STM JOURNALS

Journal of
VLSI Design Tools & Technology

ISSN: 2249-474X (Online)
ISSN: 2321-6492 (Print)
Volume 15, Issue 1, 2025
January–April
DOI (Journal): 10.37591/JoVDTT

https://journals.stmjournals.com/jovdtt

Review JoVDTT

RTL to GDSII Implementation UART Using Verilog and Cadence Tools

Dhruv Lalpurwala[1,*], Vandan Parekh[1], Robinson Paul[2]

Abstract

The study outlines a thorough design flow for Universal Asynchronous Receiver and Transmitter (UART) utilizing Register Transfer Level (RTL) to Graphic Data System II (GDS II) Implementation using Verilog and Cadence tools. The process involves RTL design, logic synthesis with Genus, and physical design using Innovus, followed by verification steps including formal verification, static timing analysis, and power optimization. The results demonstrate an efficient workflow, highlighting the effectiveness of Cadence's tools in achieving a reliable and manufacturable UART design. Communicating UARTs have no shared timing system apart from the communication signal. Typically, UARTs resynchronize their internal clocks on each change of the data line that is not considered a spurious pulse. Obtaining timing information in this manner, they reliably receive when the transmitter is sending at a slightly different speed than it should. Simplistic UARTs do not do this; instead, they resynchronize on the falling edge of the start bit only and then read the center of each expected data bit, and this system works if the broadcast data rate is accurate enough to allow the stop bits to be sampled reliably.

Keywords: UART, Cadence, Verilog, transmitter, HPC, HDL, FMSD

INTRODUCTION

With the increase in Internet of Things (IoT) applications and high-performance computing (HPC), it has become crucial to have a communication protocol to meet these demands. A reliable, reusable, plug-and-play solution can help with a sharp increase in data communication. In real-time systems, the communication protocol is the central stage for peripherals to interact with the processor, so the propagation delay would be as low as possible. If communication is unreliable, the real-time system is essentially a failure. UART is often used as an Integrated Circuit (IC) for serial communication through a computer or a peripheral device.

The first step of the design flow is the Register Transfer Level (RTL) design, where the high-level architectural description is transformed into a digital representation using hardware description languages (HDLs), such as Verilog. This involves the design and implementation of the behavior and connections of individual digital components. After the RTL, functional verification was conducted to ensure that the design behaved as intended. Techniques such as hardware emulation, formal verification, and simulation are crucial for validating design correctness and compliance with functional requirements. The gate-level representation was transformed into a physically feasible layout during the physical design stage. Tasks such as floor planning, placement, clock tree synthesis, routing, and optimization are performed to meet power, performance, and area requirements.

***Author for Correspondence**
Dhruv Lalpurwala
E-mail: lalpurwaladhruv@gmail.com

[1]Student, Department of Electronics and Communication Engineering, Birla Vishvakarma Mahavidhyalaya, Anand, Gujarat, India
[2]Assistant Professor, Department of Electronics and Communication Engineering, Birla Vishvakarma Mahavidhyalaya, Anand, Gujarat, India

Received Date: December 28, 2024
Accepted Date: January 02, 2025
Published Date: January 09, 2025

Citation: Dhruv Lalpurwala, Vandan Parekh, Robinson Paul. RTL to GDSII Implementation UART Using Verilog and Cadence Tools. Journal of VLSI Design Tools & Technology. 2025; 15(1): 1–7p.

Once the final layout database is created, containing all the necessary design data, it is ready for manufacturing after a thorough review and satisfies all relevant requirements. The manufacturing process involved sending this database to a semiconductor foundry for fabrication.

The design of UART begins with RTL design using Cadence toolsets, where the high-level architecture is described using HDLs such as Verilog. Cadence tools, such as Genus, are used to synthesize the RTL into a gate-level netlist, ensuring optimal performance and area. After RTL synthesis, the physical design is facilitated using Cadence Innovus, focusing on tasks such as partitioning, placement, clock tree synthesis, routing, and timing optimization. The design was divided into manageable blocks with precise placement for power and signal routing. Static timing analysis (STA) is performed to validate the timing, while layout vs. schematic (LVS) and Design Rule Checks (DRC) ensure that the layout matches the intended functionality and manufacturing guidelines. Cadence tools ensure a smooth flow from the RTL to the final physical design, leading to efficient chip manufacturing.

RELATED WORK

The UART protocol is a widely used serial communication protocol that allows data transfer between devices without the need for a clock signal [1]. It operates by converting parallel data into a serial format for transmission and then converting the received serial data back into a parallel format. The architecture of a UART typically includes modules such as a baud rate generator, receiver, transmitter, and First-in-first-out (FIFO) buffers. The baud rate generator sets the communication speed between devices, whereas the FIFO buffers ensure smooth data transmission and reception, avoiding data loss [2].

UARTs are commonly integrated into microcontrollers and are used as standalone ICs for serial communication. They are ideal for connecting devices, such as computers and peripherals via serial ports [2]. Full-duplex UART allows simultaneous transmission and reception of data. Additionally, the design process for UARTs includes managing data flow, interrupt handling, and ensuring a proper data format through control registers, such as FIFO. The UART design process ensures efficient serial communication, making it a crucial component of embedded systems and communication interfaces. Currently, UARTs are embedded in various microcontrollers, providing a reliable and flexible communication method for both consumer and industrial applications [3].

UART DESIGN IMPLEMENTATION

Figure 1 shows a block diagram of UART. The UART design has a transmission FIFO, receiver FIFO, baud rate generator, and control unit.

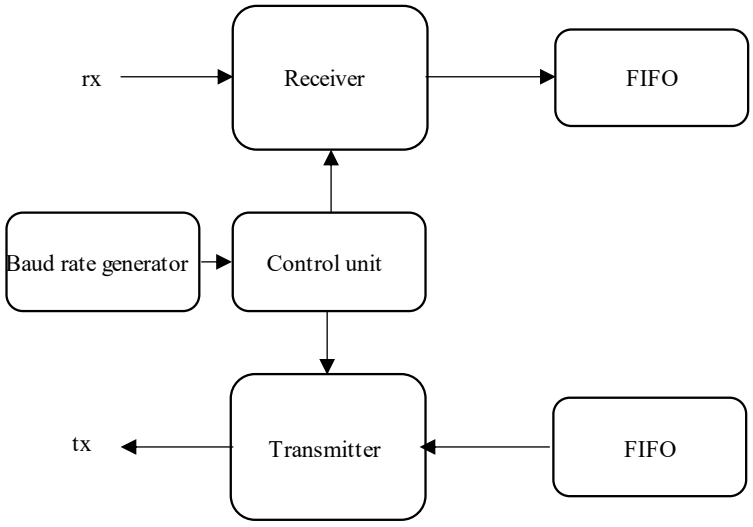

Figure 1. UART design block diagram.

The baud rate generator produces various baud rates for the transmission. It then provides this information to the control unit, producing read, write, and other control signals. A BRG is a flexible and programmable unit that produces a baud clock. FIFO helps transmit and receive data serially. The control unit helps determine when to read or write data [3, 4].

Figure 2 shows the transmitter block diagram. The transmitter module takes input clock, reset, din, and tx_start to transmit, and tx, tx_done_tick as outputs. tx_start signals, the finite state machine with datapath (FSMD) loads the data word and then gradually progresses through the start, data, and stop states to shift out the corresponding bits [5]. It signals completion by asserting a tx_done_tick signal for one clock cycle. It uses an internal shift register to transmit data, a counter that counts the number of bits to be transmitted, and a register to maintain the state machine information. It uses the FIFO buffer to temporarily store multiple data packets for sequential transmission, allowing the transmitter to efficiently handle bursts of data.

The state flow of the transmitter has five states: Initial default low energy (IDLE), START, DATA, FSM and STOP. By default, it starts in the IDLE state before the tx_start signal is switched ON [6]. It then goes to the START state. Here, the transmitter adds the start bit to the shift register and shifts the remaining data [7]. It transitions into a DATA state. Here, the transmitter shifts the data left until all eight bits have been transmitted, and the Field service management (FSM) enters the last stage, the STOP bit. The data frame is given as an output from the pin tx. During all operations, the done tick signal is set to 1 to indicate that the next data frame has to wait until the current operations are finished [8].

Figure 3 shows the UART receiver. The receiver module uses three inputs and produces two outputs. The inputs were clock, reset, and Rx. The outputs were dout and rx_done_tick. The reset signal shuts off all circuit operations. The clock signal is used to transition from one state to another. It has various internal registers, such as shift register, state register, bit-count register, and received data register. A shift register is used to shift the data frame from the transmitter.

The state register contains information regarding the three states of the receiver. The bit-count register was used as a base for the counter to pass through the state frame. The received data register stores the data from the transmitter. The dout signal indicates whether data are received and stored in the internal register. The rx_done_tick signal indicates whether the receiver is performing, that is, if the operations are ongoing, it cannot accept the next set of data frames [9].

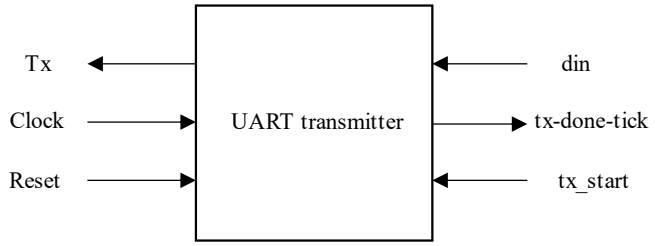

Figure 2. Transmitter block diagram.

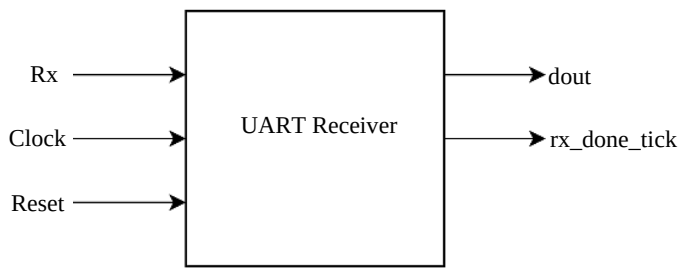

Figure 3. Receiver block diagram.

The finite state machine of the receiver has two states, namely the IDLE state and the SHIFT-DATA state. The state machine begins by waiting for the start bit to be received in the IDLE state [10]. The state machine switches to the SHIFT-DATA state after detecting the start bit, where it inserts incoming data bits one at a time until all data bits are received.

RTL TO GDSII FLOW

Figure 4 shows the complete Application specific integrated circuit (ASIC) design process, commonly known as RTL to GDSII, which is implemented using tools such as Cadence. The process begins with Design Specification and RTL coding, where designers write the RTL code in Verilog or VHSIC hardware description language based on the system specifications. Tools such as Cadence Genus are used to synthesize this RTL into a gate-level netlist. Following this, Design Simulation and Verification were performed using Cadence Xcelium, ensuring the functionality of the RTL design through comprehensive simulations.

Code coverage was then performed using Cadence Incisive Coverage to check how well the test cases exercised the RTL code, ensuring that all parts of the design were thoroughly tested. In the synthesis stage, the RTL code is converted into a gate-level representation using technology libraries, with Cadence Genus handling this.

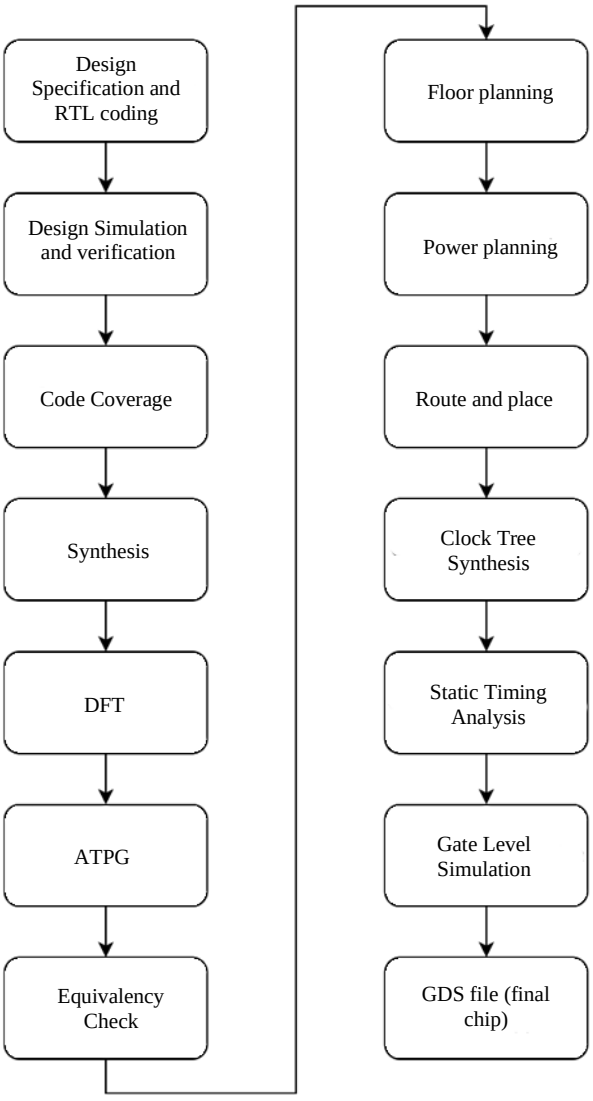

Figure 4. Flowchart of the proposed algorithm.

To make the design testable, Design for Testability (DFT) is added, allowing for easier post-manufacturing testing, and tools such as the Cadence Modus assist in this step. Next, Automatic Test Pattern Generation (ATPG) was performed to create test vectors that can detect manufacturing defects [4].

An equivalence check follows, ensuring that the synthesized design is functionally equivalent to the original RTL, a task performed by Cadence Conformal. The physical design process starts with floor planning, where the layout of the major blocks is decided using Cadence Innovus. After this, Power Planning is conducted to design the power distribution network, ensuring that the chip receives proper power, assisted by Cadence Voltus.

Once the design's floor plan and power grid are established, Place and Route are performed using Cadence Innovus, where the standard cells are placed and connected. The next step is clock tree synthesis, which ensures the distribution of the clock signal to various parts of the chip by using Cadence Innovus. STA was then conducted with Cadence Tempus to verify that the design meets all timing constraints.

In the final stages, a gate-level simulation was conducted using Cadence Xcelium to verify the behavior of the synthesized netlist [4]. Finally, the design is converted into a GDSII format using Cadence Innovus, which is the standard file used for chip fabrication.

RTL DESIGN IMPLEMENTATION

RTL design implementation using Cadence tools involves a structured approach that encompasses the design, simulation, synthesis, and verification of digital circuits. Initially, a specification is defined for the intended design, such as an arithmetic logic unit (ALU) or a simple processor. The design is coded in an HDL such as Verilog or VHDL, followed by simulation using Cadence tools such as Xcelium to verify functionality through testbenches. Once the design is validated, synthesis tools, such as Genus, convert the RTL code into a gate-level netlist, optimizing the area and timing based on set constraints.

PHYSICAL DESIGN IMPLEMENTATION

We utilized Cadence tools for RTL to GDSII flow in our backend implementation. The process begins with design synthesis, where inputs, such as HDL files (.v, .sv, .vhd), libraries, and constraints (.sdc) were provided to generate a netlist along with the area and timing reports. The synthesized output is then input into the partitioning stage alongside physical libraries (lef files) to define the design portions. This output is subsequently fed into the floor planning tool, which generates a. def file, which serves as the layout blueprint for physical design.

During the placement stage, the cells are strategically positioned on the floor plan, and the clock tree is synthesized using Resistor-Capacitor (RC) delay models to ensure optimal performance. The timing analysis tool performs in-depth timing analysis of the design and generates detailed timing reports. Cadence offers a comprehensive suite of tools for each stage: Genus for synthesis, Innovus for floor planning and placement, and tempo for clock tree synthesis. Routing was accomplished using Innovus, and verification was conducted with tools such as Cadence Virtuoso for layout checks and Cadence Pegasus for DRC/LVS checks, ultimately generating GDSII files for fabrication. The flow also produces extensive logs and reports, allowing performance analysis through metrics such as slack values from timing reports, utilization statistics from placement reports, and congestion metrics from the routing stage. These performance indicators are crucial for evaluating the effectiveness of the design and optimizing overall performance.

EXPERIMENTAL RESULTS

Figure 5 depicts the functional verification of the UART module, focusing on key signals involved in data transmission and FIFO buffer management. The primary signals under observation include the clock (clk), transmitted data (w_data[7:0]), received data (r_data[7:0]), transmitted full flag (tx_full),

and received empty flag (rx_empty). The FIFI_W signals represent the write operation to the FIFO buffer, whereas the SB_TICK signal indicates stop bit timing for proper data framing. The interaction between the transmitter and receiver sections of the UART, particularly the dynamic behavior of tx_full and rx_empty, highlights the efficiency of the buffering mechanism. This simulation aids in verifying the integrity of asynchronous serial communication, thereby ensuring that it is critical for reliable data transmission in embedded systems.

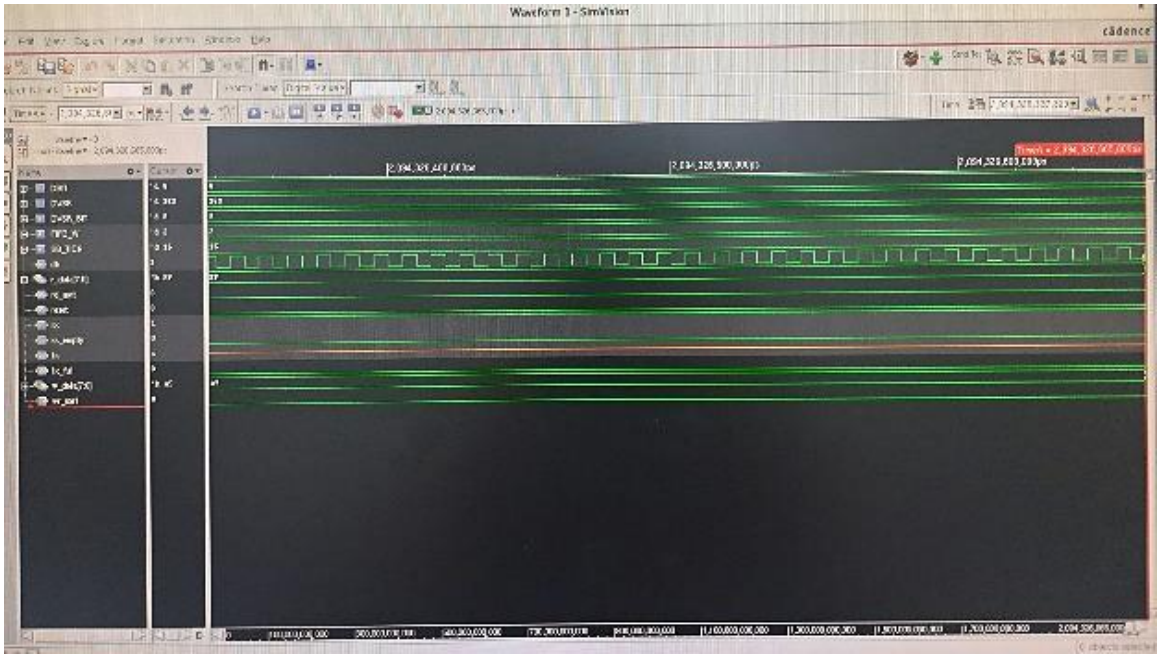

Figure 5. Functional verification.

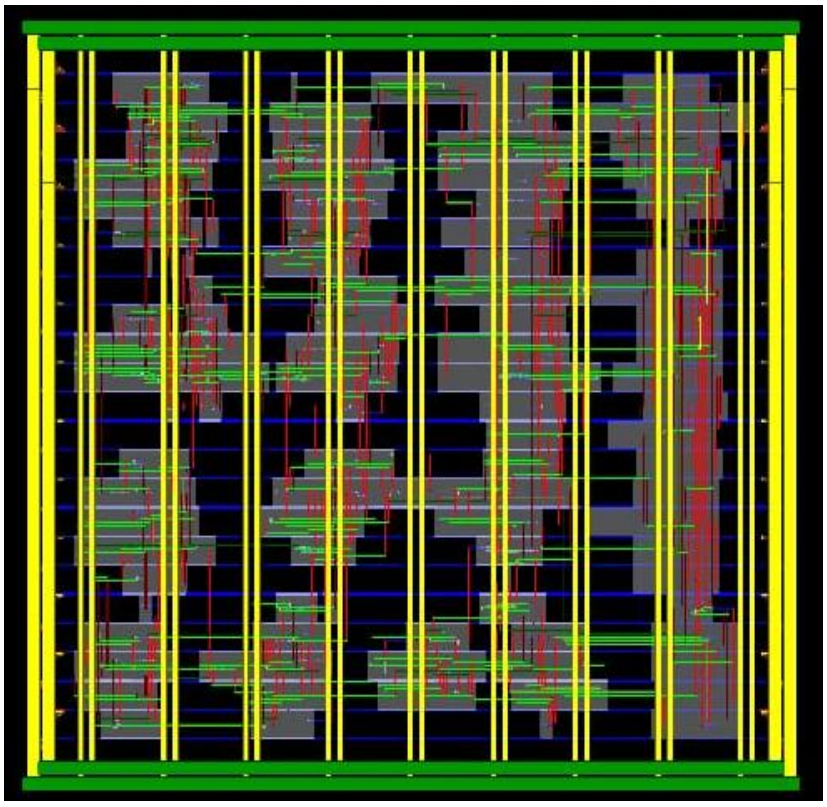

Figure 6. Physical layout.

Figure 6 depicts the physical layout of a Universal Asynchronous Receiver-Transmitter (UART) module implemented using the Cadence tool in the backend design process. The layout shows key design elements, such as vertical power rails (yellow) for distributing power, standard cell rows (gray and red) where logic gates and flip-flops are placed, and multiple routing layers (green and blue) connecting the cells according to the netlist [7]. The layout also includes an optimized clock tree in the netlist. The layout also includes an optimized clock tree to minimize clock skew and ensure synchronous operation. The outer green boundary defines the total area of the chip, and the design is optimized for timing, power, and area constraints. This layout, which is a key step in the RTL to GDSII flow, is now ready for fabrication.

CONCLUSION

This study describes the complete RTL to GDSII implementation of the UART using Verilog and Cadence tools, which successfully demonstrated a streamlined design flow from high-level RTL descriptions to physical layouts. The final design satisfied the desired performance, area, and timing constraints, validating the efficiency of the Cadence suite in optimizing complex digital circuits for real-world applications.

REFERENCES

1. Govil A, Karnwal A, Sindhu G, Singh A, Shukla DS. Design and implementation of UART using FPGA board. Int J Res Appl Sci Eng Technol. 2022;10:1187–90. doi:10.22214/ijraset.2022.41478.
2. Kavyashree S. Design and implementation of UART using Verilog. Int J Eng Comput Sci. 2015;4(12):15240–5.
3. Kumar S, Singh A. Universal asynchronous receiver and transmitter implementation on 40nm technology. Int J Electron Commun Comput Eng. 2018;9:54–8.
4. Bhatt P, Joshi D, Jadhav S. RTL to GDS implementation and verification of UART using UVM and OpenROAD. 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA. 2024. p. 713–20. doi:10.1109/CCWC60891.2024.10427771.
5. Poonam R, Kedia NN, Mandaogade SRG. FPGA implementation of UART controller with automatic baud rate generator. Int J Eng Sci Res. 2015;3:106–13.
6. Dasthagiraiah N, Subramanyam E, Supraja HKPP, Goutham KV. Design and implementation of UART on SOC. Int J Res Dev Technol. 2014;2:7–12.
7. Agrawal RK, Mishra VR. The design of high speed UART. 2013 IEEE Conference on Information & Communication Technologies, Thuckalay, India. 2013. p. 388–90. doi:10.1109/CICT.2013.6558126.
8. Mahat NF. Design of a 9-bit UART module based on Verilog HDL. 2012 10th IEEE International Conference on Semiconductor Electronics (ICSE), Kuala Lumpur, Malaysia. 2012. p. 570–3. doi:10.1109/SMElec.2012.6417210.
9. Ali Z, Wadhvani M. Design and simulation of UART for serial communication. Int J Comput Sci Eng. 2016;5:151–5.
10. Priyanka B, Gokul M, Nigitha A, Poomica JT. Design of UART using Verilog and verifying using UVM. 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India. 2021. p. 1270–3. doi:10.1109/ICACCS51430.2021.9441789.

