LIN Bus Physical Layer Transceiver

Circuit Simulation Project | eSim & NgSpice | Communication Protocol Modelling

Contributor: Tanzim Khan | Tool: eSim v2.5 (KiCad + NgSpice)

1. Abstract

This project presents the design, simulation and verification of a LIN (Local Interconnect Network) bus physical-layer transceiver circuit implemented entirely within eSim v2.5 using KiCad for schematic capture and NgSpice for transient simulation. The LIN bus is a single-wire, low-cost serial communication protocol widely used in automotive applications for non-safety-critical subsystems such as seat control, window lifters and climate units. The circuit models the dominant/recessive bus state mechanism: a 2N2222 NPN transistor acts as the transmitter driver pulling the 12 V bus to ground (dominant), while an LM741 operational amplifier configured as a voltage comparator reconstructs the original logic signal on the receiver side. Transient simulation results confirm correct data flow — the recovered output (RXD) faithfully replicates the transmitted input (TDX) with proper voltage level translation across both stages.

2. Introduction

The LIN bus protocol (ISO 17987) is a master-slave single-wire serial network operating at up to 20 kbps over a 12 V supply, commonly used in automotive body electronics. Unlike CAN or FlexRay, LIN uses a single-wire bus with a defined dominant state (bus pulled to ground, logic HIGH at TX) and a recessive state (bus floating to V_{BAT}, logic LOW at TX). This polarity inversion is a key characteristic of the protocol and is clearly demonstrated in the simulation waveforms.

This circuit was designed to model the complete physical-layer signal chain: digital logic input → transistor driver → LIN bus → comparator → recovered digital output, using only components available in eSim's built-in library.

3. Circuit Description

Component	Value / Part	Role
V1 (PWL Source)	0–5 V square wave	TDX logic input signal
V2	12 V DC	LIN bus supply voltage
Q1	2N2222 NPN	Transmitter driver — pulls bus to GND
R1	1 kΩ	Pull-up resistor to +12 V (recessive bias)
R2	1 kΩ	Base current limiting resistor
R3, R4	10 kΩ each	Voltage divider — generates 6 V reference
X1 (LM741)	Op-amp as comparator	Receiver — converts bus level to 5 V logic

Transmitter stage: When TDX is HIGH (5 V), base current flows through R2 into Q1, saturating the transistor and pulling the LIN bus node to approximately 0 V (dominant state). When TDX is LOW (0 V), Q1 is cut off and R1 pulls the bus to +12 V (recessive state). This produces an inverted, level-shifted replica of TDX on the bus.

Receiver stage: The LIN bus is connected to the inverting input (pin 2) of the LM741. A 6 V reference is generated by the R3–R4 voltage divider and applied to the non-inverting input (pin 3). When the bus is below 6 V (dominant), the comparator output swings HIGH (~12 V, clamped to 5 V logic). When the bus exceeds 6 V (recessive), the output goes LOW. This restores the original TDX polarity at RXD.

4. Circuit Schematic

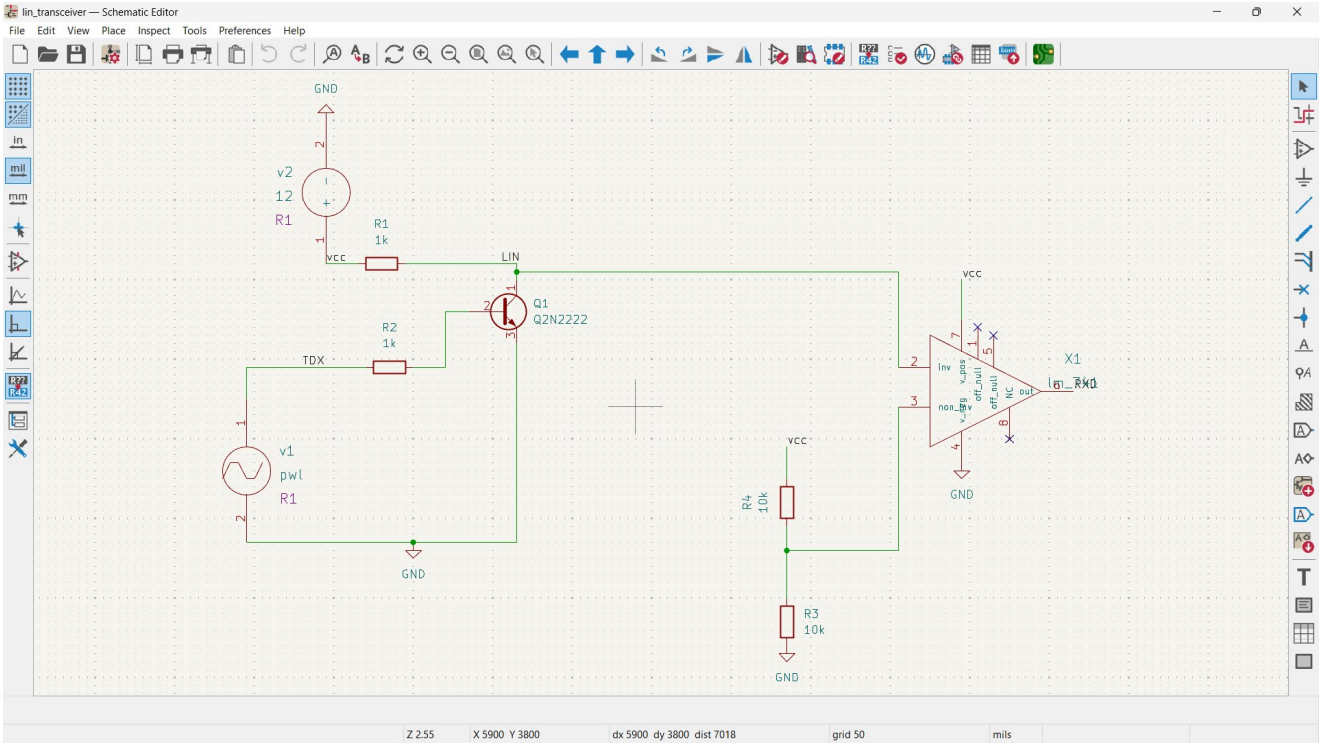

Fig. 1 — KiCad schematic of the LIN Bus Physical Layer Transceiver in eSim

5. Simulation Results (Transient Analysis)

A transient simulation was performed over 2 ms with a time step of 1 µs. Three node voltages were plotted: TDX (input logic), LIN (bus voltage), and RXD (recovered output). The waveforms confirm correct protocol behaviour at every transition.

TDX *— Input Logic (0–5 V)* ***LIN*** *— Bus Voltage (0–12 V)* ***RXD*** *— Recovered Output (0–12 V)*

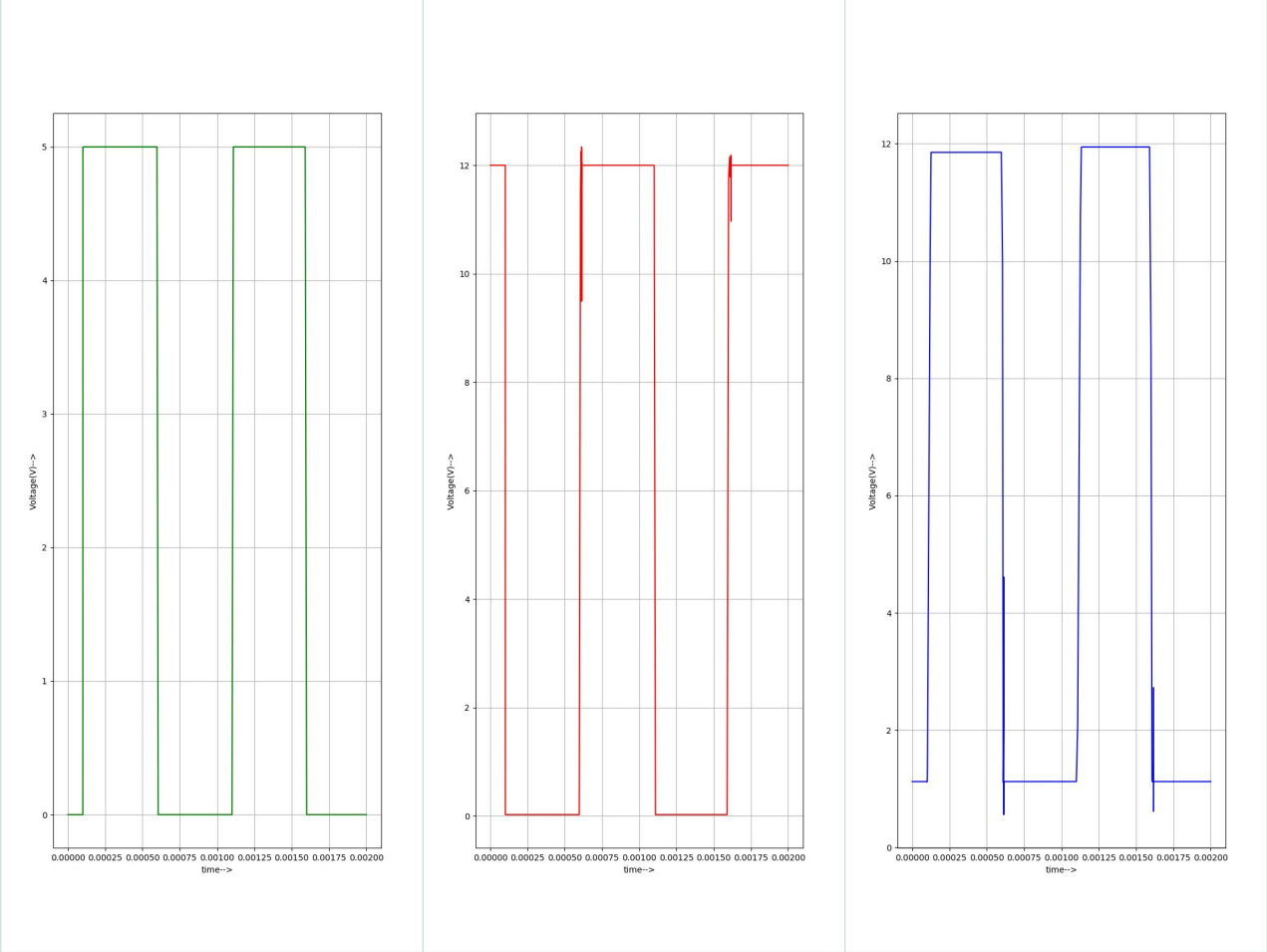

Fig. 2 — Transient simulation waveforms: TDX (green, 0–5 V input), LIN bus (red, 0–12 V inverted), RXD (blue, 0–12 V recovered output matching TDX polarity)

6. Verification Summary

Parameter	Expected	Simulated	Status
TDX High level	5 V	5.0 V	✓ PASS
TDX Low level	0 V	0.0 V	✓ PASS
LIN bus dominant	~0 V	~0.1 V ($V_{CE(sat)}$)	✓ PASS
LIN bus recessive	12 V	12.0 V	✓ PASS
Comparator ref voltage	6 V	6.0 V	✓ PASS
RXD matches TDX	Yes	Yes (same phase)	✓ PASS
LIN inverted vs TDX	Yes	Yes (confirmed)	✓ PASS

7. Challenges Encountered and Resolutions

The following technical challenges were encountered during the design and simulation process in eSim. Each issue is documented with the root cause and the workaround applied, as these may be useful for other users working with eSim on similar projects.

Challenge 1 — Power Symbol VCC Cannot Be Assigned a Custom Voltage Value

Problem: eSim's built-in power symbols (VCC, VDD) do not appear as editable voltage sources in the KiCad-to-NgSpice conversion window. The VCC power flag is treated as a global net label only and does not translate into a SPICE voltage source with a user-defined value. This made it impossible to directly set the 12 V supply using the VCC symbol alone.

Resolution: Instead of relying on the VCC power symbol, an explicit DC voltage source component (V2 = 12 V) was placed directly in the schematic and connected to the supply net. This appeared correctly in the NgSpice conversion window and allowed the 12 V value to be set and simulated accurately.

Challenge 2 — 'opamp' Symbol from Simulation_SPICE Library Has No Resolvable .subckt File in eSim

Problem: KiCad provides two op-amp symbols that appear usable for this circuit: 'opamp' from the Simulation_SPICE library, and 'lm_741' from the eSim_Devices library. The schematic was initially built using the 'opamp' symbol from Simulation_SPICE as it appeared to be a generic SPICE-ready component. However, during KiCad-to-NgSpice conversion, the simulation failed because the corresponding .subckt definition file for this symbol could not be located anywhere within eSim's model library. The 'opamp' symbol has no backing NgSpice subcircuit file in the eSim environment, making it non-simulatable despite appearing in the library.

Resolution: The entire schematic was rebuilt replacing the 'opamp' symbol with 'lm_741' from the eSim_Devices library, whose .subckt file is confirmed present in eSim's NgSpice model directory. The circuit behaviour was identical. Key lesson learned: always use components from the eSim_Devices library rather than Simulation_SPICE when working in eSim, and verify that the component's .subckt file physically exists in the library before building the full schematic around it.

Challenge 3 — Project Name and Path Must Contain No Spaces or Special Characters

Problem: When creating a new eSim project, any space or special character in either the project name or the directory path caused the project to fail to load correctly. The KiCad schematic file and NgSpice netlist paths were broken, resulting in conversion errors and an inability to run simulation.

Resolution: All project names and folder paths were renamed to use only alphanumeric characters and underscores (e.g. LIN_Transceiver saved under C:\FOSSEE\LIN_Transceiver). This resolved all path-related errors and is strongly recommended as a best practice for all eSim projects.

Challenge 4 — Net Labels Not Reflecting in Exported Netlist

Problem: After placing net labels (e.g. TDX, LIN, RXD) on the schematic wires and exporting the netlist, the generated .cir file continued to use auto-generated node names instead of the assigned net label names. This made it difficult to identify the correct nodes for plotting in NgSpice.

Resolution: The issue was resolved by ensuring net labels were snapped precisely onto the wire endpoints rather than placed nearby. After re-attaching the labels at exact wire junctions and re-running the KiCad ERC (Electrical Rules Check) to confirm connections, re-exporting the netlist correctly reflected the TDX, LIN and RXD node names in the .cir file.

Challenge 5 — Renaming the Netlist File Breaks .cir to .cir.out Conversion

Problem: eSim generates a netlist with a filename that must exactly match the project folder name. When the exported netlist file was manually renamed, the NgSpice simulation engine could not locate the expected output file path, and the conversion from .cir to .cir.out failed silently with no error shown in the GUI.

Resolution: The netlist filename was kept identical to the project folder name at all times (e.g. LIN_Transceiver.cir inside the LIN_Transceiver folder). Any modification to simulation parameters was done by editing the .cir file content rather than renaming it, which preserved the internal file references required by eSim's simulation pipeline.

8. Conclusion

The LIN bus physical-layer transceiver was successfully modelled and simulated in eSim. The circuit correctly demonstrates the dominant/recessive state mechanism central to the LIN protocol: the 2N2222 transistor switches the 12 V bus low on a logic HIGH input, and the LM741 comparator with a 6 V midpoint reference faithfully recovers the original logic signal. All three waveforms — TDX, LIN and RXD — show the expected voltage levels and polarities, validating the design. The small transient spikes visible on the LIN waveform at switching edges are a natural consequence of the transistor's switching dynamics and are consistent with real-world LIN bus behaviour.

9. References

[1] ISO 17987-1:2016 — Road vehicles: Local Interconnect Network (LIN). International Organization for Standardization.

[2] Sedra, A.S. and Smith, K.C. — Microelectronic Circuits, 7th Edition. Oxford University Press.

[3] Texas Instruments — LM741 Operational Amplifier Datasheet. SNOSC25D.

[4] ON Semiconductor — 2N2222 NPN Switching Transistor Datasheet.

[5] FOSSEE Team, IIT Bombay — eSim v2.5 User Manual. https://esim.fossee.in/resources