```
--################### MCD ###########################################################################
--# Company:    Eklektik Design
--# Engineer:   Micah Richards
--#
--# Create Date:    21:29 3/18/17
--# Module Name:    Monitor Program
--# Project Name:   MicroComputer Design
--# Target Devices: MCD Board
--# Tool versions:  Xilinx 14.7
--# Description:    Chip select controls
--#
--#################################################################################################

--################### Libraries #####################################################################
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

--################### Entity ########################################################################
entity Main_Code is
    Port (

--################### Ports #########################################################################
                CPLD_CLK, CPLD_Button: in STD_LOGIC;
                Read_Write, Add_Data_Valid, L_Data_Ready, H_Data_Ready: in STD_LOGIC;
                ROM_H_Enable, ROM_H_Output, ROM_H_Input: out STD_LOGIC;
                ROM_L_Enable, ROM_L_Output, ROM_L_Input: out STD_LOGIC;
                RAM_H_Enable, RAM_H_Output, RAM_H_Input: out STD_LOGIC;
                RAM_L_Enable, RAM_L_Output, RAM_L_Input: out STD_LOGIC;
                DUART_Enable, DUART_Read_Write, DUART_Reset: out STD_LOGIC;
                DUART_Data_ACK: in STD_LOGIC;
                CPU_Clock, CPU_Data_ACK, CPU_Reset, CPU_Halt: out STD_LOGIC;
                CPU_Valid_Periph_Add, CPU_Bus_Grant_ACK, CPU_Bus_ERR, CPU_Bus_REQ: out STD_Logic;
                CPU_Read_Write, CPU_H_Data_Ready, CPU_L_Data_Ready, CPU_Enable, CPU_Add_Valid: in STD_LOGIC;
                CPU_Bus_Grant, CPU_Valid_Memory_Add: in STD_Logic;
                ADDR: in STD_LOGIC_VECTOR (3 downto 0);
                LED: out STD_LOGIC_VECTOR (3 downto 0)
            );
end Main_Code;

--################### Behavior ######################################################################
architecture Behavioral of Main_Code is

--################### Signals #######################################################################
signal I_Count1: integer range 0 to 4500 := 1; --#### First integer for clock divider
signal I_Count2: integer range 0 to 25:= 1; --#### Second integer for clock divider
signal I_CPU_Clock: STD_LOGIC := '0'; --#### 7.5kHz clock for CPU
signal I_Reset_Triggered: STD_Logic:= '0'; --#### Asynchronous Reset Signal

begin

--################### Chip Select ###################################################################
--# Purpose: Interprets the CPU signals to enable and disable the proper chips
--#
--#################################################################################################
process (CPLD_CLK)
begin
if (rising_edge(CPLD_CLK)) then
    LED(2)          <= '1';
    LED(1)          <= '1';
    ROM_H_Enable    <= '1';
    ROM_H_Output    <= '1';
    ROM_L_Enable    <= '1';
    ROM_L_Output    <= '1';
    RAM_H_Enable    <= '1';
    RAM_H_Input     <= '1';
    RAM_H_Output    <= '1';
    RAM_L_Enable    <= '1';
    RAM_L_Input     <= '1';
    RAM_L_Output    <= '1';
    DUART_Enable    <= '1';
    CPU_Data_ACK   <= '1';

    if (CPU_Read_Write = '1' and CPU_Add_Valid = '0' ) then --#### 1 Reading
        if (ADDR = "0000") then                             --#### ROM Access
            if (CPU_H_Data_Ready = '0') then                --#### High
                ROM_H_Enable <= '0';
                ROM_H_Output <= '0';
                CPU_Data_ACK <= '0';
                LED(2) <= '0';
            end if;

            if (CPU_L_Data_Ready = '0') then                --#### Low
                ROM_L_Enable <= '0';
                ROM_L_Output <= '0';
                CPU_Data_ACK <= '0';
                LED(2) <= '0';
            end if;

        elsif (ADDR = "1111") then                          --#### RAM Access
            if (CPU_H_Data_Ready = '0') then                --#### High
                RAM_H_Enable <= '0';
                RAM_H_Output <= '0';
                CPU_Data_ACK <= '0';
                LED(2) <= '0';
            end if;

            if (CPU_L_Data_Ready = '0') then                --#### Low
                RAM_L_Enable <= '0';
                RAM_L_Output <= '0';
                CPU_Data_ACK <= '0';
                LED(2) <= '0';
            end if;

        elsif (ADDR = "0011") then                          --#### DUART Access
            if (CPU_H_Data_Ready = '0' or CPU_L_Data_Ready = '0') then
```

```
                DUART_Enable <= '0';
                LED(1) <= '0';
                DUART_Read_Write <= '1';
                CPU_Data_ACK <= DUART_Data_ACK;
                LED(2) <= DUART_Data_ACK;
            end if;
        end if;
    elsif (CPU_Read_Write = '0' and CPU_Add_Valid = '0') then --#### Writing
        if (ADDR = "1111") then                             --#### RAM Access
            if (CPU_H_Data_Ready = '0') then
                RAM_H_Enable <= '0';                        --#### High
                RAM_H_Input <= '0';
                CPU_Data_ACK <= '0';
                LED(2) <= '0';
            end if;

            if (CPU_L_Data_Ready = '0') then                --#### Low
                RAM_L_Enable <= '0';
                RAM_L_Input <= '0';
                CPU_Data_ACK <= '0';
                LED(2) <= '0';
            end if;
        elsif (ADDR = "0011") then                          --#### DUART Access
            if (CPU_H_Data_Ready = '0' or CPU_L_Data_Ready = '0') then
                DUART_Enable <= '0';
                LED(1) <= '0';
                DUART_Read_Write <= '0';
                CPU_Data_ACK <= DUART_Data_ACK;
                LED(2) <= DUART_Data_ACK;

            end if;
        end if;
    end if;
end if;

end process;

--################### Clock Select ##################################################################
--# Purpose: Allows the user to select a clock speed for
--#           Function Select
--#
--#################################################################################################
process (CPLD_CLK)
begin
if (rising_edge(CPLD_CLK)) then
        I_Count1 <= I_Count1 + 1;
        if (I_Count1 = 4500) then
            I_Count1 <= 1;
            I_Count2 <= I_Count2 + 1;
            if (I_Count2 = 25) then --#### 500Hz Divide by 4500*25
                if (CPLD_Button = '0') then
                    I_Reset_Triggered <= '1';
                else
                    I_Reset_Triggered <= '0';
                end if;
                I_Count2 <= 1;
            end if;
        end if;
    end if;
end process;

--################### Reset #########################################################################
--# Purpose: Allows the user to reset the board via the button
--#
--#################################################################################################
process (I_Reset_Triggered)
begin
    if (I_Reset_Triggered = '1') then
        CPU_Halt <= '0';
        CPU_Reset <= '0';
        DUART_Reset <= '1';
    else
        CPU_Halt <= 'Z';
        CPU_Reset <= 'Z';
        DUART_Reset <= '1';
    end if;
end process;

--################### Immediate Updates #############################################################
CPU_Clock <= CPLD_CLK;          --#### write 7.5 kHz Clock to CPU
LED(0) <= CPU_Read_Write;
LED(3) <= CPU_Add_Valid;


--################### Unused Pins Tied High #########################################################
CPU_Bus_ERR                     <= '1';
CPU_Valid_Periph_Add            <= '1';
CPU_Bus_Grant_ACK               <= '1';
CPU_Bus_REQ                     <= '1';
end Behavioral;
```