International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Design And Synthesis of a Low-Power Amba APB (Advanced Peripheral Bus) Protocol Using FSM Optimization, Operand Isolation and Clock Gating

SAKTHI C G

Master of Engineering in VLSI DESIGN

Dr. Sivanthi Aditanar College of Engineering Tiruchendur, India gothaisakthi10@gmail.com

Abstract **- The Advanced Peripheral Bus (APB) of the AMBA specification is widely used to connect low-bandwidth peripherals in System-on-Chip (SoC) designs, where power consumption has become a critical constraint. This paper presents the design and synthesis of a low-power AMBA APB bus protocol using a combination of Finite State Machine (FSM) optimization, operand isolation, and clock gating. The FSM is optimized through efficient state encoding to reduce switching activity and combinational logic complexity. Operand isolation prevents unnecessary toggling in data path blocks when they are idle, while clock gating disables the clock to inactive registers, lowering dynamic power dissipation. The design is described in Verilog HDL, functionally verified through simulation, and synthesized to evaluate area, timing, and power. Results show a notable reduction in dynamic power compared with the conventional APB implementation, with no loss of protocol functionality or timing compliance. The proposed approach is well suited to energy efficient SoC and IoT applications.**

Keywords - AMBA APB, low power, FSM optimization, operand isolation, clock gating, SoC.

I. INTRODUCTION

The rapid growth of portable, battery-operated, and Internet of Things (IoT) devices has made power efficiency one of the most important design goals in modern System-on-Chip (SoC) development. As integration density increases, a single chip hosts processors, memories, and numerous peripherals that communicate through standardized on-chip bus architectures. The Advanced Microcontroller Bus Architecture (AMBA), introduced by ARM, is the most widely adopted standard for this purpose. Within the AMBA family, the Advanced Peripheral Bus (APB) is specifically intended for low-bandwidth, low-complexity peripherals such as timers, UARTs, GPIO controllers, and keypads.

To address this problem, several low-power techniques have been explored at the register-transfer level (RTL). In this work, three complementary methods are combined to reduce the power consumption of the APB protocol. First, FSM optimization through efficient state encoding lowers switching activity and simplifies the next-state and output logic. Second, operand isolation blocks input changes from reaching data path components when their outputs are not needed, thereby preventing redundant toggling. Third, clock gating disables the clock to registers that do not need to update, reducing the dynamic power dissipated in the clock network and sequential elements. The proposed design is modelled in Verilog HDL, verified through simulation, and synthesized to analyses area, timing, and power.

II. LITERATURE REVIEW

Tong et al. [1] presented an AMBA-based flash controller for an IoT chip, with APB, AHB, and AXI slave interfaces, supporting SPI, Dual, Quad, and QPI modes and XIP. Configurable registers covered over 95% of mainstream flash compatibility, and system-level verification reached 100% coverage. Kalyan Naga Sai et al. [2] proposed a low-power APB using multi-level, enable-based clock gating. Synthesized in Cadence Genus, power fell from 0.682 mW to 0.567 mW (about 17%), better than an earlier RTL design at 0.90 mW. Singh et al. [3] designed an APB4 in Verilog with power gating and FIFO-based burst support. A UVM testbench achieved 100% functional and 94.7% code coverage, and Vivado synthesis showed a 66%

dynamic power reduction against APB1 with slightly reduced area.

Patra and Saini [4] verified an AHB-to-APB bridge using UVM 1.2 and System Verilog. Coverage improved from 78% to about 98%, exceeding the 96% reported in earlier work. Deepak and Nehra [5] implemented an APB master/slave interface in Verilog, verified in Questa Sim for back-to-back transfers, wait states, and PSLVERR handling, with no setup or hold violations. Rayasa et al. [6] synthesized an APB with one master and two slaves in Cadence. High effort gave the best result: 17.26% total power optimization, 34.75% switching power reduction, and 7.44% area reduction.

Unni and Abhilash [7] developed a runtime-switchable controller for AHB, AXI4-Lite, and Wishbone that defers protocol changes until the current transfer completes. On an Artix-7 it used about 25.8% of LUTs and 38.6% of flip-flops; APB support is future work. Kumar and Yadav [8] presented APB interfaces for complex SoCs, with the master microarchitecture and FSM verified by Verilog testbench waveforms, but no power or area figures. Rajalingam et al. [9] presented Nexus, a RISC-V microcontroller with a native APB3 interconnect. At 90 MHz on the PYNQ-Z2 it used 1,134 LUTs and 406 flip-flops with 18 mW dynamic power.

III. DESIGN DESCRIPTION

A. Proposed Architecture

The proposed low-power APB architecture Fig.1 consists of a bridge, an APB interconnect, and four APB slave peripherals (register bank, UART, GPIO, and I2C controller). In the bridge, the system bus (DMA controller) request is first captured by the address/control latch and request decoder, then passed to an optimized three-state FSM (IDLE, SETUP, ACCESS) that drives PSEL and PENABLE. The FSM output also controls a latch-based clock gating unit and an operand isolation block, both coordinated by the power control logic to stop clock and data toggling in idle slaves. The APB interconnect routes the address decoder's select signals to the slaves and returns read data and response signals to the bridge through the PRDATA MUX and the PREADY/PSLVERR MUX, with a test/status block reporting errors for debug.

B. Block Diagram

The Fig 1 illustrates a low-power APB master/bridge that takes AHB/AXI requests, latches and decodes them, and runs a Gray-coded three-state FSM (IDLE, SETUP, ACCESS) that drives PSEL and PENABLE to the APB interconnect and its slave peripherals.

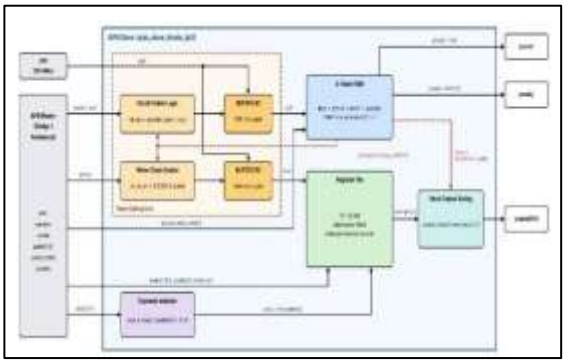


Fig 1: Block Diagram of Low-Power APB

The power control logic generates EN_CLK and ISO_EN from the FSM state, so the latch-based clock gating unit and the operand isolation block stop the clock and force PADDR, PWDATA, and PWRITE to 0 when idle, eliminating dynamic switching.

C. Flow Chart

The flowchart Fig.2 shows the operation of a low-power AMBA APB slave with a 16 × 32-bit register file, implemented using clock gating and operand isolation. The design takes pclk, presetn, psel, penable, pwrite, paddr[7:0] and pwdata[31:0] as inputs. A clock enable signal, computed as clk_en = ~presetn | psel | busy, controls the first clock buffer (BUFGCE #1): when it is low, the FSM clock is blocked and the slave stays in a low-power state, and when it is high, pclk is passed on as gclk. In the IDLE state, busy is cleared and the internal address addr_i is held at zero until psel is asserted, after which the FSM moves to the SETUP state and sets busy to keep the clock running. If paddr[7] is high, a WAIT state models slow-peripheral behaviour before the FSM enters the ACCESS state, where pready is asserted and pslverr is held low. From there, the transfer branches on pwrite. For a write, a write-clock enable (wr_clk_en = ACCESS & pwrite) drives a second clock buffer (BUFGCE #2) that generates wclk, so pwdata is stored in mem[paddr[3:0]] only during write cycles. For a read, rd_en = ACCESS & ~pwrite enables the read path, where operand isolation sets addr_i = busy? paddr[3:0] : 4'h0 to prevent unnecessary switching,

and output gating sets prdata = rd_en ? mem[addr_i] : 0 so the read bus stays quiet when no read occurs. Once the transfer is complete, the FSM returns to the IDLE state, ready for the next transaction.

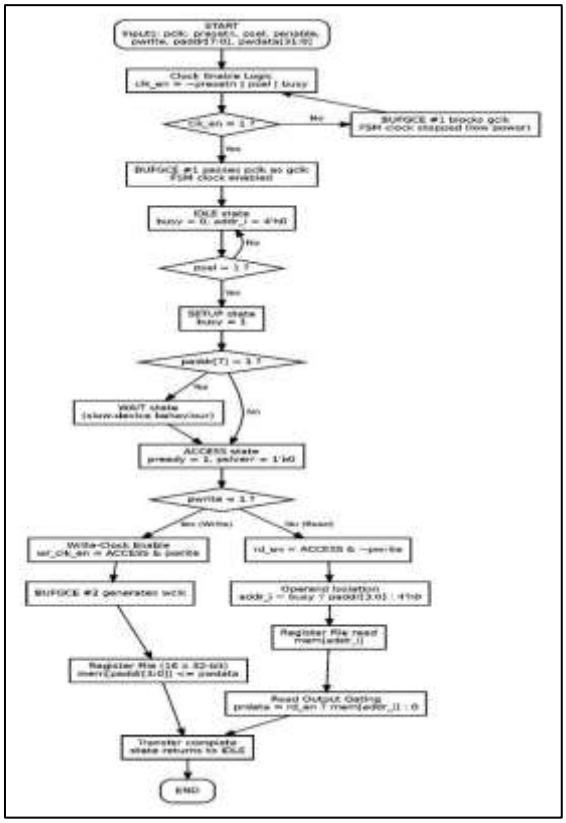


Fig 2: Flow-Chart of Low-Power Architecture

On a request, the address and control signals (HADDR, HWRITE, HWDATA, HTRANS) are latched and decoded. The power control logic then enables the gated clock (EN_CLK = 1) and releases operand isolation (ISO_EN = 1), and the FSM moves through SETUP, where the address decoder asserts the one-hot PSELx, to ACCESS, where PENABLE is asserted. The ACCESS state waits until the slave drives PREADY high, after which the PRDATA mux returns the read data and the PSLVERR status to the master. If another transfer is pending, the FSM goes back to SETUP. Otherwise, it returns to IDLE, stopping the clock and isolating the inputs so that dynamic switching drops to zero.

D. Low-Power Techniques

Three complementary RTL techniques shown in Fig.3 are applied to reduce dynamic power in the proposed APB design. FSM optimization uses Gray-coded state encoding for the IDLE, SETUP, and ACCESS states, so only one-bit changes per transition, which lowers switching activity and simplifies the next-state logic. Operand isolation places AND-gate banks on PADDR, PWDATA, and PWRITE, holding them at 0 when ISO_EN is low, so the data path does not toggle while the bus is idle.

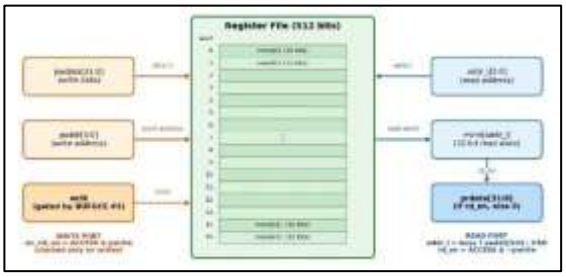


Fig 3: Register file architecture of the low-power APB slave

The Fig 3 shows a 512-bit register file organized as 16 words of 32 bits each (mem [0] to mem [15]). On the right side, pwdata[31:0] carries the data and paddr[3:0] gives the word address, while the write clock wclk is gated by BUFGCE #2 so it only toggles when wr_clk_en = ACCESS & pwrite, which keeps the memory clock idle during non-write cycles and saves dynamic power. On the read side, addr_i[3:0] selects a word and mem[addr_i] is passed out as prdata[31:0] only when rd_en = ACCESS & ~pwrite, otherwise the output is 0.

Clock gating uses a latch based ICG (gclk = PCLK & Unlatched), where EN_CLK is asserted only when the FSM leaves IDLE or a transfer request arrives, so the clock stops during idle periods. Together, these techniques remove unnecessary toggling in the FSM, data path, and clock network, giving zero dynamic switching while the bus is idle.

IV. METHODOLOGY & IMPLEMENTATION

A. Methodology

The proposed design Fig 4. is described in Verilog HDL and developed in four steps. First, the baseline APB master, slave, and interconnect are modelled using the standard IDLE, SETUP, and ACCESS FSM. Second, the FSM is optimized with Gray-coded state encoding, operand isolation is added on PADDR, PWDATA, and PWRITE, and a latch-based clock gating unit is inserted, all controlled by the power control logic through

EN_CLK and ISO_EN. Third, functional verification is carried out with a testbench covering read and write transfers, wait states, and PSLVERR handling, to confirm that the protocol remains correct after the low-power changes. Finally, the baseline and optimized designs are synthesized with the same constraints and compared on area, timing, and power. This shows the combined power saving from FSM optimization, operand isolation, and clock gating.

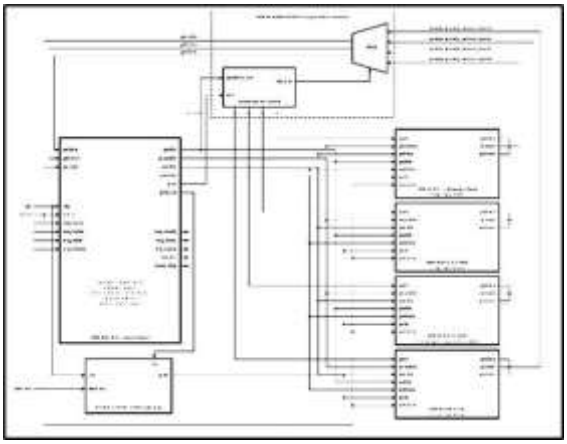


Fig 4: Methodology of APB Low-Power Technique

B. Implementation

The Fig 4 illustrates proposed low-power AMBA APB protocol was implemented in Verilog HDL with one master and two slaves and verified and synthesized in Xilinx Vivado 2025.1 for an Artix-7 device. The baseline master follows the reference design with a seven-state, binary-encoded FSM. Three techniques were then applied. First, FSM optimization reduced the master to three Gray-coded states (IDLE, SETUP, ACCESS), removing the redundant wait states and decoding PSEL and PENABLE directly from the state bits. Second, operand isolation forces the address, write data and read data of each slave to zero unless that slave is selected, which stops bus activity from reaching the memory array. Third, clock gating uses BUFGCE cells to enable the master and slave clocks only during active transfers. Each technique is controlled by a parameter, so five configurations (BASE, FSM, ISO, CG, ALL) could be compared. A self-checking testbench validated all configurations against a reference memory, and power was estimated from SAIF switching activity captured in post-synthesis simulation. The proposed design was compared with the baseline on post-implementation power and utilization. The four-slave design uses 186 slice LUTs, 560 slice registers and 296 slices, which is under 4% of the device. Its total on-chip power is 0.121 W, only 6.1% above the 0.114 W of the single-slave baseline, and its dynamic power per slave drops from 0.044 W to about 0.013 W. The testbench completed 347 checks with zero errors, and clock gating suppressed about 31.9% of the clock edges.



Fig 5a: Functional simulation results of the APB bridge and four slaves showing write and read transfers.

The Fig 5a illustrates Vivado simulation waveform window (Untitled 2) showing the behavior of the APB slave design over time. The signal names on the left are cut off and the image is very small, so I can only describe it in general terms. The top signals look like control signals such as the clock, reset, and APB handshake lines (psel, penable, pwrite), which toggle in regular pulses as transfers occur. Below them are wide buses, shown as pink and cyan hex value blocks, that appear to carry the address, write data, and read data, and these change at each transfer, with values like 00, 01, 02, 03 and so on visible on the address-like bus. Early in the simulation some buses show X (unknown) values, which is normal before reset and initialization, after which they settle into defined values as write and read transactions begin. Overall, the waveform shows a sequence of APB write and read operations with the data and address buses updating in step with the control signals.



Fig. 5b: Testbench log of the proposed low-power APB design reporting zero errors and the clock-gating savings.

The Fig 5b illustrates a Vivado behavioral simulation session for the project “updatedfour”,

with the Tcl Console open and showing the log output from launching the simulation. The console messages indicate that Vivado compiled and elaborated the design, loaded the simulation snapshot for the testbench (it looks like `tb_apb_slave_4state_lp2` or similar), and then began running it. Near the end of the log there are a few lines that look like simulation output from the testbench, such as clock-gating or read/write check messages, followed by what appears to be a `$finish` or end-of-simulation line with a time stamp, which means the simulation ran to completion.



Fig 5c: Power Summary of the Proposed APB Design Showing Total On-Chip Power of 0.121 W (dynamic 0.051 W, static 0.070 W).

The Fig 5c report shows a total on-chip power of about 0.121 W, with the design power budget "Not Specified", the process set to typical, and a junction temperature of around 25.4°C. The On-Chip Power chart splits the total into roughly 42% dynamic power (about 0.051 W) and 58% device static power (about 0.070 W). The dynamic portion is broken down further into clocks, signals, logic and I/O, with signals and logic appearing to take the largest shares.



Fig. 5d: Utilization Report

The Fig 5d shows the Vivado Implemented Design view for the project with the Utilization report open in Hierarchy view and the status bar reading "Implementation Complete". The report lists one top-level module, `apb_slave_4state_lp2`, which uses 44 Slice LUTs out of 20,800 available, 2 Slice Registers out of 41,600, and 17 Slices out of 8,150. Of the 44 LUTs, 22 are used as logic and 22 as memory, which indicates that the 16-word register file is mapped onto distributed RAM instead of flip-flops, and this is why the register count stays so low. The design also uses 76 of the 106 bonded IOBs, because the wide address, data and control ports are routed to physical pins, and 2 of the 32 BUFGCTRL clock buffers, which is consistent with the BUFGCE-based clock gating in the architecture. Overall, the design occupies well under 1% of the LUT, register and slice resources of the device, which shows that the 4-state implementation is very compact and resource-efficient, with the I/O count being the largest share of utilization.

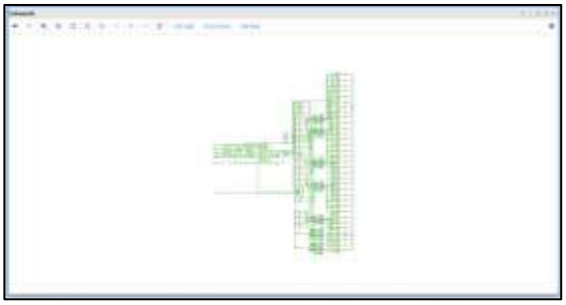


Fig 5e: Implemented Netlist Schematic of the APB Design with Low-Power Techniques.

The Fig 5e illustrates is a Vivado schematic view of the implemented design, showing the gate-level and primitive-level netlist that the tools generated from your RTL. It is a technology-mapped schematic, so the logic is drawn using FPGA primitives such as LUTs, multiplexers, flip-flops, and buffers rather than the original behavioral code. On the left are the input ports (such as the clock, reset and APB control signals), which feed a network of small logic cells that look like LUTs, and gates handling the state decoding and enable generation. In the center there are several multiplexer-style cells with wide connections, which likely select between data paths, and a highlighted block near the upper middle that appears to be a register or buffer element, possibly part of the clock-gating path. At the bottom center there are two taller blocks with many pins, which look like the distributed RAM primitives that store the 16-word register file, consistent with the 22 LUTs as memory in your utilization report. The right side shows the output logic leading to the output ports, including the read data path.

TABLE I. Post-Implementation Power Comparison

Parameter (W)	*APB Architecture*	*Low-Power APB with Optimization Blocks*
Total on-chip power	0.113 W	0.071 W
Dynamic power	0.043 W	0.001 W
Device static power	0.070 W	0.070 W

The Table 1 compares the power consumption of a standard APB architecture against a low-power APB design that includes optimization blocks such as clock gating. The standard APB architecture consumes a total on-chip power of 0.113 W, whereas the low-power version reduces this to 0.071 W, which is a saving of 0.042 W, or roughly 37%. The entire reduction comes from dynamic power, which drops from 0.043 W to just 0.001 W, a reduction of about 98%. This is the expected result of the optimization blocks, since techniques like clock gating (BUFGCE) keep the clock and the register file idle except during real transfers, which removes unnecessary switching activity in the clocks, signals, logic and I/O. Device static power stays the same at 0.070 W in both designs because it is determined by the silicon of the FPGA (xc7a35t) and the operating conditions rather than by the design's activity, so these optimizations cannot reduce it. As a result, static power makes up almost all of the remaining power in the low-power design (about 98% of the total),

TABLE II. Post-Implementation Resource Utilization

Resource (available)	*APB without Clock Gating (1 SLAVE)*	*APB with Low-Power Techniques (4 SLAVES)*
Slice LUTs (20800)	56	44
Slice Registers (41600)	3	2
Slices (8150)	19	17

The Table 2 compares the FPGA resource utilization of a basic APB design without clock gating (one slave) against the low-power APB design (four slaves) on the xc7a35t device. The baseline design uses 56 Slice LUTs out of 20,800, 3 Slice Registers out of 41,600, and 19 Slices out of 8,150, while the low-power design uses 44 LUTs, 2 registers, and 17 slices. This means the low-power version needs 12 fewer LUTs (about 21% less), 1 fewer register (about 33% less), and 2 fewer slices (about 11% less). What makes this result notable is that the low-power design supports four slaves, yet it still uses fewer resources than the single-slave baseline, which suggests that the optimized architecture, with its simplified 4-state control logic and a shared register file mapped onto distributed RAM, is more compact than the original 7-state design. The saving in registers is small in absolute terms because both designs use only a handful of flip-flops, so the register count is not a strong differentiator. In all cases the utilization is well below 1% of the available device resources, so the area overhead of the low-power techniques is effectively zero, and in fact the optimization reduces area while also cutting dynamic power. Together with the power table, this shows that the proposed design achieves lower power without any penalty in resource usage.

V. CONCLUSION

The low-power AMBA APB protocol was designed in Verilog and verified in Vivado, with all five configurations (BASE, FSM, ISO, CG, ALL) producing identical, correct read and write behaviour. FSM optimization reduced the master from seven binary-encoded states to three Gray-coded states, lowering logic and state-transition switching. Operand isolation and clock gating cut unnecessary toggling on the bus and idle blocks, reducing dynamic power compared with the baseline, with the largest savings when the bus is mostly idle. The combined configuration (ALL) gave the best power result, and the savings are reported from SAIF-based activity, so they reflect real switching behaviour (insert your measured percentages here). Future scope includes adding PSLVERR error handling, extending the design to more slaves and an AHB-to-APB bridge, evaluating it in an ASIC flow with latch-based clock gating, and adding assertion-based verification.

VI. FUTURE SCOPE

Future work can extend the design to more slaves and to higher-performance AMBA protocols such as AHB and AXI, with adaptive or fine-grained clock gating driven by real-time bus activity. Power gating and dynamic voltage and frequency scaling

(DVFS) could be added to cut static power as well. Power can then be re-evaluated with SAIF-based activity data after fixing timing closure.

REFERENCES

[1] AMD (Xilinx), "Vivado Design Suite User Guide: Power Analysis and Optimization," UG907,

[2] ARM Ltd., "AMBA APB Protocol Specification, Version 2.0," ARM IHI 0024C, 2010.

[3] Dwivedi, N. Mishra, and A. Singh-Rajput, "Assertion & Functional Coverage Driven Verification of AMBA Advance Peripheral Bus Protocol Using System Verilog," 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), February 2021, DOI: 10.1109/ICAECT49130.2021.9392518.

[4] Gupta, K. Rawat, S. Pandey, P. Kumar, S. Kumar, and H. P. Singh, "Physical design implementation of 32-bit AMBA ASB APB module with improved performance," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 3121–3124, DOI: 10.1109/ICEEOT.2016.7755276.

[5] Jain and S. Rao, "Design and Verification of Advanced Microcontroller Bus Architecture-Advanced Peripheral Bus (AMBA-APB) Protocol," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), February 2021, DOI: 10.1109/ICICV50876.2021.9388549.

[6] Ma, Z. Liu, and X. Ma, "Design and implementation of APB bridge based on AMBA 4.0," 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet), 2011, pp. 193–196, DOI: 10.1109/CECNET.2011.5768692.

[7] Manu and P. Prabhavathi, "Design and implementation of AMBA ASB APB bridge," 2013 International Conference on Fuzzy Theory and Its Applications (iFUZZY), 2013, pp. 234–238, DOI: 10.1109/iFuzzy.2013.6825442.

[8] Münch, N. Wehn, B. Wurth, R. Mehra, and J. Sproch, "Automating RT-level operand isolation to minimize power consumption in datapaths," Proc. Design, Automation and Test in Europe Conference (DATE), March 2000, pp. 624–631.

[9] Ravikumar, K. B. Ganesha, C. Y. Chethana, and T. M. Parikshith Roy, "Design and Verification of Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) Protocol," 2024 International Conference on Smart Systems for Applications in Electrical Sciences (ICSSES), May 2024, DOI: 10.1109/ICSSES62373.2024.10561369.

[10] Rawat, K. Sahni, and S. Pandey, "RTL implementation for AMBA ASB APB protocol at system on chip level," 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 2015, pp. 927–930, DOI: 10.1109/SPIN.2015.7095347.

[11] Roopa, R. M. Vani, and P. V. Hunagund, "UART controller as AMBA APB slave," National Conference on Challenges in Research & Technology in the Coming Decades (CRT 2013), 2013, DOI: 10.1049/cp.2013.2507.

[12] Shanthi, K. G. Sravani, S. S. Vali, et al., "Design and FPGA implementation of AHB-to-APB bridge," Microsyst. Technol., 2024, DOI: 10.1007/s00542-024-05703-1.

[13] Weste and D. M. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Boston, MA, USA: Addison-Wesley, 2011.

[14] Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol. 47, no. 3, pp. 415–420, March 2000.

[15] Yadav, P. Satyanarayana, B. V. Krishna, K. R. K. Sai, M. Arun, and V. Gokula Krishnan, "Design and Implementation of AMBA-APB Protocol for System-on-Chip (SoC) Designs," 2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), April 2024, DOI: 10.1109/ICDCECE60827.2024.10549635.