BLUECONNECT.AI

# BCAI-SPW-IP:

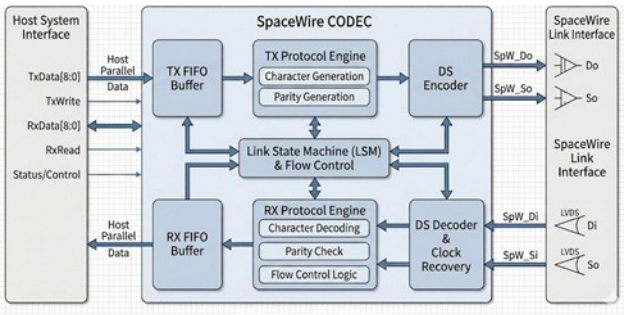
# High-Performance

# SpaceWire Codec IP Core

**Datasheet & Specifications | Version 2.2**

## 1. Overview

The BCAI-SPW-IP is a high-throughput, ultra-reliable SpaceWire (SpW) CODEC IP core, guided by the **ECSS-E-ST-50-12C** standard. Designed for demanding aerospace technologies, test-equipment bridges, and NewSpace Edge-AI applications, the core provides a seamless bridge between a host system and physical SpaceWire LVDS transceivers. Featuring advanced Clock Domain Crossing (CDC) immunity and continuous clock-recovery stabilization, it is optimized for AMD/Xilinx, Lattice, Microchip, NanoExplore, CologneChip Programmable Logic Device architectures.




## 2. Key Features

- **ECSS-E-ST-50-12C Compliance:** implements the SpaceWire state machine, initialization and data protocol handling, flow control, and parity generation/checking.
- **Zero-Gap Look-Ahead Shifting:** Utilizes instantaneous token loading to guarantee an unbroken, continuous physical clock on the wire, preventing Setup/Hold timing violations in remote receivers.
- **Leak-Proof Clock Domain Crossing Flow Control:** Employs relevant combinatorial and sequential logic to safely absorb high-frequency clock domain jitter without generating false credits or triggering spurious disconnects.
- **Continuous Stateful Parity:** Implements flawless running parity across all token types (Data, EOP, EEP, FCT, NULL).
- **Asymmetric Boot Resolution:** Native support for independent AutoStart and LinkStart modes to eliminate timer phase-mismatch delays on initialization.
- **Vendor Agnostic:** Written in synthesizable VHDL-93. Deployable on Lattice (ICE40, CrossLink), AMD/Xilinx (Spartan-7, Artix-7) or Microchip FPGA, etc..

Technical support : blueconnect.ai@proton.me

www.blueconnect.ai

## 3. Port Interface (Pinout)

| Port Name | Direction | Width | Clock Domain | Description |
|---|---|---|---|---|
| sys_rst | IN | 1 | Async | Asynchronous system reset. |
| spw_clk | IN | 1 | spw_clk | Main SpaceWire transmission clock. |
| spw_di / spw_si | IN | 1 | N/A | SpaceWire Data/Strobe inputs (from LVDS RX). |
| spw_do / spw_so | OUT | 1 | spw_clk | SpaceWire Data/Strobe outputs (to LVDS TX). |
| tx_speed_sel | IN | 2 | spw_clk | Clock divider selection (1x, 2x, 3x, 6x). |
| link_start | IN | 1 | spw_clk | ECSS LinkStart command. |
| auto_start | IN | 1 | spw_clk | ECSS AutoStart command. |
| link_disable | IN | 1 | spw_clk | ECSS LinkDisable command. |
| reg_max_credit | IN | 6 | spw_clk | Maximum credit configuration. |
| link_status | OUT | 3 | spw_clk | Current FSM state (000=ErrorReset to 101=Run). |
| error_flags | OUT | 4 | spw_clk | Latched hardware error register (See Section 4). |
| host_tx_clk | IN | 1 | host_tx_clk | Clock for host transmitting data. |
| host_tx_data | IN | 9 | host_tx_clk | 9-bit data: [8]=Control Flag, [7:0]=Payload. |
| host_tx_wr | IN | 1 | host_tx_clk | Write enable for TX FIFO. |
| host_tx_full | OUT | 1 | host_tx_clk | Write full indicator for TX FIFO. |
| host_rx_clk | IN | 1 | host_rx_clk | Clock for host reading data. |
| host_rx_data | OUT | 9 | host_rx_clk | 9-bit data: [8]=Control Flag, [7:0]=Payload. |
| host_rx_rd | IN | 1 | host_rx_clk | Read enable for RX FIFO. |
| host_rx_empty | OUT | 1 | host_rx_clk | Read empty indicator for RX FIFO. |

## 4. Hardware Error Register (error_flags)

The IP core features a dedicated 4-bit latched error register. When an anomaly occurs, the corresponding bit is permanently asserted high (1) until the host clears it by issuing a link_disable command. This allows the host software to perform post-mortem diagnostics on link failures.

- **Bit [0] - Disconnect Error (sig_disconnect):** Asserted when the physical link goes completely silent. The core monitors the recovered clock and triggers this flag if no edges are detected for exactly 850 ns (ECSS compliant inactivity threshold).
- **Bit [1] - Parity Error (sig_parity_err):** Asserted when a dynamic XOR running parity mismatch is detected on the incoming Data/Strobe lines, instantly crashing the FSM to ErrorReset and flushing buffers to prevent corrupted data processing.
- **Bit [2] - Escape Error (sig_esc_err):** Asserted when an invalid Escape (ESC) token sequence is received (e.g., an ESC token followed by another ESC token instead of an FCT, EOP, or EEP).
- **Bit [3] - Credit Error (sig_credit_err):** Asserted if the internal flow control accumulator exceeds the maximum allowed 56 outstanding credits, indicating a severe desynchronization or misbehavior from the remote transmitting node.

## 5. Performance & Specifications

The SpaceWire protocol requires 10 physical bits to transmit 1 byte of payload (8 Data bits + 1 Parity bit + 1 Control bit). The absolute theoretical maximum payload throughput is always Clock_Frequency * 0.8.

- Operating Frequency: Scalable from Tx clock = 10 MHz to 60+ MHz,  100MHz+ Rx clock on Lattice FPGAs, and up to **200 MHz Rx/Tx** on AMD/Xilinx Artix-7/Spartan-7 architectures.
- Maximum Theoretical Payload @ Tx clock = 60 MHz: 48.00 Mbps
- Validated Sustained Payload @ Tx clock = 60 MHz: 45.65 Mbps (Exceptional **95.1% Link Efficiency**).
- Memory Safety & Flow Control: Capable of sustaining massive continuous data bursts (tested to 16+ Mbyte and 39 Mega Bytes per second - over USB High-Speed) without credit underflow.

## 6. Resource Utilization & Silicon Footprint

The BCAI-SPW-IP core boasts an ultra-low silicon footprint, making it ideal for integration into size, weight, and power (SWaP) constrained environments such as micro-satellites, drones, and edge-computing platforms. When synthesized for a standard AMD/Xilinx 7-Series architecture (e.g., Spartan-7 or Artix-7), the complete SpaceWire node—including the Link FSM, zero-gap transmission engine, continuous parity tracking, and cross-domain flow controllers - consumes approximately **800 LUTs - SpW codec standalone** with direct FIFOs interfaces. A typical implementation would lead to **1249 Look-Up Tables (LUTs) and using 6 BRAM** - e.g. with USB high-speed synchronous parallel interface implementing macro-command encoding/decoding with additional bidirectional full-duplex buffers.

This design does not implement complex mechanisms like oversampling; instead, it uses a standard Data/Strobe XOR technique for RX clock recovery.

Consequently, the core leaves the vast majority of the FPGA's logic cells and Block RAM routing available for the user's primary application, AI acceleration, or multi-node routing fabrics.