**Objetivo**

* El alumn@ diseñará un sistema capaz de enviar el valor del switch en forma binaria, es decir, cuatro caracteres, uno por bit leído. La forma en que la secuencia de texto que deberá ser visualizado en la computadora es: valor binario=XXXX, donde XXXX representa el número de 4 bits.
* Demostrar a los estudiantes mediante el diseño de un módulo transmisor (TX), empleado 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 en envío de información.

**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**

Para esta práctica usaremos el código proporcionado por el profesor, este código funciona de la siguiente manera:

En la arquitectura Behavioral se declaran variables que se utilizaran en el diseño, en este se establecen diferentes procesos. El TX\_divisor se encarga de generar un pulso cada cierto tiempo para generar la temporización del transmisor. TX\_prepara prepara los datos a transmitir a un vector y seleccionar un byte de datos cuanto el pulso es alto. TX\_envia controla la transmisión de datos usando el buffer y la temporización. Los datos se transmiten cuando flag está activo e inicio es alto.

Código:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity p7 is

Port (clk : in std\_logic;

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

LED : out std\_logic;

TX\_WIRE: out std\_logic

);

end p7;

architecture Behavioral of p7 is

signal conta : integer:= 0;

signal valor : integer:= 70000;

signal INICIO : std\_logic;

signal dato : std\_logic\_vector(7 downto 0);

signal PRE : integer range 0 to 5208:=0;

signal INDICE : integer range 0 to 9:=0;

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

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

signal PRE\_VAL : integer range 0 to 41600;

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

signal i : integer range 0 to 22;

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

signal contador : integer range 0 to 49999999:=0;

signal dato\_bin: std\_logic\_vector(3 downto 0);

signal hex\_val4 :std\_logic\_vector(7 downto 0):=(others=>'0');

signal hex\_val3 :std\_logic\_vector(7 downto 0):=(others=>'0');

signal hex\_val2 :std\_logic\_vector(7 downto 0):=(others=>'0');

signal hex\_val1 :std\_logic\_vector(7 downto 0):=(others=>'0');

begin

TX\_divisor : process(clk)

begin

if rising\_edge(clk) then

contador <= contador + 1;

if(contador < 1540000) then

pulso <= '1';

else

pulso <= '0';

end if;

end if;

end process TX\_divisor;

TX\_prepara : process(clk, pulso)

type arreglo is array (0 to 20) of std\_logic\_vector(7 downto 0);

variable asc\_dato : arreglo := ("01010110", "01000001", "01001100", "01001111", "01010010", "00100000", "01000010", "01001001", "01001110", "01000001", "01010010", "01001001", "01001111", "00100000", "00111101", "00100000","00110000","00110000","00110000","00001010","00001100");

begin

asc\_dato(18) := hex\_val1; --:=

asc\_dato(17) := hex\_val2; --:=

asc\_dato(16) := hex\_val3; --:=

asc\_dato(15) := hex\_val4; --:=

if(pulso = '1') then

if rising\_edge(clk) then

if(conta = valor) then

conta <= 0;

inicio <= '1';

dato <= asc\_dato(i);

if(i = 21) then

i <= 0;

else

i <= i + 1;

end if;

else

conta <= conta + 1;

inicio <= '0';

end if;

end if;

end if;

end process TX\_prepara;

TX\_envia : process(clk, inicio, dato)

begin

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

if(flag = '0' and inicio = '1') then

flag <= '1';

BUFF(0) <= '0';

BUFF(9) <= '1';

BUFF(8 downto 1) <= dato;

end if;

if(flag = '1') then

if(PRE < PRE\_val) then

PRE <= PRE + 1;

else

PRE <= 0;

end if;

if(PRE = PRE\_val/2) then

tx\_wire <= BUFF(indice);

if(indice < 9) then

indice <= indice + 1;

else

flag <= '0';

indice <= 0;

end if;

end if;

end if;

end if;

end process TX\_envia;

LED <= pulso;

dato\_bin <= SW;

process (dato\_bin)

begin

case (dato\_bin) is

when "0000" =>

hex\_val1 <= X"30";

hex\_val2 <= X"30";

hex\_val3 <= X"30";

hex\_val4 <= X"30";

when "0001" =>

hex\_val1 <= X"31";

hex\_val2 <= X"30";

hex\_val3 <= X"30";

hex\_val4 <= X"30";

when "0010" =>

hex\_val1 <= X"30";

hex\_val2 <= X"31";

hex\_val3 <= X"30";

hex\_val4 <= X"30";

when "0011" =>

hex\_val1 <= X"31";

hex\_val2 <= X"31";

hex\_val3 <= X"30";

hex\_val4 <= X"30";

when "0100" =>

hex\_val1 <= X"30";

hex\_val2 <= X"30";

hex\_val3 <= X"31";

hex\_val4 <= X"30";

when "0101" =>

hex\_val1 <= X"31";

hex\_val2 <= X"30";

hex\_val3 <= X"31";

hex\_val4 <= X"30";

when "0110" =>

hex\_val1 <= X"30";

hex\_val2 <= X"31";

hex\_val3 <= X"31";

hex\_val4 <= X"30";

when "0111" =>

hex\_val1 <= X"31";

hex\_val2 <= X"31";

hex\_val3 <= X"31";

hex\_val4 <= X"30";

when "1000" =>

hex\_val1 <= X"30";

hex\_val2 <= X"30";

hex\_val3 <= X"30";

hex\_val4 <= X"31";

when "1001" =>

hex\_val1 <= X"31";

hex\_val2 <= X"30";

hex\_val3 <= X"30";

hex\_val4 <= X"31";

when "1010" =>

hex\_val1 <= X"30";

hex\_val2 <= X"31";

hex\_val3 <= X"30";

hex\_val4 <= X"31";

when "1011" =>

hex\_val1 <= X"31";

hex\_val2 <= X"31";

hex\_val3 <= X"30";

hex\_val4 <= X"31";

when "1100" =>

hex\_val1 <= X"30";

hex\_val2 <= X"30";

hex\_val3 <= X"31";

hex\_val4 <= X"31";

when "1101" =>

hex\_val1 <= X"31";

hex\_val2 <= X"30";

hex\_val3 <= X"31";

hex\_val4 <= X"31";

when "1110" =>

hex\_val1 <= X"30";

hex\_val2 <= X"31";

hex\_val3 <= X"31";

hex\_val4 <= X"31";

when "1111" =>

hex\_val1 <= X"31";

hex\_val2 <= X"31";

hex\_val3 <= X"31";

hex\_val4 <= X"31";

when others =>

hex\_val1 <= X"30";

hex\_val2 <= X"30";

hex\_val3 <= X"30";

hex\_val4 <= X"30";

end case;

end process;

baud<="011";

with (baud) select

pre\_val <= 41600 when "000", --1200 bauds

20800 when "001", --2400

10400 when "010", --4800

5200 when "011", --9600

2600 when "100", --19200

1300 when "101", --38400

866 when "110", --57600

432 when others; --

end architecture behavioral;

El código fue modificado para que se muestre el mensaje

VALOR BINARIO=XXXX

Para esto se tuvo que buscar el valor binario de cada letra y meterlo en la variable arreglo además de esto se modifico el tamaño para que haya suficiente espacio para cada palabra además del salto de línea y el retorno de carro

"01010110" es V

"01000001" es A

"01001100" es L

"01001111" es O

"01010010" es R

"00100000" es espacio

"01000010" es B

"01001001" es I

"01001110" es N

"01000001" es A

"01010010" es R

"01001001" es I

"01001111" es O

"00100000" es espacio

"00111101" es =

"00100000" es espacio

"00110000" es 0

"00110000" es 0

"00110000" es 0

"00001010" es espacio

"00001100" es salto de línea

En nuestro código el espacio y los 3 ceros del final serán sustituidos por los valores que nos den los switch de la tarjeta. Por lo que en el caso que el switch marque 0000

El mensaje mostrado será

VALOR BINARIO=0000\n

Ahora para mostrar el texto de forma completa de forma más rápida se modifico en la parte del divisor ya que la impresión te cada palabra es de 70000 en el contador como el texto valor binario es de aproximadamente 22 palabras hacemos la multiplicación 70000\*22 =1540000. Además, para mostrar los diferentes valores en 4 bits se modifico y crearon 4 variables donde se guardaban los valores hexadecimales de cada bit. Para gestionar los diferentes cambios se creo un process con las 4 variables. Con esto se modificó cada valor después del espacio en el texto mostrado

Video del funcionamiento del programa:

https://youtube.com/shorts/a5tvTXeZojU?si=zgVxTzga2g9sIHYF

**Bibliografía**