MOTIVATIONS

• Displaying information for people to see at a glance.
• Utilizing a display technology that can be seen at all times of the day.
• Further our knowledge in how specific parts from the class interact with each other.
FINITE STATE MACHINE

Transitions: process (reset, clock, E_fsm)
begin
  if reset = '0' then
    y <= S1;
  elsif (clock'event and clock = '1') then
    if E_fsm = '1' then
      case y is
        when S1 => y <= S2;
        when S2 => y <= S3;
        when S3 => y <= S4;
        when S4 => y <= S5;
        when S5 => y <= S6;
        when S6 => y <= S7;
        when S7 => y <= S8;
        when S8 => y <= S1;
        when others => y <= S1;
      end case;
    end if;
  end if;
end process;

Outputs: process (y)
begin
  case y is
    when S1 => z <= "0000";
    when S2 => z <= "0001";
    when S3 => z <= "0100";
    when S4 => z <= "0110";
    when S5 => z <= "1000";
    when S6 => z <= "1010";
    when S7 => z <= "1100";
    when S8 => z <= "1111";
    when others => z <= "0000";
  end case;
end process;
entity rightshifter is
generic (KBR: STRING := "LEFT");
  Port (a in STD_LOGIC_VECTOR (31 downto 0);
    qout : out STD_LOGIC_VECTOR (31 downto 0);
    w : in STD_LOGIC_VECTOR (3 downto 0);
    E : in STD_LOGIC;
    SRt, resetn : in STD_LOGIC;
    clk : in STD_LOGIC);
end rightshifter;
architecture Behavioral of rightshifter is
signal Qt : std_logic_vector (31 downto 0);
begin
  --assert (DIR = "LEFT" or DIR = "RIGHT")
  -- report "DIR can only be LEFT or RIGHT"
  -- severity error;
  process (E, SRt, w, clk, Qt, resetn)
  begin
    if resetn = '1' then
      Qt <= (others => '1');
    elsif (clk'event and clk = '1') then
      if E = '1' then
        if w = '1' then
          Qt <= a;
        else
          if DIR = "RIGHT" then
            Qt(31 downto 20) <= w;
            for i in 27 downto 0 loop
              Qt(i) <= Qt(i+1);
            end loop;
          end if;
          end if;
        end if;
      end if;
    end process;
    aout <= Qt;
    --q1: if DIR = "LEFT" generate
    -- shiftout <= Qt(5);
    -- end generate;
  end Behavioral;
TOP LEVEL DISPLAY OUTPUT
LAYOUT

- Reset (Yellow)
- Enable (Blue)
- Load (Purple)
- Words (Red)
PROBLEMS

• Displaying the words at the correct location.

• Actually getting the words to scroll across the 7-segment displays.

• Correctly scrolling the words off the screen and back onto the screen continuously.

• Incorrect finite state machine.
CONCLUSION

• Successful use of a large shifter.

• Continuous scrolling of output word.

• Improvements
  • Could be more intricate while utilizing a better display.
  • Could not use the full alphabet due to the drawbacks of the 7-segment display.