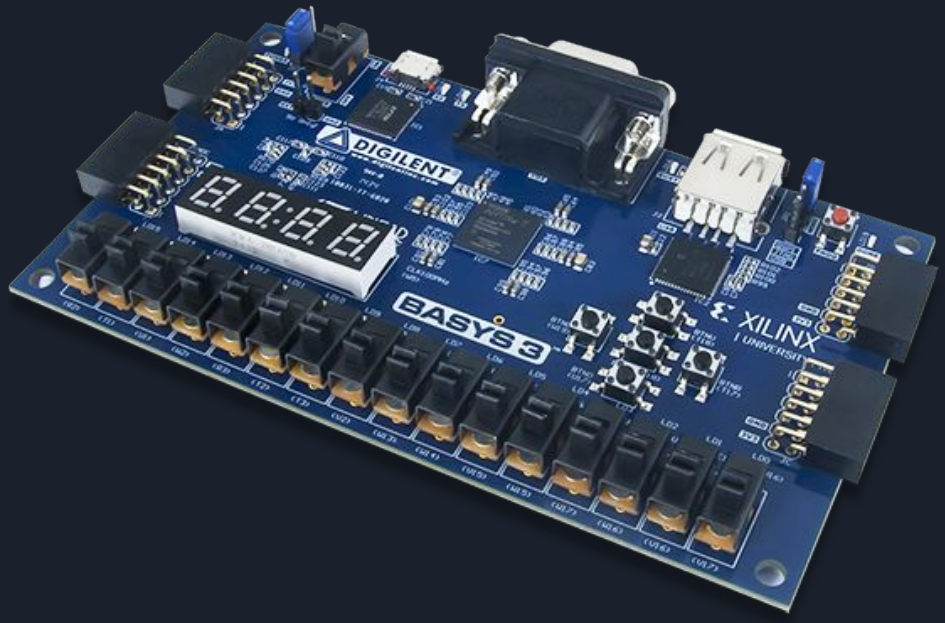
Banner on Seven Segment Displays

With Scrolling Messages at a Constant Speed

Performed by:
Alain Sfeir
Victoria Poirier
Jessica Odish
Randa Odeesh

Project Description



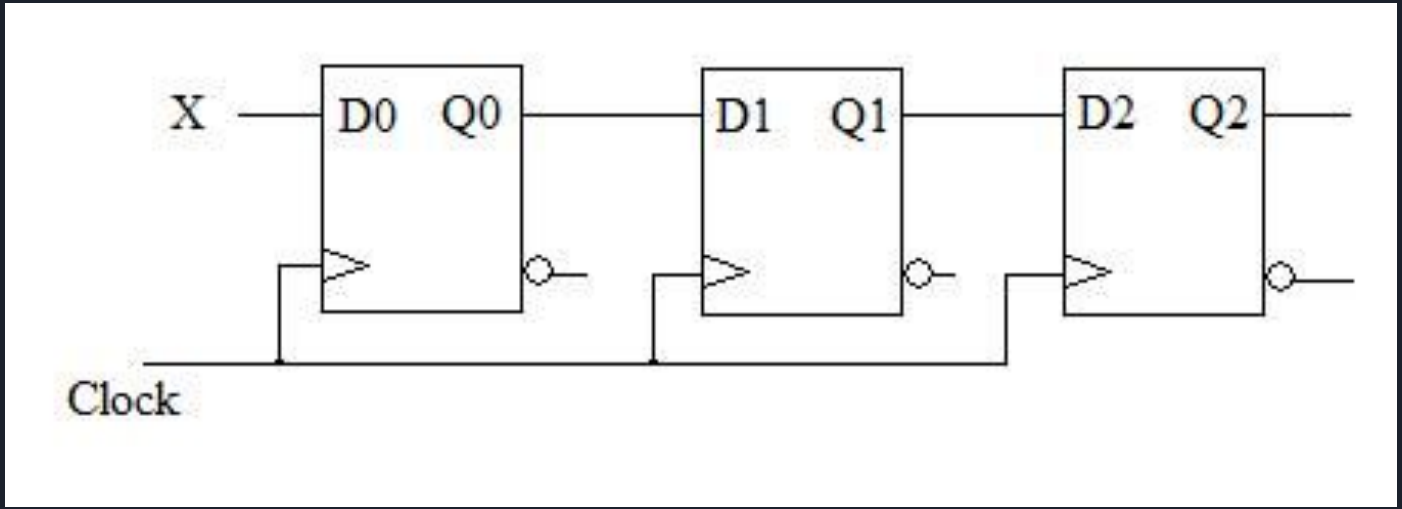
The main purpose of this project is to build and design a circuit that will interface ALL eight segment displays onto the Basys 3 Artix-7 FPGA Trainer Board

Using the switches will allow for the banner to display scrolling messages at a constant speed.

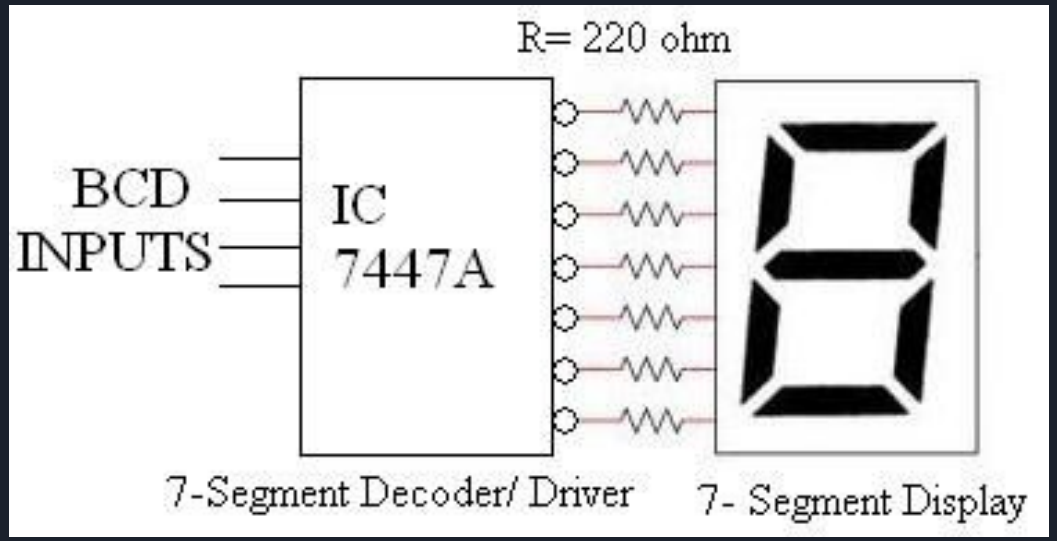
Scroll Speed Clock Module

- Controls speed of the scrolling message
- Runs as a clock
- Runs the message on constant speed depending on the one picked via a switch

Ram Shifter

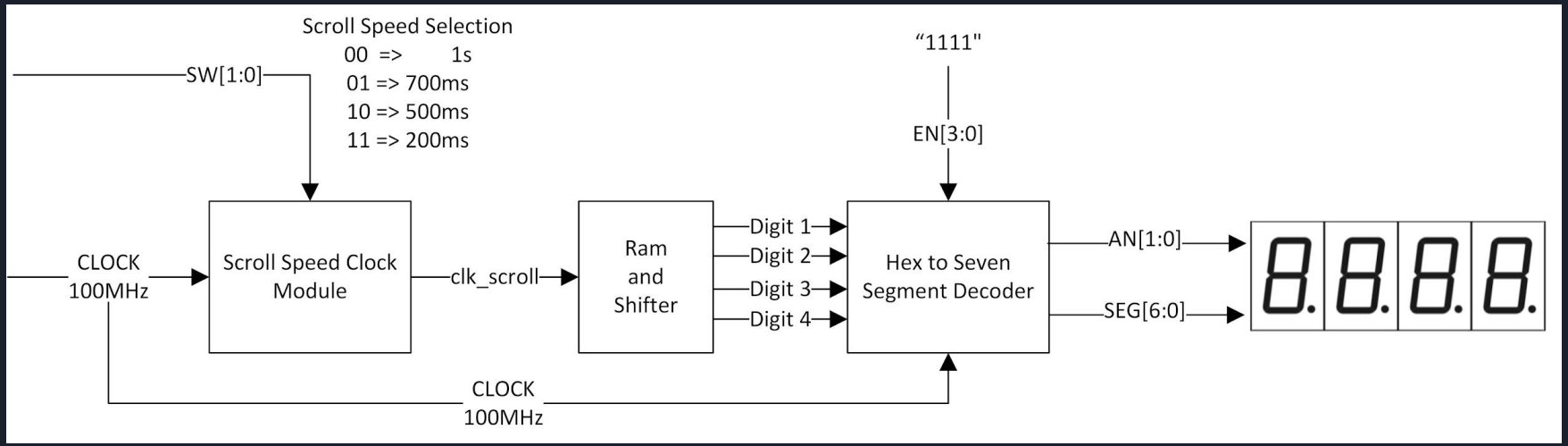


Hex to Seven Segment Decoder



Circuit Design

Keeping in mind what we wanted our circuit to do and what files were required to make that happen, this overall circuit design to the right was created to implement all the elements and components discussed previously



Experimental Setup

- Vivado’s design tools were used to check analyze the project ‘s behavioral simulation.
- Creating a test bench consists of sample input bits and output of the top file to analyze the simulations before wiring to the Basys 3 A7 hardware.
- The output parameters for the design were determined from the simulation to allow debugging the code if the results are not as expected.

RAM Shifter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ram_shifter is
    Port (
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        digit_1 : out  STD_LOGIC_VECTOR (3 downto 0);
        digit_2 : out  STD_LOGIC_VECTOR (3 downto 0);
        digit_3 : out  STD_LOGIC_VECTOR (3 downto 0);
        digit_4 : out  STD_LOGIC_VECTOR (3 downto 0)
        );
end ram_shifter;

architecture Behavioral of ram_shifter is
    constant totalCharacters : integer := 14;
    signal H :  STD_LOGIC_VECTOR(3 downto 0) := "1010";
    signal E :  STD_LOGIC_VECTOR(3 downto 0) := "1011";
    signal L :  STD_LOGIC_VECTOR(3 downto 0) := "1100";
    signal O :  STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal C :  STD_LOGIC_VECTOR(3 downto 0) := "1101";
    signal TWO :  STD_LOGIC_VECTOR(3 downto 0) := "0010";
    signal SEVEN :  STD_LOGIC_VECTOR(3 downto 0) := "0111";
    signal ZERO :  STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal SPACE :  STD_LOGIC_VECTOR(3 downto 0) := "1111";
    signal shifter : STD_LOGIC_VECTOR((totalCharacters*4)-1 downto 0) := H&E&L&L&O&SPACE&E&C&E&TWO&SEVEN&ZERO&ZERO&SPACE;
begin


    process (clk,reset)
    begin
        if (reset = '1') then
            shifter   <= H&E&L&L&O&SPACE&E&C&E&TWO&SEVEN&ZERO&ZERO&SPACE;
        elsif rising_edge(clk) then
            shifter <= shifter((totalCharacters*4)-5 downto 0) & shifter((totalCharacters*4)-1 downto (totalCharacters*4)-4);
        end if;
    end process;
    digit_4 <= shifter(((totalCharacters*4)-1)-(4*0) downto (totalCharacters*4)-(4*1));
    digit_3 <= shifter(((totalCharacters*4)-1)-(4*1) downto (totalCharacters*4)-(4*2));
    digit_2 <= shifter(((totalCharacters*4)-1)-(4*2) downto (totalCharacters*4)-(4*3));
    digit_1 <= shifter(((totalCharacters*4)-1)-(4*3) downto (totalCharacters*4)-(4*4));


end Behavioral;
```

Scroll Speed Code

```
use IEEE.STD_LOGIC_1164.ALL;
entity scroll_speed_clock is
    Port (
        clk       : in  std_logic;
        sel       : in  std_logic_vector(1 downto 0);
        reset     : in  std_logic;
        clk_scroll : out std_logic
    );
end scroll_speed_clock;

architecture Behavioral of scroll_speed_clock is
    constant threshold_1s : integer := 49999999; --
    constant threshold_700ms : integer := 34999999; --
    constant threshold_500ms : integer := 24999999; --
    constant threshold_200ms : integer := 9999999; --

    signal threshold : integer;
    signal temporal : std_logic;
    signal counter  : integer range 0 to 99999999 := 0
begin

    threshold <= threshold_1s when sel = "00" else
                 threshold_700ms when sel = "01" else
                 threshold_500ms when sel = "10" else
                 threshold_200ms;

    process (reset, clk) begin
        if (reset = '1') then
            temporal <= '0';
            counter  <= 0;
        elsif (rising_edge(clk)) then
            if (counter = threshold) then
                temporal <= NOT(temporal);
                counter  <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    clk_scroll <= temporal;

end Behavioral;
```

Decoder

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity seven_seg_controller is
    Port (
        clk     : in  STD_LOGIC;
        reset   : in  STD_LOGIC;
        digit_1 : in  STD_LOGIC_VECTOR (3 downto 0);
        digit_2 : in  STD_LOGIC_VECTOR (3 downto 0);
        digit_3 : in  STD_LOGIC_VECTOR (3 downto 0);
        digit_4 : in  STD_LOGIC_VECTOR (3 downto 0);
        en      : in  STD_LOGIC_VECTOR (3 downto 0);
        seg     : out STD_LOGIC_VECTOR (6 downto 0);
        an      : out STD_LOGIC_VECTOR (3 downto 0)
    );
end seven_seg_controller;

architecture Behavioral of seven_seg_controller is
    constant threshold : integer := 99999;
    signal digit_num : STD_LOGIC_VECTOR (3 downto 0) := "1110";
    signal digit     : STD_LOGIC_VECTOR (3 downto 0);
    signal counter   : INTEGER RANGE 0 TO threshold := 0;
    signal tick      : STD_LOGIC                    := '0';
begin

    -- 1ms Counter
    process (clk,reset)
    begin
        if (reset = '1') then
            counter   <= 0;
            digit_num <= "1110";
        elsif rising_edge(clk) then
            if (counter = threshold) then
                counter   <= 0;
                digit_num <= digit_num(2 downto 0) & digit_num(3);
            else
                counter   <= counter + 1;
                digit_num <= digit_num;
            end if;
        end if;
    end process;
```

```vhdl
    -- Digit Selction for each 7 segment display
    process(digit_num, digit_1, digit_2, digit_3, digit_4)
    begin
        case digit_num is
            -- Digit for Display 1
            when "1110" => digit <= digit_1;
            -- Digit for Display 2
            when "1101" => digit <= digit_2;
            -- Digit for Display 3
            when "1011" => digit <= digit_3;
            -- Digit for Display 4
            when "0111" => digit <= digit_4;
            when others => digit <= digit_1;
        end case;
    end process;

    -- Each 7 Segment Enable Logic
    an(0) <= digit_num(0) when en(0) = '1' else '1';
    an(1) <= digit_num(1) when en(1) = '1' else '1';
    an(2) <= digit_num(2) when en(2) = '1' else '1';
    an(3) <= digit_num(3) when en(3) = '1' else '1';

    -- Hex to Seven Segment
    process(digit)
    begin
        case digit is
            when "0000" => seg <= "1000000"; -- "0"
            when "0001" => seg <= "1111001"; -- "1"
            when "0010" => seg <= "0100100"; -- "2"
            when "0011" => seg <= "0110000"; -- "3"
            when "0100" => seg <= "0011001"; -- "4"
            when "0101" => seg <= "0010010"; -- "5"
            when "0110" => seg <= "0000010"; -- "6"
            when "0111" => seg <= "1111000"; -- "7"
            when "1000" => seg <= "0000000"; -- "8"
            when "1001" => seg <= "0010000"; -- "9"
            when "1010" => seg <= "0001001"; -- H
            when "1011" => seg <= "0000110"; -- E
            when "1100" => seg <= "1000111"; -- L
            when "1101" => seg <= "1000110"; -- C
            when "1110" => seg <= "0111111"; -- -
            when "1111" => seg <= "1111111"; -- SPACE
            when others => seg <= "1111111"; -- SPACE
```

Demo