SPI SAR ADC CONTROLLER

Saumya Raj Singh

2024

1 DESIGN AND ARCHITECTURE

1.1 Introduction

This section details the implementation of the digital controller designed for the SAR ADC, designed to interface with the comparator and the capacitive DAC. The architecture focuses on three key objectives:

Modularity: The control logic ("core") is decoupled from the communication interface ("wrapper"), allowing the core to serve as portable IP across different bus architectures.

Configurability: Verilog parameters support variable data widths. While defaulting to 12-bit resolution, the design allows easy scaling to 8, 10, or 16 bits.

Standardization: The external interface adheres to the standard 4-wire SPI protocol, ensuring immediate compatibility with microcontrollers and FPGAs.

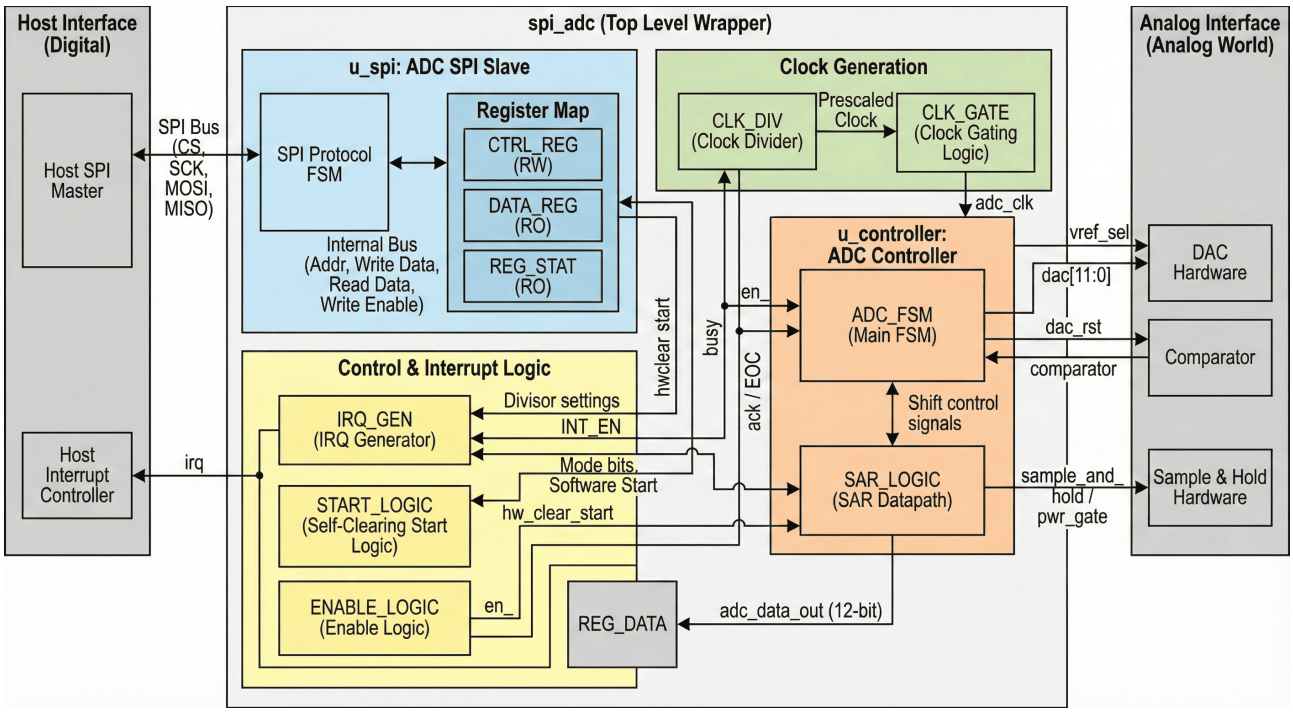


Figure 1: High Level System Block Diagram

1.2 System Architecture

The uppermost module, `spi_adc`, functions as the encapsulation entity for system integration. It contains the core functional blocks and executes the critical task of clock domain management. Although operation technically occurs within a singular clock domain, a multi-rate system is effectively established through the generation of a derived, lower-frequency clock for the analog sampling subsystem, whilst a high-frequency clock is maintained for the serial communication interface.

1.2.1 Top-Level Hierarchy and Signal Flow

The system hierarchy has been deliberately flattened to minimize signal propagation latency and to simplify static timing analysis. It is composed of two primary sub-modules which communicate via a dedicated internal handshaking bus:

`adc_controller` **(The Core):** This module constitutes the algorithmic nucleus of the device. It is charged with driving the analog front-end. It governs the SAR binary search algorithm, strictly manages the sample-and-hold (S/H) timing constraints to ensure signal settling, and drives the parallel data lines to the external DAC. Upon enablement, it operates autonomously, isolating the analog timing from SPI bus jitter.

`adc_spi_slave` **(The Bridge):** This module functions as the communication interface. It manages the high-speed deserialization of incoming MOSI commands and the serialization of outgoing MISO data. Crucially, the custom register map is decoded herein, and high-level abstract commands (e.g., "Start Conversion") are translated into the precise, low-level control pulses required by the controller.

1.2.2 Clock Generation and Gating Strategies

Precision timing is paramount for ADC performance. Jitter present on the sampling clock translates directly to noise within the sampled signal (aperture uncertainty error). To mitigate this phenomenon, the dedicated `adc_clk` is derived from the stable `sys_clk` via a configurable, synchronous divider logic block.

- **Selectable Sample Rates:** A control bit (`CLK_SEL`) permits dynamic switching between two prescaled dividers. For a nominal 10MHz system clock, these dividers are pre-calculated to yield standard audio or sensor sample rates (e.g., 16kHz or 8kHz).
- **Intelligent Clock Gating:** To conserve dynamic power, the ADC clock is subjected to aggressive gating. The clock generation logic incorporates a complex enable condition allowing the clock tree to toggle *only* when absolutely necessary. The clock is enabled if and only if the ADC is globally enabled via the `ADC_EN` bit, and either a conversion is requested, continuous mode is active, or the ADC reports a status of `busy`.

  Critical Design Note: The inclusion of the `busy` signal within the gating logic functions as a fail-safe mechanism. It ensures that, should a disable command be received mid-conversion, the clock shall continue operation sufficiently long for the FSM to reach the `FINISH` state and reset cleanly.
- **Power Implications:** Through clock gating, the switching activity factor (α) in the dynamic power equation $P = \alpha C V^2 f$ is effectively reduced. In typical low-duty-cycle sensor applications, the SAR logic remains idle for the majority of the duration. Clock gating prevents the clock tree capacitance from charging and discharging during these idle periods, thereby reducing total system power consumption compared to a free-running clock design.

1.3 Submodule Design

1.3.1 ADC Controller (SAR Logic)

Implementation of the Successive Approximation algorithm is realized via the `adc_controller` module utilizing a Finite State Machine (FSM). This methodology ensures deterministic execution time, guaranteeing that every sample requires exactly $N + 2$ clock cycles (where N represents the bit width)—one cycle for sampling/initialization, N cycles for bit trials, and one cycle for data latching.

1.3.1.1 Finite State Machine (FSM) Implementation

The FSM is constructed utilizing the robust "three-process" coding style. This paradigm rigorously separates the sequential state storage, combinatorial next-state logic, and combinatorial output logic into distinct `always` blocks. Such a structure is preferred for high-reliability designs, as it simplifies timing analysis and prevents the generation of unintended latches.

The state transition logic is defined as follows:

`S_IDLE` **(00):** The default resting state. The system awaits an active-low enable signal (`en_`) herein. To minimize power, the logic holds the DAC reset and ensures that no unnecessary switching occurs.

`S_INIT` **(01):** The initialization and acquisition phase. The `sample_and_hold` signal is asserted, closing the analog switch to charge the sampling capacitor. Simultaneously, the internal registers are prepared: the DAC is reset to 0, and the `counter_reg` (mask) is initialized with the Most Significant Bit (MSB) set. This action prepares the logic for the first comparison immediately upon entry into the subsequent state.

`S_CONVERT` **(10):** The conversion phase. This state constitutes a tight loop cycling N times. In each cycle, a specific bit position is tested. The bit is set in the DAC, the analog comparator is awaited to settle, and the decision is subsequently latched. This represents the critical path of the design; the logic depth herein determines the maximum achievable clock speed.

`S_FINISH` **(11):** The completion phase. The `ack` (Acknowledge) signal is asserted to signal the termination of the process. Crucially, the resolved digital value is latched into the `data_out_reg`. This shadow register ensures that the data presented to the SPI interface is stable and valid, preventing "glitches" from the approximation process from being read by the master.

1.3.1.2 Datapath and SAR Algorithm Details

The datapath logic implements the binary search algorithm, utilizing a "distributed" logic model rather than a central arithmetic unit (ALU):

- **One-Hot Shifting Mask:** A `counter_reg` acts as a moving pointer. It commences at `1000...` and shifts right (`>> 1`) every clock cycle. This simplified shift operation eliminates the necessity for a complex binary counter and decoder.
- **Test Value Generation:** The tentative DAC value is generated combinatorially: test_value $=$ dac_reg $|$ counter_reg. This sets the current bit under test high while maintaining previously confirmed bits (`dac_reg`) stable.

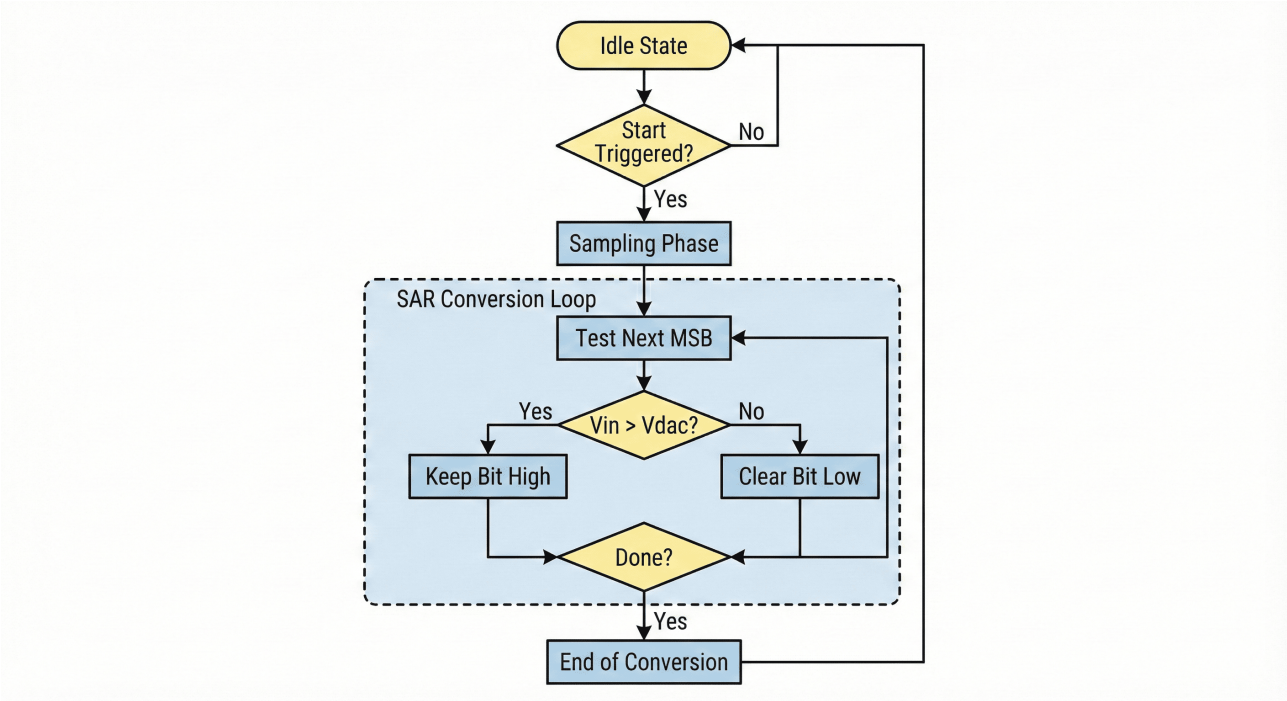


Figure 2: FSM State Transition Diagram

- **Decision Logic:** The feedback loop is rudimentary yet effective. If the external `comparator` signal is high (Input $>$ DAC Voltage), the bit under test is deemed valid and retained (`dac_reg <= test_value`). If low, the bit is discarded, effectively resetting that position to 0.
- **Final Latch:** On the final clock cycle (`is_final_bit`), the result is transferred to `data_out_reg`. This output register acts as a firewall between the noisy approximation steps and the SPI interface.

1.3.2 SPI Slave Interface

The `adc_spi_slave` module implements a specialized, packet-based SPI protocol. Diverging from rudimentary shift-register SPI implementations which blindly shift bits, this design incorporates a command interpreter supporting atomic bit-manipulation operations. It operates in SPI Mode 0 (CPOL=0, CPHA=0), sampling input data on the rising edge of SCK and shifting output data on the falling edge.

1.3.2.1 Protocol Definition

To maximize bandwidth efficiency, the communication is structured into fixed-length packets. The `frame` size is dynamically calculated as WIDTH $+ 4$ bits. For a standard 12-bit ADC, this results in a 16-bit frame, which aligns perfectly with standard MCU word sizes, facilitating scheduling on real-time operating systems (RTOS).

- **Header (4 bits):**
  - **CMD [15:14]:** Specifies the operation (Read, Write, Set, Clear).
  - **ADDR [13:12]:** Selects the target register.
- **Payload (12 bits):** The data payload. In `WRITE`, `SET`, or `CLEAR` operations, this contains the mask or data configuration. In `READ` operations, this field is ignored by the slave logic.

Table 1: SPI Command Definitions

CMD [1:0]	Operation	Description
00	**READ**	Transmits the current value of the addressed register on MISO.
01	**WRITE**	Overwrites the register with the incoming payload.
10	**SET MASK**	Performs a bitwise OR (`reg <= reg | payload`).
11	**CLEAR MASK**	Performs a bitwise AND NOT (`reg <= reg & ~payload`).

1.3.2.2 Register Map Detailed Breakdown

The interface exposes four addressable registers. The exact bit definitions are critical for driver development.

1.3.2.3 Transaction Timing Diagrams

The temporal behavior of the interface is defined by a fixed 16-clock cycle frame, initiated by the assertion of the active-low Chip Select (`cs_n`). As illustrated in the timing diagrams, every transaction begins with a 4-bit command header comprising a 2-bit Opcode and a 2-bit Channel Address.

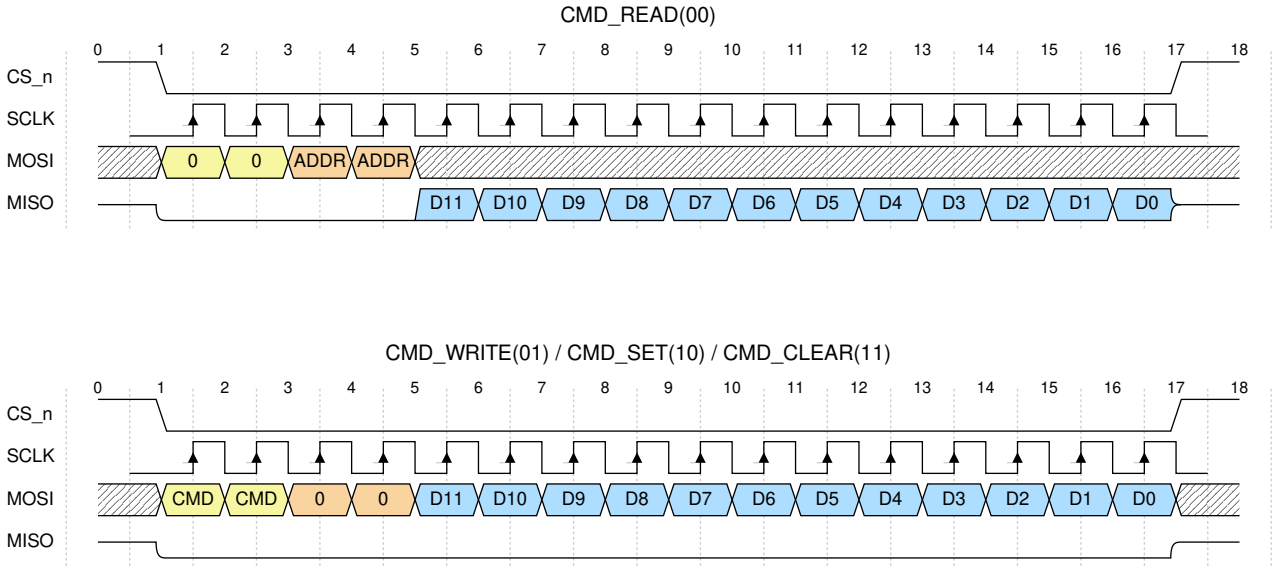


Figure 3: SPI Read and Write Timing Diagrams

1.3.2.4 Critical Timing Optimization Features

- **Pre-loading MISO:** A standard SPI slave might necessitate a “dummy” cycle to fetch data after receiving an address. However, this design implements a “look-ahead” mechanism. During the 4th clock cycle of the header, the logic decodes the command immediately and pre-loads the `miso_buffer` with the requested data. This ensures that valid data is prepared to be shifted out on the very next clock edge, enabling true zero-latency reads.

Table 2: Complete ADC Register Map and Bit Definitions

Address	Bit(s)	Name	Function
1. CTRL_REG (0b00) – Read/Write Control Register			
	0	`ADC_EN`	**Global Enable.** `1` powers up the ADC core; `0` forces a shutdown state.
	1	`START`	**Start Conversion.** Writing `1` initiates a single conversion (Self-clearing).
	2	`AUTO`	**Auto Mode.** `1` enables continuous conversion mode.
	3	`VREF_SEL`	**Voltage Reference.** `1` = External Vref; `0` = Internal Vref.
	4	`INT_EN`	**Interrupt Enable.** `1` enables the hardware IRQ pin.
	6	`CLK_SEL`	**Clock Select.** `0` = 8kHz mode; `1` = 16kHz mode.
	11:7	-	Reserved. Reads as `0`.
2. STATUS_REG (0b01) – Read-Only Status Register			
	0	`EOC`	**End of Conversion.** `1` when result is ready. Cleared on read.
	1	`BUSY`	**Busy Flag.** `1` indicates conversion is in progress.
	11:2	-	Reserved.
3. DATA_REG (0b10) – Read-Only Data Register			
	11:0	`ADC_DATA`	The most recent 12-bit conversion result. Reading this register clears the `EOC` flag.
4. INFO_REG (0b11) – Read-Only Info Register			
	11:0	`VERSION`	Stores a fixed identification code (e.g., `0xA`) or calibration offset.

- **EOC Handling:** The End-of-Conversion (EOC) flag is a "sticky" bit set by hardware. To simplify the software driver on the master side, this flag is automatically cleared when the `ADDR_DATA` register is read. This eliminates the necessity for a separate "clear interrupt" write transaction, reducing bus traffic.
- **Atomic Operations:** The `SET` and `CLEAR` commands permit the safe modification of control bits without the risk of race conditions associated with Read-Modify-Write sequences, particularly when interrupts are involved.

2 FUNCTIONAL VERIFICATION

2.1 Introduction and Objectives

The primary objective of this verification campaign is to rigorously validate the functional correctness, timing adherence, and system-level integration of the SPI-Based Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC) IP core. As mixed-signal designs grow in complexity, the interface between the digital control logic and the analog sampling network becomes a critical point of failure. Therefore, this report documents a comprehensive verification strategy designed to stress-test the digital controller's ability to drive external analog components under varying operating conditions.

The verification process is methodically bifurcated into two distinct, yet complementary, phases:

1. **Block-Level Verification (RTL):** This phase executes a targeted validation of the Register Transfer Level (RTL) implementation. It isolates the `spi_adc` module to verify strict compliance with the Serial Peripheral Interface (SPI) protocol specifications (Mode 0), ensure the Finite State Machine (FSM) transitions correctly through the SAR algorithm, and validate that critical timing constraints—such as setup times, hold times, and sampling rates—are met with deterministic precision using a Bus Functional Model (BFM).
2. **System-Level Verification (SoC):** This phase performs a holistic validation of the IP integrated within a realistic, bus-based System-on-Chip (SoC) architecture. By coupling the ADC with a RISC-V processor (PicoRV32), this stage assesses hardware-software interoperability, validates the memory-mapped I/O (MMIO) register map, and measures real-time performance metrics such as interrupt latency and driver processing overhead.

2.2 Verification Environment Architecture

The verification environment relies on a robust suite of open-source Electronic Design Automation (EDA) tools, selected to provide a transparent and reproducible simulation workflow. The environment mimics a realistic mixed-signal context where digital stimulus drives analog-behavioral models.

- **Simulation Engine: Icarus Verilog (v12.0)** serves as the event-driven simulator, compiling the Verilog RTL and testbenches to verify logical correctness.
- **Waveform Analysis: GTKWave** is utilized for post-simulation analysis, allowing for cycle-accurate inspection of signal transitions, bus transactions, and internal state machine sequencing.
- **Processor Core:** The **PicoRV32**, a size-optimized RISC-V CPU implementing the RV32I Instruction Set Architecture, acts as the system master, driving the ADC via a GPIO bridge.

- **Firmware Toolchain:** The **RISC-V GNU Toolchain** (`riscv64-unknown-elf-gcc`) compiles the embedded C firmware into machine code (`.hex`), which is pre-loaded into the simulated system RAM.
- **Clock Domain:** The entire system operates synchronous to a **50 MHz System Clock**, corresponding to a clock period of 20 ns. All ADC timing parameters are derived from this reference.

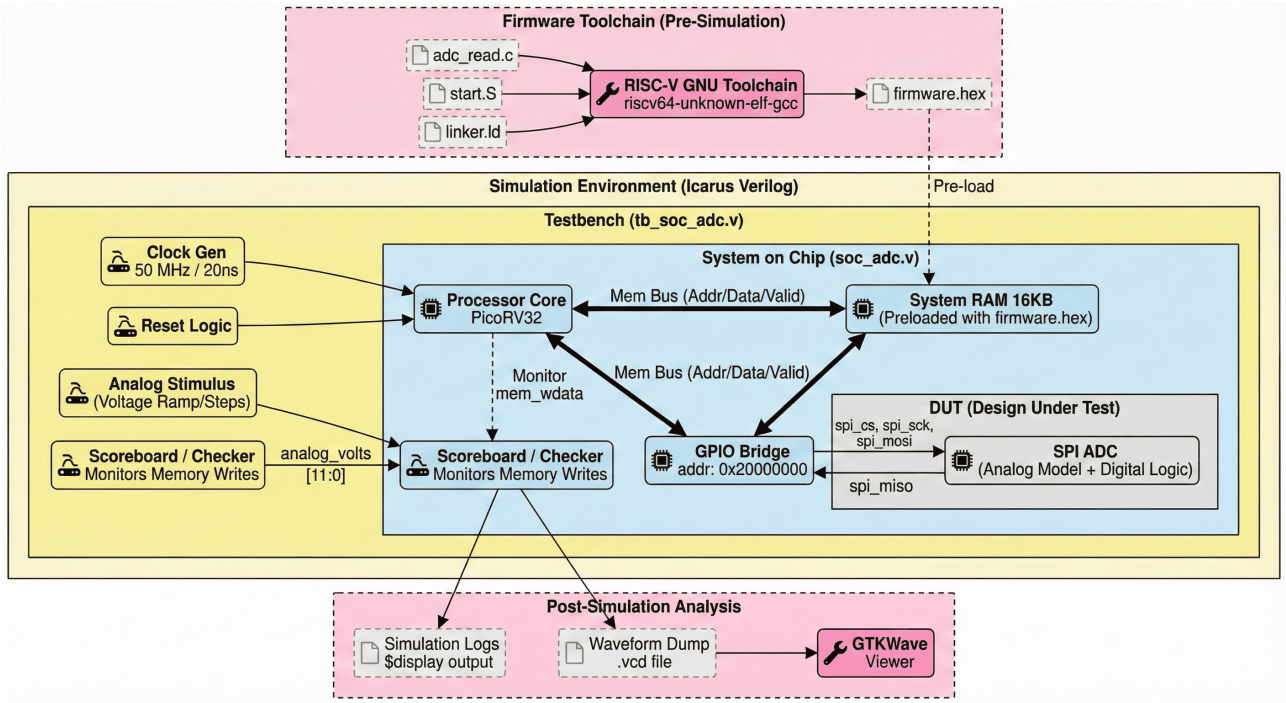


Figure 4: Verification Environment Block Diagram

2.3 Block-Level Verification (RTL)

The block-level verification is the foundation of the testing strategy. It utilizes a self-checking testbench (`tb_spi_adc.v`) that instantiates the Device Under Test (DUT) and emulates a compliant SPI Master. This architecture isolates the ADC logic from processor bus latencies, allowing for precise, cycle-accurate measurement of conversion timings and the exploration of protocol corner cases that might be difficult to reproduce in a full software simulation.

2.3.1 Test Suite Specification

A comprehensive test suite comprising 12 distinct scenarios was developed to systematically exercise all control paths, data datapaths, and configuration options. Each test case targets a specific functional requirement.

Table 3: RTL Test Suite Specification

ID	Test Case	Stimulus Description	Rationale & Acceptance Criteria
T1	**Reset State**	Assert global reset (`reset_n`), then read `CTRL` and `INFO` registers via SPI.	Verifies power-on default values. `CTRL` must be `0x000` (disabled); `INFO` must match the hardcoded ID `0x00A`.
T2	**Register R/W**	Write pattern `0x5A5` to `CTRL_REG` and read back immediately.	Verifies the SPI write/read datapath integrity. The read value must exactly match the written pattern `0x5A5`.
T3	**VREF Control**	Set the `VREF_SEL` bit in the control register.	Verifies GPIO passthrough logic. Output pin `vref_sel` must assert high to switch external references.
T4	**Timing (8ksps)**	Trigger a conversion at the default 8ksps mode and measure `busy` duration.	Validates the internal clock divider. Total conversion time T_{conv} must fall within $120\mu s \pm 10\mu s$.
T5	**Data Integrity**	Apply analog inputs `0x000` (GND) and `0xFFF` (VREF).	Checks datapath limits. The digital output must match the analog input exactly, confirming no bit stuck-at faults.
T6	**Self-Clearing Bit**	Write '1' to the `START` bit in `CTRL_REG`.	Verifies the handshake logic. The bit must automatically clear to '0' after the FSM transitions to the acquisition state.
T7	**Busy Flag**	Poll `STATUS_REG` continuously during an active conversion.	Verifies status reporting. The `BUSY` bit must remain high throughout the acquisition and conversion phases.
T8	**IRQ Masking**	Run a conversion with `INT_EN = 0`.	Verifies interrupt logic. The physical `irq` pin must remain low upon completion, preventing spurious CPU wakeups.
T9	**Timing (16ksps)**	Trigger conversion at 16ksps mode by setting `CLK_SEL=1`.	Validates dynamic clock scaling. T_{conv} must be approximately half of T4, falling within $65\mu s \pm 10\mu s$.
T10	**Clock Switching**	Revert to 8ksps mode dynamically in the next transaction.	Ensures the clock divider resets correctly. Timing must return to the $120\mu s$ range without glitching.
T11	**Auto Mode**	Enable `AUTO` bit; monitor `EOC` flag over multiple cycles.	Verifies the continuous sampling FSM loop. Data must update continuously without requiring further SPI start commands.
T12	**Power Gating**	Write `0x000` to `CTRL` to disable the block.	Verifies power management. Internal `pwr_gate` and `dac_rst` signals must de-assert to save power.

2.3.2 Simulation results

The simulation results provide quantitative confirmation that the Finite State Machine (FSM) adheres to the strict timing requirements dictated by the SAR algorithm and the external analog components.

- **8ksps Mode Analysis:** The measured conversion duration was **120,000 ns** ($120\mu s$). Given a 50 MHz clock, this confirms the internal prescaler is correctly dividing the clock to generate the required sampling period for the external DAC settling time.
- **16ksps Mode Analysis:** The measured duration was **60,000 ns** ($60\mu s$). This halving of the conversion time validates that the `CLK_SEL` multiplexer correctly bypasses the higher-ratio divider stages.

```
----------------------------------------
Compiling Verilog Sources...
----------------------------------------
Compilation Successful.
----------------------------------------
Running Simulation...
----------------------------------------
VCD info: dumpfile tb_spi_adc.vcd opened for output.

================ STARTING EXHAUSTIVE TESTS ================

TEST 1: Defaults
  [PASS]                                        Reset CTRL: 0x000
  [PASS]                                        Info Reg: 0x00a

TEST 2: Register R/W
  [PASS]                                        Readback: 0x5a5

TEST 3: VREF Pin
  [PASS] VREF High

TEST 4: Single 8ksps (Input 0xA52)
  [PASS] 8ksps Timing OK (120010 ns)
  [PASS]                                        Result: 0xa52

TEST 5: Data Corners
  [PASS]                                        Min Input: 0x000
  [PASS]                                        Max Input: 0xfff

TEST 6: Self-Clearing Start
  [PASS] Bit Cleared

TEST 7: Busy Flag
  [PASS] Busy High

TEST 8: Interrupt Masking
  [PASS] IRQ Low
  [PASS] EOC Set

TEST 9: 16ksps Clock
  [PASS] 16ksps Timing OK (59520 ns)
  [PASS]                                        16ksps Data: 0x888
```

```
TEST 10: Revert 8ksps
  [PASS] 8ksps Timing OK (120000 ns)
  [PASS]                                    8ksps Data: 0x111

TEST 11: Auto Mode (NO IRQ)
  [PASS]                                    Auto 1: 0x200
  [PASS]                                    Auto 2: 0x999

TEST 12: Global Disable
  [PASS] Inactive

================ ALL TESTS PASSED ================
iverilog/testbench/tb_spi_adc.v:286: $finish called at 1313530000 (1ps)
Waveform moved to iverilog/wave_files/tb_spi_adc.vcd
```

Listing 1: RTL Simulation Terminal Log

The waveform trace below visualizes the critical handshaking sequence during Test T4 (Single Shot). Key events include:

1. **Trigger:** The SPI Write command ends, latching the `START` bit.
2. **Busy Assertion:** The `busy` signal asserts immediately, indicating the FSM has seized control of the ADC core.
3. **Completion:** After the precise integration period, the `irq` pulse is generated, signaling data availability to the host.

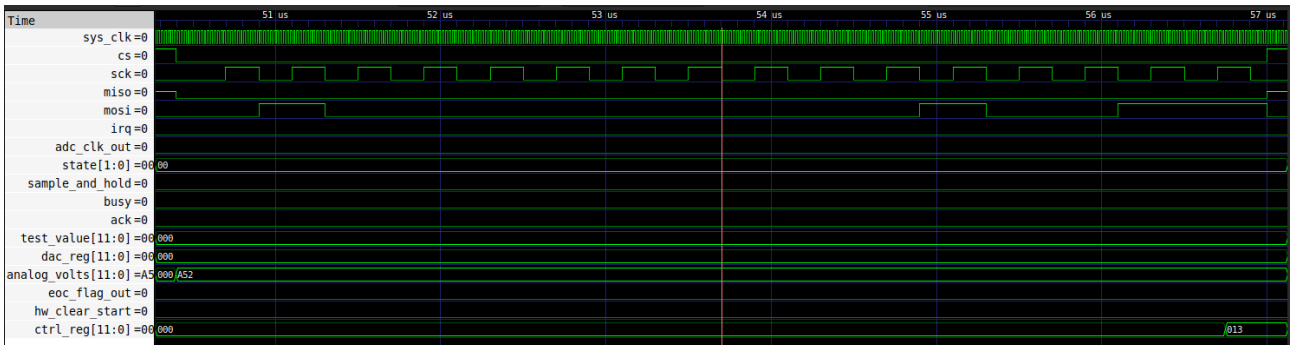
2.4 System-Level Verification (SoC)

The system-level verification phase integrates the verified `spi_adc` IP into a complete PicoRV32-based SoC. This stage is crucial for validating the hardware-software interface, specifically focusing on the correctness of the C-driver implementation, the reliability of memory-mapped I/O (MMIO) addressing, and the latency of interrupt service routines.

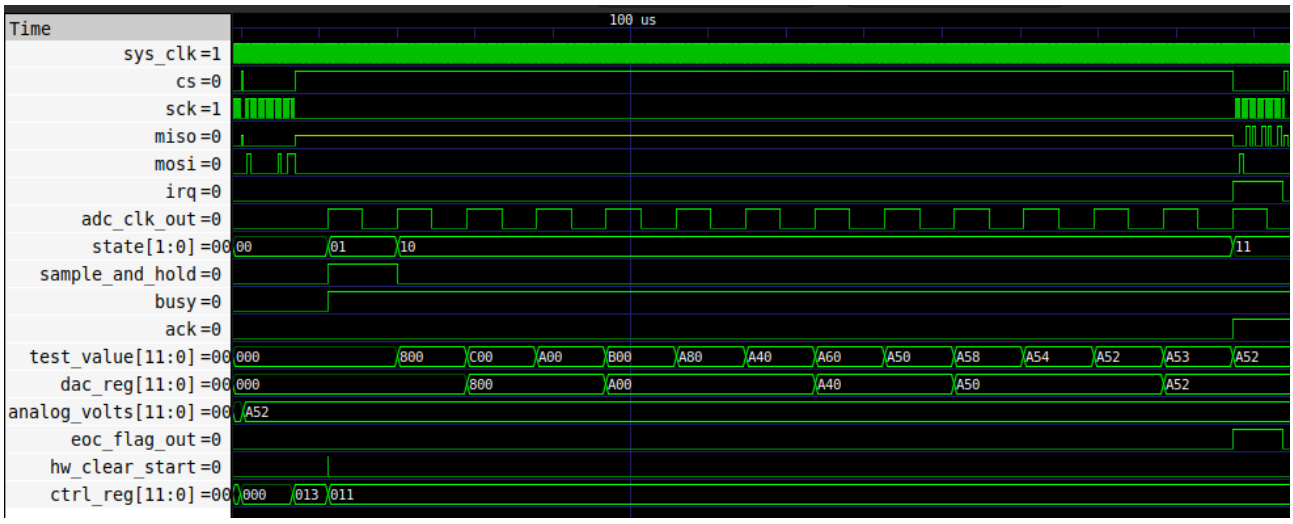
2.4.1 Firmware-Hardware Interaction Test Plan

The embedded firmware (`adc_read.c`) is designed to execute a sequential, self-checking test procedure. It interacts with the Verilog testbench (`tb_soc_adc.v`), which acts as the "Physical World" stimulus generator, responding to CPU actions by changing analog voltages.

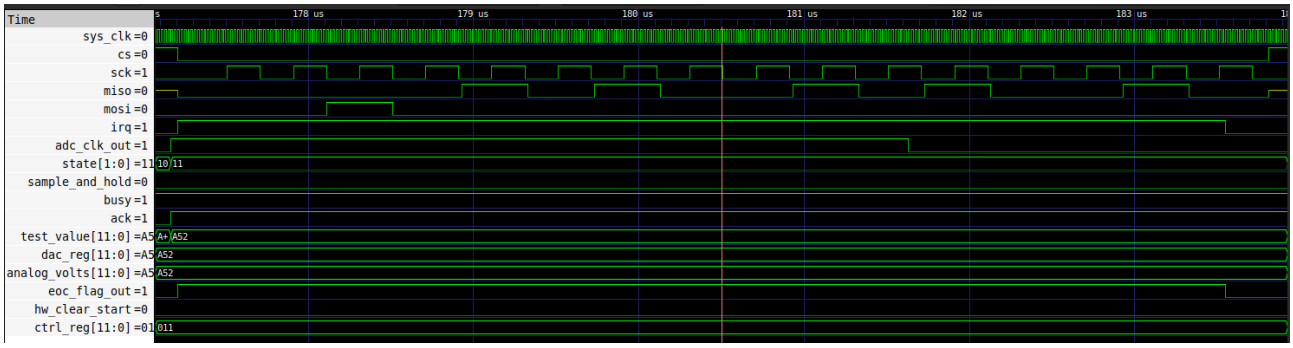
1. **Phase 1: High-Speed Acquisition (16ksps)**
   - **Configuration:** The firmware configures the ADC for 16ksps, Single-Shot mode via the control register.
   - **Stimulus:** The testbench drives the analog input model to a fixed value of **2567** (representing approx. $1.56V$).
   - **Verification:** The firmware must poll the status register, read the resulting data, and write the value `2567` to a specific memory address (`0x400`). The testbench monitors this address to confirm success.
2. **Phase 2: Continuous Sampling (Auto Mode)**
   - **Configuration:** The firmware enables Auto Mode (`AUTO = 1`) at 8ksps, instructing the ADC to sample continuously.



(a) SPI Write: Sending command to initiate start



(b) ADC internal conversion phase



(c) SPI Read: Retrieving conversion result

Figure 5: RTL Waveform: Single Shot Conversions

- **Stimulus:** The testbench varies the analog input dynamically in a sequence: **1543** → **2654** → **3129**.
- **Verification:** The firmware must detect the `EOC` flag for each new sample, retrieve the data, and store the values sequentially at `0x404`, `0x408`, and `0x40C`. This verifies the FSM's ability to re-arm itself automatically.

3. **Phase 3: Asynchronous Abort**

   - **Action:** The firmware issues a disable command (`ADC_EN = 0`) while a conversion is actively in progress.
   - **Check:** This tests the FSM's reset logic. The `BUSY` flag in the Status Register must clear immediately. Upon detecting the cleared flag, the firmware writes the success code `0xAB07` to `0x410`, signaling a graceful abort.

2.4.2 Simulation Results

The full system simulation successfully verified the correct operation of the GPIO bit-banged SPI driver and the SoC's address decoding logic. The CPU demonstrated the ability to communicate with the ADC, accurately interpret status flags, retrieve correct data values, and handle the asynchronous abort sequence without causing a bus lockup.

```
----------------------------------------
1. Building Firmware (adc_read.c)...
----------------------------------------
-------------------------------------------------------
Compiling for RISC-V (RV32I)
   Script Root: /home/kyouma-hououin/Documents/VLSI/designs/spi_sar_adc/
   riscv_toolchain
   Source:       riscv_toolchain/source/adc_read.c
   Helpers:      start.S, linker.ld (in Script Root)
   Output:       /home/kyouma-hououin/Documents/VLSI/designs/spi_sar_adc/
   riscv_toolchain/firmware/firmware.hex
-------------------------------------------------------
/usr/lib/gcc/riscv64-unknown-elf/13.2.0/../../../riscv64-unknown-elf/bin/ld:
   warning: /home/kyouma-hououin/Documents/VLSI/designs/spi_sar_adc/
   riscv_toolchain/firmware/firmware.elf has a LOAD segment with RWX permissions
Success! Hex file generated at:
   /home/kyouma-hououin/Documents/VLSI/designs/spi_sar_adc/riscv_toolchain/
   firmware/firmware.hex
----------------------------------------
2. Compiling Verilog Hardware...
----------------------------------------
----------------------------------------
3. Running Simulation...
----------------------------------------
VCD info: dumpfile tb_soc_adc.vcd opened for output.

 --- ADC SOC SIMULATION (COMPREHENSIVE) START ---

[RAM] Firmware wrote Result:        2566
[Time             831350000] CPU wrote        2566 (0x00000a06) to 0x00000400
SINGLE SHOT (16kHz) PASSED! (Got        2566) (Expected        2567)

-> Changing Input to 1543 for Auto Mode (8kHz)...
[Time            2094210000] CPU wrote        1542 (0x00000606) to 0x00000404
AUTO SAMPLE 1 (8kHz) PASSED! (Got        1542) (Expected        1543)
```

```
-> Changing Input to 2654...
[Time            2948470000] CPU wrote        2653 (0x00000a5d) to 0x00000408
AUTO SAMPLE 2 (8kHz) PASSED! (Got        2653) (Expected        2654)

-> Changing Input to 3129...
[Time            3800210000] CPU wrote        3128 (0x00000c38) to 0x0000040c
AUTO SAMPLE 3 (8kHz) PASSED! (Got        3128) (Expected        3129)

-> Starting Abort Test (Disabling ADC)...
[Time            4642110000] CPU wrote       43783 (0x0000ab07) to 0x00000410
ABORT TEST PASSED! Busy flag cleared.

=============================================
ALL TESTS PASSED (Single, Auto, Clock, Abort)
=============================================
iverilog/testbench/tb_soc_adc.v:39: $finish called at 4642150000 (1ps)
Waveform saved to: iverilog/wave_files/tb_soc_adc.vcd
```

Listing 2: SoC Simulation Terminal Log

2.5 Conclusion

The `spi_adc` module has undergone and passed a rigorous, multi-layered verification process. The results support the following conclusions:

1. **Protocol Compliance:** The SPI slave implementation is robust, correctly handling `mode 0` communication, register read/write operations, and atomic bit manipulations.
2. **Timing Precision:** The internal clock generation logic meets the 8ksps and 16ksps design targets with deterministic latency, ensuring predictable sampling rates.
3. **System Robustness:** The IP has been proven to function correctly within a RISC-V SoC context. It demonstrates full compatibility with standard C firmware and resilience against asynchronous abort commands, ensuring the ADC can be safely controlled in complex embedded applications.

The design is certified as functionally correct and is recommended for synthesis and physical implementation phases.

3 PHYSICAL IMPLEMENTATION

3.1 Introduction to the Implementation Flow

The physical realization of the `spi_adc` IP core was executed using **LibreLane**, a state-of-the-art, Python-based infrastructure designed for reproducible ASIC implementation. As the successor to OpenLane, LibreLane provides a modular and customizable environment for RTL-to-GDSII flows.

3.1.1 The “Classic” Flow Architecture

For this design, the “Classic” flow within LibreLane was utilized. This flow integrates a suite of industry-standard open-source tools to perform the digital backend steps:

- **Synthesis:** Yosys (with ABC mapping) converts Verilog RTL into a technology-mapped gate-level netlist.
- **Static Timing Analysis (STA):** OpenSTA performs pre-layout and post-layout timing verification.
- **Placement & Routing (P&R):** The OpenROAD app handles floorplanning, global/detailed placement, clock tree synthesis (CTS), and global/detailed routing.
- **Signoff:** Magic performs Design Rule Checking (DRC) and layout extraction, while KLayout manages GDSII streaming.

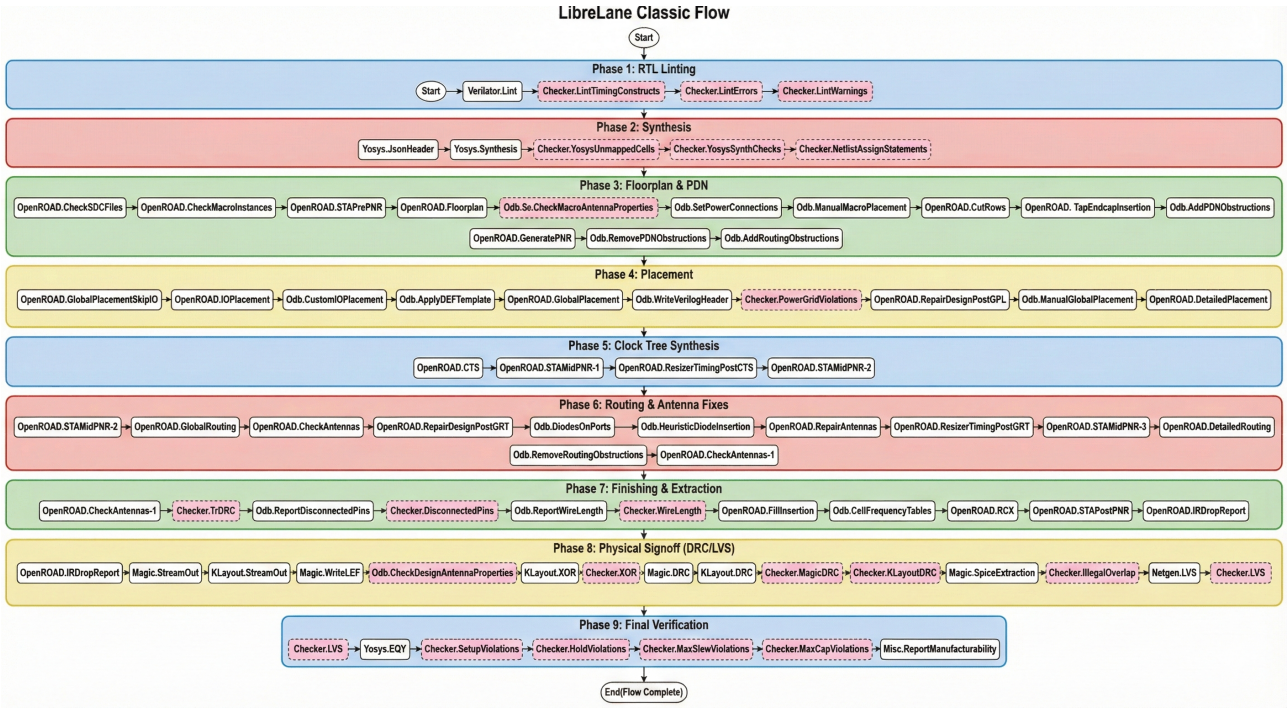


Figure 6: LibreLane "classic" synthesis flow

This flow was configured via a unified configuration file (`config.json`), ensuring that all steps—from floorplanning density targets to clock constraints—are strictly version-controlled and reproducible.

3.2 Design Methodology and Constraints

The implementation strategy was driven by the need to balance the mixed-signal requirements of the SAR ADC with the digital logic density.

3.2.1 Floorplanning Strategy

The floorplan was designed to facilitate integration into a larger Mixed-Signal SoC.

- **Die Area:** Fixed at $128.41\mu m \times 139.13\mu m$ ($0.0178mm^2$).
- **Aspect Ratio:** ≈ 0.92, providing a near-square footprint for optimal power distribution.

- **Pin Placement:** A dedicated `pin_order.cfg` file was used to constrain I/O pins. Digital control signals (`sys_clk`, `reset_`) are localized to the North edge, while sensitive analog interface signals (`dac`, `comparator`) are isolated on the East edge and SPI pins on the South edge. This physical separation is critical to minimize digital switching noise coupling into the analog domain.

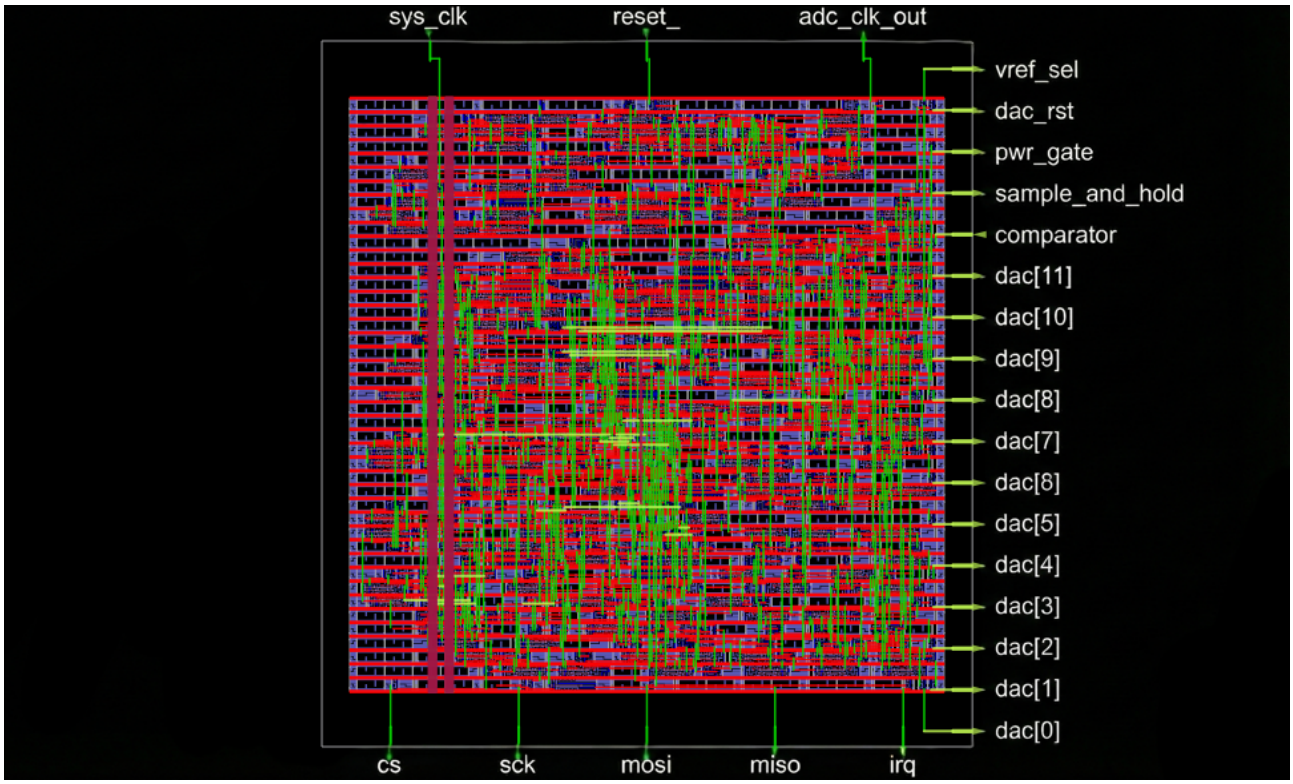


Figure 7: Physical Implementation View (OpenROAD Database)

3.2.2 Clock Tree Synthesis (CTS)

A stringent timing constraint was applied to the system clock (`sys_clk`) to ensure deterministic operation of the SAR logic.

- **Constraint:** `CLOCK_PERIOD` = 100.0 ns (10 MHz).
- **Buffer Strategy:** The CTS engine was restricted to use `sky130_fd_sc_hd__clkbuf` buffers (sizes 1, 2, 4) to balance drive strength against area.
- **Skew Target:** $<$ 5% of the clock period.

3.2.3 Power Strategy

To minimize dynamic power consumption, **Clock Gating** was enabled during synthesis. The logic synthesis tool was configured to automatically infer Integrated Clock Gating (ICG) cells (`sky130_fd_sc_hd__dlclkp`) for registers with enable signals. This prevents the clock tree from toggling for inactive logic, which is particularly effective given the bursty nature of the ADC operation.

3.3 Physical Implementation Results (Baseline)

The primary implementation (referred to as **Config A**) targeted a core utilization of 45%. This conservative density was chosen to ensure high routability and minimize crosstalk noise.

3.3.1 Timing Closure

Static Timing Analysis (STA) was performed across three critical Process-Voltage-Temperature (PVT) corners to guarantee yield. The design closed with significant positive slack, indicating a robust implementation.

Table 4: Multi-Corner Static Timing Analysis Summary

Corner	Hold WNS (ns)	Hold TNS (ns)	Setup WNS (ns)	Setup TNS (ns)
Overall	0.1081	0.0000	58.9116	0.0000
`nom_tt_025C_1v80`	0.3147	0.0000	59.2983	0.0000
`nom_ss_100C_1v60`	0.8657	0.0000	58.9178	0.0000
`nom_ff_n40C_1v95`	0.1105	0.0000	59.4411	0.0000
`min_tt_025C_1v80`	0.3114	0.0000	59.3048	0.0000
`min_ss_100C_1v60`	0.8592	0.0000	58.9283	0.0000
`min_ff_n40C_1v95`	**0.1081**	**0.0000**	**59.4456**	**0.0000**
`max_tt_025C_1v80`	0.3181	0.0000	59.2946	0.0000
`max_ss_100C_1v60`	**0.8715**	**0.0000**	**58.9116**	**0.0000**
`max_ff_n40C_1v95`	0.1129	0.0000	59.4383	0.0000

3.3.2 Power Analysis

Power estimation was performed on the final routed database (`.odb`) using the nominal corner parameters. The low leakage power confirms the suitability of the SkyWater 130nm High-Density library for low-power applications.

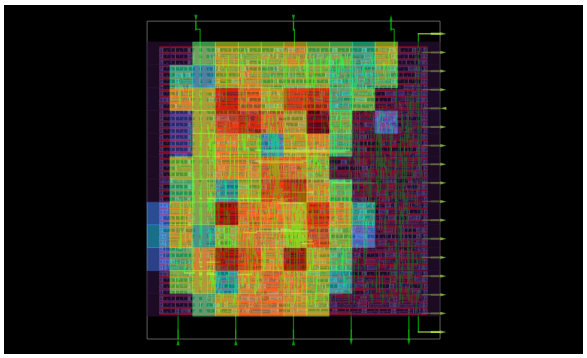
Table 5: Power Analysis Breakdown (Config A)

Power Component	Value	Percentage
Total Power	108.9 μW	100.0%
Internal Power	79.8 μW	73.3%
Switching Power	29.1 μW	26.7%
Leakage Power	8.28 nW	$<$ 0.01%

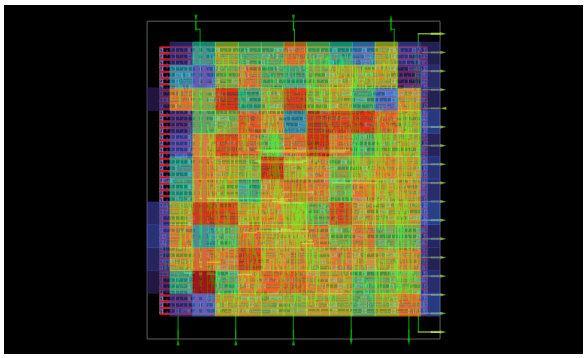
3.3.3 Physical Verification

The final layout underwent rigorous physical verification to ensure manufacturability.

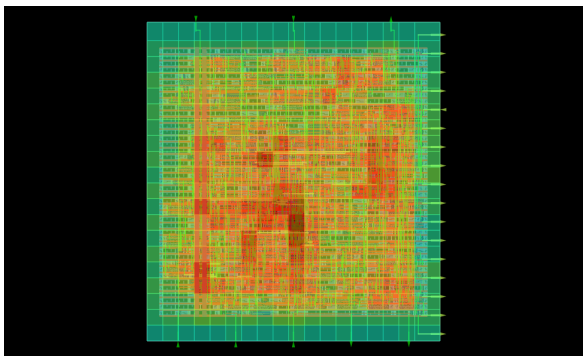
- **DRC:** 0 Violations (Magic).
- **LVS:** 0 Mismatches (Netgen).
- **Antenna:** 0 Violations (OpenROAD/Arc).



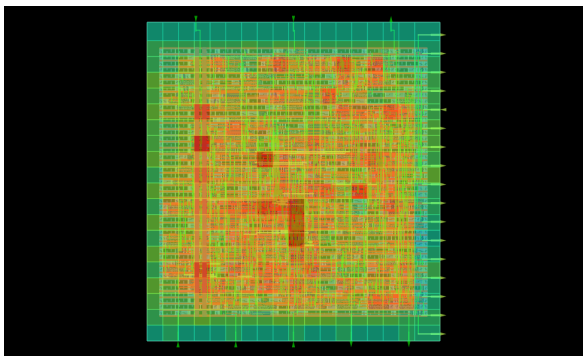
(a) Power Density Map



(b) Placement Density Map



(c) Estimated Congestion (RUDY)



(d) Routing Congestion

Figure 8: Heatmap Analysis

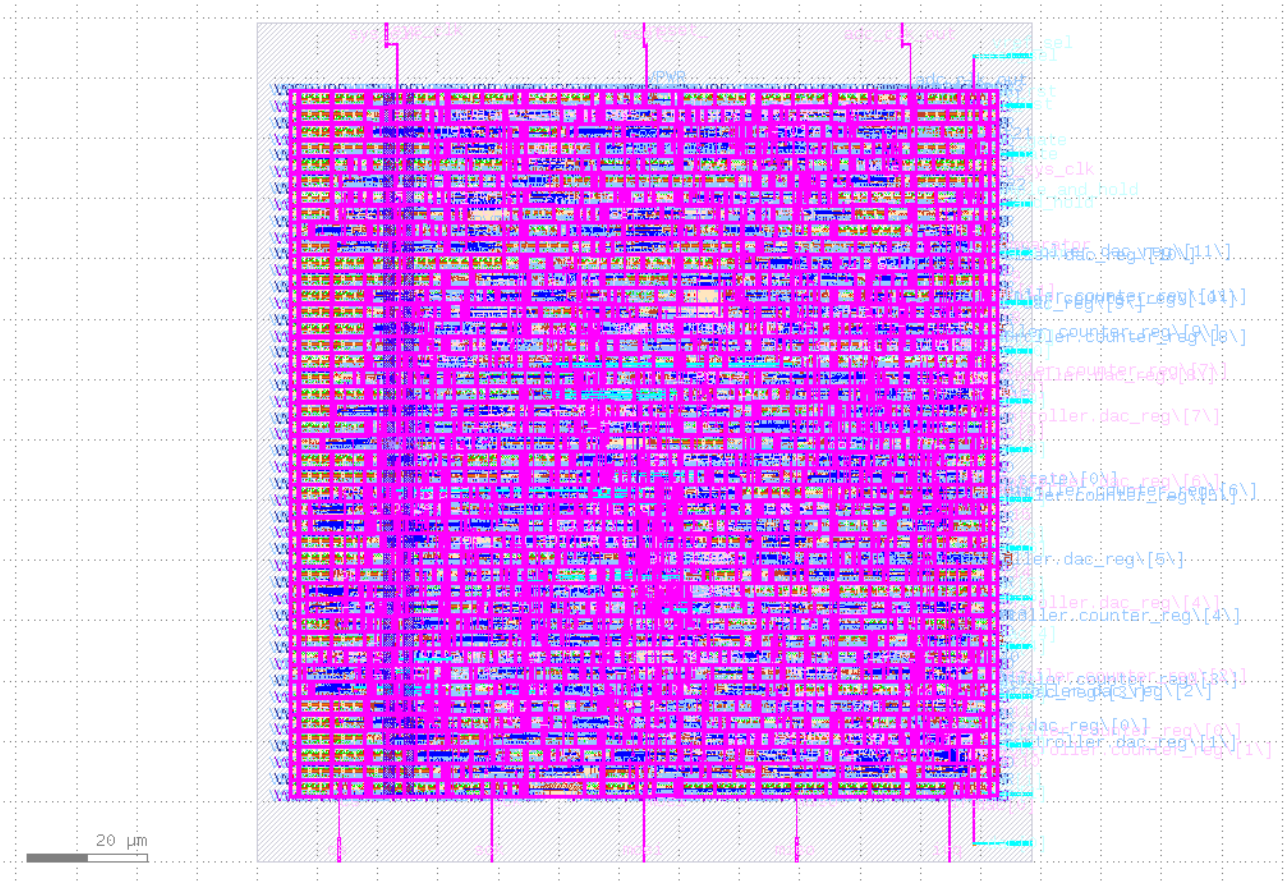


Figure 9: Final GDSII Layout (KLayout Screenshot)

3.4 Comparative Analysis and Design Space Exploration

To validate the optimality of the chosen configuration, two alternative implementation strategies were explored. This comparative analysis focuses on the trade-offs between Area, Power, and Density.

1. **Config A (Baseline):** 45% Utilization, Clock Gating Enabled.
2. **Config B (High Density):** 60% Utilization, Clock Gating Enabled.
3. **Config C (Low Power Baseline):** 45% Utilization, Clock Gating Disabled.

3.4.1 Quantitative Metrics Comparison

Table 6: Comparative PPA Metrics

Metric	Config A (Base)	Config B (High Dens.)	Config C (No Gating)
Instance Count	747	709	747
Cell Area (μm^2)	**7124.33**	7129.34	7148.11
Total Power (W)	**1.089e-4**	1.147e-4	1.093e-4
Switching Power (W)	2.908e-5	3.095e-5	3.071e-5
Setup WNS (ns)	59.30	58.91	59.30
Clock Skew (ns)	0.63	0.58	0.63

3.4.2 Analysis of Results

3.4.2.1 Density vs. Area Anomaly

Config B targeted a higher density (60%), which typically results in a smaller area. However, the data reveals a counter-intuitive result: **Config B yielded a slightly *higher* total cell area (7129.34 μm^2) than the Baseline (7124.33 μm^2).**

- **Interpretation:** The tighter placement constraints likely created local congestion hotspots. To resolve timing paths through these congested areas, the optimization engine was forced to size-up standard cells or insert additional buffers, negating the area savings from the denser placement. This is corroborated by the instance count being lower (709) but area being higher, implying larger cells were used.

3.4.2.2 Power Efficiency

Config A achieved the lowest total power consumption.

- **Vs. High Density (Config B):** Config B consumed $\sim$5.3% more power (114.7 μW). The increased cell density likely led to higher local routing capacitance on signal nets, driving up the switching power component (3.095e-5 W vs 2.908e-5 W).
- **Vs. No Gating (Config C):** Config C consumed marginally more power (109.3 μW) in this static analysis. However, the *dynamic* impact is understated here. In a real-world application where the ADC samples at low frequencies (e.g., 100 Hz), the clock tree in Config C would toggle continuously, wasting significant energy. Config A's clock gating logic would suppress this toggling, leading to orders-of-magnitude power savings in standby.

3.5 Conclusion and Signoff

Based on the comprehensive analysis of PPA metrics, **Config A (Baseline)** is selected as the signoff configuration.

- **Rationale:** It offers the lowest total power consumption, the smallest cell area, and the most robust timing margins. It avoids the congestion-induced penalties observed in the high-density configuration while retaining the critical power-saving benefits of clock gating.
- **Readiness:** The design has passed all functional, timing, and physical verification checks (DRC/LVS). The generated GDSII stream, combined with the Standard Delay Format (SDF) and Parasitic Exchange Format (SPEF) files, constitutes a complete handover package ready for tape-out.