EE214 Digital Circuits Laboratory
Wadhwani Electronics Laboratory
Electrical Engineering IIT Bombay

Problem set: 9 Date: October 24, 2025

UART - Based Counter with Clock Division

Instructions:

1. Use **Behavioral** modeling for writing VHDL description
2. Write the testbench to perform RTL simulation.
3. Demonstrate the simulations to your TA
4. Perform Pin-Planning, run on Xenon board and demonstrate to your TA.
5. Submit the entire project files in .zip format in moodle.

Part I - Clock Divider

Designing of the Clock Divider

In this experiment you will be doing clock divider. There is a 50 Mhz on board clock. You will divide it to generate 5 MHz clock and 2 Hz clock. Code for testbench will be written by you. You can run the simulation to check the waveform. You can map it in the board also so that you can assign an LED and can check if the LED is blinking.

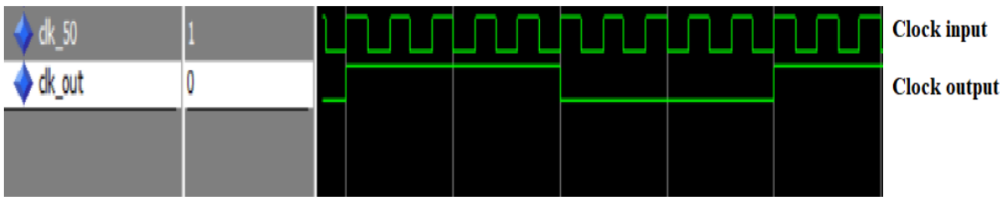

Figure 1: Waveform

NOTE: The above image(1) is for representative purpose only. May not have any similarity with the waveform that will be generated after simulation.

Method of this experiment

Suppose we need to generate f = 5 MHz from 50 MHz master clock. For this, we need a counter such that the clock out remains HIGH for 5 Input (master) Clock Cycles and LOW for next 5 Clock Cycles. In order to do this, we set-up a counter that starts from 1 and increments at every positive edge of the Input Clock (master) till the count reaches its maximum value which is 5 in this case.

$$count = 50MHz/(2 \star f) = 5$$

After the count reaches 5, count will be initialized back to 1. And clock output will go LOW till count reaches maximum again.
Note: Here we are counting from 1 to maximum count. (Not from 0 to maximum count -1). The value of count may be rounded off to nearest integer.

VHDL Description and RTL Simulation

- Generate a 50 MHz square wave input by creating your own Test bench, and 5 MHz clock and 2 Hz clock output from your design. Please verify the simulation using your own Test bench.

Part II - Integrating UART

- Here, you will understand and use a program for communicating between Xenon and a computer using UART. This program will take inputs from the computer's keyboard that can be used in programs running on the FPGA, to perform appropriate operations.

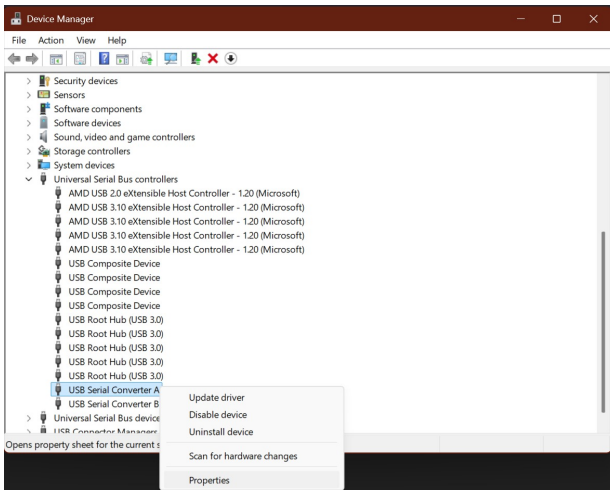

Figure 2: Step 1: Finding serial converter in device manager

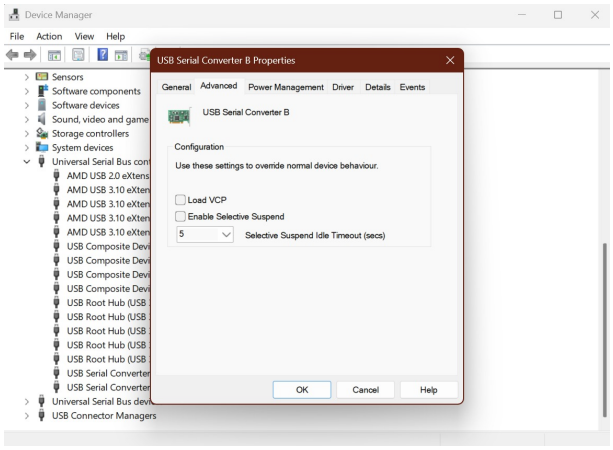

Figure 3: Step 2: In properties go to Advanced Options

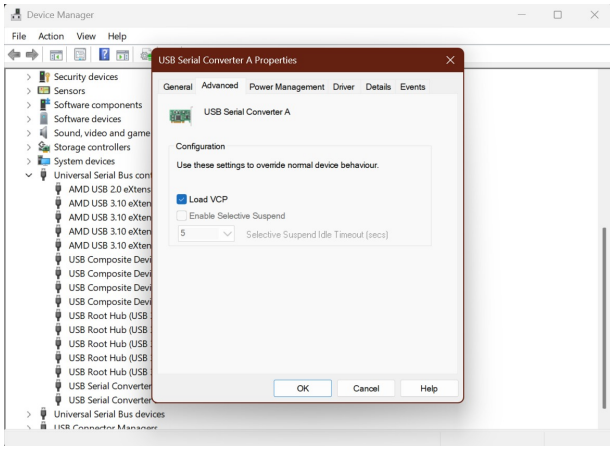

Figure 4: Step 3: Click Load VCP Option

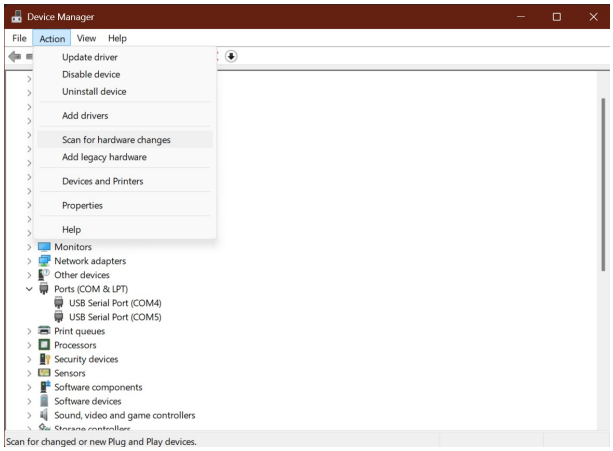

Figure 5: Step 4: Scan for Hardware Changes

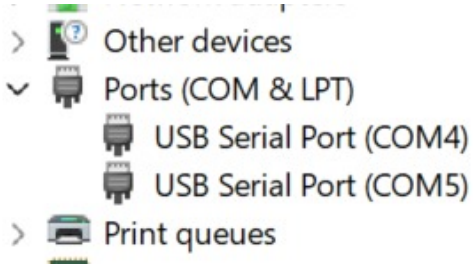

Figure 6: Step 5: You will find Serial Port under PORTS (COM,LPT)

1. For recognizing keyboard inputs on the computer and transmitting to the FPGA, through the serial terminal, you need to use the `TeraTerm` software, `RealTerm` software (or any equivalent software). The guidelines for downloading and using the TeraTerm software can be found in the following link: https://drive.google.com/file/d/1O-M1yhnHNjE3_Zf5WIJAFQeIj7CPNsPx/view?usp=sharing The download link for RealTerm software can be found in the following link: https://realterm.i2cchip.com/

Note: Set the baud rate to 9600.
Before opening RealTerm or TeraTerm, make sure to close UART JTAG.

- Design a system model with following characteristics:
  - System has a keyboard input, a 50 MHz Clock input and a reset input

– if keyboard input is "1" then implement a Johnson counter using 8 LEDs. The Johnson counter output will be displayed on LEDs such that for each clock (0.5 sec interval, using 2 Hz clock), the next state of the Johnson sequence appears on the LEDs with LED1 as the LSB and LED8 as the MSB. The pattern should loop continuously through all valid states of an 8-bit Johnson counter.

Time (s)	LED8 ... LED1
0.0	00000000
0.5	00000001
1.0	00000011
... fill in the rest ...	

– if keyboard input is "2", then implement an 8-bit ring counter. The ring counter should begin with only LED1 ON and others OFF. With each clock (0.5 sec interval, using 2 Hz clock), the logic '1' should shift to the next LED in a circular fashion (LED1 → LED2 → ... → LED8 → LED1 ...). This pattern should continue in a loop.

Time (s)	LED8 ... LED1
0.0	00000001
0.5	00000010
1.0	00000100
... fill in the rest ...	

UART Code

The code below will be your Top-level entity. Copy paste or download here and edit as required.

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

entity UART is
  generic (
    g_CLKS_PER_BIT : integer := 521    -- For 5 MHz clock and 9600 baud
  );
  port (
    i_Clk       : in  std_logic;    -- 50 MHz input clock
    i_RX_Serial : in  std_logic;
    o_LED       : out std_logic_vector(7 downto 0)  -- NEW: LED output
  );
end UART;

architecture rtl of UART is

  -- Internal 5 MHz clock from clock divider
  signal clk_5MHz : std_logic;

  -- UART RX Signals
  type t_SM_Main is (s_Idle, s_RX_Start_Bit, s_RX_Data_Bits,
                     s_RX_Stop_Bit, s_Cleanup);
  signal r_SM_Main   : t_SM_Main := s_Idle;
  signal r_RX_Data_R : std_logic := '0';
  signal r_RX_Data   : std_logic := '0';

  signal r_Clk_Count : integer range 0 to g_CLKS_PER_BIT-1 := 0;
  signal r_Bit_Index : integer range 0 to 7 := 0;
  signal r_RX_Byte   : std_logic_vector(7 downto 0) := (others => '0');
  signal r_RX_DV     : std_logic := '0';

  -- LED Counter State
  signal r_LED       : std_logic_vector(7 downto 0) := "00000001";
  signal mode        : std_logic_vector(7 downto 0) := (others => '0');

  -- Component declaration for Clock Divider
  component ClockDiv is
    port (
```

```
      clk_in       : in  std_logic;
      reset        : in  std_logic;
      clk_out_5MHz : out std_logic;
      clk_out_2Hz  : out std_logic
    );
  end component;

  signal clk_2Hz : std_logic;

begin

  -- Instantiate the clock divider (divide 50 MHz -> 5 MHz) according to your code
  u_clk_div : ClockDiv
    port map (
      clk_in       => ----
      reset        => ----
      clk_out_5MHz => ----
      clk_out_2Hz  => ----
    );

  ------------------------------------------------------------------------
  -- Double-register the incoming serial data
  ------------------------------------------------------------------------
  p_SAMPLE : process (clk_5MHz)
  begin
    if rising_edge(clk_5MHz) then
      r_RX_Data_R <= i_RX_Serial;
      r_RX_Data   <= r_RX_Data_R;
    end if;
  end process p_SAMPLE;

  ------------------------------------------------------------------------
  -- UART RX State Machine using 5 MHz clock
  ------------------------------------------------------------------------
  p_UART_RX : process (clk_5MHz)
  begin
    if rising_edge(clk_5MHz) then
      case r_SM_Main is
        when s_Idle =>
          r_RX_DV     <= '0';
          r_Clk_Count <= 0;
          r_Bit_Index <= 0;

          if r_RX_Data = '0' then
            r_SM_Main <= s_RX_Start_Bit;
          else
            r_SM_Main <= s_Idle;
          end if;

        when s_RX_Start_Bit =>
          if r_Clk_Count = (g_CLKS_PER_BIT-1)/2 then
            if r_RX_Data = '0' then
              r_Clk_Count <= 0;
              r_SM_Main   <= s_RX_Data_Bits;
            else
              r_SM_Main   <= s_Idle;
            end if;
          else
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main   <= s_RX_Start_Bit;
          end if;

        when s_RX_Data_Bits =>
          if r_Clk_Count < g_CLKS_PER_BIT-1 then
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main   <= s_RX_Data_Bits;
```

```
106             else
107               r_Clk_Count             <= 0;
108               r_RX_Byte(r_Bit_Index) <= r_RX_Data;
109
110               if r_Bit_Index < 7 then
111                 r_Bit_Index <= r_Bit_Index + 1;
112                 r_SM_Main   <= s_RX_Data_Bits;
113               else
114                 r_Bit_Index <= 0;
115                 r_SM_Main   <= s_RX_Stop_Bit;
116               end if;
117             end if;
118
119           when s_RX_Stop_Bit =>
120             if r_Clk_Count < g_CLKS_PER_BIT-1 then
121               r_Clk_Count <= r_Clk_Count + 1;
122               r_SM_Main   <= s_RX_Stop_Bit;
123             else
124               r_RX_DV     <= '1';
125               r_Clk_Count <= 0;
126               r_SM_Main   <= s_Cleanup;
127             end if;
128
129           when s_Cleanup =>
130             r_SM_Main <= s_Idle;
131             r_RX_DV   <= '0';
132          mode      <= r_RX_Byte;
133
134
135           when others =>
136             r_SM_Main <= s_Idle;
137
138         end case;
139       end if;
140     end process p_UART_RX;
141
142     ----------------------------------------------------------------------------
143     -- LED Control: Johnson or Ring counter based on mode
144     ----------------------------------------------------------------------------
145     p_LED : process (clk_2Hz)
146     begin
147       if rising_edge(clk_2Hz) then
148         case mode is
149
150           when x"31" =>
151             -- Write Logic for Johnson Counter
152
153           when x"32" =>
154              -- Write Logic for Ring Counter
155
156           when others =>
157             r_LED <= "00000001";  -- reset
158         end case;
159       end if;
160     end process p_LED;
161
162     -- Outputs
163     o_LED     <= r_led;
164
165
166 end rtl;
```

On Xenon board

- For the generated system output, show it on LED1-LED8 on Xenon board.

- Do Pin-mapping for the clock output to LEDs and 50 MHz input clock, reset to SW8 and input switch as SW1.
- Keep SW8 ON for some time then make it OFF.
- Pin Plan for Uart Rx as shown in the figure below.

Node Name	Direction	Location	I/O Bank
i_Clk	Input	PIN_26	2
i_RX_Serial	Input	PIN_25	1B

Figure 7: Uart Rx Pin Planning

- Get the LED output verified by your respective TA.
- Plain Uart RX with Testbench as a sanity check (50Mhz Clock, 115200 baud rate) UART_Rx, Testbench

Clock Source Frequency	FPGA Pin no.
1 Hz CLK	55
50 MHz CLK	26
Ext CLK	27
10 MHz CLK	29

Figure 8: Pin-mapping for on-board Clock Sources

Switch	FPGA Pin no.	LED	FPGA Pin no.
SW 8	47	LED 8	60
SW 7	46	LED 7	59
SW 6	45	LED 6	58
SW 5	44	LED 5	57
SW 4	43	LED 4	56
SW 3	41	LED 3	54
SW 2	39	LED 2	52
SW 1	38	LED 1	50

Figure 9: Pin-mapping for on-board Switches and LED's