**Objetivo**

Demostrar a los estudiantes mediante el diseño de un módulo receptor (Rx), usado 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 recepción de información. Mostrar su aplicación en el control de dispositivos periféricos desde una terminal remota.

**Introducción**

En esta práctica se requerirán conocimientos sobre diferentes conceptos vistos tanto en clases pasadas como en la materia de diseño digital moderno. Algunos conceptos importantes que será bueno recordar serán: la lógica para crear registros en VHDL y el uso de operadores lógicos. Además de esto será importante tener un hardware físico que contenga el switch y un módulo UART para la comunicación con la computadora. Además de esto asegurarse de tener conocimiento en el uso de switches de la FPGA.

Para la realización del código se deberá crear un módulo VHDL que contenga la lógica necesaria para leer el estado del switch y convertirlo en una representación binaria de 4 bits. El módulo debe incluir lógica para leer el estado del switch y convertirlo en una representación binaria de 4 bits. Esto puede hacerse utilizando operadores lógicos y registros en VHDL. Se debe tener un módulo UART configurado en VHDL para transmitir datos a la computadora. El módulo UART tomará los 4 bits binarios y los enviará en serie a la computadora. Se puedes crear una cadena de texto que tenga el formato "valor binario=XXXX", donde "XXXX" representa los 4 bits binarios generados a partir del switch. Configura la velocidad de transmisión de la comunicación UART de acuerdo con la velocidad con la que deseas que se envíe la información a la computadora.

Realiza simulaciones para verificar el funcionamiento de tu diseño y asegurarte de que la salida generada cumple con tus requisitos. Una vez se haya probado y verificado el diseño en simulación, se puede implementar en tu hardware físico, como una FPGA.

Conecta tu hardware a la computadora y monitorea la salida para asegurarte de que se esté transmitiendo correctamente la secuencia binaria formateada.

Para el desarrollo de la práctica es fundamental conocer el circuito convertidor USB TTL-serial que vamos a utilizar.

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Convertidor USB TTL-serial FT232RL

Descripción general:

El adaptador interfaz USB serial FT232RL permite comunicarnos a través del puerto USB de la PC con la UART (Universal Asynchronous Transmitter Receiver) de un microcontrolador, arduino, GPS o cualquier otro dispositivo que tenga este tipo de interfaz. Usualmente la interfaz UART se emplea para la descarga de programas (mediante bootloader), para la configuración de los parámetros de operación en equipos electrónicos, o como interfaz para mensajes de depuración en sistemas embebidos.

Las ventajas del adaptador interfaz USB serial FT232RL es que puede utilizarse con sistemas electrónicos cuyo funcionamiento es a 5 y 3.3 volts, pudiendo seleccionar el voltaje de las señales correspondientes mediante un jumper.

**Datos Técnicos importantes para el desarrollo de la practica**

* Voltaje de Operación: 5 volts.
* Puede obtener acceso a señales GND, CTS, VCC, TX, RX y DTR.
* La interfaz UART soporta 7 u 8 bits de datos, 1 ó 2 bits de parada, y paridad par/impar/marca/espacio/sin paridad
* Operación a 3.3 V o 5 V configurable mediante jumper
* Buffer de recepción de 128 Bytes y de transmisión de 256 bytes
* LED indicador de señal de transmisión y recepción

En este caso el convertidor trabajara con un voltaje de 5 V proporcionado por la FPGA.

**Desarrollo**

El alumno diseñara un sistema capaz de realizar el control de acciones sobre periféricos conectados al FPGA. El control deberá realizarse seleccionando cuatro diferentes caracteres del teclado de la computadora para ejecutar las siguientes tareas:

1.- PWM-RGB.

2.- Leer el estado del dipswitch y mostrarlo en 4 leds.

3.- Contador binario de 0 a 9.

4.- PWM en un LED.

**PWM – RGB**

Esta sección ejecutará el código que hicimos en la práctica del LED RGB. Cuando se active esta opción, se encenderán los 7 leds RGB para mostrar los colores del arcoíris.

**Estado del dipswitch y leds**

En esta sección se leerá el estado de 4 dipswitch y se encenderán los focos que asignemos a cada dipsiwtch.

**Contador binario de 0-9**

En esta sección se inicializará un contador en binario del 0-9 en los display de 7 segmentos.

**PWM – LED**

En esta sección se variará la potencia de 1 LED de la tarjeta con un PWM.

**Código utilizado:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity p8 is

Port ( clk : in std\_logic;

LEDS : out std\_logic\_vector(7 downto 0);

o7Seg : out std\_logic\_vector(6 downto 0);

switch : in std\_logic\_vector(3 downto 0);

ledtar : out std\_logic\_vector(3 downto 0);

ledR: out std\_logic;

ledG: out std\_logic;

ledB: out std\_logic;

ledextra: out std\_logic;

RX\_WIRE: in std\_logic

);

end p8;

architecture Behavioral of p8 is

component divisor is

generic (N: integer:=24);

port(reloj:in std\_logic;

div\_reloj: out std\_logic

);

end component;

component pwm is

port(reloj\_pwm:in std\_logic;

D: in std\_logic\_vector (7 downto 0);

S: out std\_logic

);

end component;

signal relojPWM: std\_logic;

signal relojCiclo: std\_logic;

signal a1: std\_logic\_vector (7 downto 0):=X"00";

signal a2: std\_logic\_vector (7 downto 0):=X"00";

signal a3: std\_logic\_vector (7 downto 0):=X"00";

signal a4: std\_logic\_vector (7 downto 0):=X"00";

signal BUFF : std\_logic\_vector(9 downto 0);

signal flag : std\_logic:='0';

signal INDICE : INTEGER RANGE 0 to 9:=0;

signal PRE : INTEGER RANGE 0 to 5208;

signal PRE\_val: INTEGER RANGE 0 to 41600;

signal baud : std\_logic\_vector(2 downto 0);

begin

D1: divisor generic map (10) port map (clk, relojPWM);

D2: divisor generic map (23) port map (clk, relojCiclo);

P1: pwm port map (relojPWM, a1, ledR);

P2: pwm port map (relojPWM, a2, ledG);

P3: pwm port map (relojPWM, a3, ledB);

P4: pwm port map (relojPWM, a4, ledextra);

RX\_dato:process (clk)

variable Cuenta : integer range 0 to 255:=0;

begin

if(clk'event and clk='1')then

if(flag='0' and RX\_WIRE='0')then

flag<='1';

INDICE<=0;

PRE<=0;

end if;

if(flag='1')then

BUFF(INDICE)<=RX\_WIRE;

if(PRE<PRE\_val)then

PRE<=PRE+1;

else

PRE<=0;

end if;

if(PRE=PRE\_val/2)then

if(INDICE<9)then

INDICE<=INDICE+1;

else

If BUFF(8 downto 1) = X"30" THEN --0

LEDS<="10000001";

elsif BUFF(8 downto 1) = X"31" THEN --1

LEDS<="11110011";

a4 <= X"00";

o7seg <="1111111";

if cuenta = 0 then

a1 <= x"FF"; --Amarillo

a2 <= x"FF";

a3 <= x"00";

elsif cuenta = 5 then

a1 <= x"00"; --Verde

a2 <= x"FF";

a3 <= x"00";

elsif cuenta = 10 then

a1 <= x"00"; --Azul

a2 <= x"60";

a3 <= x"FF";

elsif cuenta = 15 then

a1 <= x"FF"; --Rojo

a2 <= x"00";

a3 <= x"00";

elsif cuenta = 20 then

a1 <= x"FF"; --violeta

a2 <= x"60";

a3 <= x"EF";

elsif cuenta = 25 then

a1 <= x"FF"; --naranja

a2 <= x"30";

a3 <= x"00";

elsif cuenta = 30 then

a1 <= x"FF"; --rosa

a2 <= x"60";

a3 <= x"60";

end if;

cuenta := cuenta+1;

if cuenta = 35 then

cuenta :=0;

end if;

elsif BUFF(8 downto 1) = X"32" THEN --2

LEDS<="01001001";

a1 <= X"00";

a2 <= X"00";

a3 <= X"00";

a4 <= X"00";

ledtar<=switch;

o7seg <="1111111";

elsif BUFF(8 downto 1) = X"33" THEN --3

LEDS<="01100001";

a1 <= X"00";

a2 <= X"00";

a3 <= X"00";

a4 <= X"00";

if cuenta < 3 then

o7seg <="1000000";

elsif cuenta < 6 then

o7seg <="1111001";

elsif cuenta < 9 then

o7seg <="0100100";

elsif cuenta < 12 then

o7seg <="0110000";

elsif cuenta < 15 then

o7seg <="0011001";

elsif cuenta < 18 then

o7seg <="0010010";

elsif cuenta < 21 then

o7seg <="0000010";

elsif cuenta <24 then

o7seg <="1111000";

elsif cuenta < 27 then

o7seg <="0000000";

elsif cuenta < 30 then

o7seg <="0010000";

end if;

cuenta := cuenta+1;

if cuenta >= 30 then

cuenta :=0;

end if;

elsif BUFF(8 downto 1) = X"34" THEN --4

LEDS<="00110011";

a1 <= X"00";

a2 <= X"00";

a3 <= X"00";

o7seg <="1111111";

if cuenta > 5 then

a4 <= X"00";

elsif cuenta > 10 then

a4 <= X"00"; --Verde

elsif cuenta > 15 then

a4 <= X"30";

elsif cuenta > 20 then

a4 <= X"6F"

elsif cuenta > 25 then

a4 <= X"9F";

elsif cuenta > 30 then

a4 <= X"CF";

elsif cuenta > 35 then

a4 <= X"FF"; --rosa

end if;

cuenta := cuenta+1;

if cuenta = 35 then

cuenta :=0;

end if;

else

LEDS<="00000000";

end if;

flag<='0';

end if;

end if;

end if;

end if;

end process;

baud<="011";

with(baud)select

PRE\_VAL<=41600 when "000", --1200 bauds

20800 when "001", --2400 bauds

10400 when "010", --4800 bauds

5200 when "011", --9600 bauds

2600 when "100", --19200 bauds

1300 when "101", --38400 bauds

866 when "110", --57600 bauds

432 when others; --115200 bauds

end Behavioral;

**Video del funcionamiento del programa:**

**Bibliografía**