# Synopsys Vietnam

# DIGITAL TEAM ASSIGNMENT REPORT

# UART DESIGN

NGO DUC DUNG

August 9, 2024

## 1. Introduction

This report details the design and testing of a UART (Universal Asynchronous Receiver/Transmitter) block based on the given specifications. The objective is to develop a UART block that meets the provided requirements and to perform testing to ensure correct functionality

## 2. Specifications

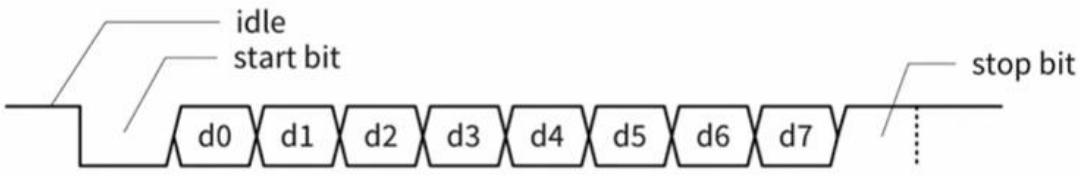
- Clock: 100MHz
- Baud Rate: 9600bps
- Bit: 8
- Priority bit: No
- Flow Control (CTS, RTS): No

## 3. Design

The UART (Universal Asynchronous Receiver/Transmitter) block is designed to facilitate serial communication by converting parallel data into a serial format for transmission and converting incoming serial data back into parallel format for reception. This design includes several key components: a **Baud Rate Generator**, a **Transmitter**, a **Receiver**, and **FIFO buffers**.

### 3.1. Technical Specifications

- Clock: 100MHz
- Baud Rate: 9600bps
- Data Bits: 8 bits
- Start Bit: 1 bit
- Stop Bit: 1 bit
- Parity Bit: 1 bit




***Frame structure of UART protocol***

### 3.2. Interface

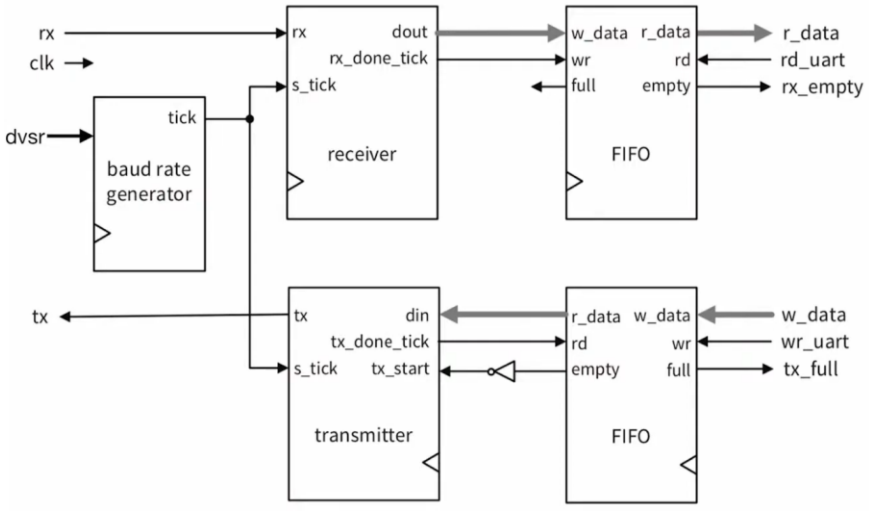
| Port name | Direction | Function |
|---|---|---|
| clk | Input | Clock signal |
| reset | Input | Asynchronous reset. Active low.<br>(0 means reset all registers inside the design) |
| rd_uart | Input | Read command enable. Active high.<br>Enable the block to read data |
| wr_uart | Input | Write command enable. Active high.<br>Enable the block to transfer data |
| rx | Input | Data received from UART |
| w_data | Input | Data to write |
| dsvr | Input | Divides a high-frequency clock to produce a lower frequency output. |
| tx_full | Output | Assert when the write command is finished. |
| x_empty | Output | Indicates that the block has data to read. |
| tx | Output | Data to transfer on UART |
| r_data | Output | Data to read |

***Interface Ports***

### 3.3. Block Diagram

The block diagram consists of the following main components:

- **Baud Rate Generator**: Generates the clock signal for the desired baud rate from the system clock.
- **Transmitte**r: Serializes and transmits data from the FIFO.
- **Receiver**: Deserializes incoming data and stores it in the FIFO.
- **FIFO (First-In, First-Out)**: Buffers data for both transmission and reception.




***Block diagram***

### 3.4. Function

The UART block is designed for serial communication, converting parallel data into serial format for transmission and converting incoming serial data back into parallel format for reception. This design operates at a baud rate of 9600 bps, with an 8-bit data frame.

**Baud Rate Generator:** The baud rate generator derives a clock signal from the 100 MHz system clock to achieve the desired baud rate of 9600 bps. The calculation is as follows:

$$Divisor\ (Dvsr) = \frac{Systems\ Clock\ Frequency}{Baud\ Rate \times Overshapimg\ Rate}$$

$$Dvsr = \frac{100 \times 10^6}{9600 \times 16} \approx 651$$

Here, an oversampling rate of 16 is commonly used for more accurate data sampling.

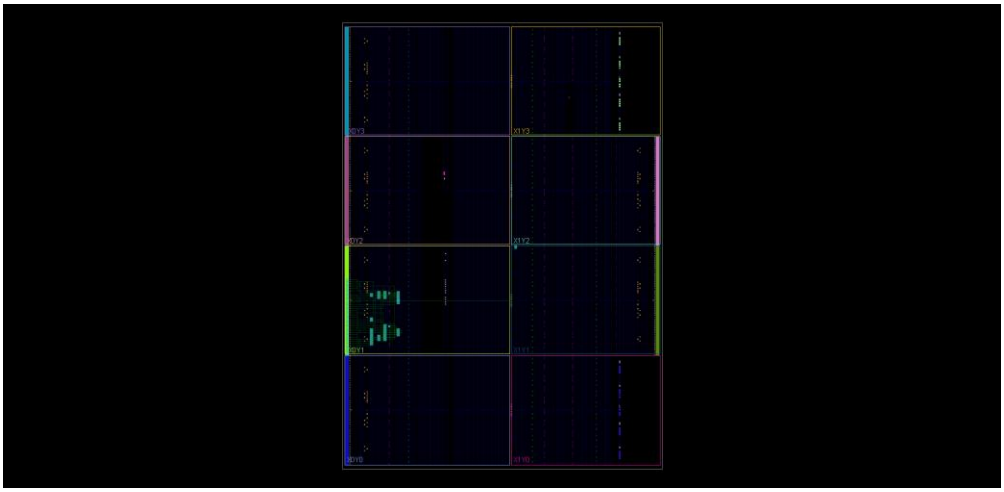
**Transmitter:** The transmitter takes parallel data from the Transmitter FIFO, serializes it, and sends it out with a start bit, 8 data bits, and a stop bit. The baud rate generator ensures that data is transmitted at the correct timing.

**Receiver:** The receiver monitors the serial line for incoming data, detects the start bit, uses the generated baud rate clock to sample and deserialize the subsequent 8 data bits, and stores the received data in the Receiver FIFO.

**FIFO Buffers:** Both the Transmitter and Receiver are equipped with FIFO buffers, each with a depth of 8 entries, allowing for efficient data management and reducing the risk of overflow or data loss during high data traffic.

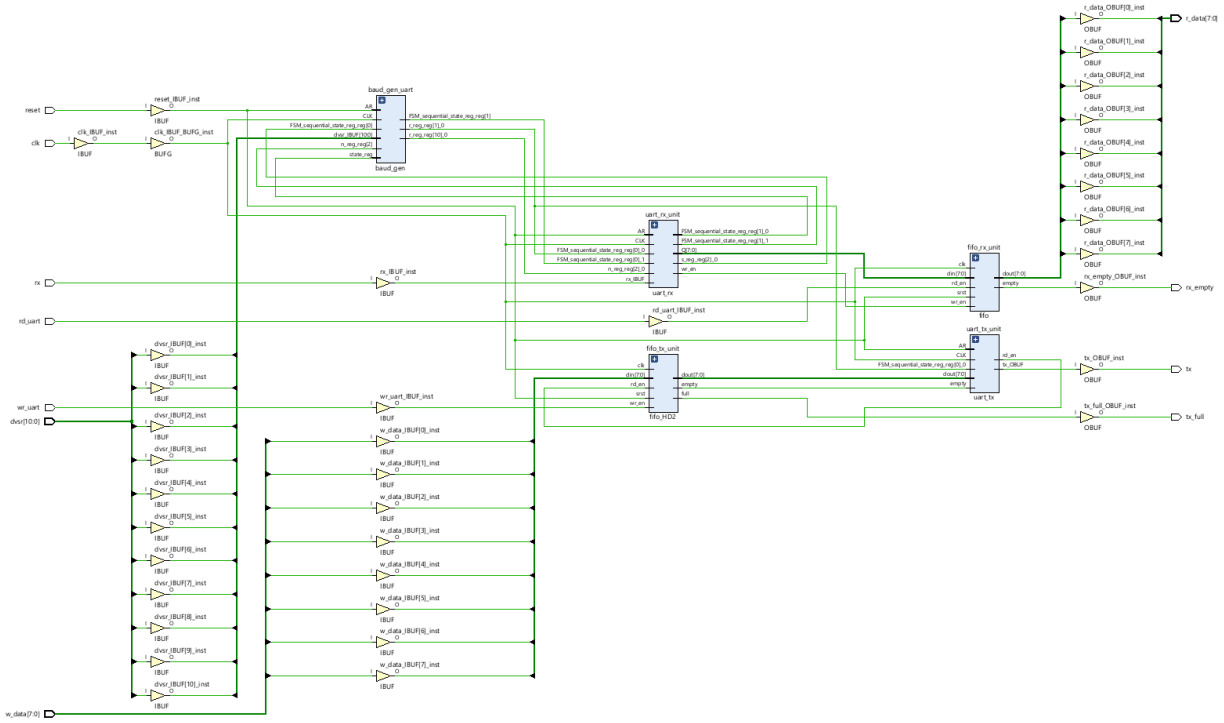
### 3.5. Schematic

After finalizing the block diagram and function design, we proceed to implement the design onto the FPGA, resulting in the device view. This view provides an overview of how the design elements are mapped to the physical resources of the FPGA.



***Implementation Device***

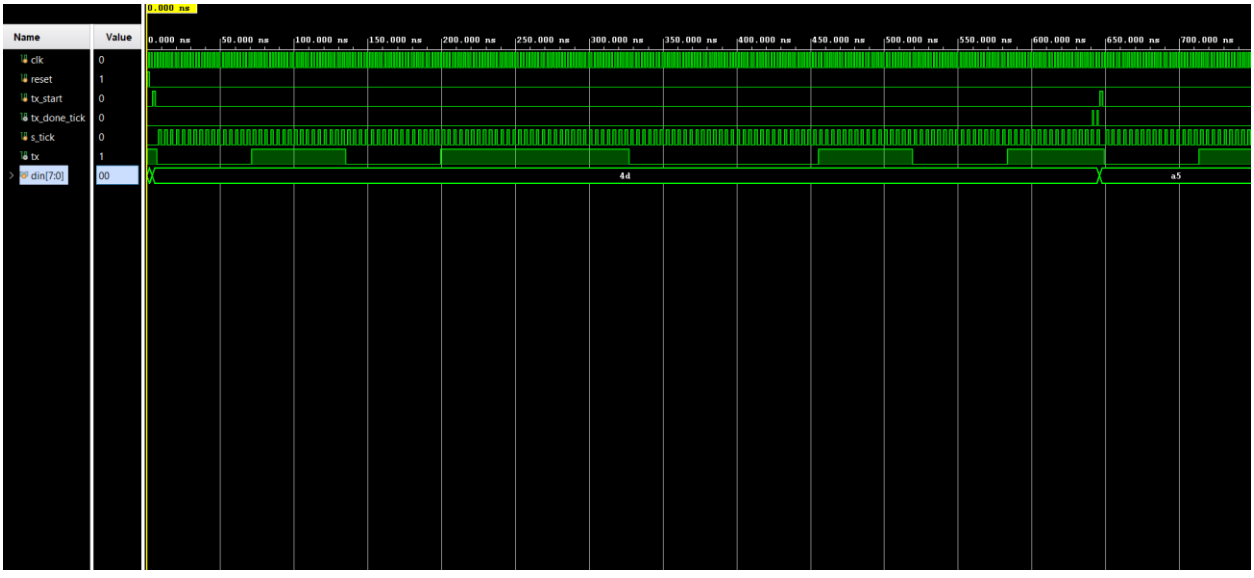
In addition to the device view, the schematic view is generated, offering a detailed look at the interconnections and layout of the design components within the FPGA. This step is essential for verifying the correct implementation of the design.



*Schematic Diagram*

# 4. Waterform and Test Vector Implementation

## 4.1. Transmitter




*Waveform of a write command*

In this simulation, we test vector the uart_tx module with two data transmissions:

**Initialization:**

- Set **reset** high for 2 ns and then low.
- Set **tx_start, din,** and **s_tick** to 0.

**Test Case 1: Transmit Data 0x4D**

- Set **din** to 8'b01001101 (0x4D).
- Assert **tx_start** for 2 ns, then deassert it.
- Generate **s_tick** pulses for 160 cycles, each pulse lasting 4 ns (total 640 ns).

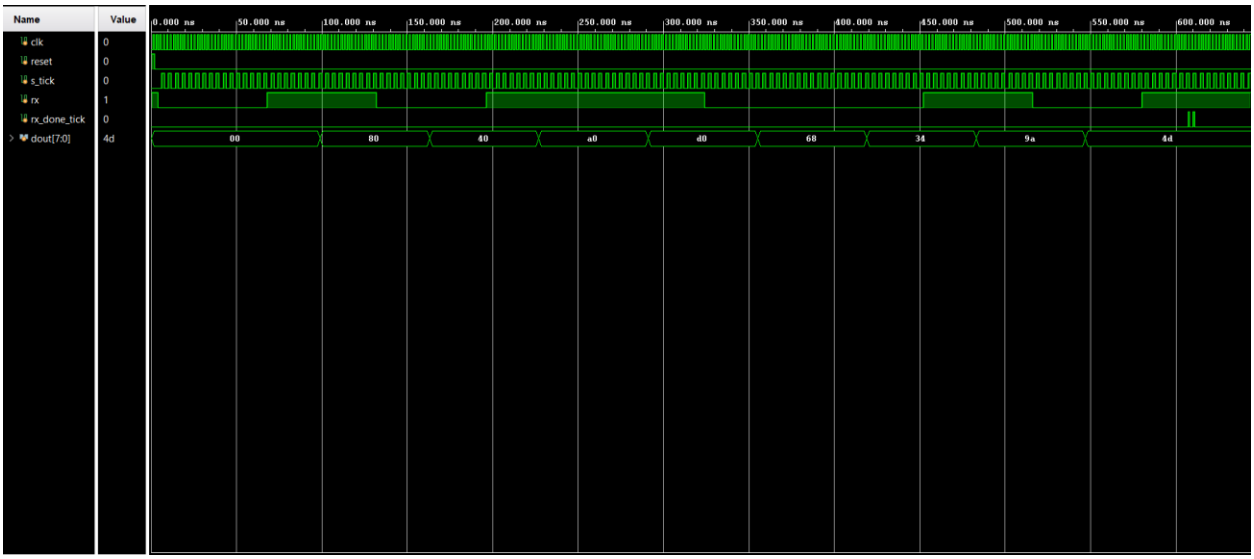
**Test Case 2: Transmit Data 0xA5**

- Set **din** to 8'b10100101 (0xA5).
- Assert **tx_start** for 2 ns, then deassert it.
- Generate **s_tick** pulses for another 160 cycles, each pulse lasting 4 ns (total 640 ns).

**Wait** for **tx_done_tick** to indicate completion of the second transmission.

**Stop** the simulation after an additional 2 ns.

### 4.2. Receiver




***Waveform of a read command***

**Initialization:**

- **reset:** Set high for 2 ns, then low to start normal operation.
- **clk:** Generates a 100 MHz clock with a 2 ns period.
- **rx:** Set to 1 (idle state).
- **s_tick:** Initially 0.

**Test Case: Receive 0x4D**

- **rx:** The send_byte task sends 0x4D.

Start bit (0)
Data bits (01001101)
Stop bit (1)

- **s_tick:** Toggled to simulate baud rate clock during transmission.

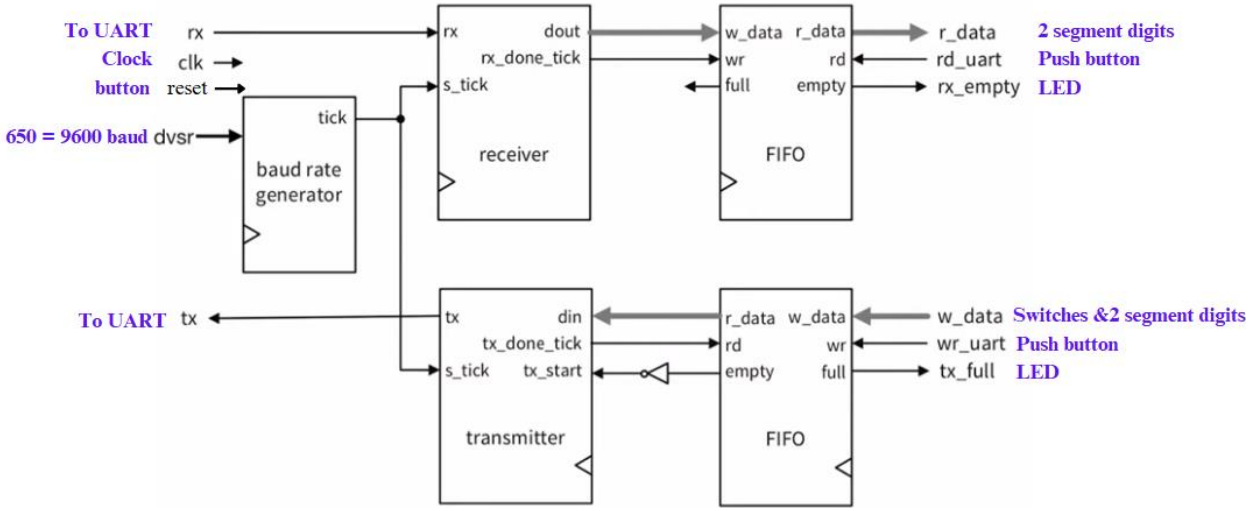
**Simulation Stop:**

- **Completion:** Wait for rx_done_tick to verify reception**.**

**End:** Stop simulation after data is received correctly.

## 5. Timing constraint file (XDC for Xilinx's FPGA)

Since the design uses the **XC7A100T-CSG324-3** FPGA, it is essential to use the **Nexys-4-DDR-Master.xdc** file for timing constraints. This XDC file is specifically designed for the **Artix-7 board**, providing predefined constraints that align with the hardware configuration and timing requirements of the **XC7A100T-CSG324-3** FPGA. By using this file, you ensure that the timing constraints are properly applied, which is crucial for optimizing the performance and reliability of your design on the **Artix-7 platform.**




***Block Diagram Pins and XDC***

Our timing constraint file is named Nexys-4-DDR-Master.xdc and includes the following key constraints:

- **clk:** The clock signal with a period of 10 ns (corresponding to a frequency of 100 MHz).
- **reset:** Enables a pull-up resistor to keep the signal high when unpressed.
- **r_data**: Displays read data on 2 seven-segment digits.
- **w_data**: Involves switches for input and displays the data on 2 seven-segment digits.
- **rd_uart, wr_data:** The push button might be used to trigger read or write operations.
- **rx_empty, tx_full:** signals and connect them to LEDs
- **tx, rx:** To configure tx and rx for the USB-RS232 interface in the Nexys-4-DDR-Master.xdc file