EC383 - Mini Project in VLSI Design

Toast RISC V Core

End Sem Evaluation Report

Submitted by

Nikhil Reddy 201EC241

Palgun N P 201EC141

6th Semester B.Tech (ECE)

Course Instructor: Dr. Ramesh Kini M.

Department of Electronics and Communications Engineering,

National Institute of Technology Karnataka, Surathkal

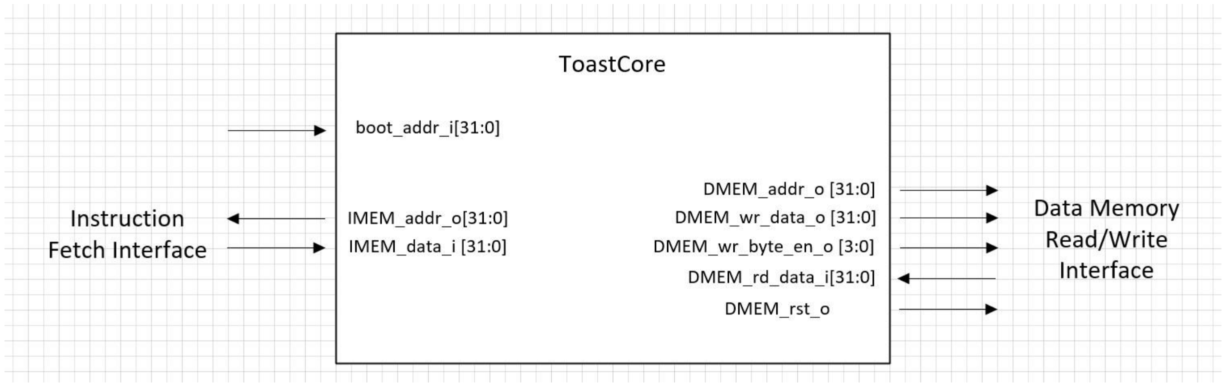
Introduction:

RISC-V is an open-source instruction set architecture (ISA) for processors. A RISC-V 32-bit core refers to a processor core that implements the RISC-V ISA with a 32-bit data width. These cores are known for their small size, low power consumption, and high performance, making them suitable for various applications such as embedded systems, IoT devices, and mobile devices. The open-source nature of RISC-V allows for customization and innovation, fostering a collaborative ecosystem of developers and researchers.

Literature Survey:

Toast Core:

Github Repo: https://github.com/georgeyhere/Toast-RV32i
Toast is a RISC-V soft core written in Verilog that implements a subset of the RV32I ISA version 2.2.

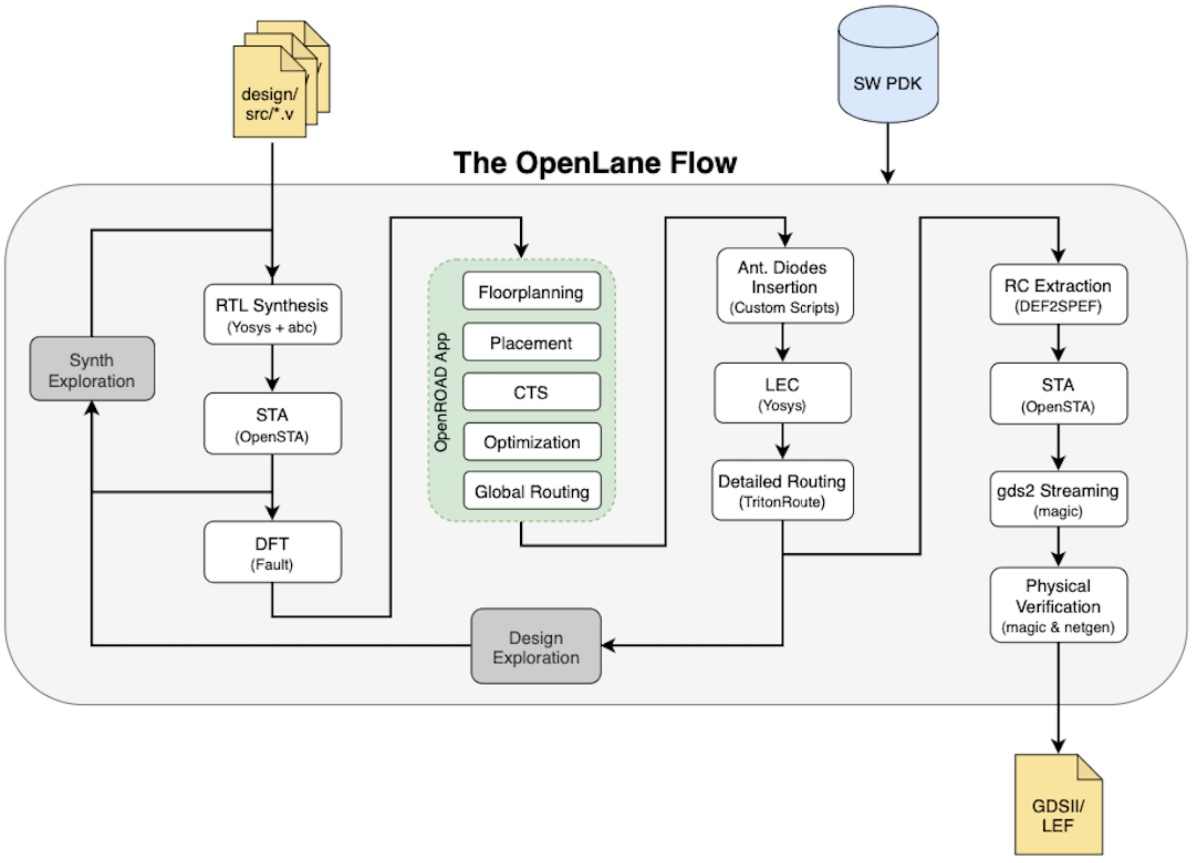


Documentation: https://toast-core.readthedocs.io/en/latest/

Toast currently is capable of passing all RV32ui unit tests and has passed post-synthesis timing at 100MHz targeting Xilinx Artix-7 FPGA.
Toast does not support interrupt handling, nor the CSR, FENCE, EBREAK, or ECALL instruction

Docs Openlane: https://openlane.readthedocs.io/en/latest/flow_overview.html

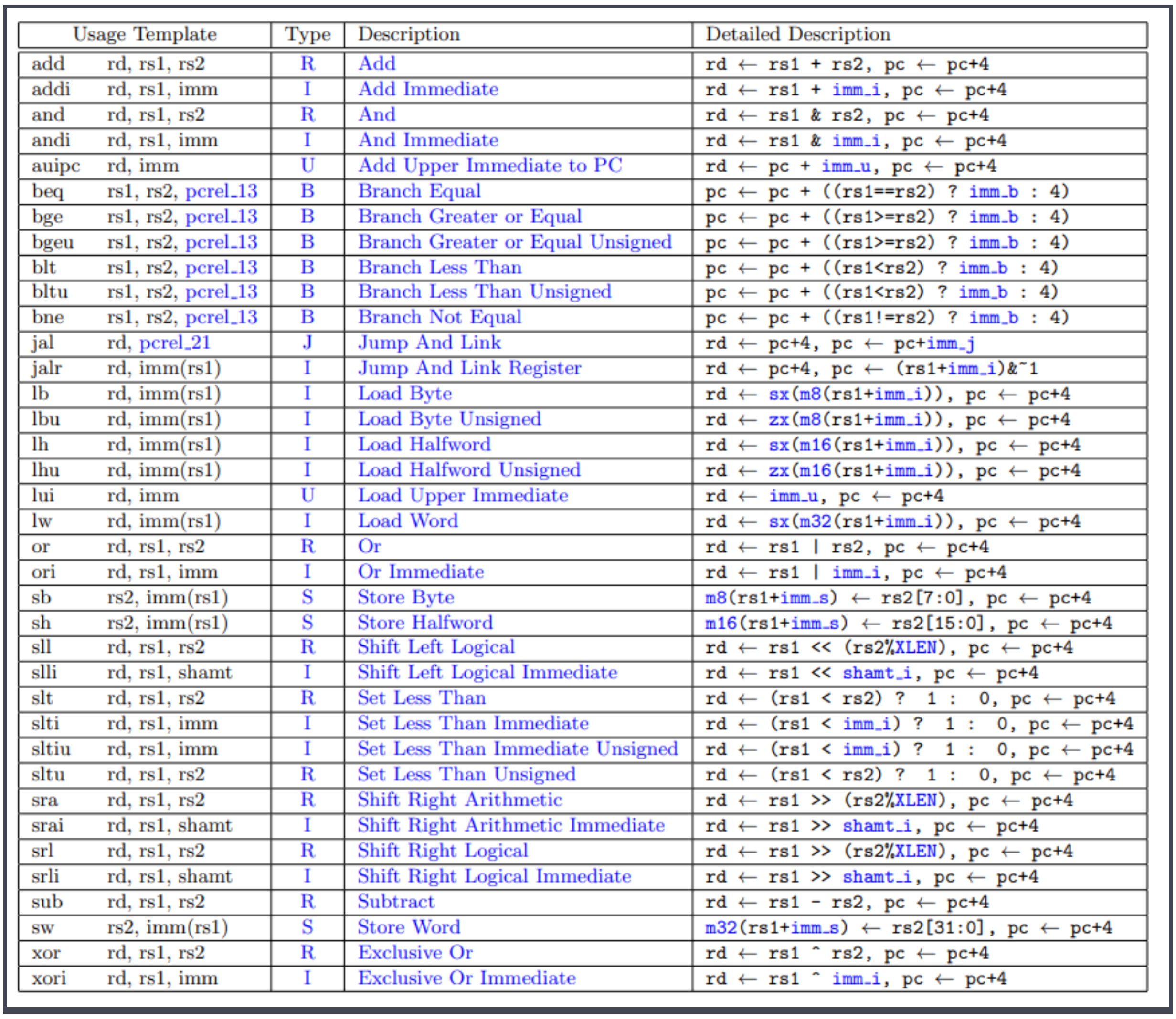
During our literature survey, we explored various tools for digital chip design and came across OpenLANE, a notable open-source solution that provides a complete RTL-to-GDSII flow. OpenLANE facilitates the conversion of chip designs from high-level languages like Verilog or VHDL to the final physical layout for fabrication. It encompasses synthesis, placement and routing, timing analysis, and physical verification, making it a comprehensive tool for digital chip design. Being open-source, OpenLANE is freely available for use and modification, making it a cost-effective and efficient choice for academics, researchers, and small companies. Leveraging the capabilities of OpenLANE, we successfully designed and implemented our project during our study.



RISC V 32I ISA:

Docs: https://github.com/johnwinans/rvalp/releases

RISC-V extension modules may be included by an implementer interested in optimizing a design for one or more purposes. Available extension modules include M (integer math), A (atomic), F (32-bit floating point), D (64-bit 209 floating point), Q (128-bit floating point), C (compressed size instructions), and others. The extension name G is used to represent the combined set of IMAFD extensions as it is expected to be a common combination.



Usage Template		Type	Description	Detailed Description
add	rd, rs1, rs2	R	Add	rd ← rs1 + rs2, pc ← pc+4
addi	rd, rs1, imm	I	Add Immediate	rd ← rs1 + imm_i, pc ← pc+4
and	rd, rs1, rs2	R	And	rd ← rs1 & rs2, pc ← pc+4
andi	rd, rs1, imm	I	And Immediate	rd ← rs1 & imm_i, pc ← pc+4
auipc	rd, imm	U	Add Upper Immediate to PC	rd ← pc + imm_u, pc ← pc+4
beq	rs1, rs2, pcrel_13	B	Branch Equal	pc ← pc + ((rs1==rs2) ? imm_b : 4)
bge	rs1, rs2, pcrel_13	B	Branch Greater or Equal	pc ← pc + ((rs1>=rs2) ? imm_b : 4)
bgeu	rs1, rs2, pcrel_13	B	Branch Greater or Equal Unsigned	pc ← pc + ((rs1>=rs2) ? imm_b : 4)
blt	rs1, rs2, pcrel_13	B	Branch Less Than	pc ← pc + ((rs1<rs2) ? imm_b : 4)
bltu	rs1, rs2, pcrel_13	B	Branch Less Than Unsigned	pc ← pc + ((rs1<rs2) ? imm_b : 4)
bne	rs1, rs2, pcrel_13	B	Branch Not Equal	pc ← pc + ((rs1!=rs2) ? imm_b : 4)
jal	rd, pcrel_21	J	Jump And Link	rd ← pc+4, pc ← pc+imm_j
jalr	rd, imm(rs1)	I	Jump And Link Register	rd ← pc+4, pc ← (rs1+imm_i)&~1
lb	rd, imm(rs1)	I	Load Byte	rd ← sx(m8(rs1+imm_i)), pc ← pc+4
lbu	rd, imm(rs1)	I	Load Byte Unsigned	rd ← zx(m8(rs1+imm_i)), pc ← pc+4
lh	rd, imm(rs1)	I	Load Halfword	rd ← sx(m16(rs1+imm_i)), pc ← pc+4
lhu	rd, imm(rs1)	I	Load Halfword Unsigned	rd ← zx(m16(rs1+imm_i)), pc ← pc+4
lui	rd, imm	U	Load Upper Immediate	rd ← imm_u, pc ← pc+4
lw	rd, imm(rs1)	I	Load Word	rd ← sx(m32(rs1+imm_i)), pc ← pc+4
or	rd, rs1, rs2	R	Or	rd ← rs1 | rs2, pc ← pc+4
ori	rd, rs1, imm	I	Or Immediate	rd ← rs1 | imm_i, pc ← pc+4
sb	rs2, imm(rs1)	S	Store Byte	m8(rs1+imm_s) ← rs2[7:0], pc ← pc+4
sh	rs2, imm(rs1)	S	Store Halfword	m16(rs1+imm_s) ← rs2[15:0], pc ← pc+4
sll	rd, rs1, rs2	R	Shift Left Logical	rd ← rs1 << (rs2%XLEN), pc ← pc+4
slli	rd, rs1, shamt	I	Shift Left Logical Immediate	rd ← rs1 << shamt_i, pc ← pc+4
slt	rd, rs1, rs2	R	Set Less Than	rd ← (rs1 < rs2) ? 1 : 0, pc ← pc+4
slti	rd, rs1, imm	I	Set Less Than Immediate	rd ← (rs1 < imm_i) ? 1 : 0, pc ← pc+4
sltiu	rd, rs1, imm	I	Set Less Than Immediate Unsigned	rd ← (rs1 < imm_i) ? 1 : 0, pc ← pc+4
sltu	rd, rs1, rs2	R	Set Less Than Unsigned	rd ← (rs1 < rs2) ? 1 : 0, pc ← pc+4
sra	rd, rs1, rs2	R	Shift Right Arithmetic	rd ← rs1 >> (rs2%XLEN), pc ← pc+4
srai	rd, rs1, shamt	I	Shift Right Arithmetic Immediate	rd ← rs1 >> shamt_i, pc ← pc+4
srl	rd, rs1, rs2	R	Shift Right Logical	rd ← rs1 >> (rs2%XLEN), pc ← pc+4
srli	rd, rs1, shamt	I	Shift Right Logical Immediate	rd ← rs1 >> shamt_i, pc ← pc+4
sub	rd, rs1, rs2	R	Subtract	rd ← rs1 - rs2, pc ← pc+4
sw	rs2, imm(rs1)	S	Store Word	m32(rs1+imm_s) ← rs2[31:0], pc ← pc+4
xor	rd, rs1, rs2	R	Exclusive Or	rd ← rs1 ^ rs2, pc ← pc+4
xori	rd, rs1, imm	I	Exclusive Or Immediate	rd ← rs1 ^ imm_i, pc ← pc+4

<u>Problem Statement:</u> Developing an Open-Source RISC-V Core using OpenLANE

As part of our project, we aim to design and implement a RISC-V core using OpenLANE, an open-source tool for digital chip design. The RTL (Register Transfer Level) design for the core has been obtained from a GitHub repository.

However, there are several challenges that need to be addressed in this project:

1. Integration with OpenLANE Flow: Integrating the RTL design into the OpenLANE flow requires understanding and configuring the various steps in the RTL-to-GDSII flow, including synthesis, placement, routing, timing analysis, and physical verification. Ensuring proper integration and compatibility of the RTL design with the OpenLANE flow is essential for a successful chip design.
2. Performance Optimization: As part of the project, we aim to optimize the performance of the RISC-V core. This involves fine-tuning the design

parameters, such as pipeline depth, clock frequency, and memory hierarchy, to achieve desired performance metrics such as throughput, latency, and power consumption.

3. Documentation and Collaboration: Proper documentation of the design process, methodologies, and results is crucial for ensuring reproducibility and collaboration within the team. Creating comprehensive documentation that captures the design decisions, optimizations, and challenges faced during the project is an important aspect of the project.

Overall, the goal of our project is to successfully design and implement a RISC-V core using OpenLANE, while addressing the challenges related to RTL verification, integration with the OpenLANE flow, performance optimization, documentation, and cost/resource constraints. By overcoming these challenges, we aim to contribute to the open-source RISC-V community and create a functional and optimized RISC-V core design using OpenLANE.

Process Development Kit:

Docs: https://skywater-pdk.readthedocs.io/en/main/

Skywater130nm refers to a semiconductor process node developed by SkyWater Technology Foundry, which is a semiconductor foundry that provides manufacturing services for integrated circuits (ICs). The "130nm" indicates the minimum feature size or the smallest dimensions of the transistors that can be fabricated using this process technology.

The Skywater130nm process technology is based on a 130-nanometer (nm) node, which is considered to be a mature technology node in the semiconductor industry. It offers a good balance between performance, power consumption, and cost-effectiveness, making it suitable for a wide range of applications, including consumer electronics, automotive, industrial, and medical devices.

The Skywater130nm process technology provides various options for designing and fabricating digital, analog, and mixed-signal ICs. It supports both digital and analog circuitry, including standard logic gates, memory cells, and analog devices such as operational amplifiers and data converters. The process also allows for the integration of passive components such as resistors and capacitors.

One notable aspect of the Skywater130nm process technology is its open-source nature. SkyWater Technology Foundry has made the process design rules, standard cell libraries, and other design resources available as open-source, enabling greater accessibility and collaboration in the development of ICs. This has garnered interest from academia, researchers, and small companies looking for cost-effective fabrication solutions.

Overall, the Skywater130nm process technology is a mature and cost-effective semiconductor process node that offers a wide range of design and fabrication options, and its open-source nature has further contributed to its popularity in the semiconductor community.

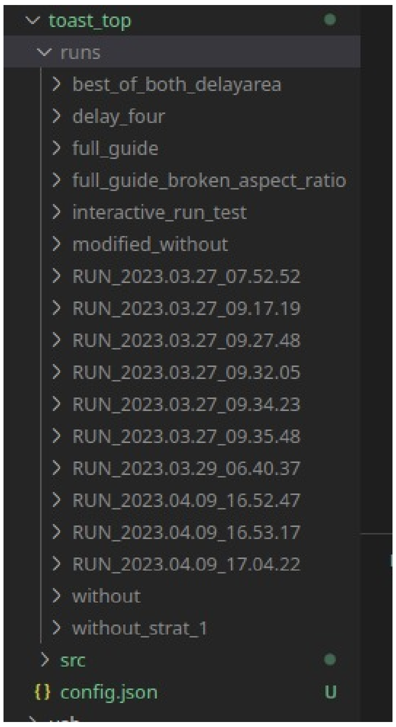
Openlane Setup:

Docs:https://openlane.readthedocs.io/en/latest/tutorials/digital_guide.html

- Clone the OpenLANE repository: Start by cloning the OpenLANE repository from GitHub (https://github.com/efabless/openlane) to your local machine.
- Install the necessary dependencies: Follow the instructions in the OpenLANE documentation to install the necessary dependencies, including the required EDA (Electronic Design Automation) tools such as Yosys, Magic, and OpenROAD.
- Prepare the design files: Gather the RTL (Register Transfer Level) design files for the RISC-V from the Toast GitHub repository (https://github.com/georgeyhere/Toast-RV32i/tree/main/rtl).
- Define the design configuration: Created a configuration file that defines the design parameters, such as the target process technology (e.g., Skywater130nm), the clock frequency, and the power constraints.
- Run the RTL synthesis: Used the OpenLANE synthesis flow to convert the RTL design files into a gate-level netlist. This involves using the Yosys synthesis tool to perform logic synthesis, optimizing the design for area, power, and performance.
- Perform floorplanning: Used the OpenLANE floorplanning flow to define

the physical layout of the RISC-V core on the chip. This involves specifying the placement of different functional blocks, I/O pads, and power/ground rails, considering design constraints and performance requirements.

- Run placement and routing: Used the OpenLANE placement and routing flow to automatically place and route the RISC-V core on the chip. This involves optimizing the placement of cells to minimize wire length and congestion, and routing the interconnections between cells to meet timing constraints.
- Perform physical verification: Used the OpenLANE physical verification flow to perform various checks on the layout, such as DRC (Design Rule Checking) and LVS (Layout vs. Schematic) to ensure that the design meets the fabrication requirements and matches the intended RTL design.
- Generate the final GDSII layout: Once the layout was physically verified, we used the OpenLANE flow to generate the final GDSII layout file, which is the format used for chip fabrication.
- Review and iterate: Reviewed the generated layout and performed any necessary iterations to refine the design, optimize performance, fixed any potential issues identified during physical verification.



```
v toast_top
  v runs
    > best_of_both_delayarea
    > delay_four
    > full_guide
    > full_guide_broken_aspect_ratio
    > interactive_run_test
    > modified_without
    > RUN_2023.03.27_07.52.52
    > RUN_2023.03.27_09.17.19
    > RUN_2023.03.27_09.27.48
    > RUN_2023.03.27_09.32.05
    > RUN_2023.03.27_09.34.23
    > RUN_2023.03.27_09.35.48
    > RUN_2023.03.29_06.40.37
    > RUN_2023.04.09_16.52.47
    > RUN_2023.04.09_16.53.17
    > RUN_2023.04.09_17.04.22
    > without
    > without_strat_1
  > src
  {} config.json                  U
```

Directory Layout

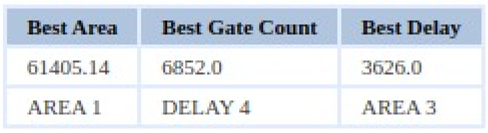
Config File:

Optimized for both delay and area

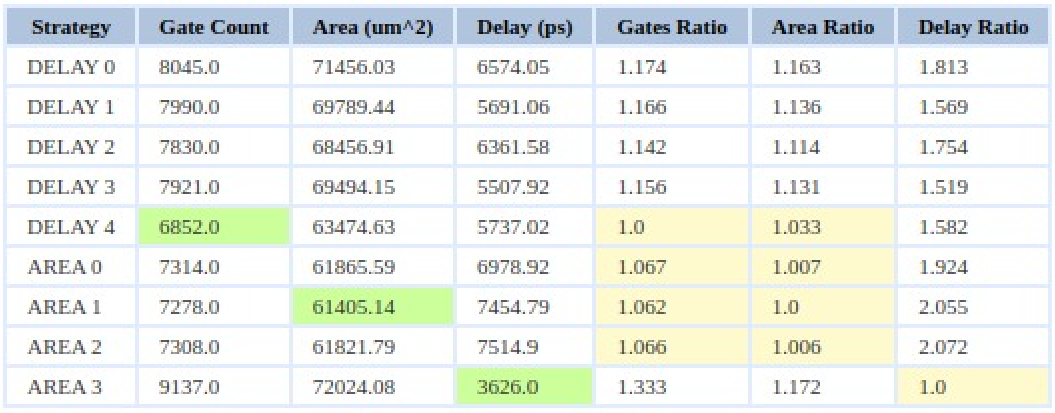
```
{
    "DESIGN_NAME": "toast_top",
    "VERILOG_FILES": "dir::src/*.v",
    "CLOCK_PORT": "clk_i",
    "CLOCK_PERIOD": 10.0,
    "DESIGN_IS_CORE": true,
    "SYNTH_MAX_FANOUT": 17,
    "SYNTH_STRATEGY": "DELAY 4",
    "PL_TARGET_DENSITY": 0.52,
    "FP_PDN_AUTO_ADJUST": "FALSE"
}
```

Synthesis Exploration:

The table below shows the best results achieved for each metric:



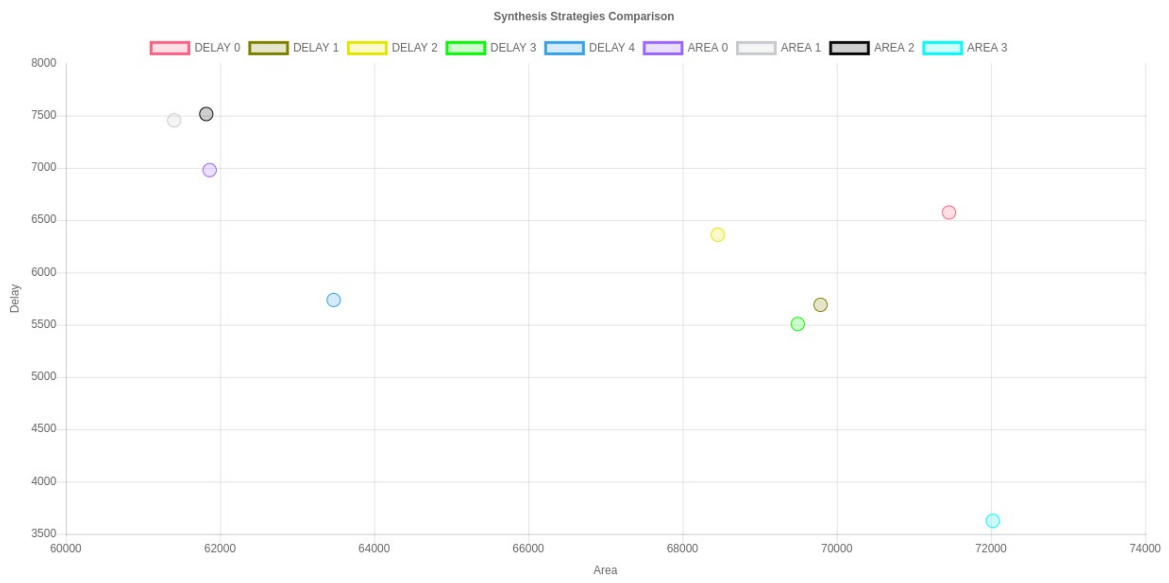
Best Area	Best Gate Count	Best Delay
61405.14	6852.0	3626.0
AREA 1	DELAY 4	AREA 3



Strategy	Gate Count	Area (um^2)	Delay (ps)	Gates Ratio	Area Ratio	Delay Ratio
DELAY 0	8045.0	71456.03	6574.05	1.174	1.163	1.813
DELAY 1	7990.0	69789.44	5691.06	1.166	1.136	1.569
DELAY 2	7830.0	68456.91	6361.58	1.142	1.114	1.754
DELAY 3	7921.0	69494.15	5507.92	1.156	1.131	1.519
DELAY 4	6852.0	63474.63	5737.02	1.0	1.033	1.582
AREA 0	7314.0	61865.59	6978.92	1.067	1.007	1.924
AREA 1	7278.0	61405.14	7454.79	1.062	1.0	2.055
AREA 2	7308.0	61821.79	7514.9	1.066	1.006	2.072
AREA 3	9137.0	72024.08	3626.0	1.333	1.172	1.0

Scatter Plot:

The scatter chart below shows the relationship between area and delay for each strategy:



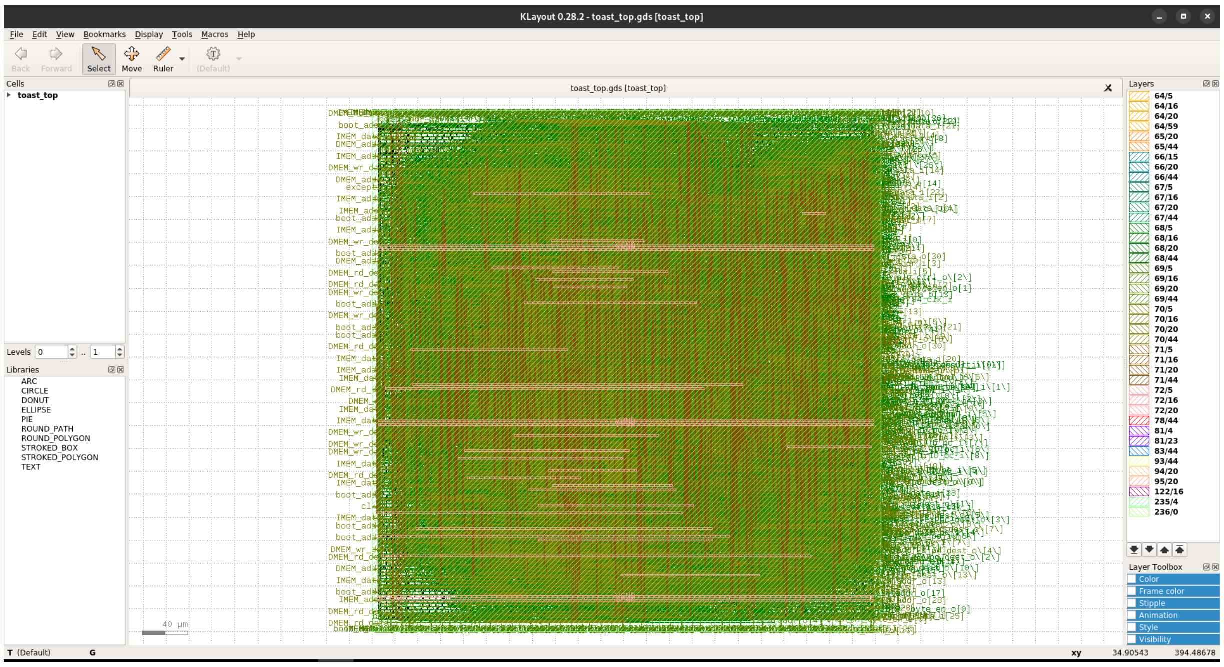
The chart shows that strategies with lower delay tend to have higher area, and vice versa. However, We can see that the DELAY 4 strategy stands out as having both low delay and low area.

<u>Conclusion:</u>

Based on the results of the exploration analysis, the best strategy for minimizing area, delay, and gate count is DELAY 4. Hence results without a synthesis strategy and subsequently DELAY 4 are observed. Other strategies can be adopted based on the relevant delay vs area tradeoff that is allowable.

KLayout:

KLayout is a free and open-source layout viewer and editor for electronic design automation (EDA) purposes. It is widely used in the semiconductor industry and academia for viewing, editing, and analyzing layout designs of integrated circuits (ICs). KLayout provides a user-friendly interface with powerful features for visualizing and manipulating layout data, including zooming, panning, and rotating views, layer management, measurement tools, and various editing functions. It supports multiple layout file formats, including GDSII, OASIS, DXF, and CIF, making it compatible with different design environments. KLayout is highly extensible and customizable, allowing users to create their own macros and scripts for automating tasks and enhancing productivity. It is available for Windows, Linux, and macOS platforms, making it a popular choice among IC designers and researchers for layout editing and analysis.



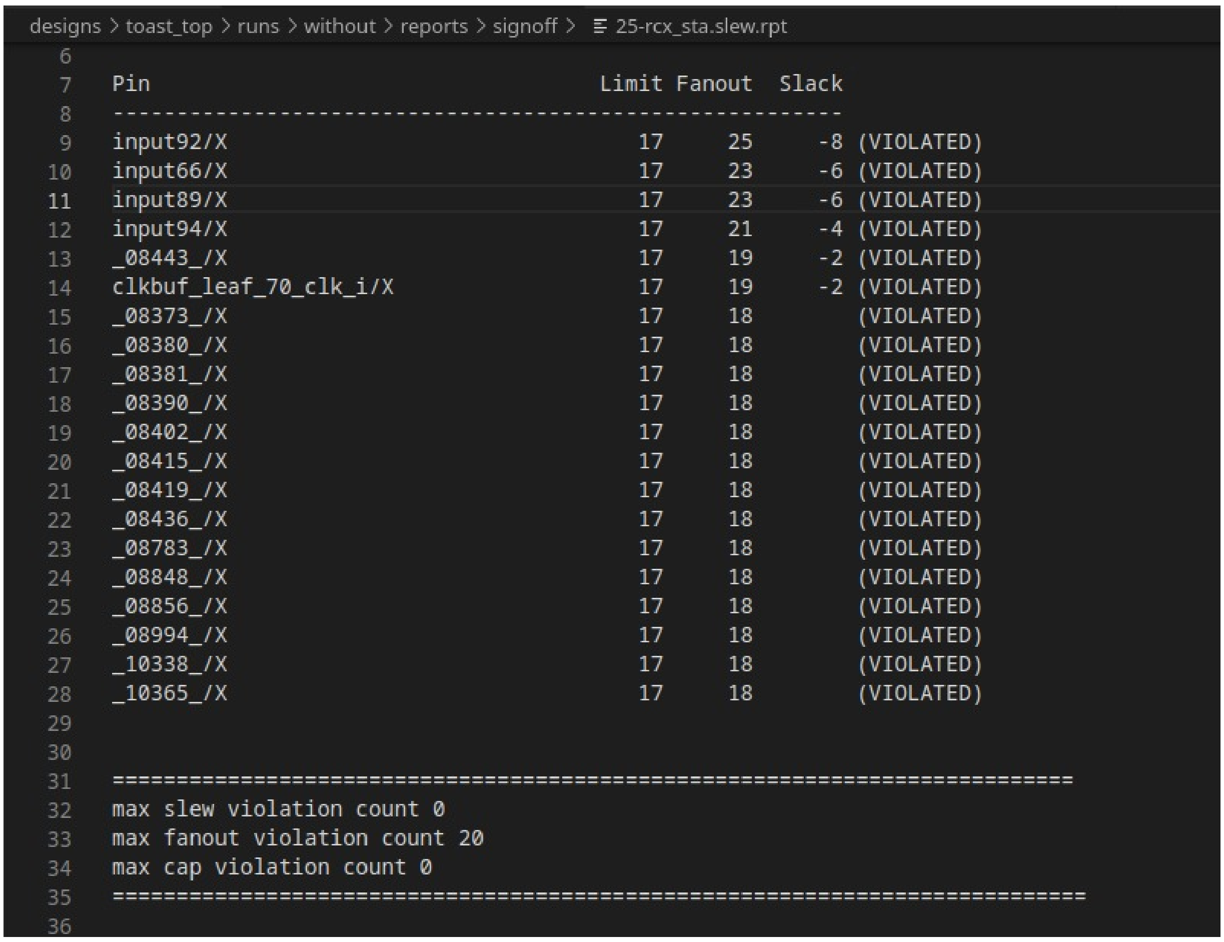
Results:

Without any synthesis strategy:

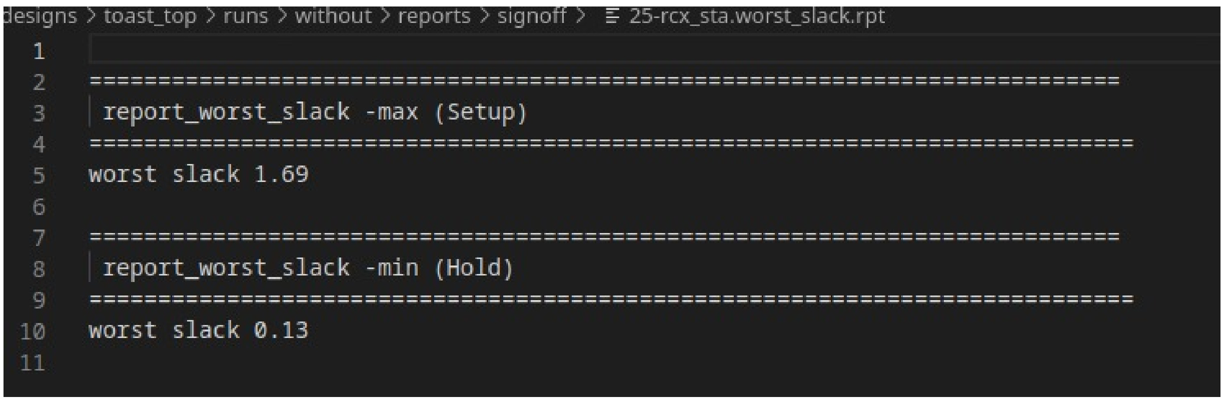
If we use the AREA 3 strategy which optimizes the design for the delay but with the maximum area, using a max fanout of 17 accounts for no fanout violations.

However the same with DELAY 4 creates fanout violations and thus we need to increase the max fanout to 25.

It can be observed that currently, 20 fanout violations are present due to this.

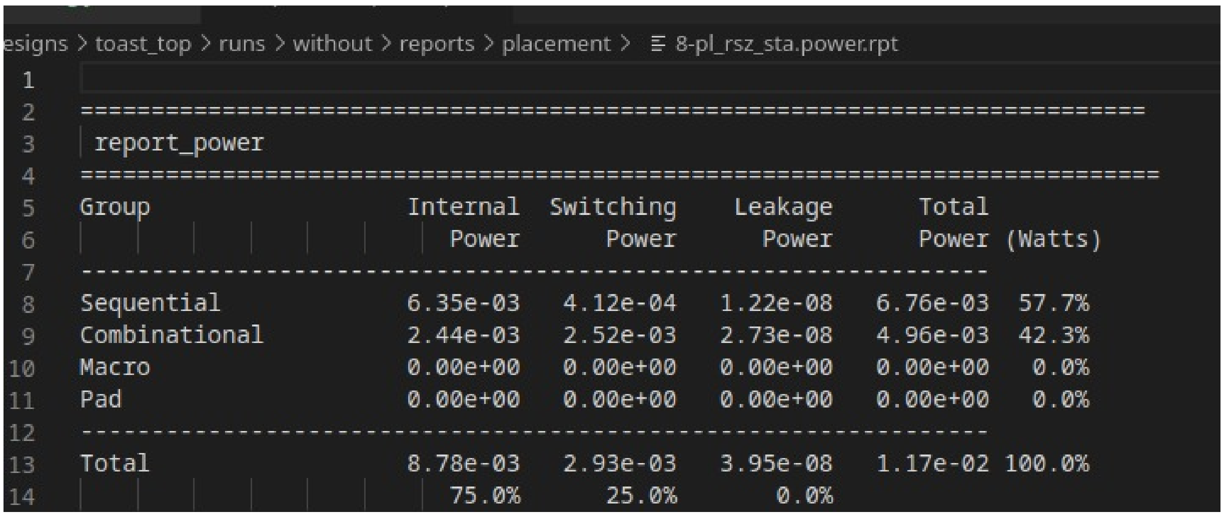

designs > toast_top > runs > without > reports > signoff > 25-rcx_sta.slew.rpt

```

Pin                                  Limit Fanout  Slack
--------------------------------------------------------
input92/X                               17     25     -8 (VIOLATED)
input66/X                               17     23     -6 (VIOLATED)
input89/X                               17     23     -6 (VIOLATED)
input94/X                               17     21     -4 (VIOLATED)
_08443_/X                               17     19     -2 (VIOLATED)
clkbuf_leaf_70_clk_i/X                  17     19     -2 (VIOLATED)
_08373_/X                               17     18        (VIOLATED)
_08380_/X                               17     18        (VIOLATED)
_08381_/X                               17     18        (VIOLATED)
_08390_/X                               17     18        (VIOLATED)
_08402_/X                               17     18        (VIOLATED)
_08415_/X                               17     18        (VIOLATED)
_08419_/X                               17     18        (VIOLATED)
_08436_/X                               17     18        (VIOLATED)
_08783_/X                               17     18        (VIOLATED)
_08848_/X                               17     18        (VIOLATED)
_08856_/X                               17     18        (VIOLATED)
_08994_/X                               17     18        (VIOLATED)
_10338_/X                               17     18        (VIOLATED)
_10365_/X                               17     18        (VIOLATED)


===========================================================================
max slew violation count 0
max fanout violation count 20
max cap violation count 0
============================================================================

```


designs > toast_top > runs > without > reports > signoff > 25-rcx_sta.worst_slack.rpt

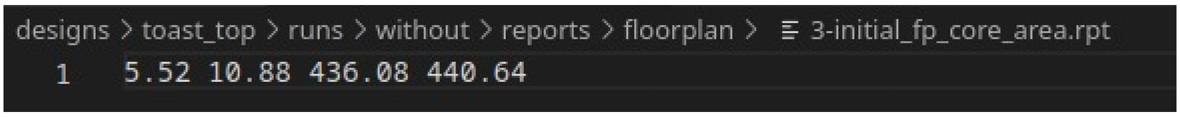
```

===========================================================================
 report_worst_slack -max (Setup)
============================================================================
worst slack 1.69

===========================================================================
 report_worst_slack -min (Hold)
============================================================================
worst slack 0.13

```

We observe the maximum static timing analysis setup and hold time slack for the worst cases. There exist no hold or setup violations. However important to note the higher value of setup slack compared to the lower value of hold slack. We can see that both values of slack are positive in this case.

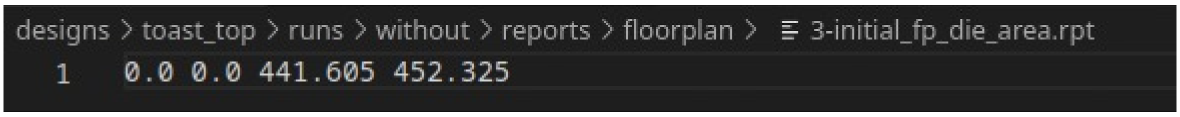


```
esigns > toast_top > runs > without > reports > placement > 8-pl_rsz_sta.power.rpt

==========================================================================
 report_power
==========================================================================
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
----------------------------------------------------------------
Sequential             6.35e-03   4.12e-04   1.22e-08   6.76e-03  57.7%
Combinational          2.44e-03   2.52e-03   2.73e-08   4.96e-03  42.3%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  8.78e-03   2.93e-03   3.95e-08   1.17e-02 100.0%
                          75.0%      25.0%       0.0%
```

We can observe that the total power consumed in this case is 11.7mW.



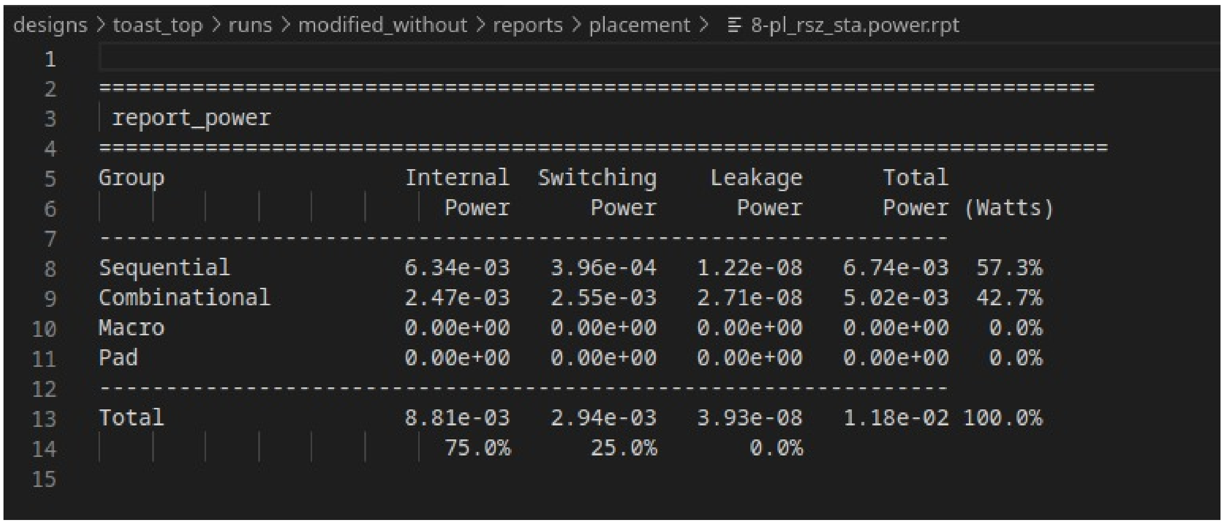
```
designs > toast_top > runs > without > reports > floorplan > 3-initial_fp_core_area.rpt
5.52 10.88 436.08 440.64
```



```
designs > toast_top > runs > without > reports > floorplan > 3-initial_fp_die_area.rpt
0.0 0.0 441.605 452.325
```

After changing the max fanout to 25 keeping no synthesis strategy:

By changing max fanout 25 we can now observe that there are no more fanout violations, but however increasing this max fanout could lead to severe implications with respect to sizing and power consumed. The same effect is visualized with the following results.

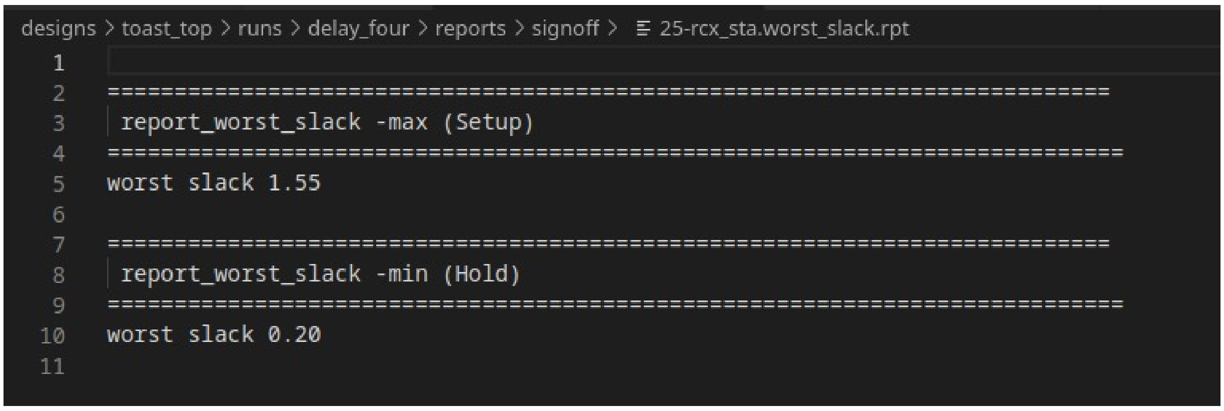


```
designs > toast_top > runs > modified_without > reports > placement > 8-pl_rsz_sta.power.rpt

==========================================================================
 report_power
==========================================================================
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
----------------------------------------------------------------
Sequential             6.34e-03   3.96e-04   1.22e-08   6.74e-03  57.3%
Combinational          2.47e-03   2.55e-03   2.71e-08   5.02e-03  42.7%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  8.81e-03   2.94e-03   3.93e-08   1.18e-02 100.0%
                          75.0%      25.0%       0.0%
```

It can now be observed that the total power consumed is 11.8mW as compared to the 11.7mW as shown before. Hence it is important to note the increase in power consumed by increasing the max fanout.

Using the DELAY 4 strategy:

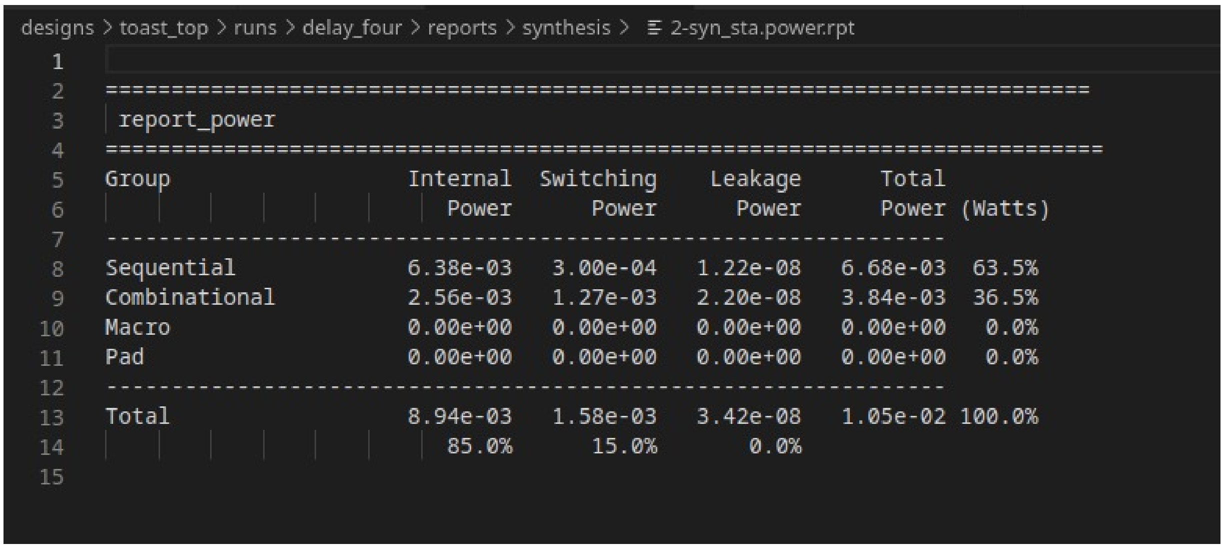
The DELAY 4 strategy minimizes both the area and delay tradeoff equally compared to the other strategies. We can now observe the various parameters and confirm the optimized approach of synthesis. Starting with the slack violations:



designs > toast_top > runs > delay_four > reports > signoff > 25-rcx_sta.worst_slack.rpt

```

===========================================================================
 report_worst_slack -max (Setup)
============================================================================
worst slack 1.55

===========================================================================
 report_worst_slack -min (Hold)
============================================================================
worst slack 0.20

```

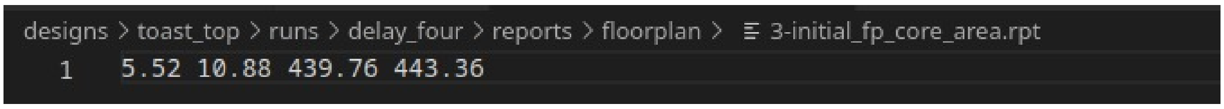
We can observe here as well that there are no hold or setup violations. We can see that the setup slack decreases and that the hold slack increases. Important that both remained positive throughout the optimization process.



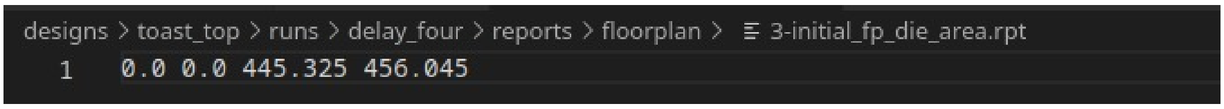
designs > toast_top > runs > delay_four > reports > synthesis > 2-syn_sta.power.rpt

```

===========================================================================
 report_power
============================================================================
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
----------------------------------------------------------------
Sequential             6.38e-03   3.00e-04   1.22e-08   6.68e-03  63.5%
Combinational          2.56e-03   1.27e-03   2.20e-08   3.84e-03  36.5%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  8.94e-03   1.58e-03   3.42e-08   1.05e-02 100.0%
                          85.0%      15.0%       0.0%

```

We can finally conclude through the total power consumed that the optimized DELAY 4 strategy yields a lower power consumption of 10.5mW as compared to the higher power consumption of the previously adopted methods.



```
designs > toast_top > runs > delay_four > reports > floorplan > 3-initial_fp_core_area.rpt
5.52 10.88 439.76 443.36
```

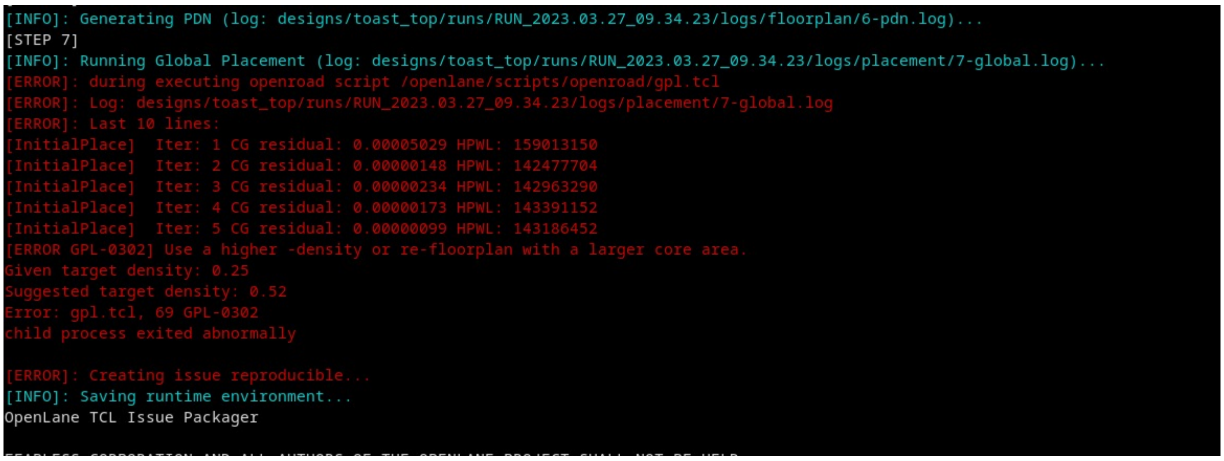


```
designs > toast_top > runs > delay_four > reports > floorplan > 3-initial_fp_die_area.rpt
0.0 0.0 445.325 456.045
```

The above is the observed core area dimensions and the die area dimensions as well which can also be included in the config file for the specification.

Additional Optimisation performed:

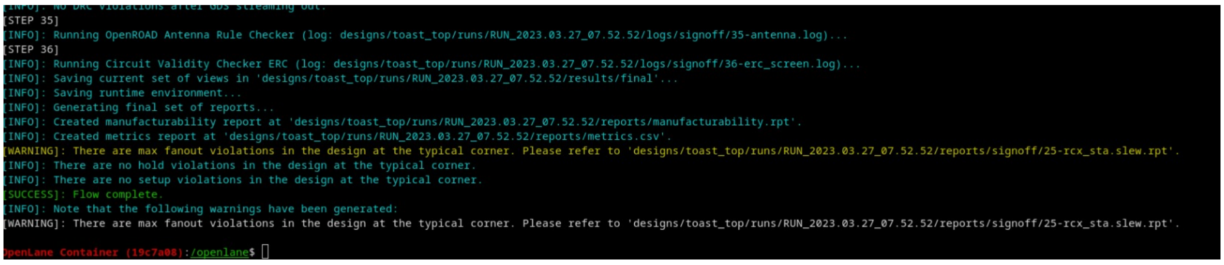
Initially we started out with a lower floor plan density of 0.25. Leading us to fall into trouble during the run_floorplan stage of the interactive synthesis. On attaining the following error



```
[INFO]: Generating PDN (log: designs/toast_top/runs/RUN_2023.03.27_09.34.23/logs/floorplan/6-pdn.log)...
[STEP 7]
[INFO]: Running Global Placement (log: designs/toast_top/runs/RUN_2023.03.27_09.34.23/logs/placement/7-global.log)...
[ERROR]: during executing openroad script /openlane/scripts/openroad/gpl.tcl
[ERROR]: Log: designs/toast_top/runs/RUN_2023.03.27_09.34.23/logs/placement/7-global.log
[ERROR]: Last 10 lines:
[InitialPlace]  Iter: 1 CG residual: 0.00005029 HPWL: 159013150
[InitialPlace]  Iter: 2 CG residual: 0.00000148 HPWL: 142477704
[InitialPlace]  Iter: 3 CG residual: 0.00000234 HPWL: 142963290
[InitialPlace]  Iter: 4 CG residual: 0.00000173 HPWL: 143391152
[InitialPlace]  Iter: 5 CG residual: 0.00000099 HPWL: 143186452
[ERROR GPL-0302] Use a higher -density or re-floorplan with a larger core area.
Given target density: 0.25
Suggested target density: 0.52
Error: gpl.tcl, 69 GPL-0302
child process exited abnormally

[ERROR]: Creating issue reproducible...
[INFO]: Saving runtime environment...
OpenLane TCL Issue Packager
```

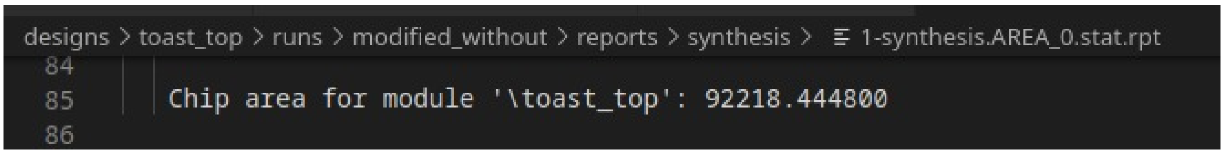
New optimized floorplan density was observed to be 0.52 which yielded optimal performance.

By default, the max fanout is limited to 10. On going through the logs and the reports for the slew and max fanout violations, it could be concluded that increasing the max fanout to 17 could remove all these violations.



```
[INFO]: No DRC violations after GDS streaming out.
[STEP 35]
[INFO]: Running OpenROAD Antenna Rule Checker (log: designs/toast_top/runs/RUN_2023.03.27_07.52.52/logs/signoff/35-antenna.log)...
[STEP 36]
[INFO]: Running Circuit Validity Checker ERC (log: designs/toast_top/runs/RUN_2023.03.27_07.52.52/logs/signoff/36-erc_screen.log)...
[INFO]: Saving current set of views in 'designs/toast_top/runs/RUN_2023.03.27_07.52.52/results/final'...
[INFO]: Saving runtime environment...
[INFO]: Generating final set of reports...
[INFO]: Created manufacturability report at 'designs/toast_top/runs/RUN_2023.03.27_07.52.52/reports/manufacturability.rpt'.
[INFO]: Created metrics report at 'designs/toast_top/runs/RUN_2023.03.27_07.52.52/reports/metrics.csv'.
[WARNING]: There are max fanout violations in the design at the typical corner. Please refer to 'designs/toast_top/runs/RUN_2023.03.27_07.52.52/reports/signoff/25-rcx_sta.slew.rpt'.
[INFO]: There are no hold violations in the design at the typical corner.
[INFO]: There are no setup violations in the design at the typical corner.
[SUCCESS]: Flow complete.
[INFO]: Note that the following warnings have been generated:
[WARNING]: There are max fanout violations in the design at the typical corner. Please refer to 'designs/toast_top/runs/RUN_2023.03.27_07.52.52/reports/signoff/25-rcx_sta.slew.rpt'.

OpenLane Container (19c7a08):/openlane$ 
```

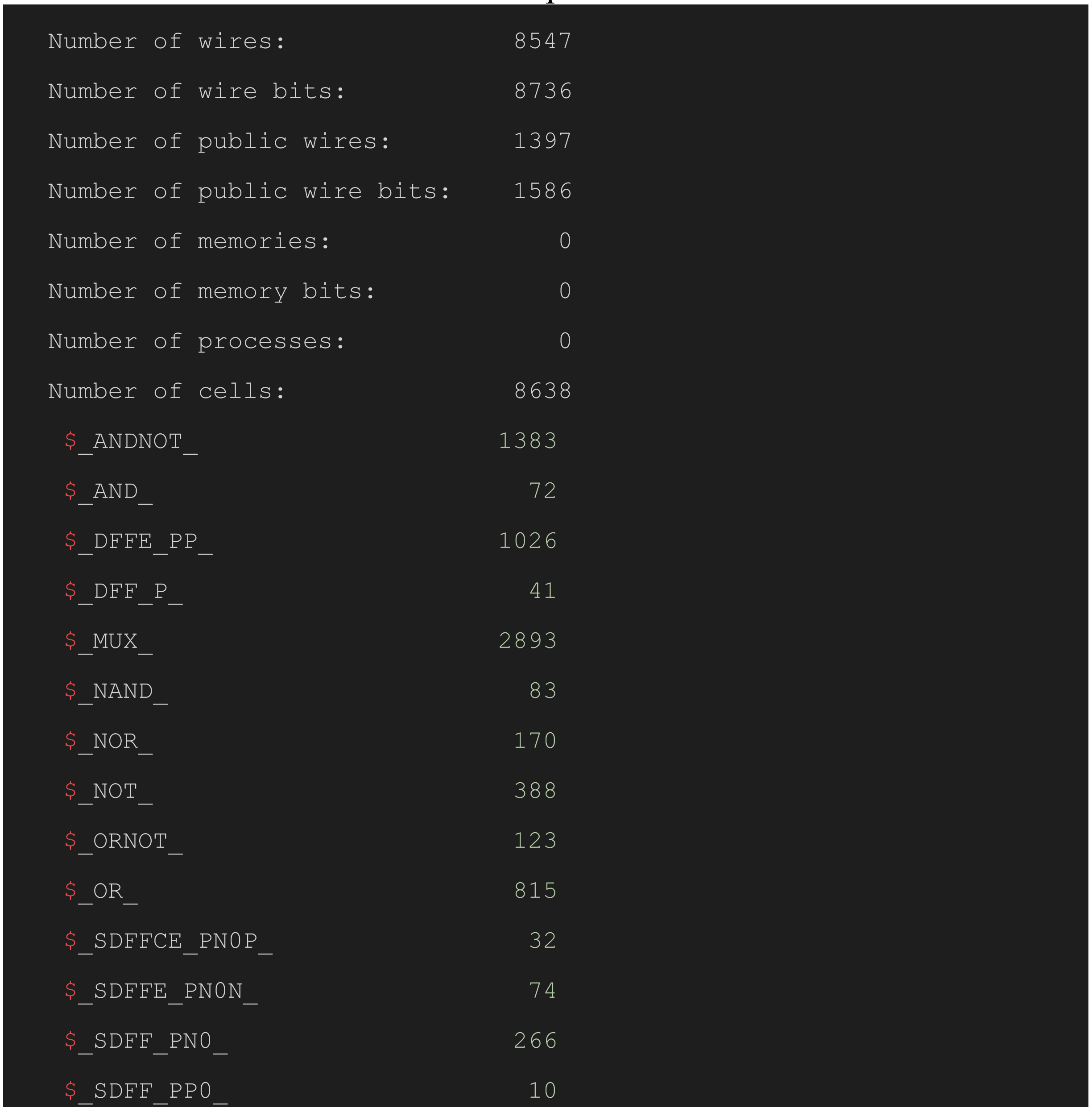
After updating the max fanout to 17 it could be observed that there were no more fanout violations.


designs > toast_top > runs > modified_without > reports > synthesis > 1-synthesis.AREA_0.stat.rpt

```
Chip area for module '\toast_top': 92218.444800
```

This is the total are consumed by the chip for this design which is 92218.4448mm^2.

Initial statistics for the RISC - V core processor



```
   Number of wires:               8547
   Number of wire bits:           8736
   Number of public wires:        1397
   Number of public wire bits:    1586
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:               8638
     $_ANDNOT_                    1383
     $_AND_                         72
     $_DFFE_PP_                   1026
     $_DFF_P_                       41
     $_MUX_                       2893
     $_NAND_                        83
     $_NOR_                        170
     $_NOT_                        388
     $_ORNOT_                      123
     $_OR_                         815
     $_SDFFCE_PN0P_                 32
     $_SDFFE_PN0N_                  74
     $_SDFF_PN0_                   266
     $_SDFF_PP0_                    10
```

```
    $_XNOR_                       39

    $_XOR_                       254
```

Statistics after technology mapping with DELAY 4 strategy:

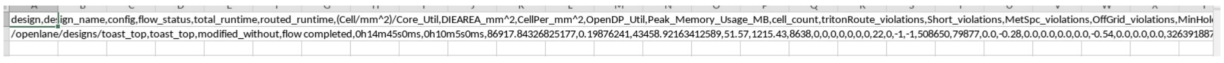
```
  Number of wires:              7605

  Number of wire bits:          9340

  Number of public wires:         99

  Number of public wire bits:   1834

  Number of memories:              0

  Number of memory bits:           0

  Number of processes:             0

  Number of cells:              9183

    $_ANDNOT_                   1383

    $_AND_                        72

    $_MUX_                      4407

    $_NAND_                       83

    $_NOR_                       170

    $_NOT_                       388

    $_ORNOT_                     123

    $_OR_                        815

    $_XNOR_                       39

    $_XOR_                       254

    sky130_fd_sc_hd__dfxtp_2    1449
```

Overall, the report shows that technology mapping can significantly reduce circuit size and the number of cells required to implement a circuit while maintaining its functionality. This reduction can lead to improved performance, reduced power consumption, and lower costs.

The circuit has 8547 wires, 8736 wire bits, 1397 public wires, and 1586 public wire bits in its initial form. The circuit comprises 8638 cells of different types, such as AND gates, D flip-flops, multiplexers, etc.

The circuit's size has reduced significantly. The circuit has 7605 wires, 9340 wire bits, 99 public wires, and 1834 public wire bits after technology mapping using the DELAY 4 strategy.

Metrics:



```
design,design_name,config,flow_status,total_runtime,routed_runtime,(Cell/mm^2)/Core_Util,DIEAREA_mm^2,CellPer_mm^2,OpenDP_Util,Peak_Memory_Usage_MB,cell_count,tritonRoute_violations,Short_violations,MetSpc_violations,OffGrid_violations,MinHol
/openlane/designs/toast_top,toast_top,modified_without,flow completed,0h14m45s0ms,0h10m5s0ms,86917.84326825177,0.19876241,43458.92163412589,51.57,1215.43,8638,0,0,0,0,0,0,0,22,0,-1,-1,508650,79877,0.0,-0.28,0.0,0.0,0.0,0.0,-0.54,0.0,0.0,0.0,326391887
```

Derivables:

Our forked Repository

https://github.com/nikhilreddy2002/Toast-RV32i

Created a Pull request in the Toast repository
https://github.com/georgeyhere/Toast-RV32i/pull/3

Conclusion:

In conclusion, implementing a RISC-V processor in OpenLANE using the "delay 4" strategy can result in several benefits and improvements in the design. The "delay 4" strategy in OpenLANE refers to a specific set of design parameters and optimizations that focus on reducing the critical path delay in the design, thereby improving the overall performance of the processor.

Some of the key parameters and optimizations associated with the "delay 4" strategy in OpenLANE include:

1. High effort synthesis optimizations: By using high-effort synthesis optimizations, such as aggressive clock gating, technology mapping, and gate-level optimization, the critical path delay in the design can be significantly reduced. This can result in improved performance by allowing the processor to operate at higher clock frequencies.

2. Multi-stage optimization: OpenLANE's "delay 4" strategy involves multi-stage optimization, where the design is iteratively optimized at different stages, such as placement, clock tree synthesis, and routing. This helps in identifying and resolving timing violations and

improving the performance of the design.

3. Advanced placement techniques: OpenLANE's "delay 4" strategy also includes advanced placement techniques, such as global placement with intelligent clustering and multi-objective placement, which can optimize the placement of cells to minimize the critical path delay and improve overall performance.

4. Comprehensive physical verification: OpenLANE's "delay 4" strategy includes thorough physical verification checks, such as Design Rule Checking (DRC) and Layout vs. Schematic (LVS), to ensure that the design meets the fabrication requirements and matches the intended RTL design. This helps in identifying and fixing any layout-related issues that can impact the performance of the design.

The implementation of a RISC-V processor in OpenLANE using the "delay 4" strategy can result in improved performance in terms of higher operating frequency, reduced critical path delay, and better overall design optimization. This can lead to a more efficient and high-performing RISC-V processor design, which is crucial for various applications in the field of digital chip design and embedded systems. However, it's important to note that the specific improvements and benefits may vary depending on the design requirements, target process technology, and other factors, and thorough testing and verification should be performed to validate the design's performance.

D

References:

1. George, Y. (2022). Toast RISC V 32I Core [GitHub repository]. Retrieved from https://github.com/georgeyhere/Toast-RV32i
2. The OpenROAD Project. (n.d.). OpenLane: The Open-Source Digital ASIC Implementation Flow [Documentation]. Retrieved from https://openlane.readthedocs.io/en/latest/
3. University of California, Berkeley. (2019, October 29). Introduction to RISC-V and the RV32I Instructions [Video]. YouTube. https://www.youtube.com/watch?v=LKB5I12LctU&t=1211s
4. Winans, J. (2020). RVALP: A RISC-V Assembler, Linker, and Simulator [GitHub repository]. Retrieved from https://github.com/johnwinans/rvalp/releases
5. RISC-V Foundation. (2021). The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.3. Retrieved from https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC-and-Priv-v1.11/riscv-spec-20210512.pdf
6. SkyWater Technology Foundry. (n.d.). SkyWater PDK: Open Source Process Design Kit [Documentation]. Retrieved from https://skywater-pdk.readthedocs.io/en/main/
7. KLayout. (n.d.). KLayout Documentation. Retrieved from https://www.klayout.de/doc.html
8. Shafik, R. (2018). Digital Logic Design Course [Video playlist]. YouTube. Retrieved from https://www.youtube.com/playlist?list=PLJ5C_6qdAvBELELTSPgzYkQg3HgclQh-5