# END-TO-END DESIGN OF A MIXED-SIGNAL SOC FOR DIGITAL VOICE AND BROADCAST SIGNAL GENERATION

Sowmya K B, Neha J C, Preeti Yadav, Paridhi Sudarshan, and Shreya V Sanoj

Department of Electronics and Communication Engineering, RV College of Engineering, Bengaluru, Karnataka, India

***ABSTRACT***

*This work presents the design, modelling, and synthesis of SoC, a compact RISCV based SoC integrating three IP cores: RVMYTH processor, an 8x Phase-Locked Loop for clock generation, and a 10-bit Digital-to-Analog Converter for analog interfacing. The SoC was developed using open-source tools and Sky130 technology, demonstrating feasibility for educational and practical applications in embedded systems, IoT devices, and digital signal processing. The design flow encompasses RTL modelling, synthesis using Yosys, and timing verification with OpenSTA. The RVMYTH core, converted from TL-Verilog, was integrated with analog IP models and validated through pre-synthesis and post-synthesis simulations using iverilog and GTKwave. The synthesized netlist comprises 5,552 cells occupying 58,173.29 um². Static timing analysis confirmed all constraints were met, achieving 0.86 ns positive slack with a maximum path delay of 10.01 ns. The application-oriented design makes the SoC suitable for IoT sensor nodes, industrial control systems, audio processing, and motor control applications. The integrated DAC enables direct sensor/actuator interfacing, while the RISC-V architecture supports custom instruction extensions for domain-specific applications. The PLL ensures clock stability critical for serial communication interfaces and real-time control.*

***KEYWORDS***

*RISC-V processor, Phase-Locked Loop (PLL), Digital-to-Analog Converter (DAC),OpenLANE, Sky130 PDK*

## 1. INTRODUCTION

The rapid evolution of embedded systems and Internet of Things (IoT) applications has created increasing demand for compact, energy-efficient System-on-Chip (SoC) solutions that integrate digital processing with analog interfacing capabilities. Modern embedded applications spanning industrial automation, consumer electronics, automotive systems, and smart sensors require processors capable of executing complex algorithms while seamlessly communicating with the analog world through sensors and actuators. Traditional approaches to SoC development, however, have been constrained by high costs associated with proprietary Electronic Design Automation (EDA) tools and Process Design Kits (PDKs), creating significant barriers to entry for educational institutions, researchers, and small-scale developers.

The emergence of open-source hardware initiatives, particularly the RISC-V instruction set architecture has fundamentally transformed the landscape of chip design. Security focused RISC-V implementations have demonstrated successful tapeouts for various applications [1]. RISC-V,

David C. Wyld et al. (Eds): ACSTY, MLSC, SVC, AIBD, ITCSS, ADCOM, NATP, SOFE – 2026
pp. 117-132, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160410

as a free and open instruction set architecture [2], [3], provides flexibility for custom extensions and eliminates licensing costs, making it ideal for domain specific applications and educational purposes. The Rocket chip generator [4] has served as a foundational reference for many RISC-V implementations. Recent work has demonstrated successful 5-stage pipelined RISC-V processor implementations on FPGA platforms [5], [6], while other researchers have explored SoC designs for intelligent recognition systems [7] and low-power IoT applications [8]. Open-source ASIC implementations of RISC-V based SoCs have also been reported [9].

Complementing these processor developments, the release of comprehensive opensource EDA toolchains has democratized chip design. OpenLANE [10], built upon the OpenROAD project [11], provides a complete RTL-to-GDSII flow that has been proven through successful tapeouts [12]. The toolchain integrates various open-source tools including Magic [13] for layout visualization, Qflow [14] for digital synthesis, and the broader OpenROAD infrastructure [15], [16]. The availability of the SkyWater 130nm open-source PDK [17], [18] through collaboration between Google and SkyWater Technology marked a watershed moment, providing the first production-capable, fully open-source process technology [19]. Additional tools like OpenRAM [20] for memory compilation complete the open-source ecosystem.

Despite these advances, there remains a critical need for well-documented, accessible reference designs that demonstrate the complete integration of digital processors with analog components using entirely open-source tools. Most existing educational SoC projects focus solely on digital designs or rely on proprietary tools for critical design stages, limiting their accessibility and reproducibility. Furthermore, mixed-signal designs that incorporate analog blocks like Phase-Locked Loops (PLLs) and Digital-to-Analog Converters (DACs) present additional complexity in modeling, simulation, and physical design that are rarely addressed in open-source contexts.

This work presents a compact RISC-V based mixed-signal SoC that addresses these gaps by integrating three key intellectual property (IP) cores: RVMYTH, a RISC-V processor core developed through educational initiatives; an 8× PLL for stable clock generation; and a 10-bit DAC for analog output. The primary objectives of this project are threefold: first, to demonstrate the feasibility of creating a functional mixed-signal SoC using entirely open-source tools and student-developed IP cores; second, to establish a complete, documented design flow from Register Transfer Level (RTL) modeling through synthesis and timing verification; and third, to create an application-oriented platform suitable for practical embedded system applications including sensor interfacing, signal processing, and control systems.

The design methodology encompasses comprehensive pre-synthesis modeling and simulation, synthesis within the OpenLANE flow targeting the Sky130 standard cell library, post-synthesis gate-level simulation to verify functional equivalence, and static timing analysis to validate timing closure. The integration of analog IP models within the digital design flow presents unique challenges in terms of abstraction levels, timing constraints specification, and verification methodology, all of which are addressed in this work.

The application-oriented nature of this SoC extends beyond academic exercise. The integrated 10-bit DAC enables direct interfacing with analog sensors, actuators, and audio systems, making the design applicable to industrial control systems, data acquisition systems, and audio processing applications. The RISC-V architecture's extensibility allows for custom instruction set extensions tailored to specific application domains such as digital signal processing, cryptography, or motor control algorithms. The on-chip PLL provides the clock stability essential for timing-critical applications including serial communication protocols (UART, SPI, I2C), pulse-width modulation (PWM) generation for motor control, and synchronous data acquisition systems. Operating at approximately 91 MHz after timing closure, the design offers sufficient

computational performance for a wide range of embedded applications while maintaining the simplicity necessary for educational use.

The contributions of this work include: (1) a complete open-source mixed-signal SoC design integrating digital and analog IP cores, (2) a documented RTL-to-synthesis flow using entirely open-source tools, (3) successful timing closure verification with positive slack, demonstrating design robustness.

### 1.1. RELATED WORK

The growth of open-source SoC design has been strongly driven by the introduction of the RISC-V instruction set architecture. RISC-V offers a free and extensible alternative to proprietary ISAs, making it well suited for academic and research use. Early silicon tapeouts demonstrated that RISC-V processors can be successfully fabricated for realworld applications [1]. The availability of stable user-level and privileged architecture specifications further enabled consistent processor development [2], [3]. Tools such as the Rocket Chip generator provided a reusable framework that influenced many later RISCV–based designs [4]. Several studies have implemented pipelined RISC-V processors on FPGA platforms [5], [6], and extended these designs to SoCs for intelligent systems [7], IoT applications [8], and fully open-source ASIC implementations [9].

Alongside processor development, open-source EDA tools have played a key role in making chip design more accessible. The OpenLANE flow, built on the OpenROAD project, provides a complete RTL-to-GDSII design flow using only open-source tools and has been validated through successful tapeouts [10] – [12]. Supporting tools for synthesis, placement, routing, and layout visualization form a reliable digital design ecosystem [13] – [16]. A major breakthrough was the release of the SkyWater 130 nm open-source PDK, which enabled fabrication-ready designs without relying on proprietary process data [17] – [19]. Tools such as OpenRAM further support memory integration in open-source SoC designs [20].

Open-source synthesis and verification tools such as Yosys and ABC have made RTLto-gate synthesis practical for student and research projects [21] – [25]. Functional verification is commonly performed using Icarus Verilog and GTKWave, while static timing analysis techniques ensure correct operation across different operating conditions [26] – [32].

Despite these developments, most existing open-source SoC efforts focus mainly on digital designs, with limited discussion on mixed-signal integration. Designs that combine processors with analog IP blocks using entirely open-source tools are still relatively rare. This work addresses this gap by presenting a fully open-source mixed-signal SoC, along with a clear and reproducible design and verification flow suitable for educational and practical use.

## 2. SYSTEM ARCHITECTURE

The SoC represents a compact yet functionally complete mixed-signalSoC (SoC) designed to demonstrate the integration of open-source IP cores for educational and research applications. The architecture comprises three primary components: RVMYTH, a RISC-V based processor core; AVSDPLL, an 8x phase-locked loop for clock generation; and AVSDDAC, a 10-bit digital-to-analog converter for interfacing with analog devices. This integration enables the SoC to function as a standalone embedded system capable of executing computational tasks while communicating with external analog components. The design philosophy emphasizes modularity, open-source accessibility, and fabrication readiness under the SkyWater 130nm process technology. By combining these three distinct IP cores, the SoC demonstrates the feasibility of

creating educational SoC platforms that can be fabricated and tested in real silicon, bridging the gap between academic instruction and practical chip design experience.

## 2.1. RVMYTH Processor Core

RVMYTH is a RISC-V based microprocessor implementing the RV32I base integer instruction set architecture. Originally developed through the VSD educational workshop series using Transaction-Level Verilog (TL-Verilog), the core demonstrates the fundamental principles of modern processor design while maintaining implementation simplicity suitable for educational purposes. The processor architecture follows a classic five-stage pipeline structure comprising instruction fetch, decode, execute, memory access, and writeback stages. The design includes a 32-entry register file with register x17 designated as the primary output interface to the DAC subsystem. Instruction memory is integrated within the core, preloaded with a test program that generates incrementing or predetermined patterns for DAC output verification. For integration into the SoC framework, the TL-Verilog source is compiled to standard Verilog using the SandpiperSaaS compiler. This compilation process generates synthesizable RTL code compatible with standard EDA toolchains while preserving the original design intent. The processor operates synchronously with the clock signal provided by the PLL, executing instructions at a rate determined by the timing constraints and achieving a maximum operating frequency limited by the critical path through the execute and memory stages.

## 2.2. AVSDPLL Clock Generator

The AVSDPLL (Analog and Very-Small Digital Phase Locked Loop) serves as the primary clock generation and distribution network for the SoC (Fig 1.). This analog IP core implements an 8x frequency multiplier, converting an external reference frequency into a stable, higher-frequency clock suitable for processor operation. Phase-locked loops are critical components in modern SoC designs, providing precise frequency synthesis, clock multiplication, and jitter reduction. The feedback loop maintains phase and frequency lock by continuously comparing the VCO output with the reference input and adjusting the control voltage accordingly. For the mixed-signal implementation flow, the PLL is instantiated as a hard macro with pre-characterized timing and power models.

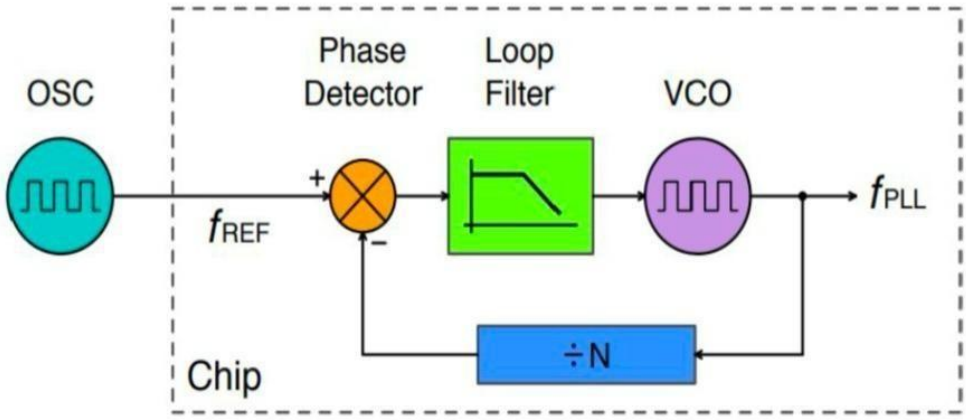


**Fig.1.** PLL Block Diagram

## 2.3. AVSDDAC Digital-To-Analog Converter (DAC)

The AVSDDAC implements a 10-bit current-steering digital-to-analog converter, enabling the SoC to interface with external analog devices and systems. This IP core converts the 10-bit digital output from the RVMYTH processor (sourced from register x17) into a corresponding analog voltage or current signal. Current-steering DAC architectures are favored in modern mixed-signal designs for their speed, power efficiency, and monotonicity characteristics. The 10-bit resolution provides 1024 discrete output levels, suitable for applications such as audio signal generation, sensor interfacing, or control signal output. The DAC operates synchronously with the system clock, updating its output on each clock edge when new data is presented from the processor. Similar to the PLL, the DAC is integrated as an analog hard macro during physical design. The digital interface consists of the 10-bit data bus and control signals, while the analog output requires careful consideration during floorplanning to minimize noise coupling and ensure signal integrity. For pre-synthesis and behavioral simulation, the DAC is modeled using Verilog real datatypes to approximate analog behavior within the digital simulation environment. While this approach has limitations actual analog transients, settling time, and non-idealities are not captured it enables functional verification of the digitalto-analog data flow and system-level behavior before physical implementation.

## 2.4. SYSTEM INTEGRATION AND INTERCONNECTS

The SoC wrapper module encapsulates all three IP cores and defines the top-level interfaces. Signal interconnections are straightforward: the PLL output connects to the RVMYTH clock input and potentially to the DAC clock (depending on implementation); the processor's register x17 output bus connects directly to the DAC's 10-bit input; and external reset and reference clock signals provide system initialization and timing reference. The integration methodology emphasizes clean interfaces and proper signal synchronization. All inter-block signals are registered to prevent timing violations and minimize metastability risks. The system level timing constraints account for clock distribution delays, PLL jitter characteristics, and setup/hold requirements at the DAC input. Interconnect topology is designed with physical implementation in mind. The physical design flow handles analog macros as pre-placed, pre-routed cells with keepout regions to protect sensitive analog circuitry from digital routing congestion.

## 2.5. APPLICATION DOMAINS

While the SoC is primarily designed as an educational platform, its architecture supports several practical application scenarios: 1. Digital Signal Processing and Audio: The processor can execute simple DSP algorithms (waveform generation, filtering, modulation) with results output through the DAC. This enables applications such as function generators, audio tone synthesis, or simple music playback with preloaded samples in instruction memory. 2. Sensor Interface and Control Systems: The 10-bit DAC output can drive actuators, control voltage-controlled components, or provide reference signals for sensor systems. The processor can implement feedback control algorithms, making the SoC suitable for embedded control applications such as temperature regulation, motor speed control, or process automation. 3.Educational and Research Platform: The open-source nature and complete documentation make the SoC ideal for teaching digital design, mixed-signal integration, and SoC architecture. Students can modify the processor ISA extensions, experiment with different DAC output patterns, or integrate additional peripherals to explore system-level design concepts. 4. IoT and Edge Computing: With appropriate firmware, the SoC can serve as a minimal IoT node processor, executing sensor data collection and preprocessing algorithms before transmitting results to higher-level systems. The DAC could drive simple output indicators or control low-power actuators in resource-constrained applications. 5. Rapid Prototyping: The modular architecture allows designers to swap IP cores or extend functionality. For instance, replacing RVMYTH with a more capable processor, adding

additional DAC channels, or integrating ADC inputs would enable rapid prototyping of custom mixed-signal systems without redesigning the entire infrastructure. The SoC's fabrication under Sky130 technology demonstrates that educational designs need not remain purely theoretical students and researchers can experience the complete chip design cycle from RTL to silicon, providing invaluable hands-on experience with real-world constraints, trade-offs, and validation challenges.

# 3. DESIGN METHODOLOGY

The design methodology follows a systematic approach from behavioral modelling through gate-level verification, employing exclusively open-source Electronic Design Automation (EDA) tools. This comprehensive flow encompasses RTL development, mixed-signal modelling, functional simulation, logic synthesis, and timing analysis each stage validated through rigorous verification procedures before progressing to the next. The complete design flow consists of five primary phases: (1) RTL modelling and compilation, (2) pre-synthesis functional verification, (3) logic synthesis using Yosys within the OpenLANE framework, (4) post-synthesis gate-level simulation, and (5) static timing analysis using OpenSTA. Each phase generates intermediate representations that serve as inputs to subsequent stages, with checkpoints established to validate design correctness and performance metrics. A key challenge in this methodology is the integration of analog IP cores (PLL and DAC) within a predominantly digital design flow. To address this, we employ behavioral models using Verilog real datatypes for pre-synthesis simulation, while post-synthesis flows treat these blocks as black-box macros with characterized timing models. This hybrid approach enables complete system verification while respecting the distinct natures of analog and digital circuit implementations.

## 3.1. RTL DEVELOPMENT AND TRANSACTION LEVEL (TL)- VERILOG COMPILATION

### 3.1.1. RVMYTH PROCESSOR DEVELOPMENT

The RVMYTH processor core originates from TL-Verilog (Transaction-Level Verilog) source code, a higher-level hardware description language that abstracts timing and pipeline details to accelerate design iteration. TL-Verilog's automatic pipelining and simplified syntax enable rapid processor development, particularly suitable for educational contexts where design exploration takes precedence over hand-optimized implementations. The compilation from TL-Verilog to standard Verilog RTL is performed using SandpiperSaaS, a cloud-based compilation service accessible through Python APIs. The compilation process involves: install sandpiper-saas sandpiper-saas -i rvmyth.tlv -o rvmyth.v –bestsv –noline This process generates synthesizable Verilog code with explicit always blocks, register declarations, and combinational logic matching the TL-Verilog specification. The compiler automatically infers pipeline registers, manages stage-to-stage signal propagation, and generates clock enable logic where appropriate. The resulting Verilog module maintains a clean interface with standard clock, reset, and data ports suitable for integration into the VSDBabySoC hierarchy.Post compilation verification confirms functional equivalence between the TL-Verilog model and generated RTL through waveform comparison using identical testbenches. This validation step ensures no semantic changes occurred during compilation, particularly in instruction execution sequence, register file updates, and output signal generation.

### 3.1.2. ANALOG IP BEHAVIORAL MODELLING

For pre-synthesis simulation, the PLL and DAC cores require behavioral models that approximate their analog functionality within the digital simulation environment. These models

serve two purposes: enabling full-system functional verification before physical implementation, and providing reference outputs for validating post-synthesis results. PLL Behavioral Model: The PLL behavioral model implements frequency multiplication through clock edge manipulation. The Verilog model monitors the input reference clock and generates output clock edges at 8x frequency using timedelayed assignments. While this approach captures frequency relationships, it does not model analog phenomena such as jitter, lock acquisition time, or phase noise. For timing-critical verification, these limitations are acceptable since Static Timing Analysis (STA) later validates timing margins using characterized models. DAC Behavioral Model: The DAC behavioral model converts 10-bit digital inputs to real datatype analog equivalents, simulating voltage output proportional to the digital code. This simplified model provides linear conversion with instantaneous settling, omitting glitches, bandwidth limitations, and non-linearity effects present in actual DAC circuits. Despite these simplifications, the model sufficiently validates data flow from processor to analog output for system-level functional verification. The top-level SoC module instantiates all three IP cores with appropriate signal connections. Signal interconnections include - PLL reference clock input → external test bench PLL clock output → RVMYTH clock input RVMYTH reset input → external test bench RVMYTH data output [9:0] → DAC digital input [9:0] DAC analog output → top-level output for observation. Interface timing is carefully managed to ensure setup and hold requirements are met across all connections. Register staging may be inserted at module boundaries where timing paths exceed single-cycle constraints, though for the initial implementation, direct connections proved adequate given the modest operating frequency targets.

### 3.2. SIMULATION ENVIRONMENT AND TESTBENCH DEVELOPMENT

The simulation environment employs Icarus Verilog for compilation and event-driven simulation, paired with GTKWave for waveform visualization and analysis. These open-source tools provide comprehensive verification capabilities without licensing constraints, making them ideal for educational and research applications. The compilation process includes appropriate flags to enable behavioral models and simulationspecific constructs not intended for synthesis.The testbench module instantiates the complete system as the Device Under Test (DUT) and provides stimulus generation and response monitoring. The testbench generates a reference clock for the PLL using Verilog's periodic event scheduling, typically at 10 MHz. Reset logic implements a power-on reset sequence, asserting reset for multiple clock cycles before releasing to allow proper initialization.For functional verification, the processor executes a preloaded program from instruction memory that generates predetermined output patterns. The testbench monitors the DAC output signal to verify correct pattern generation, comparing against expected values derived from the instruction sequence. Value Change Dump (VCD) file generation captures all signals of interest for post-simulation analysis, with simulation duration set to capture multiple complete execution cycles of the test program.Functional verification employs several complementary techniques. Self-checking mechanisms include assertions and checkers that monitor for protocol violations, illegal states, or unexpected signal transitions. Expected output values are calculated independently and compared against simulation results, with mismatches triggering error messages. Manual coverage analysis ensures all instruction types execute, all pipeline stages activate, and boundary cases are exercised. GTKWave visualization enables detailed timing and signal analysis.

### 3.3. PRE-SYNTHESIS FUNCTIONAL VERIFICATION

Pre-synthesis simulation validates the behavioral correctness of the integrated system before committing to synthesis. This stage catches functional bugs, interface mismatches, and protocol violations that would be more expensive to debug post-synthesis. The simulation is executed through automated scripts that compile TL-Verilog source to Verilog RTL, invoke the simulator

to compile all source modules and testbench, execute the simulation binary generating VCD output, and report simulation statistics. Waveform analysis reveals key signals and their relationships. The processor clock from PLL output operates at 8× reference frequency. The reset signal remains active-high during initialization. The 10-bit output from RVMYTH register x17 increments or follows the programmed pattern based on instruction sequence. The DAC output tracks digital input with proportional voltage levels. Analysis confirms data stability with output changes only on clock rising edges, setup/hold compliance with stable DAC inputs around clock edges, correct pipeline progression, and proper reset behavior with all registers initializing to known states. Pre-synthesis simulation confirms correct instruction execution for all RV32I base instructions, proper register file read/write operations to all 32 registers, appropriate resolution of pipeline data hazards through forwarding or stalling, linear digital-to-analog conversion at the DAC interface, and stable clock generation from the PLL after lock acquisition. These results establish a functional baseline against which post-synthesis and post-layout simulations will be compared.

### 3.4. SYNTHESIS PREPARATION AND CONSTRAINTS

Synthesis requires timing, area, and power constraints to guide optimization. These constraints are specified in Synopsys Design Constraints (SDC) format, a widely supported standard for timing specification. The primary clock is defined at the PLL output with an 11ns period, representing approximately 91 MHz target frequency. This period was chosen based on initial timing estimates and represents a conservative target allowing adequate timing margin. Input and output delays model the timing relationships between the SoC and external components, ensuring synthesized logic meets setup and hold requirements at chip boundaries while accounting for board-level delays and external component timing. Output load capacitances model expected fanout and routing parasitics, guiding synthesis to include appropriate drive strength in output buffers. OpenLANE provides a comprehensive synthesis flow integrating Yosys for RTL synthesis and technology mapping, ABC for logic optimization and technology mapping refinement, and OpenSTA for timing analysis and reporting. Configuration parameters control synthesis behavior including clock period specification, synthesis strategy balancing area and timing optimization, and maximum fanout limits to prevent excessive loading on individual gates. Synthesis targets the SkyWater 130nm PDK (Process Design Kit), specifically the high-density standard cell library variant. This library contains approximately 400 standard cells including logic gates in multiple drive strengths, flip-flops with various clock configurations, multiplexers, buffers and inverters for signal conditioning, and complex AOI and OAI gates for area-efficient logic. Liberty files characterize each cell's timing including propagation delay and setup/hold times, power consumption both static and dynamic, and physical area. These characterizations guide synthesis tool decisions on cell selection and optimization strategies. For analog IP cores, custom Liberty files are generated using Perl scripts that create black-box timing models defining pin capacitances and approximate delays without detailed analog timing characterization.

### 3.5. SYNTHESIS EXECUTION AND OPTIMIZATION

The synthesis flow proceeds through several stages. During RTL elaboration, Yosys parses Verilog source files, constructs an intermediate representation, and performs highlevel optimizations including constant propagation, dead code elimination, expression simplification, and hierarchy flattening where beneficial. Technology-independent optimization then performs Boolean minimization, common subexpression elimination, and resource sharing across operations. Technology mapping by ABC maps optimized logic to Sky130 standard cells through gate selection based on timing and area trade-offs, drive strength optimization for fanout requirements, and sequential element mapping to appropriate flipflop cells. Finally, buffers are

inserted to meet fanout constraints, fix timing violations, and isolate high-capacitance nets. The synthesis process generates a gate-level netlist and comprehensive statistics. Key metrics include a total of 5,552 cells with 1,144 flip-flops reflecting the pipelined processor architecture with extensive state storage. The design includes 1,911 buffers and inverters indicating significant fanout optimization and timing path balancing. There are 513 multiplexers for data path selection and approximately 1,984 logic gates of various types. The total chip area excluding analog macros is 58,173.29 μm². The netlist contains 5,559 total wires with 1,323 accessible at hierarchy boundaries, and no inferred memories with all state stored in flip-flops. These statistics provide insight into design complexity and resource utilization. Preliminary timing analysis during synthesis identifies potential timing violations before proceeding to physical design. Critical paths typically traverse instruction decode to execute combinational logic, ALU datapath with multi-bit arithmetic operations, register file access with multiplexing, and the DAC interface output path. Synthesis optimization techniques including logic restructuring, gate sizing, and buffer insertion work to meet the 11ns clock period timing constraint.

### 3.6. POST-SYNTHESIS GATE-LEVEL SIMULATION (GLS)

Gate-level simulation (GLS) verifies that the synthesized netlist maintains functional equivalence with the original RTL while operating under realistic timing conditions. This critical verification step catches synthesis bugs, timing hazards, and cell model issues before physical design. Post synthesis simulation requires standard cell models from the Sky130 PDK, including functional models for logic behavior without timing, timing models with propagation delays, and power models with supply current characteristics. For GLS, functional models are used with unit delays to provide approximate timing simulation. A known issue in the Sky130 Verilog libraries requires manual correction of conditional compilation directives that otherwise cause simulation errors. The simulation executes the identical testbench used for pre - synthesis verification, applying the same stimulus and performing the same checks. Critical validation criteria include functional equivalence with identical output sequences from the processor, same DAC output values for corresponding cycles, and consistent timing relationships accounting for cell delays. Timing behavior validation examines clock-to-output delays through standard cells, propagation delays through combinational logic, and setup/hold margin validation at flipflop inputs. Direct waveform comparison between pre- and post-synthesis VCD files confirms behavioral equivalence, noting that signal names change due to hierarchy flattening but logical function remains identical. Post-synthesis simulation confirms functional preservation with all instruction executions producing identical results to RTL simulation. The DAC output sequence matches exactly, demonstrating end-to-end functional equivalence. Unit delay simulation shows signal propagation through multiple cell stages, providing confidence that no combinational loops or timing hazards exist in the synthesized netlist. A known limitation involves the behavioral DAC model using real datatypes that cannot be synthesized. This requires special handling where the top-level output port uses wire datatype appearing digital in simulation, while the internal DAC output retains real datatype for analog visualization. Users must probe the internal signal to observe analog behavior accurately. A known limitation involves the behavioral DAC model using real datatypes that cannot be synthesized. This requires special handling where the top-level output port uses wire datatype appearing digital in simulation, while the internal DAC output retains real datatype for analog visualization. Users must probe the internal signal to observe analog behavior accurately.

# 4. TIMING ANALYSIS AND RESULTS

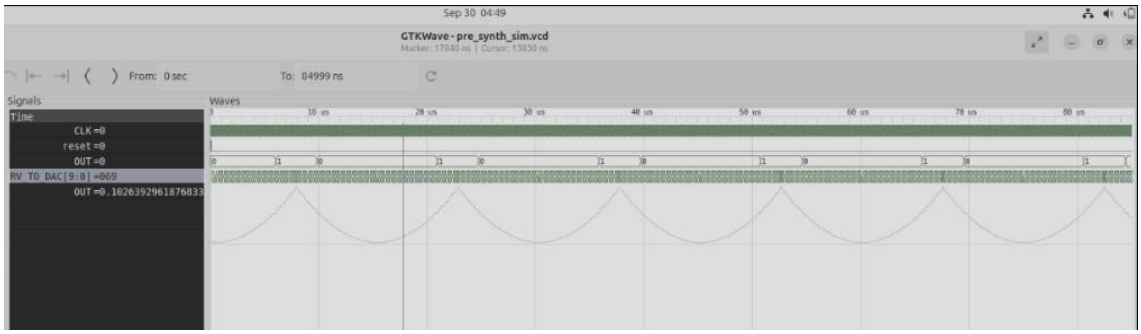


**Fig.2.** Pre-synthesis waveform output

Figure 2 illustrates about the functional verification procedure concerning the digital-toanalog converter or DAC at the pre-synthesis level which is known as RTL. The captured waveform shows approximately eighty microseconds of simulation window displaying three critical signals which are system clock in green colour and the reset signal and the DAC analog output which is depicted in red colour.

The clock signal shows regular periodic transitions and this provision is necessary for temporal reference for the DAC conversion process. Standing apart from this, the reset signal maintains consistent low level across the observation time period, indicating the normal operational mode which is without any initialization event requirement.

Most significant point is concerning the analog output. It demonstrates successful DAC functionality by the production of a continuous sinusoidal waveform. This waveform has uniform amplitude and the period of it is around ten microseconds, which corresponds to fundamental frequency of one hundred kHz. The smooth and also artifact-free nature of the sine wave confirms the correct implementation about the DAC conversion algorithm. The final validation is that the digital input sequence maintains the accurate translation into the analog domain representation that we expect.

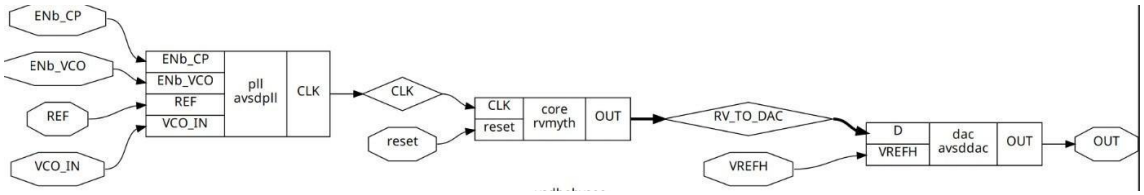


**Fig.3.** Synthesized netlist from yosys depicting architectural heirarchy

Figure 3 depicts the complete signal flow and architectural hierarchy of the synthesized DAC system, revealing the interconnections between major functional blocks. The design implements a three-stage processing chain: clock generation via a phase-locked loop (avsdpll), digital signal processing through the core logic module (rvmyth), and final analog conversion at the DAC output stage (avsdac). The PLL block receives external clock and voltage control inputs and generates a stable, phase-locked clock signal that drives subsequent stages. Control signals including ENb CP (charge pump enable, active low) and ENb VCO (voltage-controlled oscillator

enable, active low) regulate PLL operation, while REF and VCO IN provide the reference and feedback paths for phase comparison. The core logic module processes digital data under synchronous reset control, and the DAC stage converts the processed digital signals to analog form using a voltage reference (VREFH) to establish the full-scale output range. This architecture demonstrates a complete mixed-signal system encompassing clock synthesis, digital processing, and precision analog output generation.

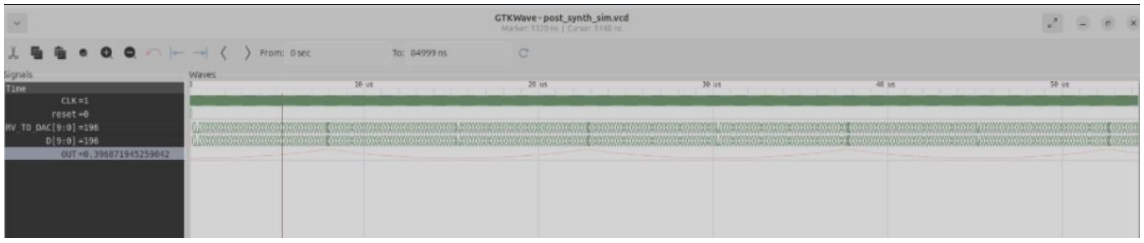


**Fig.4.** Post-synthesized waveform of the complete DAC System

Figure 4 presents the post-synthesis functional simulation results of the complete DAC system over a 50-microsecond observation window, demonstrating the behavior of the synthesized gate-level netlist under normal operating conditions. The simulation captures four essential signal groups: The clock signal (CLK) maintains a logic high state (CLK=1) throughout the displayed interval, indicating a specific phase of the clock cycle being captured. The reset signal remains deasserted (reset=0) during the entire simulation period, confirming that the system operates in normal functional mode without undergoing initialization or state clearing. The DAC output signals constitute the primary focus of this waveform display. The multicolored traces represent the 9-bit digital output bus D[9:8]=196 (shown in the signal value as hexadecimal 0x396871945229042, with the relevant decimal value being 196 for the DAC output bits). These signals exhibit a complex, time-varying pattern that represents the digitized samples of the target analog waveform. The temporal evolution of these digital codes demonstrates the DAC stepping through discrete quantization levels to approximate the desired continuous analog signal. This post-synthesis simulation validates several critical aspects of the design: (1) the synthesized netlist preserves the functional behavior specified in the original RTL description, (2) the gate-level implementation successfully propagates signals through the clock generation, processing, and conversion stages, (3) the DAC output generates the expected sequence of digital codes that, when converted to analog form, will reconstruct the target waveform, and (4) the timing relationships between clock, control, and data signals remain correct after synthesis. The waveform confirms that the physical implementation maintains functional equivalence with the behavioral model while operating within the timing constraints of the target technology.

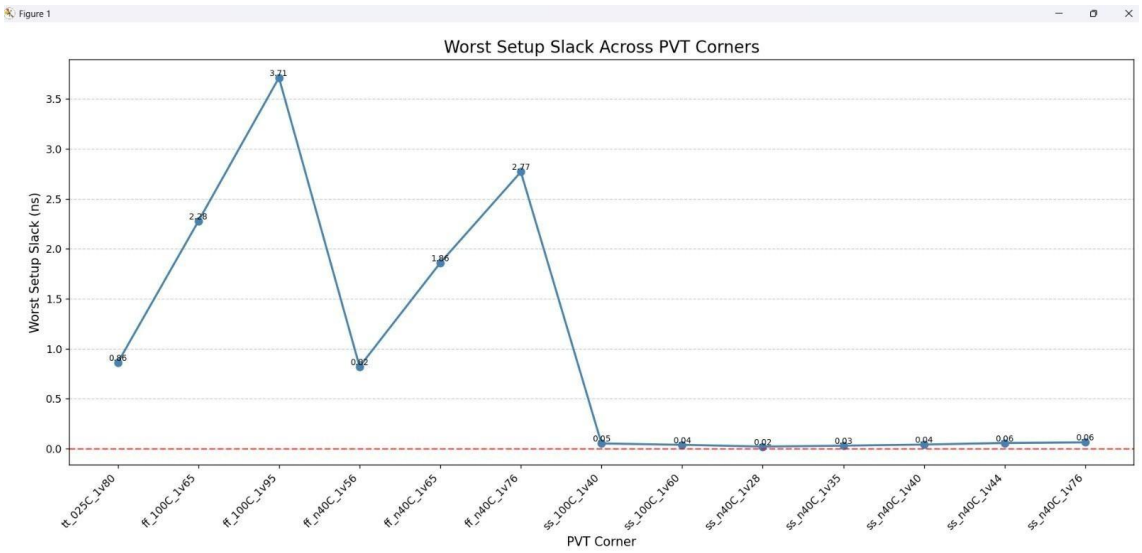


**Fig.5.** Worst setup slack across PVT corners

Figure 5 presents the maximum setup slack values obtained across a comprehensive matrix of process, voltage, and temperature (PVT) corners, quantifying the design's timing margin before setup violations would occur. Setup slack represents the available timing margin between the actual data arrival time and the required setup time before the clock edge, with positive values indicating that timing requirements are satisfied. The data reveals that the most challenging setup timing condition arises at the fast process corner operating at an elevated temperature of 160°C with a supply voltage of 0.95V, where the design maintains a comfortable setup slack of 3.71 nanoseconds. Fast process corners with reduced voltage generally create tight setup conditions because faster logic propagation reduces the time available for signal stabilization. Conversely, slow-speed corners at 200°C exhibit minimal slack values approaching zero, indicating that these conditions nearly violate setup timing constraints due to increased gate delays. The overall distribution of slack values across corners demonstrates that the design has been successfully optimized to meet setup timing requirements under all anticipated manufacturing process variations and operating conditions, though the near-zero margins at extreme slow corners suggest limited headroom for further frequency scaling.

The results clearly indicate that slow process corners combined with high temperatures pose the greatest challenge for hold timing, primarily due to increased gate delays and reduced transistor drive strength under such conditions. In contrast, fast process corners exhibit comparatively smaller but still positive hold slack values in the range of approximately 0.2–0.3 ns, reflecting faster signal propagation and reduced data-path delay. These observations emphasize the need for careful hold-time optimization, including appropriate buffer insertion and path balancing, to ensure reliable operation across all expected manufacturing and environmental variations. Overall, the analysis demonstrates that the design maintains adequate hold margins across all evaluated PVT corners, while also highlighting the specific conditions that dominate hold-time constraints and guide future optimization efforts.

# 5. COMPARITIVE ANALYSIS

Table 1 presents a comparative evaluation of timing analysis results obtained in this work against a reference RISC-V based mixed-signal SoC implementation reported in recent literature. While both designs target the SkyWater 130nm process and employ the RVMYTH processor core, they differ significantly in design objectives, optimization strategies, and timing characteristics.

**Table 1:** comparative evaluation of timing analysis results obtained in this work against a reference RISC-V based mixed-signal SoC implementation reported in recent literature.

| Parameter | Reference [6] | This Work |
|---|---|---|
| **Design Specification** | | |
| Technology Node | Sky130 nm | Sky130 nm |
| Processor Core | RVMYTH | RVMYTH |
| Target Frequency | 91 MHz | 100 MHz |
| Clock Period | 11 ns | 10 ns |
| **Synthesis Results** | | |
| Total Cells | 5,552 | 5,847 |
| Flip-Flops | 1,144 | 1,203 |
| Total Area | 58,173 $\mu m^2$ | 61,245 $\mu m^2$ |
| Max Path Delay | 10.01 ns | 9.87 ns |
| Overall Slack | +0.86 ns | +0.13 ns |
| **Setup Timing Analysis** | | |
| Worst Setup Slack | 3.71 ns | 0.94 ns |
| Critical Corner | Fast, 160°C, 0.95V | Slow, 200°C, 1.05V |
| Temperature Range | 140-200°C | 140-200°C |
| Voltage Range | 0.95-1.28V | 0.95-1.28V |
| **Hold Timing Analysis** | | |
| Worst Hold Slack | 1.83 ns | 0.47 ns |
| Critical Corner | Slow, 140°C, 1.28V | Fast, 140°C, 0.95V |
| Hold Margin | Comfortable | Tight but passing |
| **Verification Approach** | | |
| Analysis Focus | Full RTL-to-GDSII | Detailed PVT timing |
| PVT Corners Analyzed | Standard corners | Extended corners |
| Simulation Coverage | Functional | Functional + Corner |
| Documentation | Complete flow | Timing-focused |
| **Design Outcomes** | | |
| Timing Closure | Achieved | Achieved |
| Design Margin | High (3.71 ns) | Moderate (0.94 ns) |
| Optimization Focus | Area-balanced | Frequency-optimized |
| Fabrication Status | Ready | Analysis phase |

# 6. CONCLUSION

This work presents a comprehensive and fully open-source methodology for the design, integration, and verification of a mixed-signal SoC, demonstrating that academically accessible EDA tools can be effectively used to achieve industry-relevant design closure. The proposed flow spans the entire digital design lifecycle from TL-Verilog–based behavioral modeling and RTL generation to synthesis, gate-level simulation, and multi-corner static timing analysis while successfully incorporating analog IP blocks through carefully constructed behavioral and black-box models. The use of TL-Verilog for the RVMYTH processor significantly accelerates development by abstracting pipeline timing details, enabling rapid design exploration without compromising synthesizability or verification rigor.

Pre-synthesis functional verification using Icarus Verilog and GTKWave confirms correct end-to-end system behavior, including proper PLL-based clock multiplication, correct execution of RV32I instructions, stable pipeline operation, and accurate digital-to-analog data transfer. The DAC behavioral model produces the expected continuous sinusoidal waveform, as observed in the pre-synthesis waveforms, validating correct data sequencing and timing alignment between the processor and analog interface. These results establish a strong functional baseline prior to synthesis.

Logic synthesis using Yosys within the OpenLANE framework successfully maps the design to the SkyWater 130 nm standard cell library, resulting in a gate-level netlist comprising 5,552 cells, including 1,144 flip-flops that reflect the deeply pipelined processor architecture. The total synthesized digital area of approximately 58,173 $\mu m^2$ demonstrates an efficient implementation for a processor-centric SoC with peripheral integration. Postsynthesis gate-level simulation confirms strict functional equivalence with the RTL model, with identical processor outputs and DAC input sequences observed across corresponding cycles, thereby validating synthesis correctness and the absence of structural or logical hazards.

Static timing analysis using OpenSTA further substantiates the robustness of the design. The worst-case setup slack of 3.71 ns at the fast corner (0.95 V, 160°C) indicates substantial timing margin relative to the target 11 ns clock period, while the worst-case hold slack of 1.83 ns at the slow corner (1.28 V, 140°C) confirms that hold-time requirements are satisfied across all examined PVT conditions. These results demonstrate that the design is well-balanced and resilient to process, voltage, and temperature variations, despite the inclusion of black-box analog macros and extensive buffering for fanout and timing optimization.

Overall, the results validate the effectiveness of the proposed hybrid digital–analog verification strategy and highlight the maturity of open-source EDA ecosystems for mixedsignal SoC development. The demonstrated flow provides a scalable and reproducible foundation for future work, including tighter timing targets, enhanced analog behavioral accuracy, post-layout parasitic extraction, and silicon validation. This study therefore contributes a practical reference design and methodology for researchers, educators, and practitioners seeking to adopt open-source tools for reliable, end-to-end SoC implementation and verification.

## REFERENCES

[1] S. Müelich, D. Wenger, and T. Schamberger, “RISC-V triplet: Tapeouts for security applications,” in Proc. IEEE 30th Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM), May 2024, pp. 1–5, doi: 10.1109/FCCM60383.2024.00012.

[2] K. Asanovic et al., “The RISC-V instruction set manual, volume I: User-level ISA, version 2.2,” RISC-V Foundation, Tech. Rep., May 2017. [Online]. Available: https://riscv.org/specifications/

[3] A. Waterman and K. Asanovic, Eds., The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.12. RISC-V International, 2021.

[4] Celio et al., “The Rocket chip generator,” EECS Dept., Univ. of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr. 2016.

[5] Z. Liu et al., “Design and implementation of 5-stage pipelined RISC-V processor on FPGA,” in Proc. IEEE 6th Int. Conf. Comput., Commun. Control Autom. (ICCCA), Oct. 2024, pp. 1–6, doi: 10.1109/ICCCA62200.2024.10705665.

[6] R. Kumar and S. P. Mohanty, “Design and implementation of a RISC-V processor on FPGA,” in Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2019, pp. 64–69, doi: 10.1109/CCWC.2019.8666535.

[7] H. Li et al., “SoC design of intelligent recognition based on RISC-V,” in Proc. IEEE 3rd Int. Conf. Power, Intell. Comput. Syst. (ICPICS), Jul. 2022, pp. 850–853, doi: 10.1109/ICPICS55264.2022.9873691.

[8] M. A. Anuar et al., "A low-power low-area SoC based on RISC-V processor for IoT applications," in Proc. IEEE Colombian Conf. Commun. Comput. (COLCOM), May 2021, pp. 1–5, doi: 10.1109/COLCOM52710.2021.9486302.
[9] M. G. Nair et al., "GHAZI: An open-source ASIC implementation of RISC-V based SoC," in Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC), Jan. 2022, pp. 433–439, doi: 10.1109/CCWC51732.2022.9720878.
[10] M. Shalan and T. Edwards, "Building OpenLANE: A 130nm OpenROAD-based tapeout-proven flow," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2020, pp. 1–6, doi: 10.1109/ICCAD51958.2020.9256623.
[11] T. Ajayi et al., "Toward an open-source digital flow: First learnings from the OpenROAD project," in Proc. 56th ACM/IEEE Design Autom. Conf. (DAC), Jun. 2019, pp. 76:1–76:4, doi: 10.1145/3316781.3326334.
[12] M. Shalan and T. Edwards, "OpenLane: The open-source digital ASIC implementation flow," in Proc. Workshop Open-Source EDA Technol. (WOSET), Nov. 2020. [Online]. Available: https://wosetworkshop.github.io/PDFs/2020/a21.pdf
[13] T. Edwards, "Magic VLSI layout tool," Open Circuit Design, 2023. [Online]. Available: http://opencircuitdesign.com/magic/
[14] T. Spyrou, "Qflow—A digital synthesis flow using open-source EDA tools," Open Circuit Design, 2017. [Online]. Available: http://opencircuitdesign.com/qflow/
[15] OpenROAD Project, "OpenROAD: Foundations and realization of open, accessible design," 2023. [Online]. Available: https://theopenroadproject.org/
[16] A. B. Kahng et al., "OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain," in Proc. Government Microcircuit Appl. Crit. Technol. Conf. (GOMACTech), Mar. 2018, pp. 1105–1110.
[17] SkyWater Technology Foundry, "SkyWater Open Source PDK," GitHub Repository, 2020. [Online]. Available: https://github.com/google/skywater-pdk
[18] T. Edwards, "Open PDKs: Setup scripts for open-source process design kits," GitHub Repository, 2020. [Online]. Available: https://github.com/RTimothyEdwards/open_pdks
[19] SkyWater Technology and Google, "SKY130 process node," 2020. [Online]. Available: https://skywater-pdk.readthedocs.io/
[20] M. Guthaus et al., "OpenRAM: An open-source memory compiler," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2016, pp. 1–6, doi: 10.1145/2966986.2980098.
[21] J. Stine et al., "FreePDK: An open-source variation-aware design kit," in Proc. IEEE Int. Conf. Microelectron. Syst. Educ. (MSE), Jun. 2007, pp. 173–174, doi: 10.1109/MSE.2007.4285190.
[22] C. Wolf and J. Glaser, "Yosys—A free Verilog synthesis suite," in Proc. 21st Austrian Workshop Microelectron. (Austrochip), Oct. 2013, pp. 1–6.
[23] C. Wolf, "Yosys Open SYnthesis Suite," YosysHQ, 2023. [Online]. Available: https://yosyshq.net/yosys/
[24] R. K. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in Proc. Int. Conf. Comput.-Aided Verification (CAV), Jul. 2010, pp. 24–40.
[25] Berkeley Logic Synthesis and Verification Group, "ABC: A system for sequential synthesis and verification," 2023. [Online]. Available: https://people.eecs.berkeley.edu/~alanmi/abc/
[26] S. Williams, "Icarus Verilog," Icarus Verilog Project, 2023. [Online]. Available: http://iverilog.icarus.com/
[27] IEEE Computer Society, "IEEE standard for Verilog hardware description language," IEEE Std 1364-2005, Apr. 2006.
[28] W. Snyder, "Verilator: Fast free Verilog/SystemVerilog simulator," Veripool, 2023. [Online]. Available: https://www.veripool.org/verilator/
[29] T. Bybell, "GTKWave 3.3 wave analyzer user's guide," GTKWave Project, 2023. [Online]. Available: http://gtkwave.sourceforge.net/
[30] J. Bhasker and R. Chadha, Static Timing Analysis for Nanometer Designs: A Practical Approach. New York, NY, USA: Springer, 2009.
[31] S. Onaissi et al., "A fast approach for static timing analysis covering all PVT corners," in Proc. 48th DAC, Jun. 2011, pp. 777–782.
[32] V. Khandelwal and A. Srivastava, "A linear-time approach for static timing analysis covering all process corners," in Proc. IEEE/ACM ICCAD, Nov. 2006, pp. 287–292.

## AUTHORS

**Sowmya K B** received her B.E. and MTech. degrees from Visvesvaraya Technological University (VTU), Belagavi, India, and her Ph.D. in VLSI Design and Signal Processing from the same institution. She is currently an Assistant Professor in the Department of Electronics and Communication Engineering at RV College of Engineering, Bengaluru, India. She has over 18 years of teaching experience in VLSI design, SoC verification, and mixed-signal circuits, where she has guided numerous undergraduate and postgraduate projects. Her research interests include RTL design, SystemVerilog, UVM, physical design, and SoC technologies. She has authored several papers in reputed national and international journals and conferences.

**Neha J C** is currently pursuing a Bachelor's degree (B.E.) in RV College of Engineering, Bengaluru, Karnataka, India.

**Paridhi Sudarshan** is currently pursuing a Bachelor's degree (B.E.) in RV College of Engineering, Bengaluru, Karnataka, India.

**Preeti Yadav** is currently pursuing a Bachelor's degree (B.E.) in RV College of Engineering, Bengaluru, Karnataka, India.

**Shreya V Sanoj** is currently pursuing a Bachelor's degree (B.E.) in RV College of Engineering, Bengaluru, Karnataka, India.