Project Formulation: High-Density 16-Bit RISC-V Architecture Implementation Strategy on SkyWater 130nm via Industrial ASIC Flows

1. Strategic Overview and Project Charter

1.1 The Convergence of Open Silicon and Industrial Methodology

The democratization of semiconductor engineering, catalyzed by the release of the SkyWater 130nm (Sky130) Process Design Kit (PDK) and the advent of multi-project wafer (MPW) aggregation platforms like Tiny Tapeout, has fundamentally altered the accessibility landscape for custom silicon.[1] However, a significant dichotomy remains between the "hobbyist" open-source flow (typically utilizing OpenLane/OpenROAD) and the rigorous, area-optimized methodologies employed in commercial integrated circuit (IC) design.[2] This project formulation bridges that gap. It proposes the design of a **16-bit Datapath RISC-V Processor** specifically engineered to fit within the stringent **2-tile area constraint** of the Tiny Tapeout shuttle, leveraging **industrial Electronic Design Automation (EDA) tools** (Synopsys Design Compiler and Cadence Innovus) to maximize gate density and yield.

The core technical challenge addressed herein is the mapping of a standard 32-bit Instruction Set Architecture (ISA)—specifically the RISC-V RV32I or RV32E—onto a physically constrained $130nm$ planar CMOS technology. The Sky130 process, while robust, lacks the transistor density of modern FinFET nodes, featuring a standard cell height of $2.72\mu m$ and a poly pitch typical of mature nodes.[3] A standard 32-bit processor implementation often exceeds the gate budget of a single Tiny Tapeout tile. Consequently, this report articulates a "Split-Datapath" microarchitecture that processes 32-bit instructions in 16-bit time-multiplexed chunks, halving the combinational area requirements while maintaining full software compatibility with the standard RISC-V toolchain.[4]

1.2 Defining the Physical Constraints

The Tiny Tapeout platform imposes rigid boundaries on the physical implementation, which drive the architectural decisions. Understanding these constraints is prerequisite to defining the microarchitecture.

Area Constraints:

The project targets a **2-tile configuration**. According to the Tiny Tapeout specifications for the

Sky130 PDK, a single tile measures approximately $160\mu m \times 110\mu m$. However, when aggregated into a 1×2 (vertical) or 2×1 (horizontal) block, the usable area scales non-linearly due to the shared boundaries and overhead.

- **1x2 Configuration:** $160\mu m \times 225\mu m$.[6]
- **2x2 Configuration:** $334\mu m \times 225\mu m$.[6] For the purpose of this formulation, the 1×2 **vertical configuration** is selected to maximize the aspect ratio for datapath bit-slicing (16 bits fits well vertically) while adhering to the user's "2-tile maximum" constraint. This yields a total theoretical gross area of $36,000\mu m^2$.

Gate Budget Estimation: The Sky130 High Density (sky130_fd_sc_hd) standard cell library has a typical NAND2 gate area of roughly $3\mu m^2$. While the raw area suggests a capacity of 12,000 gates, practical utilization in Sky130 is severely limited by routing congestion due to the availability of only 5 metal layers.[7] Realistic utilization rates for logic-heavy designs hover between 50% and 60%. Thus, the effective gate budget is approximately **3,000 to 4,000 logic cells**.[8] A standard 32-bit RISC-V core (e.g., PicoRV32) typically demands 2,500+ cells purely for logic, excluding the register file, creating a high risk of congestion failure.[9]

I/O Constraints:

The design must interface with the Tiny Tapeout Multiplexer. This architecture abstracts the pad ring, providing the user project with a limited set of ports:

- **Inputs:** 8 dedicated (ui_in).
- **Outputs:** 8 dedicated (uo_out).
- **Bidirectional:** 8 programmable (uio_in/uio_out).
- **Control:** Clock, Reset, and Enable. Total bandwidth is limited to roughly 24 active signal pins.[10] This reinforces the decision for a 16-bit internal architecture, as a full 32-bit external bus is physically impossible without complex serialization.

1.3 Project Deliverables and Scope

The proposed project scope encompasses the full ASIC design lifecycle:

1. **Architecture:** Specification of a multi-cycle 16-bit datapath implementing the RV32E ISA.
2. **RTL Design:** SystemVerilog implementation optimized for Synopsys Design Compiler.
3. **Verification:** A UVM-based testbench integrated with the Spike instruction set simulator (ISS).
4. **Implementation:** Synthesis and Place-and-Route (PnR) scripts for Synopsys and Cadence tools, specifically adapted for the open-source PDK.
5. **Signoff:** Static Timing Analysis (STA) and Physical Verification (DRC/LVS) targeting the Tiny

Tapeout slot requirements.

---

2. Architectural Specification: The "MicroCore-16"

2.1 The Case for RV32E and 16-Bit Datapath

To reconcile the 32-bit ISA with the 4,000-gate budget, the architecture must reduce the two largest consumers of silicon area: the **Register File (RF)** and the **Arithmetic Logic Unit (ALU)**.

2.1.1 ISA Selection: RV32E vs. RV32I

The standard RV32I ISA mandates 32 general-purpose registers (GPRs). In the Sky130 hd library, a D-Flip-Flop (DFF) occupies approximately $12 - 15\mu m^2$.

- **RV32I RF Area:** $32 \text{ regs} \times 32 \text{ bits} \times 15\mu m^2 \approx 15,360\mu m^2$.
  - This consumes nearly **45%** of the total available silicon area of a 2-tile block, leaving insufficient room for ALUs, controllers, and routing overhead.
- **RV32E RF Area:** The RV32E (Embedded) extension reduces the GPR count to 16.
  - $16 \text{ regs} \times 32 \text{ bits} \times 15\mu m^2 \approx 7,680\mu m^2$.
  - This reduces the RF area burden to ~22%, making the design feasible.[5]

Decision: The project will implement **RV32E**. This maintains compatibility with standard compilers (e.g., GCC/LLVM via -march=rv32e) while reclaiming significant area.

2.1.2 Datapath Strategy: 16-Bit Multi-Cycle

While bit-serial implementations like SERV offer the ultimate area savings (executing 1 bit per cycle), they suffer from extreme latency (32+ cycles per instruction) and high control logic overhead relative to the datapath size.[4] A 16-bit datapath offers a "Goldilocks" compromise: it halves the area of combinational logic compared to a 32-bit core while requiring only 2 cycles for most integer operations, maintaining acceptable throughput.[9]

Operation Principle:

The core processes 32-bit instructions in two 16-bit "chunks" (Lower Word and Upper Word).

- **ALU Width:** 16 bits.
- **Internal Buses:** 16 bits.
- **Register Access:** The 32-bit registers are accessed as pairs of 16-bit values.

Table 1: Comparative Analysis of Architecture Options for Sky130 2-Tile Implementation

Metric	**Full 32-Bit (PicoRV32)**	**Bit-Serial (SERV)**	**Proposed 16-Bit Split**
Logic Gate Count	~2,500+	~200 - 500	**~1,200 - 1,500**
Register File Area	High (100%)	Low (uses RAM)	**Medium (uses RV32E)**
Cycles per ADD	1	32	**2**
Routing Congestion	Severe	Very Low	**Moderate**
I/O Complexity	Requires Muxing	Native Serial	**Native 16-bit**
Feasibility (2-Tile)	Low Risk of Fit	High Margin	**Optimal Fit**

2.2 Micro-Architecture Implementation Details

2.2.1 Fetch and Decode Unit

The instruction fetch unit acts as an adapter between the 32-bit instruction requirement and the 16-bit memory interface.

- **State 0 (FETCH_L):** Drive PC[15:0] to address bus. Latch incoming data to IR[15:0]. Increment PC + 2.
- **State 1 (FETCH_H):** Drive PC[15:0] to address bus. Latch incoming data to IR[31:16]. Increment PC + 2.
- **State 2 (DECODE):** The full 32-bit instruction is now available in the Instruction Register (IR). The Control Unit decodes the opcode to determine if the operation requires 1 pass (logic), 2 passes (arithmetic), or more (shifts).

2.2.2 The Split ALU

The Arithmetic Logic Unit is the heart of the area reduction strategy. It does not simply truncate operations; it must handle the propagation of state between the lower and upper 16-bit operations.

- **Addition/Subtraction:**
  - *Cycle 1:* Calculate A[15:0] + B[15:0]. Store result to Res[15:0]. Capture the **Carry Out** bit in a temporary flag register (C_flag).
  - *Cycle 2:* Calculate A[31:16] + B[31:16] + C_flag. Store result to Res[31:16].

- **Logical (AND/OR/XOR):**
  - These are bitwise independent. Cycle 1 and Cycle 2 operate identically without carry propagation.
- **Shifts (SLL/SRL/SRA):**
  - A 32-bit barrel shifter is area-prohibitive ($O(N^2)$ complexity).
  - *Optimization:* The design will utilize a **serial shifter**. A single 16-bit shift register performs the operation over multiple clock cycles. While this increases latency (up to 32 cycles), it reduces the shifter area to near zero, utilizing the existing datapath registers.[9]

2.2.3 Memory Interface and Pin Muxing

The Tiny Tapeout interface provides 8 input pins (ui_in) and 8 bidirectional pins (uio_in). This allows for a raw 16-bit input channel.

- **Input Mode:** When reading data or instructions, ui_in carries bits [7:0] and uio_in carries bits [15:8].
- **Output Mode:** uo_out carries bits [7:0] and uio_out carries bits [15:8] of the address or write data.
- **Control:** The core generates a dir_select signal to switch the bidirectional pads between address driving (Output) and data reading (Input).

---

3. The Industrial-Open Bridge: Toolchain Methodology

The prompt explicitly requests the use of **industrial tools**. While the Sky130 PDK is open-source, it is primarily distributed in formats optimized for open-source tools (Magic, Netgen). Bridging this to Synopsys/Cadence requires a specific data preparation methodology.[13]

3.1 PDK Migration to Industrial Formats

The "raw" open-source PDK lacks the compiled databases required by commercial tools. The following setup phase is critical before any design work begins.

3.1.1 Logic Library Preparation (Synopsys DC)

Synopsys Design Compiler cannot read raw .lib files efficiently if they contain non-standard attributes often found in open PDKs.

1. **Sanitization:** Scripts must parse the sky130_fd_sc_hd__tt_025C_1v80.lib to remove "noise" attributes (like CCS noise tables if incomplete) that cause the Library Compiler to fail.[14]
2. **Compilation (.db):** Use the Synopsys lc_shell to compile the sanitized Liberty file into the Synopsys Database format.

```Tcl
# Tcl snippet for lc_shell
read_lib sky130_fd_sc_hd__tt_025C_1v80_sanitized.lib
write_lib sky130_fd_sc_hd__tt_025C_1v80 -format db -output
sky130_fd_sc_hd__tt_025C_1v80.db
```

This step ensures that the synthesis engine has accurate timing and power models.

3.1.2 Physical Library Preparation (Cadence Innovus)

Innovus requires LEF files for physical abstraction and .captable files for parasitic extraction.

- **LEF:** The standard sky130_fd_sc_hd.lef is compatible. However, the Technology LEF (rtk-tech.lef) must be generated to define the routing layers (Met1-Met5) and via definitions strictly according to the PDK rules. The mflowgen ADK scripts provide a reliable source for this generated LEF.[13]
- **Captables:** Parasitic extraction is vital for timing closure at 130nm. The .captable file is generated from the PDK's Interconnect Technology File (ITF) using the Cadence generateCapTbl utility. This allows the tool to accurately calculate wire delays, which are significant in 130nm due to the resistance of aluminum interconnects.

3.2 Toolchain Environment Configuration

A robust environment setup script is required to point the tools to these generated artifacts.

- **Synopsys:** Set search_path to include the directory containing the compiled .db files. Define target_library and link_library to point to the Sky130 HD standard cells.
- **Cadence:** Set init_lef_file to the merged list of Tech LEF and Macro LEFs. Set init_mmmc_file to the Multi-Mode Multi-Corner timing constraint file.

---

4. Front-End Design and Area Optimization

4.1 SystemVerilog Coding for Area Minimization

In an area-constrained environment, the RTL coding style directly influences the synthesis quality. The design employs "Resource Sharing" at the RTL level to guide the synthesizer.

Bad Coding Style (Area Expensive):

Code snippet

```
// Infers two separate adders
assign next_pc = pc + 4;
assign alu_res = rs1 + rs2;
```

Optimized Coding Style (Area Efficient):

The design should multiplex inputs to a single adder instance.

Code snippet

```
// Infers one adder with multiplexed inputs
always_comb begin
   case (state)
      FETCH:   {op_a, op_b} = {pc, 16'd2};
      EXECUTE: {op_a, op_b} = {rs1, rs2};
      default: {op_a, op_b} = {16'b0, 16'b0};
   endcase
   adder_out = op_a + op_b;
end
```

By explicitly defining the resource sharing in SystemVerilog, the designer prevents the synthesis tool from inferring multiple arithmetic units that might be optimized away only at high effort levels.[15]

4.2 Synthesis Strategy with Synopsys Design Compiler

The synthesis stage is where the logical architecture is mapped to the physical gates. The strategy focuses on **Area Recovery**.

4.2.1 Constraints Management (SDC)

The Synopsys Design Constraints (SDC) file must reflect the Tiny Tapeout environment.

- **Clock Frequency:** The Tiny Tapeout IO pads are limited to ~50-66 MHz. However, the internal core can run faster. A conservative target of **50 MHz** is selected to ease timing pressure and allow the tool to focus on area.
  - create_clock -name core_clk -period 20.0 [get_ports clk]
- **Area Constraint:** This is the most critical directive.
  - set_max_area 0
  - Setting the max area to zero instructs the Design Compiler to continue optimizing for area indefinitely until no further reductions are possible, rather than stopping once a

specific gate count is reached.[17]

4.2.2 Optimization Directives

The synthesis script utilizes compile_ultra, Synopsys's advanced optimization engine.

- **Command:** compile_ultra -gate_clock -no_autoungroup
  - -gate_clock: Inserts Integrated Clock Gating (ICG) cells (sky130_fd_sc_hd__dlclk). This reduces the switching power and the size of the clock tree buffers.
  - **Boundary Optimization:** Disabled (set_boundary_optimization false) to ensure that the port interface remains exactly as specified for the Tiny Tapeout wrapper.[18]

4.2.3 Leakage vs. Dynamic Power Trade-off

While power is not the primary constraint, area is. High-Vt (HVT) cells typically have lower leakage but are slower. Low-Vt (LVT) cells are faster but leakier. In Sky130, the cell areas are generally similar, but the drive strengths differ. The synthesis library group should include all drive strengths to allow DC to downsize gates to the minimum drive strength required for the 50 MHz target, thereby saving area.

---

5. Verification Methodology: Ensuring Correctness without Silicon

Given the complexity of the split 16-bit datapath, verifying that the core behaves exactly like a standard 32-bit RISC-V processor is paramount. A simple directed test is insufficient.

5.1 UVM Architecture for Split-Datapath Verification

The project employs the **Universal Verification Methodology (UVM)** to create a robust verification environment.[19]

Table 2: UVM Environment Components

Component	Functionality	Adaptation for MicroCore-16
Agent	Drives/Monitors Pins	Drives 16-bit chunks; Monitors must reassemble 32-bit words.
Driver	Signal toggling	Handles the TDM protocol (splitting 32-bit transaction

		into two 16-bit bus cycles).
Monitor	Passive observation	Observes retire signal; reconstructs the full instruction and result for checking.
Scoreboard	Comparison Logic	Compares reconstructed DUT output against the Reference Model.
Ref Model	Golden Standard	**Spike ISS** (Instruction Set Simulator) connected via DPI-C.

5.2 Spike Integration via DPI-C

The **Spike ISS** serves as the golden reference. It executes RISC-V instructions functionally.

- **Mechanism:** The UVM testbench uses the SystemVerilog Direct Programming Interface (DPI-C) to step the Spike simulator.
- **Lock-Step Verification:**
  1. The DUT executes an instruction (taking e.g., 6 clock cycles).
  2. Upon completion (uo_out signals valid writeback), the UVM Monitor captures the result.
  3. The Scoreboard calls spike_step(1) via DPI.
  4. The Scoreboard retrieves the register state from Spike and compares it with the DUT's result.
  5. Any mismatch asserts a UVM_FATAL error. This methodology ensures that the custom 16-bit microarchitecture is architecturally identical to the standard.[21]

5.3 Formal Verification Strategy

To complement simulation, **Formal Property Verification (FPV)** is used to prove the absence of deadlock and hazard conditions.

- **Tool:** Synopsys VC Formal or SymbiYosys (open source).
- **RISC-V Formal Interface (RVFI):** The core implements the RVFI port, a side-channel bus that broadcasts the internal commit state.
- **Properties:** The riscv-formal framework checks properties like:
  - *PC Consistency:* Does the PC increment correctly after every non-branch instruction?
  - *Register Liveness:* Is the value read from R1 the last value written to R1?

    - This is particularly important for verifying the 16-bit "Carry" flag propagation logic, ensuring that the upper half of an ADD always receives the correct carry from the lower half.[22]

---

6. Physical Design and Implementation (Place & Route)

The Physical Design (PD) phase determines whether the synthesized logic fits within the 2-tile boundary. This is performed using **Cadence Innovus**.

6.1 Floorplanning for Tiny Tapeout

Unlike a standard rectangular block, the Tiny Tapeout floorplan is pre-defined.

- **Template Import:** The design begins by importing the tt_analog_1x2.def (or digital equivalent) provided by the platform. This DEF file contains the fixed locations of the I/O pins and the pre-routed Power Distribution Network (PDN) on Metal 4 and Metal 5.[6]
- **Placement Constraints:**
    - **Core Area:** Defined inside the power ring.
    - **Pin Alignment:** The RTL ports (ui_in, uo_out) must be physically assigned to the locations specified in the DEF. In Innovus, this is done by applying a fixed status to the imported pin locations.
    - **Obstructions:** Metal 5 is strictly off-limits for signal routing as it carries the vertical power straps for the whole chip. A routing obstruction must be created: create_route_type -name top_power -layer Metal5 -shield_net VDD.

6.2 Placement and Congestion Management

Congestion is the primary risk in Sky130 due to the low metal count.

- **Cell Padding:** To alleviate congestion, cell padding (halos) can be applied to complex cells (like Flip-Flops) to force the placer to spread them out, reserving space for routing vias.
    - *Command:* specifyCellPad * 2 (Adds 2 sites of padding).
- **Tap Cells:** Sky130 requires substrate tap cells (sky130_fd_sc_hd__tapvpwrvgnd_1) to be placed in a regular grid (checkerboard) to prevent latch-up. Innovus addWellTap command is used to insert these every ~13 microns.[24]

6.3 Clock Tree Synthesis (CTS)

The clock tree consumes significant area and power.

- **Strategy:** Use a "Fishbone" or H-Tree topology.
- **Target:** Skew < 500ps. Given the small die size, this is easily achievable.
- **Buffers:** Restrict the CTS tool to use sky130_fd_sc_hd__clkbuf_2 and _4. Avoid the largest drivers (_16) as they cause excessive electromigration (EM) risk on the thin power rails of the standard cells.[25]

6.4 Detailed Routing and Antenna Repair

The Sky130 process is sensitive to **Process Antenna Effects**—charge accumulation on long floating wires during plasma etching that can destroy gate oxides.

- **Detection:** Innovus calculates antenna ratios during routing.
- **Repair:** The "Diode Insertion" method is mandatory. When a violation is found, the router acts to:
  1. Jump to a higher metal layer (jumping up reduces the antenna ratio of the lower segment).
  2. Insert a reverse-biased diode cell (sky130_fd_sc_hd__diode_2) near the gate input.
  - *Area Impact:* Diode insertion can consume 5-10% of the available area. The floorplan utilization target must be kept below 70% to leave room for these diodes.[26]

---

7. Signoff and Submission

7.1 Static Timing Analysis (STA)

Signoff timing is performed using **Synopsys PrimeTime** or the Innovus built-in engine.

- **Corners:** Timing must be closed at the "Worst" corner (Slow-Slow Process, 1.6V, 100°C) to ensure robustness.
- **Setup/Hold:** Hold time violations are critical. Sky130 allows for hold fixing by inserting delay buffers (sky130_fd_sc_hd__dlygate). This must be done post-route.

7.2 Physical Verification (DRC/LVS)

Before submission, the design must pass Design Rule Checks (DRC) and Layout vs. Schematic (LVS).

- **Tool:** Mentor Calibre is the gold standard, but for Sky130, **Magic** and **KLayout** are the reference signoff tools for the shuttle.
- **Integration:** The Innovus-generated GDSII is streamed out and read into Magic. The specific Tiny Tapeout DRC deck is run. Common errors include "Metal 1 spacing" (due to dense pin access) and "Latch-up distance" (missing tap cells).

7.3 Submission to Tiny Tapeout

The final artifact is a GDSII file and a Verilog netlist.

- **Wrapper Integration:** The user logic is instantiated inside the tt_user_project wrapper.
- **GitHub Action:** Tiny Tapeout uses a GitHub Actions CI/CD pipeline. Even though the design was built with industrial tools, the GDS is committed to the repository. The pipeline runs a final "Pre-check" using OpenLane to verify that the GDS fits the slot and connects to the pins correctly.

8. Conclusion and Future Outlook

This report delineates a comprehensive methodology for implementing a **16-bit RISC-V processor** within the severe constraints of a **2-tile Tiny Tapeout** payload. The feasibility of this endeavor rests on two pillars:

1. **Microarchitectural Innovation:** The adoption of a 16-bit multi-cycle datapath and the RV32E ISA reduces the silicon area requirement by approximately 50% compared to standard implementations, fitting the ~4,000 gate budget of the Sky130 layout.
2. **Industrial Rigor:** The utilization of Synopsys Design Compiler and Cadence Innovus enables aggressive area optimization (set_max_area 0) and sophisticated congestion management that surpasses the capabilities of default open-source flows.

This hybrid approach—using industrial tools to target an open-source shuttle—represents a high-maturity design flow. It mitigates the risks of routing congestion and timing closure, ensuring that the "MicroCore-16" is not just a theoretical model, but a manufacturable, functional silicon reality. As 130nm nodes remain relevant for IoT and edge-compute, such area-optimized architectures define the baseline for cost-effective embedded computing.

9. Appendix: Technical Data and Constraints

Table 3: SkyWater 130nm (Sky130) Process Characteristics

Parameter	**Value**	**Implications for Design**
Feature Size	130 nm	Low density; requires larger area for logic.
Metal Layers	5 (Al)	High congestion risk; limits bus widths.
Std Cell Height	$2.72\mu n$	Determines row pitch and placement grid.
Gate Density	$\sim 60 -$	Roughly 1,000 gates per $160 \times$ tile.

Supply Voltage	1.8V (Core)	Higher power than modern nodes (0.8V).

Table 4: Tiny Tapeout 4+ Pin Mapping

RTL Signal	**TT Port**	**Direction**	**Function**
clk	clk	Input	System Clock (50 MHz)
rst_n	rst_n	Input	Active Low Reset
data_in[7:0]	ui_in[7:0]	Input	Data Bus Lower Byte
data_in[15:8]	uio_in[7:0]	Bidir (In)	Data Bus Upper Byte
data_out[7:0]	uo_out[7:0]	Output	Address/Data Out Lower
data_out[15:8]	uio_out[7:0]	Bidir (Out)	Address/Data Out Upper

Works cited

1. SKY's the Limit with the SKY130 Open-Source PDK - Skywater Technology, accessed February 18, 2026, https://www.skywatertechnology.com/sky130-open-source-pdk/
2. GreenRio: A Linux-Compatible RISC-V Processor Designed for Open-Source EDA Implementations, accessed February 18, 2026, https://riscv.org/blog/greenrio-a-linux-compatible-risc-v-processor-designed-for-open-source-eda-implementations/
3. 128 Tiny PLL, accessed February 18, 2026, https://www.tinytapeout.com/chips/ttsky25a/tt_um_tiny_pll
4. SERV: The Little CPU That Could - Tales from Beyond the Register Map, accessed February 18, 2026, http://blog.award-winning.me/2022/10/serv-little-cpu-that-could.html
5. (PDF) An Educational RISC-V-Based 16-Bit Processor - ResearchGate, accessed February 18, 2026,

https://www.researchgate.net/publication/386447067_An_Educational_RISC-V-Based_16-Bit_Processor
6. Analog Specs :: Quicker, easier and cheaper to make your own chip!, accessed February 18, 2026, https://tinytapeout.com/specs/analog/
7. Skywater 130nm Realistic Maximum Clock Std. Cells : r/chipdesign - Reddit, accessed February 18, 2026, https://www.reddit.com/r/chipdesign/comments/1anmu3q/skywater_130nm_realistic_maximum_clock_std_cells/
8. Crowd Sourced RISC-V (closed) - Tiny Tapeout, accessed February 18, 2026, https://tinytapeout.com/competitions/risc-v-peripheral/
9. cbiffle/hapenny: Small 32-bit RISC-V CPU with a half-width ... - GitHub, accessed February 18, 2026, https://github.com/cbiffle/hapenny
10. 975 ALU 74181 - Tiny Tapeout, accessed February 18, 2026, https://tinytapeout.com/runs/tt07/975
11. Wildcat: Educational RISC-V Microprocessors - arXiv, accessed February 18, 2026, https://arxiv.org/html/2502.20197v1
12. Booting Operating Systems on RISC-V Processors - Aaltodoc, accessed February 18, 2026, https://aaltodoc.aalto.fi/bitstreams/2596627b-8cd6-41ac-bf32-f0f36442f573/download
13. heavySea/skywater-130nm-adk - GitHub, accessed February 18, 2026, https://github.com/heavySea/skywater-130nm-adk
14. Modeling problems on sky130_fd_sc_hd liberty files · Issue #183 ..., accessed February 18, 2026, https://github.com/google/skywater-pdk/issues/183
15. lowRISC Verilog Coding Style Guide - GitHub, accessed February 18, 2026, https://github.com/lowRISC/style-guides/blob/master/VerilogCodingStyle.md
16. How to minimize route time in verilog code - Stack Overflow, accessed February 18, 2026, https://stackoverflow.com/questions/42741561/how-to-minimize-route-time-in-verilog-code
17. Design Compiler User Guide.pdf - KFUPM, accessed February 18, 2026, https://faculty.kfupm.edu.sa/COE/aimane/coe561/Design%20Compiler%20User%20Guide.pdf
18. Synopsys ® Synthesis Methodology Guide - PLDWorld.com, accessed February 18, 2026, http://www.pldworld.com/_actel/html/digital.library/q1_2003/pdfs/synopsys_MG.pdf
19. UVM environment for RISC-V processors - WebThesis - Politecnico di Torino, accessed February 18, 2026, https://webthesis.biblio.polito.it/18053/1/tesi.pdf
20. Design and verification of RISC-V CPU based on HLS and UVM - IEEE Xplore, accessed February 18, 2026, https://ieeexplore.ieee.org/iel7/9574103/9574447/09574575.pdf
21. UVM based design veri cation of a RISC-V CPU core - POLITesi, accessed February 18, 2026, https://www.politesi.polimi.it/retrieve/5b933d8b-7dc3-495e-826f-d2be551d0160/

2024_10_Occhineri.pdf
22. Comprehensive Formal Verification of Observational Correctness for the CHERIoT-Ibex Processor - ResearchGate, accessed February 18, 2026, https://www.researchgate.net/publication/388848527_Comprehensive_Formal_Verification_of_Observational_Correctness_for_the_CHERIoT-Ibex_Processor
23. YosysHQ/riscv-formal: RISC-V Formal Verification Framework - GitHub, accessed February 18, 2026, https://github.com/YosysHQ/riscv-formal
24. Comprehensive RTL-to-GDSII Workflow for Custom Embedded FPGA Architectures Using Open-Source Tools - MDPI, accessed February 18, 2026, https://www.mdpi.com/2079-9292/14/19/3866
25. Physically Aware Design of Generated Systems-on- Chip - EECS at Berkeley, accessed February 18, 2026, https://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-248.pdf
26. BLAKE2s Hashing Accelerator: A Solo Tapeout Journey · Tales on ..., accessed February 18, 2026, https://essenceia.github.io/projects/blake2s_hashing_accelerator_a_solo_tapeout_journey/