*Project: RMC75E FPGA TEST BENCH*

*Module: MDTTopSimp.vhd*

*Author: Satchel Hamilton*

*Company: Delta Motion*

*Date: 7/13/2023*

*Last updated: July 13, 2023*

Module Overview:

MDTTopSimp with SPWM, Start/Stop, and SSI type transducers. It operates using three different incoming clock signals running at a rate of 60MHz. The module has various input and output signals and includes a state machine to control the count sequence of the MDT counters and handle the start and termination of the count cycle.

Inputs:

- `SysReset`: Main system reset signal.

- `H1\_CLK`: 60MHz clock for the MDT interface.

- `H1\_CLKWR`: CPU clock for reads and writes.

- `H1\_CLK90`: 60MHz clock with a 90-degree phase shift for MDT counters.

- `SynchedTick60`: Synchronized tick signal at 60MHz.

- `intDATA`: Input data signal (32 bits).

- `PositionRead`: Signal indicating a position read operation.

- `StatusRead`: Signal indicating a status read operation.

- `ParamWrite`: Signal indicating a parameter write operation.

- `M\_RET\_DATA`: Input data signal for M\_RET\_DATA.

- `SSISelect`: Signal indicating the selection of SSI transducer.

Outputs:

- `mdtSimpDataOut`: Output data signal from the MDT (32 bits).

- `M\_INT\_CLK`: Output internal clock signal.

- `SSI\_DATA`: Output SSI data signal.

Internal Signals and Processes:

- `DelayDone`: Marks the end of the 1.6us Interrogation pulse.

- `RetPulseDelayDone`: Marks the setting of the NoTransducer flag.

- `PWMMagnetFault`: Indicates a PWM magnet fault condition.

- `PWMMagnetFaultLatch`: Latches the PWMMagnetFault or itself and not ClearCounter at each rising edge of H1\_CLK.

- `RisingEdgeA`: Process that shifts the value of RisingA array at every rising edge of H1\_CLK.

- `RisingAPosEdgeFound` and `RisingANegEdgeFound`: Indicate positive or negative edge in RisingA.

- `RisingACountEnable` and `RisingACountDisable`: Enable and disable the RisingACountEnableLatch based on certain conditions.

- `CountRA`: 18-bit synchronous counter that counts at every rising edge of H1\_CLK.

- `LeadingCountDecode` and `TrailingCountDecode`: Determine whether to add or subtract counts based on the value of Edge and other signals.

- `LeadingCount` and `TrailingCount`: Latch the counts at the start and end of the count cycle.

- `RisingEdgeB` and `FallingEdgeB`: Similar to RisingEdgeA and FallingEdgeA, triggered on H1\_CLK90.

- `XfrToH1\_CLK`: Process that combines bits from RisingB, FallingA, and FallingB to form Edge at every rising edge of H1\_CLK.

- `MDTPosition`: 20-bit register that receives the result of CountRA and the Leading and Trailing counts.

Entity and Architecture:

The entity `MDTTopSimp` lists the inputs and outputs for the module. The architecture `MDTTopSimp\_arch` defines the internal implementation of the module.

Usage:

This module is used as the MDT interface in the RMC75E modular motion controller, providing an interface and control for communication with PWM, Start/Stop, and SSI type transducers. The module handles the counting sequence, detects pulses, and calculates position data based on the received signals.

Notes and Recommendations:

1. Ensure all periods and timing values are defined as functions of the `H1\_CLK\_PERIOD` to synchronize the simulation properly.

2. Generate a proper testbench to verify the functionality of the `MDTTopSimp` module. The testbench should cover different scenarios and test cases, including valid and invalid input combinations, edge cases, and corner cases. It is recommended to simulate various transducer types (SPWM, Start/Stop, SSI) and test different operations such as position read, status read, and parameter write.

3. Carefully review the state machine implementation in the `MDTTopSimp\_arch` architecture. Ensure that all possible states and transitions are correctly defined and handled. Verify that the state machine accurately controls the count sequence and handles start and termination conditions.

4. Pay attention to the synchronization and timing requirements of the input and output signals. Make sure that the signals are properly synchronized with the clock signals to avoid timing violations and ensure correct data processing.

5. Validate the output signals of the `MDTTopSimp` module against the expected results for different input scenarios. Verify that the output data (`mdtSimpDataOut`), internal clock (`M\_INT\_CLK`), and SSI data (`SSI\_DATA`) are generated correctly and align with the defined specifications.

6. Perform functional and performance testing to ensure the module operates as intended within the specified constraints. Test the module with different clock frequencies, input data patterns, and transducer configurations to validate its robustness and reliability.

7. Verify the module's behavior during reset conditions (`SysReset`). Ensure that the module properly initializes and returns to a known state after a reset signal is asserted.

8. Consider implementing appropriate error handling mechanisms or fault detection logic within the module. This can help identify and handle exceptional conditions or faults that may occur during operation, such as PWM magnet faults or invalid input data.

9. Document the module's interface, functionality, and usage instructions in a clear and comprehensive manner. This documentation will assist other developers or users in understanding and utilizing the `MDTTopSimp` module correctly.

10. Perform integration testing with the larger system to ensure proper integration and compatibility with other modules or components. Validate the module's behavior within the overall system architecture and confirm that it meets the system requirements and performance expectations.

11. Continuously monitor and review the module's performance, considering possible optimizations or improvements. Gather feedback from testing and usage to identify any potential issues or areas for enhancement and address them accordingly.