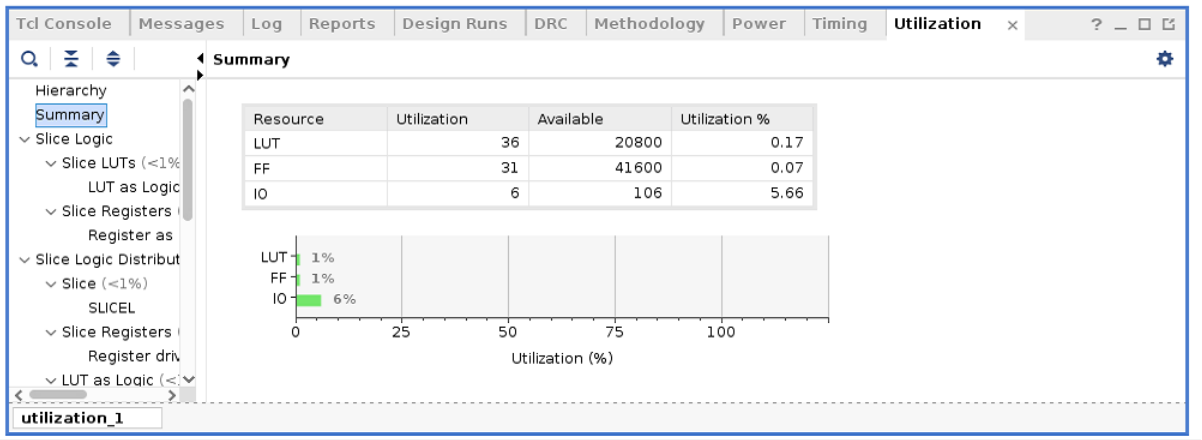
Constraints in simple are the specifications of your design like timing specifications, ports declaration, input/output delays, etc.

Bitstream

A bitstream is a binary sequence that comprises a sequence of bits. These are used in FPGA applications for programming purposes and to establish communication channels. FPGA bitstream is a file containing the programming data associated with your FPGA chip.

Counter Timing, Power and Area

Implementation of a design also gives details like the timing summary, device utilization, power analysis, etc. The below snippets show the brief timing sumary, implementation and power analysis of the up-counter design.

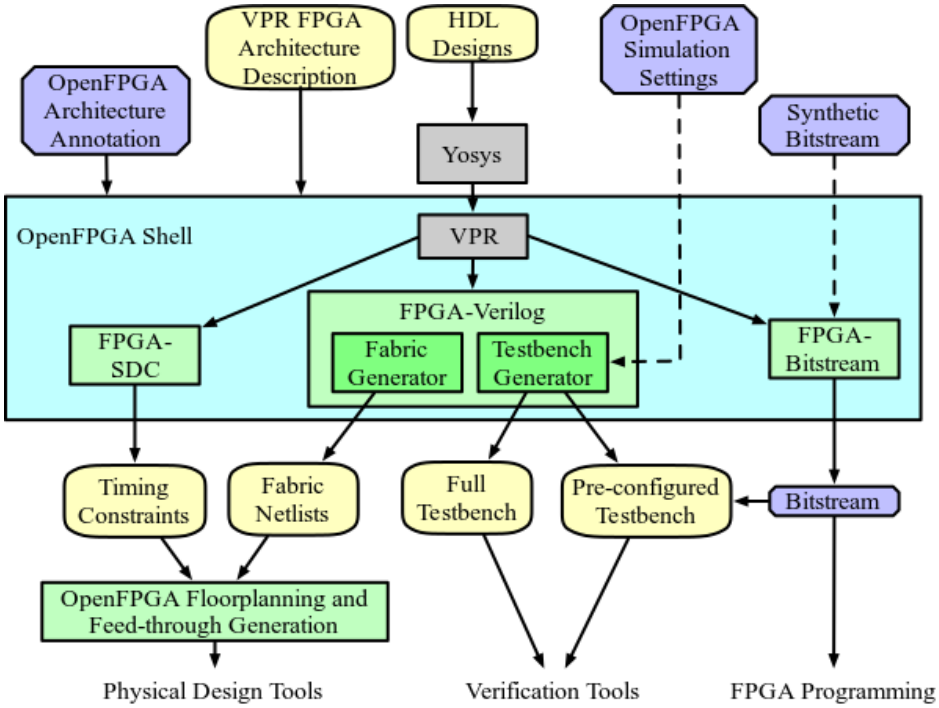


Day 2 - Exploring OpenFPGA, VPR and VTR

Introduction To OpenFPGA

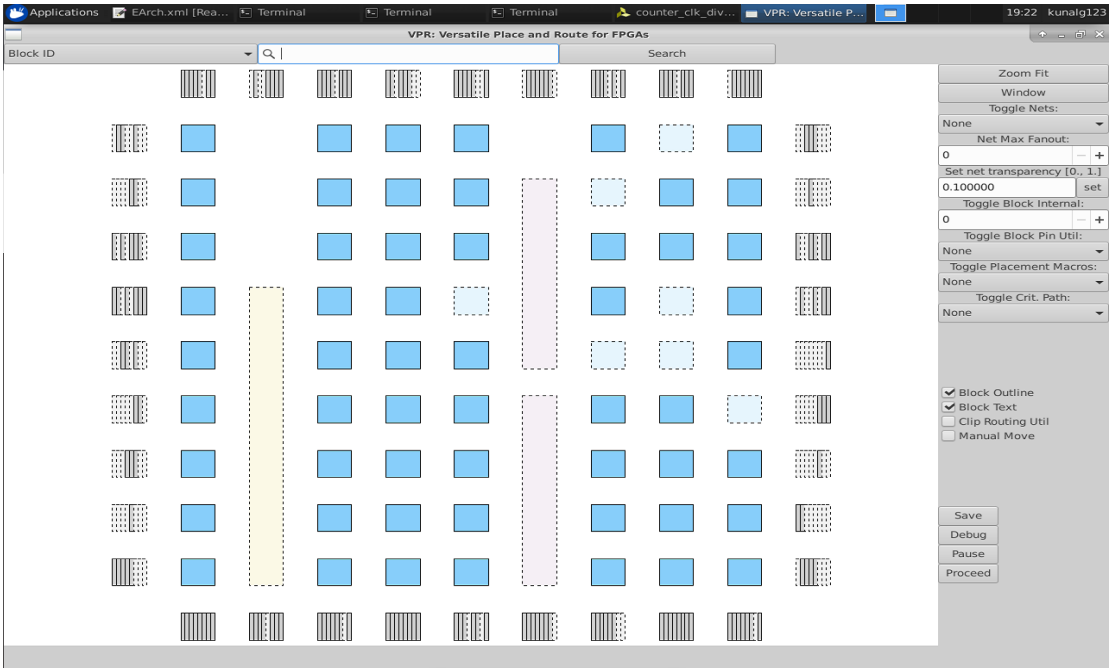
The OpenFPGA framework is the first open-source FPGA IP generator which supports highly-customizable homogeneous FPGA architectures. OpenFPGA provides a full set of EDA support for customized FPGAs, including Verilog-to-bitstream generation and self-testing verification. OpenFPGA targets to democratizing FPGA technology and EDA techniques, with agile prototyping approaches and constantly evolving EDA tools for chip designers and researchers.

Some key features of OpenFPGA are: - Use of Automation Techniques - Reduction of FPGA developement cycle to few days - Provides open source design tools



VPR

VPR (Versatile Place and Route) is an open source academic CAD tool designed for the exploration of new FPGA architectures and CAD algorithms, at the packing, placement and routing phases of the CAD flow. As input, VPR takes a description of an FPGA architecture along with a technology-mapped user circuit. It then performs packing, placement, and routing to map the circuit onto the FPGA. The output of VPR includes the FPGA configuration needed to implement the circuit and statistics about the final mapped design (eg. critical path delay, area, etc).



o invoke VPR from terminal:

```
$VTR_ROOT/vpr/vpr \
$VTR_ROOT/vtr_flow/arch/timing/EArch.xml \
<blif-file-path \
--route_chan_width 100 \
--disp on
```

The basic VPR flow involves below mentioned steps:

- Packing - combinines primitives into complex blocks
- Placment - places complex blocks within the FPGA grid
- Routing - determines interconnections between blocks
- Analysis - analyzes the implementation

VTR

The Verilog to Routing (VTR) project provides open-source CAD tools for FPGA architecture and CAD research. The VTR design flow takes as input a Verilog description of a digital circuit, and a description of the target FPGA architecture.

To invoke VTR from command-line:

```
```

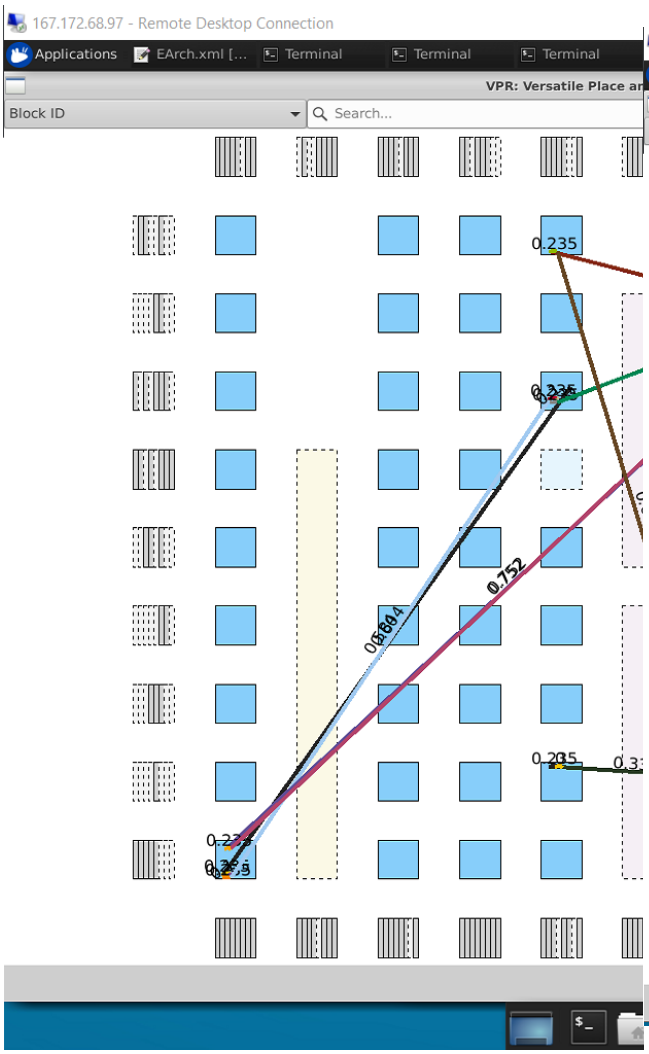
```
$VTR_ROOT/vtr_flow/scripts/run_vtr_flow.py \
$VTR_ROOT/doc/src/<verilog-file-path>
$VTR_ROOT/vtr_flow/arch/timing/EArch.xml \
-temp_dir . \
--route_chan_width 100

```

VTR Flow

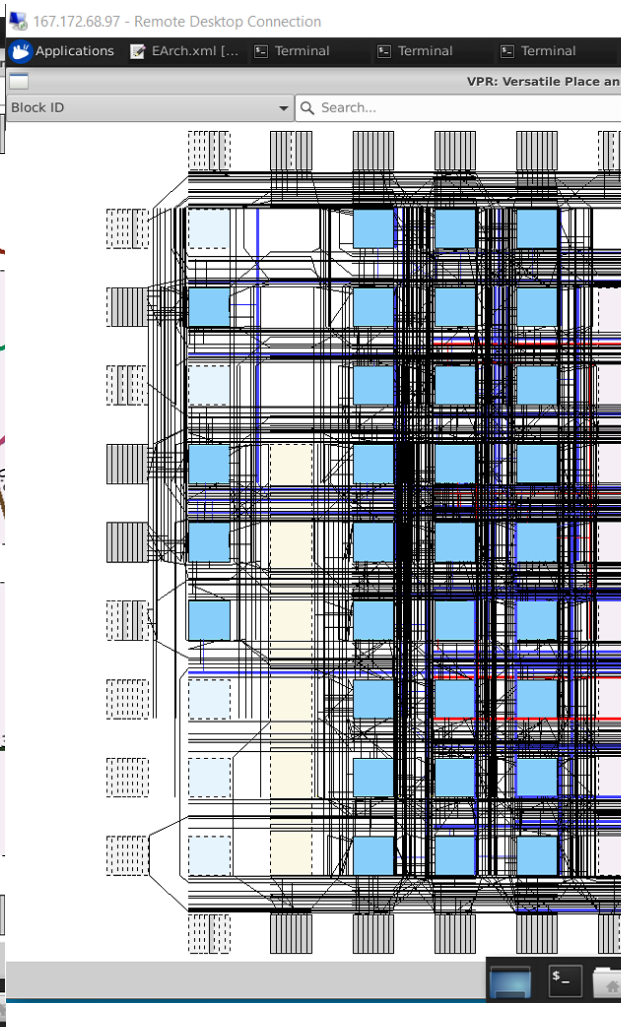
The basic VTR flow perfoms:

- Elaboration & Synthesis (ODIN II)
- Logic Optimization & Technology Mapping (ABC)
- Packing, Placement, Routing & Timing Analysis (VPR)

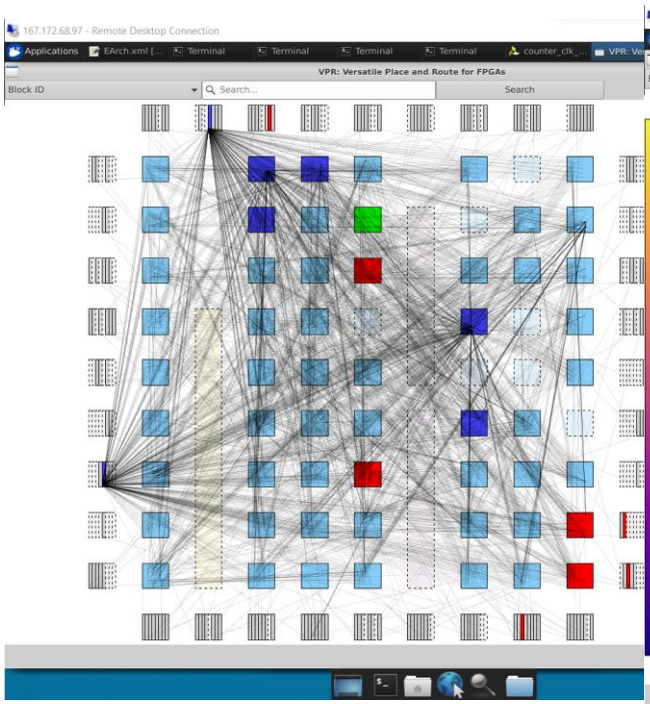
The snippets mentioned below show some of the stages from the VTR flow



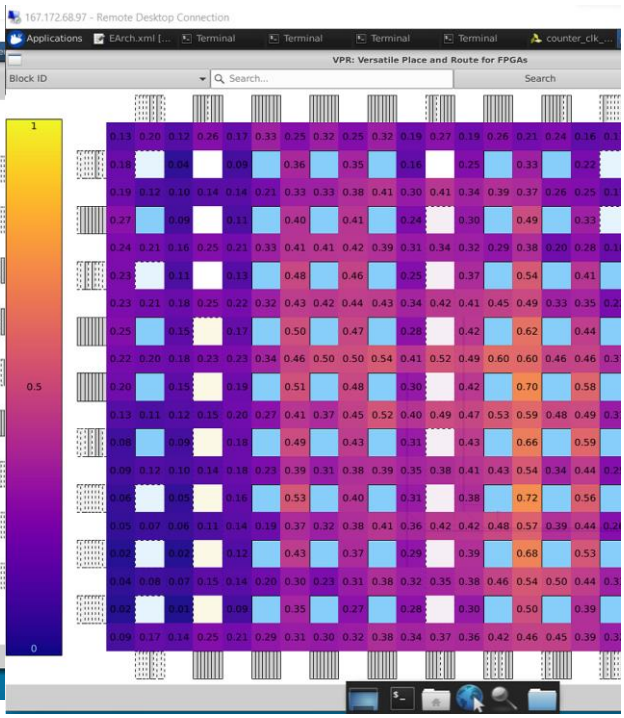
Critical Paths



Nets



Logical Connections



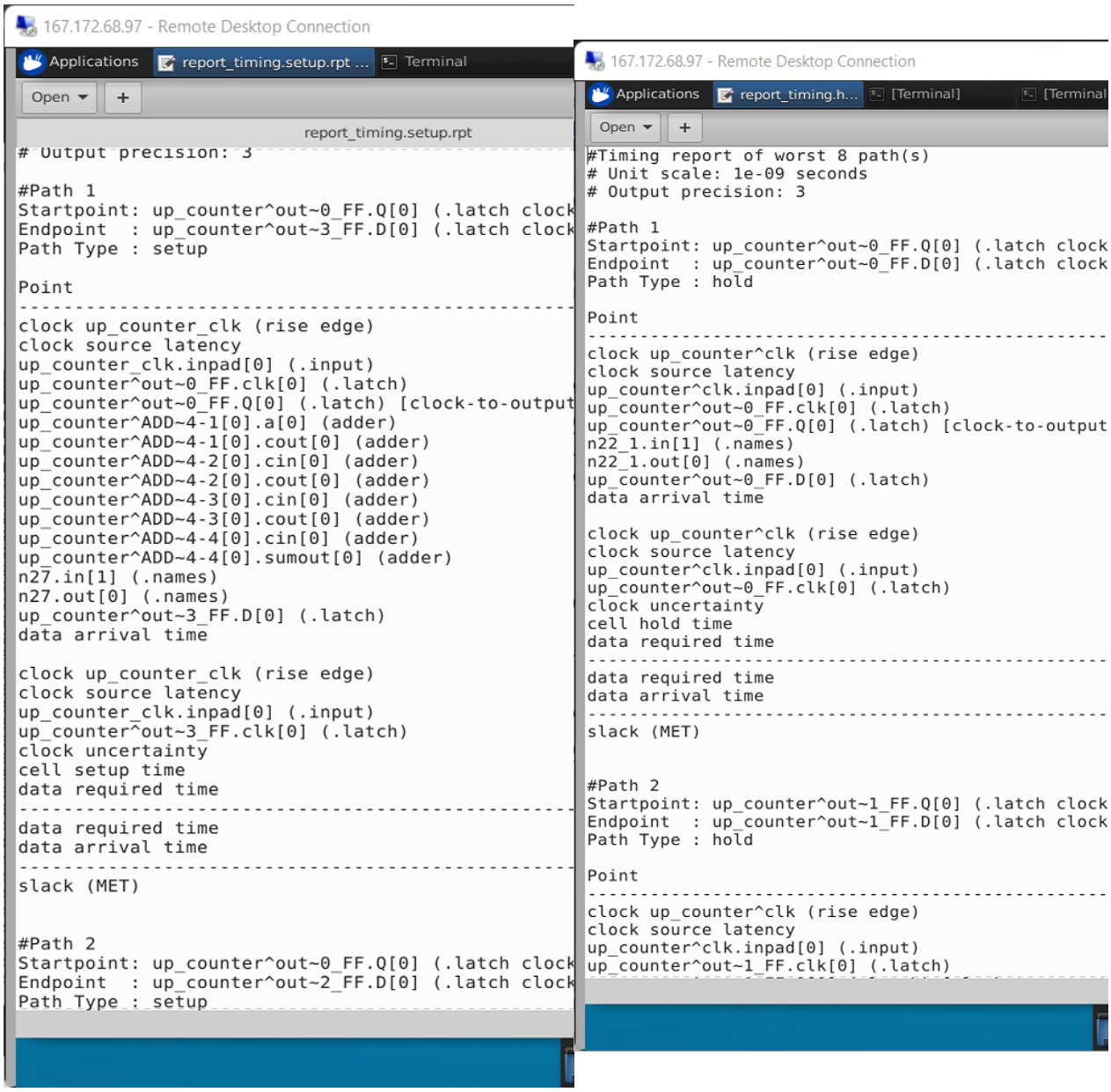
Routing Utilizations

Timing Analysis VTR Flow

In order to perform timing analysis, a constraint file needs to be created. This constraint file is provided as an input to tool. To perform timing analysis from command-line, below mentioned switch should be enabled.

```
 ## .sdc constraint file is required
 --sdc_file <sdc-file-path>
```

The snippets below show the setup and hold timing reports generated by the tool.

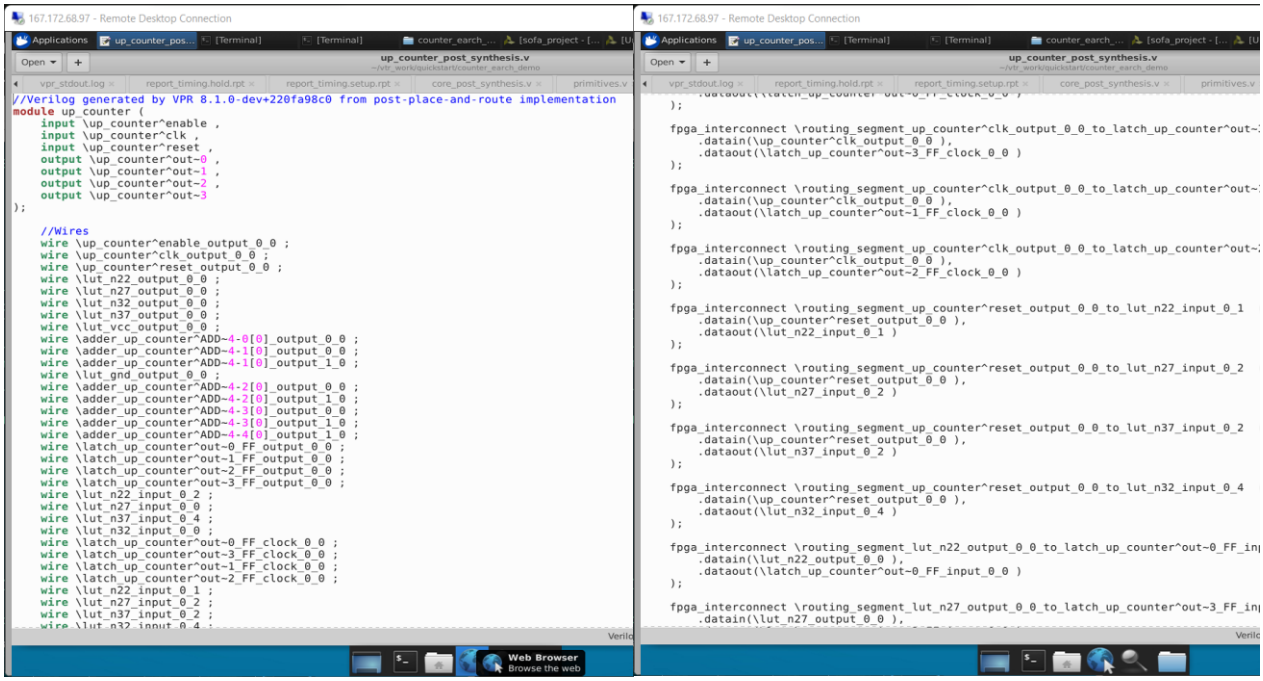


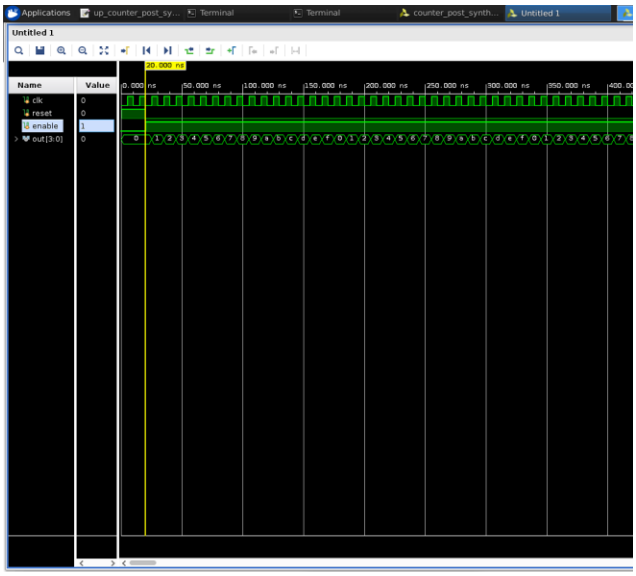
Post Synthesis Simulation

Post Synthesis simulation in VTR flow is same as Post Implementation simulations in general. To generate the post synthesis netlist for simulation, the below mentioned switch should be enabled in during the VPR stage in VTR flow.

```
 ## To generate post synthesis netlist
 --gen_post_synthesis_netlist on
```

The snippets below show the post synthesis netlist that is generated by the tool and the a behavioural simulation of the netlist in Vivado



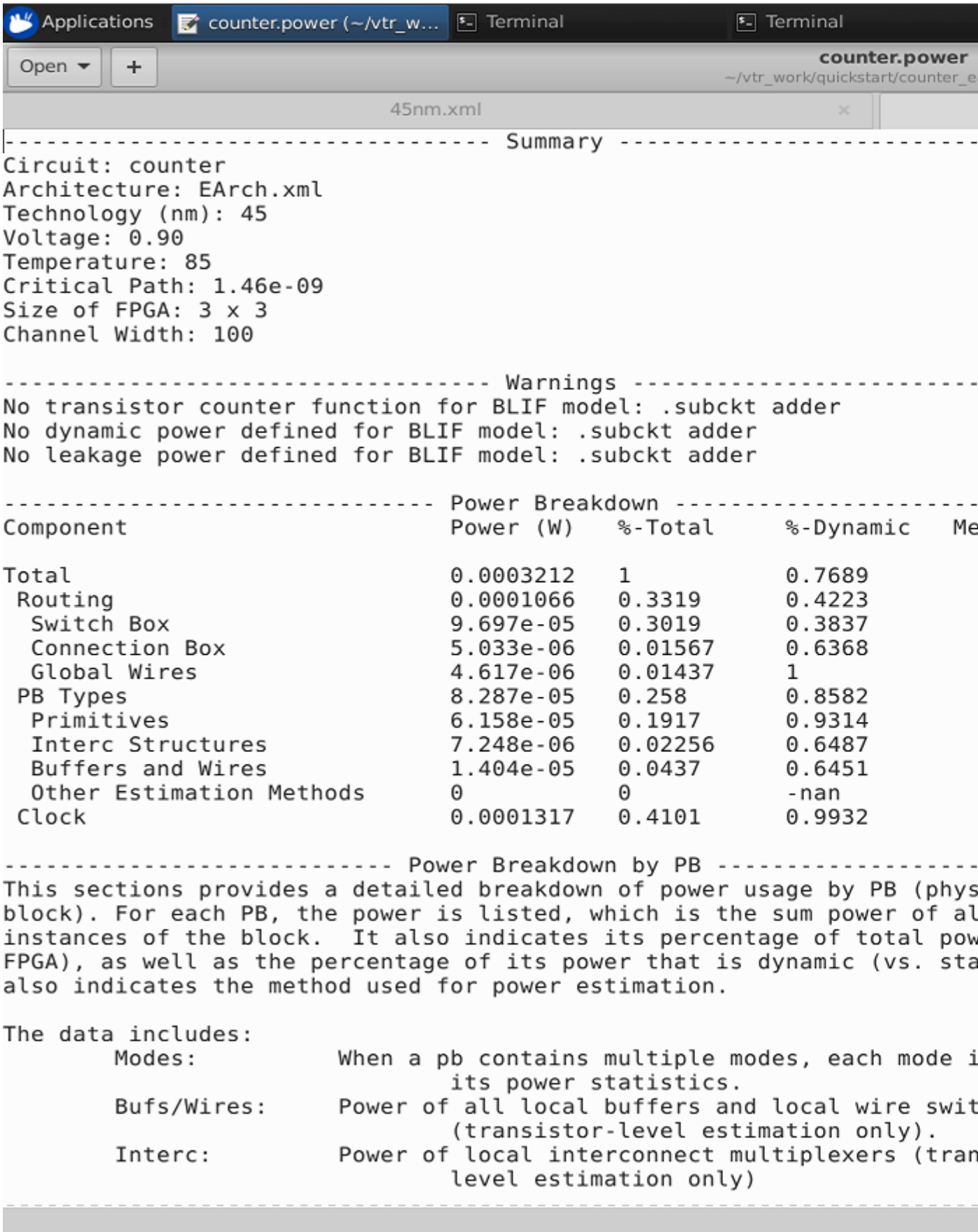


Power Analysis VTR

VTR provides a option to perform power analysis over the design. To enable power analysis in command-line, the switch mentioned below should be used.

```
## For power analysis
-power -cmos_tech
$VTR_ROOT/vtr_flow/tech/PTM_45nm/45nm.xml
```
The snippet below shows a brief summary of the power analysis report generated by the VTR

flow.


Applications
counter.power (~/vtr_w...
Terminal
Terminal
Open
+
counter.power
~/vtr_work/quickstart/counter_e
45nm.xml
---------- Summary ----------
Circuit: counter
Architecture: EArch.xml
Technology (nm): 45
Voltage: 0.90
Temperature: 85
Critical Path: 1.46e-09
Size of FPGA: 3 x 3
Channel Width: 100
---------- Warnings ----------
No transistor counter function for BLIF model: .subckt adder
No dynamic power defined for BLIF model: .subckt adder
No leakage power defined for BLIF model: .subckt adder
---------- Power Breakdown ----------
Component Power (W) %-Total %-Dynamic Me
Total 0.0003212 1 0.7689
Routing 0.0001066 0.3319 0.4223
Switch Box 9.697e-05 0.3019 0.3837
Connection Box 5.033e-06 0.01567 0.6368
Global Wires 4.617e-06 0.01437 1
PB Types 8.287e-05 0.258 0.8582
Primitives 6.158e-05 0.1917 0.9314
Interc Structures 7.248e-06 0.02256 0.6487
Buffers and Wires 1.404e-05 0.0437 0.6451
Other Estimation Methods 0 0 -nan
Clock 0.0001317 0.4101 0.9932
---------- Power Breakdown by PB ----------
This sections provides a detailed breakdown of power usage by PB (phys
block). For each PB, the power is listed, which is the sum power of al
instances of the block. It also indicates its percentage of total pow
FPGA), as well as the percentage of its power that is dynamic (vs. sta
also indicates the method used for power estimation.
The data includes:
Modes: When a pb contains multiple modes, each mode i
its power statistics.
Bufs/Wires: Power of all local buffers and local wire swit
(transistor-level estimation only).
Interc: Power of local interconnect multiplexers (tran
level estimation only)

Day 3 - RISCV Core Programming Using Vivado

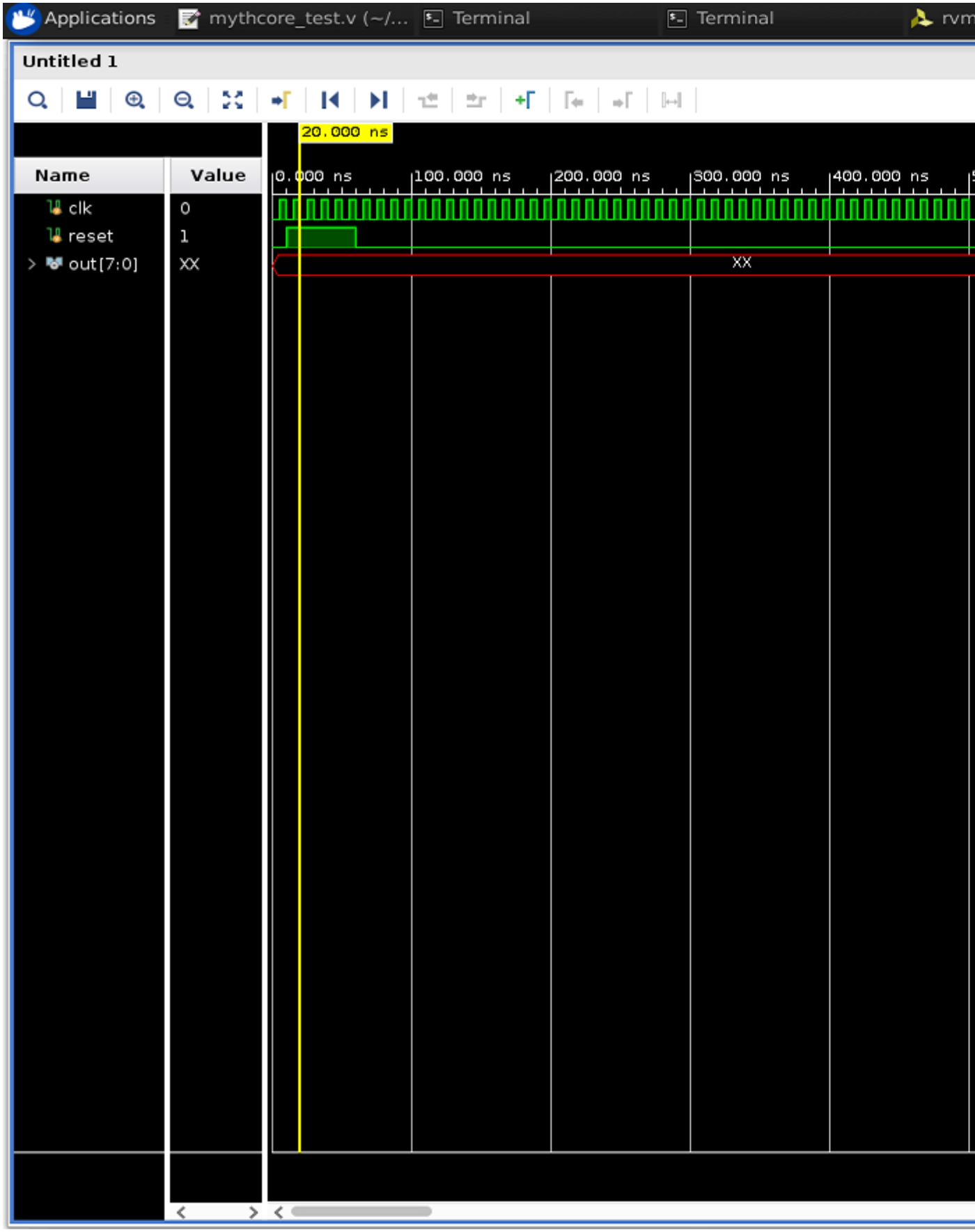
A 4-stage pipelined RISC-V core, named RVMYTH, is used in the repository. A complete RTL to Bitstream flow is implemented over the RVMYTH core. The Core is initially developed in High-level language named TL-Verilog and finally compiled to Verilog HDL.

RTL To Synthesis

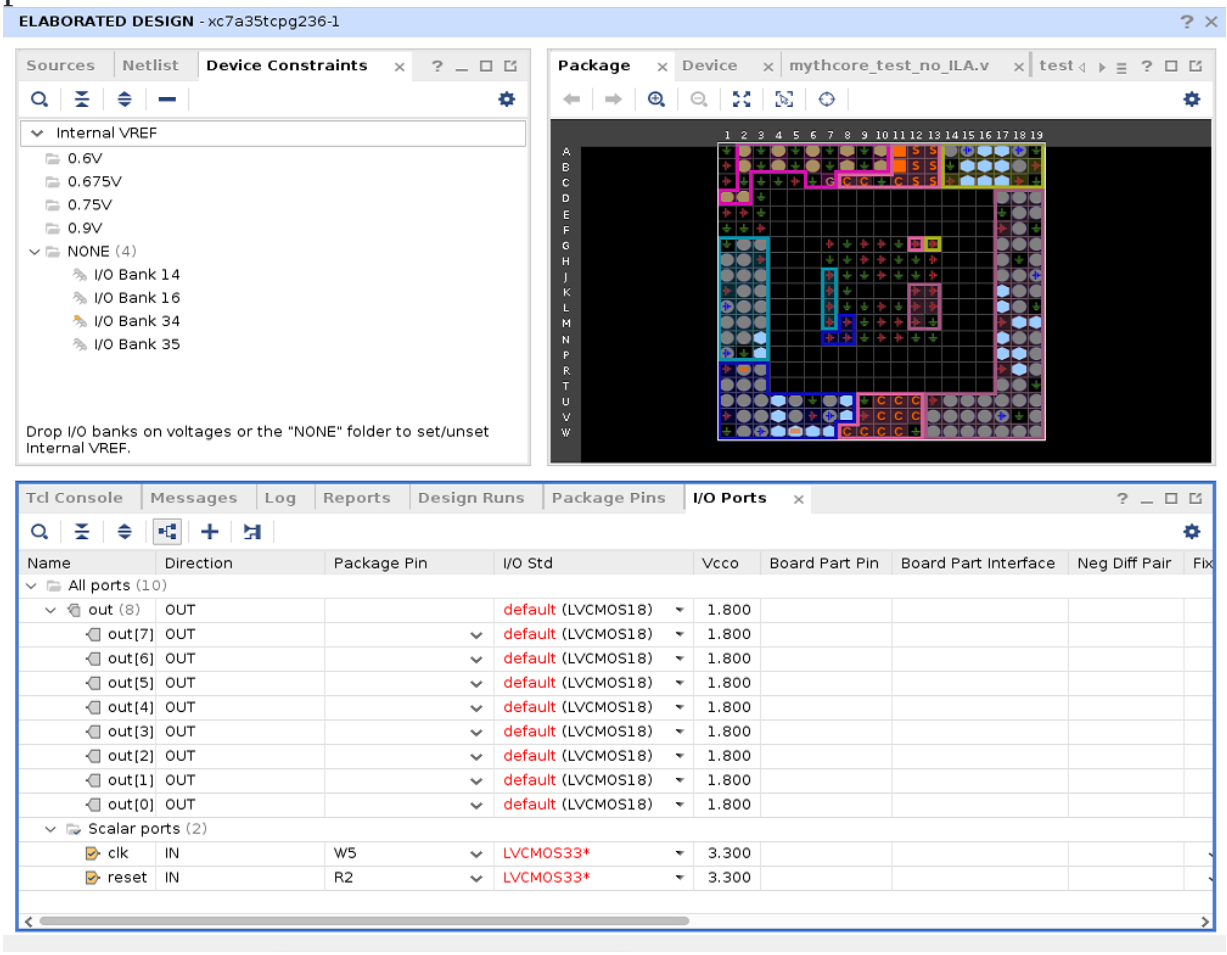
The RISC-V RTL consists of multiple blocks/modules. Some of them are:

- Instruction memory
- Data memory
- ALU
- I/O ports

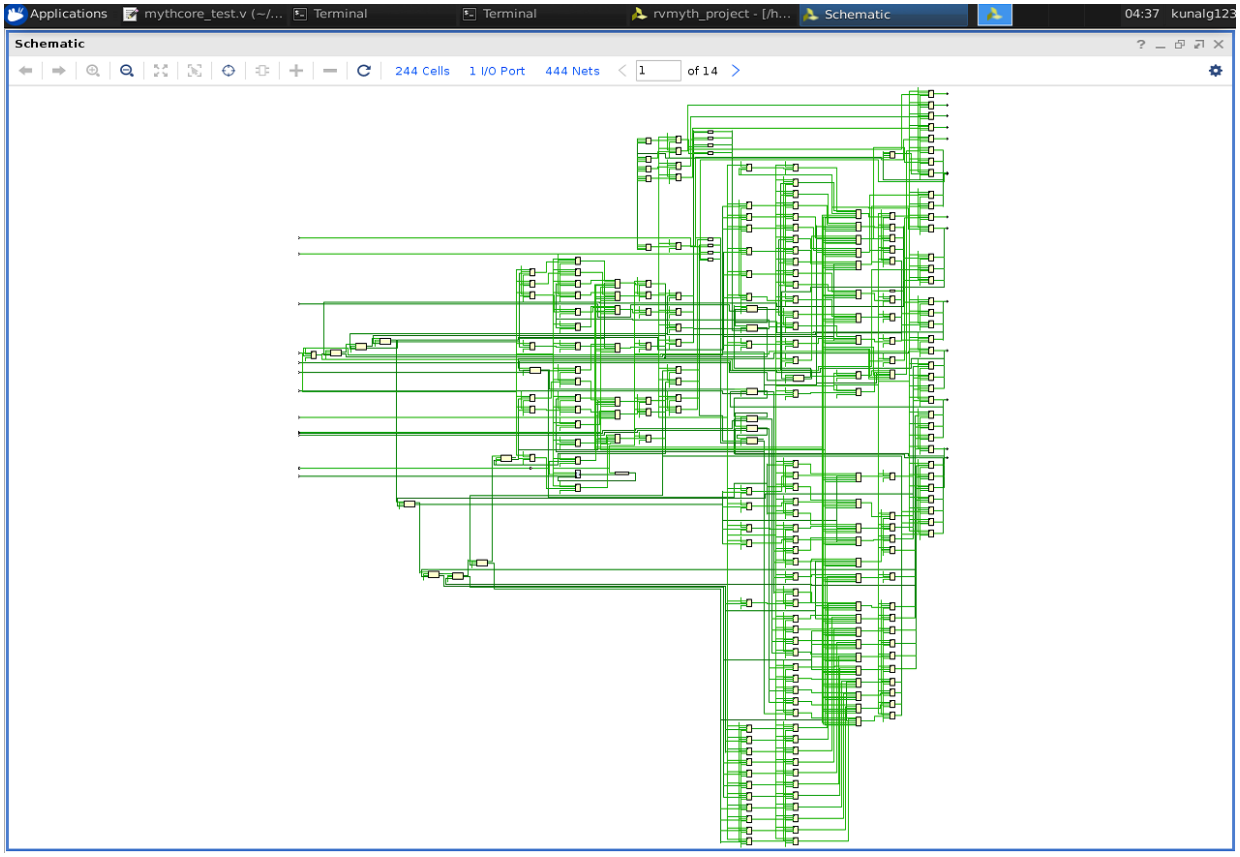
The snippets below shows the behavioural simulation of the RVMYTH RISC-V core in vivado simulator. The instruction memory contains instructions for addition of integers from 1 to 9. The output signal in the waves display the final sum.

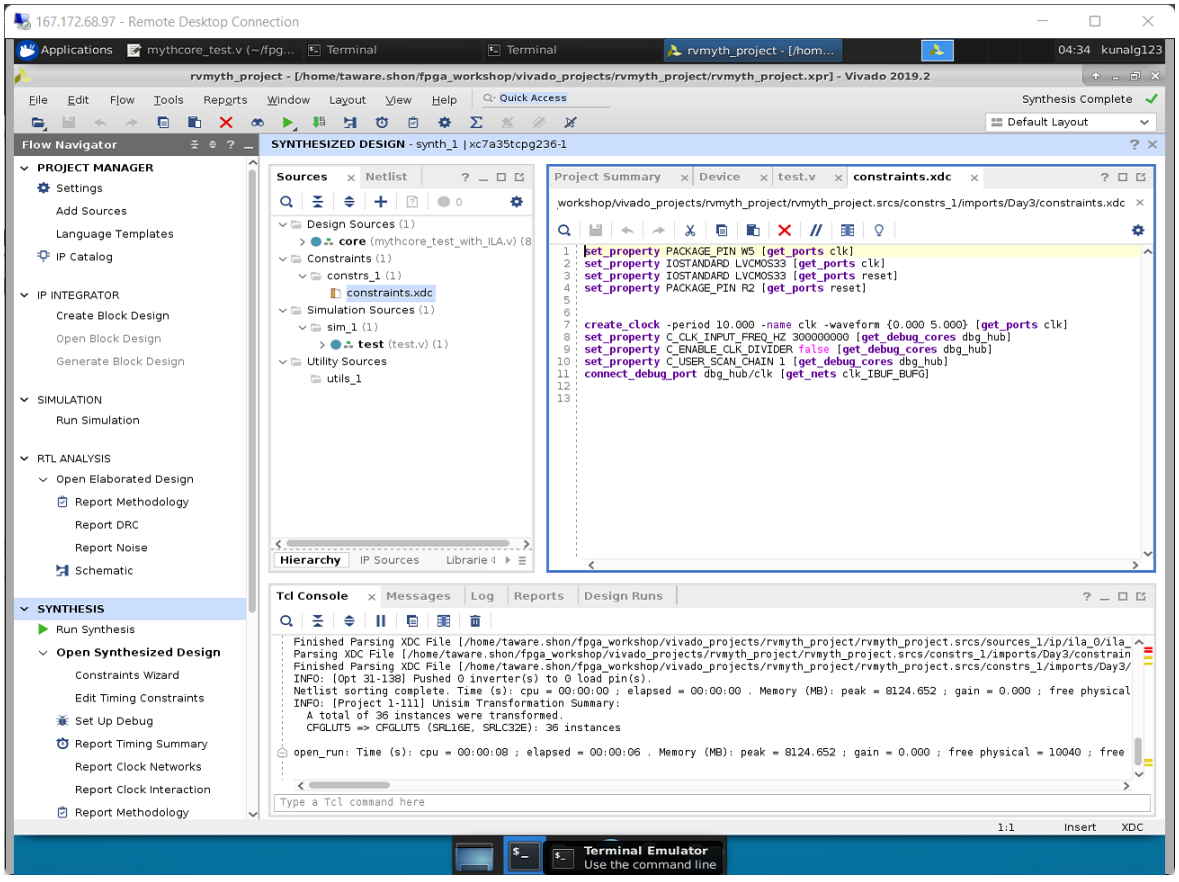

Applications
mythcore_test.v (~/...
Terminal
Terminal
rvm
Untitled 1
20.000 ns
Name
Value
0.000 ns
100.000 ns
200.000 ns
300.000 ns
400.000 ns
clk
0
reset
1
out[7:0]
XX
XX

The snippet below shows the elaboration stage where the FPGA pins are mapped to the RTL input/output
ports.



The design is synthesized in Vivado tool for Basys3 FPGA, along with some constraints. The below snippet show the schematic of design after synthesis and the constraints used for synthesis.

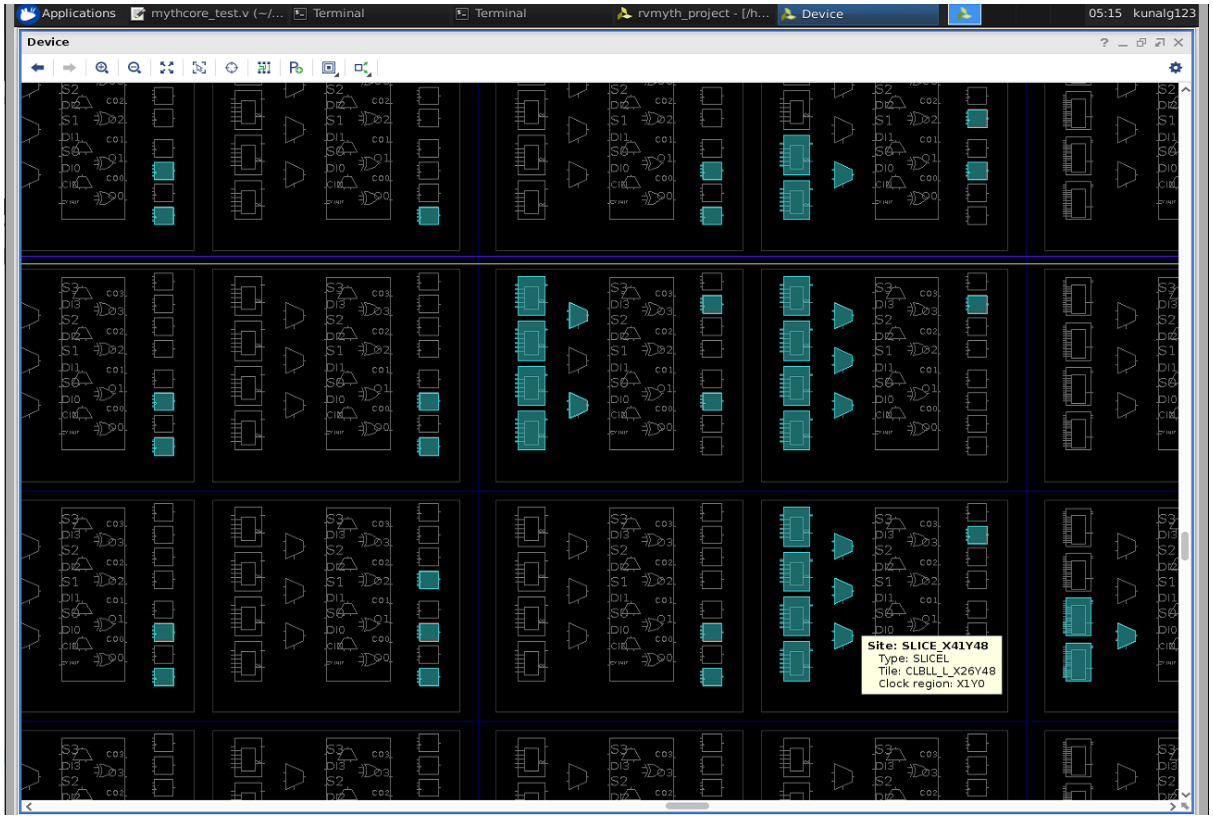

Applications
mythcore_test.v (~/...
Terminal
Terminal
rvmyth_project - [/h...
Schematic
04:37 kunalg123
Schematic
244 Cells
1 I/O Port
444 Nets
1
of 14


167.172.68.97 - Remote Desktop Connection
Applications
mythcore_test.v (~/fpg...
Terminal
Terminal
rvmyth_project - [/hom...
04:34 kunalg123
rvmyth_project - [/home/taware.shon/fpga_workshop/vivado_projects/rvmyth_project/rvmyth_project.xpr] - Vivado 2019.2
File Edit Flow Tools Reports Window Layout View Help
Quick Access
Synthesis Complete
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
SYNTHESIZED DESIGN - synth_1 | xc7a35tcpg236-1
Sources
Netlist
Design Sources (1)
core (mythcore_test_with_ILA.v) (8
Constraints (1)
constrs_1 (1)
constraints.xdc
Simulation Sources (1)
sim_1 (1)
test (test.v) (1)
Utility Sources
utils_1
Hierarchy
IP Sources
Project Summary
Device
test.v
constraints.xdc
workshop/vivado_projects/rvmyth_project/rvmyth_project.srcs/constrs_1/imports/Day3/constraints.xdc
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports reset]
set_property PACKAGE_PIN R2 [get_ports reset]
create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]
set_property C_CLK_INPUT_FREQ_HZ 300000000 [get_debug_cores dbg_hub]
set_property C_ENABLE_CLK_DIVIDER false [get_debug_cores dbg_hub]
set_property C_USER_SCAN_CHAIN 1 [get_debug_cores dbg_hub]
connect_debug_port dbg_hub/clk [get_nets clk_IBUF_BUFG]
Tcl Console
Messages
Log
Reports
Design Runs
Finished Parsing XDC File [/home/taware.shon/fpga_workshop/vivado_projects/rvmyth_project/rvmyth_project.srcs/sources_1/ip/ila_0/ila_
Parsing XDC File [/home/taware.shon/fpga_workshop/vivado_projects/rvmyth_project/rvmyth_project.srcs/constrs_1/imports/Day3/constrain
Finished Parsing XDC File [/home/taware.shon/fpga_workshop/vivado_projects/rvmyth_project/rvmyth_project.srcs/constrs_1/imports/Day3/
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 8124.652 ; gain = 0.000 ; free physical
INFO: [Project 1-111] Unisim Transformation Summary:
A total of 36 instances were transformed.
CFGLUT5 => CFGLUT5 (SRL16E, SRLC32E): 36 instances
open_run: Time (s): cpu = 00:00:08 ; elapsed = 00:00:06 . Memory (MB): peak = 8124.652 ; gain = 0.000 ; free physical = 10040 ; free
Type a Tcl command here
1:1
Insert
XDC
Terminal Emulator
Use the command line

Synthesis To Bitstream

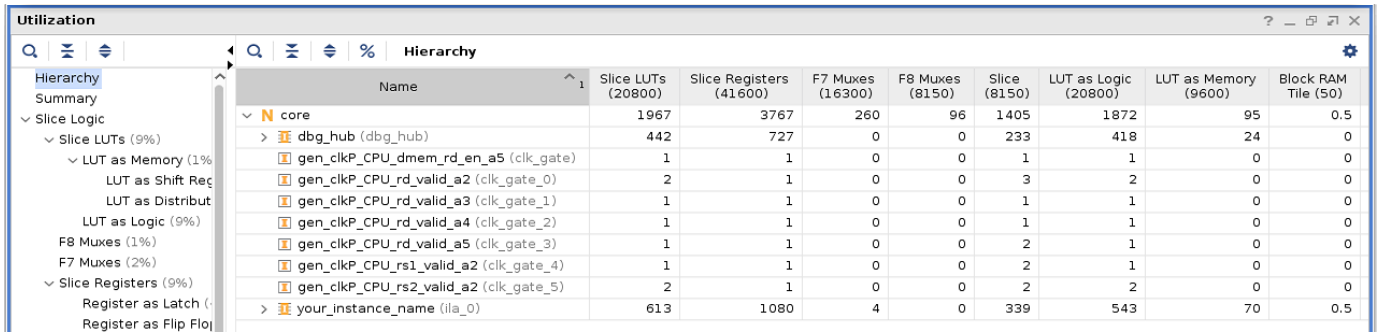
During implementation, the synthesis design is traslated to a logic design file and mapped to small blocks and sub-blocks that can fit in FPGA CLBs. These blocks are then placed and routed in a optimized way.

The snippet below shows a small part of the implemented design, the used LUTs, MUX and CLBs.

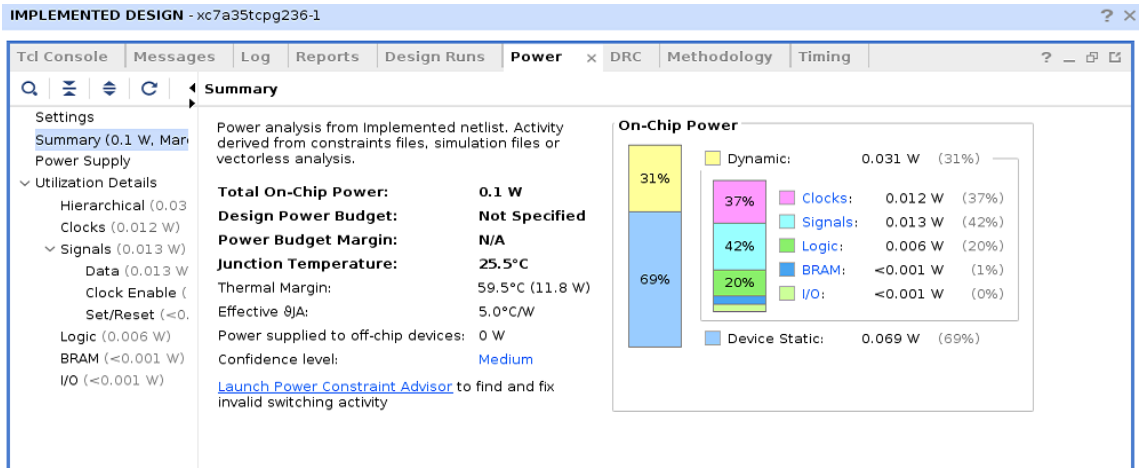


The snippet below shows the timing summary for the RVMYTH implemented core.

The snippet below shows the Device Utilization summary for the RVMYTH implemented core.



The snippet below shows the power analysis for the RVMYTH implemented core.



Day 4 - Introduction To SOFA FPGA Fabric

SOFA (Skywater Opensource FPGAs) are a series of open-source FPGA IPs designed using open-source Skywater 130nm PDKs and OpenFPGA framework.
The FPGA IP design used in this repository is FPGA1212_QLSOFA_HD_PNR which has 50MHz of maximum operating speed, 1152 LUTs, 2304 Flip-flops, 1152 soft adders. The complete design is used over OpenFPGA framework and the various reports are generated.

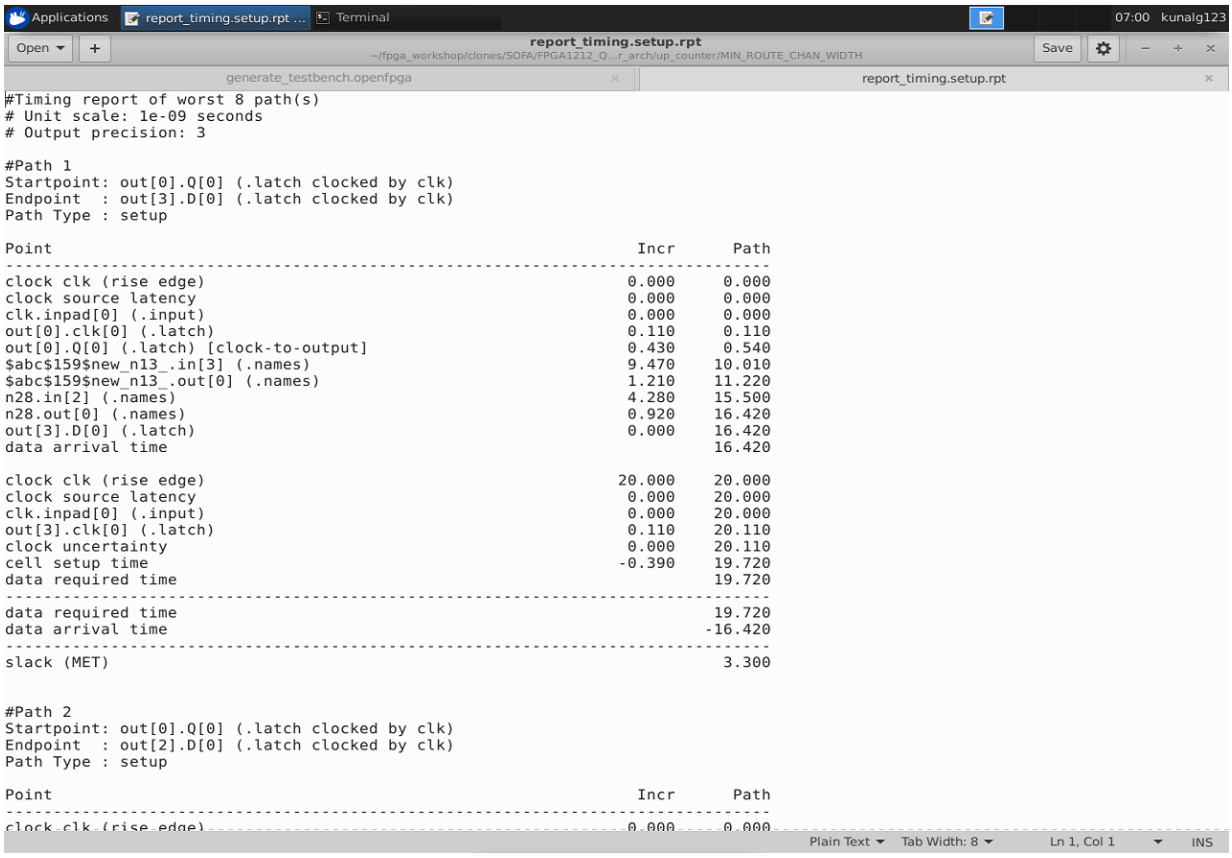
SOFA Counter Area

The snippet below shows the utilization of the counter in SOFA FPGA Fabric.

```
Circuit Statistics:
  Blocks: 17
    .input :       3
    .latch :       4
    .output:       4
    4-LUT  :       6
  Nets  : 13
    Avg Fanout:     2.5
    Max Fanout:     5.0
    Min Fanout:     1.0
  Netlist Clocks: 1
# Build Timing Graph
  Timing Graph Nodes: 45
  Timing Graph Edges: 60
  Timing Graph Levels: 8
# Build Timing Graph took 0.00 seconds (max_rss 16.7 MiB, delta_rss +0.0 MiB)
Netlist contains 1 clocks
  Netlist Clock 'clk' Fanout: 4 pins (8.9%), 4 blocks (23.5%)
```

SOFA Counter Timing

The snippet below shows the setup and hold timing summary the counter in SOFA FPGA Fabric.



Applications report_timing.setup.rpt ... Terminal 07:00 kunalg123

Open + report_timing.setup.rpt ~/fpga_workshop/clones/SOFA/FPGA1212_Q...r_arch/up_counter/MIN_ROUTE_CHAN_WIDTH Save

generate_testbench.openfpga report_timing.setup.rpt

```
#Timing report of worst 8 path(s)
# Unit scale: 1e-09 seconds
# Output precision: 3

#Path 1
Startpoint: out[0].Q[0] (.latch clocked by clk)
Endpoint  : out[3].D[0] (.latch clocked by clk)
Path Type : setup

Point                                                                  Incr      Path
-------------------------------------------------------------------------------------
clock clk (rise edge)                                                 0.000     0.000
clock source latency                                                  0.000     0.000
clk.inpad[0] (.input)                                                 0.000     0.000
out[0].clk[0] (.latch)                                                0.110     0.110
out[0].Q[0] (.latch) [clock-to-output]                                0.430     0.540
$abc$159$new_n13_.in[3] (.names)                                      9.470    10.010
$abc$159$new_n13_.out[0] (.names)                                     1.210    11.220
n28.in[2] (.names)                                                    4.280    15.500
n28.out[0] (.names)                                                   0.920    16.420
out[3].D[0] (.latch)                                                  0.000    16.420
data arrival time                                                              16.420

clock clk (rise edge)                                                20.000    20.000
clock source latency                                                  0.000    20.000
clk.inpad[0] (.input)                                                 0.000    20.000
out[3].clk[0] (.latch)                                                0.110    20.110
clock uncertainty                                                     0.000    20.110
cell setup time                                                      -0.390    19.720
data required time                                                             19.720
-------------------------------------------------------------------------------------
data required time                                                             19.720
data arrival time                                                             -16.420
-------------------------------------------------------------------------------------
slack (MET)                                                                    3.300


#Path 2
Startpoint: out[0].Q[0] (.latch clocked by clk)
Endpoint  : out[2].D[0] (.latch clocked by clk)
Path Type : setup

Point                                                                  Incr      Path
-------------------------------------------------------------------------------------
clock clk (rise edge)                                                 0.000     0.000
```

Plain Text Tab Width: 8 Ln 1, Col 1 INS

References

- VLSI System Design: https://www.vlsisystemdesign.com/ip/
- RISC-V based Microprocessor: https://github.com/shivanishah269/risc-v-core
- 4-stage RISC-V Core: https://github.com/ShonTaware/RISC-V_Core_4_Stage
- SOFA: https://github.com/lnis-uofu/SOFA
- OpenFPGA: https://openfpga.readthedocs.io/en/master/
- VPR: https://docs.verilogtorouting.org/en/latest/vpr/
- VTR: https://docs.verilogtorouting.org/en/latest/

Acknowledgement

- Kunal Ghosh, Co-founder, VSD Corp. Pvt. Ltd.
- Nanditha Rao