*Project: RMC75E FPGA TEST BENCH*

*Module: Analog.vhd*

*Author: Satchel Hamilton*

*Company: Delta Motion*

*Date: 7/13/2023*

*Last updated: July 26, 2023*

Overview:

The **Analog** module is a wrapper that instantiates three other modules: **StateMachine**, **Serial2Parallel**, **DataBuffer**. Each module serves a specific purpose and contributes to the overall functionality of the design.

1. **StateMachine**: This module represents a state machine that controls the ADC conversion process. It receives inputs such as system reset (**SysReset**), system clock (**SysClk**), enable signal (**SlowEnable**), synchronized tick signal (**SynchedTick**), loop time (**LoopTime**), and ADC chip select (**ExpA\_CS\_L**). It generates outputs to control the ADC clock (**ExpA\_CLK**), enable signal for the shift register (**Serial2ParallelEN**), clear signal for the shift register (**Serial2ParallelCLR**), and end of conversion indicator (**WriteConversion**).
2. **Serial2Parallel**: This module converts serial data to parallel data. It receives inputs such as system clock (**SysClk**), synchronized tick signal (**SynchedTick**), control axis data (**CtrlAxisData**), ADC data (**ExpA\_DATA**), enable signal for conversion (**Serial2ParallelEN**), clear signal for conversion (**Serial2ParallelCLR**), and address for parallel data (**S2P\_Addr**). It generates parallel data output (**S2P\_Data**).
3. **DataBuffer**: This module implements a data buffer to store the converted analog data. It receives inputs such as system reset (**SysReset**), write clock (**H1\_CLKWR**), system clock (**SysClk**), enable signal (**SlowEnable**), synchronized tick signal (**SynchedTick**), synchronized 60Hz tick signal (**SynchedTick60**), read control signals (**AnlgPositionRead0**, **AnlgPositionRead1**, **ExpA0ReadCh0**, **ExpA0ReadCh1**, **ExpA1ReadCh0**, **ExpA1ReadCh1**, **ExpA2ReadCh0**, **ExpA2ReadCh1**, **ExpA3ReadCh0**, **ExpA3ReadCh1**), write conversion signal (**WriteConversion**), address for parallel data (**S2P\_Addr**), and parallel data input (**S2P\_Data**). It generates the analog data output (**DataOut**), ADC chip select signal (**ExpA\_CS\_L**), and ADC clock signal (**ExpA\_CLK**).
4. **Analog**: This module serves as the top-level entity that instantiates the other modules. It provides the input and output ports required for the entire design.

Signals:

1. `ExpA\_CS\_L` (output signal):

- Type: std\_logic

- Description: This signal is an active-low chip select signal for the ExpA module.

- Usage: It is used to enable or disable communication between the ExpA module and other components in the system.

2. `ExpA\_CLK` (output signal):

- Type: std\_logic

- Description: This signal is a clock signal for the ExpA module.

- Usage: It provides the timing reference for the ExpA module's internal operations and synchronizes its data transfers.

3. `CtrlAxisData` (input signal):

- Type: std\_logic\_vector (1 downto 0)

- Description: This signal is a 2-bit vector representing control axis data.

- Usage: It carries information related to the control axis, such as direction or control mode. The exact meaning and interpretation depend on the design and context in which it is used.

4. `ExpA\_DATA` (input signal):

- Type: std\_logic\_vector (7 downto 0)

- Description: This signal is an 8-bit vector representing data for the ExpA module.

- Usage: It carries the actual data to be processed or communicated by the ExpA module. The interpretation of the data depends on the specific requirements and functionality of the module.

State Machine:

1. State Encoding: The state encoding is defined using the **STATE\_TYPE** array. Each state is assigned a binary value using the **std\_logic** type. Here are the defined states:

constant StartState : STATE\_TYPE :="000"; constant SampleHoldState : STATE\_TYPE :="001"; constant ConvertState : STATE\_TYPE :="011"; constant ConvertWaitState1 : STATE\_TYPE :="010"; constant ConvertWaitState2 : STATE\_TYPE :="110"; constant ConvertDoneState : STATE\_TYPE :="111"; constant IncConvCountState : STATE\_TYPE :="101"; constant InterConvDelayState : STATE\_TYPE :="100";

The **State** signal is of type **STATE\_TYPE** and represents the current state of the state machine.

1. Delays between Conversions: There are different delay times between conversions based on the **LoopTime** signal. The delays are defined as follows:
2. constant eighth\_ms\_interconversion\_delay : std\_logic\_vector (10 downto 0) := "00000000101"; -- 5 counts constant quarter\_ms\_interconversion\_delay : std\_logic\_vector (10 downto 0) := "00001000000"; -- 60 counts constant half\_ms\_interconversion\_delay : std\_logic\_vector (10 downto 0) := "00010110100"; -- 180 counts constant one\_ms\_interconversion\_delay : std\_logic\_vector (10 downto 0) := "00110011100"; -- 412 counts constant two\_ms\_interconversion\_delay : std\_logic\_vector (10 downto 0) := "01101110001"; -- 881 counts constant four\_ms\_interconversion\_delay : std\_logic\_vector (10 downto 0) := "11100011100"; -- 1820 counts

* These delays are used to spread the data samples over the entire control loop period.

1. Process for State Transitions: The **StateMachine\_arch** process describes the state transitions and conditions. Let's analyze it step by step:
   * The process is sensitive to the **SysClk** signal, which represents the system clock.
   * On a rising edge of the clock, the process evaluates the state transitions and updates the corresponding signals.
   * If **SysReset** or **SynchedTick** signals are asserted, the state is set to the **StartState** (reset state).
   * If **SlowEnable** signal is asserted, the process enters the state machine logic.
   * The state machine logic is implemented using a **case** statement based on the current state.
   * Each state has different conditions and actions associated with it.
   * The state transitions and corresponding assignments are as follows:
   * **StartState**:
     + If **Converting** is asserted, the **ExpA\_CS\_L** signal is set to '0' (converter chip select active), and the state transitions to **SampleHoldState**.
     + If not converting, the **ExpA\_CS\_L** signal is set to '1' (converter chip select inactive), and the state transitions to **StartState**.
   * **SampleHoldState**:
     + If **SampleHoldDone** is asserted, the **intSerial2ParallelEN** signal is set to '1' (converter data is being received), and the state transitions to **ConvertState**.
     + If **SampleHoldDone** is not asserted, the **ExpA\_CS\_L** signal remains '0' (converter chip select still active), and the state remains in **SampleHoldState**.
   * **ConvertState**:
     + If **ConversionDone** is asserted, the **ExpA\_CS\_L** signal is set to '1' (converter chip select turned off), the **intSerial2ParallelEN** signal is set to '0' (converter data is finished), **intWriteConversion** is set to '1', and the state transitions to **ConvertWaitState1**.
     + If **ConversionDone** is not asserted, the **ExpA\_CS\_L** signal remains '0' (converter chip select still active), and the **intSerial2ParallelEN** signal remains '1' (converter data is being received), and the state remains in **ConvertState**.
   * **ConvertWaitState1**:
     + The **intWriteConversion** signal is set to '0', and the state transitions to **ConvertWaitState2**.
   * **ConvertWaitState2**:
     + The state transitions to **ConvertDoneState**.
   * **ConvertDoneState**:
     + If **EndDelay** is not asserted, the **ConversionCounterEN** signal is set to '1', and the state transitions to **IncConvCountState**.
     + If **EndDelay** is asserted, the **Serial2ParallelCLR** signal is set to '1', and the state remains in **ConvertDoneState**.
   * **IncConvCountState**:
     + The **ConversionCounterEN** signal is set to '0', and the state transitions to **InterConvDelayState**.
   * **InterConvDelayState**:
     + If **InterConversionDelayTC** is asserted, the **InterConversionDelayEN** signal is set to '0', and the state transitions to **StartState**.
     + If **InterConversionDelayTC** is not asserted, the **InterConversionDelayEN** signal is set to '1', and the state remains in **InterConvDelayState**.
   * Other states:
     + The default case sets the state to **StartState**.

Other Signals and Assignments:

* + **EndDelay** signal is set to '0' in the **StartState** and '1' in the **ConvertDoneState** when **SlowEnable** is asserted.
  + **ExpA\_CLK\_EN** signal is set based on the state and cycle count.
  + **intExpA\_CLK** signal toggles based on **SlowEnable** signal and **intExpA\_CLK(0)** value.
  + **ConversionCounter** counts the number of conversions that have occurred in the current loop time.
  + **CycleCounter** is used to keep track of the logic sequence during individual conversions.
  + **InterConversionDelayCNTR** counts the delay between conversions.
  + **intConverting**, **Converting**, and **WriteEn** signals control the conversion and write processes.
  + **asyncResetCycleCounter** and **ResetCycleCounter** signals are used to reset the **CycleCounter**.
  + **SampleHoldDone** and **ConversionDone** signals indicate the completion of sample/hold and conversion, respectively.
  + **preInterConversionDelayTC** signal determines the terminal count for the interconversion delay.

Output Signals:

1. **ExpA\_CS\_L** (out std\_logic):
   * This signal represents the chip select for the ADC.
   * It is active (low) when the state machine is in the **StartState** or **ConvertState**.
   * It is inactive (high) in other states.
2. **ExpA\_CLK** (out std\_logic):
   * This signal represents the clock for the ADC.
   * It is enabled (**ExpA\_CLK\_EN** signal) when the state machine is in the **SampleHoldState** or **ConvertState** and during specific clock cycles (**CycleCounter**).
   * The clock signal toggles (**intExpA\_CLK**) on the rising edge of the system clock (**SysClk**) and when **SlowEnable** is active (high).
3. **Serial2ParallelEN** (out std\_logic):
   * This signal is the enable signal for the shift register used for ADC output.
   * It is enabled when **intSerial2ParallelEN** is active (high), **intExpA\_CLK** is high, and **SlowEnable** is active (high).
4. **Serial2ParallelCLR** (out std\_logic):
   * This signal represents the clear signal for the shift register used for ADC output.
   * It is active (low) in the **ConvertDoneState** when **EndDelay** is active (low).
5. **WriteConversion** (out std\_logic):
   * This signal indicates the end of conversion.
   * It is active (high) when **WriteEN** is active (high).