Sergio Bou Grau

# Proyecto: documentación de una UART

El siguiente proyecto tiene como finalidad el estudio y descripción de un proyecto informático escrito en lenguaje VHDL para comprender su funcionamiento y la utilidad de cada bloque que lo componen. El proyecto consta de cinco ficheros que se enumeran y describen a continuación.

## Timing

Este fichero se encarga de gestionar la elección de la velocidad de transmisión y la generación de los relojes de la velocidad, de la transmisión y de la recepción.

En primer lugar, en este fichero tenemos incluidas la siguientes librerías:

***library ieee;***

***use ieee.std\_logic\_1164.all;***

***use ieee.numeric\_std.all;***

***use ieee.std\_logic\_unsigned.all;***

El entity está compuesto por las siguientes entradas y salidas:

***Entity timing is***

***generic (F : natural := 50000;***

***min\_baud : natural := 1200);***

***port (***

***CLK : in std\_logic;***

***RST : in std\_logic;***

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

***ClrDiv : in std\_logic;***

***Top16 : buffer std\_logic;***

***TopTx : out std\_logic;***

***TopRx : out std\_logic***

***);***

***end timing;***

En la Architecture tenemos las siguientes señales:

***signal baud\_value : natural;***

***constant max\_div : natural := ((F\*1000)/(16\*min\_baud));***

***subtype div16\_type is natural range 0 to max\_div-1;***

***signal Div16 : div16\_type;***

***signal ClkDiv : integer;***

***signal RxDiv : integer;***

A continuación tenemos un process que mediante la entrada BAUD compuesta por tres bits nos permite seleccionar entre ocho distintas frecuencias para comunicarnos mediante el protocolo UART, este process es el siguiente:

***process (CLK)***

***begin***

***if rising\_edge(CLK) then***

***case Baud is***

***when "000" => baud\_value <= ((F\*1000)/(16\*115200));***

***when "001" => baud\_value <= ((F\*1000)/(16\*57600));***

***when "010" => baud\_value <= ((F\*1000)/(16\*38400));***

***when "011" => baud\_value <= ((F\*1000)/(16\*19200));***

***when "100" => baud\_value <= ((F\*1000)/(16\*9600));***

***when "101" => baud\_value <= ((F\*1000)/(16\*4800));***

***when "110" => baud\_value <= ((F\*1000)/(16\*2400));***

***when "111" => baud\_value <= ((F\*1000)/(16\*1200));***

***when others => baud\_value <= ((F\*1000)/(16\*1200)); -- n.u.***

***end case;***

***end if;***

***end process;***

A continuación tenemos el process que realiza la cuenta para generar la señal del reloj a partir de la opción elegida en el process anterior:

***process (RST, CLK)***

***begin***

***if RST='0' then***

***Top16 <= '0';***

***Div16 <= 0;***

***elsif rising\_edge(CLK) then***

***Top16 <= '0';***

***if Div16 = baud\_value then***

***Div16 <= 0;***

***Top16 <= '1';***

***else***

***Div16 <= Div16 + 1;***

***end if;***

***end if;***

***end process;***

Otro process que tenemos en este fichero es el que realiza la cuenta para el reloj que establece la escritura, que es el siguiente:

***process (RST, CLK)***

***begin***

***if RST='0' then***

***TopTx <= '0';***

***ClkDiv <= 0;***

***elsif rising\_edge(CLK) then***

***TopTx <= '0';***

***if Top16='1' then***

***ClkDiv <= ClkDiv + 1;***

***if ClkDiv = 15 then***

***TopTx <= '1';***

***ClkDiv <= 0;***

***end if;***

***end if;***

***end if;***

***end process;***

A continuación y por último tenemos el process que realiza la cuenta del reloj de lectura:

***process (RST, CLK)***

***begin***

***if RST='0' then***

***TopRx <= '0';***

***RxDiv <= 0;***

***elsif rising\_edge(CLK) then***

***TopRx <= '0';***

***if ClrDiv='1' then***

***RxDiv <= 0;***

***elsif Top16='1' then***

***if RxDiv = 7 then***

***RxDiv <= 0;***

***TopRx <= '1';***

***else***

***RxDiv <= RxDiv + 1;***

***end if;***

***end if;***

***end if;***

***end process;***

## Transmit

En el fichero transmit podemos observar que se gestiona una maquina de estados para la transmisión de los datos.

En primer lugar tenemos las siguientes librerías:

***library ieee;***

***use ieee.std\_logic\_1164.all;***

***use ieee.numeric\_std.all;***

Y en el entity las siguientes entradas y salidas:

***entity transmit is***

***generic (NDBits : natural := 8);***

***port (CLK : in std\_logic;***

***RST : in std\_logic;***

***TopTx : in std\_logic;***

***StartTx : in std\_logic;***

***Din : in std\_logic\_vector (NDBits-1 downto 0);***

***Tx : out std\_logic;***

***TxBusy : out std\_logic );***

***end;***

En la architecture inicialmente tenemos las siguientes señales:

***type State\_Type is (Idle, Load\_Tx, Shift\_Tx, Stop\_Tx);***

***signal TxFsm : State\_Type;***

***signal Tx\_Reg : std\_logic\_vector (NDBits downto 0);***

***signal RegDin : std\_logic\_vector (NDBits-1 downto 0);***

***signal TxBitCnt : natural range 0 to 15;***

A continuación en un único process tenemos la maquina de estados que gestiona la transmisión de datos,

## Receive

## UART

## Uart\_comps