*Project: RMC75E FPGA TEST BENCH*

*Module: MDTSimpDataOut.vhd*

*Author: Satchel Hamilton*

*Company: Delta Motion*

*Date: 7/13/2023*

*Last updated: July 26, 2023*

Problem statement: There is no data showing up on the MDTSimpDataOut output line which it should receive from the PositionRead signal. We need to ensure that all the conditions for this to happen are being met inside of the test bench.

Here are some conditions we have identified:

To ensure that mdtSimpDataOut receives data from PositionRead, the following conditions need to be met:

1. PositionRead signal must be asserted ('1').

2. SSISelect signal must be deasserted ('0').

3. Either StartStopRisingEdge, StartStopFallingEdge, or PWM signals must be asserted ('1').

4. The RetPulseDelayDone signal must be asserted ('1').

5. All other signals (StatusRead, ParamWrite, SysReset, etc.) should be deasserted ('0').

Additionally, the internal state machine in the MDTTopSimp entity controls the sequence of operations. It transitions through different states to generate and handle the MDT counters' count sequence. The state machine should be properly configured to ensure the desired behavior and enable the transfer of data to mdtSimpDataOut.

The RetPulseDelayDone signal indicates that a return pulse has not been received within 50us of the start of the interrogation pulse, which triggers the setting of the NoXducer bit. The combination of these conditions will enable the transfer of data to mdtSimpDataOut.

mdtSimpDataOut can receive data only after the 50us mark has been reached: The RetPulseDelayDone signal, which is set when the Delay counter reaches the value specified by the RetPulseDelay constant (indicating that a return pulse has not been received within 50us), triggers the setting of the NoXducer bit. This condition, in combination with other factors, enables the transfer of data to mdtSimpDataOut. Therefore, mdtSimpDataOut can receive data only when RetPulseDelayDone is asserted, indicating that the return pulse delay has exceeded 50us.

There are likely more conditions that need to be met. These signals may have their own conditions that need to be met, and so on, creating a complex tree of conditions that must be met.

You may now revise the test bench to ensure the proper conditions are met. Be smart. You need to think logically. For instance, we have determined that mdtTopSimpDataOut cannot receive data until this happens:

The RetPulseDelayDone signal indicates that a return pulse has not been received within 50us of the start of the interrogation pulse, which triggers the setting of the NoXducer bit. The combination of these conditions will enable the transfer of data to mdtSimpDataOut.

So, obviously, you should not try to send data from PositionRead to mdtDataSimpOut until after this condition has been met. Be very mindful of your directives and what you learned from looking at the source code. Be sure not to break logic already present in the test bench. Revise the test bench to solve the problem as defined in the problem statement now.   
  
More in-depth analysis of required conditions:

Based on the code, we can infer several conditions that need to be met for our test case to succeed:

1. **Setting up MDTTopSimp**: This entity expects certain input signals to be initialized. The signals that are relevant to our goal are:
   * **PositionRead** should be '1'
   * **SSISelect** should be '0'
   * **StartStopRisingEdge** or **StartStopFallingEdge** or **PWM** should be '1'

These conditions will ensure that the value **X"000" & MDTPosition(19 downto 0)** is assigned to the **mdtSimpDataOut** output.

1. **Assigning values to StartStopRisingEdge, StartStopFallingEdge, and PWM**:

The values of **StartStopRisingEdge**, **StartStopFallingEdge**, and **PWM** are dependent on **TransducerSelect(1 downto 0)**. We need to assign the correct values to **TransducerSelect(1 downto 0)** via **intData(6 downto 0)** in the **ParamWrite** process. The respective conditions are:

* + **StartStopRisingEdge** will be '1' when **TransducerSelect(1 downto 0) = StartStopRisingEdgeXducer**
  + **StartStopFallingEdge** will be '1' when **TransducerSelect(1 downto 0) = StartStopFallingEdgeXducer**
  + **PWM** will be '1' when **TransducerSelect(1 downto 0) = PWMXducer**

1. **State Machine Behavior**:

The module contains a state machine that influences the output. For our goal, we need to be in the correct state to assign the position data. The state machine transitions are influenced by **SynchedTick60**, **RetPulseDelayEnable**, **CounterOverFlowRetrigger**, **MDTSelect**, **RisingACountEnableLatch**, **PWMMagnetFaultLatch**, **RetPulseDelayDone**, **RisingANegEdgeFound**, and **RisingAPosEdgeFound**. Here are some key conditions we need to set:

* + To reach **s1** state: **SynchedTick60** and **not RetPulseDelayEnable** or **CounterOverFlowRetrigger** must be '1' and **MDTSelect** should be '1'.
  + **s4** and **s5** states have important conditions related to the **RisingACountEnableLatch** signal and **PWMMagnetFaultLatch**. The control of these signals will need to be set accordingly.