Project Status Report – RMC75E Test Bench

Date: June 30, 2023

Subject: Progress Update: Formatting Data Output for SSITop Module

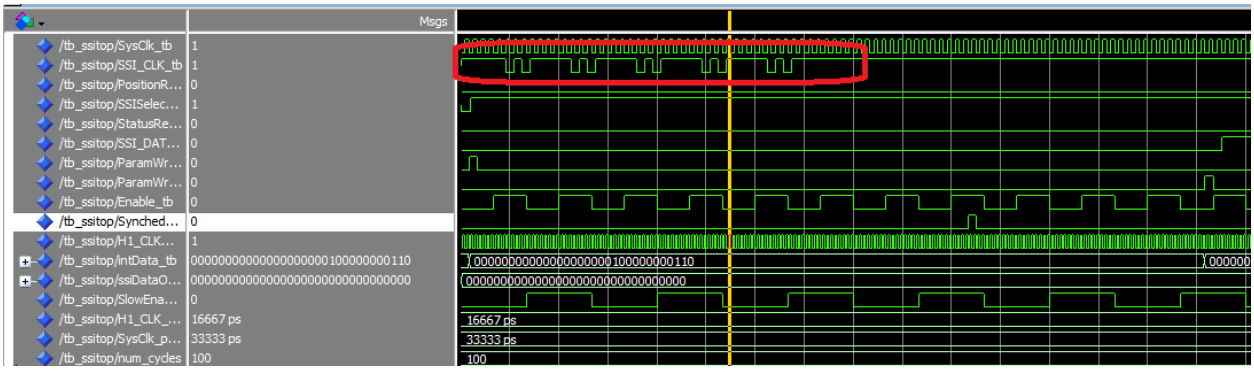
Problem Summary:

The primary focus of this phase of the project is to accurately format the data output for the SSITop Module in the RMC75E modular motion controller. Despite successful configuration and initiation of the SSI_CLK signal, a significant hurdle given the number of dependent conditions and signals required, we have encountered an issue where the data output line fails to display the correct output vector after the deassertion of PositionRead and StatusRead signals.

Progress to Date:

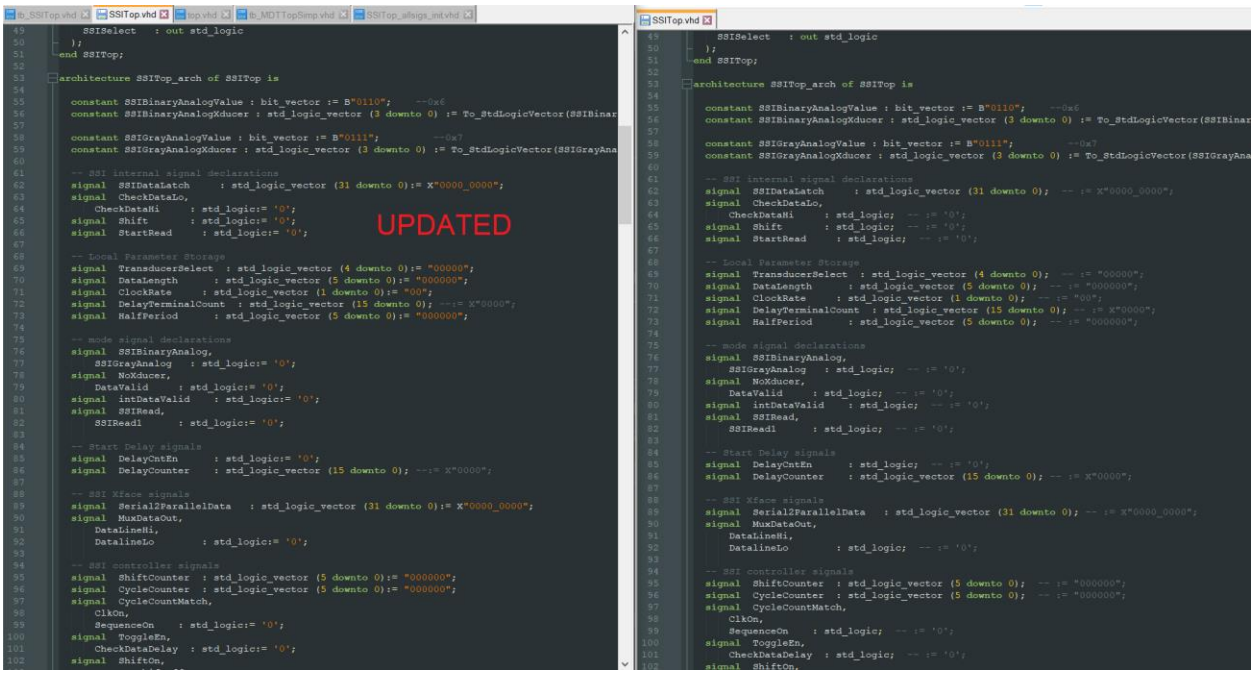
Achievements include the successful initialization of the SSI_CLK signal, with a number of negative pulses being displayed, an endeavor that necessitated managing approximately 16 other signals. Additionally, we modified the PositionRead and StatusRead signals to activate only after the second SynchedTick pulse, staggered by one 60MHz clock cycle to prevent simultaneous occurrences. A significant amount of massaging has been done to get the needed conditions to simulate properly. I also altered the source code so that all but a few signals are initialized with starting values. This wasn't so much altering the source code as it was just uncommenting the initial value declarations.

Fig. 1



Here we see the negative pulses on the SSI_CLK line that are required for correct function of the source module.

Fig. 2



(Note the uncommenting of the initial values set in the architecture declaration.)

Next Steps:

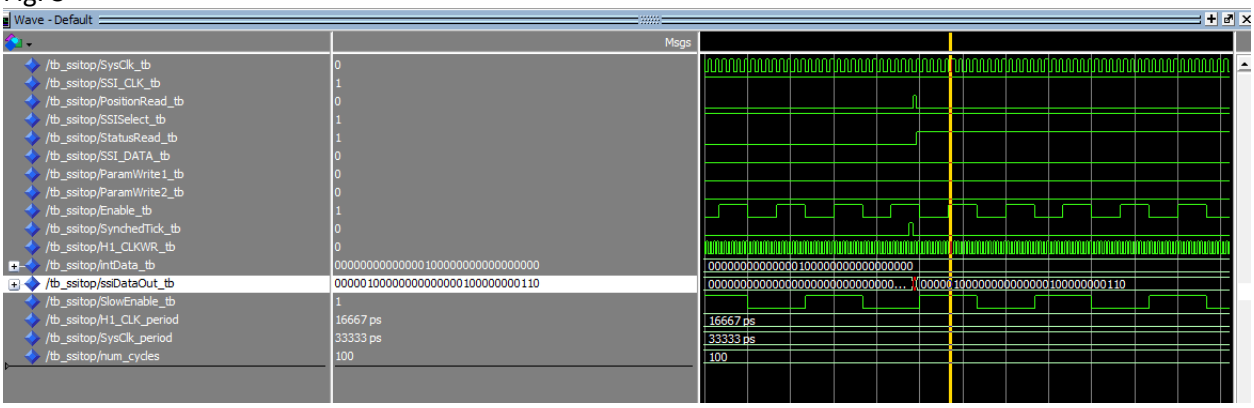
1. Run a new simulation and confirm that SSI_CLK behavior is corrected.

2. Once SSI_CLK is confirmed to be functioning correctly, assess the effect on the data output line, specifically checking the output vector after PositionRead and StatusRead signals have ended.

3. Determine why the 32-bit vector data transfer initiated by the parameter write calls is not persisting on the data output line (ssiDataOut).

4. Document the identified solutions and the resulting behavior of the system for future reference and to aid in similar troubleshooting scenarios.

Fig. 3



Note that in this earlier iteration of the simulation, the 32-bit vector writes are showing up on the ssiDataOut line, but only as long as the either the statusRead or positionRead signals is active. Once those signals are de-asserted, the output vector does not persist, and reverts to a zero vector.

Module Overview:

The SSITop interfaces with an SSI transducer. Below is an outline of expected output from some important signals based on the code:

1. ssiDataOut: This output signal depends on the active state of PositionRead or StatusRead and the state of SSISelect.

- If PositionRead and SSISelect are high, the value on ssiDataOut equals the value on SSIDataLatch.
- If StatusRead and SSISelect are high, ssiDataOut is a 32-bit value comprising several fields.
- If neither PositionRead nor StatusRead are high (or SSISelect is low), ssiDataOut will be X"0000_0000".

2. SSI_CLK: The clock signal for the SSI data transfer, it toggles every time ToggleEn and ClkOn are both high; otherwise, it retains its last value. It also sets to high during the StartRead process.

3. SSISelect: It will be high when the selected transducer is either SSIBinaryAnalog or SSIGrayAnalog, otherwise it will be low.

4. NoXducer: It indicates a break in the wire or connection to the transducer (DataLineHi or DataLineLo is high), otherwise it will be low.

The output signal ssiDataOut is primarily of interest, which is entirely dependent on the SSI_CLK signal to properly output data. It depends on several other signals and internal latches. Expected values under different conditions include:

1. When PositionRead is '1' and SSISelect is '1': ssiDataOut equals SSIDataLatch.

2. When StatusRead is '1' and SSISelect is '1': ssiData

Out is a 32-bit word composed of multiple fields.

3. Otherwise, ssiDataOut will be "0000_0000".

When StatusRead is '1' and SSISelect is '1', the ssiDataOut sets to an elaborate 32-bit word composed of several fields determined by the current state of signals such as NoXducer, DataValid, ClockRate, Datalength, and TransducerSelect.

The primary goal remains to resolve the undefined signal issue to ensure the correct data output for the SSITop Module. The 32-bit vector ssiDataOut is instrumental for outputting either status information or data from the latch, depending on the values of PositionRead, SSISelect, and StatusRead. The structure of ssiDataOut when outputting status information is as follows:

• 31 downto 20: "0" (12 bits)

• 19: NoXducer

• 18: DataValid

• 17 downto 16: "00" (2 bits)

• 15 downto 14: ClockRate

• 13 downto 8: DataLength “001000”

• 7 downto 5: "000" (3 bits)

• 4 downto 0: TransducerSelect “00110”

SSI_CLK: Signal Initialization Procedure Expanded:

To understand the signals' role in setting up the conditions required for proper initialization of **SSI_CLK**, let's go through each signal and its relationship to **SSI_CLK**:

1. **ClkOn**: This signal enables the external SCLK functions. It must be asserted for the clock signal generation to occur.
2. **ShiftOn**: This signal enables the shifting of external data into a holding register. It must be asserted for the data to be shifted in during the clock signal generation.
3. **TurnShiftOff**: This signal indicates when the shifting process should be turned off. When asserted, it stops the shifting of data.
4. **PreTurnShiftOff**: This signal determines when the **TurnShiftOff** signal will be asserted. If this signal is asserted, **TurnShiftOff** will be asserted after the current shift operation is complete.
5. **intSSI_CLK**: This is the clock generated for the SSI data transfer. It toggles at twice the frequency of the desired **SSI_CLK** output signal.
6. **ToggleEn**: This signal must run at twice the frequency of **SSI_CLK** and is used to toggle the **SSI_CLK** output. When asserted, **SSI_CLK** will toggle.
7. **Enable**: This signal indicates whether the SSI controller is enabled. If not enabled, the sequence is considered over, and the **CycleCounter** will be reset to zero.

8. **SequenceOn**: This signal determines whether the SSI read sequence is active. It is active when **SequenceOn** is asserted and **CheckDataLo** is deasserted.
9. **CycleCounter**: This signal represents the current count of the SSI read sequence. It increments when **Enable** is asserted.
10. **HalfPeriod**: This signal determines the value at which **CycleCounter** should match for **CycleCountMatch** to be asserted. It represents half of the desired period for **SSI_CLK**.
11. **CycleCountMatch**: This signal indicates when the **CycleCounter** matches the **HalfPeriod** value. It serves as a condition for toggling **ToggleEn** and generating the clock signal.
12. **ShiftCounter**: This signal counts the number of bits that have been shifted in during the SSI read cycle.
13. **StartRead**: This signal is a pulse that initiates the SSI read cycle when asserted.
14. **CheckDataHi**: This signal is used to check and latch the data line status before data transfer.
15. **CheckDataLo**: This signal is used to check and latch the data line status after data transfer.
16. **CheckDataDelay**: This signal introduces a delay before sampling the line break detection bit. It ensures that the data line has enough time to stabilize before checking for a line break.

The correct initialization and coordination of these signals are crucial for generating the desired **SSI_CLK** waveform. Issues such as **SSI_CLK** not oscillating correctly may arise if these signals are not properly synchronized or configured. It's essential to review the interdependencies among these signals and ensure they are correctly set and coordinated to achieve the desired behavior of **SSI_CLK**.

To ensure that **SSI_CLK** is defined correctly, the following signals must be in the specified state:

1. **ClkOn**: This signal must be asserted (**'1'**) to enable the external SCLK functions. It allows the clock signal generation to occur.
2. **ShiftOn**: This signal must be asserted (**'1'**) to enable the shifting of external data into a holding register during the clock signal generation.
3. **TurnShiftOff**: This signal should be deasserted (**'0'**) to keep the shifting process active. When asserted (**'1'**), it stops the shifting of data.
4. **PreTurnShiftOff**: This signal does not directly affect the state of **SSI_CLK** but determines when the **TurnShiftOff** signal will be asserted. If **PreTurnShiftOff** is asserted (**'1'**), it indicates that **TurnShiftOff** will be asserted after the current shift operation is complete.
5. **intSSI_CLK**: This clock signal is generated for the SSI data transfer. It should toggle at twice the frequency of the desired **SSI_CLK** output signal.
6. **ToggleEn**: This signal should run at twice the frequency of **SSI_CLK** and is used to toggle the **SSI_CLK** output. It needs to be asserted (**'1'**) to enable the toggling of **SSI_CLK**.
7. **Enable**: This signal should be asserted (**'1'**) to enable the SSI controller. If it is deasserted (**'0'**), the sequence is considered over, and the **CycleCounter** will be reset to zero.

8. **SequenceOn**: This signal should be asserted (**'1'**) to indicate that the SSI read sequence is active. It should be active when **SequenceOn** is asserted and **CheckDataLo** is deasserted.
9. **CycleCounter**: This signal represents the current count of the SSI read sequence. It should increment when **Enable** is asserted.
10. **HalfPeriod**: This signal represents half of the desired period for **SSI_CLK**. The **CycleCounter** should match this value for **CycleCountMatch** to be asserted.
11. **CycleCountMatch**: This signal indicates when the **CycleCounter** matches the value of **HalfPeriod**. It serves as a condition for toggling **ToggleEn** and generating the clock signal.
12. **ShiftCounter**: This signal counts the number of bits that have been shifted in during the SSI read cycle. It does not directly affect the state of **SSI_CLK**.
13. **StartRead**: This signal should be asserted (**'1'**) to initiate the SSI read cycle.
14. **CheckDataHi**: This signal is used to check and latch the data line status before data transfer. It does not directly affect the state of **SSI_CLK**.
15. **CheckDataLo**: This signal is used to check and latch the data line status after data transfer. It does not directly affect the state of **SSI_CLK**.
16. **CheckDataDelay**: This signal introduces a delay before sampling the line break detection bit. It does not directly affect the state of **SSI_CLK**.

By ensuring that these signals are properly coordinated and in the specified states, we can achieve the correct definition of the **SSI_CLK** signal.

In conclusion, progress me not always be as fast as we would like, but remains relatively consistent. We have overcome the challenge of running the SSI_CLK signal and isolated the issue with the undefined ShiftOn signal and other undefined signals. Our next steps involve troubleshooting the ssiDataOut and ensuring that the data output line correctly receives and displays the output vector after the PositionRead and StatusRead signals have ended.