University of North Carolina at Charlotte
Department of Electrical and Computer Engineering

Qorvo Radio

LO & LE – No Expander Attempted Fix

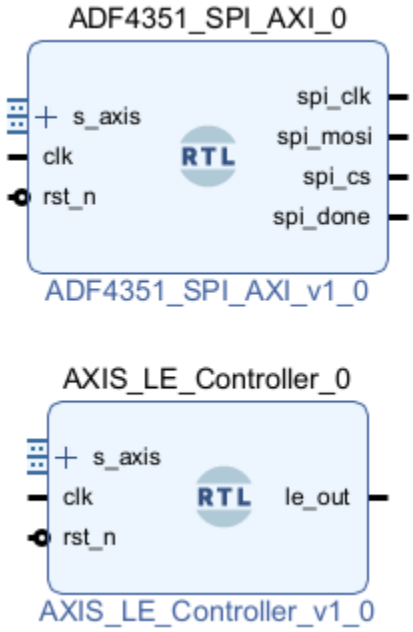

Figure 1-1 LO & LE Attempted Fix Module Diagram

Contents

Module Overview 2

Module Specifications 2

1. Inputs 2
2. Outputs 2

Operational Description 3

1. Debounce Lock Detection 3
2. Periodic Tick Generation 3

3. Continuous Lock Status Streaming.....**Error! Bookmark not defined.**
Integration and Usage 3

Module Overview

As part of the final integration phase of the QORVO_RADIO project, the team attempted to incorporate a custom-designed RF front-end PCB. However, this effort was hindered by a voltage incompatibility between the 5V MCP23S17 I/O expander and the 3V3-tolerant ADF4351 local oscillator (LO). To address this issue and avoid overvoltage damage, the team designed a corrective workaround: two AXI-Stream-driven VHDL modules—ADF4351_SPI_AXI and AXIS_LE_Controller—along with updated C control code, were developed to safely decouple the SPI data path and Load Enable (LE) control using PMOD headers driven at 3.3V logic levels.

These modules attempt to directly replace the expander's function in driving register configuration and LE pulses by creating a minimal, FPGA-native SPI control system, operating independently of the failed expander path.

Module Specifications

1. Inputs

Common:

- `clk (std_logic)`: System clock. Drives logic across both modules.
- `rst_n (std_logic)`: Active-low reset.

ADF4351_SPI_AXI:

- `s_axis_tvalid (std_logic)`: Valid signal for incoming register data via AXI Stream.
- `s_axis_tdata (std_logic_vector(31 downto 0))`: 32-bit ADF4351 register word (R0–R5).

AXIS_LE_Controller:

- `s_axis_tvalid (std_logic)`: Triggers LE pulse once register sequence is complete.
- `s_axis_tdata (std_logic_vector(31 downto 0))`: Dummy data for stream compliance.

2. Outputs

ADF4351_SPI_AXI:

- `s_axis_tready (std_logic)`: Indicates readiness to receive a new register.

- `spi_clk (std_logic)`: SPI clock output.
- `spi_mosi (std_logic)`: SPI data output.
- `spi_cs (std_logic)`: SPI chip select (optional; default low).
- `spi_done (std_logic)`: One-cycle pulse confirming SPI transfer completion.

AXIS_LE_Controller:

- `le_out (std_logic)`: LE pulse output routed to dedicated PMOD or LO test point.

Operational Description

1. AXI Stream SPI Register Shifting

The SPI module accepts a 32-bit register via AXI Stream, shifts data out on spi_mosi MSB-first, and generates clock pulses via spi_clk at a fixed divided rate. After 32 bits, a spi_done pulse is generated, signaling completion. This logic allows precise, serialized transmission of ADF4351 register configurations and replaces the expander's MOSI path using native FPGA logic.

2. Load Enable Pulse Timing

- Once register writes are complete, the LE Controller triggers a delayed, fixed-duration high pulse on le_out. This is routed directly to the ADF4351 LE test point, isolated from other logic. The delay ensures all SPI writes settle before the pulse. This isolated control allows the system to meet ADF4351 timing requirements while bypassing the 5V expander output.

Integration and Usage

While the QORVO_RADIO project achieved major milestones in system design, one critical integration goal was not fully realized: incorporating the custom-designed RF front-end PCB into the complete signal chain. Despite successful fabrication of the PCB and implementation of the control architecture, technical limitations in voltage compatibility and timing alignment prevented full end-to-end validation of the front-end configuration.

The original design called for the use of an MCP23S17 SPI I/O expander to control enable lines on the front-end, including the critical Load Enable (LE) line for the ADF4351. However, the expander required 5V input logic and interpreted 4V+ signals as high—conflicting with the LO's 3.3V maximum digital input rating. Driving MOSI and LE through the expander posed a risk of overvoltage damage and unreliable behavior.

To mitigate this, the team bypassed the expander entirely for LO communication. Instead, the SPI and LE modules were implemented on separate PMODs operating at 3.3V. This configuration enabled safe register communication while preserving LE pulse control. Power was manually patched to necessary expander output lines using jumpers for partial testing.

Despite these fixes, the front-end never produced a valid lock or RF output. Multiple VHDL and C implementations were tested, with variations in SPI clocking, register sequences, and LE timing. However, limited test time after board arrival prevented comprehensive debugging. Lock confirmation via MUXOUT was never reliably observed.

Recommendations for Future Work

- Validate LO SPI transactions independently using MUXOUT probing and debug tools.
- Consider removing the expander from the critical path entirely.
- Route fixed enables using direct 3.3V IO or through-hole jumpers.
- Redesign control logic to minimize routing and decouple analog/digital planes.
- Focus on confirming register transactions in isolation before full integration.

These modules and the corresponding C driver serve as a documented and reusable foundation for future teams continuing integration efforts. While the full RF front-end signal chain was not realized, this corrective architecture provides a safer, simplified pathway forward—built with clearer insight into the risks of mixed-voltage integration.