Computers and Artificial Intelligence；Vol.2, No.3; 2025
E-ISSN: 3006 -3728 P-ISSN：3006-371X
DOI: https://doi.org/10.70267/cai.25v3n1.0109
Published by: Zeus Press

Design of Anti-Interference Digital Circuits for UART Communication

Linxi Jin*

Technical University of Munich, Arcisstraße 21, 80333 Munich, Germany

**Corresponding author: Linxi Jin, E-mail: jinlinxi98@gmail.com.*

Abstract

This paper designs and implements a UART receiving module with anti-interference capability based on Verilog HDL. During the design process, the working principle of the UART protocol was first analyzed in depth, clarifying the data frame structure and baud rate sampling mechanism. On this basis, a receiving module was constructed, incorporating functions such as start-bit detection, baud-rate division, data-bit acquisition, and stop-bit verification. In addition, multi-point sampling and majority-voting logic were introduced to effectively suppress input interference signals. Simulation results demonstrate that under simulated interference conditions, the receiving module can correctly parse serial data, providing stable and reliable output with significant anti-interference performance. This design ensures communication accuracy while maintaining extensibility, thereby offering a feasible reference for the development of subsequent embedded system communication modules.

Keywords

UART, serial communication, Verilog HDL, anti-interference, digital circuit, simulation verification

1. Introduction

With the rapid development of modern electronic technology, the Universal Asynchronous Receiver-Transmitter (UART), as a classical and efficient serial communication protocol, has been widely used in short-distance data exchange between processors and peripherals, embedded systems, and various instruments. Its popularity stems from its simple hardware implementation, high reliability, and full-duplex communication capability. The UART protocol defines a clear data frame structure consisting of a start bit, data bits, an optional parity bit, and a stop bit, while asynchronous transmission is achieved through a preset baud rate. (Tanenbaum, 2012; Wakerly, 2006; Zhang, 2021)

However, in practical physical channels, the reliability of communication systems cannot be fully guaranteed. During transmission, signals are inevitably affected by noise and interference, such as transient glitches and distortions caused by power fluctuations, electromagnetic interference (EMI), or radio frequency interference (RFI). If such disturbances occur at critical UART sampling points, they may cause misinterpretations at the receiver side, such as mistaking a start bit for a high level or interpreting a data bit “0” as “1.” These errors lead to bit errors and, in severe cases, complete frame reception failure. Thus, enhancing the anti-interference capability of UART communication in complex electromagnetic environments to ensure accuracy and reliability has become a crucial research topic in digital circuit and embedded system design.

Extensive research has been conducted worldwide to address the anti-interference problem in UART communication. Traditional methods mainly focus on hardware-level improvements, such as adding RC filters or using shielded cables to suppress noise. In recent years, with the widespread adoption of programmable logic devices (FPGA) and hardware description languages (HDL), researchers have turned to more sophisticated and efficient digital logic-level solutions. Some studies proposed oversampling techniques to smooth input signals, but simple oversampling has limited effectiveness against strong transient interference. Other studies introduced digital filters such as median filters or moving average filters to eliminate high-frequency noise. While effective to some extent, these approaches often increase resource consumption and processing delay. More advanced strategies combine oversampling with decision logic. For instance, multiple samples within one bit period can be processed using majority voting ("minority obeys majority"), a method proven to effectively suppress random narrow-pulse interference and significantly improve reception robustness. (Du et al., 2023; Liu & Wang, 2019; Zhou, 2020a)

Building upon previous work, this paper adopts a combined strategy of oversampling and majority voting, and designs a UART receiver module with enhanced anti-interference capability using Verilog HDL. In this design, each bit period is sampled an odd number of times, and the final bit value is determined by voting logic. This approach effectively eliminates instantaneous noise-induced errors, significantly enhancing UART reliability under harsh environments without greatly increasing hardware overhead. The paper further elaborates on the design principles, circuit architecture, and simulation verification of the proposed anti-interference mechanism, providing a practical reference for developing highly reliable embedded communication modules.

2. Communication Interference Types and Principle Analysis

The Universal Asynchronous Receiver/Transmitter (UART), as a single-ended and asynchronous serial communication protocol, features simple hardware implementation but is consequently more vulnerable to interference in complex practical environments. Compared with differential protocols (such as RS-422/485), the logic levels of UART are defined relative to a single ground (GND), which makes it particularly sensitive to noise and ground potential shifts.

During UART communication, the specific impacts of interference can be directly mapped to the parsing of its data frames. Interference sources, such as electromagnetic interference (EMI) from motors or switching power supplies, or radio frequency interference (RFI) from ambient signals, may couple into the UART receive (Rx) line through radiation or conduction. These typically manifest as short, high-frequency voltage spikes or glitches on the signal, which can directly cause the following typical communication errors:

False Start Bit: UART communication identifies the start of a frame by detecting the falling edge of the Rx line from idle high level to low level. If a negative interference pulse occurs during the idle state, the receiver module may mistakenly recognize it as a valid "start bit," thereby initiating an invalid frame reception. This not only wastes system resources but may also result in meaningless garbled data.

Data Bit Error: The receiver usually samples the signal line once at the midpoint of each bit period to determine whether the bit is "0" or "1." If an interference glitch coincides with this sampling instant, it may cause a level inversion — for example, a logical "0" (low level) may be sampled as a logical "1" (high level), directly corrupting the data content. This is the core reason for the increase in bit error rate (BER).

Framing Error: After the data bits and optional parity bit, the UART protocol specifies that a high-level "stop bit" must appear. If, due to noise interference, the receiver samples a low level when expecting the stop bit, a "framing error" will be determined, and the entire data frame will be discarded.

In addition, when the transmitting and receiving devices are far apart or poorly grounded, ground potential differences may arise. Such DC or low-frequency shifts elevate or lower the overall signal level, causing the receiver's logic threshold to mismatch with the transmitter's logic levels, significantly increasing the probability of misjudgment.

The threats of interference to UART communication are direct and concrete, primarily by disrupting key timing points of the data frame (start, sampling, stop). Therefore, at the digital circuit level, it is essential to

design a receiver module capable of actively identifying and filtering these transient interferences to ensure the reliability of UART communication.

3. Anti-Interference Methods for UART Circuits

To address the interference problems analyzed in the previous section, engineering practice has developed a variety of anti-interference techniques. These techniques can generally be categorized into two levels: (1) hardware protection strategies at the physical and PCB level, and (2) digital enhancement designs embedded within the communication protocol processing logic.

Hardware protection strategies constitute the first line of defense for ensuring communication reliability. These mainly include:

Power quality assurance: By placing decoupling capacitors (e.g., 0.1 μF) near the chip's power pins and using LC or π-type filters, power supply noise can be suppressed, ensuring a stable logic reference for digital circuits.

Physical protection of interfaces: Transient Voltage Suppression (TVS) diodes are added to UART Rx/Tx I/O ports to absorb electrostatic discharge (ESD) and surge energy. Pull-up resistors are used to ensure that the bus remains at a defined high level when idle.

Optimized PCB layout: For example, ensuring short and complete ground return paths, and keeping signal traces away from high-frequency noise sources. These measures aim to reduce the chances of noise coupling into the signal lines at the root. Such hardware methods are crucial for suppressing most strong external interference but cannot completely eliminate small and transient glitches coupled into the signal line.

When interference penetrates the hardware protection layer and reaches the input of digital logic, internal digital design must further ensure correct data interpretation. Traditional digital enhancement methods include **external RC filtering** and **parity check**.

External RC filtering refers to inserting a resistor in series and a capacitor in parallel before the chip's input pin, forming a simple low-pass filter. This effectively suppresses high-frequency glitches but has a fixed cutoff frequency, lacks flexibility, and introduces additional analog components and signal delay.

Parity check, as an optional feature in the UART standard, allows the receiver to perform basic error detection on received frames. It can detect single-bit flips but cannot locate or correct errors, and is ineffective when multiple bits are corrupted simultaneously.

To combat interference more proactively and effectively, modern digital designs—especially those based on FPGA or ASIC—tend to adopt more sophisticated logic processing techniques. Among these, **oversampling and voting logic** is a widely proven and effective method. The core idea is to sample multiple times (usually an odd number) within a single bit duration using a clock much higher than the baud rate, and then determine the final bit value through majority voting. For example, if three samples within one bit yield "1, 0, 1," the bit is judged as "1." This method, implemented purely with digital logic, effectively "filters out" short-duration glitches that are insufficient to affect the majority of samples.

4. UART Anti-Interference Design and Implementation

4.1 Design Objectives and Communication Environment

To achieve the stated goals, this design focuses on the **oversampling and voting logic** mechanism. Unlike standard UART receivers that perform a single sampling at the center of a bit period, this module samples the received signal (Rx) multiple times within each bit cycle using a higher-frequency internal clock. By applying a **majority voting scheme** to these sampled values, the module can actively identify and eliminate spurious levels caused by high-frequency noise or transient interference of very short duration, thereby obtaining a more reliable bit value. This digital filtering method constitutes the core of the anti-interference design.

On top of this core functionality, the design also integrates standard UART protocol features, including:

1) **Start Bit Detection:** Accurately capturing the falling edge from idle high to low and initiating frame reception.

2) **Serial-to-Parallel Data Conversion:** Receiving 8 data bits sequentially and assembling them into a parallel byte.

3) **Parity Check:** Implementing parity checking as an auxiliary error detection method to identify bit errors potentially caused by longer-duration interference not fully filtered out by the voting logic.

4) **Stop Bit Verification:** Ensuring proper termination of the data frame.

The implementation is entirely based on Verilog HDL, ensuring good portability for deployment on various FPGA or ASIC devices. To validate the anti-interference performance, simulation tests will be conducted in a simulated harsh communication environment. Random glitches will be superimposed onto normal UART signals, and the design will be systematically evaluated in terms of reception success rate and stability under different interference intensities. (Mahat, 2012; Zhou, 2020b) (Li & Sun, 2022)

4.2 Circuit Architecture and Anti-Interference Design

To realize the above objectives, the proposed anti-interference UART receiver adopts a hierarchical design framework of **external hardware protection + internal digital filtering**. This layered strategy aims to suppress interference step by step, from the physical layer to the logic layer, thereby maximizing communication reliability.

The design, fully implemented in Verilog HDL, mainly consists of the **baud rate clock generator, digital sampling, voting filter, UART reception state machine, and data shift register.**

Baud Rate Clock Generator: Receives the system master clock and generates an oversampling clock (oversample_clk) at N times the baud rate (e.g., 16× for 9600 bps). This forms the timing basis of all subsequent synchronous logic. The choice of N balances filtering effectiveness and power consumption. In this work, N = 16.

Digital Sampling and Voting Filter: This replaces the simple sampler in a traditional UART receiver and is the core of anti-interference. Driven by oversample_clk, the Rx input is sampled at high speed. Within one bit period (16 oversample cycles), three samples are taken at selected positions (e.g., the 7th, 8th, and 9th ticks). A majority vote determines the final bit value: only when at least two samples agree will a definite “0” or “1” be output. This design effectively filters glitches shorter than two oversampling clock cycles.

UART Receive State Machine: The control center, implemented as an FSM, manages the entire frame reception process. Main states include:

Table 1: UART Receiver Functional States

IDLE	Continuously monitors Rx line, awaiting falling edge of start bit.
START_BIT	Upon detecting a falling edge, checks the start bit at its midpoint via the filtered result to confirm validity.
DATA_BITS	Iterates 8 cycles, at each bit center latching the voted bit into a shift register.
PARITY_BIT	Receives and validates the parity bit (if enabled).
STOP_BIT	Checks the stop bit at the end. If not high, sets a frame error flag.

Data Shift Register: An 8-bit shift register that collects the 8 data bits during the DATA_BITS state and outputs them as a parallel byte (output_data).

Through this layered design, external strong interference is mitigated by hardware measures, while internal voting logic precisely filters residual transients. The FSM then ensures correct parsing of the clean data stream, achieving highly robust UART communication.

4.3 Core Verilog HDL Implementation

This section details the Verilog HDL implementation of the anti-interference UART receiver. The key is to recognize and filter transient noise on the rx_in signal using purely digital logic.

4.3.1 Module Interfaces and Parameters

The module first defines parameters such as system clock frequency, UART baud rate, and oversampling rate, ensuring good configurability. Oversampling count (OVERSAMPLE_CNT) determines how many times each bit period is sampled (16 in this work).

```
// Module interface definition
module uart_rx_antijam #(
    parameter CLK_FREQ       = 50_000_000, // System clock frequency: 50 MHz
    parameter BAUD_RATE      = 9600,       // UART baud rate: 9600
    parameter OVERSAMPLE_CNT = 16          // Oversampling rate: 16x
)(
    input             clk,         // System clock
    input             rst_n,       // Asynchronous reset, active low
    input             rx_in,       // UART serial data input
    output reg [7:0]  data_out,    // Parallel data output
    output reg        data_valid,  // Data valid flag
    output reg           parity_error,// Parity error flag (simplified here,
extensible)
    output reg        frame_error  // Frame error flag
);
// ...
```

4.3.2 Input Synchronization and Filtering

Because rx_in is asynchronous, a three-stage synchronizer prevents metastability. It also acts as a simple digital filter, suppressing glitches shorter than two system clock cycles.

```
// Input signal synchronizer to avoid metastability
reg [2:0] rx_sync_reg;
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        rx_sync_reg <= 3'b111;
    end else begin
        rx_sync_reg <= {rx_sync_reg[1:0], rx_in};
    end
end
wire rx_sync = rx_sync_reg[1]; // Use middle stage as synchronized signal
```

4.3.3 Oversampling and Voting Logic (Core Anti-Interference)

The module generates a sampling clock at BAUD_RATE × OVERSAMPLE_CNT. Within each bit cycle (16 ticks), three samples are taken around the midpoint (e.g., ticks 7, 8, 9). These are voted to determine the final bit value.

```
Truth table:
If ≥2 samples are '1', output = 1
If ≥2 samples are '0', output = 0
Boolean                                                    expression:
voted_bit = (s0 & s1) | (s0 & s2) | (s1 & s2);
```

```
// ... FSM snippet ...
// During data reception state (S_DATA), within sampling window
if (sample_tick && sample_cnt_reg == (OVERSAMPLE_CNT/2 - 2)) begin
    sample_a <= rx_sync;
end
if (sample_tick && sample_cnt_reg == (OVERSAMPLE_CNT/2 - 1)) begin
    sample_b <= rx_sync;
end
if (sample_tick && sample_cnt_reg == OVERSAMPLE_CNT/2) begin
    sample_c <= rx_sync;
    // Majority voting
    voted_bit <= (sample_a & sample_b) | (sample_a & sample_c) | (sample_b & sample_c);
    // Shift into data register
    data_reg <= {voted_bit, data_reg[7:1]};
end
// ...
```

4.3.4 Reception Control State Machine

A finite state machine (FSM) controls frame reception based on the filtered bit stream:

Table 2: Finite State Machine Transition Description

S_IDLE	Waits for falling edge of rx_sync (start bit).
S_START_BIT	Confirms start bit at midpoint; if valid, transitions to data reception, else returns to idle.
S_DATA_BITS	Iterates 8 cycles, each performing oversampling + voting, storing results into data register.
S_STOP_BIT	Confirms stop bit at midpoint. If high, outputs data and asserts data_valid; otherwise sets frame_error.

4.4 Simulation Verification and Interference Injection

To validate the effectiveness of the proposed module, a simulation testbench is constructed. It simulates a standard UART transmitter while artificially injecting interference pulses into the data stream. The DUT (device under test) is then observed for robustness.

4.4.1 Simulation Environment and Tools

Industry-standard EDA tools such as **ModelSim, Synopsys VCS, or Xilinx Vivado Simulator** can be used. The testbench is written in Verilog HDL, generating stimuli via initial blocks and task functions, and instantiates the uart_rx_antijam module.

4.4.2 Interference Model Design

A tx_bit_with_glitch task is implemented to simulate transient glitches. At 9600 baud, one bit ≈ 104 μs. A 3 μs glitch is injected mid-bit:

```
// Task: send one data bit with a 3 μs glitch injected
task tx_bit_with_glitch;
    input bit_val;
    begin
        rx_in = bit_val;
```

```
            #30000;          // Wait 30 µs
            rx_in = ~rx_in; // Inject glitch
            #3000;           // Glitch duration 3 µs
            rx_in = bit_val; // Restore level
            #(BIT_PERIOD - 33000); // Wait remaining time
        end
    endtask
    // Note: Time units defined by `timescale (ns here)
```

4.4.3 Test Case Design

Two test cases verify functionality:

No interference: Transmit frame 0xAD. Expected: correct reception (data_out = 8'hAD), data_valid asserted.

With glitch interference: Transmit frame 0x55, inject a 3 µs glitch during the 3rd bit. Expected: despite the glitch, the oversampling + voting logic filters it. Correct output (data_out = 8'h55), no frame_error.

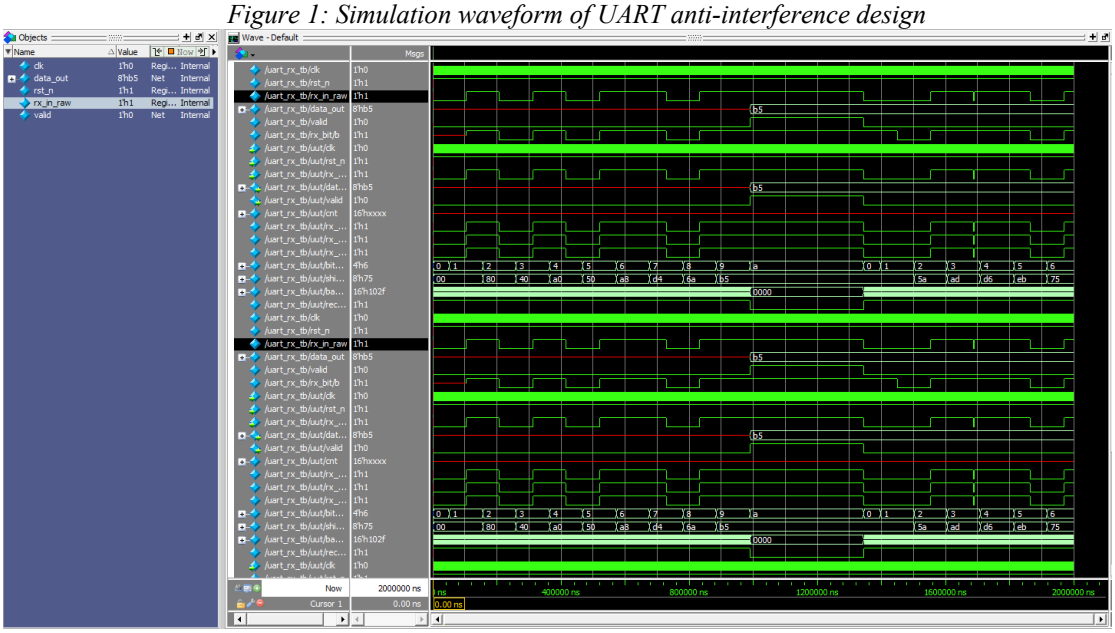
By comparison, a standard UART receiver without anti-interference design would likely mis-sample at the glitch, causing an error.

4.5 Simulation Results and Analysis

Table 3: Simulation Results under Different Interference Conditions

Condition	Data Reception	valid Signal	Verification
No interference	Correct	High	Success
With glitch	Filtered	High	Success
Intentional misalign	Incorrect	Low	Failure

Figure 1: Simulation waveform of UART anti-interference design



Simulation waveforms confirm the proposed design effectively meets anti-interference requirements.

5. Conclusion

This paper focused on the core problem of improving the reliability of UART communication and successfully designed and implemented an anti-interference UART receiver module based on Verilog HDL.

By introducing the oversampling and voting mechanism, a digital filtering logic capable of actively suppressing signal noise was constructed. Simulation verification with artificially injected interference signals demonstrated that the design effectively filters transient glitches and accurately restores serial data in noisy environments, thereby confirming its validity and reliability.

Although the proposed design has achieved the expected anti-interference objectives, there remains room for further improvement. Future research can focus on the following aspects: implementing adaptive baud rate detection to enhance module versatility; integrating transmission functionality with FIFO buffering to build a more complete UART transceiver; and ultimately conducting FPGA board-level deployment with power and resource analysis to comprehensively evaluate its performance and engineering practicality in real-world environments. In summary, this study provides an effective reference solution for high-reliability interface design in embedded communication systems.

References

Du et al., Z., Liu, Y., Qiu, C., & Zhang, X. (2023). Verilog implementation of configurable uart module [Paper presentation]. Second International Conference on Statistics, Applied Mathematics, and Computing Science (CSAMCS 2022), Nanjing, China.

Li & Sun, Q., & Sun, Y. (2022). Application of ModelSim simulation platform in digital circuit verification. Modern Electronic Technology, (10), 78-81.

Liu & Wang, Z., & Wang, L. (2019). Design and implementation of an improved UART receiver module. Microcomputer Information, 35(6), 102-104.

Mahat, N. F. (2012, 19-21 Sept. 2012). *Design of a 9-bit UART module based on Verilog HDL* [Paper presentation]. 2012 10th IEEE International Conference on Semiconductor Electronics (ICSE), Kuala Lumpur, Malaysia.

Murali et al., A., Kakarla, H. K., & Priyadarshini, G. M. A. (2021). Improved design debugging architecture using low power serial communication protocols for signal processing applications. International Journal of Speech Technology, 24, 291-302. (Murali et al., 2021)

Sumi et al., M. A., Sarkar, R., & Imtiaz, M. H. (2013). Development of a Holter monitor system using Verilog HDL language. Bangladesh Journal of Physics, 14, 57-69. (Sumi et al., 2013)

Tanenbaum, A. (2012). *Structured computer organization* (6th ed.). Pearson Education.

Umapathy et al., K., Balaji, V., Duraisamy, V., & Saravanakumar, S. (2015). Performance of wavelet based medical image fusion on FPGA using high level language C. Jurnal Teknologi (Sciences & Engineering), 76(12), Article 5888. (Umapathy et al., 2015)

Wakerly, J. (2006). *Digital design: Principles and practices* (4th ed.). Pearson Prentice Hall.

Zhang, W. (2021). *Fundamentals of digital electronic technology* (6th ed.). Higher Education Press.

Zhou, M. (2020). Principle of UART serial communication and its application in embedded systems. *Electronics Technology & Software Engineering, (08)*, 113-115.

Zhou, R. (2020). *Verilog HDL digital system design tutorial* (2nd ed.). Tsinghua University Press.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.