

Overall Project Goal (keep this fixed)

Design and verify a configurable UART (TX + RX) supporting:

- Configurable baud rate
 - 8 data bits
 - Optional parity (start with none)
 - 1 stop bit
 - Loopback / self-checking testbench
-

Week 1 — UART Fundamentals + Architecture

Day 1 – UART Protocol Understanding

Learn

- UART frame format:
Start(0) → Data[0:7] → (Parity) → Stop(1)
- Baud rate vs clock frequency
- Oversampling concept (16x preferred, but 8x ok for first version)

Deliverable

- Write a **1-page UART protocol note** (very useful for interviews)
-

Day 2 – Design Architecture

Define blocks

UART_TOP

└─ UART_TX

└─ UART_RX

└─ Baud_Generator

└─ Control / Status Registers (optional)

Key signals

- tx_start, tx_data[7:0], tx_busy
- rx_data[7:0], rx_valid
- txd, rxd

Deliverable

- Block diagram (even hand-drawn → screenshot)
-

Day 3 – Baud Rate Generator

Implement

- Parameterized baud generator:
- $\text{baud_tick} = \text{clk} / \text{baud_rate}$
- Generate **baud_tick** for TX
- Generate **oversample_tick** for RX (8x or 16x)

Verification

- Simple SV testbench to check tick interval
-

Day 4 – UART TX Design

Implement TX FSM

States:

IDLE → START → DATA → STOP → IDLE

Key points

- Shift LSB first
 - Hold each bit for 1 baud_tick
 - Assert tx_busy correctly
-

Day 5 – TX Verification

Testbench

- Drive tx_start, random tx_data
- Sample txd at baud boundaries
- Check waveform manually

Deliverable

- TX-only verified
-

Week 2 — UART RX Design + Verification

Day 6 – RX Theory

Understand

- Start bit detection (falling edge)
 - Sampling at **middle of bit**
 - Why oversampling matters
-

Day 7 – UART RX FSM

Implement RX FSM

States:

IDLE → START → DATA → STOP → DONE

Important

- Sample start bit at mid-bit
 - Shift data LSB first
 - Generate rx_valid pulse
-

Day 8 – RX Verification

Testbench

- Manually drive rxd waveform
- Check:

- Correct data reconstruction
 - Correct rx_valid timing
-

Day 9 – TX–RX Loopback Test

Connect

txd → rxd

Test

- Random data sequences
- Back-to-back frames
- Different baud rates

Deliverable

- First **end-to-end working UART**
-

Day 10 – Corner Cases

Test

- Reset in middle of frame
 - Back-to-back transmissions
 - Idle line behavior
-

Week 3 — Advanced Verification (Resume-Grade)

Day 11 – SystemVerilog Interfaces

Create:

```
interface uart_if;  
  
    logic txd, rxd;  
  
    logic tx_start;  
  
    logic [7:0] tx_data;
```

```
logic rx_valid;  
logic [7:0] rx_data;  
endinterface
```

Day 12 – Driver & Monitor

Driver

- Sends frames (bit-level or transaction-level)

Monitor

- Samples line
 - Reconstructs UART frame
 - Sends packets to scoreboard
-

Day 13 – Scoreboard

Implement

- Compare transmitted data vs received data
 - Count mismatches
 - Print pass/fail summary
-

Day 14 – Assertions (Very Important)

Add SVA

Examples:

- Start bit must be 0
- Stop bit must be 1
- rx_valid only asserted after stop bit
- tx_busy behavior

👉 This is **huge for interviews**

Day 15 – Coverage

Add:

- Data coverage (0x00, 0xFF, random)
- Back-to-back frame coverage
- Reset scenarios

Week 4 (Optional but Strong) — Polish & Extensions

Day 16 – Parity Support

- Even parity generation (TX)
- Parity checking (RX)
- Error flag

Day 17 – Documentation

Create:

- README.md
- Block diagram
- FSM diagrams
- Verification strategy

Day 18 – Resume Framing

Resume bullet

Designed and verified a configurable UART protocol in SystemVerilog with baud rate control, TX/RX FSMs, loopback testing, assertions, and functional coverage.