Design and Verification of a Pipelined RV32I Processor

The purpose of this challenge is to design, implement, and verify a complete pipelined RV32I processor core in SystemVerilog. The design must correctly support the full RV32I instruction set as defined in the official RISC-V specification (https://riscv.org/technical/specifications/), with the exception of ECALL, EBREAK, FENCE, FENCE.TSO, and PAUSE. For these instructions, as well as for any unknown or invalid instruction, the processor must assert an illegal instruction flag.

The processor should be implemented as a pipelined microarchitecture. The depth of the pipeline is left to the designer's discretion, but correct handling of hazards (data, control, and structural) is mandatory to ensure both correctness and high performance. The design should target high frequency operation while maintaining reasonable area and power efficiency, as the ultimate goal is to achieve a processor suitable for AI-centric edge workloads.

The program counter must reset to **0x00000000** when the design is compiled without the tracer defined. When the simulation is compiled with the **+define+tracer** option (discussed later in the *Tracer Integration* section), the program counter must instead reset to **0x80000000**.

Processor Interface

The processor must expose the following external interface signals:

- input logic clk_i – clock input
- input logic resetn_i – active-low reset
- output logic illegal_inst_o - should get asserted for any unknown instruction
- output logic [31:0] imem_addr_o – instruction memory address output
- input logic [31:0] imem_inst_i – instruction memory data input

- output logic [31:0] mem_addr_o – data memory address output
- output logic [31:0] mem_dat_o – data memory write data output
- input logic [31:0] mem_dat_i – data memory read data input
- output logic mem_write_o – memory write enable
- output logic mem_wstrb_o – memory write strobe signal
- output logic mem_read_o – memory read enable
- input logic mem_ack_i – memory acknowledge (indicates readiness of memory data transfer)

The top module must be named **rv32i_top**. Any design with a different top-level name will not be considered for evaluation.

Tracer Integration

A tracer module will be provided to all miners, along with its required packages (pkg.sv, tracer_pkg.sv) and support files (imem.sv, dmem.sv, testbench files). These will allow you to test your design locally, but do not need to be submitted.

The tracer must be instantiated in the datapath of your design to capture every retired instruction. The log must show instructions in program order and only when they are retired (i.e., after they commit their result to the register file). This log will be compared against a golden Spike reference during evaluation.

The tracer and all associated logic must be conditionally included using `ifdef tracer`. This ensures that tracer logic is only present during simulation and does not impact synthesis, area, or frequency.

Here is the example tracer instance, with the critical signals being connected to the tracer module. The signals that are not connected here are not required from our side, but you can connect those as well, to get the better log file with more detail for your own debugging.

```
`ifdef tracer
    tracer tracer_inst (
        .clk_i          (clk),
```

```
        .rst_ni          (reset_n),
        .hart_id_i       (1),
        .rvfi_insn_t     (rvfi_insn),
        .rvfi_rs1_addr_t (rvfi_rs1_addr),
        .rvfi_rs2_addr_t (rvfi_rs2_addr),
        .rvfi_rs3_addr_t (),
        .rvfi_rs3_rdata_t(),
        .rvfi_mem_rmask  (),
        .rvfi_mem_wmask  (),
        .rvfi_rs1_rdata_t(rvfi_rs1_rdata),
        .rvfi_rs2_rdata_t(rvfi_rs2_rdata),
        .rvfi_rd_addr_t  (rvfi_rd_addr),
        .rvfi_rd_wdata_t (rvfi_rd_wdata),
        .rvfi_pc_rdata_t (rvfi_pc_rdata),
        .rvfi_pc_wdata_t (rvfi_pc_wdata),
        .rvfi_mem_addr   (rvfi_mem_addr),
        .rvfi_mem_wdata  (rvfi_mem_wdata),
        .rvfi_mem_rdata  (rvfi_mem_rdata),
        .rvfi_valid      (rvfi_valid)
    );
`endif
```

The tracer logic itself does not need to be synthesizable and may use non-synthesizable constructs. When running OpenLane synthesis, tracer logic will be excluded automatically.

For reference on tracer-style signal interfaces, miners may review the RISC-V Formal Interface (RVFI):
👉 https://github.com/SymbioticEDA/riscv-formal/blob/master/docs/rvfi.md

Submission Requirements

Miners must submit a single zip archive containing:

- RTL source files of the processor.
- A rtl.f file at the root listing all RTL source files in proper compilation order.

No testbench files, tracer files, or memory models are required in the submission. These will be attached during evaluation.

Functional Verification

Functional correctness is evaluated using **Verilator** with the tracer enabled (+define+tracer).

- The **functionality score** is derived from the ratio of correctly executed instructions compared against a Spike golden reference.
- If the tracer is missing, not connected, or fails to log instructions, the functionality score will be **zero**.
- If the functionality score is **below 0.7**, the overall score will also be **zero**, and the design will not be eligible for emissions.

Local Verification Flow

You can and should test your design locally before submission. The following workflow is supported:

1. The tb_top.sv, imem.sv , dmem.sv and all the tracer related files are already attached to help you out in setting up the verification environment.
2. You will have to initialize the imem and dmem memories by modifying the path to .mem files in the **$readmemh** system call in both the imem.sv and dmem.sv files.
3. Using the provided files, and you design, run Verilator simulation with tracer enabled to generate a **log file**.

4. Use the provided script core_log_to_trace_csv.py (in the scripts folder) to convert the log into CSV format.
5. Compare this CSV against the Spike reference CSV using the provided instr_trace_compare.py.
6. We have already provide you multiple test cases in the form of **dmem_*.mem** and **imem_*.mem** files along with the reference golden output file generated by the Spike simulator.

Note: The provided test cases are intended only as examples to help you understand the types of instructions and scenarios that will be used during evaluation. In practice, your processor will be tested against a much larger and more diverse set of cases. For broader coverage and to better prepare your design, you are encouraged to use riscv-dv to generate randomized instruction sequences:

👉 https://github.com/chipsalliance/riscv-dv

For a detailed explanation of the evaluation environment, directory structure, and testing methodology, please refer to the accompanying **Processor Validation Guide** pdf document.

Synthesis and Implementation

Your design must be synthesizable with **Yosys** and implementable in **OpenLane** using the **sky130A PDK**.

- If synthesis fails, or the design includes unsupported constructs, the overall score will be **zero**.
- OpenLane reports will provide area (μm²), worst negative slack (WNS), and achievable frequency (Fmax).
- Designs that are not synthesizable, or that fail timing closure, will not be eligible for ranking.

Tracer logic will not be included in synthesis runs, ensuring that functional verification hooks do not affect PPA results.

Scoring

Final scores are based on the following criteria:

- **Functionality (50%)** – Must meet or exceed the 0.7 threshold to qualify. Any design with tracer missing, incorrect logging, or synthesis failure will receive zero overall score.
- **Performance (25%)** – Measured as the actual execution time for the test program. This is calculated using the number of clock cycles taken (from simulation) multiplied by the time for one clock cycle (derived from the maximum frequency reported by OpenLane). Designs that both run with fewer cycles and achieve higher frequency will score higher.
- **Area Efficiency (25%)** – Smaller area relative to the reference target will score higher.
- **Power (0%)** – Not weighted in this challenge.

If the functionality gate is not met (score < 0.7, tracer missing, or synthesis failure), the overall score is zero regardless of area or performance.

Note: The evaluation weights or functionality gate may be adjusted during the course of the challenge as submissions improve. Any such changes will be communicated promptly on the Discord channel.

Recommendations for Miners

Designers are strongly encouraged to:

- Always run Verilator with **+define+tracer** before submission to ensure logs are generated.
- Confirm that tracer logs match Spike reference outputs using the provided scripts.

- Verify that instructions are logged only at retirement and in correct program order.
- Use riscv-dv to explore edge cases and corner conditions.
- Run Yosys and OpenLane locally to ensure your design is synthesizable and to estimate area and timing.

Disqualification Conditions

Your design will be disqualified (score = 0) if:

- The top module is not named rv32i_top.
- The required processor interface is not implemented exactly.
- No tracer is instantiated in the datapath.
- The tracer does not generate valid logs under +define+tracer.
- The functionality score is below 0.7.
- The design fails Yosys synthesis or cannot complete OpenLane flow.