```
-- AD_Controller
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
library synplify;
use synplify.attributes.all;

entity AD_Controller is port

(clock              : in  std_logic;
resetn              : in  std_logic;
state_out           : out std_logic_vector(4 downto 0);

-- i/o to serial controller
AD_RUN              : in std_logic;

MUX_Selector        : out std_logic_vector(3 downto 0);
MUX_Enable          : out std_logic;

AD_RW               : out std_logic;
AD_CE               : out std_logic;
AD_CS               : out std_logic;
AD_DATA             : in  std_logic_vector(7 downto 0);
AD_Status           : in  std_logic;

--QPLL_A_Count        : in  std_logic_vector(7 downto 0);
--QPLL_B_Count        : in  std_logic_vector(7 downto 0);

RAM_ADDRESS         : out std_logic_vector(4 downto 0);
RAM_DATA_D          : out std_logic_vector(7 downto 0);
RAM_WRITE_ENABLE    : out std_logic);

end AD_Controller;

architecture rtl of AD_Controller is
attribute syn_radhardlevel of rtl : architecture is "tmr";

type state_values is (st0, st1, st2, st4, st5, st6, st7, st8, st9, st12, st13, st14, st18,
 st19, st20, st21, st22 );
--  st3,  st10, st11,  st15, st16, st17,
signal pres_state: state_values;

signal count        : std_logic_vector(3 downto 0);
signal AD_Status_reg :std_logic;
signal load_timer           : std_logic;
signal timer_en             : std_logic;
signal timed_out            : std_logic;
signal timer                : std_logic_vector(3 downto 0);


--signal AD_result    : std_logic_vector(7 downto 0);

begin
    -- fsm register
    state_reg: process (clock, resetn, AD_RUN)
    begin
    if (resetn = '0')  then
        pres_state  <= st0;
        count       <= "0000";
        MUX_Enable  <= '0';
        AD_RW       <= '1';
        AD_CE       <= '1';
        AD_CS       <= '1';
        RAM_WRITE_ENABLE <= '0';
        RAM_DATA_D <= "00000000";
        RAM_ADDRESS <= "00000";
        MUX_Selector <= "0000";
```

```
        AD_Status_reg <= '0';
        timer <= "1010";
        load_timer <= '1';
        timer_en <= '0';
        --AD_result   <= "00000000";

    elsif clock'event AND clock = '1' then
        MUX_Selector <= count;
        AD_Status_reg <= AD_Status;

-------------------------------------------------
            if load_timer = '1' then
                timer <= "1010"; --
            elsif timer_en = '1' then --
                timer <= timer - 1;
            else
                timer <= timer;
            end if;

            if timer = "0000" then
                timed_out <= '1';
            else
                timed_out <= '0';
            end if;
-------------------------------------------------


        case count is
            when "0000" => RAM_ADDRESS <= "10000"; -- '16'
            when "0001" => RAM_ADDRESS <= "10001"; -- '17'
            when "0010" => RAM_ADDRESS <= "10010"; -- '18'
            when "0011" => RAM_ADDRESS <= "10011"; -- '19'
            when "0100" => RAM_ADDRESS <= "10100"; -- '20'
            when "0101" => RAM_ADDRESS <= "10101"; -- '21'
            when "0110" => RAM_ADDRESS <= "10110"; -- '22'
            when "0111" => RAM_ADDRESS <= "10111"; -- '23'
            when "1000" => RAM_ADDRESS <= "11000"; -- '24'
            when "1001" => RAM_ADDRESS <= "11001"; -- '25'
            when others => RAM_ADDRESS <= "00000"; -- '0'
        end case;

        case pres_state is
            when st0 => -- initial state
                state_out <= ("00000");
                AD_RW <= '1';
                AD_CE <= '1';
                AD_CS <= '1';
                count <= "0000";
                if (AD_RUN = '1') then
                    pres_state <= st1;
                    MUX_Enable <= '1';
                    load_timer <= '0';
                else
                    pres_state <= st0;
                    MUX_Enable <= '0';
                    load_timer <= '1';
                end if;

            when st1 => -- mux switch time
                state_out <= ("00001");
                timer_en <= '1';
                if timed_out = '1' then
                     pres_state <= st2;
                else
                     pres_state <= st1;
                end if;

            when st2 => -- mux switch time
```

```
                state_out <= ("00010");
                load_timer <= '1';
                timer_en <= '0';
                pres_state <= st4;

 --           when st3 => -- mux switch time
  --              state_out <= ("00011");
   --             pres_state <= st4;

            when st4 => -- mux switch time
                state_out <= ("00100");
                AD_CE <= '0';
                AD_CS <= '0';
                pres_state <= st5;

            when st5 => -- start the conversion
                state_out <= ("00101");
                if (AD_RUN = '1') then
                    AD_RW <= '0';
                    pres_state <= st6;
                else
                    pres_state <= st0;
                end if;

            when st6 => --
                state_out <= ("00110");
                if (AD_Status_reg = '1') then -- conversion started
                    AD_RW <='1';
                    pres_state <= st7;
                else -- wait
                    pres_state <= st6;
                end if;

            when st7 => -- wait for bus access time of 250 ns
                state_out <= ("00111");
                AD_RW <='1';
                load_timer <= '0';
                pres_state <= st8;

            when st8 => -- wait for bus access time of 250 ns
                state_out <= ("01000");
                timer_en <= '1';
                if timed_out = '1' then
                     pres_state <= st9;
                else
                     pres_state <= st8;
                end if;


            when st9 => -- wait for bus access time of 250 ns
                state_out <= ("01001");
                pres_state <= st12;

--            when st10 => -- wait for bus access time of 250 ns
--                state_out <= ("01010");
--                pres_state <= st11;

--            when st11 => -- wait for bus access time of 250 ns
--                state_out <= ("01011");
--               pres_state <= st12;

            when st12 => --
                state_out <= ("01100");
                load_timer <= '1';
                timer_en <= '0';
                if (AD_Status_reg = '0') then -- conversion finished

                    pres_state <= st13;
```

```
                else -- wait
                    pres_state <= st12;
                end if;

            when st13 => -- wait for bus access time of 250 ns
                        -- data is valid at a maximum of 250ns after status goes low
                state_out <= ("01101");
                load_timer <= '0';
                pres_state <= st14;

            when st14 => -- wait for bus access time of 250 ns
                state_out <= ("01110");
                timer_en <= '1';
                if timed_out = '1' then
                     pres_state <= st18;
                else
                     pres_state <= st14;
                end if;

 --           when st15 => -- wait for bus access time of 250 ns
 --               state_out <= ("01111");
 --               pres_state <= st16;

--            when st16 => -- wait for bus access time of 250 ns
--                state_out <= ("10000");
--                pres_state <= st17;

--            when st17 => -- wait for bus access time of 250 ns
--                pres_state <= st18;

            when st18 => --  put result of conversion in AD_result reg.
                state_out <= ("10010");
                load_timer <= '1';
                timer_en <= '0';
                RAM_DATA_D <= AD_DATA; --AD_result <= AD_DATA;
                RAM_WRITE_ENABLE <= '1';
                pres_state <= st19;

            when st19 => -- put AD_result at RAM address
                state_out <= ("10011");
                --RAM_DATA_D <= AD_result;
                pres_state <= st20;

            when st20 => -- put AD_result at RAM address disable the data from convertor
                state_out <= ("10100");
                AD_RW <= '1';
                AD_CE <= '1';
                AD_CS <= '1';
                --RAM_WRITE_ENABLE <= '0';
                pres_state <= st21;

            when st21 => --
                state_out <= ("10101");
                RAM_WRITE_ENABLE <= '0';
                if (AD_RUN = '1') then
                    if count = "1001" then
                        count <= "0000"; -- restart at count "0000"
                    else
                        count <= count + 1; -- do conversion on next channel
                    end if;
                    load_timer <= '0';
                    pres_state <= st1;
                else
                    pres_state <= st0;
                end if;

        end case;
    end if;
```

```
    end process;
end rtl;
```