Lab 7

Carter Owens, Kyle Turley

Procedures

Introduction. The goal of this lab was to design and implement a simple ADC system that displays a voltage to the 7-Segment display.

The procedures for this lab were as follows:

1. Create a Quartus project for the ADC 7-Segment display
2. Create PLL and ADC modules from the IP Catalog
3. Design and implement a ADC module that reads a voltage from header pins
4. Create a simple breadboard design that utilizes the FPGA board header pins
5. Connect the modules and test the system on the FPGA board

Key Requirements: - Read the raw voltage values with an ADC - Send the ADC values to the 7-Segment display

Issues, Errors, and Stumbles. The first major hurdle we had to overcome was determining which pins we needed to wire to our potentiometer to the board. We had to read the DE-10 schematic sheet as well as look at the ADC sample project to know for sure where the FPGA will be sending and receiving ADC related signals. Next, had issues when trying to use the ADC component in our project. In addition, when we implemented our ADC onto our board, the 7-Segments only reported a zero value. After long, long inspection, we determined that we needed to pulse the `cmd_valid` signal, and the best way to do so was in a state machine.

Results

To test our board, we decided that we should directly connect the 5V output on the header to the first channel of the ADC. Once our implementation was working, we found that the hex values were very high: beyond `x48c`. When the wire was disconnected, the ADC reported a value of `x001` or `x002`.

Figures and Code

See appendix for source code.

Conclusion

In this lab, an (ADC) system was designed and implemented to display a voltage reading on a 3-digit 7-segment display. The project met all key requirements: the reading of raw voltage values with an ADC, and the subsequent display of these values.

The process, however, was not without its challenges. Initial difficulties arose in identifying the correct pins for wiring the potentiometer to the FPGA board, which required careful examination of the DE-10 schematic and sample projects. Further complications emerged during the integration of the ADC component. A significant hurdle was encountered when the 7-segment displays consistently showed a zero value. Through extensive troubleshooting, it was determined that the cmd_valid signal needed to be pulsed, a problem that was effectively resolved by implementing a state machine.

Upon successful implementation, testing the system by directly connecting a 5V output to the ADC's first channel yielded high hexadecimal values, exceeding x48c. This result indicates a successful conversion of the analog input to a digital signal, demonstrating the functionality of the designed system. The successful completion of this lab provides a foundational understanding of ADC principles and their practical application in digital systems.

Appendix

ADC_LAB (VHDL)

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

entity adc_lab is
    port (
        -- INPUTS --
        clk : in std_logic;

        -- OUTPUTS --
        hex0 : out std_logic_vector(7 downto 0);
        hex1 : out std_logic_vector(7 downto 0);
        hex2 : out std_logic_vector(7 downto 0)
    );
end entity adc_lab;

architecture behavioral of adc_lab is
    -- SIGNALS & CONSTANTS --
    type states is (IDLE, SENDING);

    signal state : states;

    signal clk_10mhz : std_logic;
    signal locked : std_logic;

    signal cmd_valid : std_logic := '0';
    signal cmd_ready : std_logic;
    signal resp_valid : std_logic;
    signal resp_data : std_logic_vector(11 downto 0);

    signal seg_count : unsigned(11 downto 0) := (others => '0');
    signal count_1hz : integer;


    -- COMPONENTS
    component seg
         port(
            point   : in std_logic;
            count   : in unsigned(3 downto 0);
            output: out std_logic_vector(7 downto 0)
         );
    end component seg;

    component pll_10mhz2 is
        port (
            inclk0  : in std_logic  := '0';
            c0          : out std_logic;
            locked  : out std_logic
        );
    end component pll_10mhz2;

    component adc is
        port (
```

```
            clock_clk              : in  std_logic                     := 'X';             -- clk
            reset_sink_reset_n     : in  std_logic                     := 'X';             -- reset_n
            adc_pll_clock_clk      : in  std_logic                     := 'X';             -- clk
            adc_pll_locked_export  : in  std_logic                     := 'X';             -- export
            command_valid          : in  std_logic                     := 'X';             -- valid
            command_channel        : in  std_logic_vector(4 downto 0)  := (others => 'X'); -- channel
            command_startofpacket  : in  std_logic                     := 'X';             -- startofpacke
            command_endofpacket    : in  std_logic                     := 'X';             -- endofpacket
            command_ready          : out std_logic;                                        -- ready
            response_valid         : out std_logic;                                        -- valid
            response_channel       : out std_logic_vector(4 downto 0);                     -- channel
            response_data          : out std_logic_vector(11 downto 0);                    -- data
            response_startofpacket : out std_logic;                                        -- startofpacke
            response_endofpacket   : out std_logic                                         -- endofpacket
        );
    end component adc;

begin

    -- INSTANCES --
    seg0_impl : seg port map(point => '0',  count => seg_count(3 downto 0), output => hex0);
    seg1_impl : seg port map(point => '0',  count => seg_count(7 downto 4), output => hex1);
    seg2_impl : seg port map(point => '0',  count => seg_count(11 downto 8), output => hex2);

    pll_10mhz_impl : pll_10mhz2 port map (inclk0 => clk, c0 => clk_10mhz, locked => locked);

    adc_impl : adc
        port map (
            clock_clk              => clk,                    --          clock.clk
            reset_sink_reset_n     => '1',                    --     reset_sink.reset_n
            adc_pll_clock_clk      => clk_10mhz,              --  adc_pll_clock.clk
            adc_pll_locked_export  => locked,                 -- adc_pll_locked.export
            command_valid          => cmd_valid,              --        command.valid
            command_channel        => "00001",                --               .channel
            command_startofpacket  => 'X',                    --               .startofpacket
            command_endofpacket    => 'X',                    --               .endofpacket
            command_ready          => cmd_ready,              --               .ready
            response_valid         => resp_valid,             --       response.valid
            response_channel       => open,                   --               .channel
            response_data          => resp_data,              --               .data
            response_startofpacket => open,                   --               .startofpacket
            response_endofpacket   => open                    --               .endofpacket
        );

    -- DIRECT BEHAVIOR --

    process (clk) begin
        if rising_edge(clk) then
            case (state) is
                when IDLE =>
                    count_1hz <= count_1hz + 1;

                    if count_1hz > 50_000_000 then
                        cmd_valid <= '1';
                    end if;
```

```
                    if cmd_ready = '1' then
                        state <= SENDING;
                    end if;

                when SENDING =>
                    cmd_valid <= '0';

                    if resp_valid = '1' then
                        state <= IDLE;
                        count_1hz <= 0;
                        seg_count <= unsigned(resp_data);
                    end if;
            end case;
        end if;
    end process;

end architecture behavioral;
```