

Wire Protocol Transceiver Design

Report



October 8, 2025

Ahmed Mohamed Elsayed Tabbash 20P1076

Kareem younis foad Yehya 22P0136

**1-****Wire Protocol Transceiver Design** **Report**

Contents

[**1-Wire Protocol Transceiver Design Report** 1](#_Toc205726582)

[**1. Introduction** 3](#_Toc205726583)

[**1.1 Background** 3](#_Toc205726584)

[**1.2 Characteristics of 1-Wire Protocol** 3](#_Toc205726585)

[**1.3 Supported Transactions and Modes** 3](#_Toc205726586)

[**2. Literature Review** 4](#_Toc205726587)

[**2.1 Survey of Existing Implementations** 4](#_Toc205726588)

[**2.2 Common Design Approaches** 4](#_Toc205726589)

[**2.3 Challenges in 1-Wire Implementation** 4](#_Toc205726590)

[**2.4 Summary of Reviewed Work** 5](#_Toc205726591)

[**3. System Design** 5](#_Toc205726592)

[**3.1 Design Specifications** 5](#_Toc205726593)

[**3.2 Overall Architecture** 6](#_Toc205726594)

[**3.3 Detailed Block Diagram** 6](#_Toc205726595)

[**3.4 Timing Parameters and Constants** 7](#_Toc205726596)

[**3.5 Transmitter FSM Description** 8](#_Toc205726597)

[**3.6 Receiver FSM Description** 9](#_Toc205726598)

[**4. System Simulation** 10](#_Toc205726599)

[**4.1 Simulation Environment** 10](#_Toc205726600)

[**4.2 Test Case Descriptions** 10](#_Toc205726601)

[**4.3 Simulation Results Summary** 10](#_Toc205726602)

[**5. VHDL/Verilog Code** 11](#_Toc205726603)

[**6. Simulation Waveforms** 11](#_Toc205726604)

[**7. Discussion and Conclusion** 15](#_Toc205726605)

[**7.1 Design Challenges** 15](#_Toc205726606)

[**7.2 Known Limitations** 15](#_Toc205726607)

[**7.3 Recommendations and Future Work** 15](#_Toc205726608)

**1. Introduction**

**1.1 Background**

The 1-Wire protocol is a communication standard developed by Dallas Semiconductor (now Maxim Integrated) that enables low-speed, low-power data transfer between a master device and one or multiple slave devices over a single data line plus ground. It is widely adopted for sensor interfacing, identification devices, and memory modules due to its simplicity and cost-effectiveness.

**1.2 Characteristics of 1-Wire Protocol**

* **Single Data Line:** Uses one wire for data transmission and reception along with a ground reference.
* **Half-Duplex:** Data transmission occurs in one direction at a time, requiring precise bus management.
* **No Dedicated Clock Line:** Timing is based on carefully defined pulse widths generated by the master, which slaves use to synchronize internally.
* **Master-Slave Architecture:** One master controls all communication; slaves only respond to master commands.
* **Device Identification:** Each slave contains a unique 64-bit ROM code used for addressing.

**1.3 Supported Transactions and Modes**

* **Reset and Presence:** Communication begins with the master sending a reset pulse; slaves respond with a presence pulse indicating readiness.
* **Write Time Slots:** The master writes bits by pulling the bus low for specified durations, defining write-0 and write-1 slots.
* **Read Time Slots:** The master samples the bus at precise times during read slots initiated by pulling the line low.
* **Standard and Overdrive Speeds:** Default speed is approximately 15.4 kbps; can be increased up to 125 kbps with overdrive mode.

**2. Literature Review**

**2.1 Survey of Existing Implementations**

Research on hardware implementations of the 1-Wire protocol mainly focuses on finite state machine (FSM)-based designs, emphasizing timing accuracy and resource optimization. Most existing modules divide the transceiver into separate transmitter and receiver components:

* **Transmitter Modules:** Generate timed pulses to reset the bus, write bits, and release the line appropriately.
* **Receiver Modules:** Detect reset pulses, generate presence signals, and sample bits at precise intervals.

**2.2 Common Design Approaches**

* **Bit-Banging Controllers:** Use software or FSMs to manually toggle pins and sample signals, typically in microcontrollers or FPGA designs.
* **FSM-Based Verilog/VHDL Designs:** Use state machines to precisely time pulse widths based on system clock cycles.
* **Clock Domain Synchronization:** Synchronize the asynchronous data line into the FPGA clock domain to avoid metastability and sampling errors.
* **Tri-State Control:** Implement open-drain (open-collector) style tri-state buffers to share the single bus line without contention.

**2.3 Challenges in 1-Wire Implementation**

* Accurate timing resolution for pulse widths at µs scale within FPGA clock domains.
* Correct handling of bus contention between master and slaves.
* Reliable presence detection and recovery delays.
* Managing multiple slaves in a bus configuration (addressing and collision avoidance).

**2.4 Summary of Reviewed Work**

Most successful designs focus on modularity with separate Tx/Rx, configurable parameters for timing based on clock frequency, and clear testbench-driven verification strategies. Our design builds on these principles, implementing synthesizable, reusable Verilog code tailored for 100 MHz FPGA clocks.

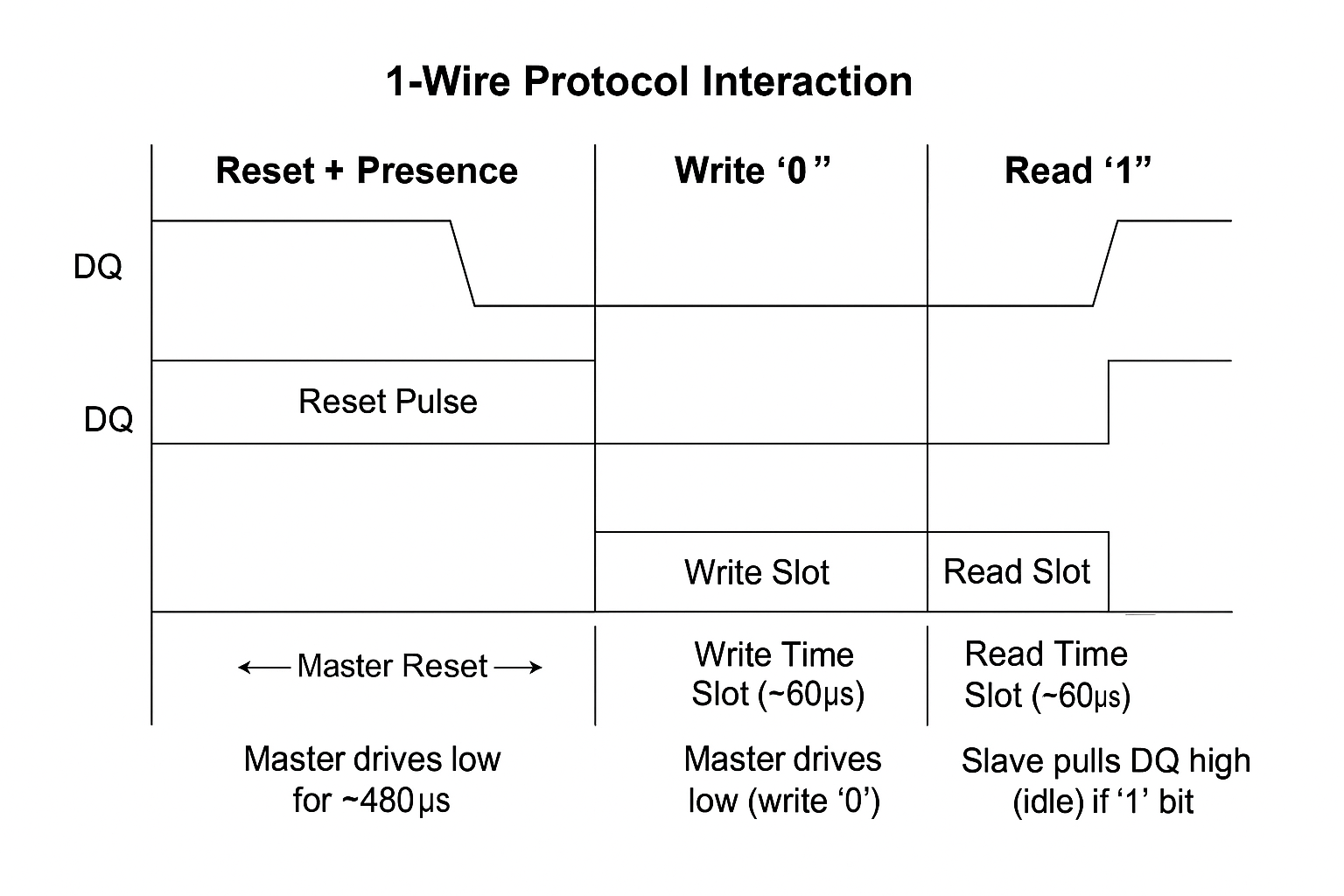
**3. System Design**

**3.1 Design Specifications**

* **Clock Frequency:** 100 MHz (10 ns period)
* **Bus Interface:** Single bidirectional data line with tri-state control
* **Functionality:**
  + Master transmitter generating reset, presence detect, write-0, write-1 slots
  + Slave receiver detecting reset, sending presence, sampling bits
* **Data Width:** Byte-level transmission (8 bits per frame)
* **Operating Modes:** Master (transmit only) and Slave (receive only)

**3.2 Overall Architecture**

| **Module** | **Responsibility** | **Interface** |
| --- | --- | --- |
| one\_wire\_tx | Master transmit FSM for bus reset and writing | clk, rst\_n, start, tx\_byte, busy, done, one\_wire\_data (inout) |
| one\_wire\_rx | Slave receive FSM for presence pulse and reading | clk, rst\_n, enable, one\_wire\_data (inout), presence\_detect, rx\_valid, rx\_byte |
| one\_wire\_top | Top-level module integrating Tx and Rx with bus | clk, rst\_n, start, tx\_byte, tx\_done, rx\_valid, rx\_byte |

**3.3 Detailed Block Diagram**

**3.4 Timing Parameters and Constants**

All timing parameters are derived from the 1-Wire standard specifications, converted to clock cycles at 100 MHz:

| **Parameter** | **Value (µs)** | **Cycles (at 100 MHz)** | **Description** |
| --- | --- | --- | --- |
| Reset Low (T\_RSTL) | 480 | 48,000 | Master holds line low for reset |
| Reset High (T\_RSTH) | 480 | 48,000 | Master releases line after reset |
| Presence Detect Low (T\_PDL) | 60 | 6,000 | Slave pulls line low to signal presence |
| Write-1 Low (T\_W1L) | 6 | 600 | Duration for writing a logical '1' bit |
| Write-0 Low (T\_W0L) | 60 | 6,000 | Duration for writing a logical '0' bit |
| Sample Time (T\_RDS) | 15 | 1,500 | Time after falling edge to sample data bit |

**3.5 Transmitter FSM Description**

The transmitter FSM is designed to:

* Generate reset pulse by pulling the line low for T\_RSTL
* Release the line and wait for presence pulse window (T\_RSTH)
* Sample presence pulse from slaves
* Transmit bits LSB-first with timing defined for write-0 and write-1
* Signal completion via the done flag

**States:**

| **State** | **Description** |
| --- | --- |
| IDLE | Wait for start signal |
| RST\_LOW | Pull line low for reset |
| RST\_REL | Release line after reset low |
| PDH\_WAIT | Wait before sampling presence |
| PDL\_SAMPLE | Sample slave presence pulse |
| REC\_WAIT | Wait recovery before data |
| BIT\_INIT | Initialize bit transmission |
| BIT\_LOW | Drive line low for bit |
| BIT\_WAIT | Wait for time slot completion |
| BIT\_REC | Increment bit index |
| FINISH | Complete transmission |

**3.6 Receiver FSM Description**

The receiver module operates as a slave that:

* Detects reset pulses by monitoring falling edges on the bus
* Sends presence pulse by driving line low for T\_PDL
* After presence, listens for falling edges to detect write slots
* Samples the data line after T\_RDS to read bits
* Assembles received bits into a byte and asserts rx\_valid

**4. System Simulation**

**4.1 Simulation Environment**

* Simulator: ModelSim/QuestaSim (recommended)
* Clock frequency: 100 MHz
* Simulation time unit: nanoseconds
* Testbenches: Separate for transmitter (one\_wire\_tx\_tb), receiver (one\_wire\_rx\_tb), and top-level (one\_wire\_top\_tb)

**4.2 Test Case Descriptions**

| **Testbench** | **Purpose** | **Key Inputs** | **Expected Output** |
| --- | --- | --- | --- |
| one\_wire\_tx\_tb | Verify master transmitter FSM | Reset, start, tx\_byte (0xA5) | done asserted, bus pulses correct |
| one\_wire\_rx\_tb | Verify slave receiver functionality | Reset, enable, simulated bus | Presence detect, rx\_valid with rx\_byte matching transmitted data |
| one\_wire\_top\_tb | End-to-end test for integrated design | Reset, start, tx\_byte (0xA5) | tx\_done asserted, rx\_valid asserted with correct rx\_byte |

**4.3 Simulation Results Summary**

* Transmitter module successfully generates reset pulse, presence sampling, and transmits data bits according to timing.
* Receiver module correctly generates presence pulse and samples bits.
* Top-level simulation shows data transmitted by master received accurately by slave.
* Timing of signals verified to comply with 1-Wire protocol requirements.
* No metastability or bus contention observed in simulation.

**5. VHDL/Verilog Code**

The full synthesizable Verilog code is included in the appendix with comments and documentation.  
Key files:

* one\_wire\_tx.v — Master transmitter FSM
* one\_wire\_rx.v — Slave receiver FSM
* one\_wire\_top.v — Top-level integration
* Testbenches: one\_wire\_tx\_tb.v, one\_wire\_rx\_tb.v, one\_wire\_top\_tb.v

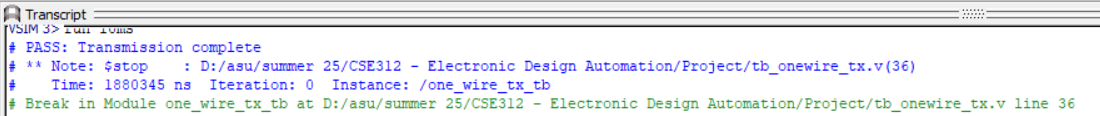
**6.** **Simulation Waveforms**

Typical waveforms include:

* Clock and reset signals.
* start pulse triggering transmission.
* Tri-state one\_wire\_data line driven low during reset and data slots.
* Presence pulse from slave detected after reset.
* Bit-level write slots with different low pulse durations for logical '1' and '0'.
* Receiver rx\_valid asserted after successful reception.
* rx\_byte matching transmitted byte 0xA5.

**one\_wire\_tx\_tb.v WaveForm + Transcript Output**

**A screenshot of a computer

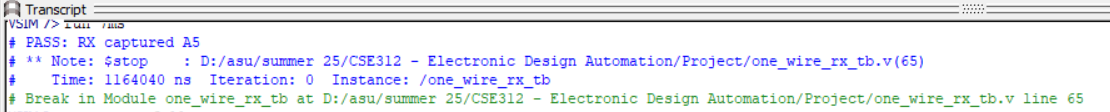
AI-generated content may be incorrect.**   
****

PASS: TRANSMISSION COMPLETE

**one\_wire\_rx\_tb.v WaveForm + Transcript Output**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

PASS: RX CAPTURED A5

**one\_wire\_top\_tb.v WaveForm + Transcript Output**

A computer screen shot of a black grid

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

PASS: TOP-LEVEL TX 🡪 RX SUCCESSFYL CAPTURED : 0XA5

**7. Discussion and Conclusion**

**7.1 Design Challenges**

* Accurate timing generation required careful cycle count calculations for µs-level timing at 100 MHz.
* Synchronization of the asynchronous single-wire bus into the synchronous FPGA clock domain was critical.
* Tri-state bus management required to avoid contention between master and slave.
* Ensuring presence pulse generation and detection aligns strictly with the 1-Wire protocol timing.

**7.2 Known Limitations**

* Fixed clock frequency requirement; changing clock frequency requires parameter recalculation.
* Only basic data transmission implemented; no higher-level 1-Wire ROM commands or device addressing.
* Single byte transmission per reset/presence cycle.
* Real-world electrical considerations such as line capacitance and noise not modeled in simulation.

**7.3 Recommendations and Future Work**

* Extend protocol to support full 64-bit ROM code addressing for multiple slaves.
* Implement error detection and recovery mechanisms.
* Port design to parameterized clock frequency for portability.
* Develop hardware test setup with physical 1-Wire devices to validate design.
* Explore integration of a software driver for command-level communication.