به نام خدا

پروژه پایانی درس VHDL

استاد اسحاقی

اعضای گروه:

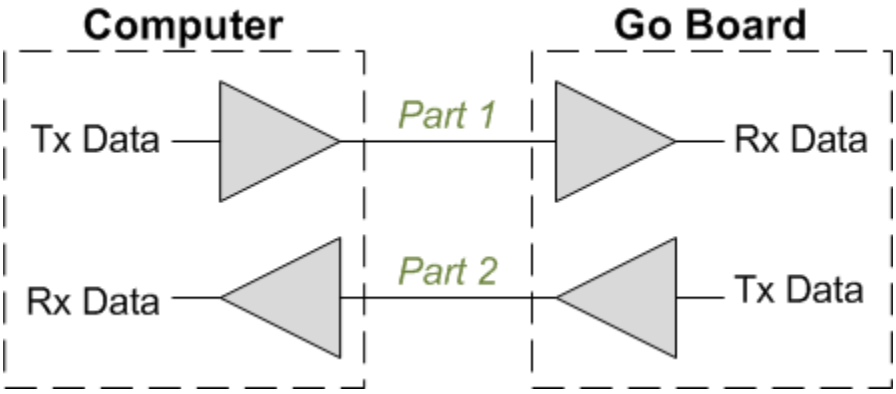
حسین کرمی

پوریا عالیشاه کمندی

مقدمه:

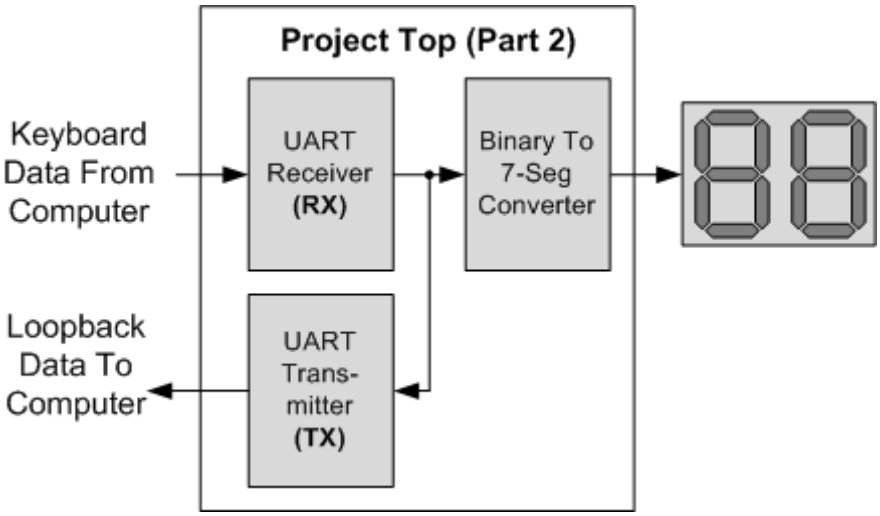
UART که مخفف universal asynchronous receiver-transmitter است، یک نوع سیستم ارتباطی بین است که با استفاده از یک رشته سیم اطلاعات را به صورت متوالی انتقال میدهد.

در اینجا یک نمونه انتقال uart را



مشاهده میکنید، که بین برد FPGA و سیستم کامپیوتری هدف است که با یک کانال ارتباطی بین یکدیگر دیتارا رد و بدل میکنند.

*در تصویر بعدی نیز ارتباط سریال بین برد و سیستم کامپیوتری را مشاهده میکنید.



در طراحی ارتباط UART باید به چند مشخصه توجه کرد:

1. سرعت انتقال یا تعداد بیت بر ثانیه(BUAD)
2. بیت مقایسه گر یا بیت (Priority Bit)
3. بیت شروع و بیت استاپ (START And STOP Bit)
4. اندازه دیتا برای انتقال

ما سیستم انتقال دهنده را برای برد خود طراحی میکنیم.

سرعت این سیستم برابر با 115200 بیت بر ثانیه است و بیت مقایسه گر نیز ندارد.

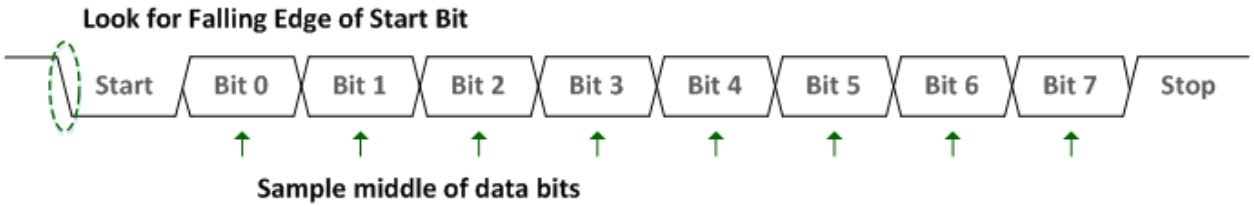
```
entity UART_TX is
  generic (
    g_CLKS_PER_BIT : integer := 87     -- Needs to be set correctly
    );
  port (
    i_Clk       : in  std_logic;
    i_TX_DV     : in  std_logic;
    i_TX_Byte   : in  std_logic_vector(7 downto 0);
    o_TX_Active : out std_logic;
    o_TX_Serial : out std_logic;
    o_TX_Done   : out std_logic
    );
end UART_TX;
```

در اینجا ما entity خود را با نام، UART_TX تعریف میکنیم که 3 ورودی:

1. iClk : ورودی کلاک
2. i_TX_DV : ورودی تنظیم بیت
3. i_TX_Byte : دیتا ورودی

همچنین 3 خروجی:

1. o_TX_Active : فعال بودن
2. o_TX_Serial : بیت دیتا(چرا که دیتا به صورت بیت به بیت وارد میشود.)
3. o_TX_Done : بیت پایان دیتا است



همانطور که مشاهده میکنید، ابتدا بیتی به عنوان بیت ورودی داده میشود که بیت 0 است که یعنی شروع انتقال دیتا اصلی ماست، بعد از انتقال 8 بیت دیتا ما، و بعد بیت استاپ داده میشود که مقدار 1 دارد و به معنی پایان انتقال است.

در این قسمت از کد ما یک Generic تعریف کردیم که مقدار کلاک در هر بیت انتقال را به ما نشان میدهد که مقدار:

$$CLK\ PER\ BIT = \frac{CLK\ OF\ BOARD}{BUAD\ OF\ UART\ SYSTEM}$$

که با توجه به اینکه ما کلاک FPGA را 10 مگا هرتز در نظر میگیریم، این عدد برابر با:

$$CLK\ PER\ BIT = \frac{10000000}{115200} = 87$$

که این یعنی در هر بیت انتقال دیتا، 87 کلاک صورت میگیرد.

```
architecture RTL of UART_TX is

  type t_SM_Main is (s_Idle, s_TX_Start_Bit, s_TX_Data_Bits,
                     s_TX_Stop_Bit, s_Cleanup);
  signal r_SM_Main : t_SM_Main := s_Idle;
  signal r_Clk_Count : integer range 0 to g_CLKS_PER_BIT-1 := 0;
  signal r_Bit_Index : integer range 0 to 7 := 0;  -- 8 Bits Total
  signal r_TX_Data   : std_logic_vector(7 downto 0) := (others => '0');
  signal r_TX_Done   : std_logic := '0';
```

حال نوبت به architecture میرسد.

ابتدا تایپی از حالت های ماشین خود ایجاد میکنیم که داری 5 حالت که به ترتیب نشان دهنده:

1. حالت اولیه
2. شروع انتقال دیتا
3. درحال ارسال
4. حالت توقف
5. حالت ریست و پاک

حال سیگنالی از تایپ خود ایجاد کرده و حالت اولیه برای مقدار اولیه ان قرار میدهیم، و تعداد شمارنده کلاک که از 0 تا 86 میباشد را ایجاد کرده و سپس تعداد بیت مورد انتقال، سیگنالی برای بیت دیتا ورودی مورد انتقال و همچنین بیت انجام را تعریف میکنیم.

```
process (i_Clk)
   begin
    if rising_edge(i_Clk) then

      case r_SM_Main is


        when s_Idle =>
          o_TX_Active <= '0';
          o_TX_Serial <= '1';         -- Drive Line High for Idle
          r_TX_Done   <= '0';
          r_Clk_Count <= 0;
          r_Bit_Index <= 0;
```

حال در این قسمت، process را تعریف میکنیم تا بتوانیم به صورت ترتیبی کدهای خود را بنویسیم.

و این process حساس به کلاک بوده و زمانی که کلاک به صورت صعودی بود وارد if شده و مقدار حالت کنونی سیستم را مورد برسی قرار میدهیم.

حال اگر حالت اولیه سیستم برابر با s_Idel بود تنظیمات بالا را برای سیگنال و خروجی انجام میدهیم.

```
if i_TX_DV = '1' then
            r_TX_Data <= i_TX_Byte;
            r_SM_Main <= s_TX_Start_Bit;
          else
            r_SM_Main <= s_Idle;
        end if;
          o_TX_Active <= '1';
          o_TX_Serial <= '0';
```

حال اگر مقدار i_TX_DV برابر 1 بود، مقدار بیت مورد انتقال را به سیگنال r_TX_Data داده و حالت کنونی سیستم را به شروع انتقال تغییر میدهیم. و در غیر این صورت حالت را به حالت اولیه بازمیگردانیم.

سپس بیت خروجی o_TX_Active را برابر 1 قرار میدهیم تا نشان دهد که UART فعال بوده و سپس با توجه به تصویر بالا، مقدار 0 را به o_TX_Serial نسبت میدهیم که یعنی شروع انتقال.

```
          if r_Clk_Count < g_CLKS_PER_BIT-1 then
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main   <= s_TX_Start_Bit;
          else
            r_Clk_Count <= 0;
            r_SM_Main   <= s_TX_Data_Bits;
        end if;
          o_TX_Serial <= r_TX_Data(r_Bit_Index);
```

حال در این قسمت چک میکنیم که تعداد کلاک های شمارش شده از مقدار تعداد کلاک در هر بیت کتمر است یا نه، که اگر کمتر بود شمارنده کلاک را یک واحد اضافه نموده و سپس حالت سیستم را به شروع انتقال تغییر میدهیم. و در غیر اینصورت، مقدار شمارنده کلاک را به صفر و حالت کنونی سیستم را به حالت در حال انتقال قرار میدهیم.

و بعد از بررسی این شروط، بیتی از 8 بیت سیگنال r_TX_Data به خروجی میدهیم که این بیت با توجه به r_Bit_Index انتقال میابد.

```
          if r_Clk_Count < g_CLKS_PER_BIT-1 then
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main   <= s_TX_Data_Bits;
          else
            r_Clk_Count <= 0;

            -- Check if we have sent out all bits
            if r_Bit_Index < 7 then
              if r_Bit_Index /= 7 then
                  r_Bit_Index <= r_Bit_Index + 1;
               end if;
              r_SM_Main   <= s_TX_Data_Bits;
            else
              r_Bit_Index <= 0;
              r_SM_Main   <= s_TX_Stop_Bit;
              end if;
            end if;
          o_TX_Serial <= '1';
```

در if و else بعدی، همان شرایط را چک نموده، و انبار در else، شماره خانه ایی که میخواهیم انتقال دهیم را ست میکنیم که اگر این شماره کمتر از 7 بودبایدیک واحد به شماره خانه اضافه نموده و حالت سیستم هم، حالت درحال انتقال میباشد. و در غیر اینصورت مقدار این شماره خانه برابر با صفر شده و حالت کنونی سیستم هم به حالت توقف در میاید چرا که تمامی این داده ها انتقال داده شده اند.

و بعد از بررسی شروط با توجه به دیاگرام ابتدایی در رابطه با انتقال دیتا، مقدار خروجی را 1 میکنیم که به معنی توقف انتقال دیتا میباشد.(STOP BIT)

```
-- Wait g_CLKS_PER_BIT-1 clock cycles for Stop bit to finish
          if r_Clk_Count < g_CLKS_PER_BIT-1 then
            r_Clk_Count <= r_Clk_Count + 1;
            r_SM_Main   <= s_TX_Stop_Bit;
          else
            r_TX_Done   <= '1';
            r_Clk_Count <= 0;
            r_SM_Main   <= s_Cleanup;
        end if; -- Stay here 1 clock when s_Cleanup =>
          o_TX_Active <= '0';
          r_TX_Done   <= '1';
          r_SM_Main   <= s_Idle;
        when others =>
          r_SM_Main <= s_Idle;
```

```
      end case;
    end if;
  end process ;

  o_TX_Done <= r_TX_Done;

end RTL;
```

حال در این قسمت تعداد کلاک ها را چک میکنیم و بعد از موافق بودن شرایط، کلاک را یک واحد افزایش داده و سپس حالت را به حالت استاپ قرار میدهیم، و در غیر اینصورت به سیگنال r_TX_Done را به 1 تغییر داده، تعداد شمارش کلاک را به صفر و حالت را به ریست یا CLEANUP تغییر میدهیم. و بعد سیگنال خروجی o_TX_Active را به صفر که به معنی غیرفعال بودن UART اشاره دارد تغییر میدهیم و بیت پایانی که r_TX_Done است را به 1 تغییر میدهیم و حالت سیستم را دوباره به حالت اولیه تغییر میدهیم.

و در قسمت پایانی اگر UART در هنگام شروع در حالتی به غیر از حالت اولیه بود، حالتش را به حالت اولیه که s_Idel میباشد، تغییر میدهیم.

و در پایان نیز مقدار r_TX_Done را به پورت خروجی o_TX_Done نسبت میدهیم. که به معنی پایان کار است.