Verification of a SPI Protocol

(Serial Peripheral Interface)

Description:

The spi_serializer module is responsible for serializing data from a FIFO buffer into a SPI-compatible serial data. The module operates as follows:

Data Serialization:

- During the SHIFT state, the module shifts out the data from the shift register bit by bit, synchronized with the SPI clock (sclk). The most significant bit (MSB) of the data is shifted out first. The mosi output provides the serialized data stream compatible with the SPI protocol.

Clock Generation:

- clk_div toggles every clock cycle, effectively dividing the clock frequency by 2, this generates a slower SPI clock sclk.
- sclk is toggled at a rate determined by ClockDivider.
- sclk_enable signal will be used to control when the sclk should be toggled. It should be active during the SHIFT state.

Done Signal:

- The done signal is set high when all data bits have been shifted out and all of them were display in the MOSI output indicating the serialization process is complete, so we can move to IDLE state where is turn off.

Reset Logic:

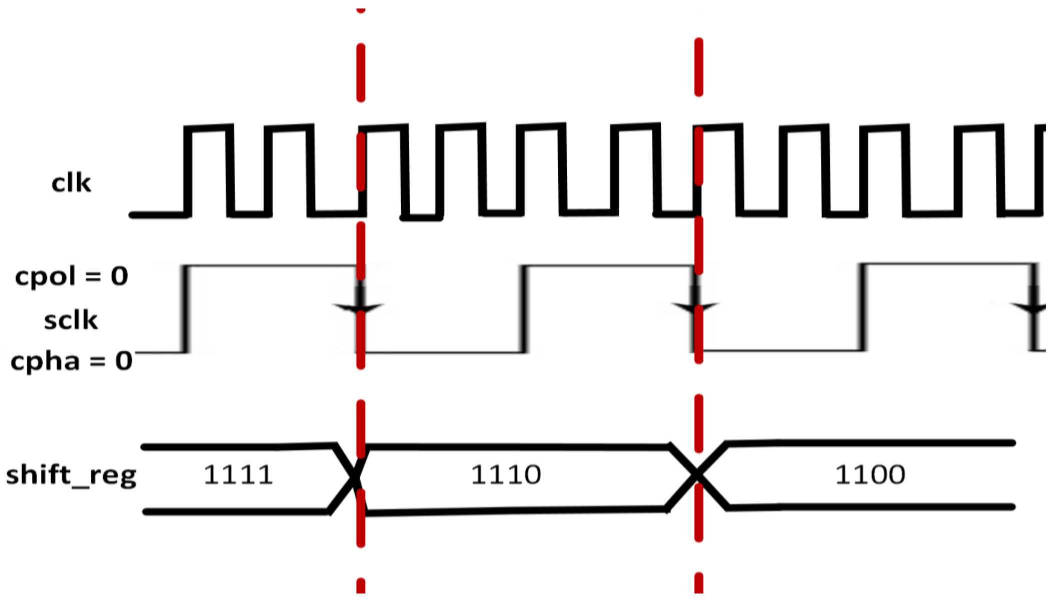
- This signal has the same level of hierarchy as the empty signal, once any of them are activated, during the next positive edge of clk state goes to IDLE and all our internal signals are initialized at 0.

MOSI Signal:

- Data Serialization: The mosi signal transmits the serialized bits of readData from the FIFO as an output.
- Bit-by-Bit Transmission: Data is sent out one bit at a time, synchronized with the SPI clock signal on the falling edge of sclk so that the data is stable (and can be sampled by a possible slave module on the rising edge of sclk).

SPI Mode:

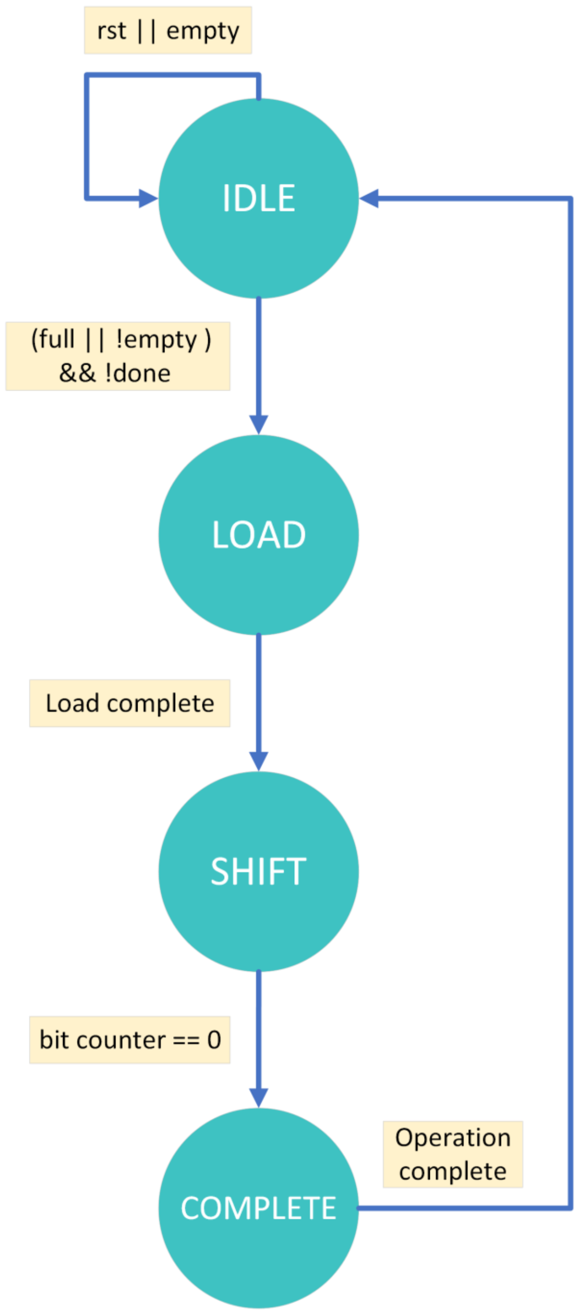
Mode 0 (CPOL = 0, CPHA = 0): CPOL = 0: The idle state of the clock is low (0). CPOL = 1: The idle state of the clock is high (1). CPHA = 0: Data is sampled on the leading (first) clock edge and shifted out on the trailing (second) clock edge. CPHA = 1: Data is sampled on the trailing (second) clock edge and shifted out on the leading (first) clock edge.	• The clock (SCLK) starts at a low level when idle. • When falling edge, the transmitter shifts left the next bit of read_data onto the mosi line. • When rising edge, the receiver samples the bit present on the mosi line (*There is no explicit sampling mentioned in this code).



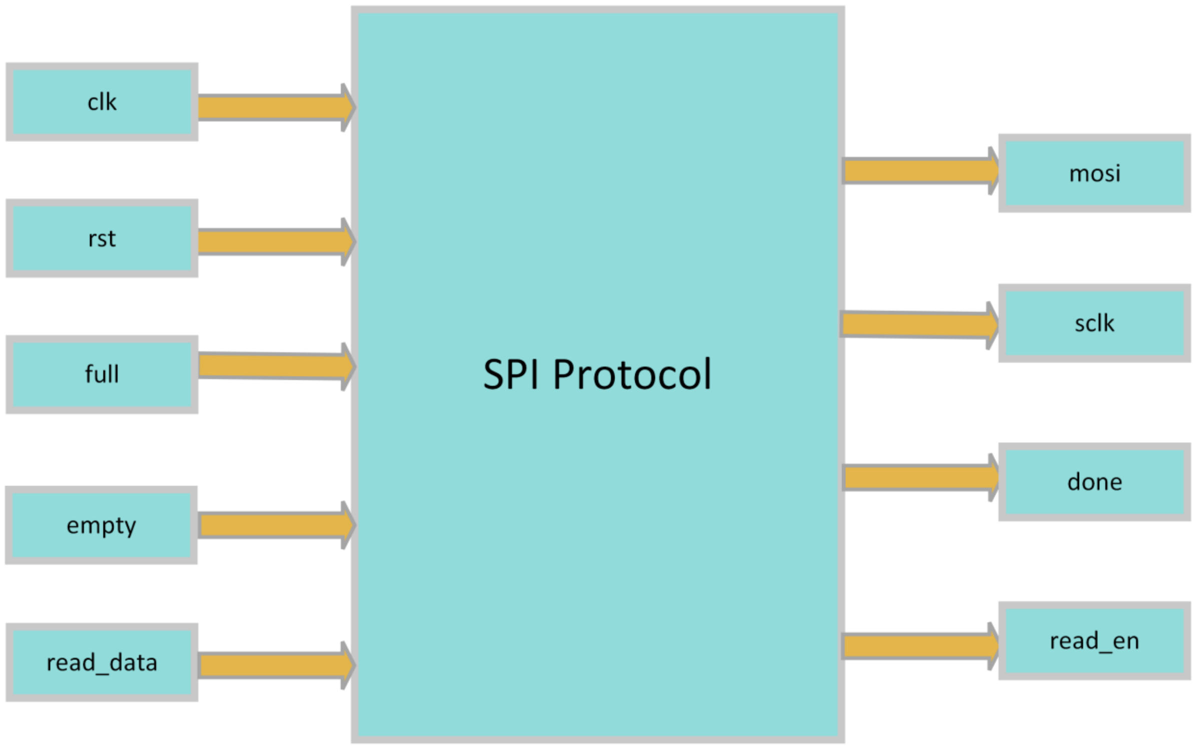
FSM states are defined as follows:

- IDLE: Initial state where the module waits for the condition to start serialization, in this case, when the FIFO is full or not empty and done is not active.
- LOAD: State where the module loads data from the FIFO, read_en is active only in this state.
- SHIFT: State where the module shifts out serially the bits of the read_data every falling edge of the sclk.
- COMPLETE: State indicating the completion of the serialization process, the done signal is activated in this state.

State Machine Diagram.



Block Diagram:



Input Definitions

- clk : Clock signal to synchronize the operations.
- rst : Reset signal to initialize the module.
- full: Signal indicating that the FIFO is full, triggering the start of serialization.
- empty: Signal indicating that the FIFO is empty, preventing serialization when active.
- readData ([DataWidth-1:0]): Data to be serialized.

Output Definitions.

- sclk : SPI clock signal.
- mosi : SPI Master Out Slave In data line (serial data output).
- done: Signal indicating that the serialization process is complete.

Internal Components:

- State Variables: state and next_state used to track the state of the finite state machine.
- Shift Register: shift_reg used to hold the data being serialized before being sent out.
- Bit Counter: bit_counter used to keep track of the number of bits shifted out during serialization.
- SCLK Generator: Generates the SPI clock by toggling sclk.

Formal Verification:

Properties

N#	Name	Category	Status

Verify the correct write operation.

Properties defined to cover this spec:

Assumptions:

Assertions:

Covers:

Verify the correct read operation:

Properties defined to cover this spec:

Assumptions:

Assertions:

Covers

Verify the correct full signal:

Assertions:

Covers:

Verify the correct empty signal:

Assertions:

Covers:

Verify the correct simultaneous write and read operation:

Properties defined to cover this spec:

Covers:

Results:

Bug fixing:

Solution: