RISC-V based micro-controller using OpenLane

EC383 Mini Project in VLSI Design

Final project report

Electronics and Communication Engineering
National Institute of Technology

April 8, 2022

Kruti Deepan Panda, 191EC126.
Rahul Magesh, 191EC145.

Abstract

This project explores the higly customizable PicoRV32 and explores its various configurations. We have tried running the base core enabled with PCPI, AXI version of the core with PCPI and multiply module enabled, AXI version of the core with PCPI, multiply module, 2 clock cylce ALU and 2 clock cycle compare enabled. We tried to fix errors and STA violations that occured during our runs and tried to determine the highest possible clock frequenct the core can run in the most stable manner possible.

1 Introduction

In this project we explore the OpenLane flow [3] and the various combinations of the PicoRV32 processor [5] to add additional functionality to it.

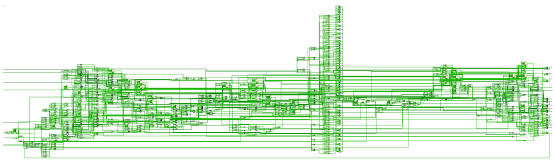
2 Literature survey

1. OpenLane tutorials
   - https://github.com/The-OpenROAD-Project/OpenLane
   - https://inst.eecs.berkeley.edu/~cs250/fa20/labs/lab1/
   - https://www.researchgate.net/publication/355051535_IBTIDA_Fully_open-source_ASIC_implementation_of_Chisel-generated_System_on_a_Chip
   - https://openlane.readthedocs.io/en/latest/docs/source/advanced_readme.html
   - https://openlane-docs.readthedocs.io/en/rtd-develop/doc/OpenLANE_commands.html
2. Understanding IR drop
   - https://vlsi-backend-adventure.com/ir_analysis.html
3. Antenna Diodes
   - https://www.edn.com/antenna-violations-resolved-using-new-method/
4. Congestion fixes
   - https://lmr.fi/int/congestion-in-vlsi-physical-design-flow/
5. Fixing hold violations
   - https://vlsiuniverse.blogspot.com/2017/02/fixing-hold-violations.html?m=1
6. Understanding routing
   - https://www.ifte.de/books/eda/chap5.pdf

3 Project details

In this section we elaborate on the various aspects of our project.

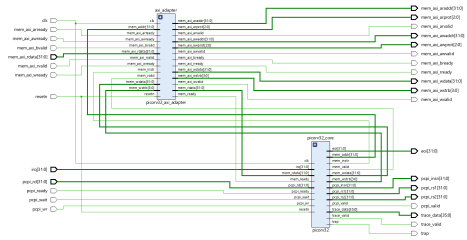
3.1 PicoRV32



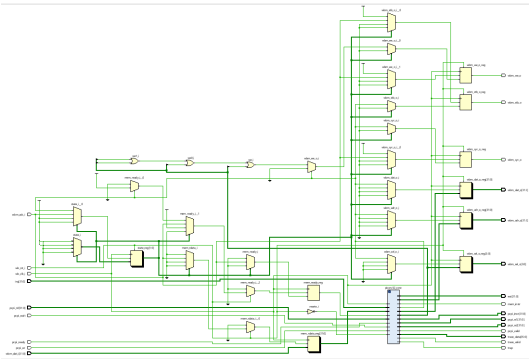
Base core

The PicoRV32 is a higly customizable core by YosysHQ.The PicoRV32 is a small core in terms of area but has high configurability. It can be configured as RV32E, RV32I, RV32IC, RV32IM, or RV32IMC core, and optionally contains a built-in interrupt controller. [5] On top of that the core supports multiple memory interfaces. The core exists in three variations: picorv32, picorv32_axi and picorv32_wb. The first provides a simple native memory interface, that is easy to use in simple environments. picorv32_axi provides an AXI-4 Lite Master [4] interface that can easily be integrated with existing systems that are already using the AXI standard. picorv32_wb provides a Wishbone master interface [5].

The core also includes a separate core picorv32_axi_adapter is provided to bridge between the native memory interface and AXI4. This core can be used to create custom cores that include one or more PicoRV32 cores together with local RAM, ROM, and memory-mapped peripherals, communicating with each other using the native interface, and communicating with the outside world via AXI4. [5]



Axi Interface Enabled



Wishbone interface enabled

PicoRV32 comes with internal verilog parameters which can be modified to generate our desired core. These internal parameters can be modified to implement additional features of the base core. When a parameter is set the corresponding logic or modules is generated using the *generate* keyword in verilog. Some parameters are explained below.

3.1.1 COMPRESSED_ISA

This enables support for the RISC-V Compressed Instruction Set [5]. By default it is 0.

3.1.2 ENABLE_PCPI

Set this to 1 to enable the Pico Co-Processor Interface (PCPI) [5]. By default it is 0.

3.1.3 ENABLE_MUL

This parameter internally enables PCPI and instantiates the *picorv32_pcpi_mul* core that implements the MUL[H[SU|U]] instructions. The external PCPI interface only becomes functional when ENABLE_PCPI is set as well [5]. By default it is 0.

3.1.4 ENABLE_FAST_MUL

This parameter internally enables PCPI and instantiates the *picorv32_pcpi_fast_mul* core that implements the MUL[H[SU|U]] instructions. The external PCPI interface only becomes functional when ENABLE_PCPI is set as well [5]. By default it is 0.

3.1.5 ENABLE_DIV

This parameter internally enables PCPI and instantiates the *picorv32_pcpi_div* core that implements the DIV[U]/REM[U] instructions. The external PCPI interface only becomes functional when ENABLE_PCPI is set as well [5].

3.1.6 TWO_CYCLE_COMPARE

This relaxes the longest data path a bit by adding an additional FF stage at the cost of adding an additional clock cycle delay to the conditional branch instructions. By default it is 0 [5].

Note: Enabling this parameter will be most effective when retiming (aka "register balancing") is enabled in the synthesis flow.

3.1.7 TWO_CYCLE_ALU

This adds an additional FF stage in the ALU data path, improving timing at the cost of an additional clock cycle for all instructions that use the ALU. By default it is 0 [5].

Note: Enabling this parameter will be most effective when retiming (aka "register balancing") is enabled in the synthesis flow.

3.2 Pico Co-Processor Interface (PCPI)

The thing that makes PicoRV32 ideal for use in microcontrollers is its Pico Co-Processor Interface (PCPI) feature. The PCPI is an interface that can be enabled by changing verilog parameters as mentioned above. PCPI helps adding additional functionality to the core easier provided they are non-branching instructions.

```
output        pcpi_valid
output [31:0] pcpi_insn
output [31:0] pcpi_rs1
output [31:0] pcpi_rs2
input         pcpi_wr
input  [31:0] pcpi_rd
input         pcpi_wait
input         pcpi_ready
```

When an unsupported instruction is found by PicoRV32 occurs it asserts *pcpi_valid*. The unsopported instruction is sent to *pcpi_insn* for the co-processor to recognise it. The decoded values of registers is made available through *pcpi_rs1* and *pcpi_rs2* and its output can be

sent to *pcpi_rd*. The *pcpi_ready* needs to asserted when the execution of the instruction is over.
When no external PCPI core acknowledges the instruction within 16 clock cycles, then an illegal instruction exception is raised and the respective interrupt handler is called. A PCPI core that needs more than a couple of cycles to execute an instruction, should assert *pcpi_wait* as soon as the instruction has been decoded successfully and keep it asserted until it asserts *pcpi_ready*. This will prevent the PicoRV32 core from raising an illegal instruction exception. [5]

3.3 Stages of the Flow

3.3.1 Synthesis

This is the very first stage of the flow. Most of the initial preparation happens in this stage. OpenLane first goes through the mentioned source files and looks through it for optimising it. Initially it tries to remove dead code. Dead code analysis involves unnecessary wires and modules that has never been used by any module. Dead code analysis is important as it helps OpenLane to focus on only the relevant verilog code. At the end of the synthesis stage the verilog file is replaced by optimized wires and standard modules by Skywater. We can also run synthesis exploration and understand which of the multiple variations of optimizations is better suited for our use case.

3.3.2 Floorplanning

Chip Floorplanning is the arrangement of logical block, library cells, pins on silicon chip. It makes sure that every module has been assigned an appropriate area and aspect ratio, every pin of the module has connection with other modules or periphery of the chip and modules are arranged in a way such that it consumes lesser area on a chip [2].
We also have power planning. Power planning is a step in which power grid network is created to distribute power to each part of the design equally. This step deals with the unwanted voltage drop and ground bounce. Steady state IR Drop is caused by the resistance of the metal wires comprising the power distribution network. By reducing the voltage difference between local power and ground, steady-state IR Drop reduces both the speed and noise immunity of the local cells and macros [2].

3.3.3 Placement

Placement is the step when it is actually decided where the different cells will be placed on the die. Placement does not just place the standard cells available in the synthesized netlist. It also optimizes the design, thereby removing any timing violations created due to the relative placement on die [2].

OpenLane does placement in two stages -

- Global Placement
- Detailed Placement

3.3.4 Clock Tree Synthesis

Clock Tree Synthesis(CTS) is a process which makes sure that the clock gets distributed evenly to all sequential elements in a design. The goal of CTS is to minimize the clock latency and skew [2]. There are several CTS techniques like:

- H - Tree
- X - Tree
- Fish bone

In OpenLANE, clock tree synthesis is carried out using TritonCTS tool. CTS should always be done after the floorplanning and placement as the CTS is carried out on a placement.def file that is created during placement stage [2].

3.3.5 Routing

OpenLANE uses TritonRoute, an open source router for modern industrial designs. The router consists of several main building blocks, including pin access analysis, track assignment, initial detailed routing, search and repair, and a DRC engine. The routing process is implemented in two stages:

- Global Routing - Routing guides are generated for interconnects.
- Detailed Routing - Tracks are generated interatively.

TritonRoute 14 ensures there are no DRC violations after routing.
OpenLane supports running multithreaded runs for routing.

3.4 config.tcl parameters

In this section, we elaborate on the config parameters we tampered with for optimizing our design.

3.4.1 CLOCK_PERIOD

This parameter is used to set the time period(in nanoseconds) of the clock signal used for timing the circuit. Note that we try reducing this value as much as possible for a faster design.
Default Value = "24.0"
Value used = "20"

3.4.2 CLOCK_PORT

The name of the design's clock port used in Static Timing Analysis.
Value = "clk".

3.4.3 FP_CORE_UTIL

This parameter is used to control the core utilization percentage. Note that having an overly high value of core utilisation in floor planning leads to routing congestion.
Default Value = "50"
Value used = "45"

3.4.4 ROUTING_CORES

Specifies the number of threads to be used in TritonRoute. Can be overriden via environment variable [3].
Default Value = "2"
Value used = "8"

3.4.5 PL_TARGET_DENSITY

The desired placement density of cells. It reflects how spread the cells would be on the core area. This value is chosen around 1-5% higher than FP_CORE_UTIL.
Default Value = "0.55"
Value used = "0.47"

3.4.6 CLK_BUFFER

This is the root clock buffer of the clock tree.
Default value = "sky130_fd_sc_hd__clkbuf_16"
Note that _16 refers to the drive strength of the clock buffer. We have decreased the drive strength to 8 in our design to help us deal with hold timing violations.
Value used = "sky130_fd_sc_hd__clkbuf_4"

3.4.7 FP_IO_MIN_DISTANCE

The minimum distance between the IO pads in microns.
Default Value = "3"
Value used = "3"

3.4.8 PL_ROUTABILITY_DRIVEN

Specifies whether the placer should use routability driven placement. 0 = false, 1 = true [3].
Default Value = "0"
Value used = "0"

3.4.9 SYNTH_STRATEGY

This parameter sets the strategy for abc logic synthesis and technology mapping. Possible values are DELAY/AREA 0-4/0-3; the first part refers to the optimization target of the synthesis strategy (area vs. delay) and the second one is an index.
Default Value = "AREA 0"
Value used = "AREA 2"

3.4.10 DIODE_INSERTION_STRATEGY

Specifies the insertion strategy of diodes to be used in the flow. 0 = No diode insertion, 1 = Spray diodes, 2 = insert fake diodes and replace them with real diodes if needed. 3= use FastRoute Antenna Avoidance flow, 4 = Use Sylvian's Custom Script for diode insertion on design pins and smartly inserting needed diodes inside the design, 5 = a mix of strategy 2 and 4.
Default Value = "3"
Value used = "3"
Note that diode insertion can help us deal with pin violations caused by antenna errors, however this will be at the cost of additional delay and leakage power consumption. We have set the value to 4 during trial 6 while trying to reduce the no. of pin violations due to antenna error.

3.4.11 SYNTH_MAX_FANOUT

The max load that the output ports can drive.
Default Value = "5"
Value used = "6"

3.4.12 GLB_RT_OVERFLOW_ITERS

The maximum number of iterations waiting for the overflow to reach the desired value.
Default Value = "64"
Value Used = "55"

3.4.13 DRT_OPT_ITERS

Specifies the maximum number of optimization iterations during Detailed Routing in TritonRoute.
Default Value = "64"
Value Used = "64"

3.4.14 SYNTH_CAP_LOAD

The capacitive load on the output ports in femtofarads.
Default Value = "33.5"
Value Used = "50"
Note that we can increase the capacitance to fix hold violations by increasing delay.

3.4.15 CTS_SINK_CLUSTERING_SIZE

Specifies the maximum number of sinks per cluster.
Default Value = "25"
Value Used = "60"

3.4.16 CTS_SINK_CLUSTERING_MAX_DIAMETER

Specifies maximum diameter (in micron) of sink cluster.
Default Value = "50"
Value Used = "60"

3.4.17 PL_RESIZER_HOLD_MAX_BUFFER_PERCENT

Specifies a max number of buffers to insert to fix hold violations. This number is calculated as a percentage of the number of instances in the design.
Default Value = "50"
Value Used = "99"

3.4.18 PL_RESIZER_SETUP_MAX_BUFFER_PERCENT

Specifies a max number of buffers to insert to fix hold violations. This number is calculated as a percentage of the number of instances in the design.
Default Value = "50"
Value Used = "99"

3.4.19 GLB_RESIZER_HOLD_MAX_BUFFER_PERCENT

Specifies a max number of buffers to insert to fix hold violations. This number is calculated as a percentage of the number of instances in the design.
Default Value = "50"
Value Used = "99"

3.4.20 GLB_RESIZER_SETUP_MAX_BUFFER_PERCENT

Specifies a max number of buffers to insert to fix setup violations. This number is calculated as a percentage of the number of instances in the design.
Default Value = "50"
Value Used = "99"

4 Errors and Fixes

4.1 Errors

- Routing congestion issues.
- [INFO GRT-0101] Running extra iterations to remove overflow.
  [INFO GRT-0103] Extra Run for hard benchmark.
- [ERROR PPL-0072] Number of pins (409) exceed number of valid positions (384).
  [ERROR DPL-0036] Detailed placement failed.
  Error: resizer.tcl, 79 DPL-0036
- [ERROR DPL-0036] Detailed placement failed. Error: resizer.tcl, 79 DPL-0036

4.2 Fixes

1. Fixes for routing congestion:
   - Reduce placement utilization or increase area if set manually.
2. Fix for abnormally high routing time:
   - Increase Area or fp_core_util
3. Fix for Insufficient pins:
   - Increase placement utilization
4. Possible Fixes for hold violations:
   - Increase Capacitance Increase core utilization percentage
   - Decrease number of CTS sinks
   - Decrease drive strength of buffers
   - Increase no. of buffers placed

   Note that most of the hold violation fixes reduce setup slack so there is a ideal balance between the two.
5. Possible Fixes for setup violations:
   - Reduced clock speed
   - Decrease Capacitance
   - Increase number of CTS sinks
   - Increase drive strength of buffers
   - Increase no. of buffers placed

5 Implementation

Since the usage of fastmul, axi, mul or div features increases the peak memory usage during routing we focused on improving the base core with pcpi enabled.
Our goal is to get the best possible clock frequency while reducing all issues that may occur.

We decided to use sky130_fd_sc_hd standard cell library for its high density cells. [1]
This library enables higher routed gated density, lower dynamic power consumption, comparable timing and leakage power. As a trade-off it has lower drive strength and does not support any drop in replacement medium or high speed library.

Other features of it include -

- sky130_fd_sc_hd includes clock-gating cells to reduce active power during non-sleep modes.
- Latches and flip-flops have scan equivalents to enable scan chain creation.
- Multi-voltage domain library cells are provided.
- Routed Gate Density is 160 kGates/mm^2 or better.
- Body Bias-able

Below is a table showing the PDK properties:

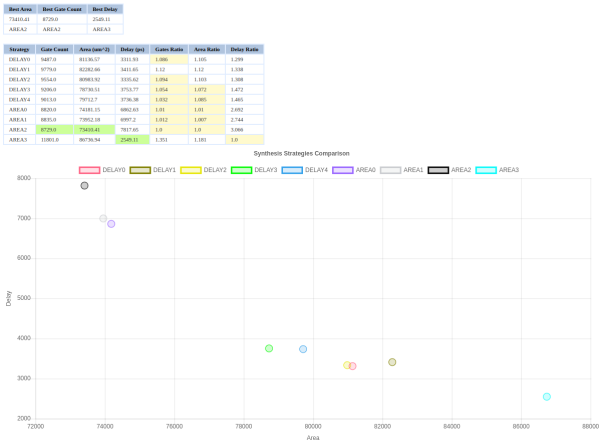
Details	Implemented
VDD	1.8V
NMOS devices used	sky130_fd_pr__nfet_01v8
PMOS devices used	sky130_fd_pr__pfet_01v8
inverters, buffers	56
AND, OR, NAND, NOR	153
XOR, XNOR	8
AND-OR-INV, OR-AND-INV	115
AND-OR, OR-AND	132
Adders, Comparators, Multiplexors	31
Latches and Flip-Flops	60
Custom power gating, bus cells	51

We ran the PicoRV32 with the following configuration :

```
/***************************************************************
 * picorv32
 ***************************************************************/

module picorv32_base #(
    parameter [ 0:0] ENABLE_COUNTERS = 1,
    parameter [ 0:0] ENABLE_COUNTERS64 = 1,
    parameter [ 0:0] ENABLE_REGS_16_31 = 1,
    parameter [ 0:0] ENABLE_REGS_DUALPORT = 1,
    parameter [ 0:0] LATCHED_MEM_RDATA = 0,
    parameter [ 0:0] TWO_STAGE_SHIFT = 1,
    parameter [ 0:0] BARREL_SHIFTER = 0,
    parameter [ 0:0] TWO_CYCLE_COMPARE = 1,
    parameter [ 0:0] TWO_CYCLE_ALU = 1,
    parameter [ 0:0] COMPRESSED_ISA = 1,
    parameter [ 0:0] CATCH_MISALIGN = 1,
    parameter [ 0:0] CATCH_ILLINSN = 1,
    parameter [ 0:0] ENABLE_PCPI = 1,
    parameter [ 0:0] ENABLE_MUL = 0,
    parameter [ 0:0] ENABLE_FAST_MUL = 1,
    parameter [ 0:0] ENABLE_DIV = 1,
    parameter [ 0:0] ENABLE_IRQ = 1,
    parameter [ 0:0] ENABLE_IRQ_QREGS = 1,
    parameter [ 0:0] ENABLE_IRQ_TIMER = 1,
    parameter [ 0:0] ENABLE_TRACE = 0,
    parameter [ 0:0] REGS_INIT_ZERO = 0,
    parameter [31:0] MASKED_IRQ = 32'h 0000_0000,
    parameter [31:0] LATCHED_IRQ = 32'h ffff_ffff,
    parameter [31:0] PROGADDR_RESET = 32'h 0000_0000,
    parameter [31:0] PROGADDR_IRQ = 32'h 0000_0010,
    parameter [31:0] STACKADDR = 32'h ffff_ffff
) (
    input clk, resetn,
```

6 Results



Synthesis exploration of base core

6.1 Outputs of interactive flow

6.1.1 .\flow.tcl – interactive

This command is used to enter the interactive mode of openlane.

6.1.2 prep -design simple_risc

It prepares a run in openlane or loads a previously stopped run in order to proceed with it. It calls trim_lib, prep_lefs, source_config, and other procs to set all the needed environment variables. It has similar flags to .\flow.tcl

6.1.3 run_synthesis

Runs yosys synthesis on the current design as well as OpenSTA timing analysis

on the generated netlist. The logs are produced under /<run_path</logs/synthesis/, the timing reports are under /<run_path</reports/synthesis/, and the synthesized netlist under /<run_path</results/synthesis/
We observed that our worst slack for setup was 6.31 ns and worst slack for hold was 0.16 ns.

```
==========
report_checks –slack_max -0.01
==========
No paths found.
check_report_end
check_slew
==========
report_check_types -max_slew -max_cap -max_fanout -violators
==========
==========
max slew violation count 0
max fanout violation count 0
max cap violation count 0
==========
check_slew_end
tns_report
==========
report_tns
==========
tns 0.00
tns_report_end
wns_report
==========
report_wns
==========
wns 0.00
wns_report_end
worst_slack
==========
report_worst_slack -max (Setup)
==========
worst slack 6.31
==========
report_worst_slack -min (Hold)
==========
worst slack 0.16
worst_slack_end
clock_skew
```

```
========================================================================
 report_clock_skew
========================================================================
Clock clk
Latency      CRPR       Skew
_39323_/CLK ^
  13.68
_39323_/CLK ^
  12.38      0.00       1.30

clock_skew_end
power_report

========================================================================
 report_power
========================================================================
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
------------------------------------------------------------------------
Sequential             4.74e-03   2.85e-04   1.81e-08   5.02e-03  29.1%
Combinational          6.69e-03   5.57e-03   5.89e-08   1.23e-02  70.9%
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                  1.14e-02   5.86e-03   7.69e-08   1.73e-02 100.0%
                          66.1%      33.9%       0.0%
power_report_end
area_report

========================================================================
 report_design_area
========================================================================
Design area 209372 u^2 100% utilization.
area_report_end
```

6.1.4 run_floorplan

Runs init_floorplan, followed by one of the io placement functions: if ::env(FP_PIN_ORDER_CFG) is defined then place_io_ol is run; otherwise, if ::env(FP_CONTEXT_DEF) and ::env(FP_CONTEXT_LEF) are defined it runs place_contextualized_io, if nothing of those is defined then it runs the vanilla place_io. Then it runs tap_decap_or on the processed design. Finally, power grid is generated utilizing ::env(VDD_NETS), ::env(GND_NETS), and ::env(SYNTH_USE_PG_PINS_DEFINES) if they are defined, otherwise vanilla gen_pdn is used. The resulting files are under /<run_path>/tmp/floorplan/ and /<run_path>/results/floorplan/.

At the end of floorplanning stage we saw that our design had 409 pins; 21898 components and 162820 component-terminals; 22000 nets and 75187 connections; 500 endcaps; 6552 tapcells.

An IR report helps us understand the robustness of our PDN (Power Delivery Network). An IR drop is defined as the average of the peak currents in the power network multiplied by the effective resistance of the power supply pads to the center of the chip. Our IR report states that we have -
Worstcase voltage: 1.80e+00 V
Average IR drop : 3.54e-09 V
Worstcase IR drop: 5.10e-07 V

6.1.5 run_placement

Runs global placement (global_placement or random_global_placement based on the value of PL_RANDOM_GLB_PLACEMENT),

then applies the optional optimizations repair_wire_length followed by run_openPhySyn if enabled, then runs the detailed placement (detailed_placement_or).
We had our worst slack for setup as 4.28 and worst slack for hold as 0.17.
After Openlane finished with the resizing step, we had worst slack for setup as 5.59, and worst slack for hold is 0.20. All our timing violations are dealt with here.
Placement Analysis

---

total displacement 0.0 u
average displacement 0.0 u
max displacement 0.0 u
original HPWL 1130746.5 u
legalized HPWL 1146775.5 u
delta HPWL 1 %

[INFO DPL-0020] Mirrored 378 instances
[INFO DPL-0021] HPWL before 1146775.5 u
[INFO DPL-0022] HPWL after 1130746.5 u
[INFO DPL-0023] HPWL delta -1.4 %

6.1.6 run_cts

Runs clock tree synthesis using the openroad app on the processed design. The resulting file is under /<run_path>/results/cts/. It also generates a the updated netlist using yosys and stores the results under /<run_path>/results/synthesis and runs yosys logic verification if enabled.
Our design has -

- Total number of Clock Roots: 1.
- Total number of Buffers Inserted: 388.
- Total number of Clock Subnets: 388.
- Total number of Sinks: 2140.

report_check_types -max_slew -max_cap -max_fanout -violators
max slew violation count 0
max fanout violation count 0
max cap violation count 0
==========
check_slew_end
tns_report
==========
report_tns
==========
tns 0.00
tns_report_end
wns_report
==========
report_wns
==========
wns 0.00
wns_report_end
worst_slack
==========
report_worst_slack -max (Setup)
==========
worst slack 5.55
==========
report_worst_slack -min (Hold)
==========
worst slack 0.10
worst_slack_end
clock_skew

```
=========================================================================
 report_clock_skew
=========================================================================
Clock clk
Latency      CRPR       Skew
_39794_/CLK ^
   2.92
_40811_/CLK ^
   1.98     -0.09       0.86

clock_skew_end
power_report

=========================================================================
 report_power
=========================================================================
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
-------------------------------------------------------------------------
Sequential             4.68e-03   4.79e-04   1.81e-08   5.16e-03  20.1%
Combinational          8.88e-03   1.16e-02   7.61e-08   2.05e-02  79.9%
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                  1.36e-02   1.21e-02   9.43e-08   2.56e-02 100.0%
                          52.9%      47.1%       0.0%
power_report_end
area_report

=========================================================================
 report_design_area
=========================================================================
Design area 223111 u^2 48% utilization.
area_report_end
```

6.1.7 run_resizer_timing

This command resizes the layout to optimize timing. It is used to fix negative slack/timing violations.

6.1.8 run_routing

Runs diode insertion based on the strategy, followed by global_routing, then ins_fill_cells, detailed_routing, and finally SPEF extraction on the processed design. The resulting file is under /<run_path>/results/routing/. It also generates a pre_route netlist using yosys and stores the results under /<run_path>/results/synthesis, and it runs yosys logic verification if enabled.

```
======================== Slowest Corner ================================

Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
----------------------------------------------------------------
Sequential             3.63e-03   4.76e-04   3.20e-05   4.14e-03  18.6%
Combinational          7.03e-03   1.10e-02   1.05e-04   1.81e-02  81.4%
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                  1.07e-02   1.15e-02   1.37e-04   2.23e-02 100.0%
                          47.9%      51.5%       0.6%

======================== Typical Corner ================================

Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
----------------------------------------------------------------
Sequential             4.68e-03   6.06e-04   1.81e-08   5.29e-03  18.8%
Combinational          8.89e-03   1.40e-02   1.68e-07   2.29e-02  81.2%
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                  1.36e-02   1.46e-02   1.87e-07   2.82e-02 100.0%
                          48.2%      51.8%       0.0%


======================== Fastest Corner ================================

Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power
----------------------------------------------------------------
Sequential             5.44e-03   7.17e-04   3.40e-08   6.16e-03  18.8%
Combinational          1.01e-02   1.65e-02   2.55e-07   2.66e-02  81.2%
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                  1.56e-02   1.72e-02   2.89e-07   3.28e-02 100.0%
                          47.5%      52.5%       0.0%
power_report_end
area_report

===========================================================================
 report_design_area
===========================================================================
Design area 444930 u^2 96% utilization.
area_report_end
```

6.1.9 run_antenna_check

Used to check for pin violations due to antenna errors.
Number of pins violated: 165
Number of nets violated: 142

6.1.10 run_magic

Streams the final GDS and a mag view. The resulting file is under /<run_path>/results/magic/.

6.1.11 run_magic_drc

Runs a drc check on the CURRENT_DEF. The resulting file is under /<run_path>/logs/magic/magic.drc .

6.1.12 run_klayout

Generates the klayout view.

6.1.13 run_klayout_gds_xor

Generates xor between klayout and gds for comparison.

6.1.14 run_magic_spice_export

Runs spice extractions on the processed design. Based on the value of MAGIC_EXT_USE_GDS either the GDS or the DEF/LEF is used for the extraction. The resulting file is under /<run_path>/results/magic/ .

6.1.15 run_lvs

Runs an lvs check between an extracted spice netlist EXT_NETLIST (so run_magic_spice_export should be run before it.) and the current verilog netlist of the processed design CURRENT_NETLIST. The resulting file is under /<run_path>/results/lvs/ and /<run_path>/reports/lvs/. The LVS(Layout Version Schematic) could be on the block/cell level or on the device/transistor level, this is controlled by the extraction type set by MAGIC_EXT_USE_GDS. If the GDS is used in extraction then the LVS will be run down to the device/transistor level, otherwise it will be run on the block/cell level which is the default behavior in OpenLANE.

6.1.16 run_lef_cvc

Runs CVC on the output spice, which is a Circuit Validity Checker. Voltage aware ERC checker for CDL netlists. The output files exist under <run-path>/results/cvc/. It is controlled by ::env(RUN_CVC)
Power nets 81711
! Checking overvoltage errors
! Checking nmos possible leak errors:
! Checking pmos possible leak errors:
! Checking mos floating input errors:
! Checking expected values:
CVC: Error Counts
CVC: Fuse Problems: 0
CVC: Min Voltage Conflicts: 0
CVC: Max Voltage Conflicts: 0
CVC: Leaks: 0
CVC: LDD drain->source: 0
CVC: HI-Z Inputs: 0
CVC: Forward Bias Diodes: 0
CVC: NMOS Source vs Bulk: 0
CVC: NMOS Gate vs Source: 0
CVC: NMOS Possible Leaks: 0
CVC: PMOS Source vs Bulk: 0
CVC: PMOS Gate vs Source: 0
CVC: PMOS Possible Leaks: 0
CVC: Overvoltage-VBG: 0
CVC: Overvoltage-VBS: 0
CVC: Overvoltage-VDS: 0
CVC: Overvoltage-VGS: 0
CVC: Model errors: 0
CVC: Unexpected voltage : 0
CVC: Total: 0
Usage Total: Time: 1 Memory: 88824 I/O: 80
Swap: 0
Virtual net update/access 913493/25695476
CVC: Log output to /open-lane/designs/picorv32_base/runs/RUN_2022.04.10_15.39.50/repo
CVC: End: Sun Apr 10 17:00:43 2022

6.1.17 calc_total_runtime

Calculates the total runtime from the starting point of the prep -design command.

6.1.18 save_state

Saves the current state of layout and netlist.

6.1.19 generate_final_summary_report

Generates a final summary csv report of the most important statistics and configurations in the run as well as a manufacturability report with the summary of DRC, LVS, and Antenna violations. This command is controlled by the flag $::env(GENERATE_FINAL_SUMMARY_REPORT)

6.1.20 check_timing_violations

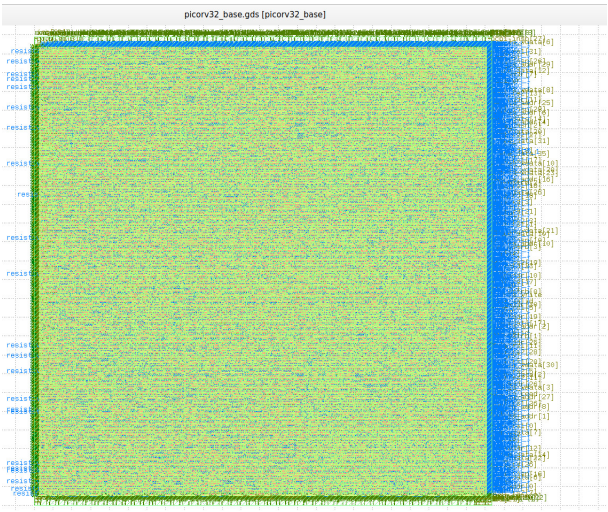
This command checks for any timing violations in the generated layouts.

6.1.21 Final summary report

Final summary report for AREA 2

Details	Implemented
VDD	1.8V
Die Area in mm^2	0.4878874905
Number of vias	211597
Wire Length	1381424
Half Perimeter wire length	1079684160
Wire count	18672
Wire bits	22310
public wires count	241
public wire bits	2984
Total Physical Cells	7052
Suggested clock frequency	50 Mhz
Suggested clock Period	20ns

6.1.22 KLayout visualization



Conclusions

The Openlane flow of PicoRV32 reported hold violations at lower clock periods which could be a consequence of this processor being designed for technologies with higher threshold voltage than sky130A.Majority of the flow runtime is utilized by routing, which is expected for a large design.

7 Future Work

Use of a better PDK with PDK with standard cells of higher drive strength can help improve our results and reduce the number of buffers required.

References

[1] Google. Skywater open source pdk. https://github.com/google/skywater-pdk, 2020.

[2] Shon Taware. Opensource physical design. https://github.com/ShonTaware/OpenSource_Physical_Design, 2022.

[3] The-OpenROAD-Project. The-openroad-project/openlane: Openlane is an automated rtl to gdsii flow based on several components including openroad, yosys, magic, netgen, fault and custom methodology scripts for design exploration and optimization. https://github.com/The-OpenROAD-Project/OpenLane.

[4] Xilinx. Amba axi4 interface protocol. https://www.edn.com/antenna-violations-resolved-using-new-method/, 2015.

[5] Yosys. Picorv32 - a size-optimized risc-v cpu. https://github.com/YosysHQ/picorv32, 2022.