Lesson 28 – ICE 6 and Lab 4 Introduction and Tips

UNITED STATES AIR FORCE ACADEMY

ECE 281

Lesson 28 Notes

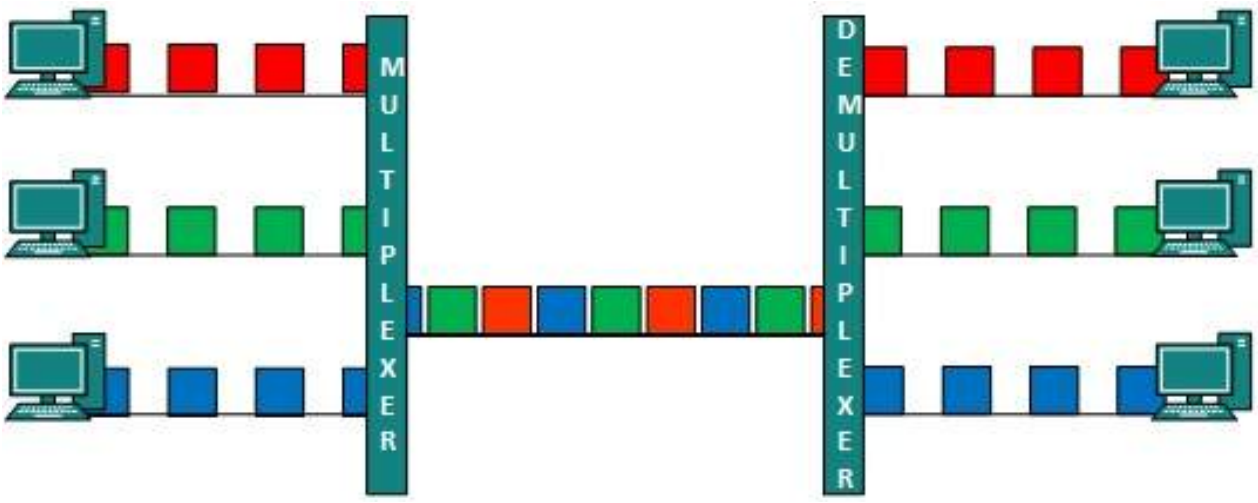
Objectives:

1. Introduce Lab 4 requirements and strategy for completion
2. Review or introduce concept of Time Division Multiplexing (TDM)
3. Practice using VHDL to accomplish TDM

Time Division Multiplexing:

For those of you that have already had ECE 215 or 315, you were introduced to a concept known as Time Division Multiplexing or TDM.

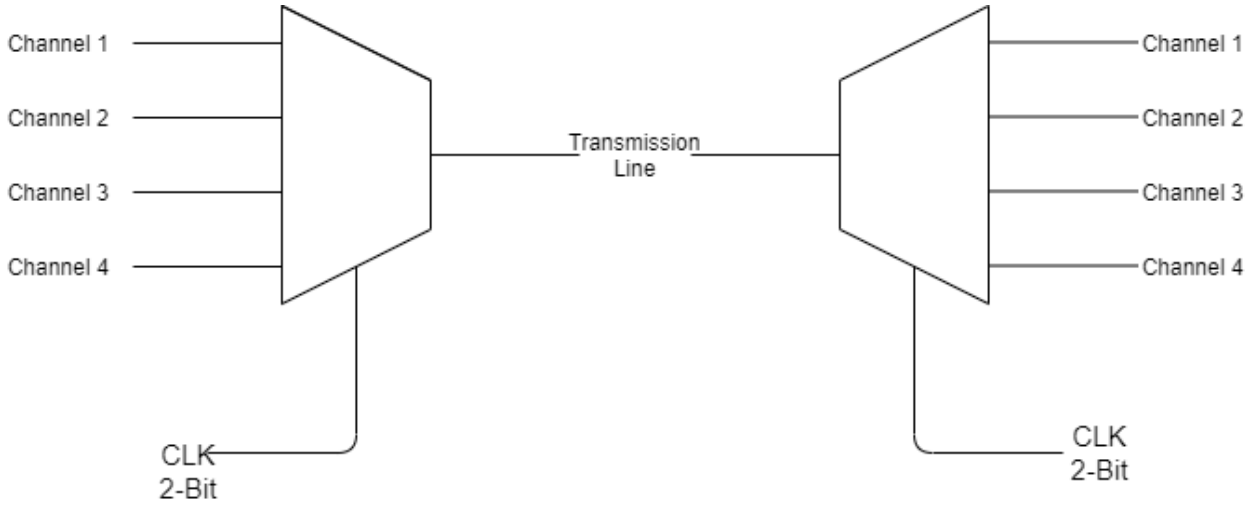
Time Division Multiplexing - is a method of transmitting and receiving independent signals over a common signal path by means of synchronized switches at each end of the transmission line so that each signal appears on the line only a fraction of time in an alternating pattern. This technique is commonly employed where the transmission medium has much higher data rate than any of the connected channels. TDM is heavily employed in modern digital communications.



ICE 6 – TDM Simulation

In ICE 6 you are going to implement TDM functionality that will ultimately be used in Lab 4. More on that soon….

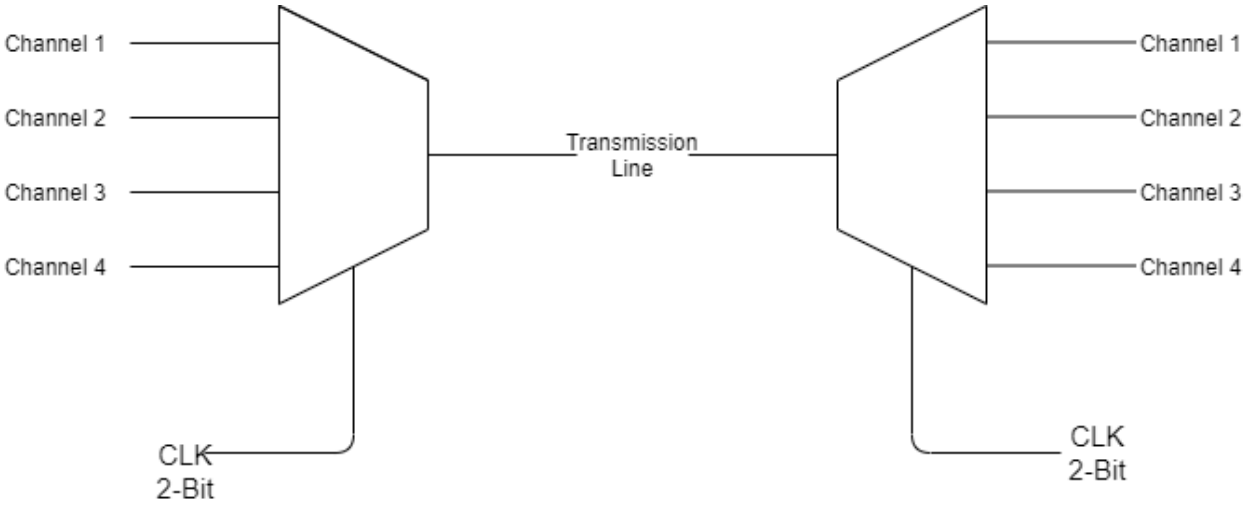
A key concept in TDM is synchronization




```
entity TDM4 is
    generic ( constant k_WIDTH : natural  := 4); -- bits in input and output
    Port ( i_CLK        : in  STD_LOGIC;
           i_RESET      : in  STD_LOGIC; -- asynchronous
           i_D3         : in  STD_LOGIC_VECTOR (k_WIDTH - 1 downto 0);
           i_D2         : in  STD_LOGIC_VECTOR (k_WIDTH - 1 downto 0);
           i_D1         : in  STD_LOGIC_VECTOR (k_WIDTH - 1 downto 0);
           i_D0         : in  STD_LOGIC_VECTOR (k_WIDTH - 1 downto 0);
           o_DATA       : out STD_LOGIC_VECTOR (k_WIDTH - 1 downto 0);
           o_SEL        : out STD_LOGIC_VECTOR (3 downto 0) -- selected data line (one-cold)
    );
end TDM4;
```

ICE 6 will have you working with 4 datasets (i_D3, i_D2, … , i_D0). In this case, that datasets will be remaining constant, **BUT THEY DO NOT HAVE TO.** ICE 6 will then have you create a single output vector (o_DATA) that will sequentially show data from each of the 4 data sets.

Question – What tools have you been given to this point to implement the TDM functionality given a 2-bit clock for synchronization of the inputs and outputs in the figure above?



Initial Goal is to create a clock that instead of just high and low, it will have 4 separate values to act as the mux and de-mux select lines.

Step 1.) How could we create a process that counted from 00 to 11 on every rising edge of the clock?

```
-- 2 Bit counter Process ----------------------------
-- counter rolls over automatically
-- asynchronous reset to "00"
twoBitCounter_proc : process(i_CLK, i_RESET)
begin
    if i_RESET = '1' then
        f_sel <= "00";
    elsif rising_edge(i_CLK) then
        f_sel <= f_sel + 1;
    end if;
end process twoBitCounter_proc;
-----------------------------------------------------
```

Step 2.) Now that we have the clock signal, how can we implement the mux that will put the appropriate input on the Transmission line?

```
-- output MUXs
o_DATA <= i_D3 when f_sel = "11" else
          i_D2 when f_sel = "10" else
          i_D1 when f_sel = "01" else
          i_D0;
```

Additional Functionality.) Notice in the TDM4.vhd, we also gave you an additional MUX for the o_SEL data line as follows:

```
o_SEL  <=  "0111" when f_sel = "11" else
           "1011" when f_sel = "10" else
           "1101" when f_sel = "01" else
           "1110";
```

This additional MUX is to enable functionality that will be used in Lab 4. More on that in the next lesson.