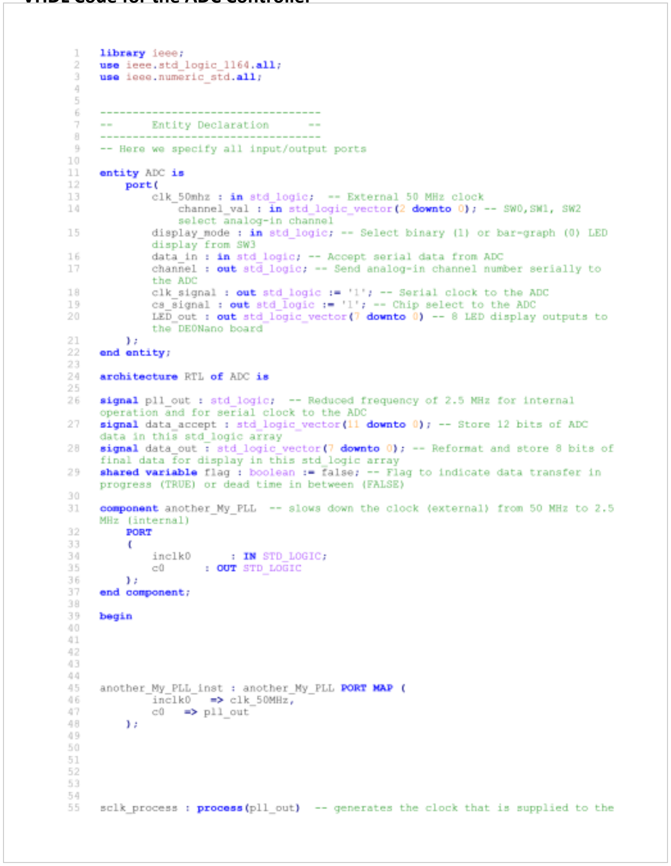
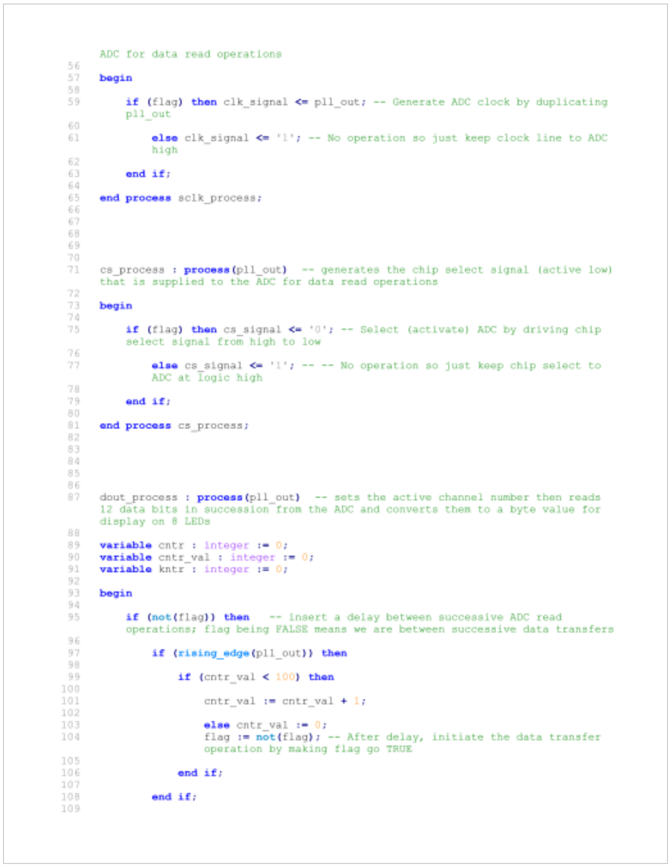
Mark Musil Lab 9 Deliverables

Friday, February 18, 2022 12:27 AM

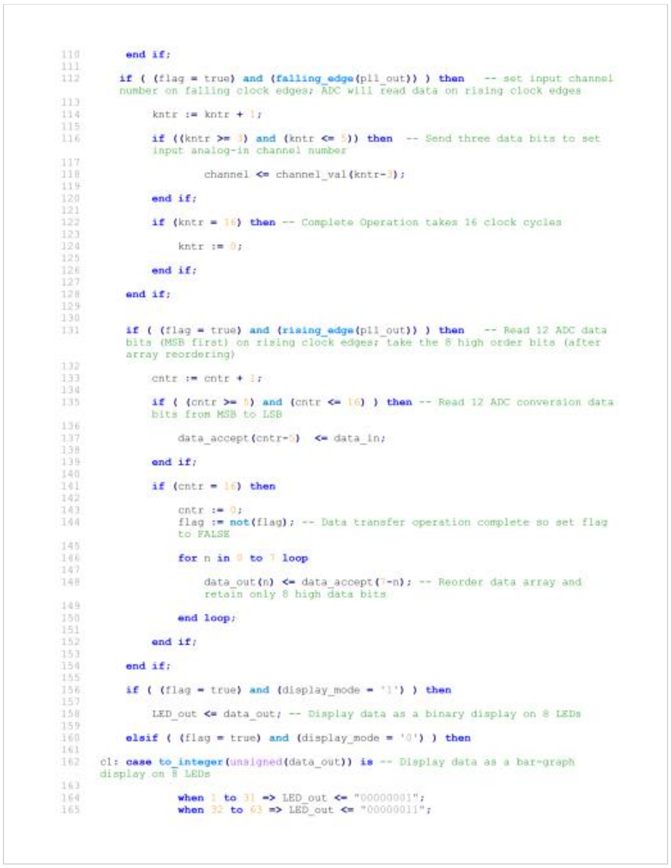
VHDL Code for the ADC Controller



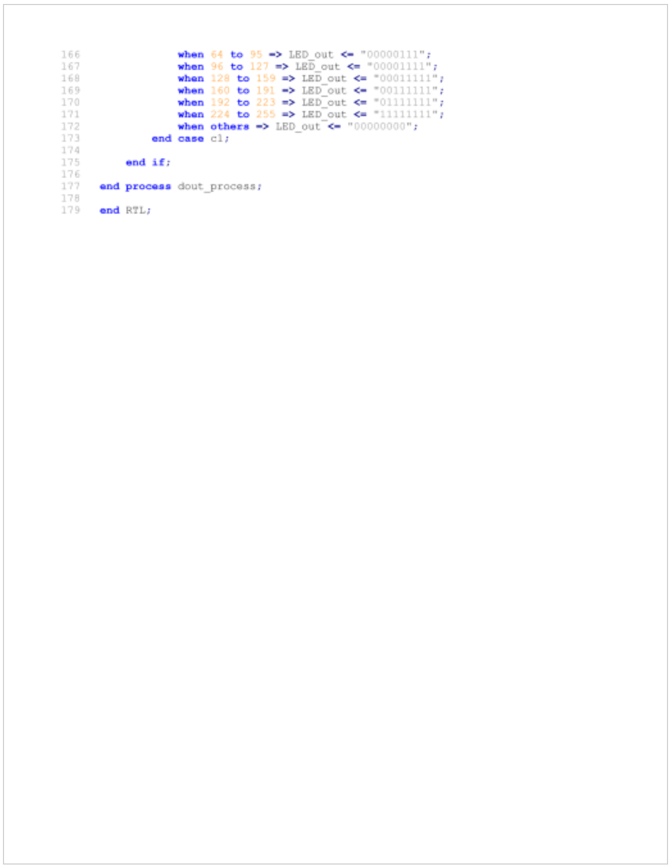
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


----------------------------------
--      Entity Declaration      --
----------------------------------
-- Here we specify all input/output ports

entity ADC is
    port(
        clk_50mhz : in std_logic;  -- External 50 MHz clock
            channel_val : in std_logic_vector(2 downto 0); -- SW0,SW1, SW2 select analog-in channel
        display_mode : in std_logic; -- Select binary (1) or bar-graph (0) LED display from SW3
        data_in : in std_logic; -- Accept serial data from ADC
        channel : out std_logic; -- Send analog-in channel number serially to the ADC
        clk_signal : out std_logic := '1'; -- Serial clock to the ADC
        cs_signal : out std_logic := '1'; -- Chip select to the ADC
        LED_out : out std_logic_vector(7 downto 0) -- 8 LED display outputs to the DE0Nano board
    );
end entity;

architecture RTL of ADC is

signal pll_out : std_logic;  -- Reduced frequency of 2.5 MHz for internal operation and for serial clock to the ADC
signal data_accept : std_logic_vector(11 downto 0); -- Store 12 bits of ADC data in this std_logic array
signal data_out : std_logic_vector(7 downto 0); -- Reformat and store 8 bits of final data for display in this std_logic array
shared variable flag : boolean := false; -- Flag to indicate data transfer in progress (TRUE) or dead time in between (FALSE)

component another_My_PLL  -- slows down the clock (external) from 50 MHz to 2.5 MHz (internal)
    PORT
    (
        inclk0      : IN STD_LOGIC;
        c0      : OUT STD_LOGIC
    );
end component;

begin





another_My_PLL_inst : another_My_PLL PORT MAP (
        inclk0   => clk_50MHz,
        c0   => pll_out
    );






sclk_process : process(pll_out)  -- generates the clock that is supplied to the
```



```vhdl
ADC for data read operations

begin

    if (flag) then clk_signal <= pll_out; -- Generate ADC clock by duplicating pll_out

        else clk_signal <= '1'; -- No operation so just keep clock line to ADC high

    end if;

end process sclk_process;






cs_process : process(pll_out)  -- generates the chip select signal (active low) that is supplied to the ADC for data read operations

begin

    if (flag) then cs_signal <= '0'; -- Select (activate) ADC by driving chip select signal from high to low

        else cs_signal <= '1'; -- -- No operation so just keep chip select to ADC at logic high

    end if;

end process cs_process;






dout_process : process(pll_out)  -- sets the active channel number then reads 12 data bits in succession from the ADC and converts them to a byte value for display on 8 LEDs

variable cntr : integer := 0;
variable cntr_val : integer := 0;
variable kntr : integer := 0;

begin

    if (not(flag)) then   -- insert a delay between successive ADC read operations; flag being FALSE means we are between successive data transfers

        if (rising_edge(pll_out)) then

            if (cntr_val < 100) then

                cntr_val := cntr_val + 1;

                else cntr_val := 0;
                flag := not(flag); -- After delay, initiate the data transfer operation by making flag go TRUE

            end if;

        end if;

```

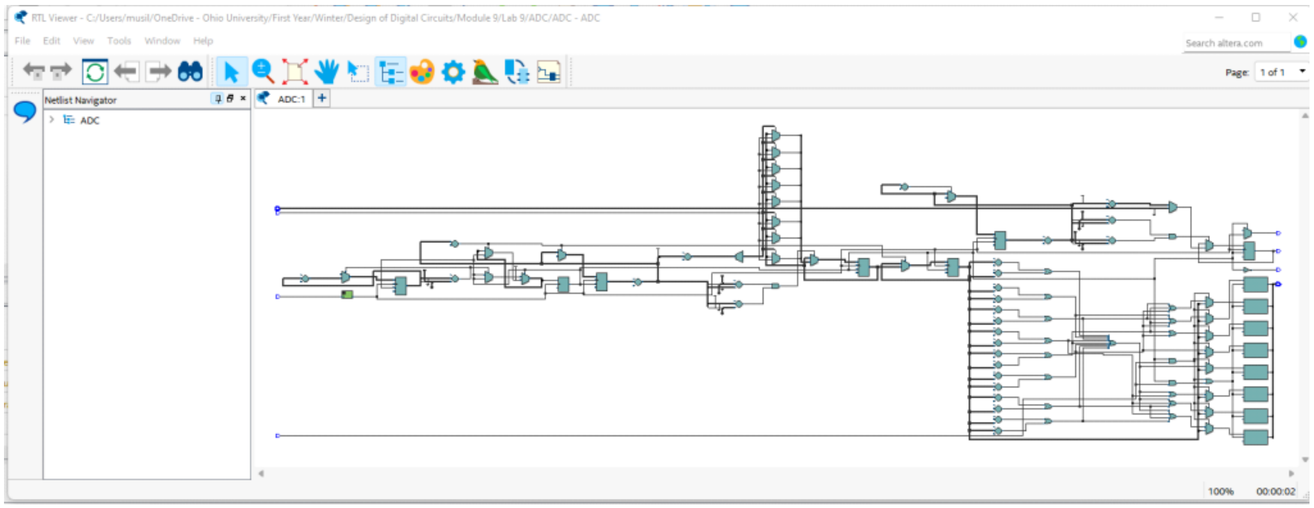


```vhdl
        end if;

      if ( (flag = true) and (falling_edge(pll_out)) ) then   -- set input channel
      number on falling clock edges; ADC will read data on rising clock edges

            kntr := kntr + 1;

            if ((kntr >= 3) and (kntr <= 5)) then  -- Send three data bits to set
            input analog-in channel number

                    channel <= channel_val(kntr-3);

            end if;

            if (kntr = 16) then -- Complete Operation takes 16 clock cycles

                kntr := 0;

            end if;

        end if;


        if ( (flag = true) and (rising_edge(pll_out)) ) then   -- Read 12 ADC data
        bits (MSB first) on rising clock edges; take the 8 high order bits (after
        array reordering)

            cntr := cntr + 1;

            if ( (cntr >= 5) and (cntr <= 16) ) then -- Read 12 ADC conversion data
            bits from MSB to LSB

                data_accept(cntr-5)  <= data_in;

            end if;

            if (cntr = 16) then

                cntr := 0;
                flag := not(flag); -- Data transfer operation complete so set flag
                to FALSE

                for n in 0 to 7 loop

                    data_out(n) <= data_accept(7-n); -- Reorder data array and
                    retain only 8 high data bits

                end loop;

            end if;

        end if;

        if ( (flag = true) and (display_mode = '1') ) then

            LED_out <= data_out; -- Display data as a binary display on 8 LEDs

        elsif ( (flag = true) and (display_mode = '0') ) then

    c1: case to_integer(unsigned(data_out)) is -- Display data as a bar-graph
    display on 8 LEDs

                when 1 to 31 => LED_out <= "00000001";
                when 32 to 63 => LED_out <= "00000011";
```



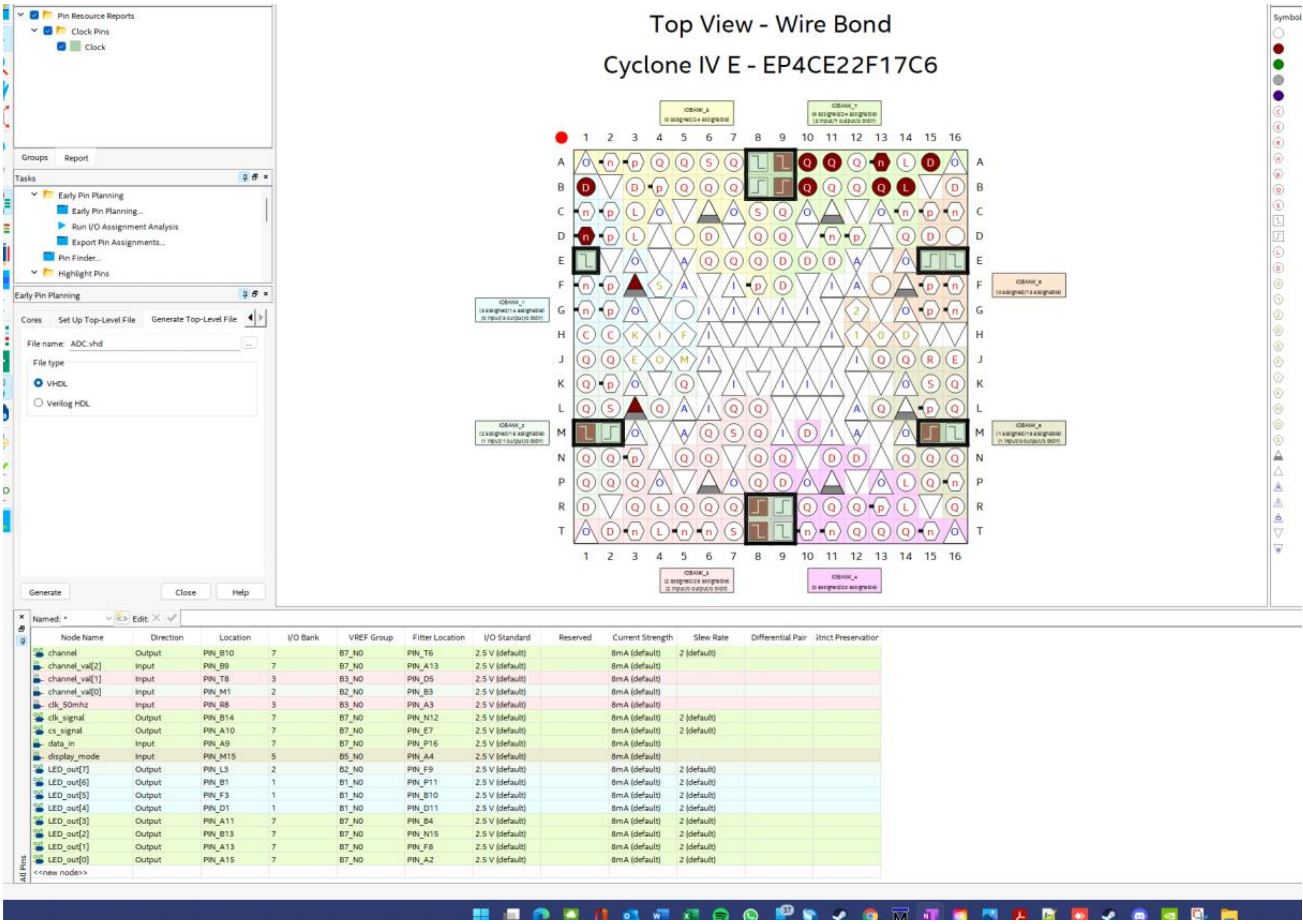
```vhdl
                when 64 to 95 => LED_out <= "00000111";
                when 96 to 127 => LED_out <= "00001111";
                when 128 to 159 => LED_out <= "00011111";
                when 160 to 191 => LED_out <= "00111111";
                when 192 to 223 => LED_out <= "01111111";
                when 224 to 255 => LED_out <= "11111111";
                when others => LED_out <= "00000000";
            end case c1;

        end if;

    end process dout_process;

    end RTL;
```

RTL View


RTL Viewer - C:/Users/musil/OneDrive - Ohio University/First Year/Winter/Design of Digital Circuits/Module 9/Lab 9/ADC/ADC - ADC
File Edit View Tools Window Help
Search altera.com
Page: 1 of 1
Netlist Navigator
ADC
ADC:1
100%
00:00:02

Screen clipping taken: 2/18/2022 12:58 AM

Pin Planner



File name: ADC.vhd

File type: [✓ VHDL] Verilog HDL

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
channel	Output	PIN_B10	7	B7_N0	PIN_T6	2.5 V (default)		8mA (default)	2 (default)		
channel_val[2]	Input	PIN_B9	7	B7_N0	PIN_A13	2.5 V (default)		8mA (default)			
channel_val[1]	Input	PIN_T8	3	B3_N0	PIN_D5	2.5 V (default)		8mA (default)			
channel_val[0]	Input	PIN_M1	2	B2_N0	PIN_B3	2.5 V (default)		8mA (default)			
clk_50mhz	Input	PIN_R8	3	B3_N0	PIN_A3	2.5 V (default)		8mA (default)			
clk_signal	Output	PIN_B14	7	B7_N0	PIN_N12	2.5 V (default)		8mA (default)	2 (default)		
cs_signal	Output	PIN_A10	7	B7_N0	PIN_E7	2.5 V (default)		8mA (default)	2 (default)		
data_in	Input	PIN_A9	7	B7_N0	PIN_P16	2.5 V (default)		8mA (default)			
display_mode	Input	PIN_M15	5	B5_N0	PIN_A4	2.5 V (default)		8mA (default)			
LED_out[7]	Output	PIN_L3	2	B2_N0	PIN_F9	2.5 V (default)		8mA (default)	2 (default)		
LED_out[6]	Output	PIN_B1	1	B1_N0	PIN_P11	2.5 V (default)		8mA (default)	2 (default)		
LED_out[5]	Output	PIN_F3	1	B1_N0	PIN_B10	2.5 V (default)		8mA (default)	2 (default)		
LED_out[4]	Output	PIN_D1	1	B1_N0	PIN_D11	2.5 V (default)		8mA (default)	2 (default)		
LED_out[3]	Output	PIN_A11	7	B7_N0	PIN_B4	2.5 V (default)		8mA (default)	2 (default)		
LED_out[2]	Output	PIN_B13	7	B7_N0	PIN_N15	2.5 V (default)		8mA (default)	2 (default)		
LED_out[1]	Output	PIN_A13	7	B7_N0	PIN_F8	2.5 V (default)		8mA (default)	2 (default)		
LED_out[0]	Output	PIN_A15	7	B7_N0	PIN_A2	2.5 V (default)		8mA (default)	2 (default)		
<<new node>>											

Screen clipping taken: 2/18/2022 2:08 AM