```
LIBRARY std;
  USE std.textio.ALL;

LIBRARY ieee;
  USE ieee.std_logic_textio.ALL;

LIBRARY Common_test;
  USE Common_test.testutils.all;

ARCHITECTURE test OF mainCircuit_tester IS

  constant clockPeriodInn  : time := 1.0/g_clockFrequency * 1 sec;
  constant clockPeriod  : time := 2.0/g_clockFrequency * 1 sec;
  signal sClock         : std_uLogic := '1';
  signal sReset         : std_uLogic ;
  signal testInfo       : string(1 to 40) := (others => ' ');

  signal counter_test        : unsigned(5 downto 0);
  signal trigger_counter     : unsigned(18 downto 0);

  signal sClock_fpga         : std_uLogic := '1';
  constant value_8           : natural := 8;
  constant value_16          : natural := 16;
  constant value_19          : natural := 19;
  constant value_32          : natural := 32;

  signal ADC_WRITE_INIT      : unsigned(15 downto 0);
  signal count_miso          : unsigned(5 downto 0);
  signal FRAM_WREN_INIT      : unsigned(7 downto 0);
  signal FRAM_WRDI_INIT      : unsigned(7 downto 0);
  signal pretrigVal          : unsigned(7 downto 0);
  signal count_memoryMax     : unsigned(18 downto 0);
  signal count_memory        : unsigned(18 downto 0);
  signal FRAM_READ_INIT      : unsigned(31 downto 0);
  signal FRAM_READ_ADD       : unsigned(18 downto 0);
  signal FRAM_WRITE_INIT     : unsigned(31 downto 0);
  signal init_done           : std_uLogic ;
  signal miso_start          : std_uLogic ;
  signal clk_start          : std_uLogic ;
  signal selector            : unsigned(3 downto 0);

  -- An example of procedure (function which returns nothing)
  -- Here checks a value and log given error message if sim is not the same
  procedure checkMeas(
    msg :           string;
    measArg :       std_ulogic) is
  begin

    std.textio.write(std.textio.output, LF & "==================" & LF);
    std.textio.write(std.textio.output, "Testing " & msg  & LF);

    assert (meas_1mhz = measArg)
      report ("meas_1mhz error - expected " & to_string(measArg)) severity error;
    if (meas_1mhz = measArg) then
      report " ** Ok" severity note;
```

```
    end if;

    std.textio.write(std.textio.output, "==================" & LF);

    -- Force clock synch.
    wait until clk'event and clk = '1';
  end procedure checkMeas;

BEGIN

 --------------------------------------------------------------------------------
                                                                -- reset and clock
  rst <= sReset;

  sClock <= not sClock after clockPeriodInn/2;
  clk <= sClock;



  sClock_fpga <= not sClock_fpga after clockPeriod/2;
  fpga_sck <= sClock_fpga ;

  --------------------------------------------------------------------------------
                                                                -- tester
  process
    variable state : std_ulogic := '0';
  begin
    -- Outputs default values
    sReset <= '1';
    acq_pretrig <= '0';
    acq_trig <= '0';
    adc_sdo <= (others => '1');
    fpga_m <= (others => '0');
    fpga_mosi <= '0';
    clk_en <= '0';
    clk_start <= '0';
  --------------------------------------------------------------------------------
                                                                  -- TESTER_MCU
    counter_test <= (others => '0');
    count_memory <= (others => '0');
    count_memoryMax <= (others => '1');
    ADC_WRITE_INIT <= "1010001010000000"; --1 010 00 10 10000000
    FRAM_READ_INIT <= SHIFT_LEFT(RESIZE("0000001100000",FRAM_READ_INIT),19);
    FRAM_WRDI_INIT <= "00000100";
    FRAM_WREN_INIT <= "00000110";
    pretrigVal <= "01010000";
    FRAM_WRITE_INIT <= SHIFT_LEFT(RESIZE("0000001000000",FRAM_READ_INIT),19);
    init_done <= '0';
    miso_start <= '0';
    selector <= (others => '0');
    count_miso <= (others => '0');
    FRAM_READ_ADD <= (others => '0');

    testInfo <= pad("Init", testInfo'length);
    wait for 20*clockPeriod;
```

```
sReset <= '0';
wait until rising_edge(sClock_fpga);

while true loop
  -- Wait until toggler should toggle
  wait for clockPeriod;


  if selector = "0000" then
    selector <= "1110";

  elsif selector = "1110" then
    fpga_m <= "1110";
    if clk_start = '0' then
      wait for clockPeriod;
      clk_start <= '1';
    end if;
    acq_pretrig <= '1';
    fpga_mosi <= pretrigVal(pretrigVal'high);
    pretrigVal <= SHIFT_LEFT(pretrigVal,1);
    counter_test <= counter_test+1 ;
      if counter_test = value_8 then
        counter_test <= (others => '0');
        clk_start <= '0';
        selector <= "0001";
        acq_pretrig <= '0';
      end if ;

  elsif selector = "0001" then
    fpga_m <= "0001";
    if clk_start = '0' then
      wait for clockPeriod;
      clk_start <= '1';
    end if;
    fpga_mosi <= ADC_WRITE_INIT(ADC_WRITE_INIT'HIGH);
    ADC_WRITE_INIT <= SHIFT_LEFT(ADC_WRITE_INIT,1);
    counter_test <= counter_test +1;
    if counter_test = value_16-1 then
      counter_test <= (others => '0');
      clk_start <= '0';
      selector <= "1000";
    end if ;

  elsif selector = "1000" then
    fpga_m <= "1000";
    if clk_start = '0' then
      wait for clockPeriod;
      clk_start <= '1';
    end if;
    if init_done <= '0' then
      fpga_mosi <= FRAM_WREN_INIT(FRAM_WREN_INIT'high);
      FRAM_WREN_INIT <= SHIFT_LEFT(FRAM_WREN_INIT,1);
      counter_test <= counter_test +1;
      if counter_test = value_8-1 then
        counter_test <= (others => '0');
```

```
      clk_start <= '0';
      selector <= "1111";
    end if ;
  else
    fpga_mosi <= FRAM_WRITE_INIT(FRAM_WRITE_INIT'high);
    FRAM_WRITE_INIT <= SHIFT_LEFT(FRAM_WRITE_INIT,1);
    counter_test <= counter_test +1;
    if counter_test = value_32-1 then
      counter_test <= (others => '0');
      clk_start <= '0';
      selector <= "1010";
    end if ;
  end if ;

elsif selector = "1111" then
  fpga_m <= "1111";
  if clk_start = '0' then
    wait for clockPeriod;
    clk_start <= '1';
  end if;
  counter_test <= counter_test +1;
    if counter_test = value_8-1 then
      counter_test <= (others => '0');
      init_done <= '1';
      clk_start <= '0';
      selector <= "1000";
    end if ;

elsif selector = "1010" then
  fpga_m <= "1010";
  if clk_start = '0' then
    wait for clockPeriod;
    clk_start <= '1';
  end if;
  count_memory <= count_memory+1;
  if count_memory = count_memoryMax then
    acq_trig <= '1';
  end if ;
  if out1 = '1' then
    clk_start <= '0';
    selector <= "1011";
  end if ;

elsif selector = "1011" then
  fpga_m <= "1011";
  if clk_start = '0' then
    wait for clockPeriod;
    clk_start <= '1';
  end if;
  if fpga_miso = '1' and miso_start = '0' then
    --count_miso <= count_miso+1;
    miso_start <= '1';
    acq_trig <= '0';
  elsif miso_start = '1' and count_miso <19 then
    count_miso <= count_miso+1;
```

```
          FRAM_READ_ADD <= SHIFT_LEFT(FRAM_READ_ADD,1);
          FRAM_READ_ADD(0) <= fpga_miso;
        elsif count_miso = 19 and miso_start = '1' then
          miso_start <= '0';
          count_miso <= (others => '0');
          FRAM_READ_INIT <= RESIZE(FRAM_READ_INIT+
                            RESIZE(FRAM_READ_ADD,
                              FRAM_READ_INIT),
                            FRAM_READ_INIT);
          clk_start <= '0';
          selector <= "0100";
        end if;

      elsif selector = "0100" then
        fpga_m <= "0100";
        if clk_start = '0' then
          wait for clockPeriod;
          clk_start <= '1';
        end if;
        fpga_mosi <= FRAM_READ_INIT(FRAM_READ_INIT'high);
        FRAM_READ_INIT <= SHIFT_LEFT(FRAM_READ_INIT,1);
      end if ;
      -- Invert state and loop
      state := not state;

    end loop;

  end process;

END ARCHITECTURE test;
```