EE 214: Digital Circuits Lab
Dev Arora (23B1271)
Arjun Singh (23B1272)

ADC to LCD via SPI

Task 2A

October 30, 2024

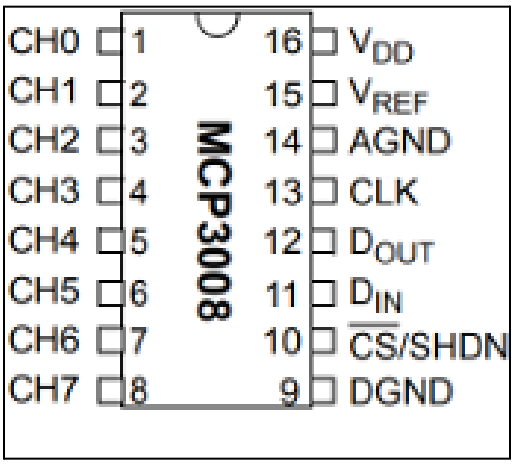
Work Distribution:

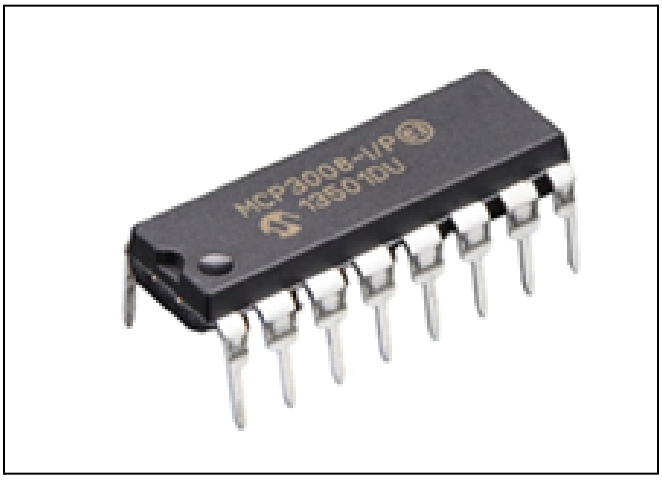
Dev & Arjun:

Both of us sat together and wrote the code for master, we had to write from the start again because the previous code lacked certain parameters which our RA asked us to add. We wrote an additional test bench for master, top level entity for LCD_VHDL, made circuit to test on hardware and did appropriate pin planning.

ADC Module - MCP3008:

We have used this ADC module which would act as the peripheral with which a controller would talk via SPI. The MCP3008 is a 10-bit analog-to-digital converter (ADC) from Microchip Technology, well-suited for applications requiring multiple input channels and precise digital representation of analog signals.





Some salient features:

- The MCP3008 provides a 10-bit resolution, allowing it to represent input signals with up to 1024 discrete values (from 0 to 1023).

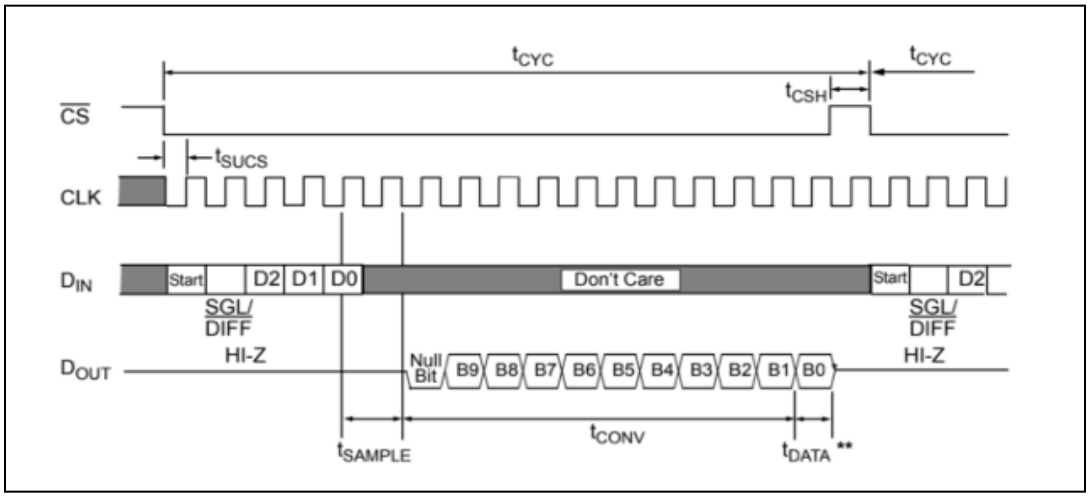
- It offers 8 single-ended channels or 4 differential pairs, providing flexibility for multiple sensors.
- The MCP3008 uses Serial Peripheral Interface(SPI) communication for data transfer, which makes it easy to interface with microcontrollers,FPGAs, and processors that support SPI.

ADC and SPI communication design:

ADC Working:

- ADC converts input voltage to a corresponding digital value with respect to a reference Voltage.
- For this Task, V_{ref} = 3.3V.
  Say V_{in} = 2.0V, then for a 10 bit ADC, $x_{10} = (2.0 * 2^{10})/3.3 \simeq 621$
  ADC output is x_2 = **1001101101**

ADC configuration and SPI communication:



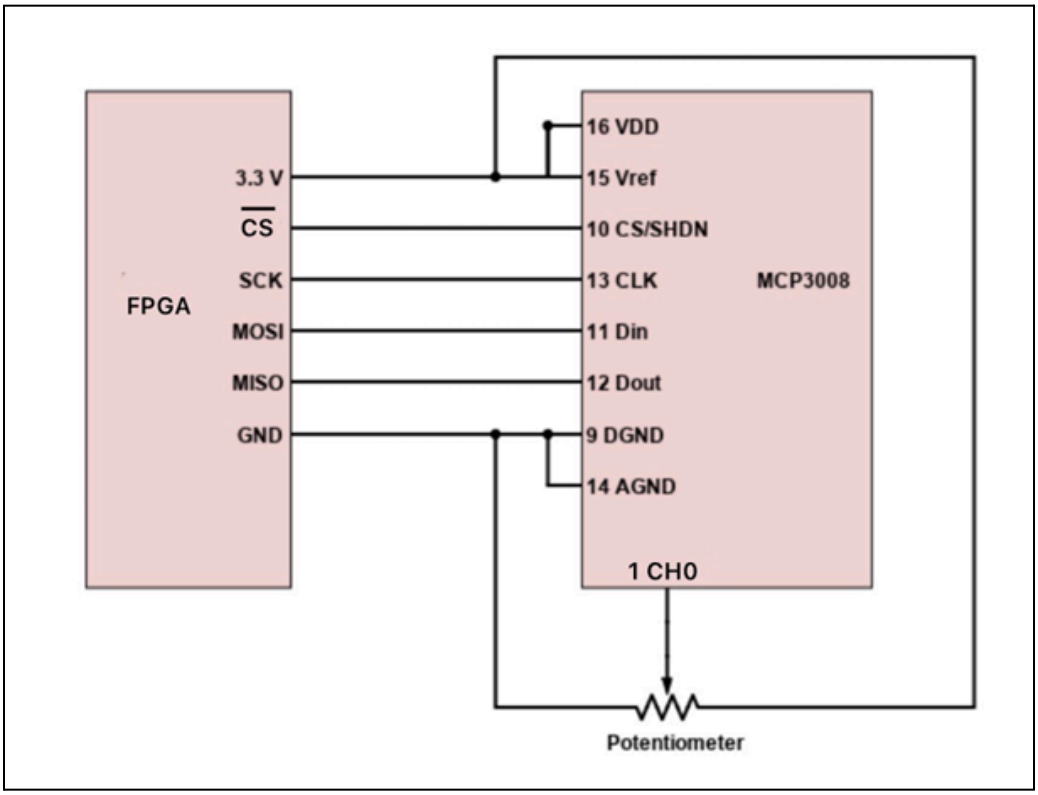
- The master & the ADC module require a total of 17 rising- & falling-edges when CS has been pulled low to complete a single iteration of the ADC sending a 10-bit voltage measurement to the master.
- The master sends 5 setup bits (**11000**) one-by-one on each of the first 5 falling edges of SCLK. We set SCLK to idle at **1** so that when it starts (with CS=0), the first edge is a falling edge. This lets the master send a start bit (**1**) right away, which the ADC reads on the next rising edge to activate.
- The MISO line stays in a Hi-Z (high impedance) state initially. The master still reads MISO on rising edges and stores the bits, but ignores the first ones, outputting only the last 10 bits, which contain the actual voltage data.
- As shown in Fig. 4, we need to connect the potentiometer output to **CH0** (Channel 0). The 4 bits following the start bit set the input type (single-ended or differential) and choose the channel(s) for analog input.. For selecting **single-ended mode** & **CH0** these 4 bits need to be **1000**.

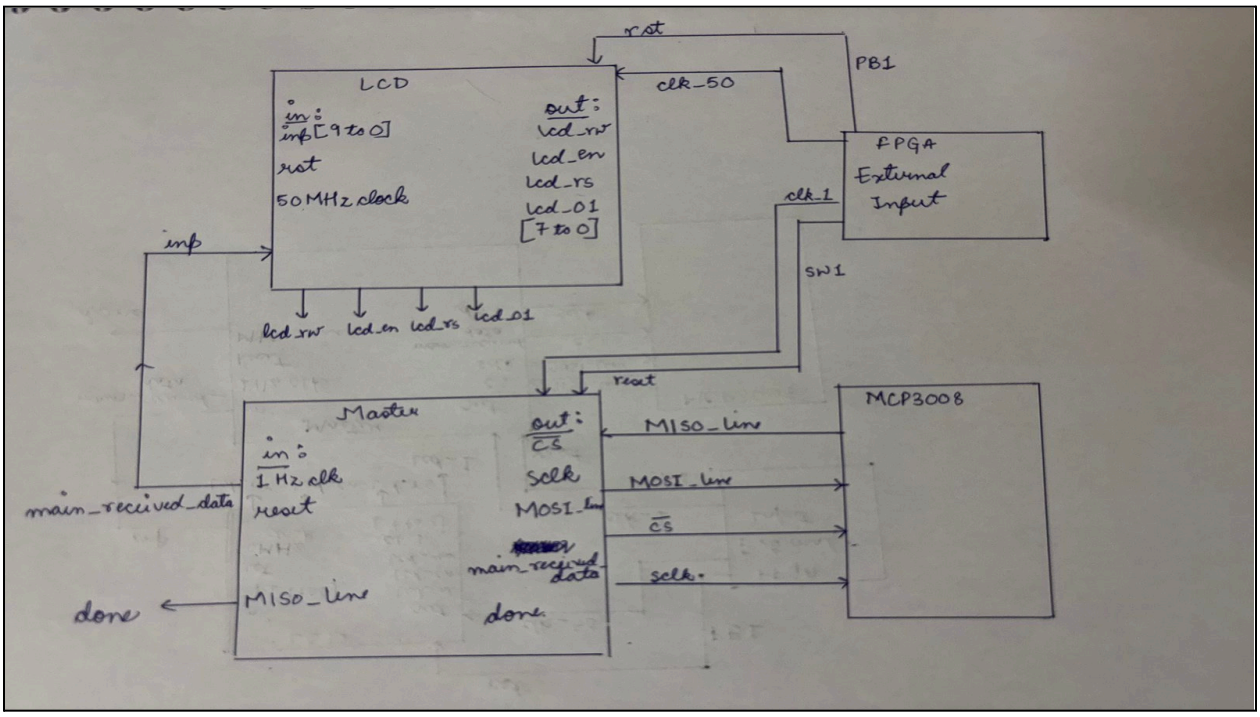
- We use counters to track the number of falling edges since CS was last pulled low. Once this count exceeds 16 (0–16 indexing), the master stops writing to MOSI or reading from MISO. A new communication cycle begins only when CS is toggled high, then low again. CS is controlled by a reset input, mapped to a switch, with **reset = 0** setting **CS = 0** and **reset = 1** setting **CS = 1**.
- In our design, each new measurement requires closing the switch, which resets the counter and master's data register, then opening it again. This triggers the ADC to send a fresh measurement to the master via SPI.

Circuit Design and Explanation:

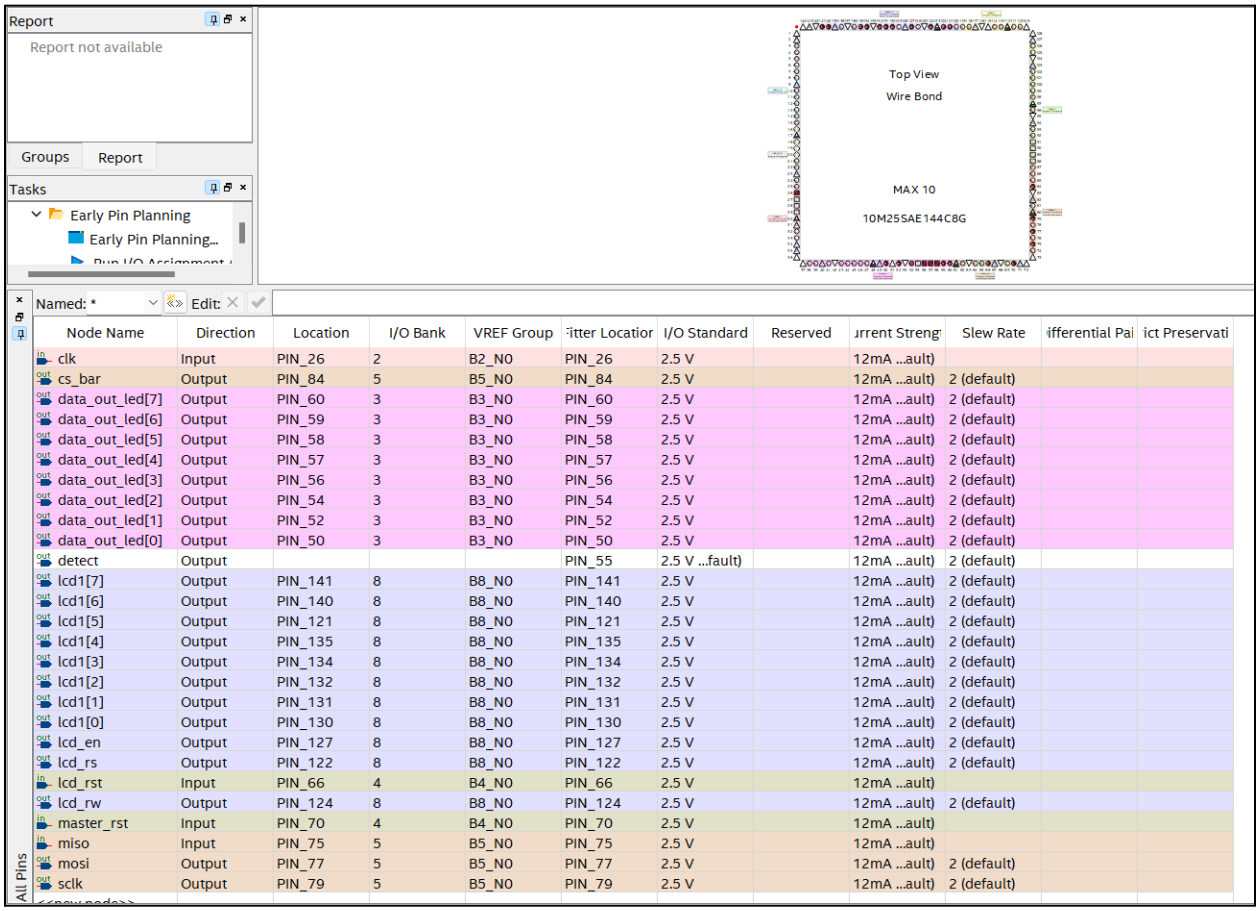
The master module sends the CS bar, SCLK, and MOSI signals to the ADC. The ADC then sends back the MISO signal, which gets stored in a register called "main received data." After all 10 bits of the ADC output are transferred, the master signals that it's done by pulling the "done" signal high, indicating the data is now stored.

When the "done" signal is high, the register value is ready to display on the LCD. It's sent as the "inp" signal to the LCD module, which configures the display to show the digital output.





Pin Planning:

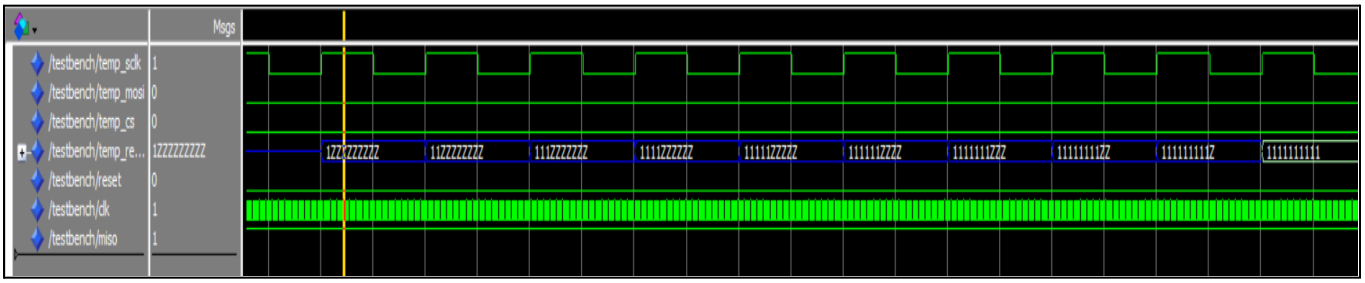


Node Name	Direction	Location	I/O Bank	VREF Group	itter Locatior	I/O Standard	Reserved	urrent Streng	Slew Rate	ifferential Pai	ict Preservati
clk	Input	PIN_26	2	B2_N0	PIN_26	2.5 V		12mA ...ault)			
cs_bar	Output	PIN_84	5	B5_N0	PIN_84	2.5 V		12mA ...ault)	2 (default)		
data_out_led[7]	Output	PIN_60	3	B3_N0	PIN_60	2.5 V		12mA ...ault)	2 (default)		
data_out_led[6]	Output	PIN_59	3	B3_N0	PIN_59	2.5 V		12mA ...ault)	2 (default)		
data_out_led[5]	Output	PIN_58	3	B3_N0	PIN_58	2.5 V		12mA ...ault)	2 (default)		
data_out_led[4]	Output	PIN_57	3	B3_N0	PIN_57	2.5 V		12mA ...ault)	2 (default)		
data_out_led[3]	Output	PIN_56	3	B3_N0	PIN_56	2.5 V		12mA ...ault)	2 (default)		
data_out_led[2]	Output	PIN_54	3	B3_N0	PIN_54	2.5 V		12mA ...ault)	2 (default)		
data_out_led[1]	Output	PIN_52	3	B3_N0	PIN_52	2.5 V		12mA ...ault)	2 (default)		
data_out_led[0]	Output	PIN_50	3	B3_N0	PIN_50	2.5 V		12mA ...ault)	2 (default)		
detect	Output				PIN_55	2.5 V ...fault)		12mA ...ault)	2 (default)		
lcd1[7]	Output	PIN_141	8	B8_N0	PIN_141	2.5 V		12mA ...ault)	2 (default)		
lcd1[6]	Output	PIN_140	8	B8_N0	PIN_140	2.5 V		12mA ...ault)	2 (default)		
lcd1[5]	Output	PIN_121	8	B8_N0	PIN_121	2.5 V		12mA ...ault)	2 (default)		
lcd1[4]	Output	PIN_135	8	B8_N0	PIN_135	2.5 V		12mA ...ault)	2 (default)		
lcd1[3]	Output	PIN_134	8	B8_N0	PIN_134	2.5 V		12mA ...ault)	2 (default)		
lcd1[2]	Output	PIN_132	8	B8_N0	PIN_132	2.5 V		12mA ...ault)	2 (default)		
lcd1[1]	Output	PIN_131	8	B8_N0	PIN_131	2.5 V		12mA ...ault)	2 (default)		
lcd1[0]	Output	PIN_130	8	B8_N0	PIN_130	2.5 V		12mA ...ault)	2 (default)		
lcd_en	Output	PIN_127	8	B8_N0	PIN_127	2.5 V		12mA ...ault)	2 (default)		
lcd_rs	Output	PIN_122	8	B8_N0	PIN_122	2.5 V		12mA ...ault)	2 (default)		
lcd_rst	Input	PIN_66	4	B4_N0	PIN_66	2.5 V		12mA ...ault)			
lcd_rw	Output	PIN_124	8	B8_N0	PIN_124	2.5 V		12mA ...ault)	2 (default)		
master_rst	Input	PIN_70	4	B4_N0	PIN_70	2.5 V		12mA ...ault)			
miso	Input	PIN_75	5	B5_N0	PIN_75	2.5 V		12mA ...ault)			
mosi	Output	PIN_77	5	B5_N0	PIN_77	2.5 V		12mA ...ault)	2 (default)		
sclk	Output	PIN_79	5	B5_N0	PIN_79	2.5 V		12mA ...ault)	2 (default)		

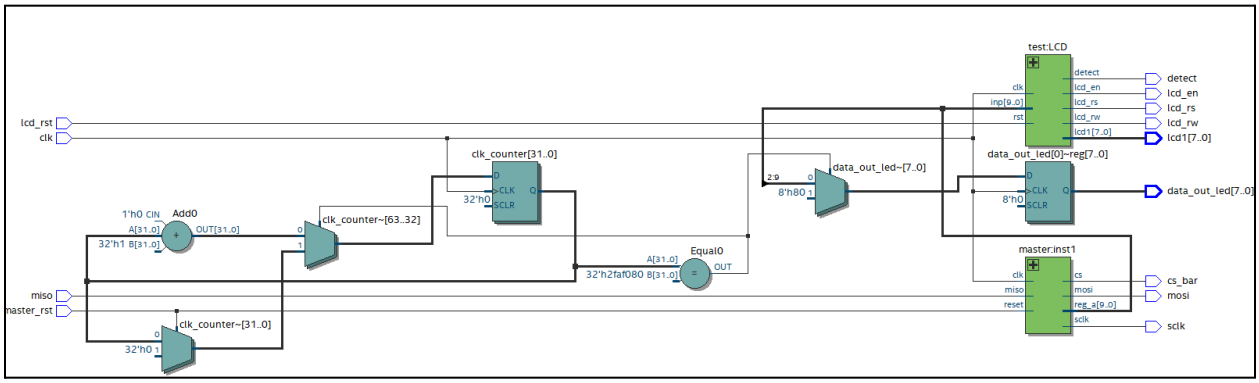
All Pins

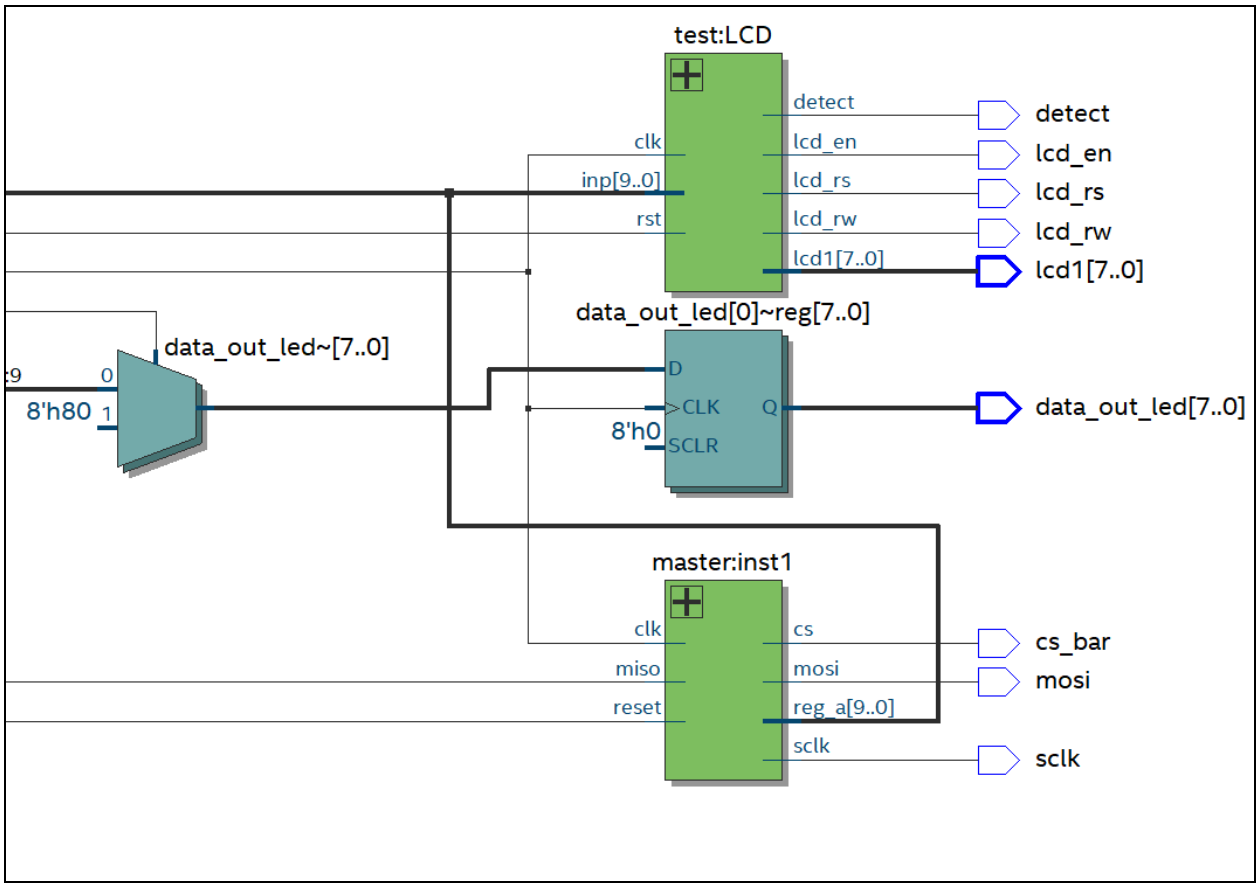
Results and Observation:

Waveform:

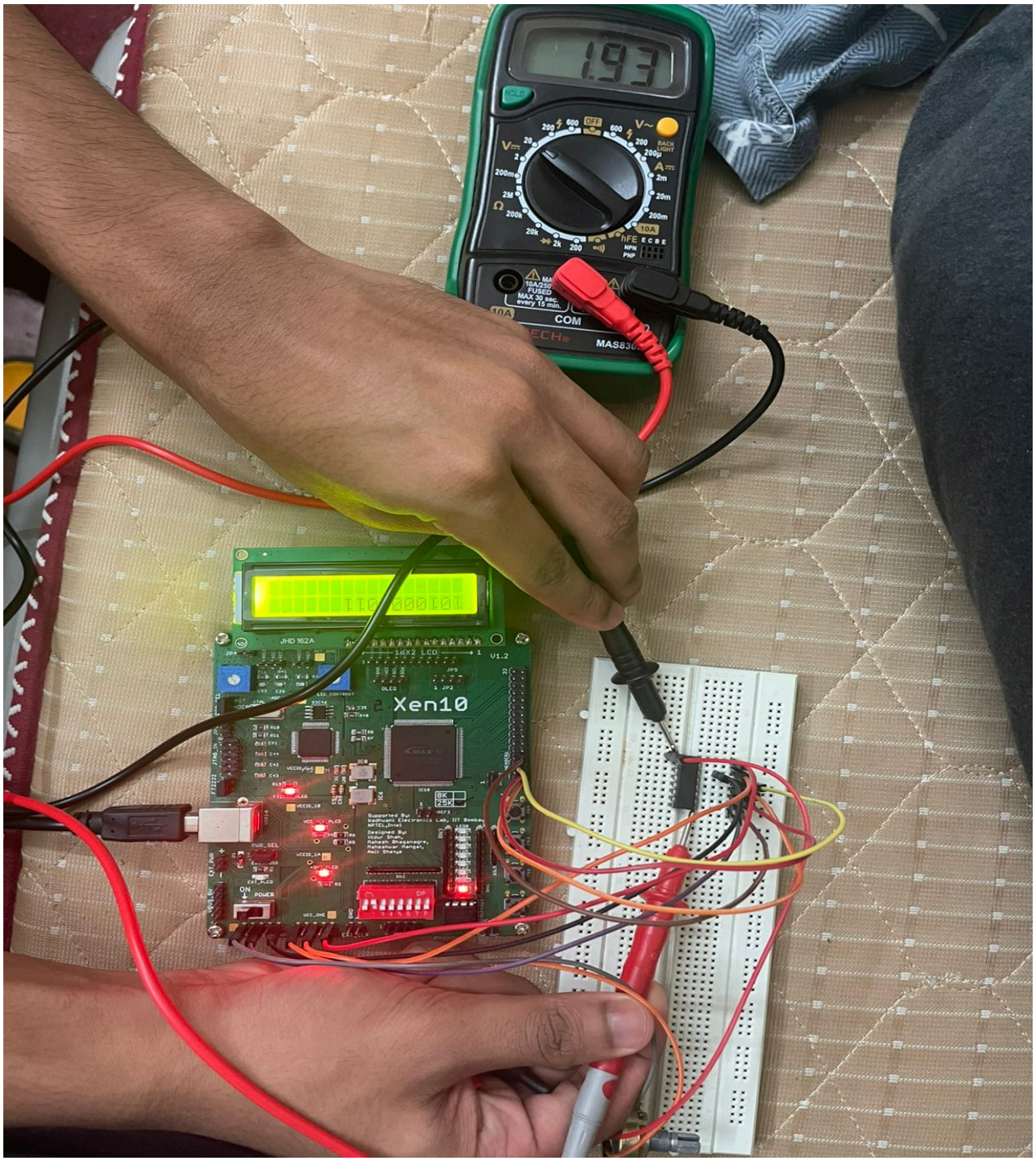


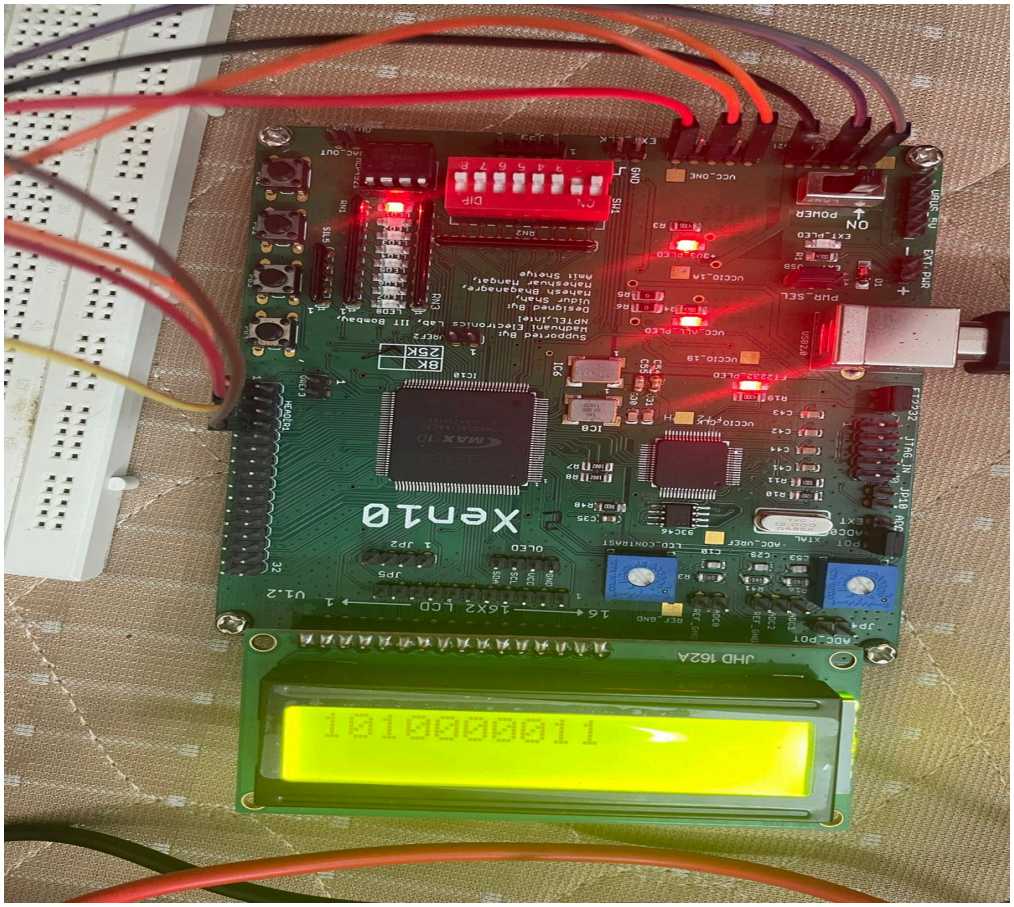
Netlist Viewer:

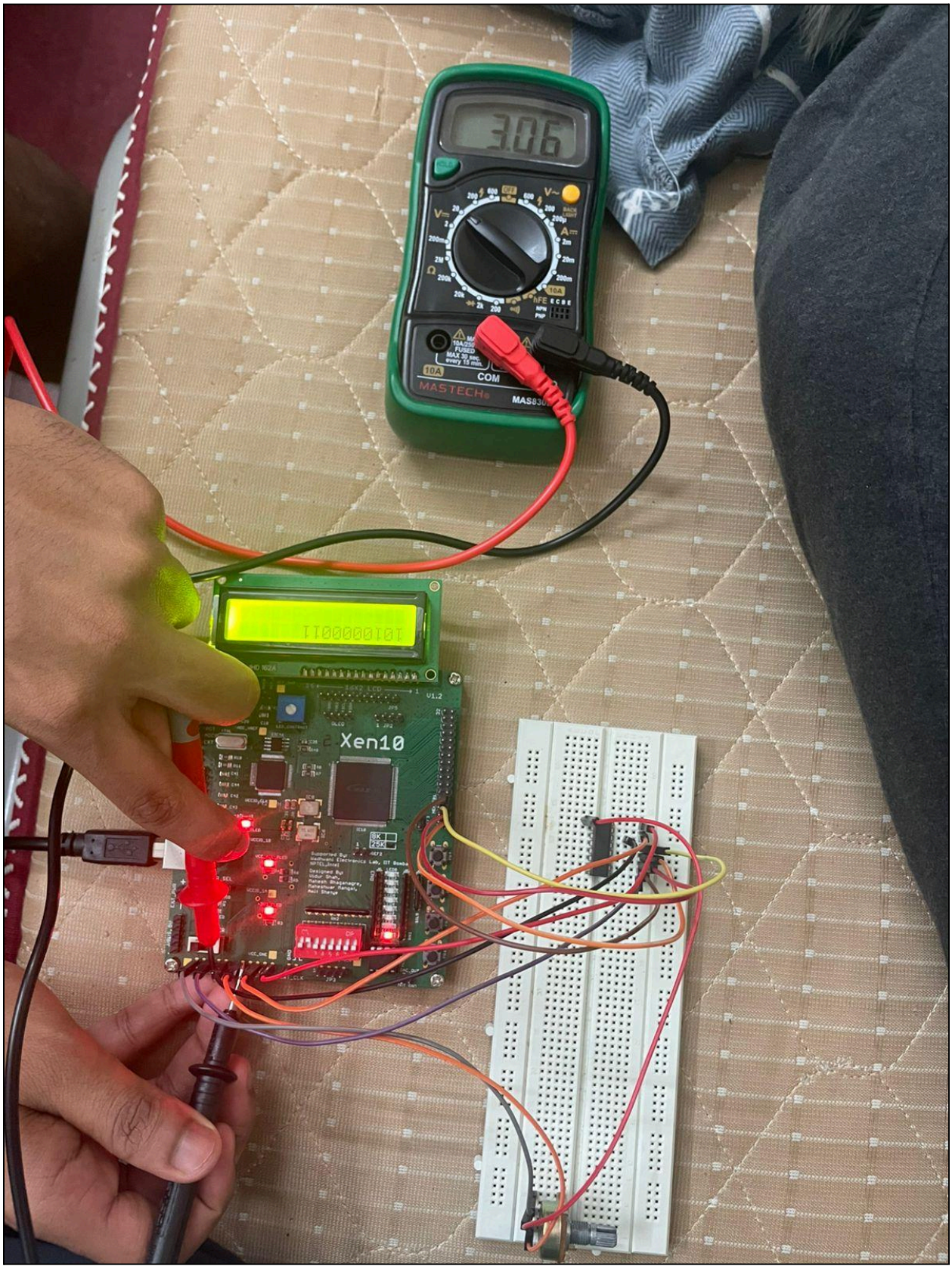




LCD output:




Xen10
JHD 162A
1010000011


3.06
COM
MASTECH
Xen10

Master Module: VHDL Code

VHDL Code for Master Module

```
library ieee;
use ieee.std_logic_1164.all;

entity master is
    port(
        clk : in std_logic;
        reset : in std_logic;
        miso : in std_logic;
        mosi : out std_logic;
        reg_a : out std_logic_vector(9 downto 0) :="ZZZZZZZZZZ";
        sclk : out std_logic;
        cs : out std_logic
    );
end entity;

architecture archi_master of master is

signal temp_sclk, temp_cs : std_logic;
constant divider_value : integer := 25;
signal clk_counter : integer := 0;
signal adc_initial : std_logic_vector(4 downto 0) :="11000";
signal counter_rise, counter_fall : integer :=0;

begin

clk_process : process(clk, reset)
begin
    if reset='1' then
        clk_counter <= 0;
        temp_sclk <='0';
    elsif rising_edge(clk) then
        if clk_counter = divider_value - 1 then
            temp_sclk <= NOT temp_sclk;
            clk_counter <=0;
        else
            clk_counter <= clk_counter +1;
        end if;
    end if;
end process;

communication : process(temp_sclk,reset)
begin
    if reset ='1' then
```

```
        counter_rise <=0;
        counter_fall <=0;
        reg_a <= "ZZZZZZZZZZ";
        temp_cs <= '1';
        mosi <= 'Z';

    else
        if rising_edge(temp_sclk) then
            if temp_cs = '0' then
                counter_rise <= counter_rise +1;
                case counter_rise is
                    when 7 =>  reg_a(9) <= miso;
                    when 8 =>  reg_a(8) <= miso;
                    when 9 =>  reg_a(7) <= miso;
                    when 10 => reg_a(6) <= miso;
                    when 11 => reg_a(5) <= miso;
                    when 12 => reg_a(4) <= miso;
                    when 13 => reg_a(3) <= miso;
                    when 14 => reg_a(2) <= miso;
                    when 15 => reg_a(1) <= miso;
                    when 16 => reg_a(0) <= miso;
                    when others => null;
                end case;
            end if;

        elsif falling_edge(temp_sclk) then
            counter_fall <= counter_fall +1;
                if counter_rise < 17 then temp_cs <= '0';
                elsif counter_rise > 17 then temp_cs <= '1'; end if
                    ;
                --if counter_rise =0 then temp_cs <= '0'; end if;

                case counter_fall is
                    when 0 => mosi <= adc_initial(4);
                    when 1 => mosi <= adc_initial(3);
                    when 2 => mosi <= adc_initial(2);
                    when 3 => mosi <= adc_initial(1);
                    when 4 => mosi <= adc_initial(0);
                    when others => mosi <= '0';

                end case;
            end if;
        end if;
end process;

cs <= temp_cs;
sclk <= temp_sclk;

end architecture;
```

Top-Level Module: VHDL Code

VHDL Code for Top-Level Module

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity top_level is
    port (
        clk                    : in std_logic;
          miso                   : in std_logic;
          lcd_rst        : in std_logic;
          master_rst     : in std_logic;
          cs_bar         : out std_logic := '1';
          sclk           : out std_logic := '0';
          mosi   : out std_logic := 'Z';
          lcd_rw                 : out std_logic;

          lcd_en                 : out std_logic;

        lcd_rs                 : out std_logic;

        lcd1: out std_logic_vector(7 downto 0);
          detect: out std_logic;
          data_out_led: out std_logic_vector(7 downto 0)
    );
end entity;

architecture archi_top_level of top_level is

    component master is
    port(
        clk : in std_logic;
        reset : in std_logic;
        miso : in std_logic;
        mosi : out std_logic;
        reg_a : out std_logic_vector(9 downto 0) :="ZZZZZZZZZZ";
        sclk : out std_logic;
        cs : out std_logic
    );
    end component;

    component test is
     port(
          inp           : in std_logic_vector(9 downto 0);
```

```
        clk           : in  std_logic;
        rst           : in  std_logic;
        lcd_rw        : out std_logic;

        lcd_en        : out std_logic;

        lcd_rs        : out std_logic;

        lcd1          : out std_logic_vector(7 downto 0);
        detect        : out std_logic
        );
    end component;

    signal data_out_master: STD_LOGIC_VECTOR(9 downto 0);
    signal clk_counter: integer := 0;
    begin
            inst1: master port map (clk=>clk,miso=>miso,reset=>
                master_rst,cs=>cs_bar,sclk=>sclk,mosi=>mosi,reg_a=>
                data_out_master);
            LCD: test port map(inp=>data_out_master,clk=>clk,rst=>
                lcd_rst,lcd_rw=>lcd_rw,lcd_en=>lcd_en,lcd_rs=>
                lcd_rs,lcd1=>lcd1,detect=>detect);

            led_process: process(clk)
                begin
                if rising_edge(clk) then
                    if clk_counter = 50000000 then
                        data_out_led <= "00000001";
                        if master_rst = '1' then
                            clk_counter <= 0;
                        end if;
                    else
                        data_out_led <= data_out_master(9 downto 2)
                            ;
                        clk_counter <= clk_counter + 1;
                    end if;
                end if;
            end process;
end architecture;
```

Testbench Module: VHDL Code

VHDL Code for Testbench Module

```
library ieee;
use ieee.std_logic_1164.all;

entity testbench is
end testbench;

architecture archi_testbench of testbench is

component master is
    port(
        clk : in std_logic;
        reset : in std_logic;
        miso : in std_logic;
        mosi : out std_logic;
        reg_a : out std_logic_vector(9 downto 0) :="ZZZZZZZZZZ";
        sclk : out std_logic;
        cs : out std_logic
    );
end component;

signal temp_sclk : std_logic;
signal temp_mosi,temp_cs : std_logic;
signal temp_reg_a : std_logic_vector(9 downto 0);
signal reset,clk : std_logic:='0';
signal miso : std_logic;
constant clock_period : time := 20 ns;
begin
miso<='1';
inst1: master port map (clk, reset, miso, temp_mosi, temp_reg_a,
    temp_sclk, temp_cs);

clock_process: process
begin
clk <= not clk after clock_period/2;
wait for clock_period/2;
end process;

reset_process: process
begin
    reset <= '1';
    wait for 10 ns;
    reset <='0';
    wait;
```

```
end process;
end architecture;
```