Project Status Report – RMC75E Test Bench

Date: July 14, 2023

Subject: Progress Update

Current Phase: Refining Analog, State Machine, Serial-to-Parallel, and Data Buffer test units.

Work to Date: This week I was able to massage all the internal signals into place so that the Quad test module correctly simulates the DUT and all of its subcomponents. The quad module is essentially just a wrapper module that instantiates six instances of the Quadxface module, which is responsible for handling quadrature encoding for expansion modules on the RMC75E modular motion controller. It receives various input signals such as clock signals, stimulus signals, data signals, and control signals. It performs several tasks related to quadrature encoding, including counting and direction detection.

Challenges: Now working on the analog module, which is similarly a sort of wrapper module that instantiates the sub modules needed for analog data processing and analog-to-digital conversion for the expansion module. There were some issues with the expansion module clock, which provides the timing reference for the expansion module's internal operations and synchronizes its data transfers. The signal was becoming intermittently undefined on transition from high to low.

The issue seemed to be caused by a rather oddly placed value assignment for the clock inside the wrapper module. This is odd because the wrapper probably shouldn't be driving that signal at all, I'm not even sure it should be able to drive that signal without throwing an error since that signal is an output signal defined in one of the sub modules. It's also odd because of its placement. It is placed within the component instantiation section of the logic, rather than the signal declaration section. I am sure Dennis had a good reason for including it, but that reason is not apparent to me at this time.

```
  component DataBuffer is
    port (
      H1_CLKWR      : in std_logic;
      SysClk        : in std_logic;
      SynchedTick     : in std_logic;
      SynchedTick60   : in std_logic;
      AnlgPositionRead0 : in std_logic;
      AnlgPositionRead1 : in std_logic;
      ExpA0ReadCh0    : in std_logic;
      ExpA0ReadCh1    : in std_logic;
      ExpA1ReadCh0    : in std_logic;
      ExpA1ReadCh1    : in std_logic;
      ExpA2ReadCh0    : in std_logic;
      ExpA2ReadCh1    : in std_logic;
      ExpA3ReadCh0    : in std_logic;
      ExpA3ReadCh1    : in std_logic;
      WriteConversion   : in std_logic;
      S2P_Addr      : inout std_logic_vector (3 downto 0);
      S2P_Data      : in std_logic_vector (15 downto 0);
      DataOut       : out std_logic_vector (15 downto 0)
    );
  end component;

  signal  Serial2ParallelEN,
      Serial2ParallelCLR  : std_logic;  -- := '0';
  signal  WriteConversion   : std_logic;  -- := '0';
  signal  SignExtend      : std_logic_vector(31 downto 16); -- Used to sign extend DataOut value
  signal  DataOut       : std_logic_vector (15 downto 0); -- := X"0000";
  signal  S2P_Addr      : std_logic_vector (3 downto 0);  -- := X"0";
  signal  S2P_Data      : std_logic_vector ( 15 downto 0);  -- := X"0000";

begin

  -- ExpA_CS_L <= '0';
  -- ExpA_CLK <= '0';

  SignExtend <= DataOut(15) & DataOut(15) & DataOut(15) & DataOut(15) &
          DataOut(15) & DataOut(15) & DataOut(15) & DataOut(15) &
          DataOut(15) & DataOut(15) & DataOut(15) & DataOut(15) &
          DataOut(15) & DataOut(15) & DataOut(15) & DataOut(15);

  AnlgDATA <= SignExtend & DataOut when AnlgPositionRead0 = '1' or AnlgPositionRead1 = '1' or
              ExpA0ReadCh0 = '1' or ExpA0ReadCh1 = '1' or ExpA1ReadCh0 = '1' or ExpA1ReadCh1 = '1' or
              ExpA2ReadCh0 = '1' or ExpA2ReadCh1 = '1' or ExpA3ReadCh0 = '1' or ExpA3ReadCh1 = '1' else
        X"0000_0000";

  StateMach_1 : StateMachine
    port map (
      SysReset      => SysReset,
      SysClk        => SysClk,
```

At any rate, commenting out this odd bit of code seems to allow the expansion module clock to become fully defined. However, we still aren't out of the woods yet, as the clock appears to have an irregular wave pattern that is simply not correct.

I am still trying to figure out what is causing this irregular behavior. Yesterday, I refined the test bench for the state machine module, which is where this signal originates, to see if I could not isolate the issue. The thinking here is that if I can get the submodule to behave correctly, I should be able to take that and apply it to the analog simulation and get everything to work as it should.

It's possible the issue is related to the loop time variable which determines the correct interval between synchronization ticks. Currently I have the loop time variable set to "000", which according to the comments in the source module should correspond to an interval period of 125 us.

Next Steps:

I have some ideas for redesigning the analog test bench that might help resolve the issue. Additionally, I will just continue to review the source code, the interactions between the various signals and conditions within the analog wrapper and the submodules it instantiates. Hopefully this will allow me to determine why this irregular behavior is occurring.