Universidad Nacional Autonoma de Mexico Facultad de ingeniería

Diseño Digital VLSI

Grupo : 4

Práctica: 9

DISEÑO DE UN TRANSMISOR PARA COMUNICACIÓN SERIAL

Edgar Daniel Barcenas Martinez

OBJETIVO:

Demostrar a los estudiantes mediante el diseño de un módulo transmisor (TX) útil en comunicaciones de tipo serial UART (*Universal Asyncrhonous Receiver Transmitter*), la utilidad de este módulo, así como la importancia de su presencia en la arquitectura de un procesador para aplicaciones electrónicas de envío de información.

ESPECIFICACIONES:

Utilizando un FPGA y un switch de 4 posiciones, diseñar un módulo Transmisor serial, el cual sea capaz de leer el valor binario del switch, procesarlo en el FPGA y posteriormente enviarlo a la computadora, en donde el dato deberá estar en formato hexadecimal. La conexión entre el FPGA y la computadora deberá realizarse empleando un circuito convertidor USB TTL-Serial. La figura 8.1 muestra el diagrama de bloques del sistema.

DESARROLLO:

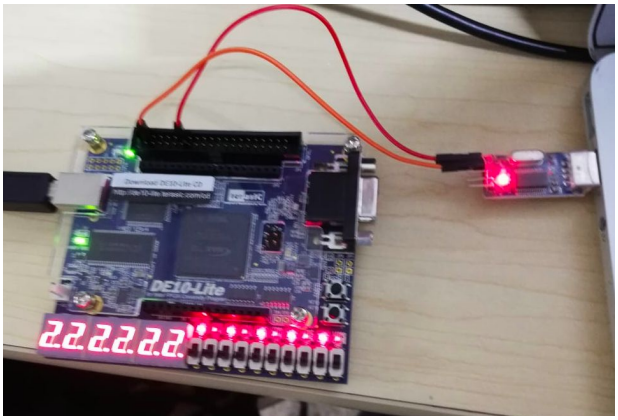
```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4  entity TX is
5      port( Clk : IN STD_LOGIC;
6      SW : IN STD_LOGIC_VECTOR(3 downto 0);
7      LED : OUT STD_LOGIC;
8      TX_WIRE : OUT STD_LOGIC);
9  end entity;
10
11 architecture behaivoral OF TX IS
12     signal conta : INTEGER := 0;
13     signal valor : INTEGER := 70000;
14     signal INICIO: STD_LOGIC;
15     signal dato : STD_LOGIC_VECTOR(7 DOWNTO 0);
16     signal PRE : INTEGER RANGE 0 TO 5208 := 0;
17     signal INDICE: INTEGER RANGE 0 TO 9 := 0;
18     signal BUFF : STD_LOGIC_VECTOR(9 DOWNTO 0);
19     signal Flag : STD_LOGIC := '0';
20     signal PRE_val: INTEGER range 0 to 41600;
21     signal baud : STD_LOGIC_VECTOR(2 DOWNTO 0);
22     signal i : INTEGER range 0 to 4;
23     signal pulso : STD_LOGIC:='0';
24     signal conta2: integer range 0 to 49999999 := 0;
25     signal dato_bin: STD_LOGIC_VECTOR(3 DOWNTO 0);
26     signal hex_val: STD_LOGIC_VECTOR(7 DOWNTO 0):= (others => '0');
27     --signal contador: STD_LOGIC_VECTOR(7 DOWNTO 0) := (others => '0');
28     signal contador : INTEGER := 0;
29 begin
```

```
31  TX_divisor : process(Clk, contador)
32  begin
33     if rising_edge(Clk) then
34         contador <= contador + 1;
35         if (contador < 140000) then
36         #--Proceso divide la frecuencia de 50 MHz hasta los  14000
37             pulso <= '1';
38         #--140000/20ns=2.8miliseg. esto indica que cada 2.8 milise
39         else
40             pulso <= '0';
41         end if;
42  end if;
43  end process TX_divisor;
44
45  TX_prepara : process(Clk, pulso) #--Verifica la señal de pulso, s
46  type arreglo is array (0 to 1) of STD_LOGIC_VECTOR(7 downto 0);
47  variable asc_dato : arreglo := (X"30",X"0A");#--Este es el dato q
48  begin
```

```
49   asc_dato(0):=hex_val;--si hay un pulso va a enviar un dato
50   if (pulso='1') then
51      if rising_edge(Clk) then
52          if (conta=valor) then
53              conta <= 0;
54              INICIO <= '1';
55              Dato <= asc_dato(i);
56              if (i = 1) then
57                  i <= 0;
58              else
59                  i <= i + 1;
60              end if;
61          else
62              conta <= conta+1;
63              INICIO <= '0';
64          end if;
65      end if;
66   end if;
67   end process TX_prepara;
```

```
69  TX_envia : process(Clk,INICIO,dato)--
70  begin
71  if(Clk'EVENT and Clk = '1') then
72      if(Flag = '0' and INICIO = '1') then
73          Flag<= '1';
74          BUFF(0) <= '0';
75          BUFF(9) <= '1';
76          BUFF(8 DOWNTO 1) <= dato;
77      end if;
78      if(Flag = '1') then
79          if(PRE < PRE_val) then
80              PRE <= PRE + 1;
81          else
82              PRE <= 0;
83          end if;
84          if(PRE = PRE_val/2) then
85              TX_WIRE <= BUFF(INDICE);
86              if(INDICE < 9) then
87                  INDICE <= INDICE + 1;
88              else
89                  Flag <= '0';
90                  INDICE <= 0;
91              end if;
92          end if;
93      end if;
94  end if;
95  end process TX_envia;
96
97  LED <= pulso;
98  dato_bin<=SW;
99  baud<="011";
```

```
101 with(dato_bin) select
102             hex_val <= X"30" when "0000",
103                        X"31" when "0001",
104                        X"32" when "0010",
105                        X"33" when "0011",
106                        X"34" when "0100",
107                        X"35" when "0101",
108                        X"36" when "0110",
109                        X"37" when "0111",
110                        X"38" when "1000",
111                        X"39" when "1001",
112                        X"41" when "1010",
113                        X"42" when "1011",
114                        X"43" when "1100",
115                        X"44" when "1101",
116                        X"45" when "1110",
117                        X"46" when "1111",
118                        X"23" when others;
119 with (baud) select
120 PRE_val <= 41600 when "000", -- 1200 bauds
121                20800 when "001", -- 2400 bauds
122                10400 when "010", -- 4800 bauds
123                5200 when "011", -- 9600 bauds
124                2600 when "100", -- 19200 bauds
125                1300 when "101", -- 38400 bauds
126                866 when "110", -- 57600 bauds
127                432 when others; --115200 bauds
128 end architecture behaivoral;
```



PRACTICA COMPLEMENTARIA:

CONCLUSIONES:

En esta práctica se demostró el uso que tienen los módulos de transmisor (TX) , se usó uso de un software llamado Puffy el cual es un cliente SSH, Telnet, rlogin y TCP para la comunicación entre el FPGA y la computadora. Se utilizó un dispositivo TTL.