

Circuit Simulation Project

<https://esim.fossee.in/circuit-simulation-project>



The Circuit Simulation Project is an initiative of FOSSEE, IIT Bombay that promotes the use of eSim for designing and simulating electrical and electronic circuits for learning and application purposes. The objective is to create and document original, application-based, or proof-of-concept circuit designs using eSim to build an open-source resource database.

Name of the participant : Chethan Aithal

Affiliation / Institution : Department of Electronics and Communication Engineering, SJB Institute of Technology, Bengaluru, Karnataka, India

Title of the circuit : Single Edge Nibble Transmission (SENT) Protocol Implementation in eSim

Theory/Description :

SENT (Single Edge Nibble Transmission), standardised as SAE J2716, is a point-to-point signalling scheme used to send sensor readings from a sensor IC to an ECU over a single digital wire. It sits between a plain PWM/analog output, which is cheap but noise-sensitive, and a full serial bus such as LIN or CAN, which is accurate but adds cost that is not justified for one sensor. SENT keeps PWM-like wiring while transmitting fully digital, checksum-protected data. The protocol encodes information as the time between falling edges rather than as voltage levels, so the receiver only has to measure edge spacing, which tolerates moderate voltage drops on the harness.

This project is a Verilog implementation based on the SENT SAE J2716 fast-channel frame structure — synchronisation/calibration pulse, status nibble, six data nibbles, CRC nibble and pause — rather than a full implementation of every option in the SAE J2716 specification (e.g. slow-channel messaging or all defined tick-length variants are outside this project's scope). The transmitter, receiver and CRC-4 checker are implemented in synthesisable Verilog and connected through a single wire inside a top-level loopback module, `sent_top.v`, so the transmit-to-decode chain implemented here can be exercised and self-checked in one simulation.

Frame structure (sent_tx.v):

Every frame opens with a fixed 56-tick synchronisation/calibration pulse. Both sent_tx and sent_rx are driven from the same system clock in this loopback simulation (sent_top.v connects a single clk input to both instances), so the synchronisation pulse is not needed to recover a separate receiver clock; rather, it lets the receiver calibrate the tick length purely from the timing of transitions on sent_line, which is how a real SENT link (where transmitter and receiver do not share a forwarded clock) would also operate. This sync pulse is followed by a status nibble, six data nibbles (D5..D0, carrying the 24-bit sensor_data_in value) and a CRC nibble, and the frame closes with a pause pulse. Every nibble symbol's high time is 12 + nibble_value ticks, applied identically on encode (symbol_total_ticks function) and decode (rounded_ticks – 12).

Symbol	Order in frame	Tick width	Carries
SYNC	1st	56 (fixed)	Calibration reference for tick length
STATUS	2nd	12 + value	status_in[3:0]
D5 .. D0	3rd – 8th	12 + value (each)	sensor_data_in[23:0], six nibbles
CRC	9th	12 + value	4-bit CRC of the six data nibbles + trailing zero nibble
PAUSE	10th (last)	20 (fixed, PAUSE_TICKS)	Frame closure before next SYNC

Table 1. SENT frame symbols generated by sent_tx.v, in transmission order.

The tick period is set by the TICK_CLKS parameter (10 clock cycles here, for fast simulation). Re-targeting the design to real SENT timing (3–90 μ s ticks) only requires changing TICK_CLKS and the input clock frequency; the FSM logic is untouched.

CRC-4 data integrity (sent_crc4.v):

Data integrity is handled by sent_crc4.v, which implements the CRC-4 recommended in SAE J2716. The implemented CRC block calculates the CRC from the six 4-bit sensor-data nibbles (data_in[23:0]) followed by the additional trailing zero nibble used by the implemented CRC procedure; the status nibble is transmitted as a separate field and is not included in the CRC calculation in this implementation, since sent_crc4.v takes only the 24-bit data_in bus as its input. The same crc_step function is instantiated once inside sent_tx.v, to compute the CRC nibble that is transmitted, and once inside sent_rx.v, to independently recompute the CRC from the data nibbles actually received; the two are compared to produce crc_ok.

Parameter	Value used in sent_crc4.v
Polynomial	$x^4 + x^3 + x^2 + 1$ (4'hD)
Initial seed	4'h5
Nibble order	Six data nibbles, most-significant-nibble first
Padding	One trailing zero nibble appended (recommended 2010+ method)

Table 2. CRC-4 configuration implemented in sent_crc4.v.

Receiver operation (sent_rx.v):

Since sent_rx shares the same clk as sent_tx in this loopback (see Frame structure above), the information it must recover is carried entirely in sent_line's edge timing rather than in a forwarded clock — so sent_rx decodes it by measurement alone: sent_line first passes through a two-flip-flop synchroniser (sent_d1, sent_d2) before edge detection — standard defensive practice for a signal driving edge-sensitive logic, and consistent with how the same receiver would also handle a sent_line supplied by an independent transmitter clock in a non-loopback configuration. A free-running counter (edge_clk_count) measures clock cycles between consecutive falling edges; this count is rounded to the nearest tick and classified: 55–57 ticks is a sync pulse, 12–27 ticks is a data/status/CRC nibble. An interval outside these windows, or one that deviates from the nearest tick boundary by more than TOLERANCE_CLKS (2 clock cycles in this implementation), is flagged on timing_error rather than silently misinterpreted — mirroring how a real SENT receiver must reject glitches or bit-rate drift. The eighth nibble after a valid sync pulse is treated as CRC; at that point status_out, data_out and crc_received are latched together and data_valid pulses for one clock, giving a single, unambiguous sampling point for a complete frame.

Loopback architecture (sent_top.v):

sent_top.v wires the transmitter's sent_line output directly to the receiver's sent_line input. This makes the design self-checking in simulation: a testbench only drives status_in and sensor_data_in, and can directly compare decoded_status/decoded_data against the applied values while observing crc_ok, timing_error and sync_detected as pass/fail indicators — no external SENT sensor or bus analyser is required.

Integration with eSim via NgVeri:

RTL functionality was first verified through digital simulation of sent_top.v (Icarus Verilog/Verilator), which confirmed the FSM, tick counting and CRC logic independently of eSim. Separately, the Verilog design was interfaced with the eSim environment using NgVeri and ADC/DAC bridge models: the RTL files are opened in the Makerchip tab, and NgVeri's “Run Verilog to Ngspice Converter” (built on Verilator) compiles the Verilog into an XSPICE code model that ngspice can load as a d_cosim block. Once generated, this model can be placed as a component in eSim's Eeschema schematic editor alongside SPICE-level elements and voltage sources, which is the mechanism eSim provides for mixed-signal integration of this digital design. The bridge blocks are required because the NgVeri-generated model exchanges purely digital (0/1) values, while the surrounding eSim schematic is solved as a SPICE circuit

in continuous voltage/current; the ADC/DAC bridges perform this digital-to-analog and analog-to-digital conversion at the model boundary so the two simulation domains can co-simulate in one ngspice run.

Reason to simulate with eSim :

A pure RTL testbench (Icarus Verilog/Verilator) confirms the FSM, tick counting and CRC logic are functionally correct in isolation, but it does not place the design in a circuit environment. The NgVeri interface allows the Verilog protocol model to interact with SPICE-based sources and bridge elements within the eSim simulation environment, which is the basis for extending this RTL-level verification toward mixed-signal integration in future work — for example, introducing pull-up/loading elements on sent_line to study the effect of realistic circuit loading on edge timing and the margin left in TOLERANCE_CLKS. This project's simulation evidence, presented in the Expected Results section, is digital RTL simulation output; no such loading experiment has been carried out to date.

Because eSim is built on open-source tools (KiCad, ngspice, Verilator), the NgVeri-based integration path is reproducible by anyone without a paid mixed-signal simulator licence, which fits the goal of the CSP open-source database.

Expected Outcome/outputs :

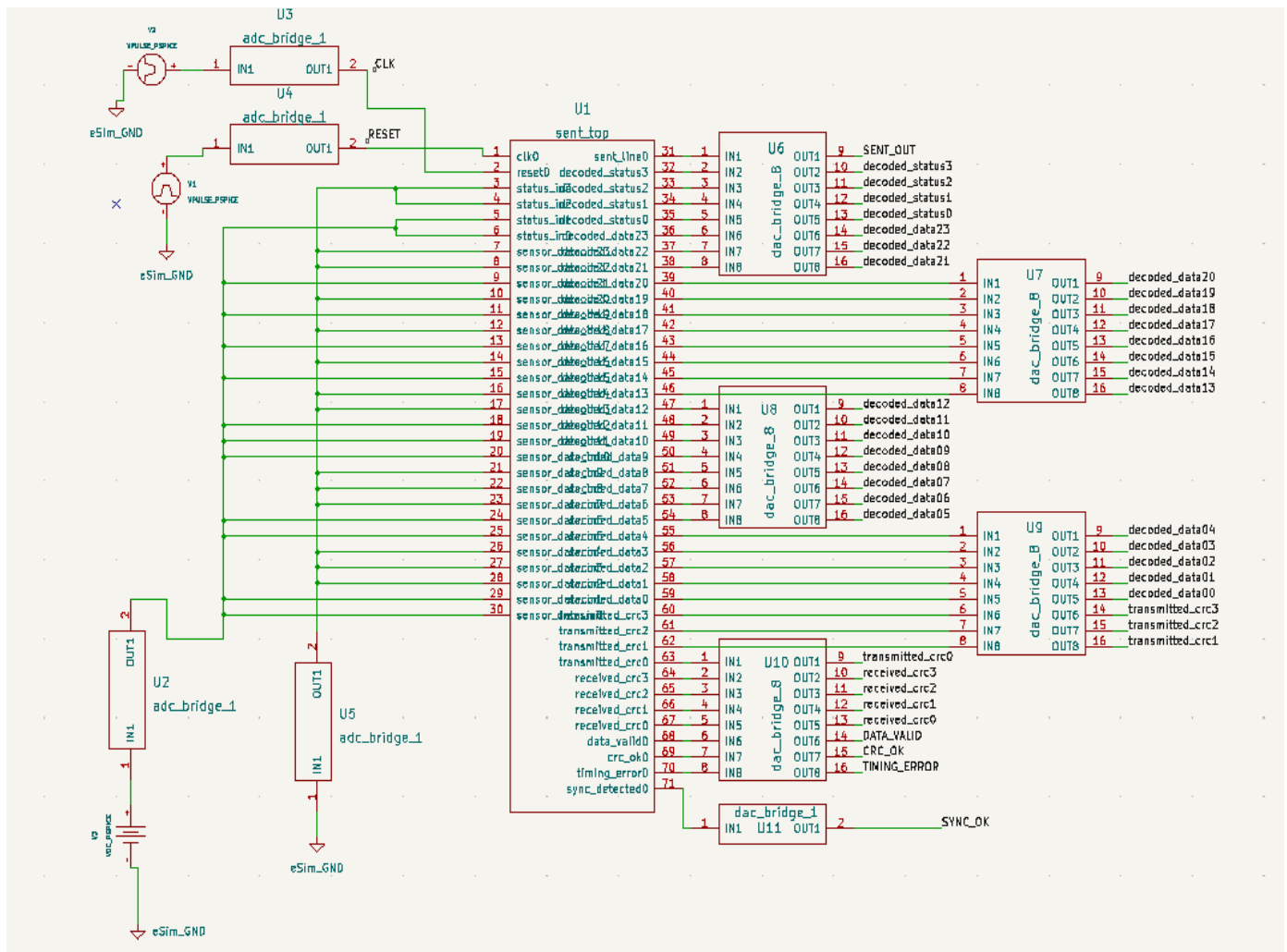
With reset released and status_in / sensor_data_in held at fixed test values, sent_tx.v is expected to free-run and produce a repeating frame on sent_line: a 56-tick sync pulse, eight nibble pulses (status, D5..D0, CRC), then the fixed pause, after which the sequence repeats. Driven from the same wire, sent_rx.v is expected to assert sync_detected once per frame, decode the following seven nibbles, and pulse data_valid on the CRC nibble's closing edge while presenting decoded_status = status_in and decoded_data = sensor_data_in.

Output signal	Expected behaviour in clean loopback
sync_detected	One pulse per frame, at the edge closing the 56-tick sync pulse
data_valid	One pulse per frame, at the edge closing the CRC nibble
decoded_status / decoded_data	Equal to status_in / sensor_data_in applied at the transmitter
crc_ok	1 on every frame (receiver's local CRC matches transmitted CRC)
received_crc vs transmitted_crc	Numerically equal
timing_error	0 throughout; should assert only if sent_line is deliberately corrupted

Table 3. Expected receiver behaviour for a clean, uncorrupted loopback frame.

Circuit Diagram(s) :

This project is an RTL/mixed-signal design rather than a discrete analog schematic, so the “circuit” corresponds to the module hierarchy of sent_top.v — sent_tx with its internal sent_crc4 instance, the sent_line net, and sent_rx with its own internal sent_crc4 instance — shown in the block diagram below. After the Verilog is converted through NgVeri, the resulting XSPICE symbol(s) can be placed on an eSim schematic sheet, for example with a pull-up resistor and load capacitance on sent_line, to obtain a circuit-level schematic for this section.



Block Diagram (s) :

The diagram reflects the actual connectivity in sent_top.v.

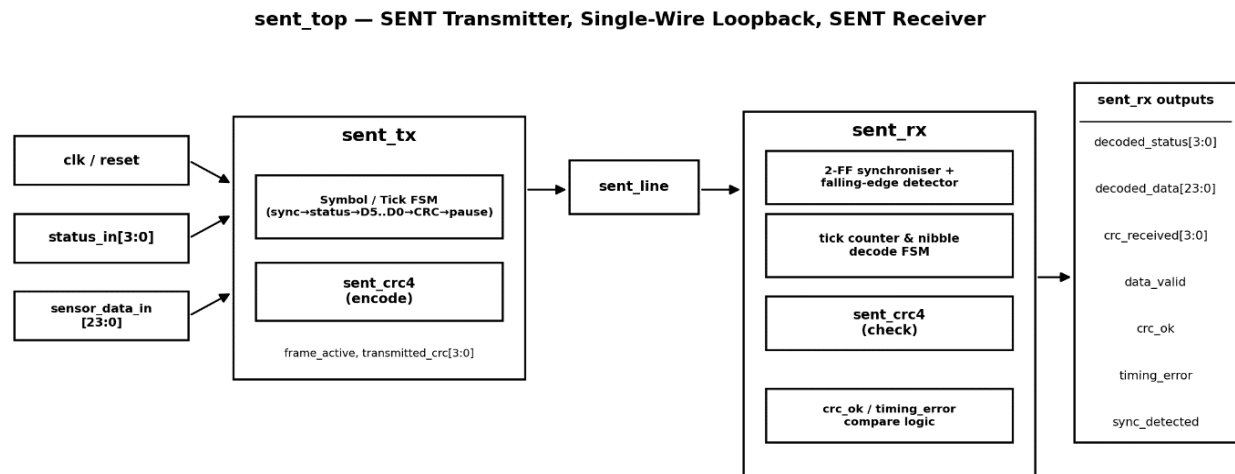


Fig. 1. Block diagram of sent_top — SENT transmitter, single-wire loopback, SENT receiver.

Signal (sent_top.v)	Dir.	Width	Description
clk, reset	in	1 bit each	System clock and asynchronous active-high reset
status_in	in	4 bits	Status nibble applied to the transmitter
sensor_data_in	in	24 bits	Sensor payload applied to the transmitter
sent_line	internal	1 bit	Loopback net: tx output tied to rx input
decoded_status, decoded_data	out	4 bits, 24 bits	Values recovered by the receiver
transmitted_crc, received_crc	out	4 bits each	CRC computed at tx vs. CRC checked at rx
data_valid, crc_ok	out	1 bit each	Frame-complete pulse; CRC match flag
timing_error, sync_detected	out	1 bit each	Edge-timing fault flag; sync-pulse flag

Table 4. Top-level port summary of sent_top.v.

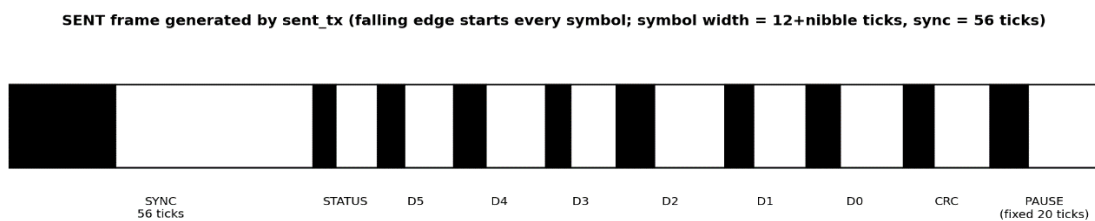


Fig. 2. SENT frame layout as generated by sent_tx.v.

Expected Results (Input, Output waveforms and/or Multimeter readings) :

The following presents the simulation-based waveform verification of the SENT transmitter, receiver, CRC, and timing-check logic, obtained from digital simulation of the Verilog design integrated into eSim via the NgVeri co-simulation interface with ADC/DAC bridge connections, for the applied stimulus `status_in = 4'h3`, `sensor_data_in = 24'h2A9C63`. No physical hardware measurements were performed; all results below are logic-level simulation results.

Simulation Output and Waveform Analysis:

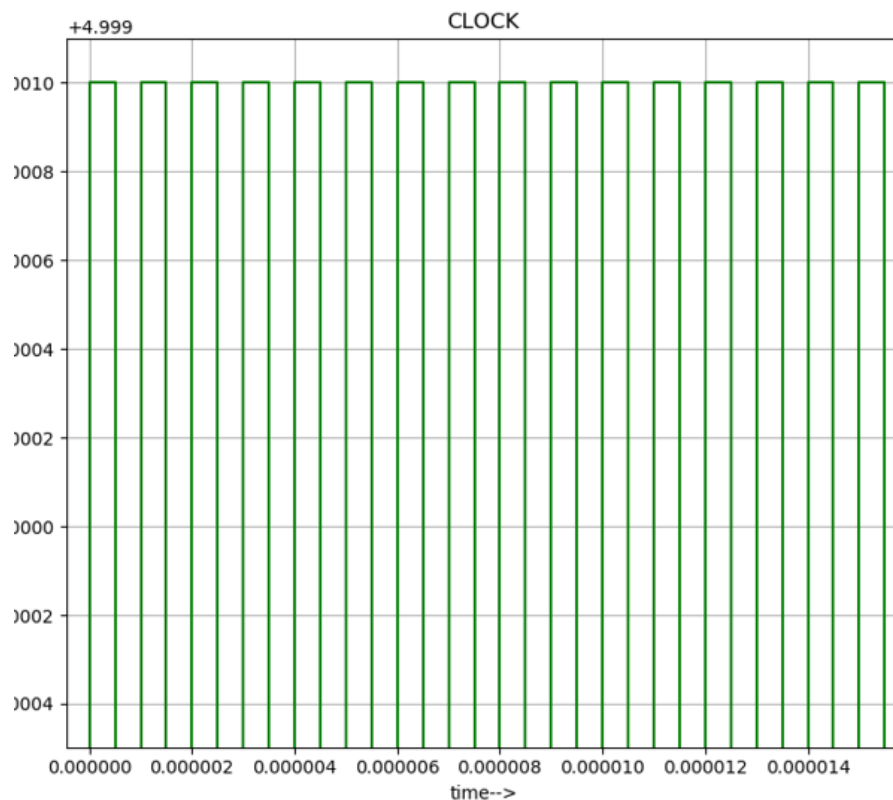


Fig. 3. System clock (CLK) waveform used as the synchronous timing reference for the transmitter and receiver.

CLK is the synchronous reference clock used throughout `sent_tx.v` and `sent_rx.v`. In this simulation `TICK_CLKS = 10`, so every SENT tick corresponds to ten clock periods; both the transmitter's symbol/tick FSM and the receiver's edge-to-tick conversion are timed from this clock. Fig. 3 shows CLK toggling regularly and continuously for the duration of the run, confirming that a stable timing reference was present throughout the test.

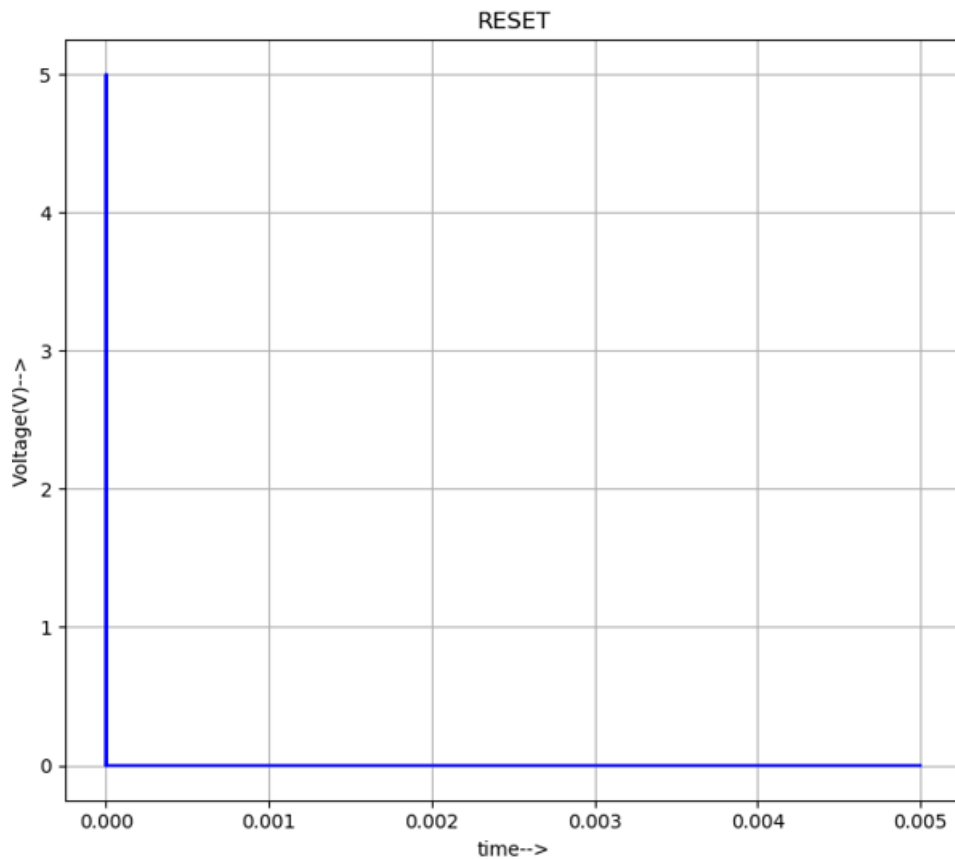


Fig. 4. Reset (RESET) waveform showing system initialization prior to SENT communication.

RESET is an asynchronous, active-high reset — both `sent_tx.v` and `sent_rx.v` use always `@(posedge clk or posedge reset)` blocks, so assertion of reset immediately forces the transmitter and receiver state machines to a known state without waiting for a clock edge. Fig. 4 shows RESET asserted at the start of the simulation and then de-asserted, after which the transmitter begins generating the SENT frame and the receiver becomes active. This confirms that communication begins only after a defined initialization sequence, consistent with the reset handling in `sent_tx.v` and `sent_rx.v`.

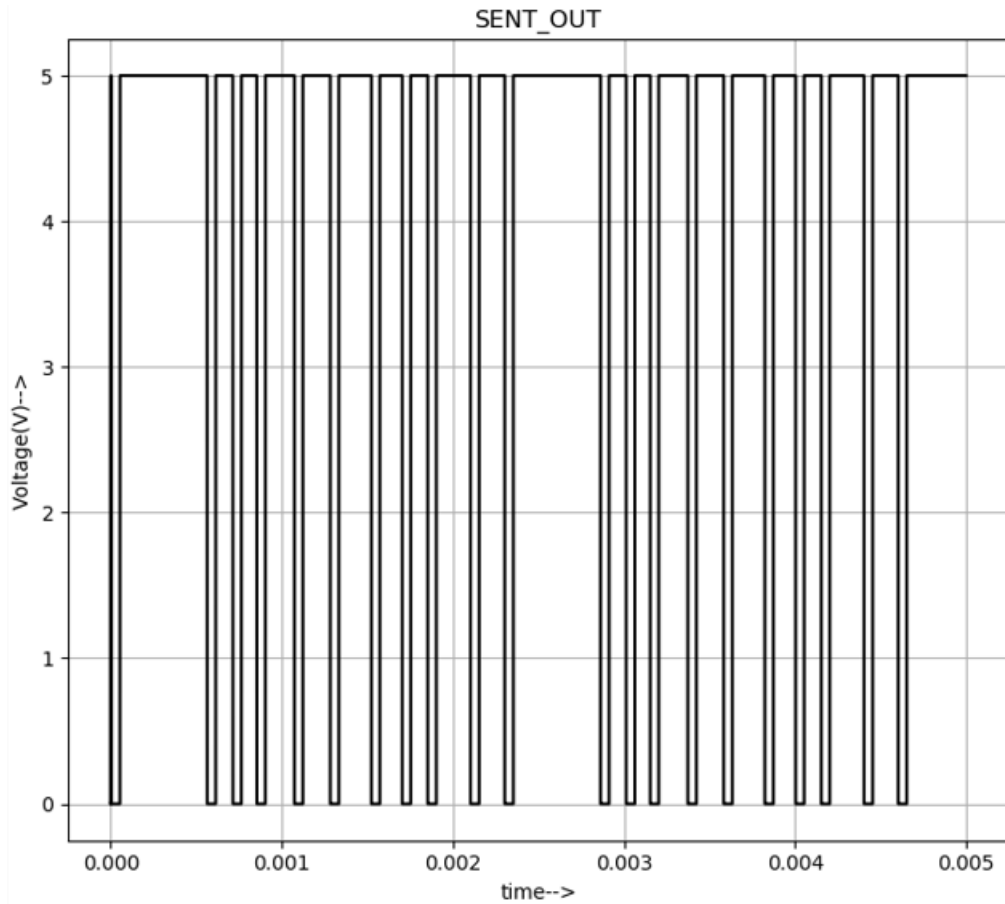


Fig. 5. SENT_OUT waveform showing the transmitted, pulse-width encoded frame.

SENT_OUT is the single-line output of sent_tx.v; information is represented purely through the timing between falling edges rather than voltage level. A synchronization interval is transmitted first, followed by the status nibble, the six data nibbles, the CRC nibble, and the pause interval, after which the sequence repeats. Fig. 5 shows this structure directly: one wider pulse (synchronization) followed by a series of narrower pulses (status, data, CRC), confirming that the transmitter generates a frame consistent with the structure implemented in sent_tx.v.

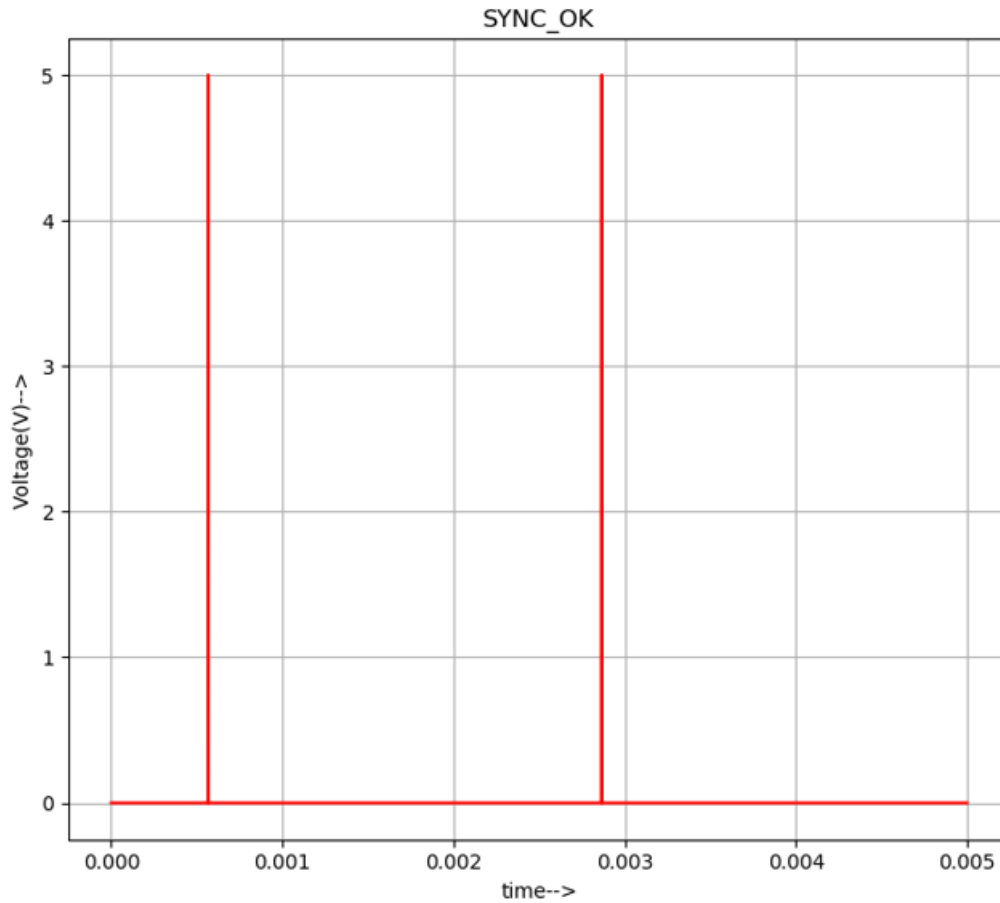


Fig. 6. sync_detected waveform confirming recognition of the synchronization interval.

The receiver measures the width of the first pulse in every frame and compares it against the expected synchronization range before decoding subsequent nibbles. Fig. 6 shows sync_detected producing a discrete pulse once per frame, confirming that the receiver correctly identifies the calibration/synchronization interval transmitted by sent_tx.v before nibble decoding begins.

A dedicated decoded_status waveform capture was not among the supplied images; however, the decoded status value recorded for this simulation is decoded_status = 4'b0011 = 4'h3, matching the applied status_in = 4'h3 exactly. This confirms correct reconstruction of the status nibble by the receiver's decode logic in sent_rx.v.

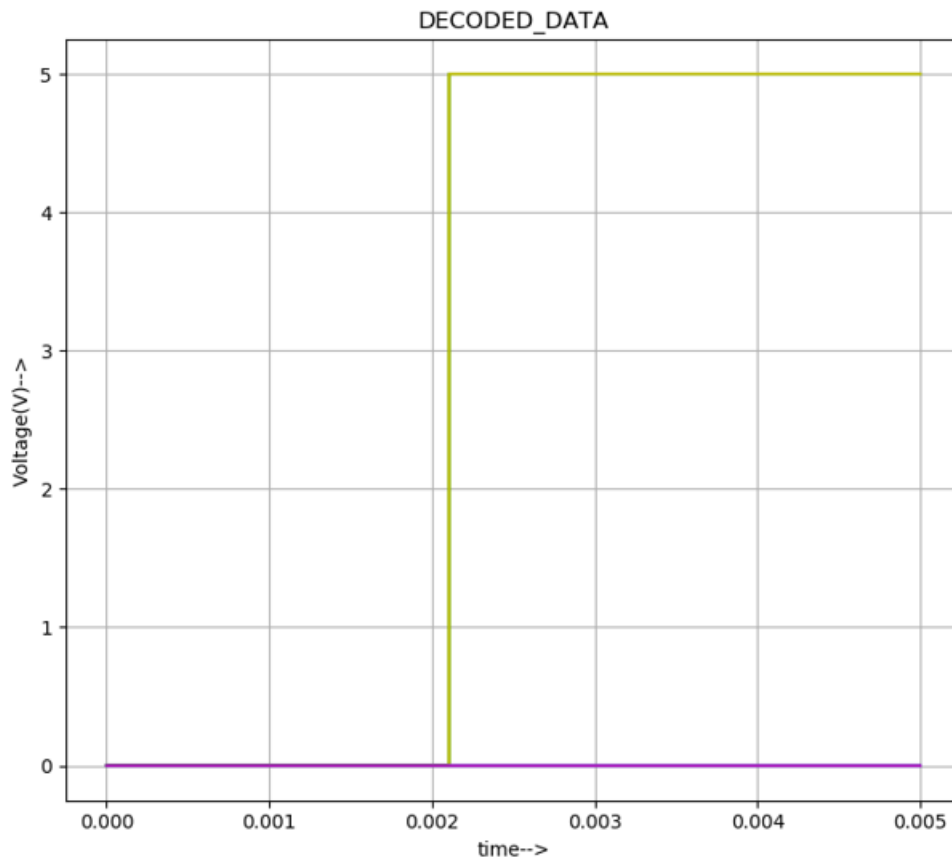


Fig. 7. Decoded least-significant sensor-data nibble (decoded_data[3:0]), consistent with the recorded value 4'h3.

The applied sensor input for this simulation is sensor_data_in = 24'h2A9C63, which breaks down into six transmitted nibbles: [23:20] = 4'h2, [19:16] = 4'hA, [15:12] = 4'h9, [11:8] = 4'hC, [7:4] = 4'h6, [3:0] = 4'h3. Fig. 7 shows this signal settling to a stable logic level once the frame has been fully received, consistent with the recorded value decoded_data[3:0] = 4'h3 = sensor_data_in[3:0]. As the waveform is a two-level (logic-low/logic-high) trace rather than a multi-bit bus plot, the specific hexadecimal value is confirmed from the accompanying simulation record rather than read directly from the trace amplitude.

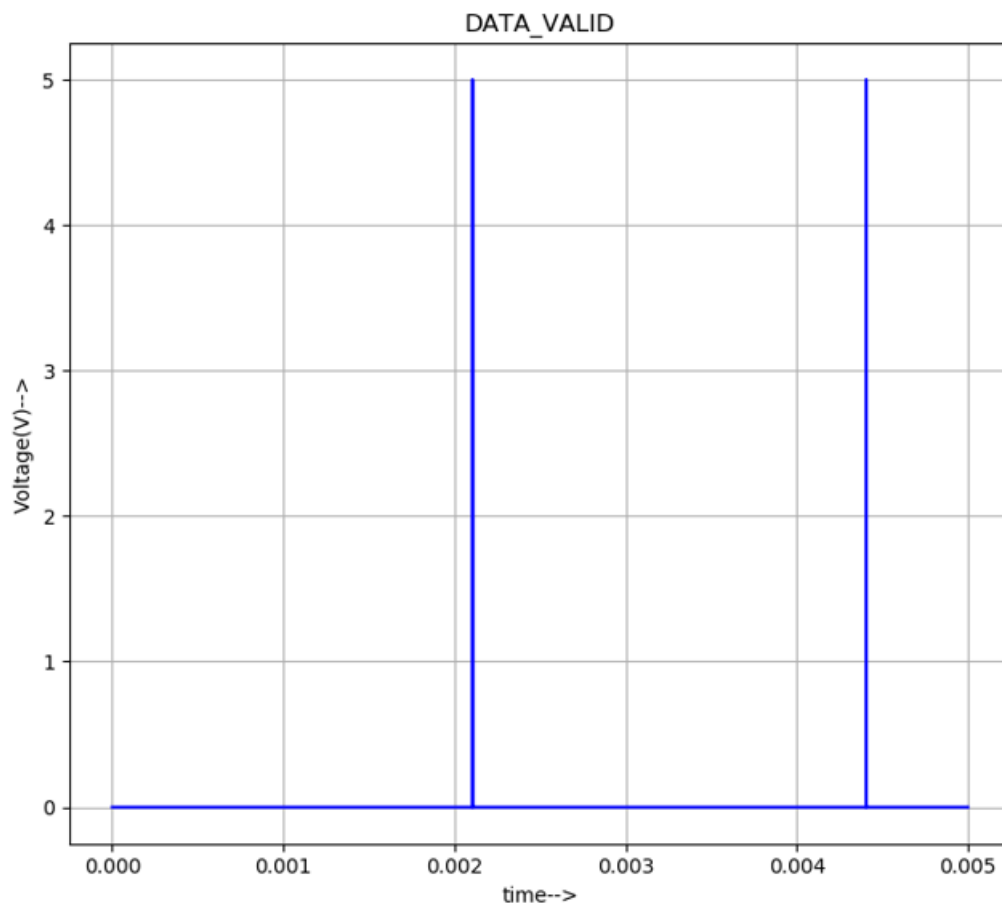
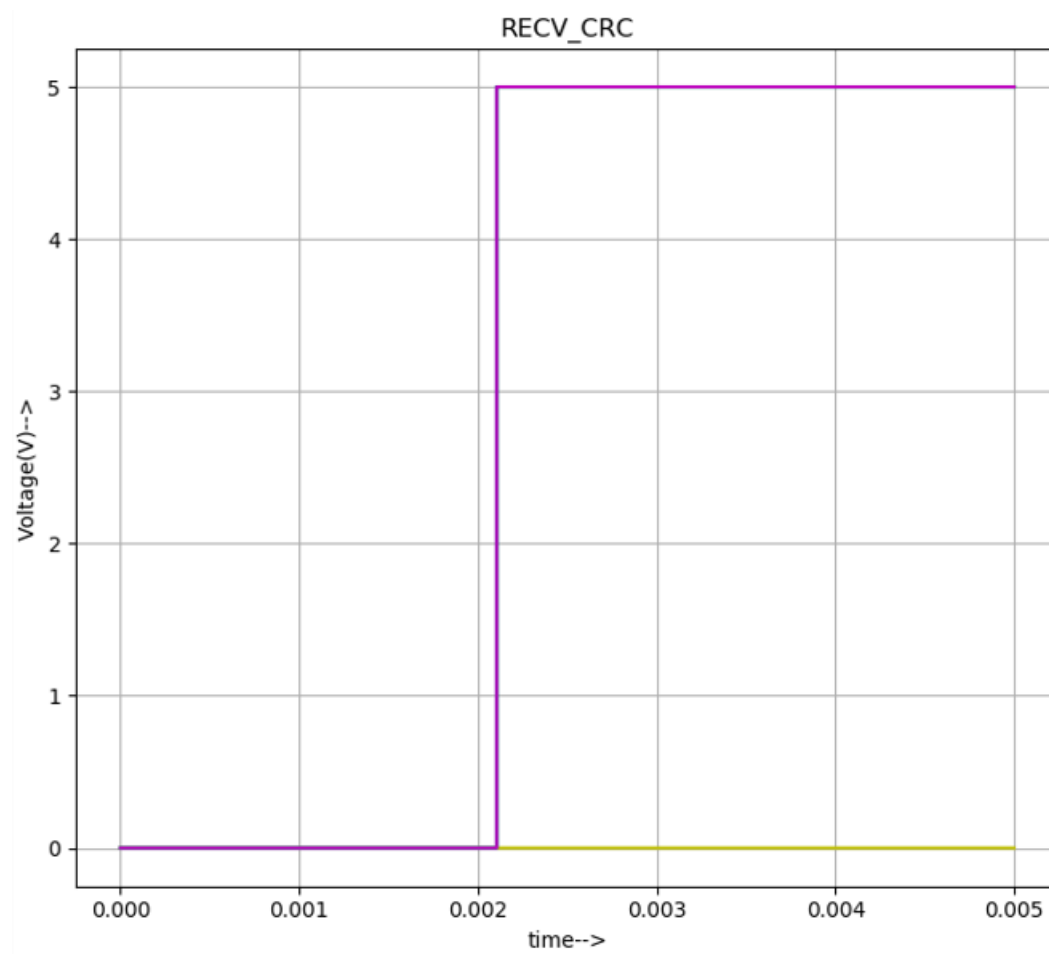
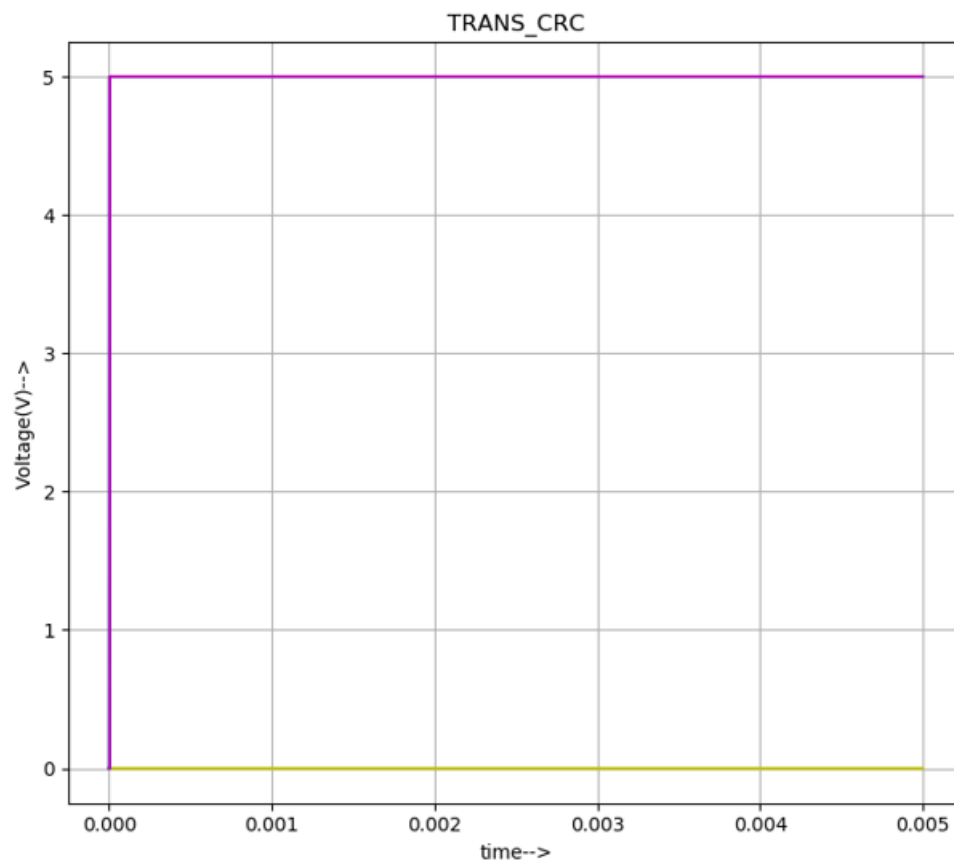


Fig. 8. data_valid waveform confirming completion of frame decoding.

data_valid is asserted by sent_rx.v once a complete frame — synchronization, status, six data nibbles, and CRC — has been received and the decoded outputs are ready to be read. Fig. 8 shows data_valid producing a pulse after the frame has been fully decoded, confirming that the receiver signals frame completion rather than presenting partially decoded data.



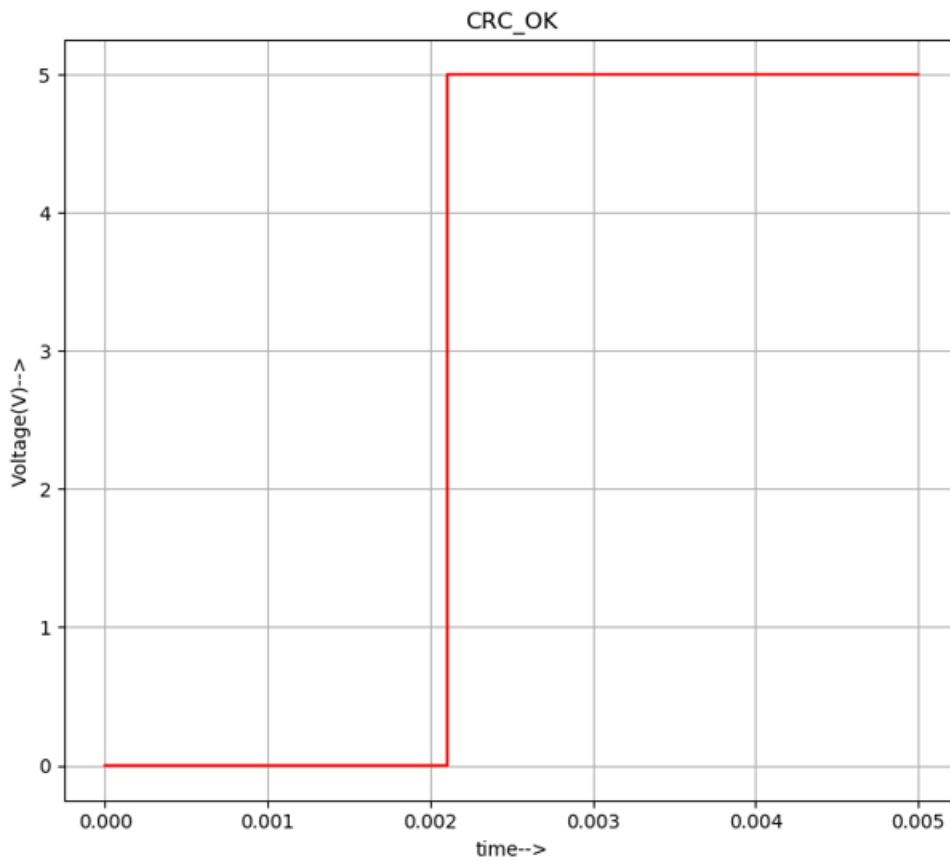


Fig. 9. Transmitted CRC, received CRC, and crc_ok waveforms confirming CRC verification.

sent_tx.v computes a CRC nibble over the transmitted data using its internal sent_crc4 instance, and sent_rx.v independently recomputes the CRC from the received nibbles using its own sent_crc4 instance, comparing the two to produce crc_ok. Fig. 9 shows transmitted_crc and received_crc settling to the same logic level, with crc_ok transitioning to and remaining at logic-high, confirming that the receiver's independently computed CRC matched the transmitted CRC for this frame. For sensor_data_in = 24'h2A9C63 processed through sent_crc4.v (seed 4'h5, polynomial 4'hD, six data nibbles plus the trailing zero nibble), the computed CRC is transmitted_crc = received_crc = 4'hD.

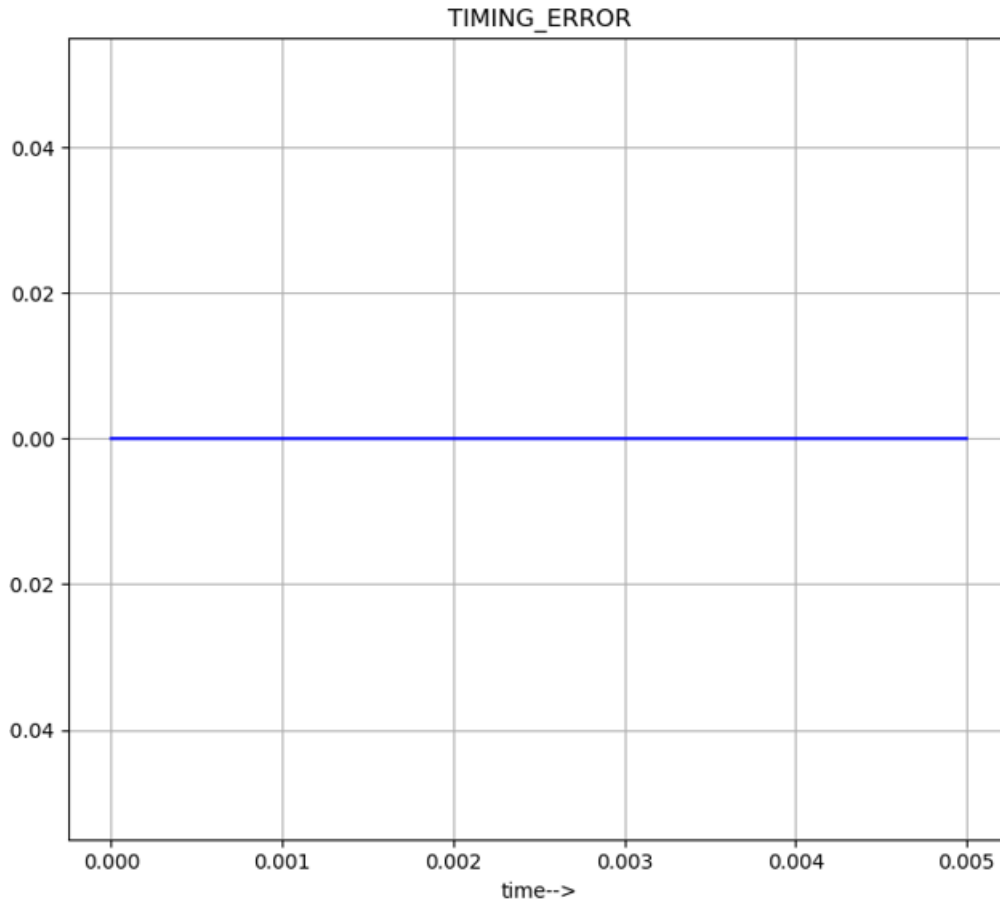


Fig. 10. timing_error waveform confirming absence of timing faults during the tested frame.

sent_rx.v compares each measured inter-edge interval against the valid SENT tick ranges (55–57 ticks for synchronization, 12–27 ticks for a nibble) and flags timing_error if a measured interval falls outside these bounds by more than TOLERANCE_CLKS. Fig. 10 shows timing_error remaining at logic-low for the duration of the simulation, indicating that no timing violation was observed during this communication. No deliberate fault or glitch injection was performed in this test; this result reflects normal, error-free frame timing only.

A dedicated decoded_data[23:0] bus-level waveform capture was not among the supplied images. The 24-bit reconstructed value is confirmed instead from the simulation record: decoded_data = 24'h2A9C63, matching sensor_data_in exactly, and is consistent with the nibble-level result shown in Fig. 7.

Overall Verification Result:

The simulation confirms correct operation of the clock and reset sequencing, generation of a structurally correct SENT frame by the transmitter, synchronization detection, status and sensor-data reconstruction, CRC generation and verification, data-valid signalling, and timing-error monitoring, for the applied stimulus status_in = 4'h3 and sensor_data_in = 24'h2A9C63. These results were obtained through digital simulation of the Verilog design integrated into

eSim via the NgVeri co-simulation interface with ADC/DAC bridge connections; no physical hardware measurements were performed.

Signal / Parameter	Expected / Applied Value	Observed Result	Verification Status
CLK	Continuous clock, TICK_CLKS = 10	Regular, periodic toggling	Verified from waveform
RESET	Assert, then release	Asserted then de-asserted at start	Verified from waveform
status_in	4'h3	Decoded_status=3'b0011	Verified from schematic and simulation output
sensor_data_in	24'h2A9C63	Correct decoded sensor data observed	Verified from schematic and simulation output
SENT_OUT	Sync + 8 nibble pulses	Frame structure observed	Verified from waveform
sync_detected	Pulse per frame	Pulse observed	Verified from waveform
decoded_status	4'b0011 / 4'h3	4'h3	Verified (matches applied status)
decoded_data[3:0]	4'h3	Consistent with 4'h3	Verified from waveform + record
decoded_data[7:4]	4'h6	Consistent with 4'h6	Verified from record
decoded_data[11:8]	4'hC	Consistent with 4'hC	Verified from record
decoded_data[15:12]	4'h9	Consistent with 4'h9	Verified from record
decoded_data[19:16]	4'hA	4'hA	Verified from record
decoded_data[23:20]	4'h2	4'h2	Verified from record
decoded_data (full)	24'h2A9C63	24'h2A9C63	Verified from record
transmitted_crc	4'hD	4'hD	Verified from record
received_crc	4'hD	4'hD	Verified from record
crc_ok	HIGH on match	HIGH	Verified from waveform
data_valid	Pulse on completion	Pulse observed	Verified from waveform
timing_error	LOW (no fault)	LOW	Verified from waveform

Table 5. Final verification table for status_in = 4'h3, sensor_data_in = 24'h2A9C63.

Research Paper/Journal/etc. :

Field	Detail
Title	J2716 SENT — Single Edge Nibble Transmission, Updates and Status
Author	M. Costin, R. Horner, J. Jaegers, D. Daecke, A. Grossmann, B. Opitz, M. ten Brinke, T. Tiek, E. Visser
Page No.	SAE Technical Paper 2011-01-1034 (not paginated as a journal article)
Link	https://doi.org/10.4271/2011-01-1034

Field	Detail
Title	Design and Implementation of Automotive SENT Interface
Author	Jong-Bae Lee, Seongsoo Lee
Page No.	Vol. 21, No. 3, pp. 256–259 (2017)
Link	https://www.kci.go.kr/kciportal/landing/article.kci?arti_id=ART002271356&

Together, these papers provide the technical foundation for the project by covering the SENT frame structure, timing, CRC handling, and automotive SENT interface implementation. The SAE paper supports the protocol and CRC methodology used in the `sent_tx.v` and `sent_rx.v` modules, while the work by Jong-Bae Lee and Seongsoo Lee provides reference for the practical transmitter–receiver interface architecture.

Source/Reference(s) :

- SAE International, “SENT — Single Edge Nibble Transmission for Automotive Applications,” SAE J2716, Rev. 2010 — base protocol specification (frame timing, tick definition, CRC-4 polynomial and seed).
- SENT (protocol), Wikipedia — overview of the SAE J2716 signalling scheme and typical automotive use. [https://en.wikipedia.org/wiki/SENT_\(protocol\)](https://en.wikipedia.org/wiki/SENT_(protocol))
- PEAK-System, “The SENT protocol (SAE J2716) explained.” <https://www.peak-system.com/know-how/blog/the-sent-protocol-sae-j2716-explained-basics-application-and-practical-use-1/>
- Texas Instruments, “Multi-Channel SAE-J2716 (SENT) Decoder Using NHET,” App. Report SPRAB22 — tick-measurement, tolerance and CRC-checking reference. <https://www.ti.com/lit/pdf/sprab22>
- FOSSEE, IIT Bombay — eSim User Manual, NgVeri / mixed-signal Verilog co-simulation chapter. https://static.fossee.in/esim/manuals/eSim_Manual_2.3.pdf
- ngspice documentation — mixed-signal co-simulation with Verilog via Verilator and the `d_cosim` code model. <https://ngspice.sourceforge.io/extras.html>