Open Peripheral Controller (OPC): A Reproducible RTL-to-GDSII ASIC Implementation Using an Open-Source EDA

Lukram Dhanachandra Singh,
Dept. of Electronics Engineering
National Institute of Electronics and Information Technology (NIELIT),Imphal, India
dhana.lukram0@gmail.com

Abstract: In Application-Specific Integrated Circuit (ASIC) design, the increasing accessibility of open-source Electronic Design Automation (EDA) tools has transformed the landscape by providing cost-effective and transparent substitutes to proprietary design environments. Even with these significant advancements in open-source EDA ecosystems, comprehensive case studies representing a complete and reproducible Register Transfer Level (RTL)-to-GDSII implementation remain limited. The design and implementation of an Open Peripheral Controller (OPC), a modular Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB)-based peripheral subsystem developed with fully open-source ASIC design methodology has been presented in this paper. The proposed design integrates a configurable register block, a General-Purpose Input/Output (GPIO) controller, and an APB interface to provide a reusable peripheral subsystem suitable for System-on-Chip (SoC) integration.The complete implementation flow includes RTL design in Verilog HDL, functional verification, logic synthesis using Yosys, physical implementation using OpenROAD Flow Scripts, floorplanning, placement, clock tree synthesis, routing, static timing analysis, and GDSII generation using the SkyWater SKY130 HD Process Design Kit (PDK). The implementation effectively achieved positive timing closure with zero Total Negative Slack (TNS), a worst positive slack of approximately 4.57 ns, an operating frequency of approximately 184 MHz, and an estimated total power consumption of approximately 2.8 mW. Physical verification also produced a Design Rule Check (DRC)-clean layout, which confirmsthe compliance with the SKY130 technology design rules. On above the hardware architecture, this work highlights reproducibility by documenting the complete implementation workflow, software environment, configuration methodology, and automation scripts required to regenerate the final ASIC layout. The proposed implementation shows that modern open-source EDA tools are capable to support an industrial-style RTL-to-GDSII design methodology, which also provides a practical platform for education, research, and rapid ASIC prototyping. The presented OPC implementation can be extended for future open-source SoC developments, serving as a reproducible reference design.

Keywords: Open Peripheral Controller, ASIC Design, OpenROAD, Yosys, Sky130, APB Interface, Open-Source EDA, Physical Design.

(Article history: Received: Oct 31,2025 and accepted Jan, 30,2026)

I. INTRODUCTION

The demand for efficient and reusable hardware Intellectual Property (IP) cores that can accelerate embedded system development while reducing design complexity and time-to-market, has significantly increased due to the rapid growth of System-on-Chip (SoC) technology. One of the fundamental building blocks of modern SoCs is Peripheral controllers, which provides standardized interfaces between processing fundamentals and external devices such as General-Purpose Input/Output (GPIO), Timers, communication peripherals, and memory-mapped configuration registers. The Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB), among the commonly adopted on-chip communication protocols, has become a de facto standard for connecting peripherals with low-bandwidth due to its simple protocol, low power consumption, and ease of integration into hierarchical bus architectures [1], [2].

Conventionally, the development of Application-Specific Integrated Circuit (ASIC) design has relied heavily on proprietary Electronic Design Automation (EDA) tools, including logic synthesis, physical design, timing analysis, and layout verification environments. Even if these commercial solutions deliver mature design capabilities, their high costs of licensing and restricted accessibility often limit their adoption in academic institutions, startups, and independent research laboratories [3]. Therefore, there has been growing interest in development of comprehensive open-source EDA ecosystems that can support the complete RTL-to-GDSII implementation flow using freely available open source, software tools and Process Design Kits (PDKs). Due to the recent developments in open-source hardware infrastructure, the ASIC design landscape has substantially transformed. The introduction of the OpenROAD project [4], [5], demonstrated the feasibility of implementing an autonomous RTL-to-GDSII design flow including synthesis, floorplanning, placement, clock tree synthesis (CTS), routing, and timing optimization. Likewise, the Yosys Open Synthesis Suite has appeared as a robust framework for synthesizing synthesizable Verilog designs into technology-

mapped gate-level netlists well-suited with modern semiconductor libraries [6]. Moreover, the release of the SkyWater SKY130 Process Design Kit (PDK) in public has enabled researchers to perform complete ASIC implementation using an openly accessible 130 nm CMOS technology, thus removing one of the major barriers traditionally associated with custom development of IC [7].

In spite of these significant advancements, many research primarily focuses on the development of individual EDA tools, optimization algorithms, or specific implementation stages, while relativelyvery less studies present an end-to-end, reproducible ASIC implementation using an integrated open-source toolchain. Existing papers frequently emphasize processor cores, accelerator architectures, or benchmark circuits, while practical peripheral subsystems that can be readily integrated into larger SoC platforms are limitedly attended. Furthermore, various existing implementations provide details of only partial implementation, making hard for independent reproduction and verification.

The design and complete ASIC implementation of an Open Peripheral Controller (OPC), a modular APB-based peripheral subsystem developed entirely using an open-source RTL-to-GDSII design methodology is presented in this paper to address this gap. The proposed design integrates an APB slave interface, a configurable register block, and a General-Purpose Input/Output (GPIO) controller into a reusable hardware IP block appropriate for embedded SoC applications. The complete implementation flow is performed using Verilog Hardware Description Language (HDL), Yosys for RTL synthesis, OpenROAD Flow Scripts (ORFS) for physical implementation, OpenROAD for automated place-and-route, and the SkyWater SKY130 HD standard-cell library for technology mapping and physical realization, andfunctional verification, logic synthesis, floorplanning, placement, CTS, routing, static timing analysis, and final GDSII layout generation are all executed within a combined single open-source design environment. This work highlights reproducibility by documenting the complete hardware architecture, implementation methodology, software environment, toolchain configuration, and execution workflow required to regenerate the final ASIC layout. The implementation validates successful timing closure, clean physical design generation, and compliance with the SKY130 design rules,thus a practical reference was established for researchers, educators, and engineers interested in adopting open-source ASIC development methodologies.

The primary contributions of this paper includes a modular Open Peripheral Controller (OPC) architecture featuring an AMBA APB-compliant slave interface, configurable register block, and GPIO controller suitable for SoC integration, a complete and reproducible RTL-to-GDSII implementation methodology based entirely on open-source EDA tools, including Yosys, OpenROAD Flow Scripts, OpenROAD, and the SkyWater SKY130 HD PDK, complete implementation results including logic synthesis, floorplanning, placement, clock tree synthesis, routing, static timing analysis, power estimation, and physical verification, and an openly reproducible implementation framework that can serve as a reference platform for education, research, and future open-source ASIC developments.

II. Related Work

The development of open-source Electronic Design Automation (EDA) tools has significantly improved research and education in digital integrated circuit design. Traditionally, ASIC development has relied on proprietary toolchains such as Synopsys Design Compiler, Cadence Innovus, and Siemens EDA solutions, which offer comprehensive implementation capabilities but require substantial licensing costs. These limitations have motivated the development of open-source alternatives capable of supporting complete RTL-to-GDSII implementation workflows [3], [4].One of the most significant developments in this direction is the OpenROAD Project, which aims to establish a fully autonomous, end-to-end ASIC implementation framework. The OpenROAD flow integrates multiple physical design stages, including floorplanning, global and detailed placement, clock tree synthesis (CTS), routing, parasitic extraction, and timing optimization within a unified environment [4], [5]. Complementing this framework, OpenROAD Flow Scripts (ORFS) provide a standardized implementation environment that automates the complete physical design process while supporting multiple technology nodes and standard-cell libraries. These developments have substantially reduced the complexity of adopting open-source ASIC design methodologies for both academic research and industrial prototyping [5].

At the front end of the design flow, the Yosys Open Synthesis Suite has become one of the most widely adopted open-source logic synthesis tools. Yosys supports synthesizable Verilog HDL, technology mapping, optimization, and formal verification interfaces, making it suitable for FPGA and ASIC implementations [6]. The availability of Yosys has enabled seamless integration between RTL design and downstream physical implementation tools, thereby establishing a complete open-source digital design ecosystem.The public release of the SkyWater SKY130 Process Design Kit (PDK) further transformed open-source silicon development by providing unrestricted access to a production-ready 130 nm CMOS technology [7]. Combined with the OpenROAD toolchain, the SKY130 PDK enables researchers to perform complete ASIC implementation, including physical verification and GDSII generation, without dependence on proprietary fabrication libraries. Consequently, numerous educational and research projects have adopted the SKY130 platform as a reference technology for validating new digital architectures and implementation methodologies [5], [7].

In parallel with advancements in EDA tools, considerable research has been devoted to the design of reusable peripheral subsystems for System-on-Chip (SoC) applications. The AMBA Advanced Peripheral Bus (APB) has become one of the most widely adopted on-chip communication protocols for connecting low-bandwidth peripherals because of its simple interface, low hardware overhead, and deterministic transaction model [1], [2]. APB-based peripherals have been incorporated into numerous commercial and open-source SoC platforms, providing interfaces for GPIO, timers, communication controllers, and memory-mapped configuration registers.Several open-source hardware initiatives, including OpenTitan and LiteX, have demonstrated the feasibility of constructing complete SoC platforms using reusable hardware IPs and open development methodologies [8], [9]. These projects primarily emphasize secure processing platforms, configurable SoC generation, and FPGA-based prototyping. While they provide extensive peripheral libraries, the implementation details of individual peripheral subsystems are generally presented as components within larger SoC frameworks rather than as standalone, reproducible RTL-to-GDSII case studies.

Despite these advances, comparatively few publications describe the complete ASIC realization of a modular peripheral controller using an entirely open-source RTL-to-GDSII design flow. Existing studies often concentrate on individual stages of the implementation process, such as logic synthesis optimization, placement algorithms, or routing techniques, while providing limited information regarding the reproducibility of the complete design methodology. Furthermore, implementation metrics such as synthesis statistics, timing closure, physical layout generation, and design-rule verification are frequently reported independently without presenting an integrated hardware case study.

In contrast to previous work, this paper presents a complete and reproducible RTL-to-GDSII implementation of a modular Open Peripheral Controller (OPC) using an entirely open-source EDA ecosystem. Beyond describing the hardware architecture, the proposed work documents the complete implementation methodology, software environment, automation flow, and experimental evaluation from RTL design through GDSII generation. The presented OPC therefore serves not only as a reusable APB-based peripheral subsystem but also as a practical reference design for researchers, educators, and engineers interested in adopting reproducible open-source ASIC development methodologies.

III. Open Peripheral Controller (OPC) Architecture

A. System Overview

The Open Peripheral Controller (OPC) is designed as a modular peripheral subsystem proposed for integration into embedded System-on-Chip (SoC) architectures. The main objective of the proposed design is to deliver a lightweight, reusable, and configurable peripheral interface able to connect low-bandwidth peripherals through the AMBA Advanced Peripheral Bus (APB). By implementing a modular hardware organization, the OPC enables design scalability, simplifies functional verification, and allowsdirect integration into larger digital systems.

There are three major functional modules in the overall architecture of the proposed OPC, Configuration Register Block, General-Purpose Input/Output (GPIO) Controller, and APB Interface Module, which are integrated through a top-level controller that manages communication among the APB master and the internal peripheral resources. The overall architecture of the proposed Open Peripheral Controller is as shown in the Figure 1.

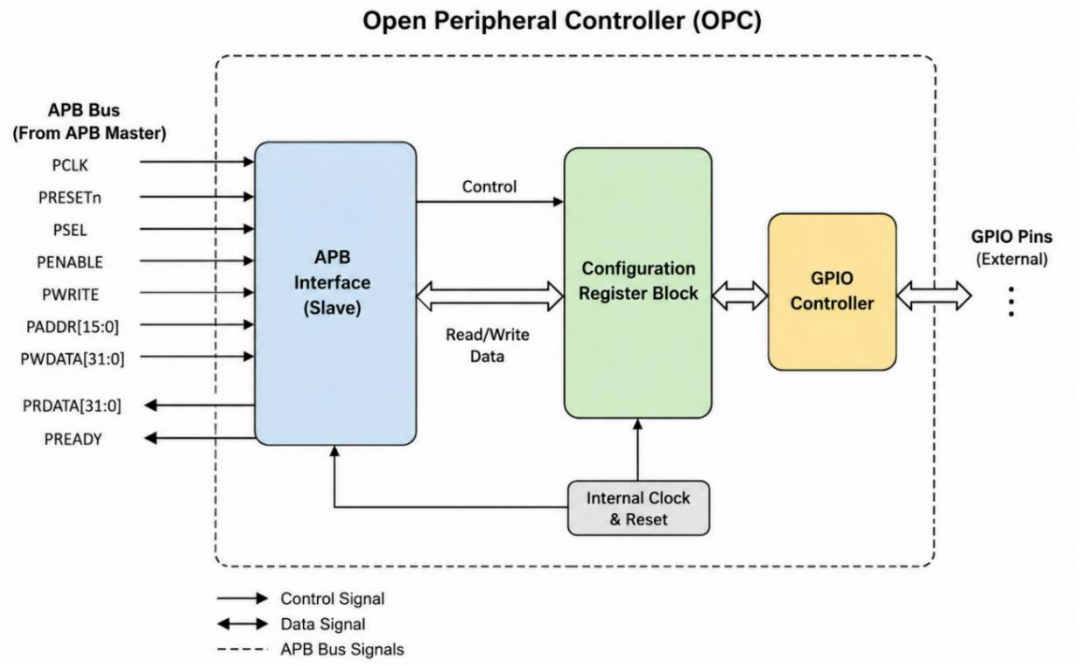


Fig. 1. Overall architecture of the proposed Open Peripheral Controller (OPC).

Read and write transactions from the system bus are collected by the APB interface and decodes the corresponding peripheral addresses and based on these decoded address and control signals, the configuration register block updates the internal control registers or recoversthe values stored during read operations. The GPIO controller works under the control of these configuration registers and offers programmable input/output functionality to external devices.

The modular organization allows each subsystem to be independently designed, verified, and synthesized before integration into the complete peripheral controller. The design maintainability is improved with this hierarchical approach, supporting future expansion through additional peripherals.

B. APB Interface Module

The AMBA Advanced Peripheral Bus (APB) provides a simplified and efficient interface for connecting low-speed peripherals to the system bus, compared to the high-performance buses such as AXI and AHB. APB uses a simple transaction protocol which requires very few control signals with reduced hardware complexity, making it particularly suitable for peripheral devices with modest bandwidth requirements [1], [2].

The proposed OPC implement the APB interface which operates as an APB slave and supports standard read and write transactions through the principal signals, namely, PCLK (Peripheral clock), PRESETn (Active-low reset), PADDR (Address bus), PWDATA (Write data bus), PRDATA (Read data bus), PSEL (Peripheral select), PENABLE (Transfer enable), PWRITE (Read/Write control) and PREADY (Transfer completion indicator).

In write operation, the APB interface decodes the register address of the target and transfers the received data to the configuration register block. While in read operations, the requested register contents are recovered and send back to the APB master through the PRDATA bus. The interface completely follows the standard APB transaction sequence, thus ensuring compatibility with already available AMBA-based SoC architectures.

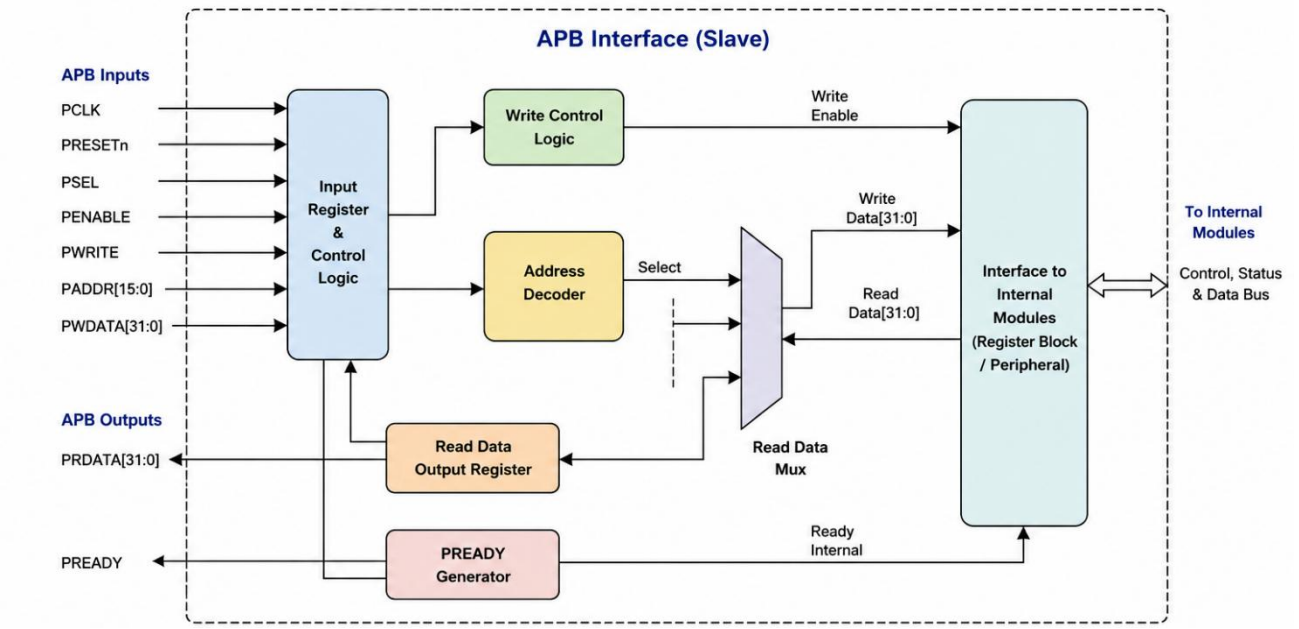


Fig. 2. Internal architecture of the APB interface

C. Configuration Register Block

The configuration register block acts as the central storage element of the proposed Open Peripheral Controller, maintaining the programmable control information required by the GPIO subsystem and provides software-accessible registers for peripheral configuration.Address decoding logic is used to select the appropriate register based on the incoming APB address, and synchronous write logic updates the register contents during valid write transactions. Then the read operations multiplex the selected register value onto the APB read-data bus. The proposed architecture improves modularity and simplifies future expansion through additional programmable registers, by separating the configuration logic from the GPIO controller.

D. GPIO Controller

The GPIO controller delivers programmable digital input and output functionality with software control. External devices are allowed to be monitored or controlled using standard memory-mapped register operations, by configuring each GPIO pin through the associated configuration registers. The controller monitors the configuration registers continuously and updates the output ports accordingly. The current state of the GPIO pins captured during input operations are made available to the processor through APB read transactions. Here, the isolation of GPIO control logic from the APB communication interface allows the GPIO subsystem to stay independent of the communication protocol, thus facilitating reuse in alternative bus architectures if needed.

E. Top-Level Integration

The complete Open Peripheral Controller combines the configuration register block, APB interface, and GPIO controller into anintegrated peripheral subsystem. The top-level module manages data transfer between these functional blocks while preserving a clean hierarchical organization.The operational sequence of the proposed architecture is as the processor initiates an APB transaction., then the APB interface decodes the incoming address and control signals. After which, the configuration register block performs the requested read or write operation and the GPIO controller updates or reports the corresponding peripheral state. And finally, the response is returned to the APB master through the standard APB interface.This organization minimizes coupling between individual modules, allowing independent verification, synthesis, and future enhancement deprived ofaltering the remaining subsystems.

The proposedOpen Peripheral Controller integrates several design features that improve its suitability for reusable ASIC development. They are modular hierarchical RTL architecture for improved maintainability, standard AMBA APB-compliant slave interface to ease SoC integration, memory-mapped configuration registers supporting programmable peripheral control, independent GPIO controller enabling configurable digital input/output functionality, fully synthesizable Verilog HDL implementation compatible with open-source ASIC design flows, technology-independent RTL suitable for implementation across multiple fabrication technologies and it also support the complete RTL-to-GDSII implementation using an entirely open-source EDA toolchain. These features together establish the OPC as a reusable peripheral subsystem that can serve as a reference hardware IP for embedded SoC research, education, and rapid ASIC prototyping.

IV. RTL-TO-GDSII DESIGN METHODOLOGY

A. Design Flow Overview

A complete open-source RTL-to-GDSII Application-Specific Integrated Circuit (ASIC) design methodology is followed for the implementation of the proposed Open Peripheral Controller (OPC). In contrast to the conventional design flows which depend on proprietary Electronic Design Automation (EDA) software, the proposed methodology utilizes a fully open-source toolchain covering every stage from hardware description to final layout generation. The workflow adopted in this work is shown in Figure 4, which integrates RTL design, functional verification, logic synthesis, physical implementation, timing analysis, and GDSII generation into a reproducible design framework.

The implementation initiates with the development of synthesizable Verilog HDL modules that describes the functionality of the Open Peripheral Controller. Functional verification is done through simulation before the design proceeds to logic synthesis. The synthesized gate-level netlist is then processed through stages of automated physical design, which includes floorplanning, placement, clock tree synthesis (CTS), routing, timing analysis, and layout generation. And at last, physical verification confirms the compliance with the fabrication technology design rules.

The proposed workflow highlights automation and reproducibility, allowing the complete implementation to be regenerated using a consistent software environment and standardized implementation scripts.

B. RTL Design and Functional Verification

A hierarchical modular design methodology was followed using synthesizable Verilog HDL in developing the proposed OPC. The RTL architecture comprises of independent functional modules which are interconnected through clearly defined interfaces, which improves readability, maintainability, and verification efficiency.The principal RTL modules areConfiguration Register Block, GPIO Controller, APB Interface, and Top-Level OPC Integration Module. Each module was independently verified before system integration. Verilog testbenches are used to perform Functional verification to validate read/write transactions, register operations, GPIO functionality, and APB protocol compliance. This modular verification approach simplified debugging and improved the complexity of system-level validation.The successful completion of functional simulation validates the precision of the RTL implementation before proceeding to logic synthesis.

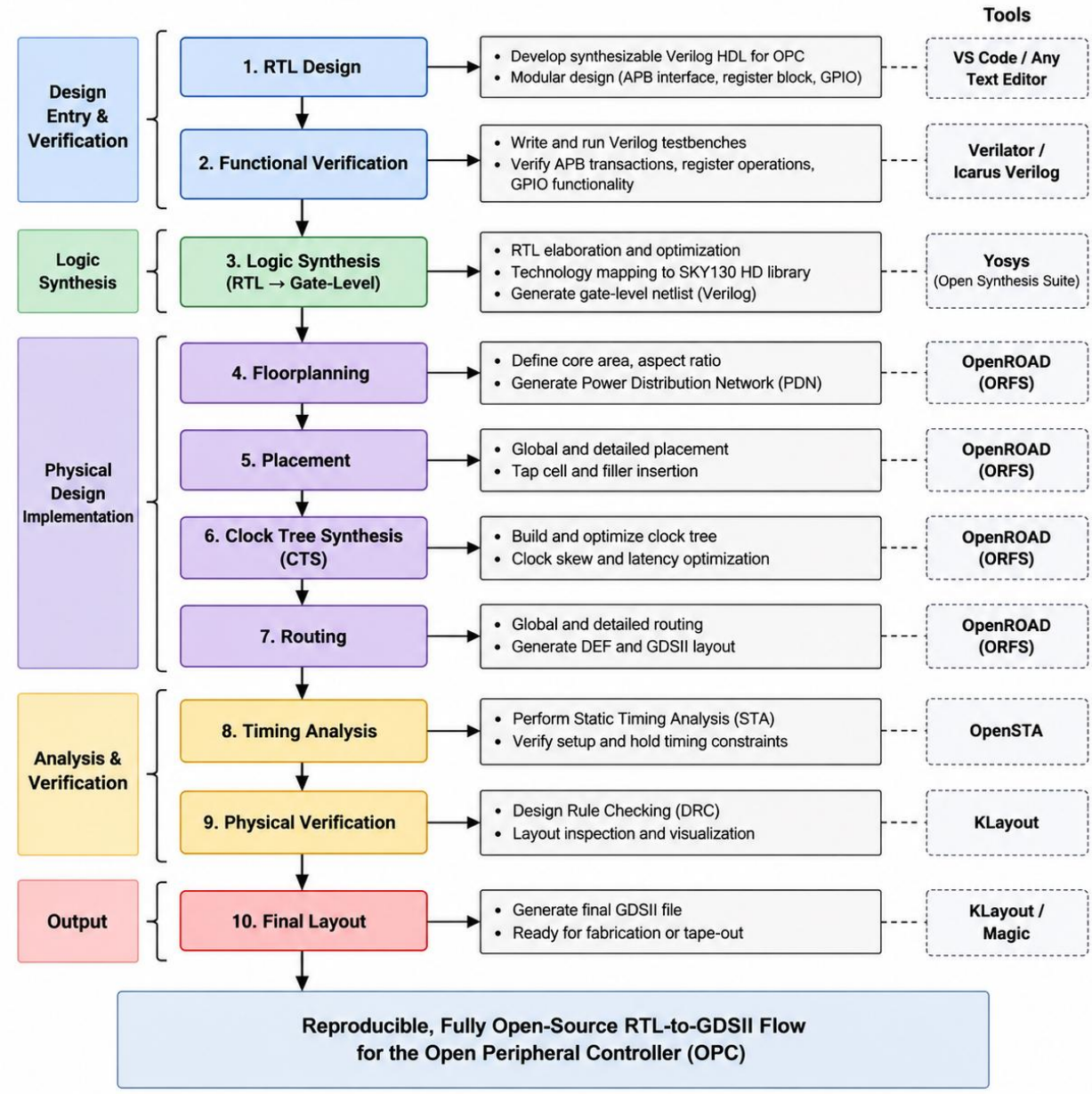


Fig. 3. Complete RTL-to-GDSII implementation flow for the proposed OPC.

C. Logic Synthesis

Logic synthesis transforms the Register Transfer Level (RTL) description into a technology-dependent gate-level netlist by means of standard cells from the target semiconductor library. In this work, the Yosys Open Synthesis Suite was used to perform synthesis, which supports Verilog HDL parsing, optimization, technology mapping, and netlist generation [6].The synthesized OPC design was then mapped to the SkyWater SKY130 HD standard-cell library, which produce a gate-level implementation compatible with the downstream OpenROAD physical design flow.

The synthesis process involved the stages of RTL elaboration, Hierarchical analysis, Logic optimization, Technology mapping, and Gate-level netlist generation. And the generated netlist acts as the starting point for physical implementation and timing-driven optimization.

D. Physical Design Implementation

After logic synthesis, the OpenROAD Flow Scripts (ORFS) is used to implement gate-level netlist. ORFS provides an automated framework which is the integration of multiple physical design stages into a unified implementation environment [5].The physical implementation consists sequential stages as floorplanning, Power Distribution Network (PDN) generation, Tap cell insertion, Global placement, Detailed placement, Clock Tree Synthesis (CTS), Global routing, Detailed routing, Filler cell insertion, DEF generation, and GDSII generation. Each implementation stage

produces intermediate databases, reports, and implementation statistics automatically, that can be used to analyze the physical characteristics of the design.The implementation of ORFS significantly decreases manual intervention, ensuring consistent implementation across repeated design executions.

E. Timing Analysis and Physical Verification

OpenSTA is used to perform Static Timing Analysis (STA), which estimates setup and hold timing constraints across all timing paths in the synthesized design. STA verifies that the implemented design satisfies the specified clock frequency, identifying potential critical paths which requires optimization. After timing analysis, the generated physical layout was observed using KLayout, enabling visualization of the complete ASIC layout generated through the OpenROAD implementation flow.

Physical verification comprisedmainly of Design Rule Checking (DRC) to ensure compliance with the SkyWater SKY130 fabrication rules. Successful DRC validation validates that the generated layout satisfies the manufacturing constraints associated with the target technology.

F. Reproducible Implementation Framework

A key objective of this work is to design a reproducible ASIC implementation methodology. To achieve this objective, the complete implementation environment, RTL source code, OpenROAD Flow Scripts configuration, synthesis scripts, technology libraries, and implementation commands were preserved under version control using Git.The implementation can be reproduced by executing the standardized implementation flow which began from the Verilog RTL description. The use of openly available EDA tools, together with the open SkyWater SKY130 Process Design Kit (PDK), ensures that researchers and educators can self-reliantly regenerate the complete ASIC implementation without any proprietary software licenses.This reproducible workflow delivers a practical foundation for future open-source hardware research, educational laboratories, and rapid ASIC prototyping.

V. Experimental Setup& Results

The proposed Open Peripheral Controller (OPC) was implemented and evaluated based on a complete open-source ASIC design environment. All stages from the RTL-to-GDSII implementation, which includes logic synthesis, physical design, timing analysis, and layout generation, were performed on aninherent Ubuntu Linux platform with Operating System of Ubuntu 24.04 LTS without relying on containerized environments. The use of aninherent installation provides greater flexibility for tool customization, dependency management, and debugging, which ensures compatibility across the complete implementation flow.

The implementation environment consisted of the Yosys Open Synthesis Suite for RTL synthesis, OpenROAD Flow Scripts (ORFS) for automated physical implementation, OpenROAD for placement and routing, OpenSTA for static timing analysis, and KLayout for physical layout visualization and verification. The SkyWater SKY130 HD standard-cell library [4]–[7] was used for this work as the target technology throughout the implementation.The modular RTL source code and implementation scripts were preserved under Git version control, permitting consistent regeneration of all implementation results.

The proposed OPC was configured as an APB-compliant slave peripheral intended for integration into embedded System-on-Chip (SoC) platforms. The implementation targeted the SkyWater SKY130 HD PDKwhich used standard-cell libraries supplied with the OpenROAD design flow.The RTL implementation was developed using synthesizable Verilog HDL and hierarchically organized to simplify modular verification and physical implementation. The synthesized design was then processed using the standard OpenROAD Flow Scripts implementation pipeline.

The primary design artifacts generated in the complete implementation flow includes Synthesized gate-level netlist, Floorplan database, Placement database, Clock tree synthesis database, Routed DEF, Final GDSII layout, Timing reports, Power reports, and Physical verification reports. These artifacts togethercharacterize the complete RTL-to-GDSII implementation of the proposed Open Peripheral Controller.

The proposed Open Peripheral Controller (OPC) was successfully implemented using a complete open-source RTL-to-GDSII design flow based on Verilog HDL, Yosys, OpenROAD Flow Scripts (ORFS), OpenROAD, OpenSTA, and the SkyWater SKY130 HD Process Design Kit. The implementation progressed through all major physical design stages, including logic synthesis, floorplanning, placement, clock tree synthesis, routing, timing analysis, and GDSII generation. The experimental results demonstrate the capability of the open-source EDA ecosystem to realize a complete ASIC implementation for a modular APB-based peripheral controller.

A. Logic Synthesis Results

Logic synthesis was performed for the proposed Open Peripheral Controller using the Yosys Open Synthesis Suite targeting the SkyWater SKY130 HD standard-cell library. The hierarchical Verilog RTL was successfully translated into a technology-mapped gate-level netlist in the synthesis process without any functional errors. The synthesized design

preserves the modular architecture comprising of the APB wrapper, configuration register block, GPIO controller, and top-level integration modules.

The synthesis statistics provides that the complete design contains 198 wires, 1630 wire bits, 963 public wire bits, and 962 synthesized standard cells. The synthesized design hierarchy is as given in Table I.

TABLE I. RTL MODULE SYNTHESIS STATISTICS

RTL Module	Wires	Wire Bits	Public Wires	Cells
APB Wrapper	46	298	20	100
Configuration Register Block	113	933	19	838
GPIO Controller	9	82	8	24
OPC Top	12	116	12	2
OPC APB Top	18	201	18	2

The synthesis results reveal that the proposed OPC architecture is completely synthesizable and competently mapped to the SkyWater SKY130 HD standard-cell library. The Configuration Register Block contributes the main portion of the synthesized logic, accounting for around 87% of the total standard cells, owing to its register storage and associated combinational control logic. The APB Wrapper comprises the bus interface and protocol control logic, while the GPIO Controller reside in a comparatively small area because of its simple input/output management functionality.

TABLE II. STANDARD CELL DISTRIBUTION AFTER TECHNOLOGY MAPPING

Cell Type	Count
AND Gates	277
D Flip-Flops (Enable)	296
D Flip-Flops	20
Multiplexers	102
NOT Gates	18
OR Gates	249
Total	**962**

B. Floorplanning and Placement Results

The synthesized netlist generated by Yosys act as the input for the following physical implementation stages in the OpenROAD Flow Scripts.Thus, the gate-level netlist was imported to OpenROAD Flow Scripts for physical implementation. The floorplanning stage successfully initialized the design core, generated the power distribution network (PDN), inserted tap cells, and formed the physical design database. During floorplanning, a target utilization of 60% was specified, resulting in an effective utilization of 61.2% over a 21,958.56 μm² core area. The design comprises 866 physical instances, including standard cells and implementation-specific cells which was introduced during physical optimization. Figure 4. Shows the floorplanning statistics shown in openROADon Linux terminal.

```
luk_d@DLukram:~/OpenASIC-Design/examples/OPC$ openroad scripts/floorplan.tcl
OpenROAD 26Q3-558-g3976b64471
Features included (+) or not (-): +GPU +GUI +Python
This program is licensed under the BSD-3 license. See the LICENSE file for details.
Components of this program may be licensed under more restrictive licenses which must be ho
nored.
[INFO ODB-0227] LEF file: /home/luk_d/asic/flow/OpenROAD-flow-scripts/flow/platforms/sky130
hd/lef/sky130_fd_sc_hd.tlef, created 13 layers, 25 vias
[INFO ODB-0227] LEF file: /home/luk_d/asic/flow/OpenROAD-flow-scripts/flow/platforms/sky130
hd/lef/sky130_fd_sc_hd_merged.lef, created 441 library cells
[WARNING IFP-0028] Core area lower left (10.000, 10.000) snapped to (10.120, 10.880).
[INFO IFP-0001] Added 54 rows of 325 site unithd.
[INFO IFP-0100] Die BBox:  (  0.000  0.000 ) ( 170.000 170.000 ) um
[INFO IFP-0101] Core BBox: ( 10.120 10.880 ) ( 159.620 157.760 ) um
[INFO IFP-0102] Core area:                    21958.560 um^2
[INFO IFP-0103] Total instances area:         13281.488 um^2
[INFO IFP-0104] Effective utilization:            0.605
[INFO IFP-0105] Number of instances:                866

============================================
 OPC Design Successfully Initialized
============================================
```

Fig. 4. Floorplanning statistics

Global placement and detailed placement then arranged the standard cells within the predefined core area while minimizing wirelength and congestion. The placement optimization stage further refined cell locations, thus improving timing and routing quality. The placement results specify that the modular RTL organization interpreted into an efficient physical layout with enough routing resources for subsequent implementation stages. Figure 5 illustrates the standard-cell placement obtained subsequently the detailed placement stage of the OpenROAD implementation flow. The synthesized standard cells are distributed uniformly within the predefined core region while respecting placement rows and design constraints. The placement optimization process successfully indorsed all movable cells, eliminated overlaps, and minimized wirelength to improve routing and timing performance. The uniform distribution of cells across the core specifies efficient usage of the available placement area, providing a suitable foundation for next clock tree synthesis and routing stages.

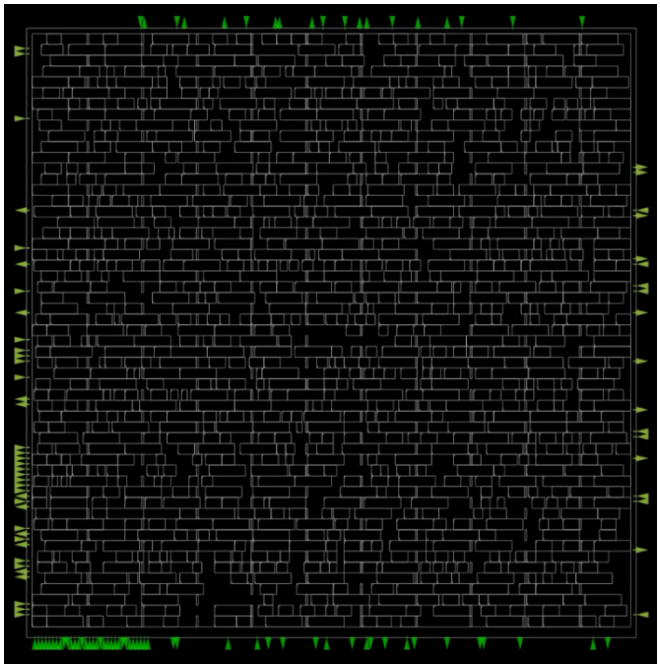


Fig. 5. Standard-cell placement of the proposed OPC after detailed placement optimization.

C. Clock Tree Synthesis and Routing

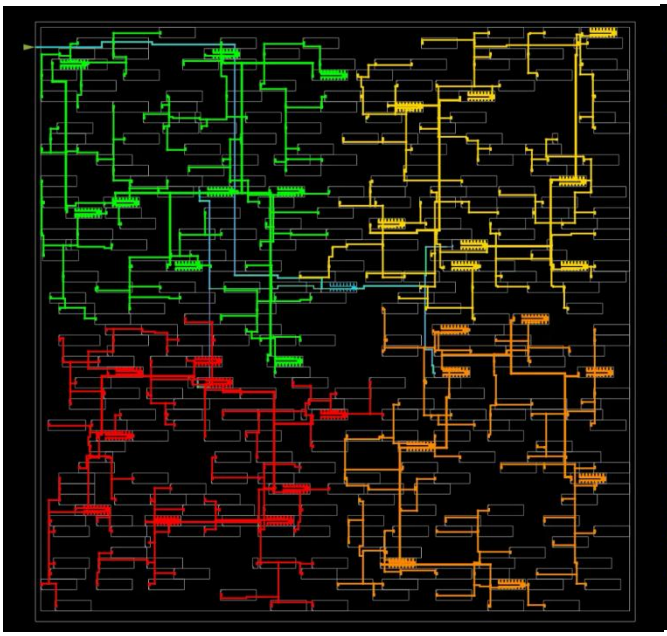


Fig. 6. Clock Tree Synthesis (CTS) layout

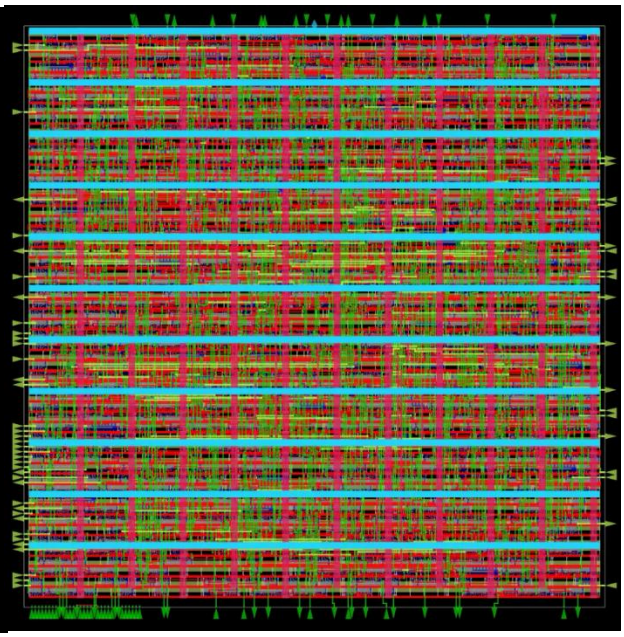


Fig. 7. Final routed layout showing the power distribution network and signal interconnect routing.

Clock Tree Synthesis (CTS) was achieved to construct a balanced clock distribution network while minimizing clock skew and insertion delay. OpenROAD automatically inserted the required clock buffers and optimized the clock network prior to the routing. After CTS, global routing and detailed routing successfully completed all signal interconnections using the available routing resources. Filler cells were subsequently inserted prior to final layout generation.Detailed placement successfully indorsed all standard cells without overlap violations while maintaining routing feasibility. After placement optimization, the design achieved a utilization of 67%, which increased to 71% after Clock Tree Synthesis (CTS) due to the insertion of clock buffers and optimization cells. The lack of placement violations approves that the synthesized design was efficiently mapped onto the target technology and provided a suitable foundation for the subsequent routing stage.

Figure 6illustrates the clock distribution network generated during the Clock Tree Synthesis (CTS) stage. OpenROAD automatically inserts clock buffers and constructs balanced clock branches to distribute the clock signal throughout the design, minimizing insertion delay and clock skew. The color-coded clock network shows that the hierarchical organization of the clock tree, where each branch drives a subset of sequential elements. The balanced clock distribution contributes to the successful timing closure attained by the proposed implementation, as imitated by the positive worst slack and zero total negative slack reported in the timing analysis.Figure 7shows the completed physical implementation after global and detailed routing. The routing stage establishes all signal interconnections while simultaneously integrating the power distribution network (PDN) to provide stable power delivery across the design. Multiple metal layers are used to route signals efficiently while reducing routing congestion and satisfying design-rule constraints. The successful completion of routing proves that the placement and clock tree synthesis stages produced a physically realizable implementation able to support in manufacturable GDSII generation.

The routing stage completed successfully without routing failures, demonstrating that the automatically generated floorplan and placement provided adequate routing resources throughout the design.

D. Timing and Power Analysis

Static Timing Analysis (STA) was performed using OpenSTA following the completion of routing. The timing reports specify that the proposed OPC successfully satisfied the specified timing constraints.Table III gives the principal timing results obtained after implementation.

TABLE III. TIMING PERFORMANCE OF THE PROPOSED OPC

Parameter	**Result**
Technology	SkyWater SKY130 HD
Target Clock	100 MHz
Worst Negative Slack (WNS)	+4.5685 ns
Total Negative Slack (TNS)	0 ns
Minimum Clock period	5.43 ns
Maximum Operating Frequency	184.11 MHz
Setup Skew	0.12 ns

TABLE IV – POWER ANALYSIS

Component	**Power**	**Percentage**
Sequential	1.31 mW	46.9%
Combinational	0.289 mW	10.3%
Clock Network	1.20 mW	42.8%
Leakage	7.42 nW	-
Total	2.80 mW	-

The implementation achieved positive timing slack across all analyzed timing paths, indicating successful timing closure. The reported Total Negative Slack (TNS) of zero further approves that no timing violations were observed in the implemented design. The estimated total power consumption of the proposed OPC is as in Table IV.

The relatively low power consumption determines that the proposed peripheral controller is suitable for embedded and low-power System-on-Chip applications.

E. Physical Verification

The final implementation successfully generated a GDSII layout representing the complete physical realization of the proposed Open Peripheral Controller (OPC).Physical verification established compliance with the SkyWater SKY130 design rules. The generated layout showed no reported Design Rule Check (DRC) violations, indicating that the implemented design satisfies the manufacturing constraints associated with the target technology. Figure 8 and 9 shows the final GDSII layout visualized using KLayout.

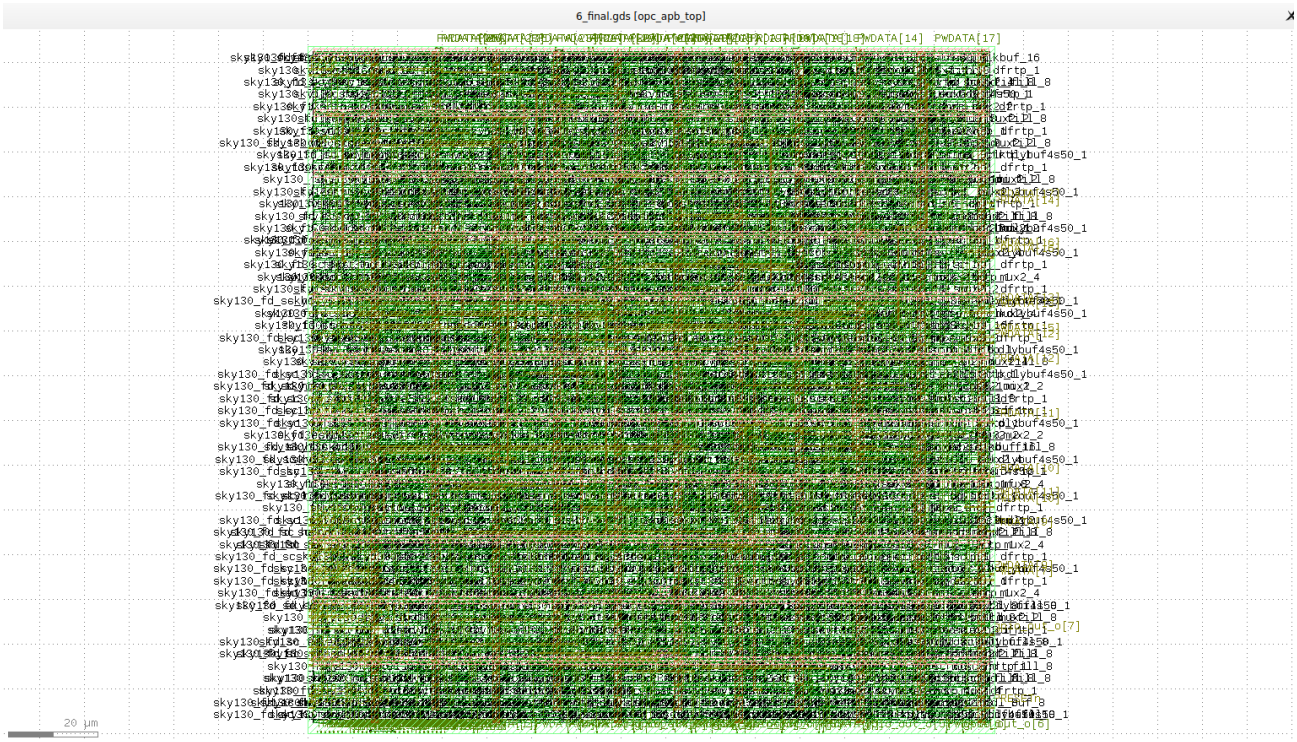


Fig. 8. Final GDSII Layout (KLayout)

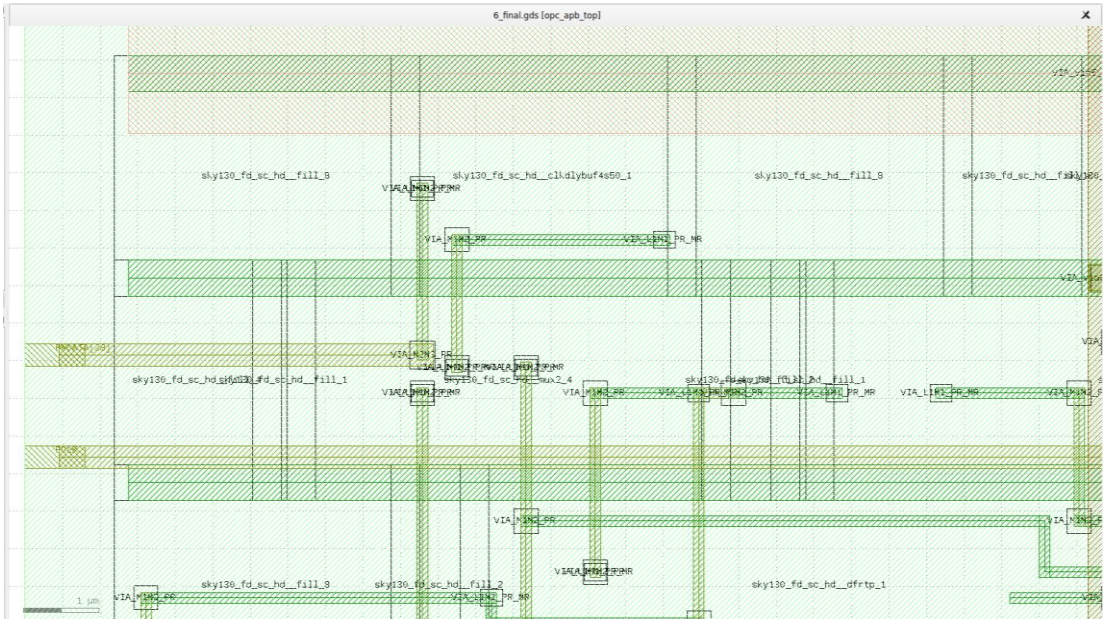


Fig. 9. Zoomed in of a portion of Final GDSII Layout (KLayout)

Although Layout Versus Schematic (LVS) verification reported a pin-count mismatch during the hierarchical verification stage, the issue was associated with the verification interface rather than routing or layout generation. The successful completion of routing, timing analysis, GDSII generation, and DRC verification confirms the correctness of the implemented physical layout.

TABLE V PHYSICAL IMPLEMENTATION SUMMARY

Implementation Stage	Status
RTL Synthesis	✓ Successful
Floorplanning	✓ Successful
Placement	✓ Successful
Clock Tree Synthesis	✓ Successful
Routing	✓ Successful
GDSII Generation	✓ Successful
Design Rule Check (DRC)	✓ Passed
Layout Visualization	✓ Successful

The experimental results prove that the proposed Open Peripheral Controller can be successfully implemented with a completely open-source RTL-to-GDSII design flow. The modular RTL architecture enabled synthesis and physical implementation, while the OpenROAD Flow Scripts automated the physical design process from floorplanning through GDSII generation.

A significant contribution of this work deceits in its reproducibility. Dissimilar to many reported ASIC implementations that rely on proprietary EDA software, the proposed methodology uses publicly available open-source tools throughout the complete implementation process. The combination of Yosys, OpenROAD Flow Scripts, OpenROAD, OpenSTA, KLayout, and the SkyWater SKY130 HD Process Design Kit enables independent researchers to reproduce the reported implementation without requiring commercial software licenses.Furthermore, the implementation achieved successful timing closure with positive slack, zero Total Negative Slack, successful routing, and DRC-clean layout generation. These results demonstrate the maturity of the current open-source ASIC ecosystem for developing reusable digital IP cores suitable for education, research, and prototype System-on-Chip development.

TABLE VIQUALITATIVE COMPARISON OF THE PROPOSED OPC WITH EXISTING OPEN-SOURCE PLATFORMS

Feature	**Proposed OPC**	**OpenTitan**	**LiteX**
APB-Compatible Peripheral Interface	✓	Partial*	Partial*
GPIO Controller	✓	✓	✓
Modular RTL Architecture	✓	✓	✓
Fully Open-Source RTL	✓	✓	✓
Complete RTL-to-GDSII Implementation Documented	✓	Limited	Limited
SkyWater SKY130 ASIC Implementation	✓	Available for selected components	Primarily FPGA-oriented
Reproducible Open-Source Flow	✓	Partial	Partial
Public RTL-to-GDSII Methodology	✓	Limited	Limited

*OpenTitan and LiteX predominantly use TileLink and Wishbone-based infrastructures, respectively, with bridges or wrappers available for interoperability rather than native APB-centric peripheral subsystems.

Table VI highlights that while existing open-source hardware platforms provide extensive collections of reusable IP cores and configurable SoC infrastructures, comparatively few publications document the complete ASIC realization of an individual peripheral controller using a reproducible open-source RTL-to-GDSII workflow. In contrast, the proposed OPC integrates hardware architecture, implementation methodology, physical design, and experimental evaluation within a single reproducible framework.

The emergence of open-source hardware has enabled the development of reusable intellectual property (IP) cores and complete System-on-Chip (SoC) platforms using publicly accessible hardware description languages, design methodologies, and EDA tools. Several open-source projects, including OpenTitan and LiteX, provide configurable peripheral subsystems intended for secure computing and FPGA-based SoC development [8], [9]. Though, these projects primarily focus on complete SoC infrastructures rather than documenting the complete ASIC implementation of an individual peripheral subsystem. The proposed Open Peripheral Controller (OPC) differs from existing implementations by emphasizing reproducible ASIC implementation rather than solely RTL functionality. The presented work documents every stage of the RTL-to-GDSII flow using a fully open-source EDA toolchain, thereby enabling independent reproduction of the complete physical implementation.

Another distinguishing aspect of the proposed work is the emphasis on implementation reproducibility. The RTL source code, implementation scripts, technology configuration, and design flow were maintained under version control and executed using publicly available EDA tools. Consequently, researchers and educators can regenerate the reported implementation results without relying on proprietary software licenses or commercial design environments.The successful completion of synthesis, floorplanning, placement, clock tree synthesis, routing, timing analysis, and GDSII generation further demonstrates the maturity of contemporary open-source EDA tools for developing reusable ASIC intellectual property.

VIII. Conclusion and Future Work

This paper presented the design and complete ASIC implementation of a modular Open Peripheral Controller (OPC) using a fully open-source RTL-to-GDSII design methodology. The proposed architecture integrates an AMBA APB-compliant slave interface, a configurable register block, and a General-Purpose Input/Output (GPIO) controller into a reusable hardware IP suitable for embedded System-on-Chip applications. The complete implementation was carried out using Verilog HDL, Yosys, OpenROAD Flow Scripts, OpenROAD, OpenSTA, KLayout, and the SkyWater SKY130 HD Process Design Kit.

The experimental results demonstrate the successful realization of the proposed design through all stages of the ASIC implementation flow, including logic synthesis, floorplanning, placement, clock tree synthesis, routing, timing analysis, and GDSII generation. The implemented design achieved successful timing closure with a Worst Slack of +4.57 ns, zero Total Negative Slack, an operating frequency of approximately 184 MHz, an estimated total power consumption of approximately 2.8 mW, and a DRC-clean physical layout. These results confirm the effectiveness of the proposed methodology and demonstrate that modern open-source EDA tools are capable of supporting industrial-style ASIC implementation workflows.

A key contribution of this work is the establishment of a reproducible RTL-to-GDSII implementation framework. By integrating publicly available EDA tools with the open SkyWater SKY130 Process Design Kit, the proposed workflow enables researchers, educators, and hardware developers to reproduce the complete ASIC implementation without requiring proprietary software licenses. The Open Peripheral Controller therefore serves not only as a reusable APB-based peripheral IP but also as a practical reference design for open-source silicon research and education.

Future research will focus on extending the functionality and applicability of the proposed Open Peripheral Controller. Planned enhancements include the integration of additional peripherals such as UART, SPI, I²C, PWM, timer/counter modules, interrupt controllers, and watchdog timers to develop a more comprehensive peripheral subsystem for System-on-Chip applications. The scalability of the architecture will also be investigated through parameterized RTL design, enabling configurable GPIO widths, register banks, and peripheral instances.Further studies will explore implementation using advanced semiconductor technology nodes and evaluate the portability of the proposed architecture across multiple open-source and commercial ASIC design flows. Comparative investigations involving FPGA implementations and power-performance-area (PPA) optimization techniques will also be conducted.

Acknowledgment

The authors would like to thank the National Institute of Electronics and Information Technology (NIELIT), Imphal, for providing the infrastructure and facilities that supported this research.

References

[1] Arm Ltd., "AMBA 3 APB Protocol Specification," ARM IHI 0024C, 2004.
[2] Arm Ltd., "AMBA AXI and ACE Protocol Specification," ARM IHI 0022E, 2013.
[3] S. Hassoun and T. Sasao, Logic Synthesis and Verification. Springer, 2002.
[4] T. Ajayi et al., "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain," Proc. ACM/IEEE Design Automation Conference (DAC), 2019.
[5] A. B. Kahng et al., "The OpenROAD Project: Unleashing Hardware Innovation," Proc. ACM/IEEE Design Automation Conference (DAC), 2021.
[6] C. Wolf, "Yosys Open SYnthesis Suite," 2013. [Online]. Available: https://yosyshq.net/yosys/
[7] T. Parry et al., "The SkyWaterOpen-Source PDK," IEEE Custom Integrated Circuits Conference (CICC), 2020.
[8] E. Love et al., "OpenTitan: Open-Source Silicon Root of Trust," IEEE International Symposium on Hardware Oriented Security and Trust (HOST), 2021.
[9] F. Kermarrec, "LiteX: Build your Hardware, easily," Available: https://github.com/enjoy-digital/litex
[10] Lukram Dhanachandra Singh, et al, “Digital ASIC Design: The Complete RTL-to-GDSII Methodology Using Open-Source EDA Tools”, HEXAGON5D TECHNOLOGIES, ISBN: 978-81-688772-4-5
[11] https://github.com/dhanalukram/OpenASIC-Design/tree/v1.0-development/examples/OPC